Image Processing

CS4243 Computer Vision and Pattern Recognition

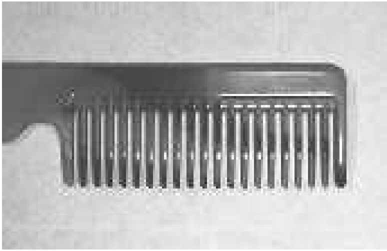
Leow Wee Kheng

Department of Computer Science
School of Computing
National University of Singapore

Outline

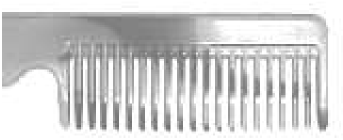
Basics of Image Processing

Suppose you have an image $f(x, y)$.

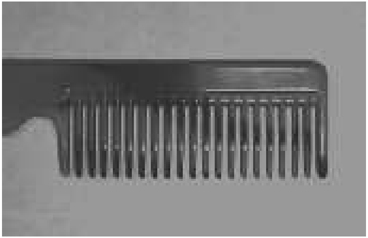


What happens if you multiply a constant c to the intensity value of each pixel?

$$r(x, y) = c\, f(x, y)$$



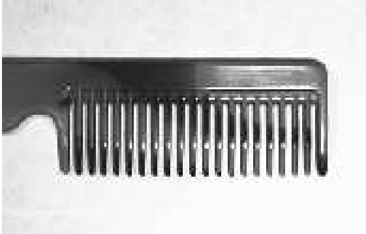
$c = 1.5$



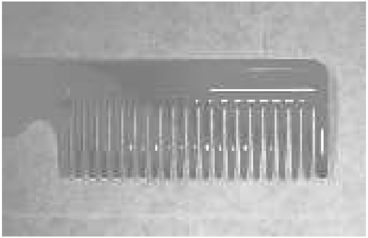
$c = 0.6$

- $c > 1$: image becomes brighter.
- $c < 1$: image becomes darker.

What if you multiply different values to different pixels?



(a) Contrast enhancement



(b) Contrast reduction

- Contrast enhancement:
  make bright pixels brighter, dark pixels darker.
- Contrast reduction:
  make bright pixels darker, dark pixels brighter.

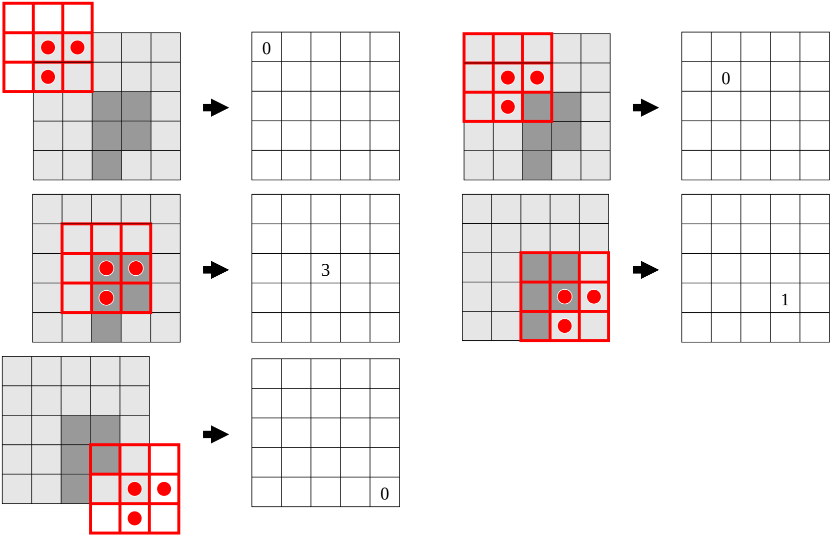
This type of image manipulation is called point processing.

Convolution & Cross Correlation

Now, try something special:

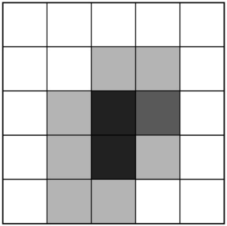
- place a grid, called mask, over a part of an image
- multiply pixels under red dot by 1
- multiply pixels under empty grid by 0
- add up the products

For the image, take dark pixel value = 1, light pixel value = 0.



The final result is

0	0	0	0	0
0	0	1	1	0
0	1	3	2	0
0	1	3	1	0
0	1	1	0	0



This type of image manipulation is called neighbourhood processing.

In particular, the above process is called template matching.
It finds the locations at which the template best matches the image.

Template matching is a kind of cross correlation.

Convolution

Convolution between image $f(x,y)$ and kernel $k(x,y)$ is

$$f(x,y) * k(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(u,v)\, k(x-u, y-v)\, du\, dv \tag{1}$$

In discrete form,

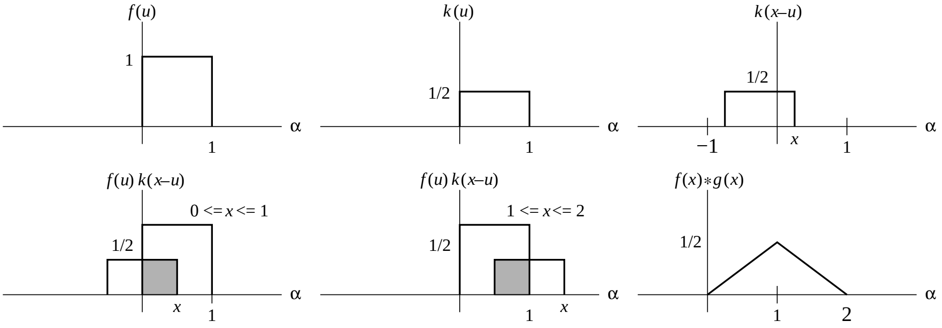
$$f(x,y) * k(x,y) = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} f(i,j)\, k(x-i, y-j) \tag{2}$$

where W and H are the the width and height of the image.

Convolution is commutative (Exercise):

$$f(x,y) * k(x,y) = k(x,y) * f(x,y). \tag{3}$$

1D Example



Demo

Cross Correlation

Cross correlation between image $f(x, y)$ and kernel $k(x, y)$ is

$$f(x, y) \circ k(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(u, v)\, k(x + u, y + v)\, du\, dv \tag{4}$$

In discrete form,

$$f(x, y) \circ k(x, y) = \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} f(i, j)\, k(x + i, y + j) \tag{5}$$

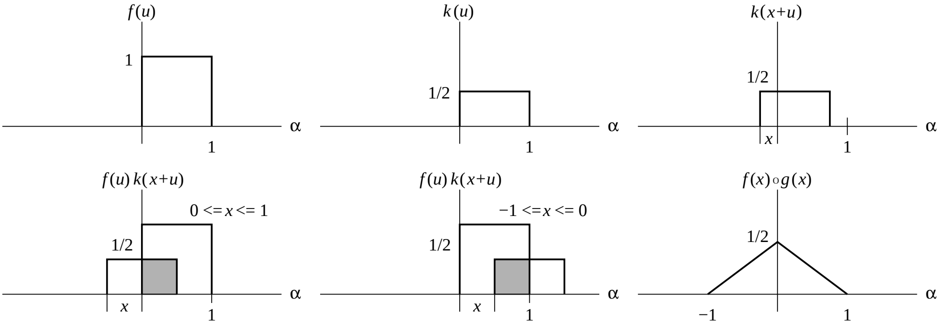
where W and H are the the width and height of the image.

If $f = k$, then it is called auto-correlation.

Cross correlation and convolution are related by (Exercise):

$$f(x, y) \circ k(x, y) = f(-x, -y) * k(x, y). \tag{6}$$

1D Example



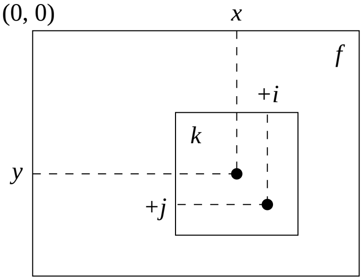
Notes:

- $+x$ slides kernel k to the left ($-x$ direction).
- $-x$ slides kernel k to the right ($+x$ direction).

More convenient way to implement cross correlation:

$$f(x, y) \circ k(x, y) = \sum_{i=-w/2}^{w/2} \sum_{j=-h/2}^{h/2} f(x+i, y+j)\, k(i, j) \tag{7}$$

where w and h are the width and height of template k.



- f has origin at the top-left (or bottom-left) corner.
- k has origin in the middle; need odd-sized mask.
- $+x, +y$ slide template towards $+x, +y$ directions.

Symmetric Kernel

Convolution is commutative. So,

$$\begin{aligned} f(x,y) * k(x,y) &= k(x,y) * f(x,y) && (8) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k(u,v)\, f(x-u, y-v)\, du\, dv && (9) \end{aligned}$$

Substituting $\mu = -u$ and $\nu = -v$ into Eq. 9 gives

$$f(x,y) * k(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x+\mu, y+\nu)\, k(-\mu, -\nu)\, d\mu\, d\nu \qquad (10)$$

Since k is symmetric, i.e., $k(x,y) = k(-x,-y)$, we obtain

$$f(x,y) * k(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x+\mu, y+\nu)\, k(\mu, \nu)\, d\mu\, d\nu \qquad (11)$$

Thus, convolution is equal to cross correlation if kernel is symmetric.

Applications

Convolve (or correlate) image with different kernels produces different results:

- uniform mask: box filter, averaging, smoothing and remove noise
- Gaussian: smoothing and remove noise
- difference mask: edge detection
- difference of Gaussian: edge detection

Box Filter

Noise can be reduced by applying a **box filter**:

1	1	1
1	1	1
1	1	1

$$f(x,y) * k(x,y) = \sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} f(x+i, y+j) \tag{12}$$

Usually, we normalize the mask values so that

$$\sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} k(i,j) = 1 \tag{13}$$

3×3 normalized box filter:

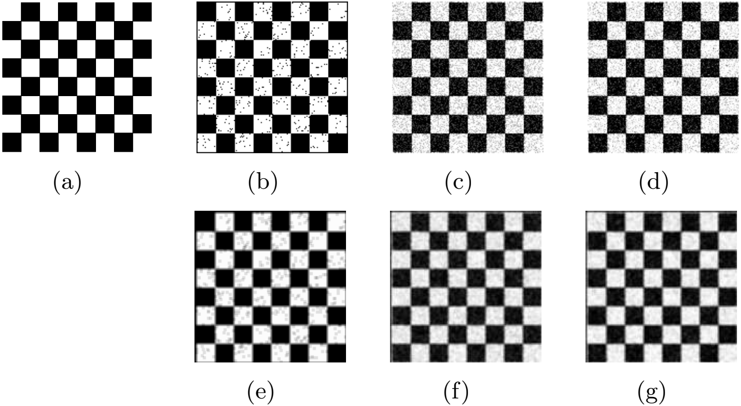
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

That is,

$$f(x,y) * k(x,y) = \frac{1}{w^2} \sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} f(x+i, y+j) \tag{14}$$

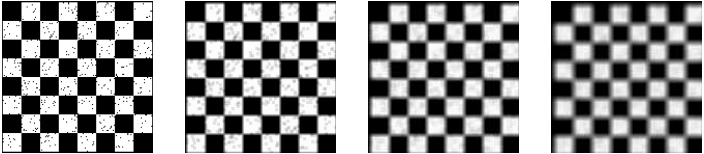
For $w = 3$, we have

$$f(x,y) * k(x,y) = \frac{1}{9} \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x+i, y+j) \tag{15}$$



(a) original image
(b) salt-and-pepper noise, isolated pixels of wrong gray value
(c) uniform noise, noise levels follow a uniform distribution
(d) Gaussian noise, noise levels follow a Gaussian distribution
(e)–(g) filtered by a box filter

Note that removing noise can also blur edges:



(a) (b) (c) (d)

(a) corrupted with salt-and-pepper noise

(b) filtered by 3×3 box filter

(c) filtered by 5×5 box filter

(d) filtered by 7×7 box filter

Why? Because box filtering = averaging neighbour pixels (Eq. 14)!

See the following "zoomed-in view":

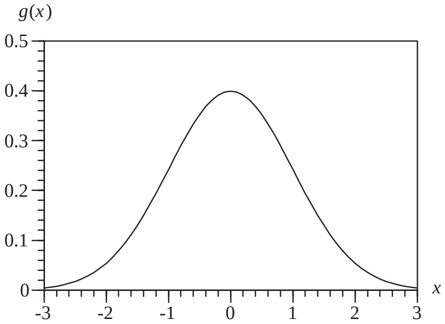
$$\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline\end{array} * \begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&1&1\\\hline 1&1&1\\\hline\end{array} = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 2&3&3&3&2\\\hline 4&6&6&6&4\\\hline 6&9&9&9&6\\\hline 4&6&6&6&4\\\hline\end{array}$$

- Averaging causes a gradual change of pixel values.
- Sharp edge is blurred.
- Filtered values at image boundaries are smaller: boundary effect.
- To reduce edge blurring, use median filter [GW92, SS01] or anisotropic filter [PM90].

Gaussian Filter

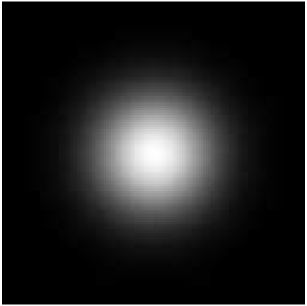
1D (normalized) Gaussian

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{16}$$



2D (normalized) Gaussian

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{17}$$



$\sigma = 15$ pixels, top view around the peak

Like box filters, 2D Gaussian can be used as a smoothing filter:

$$f(x,y) * g(x,y) = \sum_i \sum_j f(x+i, y+j)\, g(i,j) \tag{18}$$

That is, Gaussian filtering is weighted averaging.

2D Gaussian is a separable kernel:

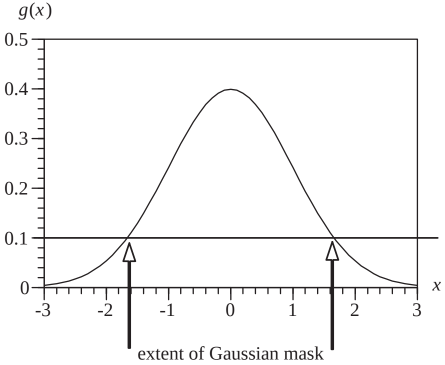
$$f(x,y) * g(x,y) = (f(x,y) * g(x)) * g(y) \tag{19}$$

First convolve f by horizontal 1-D Gaussian $g(x)$.
Then, convolve result by vertical 1-D Gaussian $g(y)$.
This method is more efficient.

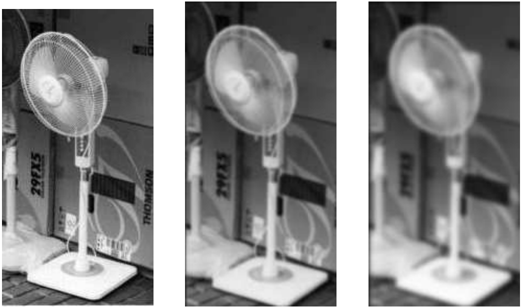
Complexity of original Gaussian smoothing is $O(WHwh)$.
Complexity of efficient Gaussian smoothing is $O(WH(w+h))$.

Notes:

- To use Gaussian, need to discretize the function.
- Size of Gaussian mask must be large enough.
- The larger the Gaussian's σ, the larger is the mask.
- Cut off the mask at a sufficiently small mask value.



Example: Gaussian smoothing.



(a) (b) (c)

(a) original image

(b) filtered by Gaussian with $\sigma = 1$.

(c) filtered by Gaussian with $\sigma = 2$.

Like box filters, Gaussian filters remove noise and blur edges.

Self Study

Edge detection ([SS01] Section 5.6–5.8):

- Difference of Gaussian (DoG): large difference indicates edge.
- Laplacian of Gaussian (LoG): zero-crossing indicates edge.
- Canny edge detector: more immune to noise than LoG.

Exercises

(1) Show that convolution is commutative, i.e., $f(x, y) * k(x, y) = k(x, y) * f(x, y)$.

(2) Show that cross correlation of the form given in Eq. 4 is related to convolution by

$$f(x, y) \circ k(x, y) = f(-x, -y) * k(x, y). \tag{20}$$

(3) Show that cross correlation of the form given in Eq. 7 is related to convolution by

$$f(x, y) \circ k(x, y) = f(x, y) * k(-x, -y). \tag{21}$$

Further Reading

- Convolution and correlation: [GW92] Section 3.3.8, [SS01] Section 5.10.
- Image filtering: [SS01] Chapter 5.
- Median filtering: [SS01] Section 5.5.
- Anisotropic filtering: [PM90].
- Nonlinear filtering (including median filtering, ansiotropic filtering): [Sze10] Section 3.3.1.
- Edge detection: [SS01] Section 5.6–5.8, [Sze10] Section 4.2.

OpenCV supports many image processing functions:

- Convolution
- Image filtering and smoothing
- Edge detection, corner detection

Reference

R. C. Gonzalez and R. E. Woods.
Digital Image Processing.
Addison-Wesley, 1992.

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

L. Shapiro and Stockman.
Computer Vision.
Prentice-Hall, 2001.

R. Szeliski.
Computer Vision: Algorithms and Applications.
Springer, 2010.