


浙江大学爱丁堡大学联合学院

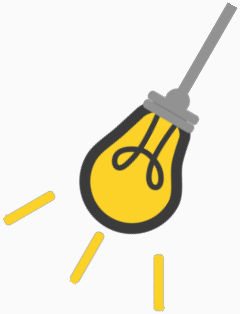
ZJU-UoE Institute

Filters

Nicola Romanò - nicola.romano@ed.ac.uk

Learning objectives

- Define convolutional filters
- Explain their use in image analysis
- Implement basic filters in Python



Types of pixel operations

Operations for manipulating pixel intensities

Two types of operations:

- Point operations - Change pixel intensity based only on its value - $I'_{(x,y)} = f(I_{x,y})$ (see Lecture 3)
- **Neighbourhood operations** - Change pixel intensity based on the intensity of the pixel and its neighbours.

Filters

Neighbourhood operations, often called **filters** allow to modify an image in a way that is not possible with point operations.

- Detect simple structures such as edges, corners, lines, etc.
- Perform operations such as smoothing, sharpening, etc.
- Noise reduction

Filters

Neighbourhood operations, often called **filters** allow to modify an image in a way that is not possible with point operations.

- Detect simple structures such as edges, corners, lines, etc.
- Perform operations such as smoothing, sharpening, etc.
- Noise reduction

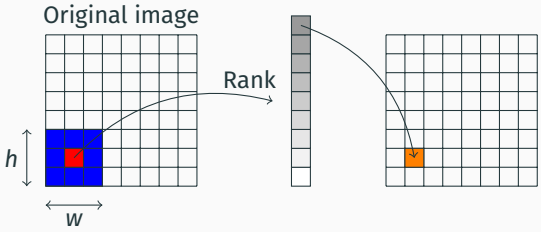
Today we will look at:

- **Rank filters** - the new pixel value is a function of the rank of the pixel values of the neighbourhood
- **Convolutional filters** - the new pixel value is a weighted sum of the pixel values of the neighbourhood

Rank filters

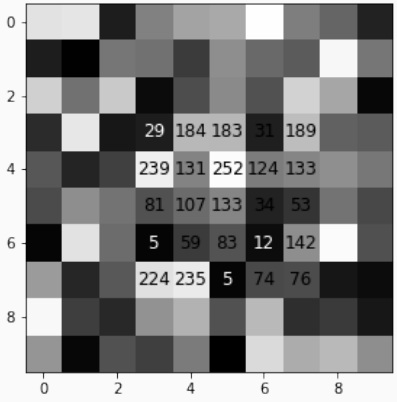
Convolutional filters

Rank filters



- We decide on a window size $w \times h$ (other non-rectangular shapes are possible)
- We traverse each pixel in the image and take its $w \times h$ neighbourhood
- We rank the intensity of each pixel in the neighbourhood
- We take a specific value (e.g. minimum, maximum, median) and set the output value to this value

Example - median filter



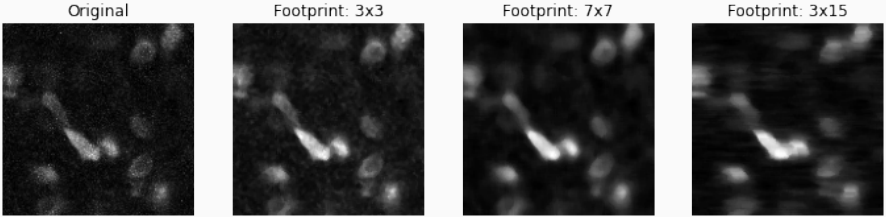
Rank filters in Scikit Image

Rank filters are implemented in Scikit Image in the ‘skimage.rank‘ module.
These filters require a *footprint* of the pixel neighbourhood, as a matrix of 0 and 1.

For example

```
from skimage.rank import median

# 3x3 neighbourhood
footprint = np.ones(3, 3)
img_median = median(img, selem=footprint)
```



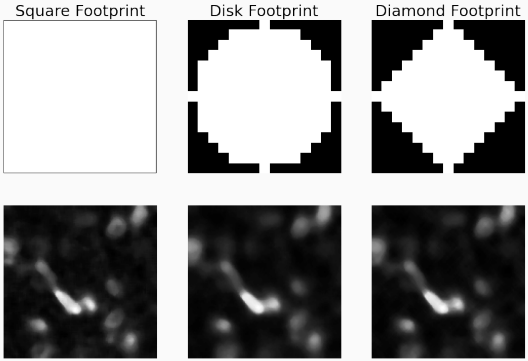
Non-rectangular footprints

The footprint can be a *non-rectangular* matrix. For example, you can generate a **circular** shape using the `skimage.morphology.disk` function or a **diamond** shape using the `skimage.morphology.diamond` function.

```
from skimage.morphology import disk,
diamond

dsk = disk(7) # A disk, radius 7
dia = diamond(7) # A diamond, radius 7

fig, ax = plt.subplots(1, 2)
ax[0].imshow(dsk, cmap="gray")
ax[1].imshow(dia, cmap="gray")
```



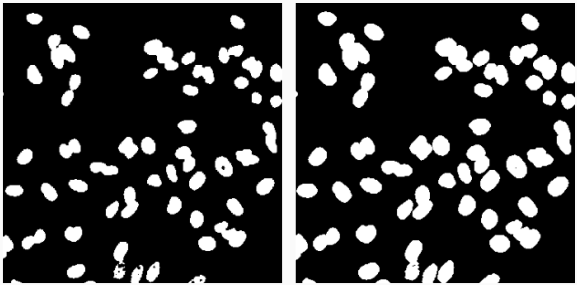
Rank filters - use cases

Rank filters are very simple, but have useful applications.
The **median filter** is used to remove noise and to smooth images.

Rank filters - use cases

Rank filters are very simple, but have useful applications.
The **median filter** is used to remove noise and to smooth images.

The **maximum filter** can be used in binary images to remove small "holes". It is also a very common filter used in modern neural networks for image analysis.

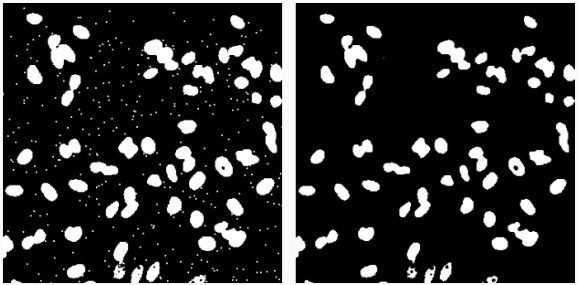


Rank filters - use cases

Rank filters are very simple, but have useful applications.
The **median filter** is used to remove noise and to smooth images.

The **maximum filter** can be used in binary images to remove small "holes". It is also a very common filter used in modern neural networks for image analysis.

The **minimum filter** can be used to remove small bright spots.

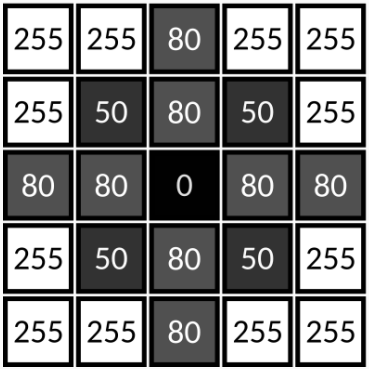


Convolutional filters

A **convolutional filter** consists of a small matrix, called a **kernel**, that is used to process an image. Convolution takes each pixel of the image, together with its neighbours, and adds them together, weighting each neighbour by the value of a kernel of the same size of the neighbourhood.

Example of a convolutional filter

Image

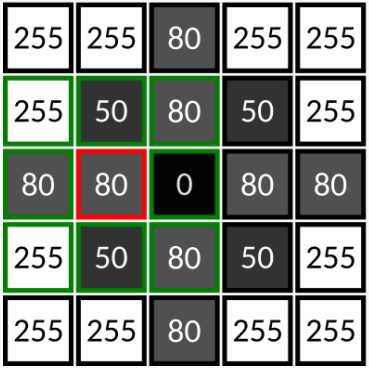


Kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Example of a convolutional filter

Image



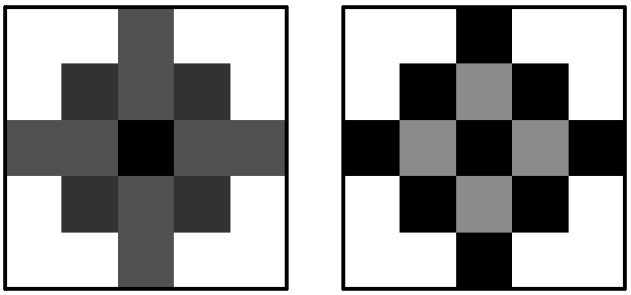
Kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The convolved pixel value will be

$$255 * 0 + 50 * (-1) + 80 * 0 + 80 * (-1) + 80 * 5 + 0 * (-1) + 255 * 0 + 50 * (-1) + 80 * 0 = \mathbf{220}$$

Example of a convolutional filter - result



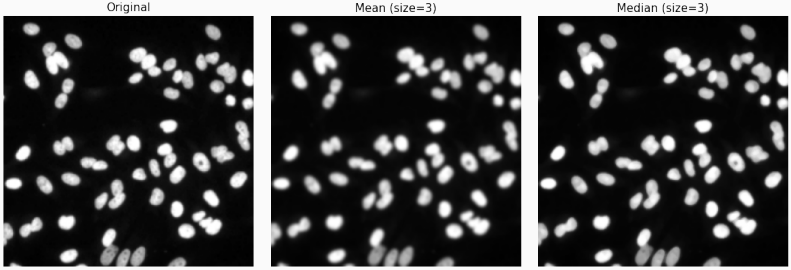
Our image after applying the convolutional filter

Common convolutional filters - averaginig filter

The **averaging filter** is a simple filter that is used to reduce noise in an image.
It simply takes the average of the pixel values in the neighbourhood.

Example 3x3 kernel:

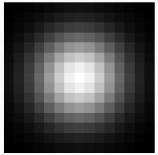
$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

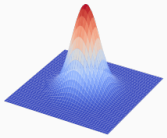


Not as good as the **median filter**, as it is sensitive to outliers and does not preserve edges as well (can you get an intuition as to why?).

Common convolutional filters - Gaussian filter

The **Gaussian filter** is a filter that is used to smooth images.
It uses a **Gaussian** function to weight the pixel values in the neighbourhood.





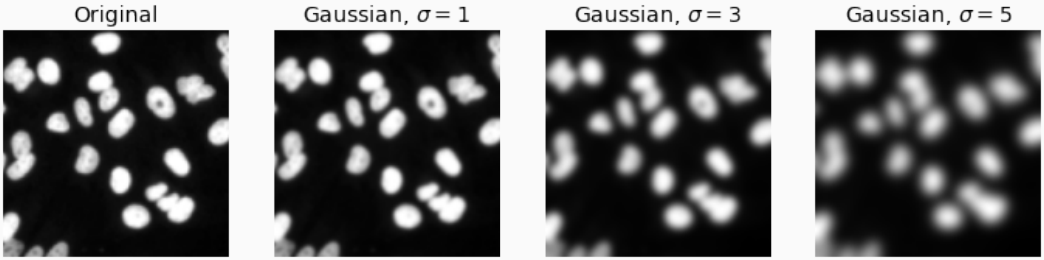
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Example of gaussian kernels 3x3 and 5x5 (approx.):

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Common convolutional filters - Gaussian filter - result

Example of gaussian filters with increasing sigma values:

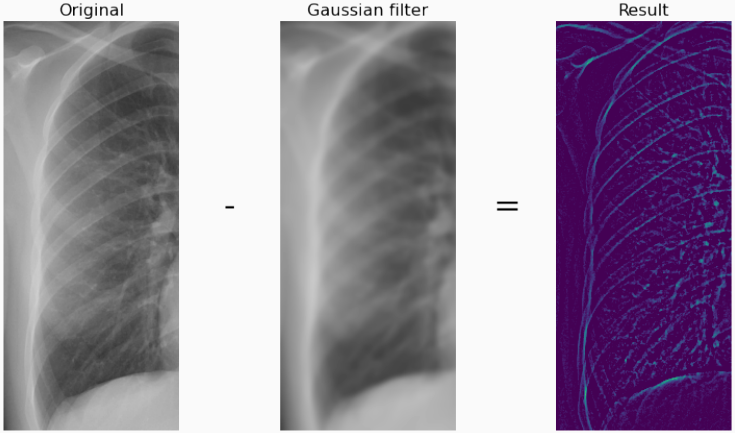


Gaussian filters result in a blurring effect, which can be useful for removing noise, or in general removing the finer detail of the image (low-pass filtering).

```
from skimage.filters import gaussian
img_blurred = gaussian(image, sigma=1)
```

Sharpening filters

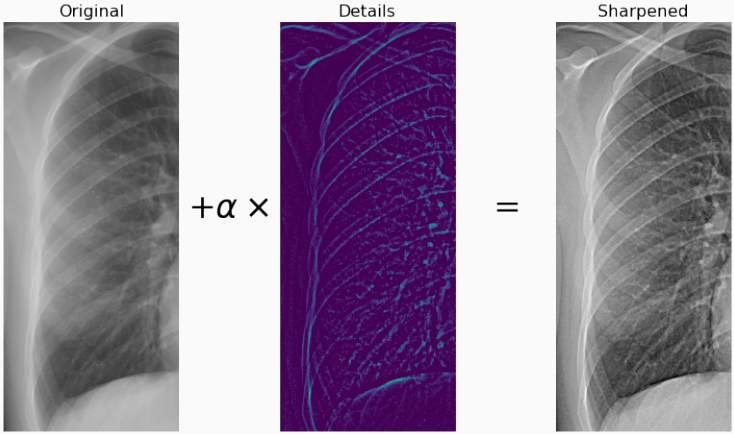
Sharpening filters are used to increase the detail of an image.



If we apply a Gaussian filter to the image we will remove detail, so if we subtract the result from the original image we will get the "details" of the image.

Sharpening filters

Sharpening filters are used to increase the detail of an image.



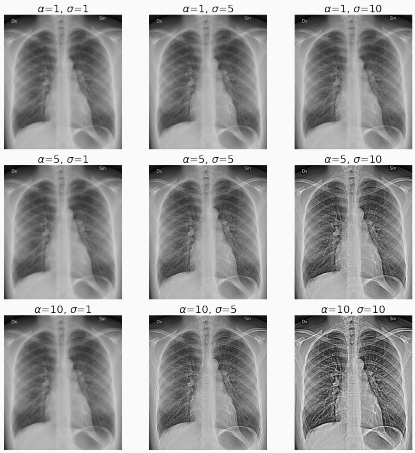
We can now add the details to the original image, thus sharpening it!

This process is called **"unsharp masking"**.

Unsharp masking in Scikit Image

You can use the `skimage.filters.unsharp_mask` function to apply unsharp masking to images.

```
from skimage.filters import unsharp_mask
img_sharpened = unsharp_mask(img, radius=15, amount=2)
```



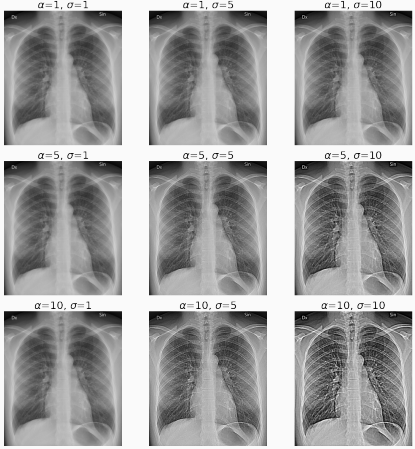
Exploring the parameter space of unsharp masking.

Unsharp masking in Scikit Image

You can use the `skimage.filters.unsharp_mask` function to apply unsharp masking to images.

```
from skimage.filters import unsharp_mask
img_sharpened = unsharp_mask(img, radius=15, amount=2)
```

Exercise: try writing your own unsharp masking function.
It should accept an image, a radius (the σ of the gaussian blur) and an amount for sharpening and return the sharpened image.



Exploring the parameter space of unsharp masking.