Anisotropic diffusion for denoising and edge detection

Simon LEBASTARD

January, 7th, 2016

1 General principles

1.1 Space-scales

A picture can have a range of different important scales. It can bear information at a low spatial frequency, while details will be held in the high frequencies of the picture.
When receiving a damaged picture, we might not be able to know what the most important "spectral scale" for the image is. Below is an exemple of a space-scale of a 1D signal (source: Perona & Malik)
From an image we received, we basically want to produce a wide range of scale space, so that if the image was damaged during transmission, information from different frequencies can be recovered.

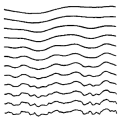

Figure 1: Scale space of a 1D signal. Source: Perona & Malik

1.2 Isotropic diffusion

Convoluting with Gaussian kernels of increasing width is comparable to solving the diffusion problem, which general equation is:

$$I_t = div(c(x,y,t)\nabla I) \tag{1}$$

In isotropic situation $c(x,y,t)$ is only a function of time, and in the case of a time-constant environment c **is a constant**. The solution to this equation in the caseis the following:

$$I(x,y,t) = \frac{1}{\sqrt{4\pi ct}} \int I(x',y',0) \exp^{-\frac{(x-x')^2+(y-y')^2}{4ct}} dx'dy' \tag{2}$$

The solution can be interpreted as the convolution of the original image with the gaussian kernel:

$$G(x,y,t) = \frac{1}{\sqrt{4\pi ct}} \exp^{-\frac{x^2+y^2}{4ct}} \tag{3}$$

the deviation of which increases linearly with time, and is a linear function of c. This result suits the intuition that the gaussian kernel expands wih time, leading to high frequences disappearing in the image.

To illustrate the different kinds of solutions I'll use the following picture of Venise. First here the result of isotropic diffusion for different kernel width. Those results were computed in the exact same manner

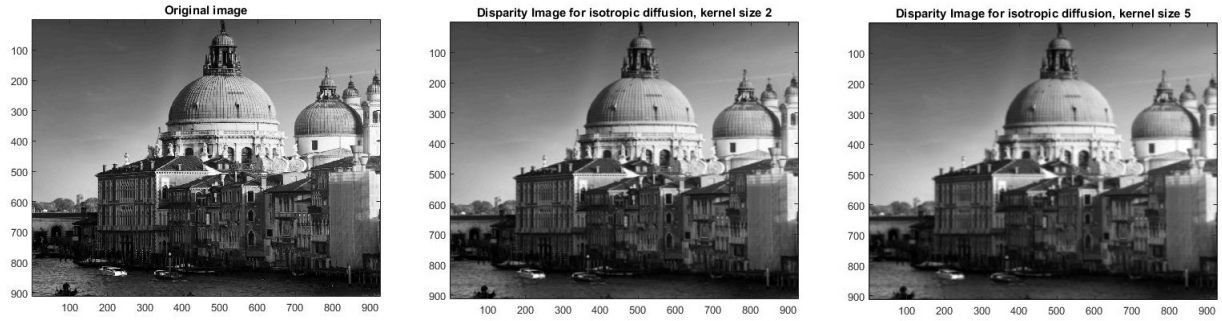

Figure 2: The result of isotropic diffusion for a constant c = 1, for various kernel width

that was tought during class. You can download the Matlab code as well as the image ressources from: `https://github.com/slebastard/TIVA_anisotropic_diffusion`

Gaussian convolution can generate a scale space that have some fine properties. However it also has some pretty obvious drawbacks, some of which are:

- Noise disappears but so do edges. The high peaks of the gradients tend to smoothen.
- Edges are naturally shifted away from their true location when the image is convoluted with an increasingly flat Gaussian kernel. That means that even if we recognize an edge in a coarse scale image, we won't be able to figure out the true position of that edge so easily.

Note that the Laplacian of Gaussian (LoG) is often used for the isotropic diffusion methods. It simply is the difference between two or more gaussian kernels of different width, and presents two advantages: it is linear, and it seems to be very close from the way mammals see.

2 Anisotropic diffusion and the Perona & Malik study

In the general case the diffusion factor $c(x, y, t)$ is a function of space and time. An ideal case would be if we could make it to set $c = 0$ on the edges of the objects and $c \neq 0$ otherwise. There would then be a blurring inside each regions, without the regions being mixed up.

For some cases of diffusion factors, taken as functions of $||\nabla I||$, the edges of the image will be preserved, even enhanced.

Perona & Malik worked on a specific range of solutions, generated from two classes of diffusion factors:

$$c_\kappa(||\nabla I||) = \exp((-\frac{||\nabla I||}{\kappa})^2) \tag{4}$$

and

$$c_{;kappa}(||\nabla I||) = \frac{1}{1 + (\frac{||\nabla I||}{\kappa})^2} \tag{5}$$

In both cases we can see that the spots with a high gradient will generate a low diffusion factor, which means that diffusion will not occur much around the edges. Those two diffusion functions are the result of compromises between blurring the spots will little intensity variations while preserving the edges.

Note that the κ parameter plays an important role: when high, only very important gradients will induce a low diffusion factor. That means that the asymptotic situation $\kappa \to \infty$ is equivalent to

isotropic diffusion, while a very low value for κ will mean that only spots with a very high gradient (that is edges) will be spared from diffusion.

From this anisotropic diffusion we can generate anisotropic scale space that have good properties.

As can be seen in the results (page 4), the low-resolution image that was generated through anisotropic diffusion make it to preserve the most important edges for a sufficiently low κ value. The major gradients in the image are preserved, they even appear enhanced because of the variation of local contrasts around an edge.

3 Applications

3.1 Anisotropic filters for denoising and edge detection

Anisotropic diffusion is an effective process for denoising (through blurring) while preserving the edges (which cannot be done with a mere gaussian kernel convolution).

Is was shown before that building a space scale with an isotropic diffusion would lead to blurring the edges, which is a problem as what we seek to do is to represent all the objects of different scales in the image without alterring them, which implies that we must be able to detect edges of all objects in the image from the scale space. Edges detection can be made easy by running the diffusion process backwards. But the problem is ill-posed and mostly leads to unstable solution.

Anisotropic diffusion with just the right diffusion factor, as a function of $||\nabla I||$ can lead to enhancing the edges in a similar way reversed diffusion would, while going forward in time (thus preventing us from any unstability issue).

In their paper, Perona & Malik compared the result of anisotropic diffusion with the canny detector in terms of edges enhancement. $||\nabla I||$ is compared for the two methods.

3.2 Thermal conduction in 'anisotropic' environment

There are several exemples of situations of anisotropic (in the sense that was defined earlier, which in physics is equivalent to the term 'inhomogeneous') diffusion:

- Diffusion in a porous environement, and more generally in inhomogeneous environement, where the diffusion constant $c(x, y, t$ will depend on the layer, material or physical state that is at position (x, y) at time t. In fact we have
$$c = \frac{k}{\rho C_V} \tag{6}$$
where ρ is the material density, k is its thermal conductivity, C_V its specific heat. Upon hypothesis a resolution scheme similar to the one explained below, for the right diffusion function, could be used

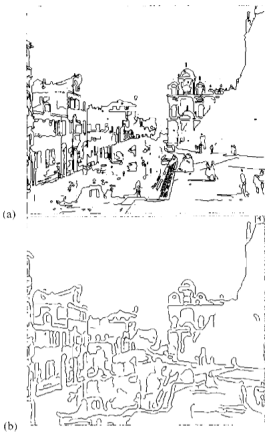

Fig. 10. Edges detected using (a) anisotropic diffusion and (b) Gaussian smoothing (Canny detector).

Figure 3: The results of anisotropic diffusion (a) and canny detection (b) on the intensity gradient map ($||\nabla I||$). Source: Perona & Malik

4 Anisotropic filters: how to

4.1 Computing the filter

To perform an anisotropic filter, the modifications in the image from one step to the next are controlled by discretizing the general diffusion equation into:

$$I_{i,j}^{t+1} = I_{i,j}^{t} + \lambda * (c_N \nabla_N I + c_S \nabla_S I + c_W \nabla_W I + c_E \nabla_E I) \tag{7}$$

according to the method from Perona & Malik, where $\nabla_N I$ will here mean the finite-difference:

$$\nabla_N I_{i,j} = I_{i-1,j} - I_{i,j} \tag{8}$$

, and c_N is the diffusion factor for the northern direction from this pixel:

$$c_{N,i,j}^{t} = g((||\nabla I||)_{i+\frac{1}{2},j}^{t}) \tag{9}$$

Here g would be one of the two functions that Perona & Malik used in for their works.
Then we would only have to run a script that, from the original image, computes the blurred image. This is what it looks like for a single time loop.

```
[w,h] = size(img);
actualized_img = zeros(size(img));

for ab = 1:w
    for or = 1:h
        dif_coef = compute_coef(img, ab, or, kappa);
        fin_diff = compute_difference(img, ab, or);
        actualized_img(ab,or) = img(ab,or) + lambda*dot(dif_coef, fin_diff);
    end
end
```

Here the compute_coef function returns the value of the diffusion factors around (ab, or), using the quadratic or exponential function with the parameter κ. The compute_difference function evaluates the finite-differences of intensity around (ab, or).

4.2 Results

Here are the results from my program, computed for different κ values at a given $\lambda = 0.25$. Results also show the influence of diffusion time at a given κ.

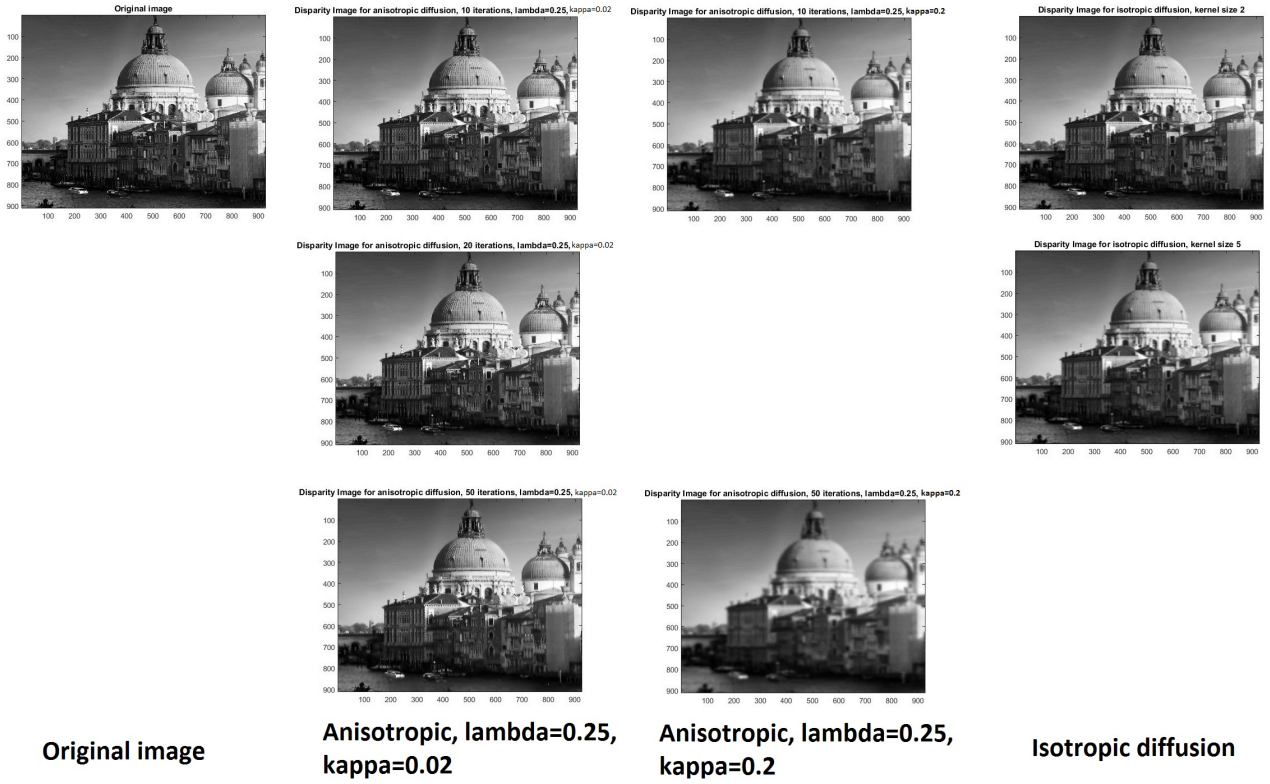

Figure 4: Results from the Matlab script for anisotropic diffusion, for different κ values and number of iterations

Once again you can checkout the code as well as the solutions and ressources used at: `https://github.com/slebastard/TIVA_anisotropic_diffusion`

5 Bibliography

- `http://image.diku.dk/imagecanon/material/PeronaMalik1990.pdf`
  Title: Scale-space and edge detection using anisotropic diffusion
  Authors: Perona & Malik
  July 1990
- `http://www.cs.utah.edu/~manasi/coursework/cs7960/p2/project2.html`
  Project title: Anisotropic diffusion
  Author: M. Datar
- `http://www.mia.uni-saarland.de/weickert/Papers/book.pdf`
  Title: Anisotropic diffusion in image processing
  Author: J. Weickert At: B.G. Teubner Stuttgart
- `http://www.sciencedirect.com/science/article/pii/0011227579900547`
  Title: Model calculations on heat flow in inhomogeneous thermal systems
  Authors: A.W Pattulo & J.C.A van der Sluijs