Applications of Numerical PDEs in Image Processing and Computer Vision

Lindsey Grigsby

University of Florida

Spring 2020

Introduction

Image processing

- the transformation of an input image into an output image
- e.g. compression, rotation, blurring, edge detection

Computer vision

- uses image processing algorithms to complete tasks related to understanding an image
- e.g. edge detection, motion tracking, object recognition

several good libraries, e.g. OpenCV (C++/Python)

Mathematical operations on images

- Can consider an image:
  - a *length*×*width* matrix of pixels
  - or, an intensity function $I(x, y)$ describing the intensity at each pixel (x, y)
- Operations
  - easy to imagine how to convert to grayscale, rotate, etc
  - easy to imagine what multiplication by a constant, taking the transpose, etc, will do
- more interesting tasks can also be intuitive
- e.g. edge detection

A simple example involving partial derivatives

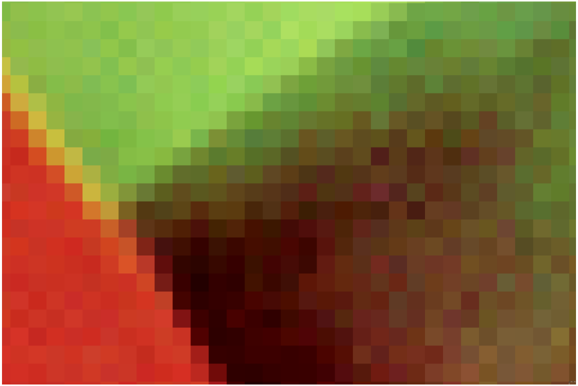

Figure 1: What is an edge? Try: a curve such that along each line perpendicular to it, pixel values change very quickly. (Then there are 2 edges here.)

A simple example involving partial derivatives

Figure 2: Zooming out, the edges detected via this method make sense.

A simple example involving partial derivatives

Formally:

- consider the image in terms of its intensity function $I(x, y)$
- say (x, y) is part of an edge if $||\nabla I(x, y)||$ is a local max (recall that $||\nabla I(x, y)|| \iff$ rate of increase in the direction of fastest increase)
- this $+$ some simple refining steps is Canny edge detection

Figure 3: The result of Canny edge detection

Examples of PDEs in image processing and computer vision

Image segmentation

- separates an image into regions based on some criteria (e.g. distinct objects)
- many methods, including several efficient PDE methods

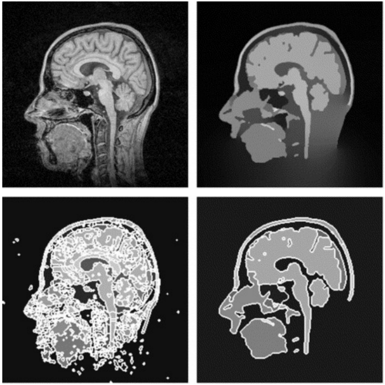

Figure 4: "Efficient image segmentation using partial differential equations and morphology", Weickert

Inpainting

- removes large imperfections from images
- two main PDE-related algorithms - one based on Navier-Stokes, and another based on the "fast marching method"

Navier-Stokes	Image inpainting
stream function Ψ	Image intensity I
fluid velocity $v = \nabla^\perp \Psi$	isophote direction $\nabla^\perp I$
vorticity $\omega = \Delta \Psi$	smoothness $w = \Delta I$
fluid viscosity ν	anisotropic diffusion ν

Figure 5: "Navier-Stokes, fluid dynamics, and image and video inpainting", Bertalmio.

Inpainting

Figure 6: An example from the same paper

Anisotropic diffusion

- removes noise from an image without blurring details necessary for interpreting the image (like edges)
- related to the heat equation

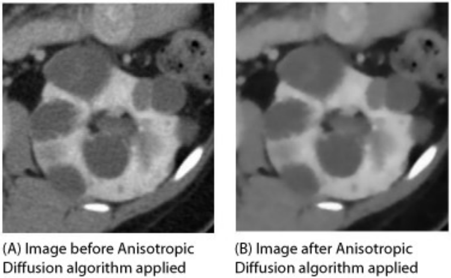

Figure 7: Noise is reduced without blurring edges

Anisotropic Diffusion

Motivation

- one way to reduce noise is via Gaussian diffusion (blurring)
- but, this blurs everything evenly

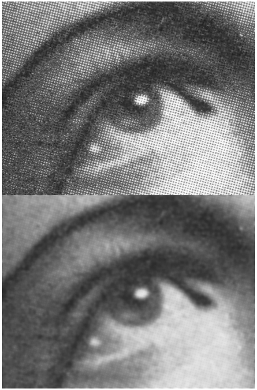

Figure 8: Original image smoothed using Gaussian diffusion

Motivation

The fix: consider what Gaussian diffusion is in terms of PDEs.

- clear analogy with heat transfer:

$$\text{heat equation} \iff \text{gaussian diffusion equation}$$
$$\frac{\delta u}{\delta t} = \alpha \Delta u \iff \frac{\delta I}{\delta t} = c \Delta I$$

  where

$$u(x, y, t) \text{ (temp at time t)} \iff I(x, y, t) \text{ (pixel value at time t)}$$

  and

$$\alpha \text{ (diffusivity of the medium)} \iff c \text{ (diffusion coefficient)}$$

Motivation

In the Gaussian diffusion equation

$$\frac{\delta I}{\delta t} = c\Delta I = div\left(c\nabla I\right)$$

(note: $\Delta =$ Laplacian and $\nabla =$ gradient),

- c controls how much the image blurs
- so if we want to prevent blurring in particular places (like edges), we could try making c a function

The anisotropic diffusion equation

Call this modified equation,

$$\begin{aligned}\frac{\delta I}{\delta t} &= div\,(c(x,y,t)\nabla I) \\ &= \nabla c \cdot \nabla I + c(x,y,t)\Delta I,\end{aligned}$$

the anisotropic diffusion equation. We want it to uniquely define a family of images (parametrized by t) which can be easily computed.

- do there even exist (non-constant) c so that this problem is well-posed and numerically stable?
- if so, will one of these c's improve on Gaussian noise reduction?

Specializing the problem

Perona and Malik investigated anisotropic diffusion in 1987.

They formalized what they wanted out of the family of images (i.e., what "blur while preserving edges" means)

Example of a tree against sky

- instead of "edges", talk about region boundaries at different resolutions
- want a family of images such that
  - First, boundaries between details on individual leaves are blurred. Then boundaries between leaves are blurred. Then sky and tree are blurred together.

Choosing the diffusion coefficient $c(x, y, t)$

If somehow we knew exactly where the region boundaries were at time any time t, then we could choose

$$c(x,y,t) = \begin{cases} 1 & (x,y,t) \in \{\text{interior of some region}\} \\ 0 & (x,y,t) \in \{\text{boundary of some region}\}. \end{cases}$$

- Then at each time t, the interiors of regions would blur and the boundaries would not - exactly what we want.
- But, we don't generally have this region info.

Choosing the diffusion coefficient $c(x, y, t)$

If somehow we could estimate where the region boundaries were at any time t, then we could choose $c(x, y, t)$ to be a decreasing function (with values between 0 and 1) of how likely it is that (x, y, t) is part of a boundary.

- Then at each time t, the probably-interiors of regions would blur significantly and the probably-boundaries would blur less.
- Can we get these boundary estimations?

Choosing the diffusion coefficient $c(x, y, t)$

We already have one way to estimate boundaries, using the fact that $||\nabla|| =$ rate of increase in the direction of fastest increase.

So we can choose

$$c(x, y, t) = g(||\nabla I(x, y, t)||)$$

where g is some monotone decreasing, non-negative function with $g(0) = 1$.

- Note that any such g should have the property of blurring interiors and preserving boundaries.

Choosing the diffusion coefficient $c(x, y, t)$

Perona and Malik suggest using either

$$g\left(||\nabla I||\right) = e^{-\left(\frac{||\nabla I||}{K}\right)^2}$$

or

$$g\left(||\nabla I||\right) = \frac{1}{1 + \left(\frac{||\nabla I||}{K}\right)^2}$$

and K is a constant.

- These choices are numerically convenient.

To summarize:

We are considering the anisotropic diffusion equation

$$\frac{\delta I}{\delta t} = \nabla c \cdot \nabla I + c(x, y, t)\Delta I$$

with $c(x, y, t) = g\left(||\nabla I(x, y, t)||\right)$, where g is some monotone decreasing, non-negative function with $g(0) = 1$.

- *If/when* this problem is well-posed and numerically stable, then the family of images it defines should have the boundary-preserving properties we seek.

Experiments: The effect of $g\left(||\nabla I||\right)$

Figure 9: Images generated using the two proposed functions g and $k = 0.05$, at $t = 128$. (Second is exponential.)

- $e^{-\left(\frac{||\nabla I||}{K}\right)^2}$ "privileges high contrast edges over low contrast ones"
- $\frac{1}{1+\left(\frac{||\nabla I||}{K}\right)^2}$ "privileges wide regions over smaller ones"

Experiments: The effect of the constant K

- K controls the width of g
- if an image is very noisy, need g wide $\rightarrow K$ large; if it has little noise, need g narrow $\rightarrow K$ small.

We can choose K experimentally or using Canny noise estimation

Experiments: The effect of the constant K

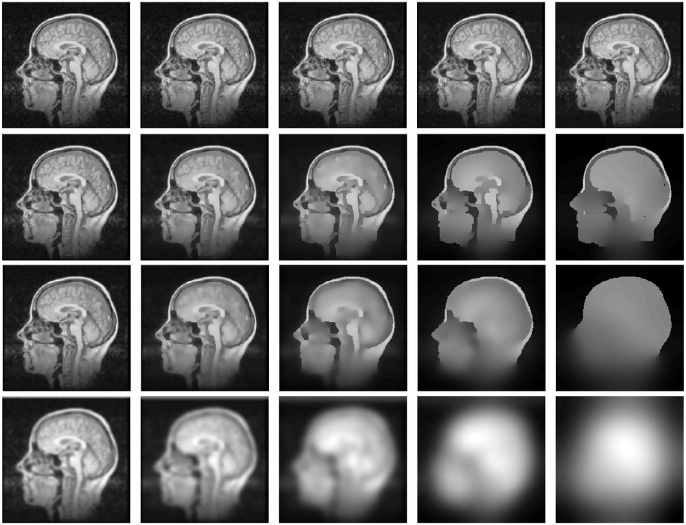

Figure 10: Top row using $k = 1$, then $k = 7.66, 14.32, 100000$

Properties

Properties of the PDE problem

- can prove that for either suggested g, the problem is well-posed and numerically stable if K is chosen correctly
- other g might also have good properties

Properties of the resulting family of images

- can prove that edges are actually enhanced
- can prove that no new features are created

Applications

- on its own, it makes images more clear
- in other algorithms, e.g., image segmentation, inpainting
- importantly, it can improve edge detection
  - makes edges detected with gradients more meaningful

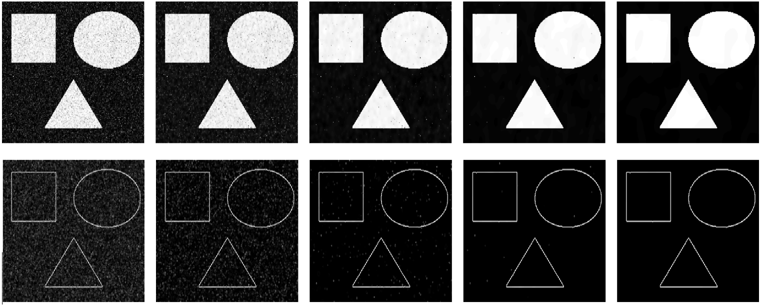

Figure 11: Top row: anisotropic diffusion; bottom row: edges recovered with gradient method

Example implementation

OpenCV's anisotropicDiffusion()

Parameters:

- **src** Source image with 3 channels.
- **dst** Destination image of the same size and the same number of channels as src .
- **alpha** The amount of time to step forward by on each iteration (normally, it's between 0 and 1).
- **K** sensitivity to the edges
- **ninters** The number of iterations

References

1. Bertalmio, M. & Bertozzi, Andrea & Sapiro, G.. (2001). Navier-Stokes, fluid dynamics, and image and video inpainting. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 1. I-355. 10.1109/CVPR.2001.990497.
2. Bin Zhou, Xiao-Lin Yang, Rui Liu and Wei Wei, 2010. Image Segmentation with Partial Differential Equations. Information Technology Journal, 9: 1049-1052.
3. Pietro Perona and Jitendra Malik (July 1990). "Scale-space and edge detection using anisotropic diffusion". IEEE Transactions on Pattern Analysis and Machine Intelligence. 12 (7): 629–639.
4. J Weickert, 2001. Efficient image segmentation using partial differential equations and morphology. Pattern Recognition Volume 34, Issue 9, September 2001, Pages 1813-1824
5. Weickert, J.: Anisotropic diffusion in image processing. B.G. Teubner, Stuttgart (1998).

References

Images only from

1. Figures 1-3: https://github.com/polyfloyd/edge-detection-rs
2. Figure 6: https://mipav.cit.nih.gov/pubwiki/index.php/Filters_(Spatial)_Anisotropic_Diffusion
3. Figure 7: http://www.codebind.com/python/opencv-python-tutorial-beginners-smoothing-images-blurring-images-opencv/
4. Figure 8: http://www.cs.utah.edu/ man-asi/coursework/cs7960/p2/project2.html
5. Figures 9-10: https://www.cs.utah.edu/ jfishbau/advimproc/project2/