數位影像處理
Digital Image Processing

Chapter 3
Spatial Filtering

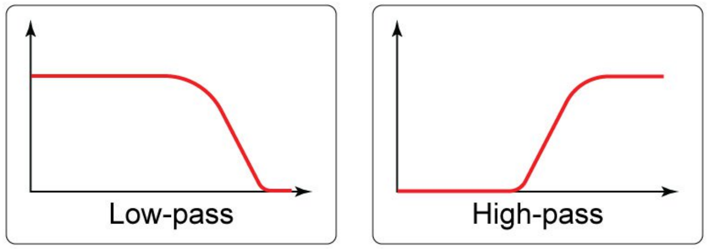
Ming-Han Tsai
Department of Information Engineering and Computer Science,
Feng Chia University

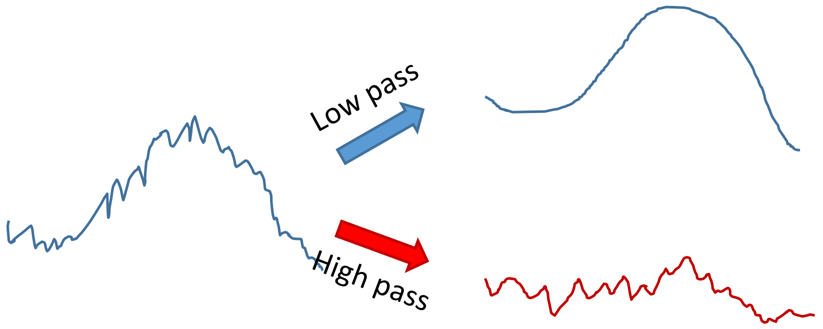
3 Spatial Filtering

- Introduction
- Correlation and Convolution
- Smooth Filter
- Threshold

1. Introduction

- Filter： frequency domain(Signal Processing)
- Only accept(pass) or reject(block) certain frequency
  - highpass filter
  - lowpass filter
  - bandpass filter





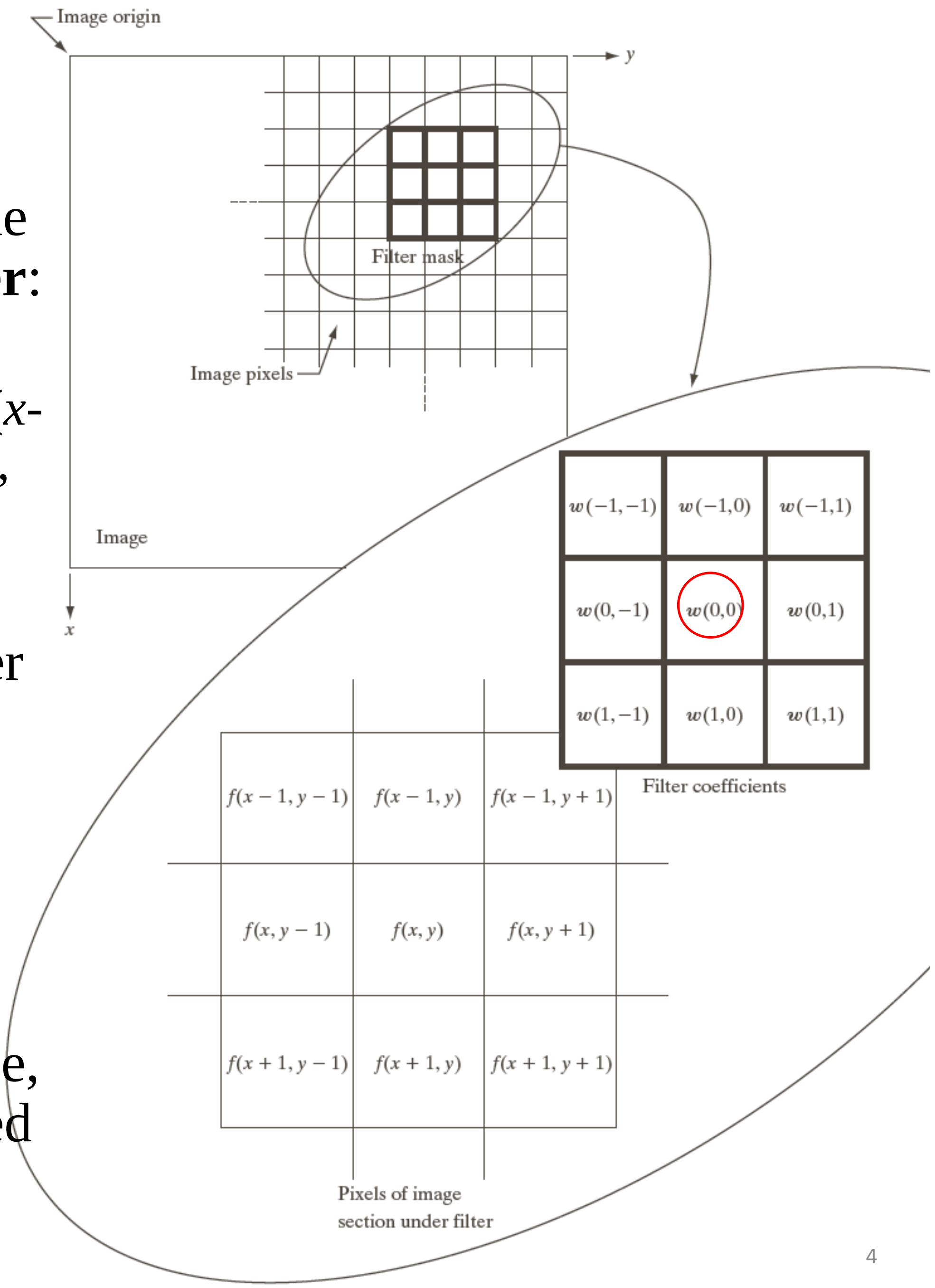
- spatial filter：also called **mask**、**kernel**、**template**、**window**…

- Filtering：For $p(x, y)$，the **filter response** $g(x, y)$ is the sum of product of **filter coefficients** and the **pixels of image section under filter**:
- $g(x, y) = w(-1,-1)\, f(x-1, y-1) + w(-1,0)\, f(x-1, y) + \ldots + w(0,0)\, f(x, y) + w(1, 1)\, f(x+1, y+1)$

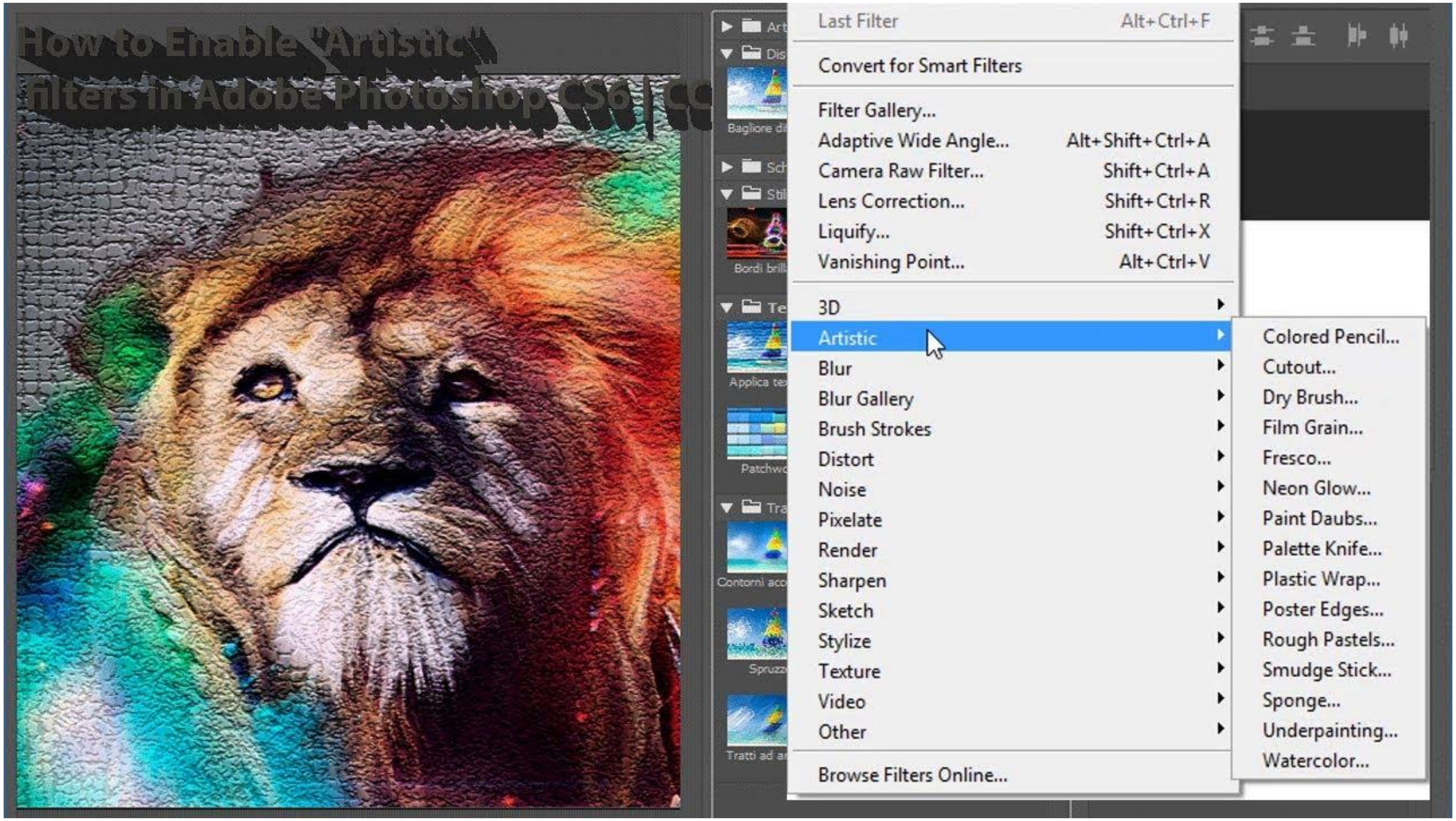
- Align the center coefficients of filter $w(0, 0)$ with image pixel $p(x, y)$

<u>**Notice**</u>

- The filter result stored in a new image
- If the result restore to original image, the content of image will be changed



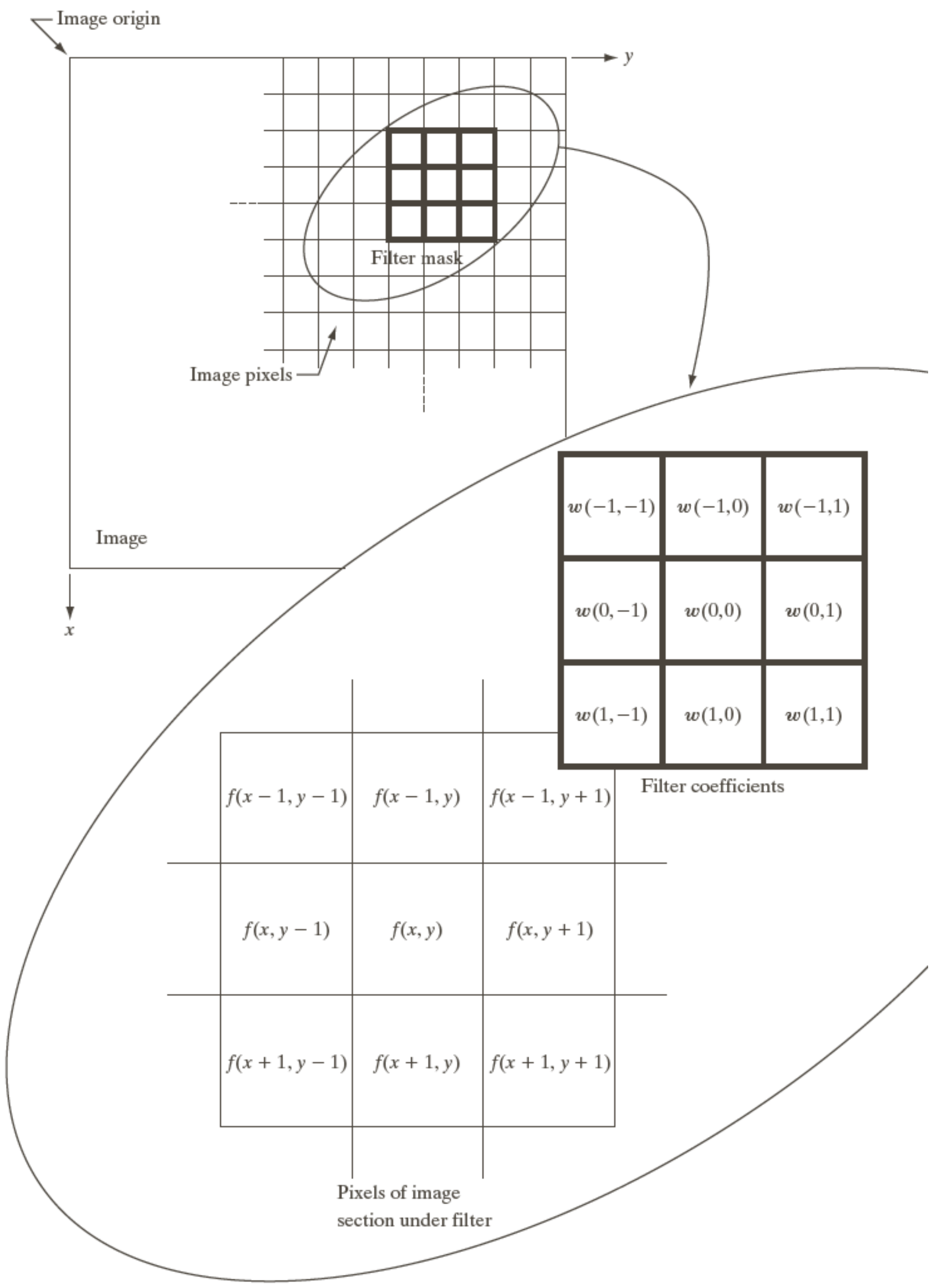
Filter in Photoshop / Image Editing



3. Correlation and Convolution

- 相關性(correlation)：
- Calculate the sum of product of each elements in correspondence location
- $g(x, y) = w(-1,-1)\, f(x-1, y-1)$
  $+ w(-1,0)\, f(x-1, y)+\ldots$
  $+w(0,0)\, f(x, y)$
  $+w(1, 1)\, f(x+1, y+1)$

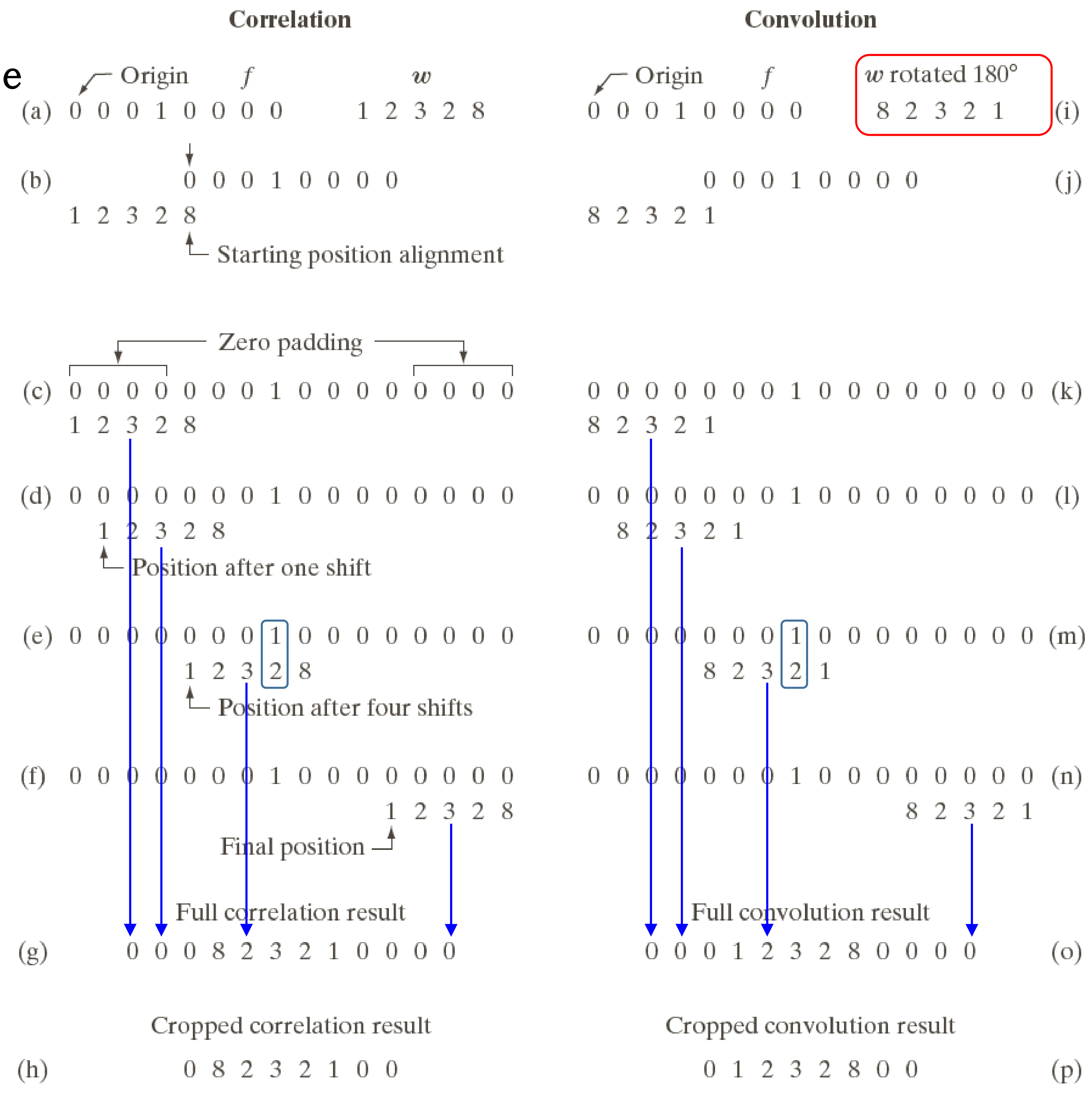
- 迴旋積(convolution)：
- Similar operation, but the filter is rotated 180º



1-D Example

f: function

w: filter

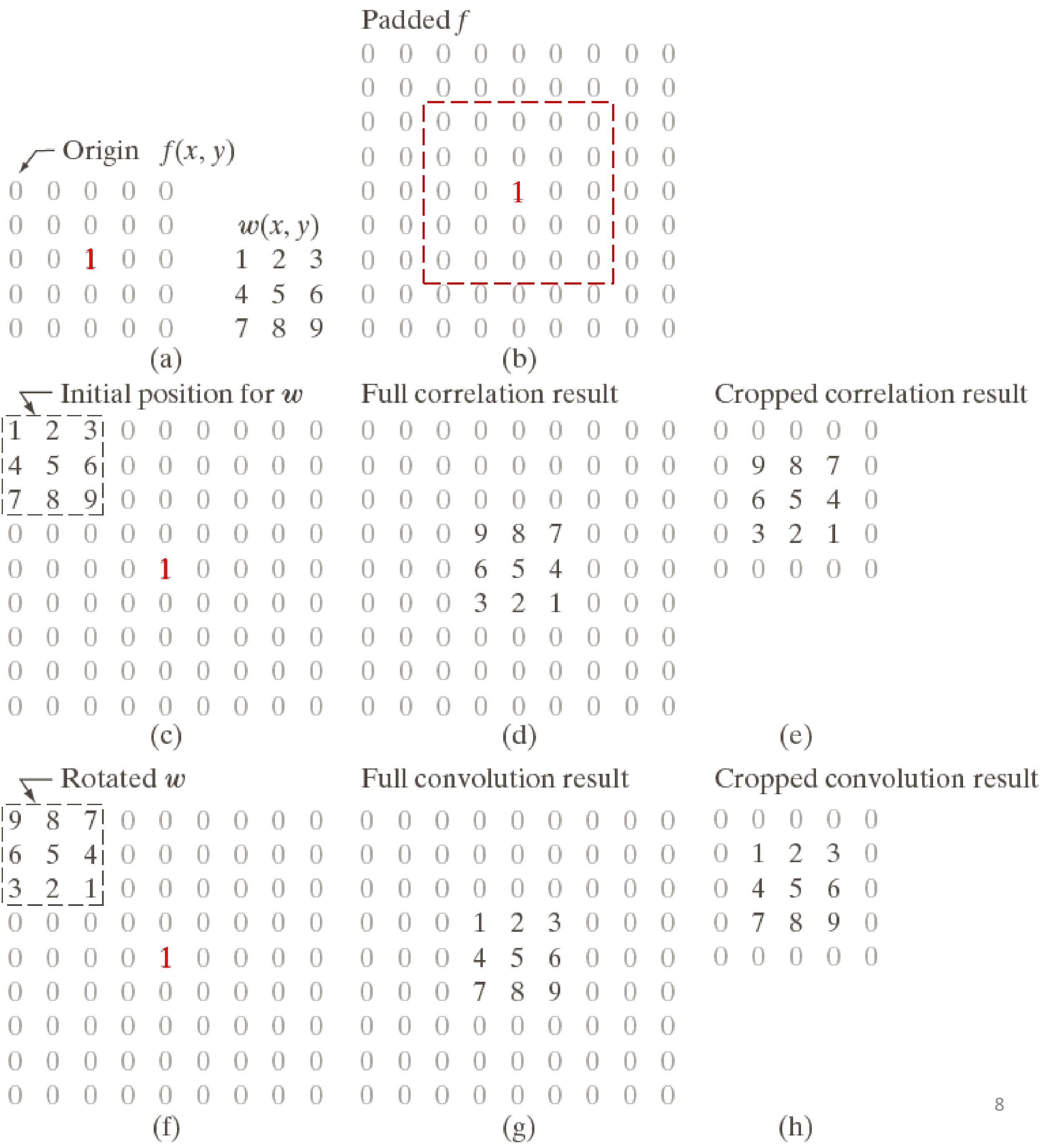


Correlation
Convolution
Origin
f
w
w rotated 180°
(a) 0 0 0 1 0 0 0 0 1 2 3 2 8
(i) 0 0 0 1 0 0 0 0 8 2 3 2 1
(b) 0 0 0 1 0 0 0 0
1 2 3 2 8
Starting position alignment
(j) 0 0 0 1 0 0 0 0
8 2 3 2 1
Zero padding
(c) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
(k) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
(d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after one shift
(l) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
(e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after four shifts
(m) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
(f) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Final position
(n) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1
Full correlation result
(g) 0 0 0 8 2 3 2 1 0 0 0 0
Full convolution result
(o) 0 0 0 1 2 3 2 8 0 0 0 0
Cropped correlation result
(h) 0 8 2 3 2 1 0 0
Cropped convolution result
(p) 0 1 2 3 2 8 0 0

2-D Example

f: function

w: filter



$$f(x,y) = \begin{matrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{matrix} \qquad w(x,y) = \begin{matrix} 1&2&3\\ 4&5&6\\ 7&8&9 \end{matrix}$$

(a)

Padded f

$$\begin{matrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{matrix}$$

(b)

Initial position for *w*

$$\begin{matrix} 1&2&3&0&0&0&0&0&0\\ 4&5&6&0&0&0&0&0&0\\ 7&8&9&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{matrix}$$

(c)

Full correlation result

$$\begin{matrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&9&8&7&0&0&0\\ 0&0&0&6&5&4&0&0&0\\ 0&0&0&3&2&1&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{matrix}$$

(d)

Cropped correlation result

$$\begin{matrix} 0&0&0&0&0\\ 0&9&8&7&0\\ 0&6&5&4&0\\ 0&3&2&1&0\\ 0&0&0&0&0 \end{matrix}$$

(e)

Rotated *w*

$$\begin{matrix} 9&8&7&0&0&0&0&0&0\\ 6&5&4&0&0&0&0&0&0\\ 3&2&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{matrix}$$

(f)

Full convolution result

$$\begin{matrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&1&2&3&0&0&0\\ 0&0&0&4&5&6&0&0&0\\ 0&0&0&7&8&9&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{matrix}$$

(g)

Cropped convolution result

$$\begin{matrix} 0&0&0&0&0\\ 0&1&2&3&0\\ 0&4&5&6&0\\ 0&7&8&9&0\\ 0&0&0&0&0 \end{matrix}$$

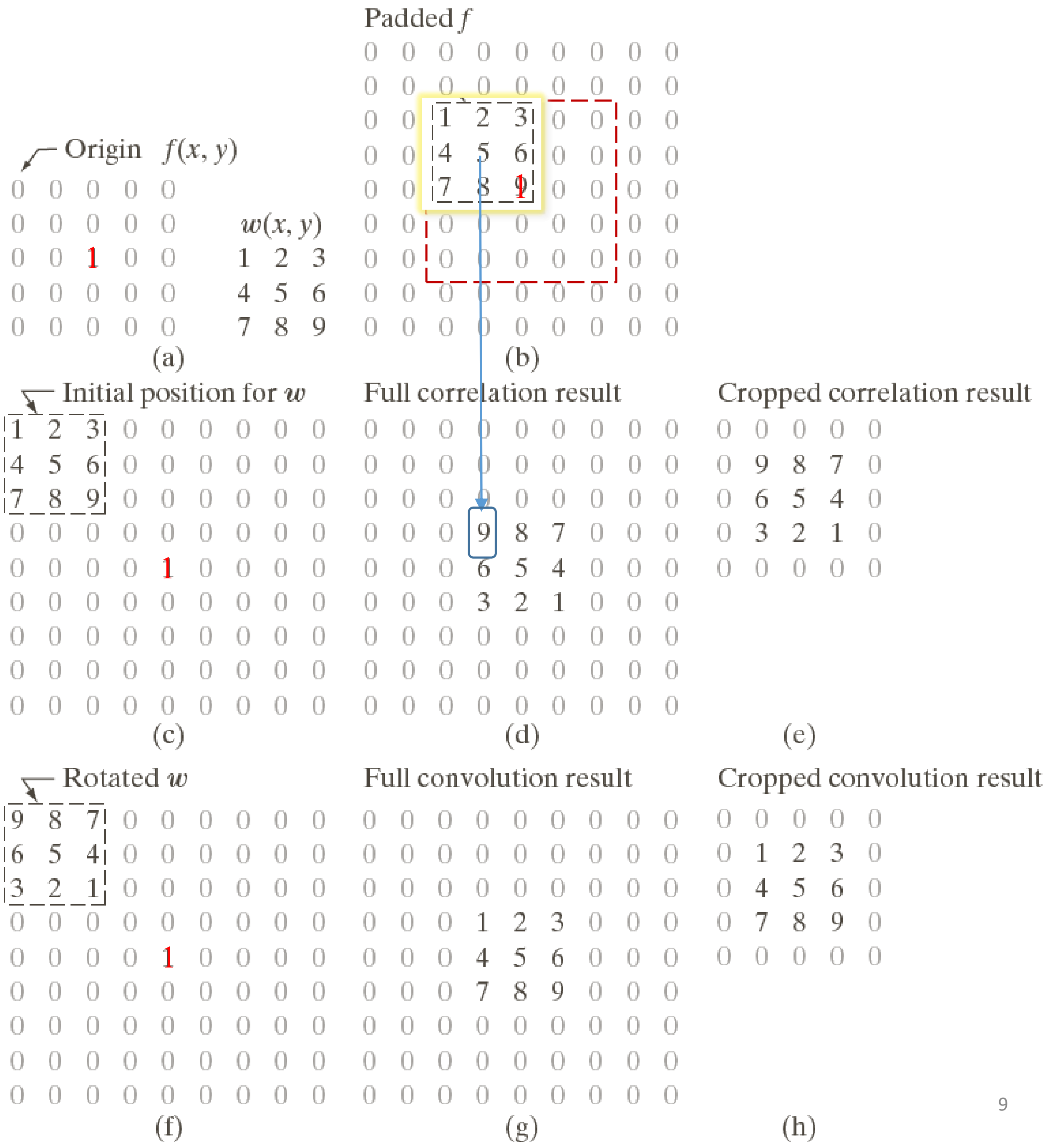
(h)

二維範例

f: function

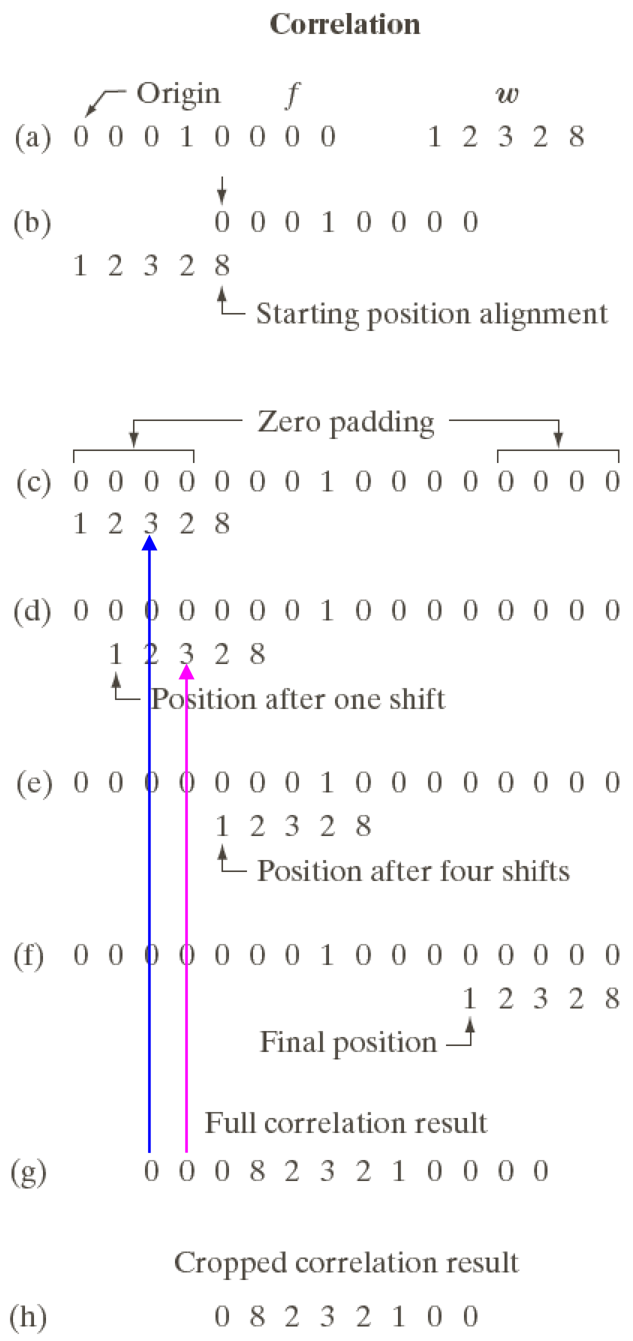
w: filter

- Correlation and Convolution is the same when the filter is symmetric



3. 空間相關性與迴旋積

- The correlation of a filter *w* and a discrete unit impulse function *I* is a reversed *w*
- *Discrete Unit Impulse Function*：
  - $I(t)\ (impulse) = \begin{cases} 1, when\ t = 0 \\ 0. otherwise \end{cases}$

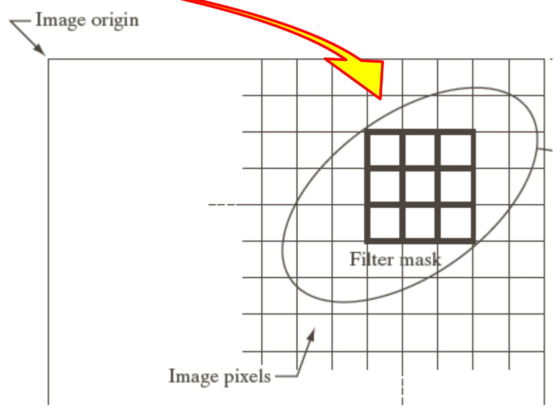

Correlation
Origin
f
w
(a) 0 0 0 1 0 0 0 0   1 2 3 2 8
(b) 0 0 0 1 0 0 0 0
1 2 3 2 8
Starting position alignment
Zero padding
(c) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
(d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after one shift
(e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Position after four shifts
(f) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 2 3 2 8
Final position
Full correlation result
(g) 0 0 0 8 2 3 2 1 0 0 0 0
Cropped correlation result
(h) 0 8 2 3 2 1 0 0

- If the filter *w* is rotated, the result of correlation of impulse function is the original *w*
  - *Convolution*

4. Smoothing Filter

- Blurry, reduce noise
- Averaging filter is a linear filter

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$



Normalization

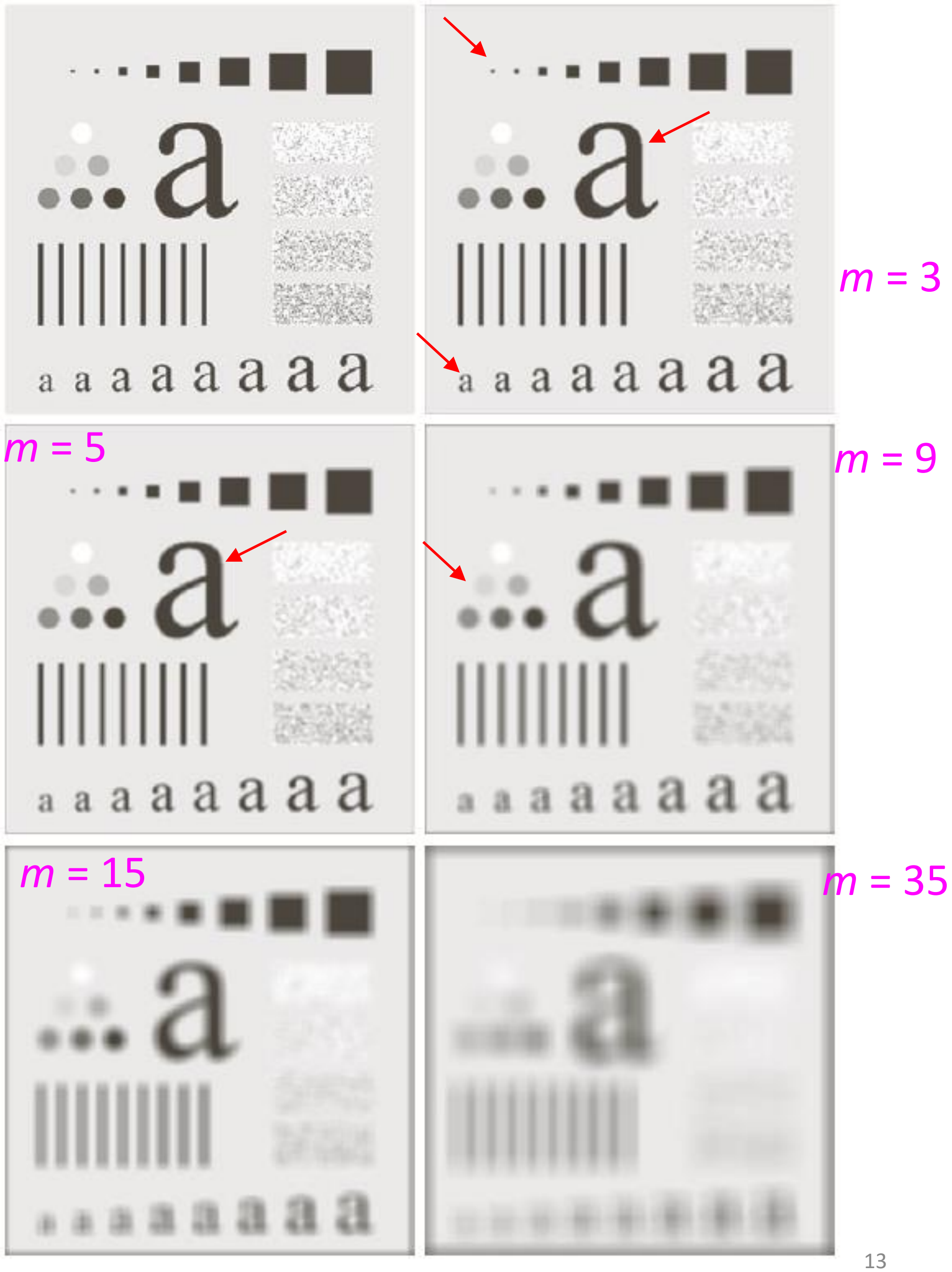
weighted average:
The center point is much more important

4. 平滑空間濾波器

- Image size：500x500
- Average Filter
- Mask size *m* = 3, 5, 9, 15, 35

1	1	1
1	1	1
1	1	1

- Look at the different size boxes in different filter scale



OpenCV: Image Blur - Average

- Smooth an image by replacing each pixel by the average pixel value computed over a rectangular neighborhood.
- void **blur**(InputArray **src**, OutputArray **dst**, Size **ksize**, Point **anchor**=Point(-1,-1), int **borderType**=BORDER_DEFAULT)
  - **src** – input image
  - **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.

➔cv::blur(image, result, cv::Size(3,3));

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Such a matrix is sometimes called a kernel or a mask.

OpenCV: Image Blur - Average

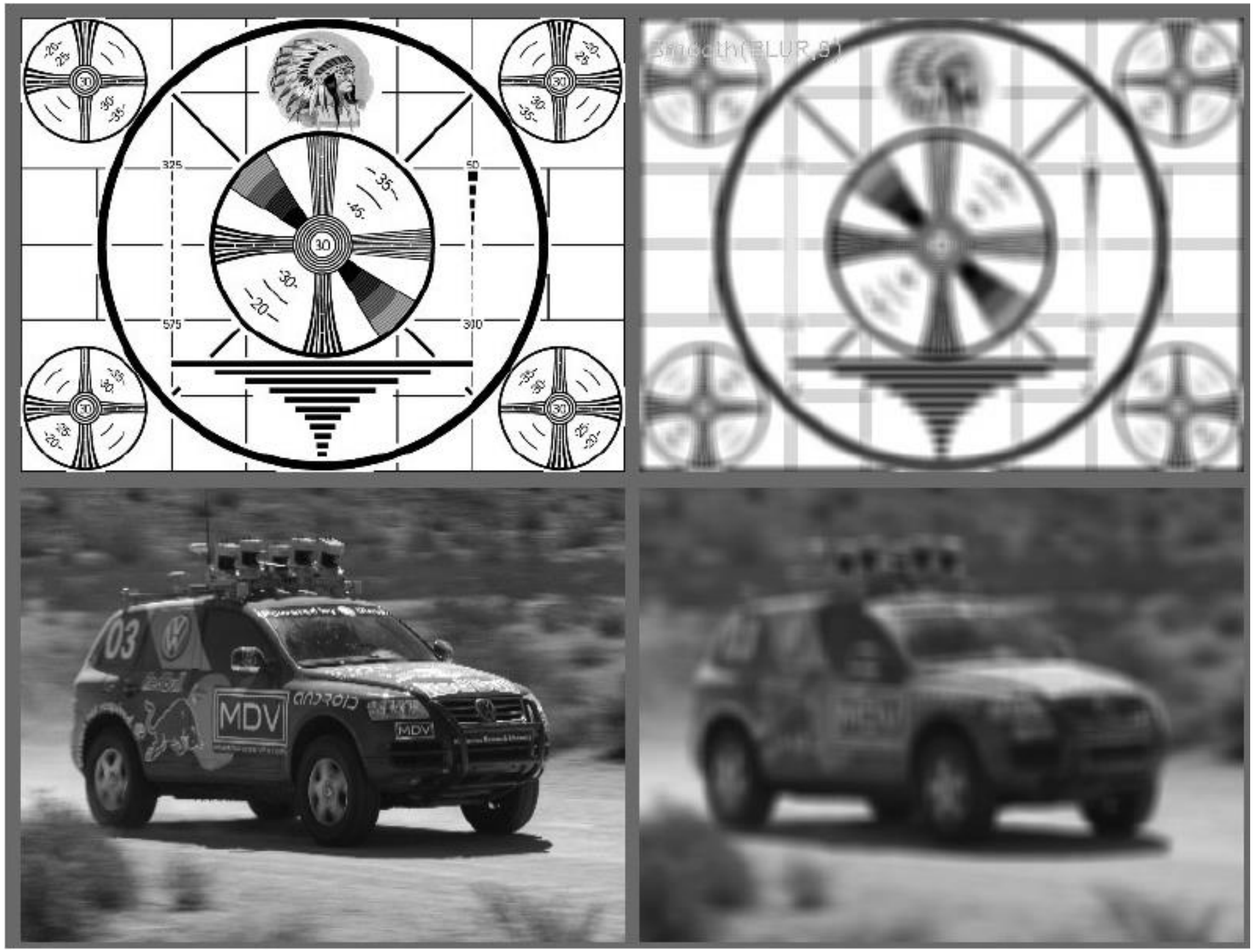
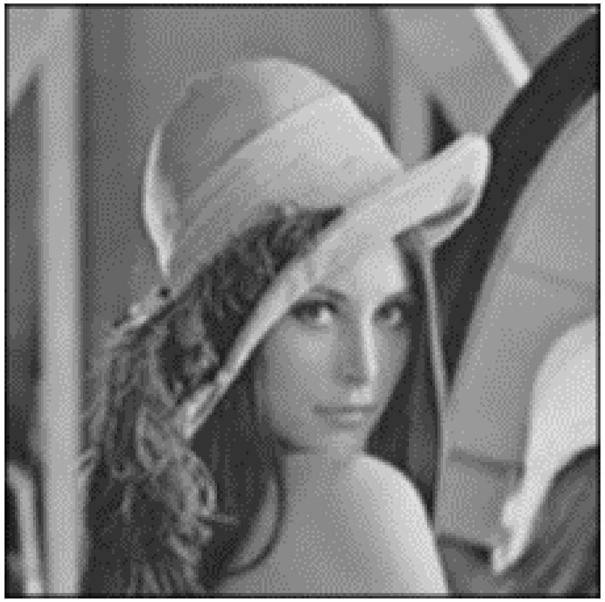
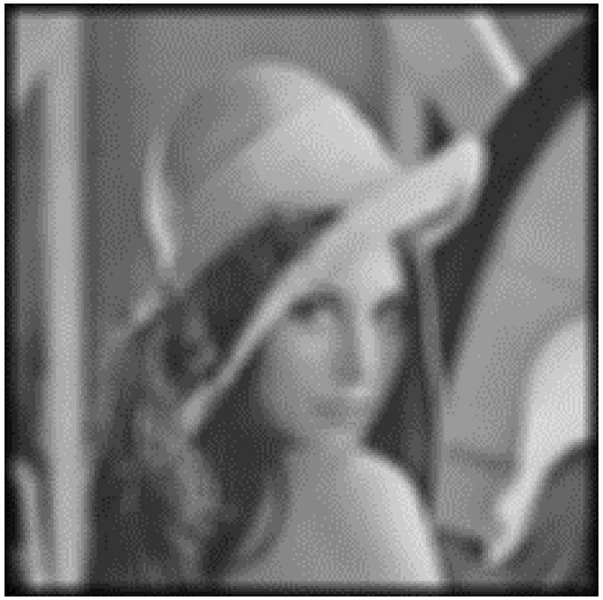


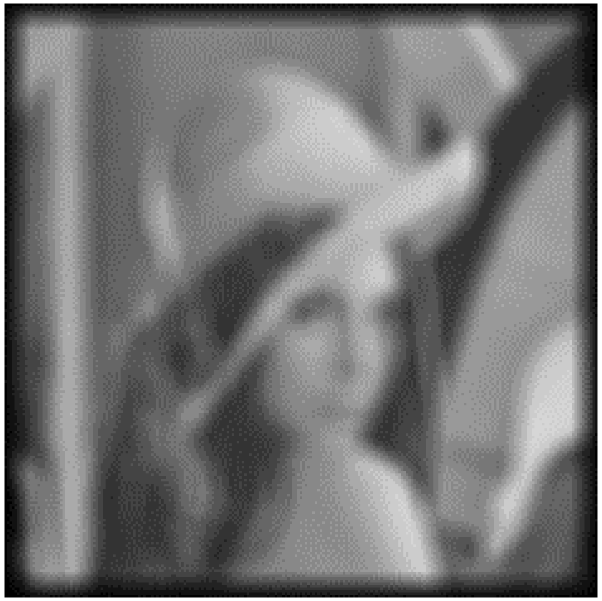
Image smoothing by block averaging

Average Blur

smooth


3*3


7*7

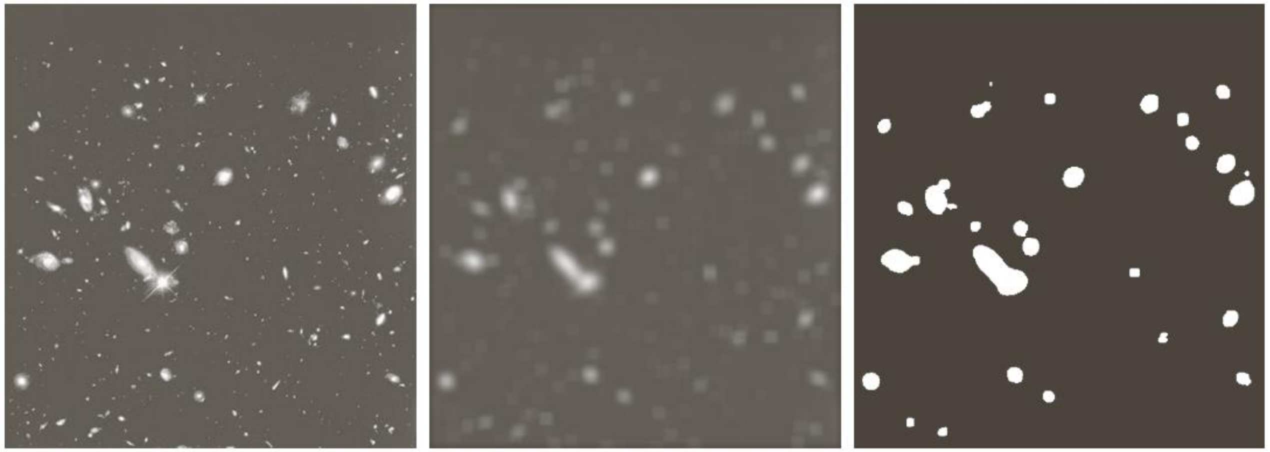

11*11

4. 平滑空間濾波器

- Smooth an image to
  - Erase small object region
  - Enlarge large object region

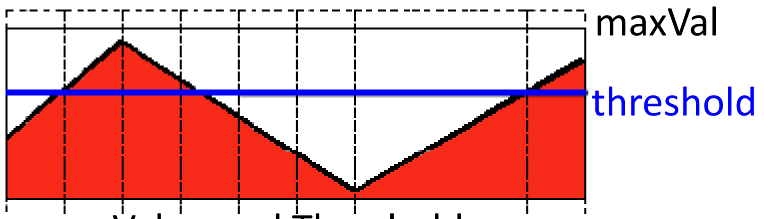
門檻化 (Thresholding)

- $f(x, y)$: pixel，$g(x, y)$: after Thresholding
- T: 門檻值 $g(x,y)=\begin{cases}255, if\ f(x,y)>T\\ 0, if\ f(x,y)\le T\end{cases}$ $or\ g(x,y)=\begin{cases}1, if\ f(x,y)>T\\ 0, if\ f(x,y)\le T\end{cases}$

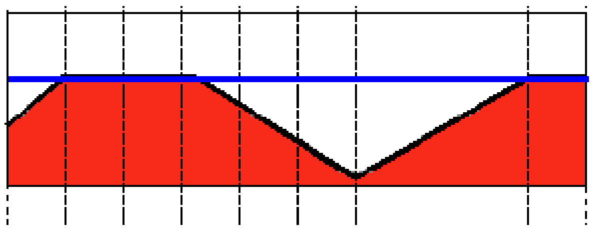

(a)哈伯望遠鏡之太空影像，528x485
(b)用15x15的平均遮罩作濾波
(c) 對(b)做門檻化

OpenCV Threshold Types

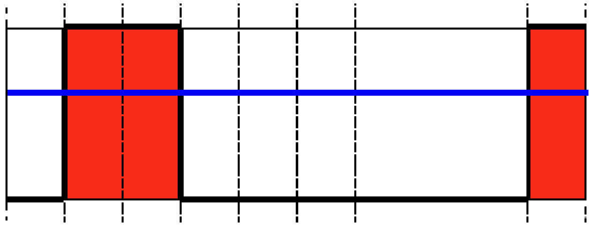
- threshold (src, dst, threshold, max_value, threshold_type);



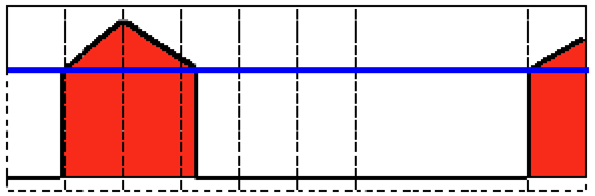
Value and Threshold



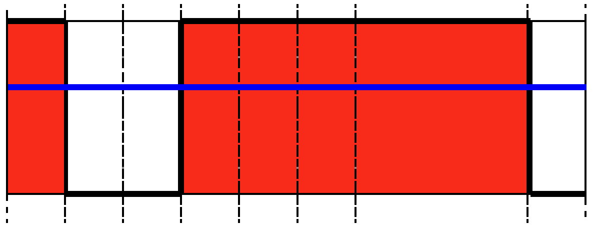
THRESH_TRUNC



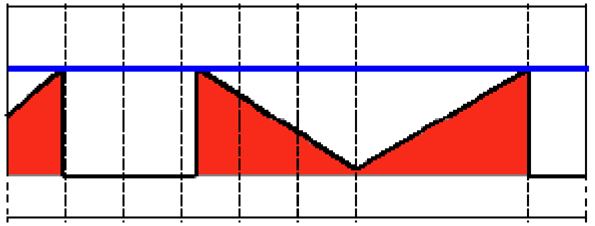
THRESH_BINARY



THRESH_TOZERO



THRESH_BINARY_INV



THRESH_TOZERO_INV

Thresholding 的應用

- cvtColor (src, dst, CV_BGR2GRAY);
- threshold (src, dst, threshold, max_value, threshold_type);

cvtColor (frame, gray, CV_BGR2GRAY);

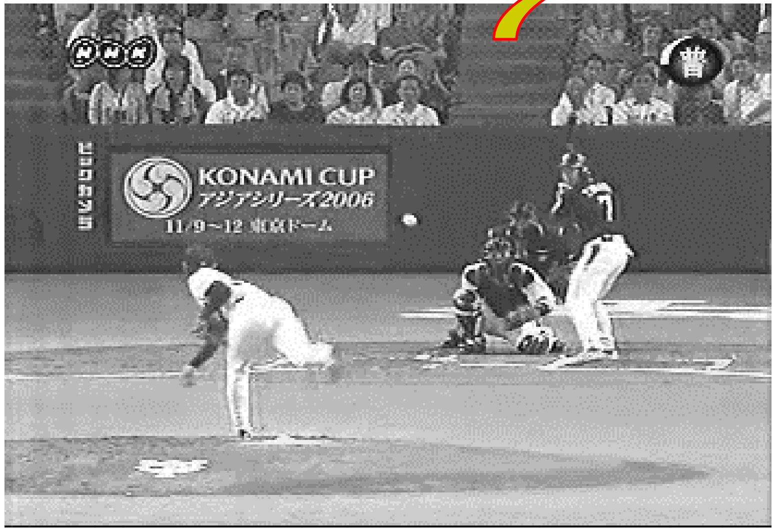
frame (3 channels)

gray (1 channel)

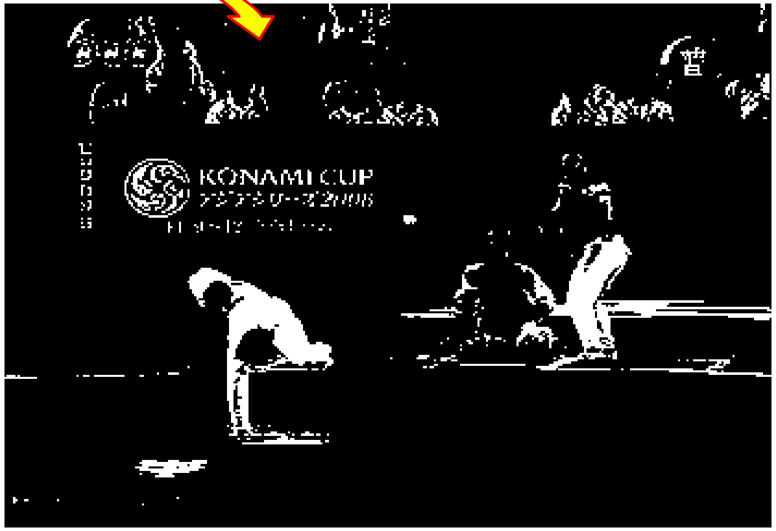
Thresholding 的應用

- cvtColor (src, dst, CV_BGR2GRAY);
- threshold (src, dst, threshold, max_value, threshold_type);

threshold (gray, white, 200, 255, **THRESH_BINARY**);



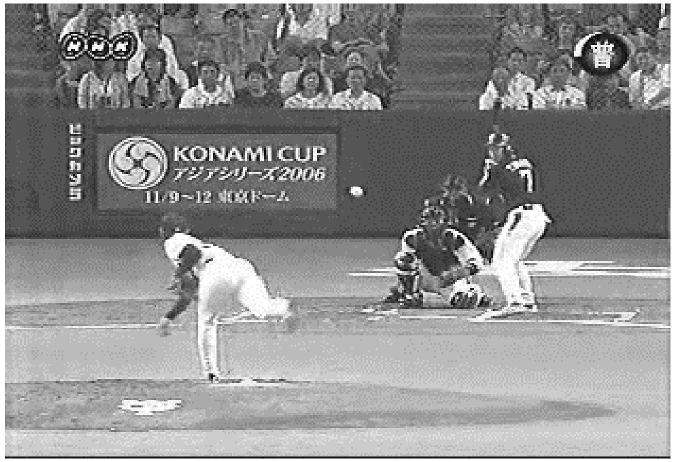
gray

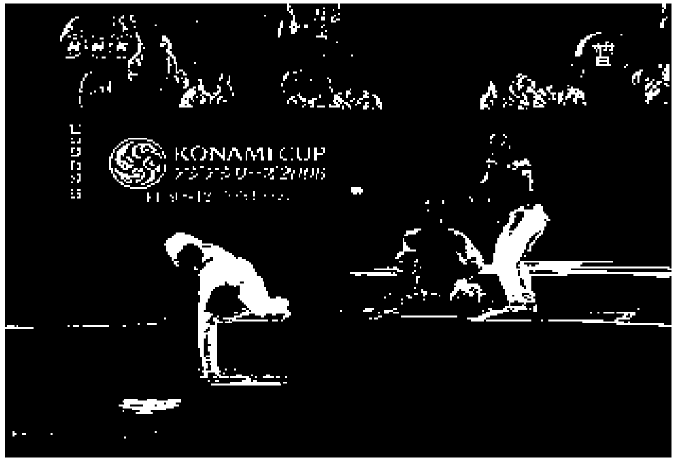


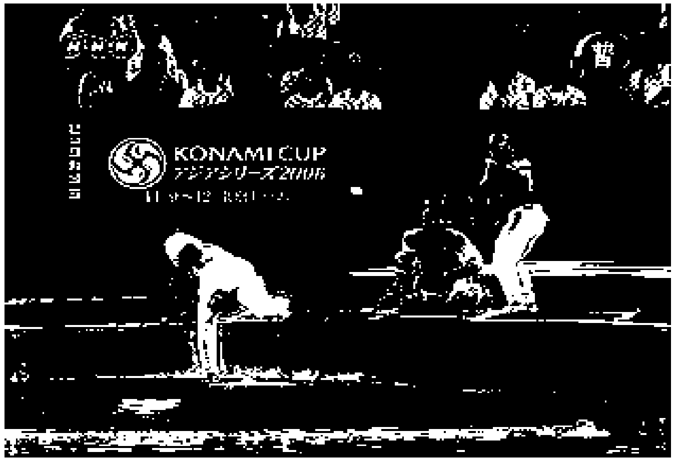
White (threshold = 200)

Thresholding 的應用

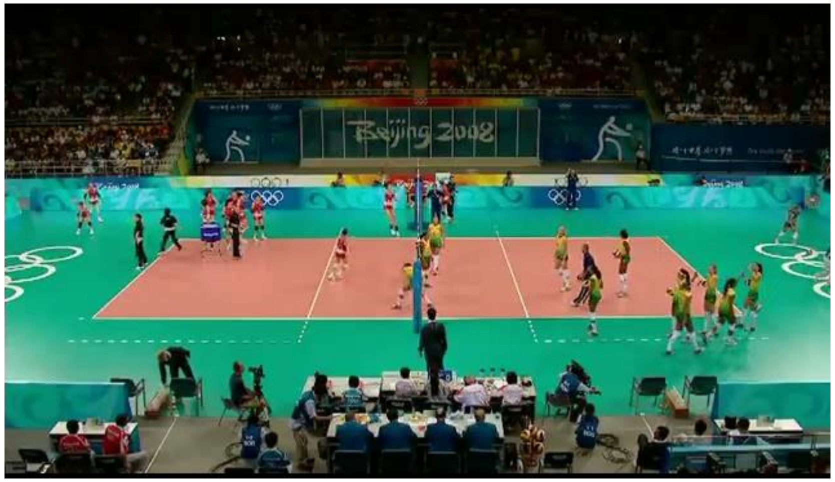
- Different thresholds

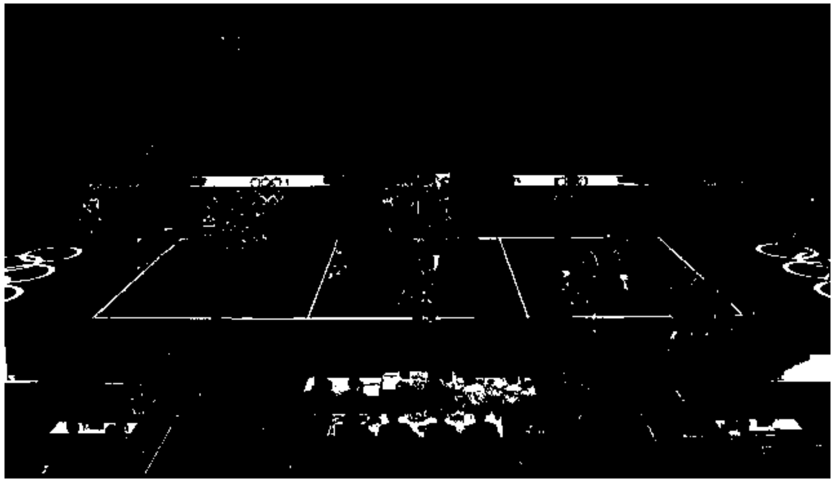


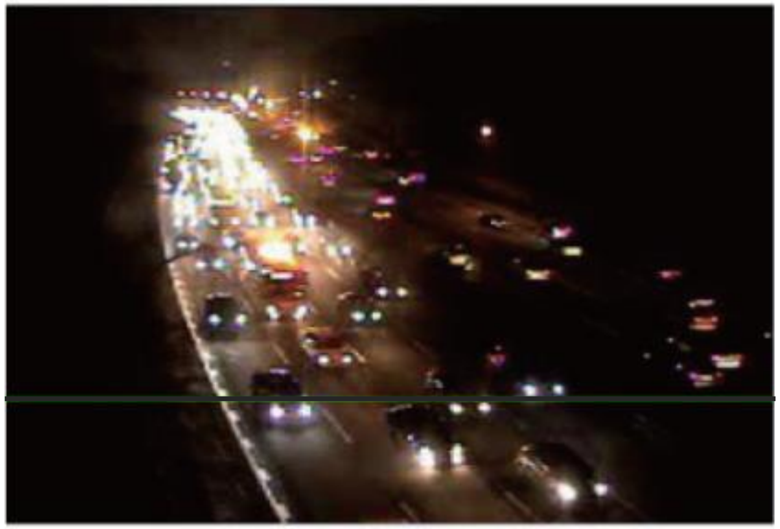

(threshold = 200)

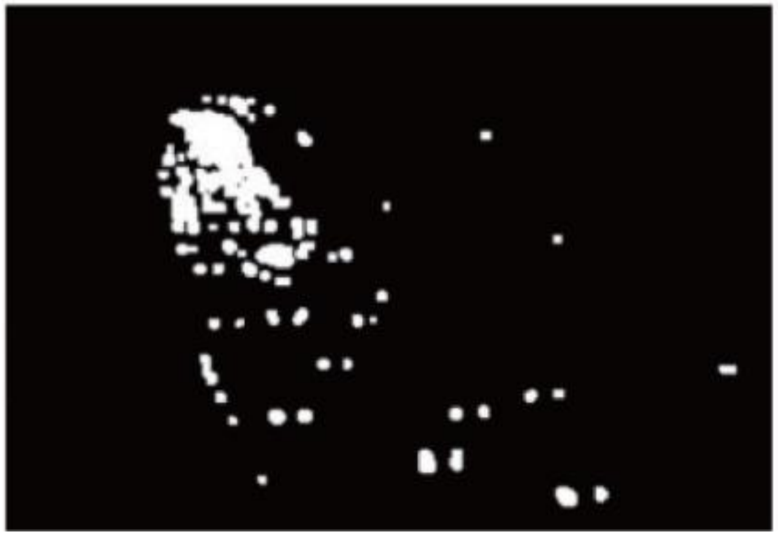

(threshold = 180)

Thresholding 的應用

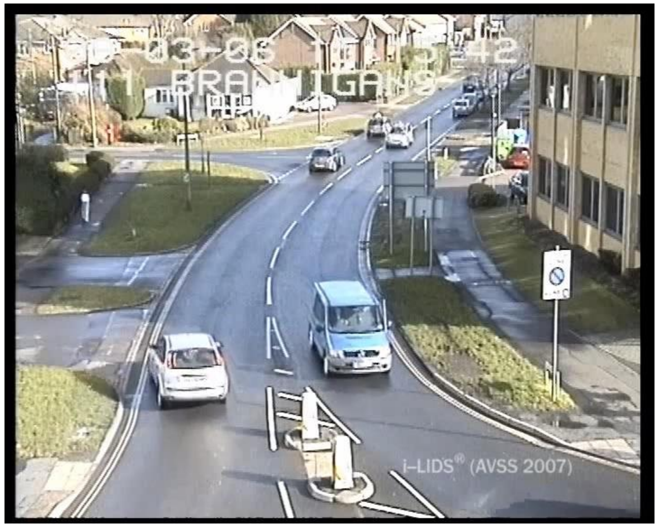


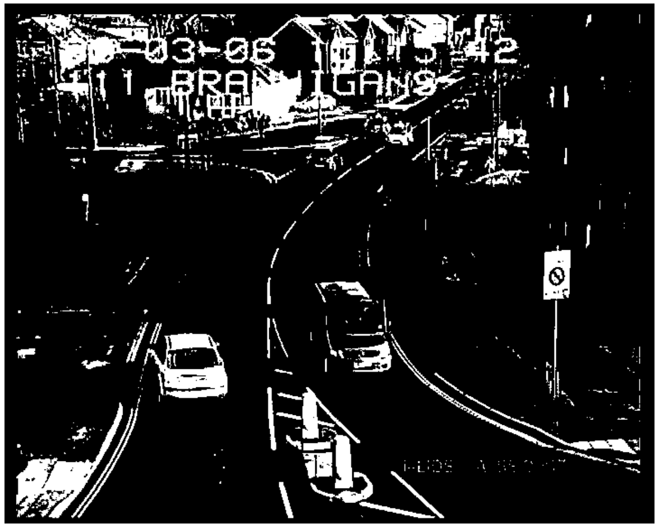
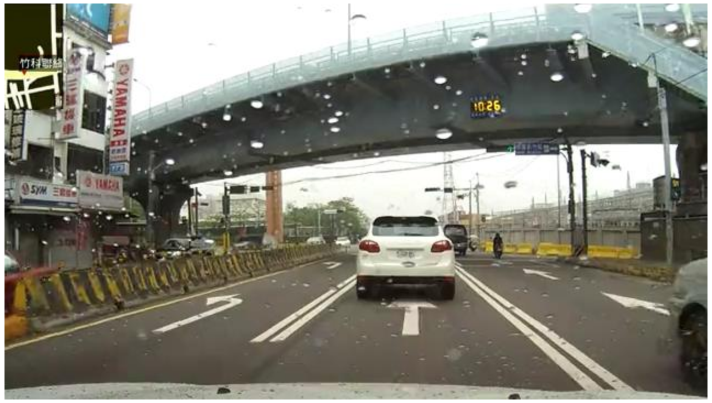


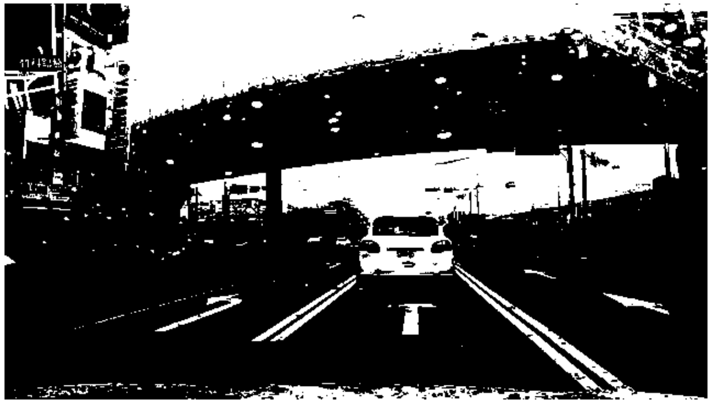




Thresholding 的應用

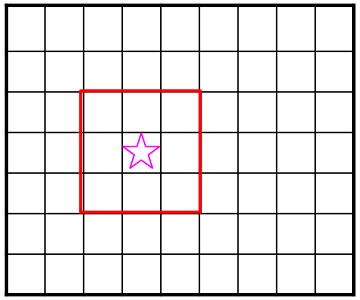




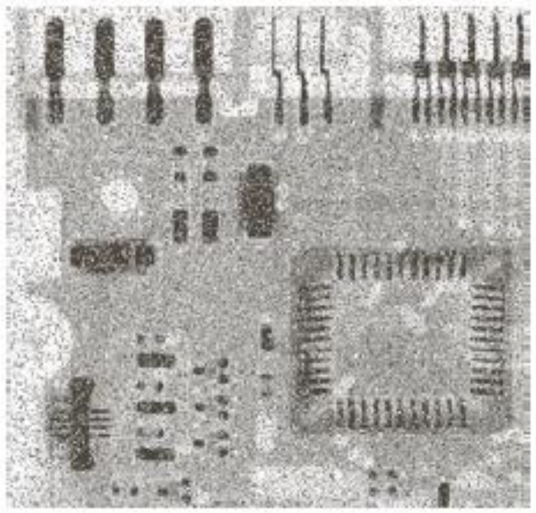
4. 平滑空間濾波器

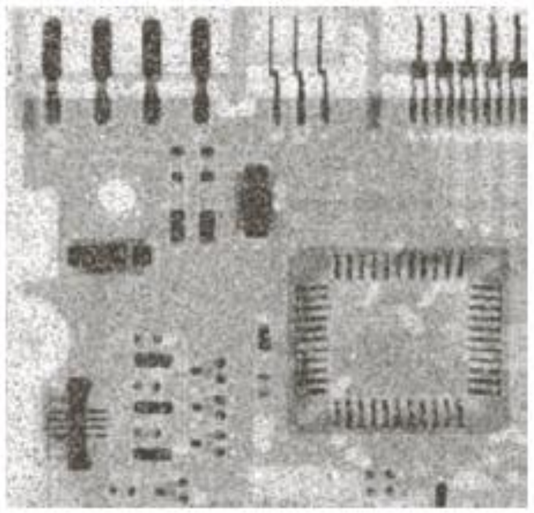
23, 23, 24, 25 ,25, 27, 27, 30, 235
Median = 25
Avg = 48.8



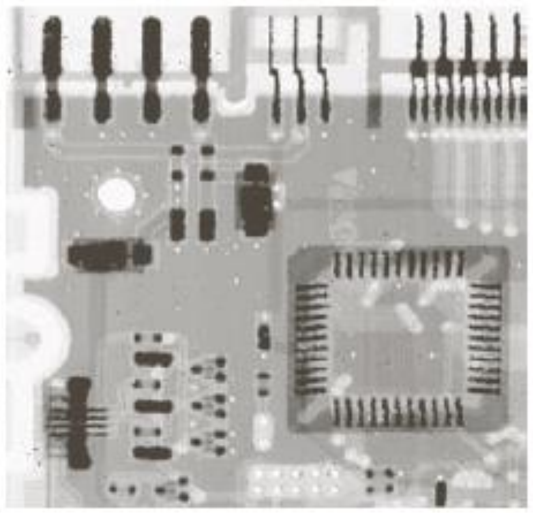
Non-Linear Filter

- 中值濾波器(median filter)
- Find the median number in a sequence of pixels in a filter
- Remove **pepper and salt noise**





3x3 average filter



3x3 median filter

Image Blur - Median

- The median filter replaces each pixel by the median or “middle” pixel (as opposed to the mean pixel) value in a square neighborhood around the center pixel.
- void **medianBlur**(InputArray **src**, OutputArray **dst**, int **ksize**)
  - **src** – input image.
  - **dst** – output image of the same size and type as src.
  - **ksize** – aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...
    It is **int** → **square kernal**

➔ cv::medianBlur(image, result, 5);

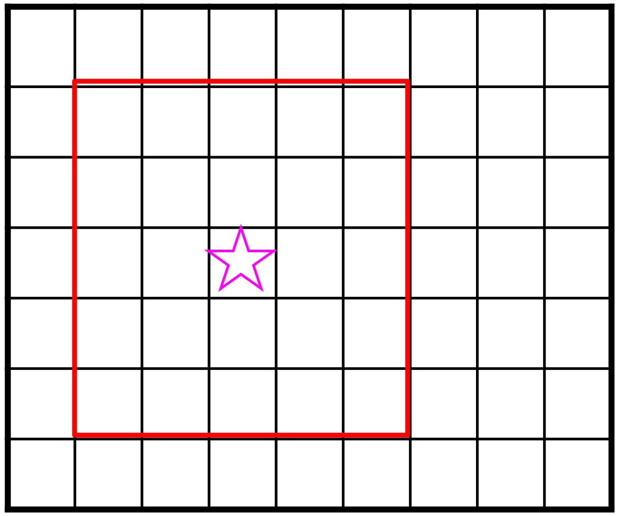
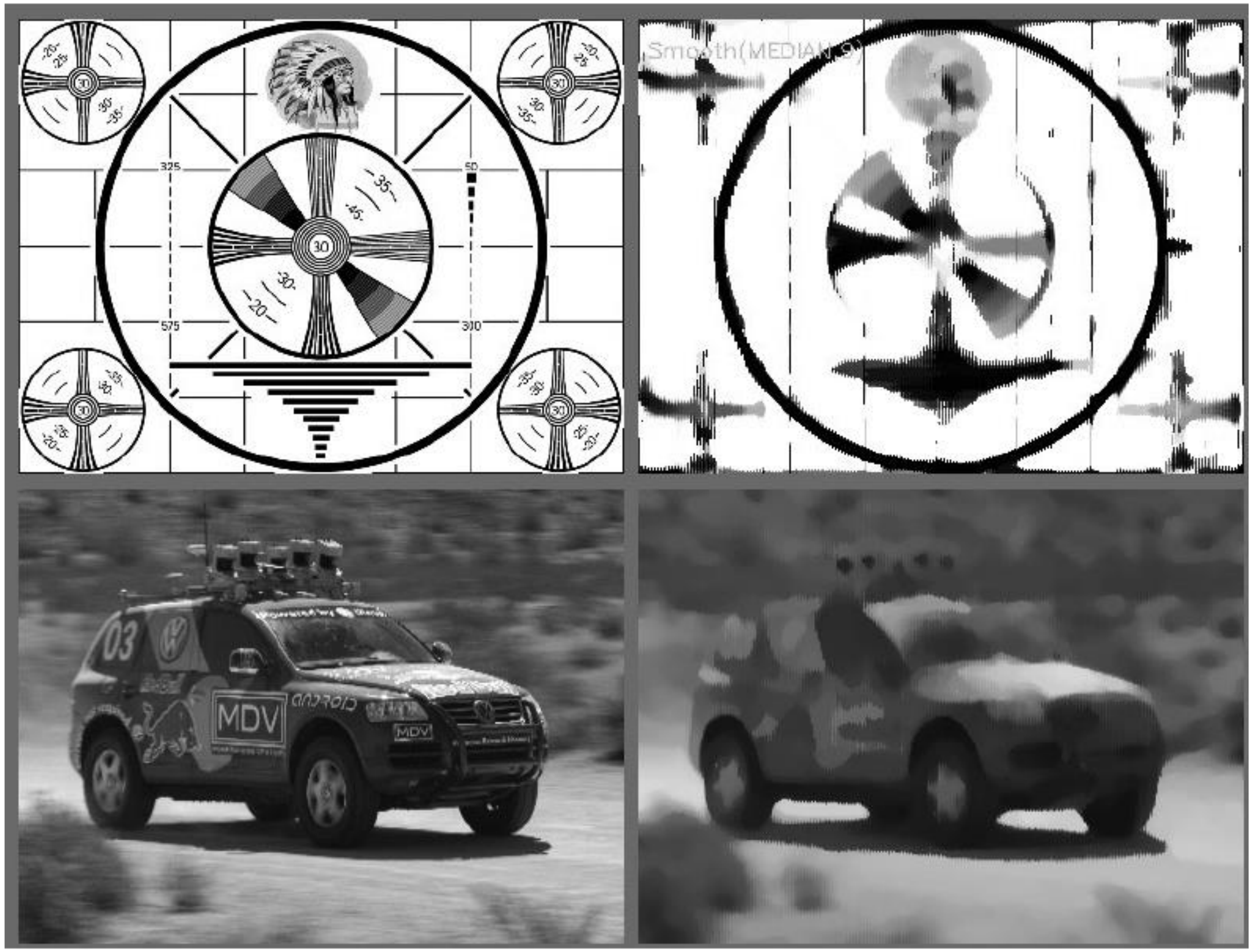


Image Blur - Median

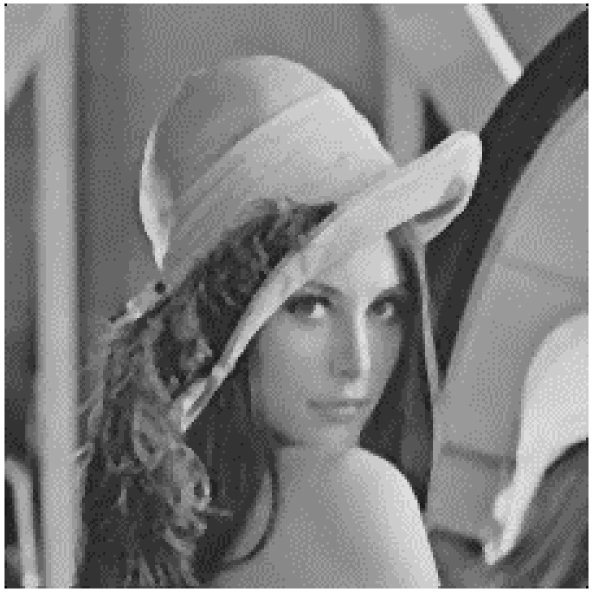


Median Blur

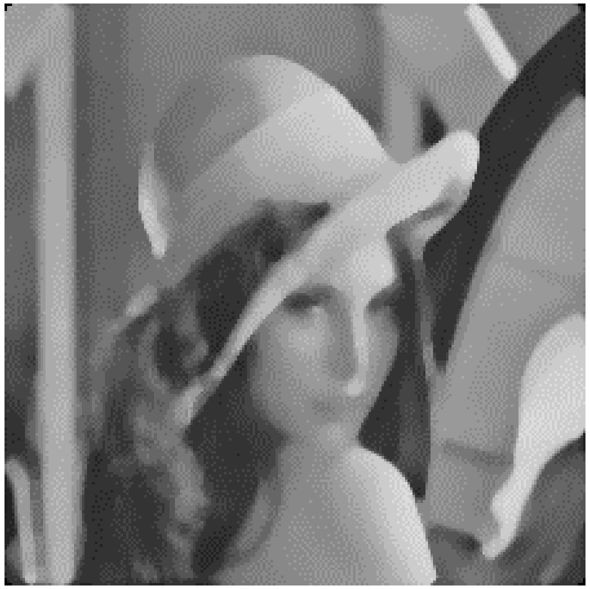
$$\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 2 & 3 \\ \hline 1 & 1 & 1 \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 3 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

1 1 1 1 1 1 1 2 3

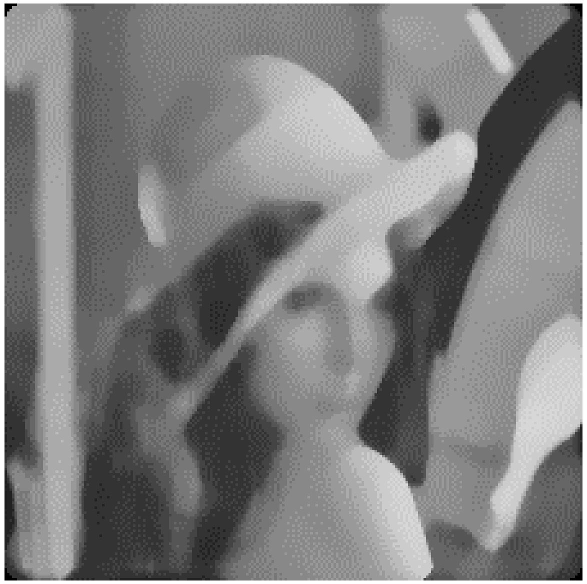
Medium number= 1



3*3



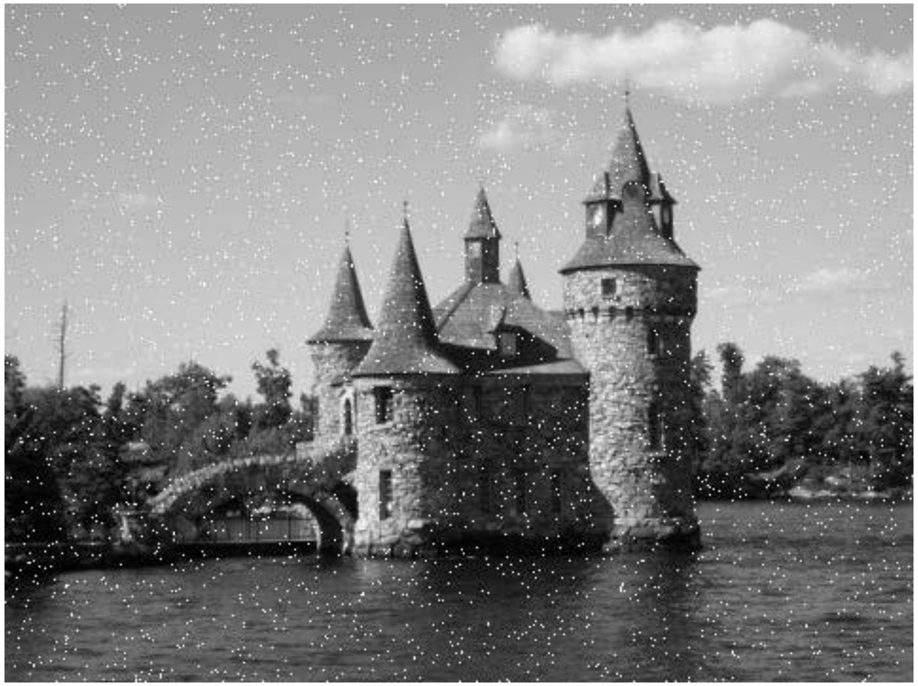
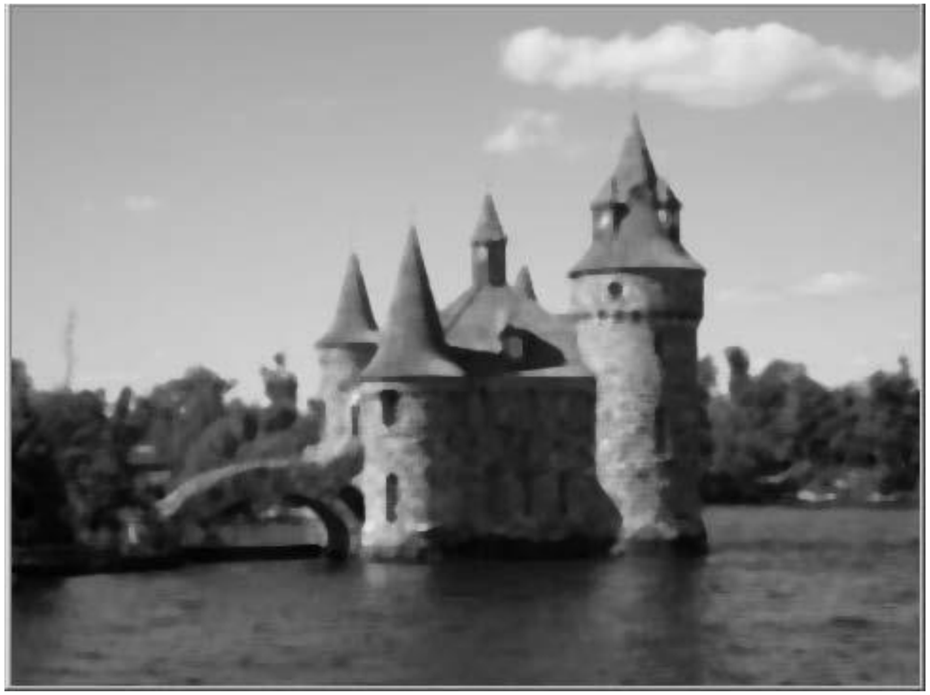
7*7



11*11

Image Blur - Median

- Simple blurring by averaging can be sensitive to image noise, especially large isolated outlier points.
- Median filtering is able to ignore the outliers by selecting the middle points.
  - particularly useful to combat salt-and-pepper noise

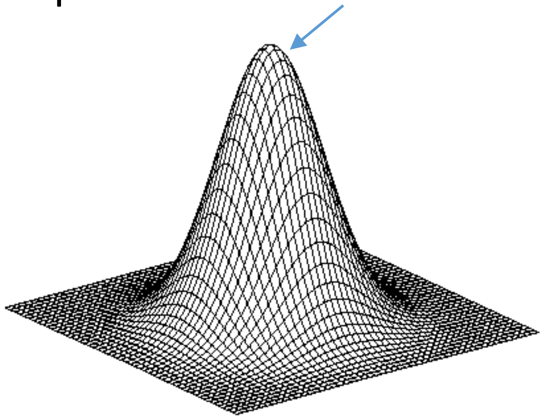



OpenCV Functions

Image Blur – Gaussian filter

- Gaussian filtering is probably the most useful.
  - convolve each point in the input array with a Gaussian kernel and then summing to produce the output array.

The pixels closer to the center are weighted more.



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$\frac{1}{273}$

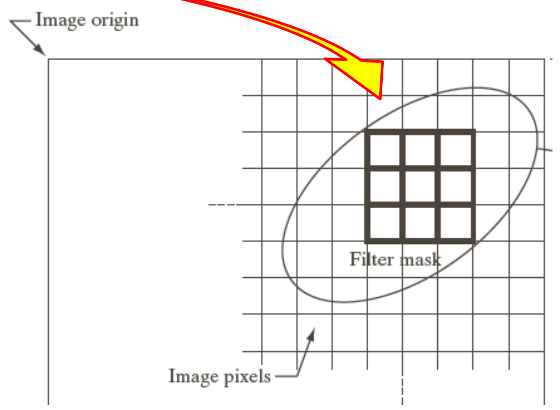
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

5x5 kernel, $\sigma_x = \sigma_y = 1$

Recall：Smoothing Filter

- Blurry, reduce noise
- Averaging filter is a linear filter

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

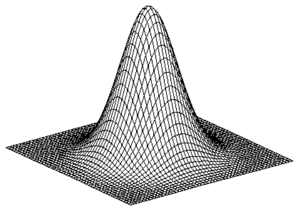


Normalization

weighted average:
The center point is much more important

Image Blur – Gaussian filter

- void GaussianBlur(InputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int **borderType**=BORDER_DEFAULT)
  - **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.
    - if sigmaY is zero, it is set to be equal to sigmaX
    - if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively.



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Image Blur – Gaussian filter

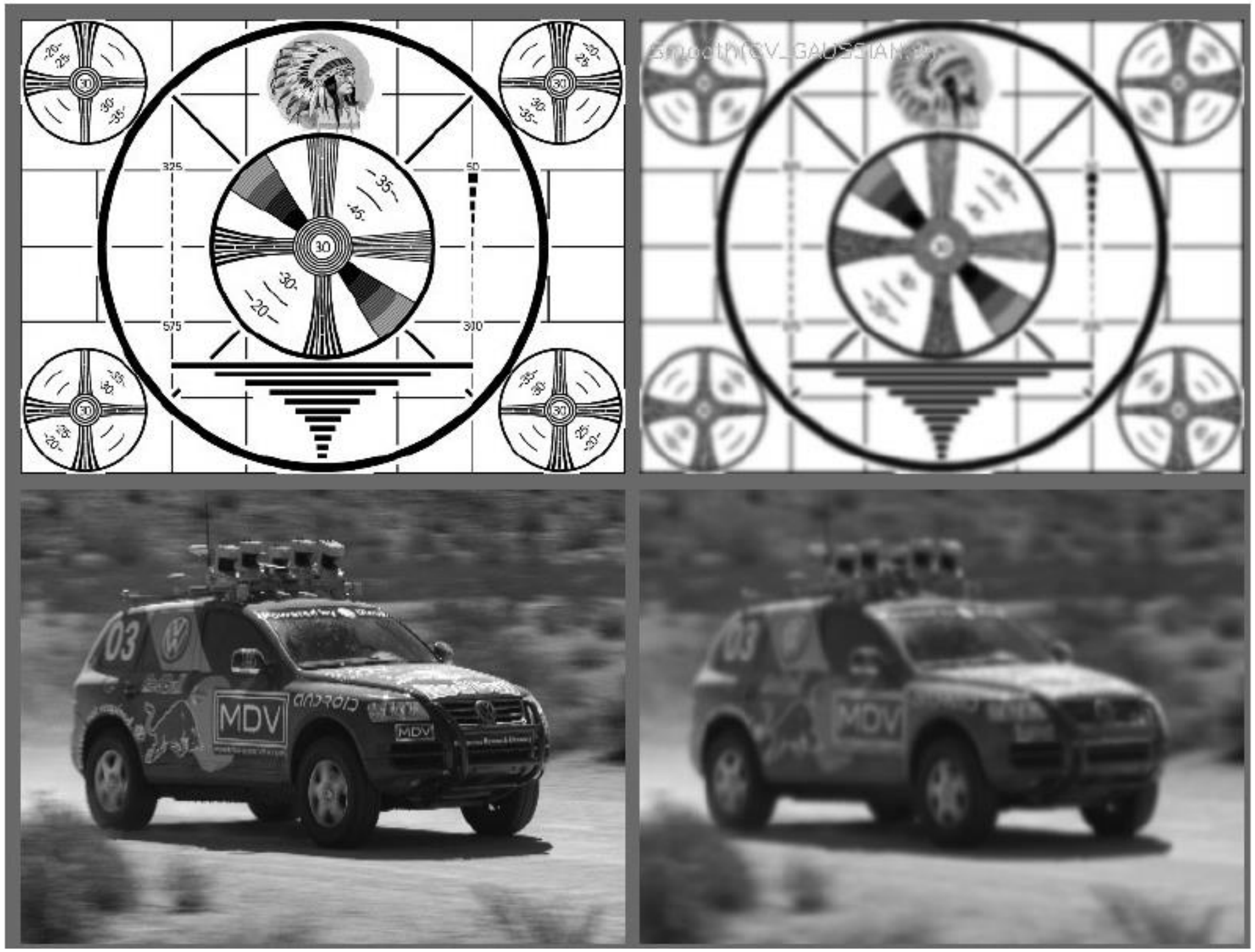
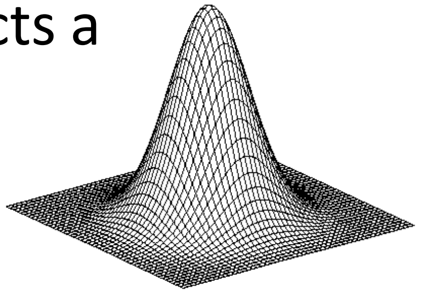


Image Blur – Bilateral filter

- Bilateral filtering →edge-preserving smoothing.
- 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.

  →Gaussian smoothing smoothes away the edges.

Image Blur – Bilateral filter



- Like Gaussian smoothing, bilateral filtering constructs a weighted average of each pixel and its neighboring components.
- The weighting has two components:
  - the first is the same weighting used by Gaussian smoothing, based on the spatial distance from the center pixel
  - The second is also a Gaussian weighting but is based on the difference in intensity from the center pixel.
- You can think of bilateral filtering as Gaussian smoothing that weights more similar pixels more highly 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 the image.

http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

Comparison – Gaussian filter

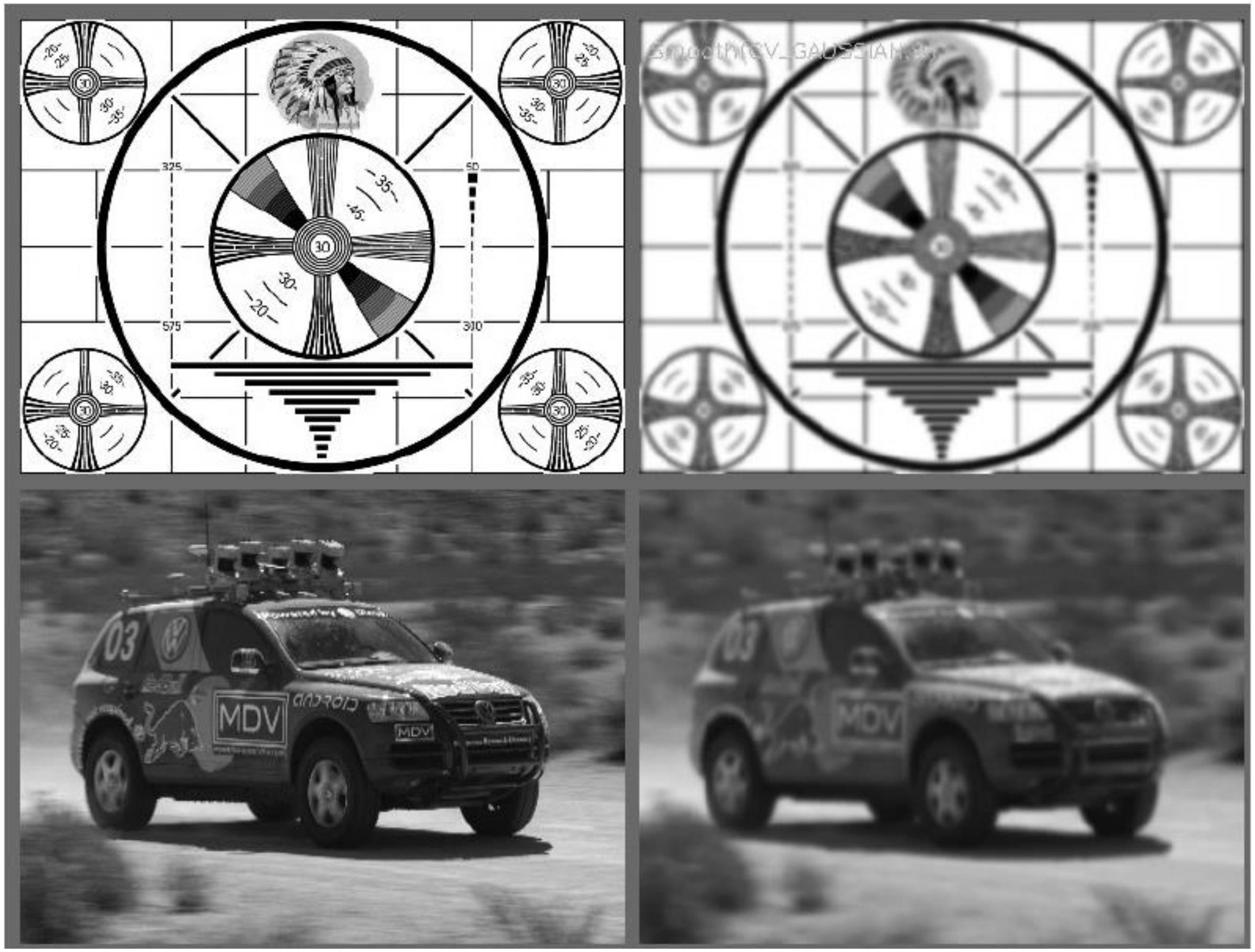


Image Blur – Bilateral filter

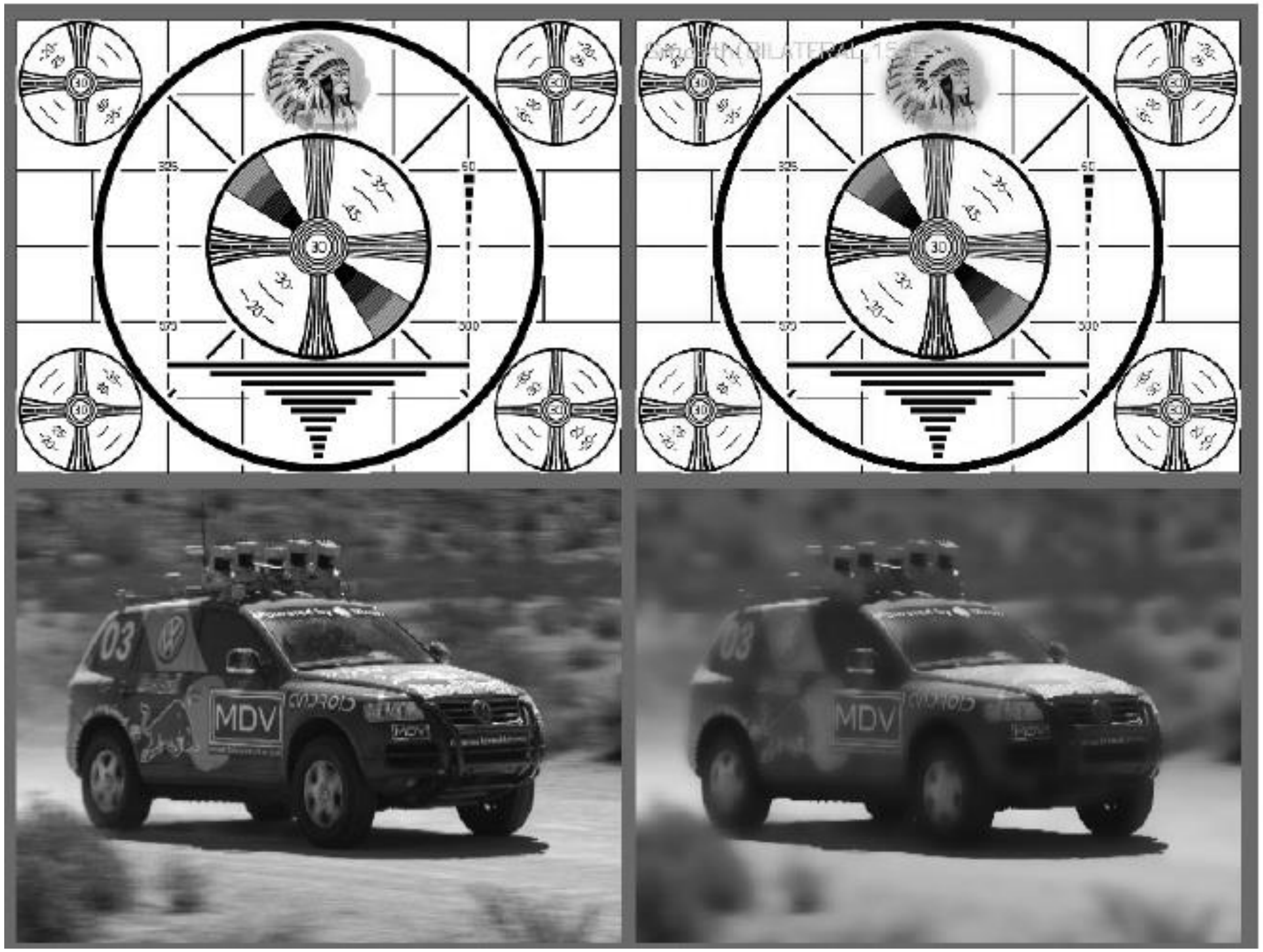
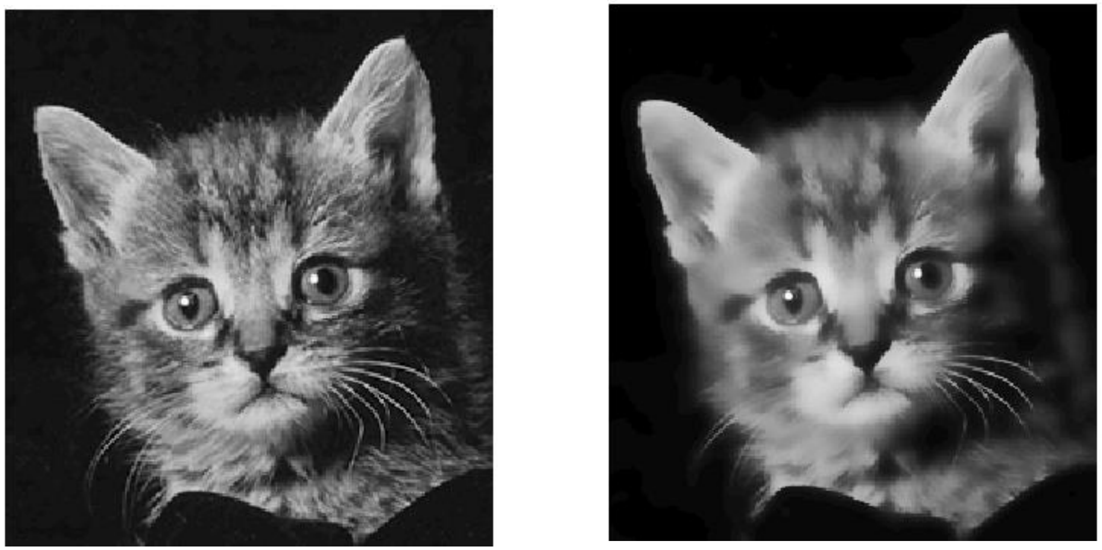


Image Blur – Bilateral filter



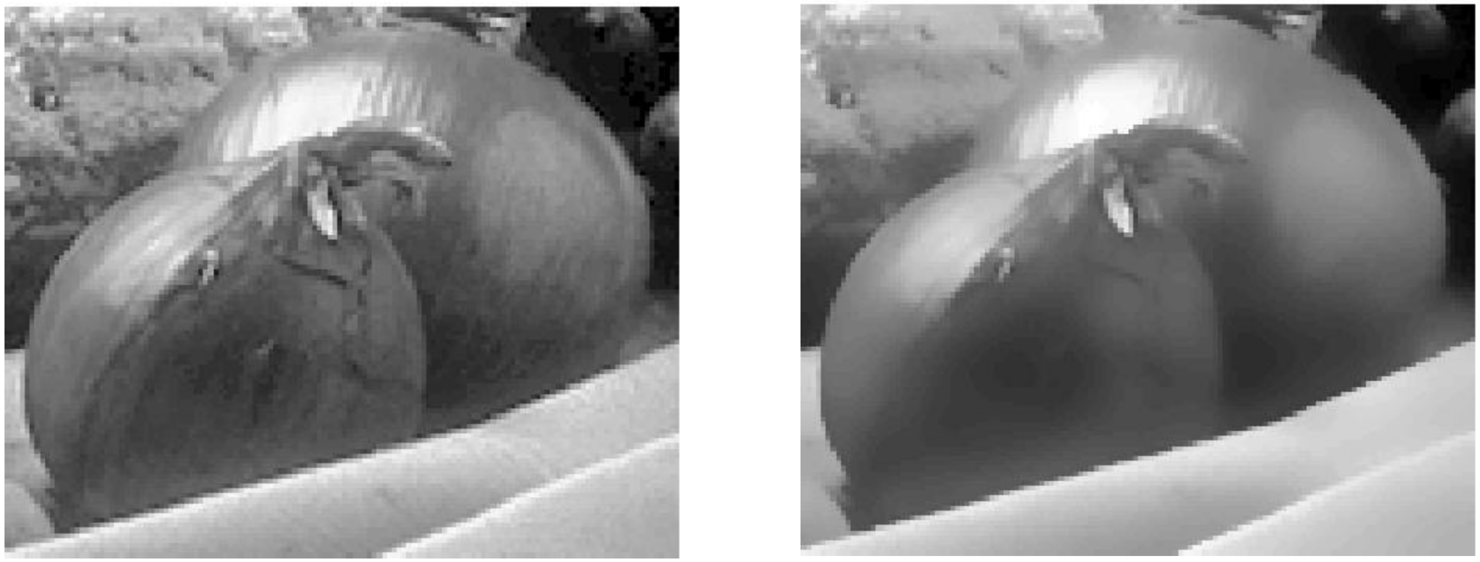
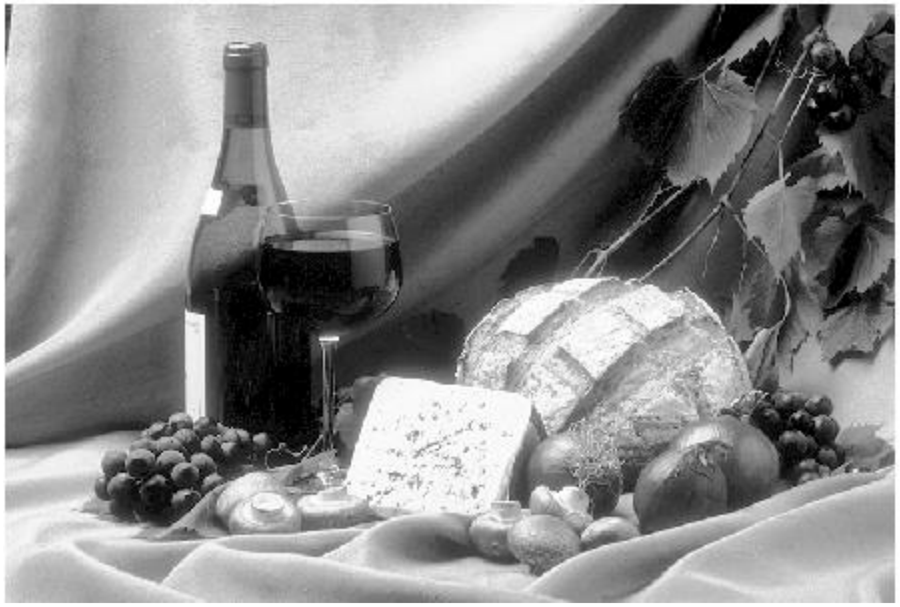
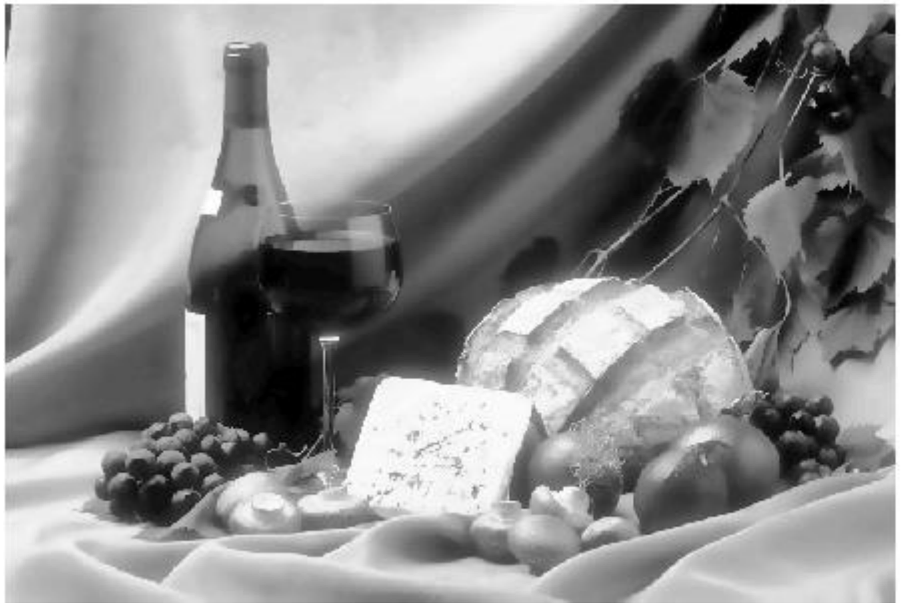


Image Blur – Bilateral filter





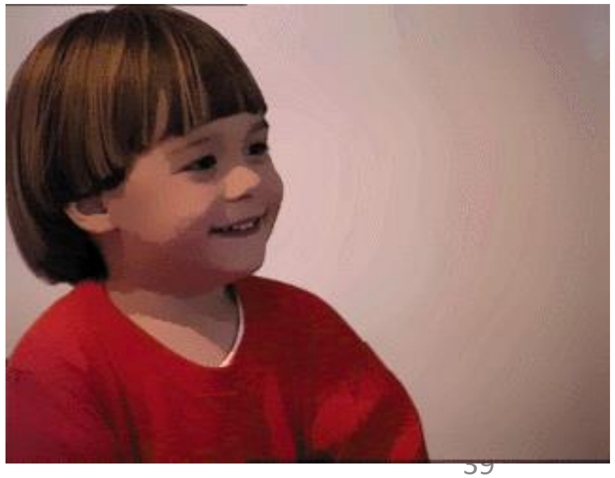
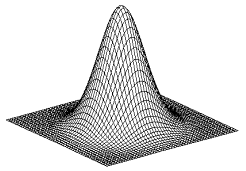


Image Blur – Bilateral filter



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

- void **bilateralFilter**(InputArray **src**, OutputArray **dst**, int **d**, double **sigmaColor**, double **sigmaSpace**, int **borderType**=BORDER_DEFAULT)
  - **d** – Diameter of each pixel neighborhood that is used during filtering.

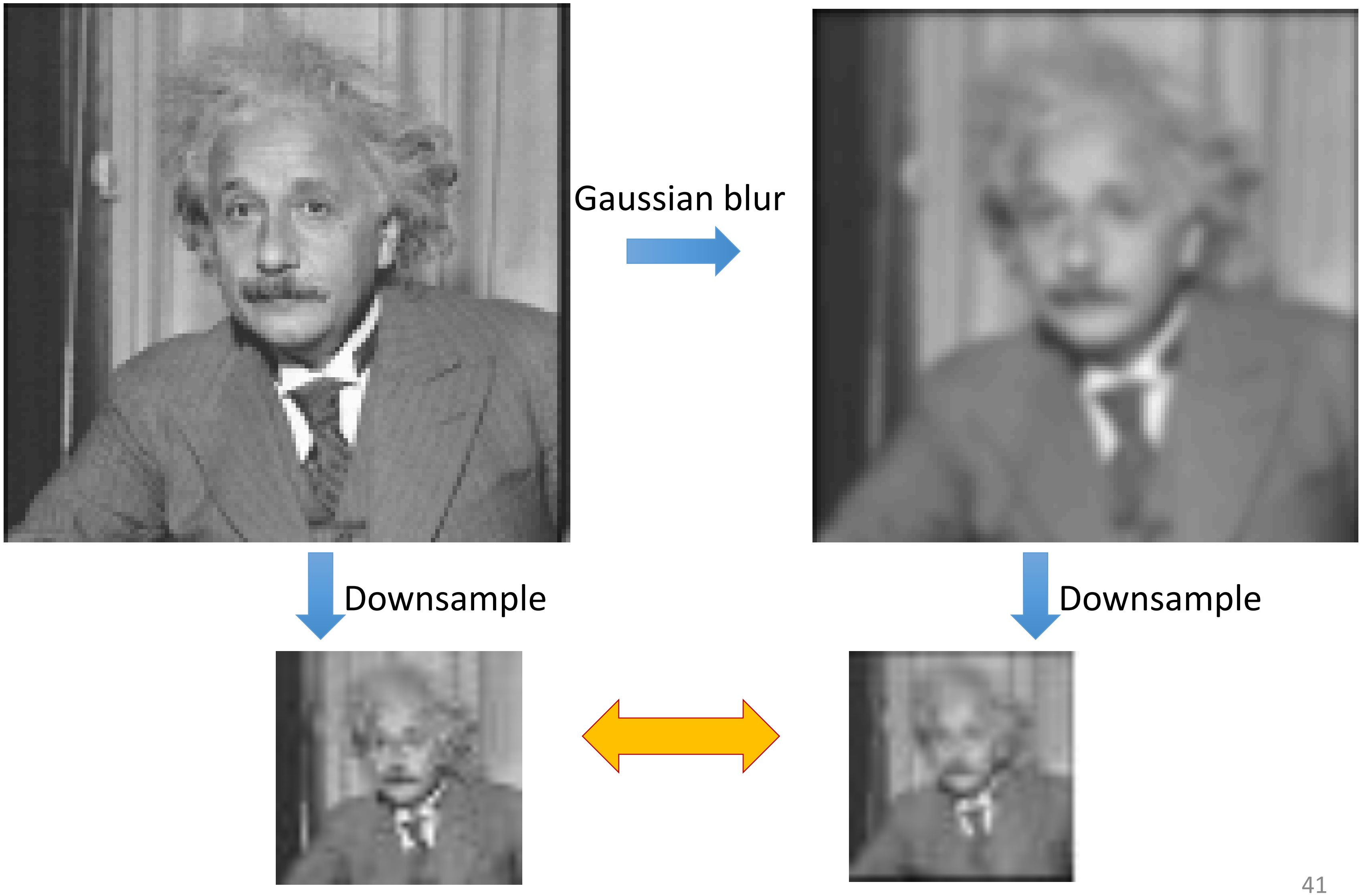
  →Large filters (d > 5) are very slow. Recommend: use d=5 for real-time applications.

  - **sigmaColor** – Filter sigma in the color space. A larger value of the parameter means that farther colors within the pixel neighborhood will be mixed together, resulting in larger areas of semi-equal color.
  - **sigmaSpace** – Filter sigma in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough.

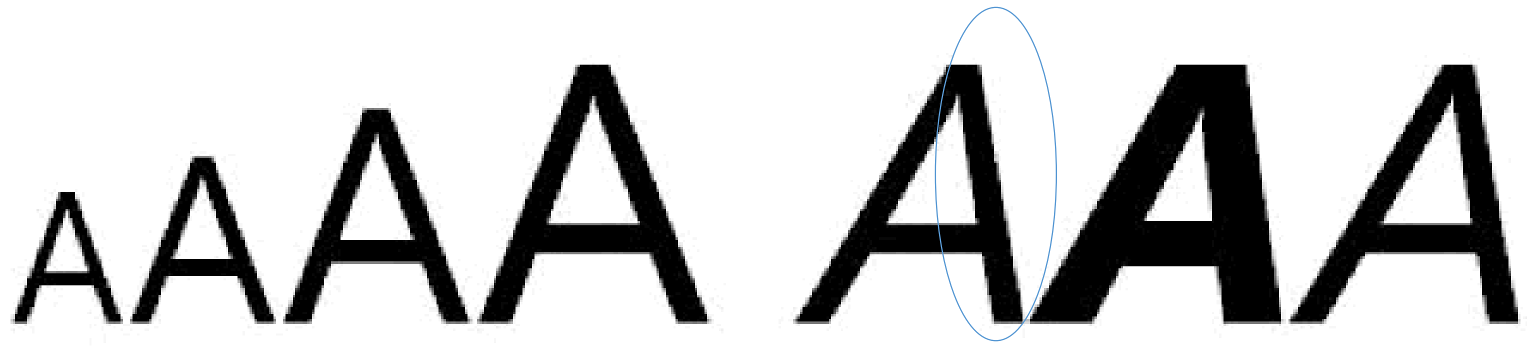
  →For simplicity, you can set the 2 sigma values to be the same.

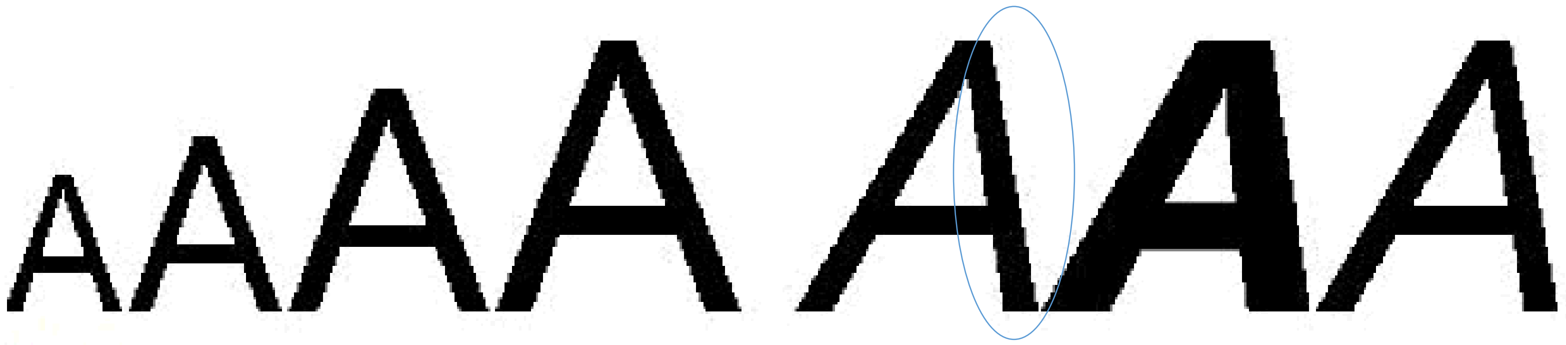
  If they are small (< 10), the filter will not have much effect, whereas if they are large (> 150), they will have a very strong effect, making the image look "cartoonish".

Some applications

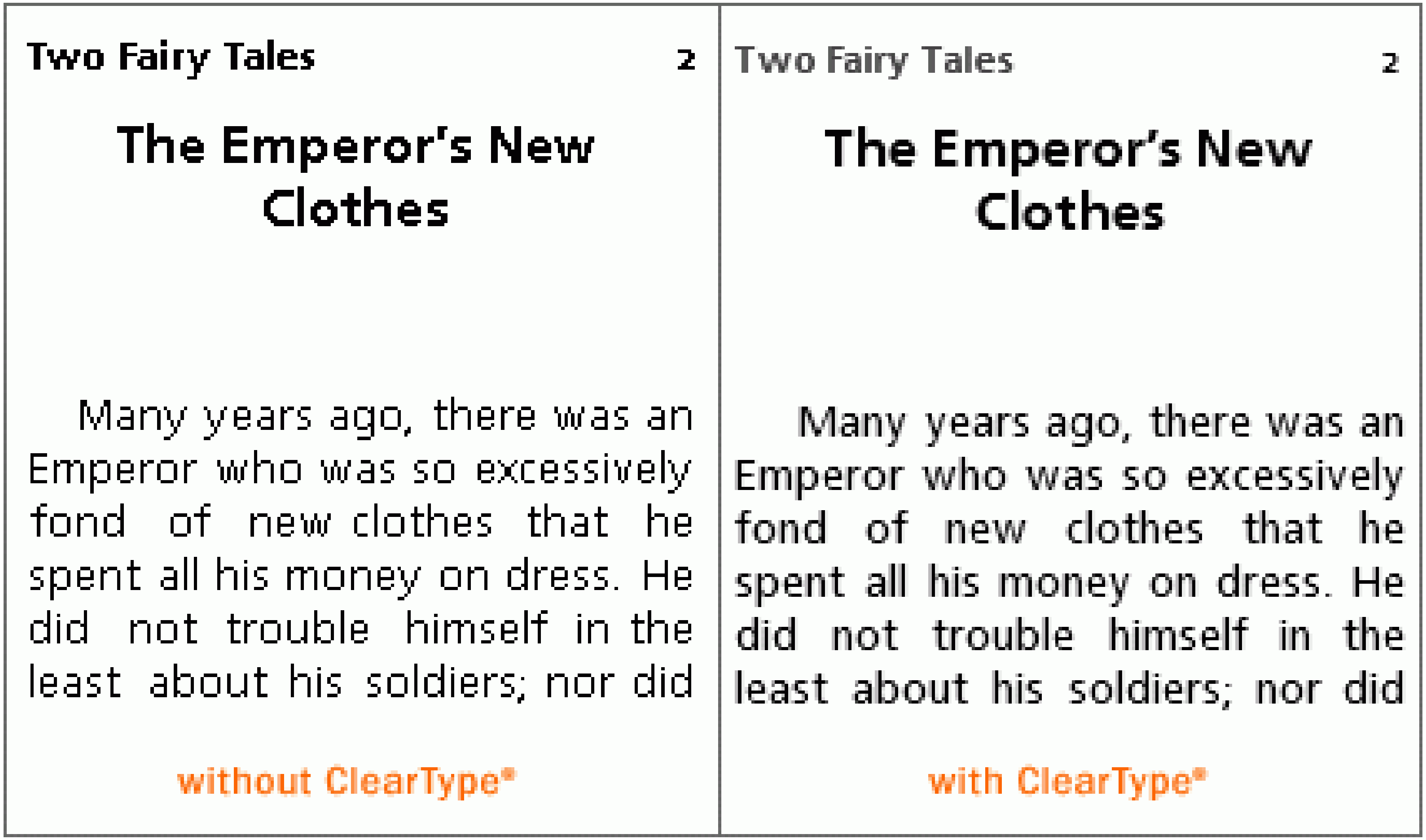


Clear Type Font


ClearType


None

Clear Type Font


Two Fairy Tales 2

The Emperor's New Clothes

Many years ago, there was an Emperor who was so excessively fond of new clothes that he spent all his money on dress. He did not trouble himself in the least about his soldiers; nor did

without ClearType®

Two Fairy Tales 2

The Emperor's New Clothes

Many years ago, there was an Emperor who was so excessively fond of new clothes that he spent all his money on dress. He did not trouble himself in the least about his soldiers; nor did

with ClearType®

- Q&A