PDEs for Image Processing

Martin Robinson

August 15, 2014

1 Image Domain

A grey-scale image is simply a two dimensional NxM array of pixel values $u_{i,j}$, where $i \in 1 \dots L_x$ and $j \in 1 \dots L_y$. However, we can also consider that $u_{i,j}$ are samples of some continuous function $u(x, y)$ defined on the domain $(x, y) \in \Omega \subset \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of positive real numbers. For the sake of simplicity, we set the domain so that $\Omega = \{x, y \in \mathbb{R}^+ : x < L_x \wedge y < L_y\}$, so that the distances between neighbouring pixels will be equal to 1.

In this case, an image consists of samples from this continuous function

$$u_{ij} = u(\mathbf{x}_{i,j}) \tag{1}$$

$$\mathbf{x}_{i,j} = \begin{pmatrix} i - 0.5 \\ j - 0.5 \end{pmatrix} \tag{2}$$

We can consider a "real" image to consist of both a source term u, which we are trying to recover, and an additive noise term n

$$u_{i,j} = u(\mathbf{x}_{i,j}) + n(\mathbf{x}_{i,j}) \tag{3}$$

The problem then becomes the recovery of the source term u from the set of pixel values $u_{i,j}$. That is, we need to construct some sort of operator to remove the noise term n.

2 Signal Processing - Low-Pass Filtering

Assuming that the noise term is comprised of mainly high frequency terms and the source term is mainly low frequency terms, one might use a low-pass filter to (approximately) recover the source term.

A low-pass filter is a filter that removes or attenuates some of the higher frequency components of the original signal. The main parameter to the low-pass filter is the *cut-off* frequency, above which most of the attenuation is performed.

A low-pass filter can be implemented by convolution of the pixel values by a function f. In one dimension and in an infinite domain, the convolution operator is defined as

$$(f \star u)(x) = \int_{-\infty}^{\infty} f(x - x')u(x')dx' \tag{4}$$

A common low-pass filter is the *Gaussian* filter, which uses a gaussian function for f

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{5}$$

We can examine the frequency response of the gaussian filter by considering its *fourier transform* ($\mathcal{F}(f) = \hat{f}$), defined as

$$\hat{f}(k) = \int_{-\infty}^{\infty} f(x) e^{-ikx} dx, \tag{6}$$

where k is the mode frequency in units of radians per unit length. Note that this integral only converges if $f(x) \to 0$ fast enough as $x \to \pm\infty$. This transforms our original function from the spatial domain (x) to the frequency domain (k). It is equivilent to the fourier series in the limit of infinite domain size $L \to \infty$. The original function can be recovered by the inverse fourier transform

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(x) e^{ikx} dk. \tag{7}$$

Some fourier transform properties

1. $\mathcal{F}$ is a linear operator: $\mathcal{F}\{af + bg\} = a\mathcal{F}\{f\} + b\mathcal{F}\{g\}$
2. Shift property: $\mathcal{F}\{f(x+a)\} = e^{ika}\hat{f}(k)$.

$$\mathcal{F}f(x+a) = \int_{-\infty}^{\infty} f(x+a) e^{-ikx} dx = \int_{-\infty}^{\infty} f(x') e^{ika} e^{-ikx'} dx' = e^{ika}\hat{f}(k). \tag{8}$$

3. Derivative property: $\mathcal{F}\{\frac{df}{dx}\} = ik\hat{f}(k)$.

$$\mathcal{F}\frac{df}{dx} = \int_{-\infty}^{\infty} \frac{df}{dx} e^{-ikx} dx = \Big[f(x) e^{-ikx} \Big]_{-\infty}^{\infty} + ik \int_{-\infty}^{\infty} f(x) e^{-ikx} dx = ik\hat{f}(k). \tag{9}$$

4. Convolution property: $\mathcal{F}\{f(x) \star g(x)\} = \hat{f}(k)\hat{g}(k)$

$$\begin{aligned}
\mathcal{F}f \star g &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x - x') g(x') dx' \right] e^{-ikx} dx && (10)\\
&= \int_{-\infty}^{\infty} g(x') e^{-ikx'} \left[\int_{-\infty}^{\infty} f(x - x') e^{-ik(x-x')} dx \right] dx' && (11)\\
&= \int_{-\infty}^{\infty} g(x') e^{-ikx'} \left[\int_{-\infty}^{\infty} f(x - x') e^{-ik\xi} d\xi \right] dx' && (12)\\
&= \hat{f}(k)\hat{g}(k) && (13)
\end{aligned}$$

5. Fourier transform of gaussian: $\hat{f}(k) = e^{-\frac{k^2}{2\sigma^2}}$

$$\begin{aligned}
\hat{f}(k) &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} e^{-ikx} dx && (14)\\
\frac{d}{dk}\hat{f}(k) &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} (-ix) e^{-ikx} dx && (15)\\
&= i\sigma^2 \int_{-\infty}^{\infty} \left(\frac{d}{dx} e^{-\frac{x^2}{2\sigma^2}} \right) e^{-ikx} dx && (16)\\
&= -k\sigma^2 \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} e^{-ikx} dx && (17)\\
&= -k\sigma^2 \hat{f}(k) && (18)
\end{aligned}$$

Therefore

$$\hat{f}(k) = Ce^{-\frac{k^2}{2\sigma^2}} \tag{19}$$

where the constant C is determined using $C = \hat{f}(0) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx$ so that $C = 1$

Using the convolution property and the fourier transform of a guassian, it can be seen that the application of a gaussian filter attenuates the frequency components of the original image by a gaussian. i.e. a low-pass filter

3 Heat Equation

Another option is to remove the noise term by applying a numerical discritisation of the *diffusion equation*. As we will see, this is equivalent to the gaussian filter, but has the advantage that we can easily modify the equation to, for example, recover edges in the image.

The diffusion, or heat equation in one-dimension is given by

$$\begin{cases} u_t(x,t) = Du_{xx}(x,t) \\ u(x,0) = u_0(x) \end{cases} \tag{20}$$

where D is the diffusion constant, $u_t(x,t) = \frac{\partial}{\partial t}u(x,t)$ and $u_{xx}(x,t) = \frac{\partial^2}{\partial^2 x}u(x,t)$. You can see that the speed of evolution u_t is proportional to the concavity u_{xx}. In other words, small features with high u_{xx} (i.e. noise) will be smoothed out first.

3.1 Fundamental Solution

We can consider the heat equation on an infinite domain $-\infty < x < \infty$ where the function u decays to zero towards infinity: $u(x,t) \to 0$ as $x \to \pm\infty$

Take the fourier transform of both sides

$$\mathcal{F}\{u_t(x,t)\} = \mathcal{F}\{Du_{xx}(x,t)\} \tag{21}$$

$$\frac{\partial}{\partial t}\int_{-\infty}^{\infty} f(x)e^{-ikx}dx = D(ik)\hat{u_x} \tag{22}$$

$$\int_{-\infty}^{\infty} \frac{\partial}{\partial t} f(x)e^{-ikx}dx = D(ik)^2\hat{u} \tag{23}$$

$$\hat{u}_t = -Dk^2\hat{u}, \tag{24}$$

$$\tag{25}$$

using the derivative property of the fourier transform and the fact that partial derivative of t does not interfere with the transform. Using this result, we can see that the solution for $\hat{u}(k,t)$ is

$$\hat{u}(k) = C(k)e^{-k^2Dt}, \tag{26}$$

where the constant C can be determined from the fourier transform of the initial conditions

$$\hat{u}(k,0) = \hat{u}_0(k) = C(k) \tag{27}$$

giving

$$\hat{u}(k,t) = \hat{u}_0(k)e^{-k^2Dt}, \tag{28}$$

This can be written as

$$\hat{u}(k,t) = \hat{u}_0(k)\mathcal{F}\left(\frac{e^{-x^2/4Dt}}{\sqrt{4\pi Dt}}\right) \tag{29}$$

Using the convolution property of the fourier transform, we can see that the inverse fourier transform of both sides gives

$$u(x,t) = \frac{1}{\sqrt{4\pi Dt}} \int_{-\infty}^{\infty} u_0(x') e^{-\frac{(x-x')^2}{4Dt}} dx' \tag{30}$$

where

$$G(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-\frac{x^2}{4Dt}} \tag{31}$$

is the Green function, or fundamental solution, to the heat equation. Note that this is the same as the gaussian low-pass filter with $\sigma = \sqrt{2Dt}$.

3.2 Finite Differences

3.2.1 Forward Time, Central Space

We will approximate the solution to the heat equation using finite differences, using a forward time, central space method.

First we calculate the time derivative using a forward difference

$$u_t(x_i, t_n) \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{32}$$

where $t_n = \Delta t n$ for the set of positive integers $n = 0, \ldots, N$ and $x_i = i - 0.5$ for the set of positive integers $i = 1, \ldots, L$. Note that this gives a simple grid spacing $\Delta x = 1$.

We approximate u_{xx} using a central difference

$$u_{xx}(x_i, t_n) \approx \frac{1}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) = u_{i+1}^n - 2u_i^n + u_{i-1}^n \tag{33}$$

Putting these into the heat equation gives

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = D(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{34}$$

$$u_i^{n+1} = u_i^n + \Delta t D(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{35}$$

$$u_i^{n+1} = \Delta t D u_{i+1}^n + (1 - 2\Delta t D) u_i^n + \Delta t D u_{i-1}^n \tag{36}$$

3.2.2 Boundary Conditions

We set our domain for x to be $0 < x < L$, and set boundary conditions at $x = 0$ and $x = L$.

A Dirichlet boundary condition gives the value of the function $u(x)$ at the boundary. For image processing, we might want to set a dirichlet boundary condition as the initial pixel values at the edges.

$$u(0,t) = u_0(0) \tag{37}$$

$$u(L,t) = u_0(L). \tag{38}$$

A Neumann boundary condition gives the derivative $\partial/\partial x$ of the function $u(x)$ at the boundary. A zero-Neumann b.c. is often used to ensure that the gradient of the image pixels at the boundary is zero.

$$\frac{\partial u}{\partial x}(0) = 0 \tag{39}$$

$$\frac{\partial u}{\partial x}(L) = 0 \tag{40}$$

This could be implemented, for example, by having a set of "mirror" pixels surrounding the image.

$$u_{0,j} = u_{1,j} \tag{41}$$
$$u_{L+1,j} = u_{L,j} \tag{42}$$

3.2.3 Matlab Code

We can write Matlab code to calculate this, using a zero-neumann boundary condition.

```
function u = isotropic_diffusion(u, dtD, N)

  % Get size of initial condition matrix
  nx = size(u);

  % Setup indexing. We don't update the boundary pixels, as they are set by
  % the boundary conditions
  j=2:nx-1;

  % Loop through timesteps
  for it=1:N
      % Use Forward Time, Centered Space
      u(i,j)= dtD*(u(i+1)+u(i-1)) + (1-2*dtD)*u(i);

      % Zero-neumann boundary conditions
      u(1) = u(2);
      u(nx) = u(nx-1);
  end
end
```

3.2.4 Stability

The error of the finite difference approximation is given by

$$E_i^n = \frac{u(x_i, t^{n+1}) - u(x_i, t^n)}{\Delta t} - D(u(x_{i+1}, t^n) - 2u(x_{i1}, t^n) + u(x_{i-1}, t^n) \tag{43}$$

Let $e_i^n = u_i^n - u(x_i, t_n)$, then subtracting Eq. 43 from Eq. 57 gives

$$E_i^n = \frac{e_i^{n+1} - e_i^n}{\Delta t} - D(e_{i+1}^n - 2e_i^n + e_{i-1}^n) \tag{44}$$

and

$$e_i^{n+1} = re_{i+1}^n + (1-2r)e_i^n + re_{i-1}^n - \Delta t E_i^n \tag{45}$$

let $e^n = max_i|e_i^n|$ and assume $1 - 2r \geq 0$

$$|e_i^{n+1}| \leq (1-2r)e^n + re^n + re^n + \Delta t|E_i^n| \tag{46}$$

or

$$|e_i^{n+1}| \leq e^n + \Delta t|E_i^n| \tag{47}$$

We assume that $e_0 = 0$, and with $E = \max_{i,n}|E_i^n|$

$$|e_i^{n+1}| \leq e^n + \Delta tE \geq e^{n-1} + 2\Delta tE \geq e^{n-2} + 3\Delta tE \geq \ldots \tag{48}$$
$$|e_i^{n+1}| \leq (n+1)\Delta tE = t_{n+1}E \tag{49}$$

We know that the approximation is first order in time, therefore $E \leq C\Delta t$ and

$$|e_i^{n+1}| \leq t_{n+1}\Delta tC \tag{50}$$

Take the limit as $n \to \infty$, $\Delta t \to 0$ and $t_n \to t$

$$|e_i^{n+1}| \leq tE \to 0 \tag{51}$$

Therefore the scheme converges for $1 - 2r \geq 0$

3.3 2D Heat Equation

The two-dimensional version of the heat equation is given as

$$\begin{cases} u_t(x,y,t) = D(u_{xx}(x,y,t) + u_{xx}(x,y,t)) \\ u(x,y,0) = u_0(x,y) \end{cases} \tag{52}$$

3.3.1 Finite Differences: Forward Time, Central Space

We will approximate the solution to the heat equation using finite differences, using a forward time, central space method.

The time derivative is unchanged from 1 dimension:

$$u_t(x_i, y_j, t_n) \approx \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} \tag{53}$$

where $t_n = \Delta t n$ for the set of positive integers $n = 0, \ldots, N$, $x_i = i - 0.5$ for the set of positive integers $i = 1, \ldots, L$ and $y_j = j - 0.5$ for the set of positive integers $j = 1, \ldots, L$.

We approximate u_{xx} and u_{yy} using a central difference

$$u_{xx}(x_i, y_j, t_n) \approx \frac{1}{\Delta x^2}(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) = u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n \tag{54}$$

$$u_{yy}(x_i, y_j, t_n) \approx \frac{1}{\Delta y^2}(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) = u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n \tag{55}$$

$$\tag{56}$$

Putting these into the heat equation gives

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = D(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n) \tag{57}$$

$$u_i^{n+1} = u_i^n + \Delta t D(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n) \tag{58}$$

$$u_i^{n+1} = \Delta t D(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n) + (1 - 4\Delta t D)u_{i,j}^n \tag{59}$$

Heat equation is linear, so the same arguments for accuracy and stability in 1D also apply for 2D.