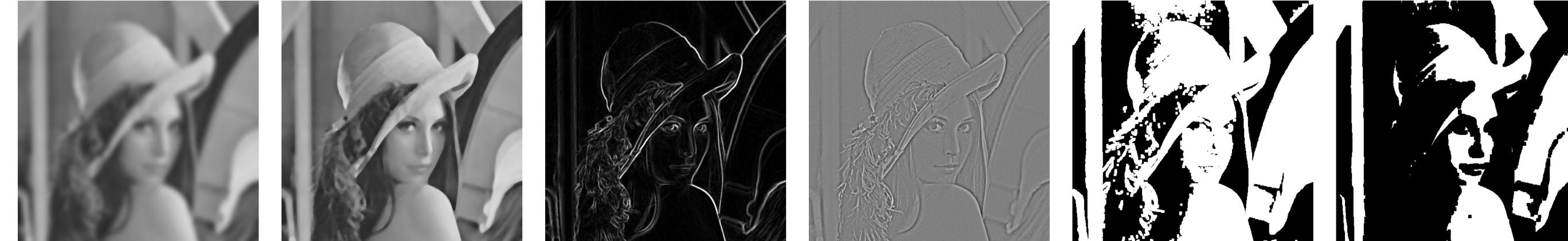




CPSC 425: Computer Vision



Lecture 6: Image Filtering (final)

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

Menu for Today (September 17, 2018)

Topics:

- Associativity and symmetry
- Non-linear Filters: Median, ReLU
- Bilateral Filter
- iClicker Quiz

Readings:

- **Today's** Lecture: [Optional] Forsyth & Ponce (2nd ed.) 4.4
- **Next** Lecture: [Optional] Forsyth & Ponce (2nd ed.) 4.4

Reminders:

- **Assignment 1:** Image Filtering and Hybrid Images due **September 24th**

Today's “fun” Example: Face on the Moon

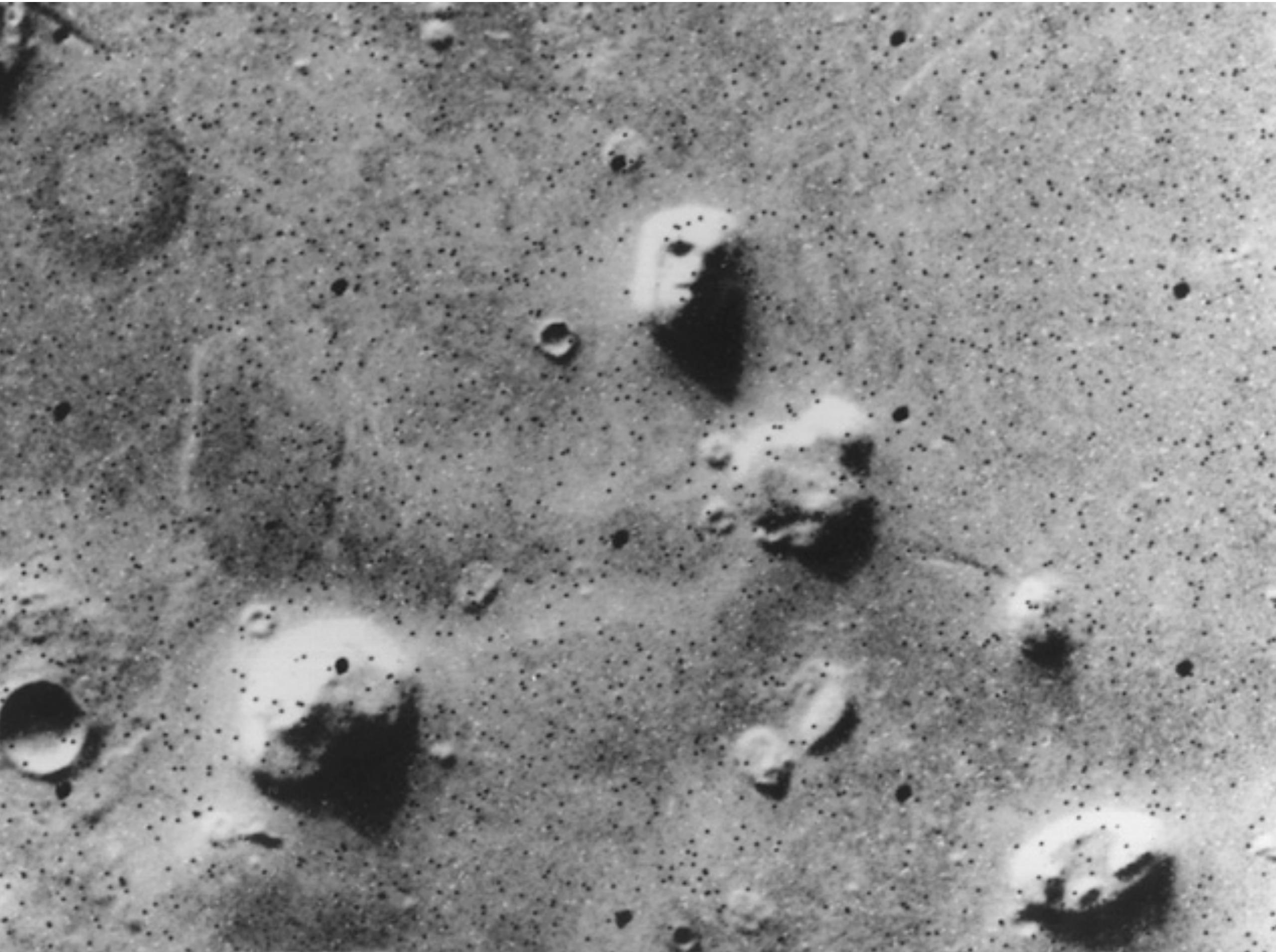
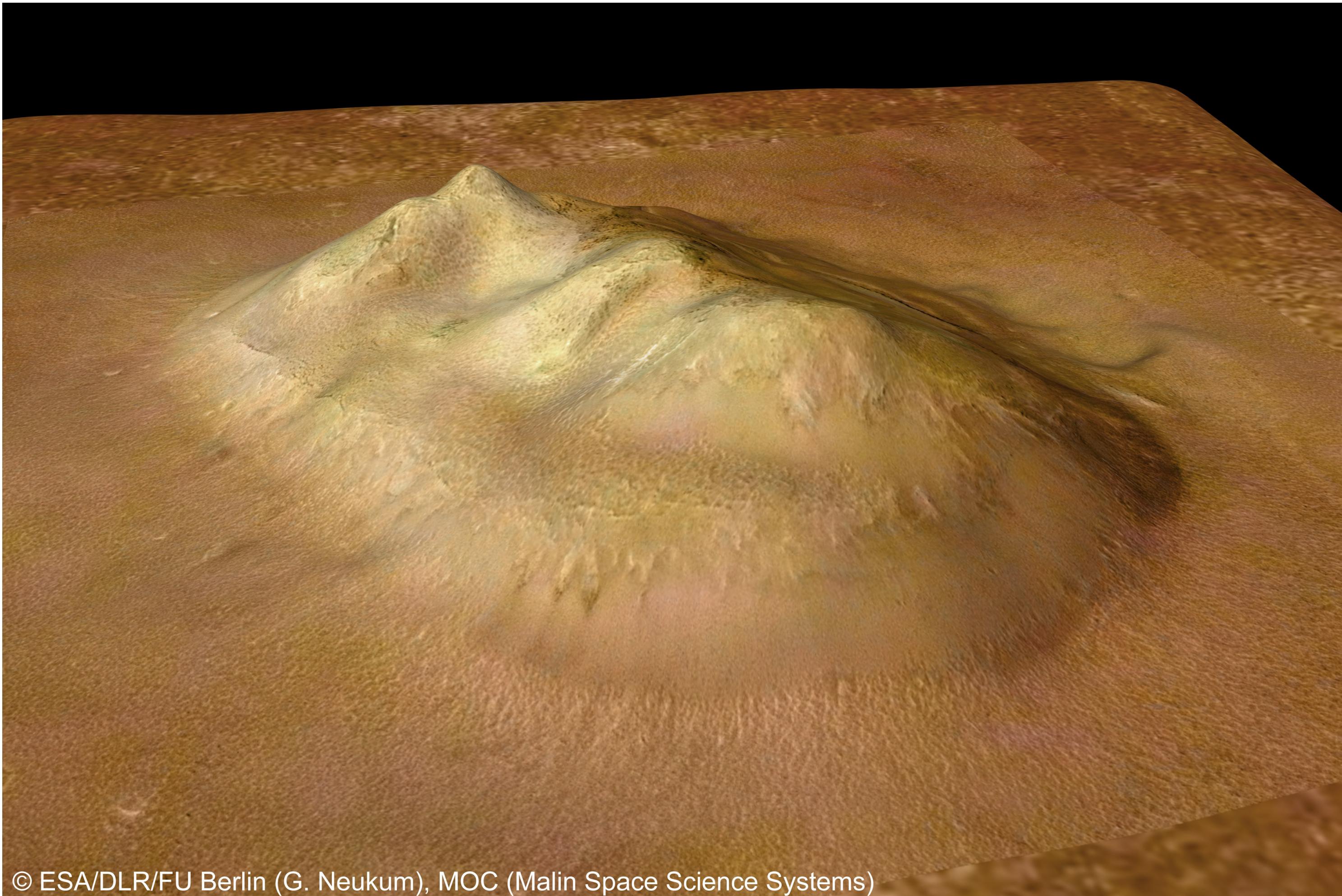


Image Credit: http://esamultimedia.esa.int/images/marsexpress/300-230906-3253-6-vk1-Cydonia_H.jpg

Today's “fun” Example: Face on the Moon



© ESA/DLR/FU Berlin (G. Neukum), MOC (Malin Space Science Systems)

Image Credit: http://esamultimedia.esa.int/images/marsexpress/311-230906-3253-6-3d5-Cydonia_H.jpg

Lecture 5: 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

Correlation vs. Convolution Interpretations

Correlation: Generally measures similarity between two signals. In our case, this would mean similarity between a filter and an image patch it is being applied to.

Convolution: Generally measures effect one signal has on another signal.

Linear Filters: Properties (recall **Lecture 4**)

Let \otimes denote convolution. Let $I(X, Y)$ be a digital image

Superposition: Let F_1 and F_2 be digital filters

$$(F_1 + F_2) \otimes I(X, Y) = F_1 \otimes I(X, Y) + F_2 \otimes I(X, Y)$$

Scaling: Let F be digital filter and let k be a scalar

$$(kF) \otimes I(X, Y) = F \otimes (kI(X, Y)) = k(F \otimes I(X, Y))$$

Shift Invariance: Output is local (i.e., no dependence on absolute position)

An operation is **linear** if it satisfies both **superposition** and **scaling**

Linear Filters: Additional Properties

Let \otimes denote convolution. Let $I(X, Y)$ be a digital image. Let F and G be digital filters

- Convolution is **associative**. That is,

$$G \otimes (F \otimes I(X, Y)) = (G \otimes F) \otimes I(X, Y)$$

- Convolution is **symmetric**. That is,

$$(G \otimes F) \otimes I(X, Y) = (F \otimes G) \otimes I(X, Y)$$

Convolving $I(X, Y)$ with filter F and then convolving the result with filter G can be achieved in single step, namely convolving $I(X, Y)$ with filter $G \otimes F = F \otimes G$

Example: Two Box Filters

```
filter = boxfilter(3)
```

```
signal.correlate2d(filter, filter, 'full')
```

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \otimes \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \frac{1}{81} \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 2 & 4 & 6 & 4 & 2 \\ \hline 3 & 6 & 9 & 6 & 3 \\ \hline 2 & 4 & 6 & 4 & 2 \\ \hline 1 & 2 & 3 & 2 & 1 \\ \hline \end{array}$$

3x3 Box 3x3 Box

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9} \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \otimes \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = \frac{1}{81} \begin{matrix} & & & & & & \\ & 1 & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{matrix}$$

3x3 Box

Output

Example: Two Box Filters

Treat one filter as padded “image”

$$\begin{array}{c} \text{Input} \\ \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \\ \frac{1}{9} \end{array} \otimes \begin{array}{c} \frac{1}{9} \\ \left[\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right] \\ \text{3x3 Box} \end{array} = \frac{1}{81} \begin{array}{c} \text{Output} \\ \left[\begin{array}{ccccccc} & & & & & & \\ 1 & 2 & & & & & \\ & & & & & & \end{array} \right] \end{array}$$

Example: Two Box Filters

Treat one filter as padded “image”

$$\begin{array}{c} \text{Input} \\ \begin{array}{|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \end{array} \otimes \begin{array}{c} \frac{1}{9} \\ \text{3x3 Box} \end{array} = \begin{array}{|c|c|c|c|c|c|c|} \hline & & & & & & \\ \hline & & & & & & \\ \hline & 1 & 2 & 3 & & & \\ \hline & & & & & & \\ \hline \end{array}$$

3x3 Box

Output

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9} \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} \otimes \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} = \frac{1}{81} \begin{matrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{matrix}$$

3x3 Box

Output

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9} \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

3x3 Box

$$\otimes \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

3x3 Box

$$= \frac{1}{81} \begin{array}{|c|c|c|c|c|c|c|c|} \hline & & & & & & & \\ \hline & 1 & 2 & 3 & 2 & 1 & & \\ \hline 2 & 4 & 6 & 4 & 2 & & & \\ \hline 3 & 6 & 9 & 6 & 3 & & & \\ \hline 2 & 4 & 6 & 4 & 2 & & & \\ \hline 1 & 2 & 3 & 2 & 1 & & & \\ \hline & & & & & & & \\ \hline \end{array}$$

Output

Example: Two Box Filters

Treat one filter as padded “image”

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

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

3x3 Box

$$\otimes \frac{1}{9} \begin{array}{|ccc|} \hline 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ \hline \end{array}$$

3x3 Box

$$= \frac{1}{81}$$

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

Output

Example: Two Box Filters

```
filter = boxfilter(3)
```

```
temp = signal.correlate2d(filter, filter, 'full')
```

```
signal.correlate2d(filter, temp, 'full')
```

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{729}$$

3x3 Box 3x3 Box 3x3 Box

1	3	6	7	6	3	1
3	9	18	21	18	9	3
6	18	36	42	36	18	6
7	21	42	49	42	21	7
6	18	36	42	36	18	6
3	9	18	21	18	9	3
1	3	6	7	6	3	1

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

$$\frac{1}{16} \otimes \begin{matrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{matrix} = \frac{1}{256}$$

The diagram illustrates the application of a separable Gaussian filter. It shows the multiplication of a 5x5 input matrix by a 1x5 filter vector. The result is a 5x5 output matrix. Only the top-left element of the output matrix is highlighted with a red border, representing the result of the convolution step.

Example: Separable Gaussian Filter

$$\frac{1}{16} \otimes \begin{matrix} 1 \\ 16 \\ \otimes \\ 1 \\ 16 \\ \end{matrix} = \frac{1}{256}$$

The diagram illustrates the convolution of a 5x5 input image with a 1D separable Gaussian filter. The input image is a 5x5 matrix with values: 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. A red box highlights the central element 4. The filter is a 1D vector: 1, 4, 6, 4, 1. The result of the convolution is a 5x5 output image where the central element is 16, highlighted by a red box.

Example: Separable Gaussian Filter

$$\frac{1}{16} \otimes \begin{matrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{matrix} = \frac{1}{256}$$

The diagram illustrates the convolution of a 5x5 input matrix with a 1D kernel. The input matrix has a value of 1 at its bottom-right corner, highlighted by a red border. The 1D kernel is shown vertically below the input. The result of the convolution is a 5x5 output matrix where the bottom-right corner also has a value of 1, highlighted by a red border. The other values in the output matrix are 0.

The input matrix is:

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

The kernel is:

1
4
6
4
1

The output matrix is:

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

Example: Separable Gaussian Filter

$$\frac{1}{16} \otimes \frac{1}{16} = \frac{1}{256}$$

The diagram illustrates the element-wise multiplication (Hadamard product) of two 5x5 matrices. The first matrix is a 5x5 identity matrix scaled by $\frac{1}{16}$. The second matrix is a 5x5 Gaussian kernel scaled by $\frac{1}{16}$. The result is a 5x5 matrix where each element is the product of the corresponding elements from the two input matrices.

The resulting matrix is:

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

Pre-Convoving Filters

Convolving two filters of size $m \times m$ and $n \times n$ results in filter of size:

$$\left(n + 2 \left\lfloor \frac{m}{2} \right\rfloor\right) \times \left(n + 2 \left\lfloor \frac{m}{2} \right\rfloor\right)$$

More broadly for a set of K filters of sizes $m_k \times m_k$ the resulting filter will have size:

$$\left(m_1 + 2 \sum_{k=2}^K \left\lfloor \frac{m_k}{2} \right\rfloor\right) \times \left(m_1 + 2 \sum_{k=2}^K \left\lfloor \frac{m_k}{2} \right\rfloor\right)$$

Gaussian: An Additional Property

Let \otimes denote convolution. Let $G_{\sigma_1}(x)$ and $G_{\sigma_2}(x)$ be two 1D Gaussians

$$G_{\sigma_1}(x) \otimes G_{\sigma_2}(x) = G_{\sqrt{\sigma_1^2 + \sigma_2^2}}(x)$$

Convolution of two Gaussians is another Gaussian

Special case: Convolving with $G_\sigma(x)$ twice is equivalent to $G_{\sqrt{2}\sigma}(x)$

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:

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
---	---	---	---	----	----	----	----	----

Output

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:

The diagram illustrates the bilateral filter kernel as a product of two components: a domain kernel and a range kernel. The domain kernel is represented by a green box containing the term $\exp^{-\frac{x^2+y^2}{2\sigma_d^2}}$. The range kernel is represented by a blue box containing the term $\exp^{-\frac{(I(X+x, Y+y) - I(X, Y))^2}{2\sigma_r^2}}$. The two boxes are separated by a thin vertical line and are multiplied together to form the full bilateral weight.

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

(with appropriate normalization)

Bilateral Filter

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



0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

(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

Range * Domain Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

multiply
→

sum to 1
→

0.11	0.16	0.03
0.16	0.26	0.01
0.11	0.16	0.01

(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
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
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

$$\sigma_r = 0.45$$

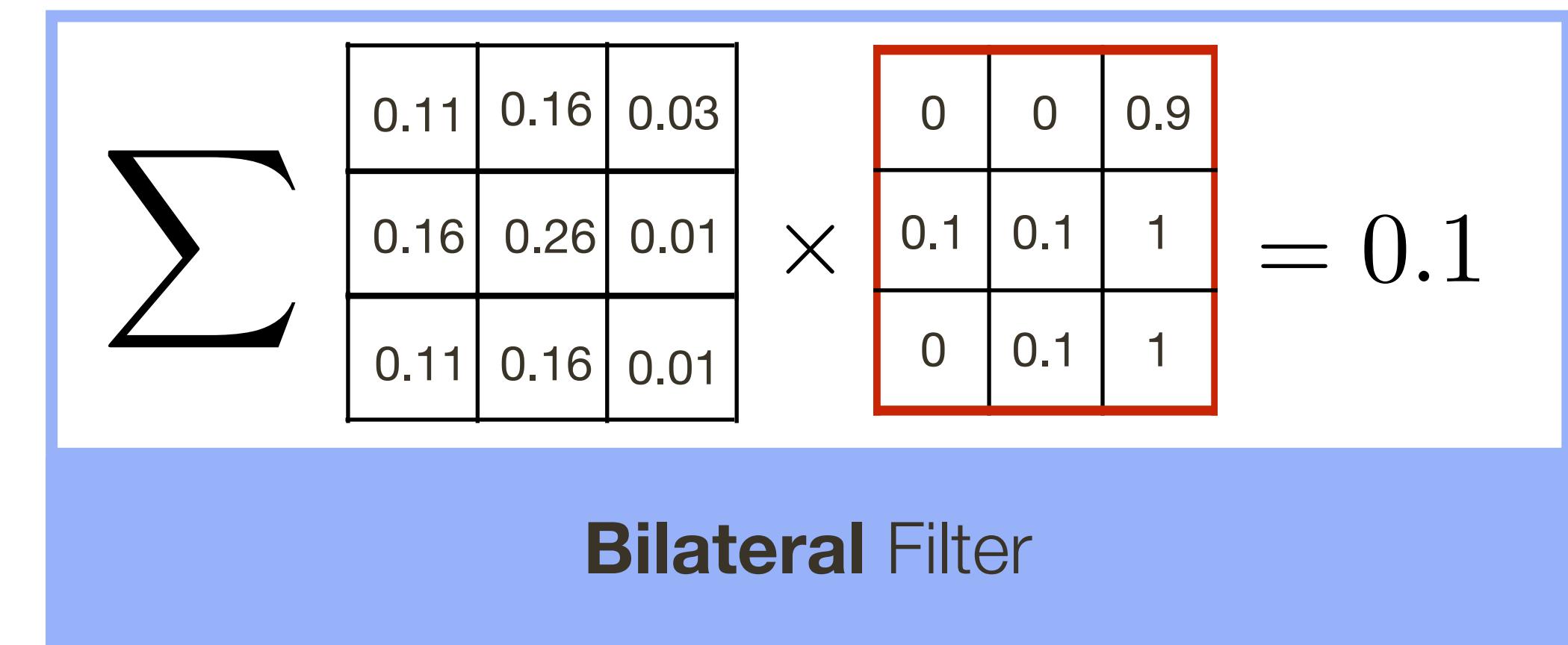
0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

multiply
→

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

(this is different for each locations in the image)

Range * Domain Kernel



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
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
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

\sum

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

0	0	0.9
0.1	0.1	1
0	0.1	1

= 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

Range * Domain Kernel

multiply

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

(this is different for each locations in the image)

0.11	0.16	0.03
0.16	0.26	0.01
0.11	0.16	0.01

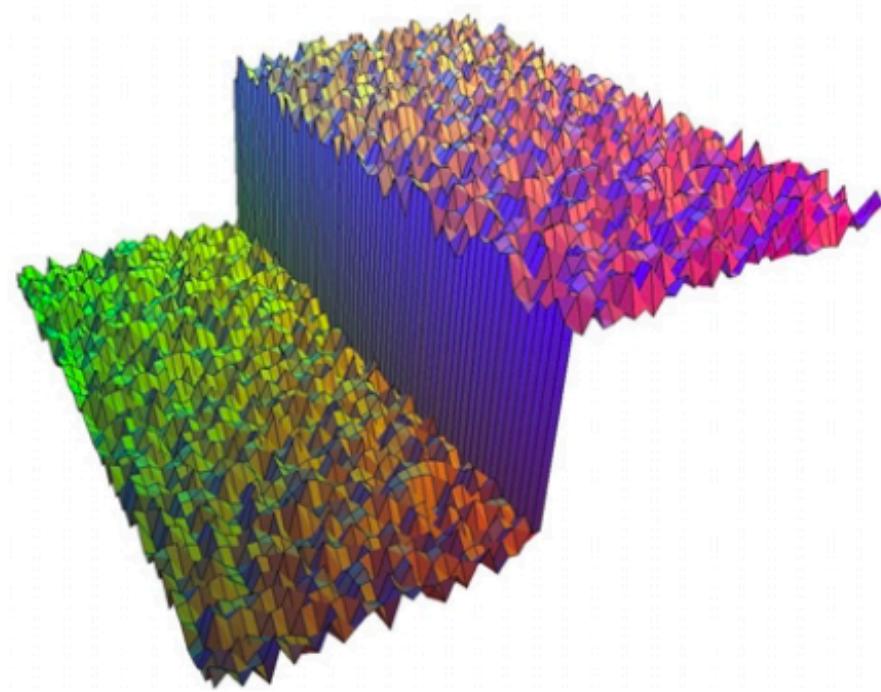
\times

0	0	0.9
0.1	0.1	1
0	0.1	1

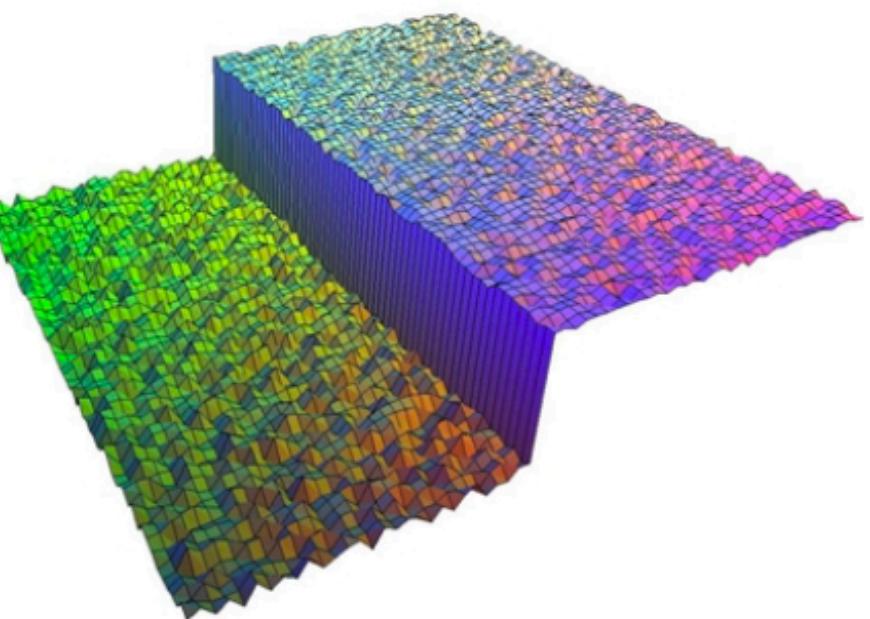
= 0.1

Bilateral Filter

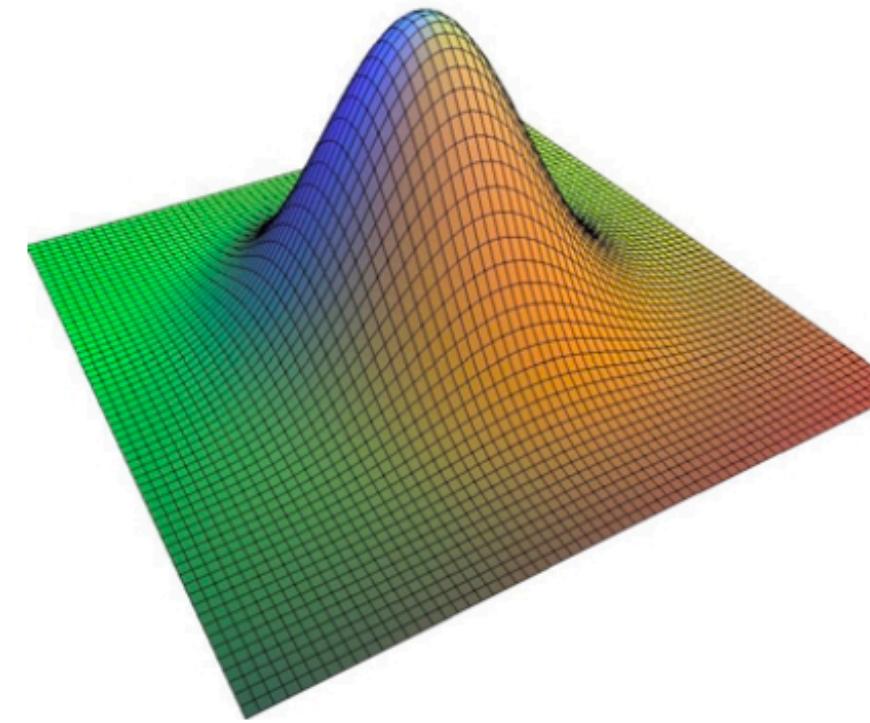
Bilateral Filter



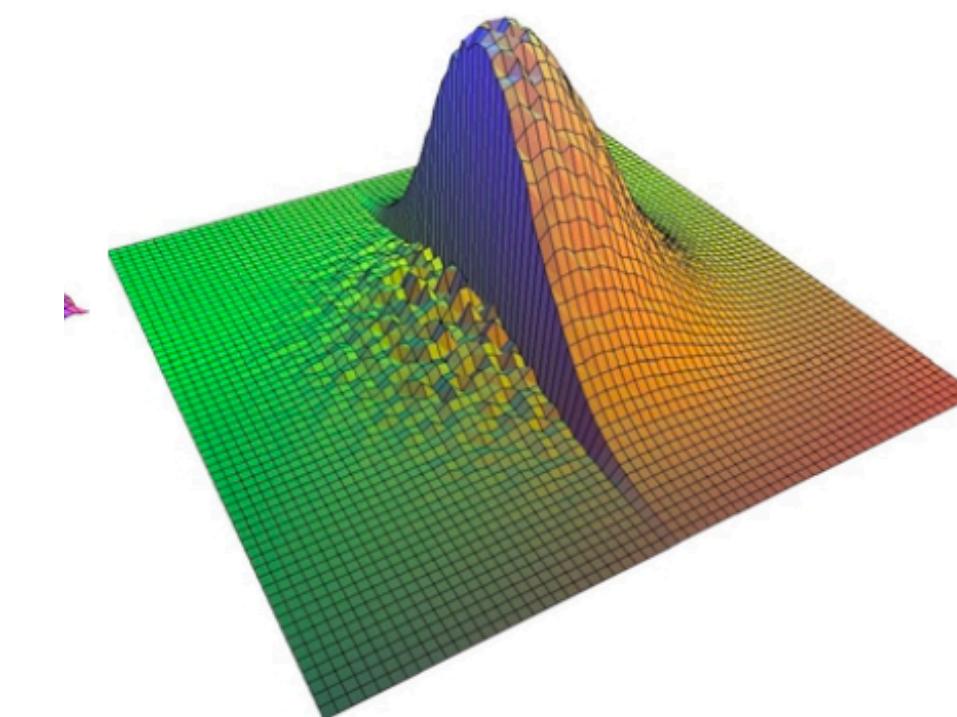
Input



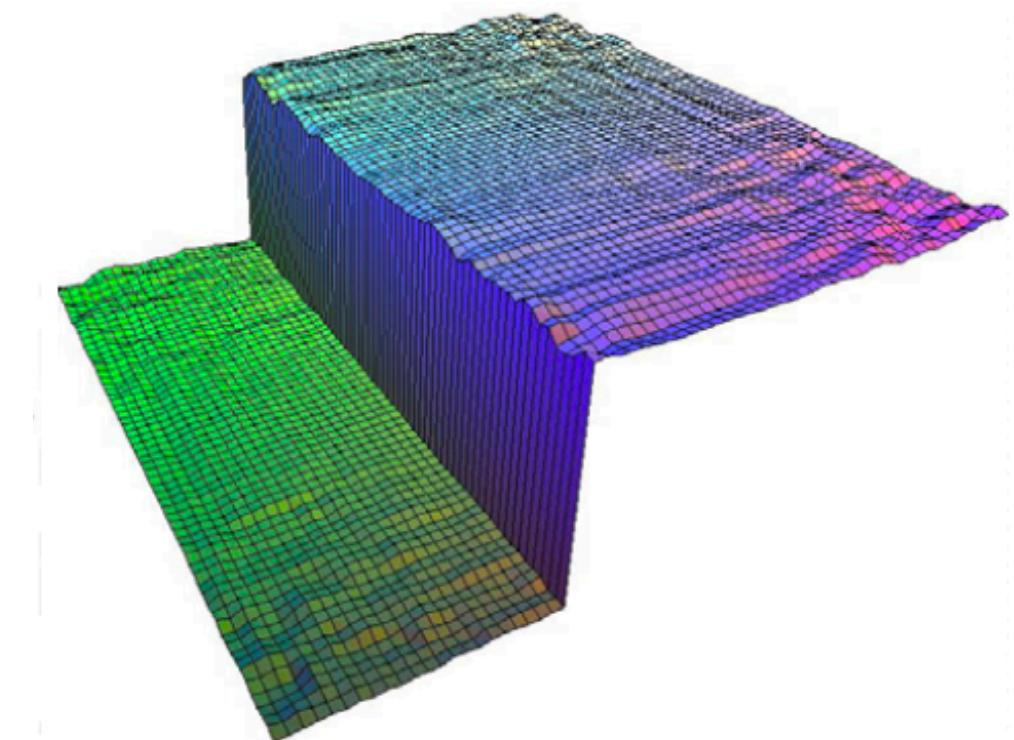
Range Kernel Influence



Domain Kernel



Bilateral Filter
(domain * range)



Output

Images from: Durand and Dorsey, 2002

Bilateral Filter Application: Denoising



Noisy Image



Gaussian Filter



Bilateral Filter

Bilateral Filter Application: Cartooning



Original Image



After 5 iterations of **Bilateral Filter**

Bilateral Filter Application: Flash Photography

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

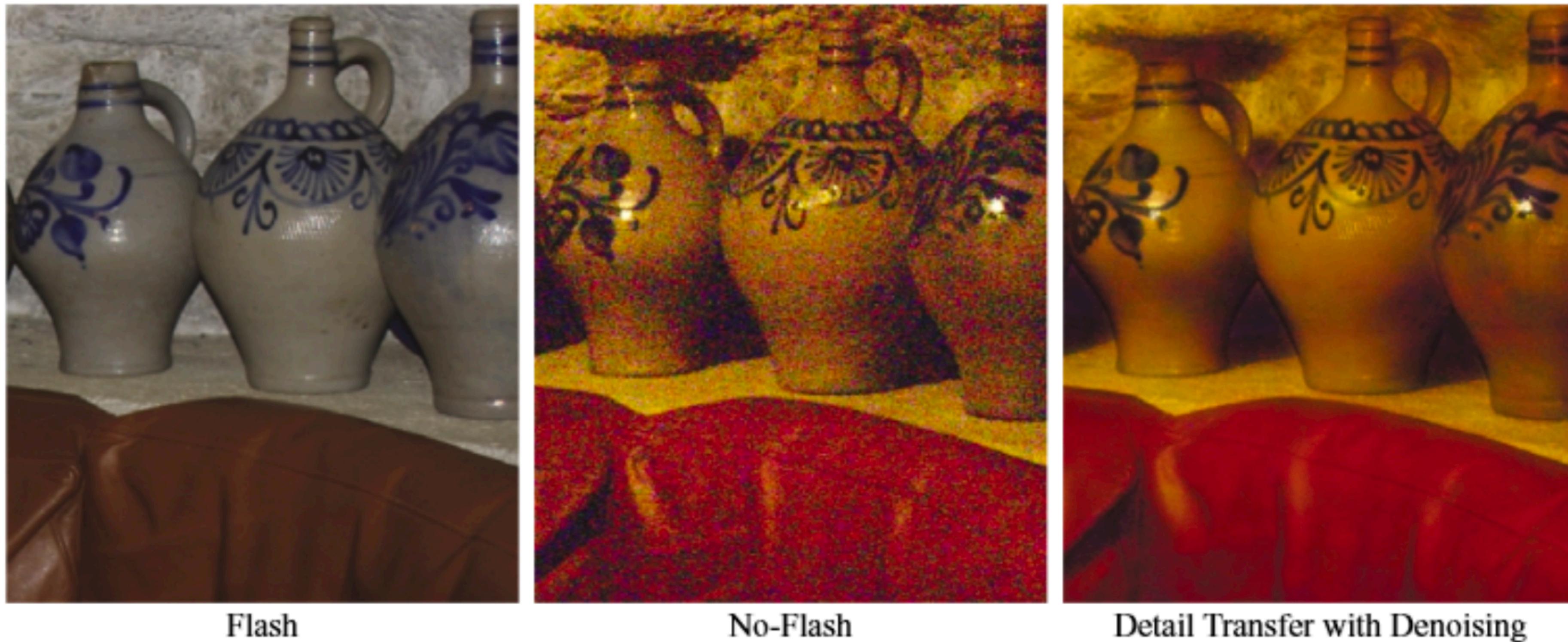
But there are problems with **flash images**:

- 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

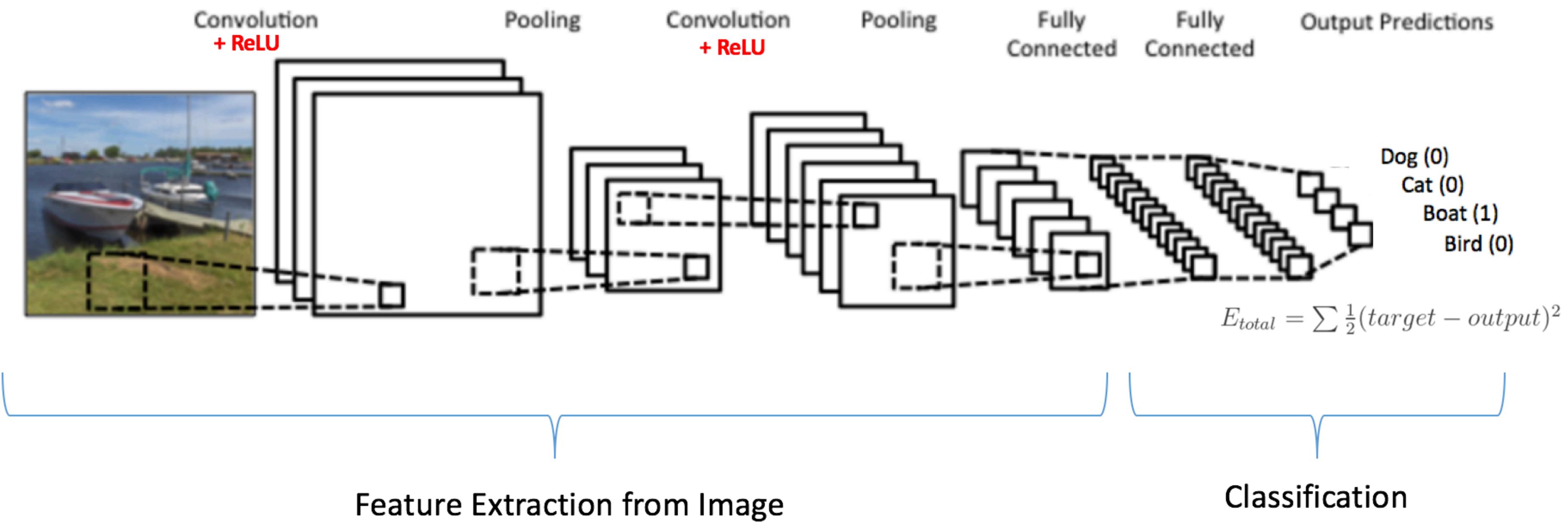
System using ‘joint’ or ‘cross’ bilateral filtering:



‘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



Please get your **iClickers** – Quiz

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