Applied Medical Image Processing Lecture Notes

1 Filters:

1.1 Example:

Filters are typically used to replace a pixel value with a new value using neighborhood information. One application of filters is to reduce noise. For example, replace every pixel by the average of neighborhood pixels (3 x 3 neighborhood).

$$I(u,v) = P_0$$
$$I_s(u,v) = 1/9 \sum_{i=0}^{8} P_i$$

Here, $I(u,v)$ represents the pixel intensity at location u, v in a 2D array. Alternatively, we could write:

$$\begin{aligned} I_s(u,v) = 1/9 \quad & [I(u-1, v_{-1}) + I(u, v_{-1}) + \\ & I(u+1, v-1) + I(u,v) + I(u+1, v) \\ & + I(u-1, v+1) + I(u, v+1) + \\ & I(u+1, v_{+1})] \end{aligned}$$

or:

$$I_s(u,v) = 1/9 \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(u+i, v+j)$$

Size of filter is called support of filter. In this example, it is 3 x3. Factors that define filter are its size, shape (Ex. rectangular or circular), and the weights (coefficients of the filter). As an example consider the average filter:

$$H(i,j) = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

which is a mapping as follows: $H : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. To apply a filter, we use the following equation:

$$I'(u,v) = \sum_{(i,j) \in R_H} I(u+i, v+j).H(x,j)$$

Here R_H is a set of coordinates covered by filter H. Note that a simple average filter results in ringing artifact. It is desirable to normalize wights in a filter to avoid changing the overall intensity scale in the image. Ex.

$$H(i,j) = 1/40 \begin{bmatrix} 3 & 5 & 3 \\ 5 & 8 & 5 \\ 3 & 5 & 3 \end{bmatrix}$$

Another example is Gaussian Filter which is defines as:

$$G_\sigma(r) = e^{-r^2/2\delta^2} \text{ or } G_\sigma(x,y) = e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}}$$

Laplacian filter is a difference filter that can be used to perform edge enhancement:

$$L = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

$$I'(u,v) = \sum_{i,j \in \mathbb{R}_H^+} I(u+i; v+j) \cdot |H(i,j)|$$
$$- \sum_{(i,j) \in \mathbb{R}_H^-} I(u+i; v+j).|H(i,j)|$$

Recall that operation that is associated with the linear filters is represented by convolution. By definition

$$I'(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i; v-j) \cdot \underbrace{H(i;j)}_{\text{kernel}}$$
$$I' = I * H$$
$$I'(u,v) = \sum_{i,j \in \mathbb{R}_H} I(u-i, v-j) \cdot H(i,j)$$
$$= \sum_{i,j} I\left(u+i; v_{+j}\right).H(-i;-j)$$
$$= \sum_{i,j} I(u+i, v+j) \cdot H^*(i,j)$$

Here $H^*(i,j)$ is a reflected filter.

Properties of convolution can be listed as follows:

1. $I * H = H * I$
2. if $s \in \mathbb{R}$ then $(s.I) * H = I * s.H = s \cdot (I * H)$
3. $$(I_1 + I_2) * H = (I_1 * H) + (I_2 * H)$$
   However, if $b \in \mathbb{R}$, then $(b + I) * H \neq b + (I * H)$
4. $$I_1 * (I_2 * I_3) = (I_1 * I_2) * I_3$$
   if $H = H_1 * H_2 * H_3 * H_3 \cdots$
   $$I * H = I * (H_1 * H_2 * \ldots * H_n)$$
   $$= ((I * H_1) * H_2 * \ldots * H_n)$$

1.2 Separability:

Separating 2D Kernels H into pair 1D kernels H_x and H_y, for example:

$$H_x = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} H_y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$I' = (I * H_x) * H_y \Rightarrow I' = I * (\underbrace{H_x * H_y}_{H_{xy}})$$

$$H_{xy} = H_x * H_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Advantage is reduction in the computational cost. For $2D$ filter we need $5 \times 3 = 15$ steps, however, for two separate 1D filtering we need $5 + 3 = 8$ steps . Other example is the 2D Gaussian filter:

$$G_\sigma(x, y) = e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}} = e^{-x^2/2\sigma^2} \cdot e^{-y^2/2\sigma^2}$$

then

$$I' = I_* H^{G,\sigma} = I_* H_x^{G,\sigma} * H_y^{G,\sigma}$$

Another example is Binomial filter. A1D Filter of order b is defined as:

$$C^{1D}_{\text{binomal},\, b} = \frac{1}{2^b} c^{1D}_{\text{binu},\, b-1} * c^{1D}_{\text{binom},\, 1}$$

$$C^{1D}_{\text{binom}\ 1} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$C^{2D}_{\text{binom},\, b} = C^{1D}_{\text{binom},\, ,b} \otimes C^{1D}_{\text{binom},\, ,b}$

Ex: $C^{1D}_{\text{binom}\ ,2} = \frac{1}{2^2} \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

$$c^{2D}_{\text{binom},2} = 1/4 \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \otimes 1/4 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$1/4 \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \cdot 1/4 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

1.3 Filters in Fourier Space:

Noises in the Fourier space is represented as high frequency component, so filters are designed to reduce the magnitude of those components. An example is the Gaussian filter. Another example is the Butterworth filter.

$$C_{Butterworth,\omega_{\max}}(u, v) = \frac{1}{1 + \left(\left(u^2 + v^2\right)/\omega^2_{\max}\right)^k}$$

Here k regulates the steepness of the dumping function at $\omega_{\max}$ which is the cut off frequency.

Other examples are Hamming and Hann filters which are defined as:

$$C_{Ha,\omega_{\max}}(u, v) = \begin{cases} \alpha - (1-\alpha)\cos\left(\frac{\sqrt{u^2+v^2}}{\omega_{\max}}\pi\right) & \text{if } u^2 + v^2 < \omega_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Note that for Hamming filter $\alpha = 0.53836$ and for Hann filter $\alpha = 0.5$.

1.4 Edge handling:

we don't process pixels at border where there is no support for the filter. This will result in image size reduction in each iteration of filter. There are several options:

- set the unprocessed pixels to the original unfiltered image value. This results in noticeable difference between filtered and unfiltered pixels.

- pad the border region with additional pixels and filter that area.
  - use constant value for pixels outside image, for example use zero. Causes artifacts with filters that have large support.
  - Border pixels extend beyond the image boundaries.
  - Image is mirrored at each of its four boundaries.
  - image repeats periodically in the horizontal and vertical directions (make sense for Fourier filtering).

1.5 Edge Preserving Filters:

1.5.1 Median Filter:

We are turning our attention towards a class of filters that not only reduce noise in images but also preserve a main characteristic of an image which is the edge information. We previously mentioned that an edge is presented as a location with a sudden change of intensity which will be reflected as a high frequency component in the Fourier space. Similarly noises are large fluctuations in intensity from one pixel to another which will be considered as a high frequency component in the Fourier space. Accordingly, filters that are solely focused on reducing high frequency components in an image will result in blurring of edges in that image. In Median Filter, we replace every pixel value by the median of the pixels in the filter region R. Imagine we have the following 3×3 neighborhood with the following pixel intensity values:

3	7	2
1	0	0
9	5	8

We could stack the numbers row by row and then sort them:

$$\begin{matrix} 3 \\ 7 \\ 2 \\ 1 \\ 0 \\ 0 \\ 9 \\ 5 \\ 8 \end{matrix} \xrightarrow{\text{sort}} \begin{matrix} 0 \\ 0 \\ 1 \\ 2 \\ 3 \\ 5 \\ 7 \\ 8 \\ 9 \end{matrix}$$

Here median is 3 which is the middle number after sorting. Mathematically, we could write:

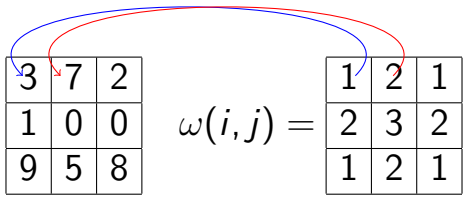
$$I'(u,v) = \text{median}\{I(u+i, v+j) \mid (i,j) \in R\}$$

Where R is a neighborhood region of center pixel (u, v). For a case that we have odd size filter, Ex. 9 ($2k+1|_{k=4}$), with sorted pixel values p_i, then:

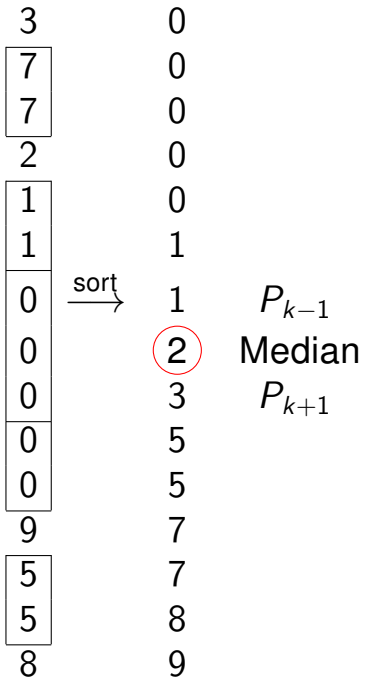
median is calculated as $\{P_0, P_1, P_2, \ldots P_{k-1} \ldots P_{2k}\} \triangleq P_k$.

For even size filter, $2k$, for $k > 0$, median is calculated as:
$\{P_0, P_1, \ldots, P_{k-1}, P_k, P_{k+1}, \ldots P_{2k-1}\} \triangleq \frac{P_{k-1}+P_{k+1}}{2}$

Given that median filter is a rank filter, it has the advantage that outlier numbers such as very high or very low values will not influence the result. Also in any regular median filter, each pixel in the filter has the same vote. However if for any reason we would like to give more weight to one Pixel relative to other one, for example pixels that are farther from the center of the filter should have less weight, then we can define a weight matrix and use that to compute the result of median filter. For example for the following pixel values and weights:



This weights will be used to replicate the pixel intensity values by the number specified by weights. For this example, we will have:



As you will see, pixel value 3 (top left corner) has weight 1 so, it is repeated only once. On the other hand pixel value 7 is associated with weight 2, therefore repeated twice, and so forth.

The edge preserving property of median filter works under the following conditions:

- The edge is straight within the neighborhood region.
- The signal difference of two regions incident to the edge exceeds the noise amplitude.
- The signal is locally constant within each of the two regions.

Problems

- Medinn Filter removes very small regions.
- Median Filter removes corners, only preserves straight edge.

- If there is more than one boundary within the filter region, then boundaries may merge.