Image Processing
INT3404 20
Week 5: Spatial filtering (cont)

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.

Email: ngocdiep@vnu.edu.vn

Schedule

Tuần	Nội dung	Yêu cầu đối với sinh viên (ngoài việc đọc tài liệu tham khảo)
1	Giới thiệu môn học	Cài đặt môi trường: Python 3, OpenCV 3, Numpy, Jupyter Notebook
2	Ảnh số (Digital image) – Phép toán điểm (Point operations) Làm quen với OpenCV + Python Điều chỉnh độ tương phản (Contrast adjust)– Ghép ảnh (Combining images)	
3	Histogram - Histogram equalization	Làm bài tập 1: điều chỉnh gamma tìm contrast hợp lý
4	Phép lọc trong không gian điểm ảnh (linear processing filtering)	Thực hành ở nhà
5	Phép lọc trong không gian điểm ảnh (linear processing filtering) (cont.) Thực hành: Cách tìm filters	Thực hành ở nhà
6	Thực hành: Ứng dụng của histogram; Tìm ảnh mẫu (Template matching)	Bài tập mid-term
7	Trích rút đặc trưng của ảnh Cạnh (Edge) và đường (Line) và texture	Thực hành ở nhà
8	Các phép biến đổi hình thái (Morphological operations)	Làm bài tập 2: tìm barcode
9	Chuyển đổi không gian – Miền tần số – Phép lọc trên miền tần số Thông báo liên quan đồ án môn học	Đăng ký thực hiện đồ án môn học
10	Xử lý ảnh màu (Color digital image)	Làm bài tập 3: Chuyển đổi mô hình màu và thực hiện phân vùng
11	Các phép biến đổi hình học (Geometric transformations)	Thực hành ở nhà
12	Nhiễu – Mô hình nhiễu – Khôi phục ảnh (Noise and restoration)	Thực hành ở nhà
13	Nén ảnh (Compression)	Thực hành ở nhà
14	Hướng dẫn thực hiện đồ án môn học	Trình bày đồ án môn học
15	Hướng dẫn thực hiện đồ án môn học Tổng kết cuối kỳ	Trình bày đồ án môn học

This week outline

1. Recall Spatial Filtering (Week 4)
2. Some properties of Convolution and correlation
3. Filter design
   1. Non-DL
   2. DL

Recall week 4: Spatial filtering

- Neighbors of a pixel
- Distance between two pixels
- Correlation

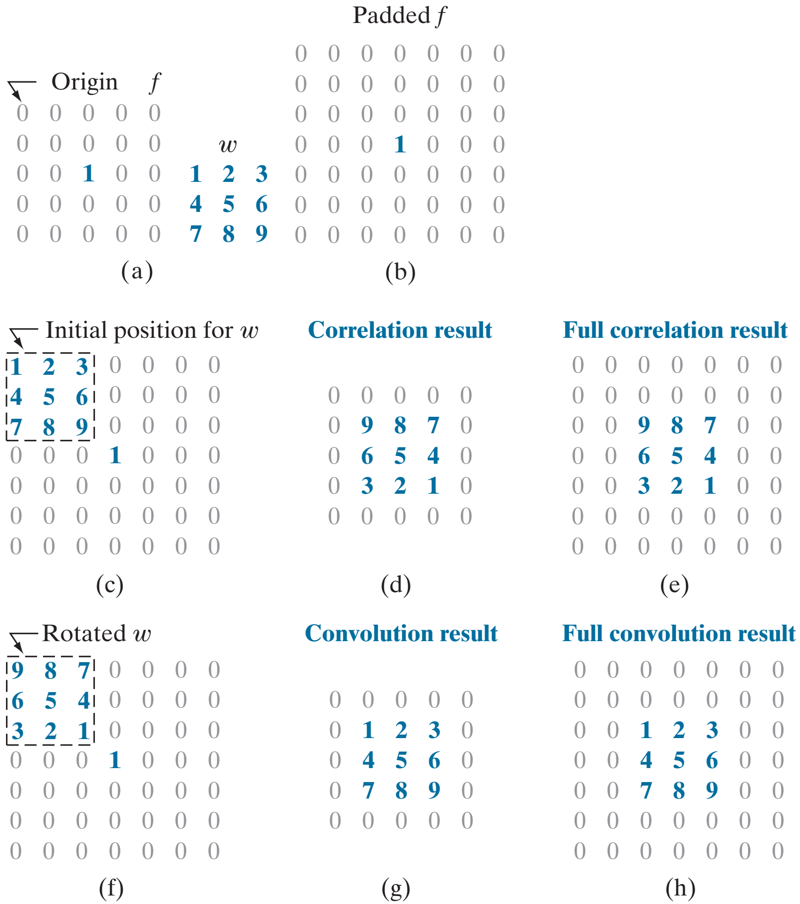
$$(w \star f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

- Convolution

$$(w \bigstar f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

- Filter kernels

Correlation and Convolution in 2D



Source: Fig. 3.30, Gonzalez

Padding

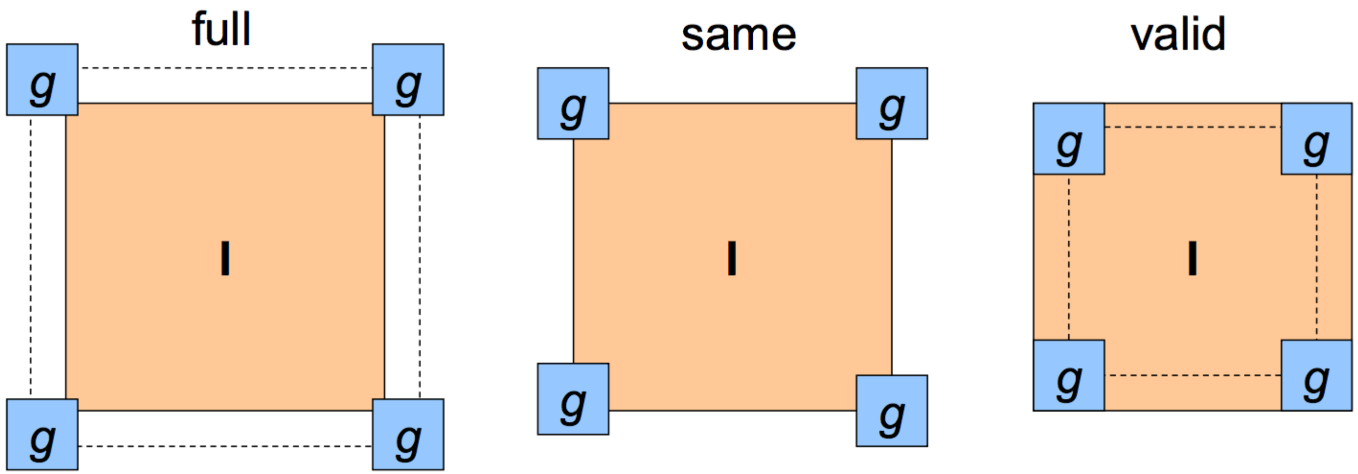


Image size: MxN
Kernel size: mxn

Output: (M + m – 1) x (N +n – 1) M x N (M-m+1) x (N – n + 1)

Padding values at borders

- Pad a constant value (black)
- Wrap around (circulate the image)
- Copy edge (replicate the edges' pixels)
- Reflect across edges (symmetric)

Filter kernels

- Smoothing/Noise reduction/Blurring
  - Box filter
  - Lowpass Gaussian filter
  - Order-statistic (nonlinear) filter
    - Max, min, median
- Other applications:
  - Shading correction
  - Unsharp masking

Properties

Fundamental properties of convolution and correlation

Property	Convolution	Correlation
Commutative	$f \star g = g \star f$	—
Associative	$f \star (g \star h) = (f \star g) \star h$	—
Distributive	$f \star (g + h) = (f \star g) + (f \star h)$	$f \,☆\, (g + h) = (f \,☆\, g) + (f \,☆\, h)$

Simpler convolution computation?

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

$$\frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

Separable filter kernels

- A 2-D function G(x, y) is separable if it can be written as the product of two 1-D functions, $G_1(x, y)$ and $G_2(x, y)$

$$G(x,y) = G_1(x,y)G_2(x,y)$$

- Associative property of convolution

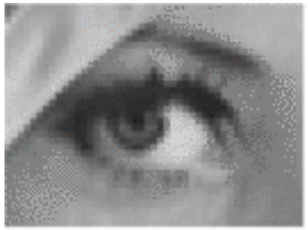
$$w \star f = (w_1 \star w_2) \star f = (w_2 \star w_1) \star f = w_2 \star (w_1 \star f) = (w_1 \star f) \star w_2$$

Computational advantage

Input size: MxN
Kernel size: mxn

$$C = \frac{MNmn}{MN\left(m+n\right)} = \frac{mn}{m+n}$$

Week 5: Filter design



Original

•0	•0	•0
•0	•0	•1
•0	•0	•0

=

?

Kernels with negative values?

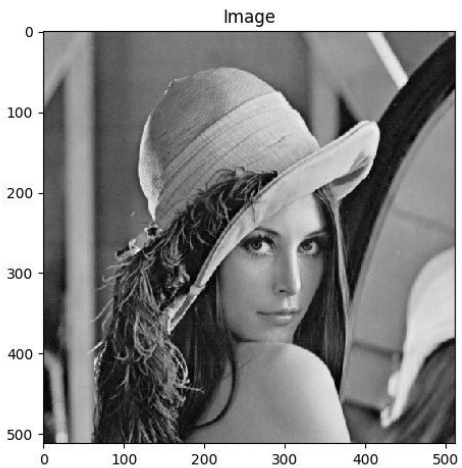
-1	0	1
-2	0	2
-1	0	1

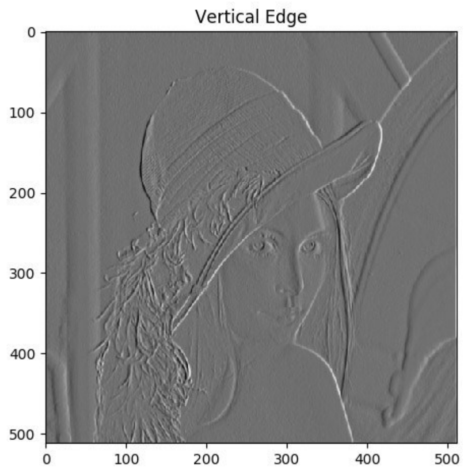
Vertical

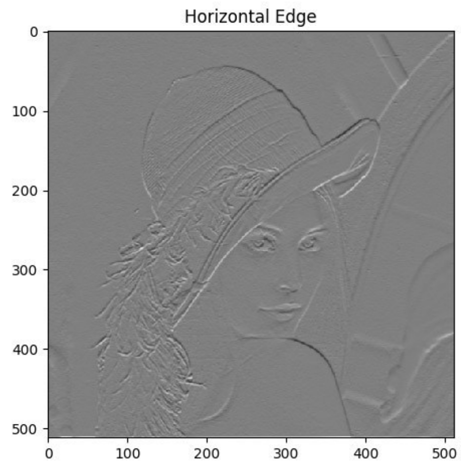
1	2	1
0	0	0
-1	-2	-1

Horizontal

Edge detection





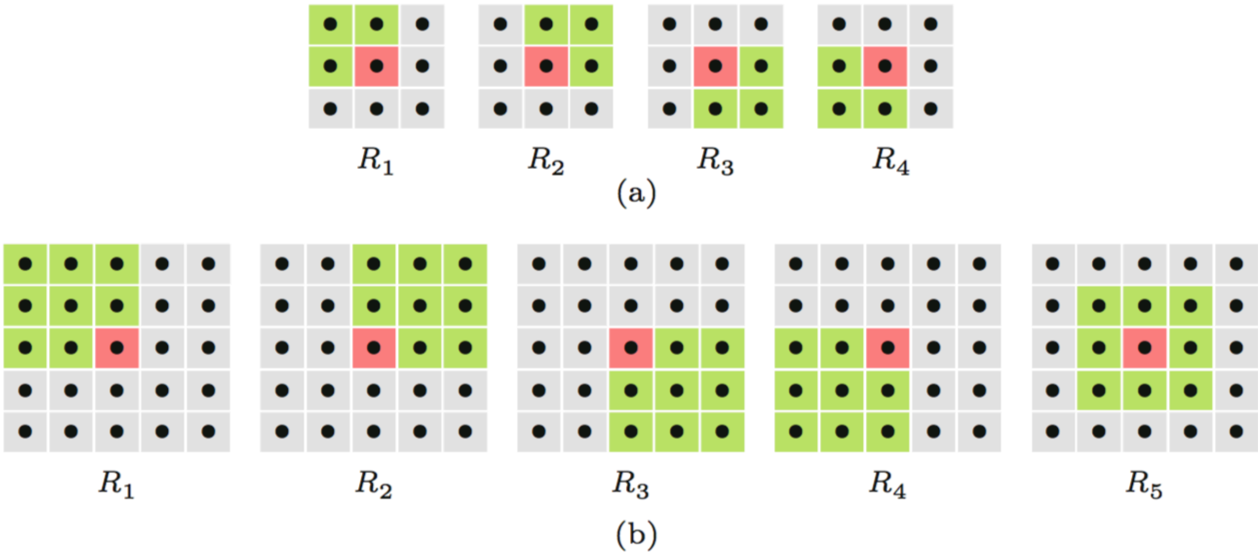


Filter learning

- With DL
  - Automatic learning <-- With a lot of engineering skills
- Without DL
  - With a lot of *a priori* knowledge

Topic:
Edge-preserving smoothing filters

Kuwahara-type filter



17.1 KUWAHARA-TYPE FILTERS

Fig. 17.1
Subregion structures for Kuwahara-type filters. The orginal *Kuwahara-Hachimura* filter (a) considers *four* square, overlapping subregions [144]. *Tomita-Tsuji* filter (b) with *five* subregions ($r = 2$). The current center pixel (red) is contained in all subregions. Das aktuelle Zentralpixel (rot) ist in allen Subregionen enthalten.

Ref:
Burger, Wilhelm, and Mark J. Burge. "Edge-Preserving Smoothing Filters." *Digital Image Processing*. Springer, London, 2016. 413-451.

Kuwahara-type filter

$$\mu_k(I,u,v) = \frac{1}{|R_k|} \cdot \sum_{(i,j)\in R_k} I(u+i,v+j) = \frac{1}{n_k} \cdot S_{1,k}(I,u,v), \tag{17.1}$$

$$\sigma_k^2(I,u,v) = \frac{1}{|R_k|} \cdot \sum_{(i,j)\in R_k} \big(I(u+i,v+j) - \mu_k(I,u,v)\big)^2 \tag{17.2}$$

$$= \frac{1}{|R_k|} \cdot \Big(S_{2,k}(I,u,v) - \frac{S_{1,k}^2(I,u,v)}{|R_k|}\Big), \tag{17.3}$$

for $k = 1, \ldots, K$, with[2]

$$S_{1,k}(I,u,v) = \sum_{(i,j)\in R_k} I(u+i,v+j), \tag{17.4}$$

$$S_{2,k}(I,u,v) = \sum_{(i,j)\in R_k} I^2(u+i,v+j). \tag{17.5}$$

The mean (μ) of the subregion with the smallest variance (σ^2) is selected as the update value, that is,

$$I'(u,v) \leftarrow \mu_{k'}(u,v), \quad \text{with } k' = \operatorname*{argmin}_{k=1,\ldots,K} \sigma_k^2(I,u,v). \tag{17.6}$$

Kuwahara-type filter

(a) RGB test image with selected details

(b) $r = 1$ (3×3 filter)

(c) $r = 2$ (5×5 filter)

(d) $r = 3$ (7×7 filter)

(e) $r = 4$ (9×9 filter)

Nagao-Matsuyama filter

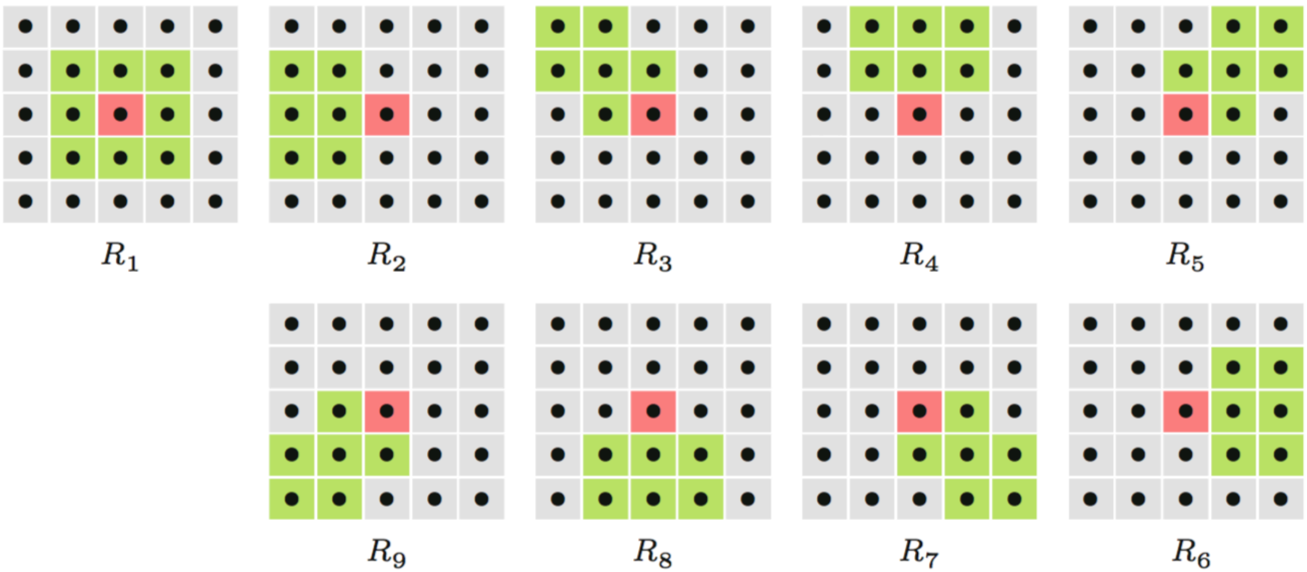


Fig. 17.2
Subregions for the 5×5 ($r = 2$) *Nagao-Matsuyama* filter [170]. Note that the centered subregion (R_1) has a different size than the remaining subregions ($R_2, \ldots, R_9$).

Domain filter vs Range filter

- Domain filter: weights depend only on the distance in the spatial domain

$$I'(u,v) \leftarrow \sum_{\substack{m=\\-\infty}}^{\infty} \sum_{\substack{n=\\-\infty}}^{\infty} I(u+m, v+n) \cdot H(m,n)$$
$$= \sum_{\substack{i=\\-\infty}}^{\infty} \sum_{\substack{j=\\-\infty}}^{\infty} I(i,j) \cdot H(i-u, j-v),$$

Cause some spatial effect upon the image: blurring or sharpening

- Range filter: weights depend only upon the differences in pixel values or range

$$I'_r(u,v) \leftarrow \sum_{\substack{i=\\-\infty}}^{\infty} \sum_{\substack{j=\\-\infty}}^{\infty} I(i,j) \cdot H_{\mathrm{r}}\big(I(i,j) - I(u,v)\big).$$

Act as a point opration

Bilateral filter

- Combining both domain filtering and range filtering

$$I'(u,v) = \frac{1}{W_{u,v}} \cdot \sum_{\substack{i= \\ -\infty}}^{\infty} \sum_{\substack{j= \\ -\infty}}^{\infty} I(i,j) \cdot \underbrace{H_{\mathrm{d}}(i-u, j-v) \cdot H_{\mathrm{r}}\big(I(i,j)-I(u,v)\big)}_{w_{i,j}},$$

where H_{d}, H_{r} are the *domain* and *range* kernels, respectively, $w_{i,j}$ are the composite weights, and

$$W_{u,v} = \sum_{\substack{i= \\ -\infty}}^{\infty} \sum_{\substack{j= \\ -\infty}}^{\infty} w_{i,j} = \sum_{\substack{i= \\ -\infty}}^{\infty} \sum_{\substack{j= \\ -\infty}}^{\infty} H_{\mathrm{d}}(i-u, j-v) \cdot H_{\mathrm{r}}\big(I(i,j)-I(u,v)\big)$$

is the (position-dependent) sum of the weights $w_{i,j}$ used to normalize the combined filter kernel.

(a) $\sigma_r = 10$

(b) $\sigma_r = 20$

(c) $\sigma_r = 50$

(d) $\sigma_r = 100$

Fig. 17.11
Bilateral filter—color example. A Gaussian kernel with $\sigma_d = 2.0$ (kernel size 15×15) is used for the domain part of the filter; working color space is sRGB. The width of the range filter is varied from $\sigma_r = 10$ to 100. The filter was applied in sRGB color space.

A Benchmark for Edge-Preserving Image Smoothing

Feida Zhu, *Student Member, IEEE,* Zhetong Liang, *Student Member, IEEE,* Xixi Jia, *Student Member, IEEE,*
Lei Zhang, *Fellow, IEEE,* and Yizhou Yu, *Fellow, IEEE*

[6] B. Ham, M. Cho, and J. Ponce, “Robust guided image filtering using nonconvex potentials,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] L. Xu, C. Lu, Y. Xu, and J. Jia, “Image smoothing via l 0 gradient minimization,” in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6. ACM, 2011, p. 174.

[8] D. Min, S. Choi, J. Lu, B. Ham, K. Sohn, and M. N. Do, “Fast global image smoothing based on weighted least squares,” *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5638–5653, 2014.

[9] L. Bao, Y. Song, Q. Yang, H. Yuan, and G. Wang, “Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree,” *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 555–569, 2014.

[10] Q. Zhang, L. Xu, and J. Jia, “100+ times faster weighted median filter (wmf),” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2830–2837.

[11] S. Paris, S. W. Hasinoff, and J. Kautz, “Local laplacian filters: Edge- aware image processing with a laplacian pyramid.” *ACM Trans. Graph.*, vol. 30, no. 4, pp. 68–1, 2011.

[12] S. Bi, X. Han, and Y. Yu, “An l 1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 78, 2015.