Friedrich-Alexander-Universität
Technische Fakultät

Lecture 3, Part 1

Image Filtering (Linear Filters)

Computer Vision
Summer Semester 2022

Prof. Bernhard Egger, Prof. Tim Weyrich, Prof. Andreas Maier

Image Filtering

- Linear Filters -- Mean, Convolution, Gaussian
- Filter separability, etc

Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

$$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

- Really important!
  - Enhance images → Denoise, resize, increase contrast, etc.
  - Extract information from images → Texture, edges, distinctive points, etc.
  - Detect patterns → Template matching

Key properties of linear filters

- **Linearity:**

  $\texttt{imfilter(I, } f_1 + f_2\texttt{) = imfilter(I,}f_1\texttt{) + imfilter(I,}f_2\texttt{)}$

- **Shift invariance:**

  Same behavior regardless of pixel location

  $\texttt{imfilter(I,shift(f)) = shift(imfilter(I,f))}$

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

Correlation and Convolution

- 2d correlation $$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

`h=filter2D(f,I)` / `h=imfilter(I,f)` **[matlab]**

- 2d convolution $$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

`h=conv2(f,I)` / `h=imfilter(I,f,'conv')` **[matlab]**

`conv2(I,f)` is the same as `filter2(rot90(f,2),I)`
Correlation and convolution are identical when the filter is symmetric.

Correlation and Convolution

- 2d correlation $$h[m,n] = \sum_{k,l} f[k,l]\, I[m+k, n+l]$$

```
h = cv2.filter2D(f,I)
```
[OpenCV]

- 2d convolution $$h[m,n] = \sum_{k,l} f[k,l]\, I[m-k, n-l]$$

```
h = cv2.filter2D(cv2.flip(f),I)
```
[OpenCV]

Convolution properties

The mathematical symbol for convolution is the asterisk operator ($*$) – not to be confused with multiplication in source code (*)!

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality, e.g., image edges

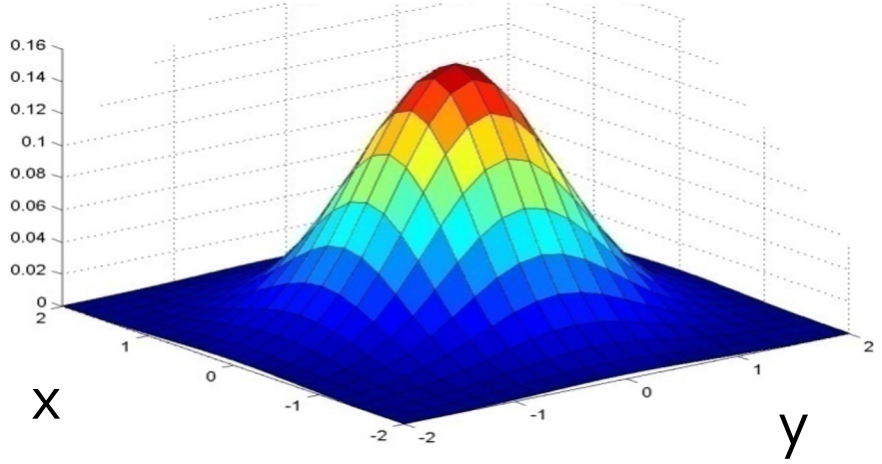
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
  - *Correlation* is ***not*** associative (rotation effect)
  - Why important?

Convolution properties

- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k(a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0] \Rightarrow a * e = a$

Important filter: Gaussian

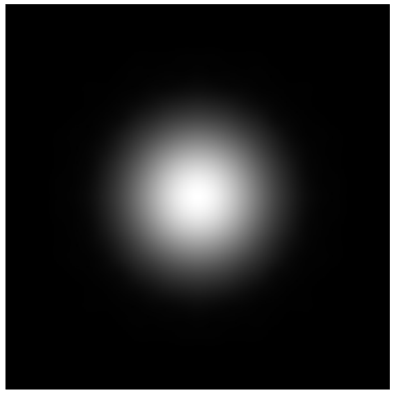
- Weight contributions of neighboring pixels by nearness



x

y					
	0.003	0.013	0.022	0.013	0.003
	0.013	0.059	0.097	0.059	0.013
	0.022	0.097	0.159	0.097	0.022
	0.013	0.059	0.097	0.059	0.013
	0.003	0.013	0.022	0.013	0.003

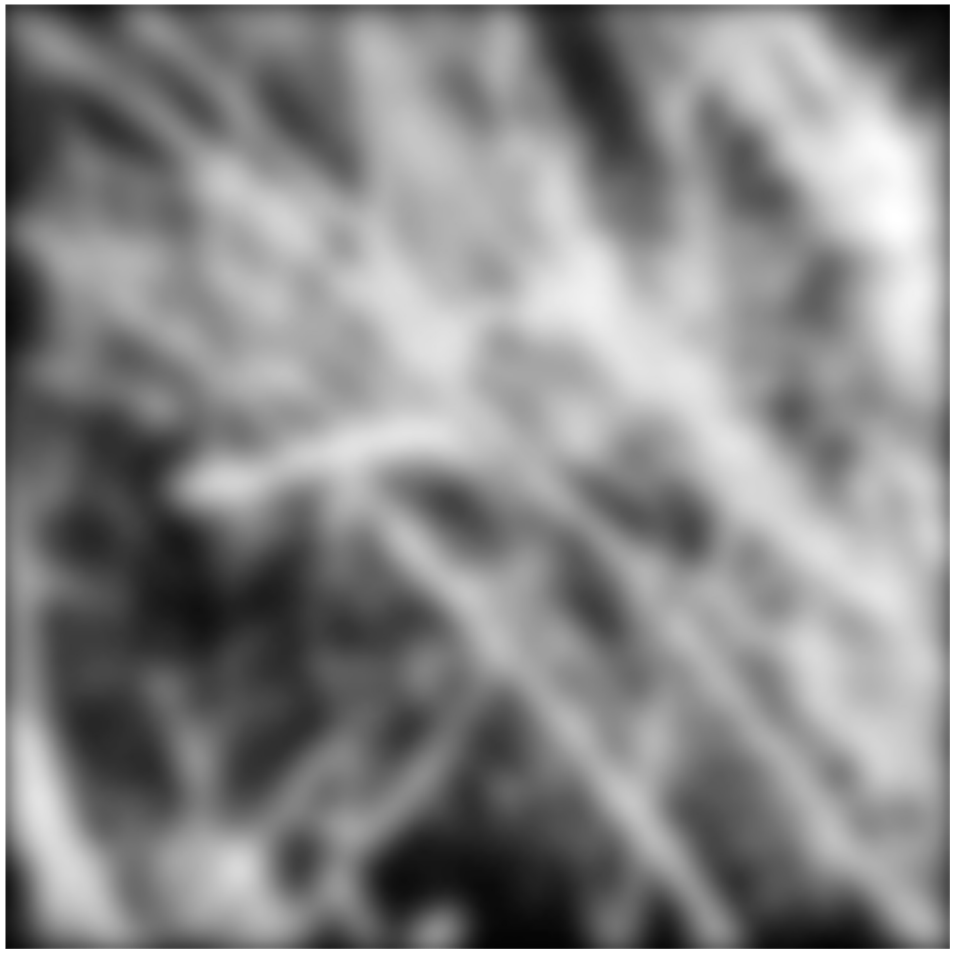
5×5, $\sigma = 1$



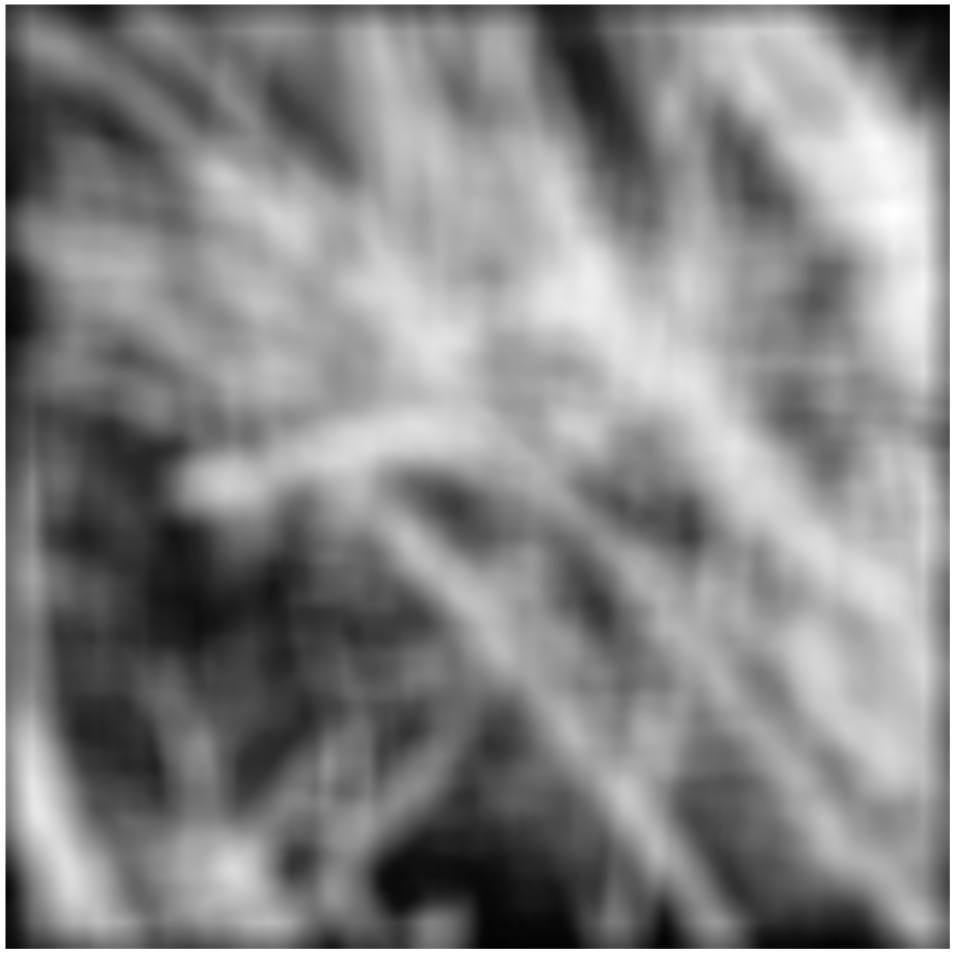
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

James
Hays

Smoothing with Gaussian filter



Smoothing with box filter



Properties of Gaussian filters

- Low-pass filter
  - Remove high-frequency components from the image
  - Images become more smooth
- Gaussian convolved with Gaussian…

  …is another Gaussian
- Convolving two times with Gaussian of width σ is same as convolving once with width $\sigma\sqrt{2}$
- *Separable kernel*
  - Factors into product of two 1D Gaussians

Separability of the Gaussian filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

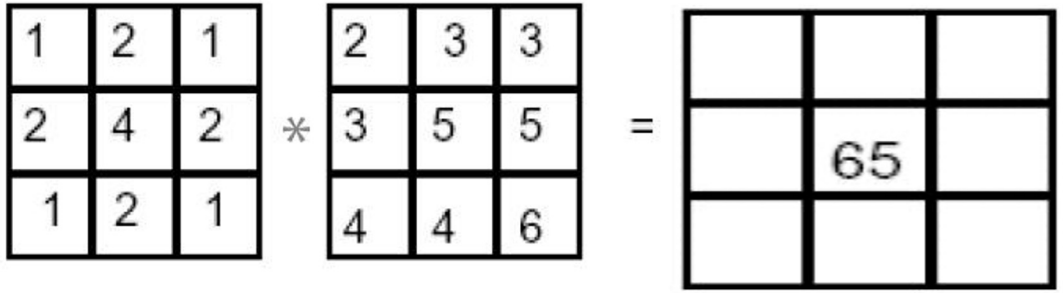
$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

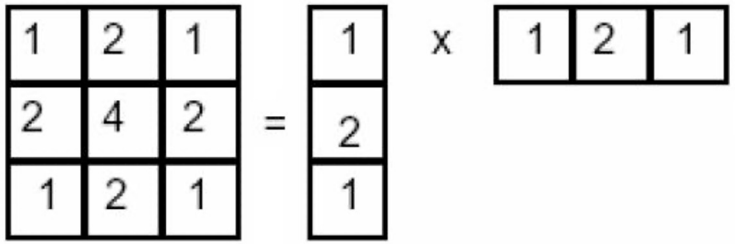
In this case, the two functions are the (identical) 1D Gaussian

Separability: numerical example

2D convolution
(center location only)



The filter factors into a product of 1D filters:



Separability: numerical example

Perform convolution along rows:

1	2	1

*

2	3	3
3	5	5
4	4	6

=

	11	
	18	
	18	

Followed by convolution along the remaining column:

1
2
1

*

	11	
	18	
	18	

=

	65	

Separability

Why is separability useful in practice?

MxN image, PxQ filter

- 2D convolution: ~MNPQ multiply-adds (computational complexity)
- Separable 2D: ~MN(P+Q) multiply-adds (computational complexity)

Speed up = PQ/(P+Q)

9×9 filter = ~4.5× faster

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Linear Filters – Examples

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

James Hays

Linear Filters – Examples

Original

0	0	0
0	0	1
0	0	0

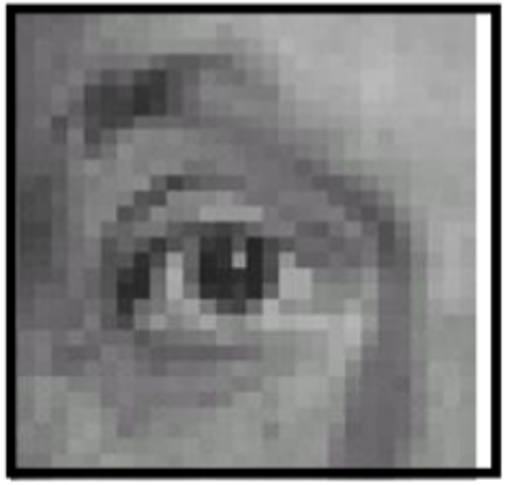
?

James Hays

Linear Filters – Examples

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

James Hays

Example: Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

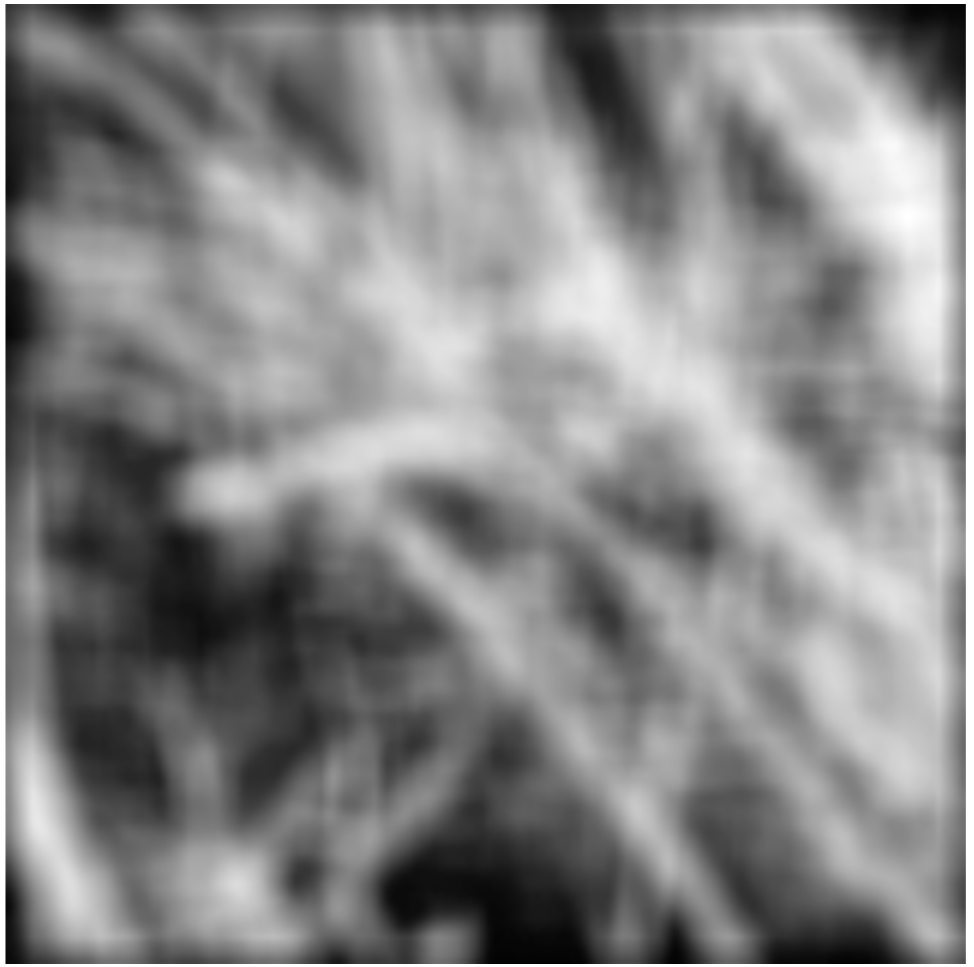
- Achieve smoothing effect (remove sharp features)

- Why does it sum to one?

$$f[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smoothing with box filter



James Hays

Box Filter – Steps

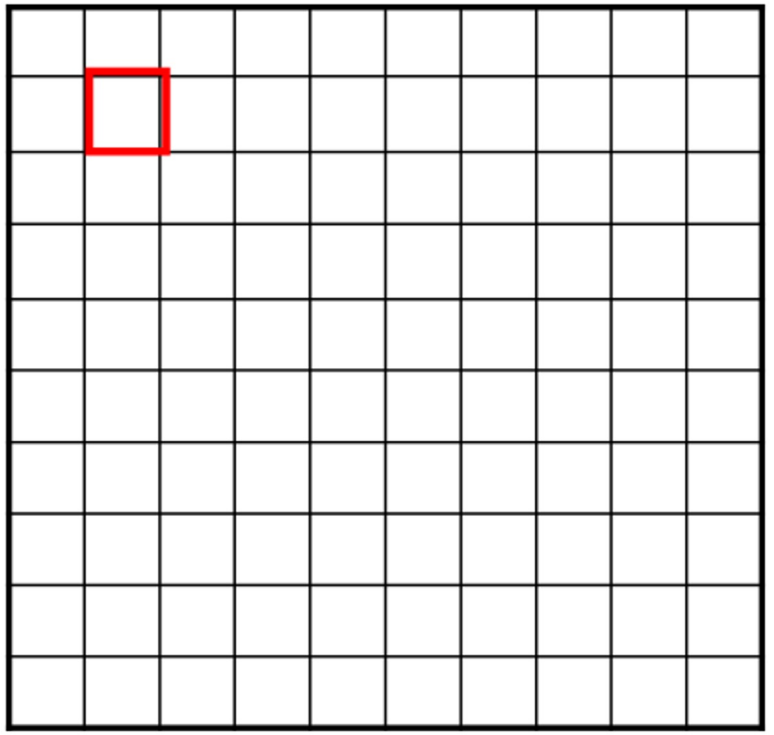
$f[\cdot,\cdot]$

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$



Box Filter – Steps

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

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							

Box Filter – Steps

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

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					

Box Filter – Steps

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

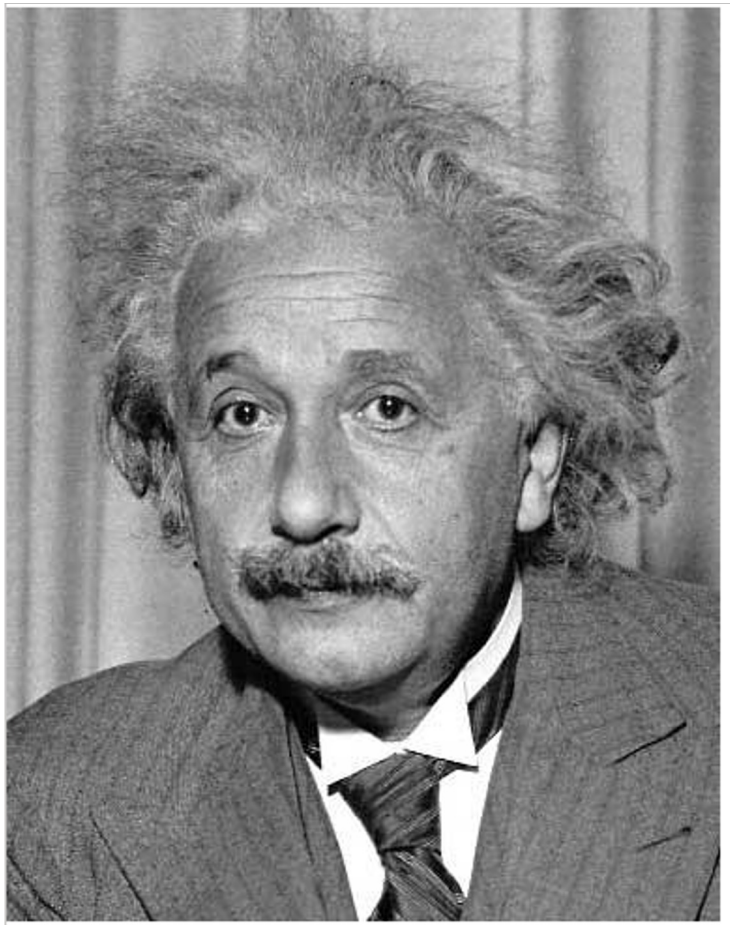
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

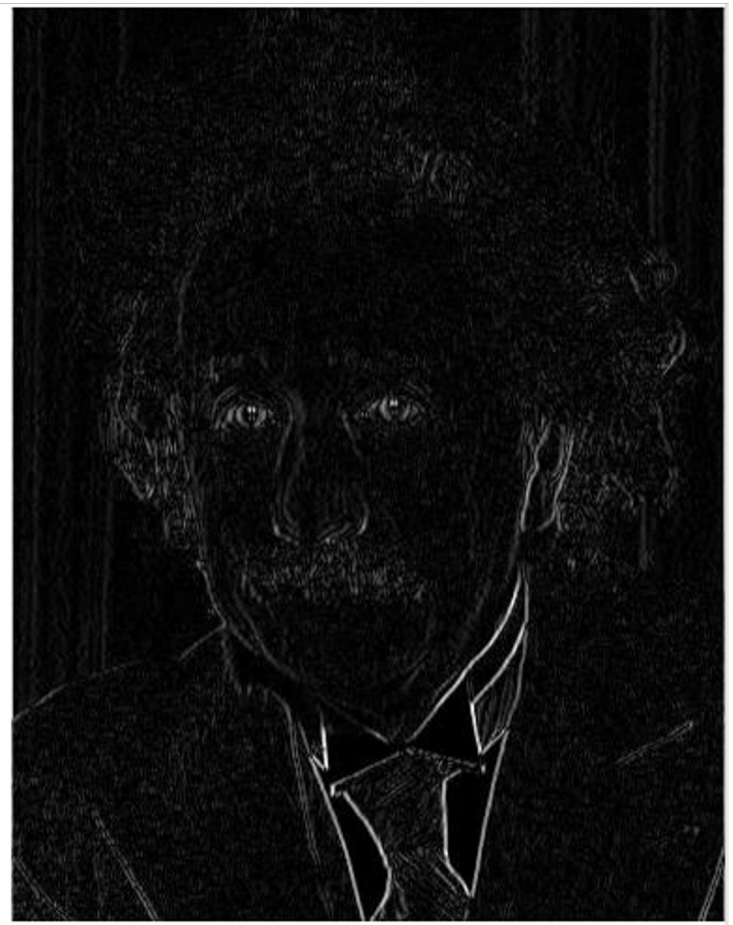
James Hays

Sobel filter



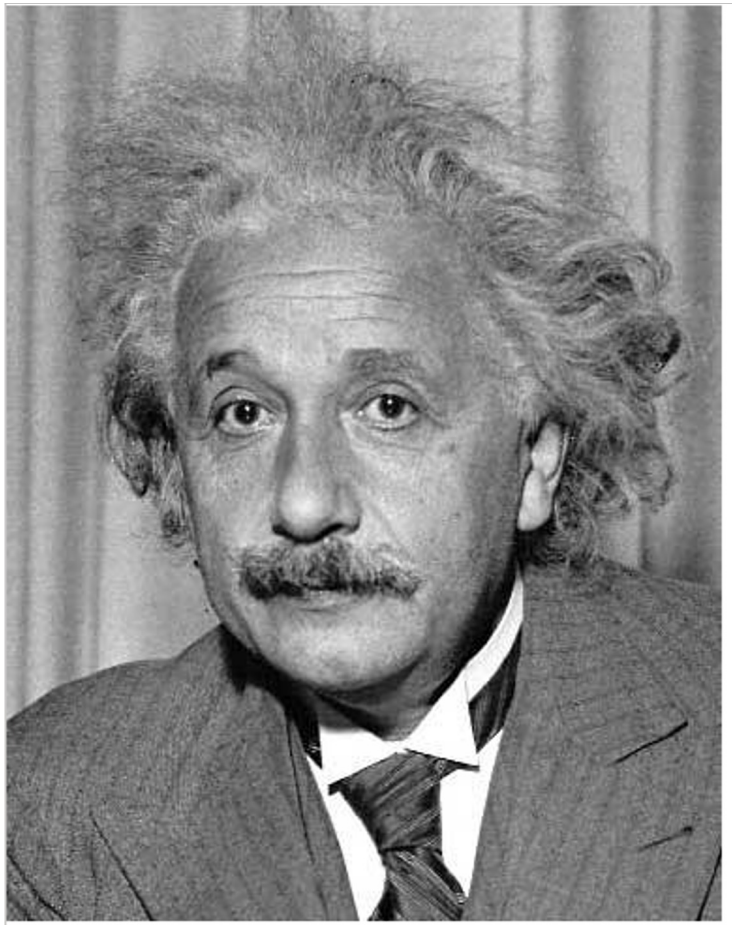
1	0	-1
2	0	-2
1	0	-1

Sobel



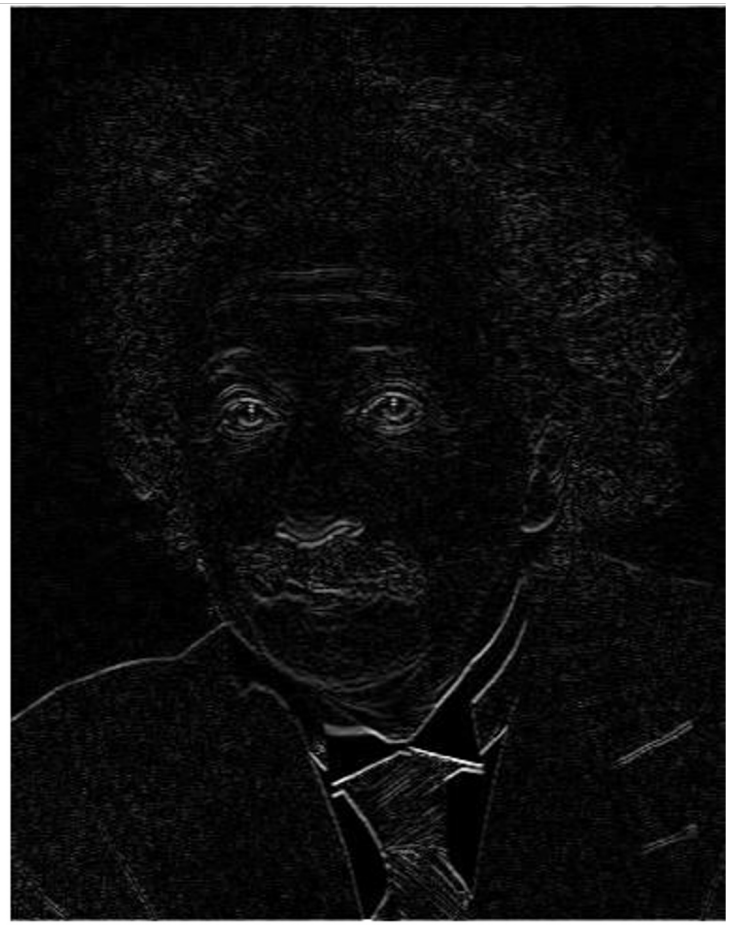
Vertical Edge
(absolute value)

Sobel filter



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

Sobel



Horizontal Edge
(absolute value)

Sobel filter

- Central differences
  - Edge enhancing
  - Slightly smoothing
  - Positive and negative output values

```
[-1 0 1]
[-2 0 2]
[-1 0 1]
```

```
>> I = im2double( imread( 'luke.jpg' ) );
>> h = imfilter( I, fspecial('sobel') );
```

Sobel filter

```
[-1 0 1]
[-2 0 2]
[-1 0 1]
```

```
imshow( h );
```

```
imshow( h + 0.5 );
```

Sobel filter

`h(:,:,1) < 0`

`h(:,:,1) > 0`

Convolution in Convolutional Neural Networks

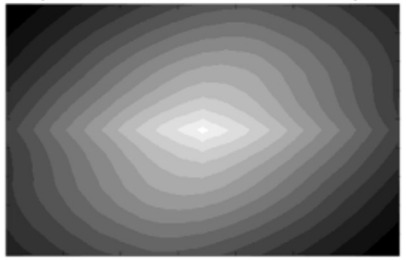
- Convolution is the basic operation in CNNs
- Learning convolution kernels allows us to learn which `features' provide useful information in images.

What does it '*really*' mean to convolve?

$$I = D * D$$

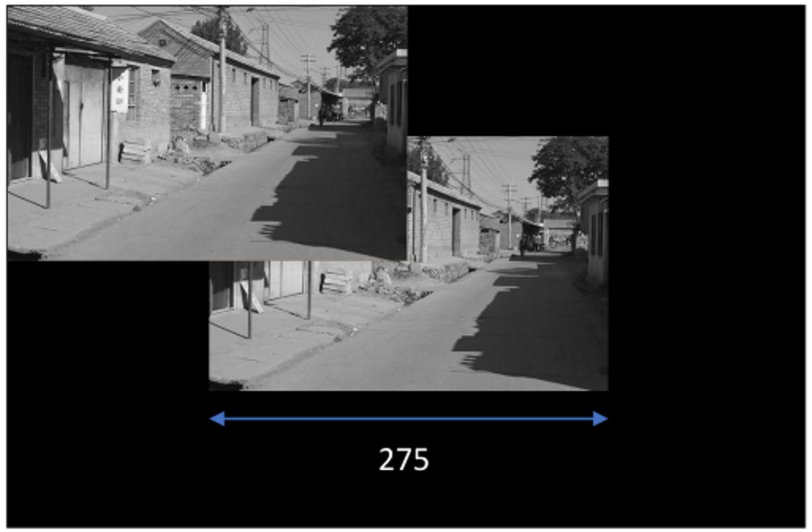
D (275 x 175 pixels)

I (from slide – 275 x 175)



(275-1)/2

(275-1)/2

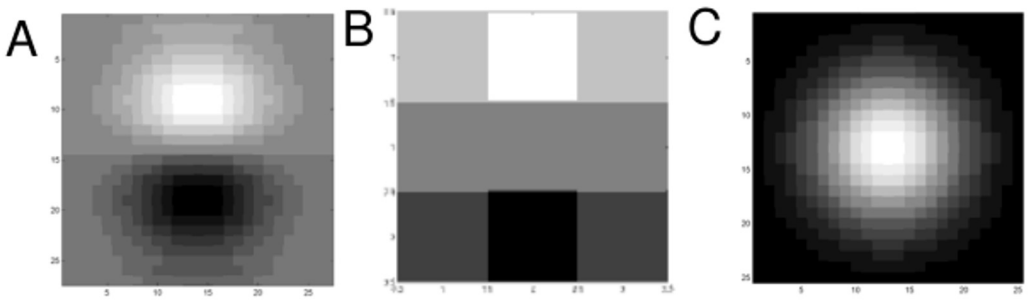


For x: 275 + (275-1)/2 + (275-1)/2
= 549

I_norm (549 x 349 pixels)

Convolution Vs. Correlation

A = B * C – “because it kind of looks like it.”



C is a Gaussian filter
(or something close to it it),
and we know that it ‘blurs’.

When the filter ‘looks like’ the image = ‘template matching’

Filtering viewed as comparing an image of
what you want to find against all image regions.

Convolution Vs. Correlation

subtract the mean

```
>> f = D[ 57:117, 107:167 ]
>> f2 = f – np.mean(f)
>> D2 = D – np.mean(D)
```

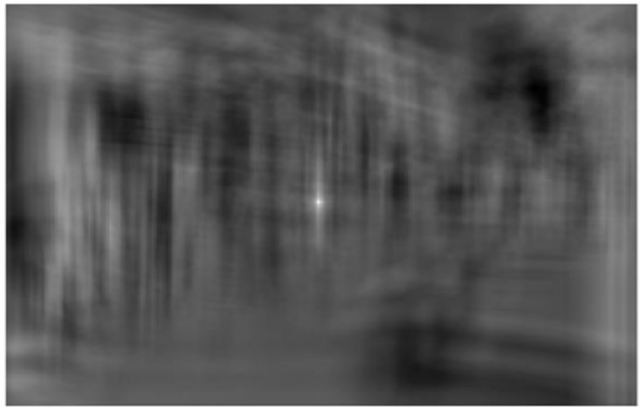
Now zero centered.
Score is higher only when dark parts match and when light parts match.

```
>> I2 = correlate2d( D2, f2, 'same' )
```

D2 (275 x 175 pixels)

f2
61 x 61

I2



Convolution Vs. Correlation

What happens with convolution?

```
>> f = D[ 57:117, 107:167 ]
>> f2 = f – np.mean(f)
>> D2 = D – np.mean(D)

>> I2 = convolve2d( D2, f2, ‘same’ )
```

D2 (275 x 175 pixels)

f2
61 x 61

I2