ECE 6560

Implementing an Eulerian PDE Approach for Computing Thickness

Isaac Lee

April 28, 2018

Table of Contents

I. Introduction

In science, analysis, and computer vision, thickness is an important metric for measuring different properties in an image to gain insight of fundamental structural performance. Measuring thickness has contributions in ultrasound imagery, medical analysis, drilling, robotics, etc. all of which conclude important dimensions in scientific and engineering approach in helping to draw potential insights from system dynamics through imaging.

This paper discusses an implementation of one particular approach of calculating thickness via an Eulerian framework [1] which is a discretized model of calculating partial differential equations. In a general sense, thickness should be the corresponding distance between two points of two boundaries. In mathematical terms, thickness is "defined as the length of correspondence trajectories, which run from one tissue boundary to the other" [1] and these trajectories are curved in general. By this definition, an inherent mathematical model can be derived using special properties and computational procedure.

II. Thickness Principles

To extrapolate a computational approach, it is necessary to configure and understand thickness properties in a mathematical sense. In this approach, thickness is defined as arbitrary corresponding trajectories that follow through a smooth vector field within the region of the two boundaries. The underlying assumptions in this idea are that correspondence trajectories do not intersect and map one to one (uniquely) from the inner boundary to the outer, orthogonality is on both boundaries, and that the most efficient distance or shortest when possible is taken.

This approach considers a general model in that thickness can be different for asymmetrical models and uses a normalized gradient of a harmonic function to calculate the vector field of unit tangents. A pair of partial differential equations (PDEs), which are guided by this vector field, are solved over the thickness region and the solutions of these PDEs calculate the thickness points directly from their respective starting boundary point and therefore the sum of these PDEs calculate thickness directly at each point.

III. Thickness Mathematical Approach

In this section, the mathematical model is outlined to measure the thickness in a region using linear PDEs.

A. Correspondence Trajectories

In this paper, the region is assumed to be in $R \subset R^n, n = 2,3$ with two closed boundaries, d_0R (inner) and d_1R (outer). Thickness is at any point $x \in R$ as the total arclength of a unique curve that remains in region R and starts on d_0R and finishes on d_1R.

B. Tangent Field

The unit vector field that the corresponding trajectories uses is simply a normalized gradient vector flow field. To calculate the tangent field, a unique harmonic function u is used which

interpolates between a potential from d_0R to d_1R. The function u is solved by solving Laplace's equation over R:

$$\Delta u = 0 \; with \; u(d_0R) = \; 0, u(d_1R \;) \; = \; 1$$

(Equation 1)

and the corresponding tangent field is:

$$\vec{T} = \frac{\nabla u}{\|\nabla u\|}$$

(Equation 2)

C. Calculating Thickness

Thickness is defined such that the unit vector field $T(x)$ guides the correspondence trajectories from one boundary point to the other boundary point through the region R and they are mapped uniquely. Therefore, the length functions L_0 and L_1, are defined such that $L_0(x)$ is the arclength function of the correspondence trajectory between d_0R and x and $L_1(x)$ is the arclength between d_1R and x, the intuition is that they start on their corresponding boundaries and end on the opposite.

The pair of length of functions follow a first-order linear PDE:

$$\nabla L_0 \cdot \vec{T} = 1, L_0(d_0R) = 0$$

$$-\nabla L_1 \cdot \vec{T} = 1, L_1(d_1R) = 0$$

(Equation 3)

So, thickness is obtained at any point by adding L_0 and L_1:

$$W(x) \; = \; L_0(x) \; + \; L_1(x)$$

(Equation 4)

IV. Numerical Approach

The approach in this paper follows the Eulerian PDE approach in [1] for thickness calculations. The tangent field is first computed by the unique harmonic function u. Here, the standard method for collecting u is defined by the discretized central difference for calculating Laplace's equation. The central difference scheme captures information from bordering pixels on both sides and provides a better and more accurate approximation.

$$\Delta u = 0$$

$$u_t = u_{xx}$$

$$u(x, t + \Delta t) \; = \; u(x, t) \; + \; \Delta t(\frac{u(x + \Delta x, t) + u(x - \Delta x, t) - 2 * u(x, t)}{\Delta x^2})$$

(Equation 5)

with CFL condition:

$$\Delta t \leq \frac{1}{2}\Delta x^2$$

(Equation 6)

The choice of $\Delta x,\ \Delta y,\ \Delta t$ influence the stability of the discretization scheme and so it is important to select the correct CFL conditions to achieve convergence and stability in the scheme. The CFL conditions selected for this scheme is:

$$\Delta x = 1, \Delta y = 1, \Delta t \leq \frac{1}{2}$$

(Equation 7)

Here, we consider $\Delta x = 1$, so the time step is therefore less than ½. From the calculated harmonic function u, the unit tangent field (2) is determined by the gradient of u in all directions divided by its norm. The scheme to determine the gradient is again, a central difference scheme. This is because it is a more accurate average approximation from both sides. The norm of the gradient is a point-wise norm.

$$u(x, t + \Delta t) = \frac{u(x + \Delta x, y, t) - u(x - \Delta x, y, t)}{2}$$

$$u(y, t + \Delta t) = \frac{u(x, y + \Delta y, t) - u(x, y - \Delta y, t)}{2}$$

$$\|\nabla u\| = \sqrt{u_x(x, y, t)^2 + u_y(x, y, t)^2}$$

(Equation 8)

To determine the length functions, the following scheme is the 2-D case:

$$D_x^- L = \frac{L[i, j] - L[i - 1, j]}{h_x}$$

$$D_x^+ L = \frac{L[i + 1, j] - L[i, j]}{h_x}$$

$$D_x^- L = \frac{L[i, j] - L[i, j - 1]}{h_y}$$

$$D_x^+ L = \frac{L[i, j + 1] - L[i, j]}{h_y}$$

(Equation 9)

which compute the spatial differences in a upwind differencing scheme. There must be a consideration for upwind differences due to the finite differencing scheme model but there is no

need for entropy conditions because these are linear PDEs and the trajectories never intersect. The trajectories, however, do flow in opposite directions for each starting boundary, so a consideration would be for choosing the proper differencing scheme such that L_0 in the backward direction use upwind along the characteristic passing through the grid point.

The upwinding differencing scheme is presented in (9) is simplified here:

$$1 = T_x[i,j]\begin{cases} D_x^- L_0[i,j], & -T_x[i,j] < 0 \\ D_x^+ L_0[i,j], & otherwise \end{cases}$$
$$+ T_y[i,j]\begin{cases} D_y^- L_0[i,j], & -T_y[i,j] < 0 \\ D_y^+ L_0[i,j], & otherwise \end{cases}$$

(Equation 10)

$$1 = T_x[i,j]\begin{cases} D_x^+ L_1[i,j], & -T_x[i,j] < 0 \\ D_x^- L_1[i,j], & otherwise \end{cases}$$
$$+ T_y[i,j]\begin{cases} D_y^+ L_1[i,j], & -T_y[i,j] < 0 \\ D_y^- L_1[i,j], & otherwise \end{cases}$$

(Equation 11)

Therefore, the total discretized upwinding scheme which define L_0 and L_1 is presented here:

$$L_0[i,j] = \frac{1 + |T_x| L_0[i \mp 1, j] + |T_y| L_0[i, j \mp 1]}{|T_x| + |T_y|}$$

(Equation 12)

$$L_1[i,j] = \frac{1 + |T_x| L_1[i \pm 1, j] + |T_y| L_1[i, j \pm 1]}{|T_x| + |T_y|}$$

(Equation 13)

V. Implementing Thickness Algorithm in MATLAB

The algorithm used to code the thickness of a region is:

1. Generate an image matrix of a desired region area (ellipse encircling a small circle)
2. Set $u(d_0 R) = 0$ and $u(d_1 R) = 1$ and the intermediate region to be 0.5
3. Solve Laplace's equation over R ($\Delta u = 0$)
4. Compute unit Tangent field: $\vec{T} = \frac{\nabla u}{\|\nabla u\|}$ with $\Delta x = 1, \Delta y = 1, \Delta t \leq \frac{1}{2}$ via central difference (8)
5. Algorithm for thickness using Iterative Relaxation [1]
    a. Set $L_0 = L_1 = 0$ at all grid points.
    b. Use (12) and (13) to update L_0 and L_1 at all points inside R.
    c. Repeat Step 2 until the values L_0 and L_1 converge.

In order to implement the described algorithm in MATLAB, $h_x = h_y = 2$ and $\Delta x = \Delta y = 1$ and $\Delta t = \frac{1}{2}$. The CFL condition and step given allows convergence and stability for the discretization scheme discussed above. The minimum number of iterations used in this experiment was 1000. More is documented in Appendix A.

VI. Results

For this experiment, the moving standard deviation of each length function is plotted against the length function L_0. The sampling rates used are 1:1 and 2:1 where the ratio represents the number of related interpolated points to each corresponding measured size of matrix.

In this test, the elliptic annulus constructed varies in its major radii by 100 while its minor radii remains constant at 50 and the inner boundary is 25 (all numeric values are rated by pixels). The annulus is increased linearly by 100 from 200 to 900 and thickness is computed for each case to determine a moving standard deviation from point to point. Since it is assumed that computing (13) will generate a unique, one to one mapping of the boundary points, only the standard deviation for the length function L_0 is considered because they will generate the identically same plot but backwards. As the major radii is increased, the discretization schemes favor from a linear increase of iterations; the tested and constant number of iterations was chosen to be 3000. The standard deviation for the thickness (4) function is not shown because it computed to be 0 on the major radii axis – which is consistent to what the mathematical model indicates in that the sum of the length functions should compose of the thickness for each corresponding trajectory.

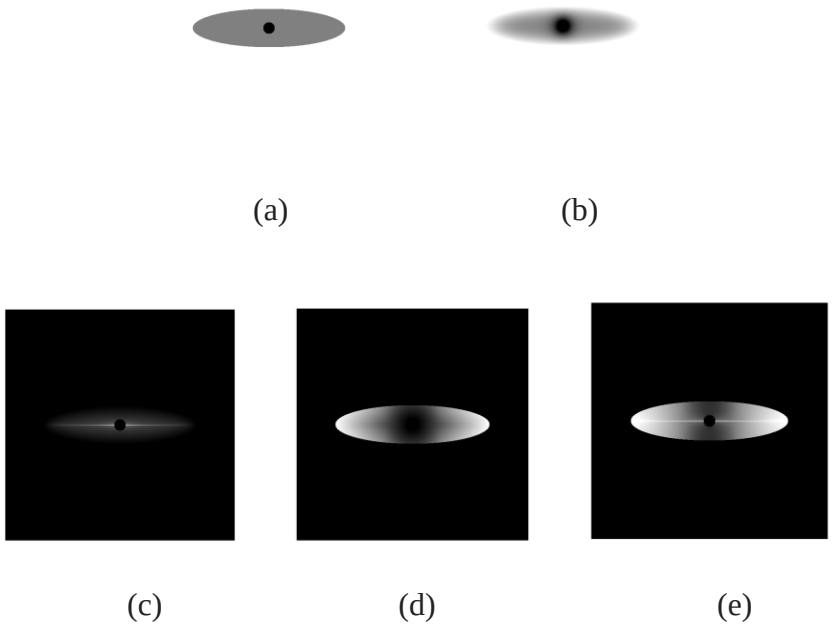


Figure 1. Major radii of 200 (a) The initial region with values set to 0 for the inner boundary, 0.5 for the inner, 1 for the outer. (b) Is the solved Laplace's equation in the region R. (c) Length L_1 (d) Length L_0 (e) Thickness $L_0 + L_1$

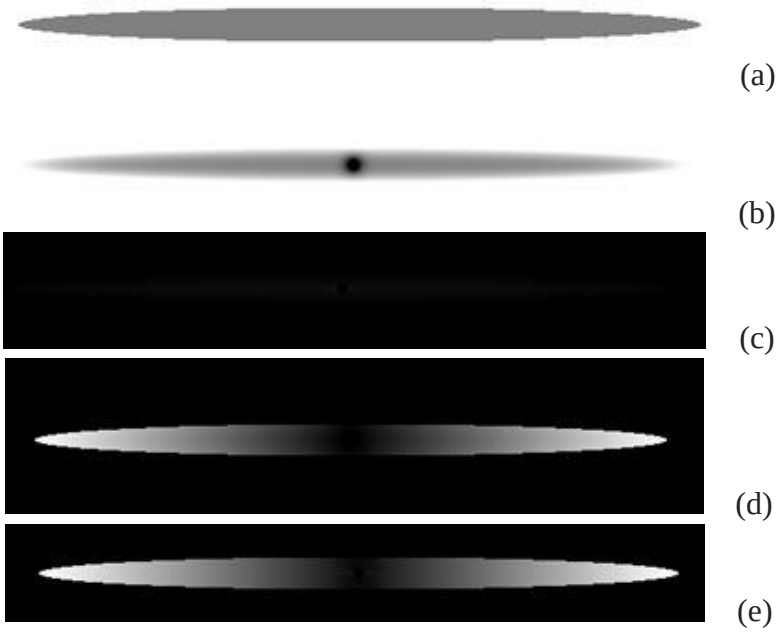
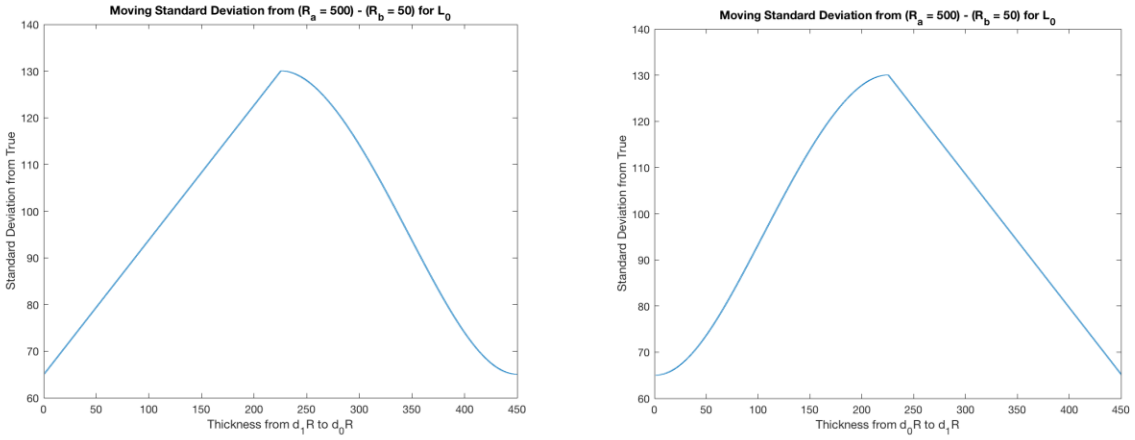


Figure 2. Similar figures to Figure 1, but for a major radii of 1000

Figures 1 and 2 represent the range of major radii used in this experiment. For the sake of brevity, only relevant plots will be shown here to provide insight for concise conclusions.



(a) (b)

Figure 3. (a) Standard deviation of L_0 mapped to the eccentricity of the annulus. (b) Standard deviation of L_1 mapped to the eccentricity of the annulus

In Figure 3a, the plot represents the standard deviation of L_0 mapped against the progressive thickness values of the length function. Figure 3b represents the similar graph, but for L_1 in order to demonstrate that the plots indeed match the mathematical model in that L_0 and L_1 are unique, one to one maps and are therefore the reverse of each other.

The eccentricity metric is:

$$eccentricity = \sqrt{r_{major}^2 + r_{minor}^2}$$

(Equation 14)

which defines an elliptical annulus in a meaningful way.

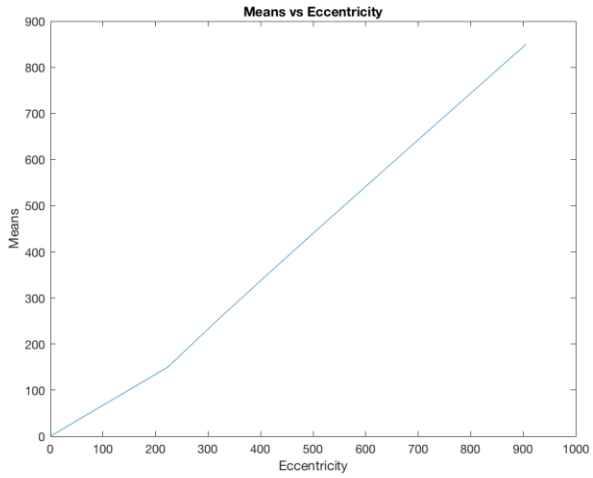


Figure 4. Means vs. Eccentricity (Sample rate 1)

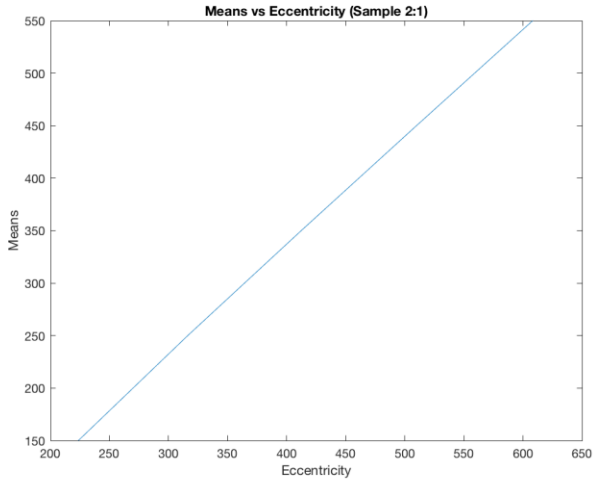


Figure 5. Means vs Eccentricity (Sample rate 2)

Figure 4 shows the relationship between the major radii axis means vs the eccentricity of the thickness function is a linear relationship. The plot helps to conclude that the length functions perform robustly for a consecutively growing major radii width. This is because of the one to one and unique property of the corresponding trajectories. Figure 5 also proves that even by

increasing the sample rate that it the linear relationship remains consistent to that of the 1:1 sample rate.

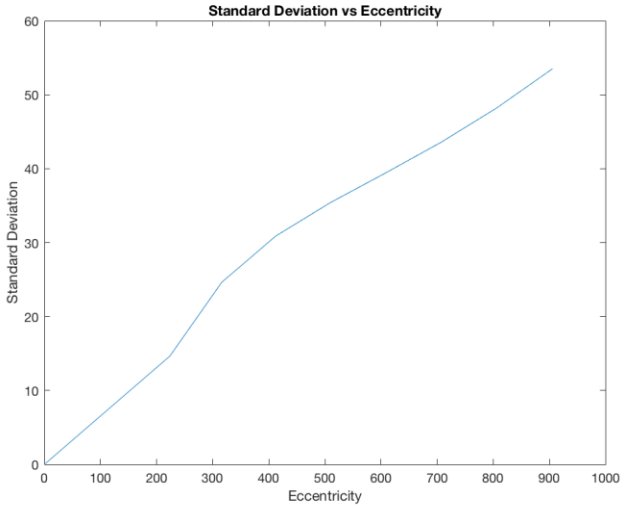


Figure 6. Standard Deviation vs Eccentricity (Sample rate 1)

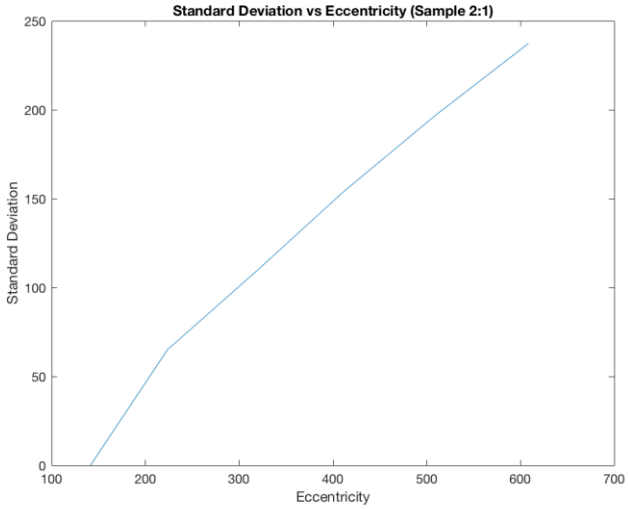


Figure 7. Standard Deviation vs. Eccentricity (Sample rate 2)

Figure 6 represents the standard deviation of the major radii axis vs the eccentricity of the elliptic annulus. The plot shows another linear relationship in that as the major radii increases (affecting the eccentricity), the standard deviation increases linearly as well. This demonstrates that with constant iterations and inner boundary and minor radii, the length functions perform robustly. Figure 7 also demonstrates similarly that with two times the sample rate, that the linear relationship is still maintained; however, the plot is cut short due to excessive time constraints.

VII. Conclusion

Overall, the thickness algorithm appears to be very robust, even against large regions and boundaries. The results qualitatively prove that the length functions perform as to expectations in that the basis of the PDEs are based on mathematical properties that are proven. However, some weaknesses associated to PDE solving is typically computation time. Computing large scale data and solving for the harmonic function $\Delta u = 0$ proved to take the bulk of the time, whereas solving the length functions proved to be relatively shorter. Calculating length functions typically took around 500 iterations to fully converge quickly whereas the larger data sets required iterations greater than 2000 to develop a fully functioning model. The fact that the length functions are inherently tied to the calculation of such harmonic functions degrade the overall time performance of the product. The data set for the 2:1 sample rate is incomplete as a result of the time performance of the inability to fully compute all consecutive samples.

The next step would be to optimize all computational solutions to converge to a tolerance such that convergence would be asymptotically/exponentially stable to a known value. However, without such knowledge or theory in optimizing discretizations, the next approach taken would be to have the user circle around the desired region of interest and the computational model would effectively be faster due to less looping around irrelevant values. Also, I would implement the cyclically alternating Gauss-Seidel algorithm which is a different and more efficient thickness algorithm used in [1]. If more practice and time were given for this project, level-set thickness calculation would be implemented in this program.

VIII. References

[1] A. Yezzi Jr., J. Prince. An Eulerian PDE Approach for Computing Tissue Thickness. IEEE Transactions on Medical Imaging, Vol. 22, NO. 10

Appendix A: MATLAB Code

```
% u must be a square matrix
% Set region of interest to 0.5 and other region potential from 0 to 1
% max_win is for quicker computation window (usually largest foci)
% floor(n/2)-max_win+2:ceil(n/2)+max_win

function [u, L0, L1] = thickness(u, max_iter)
u_prev = u;
input_u = u;
n = length(u);
% field_e = [];
n_iter = 0;
delta_t = .25;
% if max_win <= 0
%     max_win = n;
% end

if max_iter <=0
    max_iter = 5000;
end

% Compute laplace(U) = 0

while n_iter < max_iter
    n_iter = n_iter+1;
    for i = 2:n-1
        for j = 2:n-1
            if input_u(i,j) == .5
                u_prev(i,j) = u(i,j)+ delta_t*(u(i+1,j) + u(i-1,j) + u(i,j+1) +u(i,j-1) - 4*u(i,j));
            end
        end
    end

    u = u_prev;
%     grad_phi_x = (phi(2:end,:)-phi(1:end-1,:))./2;
%     grad_phi_y = (phi(:,2:end)-phi(:,1:end-1))./2;
%     field_e(n_iter) = sqrt(sum(sum(grad_phi_x.^2))+sum(sum(grad_phi_y.^2)));
%     if n_iter == 1
%         e_ratio = 1;
%     else
%         e_ratio = (field_e(n_iter-1)-field_e(n_iter))/2;
%     end
%     fprintf('%8.6f | %8.6f\n', e_ratio, field_e(n_iter))
end

% Compute Tangent
grad_phi_x = zeros(size(u));
grad_phi_y = zeros(size(u));
T_x = zeros(size(u));
T_y = zeros(size(u));
for i = 2:n-1
    for j = 2:n-1
        if input_u(i,j) == .5
            grad_phi_x(i,j) = .5*(u_prev(i+1,j)-u_prev(i-1,j));
            grad_phi_y(i,j) = .5*(u_prev(i,j+1)-u_prev(i,j-1));
            norm_u = sqrt(grad_phi_x(i,j).^2+grad_phi_y(i,j).^2);
            T_x(i,j) = grad_phi_x(i,j)./norm_u;
            T_y(i,j) = grad_phi_y(i,j)./norm_u;
        end
    end
end
```

```
L0 = zeros(size(u));
L1 = zeros(size(u));
dL0 = L0;
dL1 = L1;
iter = 0;
while iter < max_iter
    iter = iter + 1;
    for i = 2:n-1
        for j = 2:n-1
            if input_u(i,j) == .5
                % upwind differencing
                if -T_x(i,j) < 0
                    L0_x = L0(i-1,j);
                    L1_x = L1(i+1,j);
                else
                    L0_x = L0(i+1,j);
                    L1_x = L1(i-1,j);
                end
                if -T_y(i,j) < 0
                    L0_y = L0(i,j-1);
                    L1_y = L1(i,j+1);
                else
                    L0_y = L0(i,j+1);
                    L1_y = L1(i,j-1);
                end
                dL0(i,j) = (1 + abs(T_x(i,j))*L0_x + abs(T_y(i,j))*L0_y)/(abs(T_x(i,j))+abs(T_y(i,j)));
                dL1(i,j) = (1 + abs(T_x(i,j))*L1_x + abs(T_y(i,j))*L1_y)/(abs(T_x(i,j))+abs(T_y(i,j)));
            end
        end
    end
    L0 = dL0;
    L1 = dL1;
end

end
```