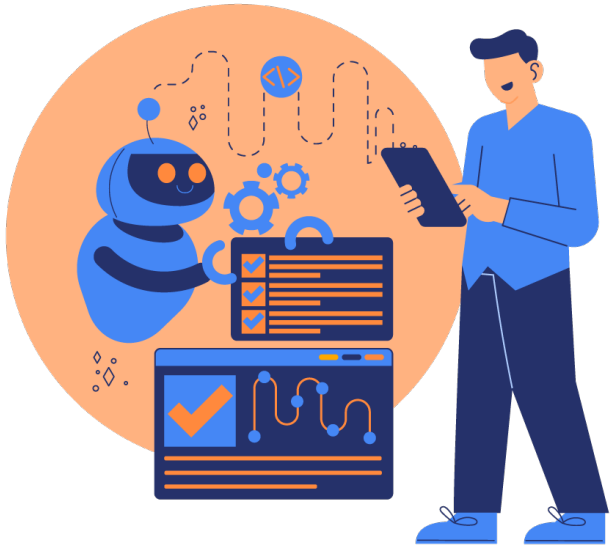
Image Processing

Dr. Mitchell Olsthoorn

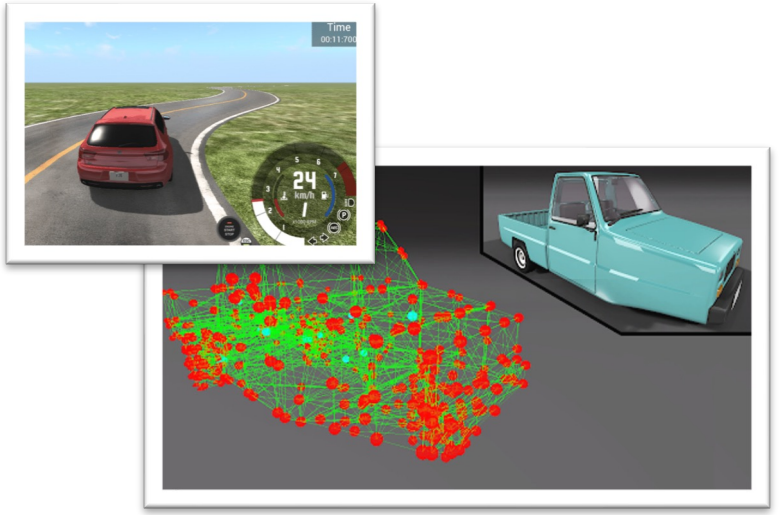
Dr. Annibale Panichella

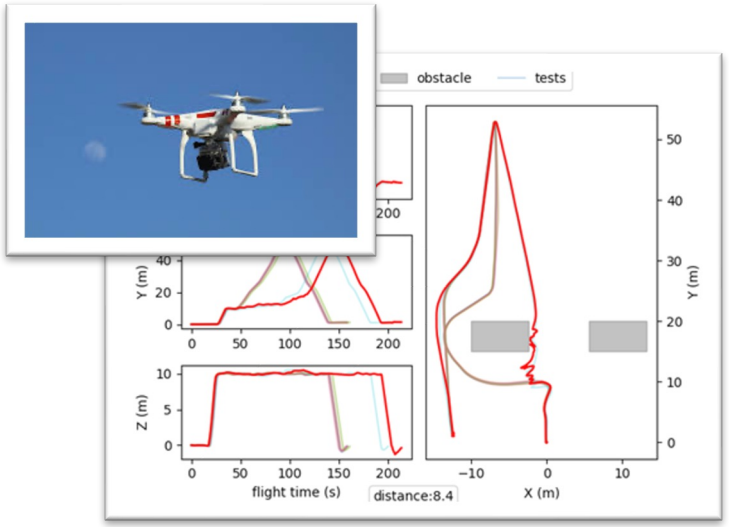
TU Delft

Dr. Mitchell Olsthoorn



Dr. Annibale Panichella





Overview

01

Digital Images

02

Image Filtering

03

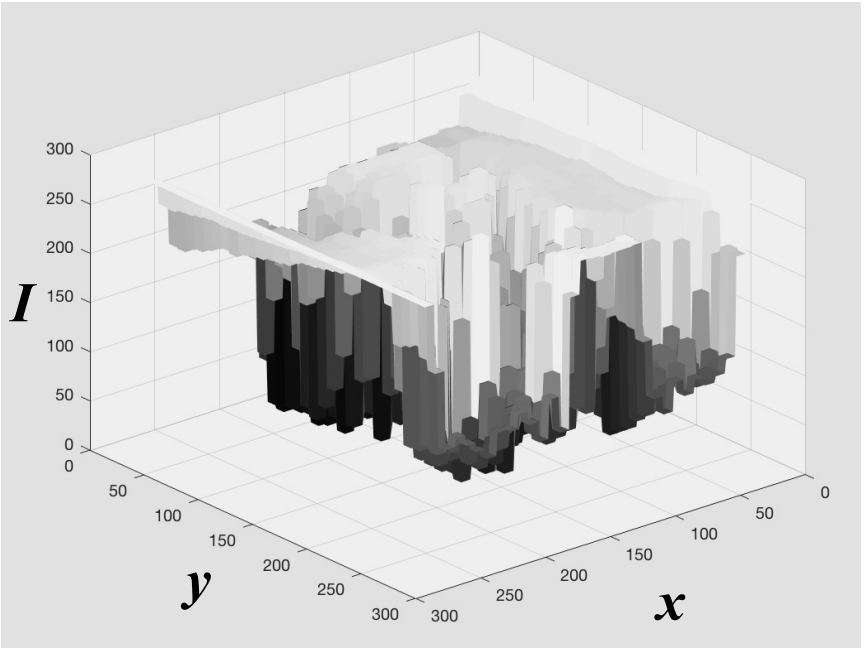
Advanced Use Cases

01

What is an Image?

Digital Images

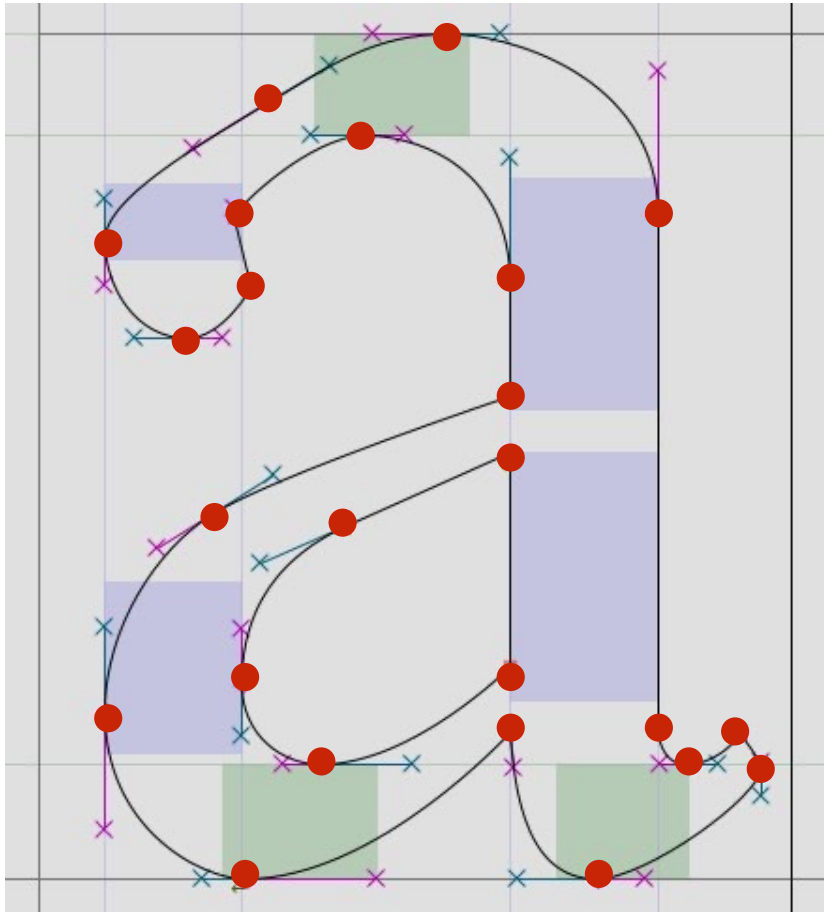
- A digital images is a two-dimensional arrays of numbers
- Each number (pixel) represents the intensity (I) of the colors



Vector Images

- Images are modeled using mathematical formulas: like polygons, curves, lines.
- The quality of the vector images is preserved when they are zoomed in
- Used by word processors (Microsoft Word), video-games, etc.

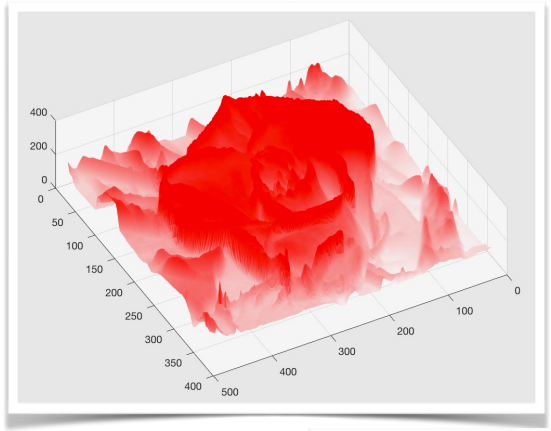
Control Points

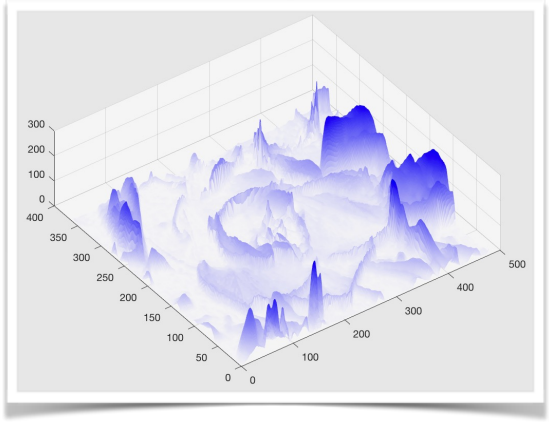


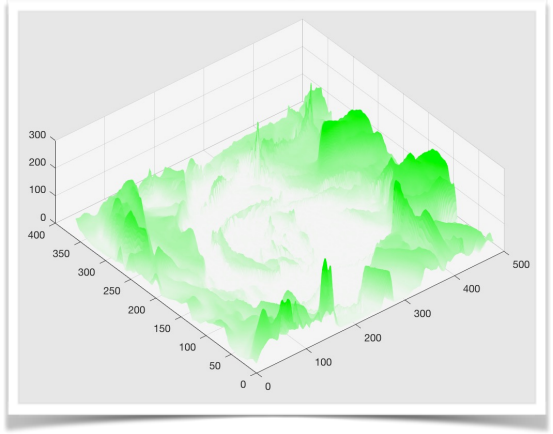
Color Digital Images

- A color images is a three-dimensional grid: each pixel is described with three colors =>

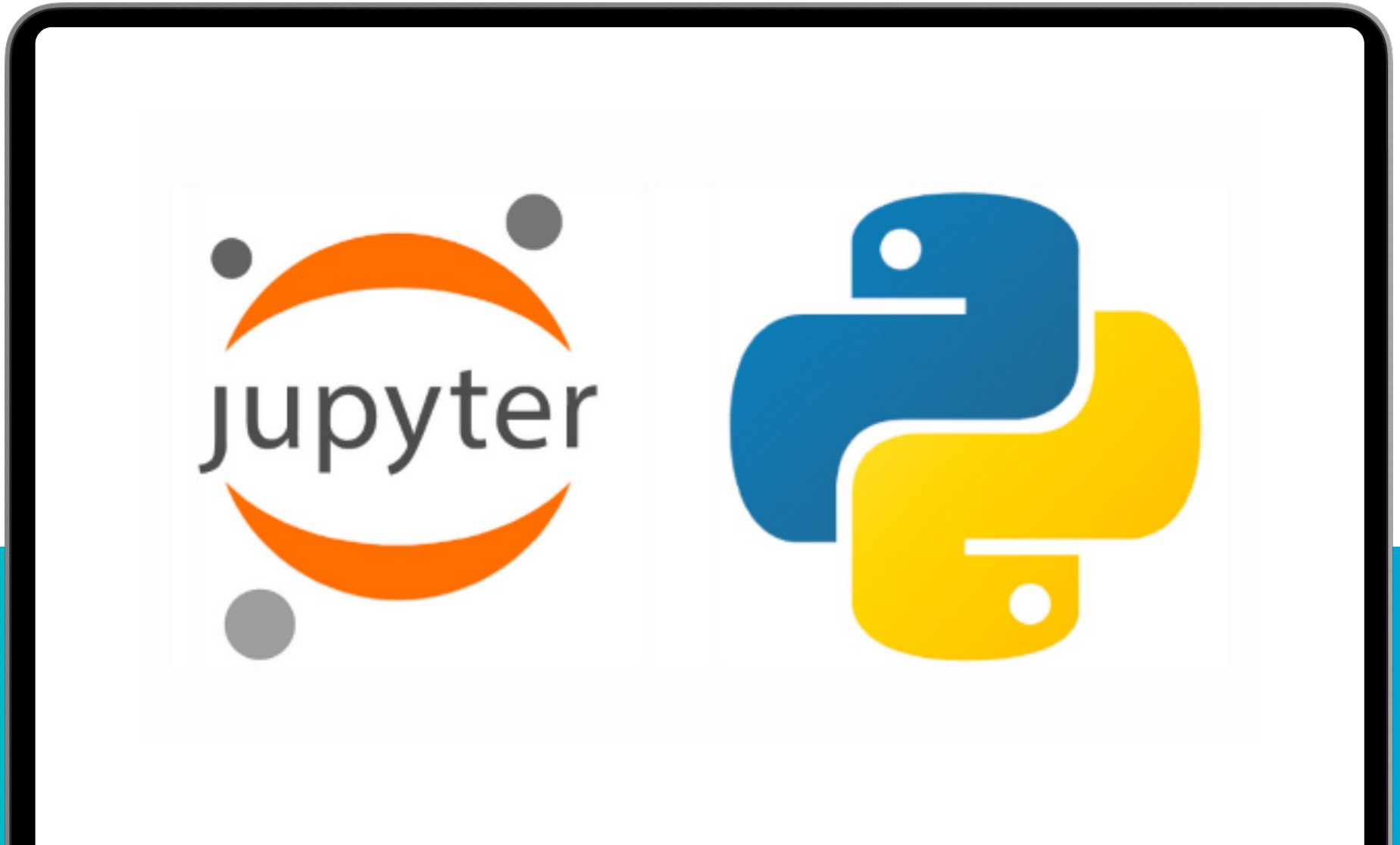
$$I(x,y) = R(x,y)+G(x,y)+B(x,y)$$







Demonstration – What is an Image?



02

What is Image Noise?

Image Noise

- Noise: any random degradation (variation of brightness or color information) on an image caused by external disturbance

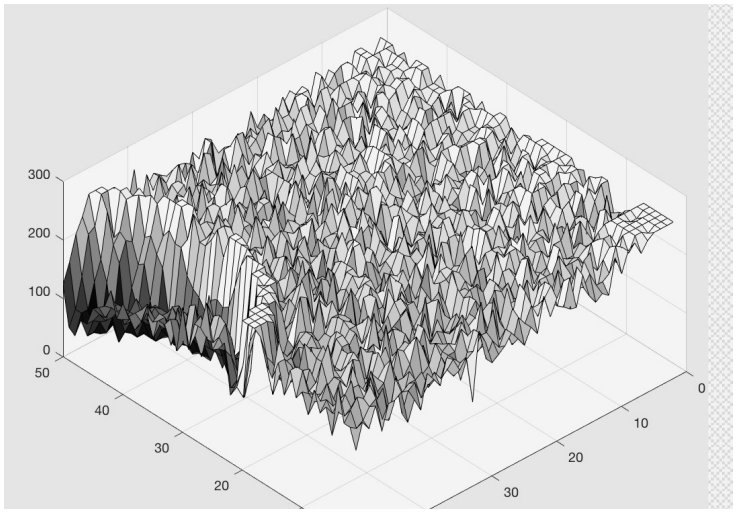
$$I(x,y) = R(x,y)+G(x,y)+B(x,y) + D(x,y)$$

Source of Noise

- **Limits of the device** used to take the picture: (ISO factor) how quickly the camera's sensor absorbs lights
- Errors in the **transmission** (e.g., if an image is sent from a satellite to a phone)
- Memory cell **failures**
- **Sensor heat** while clicking an image

Type of Noise

Salt and Pepper



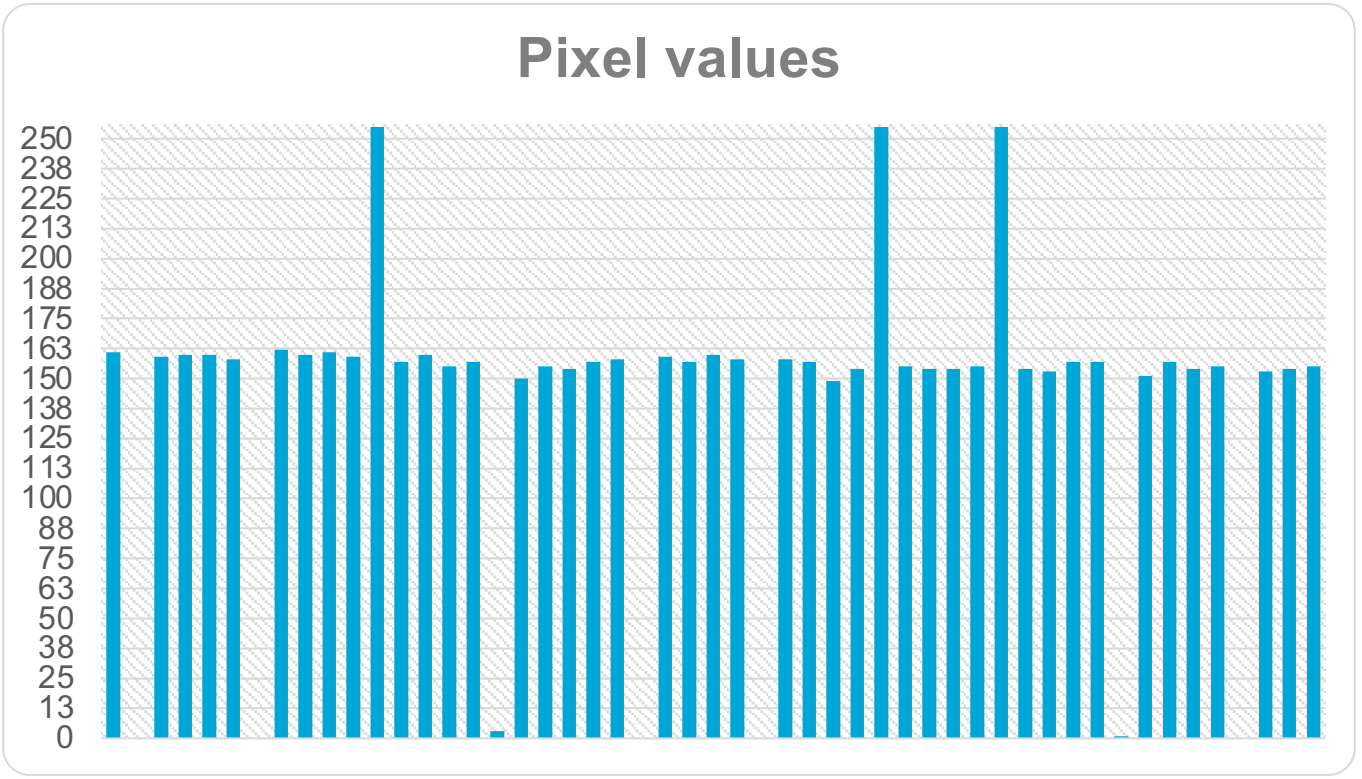
Uniform Noise

Blurry Images

03

What is the Intuition Behind Image Filtering?

Salt and Pepper Noise



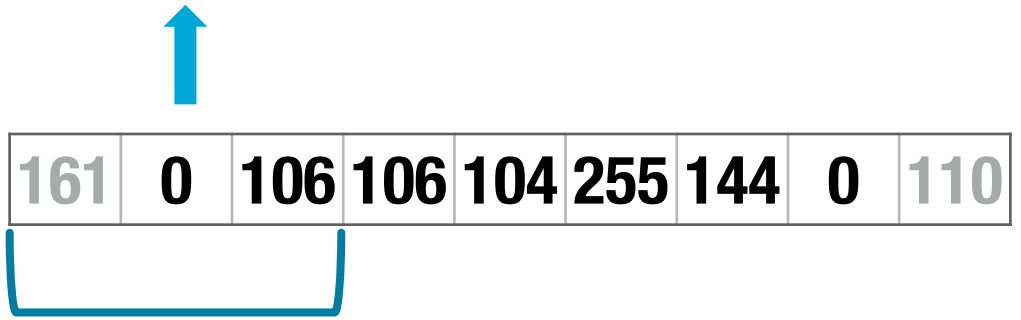
Median Filter

161	0	106	106	104	255	144	0	110

Kernel

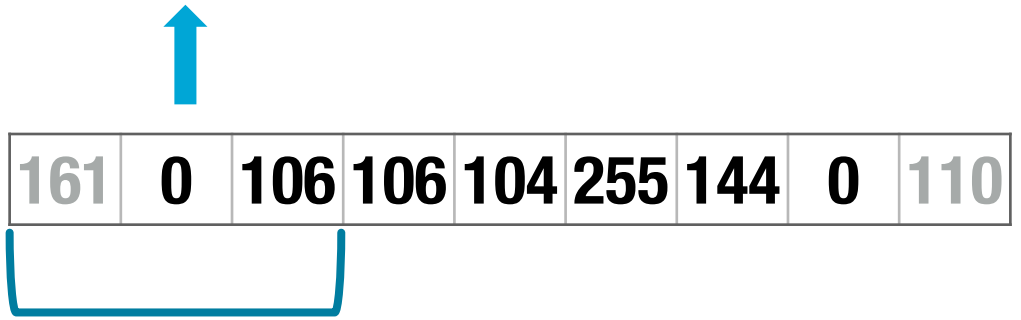
Median Filter

Median(0,106,161) = 106



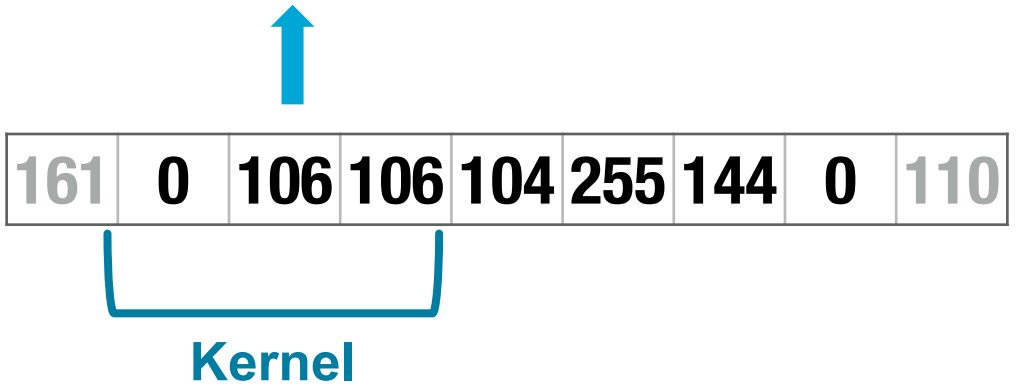
Median Filter

Median(0,106,161) = 106



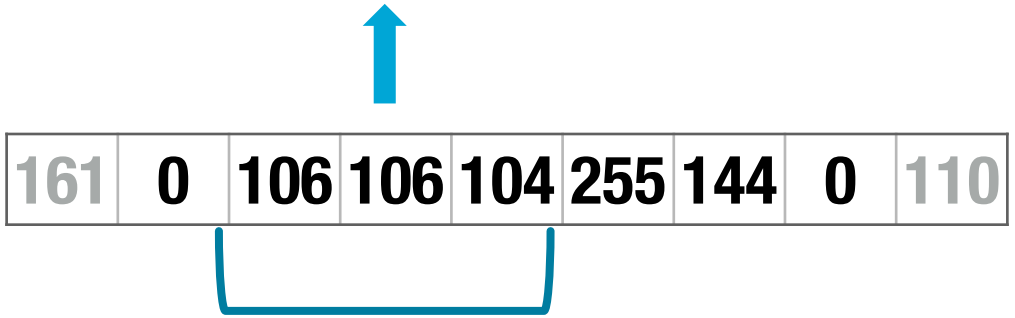
Median Filter

Median(0,106,106) = 106



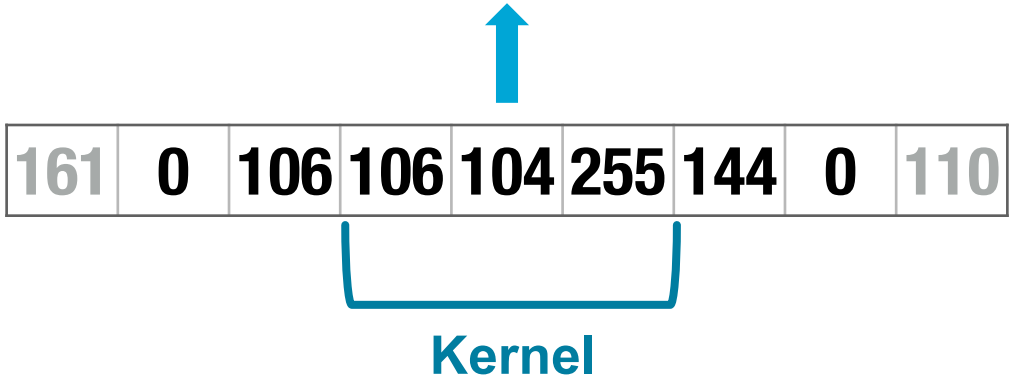
Median Filter

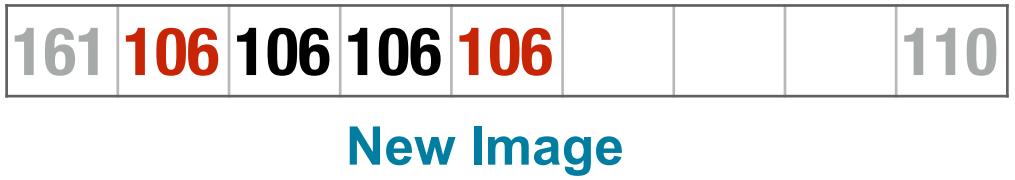
Median(104,106,106) = 106



Median Filter

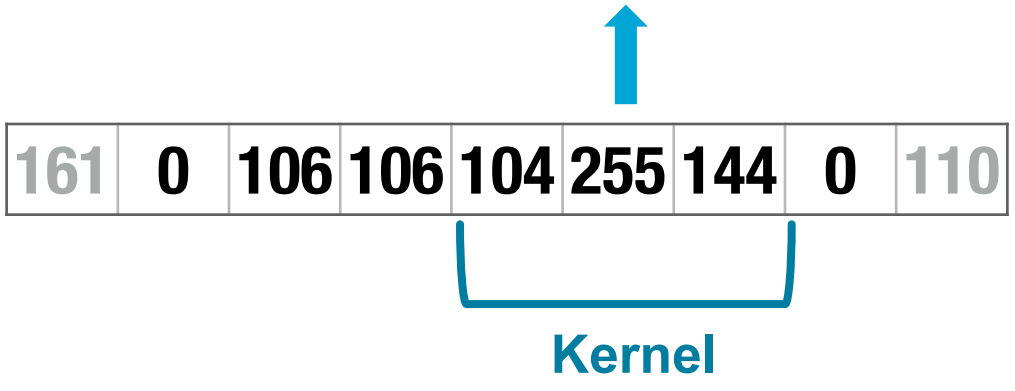
Median(104,106,255) = 106





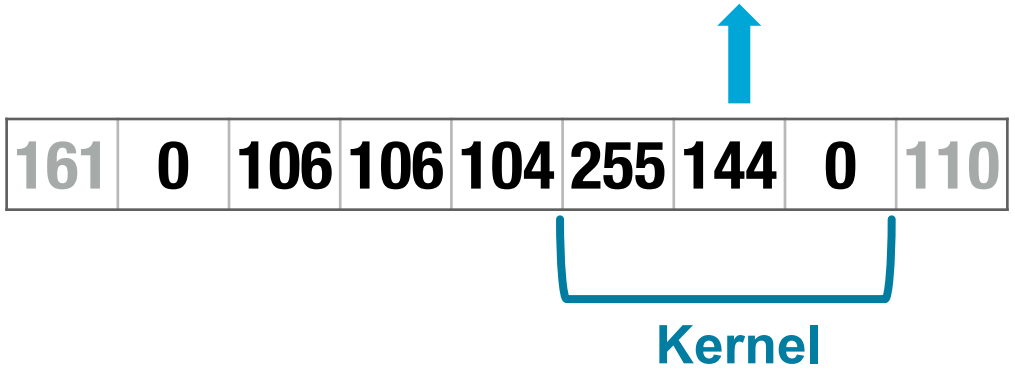
Median Filter

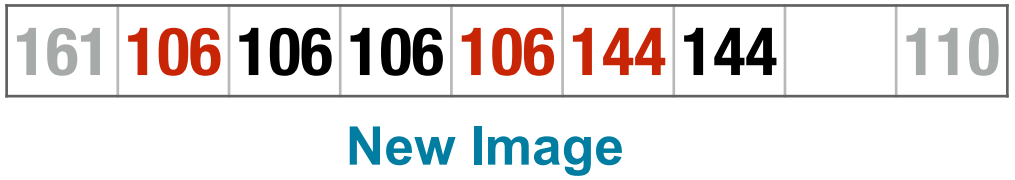
Median(104,144,255) = 144



Median Filter

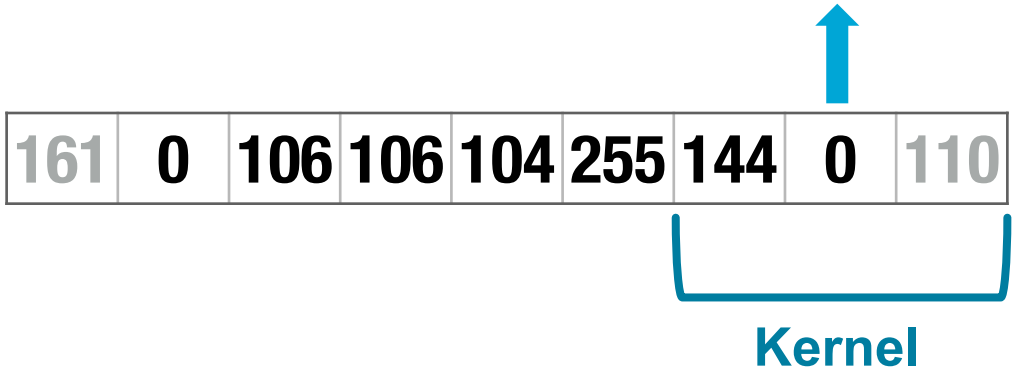
Median(0,144,255) = 144

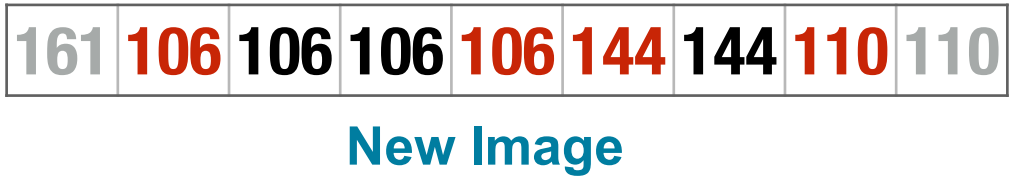




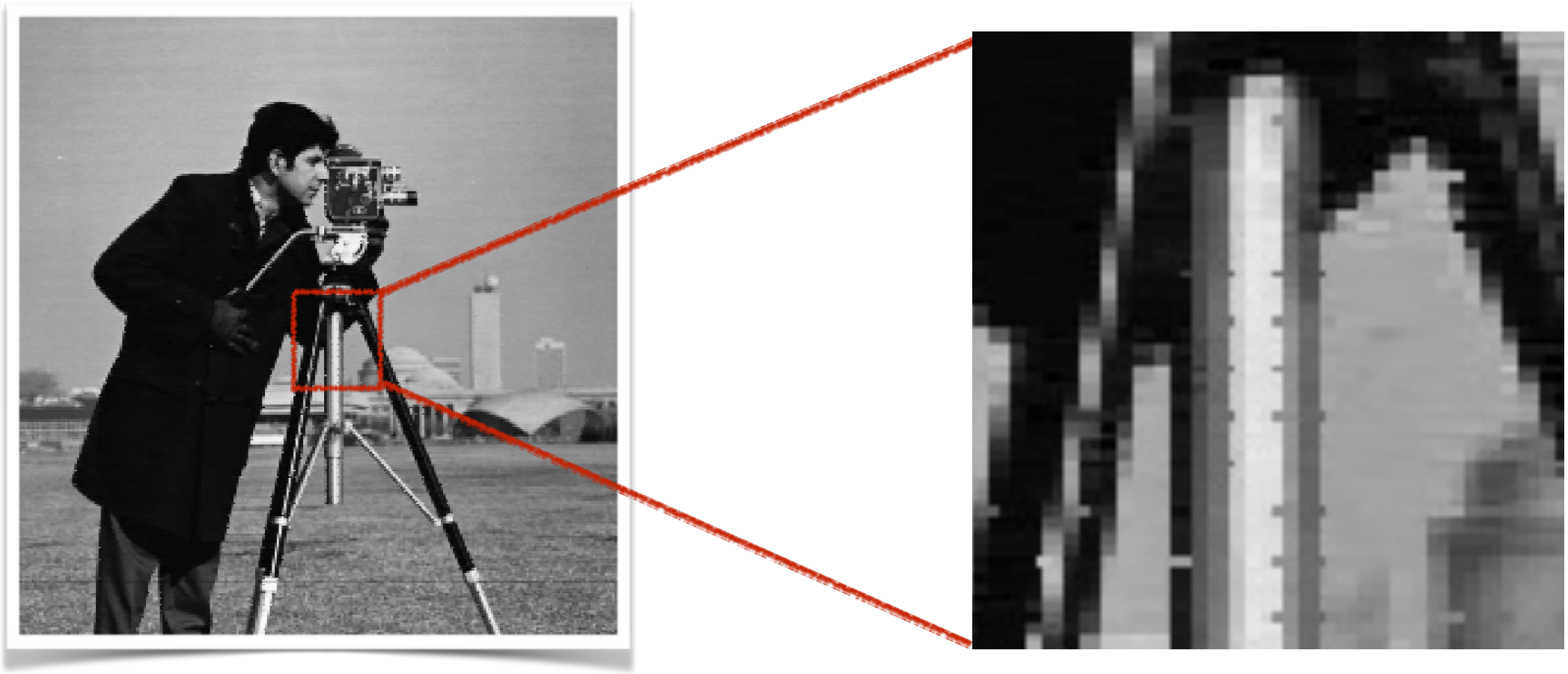
Median Filter

Median(0,110,144) = 110





Median Filter



2D Median Filter

Kernel

108	107	100	103	108	108	110	111	109
161	0	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

Median(0,100,106,106,…
…106,107,108,110,161)

2D Median Filter

Kernel

108	107	100	103	108	108	110	111	109
161	106	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

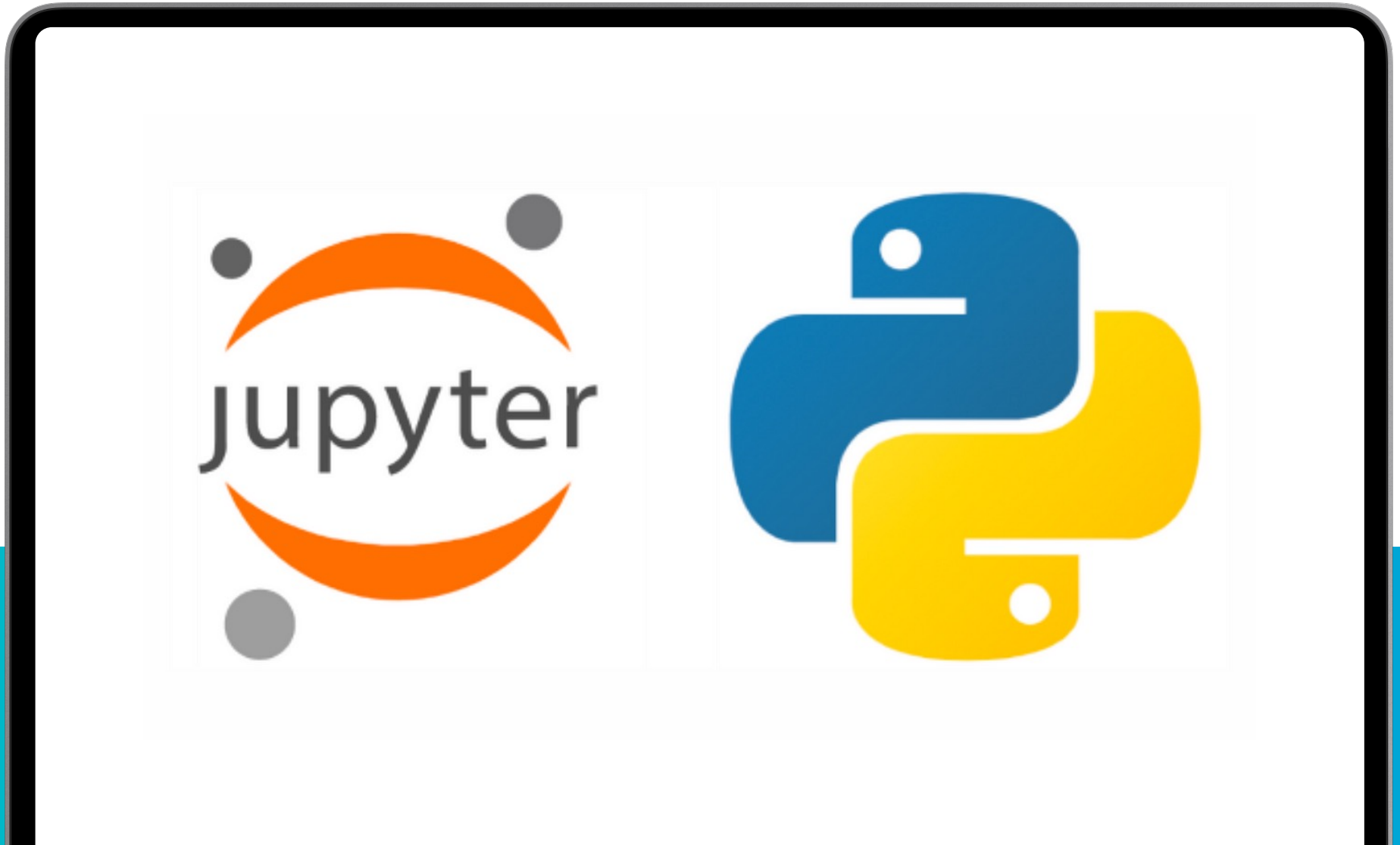
Median(0,100,106,106,…
…106,107,108,110,161)

Kernel =

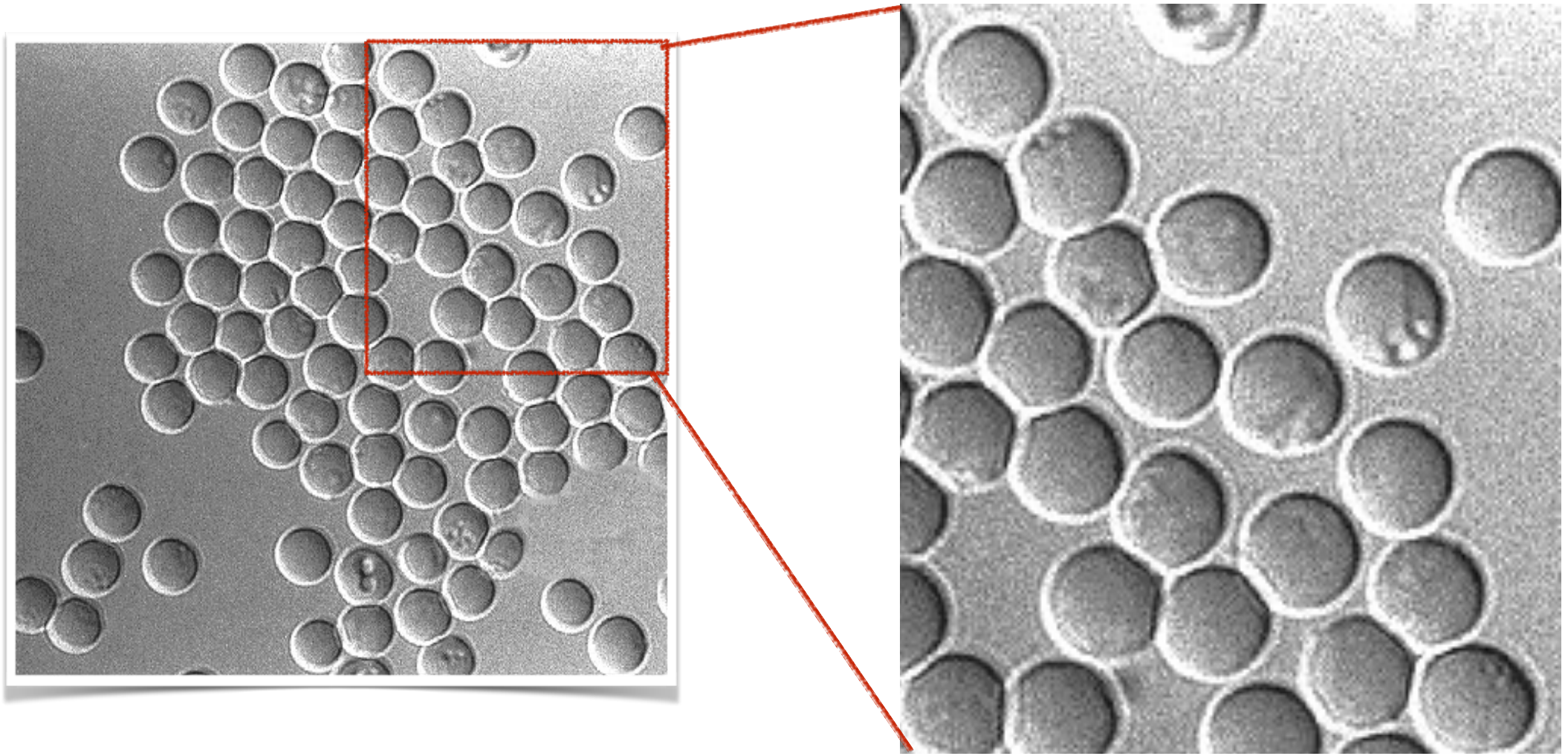
P(i-1,j-1)	*P(i-1, j)*	*P(i-1, j+1)*
P(i,j-1)	*P(i,j)*	*P(i, j+1)*
P(i+1,j-1)	*P(i+1, j)*	*P(i+1, j+1)*

$$P'_{(i,j)} = med\{P_{(i-1,j-1)} \dots P_{(i+1,j+1)}\}$$

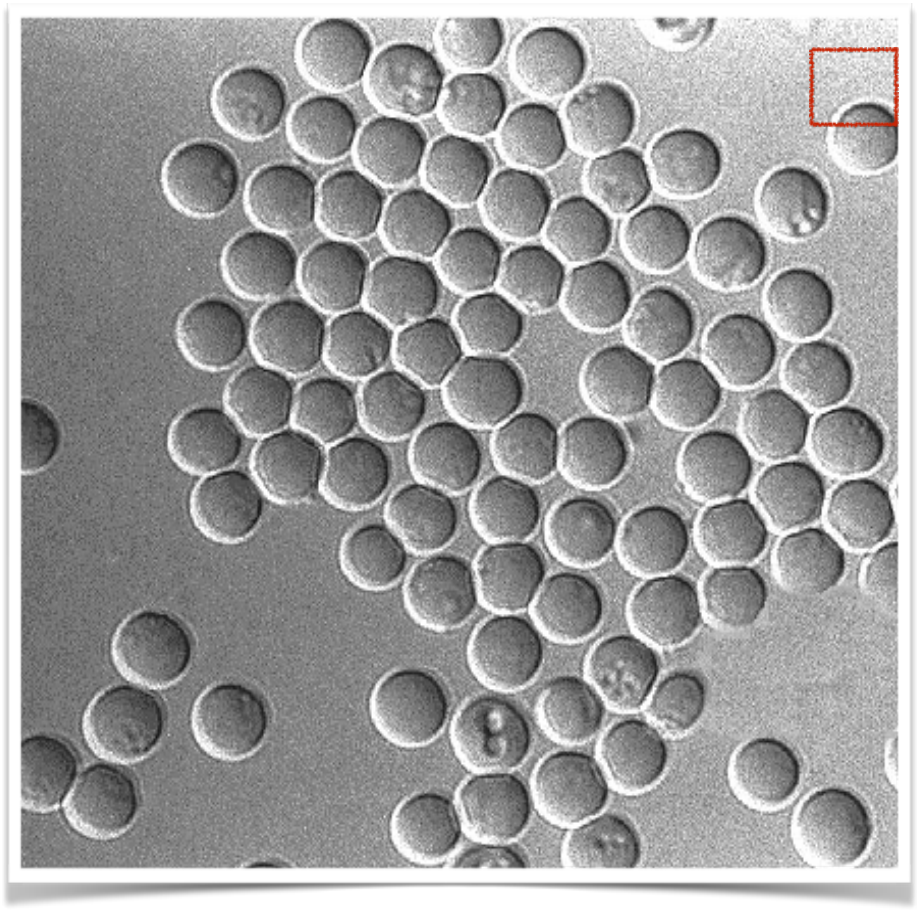
Demonstration – Median Filter

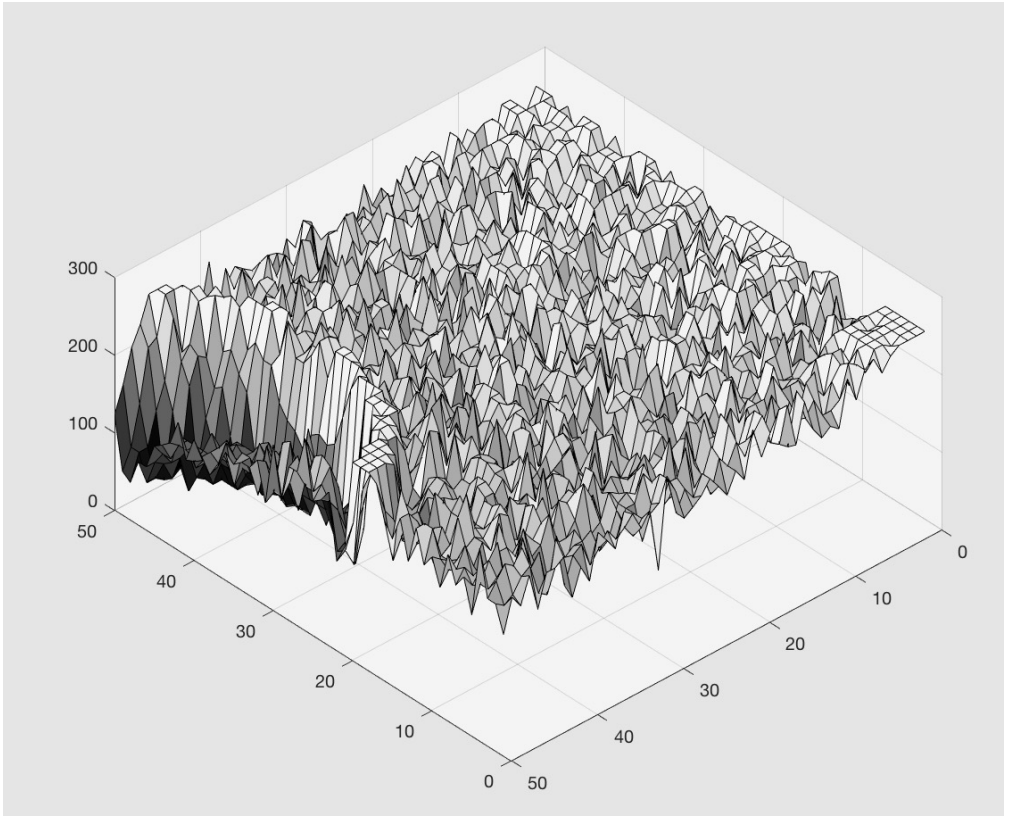


Uniform Noise



Uniform Noise





Arithmetic Mean Filter

Kernel

108	107	100	103	108	108	110	111	109
161	106	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

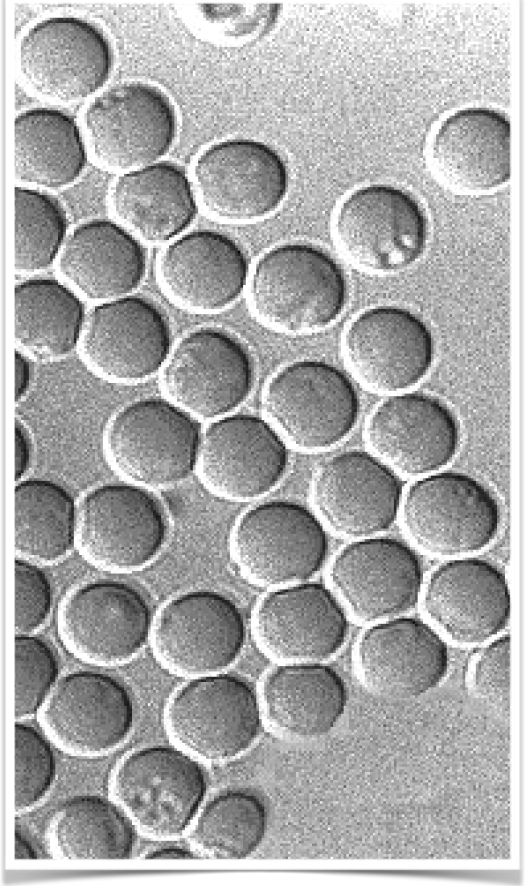
Kernel =

P(i-1,j-1)	*P(i-1, j)*	*P(i-1, j+1)*
P(i,j-1)	*P(i,j)*	*P(i, j+1)*
P(i+1,j-1)	*P(i+1, j)*	*P(i+1, j+1)*

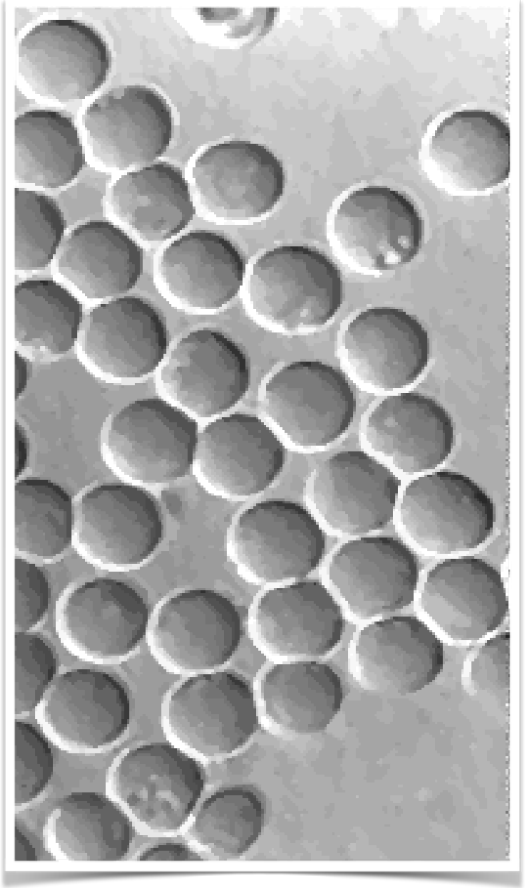
$$P'_{(i,j)} = \frac{1}{9} * \sum_{R=i-1}^{i+1} \left(\sum_{C=j-1}^{j+1} P_{(R,C)} \right)$$

Arithmetic Mean Filter

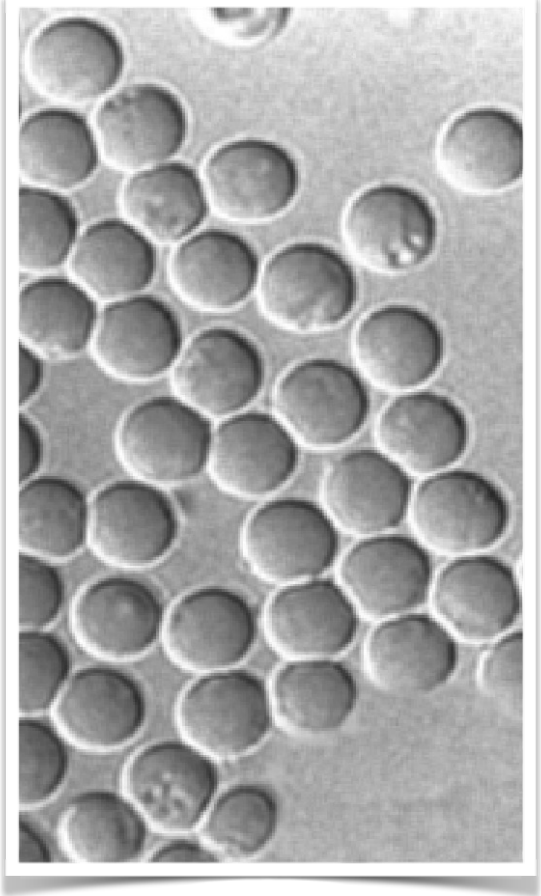
Uniform Noise



3x3 Median

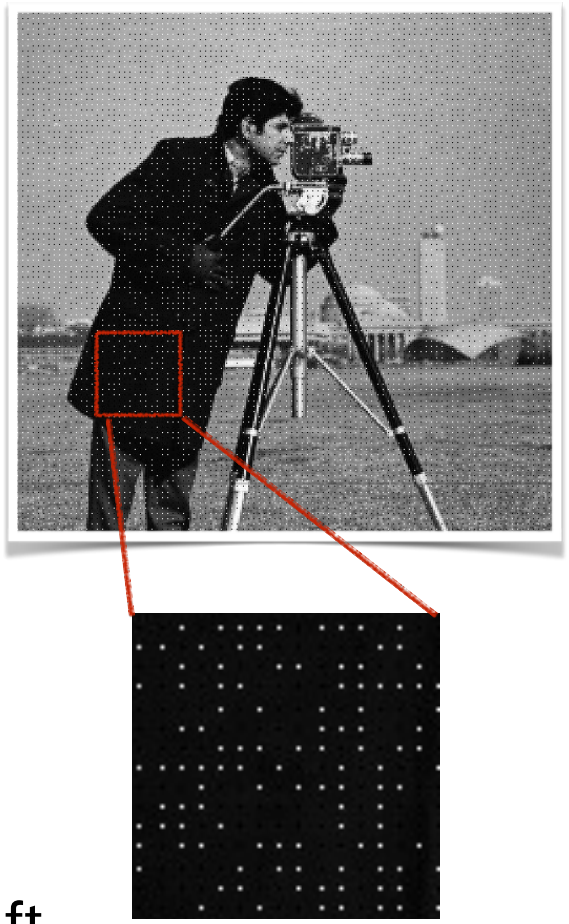


3x3 Mean

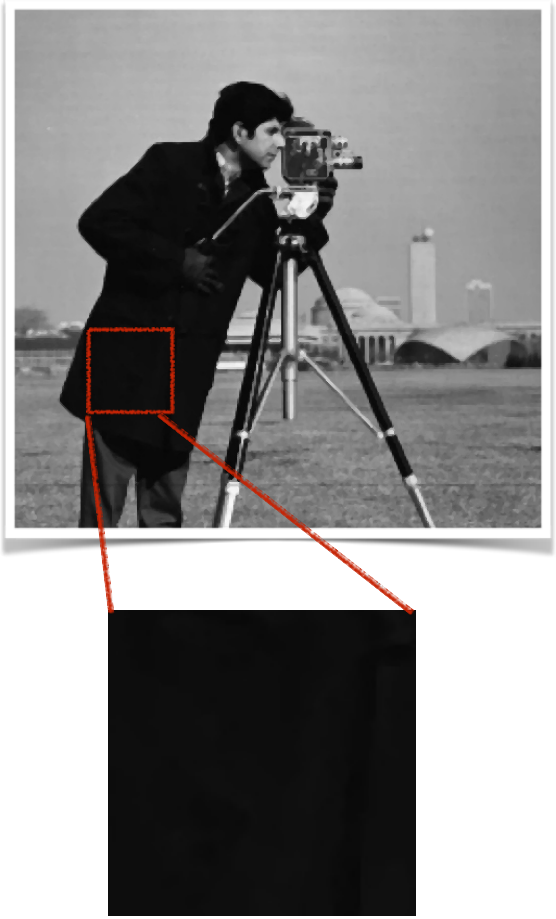


Arithmetic Mean Filter

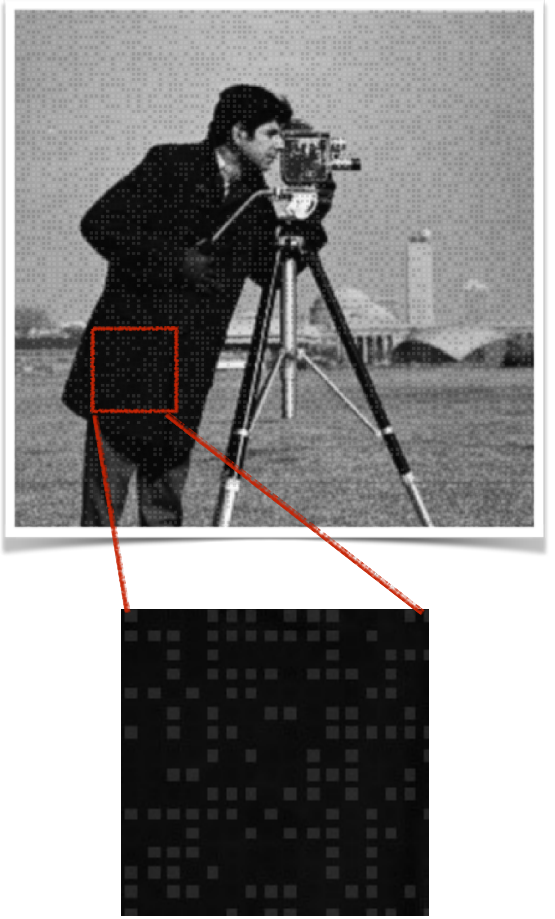
Salt and Pepper Noise



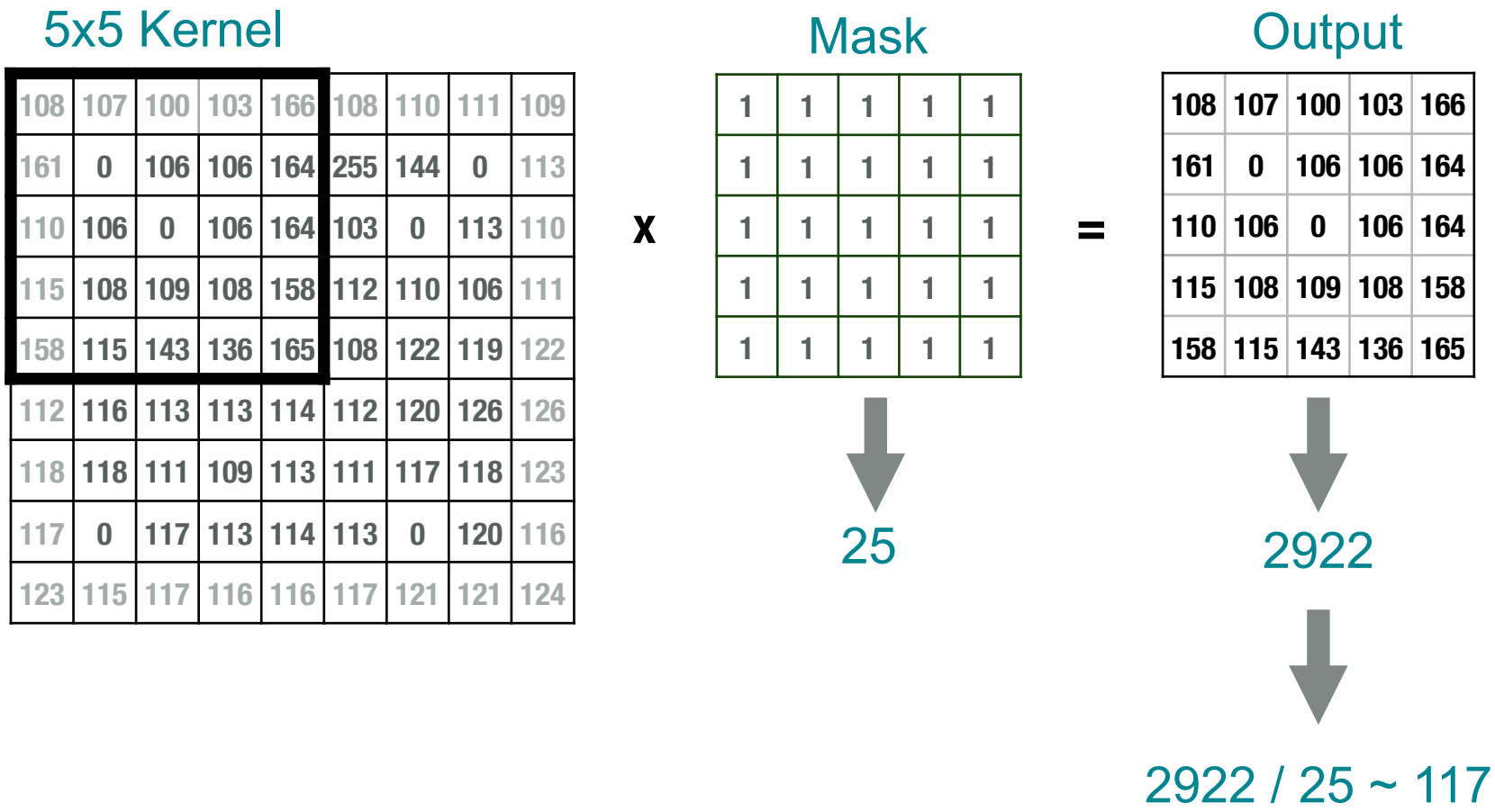
3x3 Median



3x3 Mean



Weighted Mean (Linear) Filter



Weighted Mean (Linear) Filter

5x5 Kernel

108	107	100	103	166	108	110	111	109
161	0	106	106	164	255	144	0	113
110	106	0	106	164	103	0	113	110
115	108	109	108	158	112	110	106	111
158	115	143	136	165	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

Normalization Weights

$$P'_{(i,j)} = \frac{1}{\sum_{R=i-2}^{i+2}\sum_{R=j-2}^{j+2} W_{(i,j)}} * \left(\sum_{R=i-2}^{i+2}\sum_{R=j-2}^{j+2} W_{(i,j)} * P_{(i,j)} \right)$$

$$P'_{(i,j)} = \frac{1}{25} * \left(\sum_{R=i-2}^{i+2}\sum_{R=j-2}^{j+2} 1 * P_{(i,j)} \right)$$

5x5 Mean Filter when all W(i,j) = 1

Weighted Mean (Linear) Filter

5x5 Kernel

	j-2	j-1	j	j+1	j+2
i-2	108	107	100	103	166
i-1	161	0	106	106	164
i	110	106	0	106	164
i+1	115	108	109	108	158
i+2	158	115	143	136	165

X

Mask

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Weighted Mean (Linear) Filter

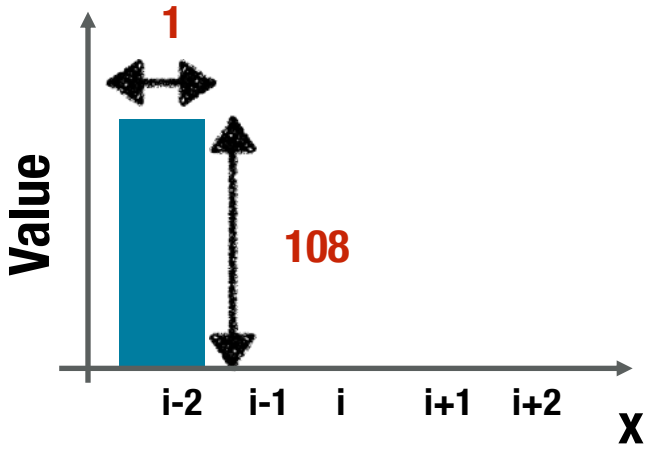
5x5 Kernel

	j-2	j-1	j	j+1	j+2
i-2	108	107	100	103	166
i-1	161	0	106	106	164
i	110	106	0	106	164
i+1	115	108	109	108	158
i+2	158	115	143	136	165

x

Mask

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1



Weighted Mean (Linear) Filter

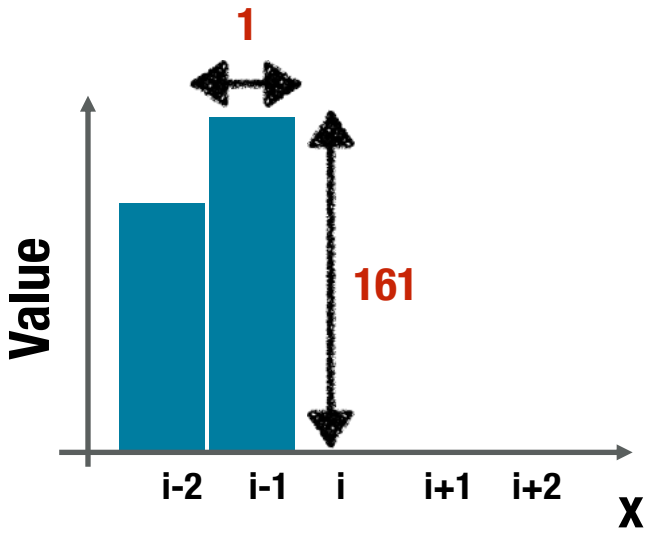
5x5 Kernel

	j-2	j-1	j	j+1	j+2
i-2	108	107	100	103	166
i-1	161	0	106	106	164
i	110	106	0	106	164
i+1	115	108	109	108	158
i+2	158	115	143	136	165

x

Mask

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1



Weighted Mean (Linear) Filter

5x5 Kernel

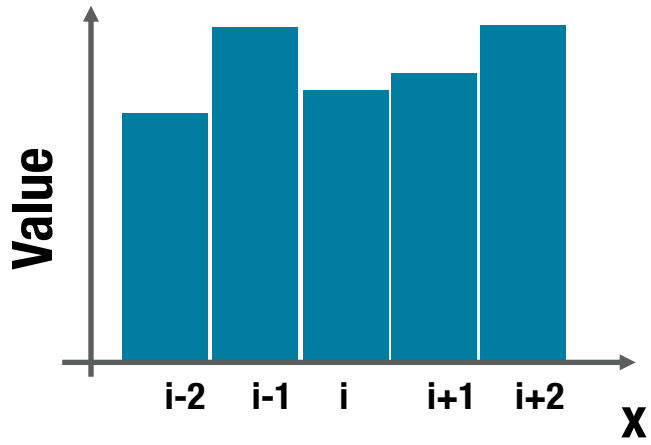
	j-2	j-1	j	j+1	j+2
i-2	108	107	100	103	166
i-1	161	0	106	106	164
i	110	106	0	106	164
i+1	115	108	109	108	158
i+2	158	115	143	136	165

x

Mask

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Area Under Curve



Weighted Mean (Linear) Filter

5x5 Kernel

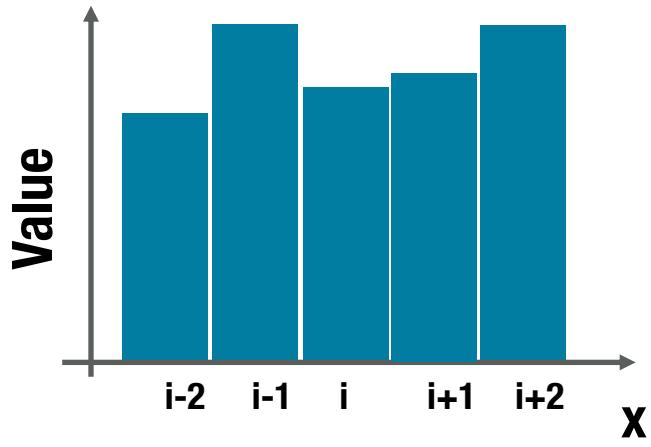
	j-2	j-1	j	j+1	j+2
i-2	108	107	100	103	166
i-1	161	0	106	106	164
i	110	106	0	106	164
i+1	115	108	109	108	158
i+2	158	115	143	136	165

x

Mask

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Area Under Curve



$$\int_{i-2}^{i+2} Kernel(x,y)\ dx$$

Weighted Mean (Linear) Filter

5x5 Kernel

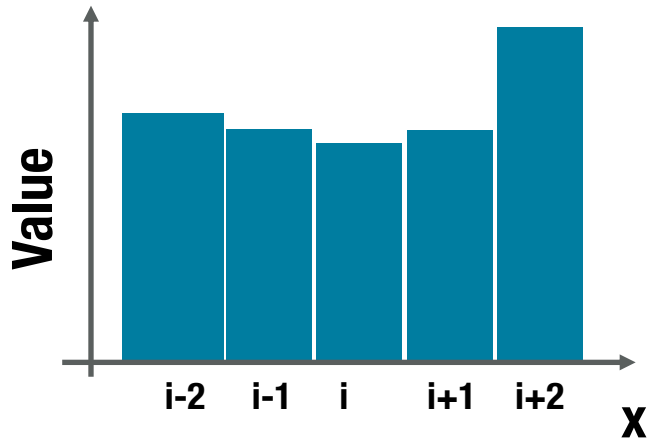
	j-2	j-1	j	j+1	j+2
i-2	108	107	100	103	166
i-1	161	0	106	106	164
i	110	106	0	106	164
i+1	115	108	109	108	158
i+2	158	115	143	136	165

x

Mask

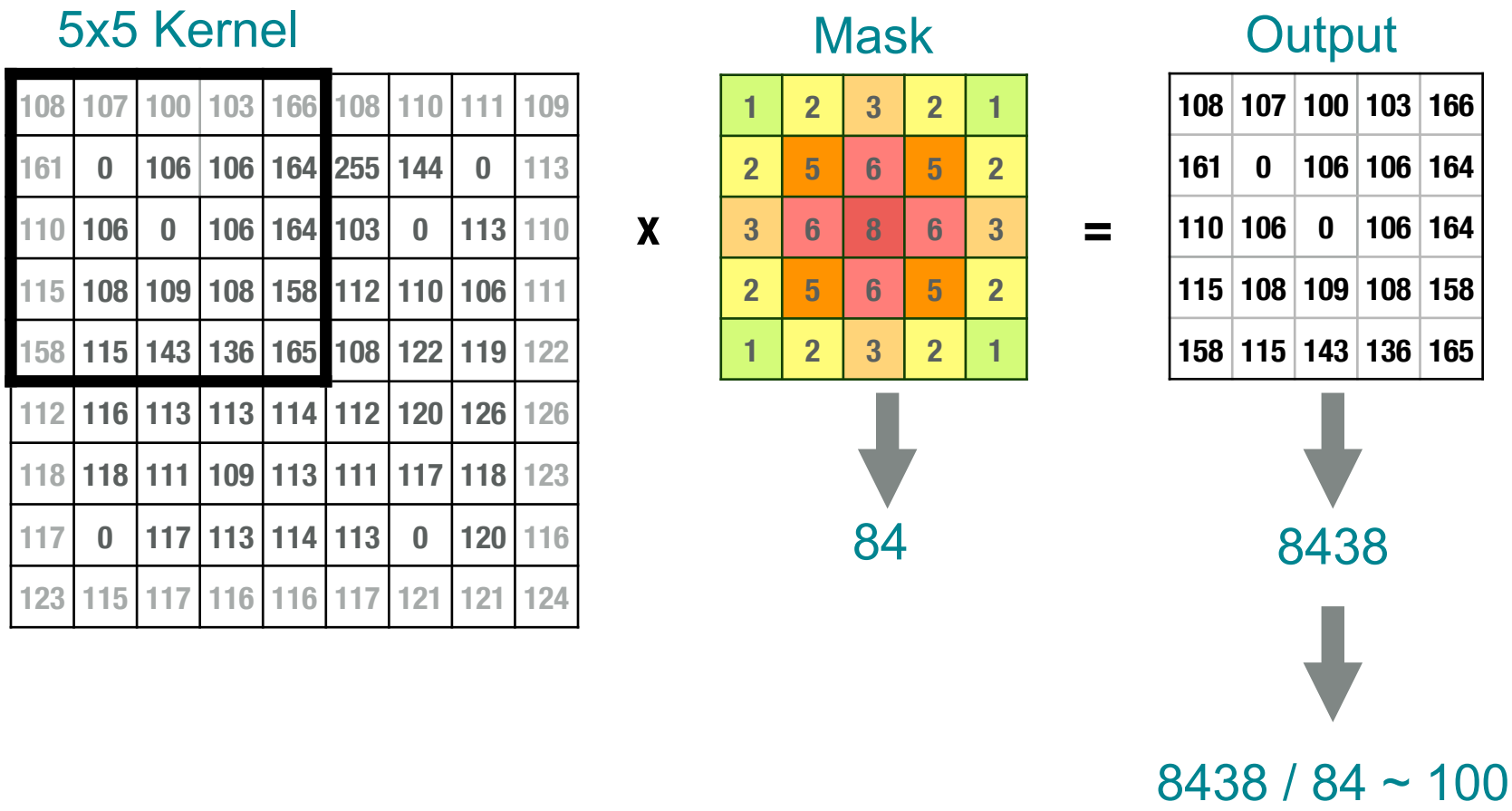
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Area Under Curve



$$\int_{j-2}^{j+2} Kernel(x,y)\ \ dy$$

Weighted Mean (Linear) Filter

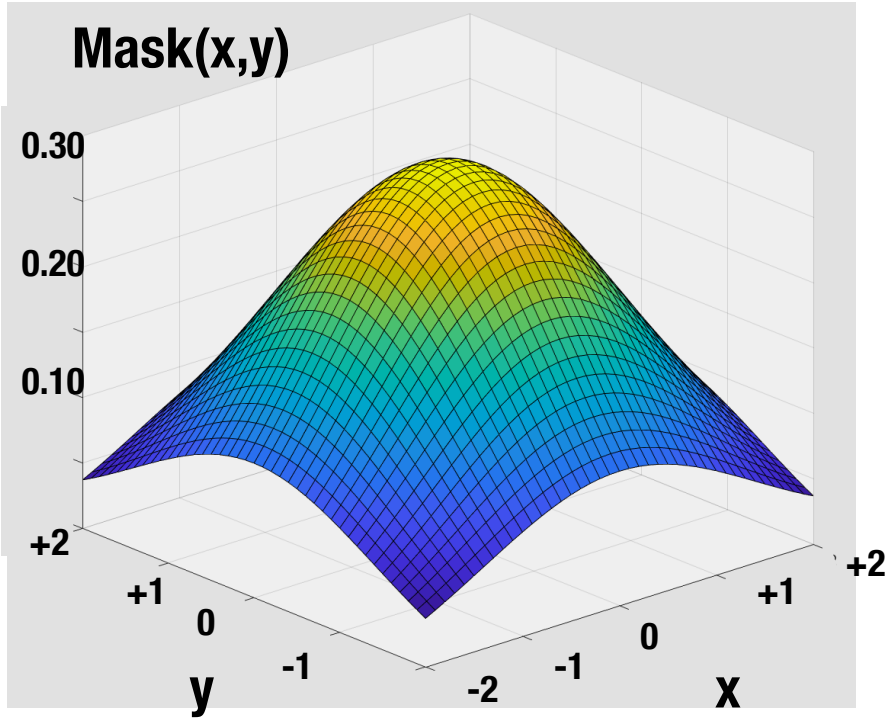


Gaussian Filter

Mask

$\frac{1}{84} *$

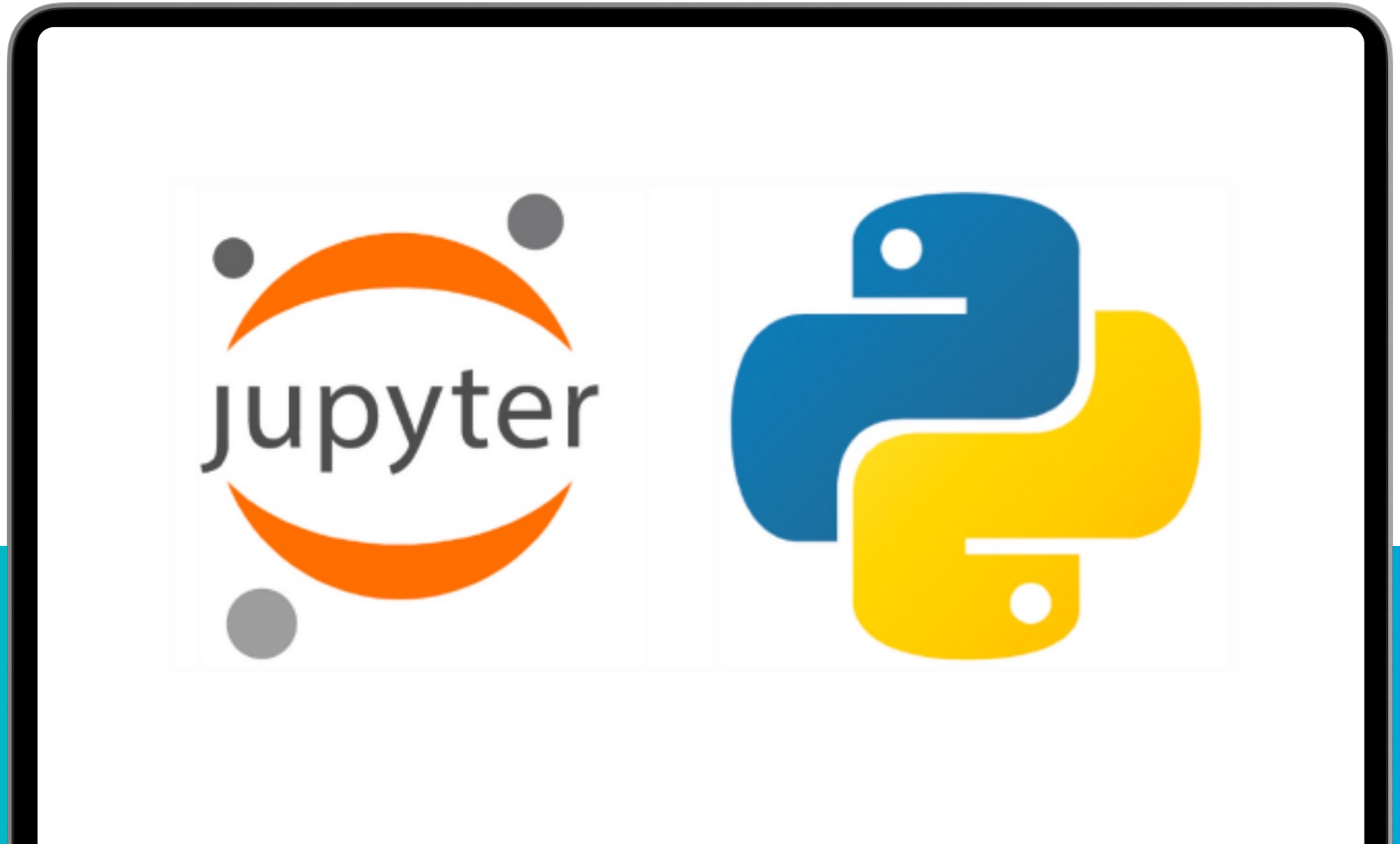
1	2	3	2	1
2	5	6	5	2
3	6	8	6	3
2	5	6	5	2
1	2	3	2	1



2-D Gaussian distribution with mean (0,0) and σ =1.4

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

Demonstration – Linear Filters



04

How Do We Sharpen Images?

From smoothing to sharpening

- We can use positive kernel to smooth images
- Smoothing => averaging the pixel => Integration

- How can we sharpen images using kernels?
- Ideas?

Kernel with Negative Values

Image to filter

108	107	100	103	108	108	110	111	109
161	0	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

Kernel

0	-1	0
-1	4	-1
0	-1	0

Output

	-480	112	5	-157	561	211	-481	
	100	-3	0	-4	-59	-470	236	
	-13	3	-1	-6	20	100	-29	
	10	-4	2	0	-24	31	0	
	6	1	5	8	-5	3	21	
	127	-13	-14	4	-11	119	-14	
	-467	127	-4	1	110	-471	125	

Values after the filter may be negative or larger than 255.

Mask with Negative Values

Image to filter

108	107	100	103	108	108	110	111	109
161	0	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

Kernel

0	-1	0
-1	4	-1
0	-1	0

Output

	0	112	5	0	255	211	0	
	100	0	0	0	0	0	236	
	0	3	0	0	20	100	0	
	10	0	2	0	0	31	0	
	6	1	5	8	0	3	21	
	127	0	0	4	0	119	0	
	0	127	0	1	110	0	125	

Values after the filter may be negative or larger than 255.
Values must be remapped to [0; 255]

Mask with Negative Values

Image to filter

108	107	100	103	108	108	110	111	109
161	0	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

Kernel

0	-1	0
-1	4	-1
0	-1	0

Output

	0	112	5	0	255	211	0	
	100	0	0	0	0	0	236	
	0	3	0	0	20	100	0	
	10	0	2	0	0	31	0	
	6	1	5	8	0	3	21	
	127	0	0	4	0	119	0	
	0	127	0	1	110	0	125	

Local peaks

Remove negative values

Replacing values > 255

Mask with Negative Values

Image to filter

108	107	100	103	108	108	110	111	109
161	0	106	106	104	255	144	0	113
110	106	106	106	104	103	0	113	110
115	108	109	108	107	112	110	106	111
115	115	111	111	110	108	122	119	122
112	116	113	113	114	112	120	126	126
118	118	111	109	113	111	117	118	123
117	0	117	113	114	113	0	120	116
123	115	117	116	116	117	121	121	124

Kernel

0	-1	0
-1	4	-1
0	-1	0

Output

	0	112	5	0	255	211	0	
	100	0	0	0	0	0	236	
	0	3	0	0	20	100	0	
	10	0	2	0	0	31	0	
	6	1	5	8	0	3	21	
	127	0	0	4	0	119	0	
	0	127	0	1	110	0	125	

Effect = Highlighting pixels that are “local” **peaks** in the image

Sharpening Filter

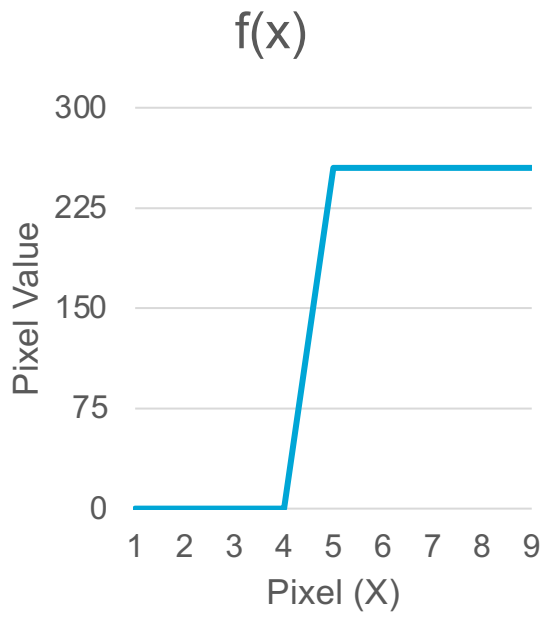
Laplacian Mask

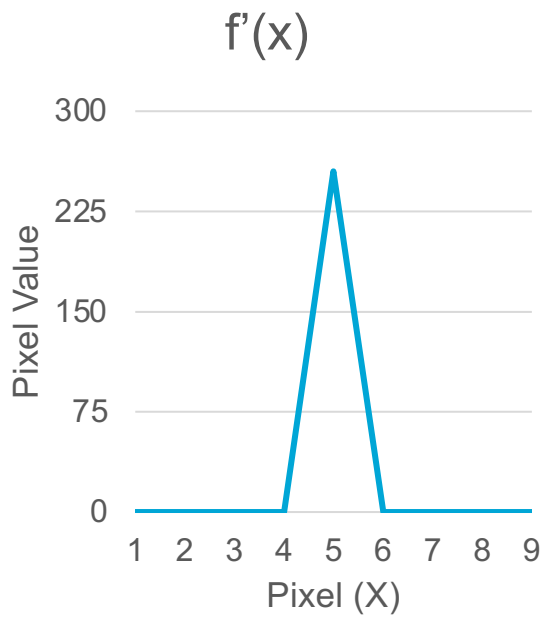
y

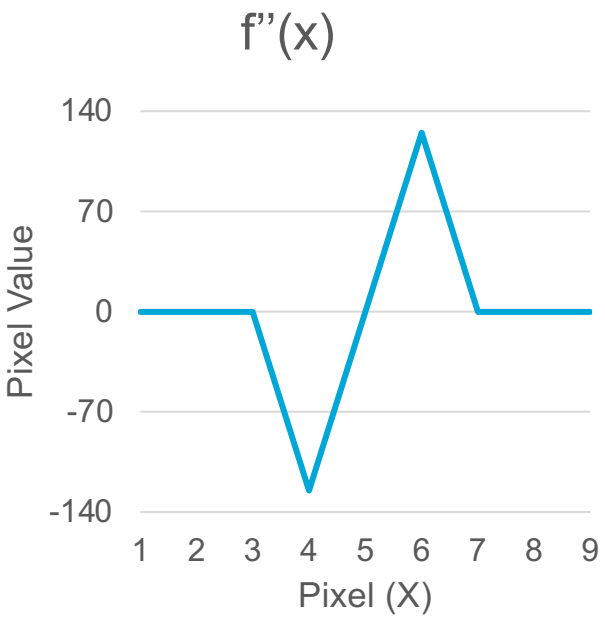
0	-1	0
-1	4	-1
0	-1	0

x

Coefficients of the Laplacian (2nd derivative)







Sharpening Filter

Laplacian Mask

y

0	-1	0
-1	4	-1
0	-1	0

x

Coefficients of the Laplacian (2nd derivative)

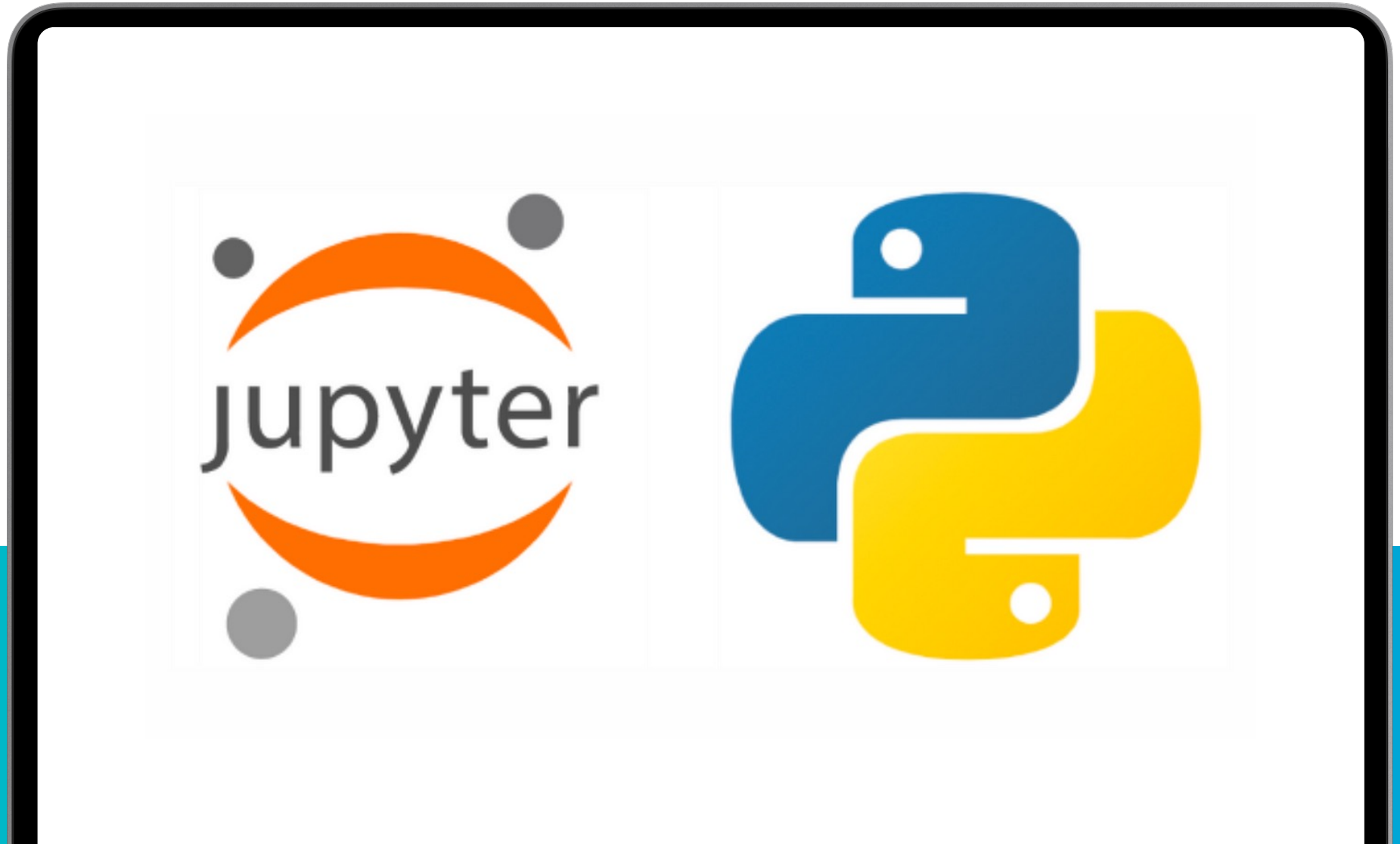
-1	2	-1

$$\frac{\partial^2 f}{\partial^2 y} = -f_{(i+1,j)} + 2f_{(i,j)} - f_{(i-1,j)}$$

-1
2
-1

$$\frac{\partial^2 f}{\partial^2 x} = -f_{(i,j+1)} + 2f_{(i,j)} - f_{(i,j-1)}$$

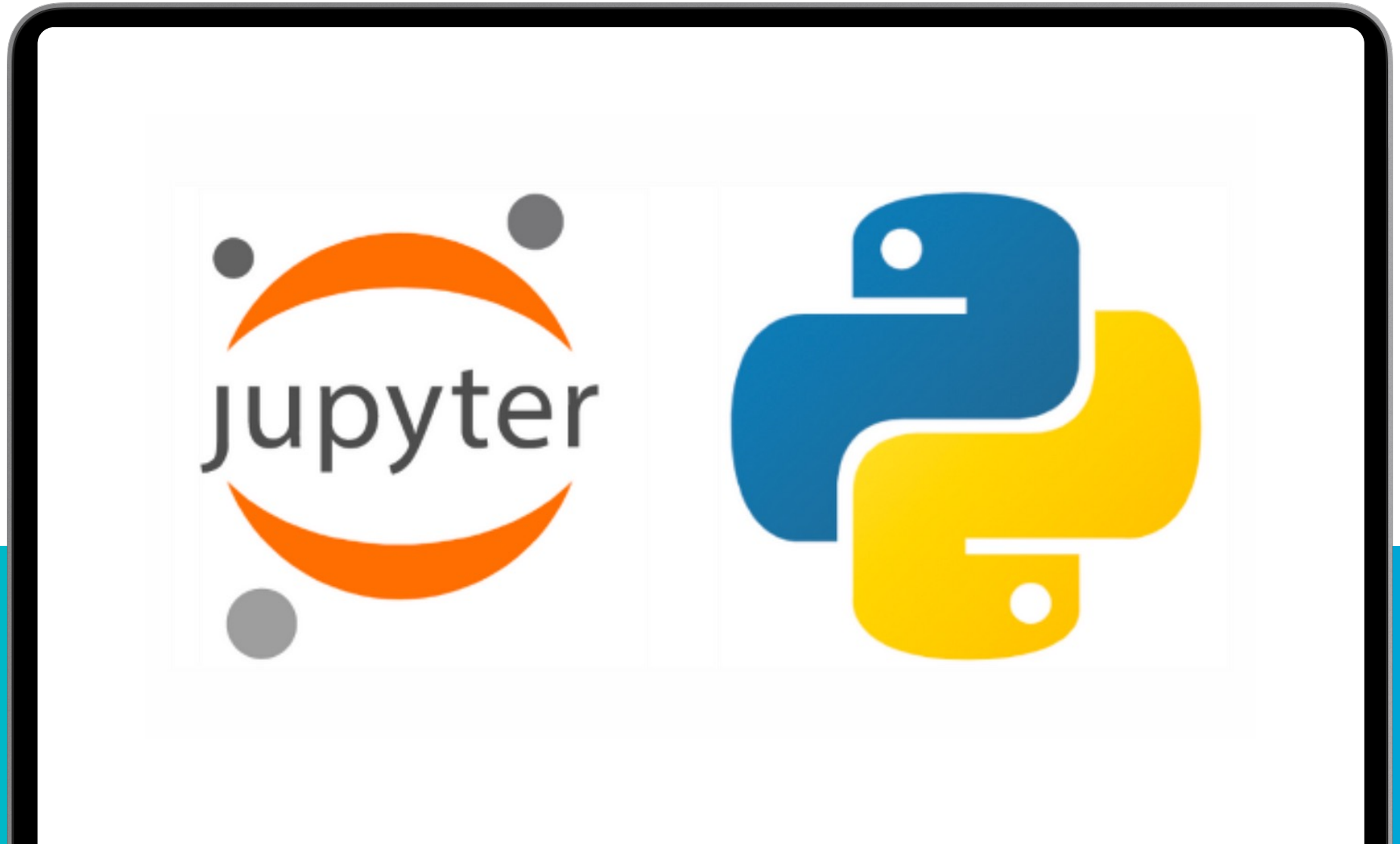
Demonstration – Sharpening Filters



05

What Can We Do With These Techniques?

Demonstration – Advanced Usages



TU Delft

Thank you for your attention!