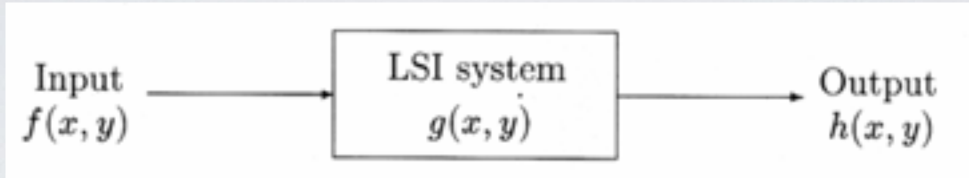
IMAGE FILTERING

INEL 6088 Computer Vision

References: Chapter 4 Jain et al, Chapter 3 Davies

CONVOLUTION



1D: $$f(x) \star g(x) = \int_{-\infty}^{+\infty} f(u)g(x-u)du \Rightarrow f(i) \star g(i) = \sum_{k=1}^{m} f(k)g(i-k)$$

2D, discrete

$$\begin{bmatrix} h4 & h3 & h2 \\ h5 & h0 & h1 \\ h6 & h7 & h8 \end{bmatrix}$$

mask

$$F(x, y) = f(x, y) * g(x, y) = \sum_i \sum_j f(i, j)g(x-i, y-j)$$

For convenience, pre-calculate $h(x, y) = g(-x, -y)$

$$F(x, y) = \sum_i \sum_j f(x+i, y+j)h(i, j)$$

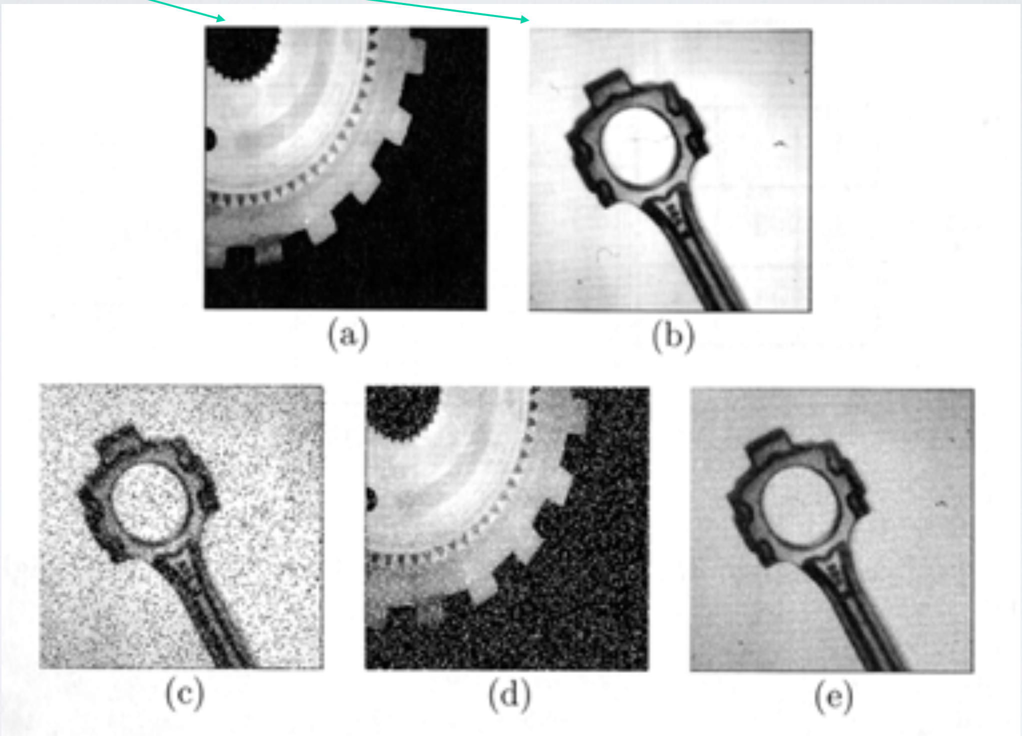
$$CONV : [[q_0 = p_0h_0 + p_1h_1 + p_2h_2 + p_3h_3 \\ +p_4h_4 + p_5h_5 + p_6 + h_6 + p_7h_7 + p_8h_8]]$$

CHARACTERISTICS OF CONVOLUTION

- Linear (convolution of a sum is the sum of the convolutions)
- Convolution of a scaled image is the scaled convolution
- Spatially invariant
- Convolution in the image domain correspond to multiplication in the (spatial) frequency domain
- Frequency-domain convolution used only for very large kernels – not common in machine vision
- The kernel defines the filter being used
- We use non-linear filters as well; formalism does not apply but many are used in a very similar way.

Common Types of Noise

Noise-free

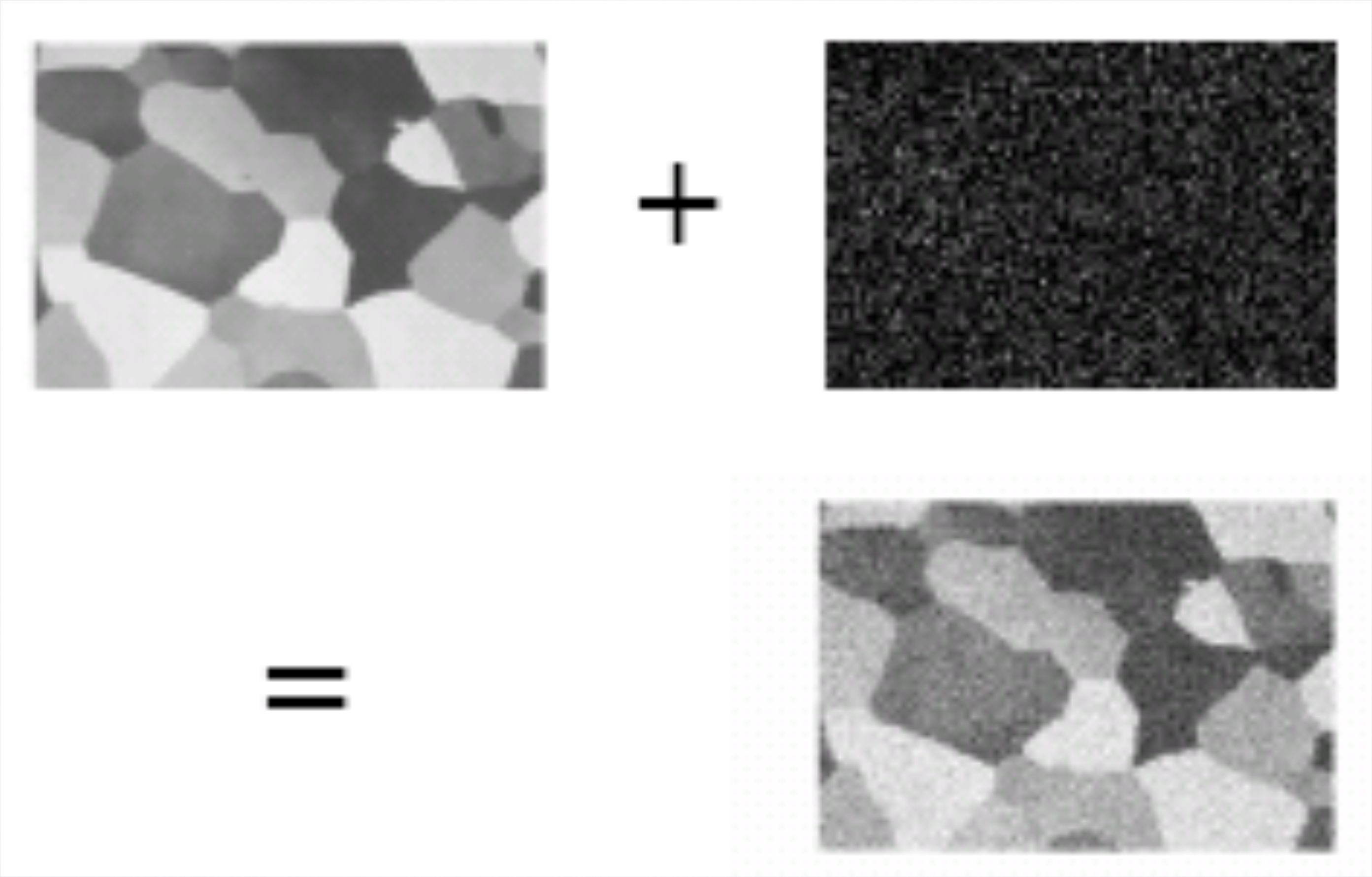


Salt & pepper - Random black & white pixels

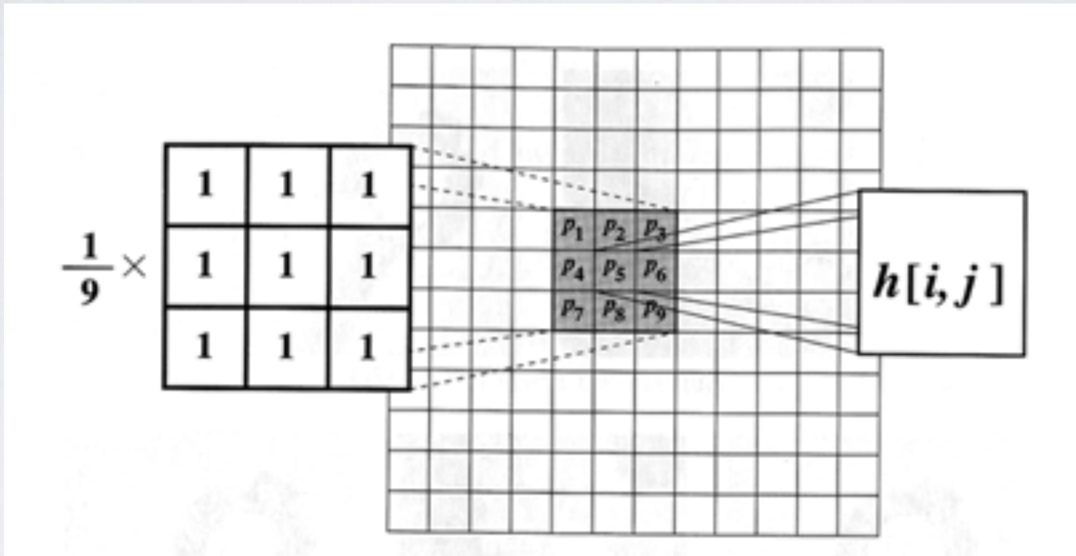
Impulse - Only random white

Gaussian – random gray level variations

ADDITIVE WHITE GAUSSIAN NOISE



MEAN FILTER



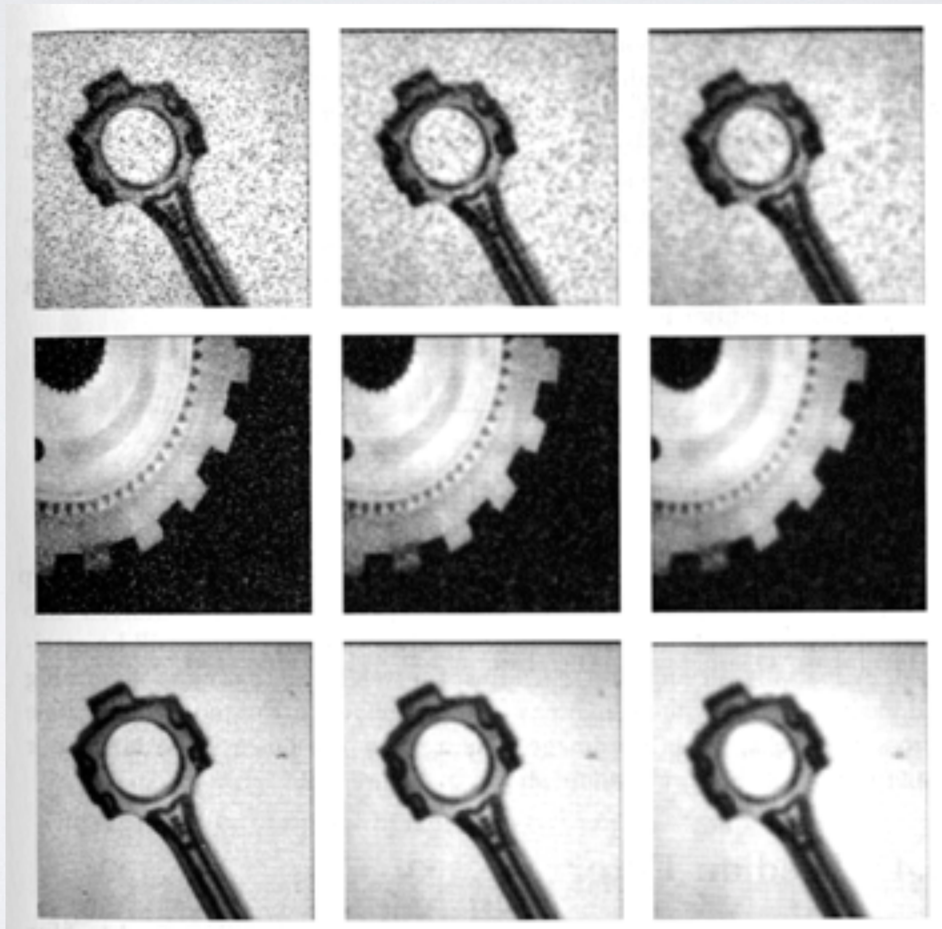
Smoothing filter: pixel weights are not equal, but are usually symmetric and add 1.

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

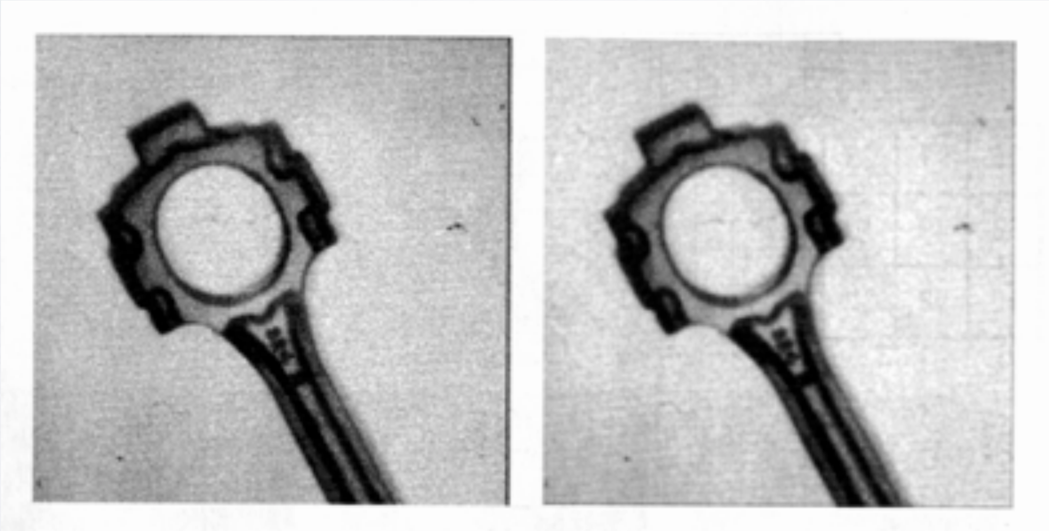
Single-peak, or main lope

Mean Filter Applied to Noisy Images

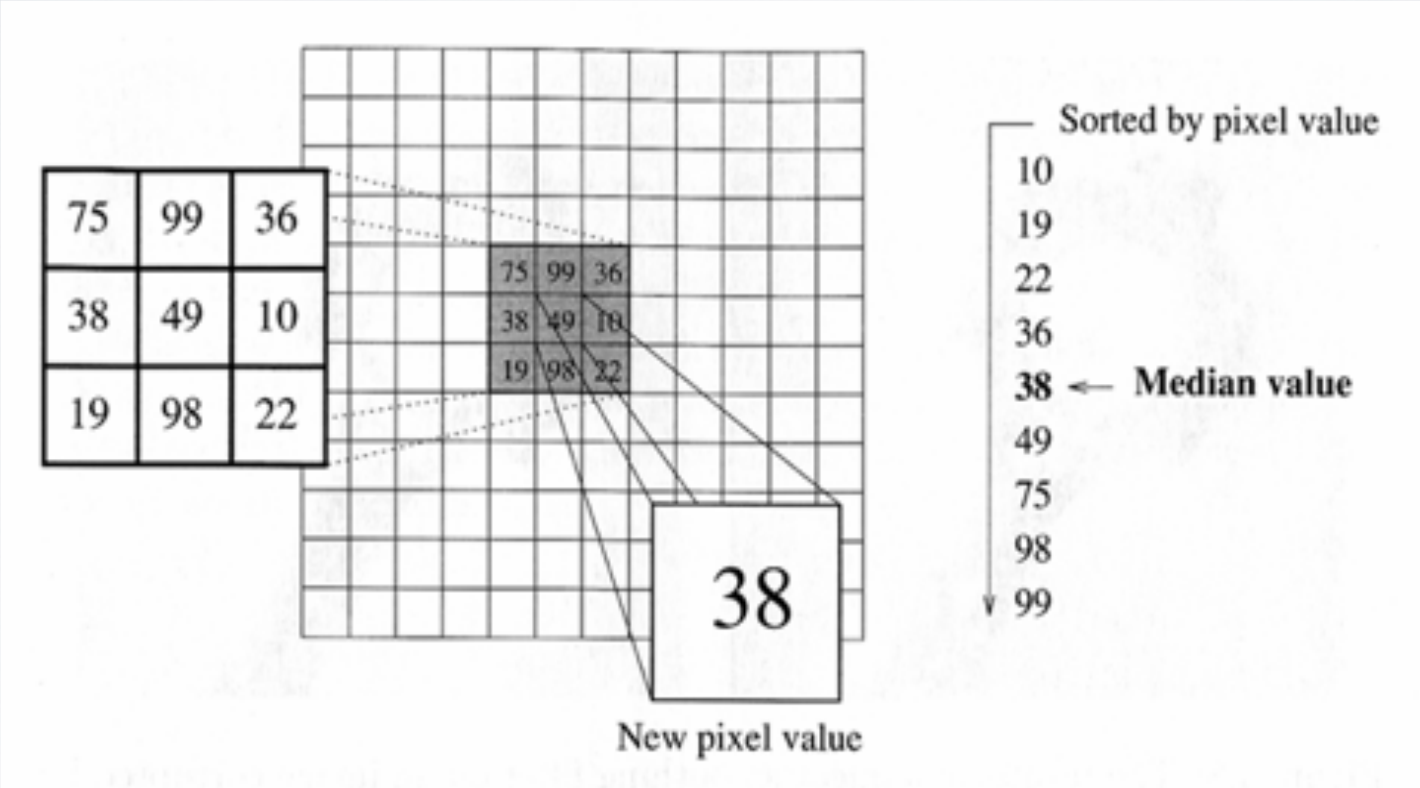
3×3 5×5 7×7



Result of using the smoothing filter seen previously



MEDIAN FILTER



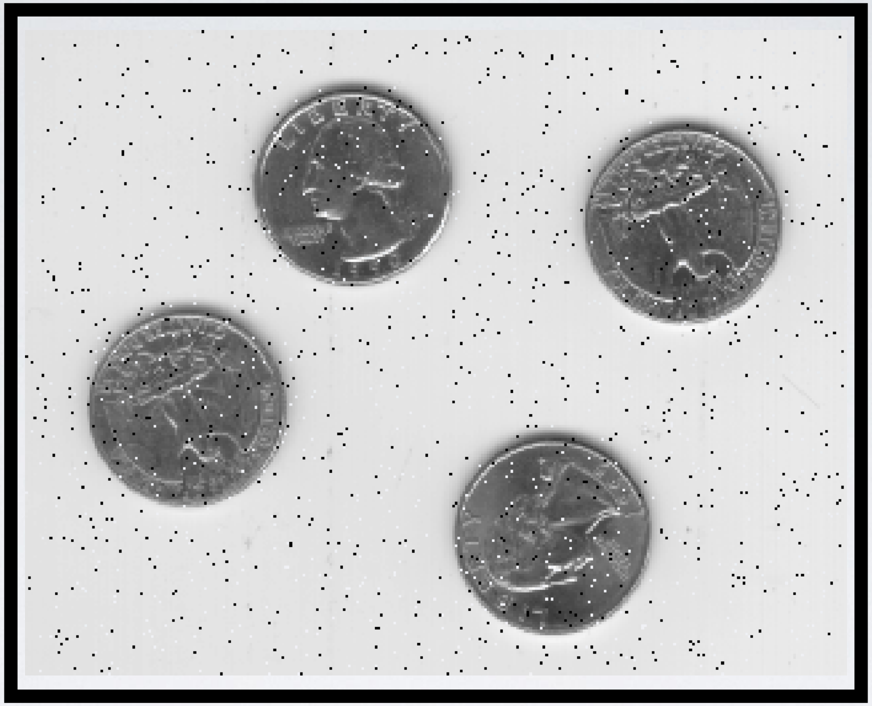
Nonlinear filter - effective to remove salt & pepper and impulse noise without removing too much image detail.

MATLAB Example

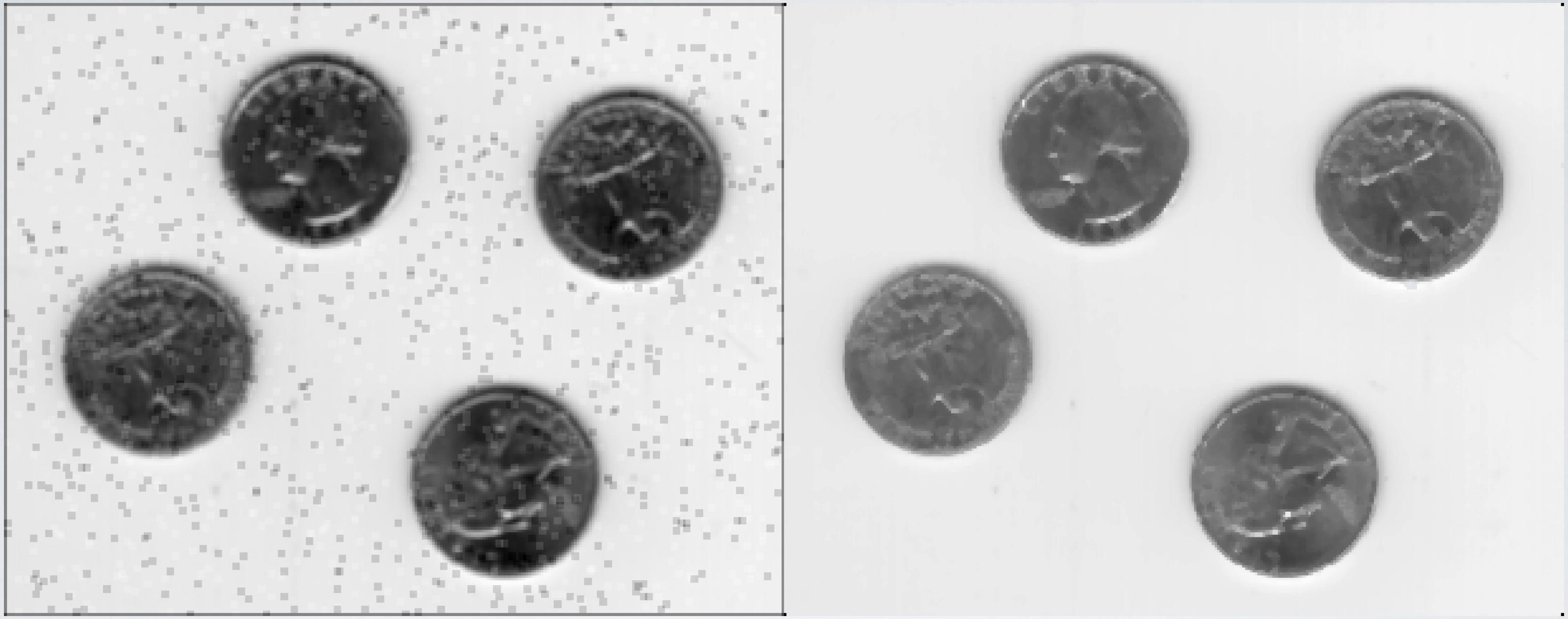
MATLAB EXAMPLE

openExample('images/CompareResultsOfAveragingFilterAndMedianFilterExample')

```
I = imread('eight.tif');
figure
imshow(I)
```



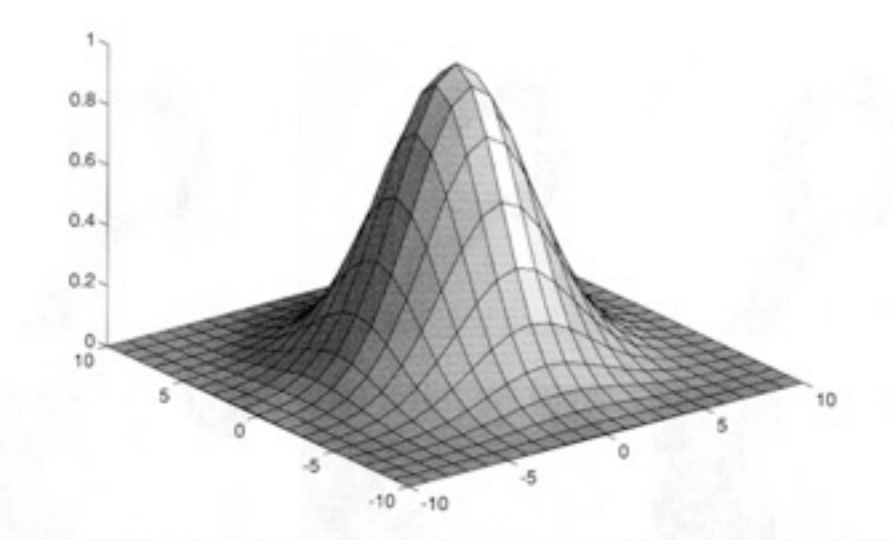
```
% For this example, add salt and pepper noise to the image. This
% type of noise consists of random pixels being set to black or %
% white (the extremes of the data range).
J = imnoise(I,'salt & pepper',0.02);
figure
imshow(J)
```



```
% Filter the noisy image, |J|,
% with an averaging filter and
% display the results. The
% example uses a 3-by-3
% neighborhood.
Kaverage =
filter2(fspecial('average',3),J)/
255;
figure
imshow(Kaverage)
```

```
% Now use a median filter to filter
% the noisy image, |J|. The example
% also uses a 3-by-3 neighborhood.
% Display the two filtered images
% side-by-side for comparison. Notice
% that |medfilt2| does a better job
% of removing noise, with less
% blurring of edges of the coins.
Kmedian = medfilt2(J);
imshowpair(Kaverage,Kmedian,'montage')
```

GAUSSIAN SMOOTHING



$$g[i,j] = \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) = \exp\left(-\frac{i^2}{2\sigma^2}\right)\exp\left(-\frac{j^2}{2\sigma^2}\right)$$

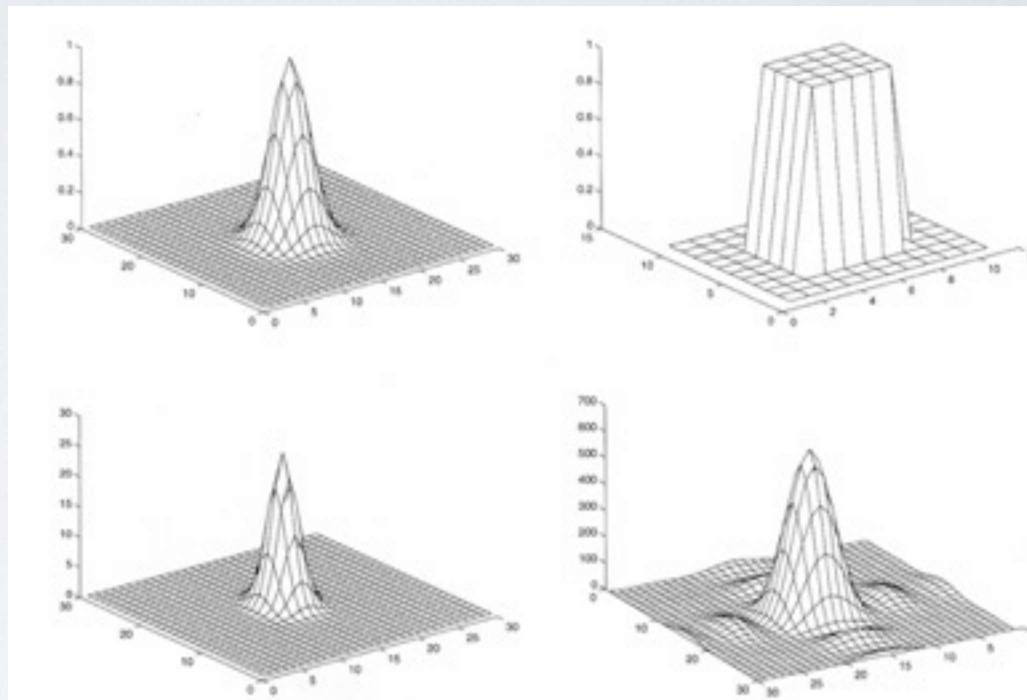
PROPERTIES OF GAUSSIAN FILTER

- Rotational symmetry
- Single lope in both space and frequency domain
- Smoothing is controlled by a single parameter, σ
- Separable
- Will spread impulse and salt & pepper noise!

Gaussian Mean

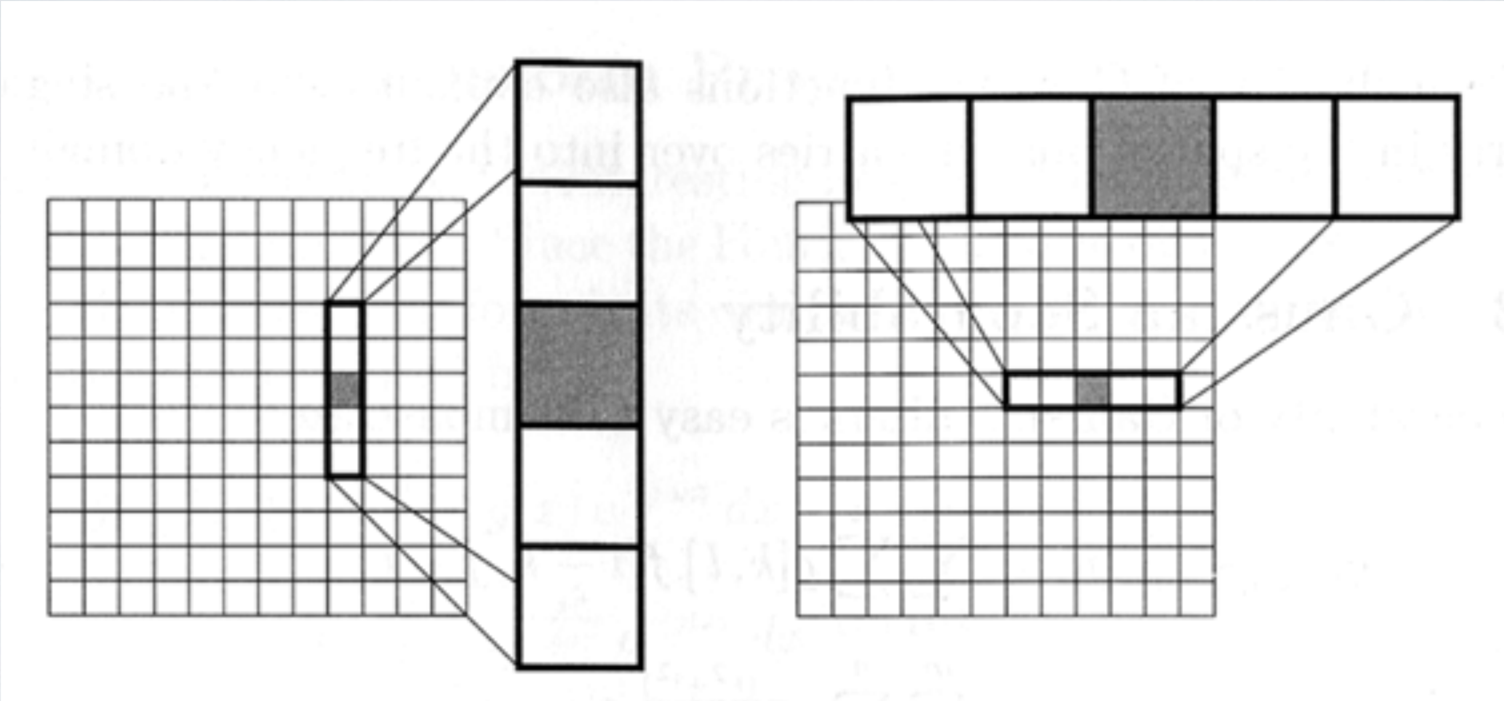
space

frequency



Multiple lopes can cause halos on image

GAUSSIAN FILTER – CAN BE APPLIED AS THE CONVOLUTION OF TWO 1D FILTERS



- To generate the 1D kernel, we must discretize the gaussian
- Require the mask width to subtend most of the gaussian area: use w = 5σ for 98.76% of the area; for a σ=0.6, use w = 3 pixels; for σ=1, use w = 5

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

apply to whole image first

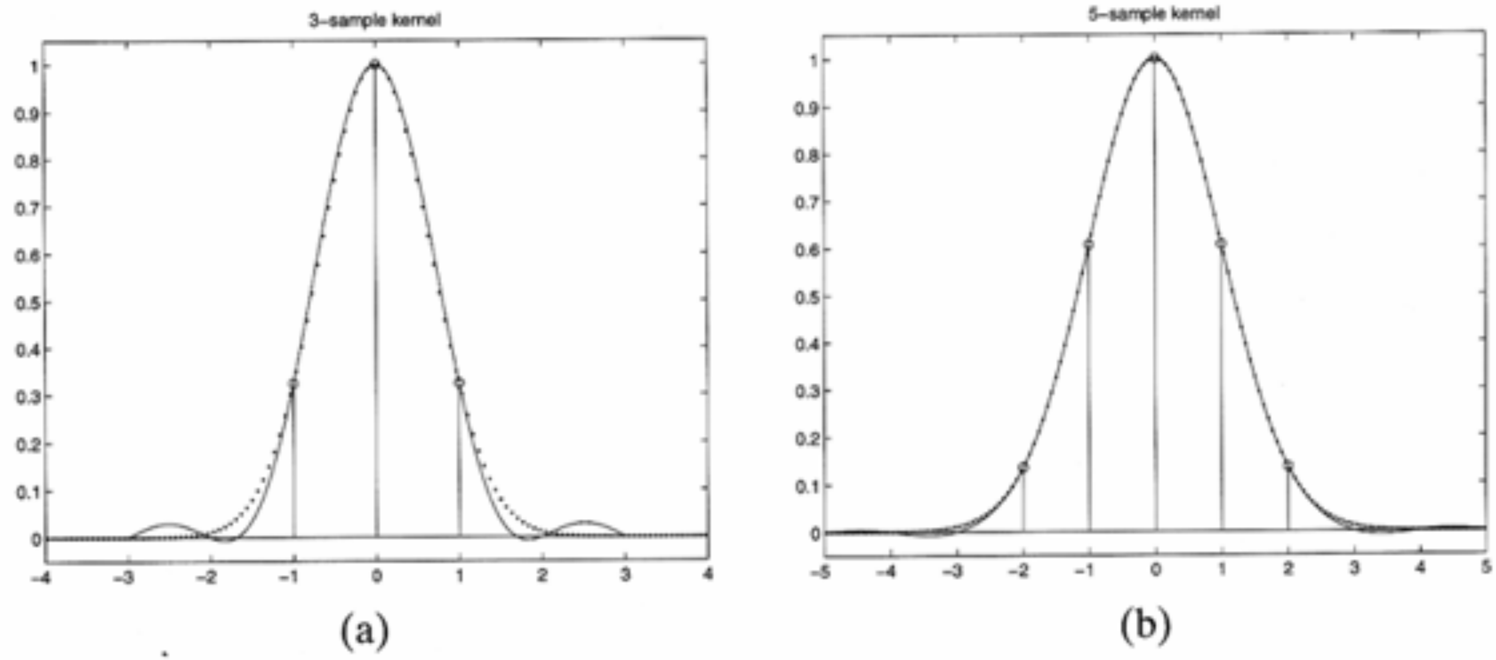
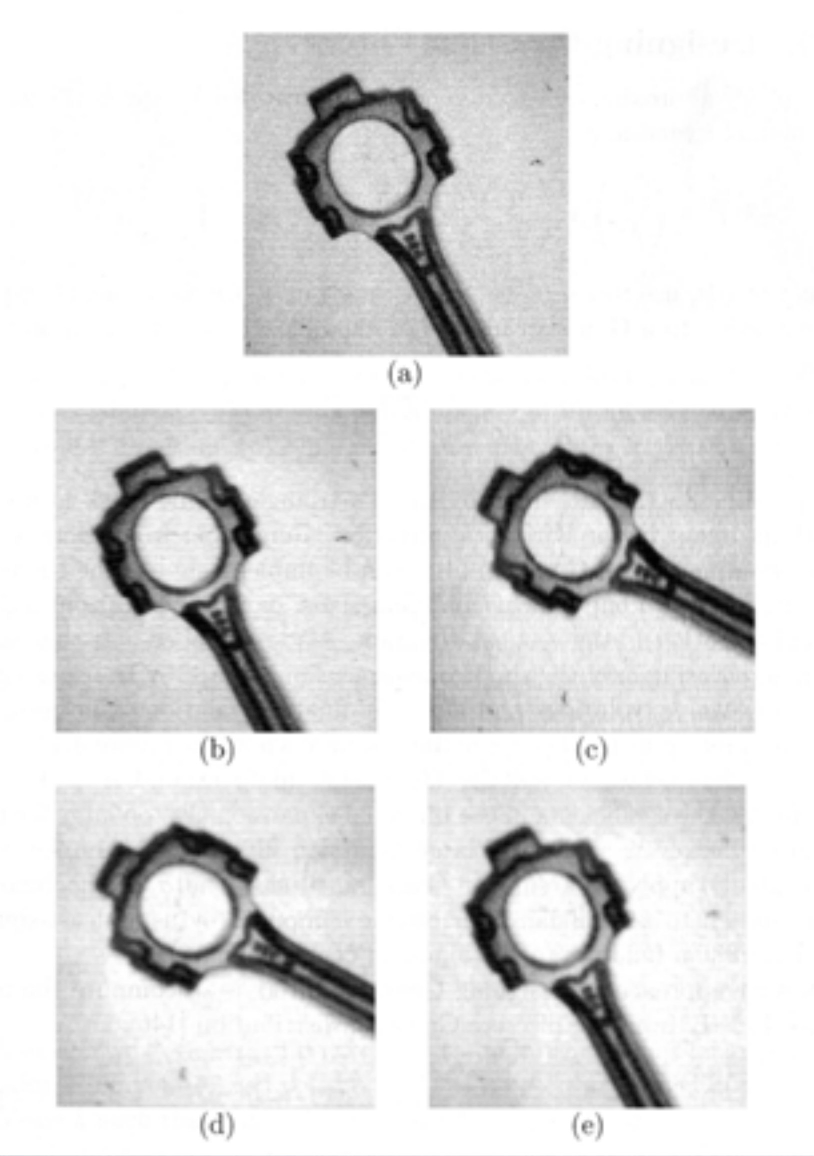


Figure 3.6 Continuous Gaussian kernels (dotted), sampled real kernels, and continuous kernels reconstructed from samples (solid), for $\sigma = 0.6$ ($w = 3$) (a) and $\sigma = 1$ ($w = 5$) (b) respectively.

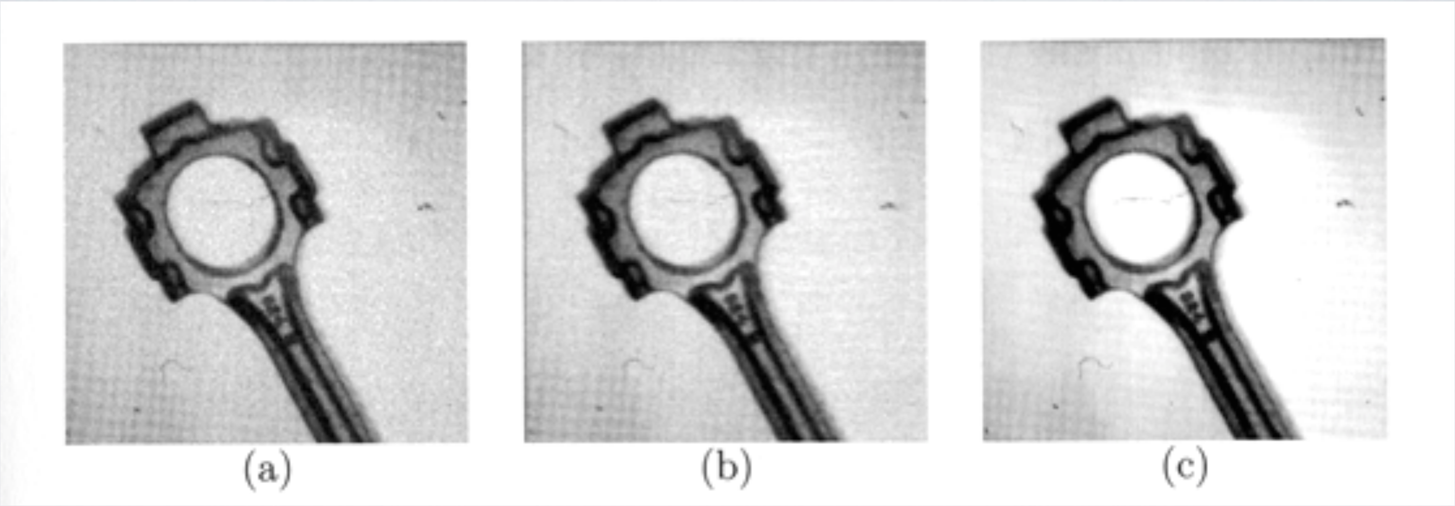


Result of using a single horizontal convolution mask *k* on (a) original noisy image.
(b) After convolution with *k*.
(c) transpose of (b).
(d) convolution of (c) with *k*.
(e) Transpose of (d) to get the final smoothed image.

DESIGN OF GAUSSIAN FILTERS

- Use the row *n* of Pascal's Triangle as a one-dimensional, n-point approximation of a Gaussian filter.

						1						
					1		1					
				1		2		1				
			1		3		3		1			
		1		4		6		4		1		
	1		5		10		10		5		1	
1		6		15		20		15		6		1



Using the fifth row of Pascal's triangle as a Gaussian filter. (a) original; (b) After smoothing in the horizontal dir. (c) After smoothing in the vertical direction

Another approach: compute the mask directly from the discrete Gaussian distribution. Below it is shown the result of using

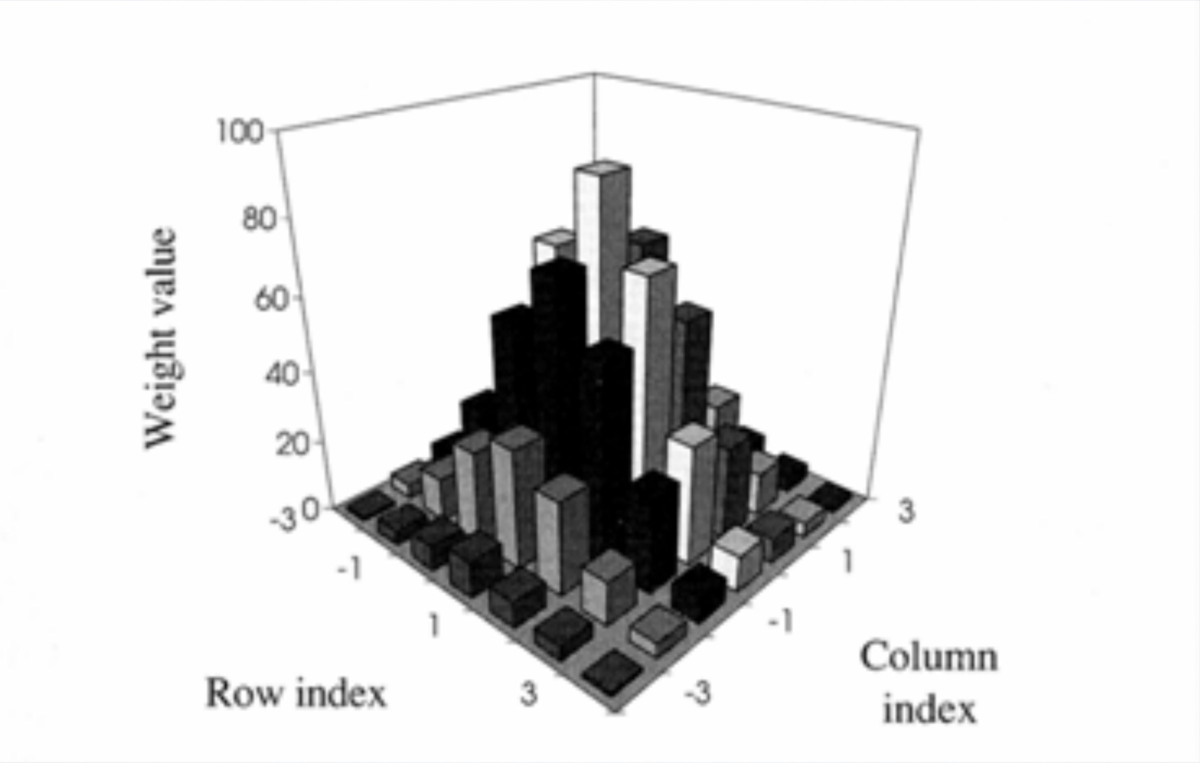
$$\frac{g[i,j]}{c} = e^{-\frac{i^2+j^2}{2\sigma^2}}$$

For n=7 and σ^2=2.

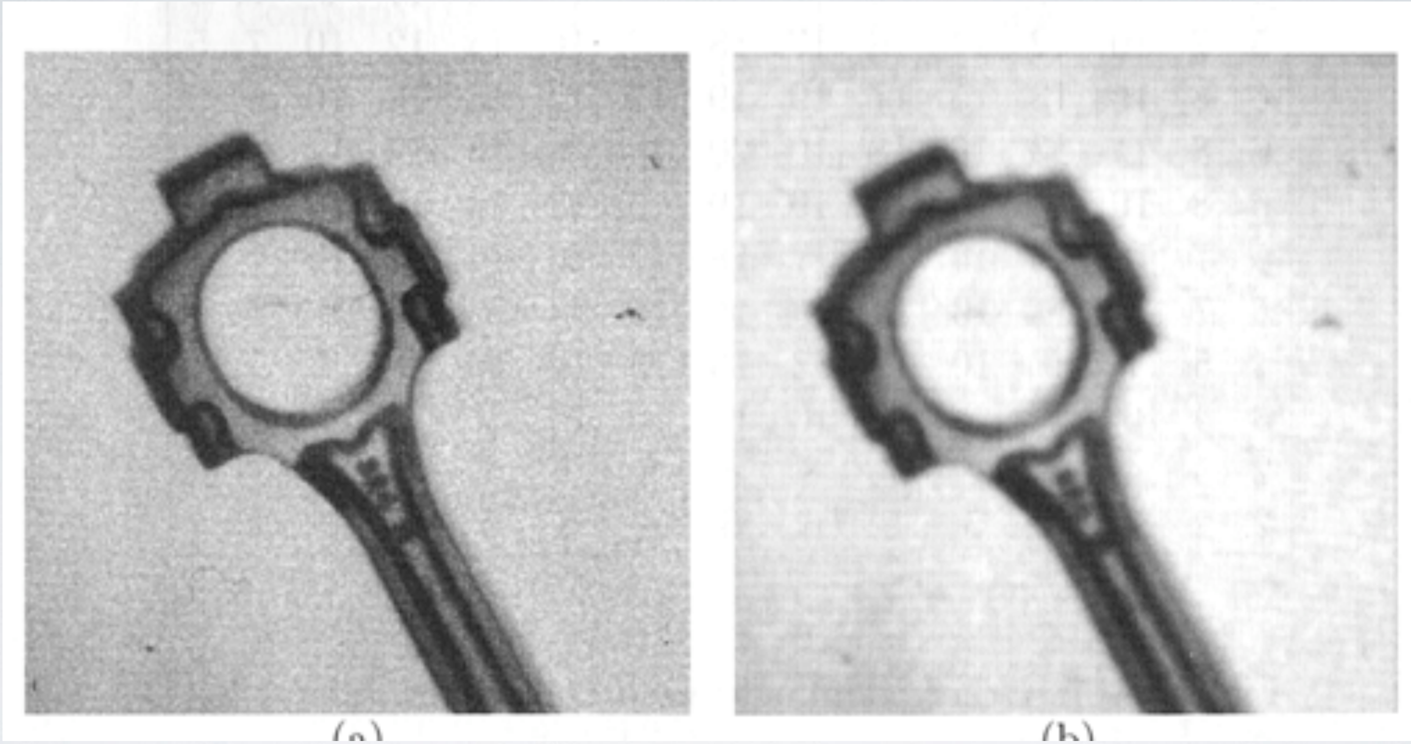
$[i, j]$	-3	-2	-1	0	1	2	3
-3	.011	.039	.082	.105	.082	.039	.011
-2	.039	.135	.287	.368	.287	.135	.039
-1	.082	.287	.606	.779	.606	.287	.082
0	.105	.368	.779	1.000	.779	.368	.105
1	.082	.287	.606	.779	.606	.287	.082
2	.039	.135	.287	.368	.287	.135	.039
3	.011	.039	.082	.105	.082	.039	.011

After multiplying by 91 so that the smallest values (corners) become 1,

$[i, j]$	-3	-2	-1	0	1	2	3
-3	1	4	7	10	7	4	1
-2	4	12	26	33	26	12	4
-1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1



3-D plot of the 7x7 Gaussian mask.



Result of Gaussian smoothing using the 7x7 mask.

Also see a MATLAB example

MATLAB EXAMPLE
SMOOTH IMAGE WITH GAUSSIAN FILTER

openExample('images/SmoothImageWithGaussianFiltersExample')

Original Image

Gaussian filtered image, σ = 2

```
% Filter the image with a Gaussian filter
% with standard deviation of 2.
Iblur = imgaussfilt(I, 2);
```

7 × 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Another commonly used Gaussian mask

The result of the convolution must be divided by the sum of the mask weights to ensure that regions of uniform intensity are not affected.

15 × 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2