Diffusion Filtering

Ajit Rajwade, CS 663, IITB

1 Isotropic Heat Equation

Consider a rectangular bucket filled with some lukewarm fluid. Suppose, I insert a hot object into this bucket. Over a period of time, the temperature of this object will reduce, starting from the interface between the fluid and the object towards the interior of the object. The temperature of the fluid will start increasing in a reverse direction. This process will continue until the entire system attains a constant temperature. Here of course, we are assuming that our system consists of only the bucket with water and the new object, ignoring the surroundings. Let the temperature at point (x, y) (we will assume this is a 2-D system for simplicity, though nothing here will fundamentally change if we considered 3-D) at time instant t be denoted as $I(x, y, t)$. The change in the temperature of any point is given by the following partial differential equation (PDE), called the (isotropic) heat equation:

$$\frac{\partial I(x,y,t)}{\partial t} = \rho\left(\frac{\partial^2 I(x,y,t)}{\partial x^2} + \frac{\partial^2 I(x,y,t)}{\partial y^2}\right). \tag{1}$$

The term on the RHS is the Laplacian operator and $\rho > 0$ is a constant. In the context of image processing, the above equation is simulated on images (rather, on image intensities) to smooth away noise, and here 'image intensity' is the equivalent of temperature. Now, the digital approximation of the image Laplacian is as follows:

$$\begin{aligned} \frac{\partial^2 I(x,y,t)}{\partial x^2} + \frac{\partial^2 I(x,y,t)}{\partial y^2} &= \quad (2) \\ I(x,y+1) - 2I(x,y) + I(x,y-1) + I(x+1,y) - 2I(x,y) + I(x-1,y) & \\ = I(x,y+1) + I(x,y-1) + I(x+1,y) + I(x-1,y) - 4I(x,y). & \end{aligned}$$

This equation can be simulated in a digital computer by iterating the following from $t = 0$ to $t = T$, where $T > 0$:

$$I(x,y,t+1) = I(x,y,t)+\rho[I(x,y+1,t)+I(x,y-1,t)+I(x+1,y,t)+I(x-1,y,t)-4I(x,y,t)]. \tag{3}$$

Note that we have a third 'time' coordinate because the intensity of the image (or the temperature of the system) is changing over time. The initial image (upon which the diffusion is performed) is $I(x,y,0)$. For a numerically robust simulation, the value of ρ should be chosen to be small - like 0.1 or 0.01 (otherwise some stability issues arise, the precise reasons for which we will not delve into during this course, as it is better done in a complete course on numerical partial differential equations). Please note that we have seen the image Laplacian before in the context of edge detection. And we also saw an image sharpening algorithm where we deducted the value of the Laplacian at each point. In this case, we are repeatedly adding the value of the Laplacian, and we can see from the code '`heat_equation.m`' that this gives us a form of intensity blurring.

Note that our image is defined over a finite, bounded domain of the form $[1,H]\times[1,W]$ (height = H, width = W). As intensity values are not defined outside this domain, the question arises: What are the gradient values along the boundary of the image? A commonly used convention is that the derivative of the image in a direction perpendicular to its boundary is zero. This is one form of the Neumann boundary conditions. It is implemented in practice by padding the image by a one-pixel thick border outside its boundary. The padded pixel values are obtained by reflection across the image boundary, i.e. $\forall x, I(x,H+1) = I(x,H); \forall x, I(x,0) = I(x,1); \forall y, I(W+1,y) = I(W,y); \forall y, I(0,y) = I(1,y)$.

As seen by executing the code '`heat_equation.m`', simulating this PDE on an image leads to blurring of the image. It turns out that simulation of this PDE is equivalent to convolution with a Gaussian. We shall prove this result in 1-D (as a homework, you may try the 2-D version yourself). We again consider an image $I(x,t)$. To this effect, consider the original PDE:

$$\frac{\partial I(x,t)}{\partial t} = \rho\frac{\partial^2 I(x,t)}{\partial x^2}. \tag{4}$$

Taking Fourier transforms on both sides using $\mathcal{F}$ as notation for the Fourier operator, we have

$$\mathcal{F}(\frac{\partial I(x,t)}{\partial t}) = \rho\mathcal{F}[(\frac{\partial^2 I(x,t)}{\partial x^2}]. \tag{5}$$

The LHS can be written as follows:

$$\mathcal{F}(\frac{\partial I(x,t)}{\partial t}) = \int_{-\infty}^{+\infty} \frac{\partial I(x,t)}{\partial t} e^{-j2\pi\mu x} dx \tag{6}$$
$$= \frac{\partial}{\partial t}\int_{-\infty}^{+\infty} I(x,t) e^{-j2\pi\mu x} dx$$
$$= \frac{\partial \hat{I}(\mu,t)}{\partial t}$$

where $\hat{I}(\mu,t)$ stands for the Fourier Transform of I at time instant t, and μ stands for the frequency. Note that we could take the partial derivative outside the integral only because the second term $e^{j2\pi\mu x}$ is independent of t. The RHS of Eqn 5 is given as follows, using integration by parts:

$$\mathcal{F}[\rho(\frac{\partial^2 I(x,t)}{\partial x^2}] = \rho(I_x(x,t)e^{-j2\pi\mu x})\Big|_{-\infty}^{\infty} - \rho\int_{-\infty}^{+\infty} I_x(x,t)[-j2\pi\mu]e^{-j2\pi\mu x}dx \tag{7}$$
$$= 0 + \rho[j2\pi\mu]\int_{-\infty}^{+\infty} I_x(x,t)e^{-j2\pi\mu x}dx(\text{ using } I_x(-\infty,t) = I_x(\infty,t) = 0) \tag{8}$$
$$= \rho j2\pi\mu\Big((I(x,t)e^{-j2\pi\mu x})\Big|_{-\infty}^{\infty} - \int_{-\infty}^{+\infty} I(x,t)[-j2\pi\mu]e^{-j2\pi\mu x}dx\Big) \tag{9}$$
$$= 0 - 4\pi^2\mu^2\rho\hat{I}(\mu,t) \text{ using } I(\infty,t) = I(-\infty,t) = 0 \text{ and by defn. of Fourier transform .} \tag{10}$$

In fact, this can be generalized in the following manner, for any integer $n > 0$:

$$\mathcal{F}[(\frac{\partial^n I(x,t)}{\partial x^n}] = (j2\pi\mu)^n\hat{I}(\mu,t). \tag{11}$$

Now, eqn 5 becomes:

$$\frac{\partial \hat{I}(\mu,t)}{\partial t} = -4\pi^2\mu^2\rho\hat{I}(\mu,t) \tag{12}$$

which is really just an ordinary differential equation (ODE) since we have eliminated the derivatives in x and have only the derivatives in t. This ODE can be solved as follows:

$$\frac{d\hat{I}(\mu,t)}{dt} = -4\pi^2\mu^2\rho\hat{I}(\mu,t) \tag{13}$$
$$\therefore \frac{d\hat{I}(\mu,t)}{\hat{I}(\mu,t)} = -4\pi^2\mu^2\rho dt \tag{14}$$
$$\therefore \log\hat{I}(\mu,t) = -4\pi^2\mu^2\rho t + c'(\mu) \text{ taking the integrals on both sides} \tag{15}$$
$$\therefore \hat{I}(\mu,t) = e^{-4\pi^2\mu^2\rho t}c(\mu) \tag{16}$$

where $c'(\mu)$ is a constant of integration, which could be function of μ, but will always be completely independent of t. Also $c(\mu) = e^{c'(\mu)}$. Setting $t = 0$ in the above equation, we now have:

$$\hat{I}(\mu, 0) = c(\mu), \tag{17}$$

which finally yields:

$$\hat{I}(\mu, t) = e^{-4\pi^2\mu^2\rho t}\hat{I}(\mu, 0). \tag{18}$$

Note that the RHS of the above equation contains the product of two Fourier transforms, one being $\hat{I}(\mu, 0)$ and the other being $e^{-4\pi^2\mu^2\rho t}$. Using the convolution theorem to transfer to the spatial domain, will give rise to the convolution of two functions:

$$I(x,t) = \frac{1}{\sqrt{4\pi\rho t}}e^{-\frac{x^2}{4\rho t}} * I(x,0) \tag{19}$$

where the former function is a Gaussian with mean 0 and standard deviation $\sqrt{2\rho t}$. Note that this was possible because of the following result pertaining to Fourier transforms of a Gaussian:

$$\mathcal{F}(e^{-ax^2}) = \sqrt{\frac{\pi}{a}}e^{-\frac{\pi^2\mu^2}{a}}. \tag{20}$$

You will find a proof of the above result at here, but the proof is not essential for the rest of the material here, or for the rest of this course.

Hence we can say that:

$$\mathcal{F}^{-1}(e^{-4\pi^2\mu^2\rho t}) = \mathcal{F}^{-1}(e^{-\frac{\pi^2\mu^2}{1/(4\rho t)}}) \tag{21}$$

$$= \sqrt{\frac{1}{4\pi\rho t}}e^{-\frac{x^2}{4\rho t}}[\text{ Note that: } a = \frac{1}{4\rho t}] \tag{22}$$

$$= \frac{1}{\sqrt{2\pi}\sqrt{2\rho t}}e^{-\frac{x^2}{2(2\rho t)}} \tag{23}$$

Thus, we have proved that executing the heat equation is equivalent to Gaussian convolution. As mentioned earlier, the standard deviation of the Gaussian is proportional to $\sqrt{t}$. The longer you run the PDE, the more the standard deviation of the Gaussian.

Now, you may then wonder why we should bother about the heat equation at all. After all, we have proved that executing the heat equation is equivalent to Gaussian convolution. In fact, the convolution operation is computationally faster and more numerically stable than executing the heat equation. Well, we still study it for two reasons. Firstly, this is a basic framework for image filtering. There are several different types of PDEs (and amongst them, the heat equation is one of the most basic PDEs), all of which perform varied operations on an image. Most of them *cannot* be expressed in the form of convolutions or any analytic, closed-form formulae. The only way to solve most of them, is to simulate them numerically. Secondly, the overall idea of simulating diffusion equations on an image is quite interesting!

2 Anisotropic Diffusion

For this section, please run the code '`perona_malik.m`' from the folder. The heat equation (or Gaussian blurring) will get rid of the noise in the image, but it will smear away the edges as well. In other words, we need to introduce some insulators in the earlier fluid and hot body example which will reduce the amount of diffusion across edges. The stronger the edge, the less the diffusion across it should be. There are many ways to modify our original heat equation. One of the most popular modifications to it was proposed by Perona and Malik in 1990. If you look at the heat equation, it can also be written as:

$$\frac{\partial I(x,y,t)}{\partial t} = \rho(\frac{\partial^2 I(x,y,t)}{\partial x^2} + \frac{\partial^2 I(x,y,t)}{\partial y^2}) = \text{div}(\rho \nabla I(x,y,t)). \tag{24}$$

The divergence of a vector field (at a point) is the measure of "outgoing-ness" of the vector field at that point. It tells you how much heat energy (say) is going out of that point or coming into it. Consider a vector field $\mathbf{v}(x,y) = (v_1(x,y), v_2(x,y))$. Its divergence is given as $\text{div}(\boldsymbol{v}(x,y)) \triangleq \frac{\partial v_1(x,y)}{\partial x} + \frac{\partial v_2(x,y)}{\partial y}$. Note that in the above equation, $\nabla I(x,y,t)$ is a vector in 2D for every (x,y,t)!

Note that in the diffusion PDE above, the term ρ (which controls the speed of diffusion) is a constant, independent of x and y. There is no reason this should be so. It could very well be forced to depend on the spatial location. In the Perona and Malik PDE, instead of allowing the image to evolve as per the divergence of the gradient vector, they evolve it as per the divergence of the gradient vector times a 'diffusivity function' denoted as $g(x,y,t)$. This diffusivity function is a decreasing function of the gradient magnitude at a given point and at a given time, i.e. we have $g(x, y, t)$ to be one of the following forms:

$$g(x, y, t) = e^{-\frac{\|\nabla I(x,y,t)\|^2}{K}} \text{ OR} \tag{25}$$
$$g(x, y, t) = \frac{1}{1 + \frac{\|\nabla I(x,y,t)\|^2}{K}}.$$

In both these formulations, we see that the higher the gradient magnitude, the lower the diffusivity. The diffusivity function is computed afresh in every iteration of the PDE, because the intensities (and hence the gradient magnitudes) are all evolving with time. The diffusivity is parameterized by K, which governs the rate of diffusion. If K is small, even smaller edges will be preserved, but you may end up undersmoothing the image. If K is large, you will be able to erase noise faster, but you might lose some of the weaker edges in the bargain. For now, let us just stick to the notion that K is creative user choice. The Perona Malik PDE is now written as:

$$\frac{\partial I(x, y, t)}{\partial t} = \text{div}(g(x, y, t)\nabla I(x, y, t)) \tag{26}$$
$$= g(x, y, t)(\frac{\partial^2 I(x, y, t)}{\partial x^2} + \frac{\partial^2 I(x, y, t)}{\partial y^2}) + \nabla I(x, y, t)\cdot\nabla g(x, y, t).$$

Again note that $\nabla I(x, y, t)$ and $\nabla g(x, y, t)$ are vectors (with one component each, in the X and Y directions) and '$\cdot$' stands for the dot product.

When you execute the heat equation for several iterations, you will end up with a constant intensity image. When you execute the Perona Malik equation (cf: '`perona_malik.m`'), you will be able to preserve significant edges (depending upon your choice of K) and yet erase the noise. If the Perona Malik PDE is run for sufficiently many iterations, you end up with a piecewise constant intensity image, which resembles a cartoon. The initial purpose of this PDE was noise removal, a task which it performs quite well, but then it also erases weaker edges and textures along with it (depending upon K).

3 Convolutions?

The heat equation is equivalent to Gaussian convolution, but there is no convolution equivalent for the Perona Malik equation. This is for the simple reason that the output of only linear and time-invariant (or space-invariant) filters can be represented as a convolution between the input and the impulse response of the filter. In case of the Perona Malik PDE, the amount of diffusion at different points in the image is necessarily different (by design!) depending on the local intensity gradient magnitude. Neither is it a linear system, so convolution does not apply for two different reasons!

4 PDEs: Other Applications

PDEs can be used for applications other than noise removal. For instance, we may want to simulate the effect of motion blur, and a diffusion PDE can be designed specifically for this purpose. To this end, we first define a vector field $\hat{v}(x,y) = [v_1(x,y), v_2(x,y)]$ which represents the direction in which we want to blur the image at each point. This vector field may be a constant, i.e. at every point, we may wish the blur direction to be the same, in which case we would write $\forall x, \forall y, \hat{v}(x,y) = [v_1, v_2]$. The relevant PDE for this purpose will be:

$$\frac{\partial I(x,y,t)}{\partial t} = \rho \frac{\partial^2 I(x,y,t)}{\partial \hat{v}(x,y)^2}. \tag{27}$$

In this case, we are taking the second **directional** derivative of $I(x,y)$ in the direction $\hat{v}(x,y)$, and this is denoted as $\frac{\partial^2 I(x,y,t)}{\partial \hat{v}(x,y)^2}$ or as $I_{\hat{v}\hat{v}}(x,y)$. Executing the aforementioned PDE will blur the image in the chosen directions, as we showed in class. Note that this PDE is **different** from the Perona Malik PDE. Now the question arises, how do we compute these directional derivatives?

Let's take a look at the first directional derivative of $I(x,y)$ in the direction $\hat{v}(x,y)$, which is denoted as $\nabla_{\hat{v}(x,y)} I(x,y)$. It is given by $\nabla I(x,y) \cdot \hat{v}(x,y) = v_1(x,y) I_x(x,y) + v_2(x,y) I_y(x,y)$. The second directional derivative is now given as follows:

$$\frac{\partial^2 I(x,y,t)}{\partial \hat{v}(x,y)^2} = \nabla_{\hat{v}(x,y)}(\nabla_{\hat{v}(x,y)} I(x,y)) \tag{28}$$

$$= \nabla_{\hat{v}(x,y)}(v_1(x,y) I_x(x,y) + v_2(x,y) I_y(x,y)). \tag{29}$$

Taking partial derivatives and further simplifying, this yields

$$\begin{aligned}\frac{\partial^2 I(x,y,t)}{\partial \hat{v}(x,y)^2} &= \frac{\partial}{\partial x}(v_1(x,y)I_x(x,y) + v_2(x,y)I_y(x,y))v_1(x,y) + \\ &\quad \frac{\partial}{\partial y}(v_1(x,y)I_x(x,y) + v_2(x,y)I_y(x,y))v_2(x,y).\end{aligned} \tag{30}$$

Further simplification yields:

$$\frac{\partial^2 I(x,y,t)}{\partial \hat{v}(x,y)^2} = v_1^2(x,y)I_{xx}(x,y) + 2I_{xy}(x,y)v_1(x,y)v_2(x,y) + v_2^2(x,y)I_{yy}(x,y). \tag{31}$$

Plugging this result back into the earlier PDE, we have the following:

$$\frac{\partial I(x,y,t)}{\partial t} = \rho(v_1^2(x,y)I_{xx}(x,y) + 2I_{xy}(x,y)v_1(x,y)v_2(x,y) + v_2^2(x,y)I_{yy}(x,y)). \tag{32}$$

Look at the code '`interesting_blurs.m`' that implements this PDE and the effect it has on an image.