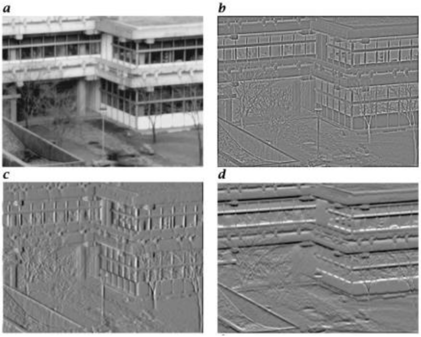
Acknowledgements

Dr Nazar Khan (PUCIT)
Ms Adeela Islam (PhD Scholar PUCIT)
Dr Rawat (UCF)

Image Filtering Applications

- Uses of filtering:
  - Enhance an image (denoise, sharp, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Example



a. Original image
b. Laplacian operator
c. Horizontal derivative
d. Vertical derivative

Correlation (**)

If we have

- *Image* = $f(x, y)$
- *Kernel* = $h(x, y)$

$$
\begin{array}{c}
f \\
\begin{array}{|c|c|c|}
\hline
f_1 & f_2 & f_3 \\
\hline
f_4 & f_5 & f_6 \\
\hline
f_7 & f_8 & f_9 \\
\hline
\end{array}
\end{array}
\;**\;
\begin{array}{c}
h \\
\begin{array}{|c|c|c|}
\hline
h1 & h2 & h3 \\
\hline
h4 & h5 & h6 \\
\hline
h7 & h8 & h9 \\
\hline
\end{array}
\end{array}
$$

Then correlation is

$f **h = f_1h_1 + f_2h_2 + f_3h_3 + f_4h_4 + f_5h_5 + f_6h_6 + f_7h_7 + f_8h_8 + f_9h_9$

(dot product)

Correlation (**)

- Compares the similarity of two sets of data
- The correlation result reaches a maximum at the time when the two signals match best
- It is the measure of relatedness of two products

Template Matching

- Correlation can also be used for *matching*
- If we want to determine whether an image f(x, y) contains a particular object, we let h(x, y) be that object (also called a *template*) and compute the correlation between f and h
- If there is a match, the correlation will be maximum at the location where h finds a correspondence in f

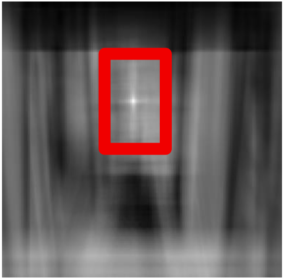
Chair detection using template matching (Naïve approach)

This is a chair h(x,y)

Find the chair in this image f(x,y)

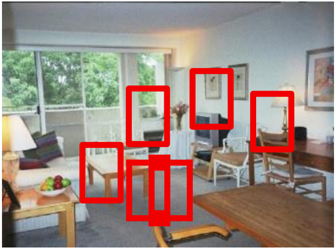
7

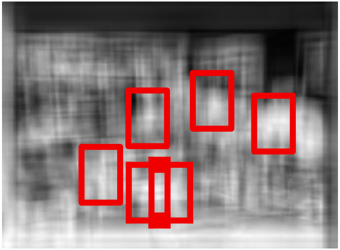
Output of correlation



Template Matching

Find the chair in this image





Epic fail!
Simple template matching is not going to make it

Convolution (*)

Same as correlation but *kernel flipped horizontally and vertically*

- *Image* = $f(x, y)$
- *Kernel* = $h(x, y)$

h

h1	h2	h3
h4	h5	h6
h7	h8	h9

flipped horizontally and vertically

f

f_1	f2	f3
f4	f5	f6
f7	f8	f9

$*$

h9	h8	h7
h6	h5	h4
h3	h2	h1

$$f * h = f_1 h_9 + f_2 h_8 + f_3 h_7 + f_4 h_6 + f_5 h_5 + f_6 h_4 + f_7 h_3 + f_8 h_2 + f_9 h_1$$

Convolution (*)

- It can be explained as the "mask/kernel convolved with an image".

$$f'(x, y) = h(x, y) * f(x, y)$$

- Or it can be explained as "image convolved with mask/kernel".

$$f'(x, y) = f(x, y) * h(x, y)$$

What is mask/kernel?

- It can be represented by a two-dimensional matrix
- The mask is usually of the order of 1×1, 3×3, 5×5, 7×7
- A mask should always be in odd number, because other wise you cannot find the **mid of the mask.**

How to perform convolution?

In order to perform convolution on an image, following steps should be taken

1. Flip the mask (horizontally and vertically) only once

2. Slide the mask onto the image

3. Multiply the corresponding elements and then add them (dot product)

4. Repeat this procedure until all values of the image has been calculated

How to perform convolution?

Mask

1	2	3
4	5	6
7	8	9

How to perform convolution?

Flip the mask horizontally

3	2	1
6	5	4
9	8	7

How to perform convolution?

Flip the mask vertically

9	8	7
6	5	4
3	2	1

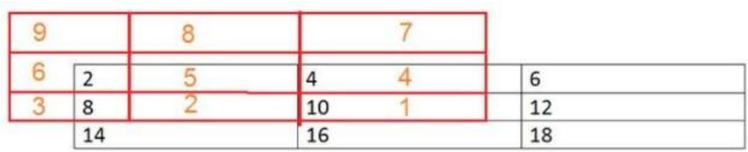
How to perform convolution?

Image

2	4	6
8	10	12
14	16	18

How to perform convolution?

Slide the mask onto the image and multiply the corresponding elements and then add them



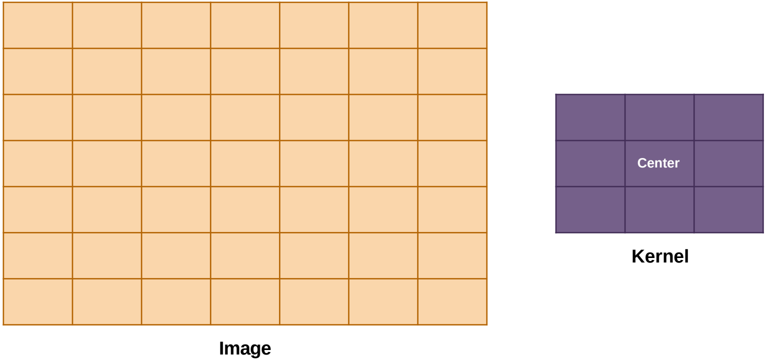
First pixel = (5*2) + (4*4) + (2*8) + (1*10)
= 10 + 16 + 16 + 10
= 52
Place 52 in the original image at the first index and repeat this procedure for each pixel of the image.

Dealing with Boundaries

- Padding with Zeros
- Copy boundary values
- Ignore boundaries

Convolution Animation

Performing Convolution on a 7×7 image with a 3×3 kernel



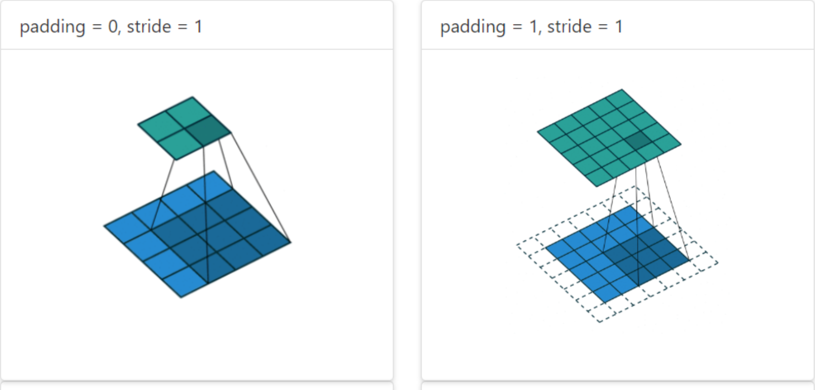
Convolution

Dealing with Boundaries (Padding with Zeros):

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

What if the filter size was 5×5 *instead of* 3×3*?*

Convolution Animation



Demo: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

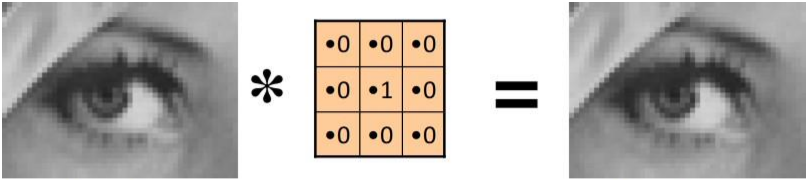
Convolution in 2D - Examples

Original

$$* \begin{bmatrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 1 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{bmatrix} = \ ?$$

Courtesy: D Lowe

Convolution in 2D - Examples



Original

Filtered
(no change)

Courtesy: D Lowe

Convolution in 2D - Examples

Original

$*$

•0	•0	•0
•0	•0	•1
•0	•0	•0

$=$?

Courtesy: D Lowe

Convolution in 2D - Examples

Original

*

•0	•0	•0
•0	•0	•1
•0	•0	•0

=

Shifted right
By 1 pixel

Courtesy: D Lowe

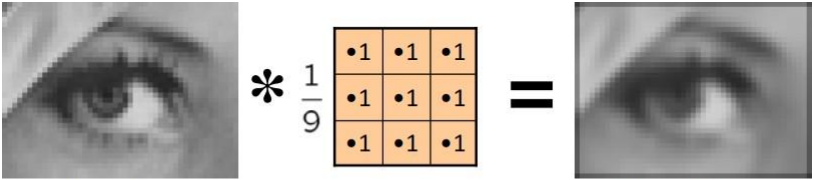
Convolution in 2D - Examples

Original

$$\ast \ \frac{1}{9} \begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix} = \ ?$$

Courtesy: D Lowe

Convolution in 2D - Examples



Original

Blur (with a box filter)

Courtesy: D Lowe

Separability

- A 2D filter is separable if it can be written as the product of a "column" and a "row"

example: box filter

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \underset{\text{column}}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} * \underset{\text{row}}{\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}}$$

- 2D convolution with a separable filter is equivalent to two 1D convolutions:
  1. First convolve the image with a one-dimensional horizontal filter
  2. Then convolve the result of the first convolution with a one-dimensional vertical filter

Separability

- **Why is separability useful?**

  If the image has $M \times M$ pixels and the filter kernel has size $N \times N$:

  - Cost (multiplication) of convolution with a non-separable filter: $= M^2 \times N^2$
  - Cost (multiplication) of convolution with a separable filter: $= 2 \times N \times M^2$

- Hence, it is computationally much cheaper

Separability Example

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} \quad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} \quad \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

$$\begin{aligned} &= 2 + 6 + 3 = 11 \\ &= 6 + 20 + 10 = 36 \\ &= 4 + 8 + 6 = \underline{18} \\ &\qquad\qquad\quad\;\; 65 \end{aligned}$$

Convolution Masks

- Different masks (Box vs prewitt) lead to different effects
- **Low pass filters: (Smoothing)**
  Low pass filtering, is employed to remove high spatial frequency noise from a digital image
- **High pass filters: (Edge Detection, Sharpening)**
  A high-pass filter can be used to make an image appear sharper. These filters emphasize fine details in the image

Averages

- Mean

$$I = \frac{I_1 + I_2 + \ldots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

- Weighted mean

$$I = \frac{w_1 I_1 + w_2 I_2 + \ldots + w_n I_n}{n} = \frac{\sum_{i=1}^{n} w_i I_i}{n}$$

Smoothing Filters

- Averaging/Mean Filters (e.g., Box filter)
- Weighted Averaging Filters (e.g., Gaussian filter)

Box Filter

- Also known as the *averaging filter*

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

- Replaces pixel with local average
- All the pixels have same weight
- Has smoothing (blurring) effect
- size of mask determines extent of smoothing

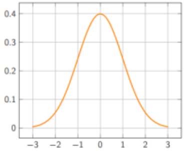
Gaussian Filter

A widely used mask for smoothing is the Gaussian mask, named after Carl Friedrich Gauss

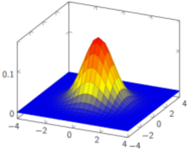
$$1D : g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

$$2D : G(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu_1)^2+(y-\mu_2)^2/2\sigma^2}$$

where μ is the $1D$ mean, (μ_1, μ_2) is the $2D$ mean and σ^2 is the variance.



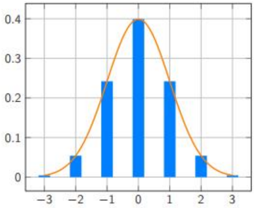
$\mu = 0, \sigma = 1$



$(\mu_1, \mu_2) = (0, 0), \sigma = 1$

Courtesy: N. Khan

Gaussian Kernel
1D Discrete approximation



$$\begin{bmatrix} 0.0044 & 0.054 & 0.242 & 0.399 & 0.242 & 0.054 & 0.0044 \end{bmatrix}$$

Slide credit: Dr Nazar

Gaussian Kernel
2D Discrete Approximation

$$\begin{bmatrix} 0.0000 & 0.0002 & 0.0011 & 0.0018 & 0.0011 & 0.0002 & 0.0000 \\ 0.0002 & 0.0029 & 0.0131 & 0.0215 & 0.0131 & 0.0029 & 0.0002 \\ 0.0011 & 0.0131 & 0.0585 & 0.0965 & 0.0585 & 0.0131 & 0.0011 \\ 0.0018 & 0.0215 & 0.0965 & 0.1592 & 0.0965 & 0.0215 & 0.0018 \\ 0.0011 & 0.0131 & 0.0585 & 0.0965 & 0.0585 & 0.0131 & 0.0011 \\ 0.0002 & 0.0029 & 0.0131 & 0.0215 & 0.0131 & 0.0029 & 0.0002 \\ 0.0000 & 0.0002 & 0.0011 & 0.0018 & 0.0011 & 0.0002 & 0.0000 \end{bmatrix}$$

Slide credit: Dr Nazar

Gaussian Kernel
2D Discrete approximation

$$\begin{bmatrix} 0.0044 & 0.054 & 0.242 & 0.399 & 0.242 & 0.054 & 0.0044 \end{bmatrix} * \begin{bmatrix} 0.0044 \\ 0.054 \\ 0.242 \\ 0.399 \\ 0.242 \\ 0.054 \\ 0.0044 \end{bmatrix}$$

Separability of Gaussian Kernels: Convolution with 2D Gaussian can be performed via two successive convolutions with 1D Gaussians which are computationally much cheaper.

Slide credit: Dr Nazar

Gaussian Filter

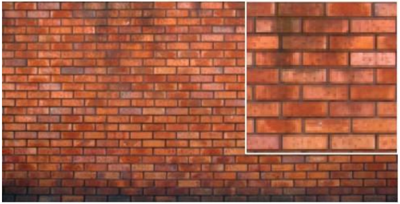
- Nearest neighboring pixels have the most influence on the output

- This kernel approximates a 2D Gaussian function

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

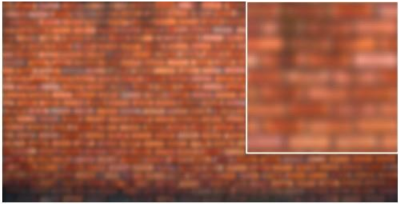
- Replaces pixel with weighted average of neighborhood
- Has smoothing (blurring) effect
- Size of mask and variance of Gaussian determines extent of smoothing

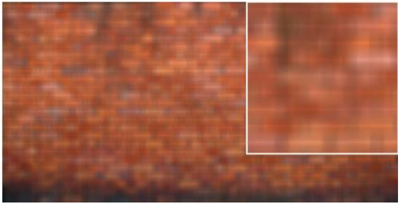
Gaussian vs. Box Filtering



original

Which blur do you like better?

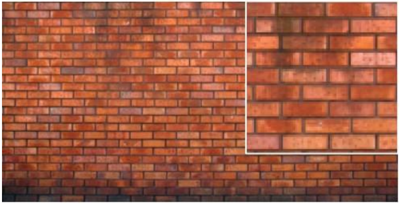




Gaussian vs. Box Filtering

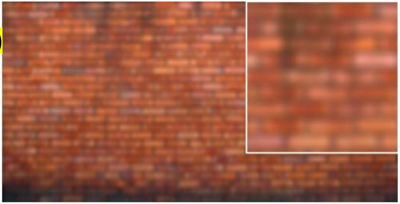
```
gaussian_kernel_1d = cv2.getGaussianKernel(kernel_size, sigma)
gaussian_kernel_2d = np.outer(gaussian_kernel_1d, gaussian_kernel_1d.T)
kernel = np.outer(probs, probs)
```

```
Box_output = cv2.filter2D(img, -1, kernel)
```

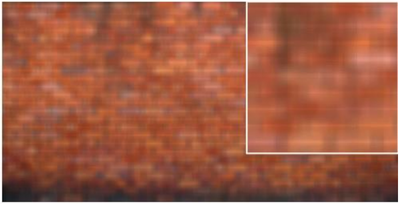


original

Which blur do you like better?



7x7 Gaussian



7x7 box

```
kernel = np.ones((7, 7),np.float32)/25
Box_output = cv2.filter2D(img, -1, kernel)
```

Noise Removal

Gaussian Noise

After Averaging

After Gaussian Smoothing

Demo
- adding noise
- smoothing with avg and gaussian filter

Sharpening Filters

- The sharpen kernel emphasizes differences in adjacent pixel values. This makes the image look more vivid.
- For a 3x3 mask, the simplest arrangement is as below

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

- When the mask is over a constant or slowly varying region the output is zero or very small

Sharpening Filters

https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

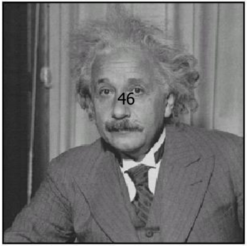
- Accentuates differences with local average

Homework:3

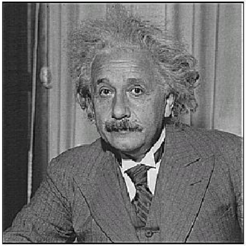
1) **2*I – neighborhood operation avg(I)**

2) Apply previously slide kernel on an image

Sharpening Filter

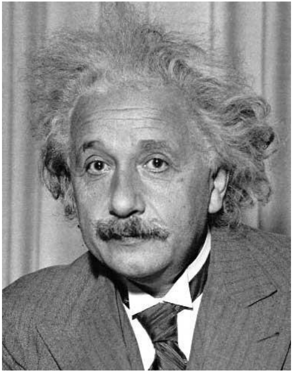


before



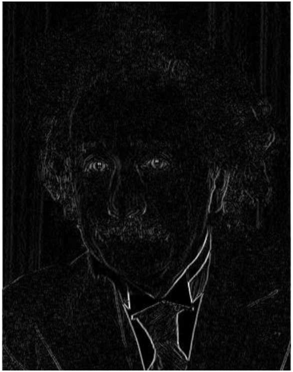
after

Sobel Filtering



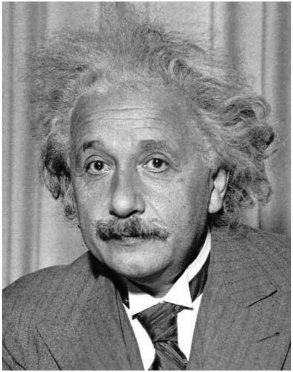
1	0	-1
2	0	-2
1	0	-1

Sobel



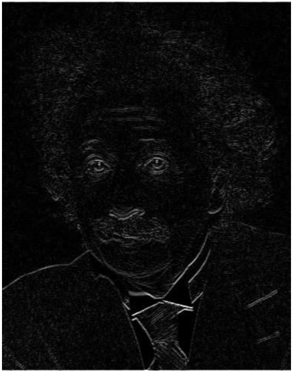
Vertical Edge
(absolute value)

Sobel Filtering



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

Sobel



Horizontal Edge
(absolute value)

Key properties of linear filters

Linearity:

`filter(f`$_1$` + f`$_2$`) = filter(f`$_1$`) + filter(f`$_2$`), as` **`did in sharping filter`**

Shift invariance: same behavior regardless of pixel location

Any linear, shift-invariant operator can be represented as a convolution

Non-linear Filtering

- Any filtering performed via convolution is **linear filtering**
- **Non-linear filtering** yields additional benefits
  - Median filtering
  - Bilateral filtering
  - Non-local means

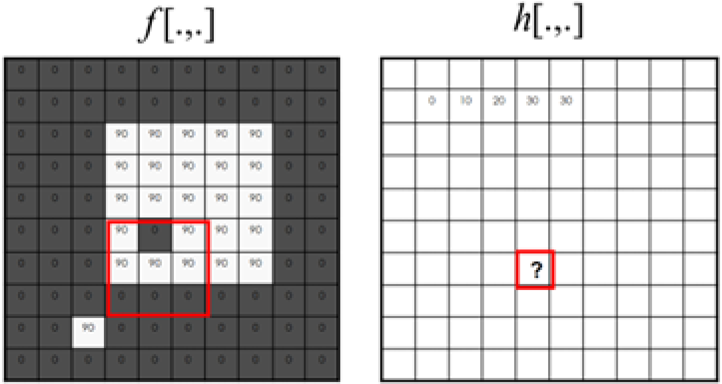
Credits: N. Khan

Median Filter

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- Advantage?
- Is it same as convolution?

Image filtering - mean

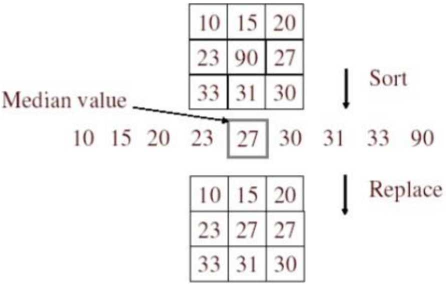
$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$



k,l

Credit: S. Seitz

Median filter



- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

Credits: K. Grauman

Image filtering - mean

$$g[\cdot,\cdot] \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				50					

Image filtering - median

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

Image filtering - median

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				90					

Median Filter