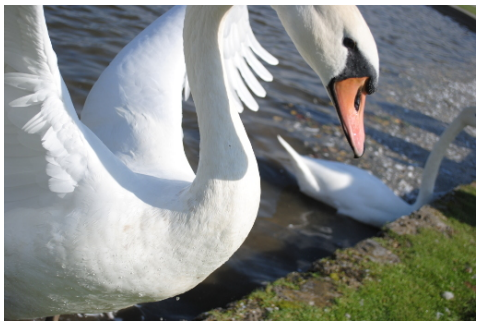
Image Processing Project for CBL Summer School

Perona–Malik Image denoising In the lectures, we looked at spatially varying diffusion, where the diffusion coefficient varied with x and y: now we go fully nonlinear with an application in image processing.

First try regular diffusion (the heat equation in 2D). You can use the `diffusion1.m` and `isotropic_diffusion.m` codes from the github repository. This is Gaussian blurring. It should get rid of the noise but it will also blur all the detail in the image and in particular the *edges*.



Photograph by Harry Biddle

The Perona–Malik equation is a modification of Gaussian diffusion that employs edge preservation by varying the diffusion coefficient across the image dependent on the image gradient ∇u, penalising it at edges (i.e. where ∇u is large). The general equation is

$$u_t = \nabla \cdot \big(g(|\nabla u|)\nabla u\big),$$

where $g(s)$ is an edge detection function with a tunable parameter λ that controls the sensitivity of the equation to visual edges. It gives a threshold to separate noise from edges. Two such functions proposed by Perona & Malik were

$$g(s) = \frac{1}{1+\frac{s^2}{\lambda^2}} \qquad \text{or} \qquad g(s) = \exp\left[-\frac{s^2}{\lambda^2}\right].$$

The two-dimensional Cartesian form of the Perona–Malik equation is

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left[g\left(\sqrt{u_x^2+u_y^2}\right)u_x\right] + \frac{\partial}{\partial y}\left[g\left(\sqrt{u_x^2+u_y^2}\right)u_y\right].$$

Following the lectures on spatially varying diffusion, this can be discretized with finite differences and forward Euler

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t\left[\frac{g_{i+\frac{1}{2},j}^n D_+^x u_{i,j}^n - g_{i-\frac{1}{2},j}^n D_-^x u_{i,j}^n}{\Delta x} + \frac{g_{i,j+\frac{1}{2}}^n D_+^y u_{i,j}^n - g_{i,j-\frac{1}{2}}^n D_-^y u_{i,j}^n}{\Delta x}\right]. \tag{1a}$$

Here D_+^α and D_-^α indicate forward and backward finite differences, in the direction indicated by the superscript α, on a grid spacing of Δx. The expressions involving g between grid points are calculated as the average of the values at the two neighbouring grid points, e.g.,

$$g_{i+\frac{1}{2},j}^n = \tfrac{1}{2}(g_{i+1,j}^n + g_{i,j}^n), \qquad g_{i-\frac{1}{2},j}^n = \tfrac{1}{2}(g_{i-1,j}^n + g_{i,j}^n), \tag{1b}$$

where the nodal $g_{i,j}^n$ are computed using central finite differences

$$g_{i,j}^n = g\left(\sqrt{(D_c^x u_{ij}^n)^2 + (D_c^y u_{ij}^n)^2}\right). \tag{1c}$$

Use the code from the github repository which loads the image and adds noise. Write a code to perform Perona–Malik denoising, using zero-Neumann boundary conditions for u. How does this set the boundary conditions for $g(|\nabla u|)$? Assuming you choose to make $\Delta x = 1$, then you might start with $\Delta t = 0.1$ and $\lambda = 5$. What happens to the diffusion near the edges? What happens to the image as you increase t? Try using both options for $g(s)$, are there any differences in the results?

Optional: try working with the full colour image. One approach is to work on each colour channel separately, although maybe R-G-B is not the best basis.