Computer Vision

Ch.8 Image Smoothing

Prof. Po-Yueh Chen (陳伯岳)
E-mail: pychen@cc.ncue.edu.tw
Ext: 8440

NCUE CSIE

Review (1/3)

➢ Noise

Common noise

- Salt-pepper noise
- Gaussian noise
- Spike noise
- Shot noise
- … etc.

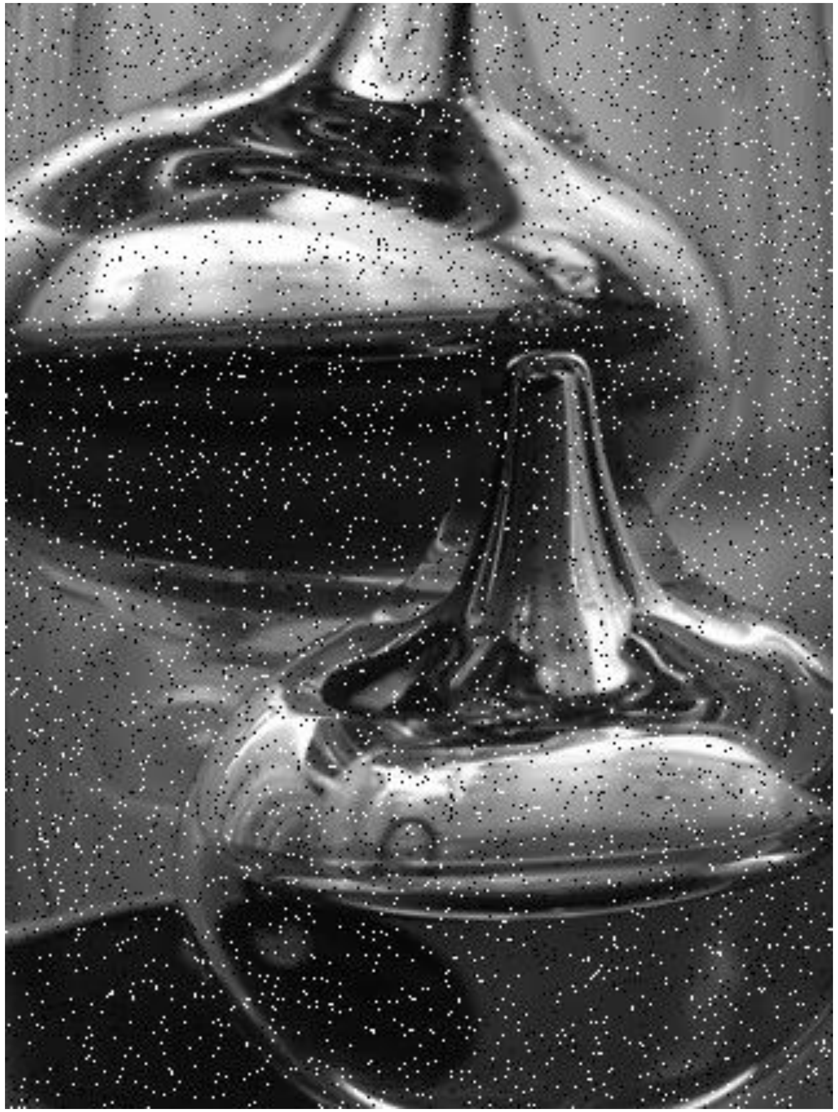
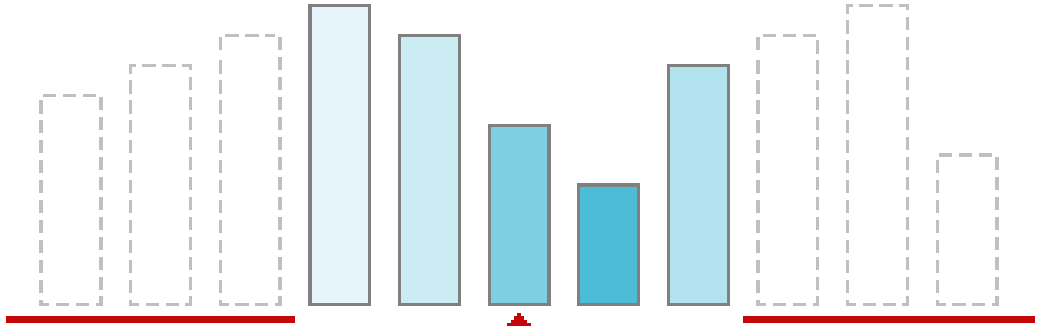


Image with salt and pepper noise

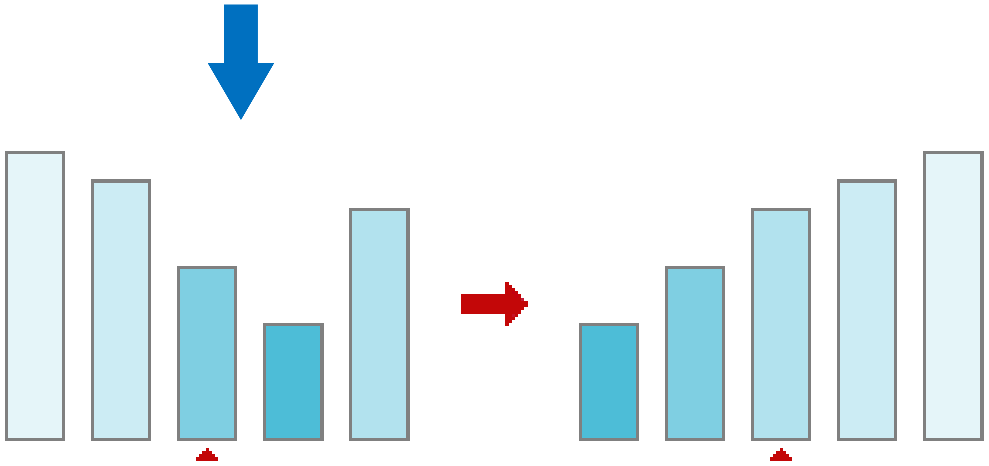
Source: https://en.wikipedia.org/wiki/File:Noise_salt_and_pepper.png

Review (2/3)

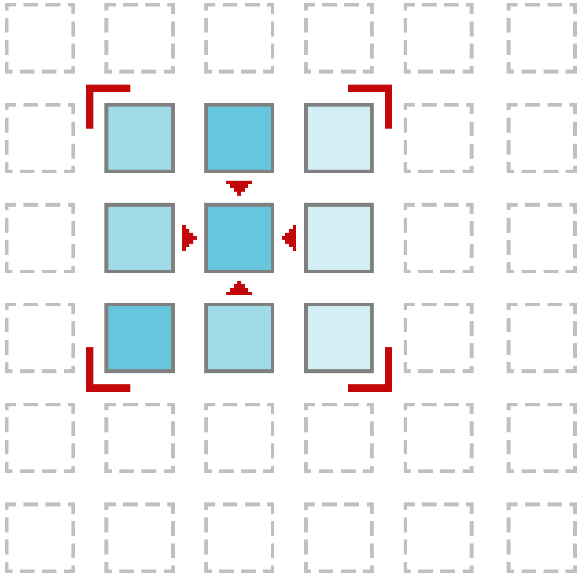
- **Median filter**



Window of size 5 in 1D.



Taking the median.

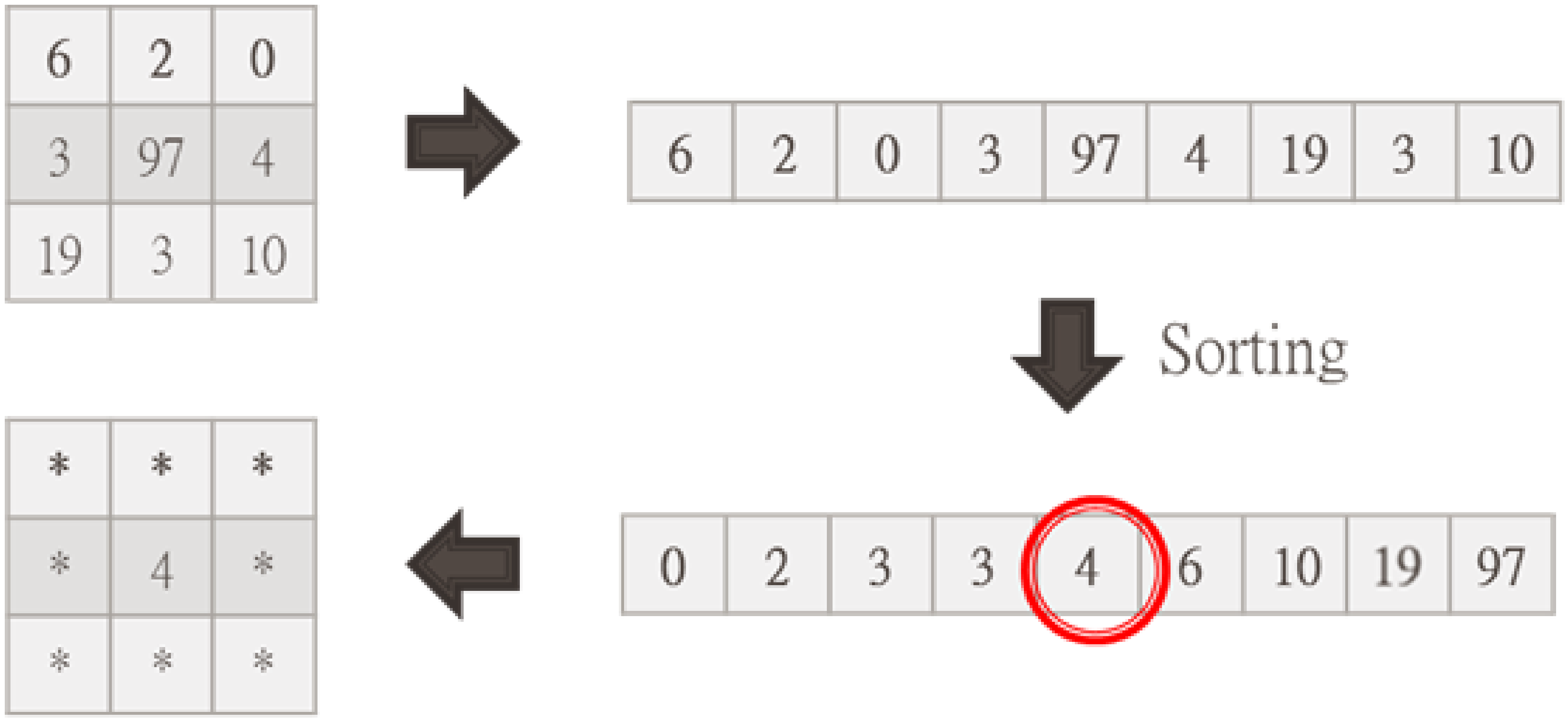


Window of size 3×3 in 2D.

Review (3/3)

➢ Median filter

- ✓ The Median Filter is a non-linear digital filtering technique, often used to remove noise from an image or signal.



Source: https://pic.pimg.tw/honglung/1484949784-577764374.png

Introduction of Image Smoothing

- ✓ **Smoothing is also called blurring.**
- ✓ **It is a simple and frequently used image processing operation.**
- ✓ **There are many reasons for smoothing.**
- ✓ **It is usually done to reduce noise or camera artifacts.**

- Image smoothing can be done by spatial filter
  - ➢ **Mean filter**
  - ➢ **Gaussian filter**
  - ➢ **Median filter**
  - ➢ **Bilateral filter**

Image Filter – Mean filter (1/3)

➢ Mean filter

- ✓ Mean filter is also called average filter.
- ✓ It smooths an image by replacing each pixel by the average pixel value computed over a rectangular neighborhood (kernel).

When performing smoothing with mean filter:

- Compute the average pixel value over this region.
- Save the result to the anchor pixel.

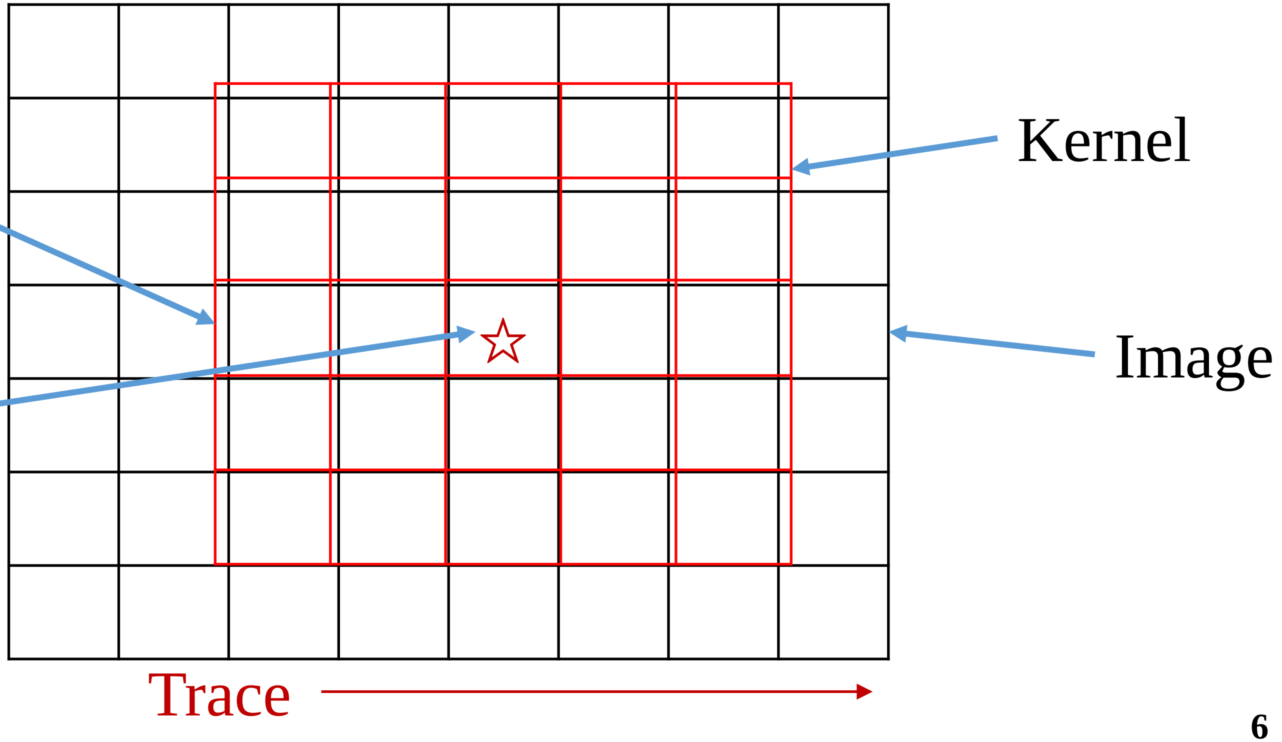


Image Filter – Mean filter (2/3)

- **Code**

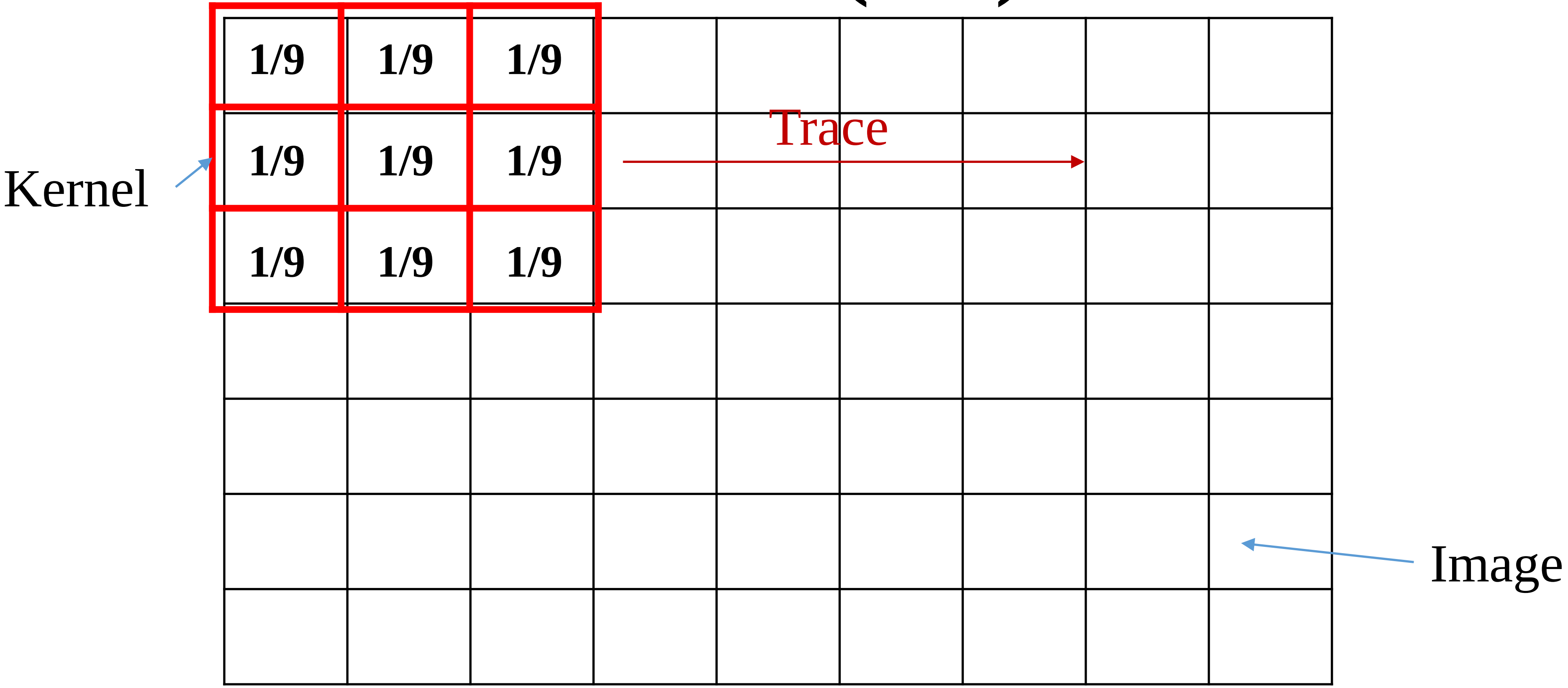
Syntax: ✓Blurs an image using the normalized box filter.

blur(src, dst, ksize, anchor, borderType);

src – Input image; it can have any number of channels.
dst – Output image of the same size and type as src.
ksize – Blurring kernel size.
anchor – Anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.
borderType – Border mode used to extrapolate pixels outside of the image.

✓ The call blur(src, dst, ksize, anchor, borderType) is equivalent to boxFilter(src, dst, src.type(), anchor, true, borderType).

Image Filter – Mean filter (3/3)



- **A 3x3 kernel is used to perform mean smoothing.**
- **It uses such a kernel to scan the whole image.**
- **At each location, it computes the average pixel value under the kernel, and saves the result to the kernel center.**

Introduction Gaussian filter & Convolutional

- ✓ **The next smoothing filter, Gaussian filter, is probably the most useful.**
- ✓ **Gaussian filtering convolves each point in the image with a Gaussian kernel and then sum to produce the output image.**

Two key points to be introduced first:

- **What is the operation of convolution?**

  ✓ **In AI and Deep Learning, the famous term CNN is the abbreviation of Convolutional Neural Network.**

- **What is Gaussian kernel?**

Image Filter – Convolution (1/14)

➢ Convolution

- ✓ Convolution is like computing the weighted sum of the pixel values under the kernel.

- ✓ For example, a 3×3 square kernel is used.

- ✓ The kernel coefficients $\boldsymbol{w(i,j)}$ are the weights of pixels $\boldsymbol{f(x+i,y+j)}$ under the kernel.

Image Filter – Convolution (2/14)

➢ Convolution

- ✓ We multiple the pixel value $f(x+i, y+j)$ by the corresponding weight $w(i,j)$, and then sum them up, as the following equation:

$$
\begin{aligned}
g(x,y) = \quad & w(-1,-1) \times f(x-1, y-1) + \\
& w(-1,0) \times f(x-1, y) + \\
& w(-1,1) \times f(x-1, y+1) + \\
& w(0,-1) \times f(x, y-1) + \\
& w(0,0) \times f(x, y) + \\
& w(0,1) \times f(x, y+1) + \\
& w(1,-1) \times f(x+1, y-1) + \\
& w(1,0) \times f(x+1, y) + \\
& w(1,1) \times f(x+1, y+1)
\end{aligned}
$$

Image Filter – Convolution (3/14)

➢ Convolution

$$
\begin{aligned}
g(x,y) = \; & w(-1,-1) \times f(x-1, y-1) + \\
& w(-1,0) \times f(x-1, y) + \\
& w(-1,1) \times f(x-1, y+1) + \\
& w(0,-1) \times f(x, y-1) + \\
& w(0,0) \times f(x, y) + \\
& w(0,1) \times f(x, y+1) + \\
& w(1,-1) \times f(x+1, y-1) + \\
& w(1,0) \times f(x+1, y) + \\
& w(1,1) \times f(x+1, y+1)
\end{aligned}
$$

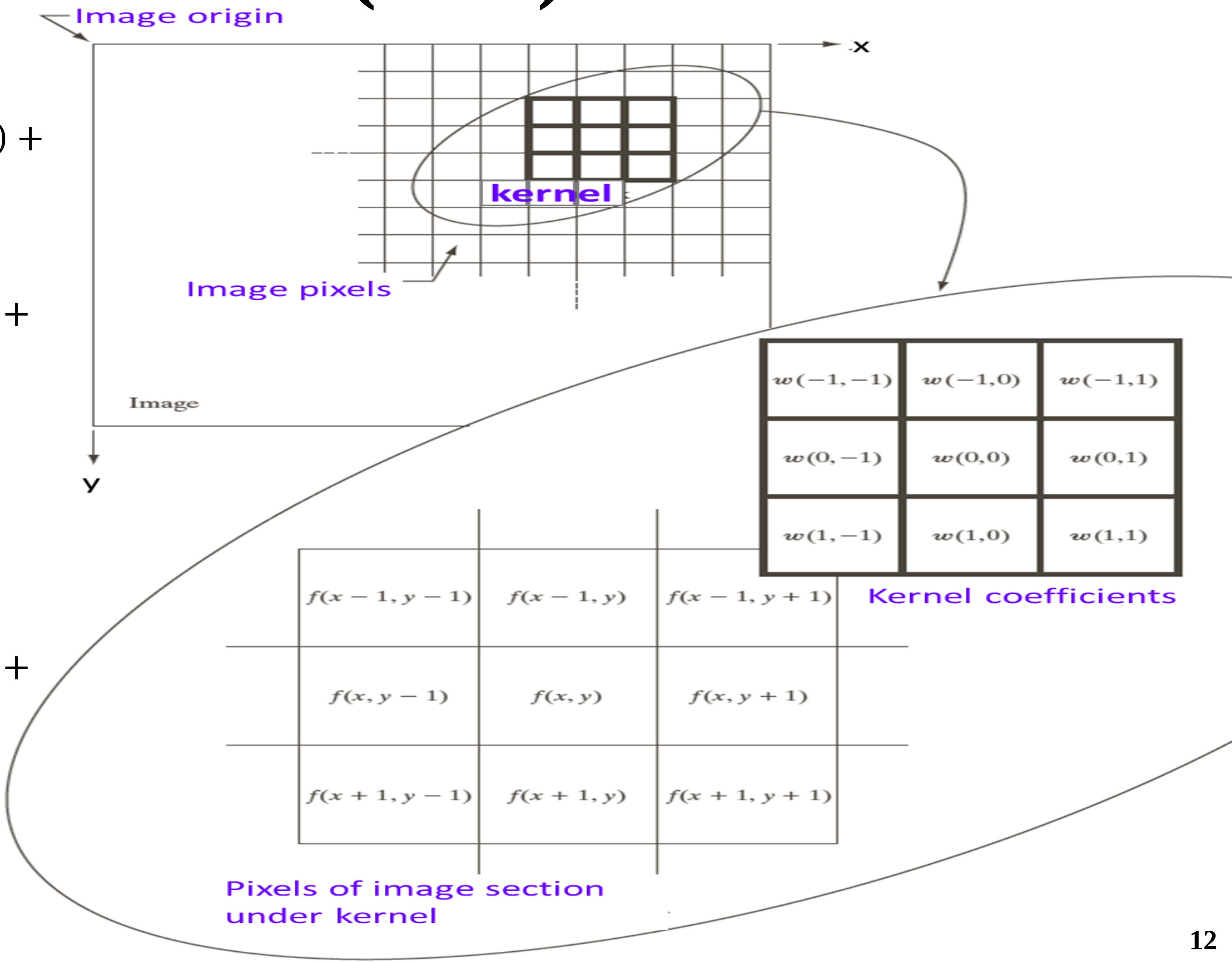
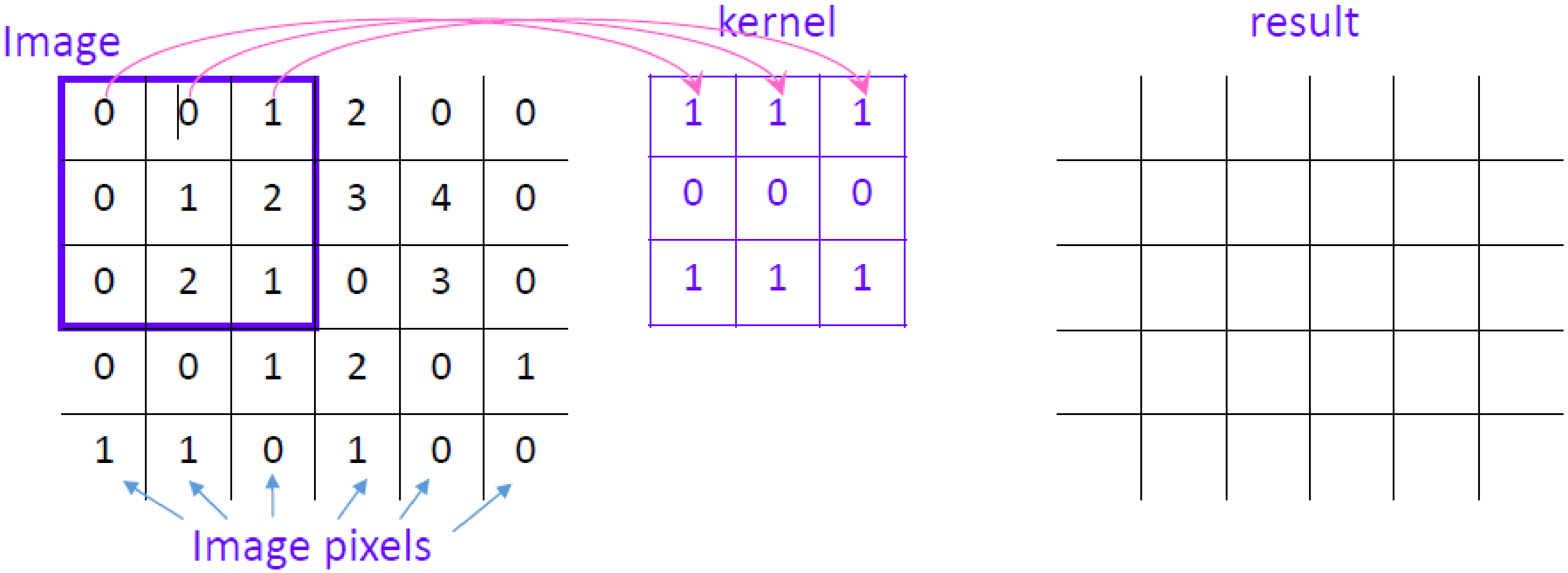


Image Filter – Convolution (4/14)

➢ Convolution



$$
\begin{aligned}
&0 \times 1 + 0 \times 1 + 1 \times 1 + \\
&0 \times 0 + 1 \times 0 + 2 \times 0 + \\
&0 \times 1 + 2 \times 1 + 1 \times 1 \\
&= 4
\end{aligned}
$$

✓ A simple example is used to explain the basic idea of convolution.

Image Filter – Convolution (5/14)

➢ Convolution

Image

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

kernel

1	1	1
0	0	0
1	1	1

result

$$
\begin{aligned}
&0 \times 1 + 0 \times 1 + 1 \times 1 + \\
&0 \times 0 + 1 \times 0 + 2 \times 0 + \\
&0 \times 1 + 2 \times 1 + 1 \times 1 \\
&= 4
\end{aligned}
$$

Image Filter – Convolution (6/14)

➢ Convolution

Image

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

kernel

1	1	1
0	0	0
1	1	1

result

$$
\begin{aligned}
&0 \times 1 + 0 \times 1 + 1 \times 1 + \\
&0 \times 0 + 1 \times 0 + 2 \times 0 + \\
&0 \times 1 + 2 \times 1 + 1 \times 1 \\
&= 4
\end{aligned}
$$

Image Filter – Convolution (7/14)

➢ Convolution

Image

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

kernel

1	1	1
0	0	0
1	1	1

result

	4				

$$0 \times 1 + 0 \times 1 + 1 \times 1 +$$
$$0 \times 0 + 1 \times 0 + 2 \times 0 +$$
$$0 \times 1 + 2 \times 1 + 1 \times 1$$
$$= 4$$

✓ To sum them up. Then we obtain the result, 4, and save it to the center pixel.

Image Filter – Convolution (8/14)

➢ **Convolution**

Image

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

kernel

1	1	1
0	0	0
1	1	1

result

	4	6			

$0 \times 1 + 1 \times 1 + 2 \times 1 +$
$1 \times 0 + 2 \times 0 + 3 \times 0 +$
$2 \times 1 + 1 \times 1 + 0 \times 1$
$= 6$

✓ Move the kernel to the next position, and perform the same computation. The result is 6.

Image Filter – Convolution (9/14)

➢ Convolution

Image

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

kernel

1	1	1
0	0	0
1	1	1

result

	4	6	7	

$$1 \times 1 + 2 \times 1 + 0 \times 1 + 2 \times 0 + 3 \times 0 + 4 \times 0 + 1 \times 1 + 0 \times 1 + 3 \times 1 = 7$$

- ✓ Move the kernel to the next position. The result is 7.

Image Filter – Gaussian filter (10/14)

- **Gaussian kernel**
  - **2D Gaussian function**

$$\boldsymbol{f(x,y) = A\,exp\left(-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)\right)}$$

A is the amplitude, (x_o, y_o) is the center,

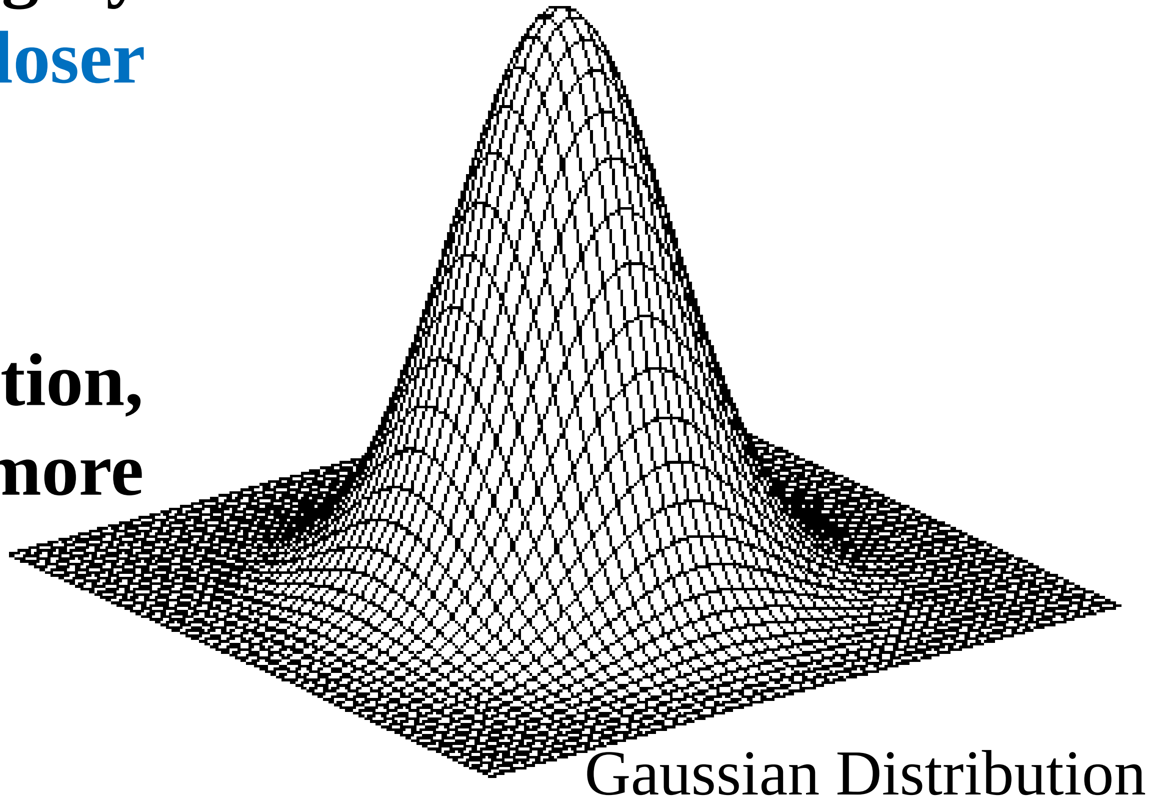
σ_x, σ_y are the standard variations in x- and y- directions.

Image Filter – Gaussian filter (11/14)

➢ Gaussian filter

- ✓ **A main characteristic of smoothing by Gaussian filter is that the pixels closer to the center are weighted more.**

- ✓ **This is because for the center location, closer pixels are usually more important than farther pixels.**

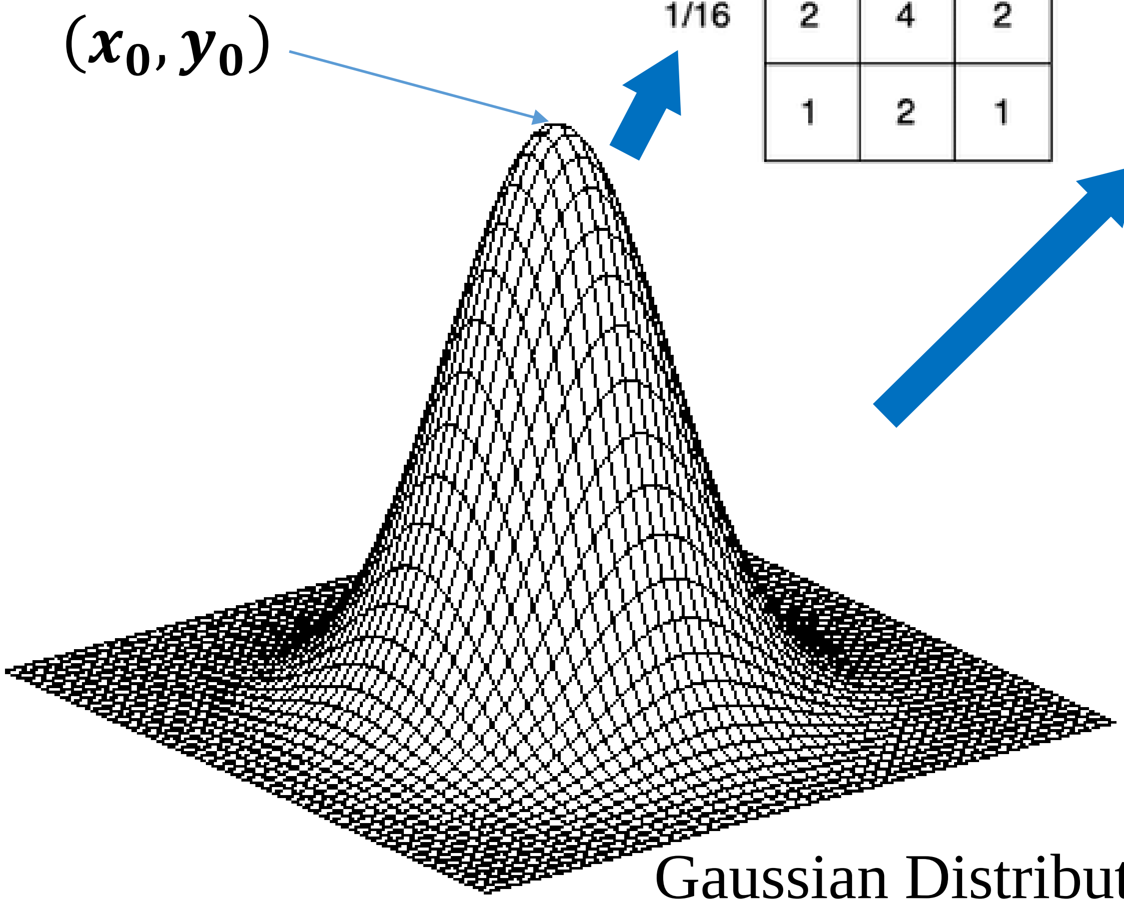


Gaussian Distribution

Image Filter – Gaussian filter (12/14)

- **Gaussian kernel**

✓ From the Gaussian function, we can generate the discrete approximation of Gaussian kernels of different sizes, e.g., 3×3, 5×5, 7×7, and so on.



Gaussian Distribution

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

Image Filter – Gaussian filter (13/14)

➢ Gaussian filter

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Gaussian kernel

✓ The pixels closer to the center are weighted more.

Image Filter – Gaussian filter (14/14)

➢ **Code**

Syntax:

GaussianBlur(src, dst, ksize, sigmaX, sigmaY, borderType);

src − Input image; it can have any number of channels.
dst − Output image of the same size and type as src.
ksize − Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero’s and then they are computed from sigma* .
sigmaX − Gaussian kernel standard deviation in X direction.
sigmaY − Gaussian kernel standard deviation in Y direction.
borderType − Pixel extrapolation method (see borderInterpolate() for details).

Image Filter – Comparison (1/3)

➢ **Gaussian filter**

Gaussian smoothing reduces noise while preserving signal.

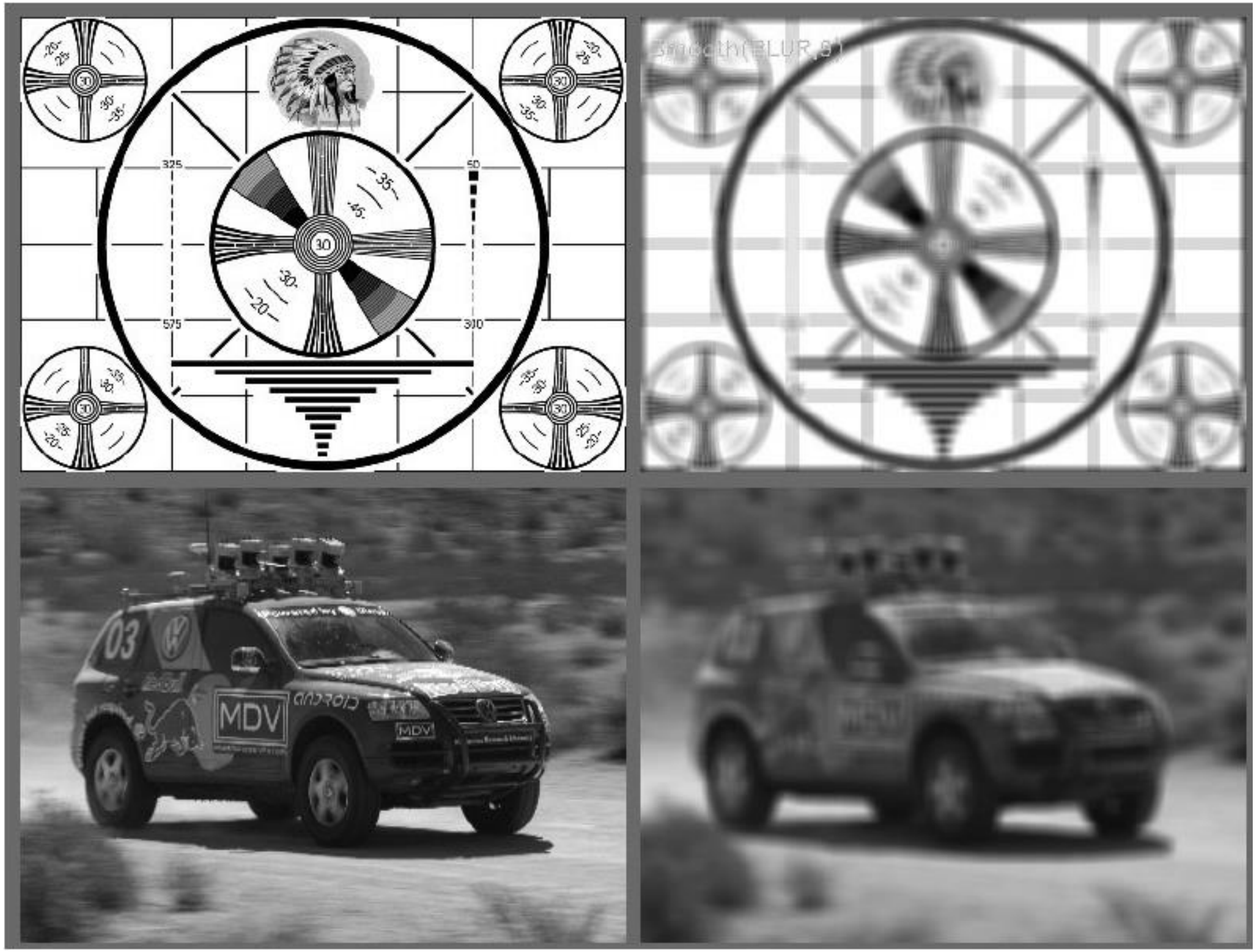
Although the image is blurred, the textures are not lost.

Original Image Gaussian Results

Image Filter – Comparison (2/3)

➢ Mean filter

Detailed textures are lost.

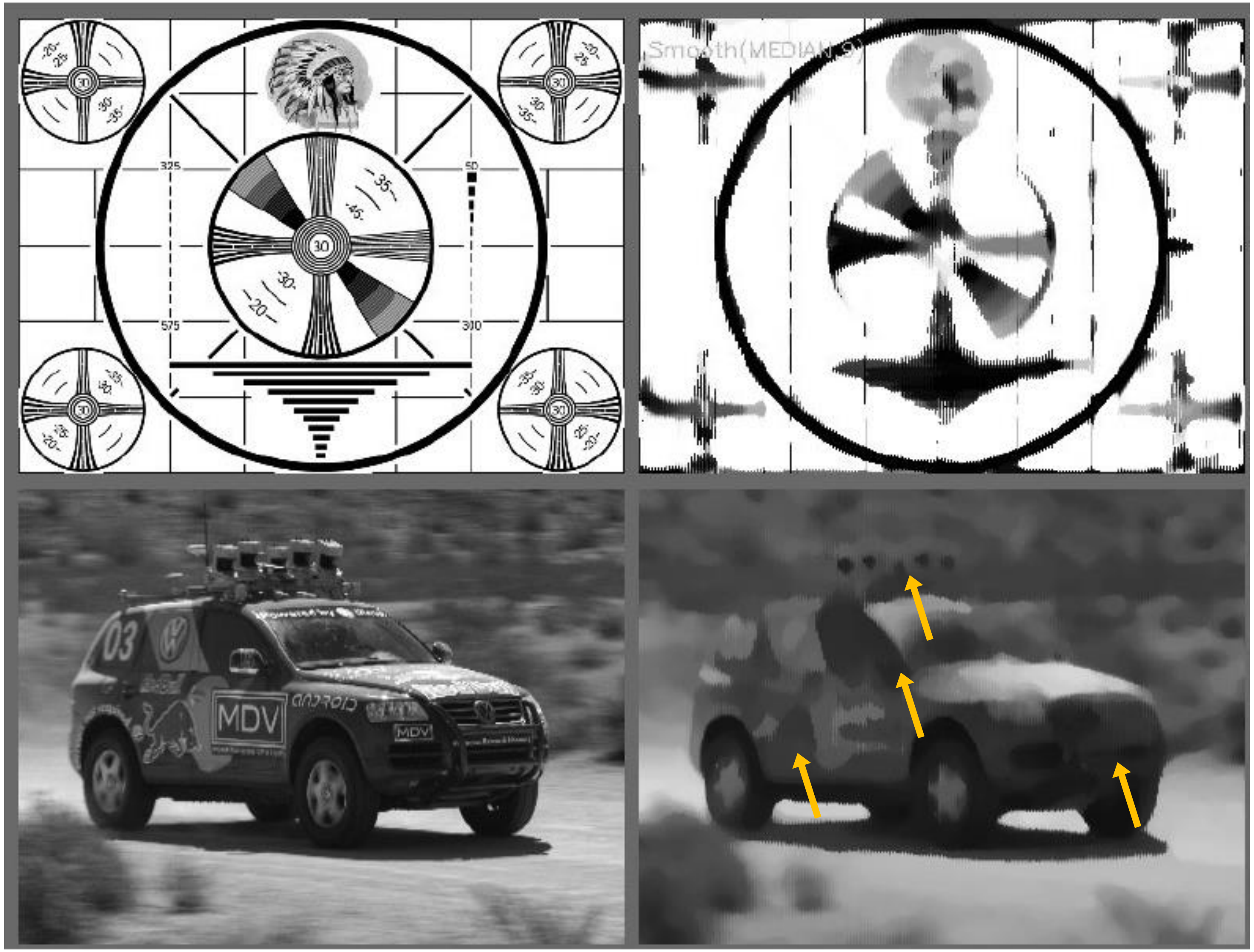


Original Image Mean Results

Image Filter – Comparison (3/3)

- **Median filter**

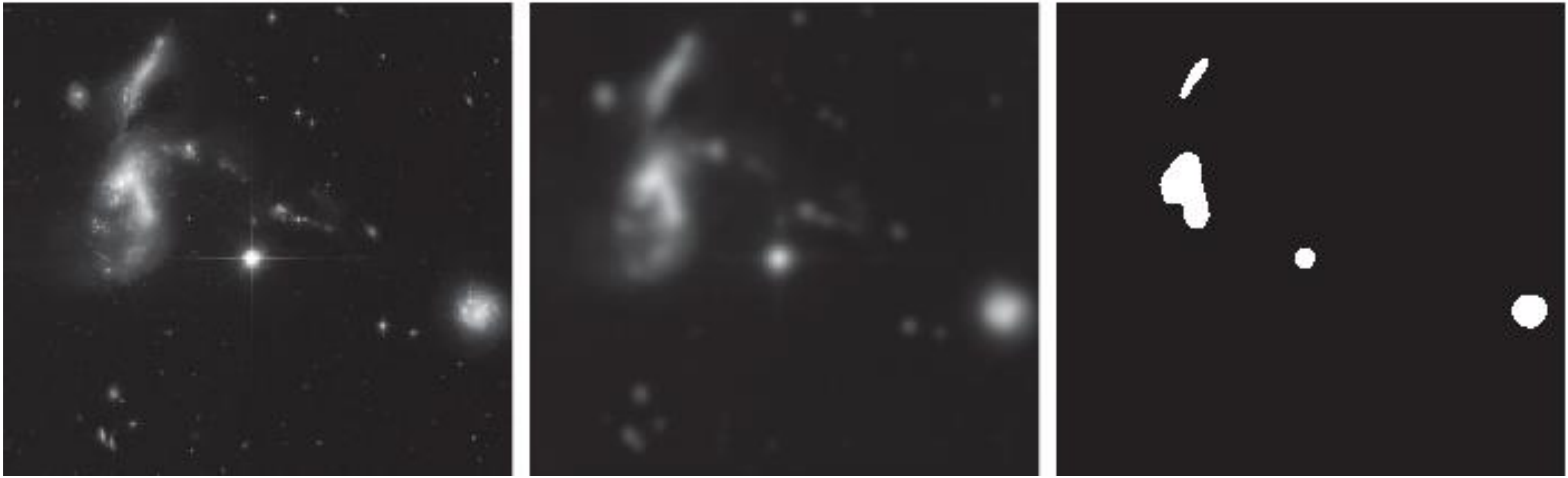
There are several areas of semi-equal color.



Original Image Median Results

Applications

Here is an example that uses the operations of blurring and thresholding for region extraction.
The objective is to eliminate irrelevant detail in the image.
"Irrelevant" refers to pixel regions that are small compared to kernel size.



(a) Telescope image (b) Result of Gaussian filtering (c) Result after thresholding

Image Filter – Bilateral filter (1/7)

✓ Bilateral filtering is known as edge-preserving smoothing.

Original Image

Bilateral Results

Although the image is blurred, Edges are still preserved.

Image Filter – Bilateral filter (2/7)

A typical motivation for Gaussian smoothing:

- Pixels in a real image should vary slowly over space and thus be correlated to their neighbors.
- Random noise can be expected to vary greatly from one pixel to the next (i.e., noise is spatially uncorrelated).

It is in this sense that Gaussian smoothing reduces noise while preserving signal.

Unfortunately, this method breaks down near edges, where you do expect pixels to be uncorrelated with their neighbors.

→ As a result, Gaussian smoothing blurs away edges.

✓ Bilateral filtering provides a mean of smoothing an image without blurring away its edges.

Image Filter – Bilateral filter (3/7)

Like Gaussian smoothing, bilateral filtering constructs a weighted average of each pixel and its neighboring pixels.

The weighting has two components:

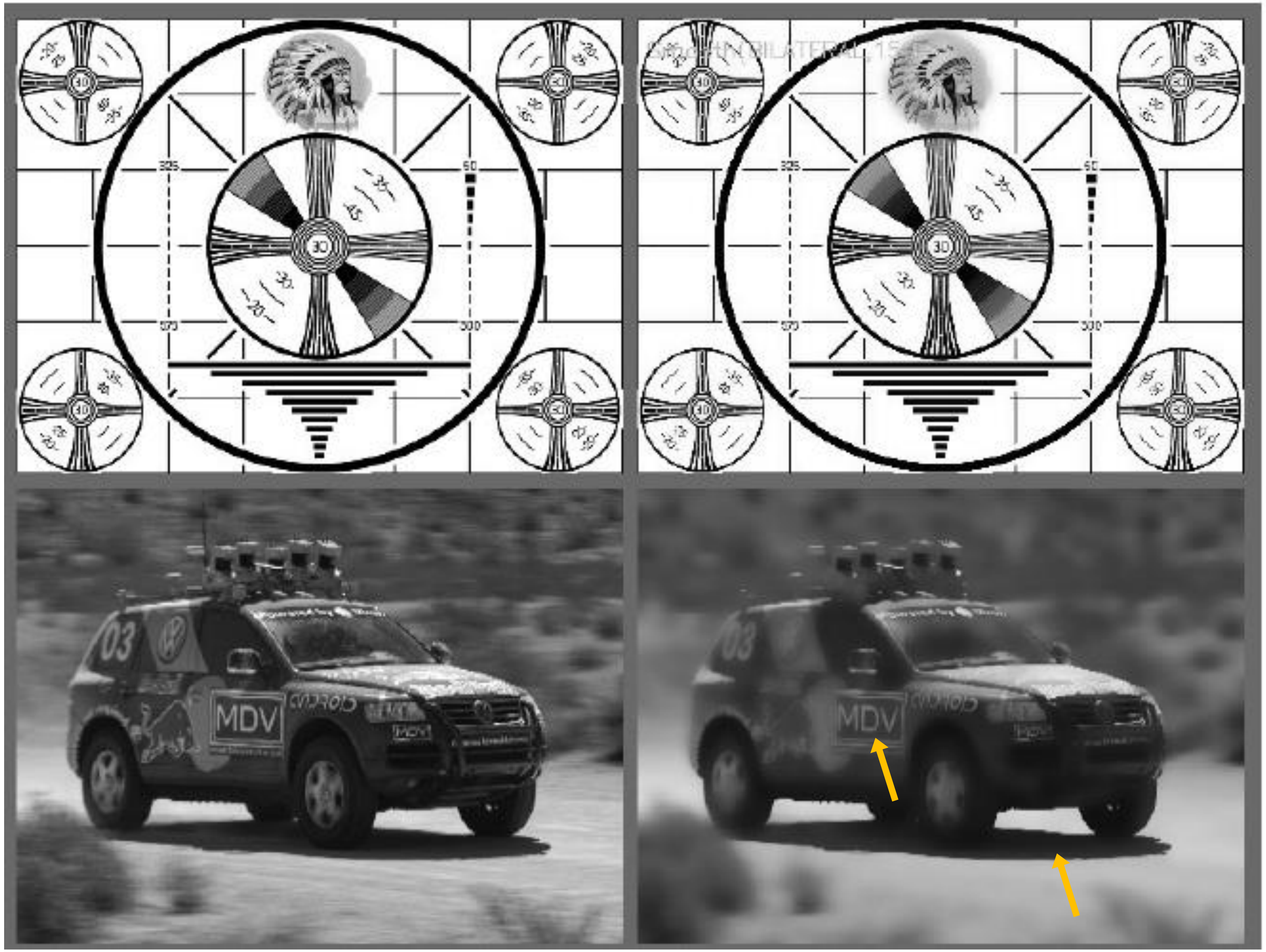
- The first is the same weighting used by Gaussian smoothing, based on the spatial distance from the center pixel. → closer pixels are weighted more.
- The second is also a Gaussian weighting but is based on the difference in intensity from the center pixel. → similar pixels are weighted more.

➢ You can think of bilateral filtering as Gaussian smoothing that weights similar pixels more, than less similar ones.

➢ The effect of this filter is typically to turn an image into what appears to be a watercolor painting of the same scene.

➢ This can be useful as an aid to segmenting an image.

Image Filter – Bilateral filter (4/7)



✓ Edges are preserved.

Image Filter – Bilateral filter (5/7)

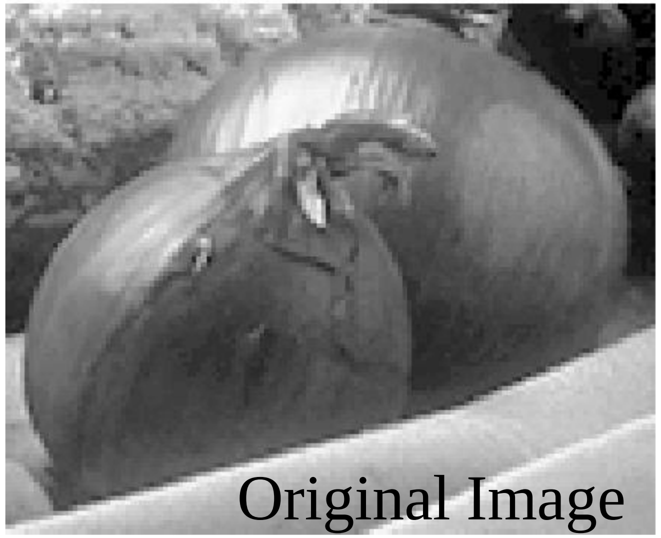
➢ Code

Syntax:

bilateralFilter(src, dst, d, sigmaColor, sigmaSpace, borderType);

src − Source 8-bit or floating-point, 1-channel or 3-channel image.
dst − Destination image of the same size and type as src.
d − Diameter of each pixel neighborhood that is used during filtering. If it is non-positive, it is computed from sigmaSpace.
sigmaColor − Filter sigma in the color space.
sigmaSpace − Filter sigma in the coordinate space.
borderType − border mode used to extrapolate pixels outside of the image, see BorderTypes

Image Filter – Bilateral filter (6/7)



Original Image

Bilateral filter

✓There are areas of semi-equal color.

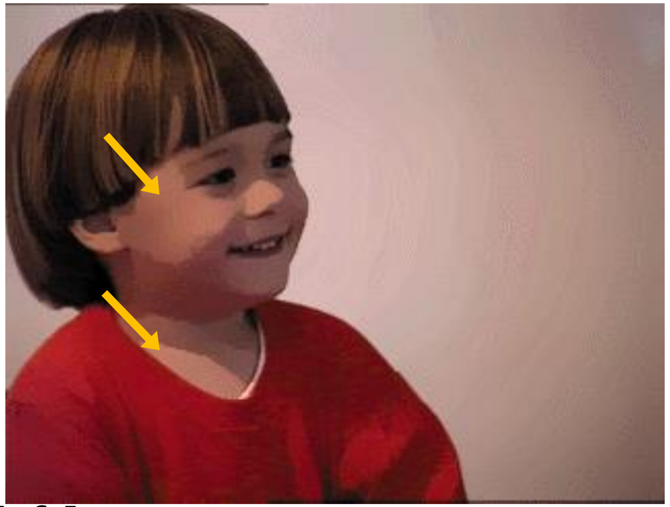
Image Filter – Bilateral filter (7/7)

Original Image

Bilateral filter

- ✓ Larger sigma results in larger areas of semi-equal color.

Original Image



Bilateral filter

- ✓ Bilateral filter make the image look like a watercolor painting.

Thanks!

Any questions?