Outline

1. Basics of Image Processing

2. Convolution & Cross Correlation

3. Applications
   - Box Filter
   - 1D Gaussian Filter
   - 2D Gaussian Filter

4. Self Study

5. Exercises

6. Further Reading
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**.
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

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 3 & 2 & 0 \\
0 & 1 & 3 & 1 & 0 \\
0 & 1 & 1 & 0 & 0
\end{array}
\]

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

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

In discrete form,

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

where \( W \) and \( H \) are the width and height of the image.

Convolution is commutative (Exercise):

\[
f(x, y) \ast k(x, y) = k(x, y) \ast f(x, y). \tag{3}
\]
1D Example

\[ f(u) \]

\[ k(u) \]

\[ k(x-u) \]

\[ f(u) k(x-u) \]

\[ f(u) k(x-u) \]

\[ f(x) \ast g(x) \]
Cross Correlation

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

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

(4)

In discrete form,

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

(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):

\[
f(x, y) \circ k(x, y) = f(-x, -y) * k(x, y).
\]  

(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:

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

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,

\[ f(x, y) * k(x, y) = k(x, y) * f(x, y) \]  \hspace{1cm} (8)

\[ = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(u, v) f(x - u, y - v) \, du \, dv \] \hspace{1cm} (9)

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

\[ f(x, y) * k(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x + \mu, y + \nu) k(-\mu, -\nu) \, d\mu \, d\nu \] \hspace{1cm} (10)

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

\[ f(x, y) * k(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x + \mu, y + \nu) k(\mu, \nu) \, d\mu \, d\nu \] \hspace{1cm} (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:

$$f(x, y) \ast k(x, y) = \sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} f(x + i, y + j)$$  \hspace{1cm} (12)

Usually, we normalize the mask values so that

$$\sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} k(i, j) = 1$$  \hspace{1cm} (13)
$3 \times 3$ normalized box filter:

\[
\begin{array}{ccc}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{array}
\]

That is,

\[
f(x, y) \ast k(x, y) = \frac{1}{w^2} \sum_{i=-w/2}^{w/2} \sum_{j=-w/2}^{w/2} f(x+i, y+j)
\]  \hspace{1cm} (14)

For $w = 3$, we have

\[
f(x, y) \ast k(x, y) = \frac{1}{9} \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x+i, y+j)
\]  \hspace{1cm} (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) corrupted with salt-and-pepper noise
(b) filtered by $3 \times 3$ box filter
(c) filtered by $5 \times 5$ box filter
(d) filtered by $7 \times 7$ box filter
Applications: Box Filter

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

See the following “zoomed-in view”:

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\] \times \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array} = \begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
2 & 3 & 3 & 3 & 3 & 2 \\
4 & 6 & 6 & 6 & 6 & 4 \\
6 & 9 & 9 & 9 & 9 & 6 \\
4 & 6 & 6 & 6 & 6 & 4 \\
\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].
1D (normalized) Gaussian

\[ g(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \] (16)
2D (normalized) Gaussian

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

(17)

\[\sigma = 15\] pixels, top view around the peak
Like box filters, 2D Gaussian can be used as a smoothing filter:

$$f(x, y) * g(x, y) = \sum_i \sum_j f(x + i, y + j) g(i, j)$$  \hspace{1cm} (18)

That is, Gaussian filtering is weighted averaging.

2D Gaussian is a separable kernel:

$$f(x, y) * g(x, y) = (f(x, y) * g(x)) * g(y)$$  \hspace{1cm} (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 $\sigma$, the larger is the mask.
- Cut off the mask at a sufficiently small mask value.
Example: Gaussian smoothing.

(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.
(1) Show that convolution is commutative, i.e.,
\[ f(x, y) \ast k(x, y) = k(x, y) \ast f(x, y). \]

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

\[ f(x, y) \circ k(x, y) = f(-x, -y) \ast k(x, y). \]  \hspace{1cm} (20)

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

\[ f(x, y) \circ k(x, y) = f(x, y) \ast k(-x, -y). \]  \hspace{1cm} (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, anisotropic 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,*  

L. Shapiro and Stockman.  
*Computer Vision.*  

R. Szeliski.  
*Computer Vision: Algorithms and Applications.*  