Image processing
- Linear filtering

Objectives

- What is image filtering
- The difference between linear and nonlinear filters
- Correlation and Convolution
- Learn how linear filters work and their implementation
- Learn how nonlinear filters work and their implementation

What is image filter

- Filtering is a technique for modifying or enhancing an image.
- Filters are just systems that form a new, and preferably enhanced, image from a combination of the original image's pixel values.
- Filtering is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel.

De-noising

Salt and pepper noise

Super-resolution

Linear and Nonlinear filters

- Linear filtering is filtering in which the value of an output pixel is a linear combination of the values of the pixels in the input pixel's neighborhood.

$$f(x, y) = \sum_{m,n} I\,(m, n) \cdot w(x - m, y - n)$$

  - I is represents the original image, f is represents the filtered pixels value, and w is represents the filter coefficients

- The general idea in non-linear image filtering is that instead of using the spatial mask in a convolution process, the mask is used to obtain the neighboring pixel values, and then ordering mechanisms produce the output pixel.

Linear and Nonlinear filters

- Noisy Image-
- Salt and pepper noise

- ***Linear filtering*** - Mean filter
- ***Nonliner filter-*** *Median filter*

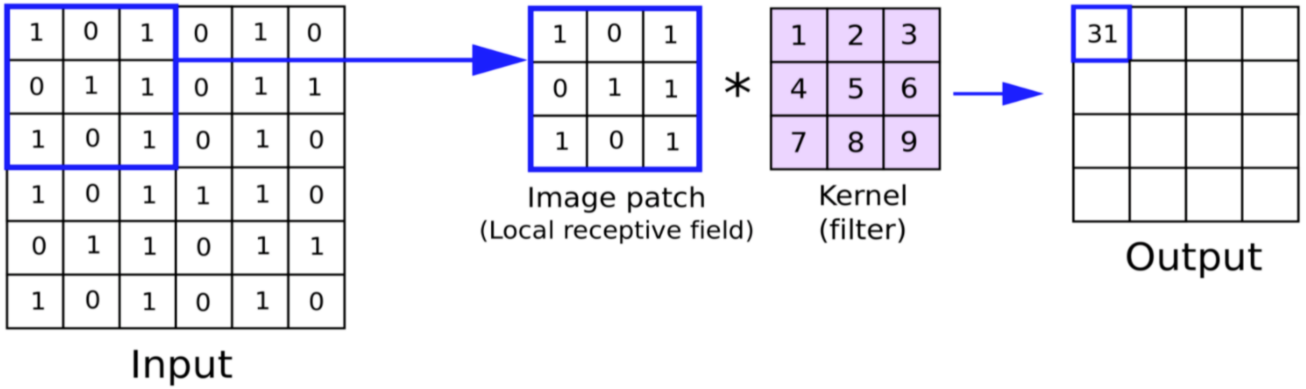
Convolution

- Linear filtering of an image is accomplished through an operation called convolution.
- In convolution, the value of an output pixel is computed as a weighted sum of neighboring pixels. The matrix of weights is called the convolution kernel, also known as the filter.
- Convolution uses the kernel to highlight a particular feature of an image.

$$(f * g)(t) \stackrel{\mathrm{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Convolution Steps

- You place it over the input image beginning from the top-left corner.
- Multiply each weight in the rotated convolution kernel by the pixel.
- Sum up the individual products from step 2.
- You then move the feature detector one cell to the right and do the same thing.
- After you have gone through the whole first row, you can then move it over to the next row and go through the same process.



Convolution- Kernel

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Correlation

- Correlation is a local operator and is a simple local image operator.

- A correlation takes an image F, a weight function W and it results in a new image G.

- The weight function W is often defined on a small subset of the sample points of F.

$$G(i,j) = (F \star_c W)[i,j] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} F[i+k, j+l]W[k,l]$$

Correlation

- In correlation, the value of an output pixel is also computed as a weighted sum of neighboring pixels.

- A correlation takes an image F, a weight function W and it results in a new image G.

- The weight function W is often defined on a small subset of the sample points of F.

$$G(i,j) = (F \star_c W)[i,j] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} F[i+k, j+l]W[k,l]$$

Correlation- Convolution

- What are the differences between correlation and convolution?
- Correlation is measurement of the similarity between two image.
- Convolution is measurement of effect of one image on the other image.
- The mathematical calculation of Correlation is same as convolution in time domain, except that the image is not reversed, before the multiplication process. If the filter is symmetric then the output of both the expression would be same.

Mean Filter

- It is often used to reduce noise in images.
- The idea of mean filtering is simply to replace each pixel value in an image with the mean (“average”) value of its neighbors, including itself.

MeanFilter

- Keeping border values unchanged

Average = round(1+4+0+2+2+4+1+0+1)/9 = 2

Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	2	2	2	1	3
1	2	1	1	1	0
1	2	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Mean Filter

- Extending border values outside with values at boundary

Input

Average = round(1+4+0+1+4+0+2+2+4)/9 = 2

1	1	4	0	1	3	1	1
1	1	4	0	1	3	1	1
2	2	2	4	2	2	3	3
1	1	0	1	0	1	0	0
1	1	2	1	0	2	2	2
2	2	5	3	1	2	5	5
1	1	1	4	2	3	0	0
1	1	1	4	2	3	0	0

Output

2	2	2	2	2	2
2	2	2	2	1	2
1	2	1	1	1	2
2	2	1	1	1	2
2	2	2	2	2	2
2	2	3	3	2	2

Mean Filter

- Extending border values outside with 0s (Zero-padding)

Input

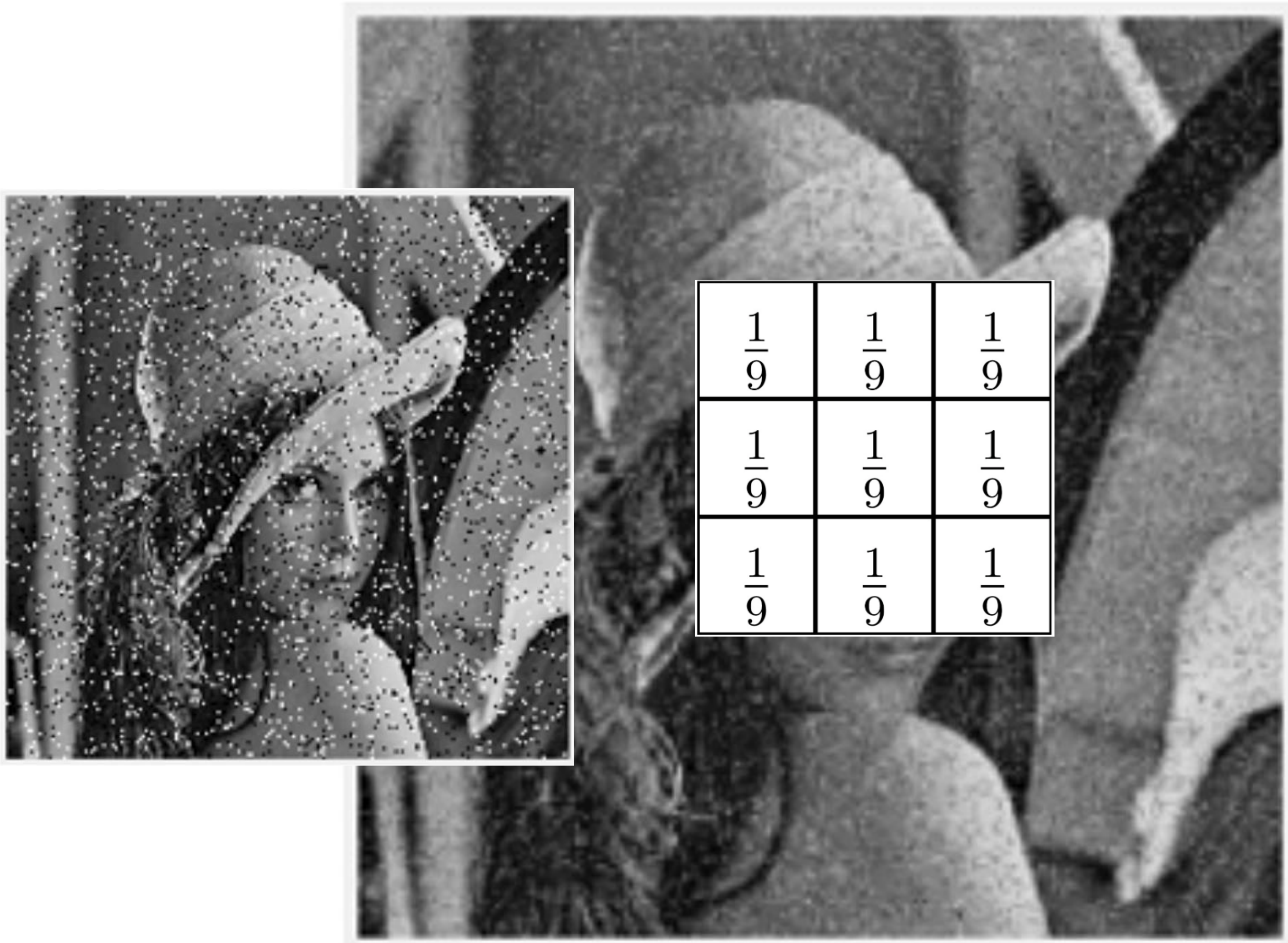
Average = round(2+5+0+3+0+0+0+0+0)/9 = 1

0	0	0	0	0	0	0	0
0	1	4	0	1	3	1	0
0	2	2	4	2	2	3	0
0	1	0	1	0	1	0	0
0	1	2	1	0	2	2	0
0	2	5	3	1	2	5	0
0	1	1	4	2	3	0	0
0	0	0	0	0	0	0	0

Output

1	1	1	1	1	1
1	2	2	2	1	1
1	2	1	1	1	1
1	2	1	1	1	1
1	2	2	2	2	2
1	2	2	2	1	1

Mean Filter



$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Median Filter

- Median filtering is a nonlinear method used to remove noise from images.
- It is widely used as it is very effective at removing noise while preserving edges.
- It is particularly effective at removing ‘salt and pepper’ type noise.
- The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels.
- The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.

Median Filter

- Keeping border values unchanged

Sorted: 0,0,1,1,1,2,2,4,4

Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Median Filter

- Extending border values outside with values at boundary

Input

1	1	4	0	1	3	1	1
1	1	4	0	1	3	1	1
2	2	2	4	2	2	3	3
1	1	0	1	0	1	0	0
1	1	2	1	0	2	2	2
2	2	5	3	1	2	5	5
1	1	1	4	2	3	0	0
1	1	1	4	2	3	0	0

Sorted: 0,0,1,1,1,2,2,4,4

Output

2	2	2	2	2	2
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	1	1	2
1	2	2	2	2	2
1	2	2	3	2	2

Median Filter

- Extending border values outside with 0s

Input

0	0	0	0	0	0	0	0
0	1	4	0	1	3	1	0
0	2	2	4	2	2	3	0
0	1	0	1	0	1	0	0
0	1	2	1	0	2	2	0
0	2	5	3	1	2	5	0
0	1	1	4	2	3	0	0
0	0	0	0	0	0	0	0

Sorted: 0,0,1,1,1,2,2,4,4

Output

0	2	1	1	1	0
0	1	1	1	1	1
0	1	1	1	2	1
0	1	1	1	1	1
1	2	2	2	2	2
0	1	1	1	1	0

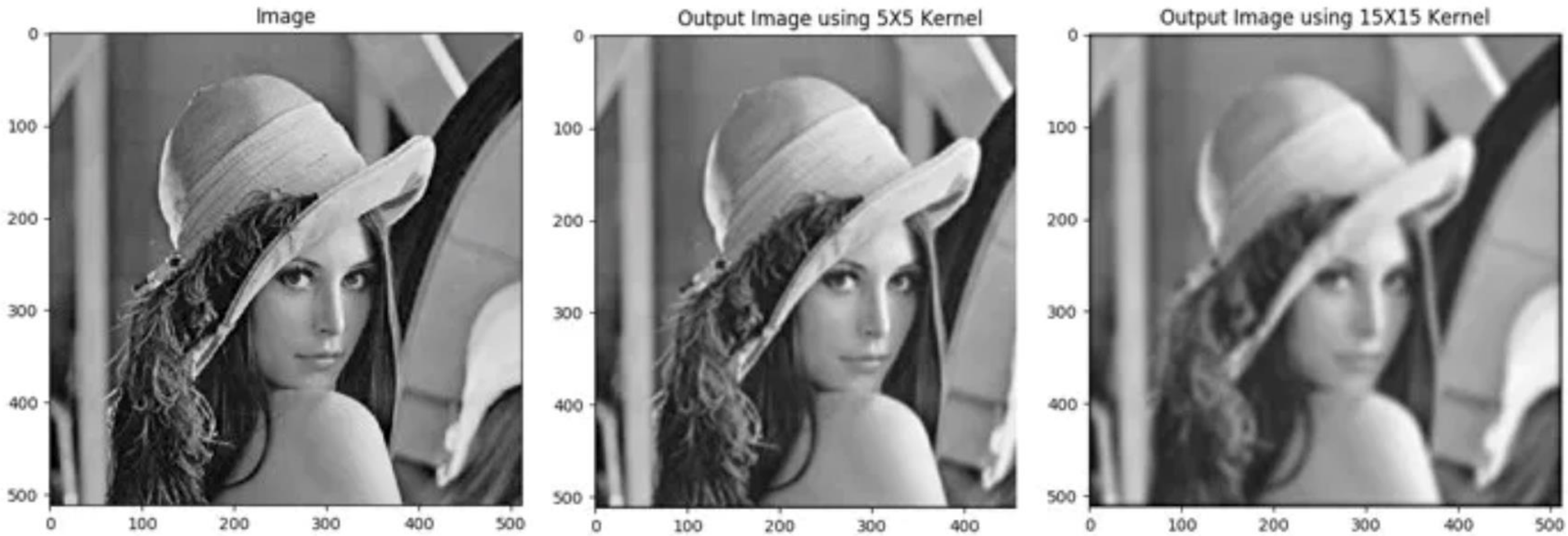
Median Filter

Gaussian Smoothing

- Gaussian smoothing is the result of blurring an image by a Gaussian function.
- Enhance image structures at different scales.
- Smooth blur resembling that of viewing the image through a translucent screen
- A smoothed function is the convolution of the original function f with the Gaussian weight function G

$$G^s(x, y) = \frac{1}{2\pi s^2} \exp\left(-\frac{x^2 + y^2}{2s^2} \right)$$

Gaussian Smoothing



Bilateral filtering

- The bilateral filter adds a tonal weight such that pixel values that are close to the pixel value in the center are weighted more than pixel values that are more different

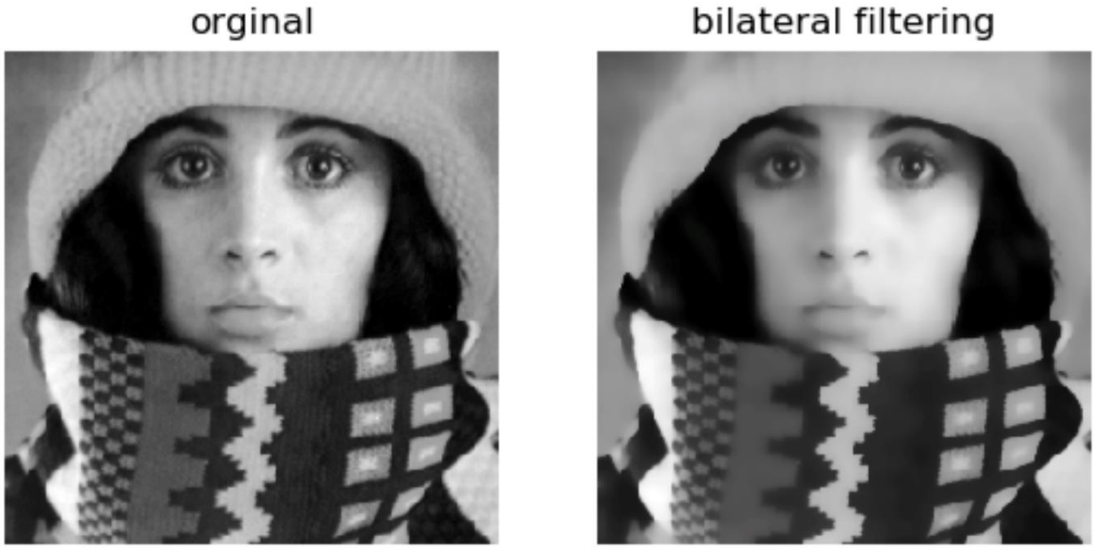
- This tonal weighting makes that the bilateral filter is capable of preserving edges (large differences in tonal value) while smoothing in the more flat regions (small tonal differences).

$$g(\mathbf{x}) = (f * G^s)(\mathbf{x}) = \int_{\mathcal{R}} f(\mathbf{y})\, G^s(\mathbf{x}-\mathbf{y})\, d\mathbf{y}$$

- Th ussian sr

Bilateral filtering

```
def bilateral( f, s, t ):
  def tonaldistsq(fp, fq):
    pass
  def spatialdistsq(p, q):
    pass
  g = empty( f.shape )
  for p in domainIterator(f.shape[:2]):
    t = 0; n = 0
    for q in nbhIterator(f.shape, start, end, p):
      w = exp( -spatialdistsq(p,q)/(2*s**2) ) * exp( -tonaldistsq(f[p],f[q])/(2*t**2) )
      t += f[q] * w
      n += w
    g[p]=t/n
```



Summary

- What is image filtering
- The difference between linear and nonlinear filters
- Correlation and Convolution
- Learn how linear filters work and their implementation
- Learn how nonlinear filters work and their implementation