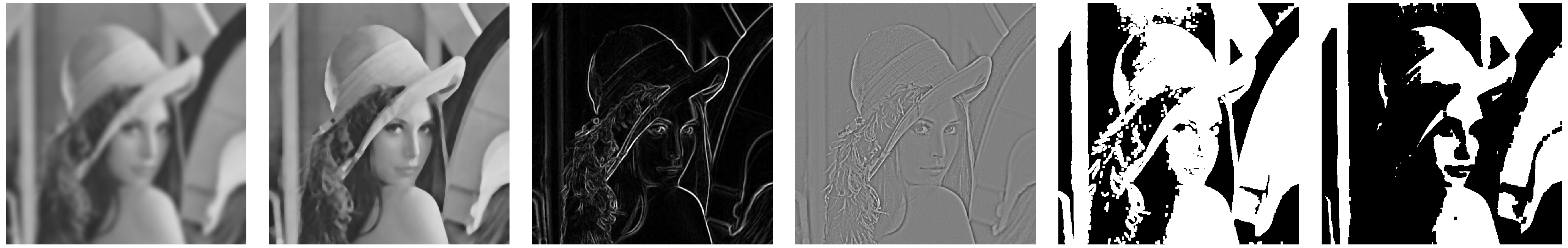
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 5: Image Filtering (final)

(unless otherwise stated slides are taken or adopted from **Bob Woodham, Jim Little** and **Fred Tung**)

Menu for Today (**January 17, 2018**)

Topics:

— Non-linear Filters: Median, ReLU

— Bilateral Filter

Redings:

— **Today's** Lecture: Forsyth & Ponce (2nd ed.) 4.4

— **Next** Lecture: Forsyth & Ponce (2nd ed.) 4.5

Reminders:

— **Assignment 1:** Image Filtering and Hybrid Images due **January 25**th

Today's "**fun**" Example: **Clever** Hans

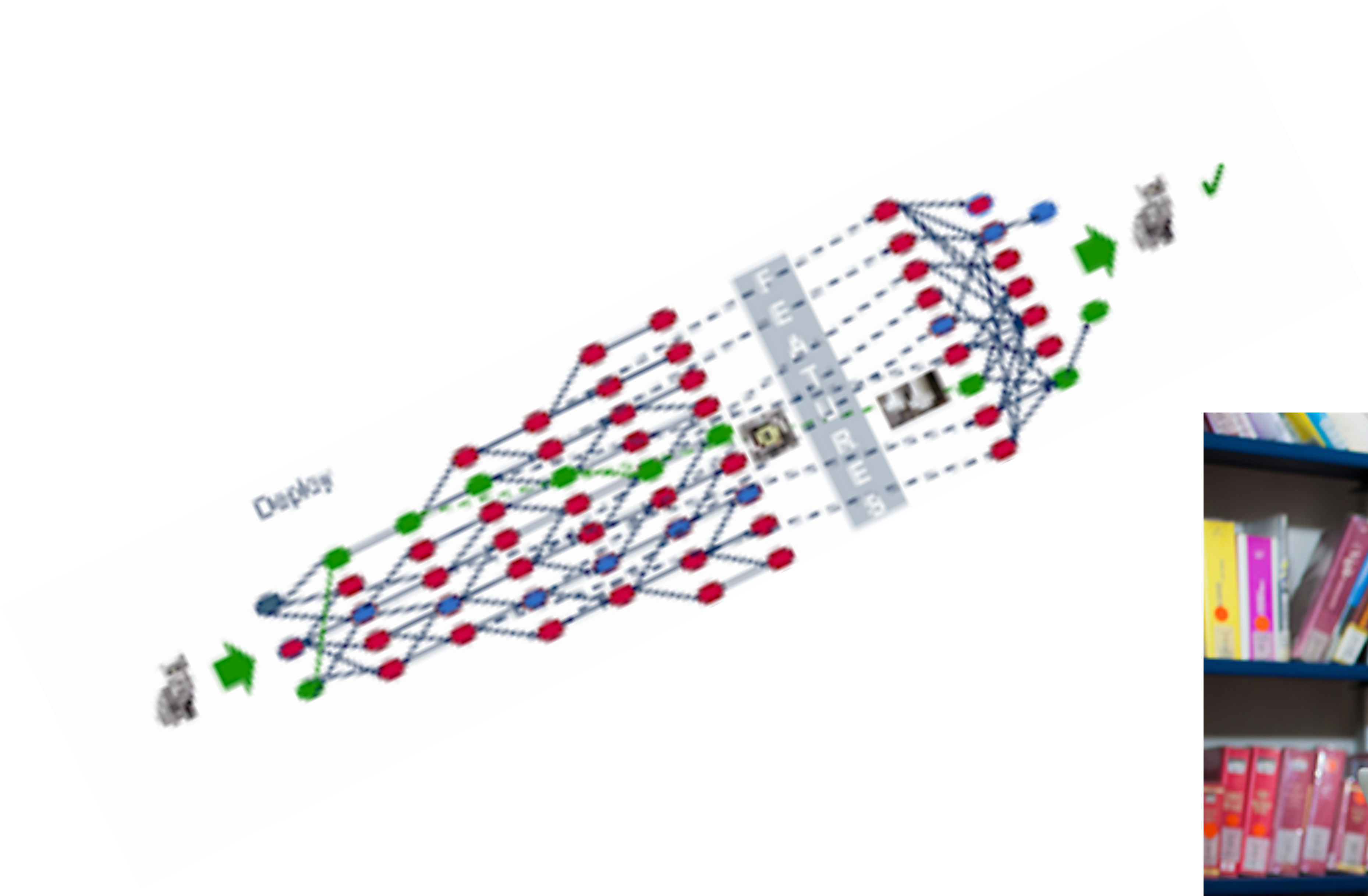
Hans could get 89% of the math questions right

Today’s “**fun**” Example: **Clever** Hans

The course was **smart**, just not in the way van Osten thought!

Hans could get 89% of the math questions right

Clever DNN



Visual **Question Answering**

AI agent

Yes

Is there zebra climbing the tree?

Lecture 4: Re-cap

Linear filtering (one intepretation):
— new pixels are a weighted sum of original pixel values
— “filter” defines weights

Linear filtering (another intepretation):
— each pixel influences the new value for itself and its neighbours
— “filter” specifies the influences

Lecture 4: Re-cap

We covered two additional linear filters: **Gaussian**, **pillbox**

Separability (of a 2D filter) allows for more efficient implementation (as two 1D filters)

The Convolution Theorem: In **Fourier** space, convolution can be reduced to (complex) multiplication

Convolution is **associative** and **symmetric** (correlation is not in general)

Lecture 4: Pre-convolving Separable Gaussian Filter

$$\frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Lecture 4: Pre-convolving Separable Gaussian Filter

$$\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256}\begin{bmatrix} \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ 1 & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \\ \; & \; & \; & \; & \; \end{bmatrix}$$

Lecture 4: Pre-convolving Separable Gaussian Filter

$$\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256}\begin{bmatrix} & & & & \\ & & & & \\ 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{bmatrix}$$

Lecture 4: Pre-convolving Separable Gaussian Filter

$$
\frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 1&4&6&4&1\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline\end{array}
\otimes
\frac{1}{16}\begin{array}{|c|}\hline 1\\\hline 4\\\hline 6\\\hline 4\\\hline 1\\\hline\end{array}
=
\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline &&&&\\\hline &&&&\\\hline 1&4&6&4&1\\\hline 4&16&24&16&4\\\hline 6&24&36&24&6\\\hline 4&16&24&16&4\\\hline 1&4&6&4&1\\\hline &&&&\\\hline &&&&\\\hline\end{array}
$$

Lecture 4: Pre-convolving Separable Gaussian Filter

$$\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Non-linear Filters

We've seen that **linear filters** can perform a variety of image transformations

— shifting

— smoothing

— sharpening

In some applications, better performance can be obtained by using **non-linear filters**.

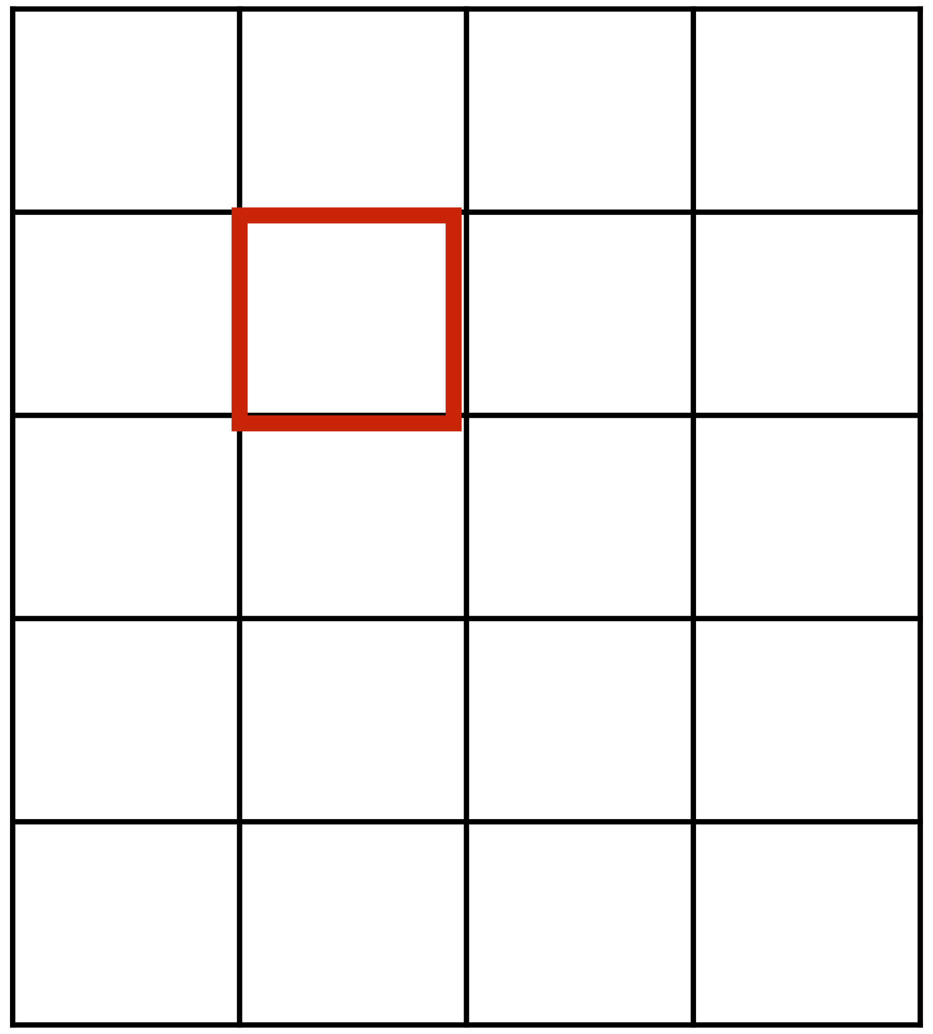
For example, the median filter selects the **median** value from each pixel's neighborhood.

Median Filter

Take the **median value** of the pixels under the filter:

5	13	5	221
4	16	7	34
24	54	34	23
23	75	89	123
54	25	67	12

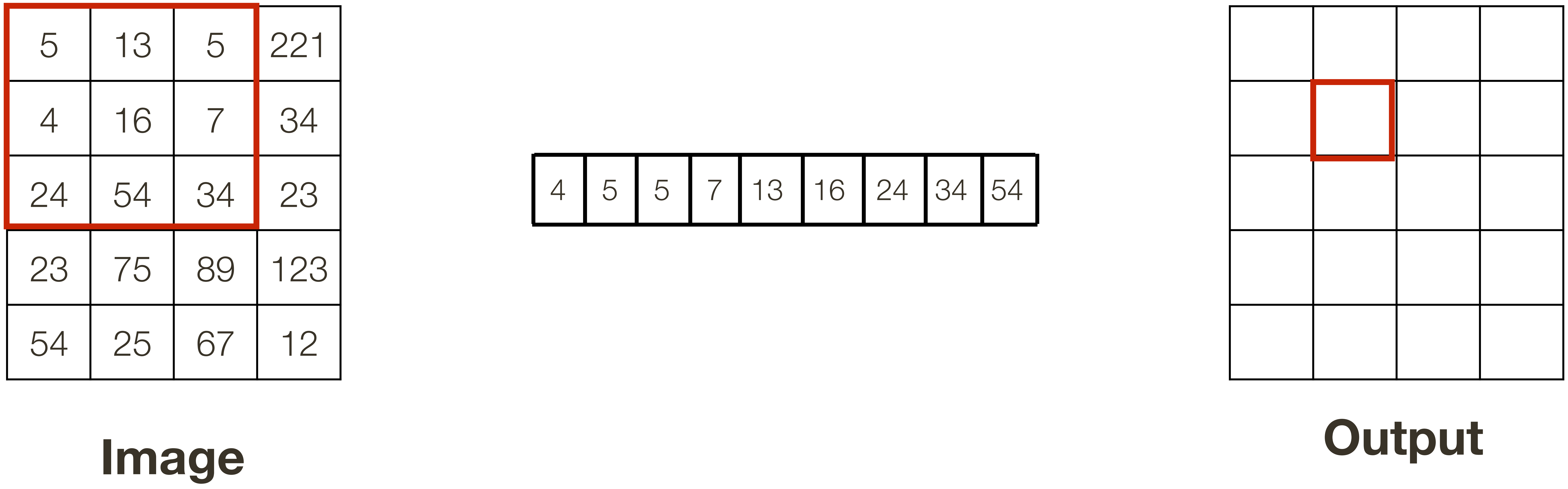
Image



Output

Median Filter

Take the **median value** of the pixels under the filter:



Median Filter

Take the **median value** of the pixels under the filter:

5	13	5	221
4	16	7	34
24	54	34	23
23	75	89	123
54	25	67	12

Image

4	5	5	7	13	16	24	34	54

	13		

Output

Median Filter

Effective at reducing certain kinds of noise, such as impulse noise (a.k.a 'salt and pepper' noise or 'shot' noise)

The median filter forces points with distinct values to be more like their neighbors

Image credit: https://en.wikipedia.org/wiki/Median_filter#/media/File:Medianfilterp.png

Bilateral Filter

An edge-preserving non-linear filter

Like a Gaussian filter:

— The filter weights depend on spatial distance from the center pixel
— Pixels nearby (in space) should have greater influence than pixels far away

Unlike a Gaussian filter:

— The filter weights also depend on range distance from the center pixel
— Pixels with similar brightness value should have greater influence than pixels with dissimilar brightness value

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

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

(with appropriate normalization)

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

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

(with appropriate normalization)

Bilateral filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by a product:

$$\exp^{-\frac{x^2+y^2}{2\sigma_d^2}} \exp^{-\frac{(I(X+x, Y+y) - I(X, Y))^2}{2\sigma_r^2}}$$

(with appropriate normalization)

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

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

(with appropriate normalization)

Bilateral filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by a product:

domain kernel	$\exp^{-\frac{x^2+y^2}{2\sigma_d^2}}$	$\exp^{-\frac{(I(X+x, Y+y)-I(X,Y))^2}{2\sigma_r^2}}$	**range** kernel

(with appropriate normalization)

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel

$$\sigma_d = 0.45$$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel

$$\sigma_d = 0.45$$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel

$$\sigma_r = 0.45$$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(this is different for each locations in the image)

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel

$$\sigma_d = 0.45$$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel

$$\sigma_r = 0.45$$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(this is different for each locations in the image)

multiply →

Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel

$\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel

$\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(this is different for each locations in the image)

multiply →

Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

sum to 1 →

0.11	0.16	0.03
0.16	0.26	0.01
0.11	0.16	0.01

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel

$\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel

$\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(this is different for each locations in the image)

multiply →

Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

$$\sum \begin{bmatrix} 0.11 & 0.16 & 0.03 \\ 0.16 & 0.26 & 0.01 \\ 0.11 & 0.16 & 0.01 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.9 \\ 0.1 & 0.1 & 1 \\ 0 & 0.1 & 1 \end{bmatrix} = 0.1$$

Bilateral Filter

Bilateral Filter

image $I(X,Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

→

image $I(X,Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel

$\sigma_d = 0.45$

$$\sum \begin{bmatrix} 0.08 & 0.12 & 0.08 \\ 0.12 & 0.20 & 0.12 \\ 0.08 & 0.12 & 0.08 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.9 \\ 0.1 & 0.1 & 1 \\ 0 & 0.1 & 1 \end{bmatrix} = 0.3$$

Gaussian Filter (only)

Range Kernel

$\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(this is different for each locations in the image)

multiply →

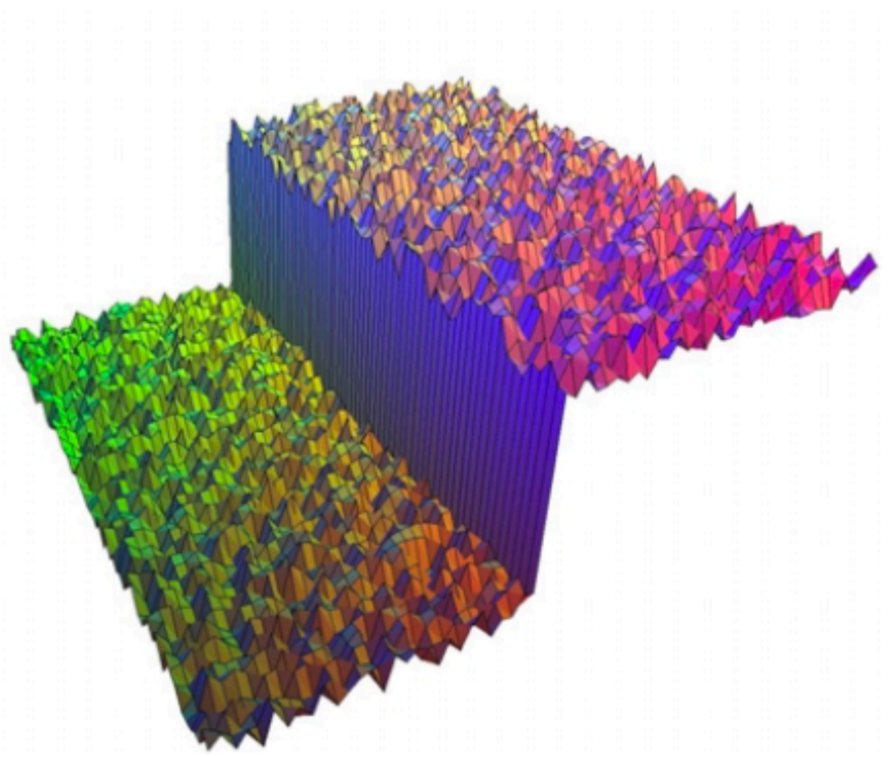
Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

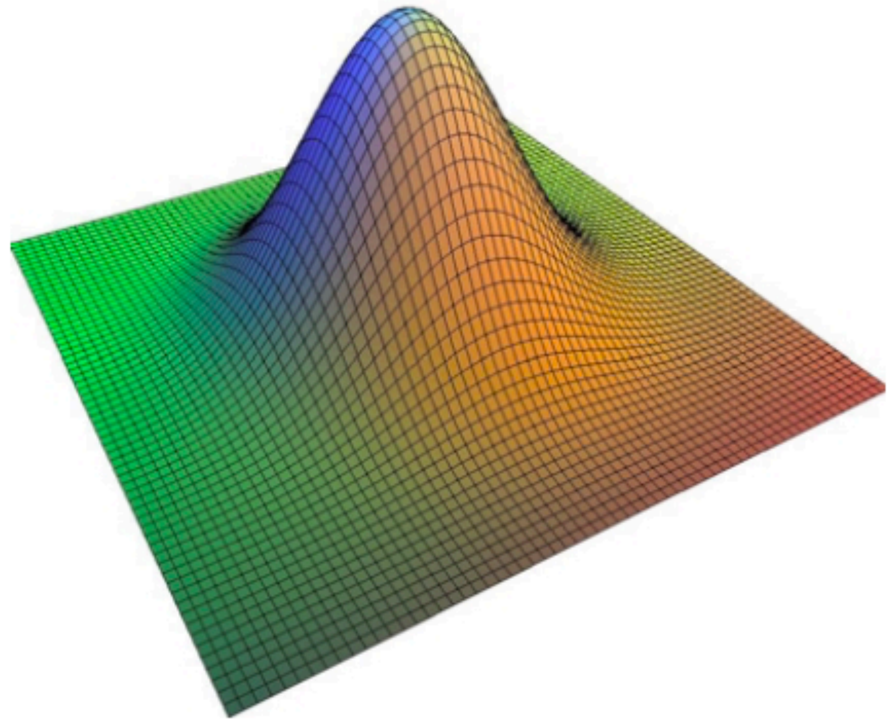
$$\sum \begin{bmatrix} 0.11 & 0.16 & 0.03 \\ 0.16 & 0.26 & 0.01 \\ 0.11 & 0.16 & 0.01 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.9 \\ 0.1 & 0.1 & 1 \\ 0 & 0.1 & 1 \end{bmatrix} = 0.1$$

Bilateral Filter

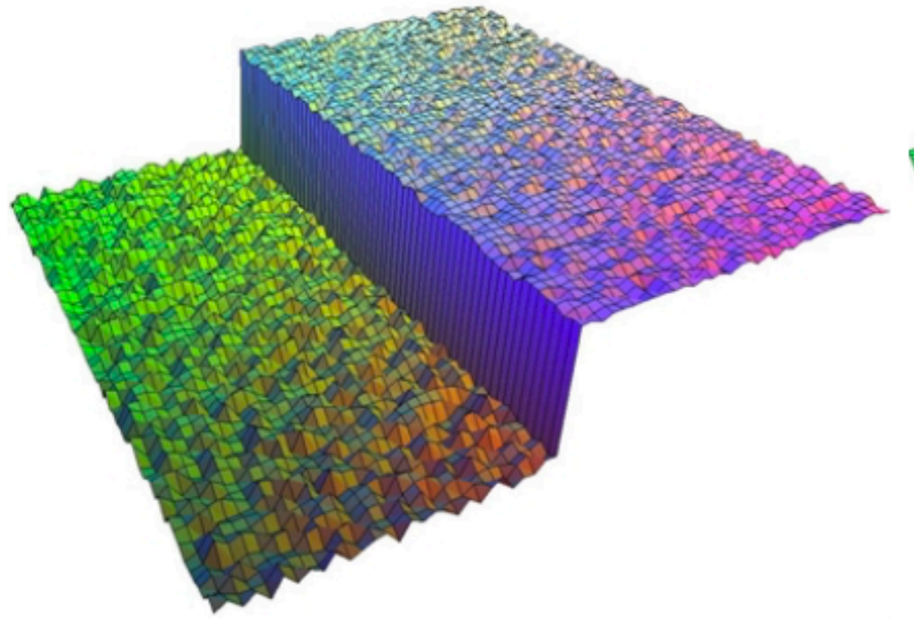
Bilateral Filter



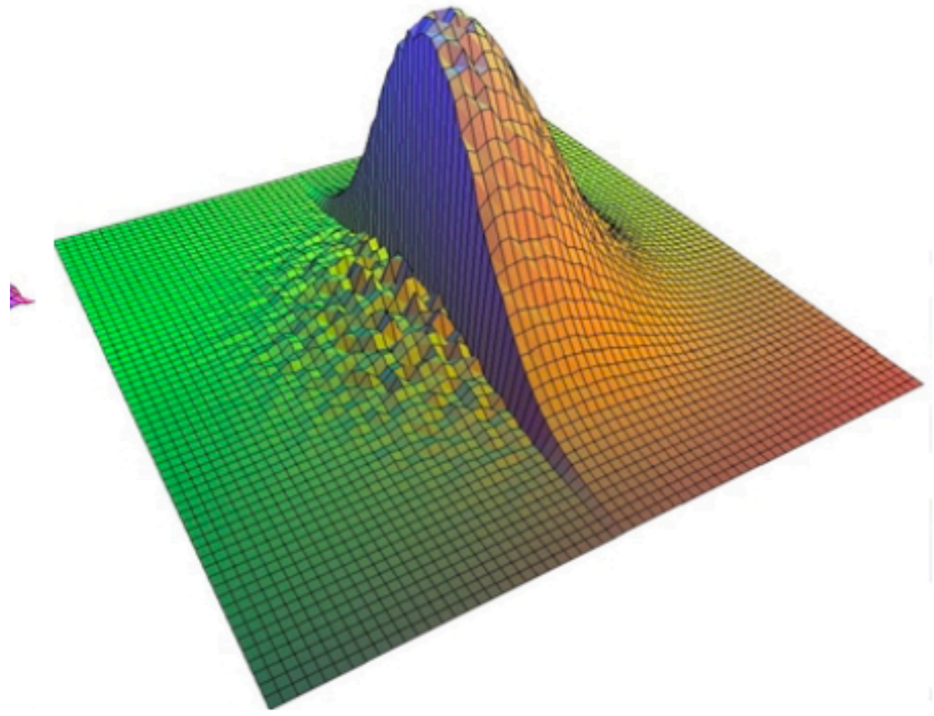
Input



Domain Kernel

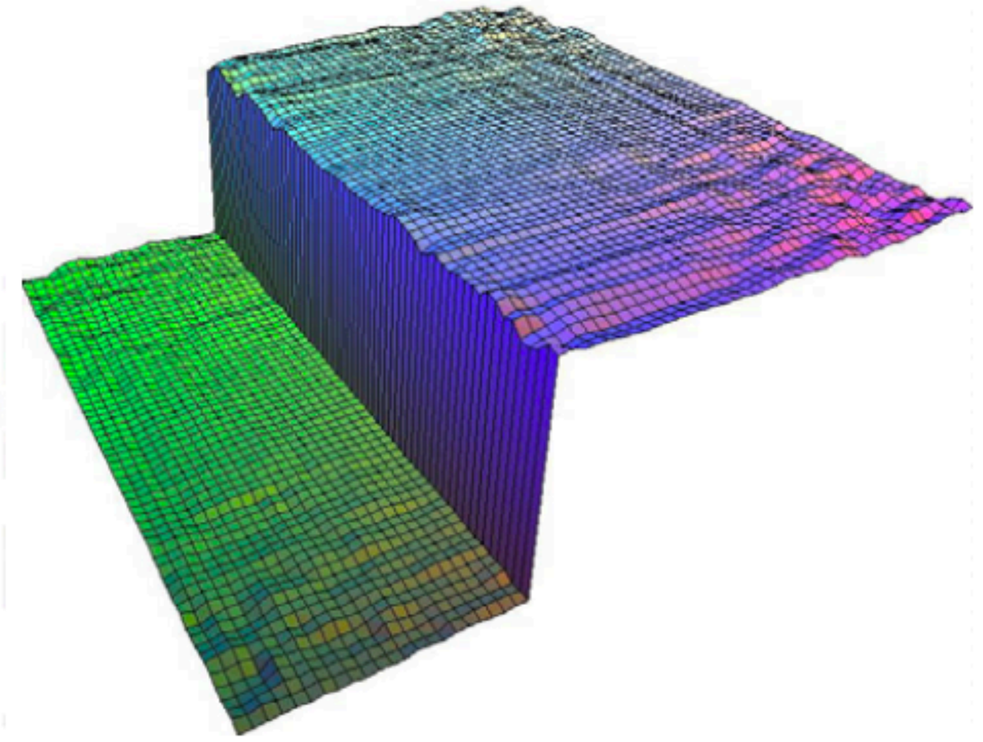


Range Kernel Influence



Bilateral Filter

(domain * range)



Output

Images from: Durand and Dorsey, 2002

Bilateral Filter Application: Denoising

Noisy Image

Gaussian Filter

Bilateral Filter

Slide Credit: Alexander Wong

Bilateral Filter Application: Cartooning

Original Image

After 5 iterations of **Bilateral** Filter

Slide Credit: Alexander Wong

Bilateral Filter Application: Flash Photography

Non-flash images taken under low light conditions often suffer from excessive **noise** and **blur**

But there are problems with **flash images**:
— colour is often unnatural
— there may be strong shadows or specularities

Idea: Combine flash and non-flash images to achieve better exposure and colour balance, and to reduce noise

Bilateral Filter Application: Flash Photography

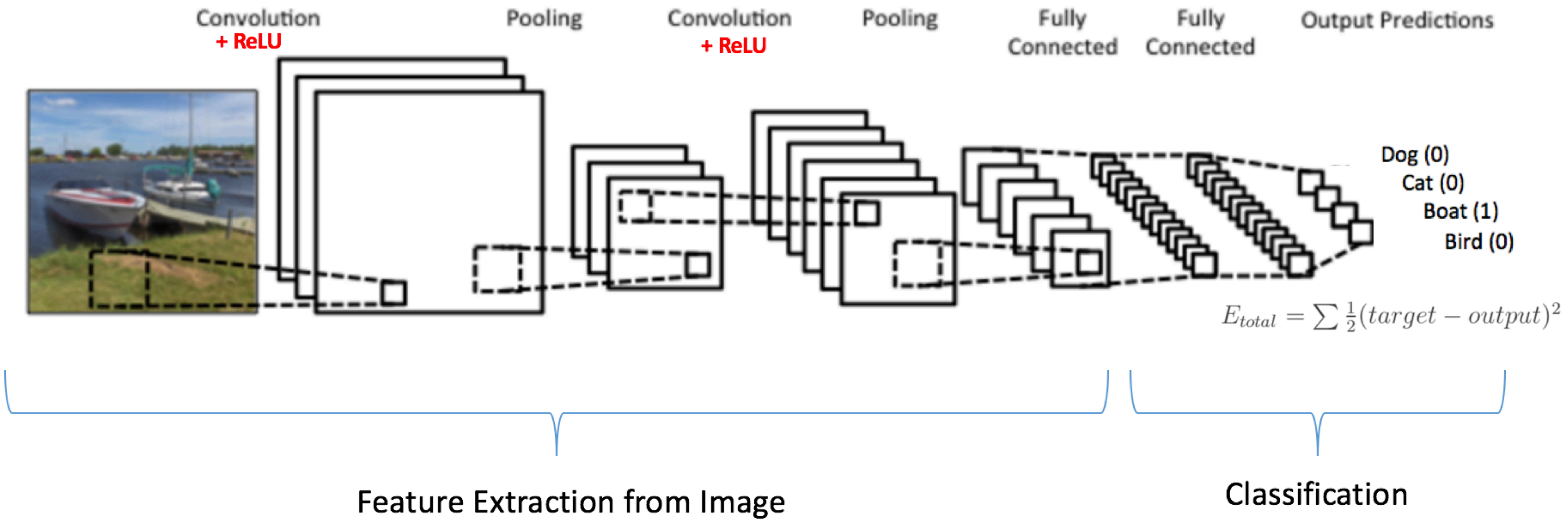
System using 'joint' or 'cross' bilateral filtering:

Flash | No-Flash | Detail Transfer with Denoising

'Joint' or 'Cross' bilateral: Range kernel is computed using a separate guidance image instead of the input image

Figure Credit: Petschnigg et al., 2004

Aside: Linear Filter with ReLU



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

→

9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1

Result of: Linear Image Filtering → After Non-linear ReLU

Summary

We covered two three **non-linear filters**: Median, Bilateral, ReLU

Separability (of a 2D filter) allows for more efficient implementation (as two 1D filters)

Convolution is **associative** and **symmetric**

Convolution of a Gaussian with a Gaussian is another Gaussian

The **median filter** is a non-linear filter that selects the median in the neighbourhood

The **bilateral filter** is a non-linear filter that considers both spatial distance and range (intensity) distance, and has edge-preserving properties