Assignment 3 — Non-Linear Anisotropic Diffusion

Image Processing and Pattern Recognition

Deadline: January 17, 2024

December 18, 2023

1. Goal

In this exercise, we will implement and discuss different non-linear anisotropic diffusion models. These models derivate from the celebrated Perona-Malik nonlinear diffusion [1]. In their paper, Perona and Malik proposed a diffusivity function which guides the diffusion process by reducing diffusion across edges. Their model showed good denoising performance and some of today's state-of-the-art image denoising algorithms build on this idea.

Specifically, we consider coherence enhancing diffusion (CED) and edge enhancing diffusion (EED). We discuss these methods in more detail in the following sections and show examples of CED and EED in Figure 1 and Figure 2 respectively.

2. Methods

Let $u\colon \Omega \to \mathbb{R}$ be the image function over the domain $\Omega \subseteq \mathbb{R}^2$. The basic diffusion equation (also known as "heat equation") is

$$\frac{\partial u}{\partial t} = \operatorname{div} \nabla u. \tag{1}$$

Here, $\nabla u = (\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y})^\top$ denotes the vector of spatial derivatives and div is the divergence of a vector field. This partial differential equation is motivated by the diffusion of, e.g., heat in some homogeneous material. To "steer" the diffusion, we introduce the *diffusion tensor* $D \in \mathbb{R}^{2\times 2}$ and modify (1) to

$$\frac{\partial u}{\partial t} = \operatorname{div} D\nabla u. \tag{2}$$

In our context, we can use the diffusion tensor to guide the diffusion process along image edges.

Figure 1: CED using the parameters detailed in Table 1: The input image is on the left, the output of CED on the right.

Figure 2: EED using the parameters detailed in Table 1: The input image is on the left, the output of EED on the right.

We can recover the isotropic diffusion equation (1) by setting $D = I$. On the other hand, if D is densely populated, the diffusion is *anisotropic*. CED and EED differ in the construction of the diffusion tensor D. For both, it is derived from the *structure tensor* that we define in the next section.

2.1. Structure Tensor

For a given image location $(i, j) \in \Omega$, the structure tensor $S_{(i,j)}(u) \in \mathbb{R}^{2\times 2}$ captures the local structure around $u(i, j)$. Specifically, we define the structure tensor as

$$S_{(i,j)}(u) = \left(G_{\sigma_g} * \begin{bmatrix} (\tilde{u}_x)^2 & \tilde{u}_x \tilde{u}_y \\ \tilde{u}_x \tilde{u}_y & (\tilde{u}_y)^2 \end{bmatrix} \right)(i, j) \tag{3}$$

where v_x is a shorthand for $\frac{\partial v}{\partial x}$, G_σ is a Gaussian kernel with variance σ^2 and $\tilde{u} = G_{\sigma_u} * u$. The convolution with G_{σ_g} as well as the indexing is understood element-wise. In Appendix A, (3) is described using more verbose notation.

The following discussion is valid for any pixel index $(i, j) \in \Omega$. Thus, we drop the index subscript as well the argument u for conciseness.

Let $\mu_1, \mu_2 \in \mathbb{R}$ and $v_1, v_2 \in \mathbb{R}^2$ be the Eigenvalues and Eigenvectors of S respectively. We use the convention that the Eigenvalues are sorted in descending order ($\mu_1 \geq \mu_2$) and the Eigenvectors are of unit length ($\|v_1\| = \|v_2\| = 1$). They encode information about the local structure of the image, namely the direction and slope of grayvalue variation. More specifically, we can distinguish between three cases:

1. $\mu_1, \mu_2 \ll$: Both Eigenvectors are small. This corresponds to a flat image region.
2. $\mu_1 \gg \mu_2$: Large gradient in direction v_1, but little change in v_2. In this case, there is an edge in the image.
3. $\mu_1, \mu_2 \gg$: Strong edges is both directions, i.e. and image corner.

In summary, the Eigenvalues determine the magnitude of grayvalue variation in the direction of the Eigenvector. If $\mu_1 \gg \mu_2$, v_1 is orthogonal to the edge (i.e. points in the direction of largest grayvalue change), whereas v_2 points in the coherence direction, i.e. along the edge. We will use this information to guide the diffusion process.

2.2. Diffusion Tensor

Given the Eigendecomposition of S as described above, we calculate the diffusion tensor via

$$D = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} v_1^\top \\ v_2^\top \end{bmatrix} = \begin{bmatrix} D^1 & D^3 \\ D^3 & D^2 \end{bmatrix}. \tag{4}$$

The choice of $\lambda_1, \lambda_2 \in \mathbb{R}^+$ depends on the application. Typically, they are a function of the eigenvalues μ_1, μ_2 and in the next section we describe the choice for CED and EED.

2.3. Coherence Enhancing Diffusion

The idea of CED is to guide the diffusion process along coherent regions, i.e. regions of similar structure. Thus, we choose

$$\begin{cases} \lambda_1 = \alpha, \\ \lambda_2 = \alpha + (1-\alpha)(1 - g(\mu_1 - \mu_2)), \end{cases} \tag{5}$$

with a small constant $\alpha > 0$ and the edge stopping function

$$g(x) = \exp\left(-\frac{x^2}{2\gamma^2}\right) \tag{6}$$

parametrized by the width $\gamma > 0$.

2.4. Edge Enhancing Diffusion

To facilitate diffusion in relatively homogeneous regions and inhibit diffusion across edges, the diffusivity λ_1 should decrease as μ_1 increases. Consequently, we choose

$$\begin{cases} \lambda_1 = (1 + \frac{\mu_1}{\delta^2})^{-\frac{1}{2}}, \\ \lambda_2 = 1. \end{cases} \tag{7}$$

where $\delta > 0$ controls the influence of μ_1. In edge enhancing diffusion, we always set $\sigma_g = 0$.[1]

2.5. Numerical Implementation

We solve (2) using a semi-implicit approach. In the discrete setting, an image of size $M \times N$ is represented as a vector $U \in \mathbb{R}^{MN}$ flattened in row-major order. We approximate the time-derivative $\frac{\partial u}{\partial t}$ and the spatial gradient ∇u by first-order finite differences. With a time discretization step $\tau \in \mathbb{R}^+$, (2) is discretized as

$$\frac{U^{t+\tau} - U^t}{\tau} = -\boldsymbol{\nabla}^\top \boldsymbol{D}(U^t) \boldsymbol{\nabla} U^{t+\tau}. \tag{8}$$

The spatial finite differences operator $\boldsymbol{\nabla} = \begin{pmatrix} \boldsymbol{\nabla}_x \\ \boldsymbol{\nabla}_y \end{pmatrix} \in \mathbb{R}^{2MN \times MN}$ is constructed by stacking the finite differences operators in the two spatial dimensions $\boldsymbol{\nabla}_x, \boldsymbol{\nabla}_y \in \mathbb{R}^{MN \times MN}$. In addition, we used the fact that the discrete divergence is $-\boldsymbol{\nabla}^\top$. Accordingly, $\boldsymbol{D}(U^t) \in \mathbb{R}^{2MN \times 2MN}$ is constructed as

$$\boldsymbol{D}(U^t) = \begin{bmatrix} \operatorname{diag} \boldsymbol{D}^1(U^t) & \operatorname{diag} \boldsymbol{D}^3(U^t) \\ \operatorname{diag} \boldsymbol{D}^3(U^t) & \operatorname{diag} \boldsymbol{D}^2(U^t) \end{bmatrix}, \tag{9}$$

[1] In other words, the convolution with G_{σ_g} in (3) is just the identity map.

where $\boldsymbol{D}^m(U^t) = (D^m_{1,1}, \dots, D^m_{M,N}) \in \mathbb{R}^{MN}$, $m \in \{1,2,3\}$ is a vector holding D^m for every pixel in U^t and diag: $\mathbb{R}^{MN} \to \mathbb{R}^{MN\times MN}$ constructs a diagonal matrix from its argument. Solving (8) for $U^{t+\tau}$ yields

$$U^{t+\tau} = \left(I + \tau \boldsymbol{\nabla}^\top \boldsymbol{D}(U^t)\boldsymbol{\nabla}\right)^{-1} U^t. \tag{10}$$

This is repeated until we reach some end time $T > 0$.

3. Tasks

3.1. Implementation

Implement CED and EED as described in the previous sections. Keep in mind that they only differ in the definition of λ_1, λ_2. We provide a function to compute $\boldsymbol{\nabla}$ (`spnabla_hp`), as well as reference images both diffusion processes. The parameters for the reference images are detailed in Table 1. You can specify the mode as a command line argument to the script, i.e. `python anisotropic_diffusion.py [mode]`.

3.2. Discussion

Parameters and Experiments Try the diffusion schemes on your own images. Experiment with different parameters, describe their influence and plot your findings. In particular,

- describe how α and γ influence the output of the CED and,
- show the influence of σ_u and δ on the output of EED.

In which cases (limits) do the schemes “degenerate” to linear homogeneous isotropic diffusion? What is the explicit “solution” to the linear homogeneous isotropic diffusion equation?

Structure Tensor and Eigenvalue Decomposition In EED, $\sigma_g = 0$, and therefore the structure tensor (4) and its Eigendecomposition become less abstract. Calculate the Eigendecomposition of the structure tensor explicitly in terms of $\nabla\tilde{u}$, i.e. derive expressions for μ_1, μ_2, v_1 and v_2 in terms of $\nabla\tilde{u}$. Interpret these expressions in our context. Do they make sense for EED? You can check your calculations by implementing them and comparing them to the results from some library.

Numerics Which convolution kernel does `spnabla_hp` implement in each spatial direction? In other words, for which convolution kernel K does $\mathrm{flat}(K * \mathrm{unflat}(U)) = \boldsymbol{\nabla}_x U$ (and similarly $\boldsymbol{\nabla}_y$) hold? Here, $\mathrm{flat} : \mathbb{R}^{M\times N} \to \mathbb{R}^{MN}$ vectorizes an image in row-major order, and $\mathrm{unflat} : \mathbb{R}^{MN} \to \mathbb{R}^{M\times N}$ reverses this operation. What other choices do exist? Do you notice anything in the output images (patters etc.) that is related to the choice of K?

Table 1: Parameters used for the reference images.

	σ_g	σ_u	α	γ	δ	τ	T
CED	1.5	0.7	5×10^{-4}	10^{-4}	—	5	100
EED	0	10	—	—	10^{-4}	1	10

References

[1] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

A. Appendix

Let $\tilde{u}$ denote the image after convolution with a Gaussian kernel with standard deviation σ_u, that is $\tilde{u} = G_{\sigma_u} * u$. For an arbitrary pixel location $(i,j) \in \Omega$, the structure tensor is

$$S_{(i,j)}(u) = \begin{bmatrix} \left(G_{\sigma_g} * \left(\frac{\partial \tilde{u}}{\partial x} \odot \frac{\partial \tilde{u}}{\partial x}\right)\right)(i,j) & \left(G_{\sigma_g} * \left(\frac{\partial \tilde{u}}{\partial x} \odot \frac{\partial \tilde{u}}{\partial y}\right)\right)(i,j) \\ \left(G_{\sigma_g} * \left(\frac{\partial \tilde{u}}{\partial y} \odot \frac{\partial \tilde{u}}{\partial x}\right)\right)(i,j) & \left(G_{\sigma_g} * \left(\frac{\partial \tilde{u}}{\partial y} \odot \frac{\partial \tilde{u}}{\partial y}\right)\right)(i,j) \end{bmatrix}.$$

In the above, $\odot$ is an element-wise product.