Image Restoration using Geometrically Stabilized Heat Equation

CS 663 Project Report
Susmit Sarkar (210010063)
Vedant Yadav (210110116)
Atharva Inamdar (210010017)

Introduction

In this project, we have implemented a research paper which has attempted to solve the problem of deblurring of an image using the reverse heat equation. We consider the stabilization of the reverse heat equation. We model the blurring as a convolution and perform the stabilization by adding a normal component of the heat equation in the normal

$$Y(x) = \int U(t)h(x-t)dt$$

$$\frac{\partial u}{\partial t} = c\triangle u,\ u(x,0) = I_0(x)$$

Here, *h(x)* is the kernel, *U(x)* is the original image, *Y(x)* is the blurred image.

Blurring is proportion to the laplacian.

Stabilized Backward Heat Equation

$$\frac{\partial u}{\partial t} = c\triangle u$$
$$u(x,\tau) = I(x)$$

This is the reverse heat equation. *I(x)* is the observation and *c* is the diffusion coefficient

$$u(x,0) = I_0(x).$$

We need to solve this for obtaining the unblurred image

$$\frac{\partial u}{\partial t} = -c\triangle u,\ u(x,0) = I(x).$$

Thus, the unblurred image will be obtained by reversing the heat equation as shown here.

Stabilized Backward Heat Equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial \eta^2} + \frac{\partial^2 u}{\partial \zeta^2} = u_{\eta\eta} + u_{\zeta\zeta}$$

$$u_{\eta\eta} = \frac{u_{xx}u_x^2 + 2 * u_{xy}u_x u_y + u_{yy}u_y^2}{u_x^2 + u_y^2}$$

$$u_{\zeta\zeta} = \frac{u_{xx}u_x^2 - 2 * u_{xy}u_x u_y + u_{yy}u_y^2}{u_x^2 + u_y^2}$$

In order to reduce sudden spike in the laplacian near the edges, we convert the heat equation into into its geometric form.

η and ζ represent the normal and tangential directions respectively.

Stabilized Backward Heat Equation

Since the diffusion along the normal diffuses across the edges, and diffusion along tangent occurs along the edges, blurring is causes more by the normal diffusion. Thus, for stabilization of reverse diffusion, the reverse diffusion across the edges has to be done at a slower rate as compared to diffusion along the edge.

$$u_t = -c\triangle u + \beta u_{\eta\eta}$$

As shown above, we add a forward component of diffusion along the normal. We ensure $c > \beta$ for overall forward diffusion.

Stopping Criterion

$$\kappa = \frac{u_{xx}u_x^2 - 2u_{xy}u_x u_y + u_{yy}u_y^2}{(u_x^2 + u_y^2)^{3/2}}$$

$$\kappa_t > \theta.$$

We have to estimate *u(x,0).* An observation can be made that the heat equation is not valid for *t < 0.* Thus, we need to stop at *t = 0.*

We use the degeneration of the curvature as the criterion for stopping.

The process is stopped when the stopping criterion exceeds a certain threshold.

Implementation Details

- The implementation has been done in MATLAB.
- The boundary condition used for the reverse heat equation is the Neumann Boundary condition, i.e. the gradient is zero along the boundary.
- The value of *c* and β is chosen to be 0.2 and 0.002 respectively.
- The value of θ is chosen to be 0.3
- The values are picked from the experiments described in the paper, which were chosen empirically.

Implementation Details

- We have implemented the reverse heat equation in the code by discretizing the derivatives.
- We update each pixels in the image using until the stopping criteria is met for that individual pixel.
- The stopping criteria is based on the divergence of the curvature, and the algorithm stops when it exceeds a threshold.

```
while pixels_reached_threshold ~= ones(H-2,W-2)

    I_new = zeros(H,W);

    for i = 2:1:H-1
        for j = 2:1:W-1 %WE donot consider the boundary
                        %as the intensity doesn't update due to 0 gradient
            %Computing the lapclacian
            %Directional derivaitves
            I_x = I(i+1,j) - I(i,j);
            I_y = I(i,j+1) - I(i,j);

            I_xx = I(i+1,j) + I(i-1,j) - 2*I(i,j);
            I_yy = I(i+1,j) + I(i-1,j) - 2*I(i,j);
            I_xy = I(i+1,j+1) + I(i,j) - I(i+1,j) - I(i,j+1);

            %Checking the stopping criterion
            k = (I_xx*I_x^2 - 2*I_xy*I_x*I_y + I_yy*I_y^2)/(I_x^2 + I_y^2)^(3/2);
            if k > threshold
                pixels_reached_threshold(i-1,j-1) = 1;
            end

            if pixels_reached_threshold(i-1,j-1) == 0
                %Diffusion along the normal and tangent to edges
                I_nn = (I_xx*I_x^2 + 2*I_xy*I_x*I_y + I_yy*I_y^2)/(I_x^2 + I_y^2);
                I_tt = (I_xx*I_x^2 - 2*I_xy*I_x*I_y + I_yy*I_y^2)/(I_x^2 + I_y^2);

                Laplacian = I_nn + I_tt;

                %Updating the image pixel
                I_new(i,j) = I(i,j) - c*Laplacian + beta*I_nn;
            end

        end
    end
```

Results

Here is the deblurring algorithm proposed in the paper applied to the barbara image:

Original image

Blurred image

Deblurred image

Here, the image was blurred with a gaussian filter(sigma = 40) and deblurred using modified parameters (c = 5, beta = 0.002, threshold = 8)