GV11 Geometry of Images. Coursework 2018

1 Introduction

This coursework asks you to implement linear and non-linear diffusion processes by solving partial differential equations (PDES) numerically. In this way you can construct both linear and non-linear *scalespaces*. It is recommended that you use Matlab for the implementation.

2 Numerical Implementation of Diffusion

2.1 1D linear diffusion

The 1D Linear diffusion equation is

$$\frac{\partial f}{\partial t} = \frac{\partial^2 f}{\partial x^2}$$

To implement this you need to

1. choose a number of point N at which to solve the problem
2. Implement the $N \times N$ matrix A representing the Laplacian
3. solve an iterative scheme

$$\begin{array}{lll} \text{Explicit scheme:} & f^{k+1} & = f^k + \Delta t \mathsf{A} f^k \\ \text{Implicit scheme:} & (I - \Delta t \mathsf{A})\, f^{k+1} & = f^k \end{array}$$

   where Δt is a chosen time step.

Choose a suitable array of values to initialise the first step f^0 (for example, take one line from a typical test image). By storing each iteration you can display the function $f(k\Delta t)$ for any integer k which should represent the diffusion of f to time $t = k\Delta t$.

Use your implementation to carry out the following tests

- Test both the explicit and implicit scheme and find the largest Δt so that the explicit scheme is stable.
- Test the theory that says that f^k is also given by the convolution of f^0 with a Gaussian filter of standard deviation $\sigma = \sqrt{2k\Delta t}$.

2.2 2D linear diffusion

The 2D Linear diffusion equation is

$$\frac{\partial f}{\partial t} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

You should implement this to produce a diffusion scheme for a $N \times N$ image. This involves reshaping f into a $N^2 \times 1$ vector and implementing the $N^2 \times N^2$ matrix A. This matrix is much

too large to be held in a dense "full" form. You must use the sparse representation `sparse` for A. The entries of A should be -4 on the diagonal, 1 on the neighbours to the diagonal, and 1 on a set of entries at a spacing of N from the main diagonal.

$$\ldots 1 \underbrace{\ldots}_{N} 1 \; -4 \; 1 \underbrace{\ldots}_{N} 1 \ldots$$

This is called 'tridiagonal with fringes'. You may find the Matlab function `spdiags` helpful.

It's easy to solve the diffusion equation using Explicit scheme, but the Implicit scheme needs the inversion of a huge matrix, which, even if sparse, is computationally challenging. The *semi-implicit* scheme, involves splitting A into A^{x} with entries

$$\ldots \; 1 \; -2 \; 1 \; \ldots$$

and A^{y} with entries

$$\ldots 1 \underbrace{\ldots}_{N} -2 \underbrace{\ldots}_{N} 1 \ldots$$

Then the *semi-implicit* scheme is

$$\begin{aligned} \left(I - \frac{\Delta t}{2}\mathsf{A}^{\mathsf{x}}\right) f^{k+1/2} &= \left(I + \frac{\Delta t}{2}\mathsf{A}^{\mathsf{y}}\right) f^{k} \\ \left(I - \frac{\Delta t}{2}\mathsf{A}^{\mathsf{y}}\right) f^{k+1} &= \left(I + \frac{\Delta t}{2}\mathsf{A}^{\mathsf{x}}\right) f^{k+1/2} \end{aligned}$$

In between these two steps, if $f^{k+1/2}$ is reshaped so that the pixels are accessed in column order, then both equations can be solved with tridiagonal solvers. This means that A^{y} on the second step is *identical* to A^{x} on the first step. A simple direct solver is perfectly efficient:

```
IAx = speye(N*N) - deltaT/2 * spdiags(dd,-1:1,N*N,N*N);
f = IAx\b;
```

where `dd` is a $N^2 \times 3$ matrix with each row being $[1 \; -2 \; 1]$ and `b` represents the righthand side $\left(I + \frac{\Delta t}{2}\mathsf{A}^{\mathsf{y}}\right) f^{k}$.

Use your implementation to carry out the following tests

- Calculate a series of images $f^0 \ldots f^n$. These should appear as successive blurring over time.
- Choose a local maximum in f^0 and track its position over time in the succeeding images. If the time step Δt is not large, the spatial position of the maximum should change only slowly over time. If the particular maximum is annihalated, then choose another and track the position of that.

2.3 2D Anisotropic diffusion

The 2D anisotropic diffusion equation is

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial x}\gamma\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}\gamma\frac{\partial f}{\partial y}$$

where γ is a diffusivity function. Implement this scheme using the Perona-Malik function

$$\gamma = \frac{1}{1 + \left(\frac{|\nabla f|}{T}\right)^2} \tag{1}$$

where T is a threshold such that γ becomes small for large edges and almost 1 for small edges.

The easiest way to implement this is to create explicit 1^{st}-order derivative operators `Dx` and `Dy` and then create

```
gam =  spdiags(reshape(gamma,[],1),0:0,N*N,N*N)
PM = - (Dx' *gam *Dx +  Dy'* gam*Dy);
```

where `gamma` is the image calculated in eq.(1).

If you examine the matrix `PM` you will find that it has the same structure as the Laplacian you used in part 2.2 but with different values, e.g. at rows $j-1 \to j+1$ you will see something like

$$\begin{array}{ccccccccccccccccccc}
\cdots & a_{j-1} & 0 & 0 & \cdots & b_{j-1} & -(a_{j-1}+b_{j-1}+c_{j-1}+d_{j-1}) & c_{j-1} & 0 & 0 & \cdots & d_{j-1} & 0 & 0 & \cdots \\
\cdots & 0 & a_j & 0 & \cdots & 0 & b_j & -(a_j+b_j+c_j+d_j) & c_j & 0 & \cdots & 0 & d_j & 0 & \cdots \\
\cdots & 0 & 0 & a_{j+1} & \cdots & 0 & 0 & b_{j+1} & -(a_{j+1}+b_{j+1}+b_j+c_{j+1}+d_{j+1}) & c_{j+1} & \cdots & 0 & 0 & d_{j+1} & \cdots
\end{array}$$

The values of each $\{a_j, b_j, c_j, d_j\}$ are between 0 and 1 for all j, and they represent the "diffusivities" between the j^{th} pixel and its neighbours to the left, right, above and below. Notice that each row adds up to zero.

Repeat the tests you performed for the question in section 2.2. Comment on the differences.

3 Report

You should write a short report, probably no more than 6-8 pages, in which you describe the steps you took and the design choices you made. Indicate any problems encountered and how you solved them. Compare and contrast the different solutions you obtained. Code, if you consider it relevant, can be attached as an appendix (i.e. not included in the 6-8 page report).