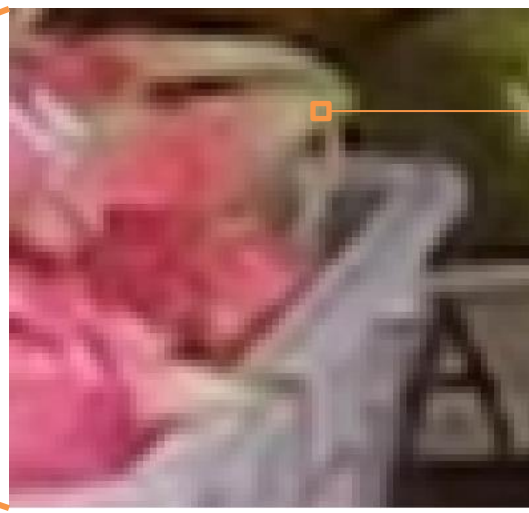


IMAGE FILTERING

HYUN SOO PARK

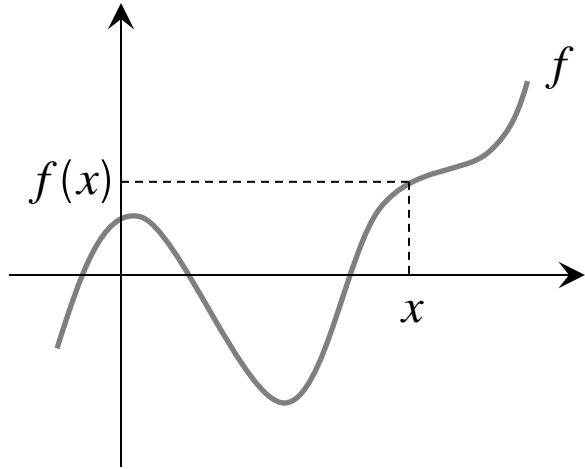


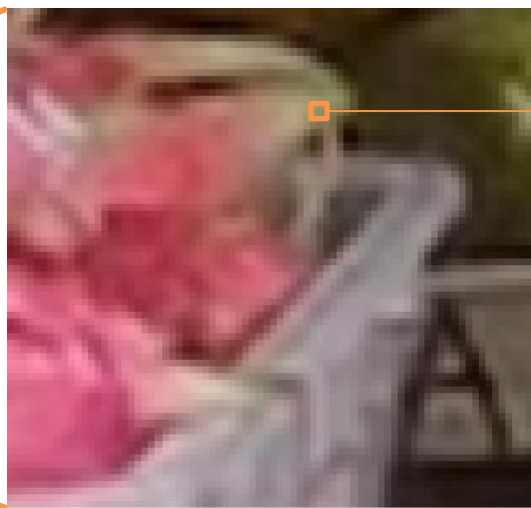




$I(x_1)$

Pixel value at x





$I(x_1)$

Pixel value at x

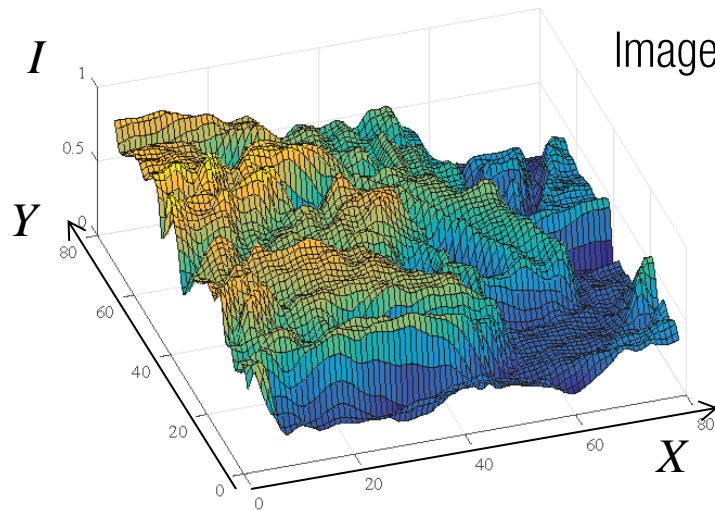
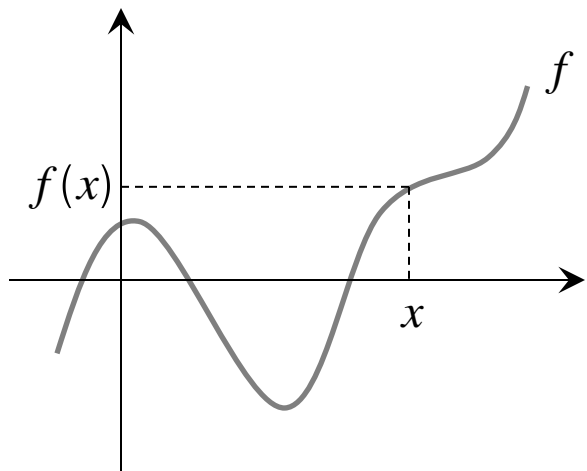
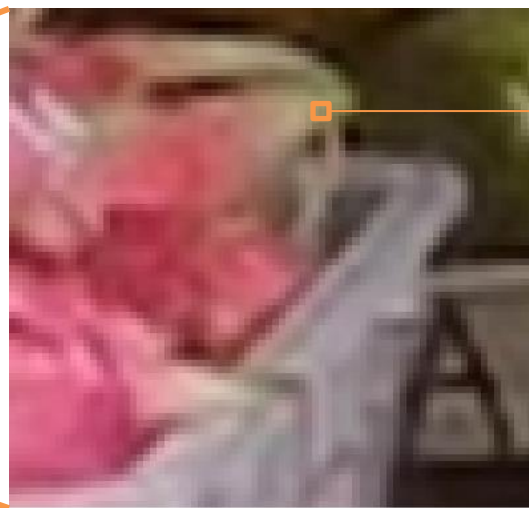


Image as a 2D function



$$I(x_1) = (212, 200, 221)$$

Pixel value at x

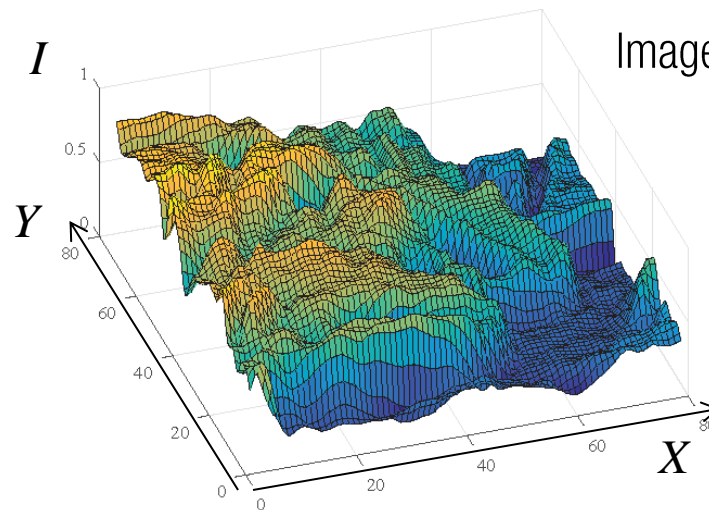
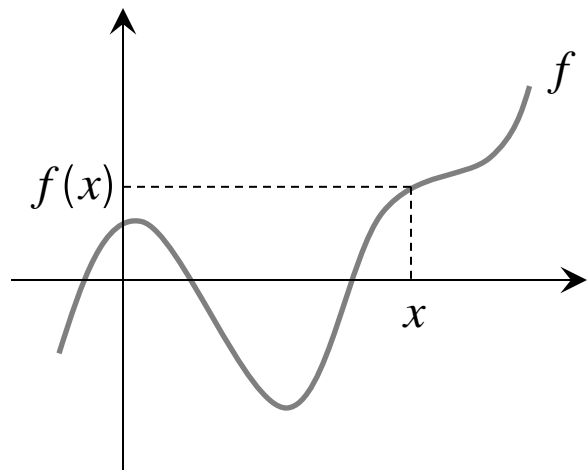
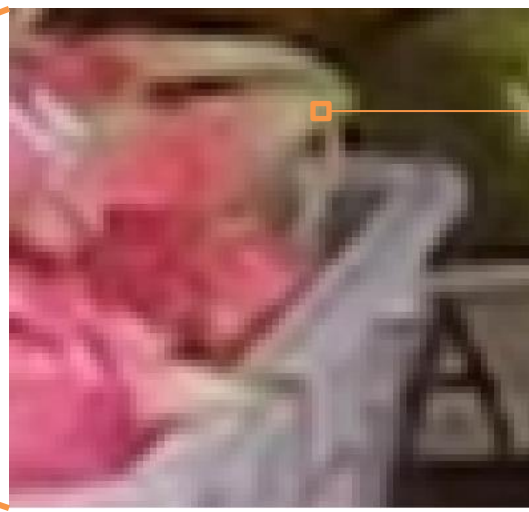


Image as a 2D function



$$I(x_1) = (212, 200, 221)$$

Pixel value at x

Select Color:

The interface displays a color selection tool. It includes a large color gradient area with a white circle indicating the selected color. To the right is a vertical color bar. Below these are several input fields: a color swatch, a white swatch, and a small red swatch. The fields are: H: 0°, S: 57%, B: 69%, R: 175, G: 75, B: 75, and a hex code field containing #AF4B4B.

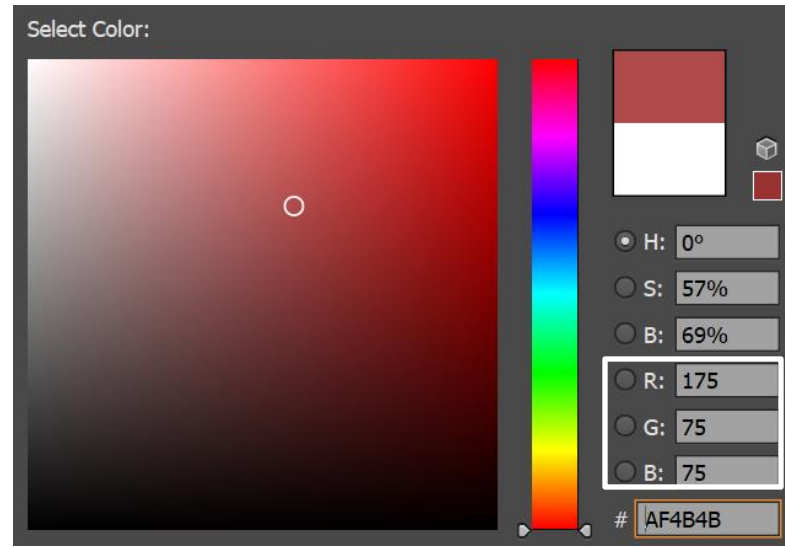
H:	0°
S:	57%
B:	69%
R:	175
G:	75
B:	75
#	AF4B4B



$I(x_1) = (212, 200, 221)$

Pixel value at x

$I(x_2) = (212, 20, 51)$





$$I(x_1) = (212, 200, 221)$$

Pixel value at x

$$I(x_2) = (212, 20, 51)$$

$I(x) \in 2^8 \times 2^8 \times 2^8$
8-bit image

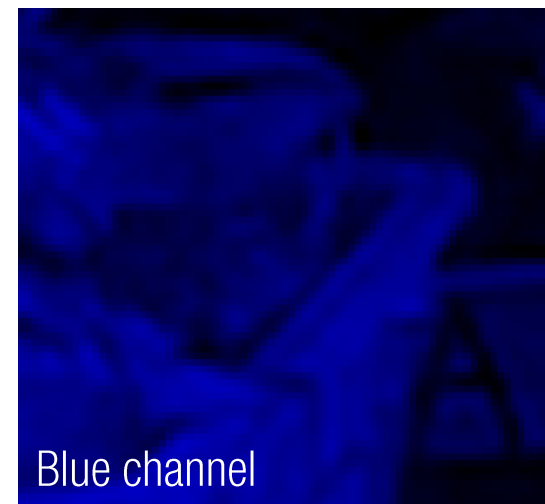
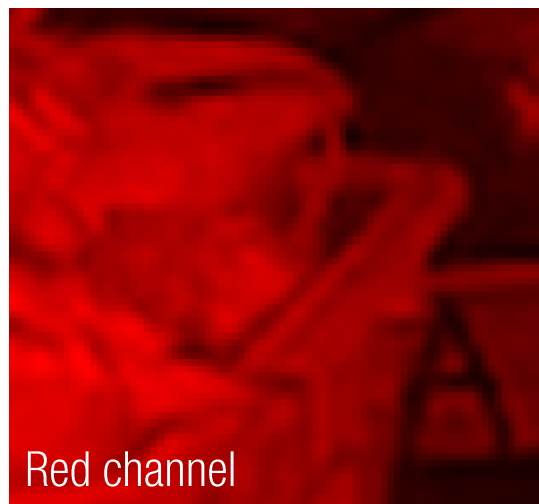


IMAGE FILTERING



$$f(I) = I_f$$

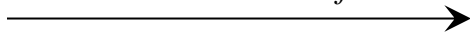


IMAGE FILTERING



$$I(x_1) = (212, 200, 221)$$

$$f(I) = I_f$$



$$\begin{aligned} f(I(x_1)) &= f(212, 200, 221) \\ &= (242, 152, 135) \end{aligned}$$

IMAGE FILTERING (PIXEL-WISE FILTERING)



$$I_f = I + 100$$



$$I_f = I - 100$$



$$I_f = 255 - I$$



$$I_r > 100 \ \& \ I_g < 100 \ \& \ I_b < 100$$

IMAGE FILTERING (PIXEL-WISE FILTERING)



$$I_r > 100 \& I_g < 100 \& I_b < 100$$



Blue/black

White/gold



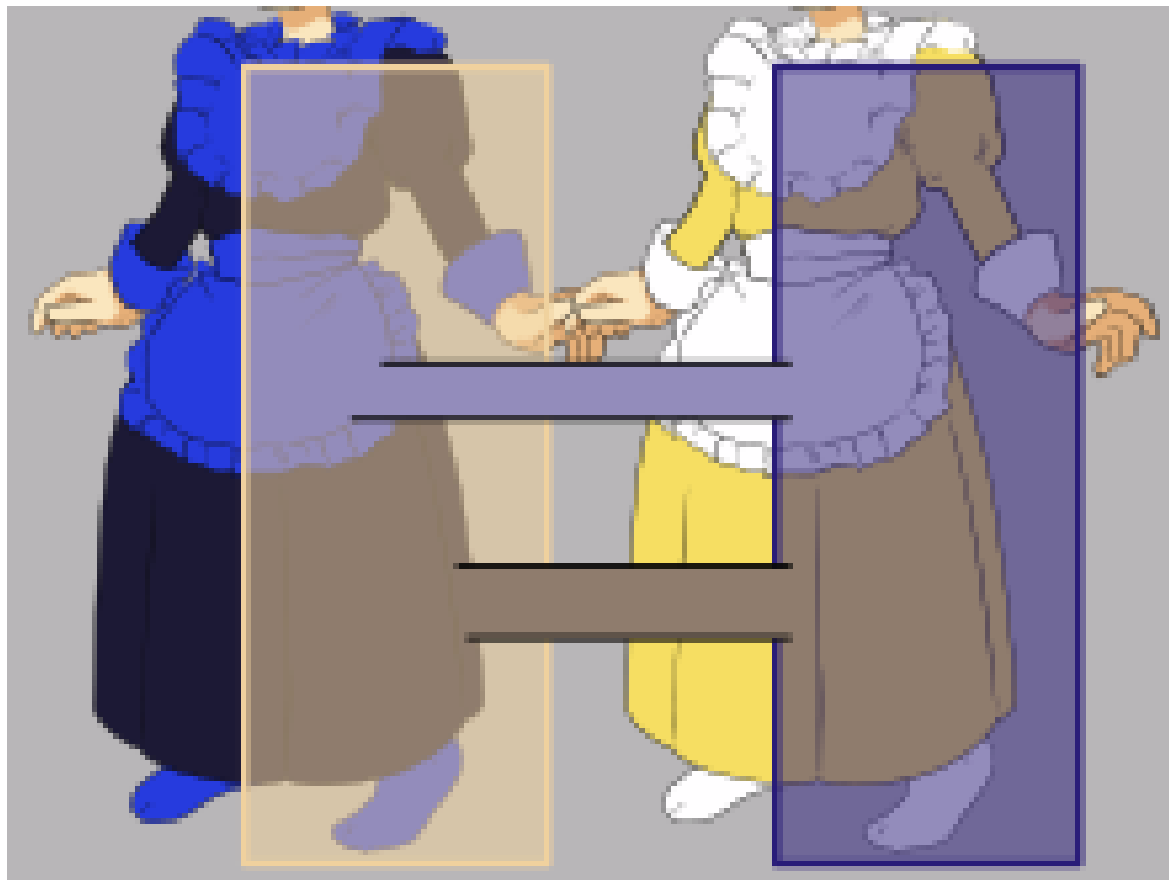


IMAGE FILTERING (PIXEL-WISE FILTERING)



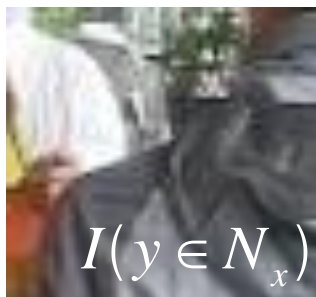
$$I_r > 100 \& I_g < 100 \& I_b < 100$$

IMAGE SPATIAL FILTERING



$$I_f(x) = f(I(y \in N_x))$$

—————→



Local patch around x

N_x : Neighboring pixels of x

IMAGE SPATIAL FILTERING

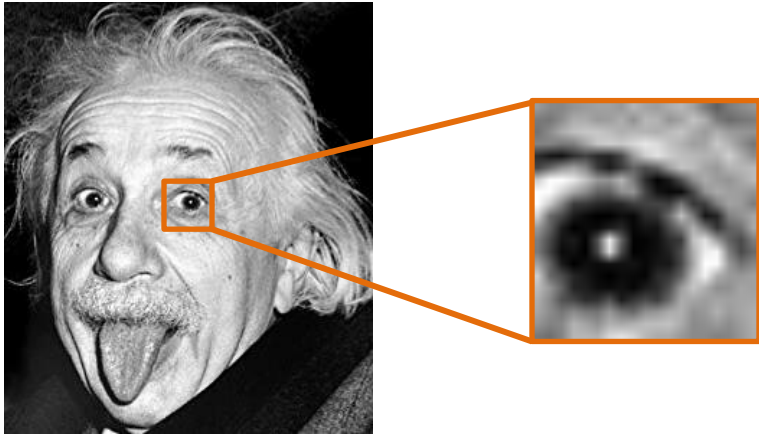
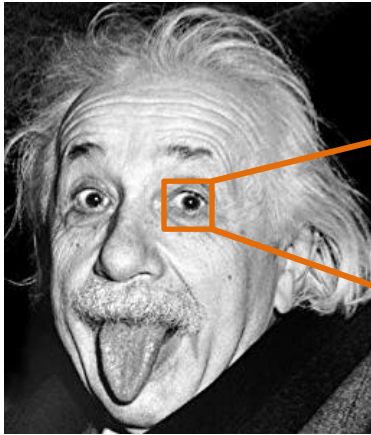


IMAGE SPATIAL FILTERING

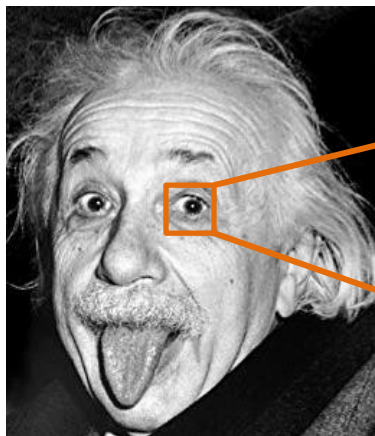


2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

I

Image

IMAGE SPATIAL FILTERING



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

I

Image

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

\otimes

a	b	c
d	e	f
g	h	i

z

Filter

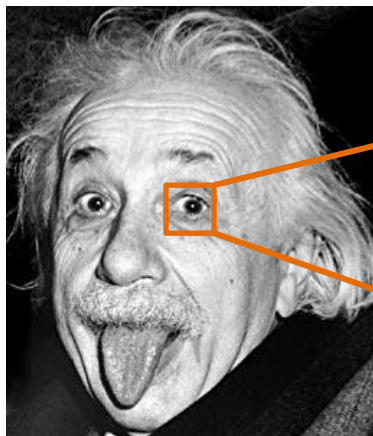
=

I_f

Filtered image

IMAGE SPATIAL FILTERING

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

Image

⊗

<i>a</i>	<i>b</i>	<i>c</i>
<i>d</i>	<i>e</i>	<i>f</i>
<i>g</i>	<i>h</i>	<i>i</i>

Filter

=

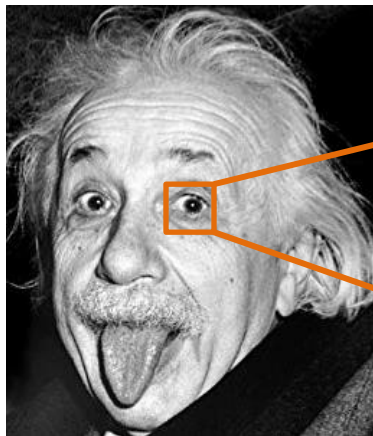
	<i>y</i>			

Filtered image

$$y = 2a + b + 4c + d + 2e + 2f + 3g + 3h + 5i$$

IMAGE SPATIAL FILTERING

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

Image

⊗

<i>a</i>	<i>b</i>	<i>c</i>
<i>d</i>	<i>e</i>	<i>f</i>
<i>g</i>	<i>h</i>	<i>i</i>

Filter

=

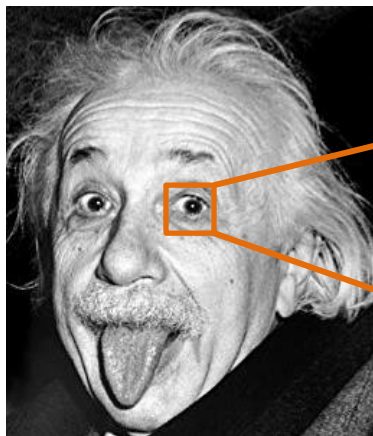
		<i>y</i>		

Filtered image

$$y = a + 4b + 4c + 2d + 2e + 3f + 3g + 5h + 8i$$

IMAGE SPATIAL FILTERING

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

Image

⊗

<i>a</i>	<i>b</i>	<i>c</i>
<i>d</i>	<i>e</i>	<i>f</i>
<i>g</i>	<i>h</i>	<i>i</i>

Filter

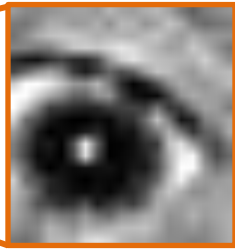
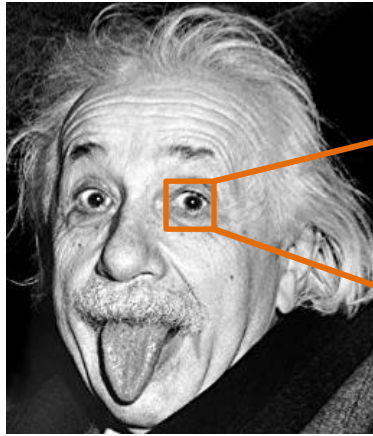
=

			<i>y</i>	

Filtered image

$$y = 5a + 8b + 9c + 2d + 6e + 7f + 2g + h + 3i$$

IDENTITY



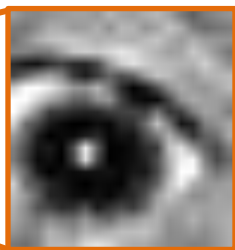
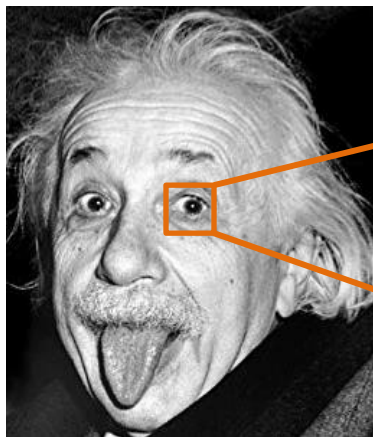
2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

$$\otimes \begin{array}{|c|} \hline \blacksquare \\ \hline \end{array} =$$

$$\otimes \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} =$$

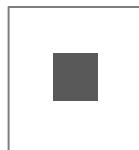
IDENTITY



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

⊗



=



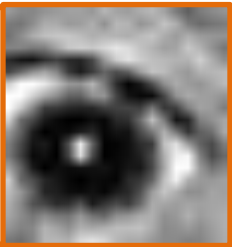
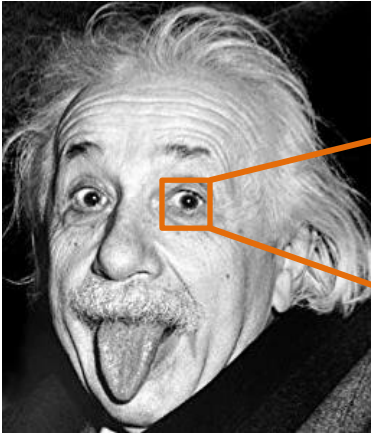
⊗

0	0	0
0	1	0
0	0	0

=

	2	2	3	
	3	5	8	
	2	2	6	

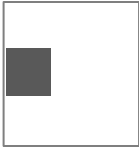
SHIFTING



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

⊗

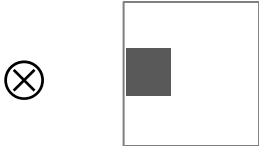
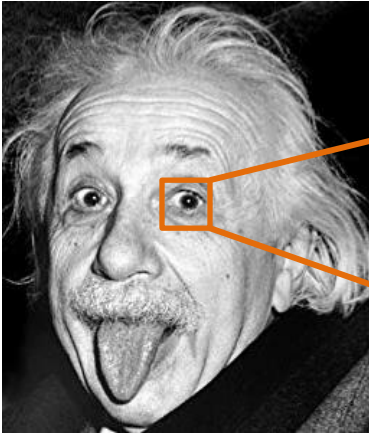


⊗

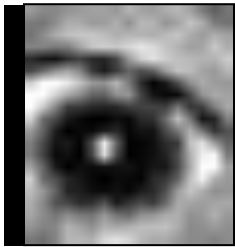
0	0	0
1	0	0
0	0	0

SHIFTING

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



=



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

⊗

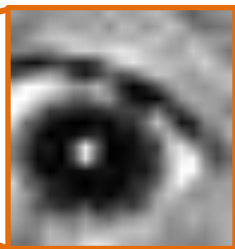
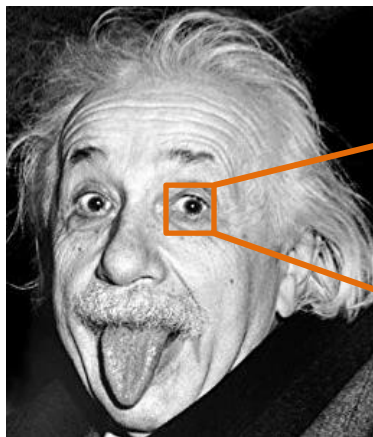
0	0	0
1	0	0
0	0	0

=

	1	2	2	
	3	3	5	
	5	2	2	

BOX FILTER (MOVING AVERAGING)

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



⊗



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

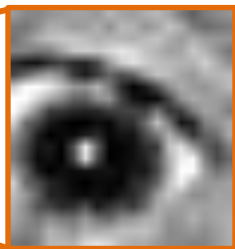
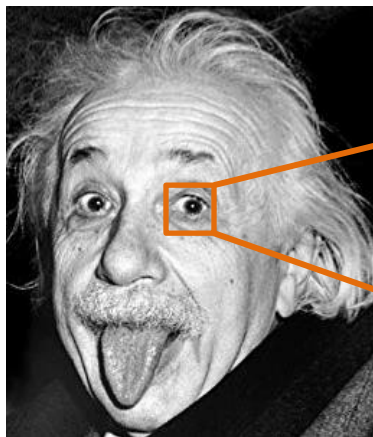
⊗

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

BOX FILTER (MOVING AVERAGING)

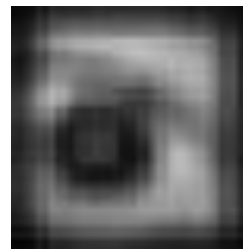
$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



⊗



=



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

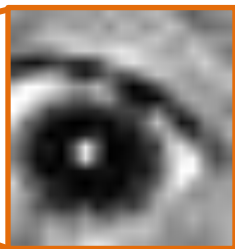
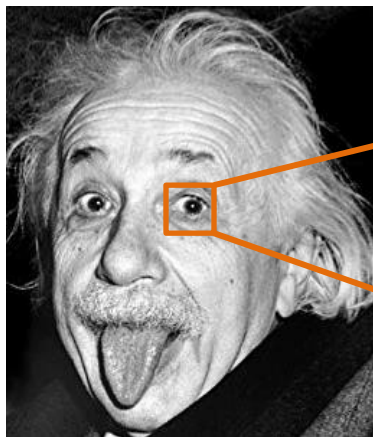
⊗

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Average

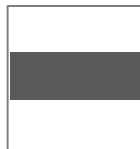
DIRECTIONAL BLUR



2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

⊗

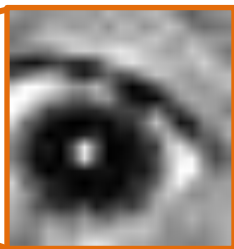
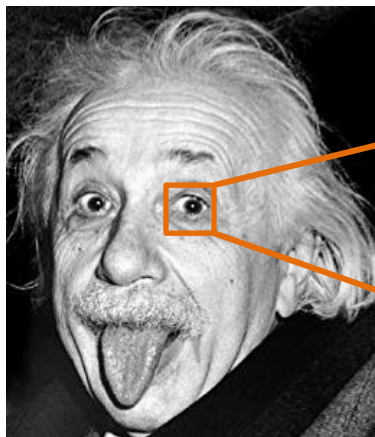


⊗

$\frac{1}{3}$

0	0	0
1	1	1
0	0	0

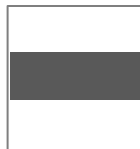
DIRECTIONAL BLUR



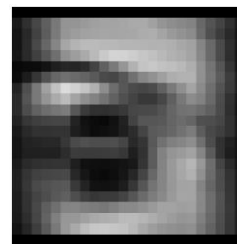
2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

⊗



=



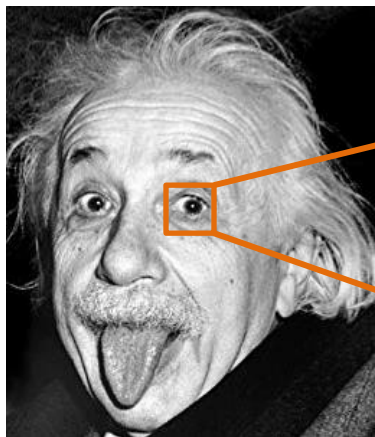
⊗

$\frac{1}{3}$

0	0	0
1	1	1
0	0	0

Horizontal blur

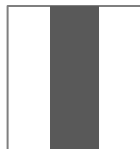
DIRECTIONAL BLUR



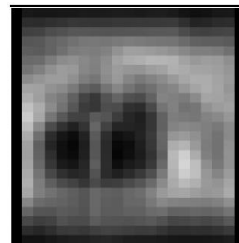
2	1	4	4	7
1	2	2	3	6
3	3	5	8	9
5	2	2	6	7
8	3	2	1	3

$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$

⊗



=



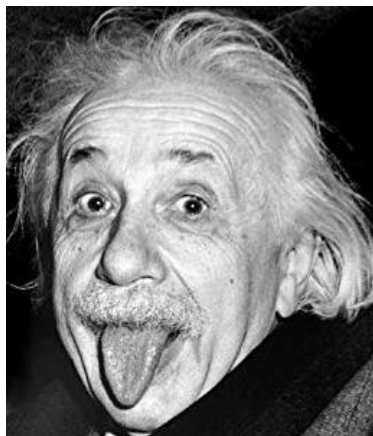
⊗

$\frac{1}{3}$

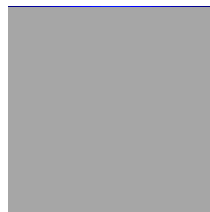
0	1	0
0	1	0
0	1	0

Vertical blur

BOX FILTER (MOVING AVERAGING)



\otimes

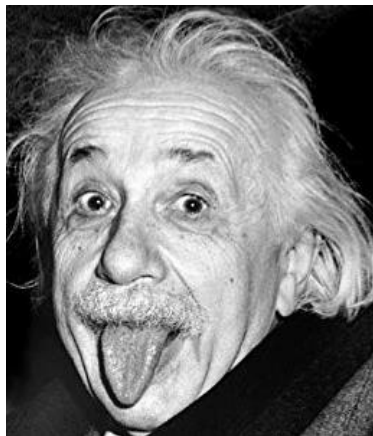


=

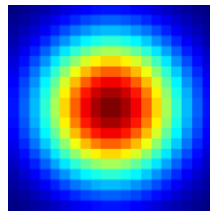
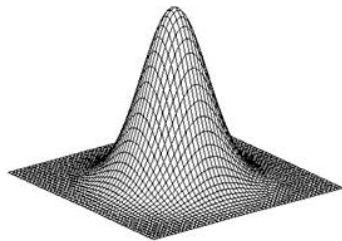


Pixel averaging

GAUSSIAN BLURRING (MOVING WEIGHTED AVERAGE)



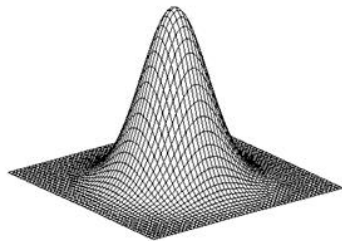
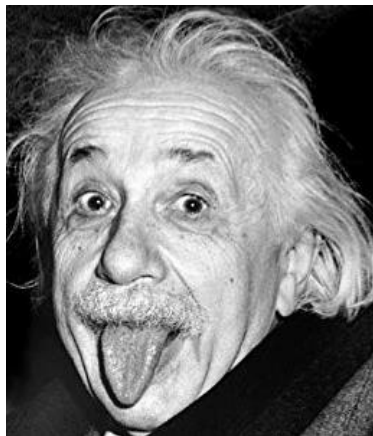
⊗



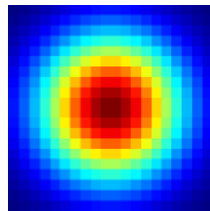
=

$$z(k,l) = \frac{1}{2\pi\sigma^2} e^{-\frac{k^2+l^2}{2\sigma^2}}$$

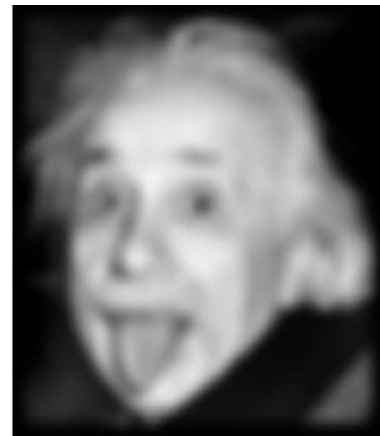
GAUSSIAN BLURRING (MOVING WEIGHTED AVERAGE)



⊗



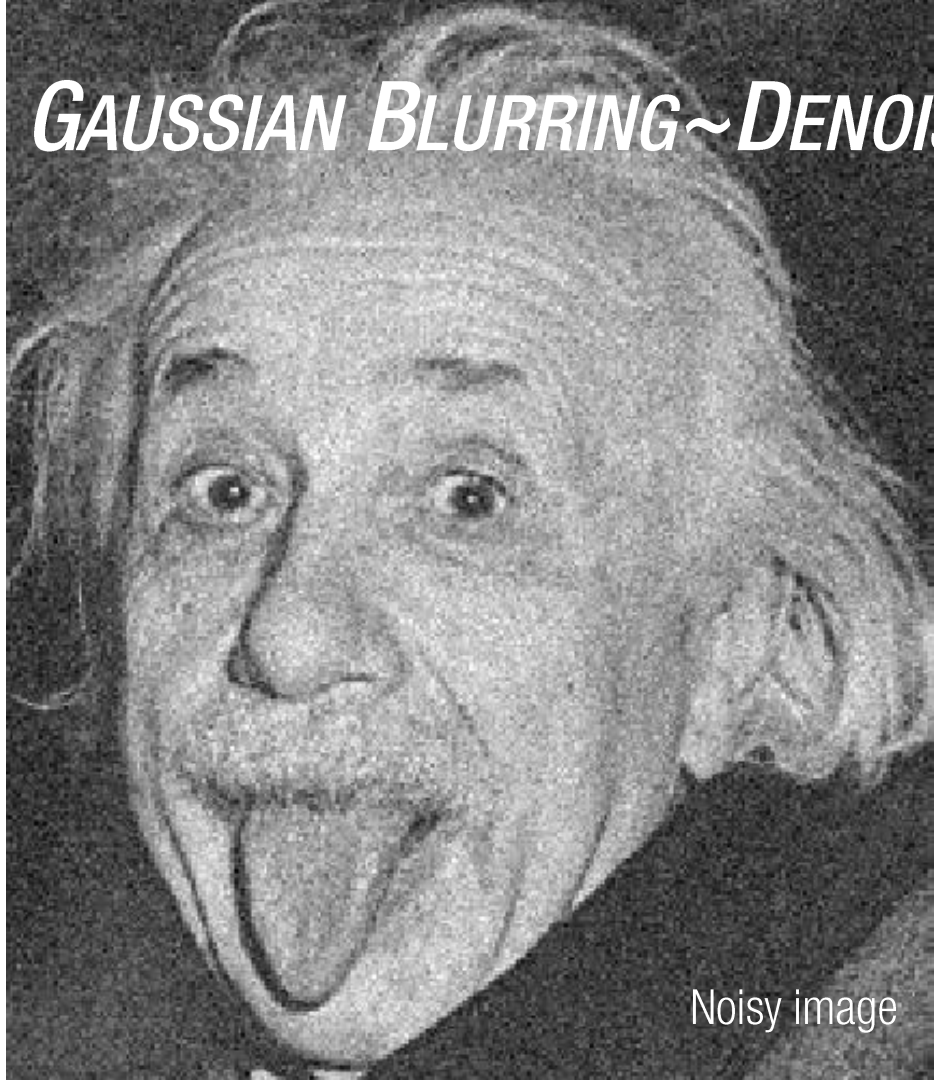
=



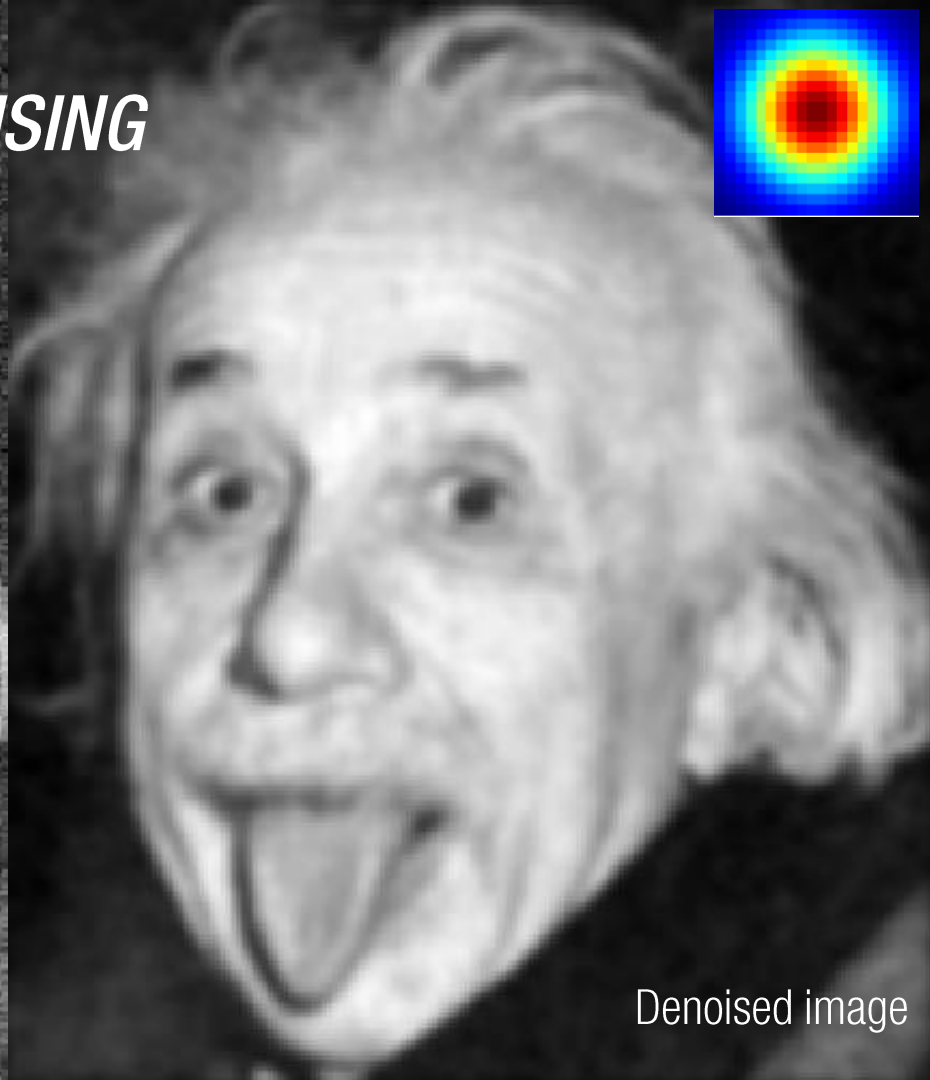
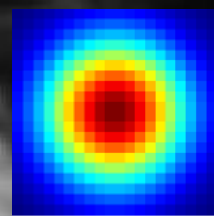
$$z(k,l) = \frac{1}{2\pi\sigma^2} e^{-\frac{k^2+l^2}{2\sigma^2}}$$

Gaussian blur

GAUSSIAN BLURRING ~ DENOISING



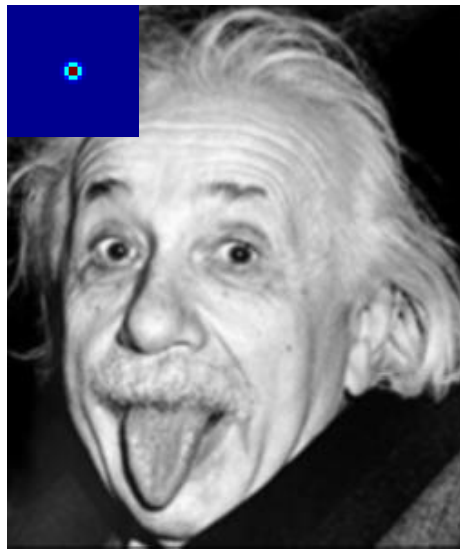
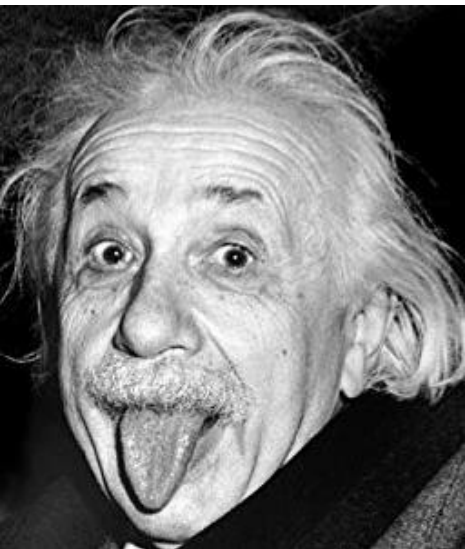
Noisy image



Denoised image

GAUSSIAN BLURRING

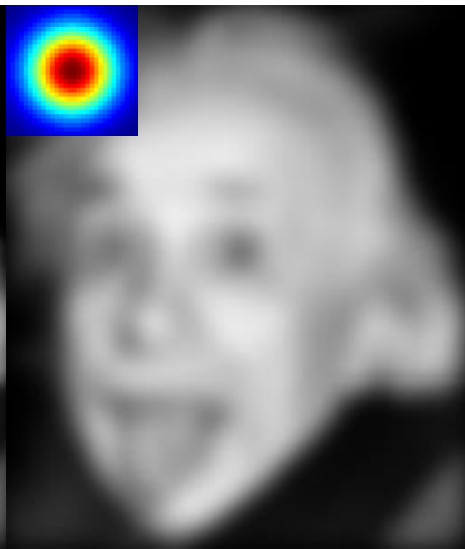
$$z(k,l) = \frac{1}{2\pi\sigma^2} e^{-\frac{k^2+l^2}{2\sigma^2}}$$



$\sigma = 1$



$\sigma = 4$



$\sigma = 7$

IMAGE SPATIAL FILTERING

```
function im_f = Filtering(im, filter)
```

```
% filtered image initialization
```

```
im_f = zeros(size(im));
```

```
center_k = floor(size(filter,1)/2)+1;
```

```
center_l = floor(size(filter,2)/2)+1;
```

```
for i = 1 : size(im,1)
```

```
    for j = 1 : size(im,2)
```

```
        % filtering
```

```
        v = 0;
```

```
        for k = 1 : size(filter,1)
```

```
            for l = 1 : size(filter,2)
```

```
                i1 = i + k - center_k;
```

```
                j1 = j + l - center_l;
```

```
                if i1 <= 0 || i1 > size(im_f,1) || j1 <= 0 || j1 > size(im_f,2)
```

```
                    continue;
```

```
                end
```

```
                v = v + im(i1,j1)*filter(k,l);
```

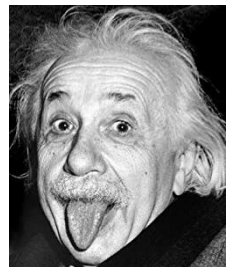
```
            end
```

```
        end
```

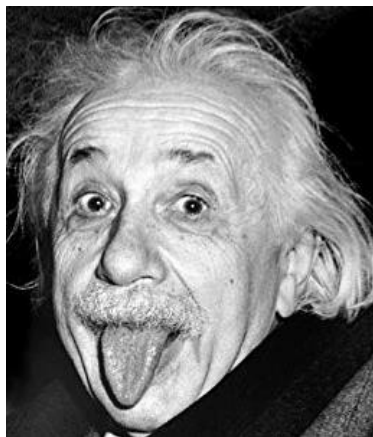
```
        im_f(i,j) = v;
```

```
    end
```

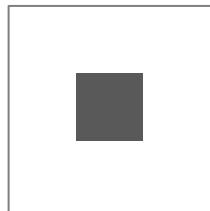
```
end
```



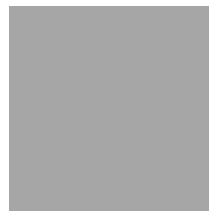
$$I_f(i, j) = \sum_{k,l} I(i+k, j+l)z(k,l)$$



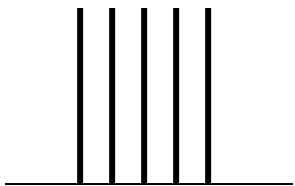
\otimes



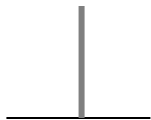
-



=



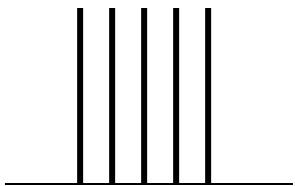
⊗



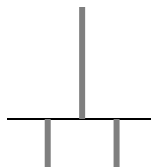
-



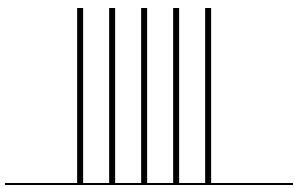
=



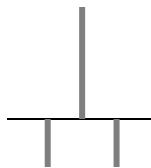
\otimes



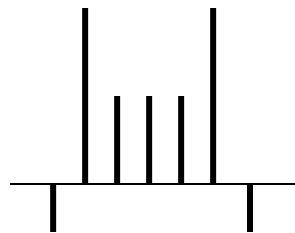
$=$



\otimes

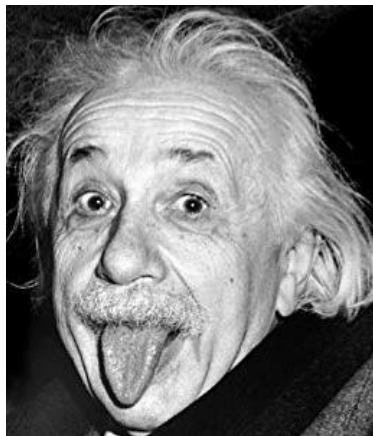


=

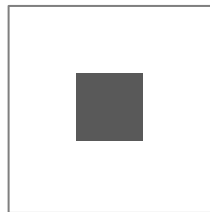


High response at edge

IMAGE SHARPENING



\otimes

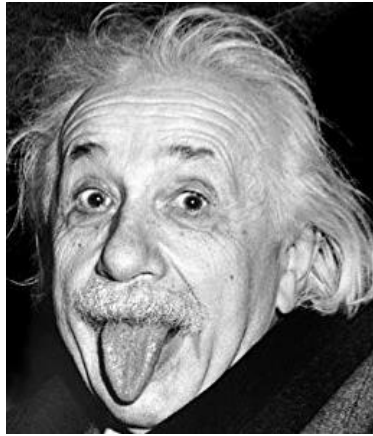


-

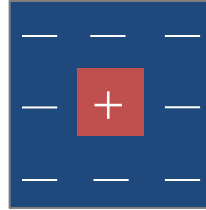


=

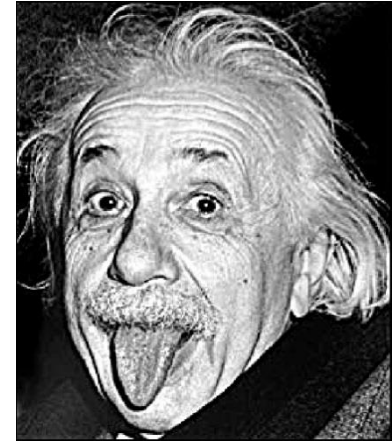
IMAGE SHARPENING



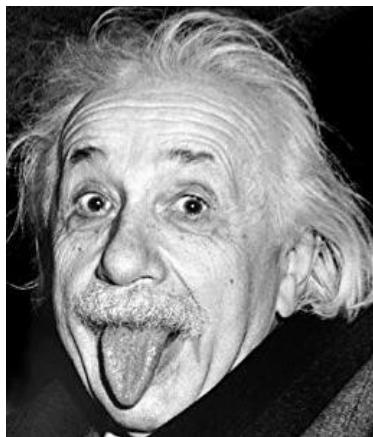
\otimes



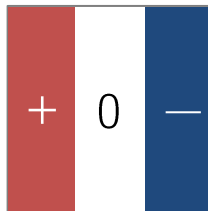
=



Sharpening

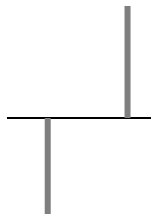
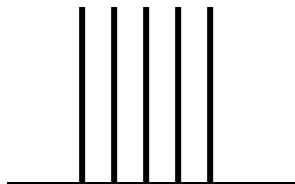


⊗

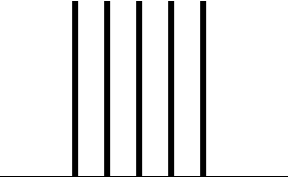


=

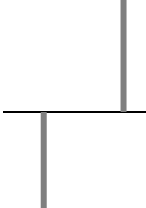
DIFFERENTIATION



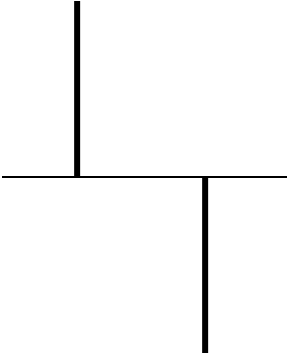
DIFFERENTIATION



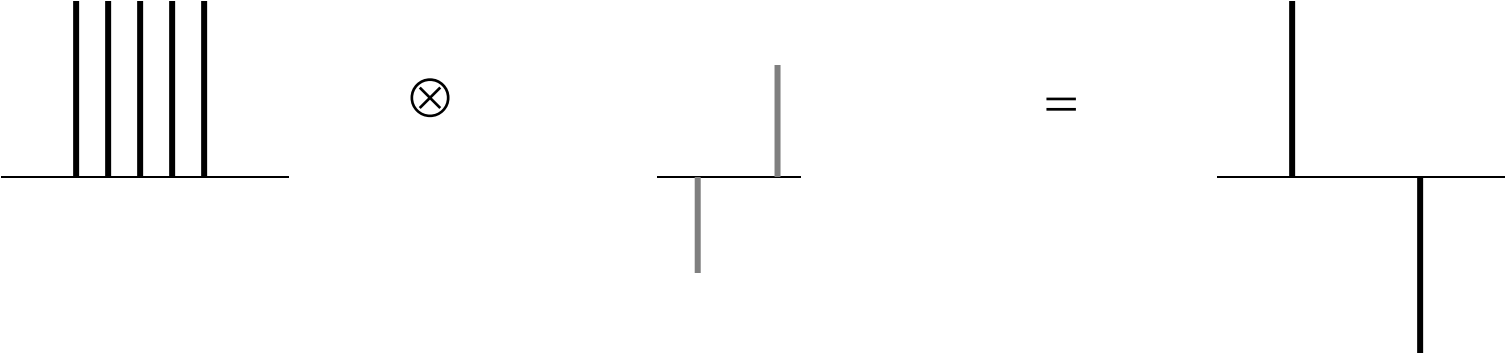
⊗



=



DIFFERENTIATION



$$\frac{df}{du} = \lim_{h \rightarrow 0} \frac{f(u+h) - f(u-h)}{2h} \longrightarrow I \otimes z = \frac{\partial I}{\partial u}$$

DIFFERENTIATION

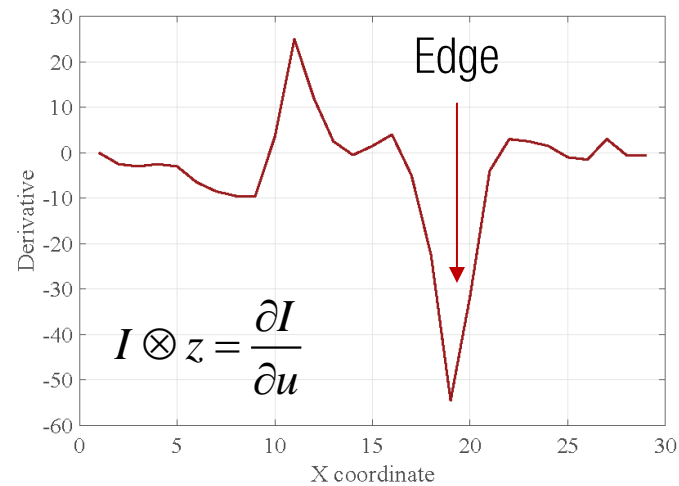
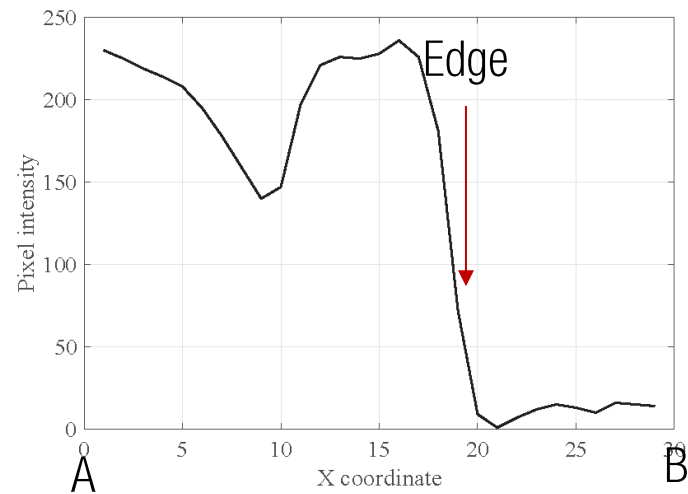
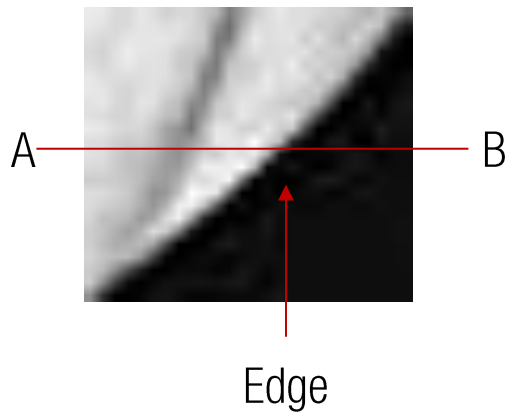
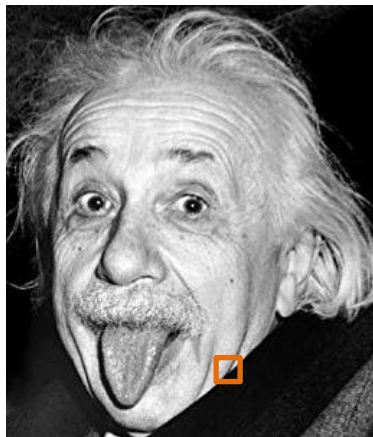
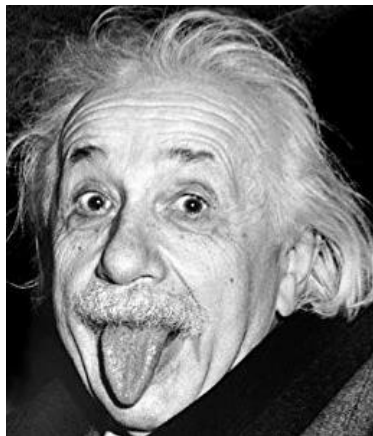
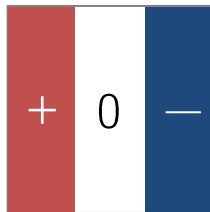


IMAGE DIFFERENTIATION

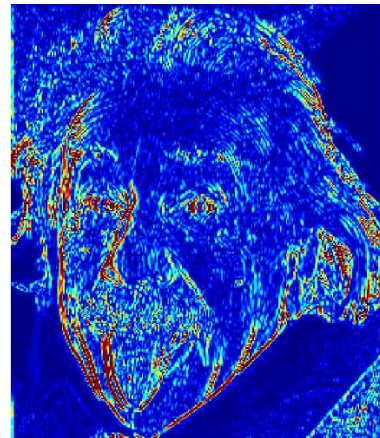


\otimes



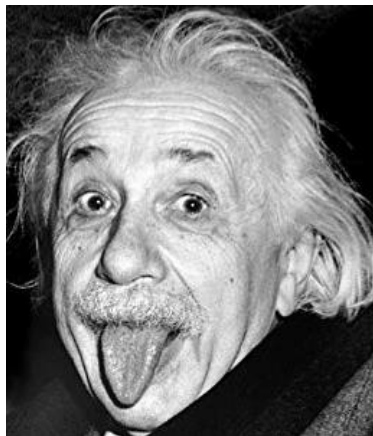
=

$$I \otimes z = \frac{\partial I}{\partial u}$$

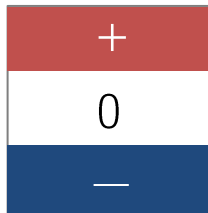


Differentiation

IMAGE DIFFERENTIATION

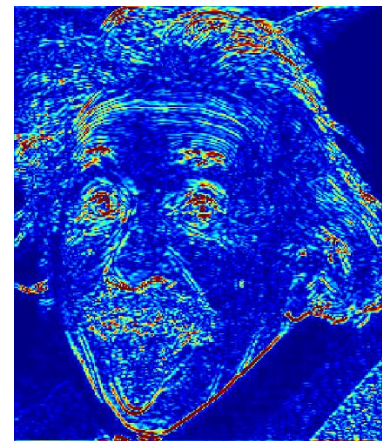


\otimes



=

$$I \otimes z = \frac{\partial I}{\partial v}$$



Differentiation

EDGE RESPONSE

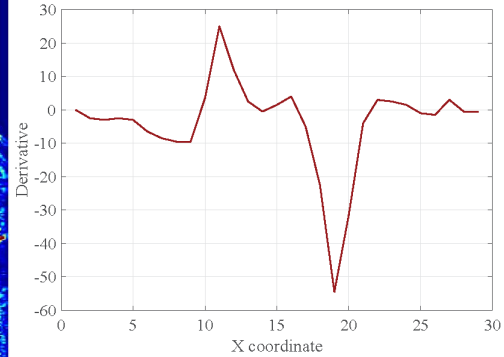
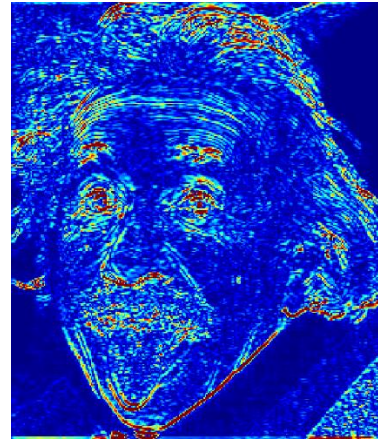
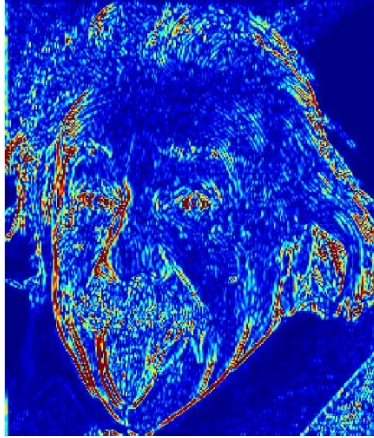
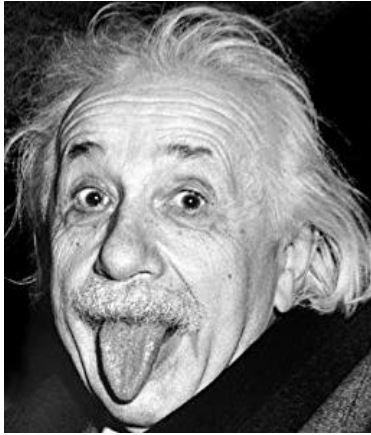


Image differentiation can be used for edge detection but it is VERY noisy.

NONLINEAR FILTER

