Applied Medical Image Processing Lecture Notes

1 Bilateral Filter:

1.1 Definition:

A filter where weights depend only on the distance in the spatial domain is called domain filter. Pixels that are closer to the filter center are given more weight and farther pixels are given less weight. To make smoothing less destructive on the edges, we have to exclude pixels or reduce their weight fi they are very dissimilar in value to the center pixel. These type of filters rely on pixel values or range as opposed to distance hence they are called range filters. Bilateral filter is the combination of the domain and range filters. Mathematically:

$$\begin{aligned} I'(u,v) &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} I(u+m,v+n)\cdot Hd(m,n) \\ &= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(i,j)\cdot Hd(i-u,j-v) \\ I'(u,v) &= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(i,j)\cdot \mathrm{Hr}(I(i,j)-I(u,v)) \end{aligned}$$

Where Hd and Hr or domain and range filter kernels, respectively. Range filter only applies to the pixels with similar intensity values. If we combine these two sets of filters, we will get:

$$\begin{aligned} I'(u,v) = \tfrac{1}{W_{u,v}}\textstyle\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(i;j)\cdot H_d(i-u,j-v)\cdot H_r(I(i,j)-I(u,v)) \\ W_{u,v} = \textstyle\sum_i\sum_j w_{i,j} = \sum_i\sum_j H_d(i-u,j-v)\cdot H_r(I(i,j)-I(u,v)) \end{aligned}$$

At a given filter position the weight assigned to each contributing pixel depends on 1) spatial position relative to the center pixel position and 2) the similarity of pixel value to the value of the center pixel.

For a homogeneous region with similar intensity, Bilateral filter is similar to a linear weight average. For the regions that contain edges, only similar pixels are smoothed, therefore it preserves the edge. This filter is space variant and non-linear. If we have a kernel with radius k then the size of kernel will be $2k+1$, accordingly the above equation can be written as:

$$I'(u,v) = \frac{\sum_{i=u-k}^{u+k} \sum_{j=v-k}^{v+k} I(i,j) \cdot H_d(i-u, j-v) \cdot H_r(I(i,j) - I(u,v))}{\sum_i \sum_j H_d(i-u, j-v) \cdot H_r(I(i,j) - I(u,v))}$$

If we substitute $i-u$ with m and $j-v$ with n, then we can rewrite the effective kernel as follows:

$$\bar{H}_{I,u,v}(i,j) = \begin{cases} \frac{H_d(i,j) \cdot H_r(I(u+i, v+j) - I(u,v))}{\sum\limits_{m=-k}^{k} \sum\limits_{n=-k}^{k} H_d(m,n) \cdot H_r(I(u+m, v+n) - I(u,v))} & \text{if } -K \leqslant i,j \leqslant K \\ 0 & \text{otherwise} \end{cases}$$

1.2 Example:

We can use Gaussian function for both the domain and range,

$$\begin{aligned} H_d^{G,\sigma_d}(m,n) &= \frac{1}{2\pi\sigma_d^2} e^{-\frac{\rho^2}{2\sigma_d^2}} \\ &= \frac{1}{2\pi\sigma_d^2} e^{-\frac{\left(m^2+n^2\right)}{2\sigma_d^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\delta_d} e^{-\frac{m^2}{2\sigma_d^2}} \right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma_d} e^{-\frac{n^2}{2\sigma_d^2}} \right) \end{aligned}$$

$$H_r^{G,\sigma_r}(x) = \frac{1}{\sqrt{2\pi}\sigma_r} e^{-\frac{x^2}{2\sigma_r^2}}$$

$$I'(u,v) = \frac{1}{w_{x,v}} \cdot \sum_{m=-k}^{k} \sum_{n=-k}^{k} I(u+m, v+n) e^{\frac{-\left(m^2+n^2\right)}{2\delta_d^2}} \cdot e^{-\frac{(I(u+m,v+n) - I(u,v))^2}{2\delta_r^2}}$$

Where

$$w(u,v) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} e^{\frac{-\left(m^2+n^2\right)}{2\sigma_d^2}} \cdot e^{-\frac{(I(u+m,v+n)-I(u,v))^2}{2\sigma_r^2}}$$

Here $k = \lceil 3.5\sigma_d \rceil$

Steps to perform bilateral filtering is summarized in Algorithm 1:

Algorithm 1 BilateralFilterGray(I, σ_d, σ_r) see reference [1]

Input: I , a grayscale image of size $M \times N$; σ_d , width of the 2D Gaussian *domain* kernel; σ_r, width of the 1D Gaussian *range* kernel;
Returns a new filtered image of size $M \times N$.

```
1: $(M, N) \leftarrow \text{Size}(I)$
2: $K \leftarrow \lceil 3.5\sigma_d \rceil$
3: $I' \leftarrow Duplicate(I)$
4: for all image coordinates $(u, v) \in M \times N$ do
5:     $S \leftarrow 0$
6:     $W \leftarrow 0$
7:     $a \leftarrow I(u, v)$
8:     for $m \leftarrow -K, \ldots, K$ do
9:         for $n \leftarrow -K, \ldots, K$ do
10:            $b \leftarrow I(u+m, v+n)$
11:            $w_d \leftarrow e^{-\frac{(m^2+n^2)}{2\sigma_d^2}}$
12:            $w_r \leftarrow e^{-\frac{(a-b)^2}{2\sigma_r^2}}$
13:            $w \leftarrow w_d \cdot w_r$
14:            $S \leftarrow S + w \cdot b$
15:            $W \leftarrow W + w$
16:        end for
17:    end for
18:    $I'(u, v) \leftarrow \frac{1}{W} \cdot S$
19: end for
20: return $I'$
```

1.3 Separable Implementation:

It is possible to accelerate the process of filtering by using a sequence of $1D$ filters. For example, we could use 1D H$_d$ and H$_r$ to get an image called I$'$

$$I'(u,v) = \frac{\sum_{m=-K}^{K} I(u+m,v) \cdot H_d(m) H_r(I(u+m,v))}{\sum_{m=-K}^{K} H_d(m) \cdot H_r(I(u+m,v) - I(u,v))}$$

followed by:

$$I''(u,v) = \frac{\sum_{n=-K}^{K} I'(u,v+n) \cdot H_d(n) H_r(I'(u,v+n))}{\sum_{n=-K}^{k} H_d(n) \cdot H_r(I'(u,v+n) - I'(u,v))}$$

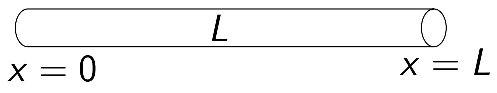
The implementation of separable bilateral filtering is summarized in Algorithm 2.

2 Anisotropic Diffusion Filter:

This filter is based on the concept of diffusion in material in order to preserve edge. The idea is to limit the diffusion process parallel to edges.

2.1 Simple Example:

Starting with a simple example, we would like to quantify the rate of temperature change at differentime points in a one dimensional example using an infinitely thin rod of length L.



Let's define $u(x,t)$ as a function representing the temperature value at location " x " and time t. In this model, heat dissipates according to

$$\frac{\partial u}{\partial t} = u_t = \frac{\partial\left(k u_x\right)}{\partial x}. \quad u_x = \frac{\partial u}{\partial x}$$

Algorithm 2 BilateralFilterGraySep(I, σ_d, σ_r) see reference [1]

```
1:  Input: $I$, a grayscale image of size $M \times N$; $\sigma_d$, width of the 2D Gaussian domain kernel; $\sigma_r$,
    width of the 1D Gaussian range kernel
2:  Output: A new filtered image of size $M \times N$
3:  $(M, N) \leftarrow \text{Size}(I)$
4:  $K \leftarrow \lceil 3.5 \cdot \sigma_d \rceil$
5:  $I' \leftarrow \textit{Duplicate}(I)$
6:  for all image coordinates $(u, v) \in M \times N$ do
7:      $S \leftarrow 0$
8:      $W \leftarrow 0$
9:      $a \leftarrow I(u, v)$
10:     for $m \leftarrow -K, \ldots, K$ do
11:         $b \leftarrow I(u + m, v)$
12:         $w_d \leftarrow e^{-\frac{m^2}{2\sigma_d^2}}$
13:         $w_r \leftarrow e^{-\frac{(a-b)^2}{2\sigma_r^2}}$
14:         $w \leftarrow w_d \cdot w_r$
15:         $S \leftarrow S + w \cdot b$
16:         $W \leftarrow W + w$
17:     end for
18:     $I'(u, v) \leftarrow \frac{1}{W} \cdot S$
19: end for
20: $I'' \leftarrow \textit{Duplicate}(I)$
21: for all image coordinates $(u, v) \in M \times N$ do
22:     $S \leftarrow 0$
23:     $W \leftarrow 0$
24:     $a \leftarrow I'(u, v)$
25:     for $n \leftarrow -K, \ldots, K$ do
26:         $b \leftarrow I'(u, v + n)$
27:         $w_d \leftarrow e^{-\frac{n^2}{2\sigma_d^2}}$
28:         $w_r \leftarrow e^{-\frac{(a-b)^2}{2\sigma_r^2}}$
29:         $w \leftarrow w_d \cdot w_r$
30:         $S \leftarrow S + w \cdot b$
31:         $W \leftarrow W + w$
32:     end for
33:     $I''(u, v) \leftarrow \frac{1}{W} \cdot S$
34: end for
35: return $I''$
```

Which state that the rate of change of temperature over time is proportional to a diffusion parameter k and the rate of change of temperature relative to the position parameter x.

For this differential equation, we can consider three different boundary conditions at positions $x = 0$ and $x = L$:

- Dirichlet:
$$u(0,t) = a$$
$$u(L,t) = b$$
- Periodic:
$$u(0,t) = U(L,t)$$
- Newman:
$$u_t(0,t) = u_t(L,t) = 0$$

For isotropic diffusion, K is constant, therefore diffusion equation can be simplified as:

$$u_t = \frac{\partial\left(ku_x\right)}{\partial x} = k\frac{\partial u_x}{\partial x}$$

if $K(x)$ changes along the bar $u_t = \frac{\partial(ku_x)}{\partial x}$ is an anisotropic diffusion. We cam extend this equation to higher dimensions:

$$u_t = \nabla \cdot (k\nabla u)$$

reminder:

$$\text{if } \vec{V} = (V_x, V_y) \text{ or } (V_1, V_2)$$
$$\text{then } \nabla \cdot \vec{V} = \frac{\partial V_1}{\partial x} + \frac{\partial V_2}{\partial y}$$
$$\nabla \vec{V} : \left\langle \frac{\partial V_1}{\partial x}, \frac{\partial V_2}{\partial y} \right\rangle$$

Using this notation for an isotropic case where K is constant

$$\begin{aligned} u_t &= \nabla \cdot (k\nabla u) \quad \nabla u = \langle u_x, u_y \rangle \\ u_t &= K\nabla \cdot (\nabla u) \\ &= K\left[\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right] \\ &= K(u_{xx} + u_{yy}) \end{aligned}$$

Note: $\nabla \cdot (\nabla u)$ is called Laplaien

$$\nabla \cdot (\nabla u) = \nabla^2 u = \Delta u$$
$$u_t = K \nabla^2 u = K \Delta u$$

2.1.1 Numerical Solution:

Numerically, we will define the first and second derivatives as follows:

$$\begin{aligned}
u_x &\approx \frac{u(x+1,y) - u(x-1,y)}{2} \\
u_y &\approx \frac{u(x,y+1) - u(x,y-1)}{2} \\
u_{xx} &\approx u(x+1,y) - 2u(x,y) + u(x-1,y) \\
u_{yy} &\approx u(x,y+1) - 2u(x,y) + u(x,y-1)
\end{aligned}$$

These definitions are based on the Finite difference method described bellow:

$$\begin{aligned}
f_x(x,y) &\approx \frac{f(x+h,y) - f(x-h,y)}{2h} \\
f_{xx}(x,y) &\approx \frac{f(x+h,y) - 2f(x,y) + f(x-hy)}{h^2} \\
&\text{we assume } h = 1
\end{aligned}$$

Also, we define numerical derivative for the time variable as follows:

$$u_t(x,y,t+\Delta t) \approx \frac{u(x,y,t+\Delta t) - u(x,y,t)}{\Delta t}$$

Assume $u^n(x,y)$ donates the n^{th} iteration at time $t = n\Delta t$, then we can solve the diffusion equation iteratively:

$$\begin{aligned}
u_t^{n+1} &= \frac{u^{n+1} - u^n}{\Delta t} \qquad \text{Given: } u_t = K\left[u_{xx} + u_{yy}\right] \\
\frac{u^{n+1} - u^n}{\Delta t} &= k\left[u_{xx}^n + u_{yy}^n\right] \\
u_{xx}^n &= u^n(x+1,y) - 2u^n(x,y) + u^n(x-1,y) \\
u_{yy}^n &= u^n(x,y+1) - 2u^n(x,y) + u^n(x,y-1) \\
u^{n+1} &= u^n + k\Delta t\left[u_{xx}^n + u_{yy}^n\right]
\end{aligned}$$

2.1.2 Diffusion equation and Images:

We can model an image as $I_A \approx u(\vec{x}, t)$, where local intensities are similar to temperature. In this setup the diffusion equation takes the following form:

$$\frac{\partial I(\vec{x}, t)}{\partial t} = K\nabla^2 I = \frac{\partial^2 I(\vec{x}, t)}{\partial x^2} + \frac{\partial^2 I(x, t)}{\partial y^2}$$

for now, let's assume $K = 1$, then $\frac{\partial I(\vec{x},t)}{\partial t} = I_{xx}(\vec{x}, t) + I_{yy}(\vec{x}, t)$, which can be solved iteratively using:

$$I^n(\vec{x}) = I^{n-1}(\vec{x}) + \alpha \cdot \left[\nabla^2 I^{n-1}(\vec{x})\right]$$

Here $I^0 = I_{\text{original image}}$ and α controls the speed of diffusion. Its value should be adjusted according to the application and image. To calculate the Laplacian of an image, we could use the following numerical method:

$$\nabla^2 I \approx I * H^2 = I * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

For isotropic diffuser, this process be haves like a Gaussian filter. After nth iteration of diffusion process, we will have:

$$I^{(n)} = I * H^{G,\delta_n}$$

$$H^{G,\delta_n}(x, y) = \frac{1}{2\pi\sigma_n^2} e^{-\frac{(x^2+y^2)}{2\sigma_n^2}}$$

$$\sigma_n = \sqrt{2t} = \sqrt{\frac{2n}{\alpha}}$$

2.2 Perona-Malik Filter:

Idea: make the diffusivity coefficient (conductivity) variable and dependant on local image structure.

$$\frac{\partial I}{\partial t} = \nabla \cdot (K \cdot \nabla I)$$
$$k = k(\vec{x}, t) = g(\|\nabla I(\vec{x}, t)\|)$$

Function g Should return large values when the image gradient is small and small values when it is large. Several options for function g is listed below:

$$g(d) = e^{-\left(\frac{d}{k}\right)^2}$$
$$g(d) = \frac{1}{1 + \left(\frac{d}{k}\right)^2}$$
$$g(d) = \frac{1}{\sqrt{1 + \left(\frac{d}{k}\right)^2}}$$
$$g(d) = \begin{cases} \left(1 + \left(\frac{d}{2k}\right)^2\right)^2 & \text{for } d \leqslant 2k \\ 0 & \text{otherwise} \end{cases}$$

$k > 0$ should be adjusted relative to the noise in image. In the original paper, it has been suggested that

$$I^n(\vec{x}) = I^{n-1}(\vec{x}) + \alpha \cdot \sum_{i=0}^{3} g\left(\left|\delta_i\left(I^{(n-1)}, \vec{x}\right)\right|\right) \cdot$$
$$I^0 = I \text{ onginal image}$$

Where (see Fig. 1):

$$\begin{cases} \delta_i(I, \vec{x}) = I\left(\vec{x} + d_i\right) - I(\vec{x}) \\ d_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad d_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad d_2 = -\begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad d_3 = -\begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{cases}$$

Algorithm 3 [1] summarizes the implementation steps for the Perona-Malik diffusion filter [3].

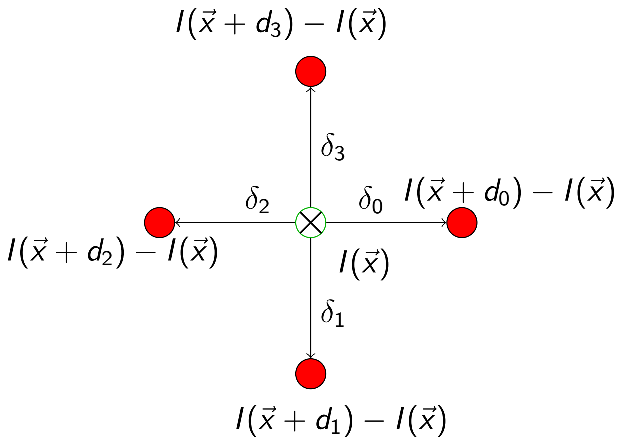

$I(\vec{x} + d_3) - I(\vec{x})$
δ_3
δ_2
δ_0
$I(\vec{x} + d_0) - I(\vec{x})$
$I(\vec{x} + d_2) - I(\vec{x})$
$I(\vec{x})$
δ_1
$I(\vec{x} + d_1) - I(\vec{x})$

Figure 1: Illustration of $\delta_i(I, \vec{x})$ operator relative to the center pixel

Algorithm 3 PeronaMalik(I, α, κ, T)

Input: I , a grayscale image of size $M \times N$; α , update rate; κ, smoothness parameter; T, number of iterations;
Returns the modified image I.

Specify the conductivity function:
$g(d) := e^{-(d/\kappa)^2}$

```
 1: $(M, N) \leftarrow \text{Size}(I)$
 2: Create maps $D_x, D_y : M \times N \to \mathbb{R}$
 3: for $n \leftarrow 1, \ldots, T$ do
 4:     for all coordinates $(u, v) \in M \times N$ do
 5:         $D_x(u, v) \leftarrow \begin{cases} I(u+1, v) - I(u, v) & \text{if } u < M-1 \\ 0 & \text{otherwise} \end{cases}$
 6:         $D_y(u, v) \leftarrow \begin{cases} I(u, v+1) - I(u, v) & \text{if } v < N-1 \\ 0 & \text{otherwise} \end{cases}$
 7:     end for
 8:     for all coordinates $(u, v) \in M \times N$ do
 9:         $\delta_0 \leftarrow D_x(u, v)$
10:         $\delta_1 \leftarrow D_y(u, v)$
11:         $\delta_2 \leftarrow \begin{cases} -D_x(u-1, v) & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases}$
12:         $\delta_3 \leftarrow \begin{cases} -D_y(u, v-1) & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases}$
13:         $I(u, v) \leftarrow I(u, v) + \alpha \cdot \sum_{k=0}^{3} g(|\delta_k|) \cdot \delta_k$
14:     end for
15: end for
16: return $I$
```

Bibliography

[1] Wilhelm Burger, Mark J. Burge , Principles of Digital Image Processing: Advanced Methods, Chapter 5 - Edge-Preserving Smoothing Filters, 2013

[2] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In “Proceedings Int’l Conf. on Computer Vision”, ICCV’98, pp. 839–846, Bombay (1998).

[3] Perona, P., and J. Malik. "Scale-space and edge detection using anisotropic diffusion." IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 12, No. 7, July 1990, pp. 629–639.

[4] Gerig, G., O. Kubler, R. Kikinis, and F. A. Jolesz. "Nonlinear anisotropic filtering of MRI data." IEEE Transactions on Medical Imaging. Vol. 11, No. 2, June 1992, pp. 221–232.