EN2550 Fundamentals of Image Processing and Machine Vision: Spatial Filtering

Ranga Rodrigo
ranga@uom.lk

The University of Moratuwa, Sri Lanka

February 8, 2022

Contents

Introduction

- We use spatial filtering to enhance images, e.g., noise filtering. We can use the same technique to locate objects in images, called template matching. After completing this lesson, we would be able to process images using filters, as done in popular image processing software such as Photoshop.
- Intensity transformations affect the brightness (intensity level)of an image. The resulting value of a pixel after an intensity operation is just dependent on the original value of the same pixel.
- In contrast, the resulting value of a pixel after a spatial filtering operation depends on the neighborhood of the pixel in question. So the space around the pixel in question matters, hence, the name spatial filtering.
- In linear spatial filtering we use the 2-D convolution operation.

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20					
2							
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30				
2							
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40			
2							
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40		
2							
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2							
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40					
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60				
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90			
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90		
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3							
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40					
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60				
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90			
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90		
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4							
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4		20					
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4		20	30				
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4		20	30	50			
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4		20	30	50	50		
5							

Figure: Convolution

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

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4		20	30	50	50	40	
5							

Figure: Convolution

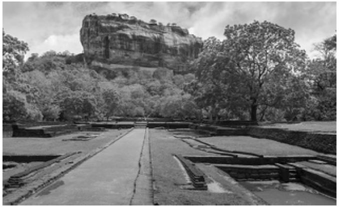
$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	180	90	90	180	90
3	0	0	180	90	180	180	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

	0	1	2	3	4	5	6
0							
1		20	30	40	40	40	
2		40	60	90	90	80	
3		40	60	90	90	80	
4		20	30	50	50	40	
5							

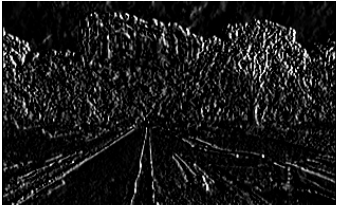
Figure: Convolution

Examples of Effect of Kernel Choices



(a) Original

(b) Avegaging



(c) Sobel horizontal

(d) Sobel vertical

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

Averaging

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

Sobel vertical

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

Sobel horizontal

Figure: 3×3 kernels

Spatial Filtering (Averaging) Using filter2D

Listing: Spatial Filtering (Averaging)

```
%matplotlib inline
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('../images/sigiriya.jpg', cv.IMREAD_REDUCED_GRAYSCALE_2)

kernel = np.ones((3,3),np.float32)/9
imgc = cv.filter2D(img,-1,kernel)

fig, axes  = plt.subplots(1,2, sharex='all', sharey='all', figsize=(18,18))
axes[0].imshow(img, cmap='gray')
axes[0].set_title('Original')
axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(imgc, cmap='gray')
axes[1].set_title('Averaging')
axes[1].set_xticks([]), axes[1].set_yticks([])
plt.show()
```

Spatial Filtering (Sobel Vertical) Using filter2D

Listing: Spatial Filtering (Sobel Vertical)

```
%matplotlib inline
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('../images/sigiriya.jpg', cv.IMREAD_REDUCED_GRAYSCALE_2)

kernel = np.array([(-1, -2, -1), (0, 0, 0), (1, 2, 1)], dtype='float')
imgc = cv.filter2D(img,-1,kernel)

fig, axes = plt.subplots(1,2, sharex='all', sharey='all', figsize=(18,18))
axes[0].imshow(img, cmap='gray')
axes[0].set_title('Original')
axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(imgc, cmap='gray')
axes[1].set_title('Averaging')
axes[1].set_xticks([]), axes[1].set_yticks([])
plt.show()
```

Convolution and Correlation

1. Spatial filtering is, in fact, convolution.
2. As the filters are typically symmetric—i.e., a 180° rotation results in the same kernel—correlation is equivalent to convolution.
3. Correlation is also the scalar product between the kernel and the underlying image patch. Therefore, it is a measure of similarity between the kernel and the underlying image patch.
4. As a result, when the kernel and the patch are "similar", the output is high. In view of this, spatial filtering seeks for patches in the image that are similar to the kernel.
5. Implementing filtering using loops (four nested for loops) in a non-C fashion is inefficient. Instead, use `filter2D`.

Convolution and Correlation

The convolution sum expression that we leaned in signal and systems is

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k].$$

In 2-D, as applicable in image processing, collation sum with a kernel $w[m,n]$ with non-zero values in $(m,n) \in ([-a,a],[-b,b])$ is

$$(w * f)[m,n] = w[m,n] * f[m,n] = \sum_{s=-a}^{a} \sum_{k=-b}^{b} w[s,t]f[m-s,n-t].$$

Correlation:

$$(w \circledast f)[m,n] = w[m,n] \circledast f[m,n] = \sum_{s=-a}^{a} \sum_{k=-b}^{b} w[s,t]f[m+s,n+t].$$

Listing: Filtering Using Loops

```
%matplotlib inline
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

def filter(image, kernel):
    assert kernel.shape[0]%2 == 1 and kernel.shape[1]%2 == 1
    k_hh, k_hw = math.floor(kernel.shape[0]/2), math.floor(kernel.shape[1]/2)
    h, w = image.shape
    image_float = cv.normalize(image.astype('float'), None, 0.0, 1.0, cv.NORM_MINMAX)
    result = np.zeros(image.shape, 'float')

    for m in range(k_hh, h - k_hh):
        for n in range(k_hw, w - k_hw):
            result[i,j] = np.dot(image_float[m-k_hh:m + k_hh + 1, n - k_hw : n + k_hw + 1].flatten(), kernel.flatten())
    return result

img = cv.imread('../images/keira.jpg', cv.IMREAD_REDUCED_GRAYSCALE_8)
```

```
f, axarr = plt.subplots(1,2)
axarr[0].imshow(img, cmap="gray")
axarr[0].set_title('Original')
kernel = np.array([(1/9, 1/9, 1/9), (1/9, 1/9, 1/9), (1/9, 1/9, 1/9)], dtype='float')
imgb = filter(img, kernel)
imgb = imgb*255.0
imgb = imgb.astype(np.uint8)

axarr[1].imshow(imgb, cmap="gray")
axarr[1].set_title('Filtered')
```

Example

Consider the image

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and the filtering kernel

$$w = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

. Appropriately pad the image. Carry out a. correlation b. convolution.

Solution

Consider the image

$$f_{\text{padded}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad w = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Correlation result and convolution result, respectively

$$(w \circledast f) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 8 & 7 & 0 \\ 0 & 6 & 5 & 4 & 0 \\ 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (w * f) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

.

Convolution: Key Properties

1. Linearity: $\mathrm{filter}(f_1 + f_2) = \mathrm{filter}(f_1) + \mathrm{filter}(f_2)$
2. Shift invariance: same behavior regardless of pixel location: $\mathrm{filter}(\mathrm{shift}(f)) = \mathrm{shift}(\mathrm{filter}(f))$
3. Theoretical result: any linear shift-invariant operator can be represented as a convolution.

Other Properties

1. Commutative: $a * b = b * a$
2. Conceptually no difference between filter and signal
3. Associative: $a * (b * c) = (a * b) * c$
4. Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
5. This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
6. Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
7. Scalars factor out: $ka * b = a * kb = k(a * b)$
8. Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Source: *Svetlana Lazebnik*

At the Edge

1. The filter window falls off the edge of the image.
2. Need to extrapolate the image.
3. Methods: various border types, image boundaries are denoted with '|'

BORDER_REPLICATE:	`aaaaaa|abcdefgh|hhhhhhh`	
BORDER_REFLECT:	`fedcba|abcdefgh|hgfedcb`	
BORDER_REFLECT_101:	`gfedcb|abcdefgh|gfedcba`	
BORDER_WRAP:	`cdefgh|abcdefgh|abcdefg`	
BORDER_CONSTANT:	`iiiiii|abcdefgh|iiiiiii`	with some specified 'i'

Sharpening

(a) Original

−

(b) Smoothed

=

(c) Original − Smoothed

(d) Original

−

(e) Original − Smoothed

=

(f) Sharpened

Figure: Sharpening (125 added to Original − Smoothed to display)

Source: *Svetlana Lazebnik*

Box Filter vs. Gaussian Filter

- What's wrong with this filtering operation?
- What's the solution?

(a) Original

(b) Kernel (much zoomed)

(c) Box Filtered

Figure: Smoothing with Box Filter

Source: *D. Forsyth*

Box Filter vs. Gaussian Filter

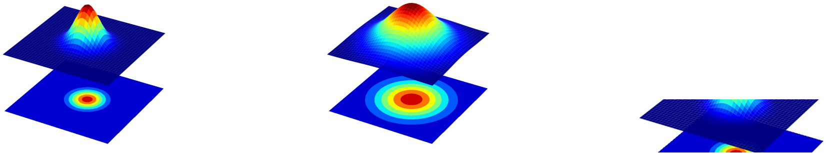
(a) Original (b) Box Filtered (c) Gaussian Filtered

Figure: Smoothing with Box and Gaussian Filter

Source: *D. Forsyth*

Gaussian Kernel

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



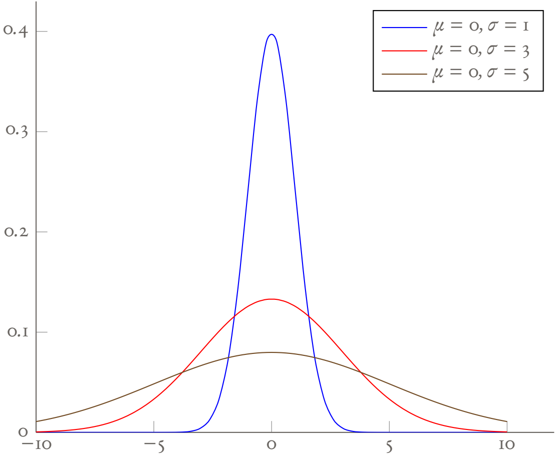
(a) $\sigma = 1$ (b) $\sigma = 2$ (c) $\sigma = 5$

Figure: 2-D Gaussians

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case).
- The Gaussian function has infinite support, but discrete filters use finite kernels.

Choosing Gaussian Kernel Width

Rule of thumb: set the filter half-width to about 3σ.



Separability of the Gaussian Filter

$$
\begin{aligned}
G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \\
&= \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right] \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right].
\end{aligned}
$$

- The 2-D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y.
- In this case, the two Gaussians are the (identical) 1-D Gaussians.

Separability of the Gaussian Filter

$$
\begin{aligned}
G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \\
&= \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \right] \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \right].
\end{aligned}
$$

- The 2-D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y.
- In this case, the two Gaussians are the (identical) 1-D Gaussians.
- Separability means that a 2-D convolution can be reduced to two 1-D convolutions (one among rows and one among columns).
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?

Separability of the Gaussian Filter

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \\
&= \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \right] \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \right].
\end{aligned}
$$

- The 2-D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y.
- In this case, the two Gaussians are the (identical) 1-D Gaussians.
- Separability means that a 2-D convolution can be reduced to two 1-D convolutions (one among rows and one among columns).
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel? $O(n^2 m^2)$.
- What is the complexity if the kernel is separable?

Separability of the Gaussian Filter

$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \\
&= \left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right]\left[\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right].
\end{aligned}
$$

- The 2-D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y.
- In this case, the two Gaussians are the (identical) 1-D Gaussians.
- Separability means that a 2-D convolution can be reduced to two 1-D convolutions (one among rows and one among columns).
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel? $O(n^2 m^2)$.
- What is the complexity if the kernel is separable? $O(n^2 m)$.

Source: *D. Forsyth*

Noise

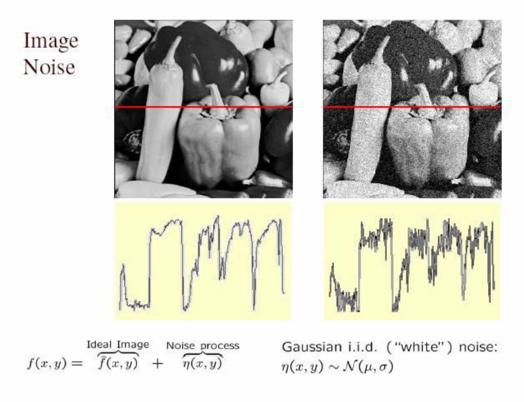
- Salt and pepper noise: contains random occurrences of black and white pixels
- Impulse noise: contains random occurrences of white pixels
- Gaussian noise: variations in intensity drawn from a Gaussian normal distribution

Figure

Source: *S. Sietz*

Gaussian Noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise



Figure

Source: *M. Hebert*

Reducing Gaussian Noise

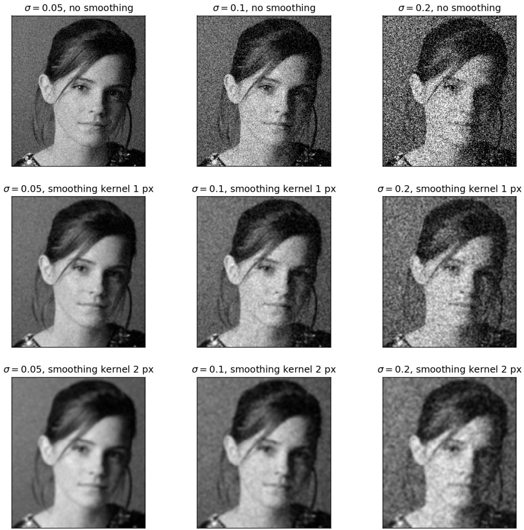


Figure: Reducing Gaussian noise: noise σ and smoothing kernel size

Smoothing with larger standard deviations suppresses noise, but also blurs the image.

Reducing Salt-and-Pepper Noise

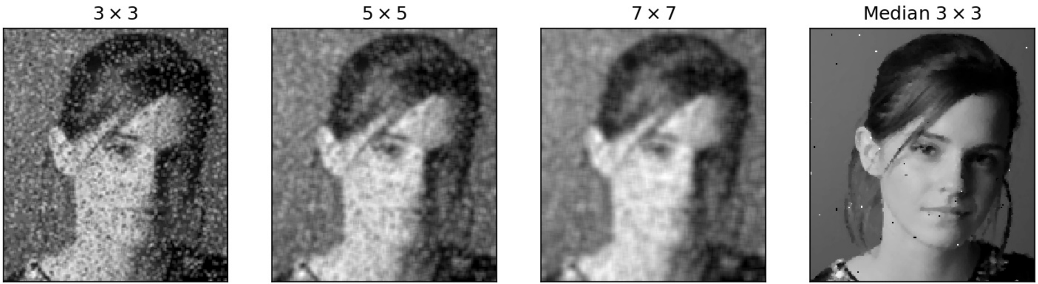
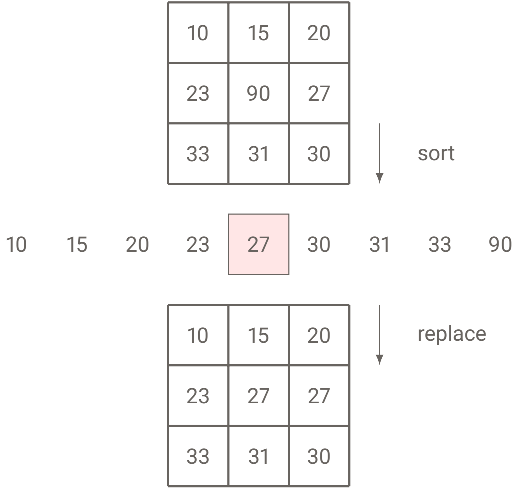


Figure: Inability to reduce salt and pepper noise with Gaussian filtering

Median Filtering



Figure

A median filter operates over a window by selecting the median intensity in the window.
Is median filtering linear?

Source: *K. Grauman*

Sharpening Revisited

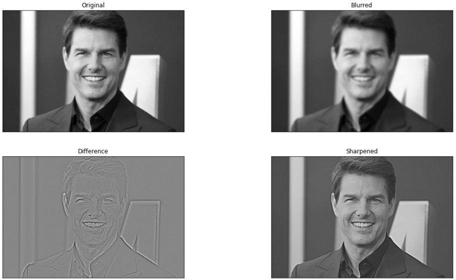


Figure: Sharpening

1. What does blurring take away?
2. We add it back.

Source: *Svetlana Lazebnik*

Unsharp Mask Filter

$$f + \alpha(f - f * g) = (1 + \alpha)f - \alpha f * g = f * ((1 + \alpha)e - g)$$