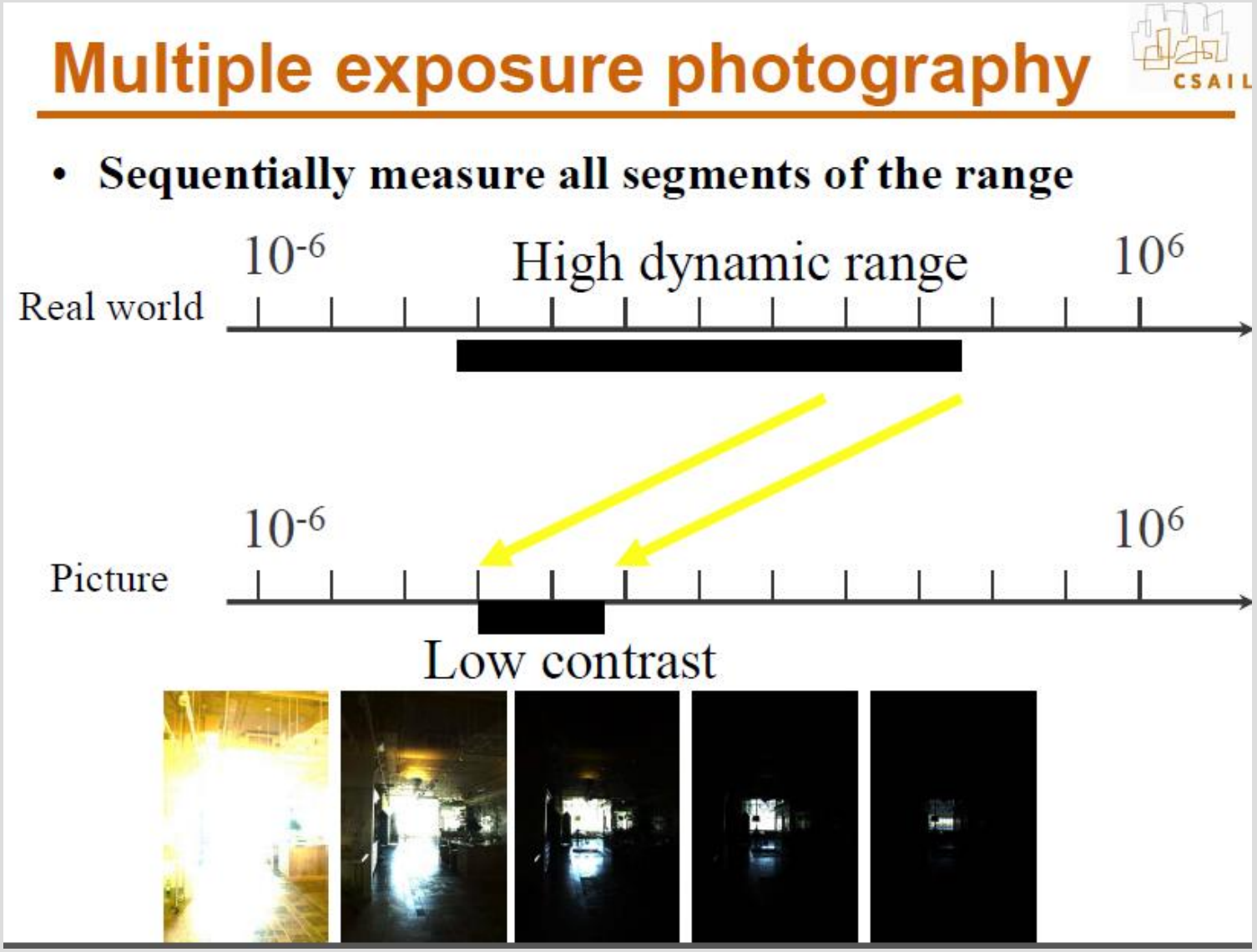
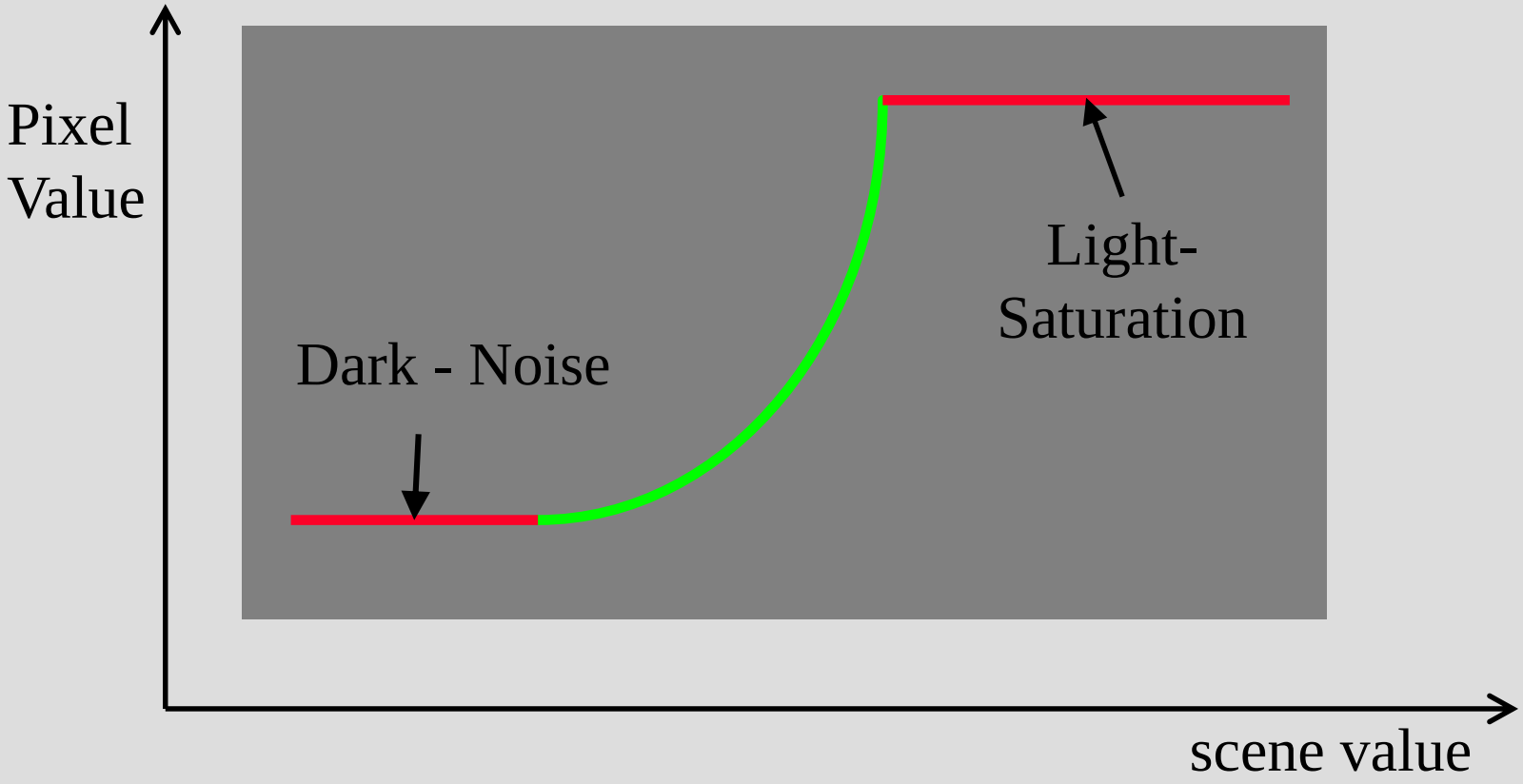
Sequential Collection of Images



Response Curve

The relation between the radiance and the pixel values is

- not linear and
- not completely known



Estimate Camera Response Curve

$$Z_{ij} = f(E_i \Delta t_j) \tag{1}$$

Since we assume f is monotonic, it is invertible, and we can rewrite (1) as:

$$f^{-1}(Z_{ij}) = E_i \Delta t_j$$

Taking the natural logarithm of both sides, we have:

$$\ln f^{-1}(Z_{ij}) = \ln E_i + \ln \Delta t_j$$

To simplify notation, let us define function $g = \ln f^{-1}$. We then have the set of equations:

$$g(Z_{ij}) = \ln E_i + \ln \Delta t_j \tag{2}$$

where i ranges over pixels and j ranges over exposure durations. In this set of equations, the Z_{ij} are known, as are the Δt_j. The unknowns are the irradiances E_i, as well as the function g, although we assume that g is smooth and monotonic.

Debevec – Malik Siggraph 97

Response Curve

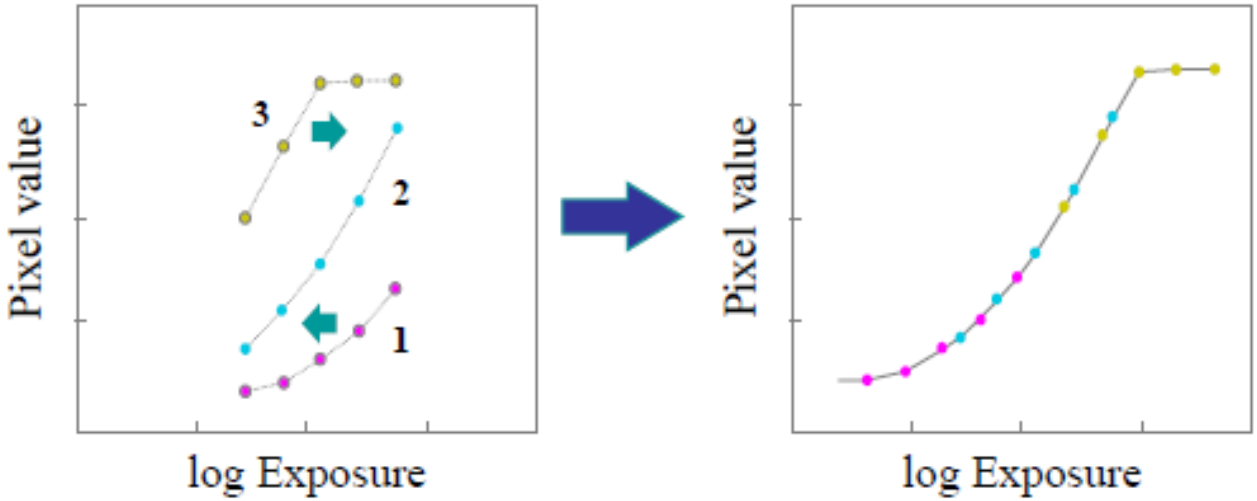


Figure 10.13 Radiometric calibration using multiple exposures (Debevec and Malik 1997). Corresponding pixel values are plotted as functions of log exposures (irradiance). The curves on the left are shifted to account for each pixel's unknown radiance until they all line up into a single smooth curve.

Linear equations

$$g(Z_{ij}) = \ln E_i + \ln t_j \quad \text{same as} \quad g(Z_{ij}) - \ln E_i = \ln t_j$$

Fix i and j and assume that Z_{ij}= m with $0 \leq m < 256$
Define vector A_{ij} = (0 … 1 …0) where the 1 is at position m
Define vector of unknowns: x_1 = (g(0) … g(255))'

Then
$g(Z_{ij}) = A_{ij}x_1$

Define the vector B_i as unit vector (0 … 1 …) with 1 at position I
Define vector of unknowns: x_2 = (ln E_1 ,… ln E_N)'

Then
$B_i\ x_2 = \ln E_i$

For
x' = (x_1 x_2), A_j = (A_{ij} B_i)

$A_j x = \ln t_j$

Linear Algebra

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \left[g(Z_{ij}) - \ln E_i - \ln \Delta t_j \right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g''(z)^2 \qquad (3)$$

Data Term **Smoothness Term**

Measurement k

$A_k x = b_k$

no weights means elements in A_k are 0 or 1

“many” measurements give the equation

Ax = b

with

x = A\b

as solution

Weights

Points near the end are not as reliable as those at the center
Introduce weights w(z)

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \{w(Z_{ij}) [g(Z_{ij}) - \ln E_i - \ln \Delta t_j]\}^2 +$$

$$\lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} [w(z) g''(z)]^2$$

Conclusions/Code

```
%
% Given a set of pixel values observed for several pixels in several
% images with different exposure times, this function returns the
% imaging system's response function g as well as the log film irradiance
% values for the observed pixels.
%
% Assumes:
%
%  Zmin = 0
%  Zmax = 255
%
% Arguments:
%
%  Z(i,j) is the pixel values of pixel location number i in image j
%  B(j)   is the log delta t, or log shutter speed, for image j
%  l      is lamdba, the constant that determines the amount of smoothness
%  w(z)   is the weighting function value for pixel value z
%
% Returns:
%
%  g(z)   is the log exposure corresponding to pixel value z
%  lE(i)  is the log film irradiance at pixel location i
%

function [g,lE]=gsolve(Z,B,l,w)

n = 256;

A = zeros(size(Z,1)*size(Z,2)+n+1,n+size(Z,1));
b = zeros(size(A,1),1);

%% Include the data-fitting equations

k = 1;
for i=1:size(Z,1)
  for j=1:size(Z,2)
    wij = w(Z(i,j)+1);
    A(k,Z(i,j)+1) = wij;  A(k,n+i) = -wij;       b(k,1) = wij * B(i,j);
    k=k+1;
  end
end

%% Fix the curve by setting its middle value to 0

A(k,129) = 1;
k=k+1;

%% Include the smoothness equations

for i=1:n-2
  A(k,i)=l*w(i+1);        A(k,i+1)=-2*l*w(i+1);  A(k,i+2)=l*w(i+1);
  k=k+1;
end

%% Solve the system using SVD

x = A\b;

g = x(1:n);
lE = x(n+1:size(x,1));
```

Problems

Ghosting: Scene is changing while taking pictures

Aligning: SIFT etc.

Lens properties: Aberations

...

Internal Processing in a Camera

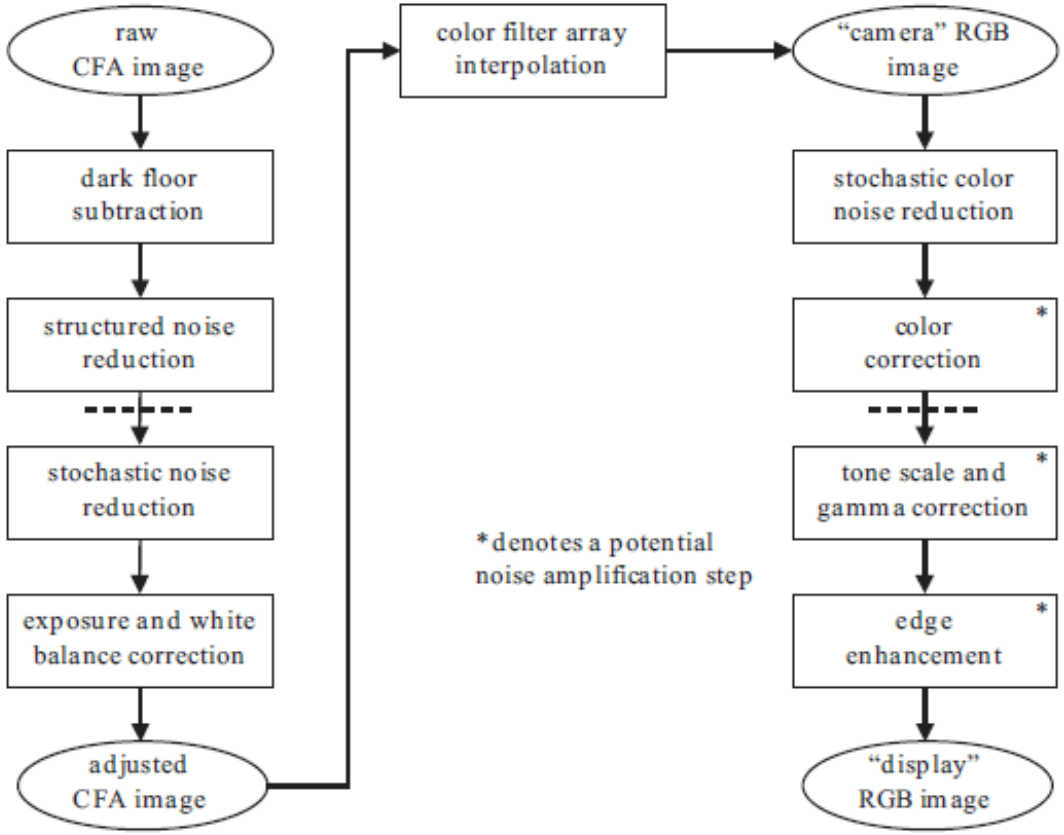


FIGURE 3.1

Reference image processing chain.

In a real camera a lot of signal processing is done that may lead to problems when the camera response curve is estimated

Tonemapping

After the HDR reconstruction we have an image with HDR

How can we ”see” it?

The second half: contrast reduction

- **Input: high-dynamic-range image**
  - (floating point per pixel)

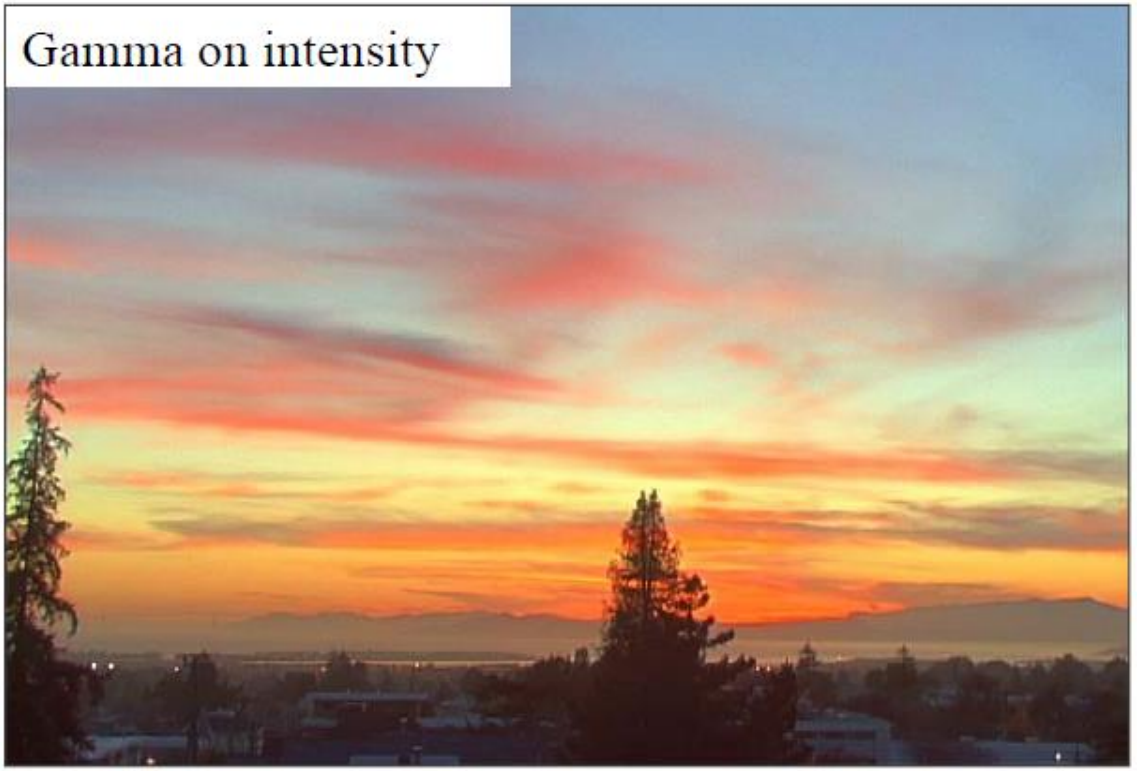
Gamma compression on intensity

- **Colors are OK,**
  but details (intensity high-frequency) are blurred

Intensity

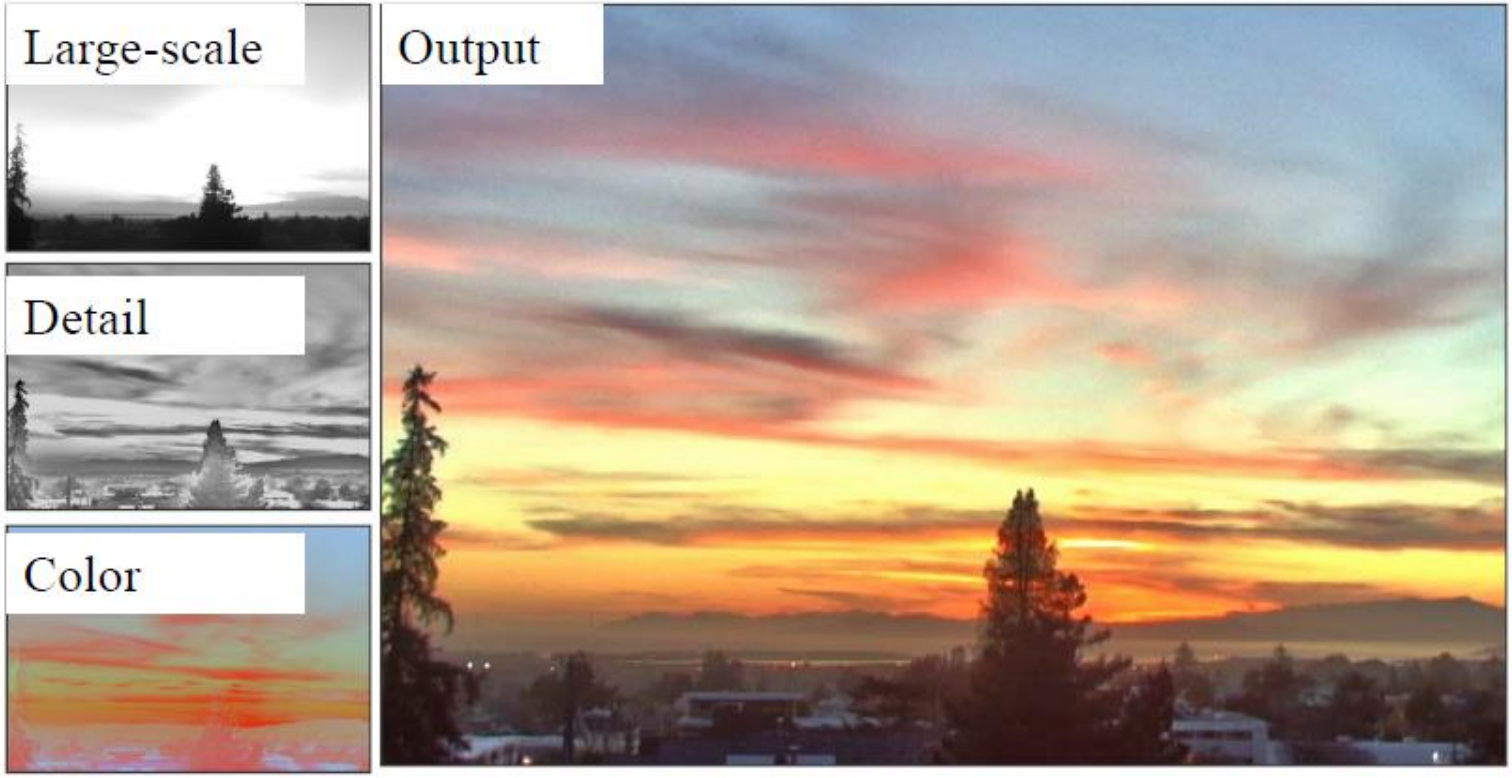
Color

Gamma on intensity



Our approach

- **Do not blur across edges**
- **Non-linear filtering**



Summary

1) Relation LDR and HDR and the basic idea behind HDR
2) Different ways to vary exposure, major advantages-drawbacks
3) Properties of the response curve and its different regions
4) Can explain Fig. 10.13 and the relation between different exposures
5) Explain the three stages:
   - Estimation of the response curve
   - Estimation of the radiance map
   - Role of tonemapping

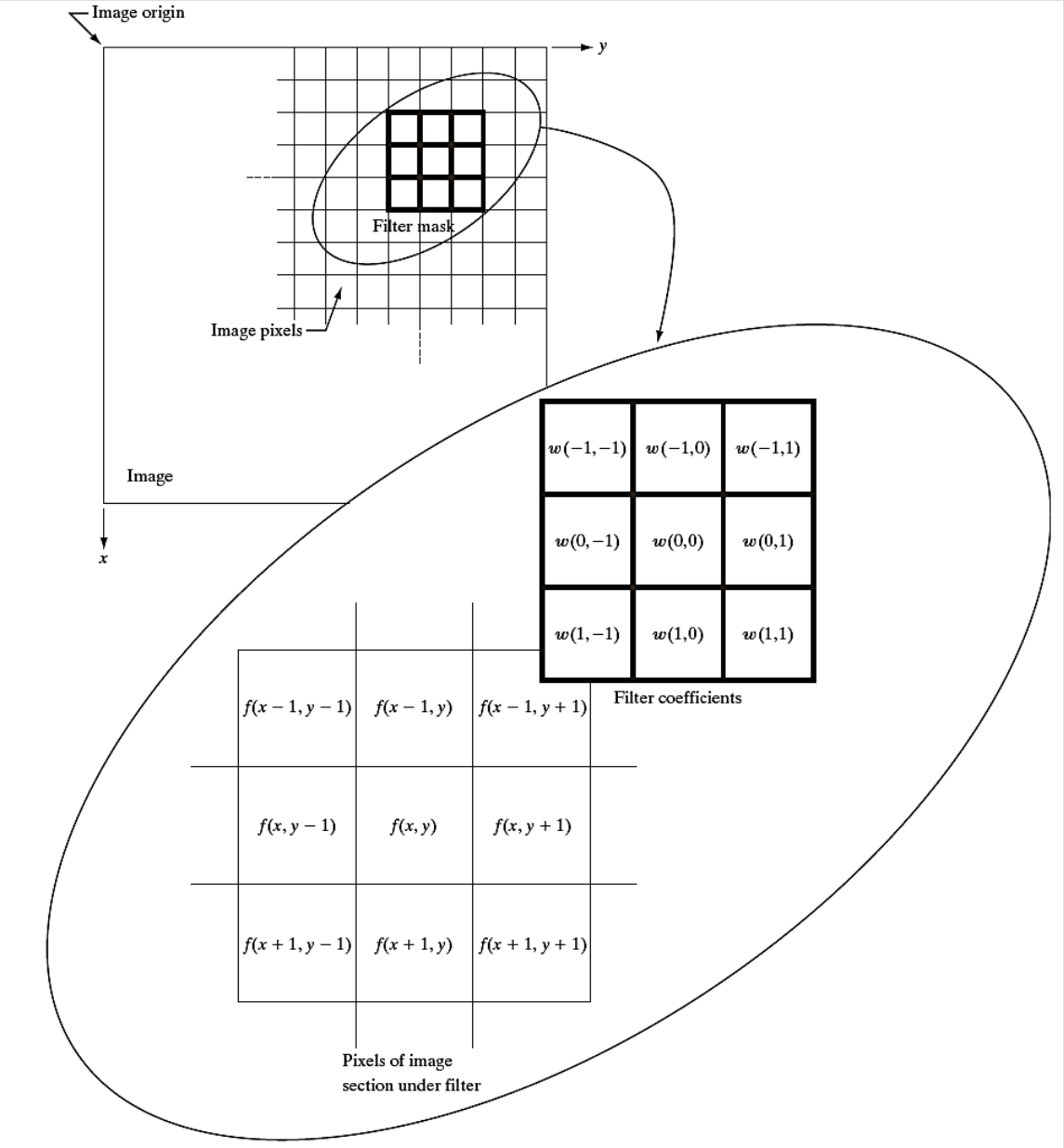
2D Filters

Reiner Lenz

2015

Filtering in the spatial domain

A mask (filter kernel) slides over the image. The centerpixel's new value (filter response) is a function of the kernel values and the pixels covered by the kernel (the neighbourhood).



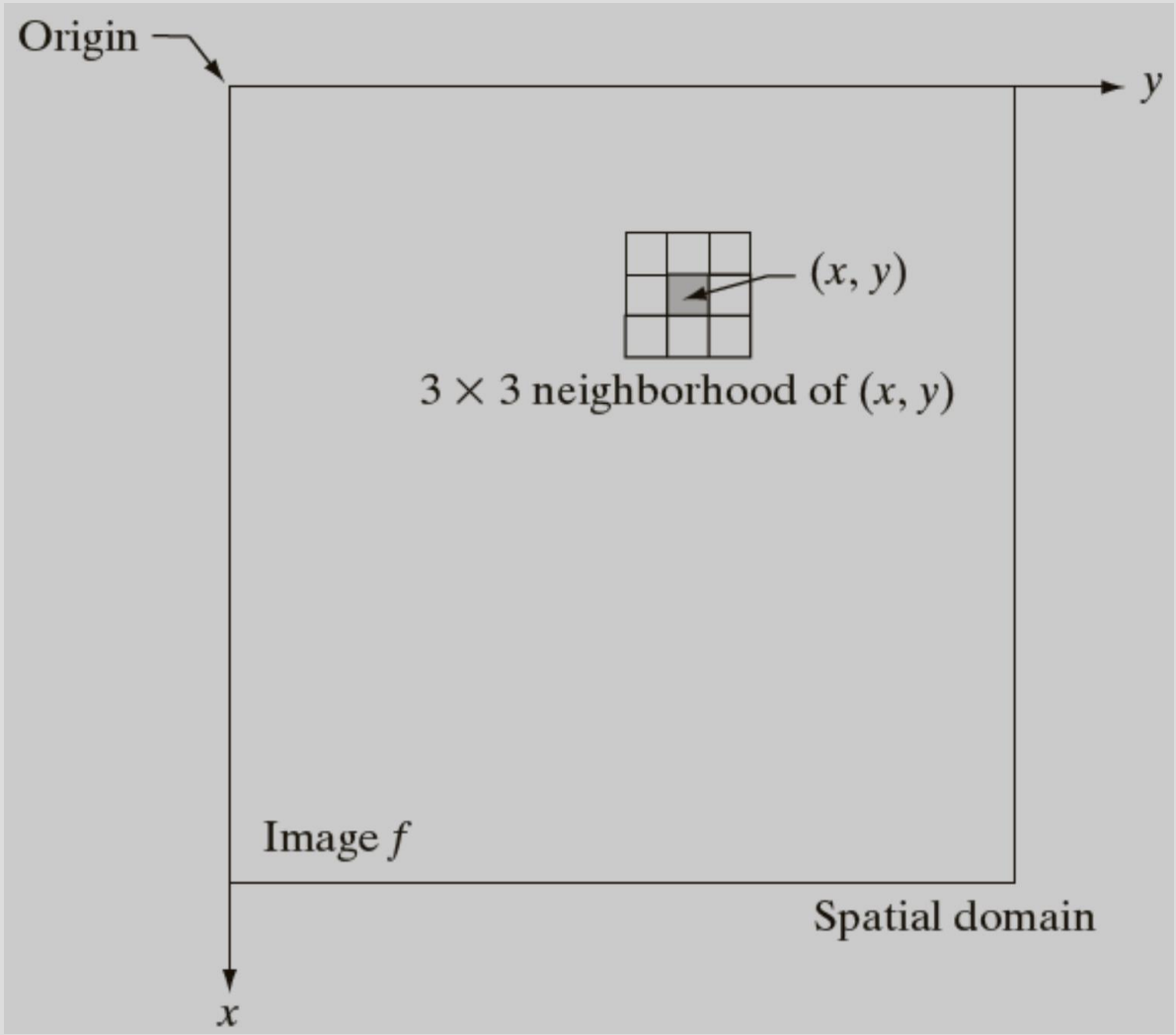


FIGURE 3.1
A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

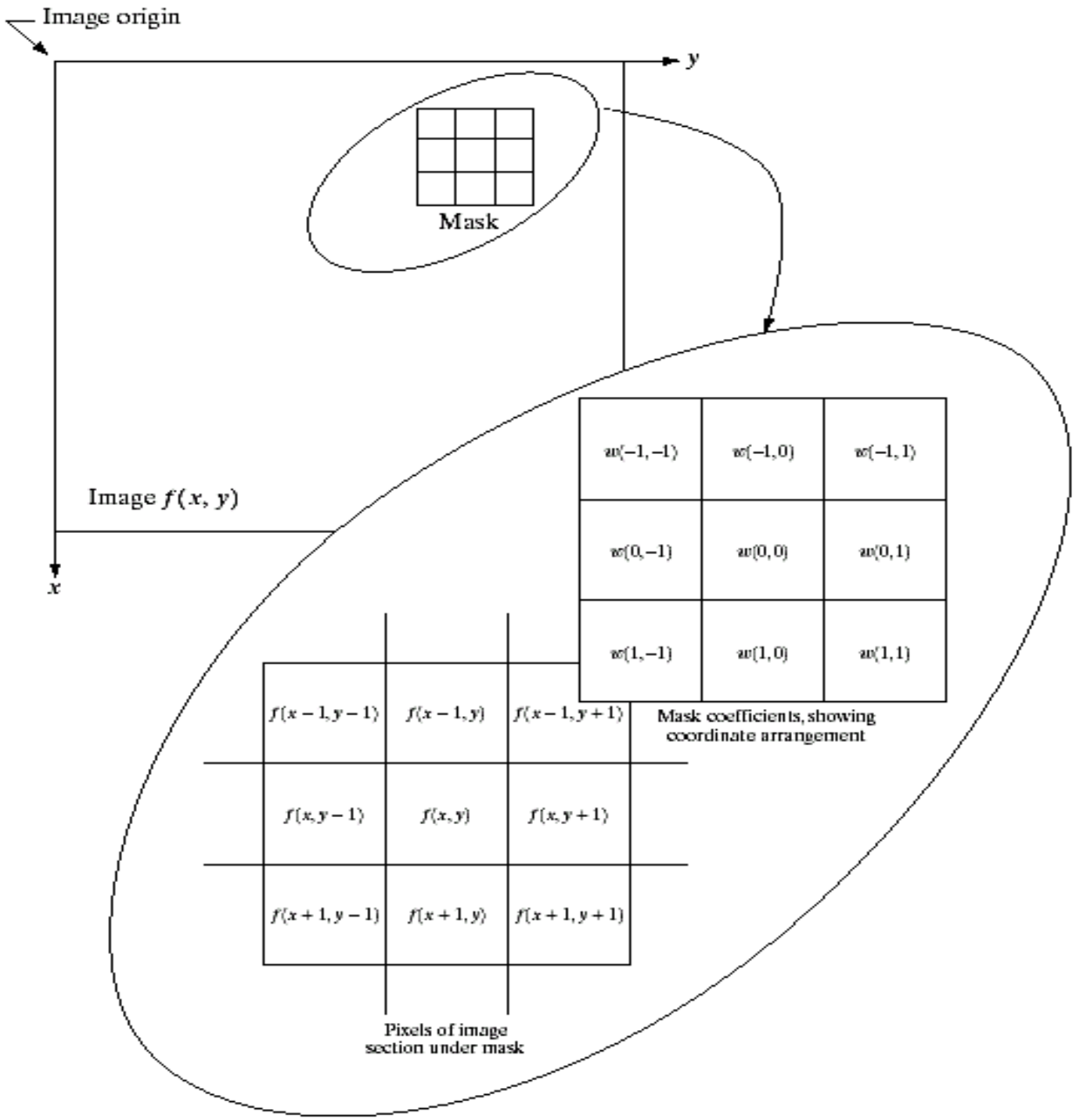


Image origin
y
Mask
Image f(x, y)
x
w(−1, −1)
w(−1, 0)
w(−1, 1)
w(0, −1)
w(0, 0)
w(0, 1)
w(1, −1)
w(1, 0)
w(1, 1)
Mask coefficients, showing coordinate arrangement
f(x − 1, y − 1)
f(x − 1, y)
f(x − 1, y + 1)
f(x, y − 1)
f(x, y)
f(x, y + 1)
f(x + 1, y − 1)
f(x + 1, y)
f(x + 1, y + 1)
Pixels of image section under mask

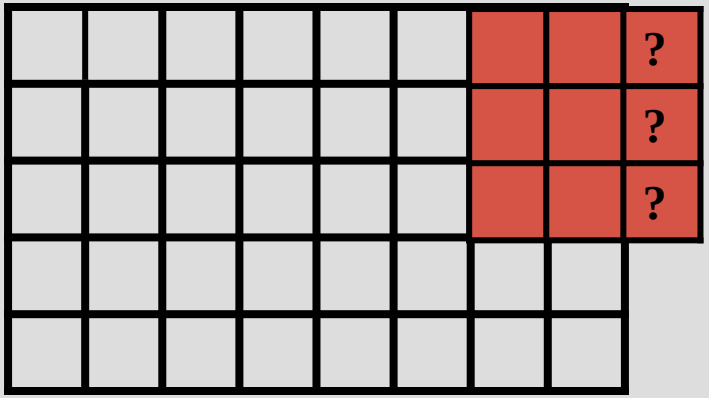
Filtering in the spatial domain

What happens when the filter kernel reaches the borders of the image?

* Don’t let the kernel reach outside => frame of unfiltered pixels in the result image

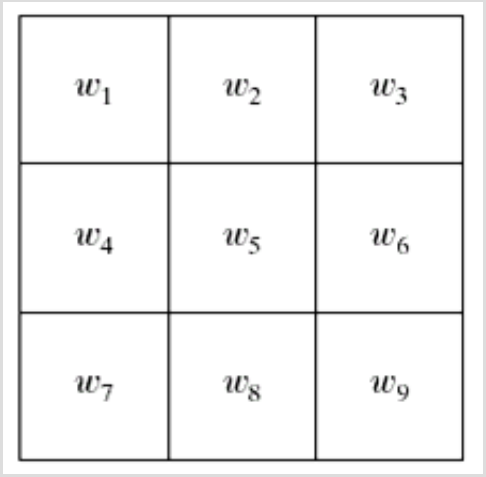
•Give the image a frame of zeroes (”zero- padding”) =>

frame of miscalculated pixels in the result



Filtering in the spatial domain

Linear filter:



$$g = \sum_{i=1}^{9} w_i f_i$$

Kernel

g : filter respons

f_i : pixels covered by kernel

Linear filtering - correlation

The ***correlation*** between two 2D-functions, the image f(x,y) and the filter kernel w(m,n).

Continous case :

$$g(x,y) = (f \circ w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x+a, y+b)w(a,b)dadb$$

Discrete case:

$$g(x,y) = \sum_{(s,t)\in neighbourhood} \sum f(x+s, y+t)w(s,t)$$

Linear filtering - convolution

The ***convolution*** between two 2D-functions: the image f(x,y) and the filter kernel w(m,n).

Continous case :

$$g(x,y) = (f \bullet w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a, y-b)w(a,b)dadb$$

Discrete case:

$$g(x,y) = \sum_{(s,t)\in neighbourhood} \sum f(x-s, y-t)w(s,t)$$

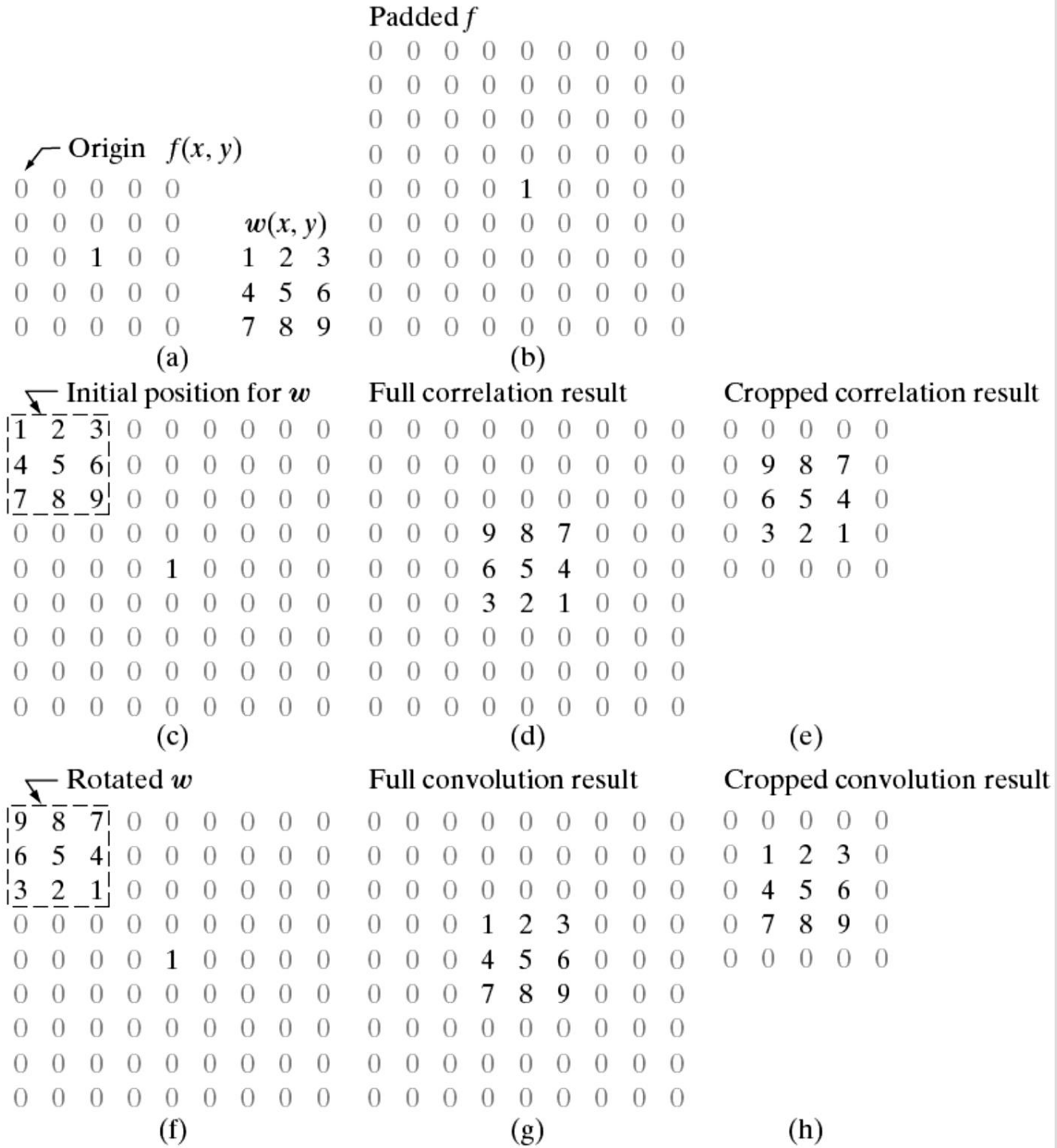
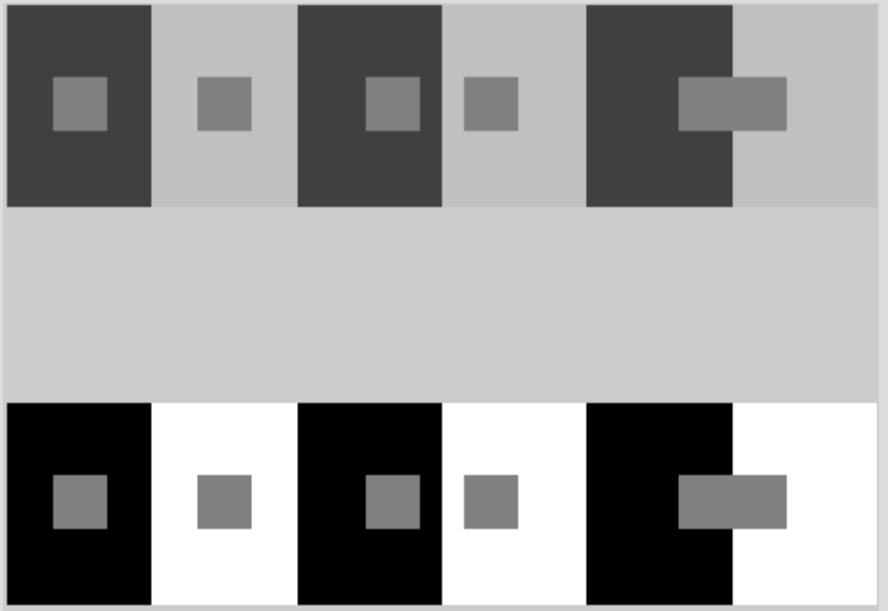


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

Matlab

```
ima = imread(fullfile(impath,'F2_08.TIF'));
subplot(2,1,1),imshow(ima)
subplot(2,1,2),imshow(ima,[])
```

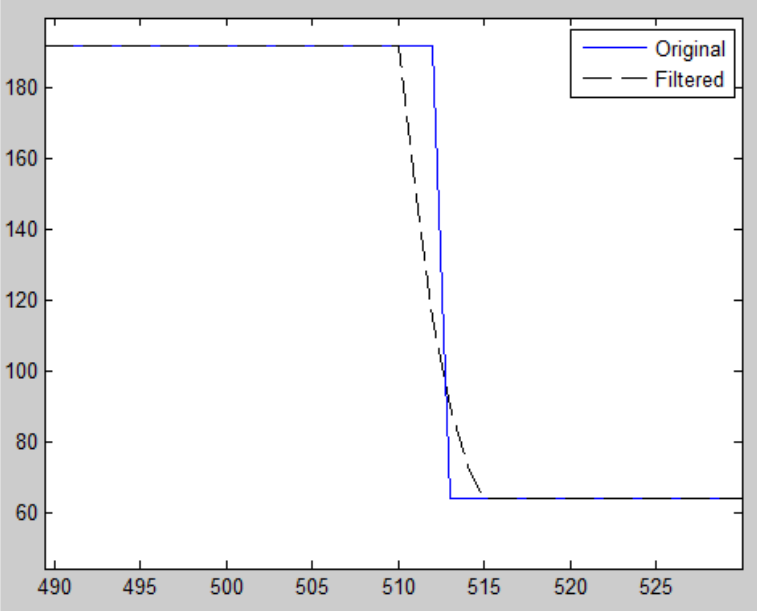
```
h = (1:5)' ;
fima= imfilter(ima,h'/sum(h));
figure
plot(ima(90,:),'b')
hold on
plot(fima(90,:),'k--')
legend('Original','Filtered')
```



```
ima(90,508:517)
 192  192  192  192  192   64   64   64   64  64

lima=ones(6,1)*double(ima(90,508:517));
for k = 1:6
  filterresult(k) = lima(k,k:k+4)*h/15;
end

filterresult =
192.0000  149.3333  115.2000   89.6000
72.5333   64.0000
>> fima(90,508:517)
192  192  192  149  115   90   73   64   64   64
```

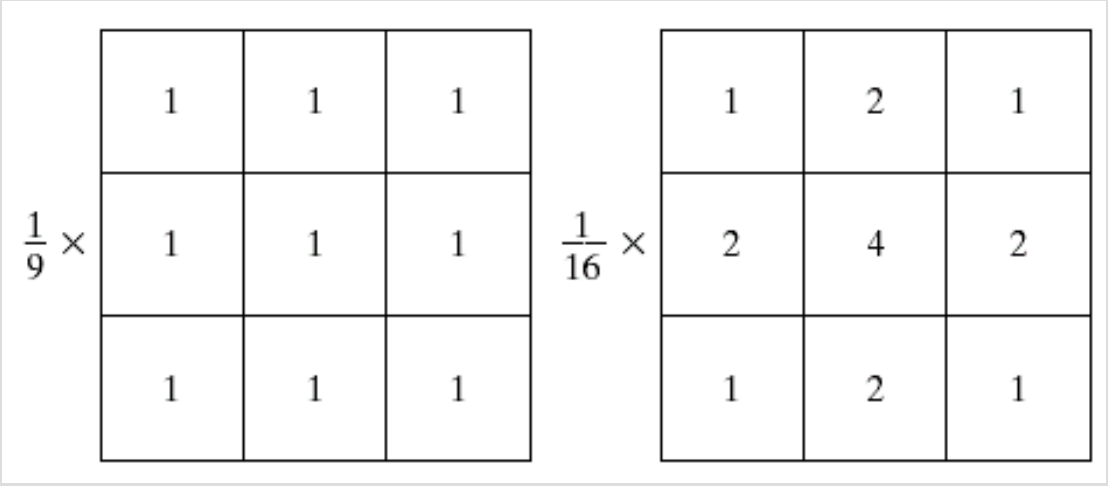


BasicFilterDemo.html

Averaging filters

Smoothes the image. Supression of noise and small objects in the image.

Example:



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

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Boxfilter

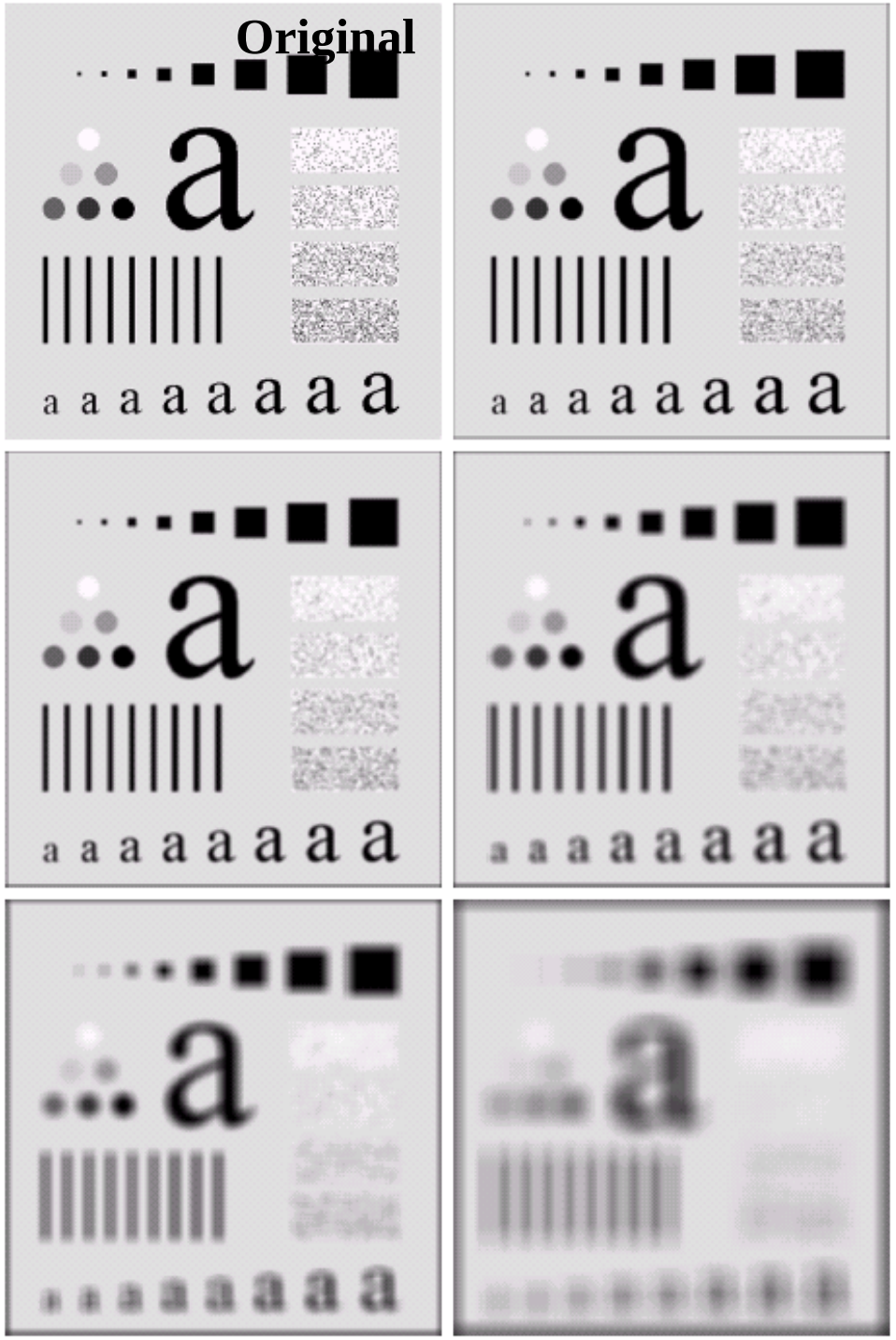
Weighted filter

Averaging filters

Different kernel sizes:

3x3, 5x5, 9x9, 15x15,

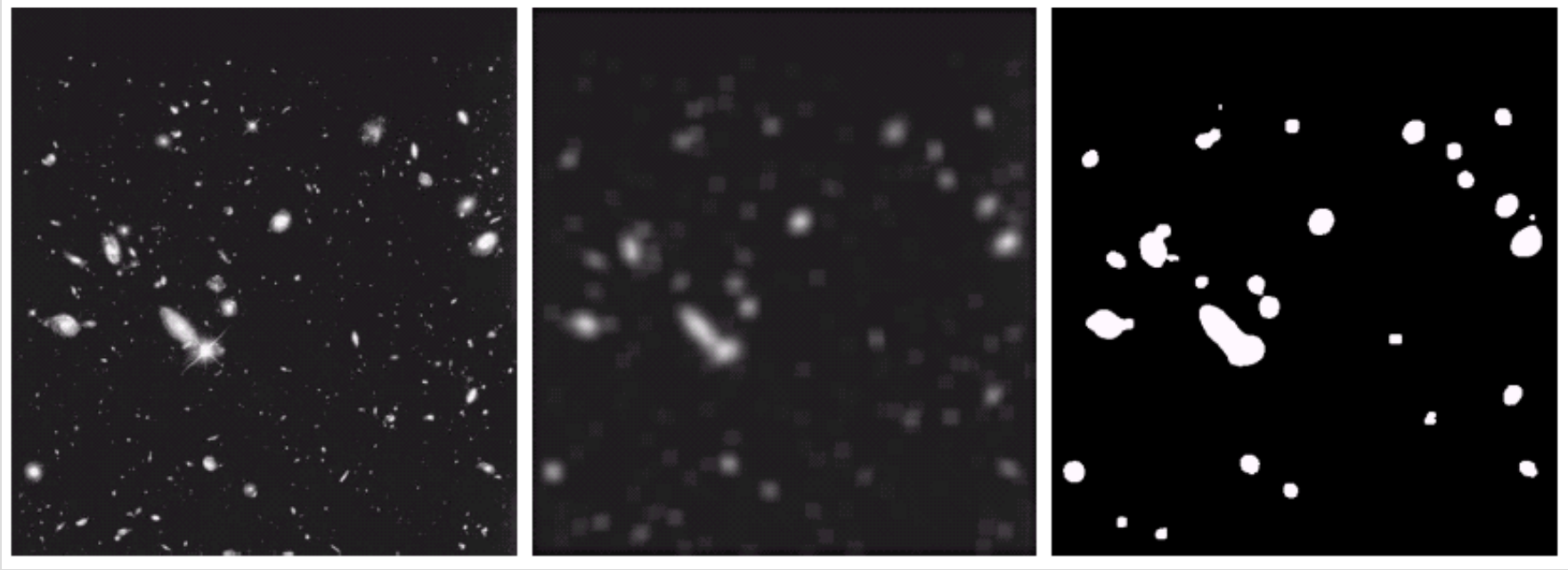
35x35



Averaging filters

Supression of small objects in the image.

Example (15x15-kernel):



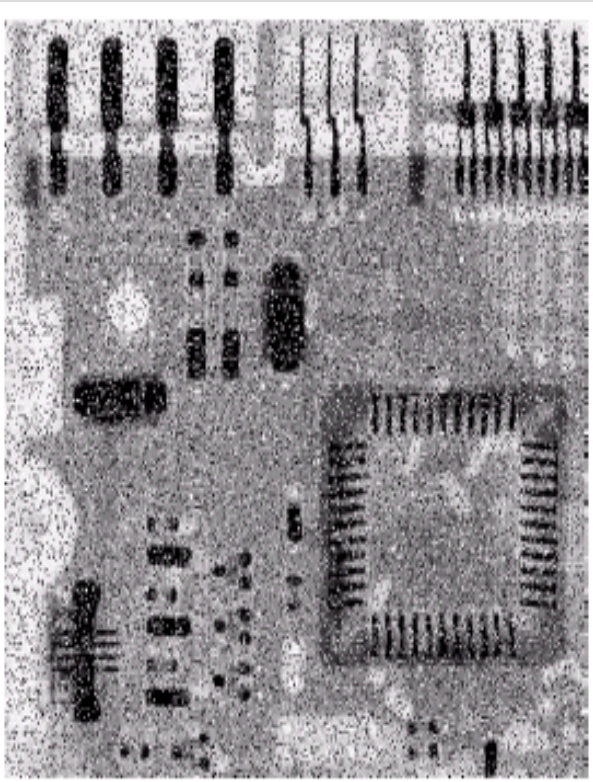
Original Result Thresholded result

Median filter

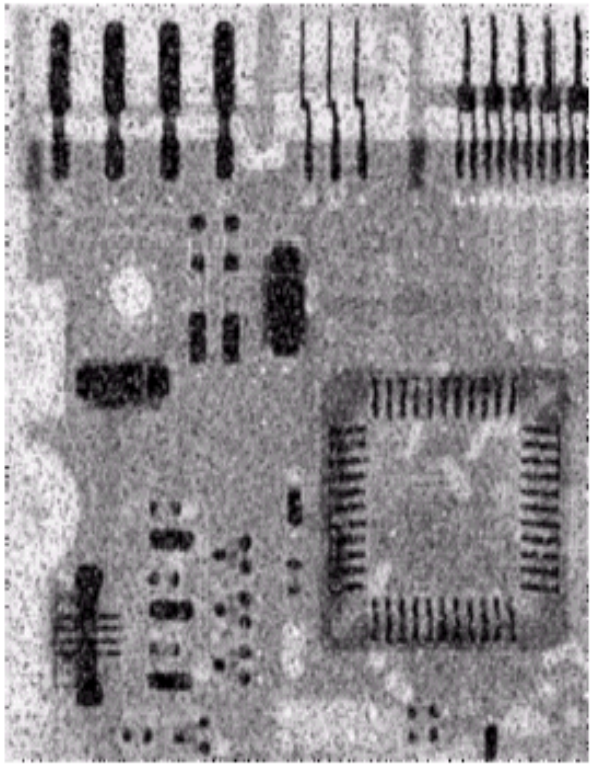
- The center pixel is replaced by the *median* of the pixel values covered by the kernel.
- Non-linear filter.
- Good for removal of ”spiky” noise (salt and pepper noise)
- No smoothing of sharp edges.

Median filter

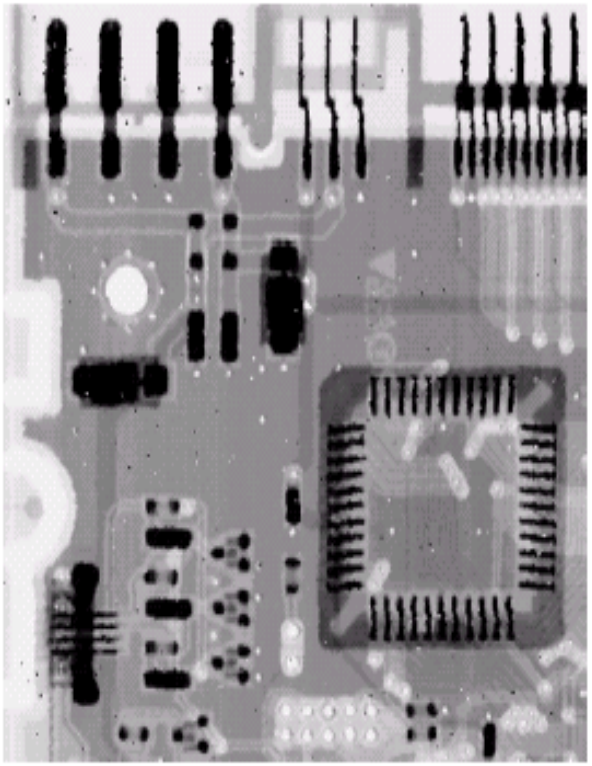
Example:



Original



3x3 average



3x3 median

Derivating filters

- Averaging (integrating) filters result in smoothing
- Derivating filters result in sharpening.

1-D Derivatives

$$\frac{\partial f}{\partial x} \approx f(x+1) - f(x)$$

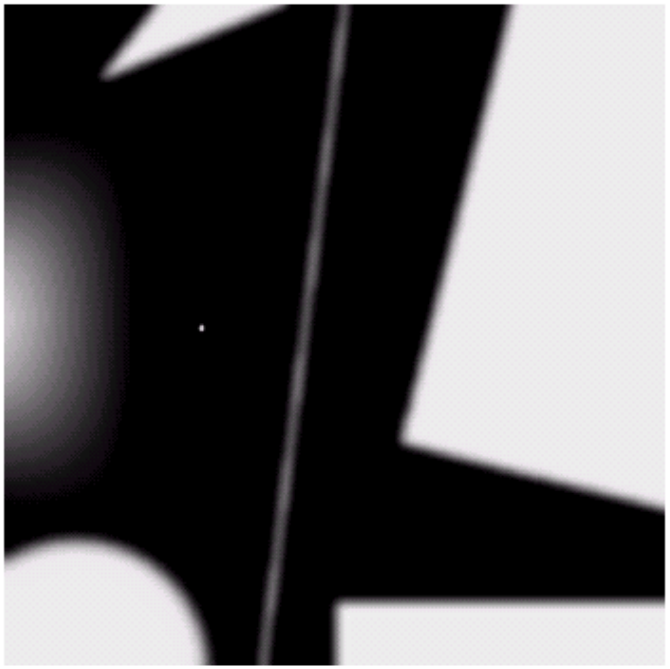
(-1 1)

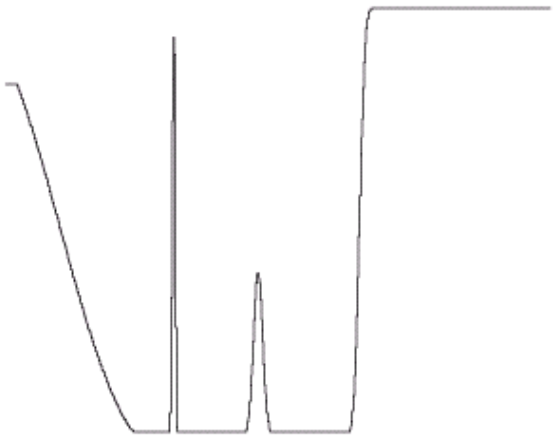
$$\frac{\partial}{\partial x^2} = \frac{\partial}{\partial x}\frac{\partial}{\partial x} \approx g(x+1) - g(x) =$$
$$f(x+2) - f(x+1) - \big(f(x+1) - f(x)\big) =$$
$$f(x+2) - f(x+1) \; - f(x+1) + f(x) =$$
$$f(x+2) - 2f(x+1) + f(\mathrm{x})$$

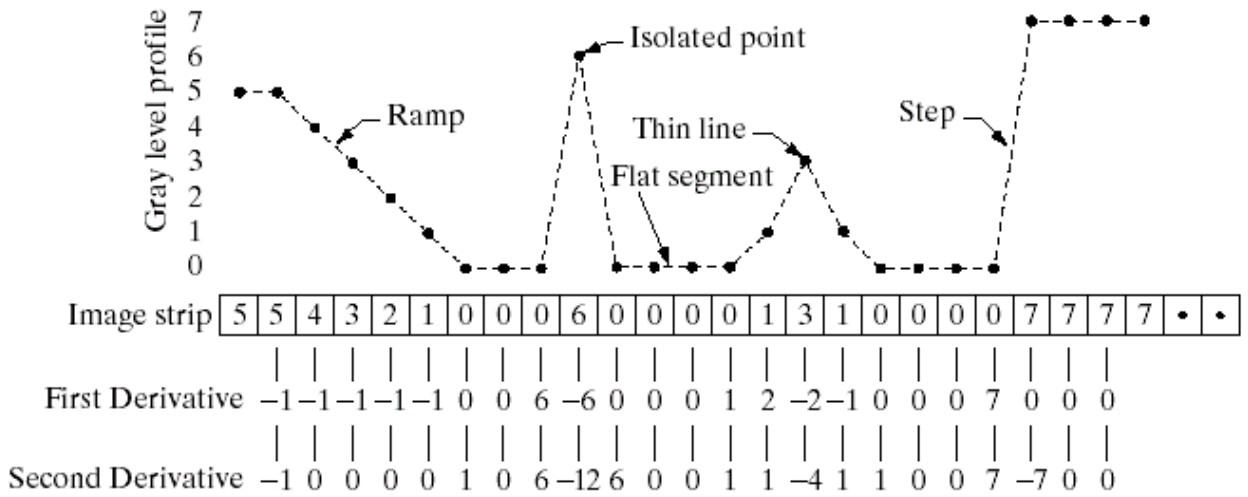
(1 -2 1)

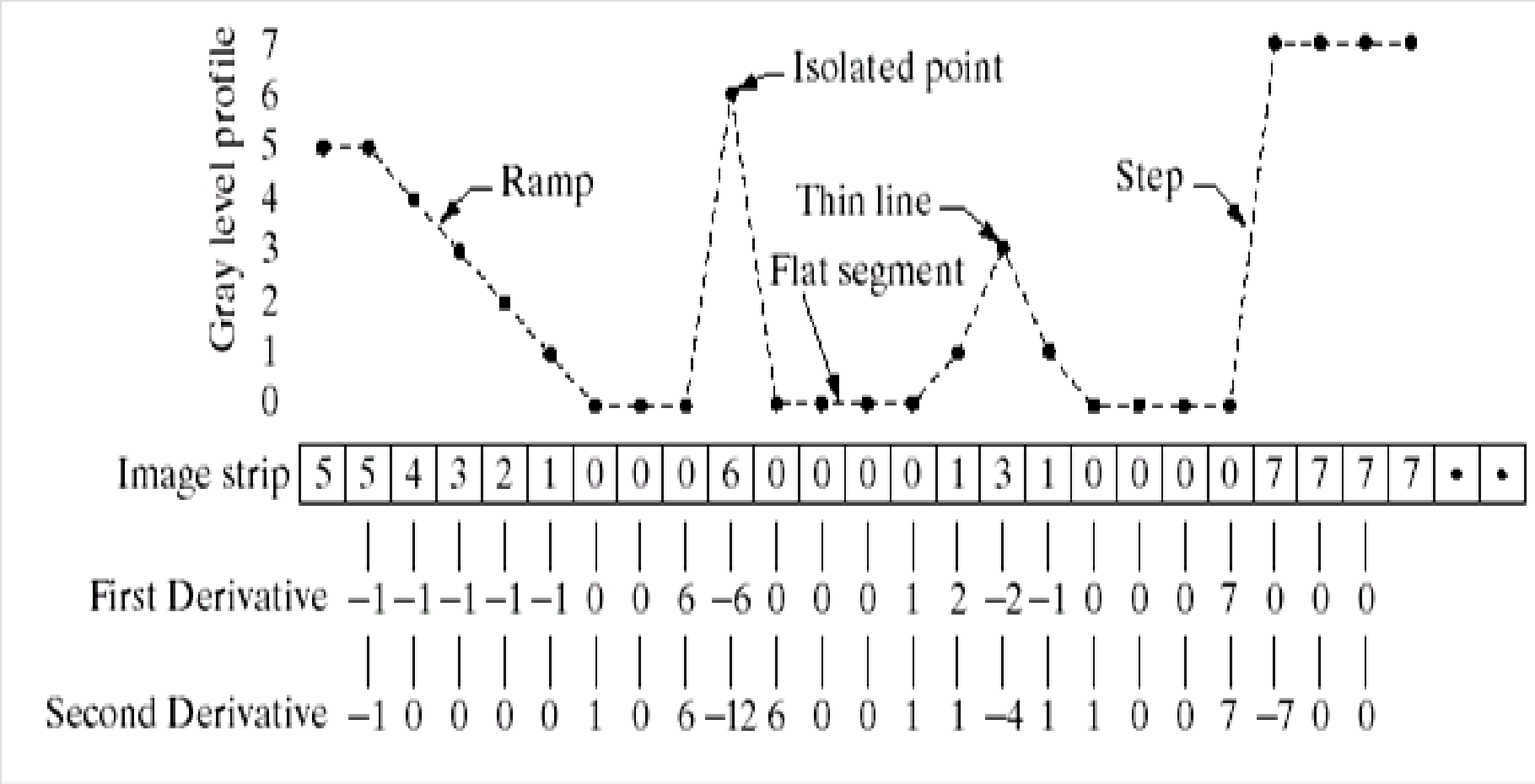
Derivating filters

Effects of 1st and 2nd derivatives on a 1D-signal (intensity profile in image).








Gray level profile
7
6
5
4
3
2
1
0
Ramp
Isolated point
Thin line
Flat segment
Step
Image strip
5 5 4 3 2 1 0 0 0 6 0 0 0 0 1 3 1 0 0 0 0 7 7 7 7
First Derivative
−1 −1 −1 −1 −1 0 0 6 −6 0 0 0 1 2 −2 −1 0 0 0 7 0 0 0
Second Derivative
−1 0 0 0 0 1 0 6 −12 6 0 0 1 1 −4 1 1 0 0 7 −7 0 0

Laplacian filter

*Commonly used for sharpening.

•Based on the 2nd derivative

•*Isotropical*, i.e. yields the same result independent of the rotation of the image (rotation invariance)

•The Laplace operator:

$$\nabla^2 f(x,y) = \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y)$$

Laplacian filter

- Discrete Laplace:

$$\nabla^2 f(i,j) = f(i,j-1) + f(i-1,j) + f(i+1,j) + f(i,j+1) - 4f(i,j)$$

Laplace-kernels

0	1	0
1	−4	1
0	1	0

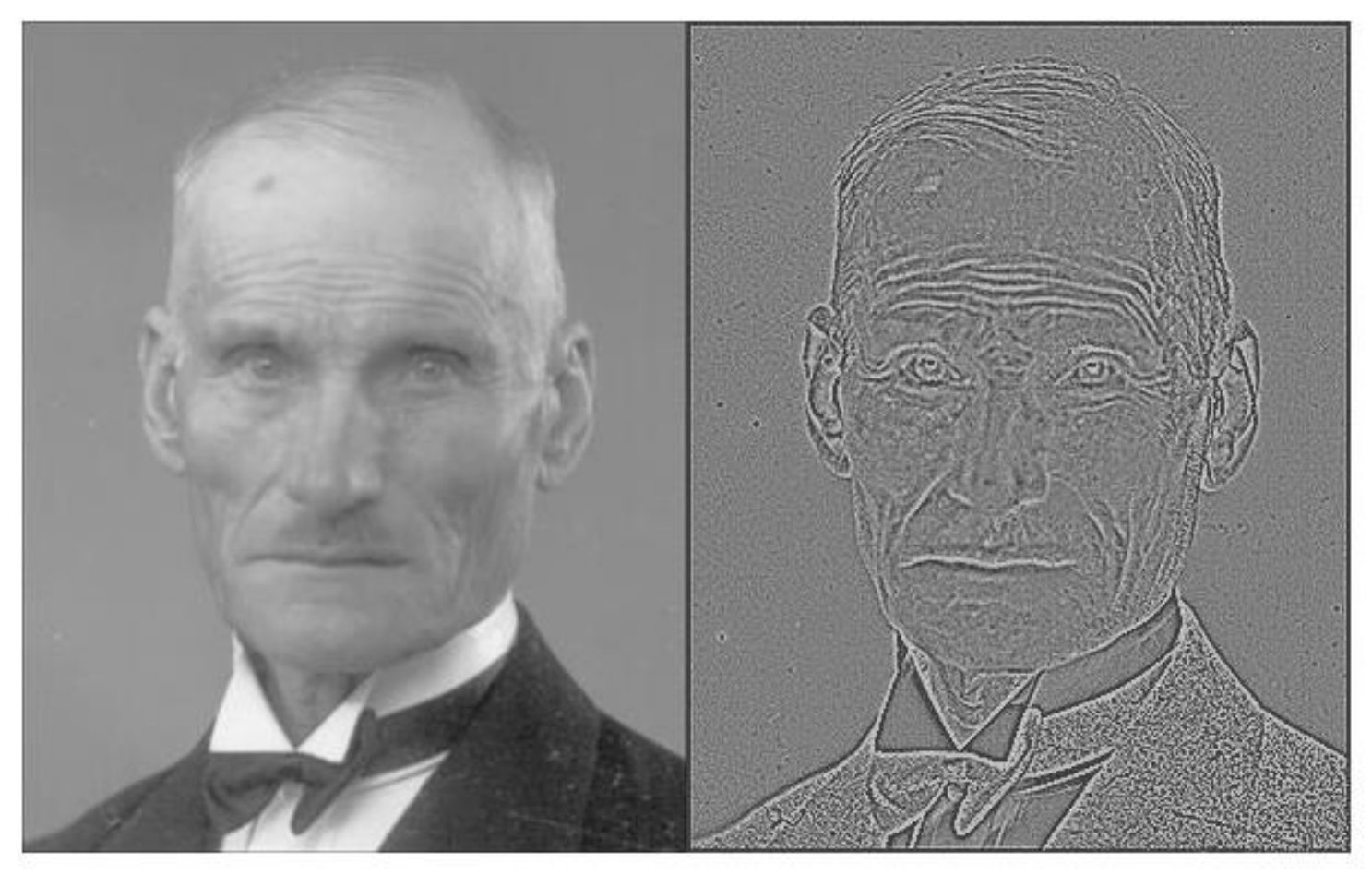
1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Laplacian filter

Example:



The Laplace image has an offset; 0-level is set to 0.5