

# GPU Programming in Computer Vision: Day 3

---

Date: Wednesday, 19. September 2018

---

## Exercise 9: Isotropic Diffusion (8P)

Implement the nonlinear isotropic diffusion

$$\partial_t u = \operatorname{div} \left( \widehat{g}(|\nabla u|) \nabla u \right),$$

for different choices of  $\widehat{g}$ . This means updating  $u$  in the following way:

$$u^{n+1} = u^n + \tau \operatorname{div}(\widehat{g}(|\nabla u^n|) \nabla u^n).$$

As a initial condition, pick  $u^0$  as the input image. For simplicity, we consider  $\widehat{g}(s) = 1$  at first.

1. Use forward differences to compute the derivatives  $v_1 := \partial_x^+ u$  and  $v_2 := \partial_y^+ u$ .  
Reuse your code from exercise 4.
2. Compute the diffusivity  $g$  from  $v_1, v_2$ . Use a “`_host_ _device_`” function for  $\widehat{g}$ . *Hint:* Note that  $g$  is *scalar*, there is only one value  $g$ , which is shared for all channels.
3. Multiply  $v_1, v_2$  by  $g$ , and store the result again in  $v_1, v_2$ .  
If you want, you can combine steps 2 and 3 into a single kernel. Note that then you don't need an array for  $g$ , because you can compute  $g$  locally in the kernel.
4. Use backward differences to compute the divergence:  $d := \operatorname{div} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \partial_x^- v_1 + \partial_y^- v_2$ .  
Reuse your code from exercise 4.  $y$
5. Compute the update step for  $u$ , update all of the  $n_c$  channels in a single kernel. You can implement this as a separate kernel, or as part of the div-kernel from step 3.
6. Compute  $N$  iterations of the diffusion and visualize the end result. Experiment with different time steps  $\tau$  and different numbers of iterations  $N$ . A necessary condition for convergence is  $\tau < 0.25/\widehat{g}(0)$ , i.e.  $\tau < 0.25$  for  $\widehat{g}(s) = 1$ . What happens if  $\tau$  is chosen too big? What happens for very large  $N$ ?
7. Compare the result to Gaussian convolution  $G_\sigma * u$  with  $\sigma = \sqrt{2\tau N}$ . What do you observe?
8. Now try using a different diffusivity function:
  - $\widehat{g}(s) = \frac{1}{\max(\varepsilon, s)}$ ,
  - $\widehat{g}(s) = \exp(-s^2/\varepsilon)/\varepsilon$ .

How do the results change in each case?

## Exercise 10: Anisotropic Diffusion

(5P, Bonus)

Implement the *anisotropic* linear diffusion

$$\partial_t u = \operatorname{div}(G \nabla u),$$

by iterating the explicit forward Euler scheme

$$u^{n+1} = u^n + \tau \operatorname{div}(G \nabla u^n).$$

Here,  $G(x, y) \in \mathbb{R}^{2 \times 2}$  is a  $2 \times 2$  diffusion tensor which is different at every pixel  $(x, y)$  in the domain and is calculated once from the input image. Its aim is to make the diffusion process preserve image edges.

1. Compute the diffusion tensor  $G$  from the input image:
  - (a) First calculate the structure tensor of the input image. Postsmooth the structure tensor with a gaussian kernel with a different standard deviation parameter  $\rho$ . Reuse parts of your code from exercise 7.
  - (b) Now calculate the Eigenvalues  $\lambda_1 \geq \lambda_2$  and corresponding eigenvectors  $e_1, e_2$  of the structure tensor  $G$  in each point. For that, extend the `__device__` function from exercise 8 to also calculate and return the eigenvectors.
  - (c) Set the diffusion tensor to  $G = \mu_1 e_1 e_1^T + \mu_2 e_2 e_2^T$  with

$$\mu_1 = \alpha, \quad \lambda_1 = \lambda_2, \\ \mu_2 = \begin{cases} \alpha & \\ \alpha + (1 - \alpha) \exp\left(-\frac{C}{(\lambda_1 - \lambda_2)^2}\right) & \text{else,} \end{cases}$$

where  $C > 0$  and  $\alpha \in (0, 1)$  are parameters.

2. Implement the diffusion as outlined in the previous exercise, except that the diffusivity is now matrix valued and constant. Perform a  $2 \times 2$  matrix-vector multiplication on the gradient instead of the scalar multiplication in exercise 10.
3. Try out different choices of  $C$ ,  $\alpha$  and iteration numbers  $N$ . Remember to choose  $\tau < 0.25$  small enough. Try it out on the image `van-gogh.png` and pick  $C = 5 \cdot 10^{-6}$ ,  $\alpha = 0.01$ ,  $\sigma = 0.5$  and  $\rho = 3$  as a start. *Hint: for debugging purposes it might be useful to visualize the diffusion tensor!*