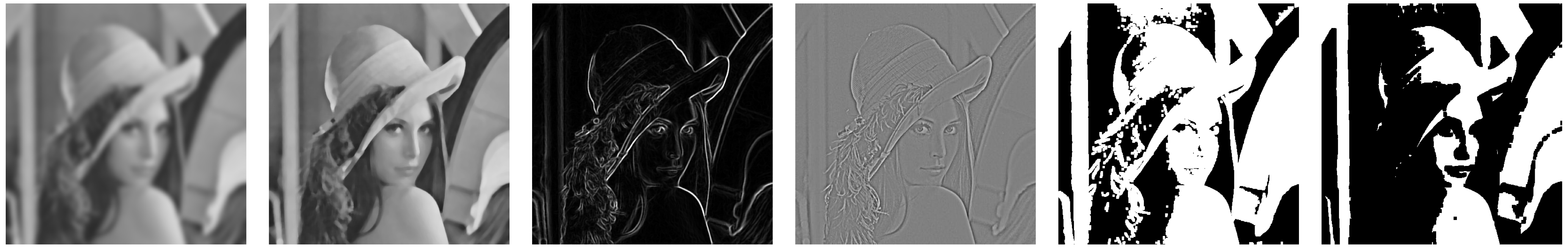
THE UNIVERSITY OF BRITISH COLUMBIA

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 21, 2020**)

Topics:

— **Non-linear** Filters: Median, ReLU

— **Bilateral** Filter

Readings:

— **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 **September 30**th

— Discussions on **Piazza** are going reasonably well (avg response time 33min)

— I will add Office Hour on **Tuesdays** @ 5pm (Zoom link will be posted)

Today’s “**fun**” Example: **Visual Question Answering**

http://vqa.cloudcv.org

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

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

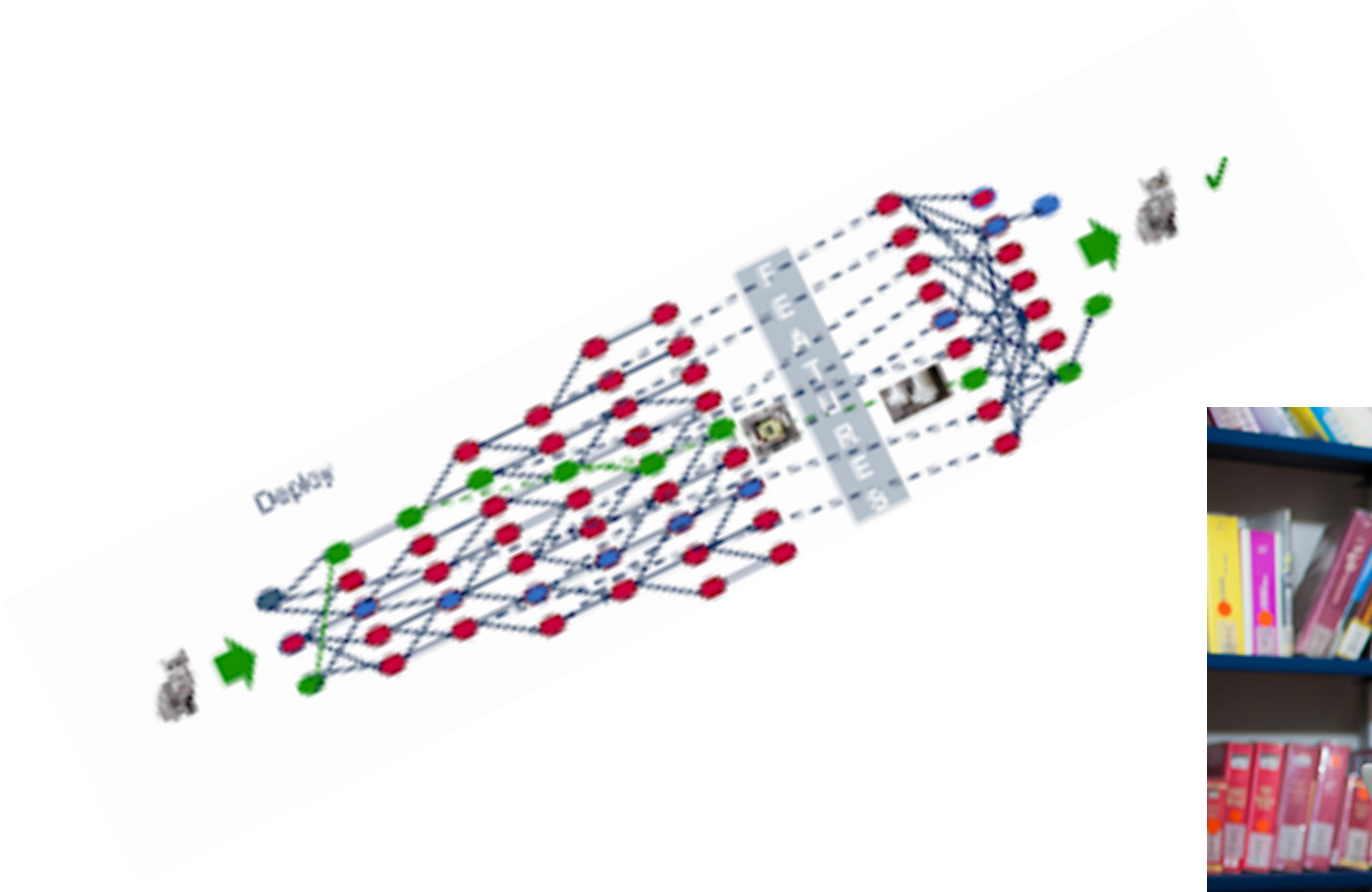
Hans could get 89% of the math questions right

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 5: Re-cap

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

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

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)

— separable filter can be expressed as an **outer product** of two 1D filters

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

Lecture 5: Re-cap The Convolution Theorem

Convolution **Theorem**:

Let $$i'(x,y) = f(x,y) \otimes i(x,y)$$

then $$\mathcal{I}'(w_x, w_y) = \mathcal{F}(w_x, w_y)\, \mathcal{I}(w_x, w_y)$$

where $\mathcal{I}'(w_x, w_y)$, $\mathcal{F}(w_x, w_y)$, and $\mathcal{I}(w_x, w_y)$ are Fourier transforms of $i'(x,y)$, $f(x,y)$ and $i(x,y)$

At the expense of two **Fourier** transforms and one inverse Fourier transform, convolution can be reduced to (complex) multiplication

Lecture 5: Assignment 1 Intuition

Preview of **Part 3** of your homework

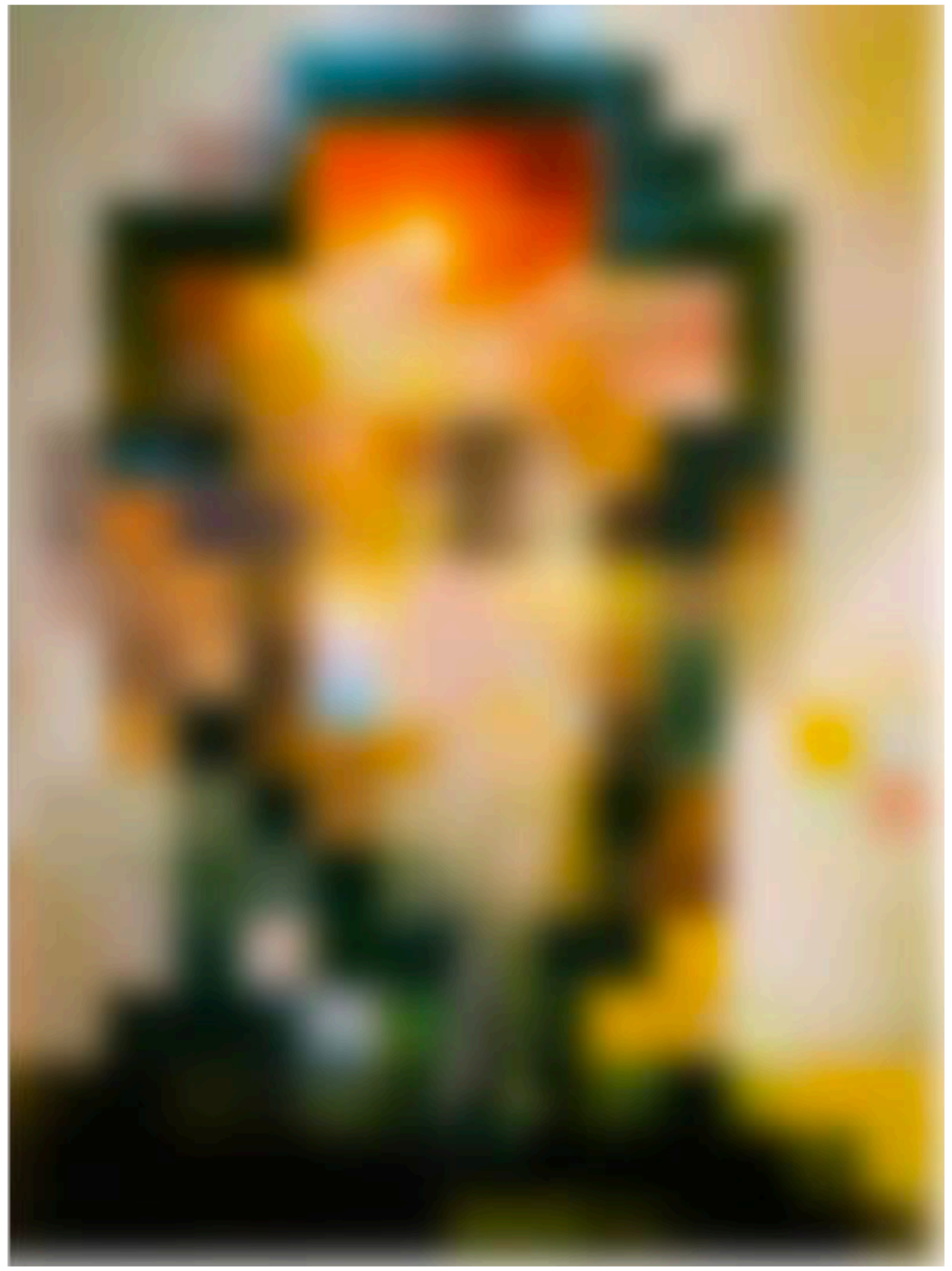
Gala Contemplating the Mediterranean Sea Which at Twenty Meters Becomes the Portrait of Abraham Lincoln (Homage to Rothko)

Salvador Dali, 1976

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

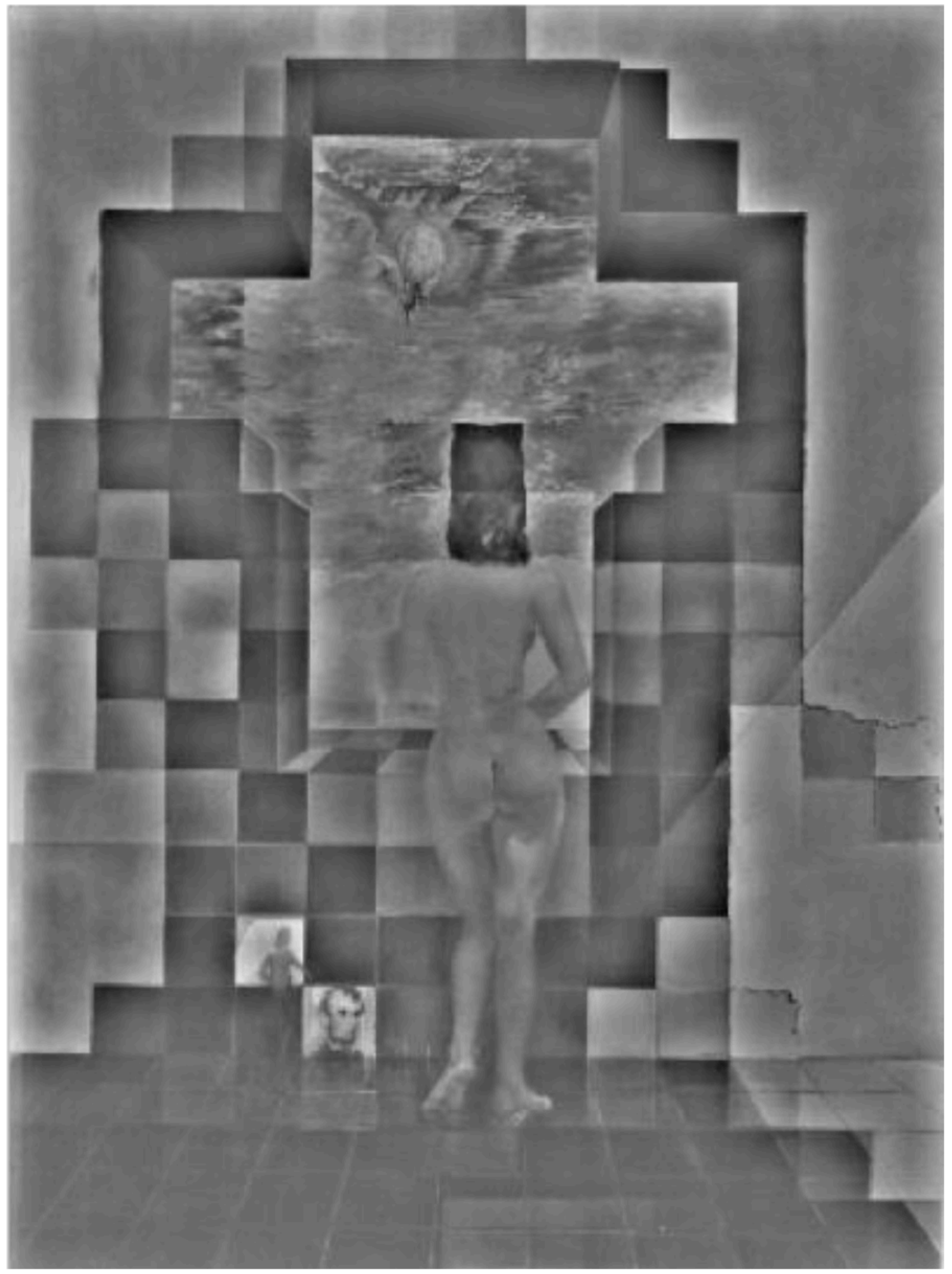
Lecture 5: Assignment 1 Intuition

Preview of **Part 3** of your homework



Low-pass filtered version

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)



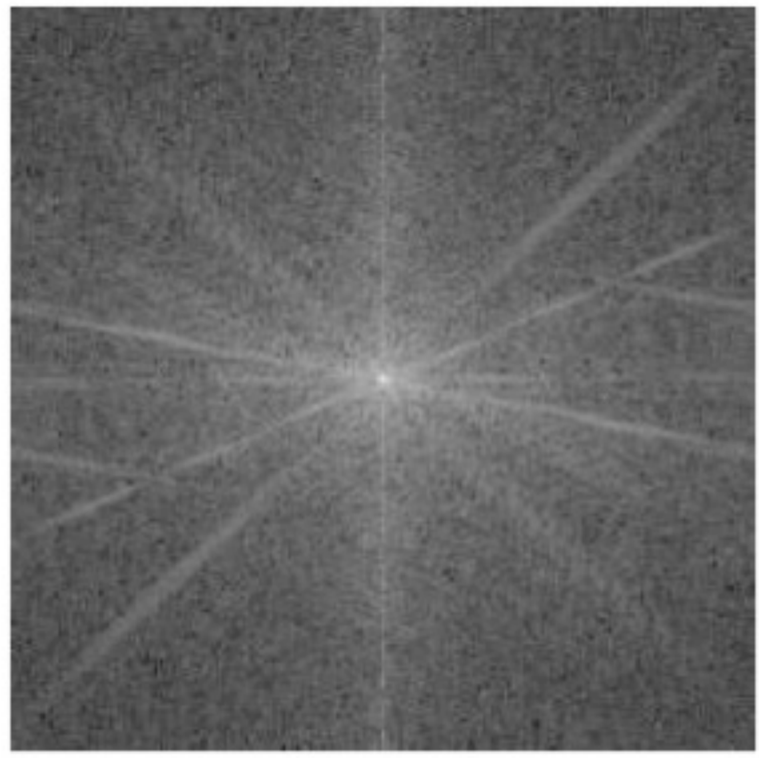
High-pass filtered version

Preview of **Part 3** of your homework

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

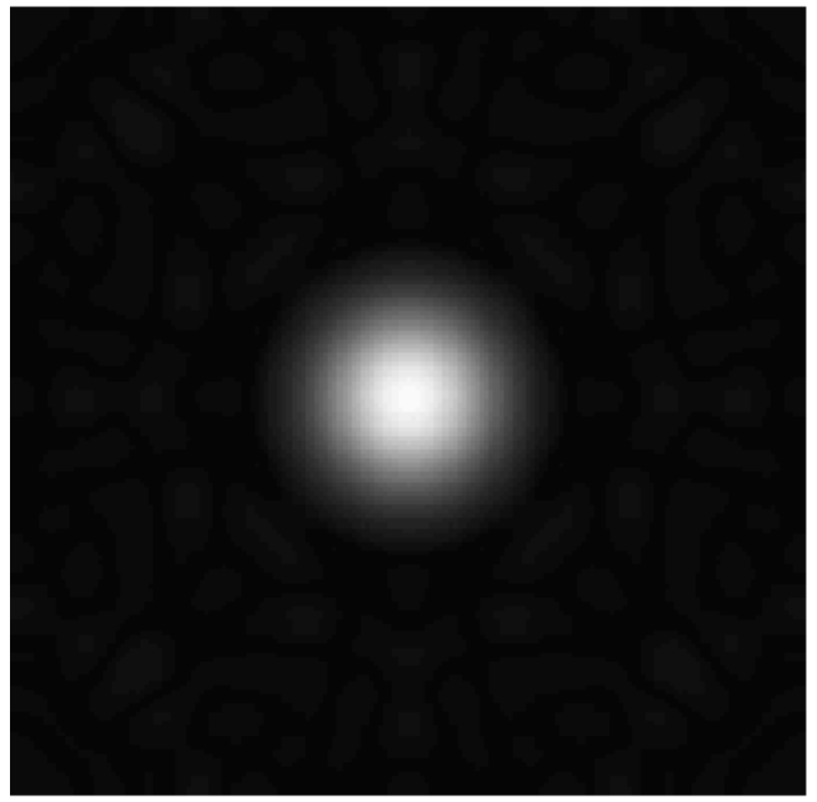
Lecture 5: Re-cap

image



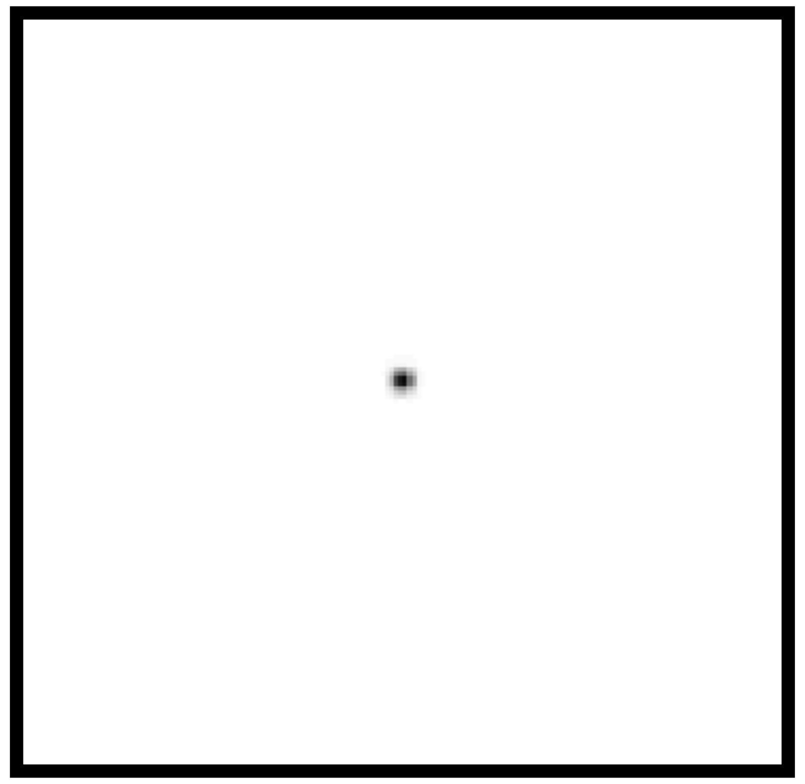
FFT (Mag)

complex
element-wise
multiplication



Low pass

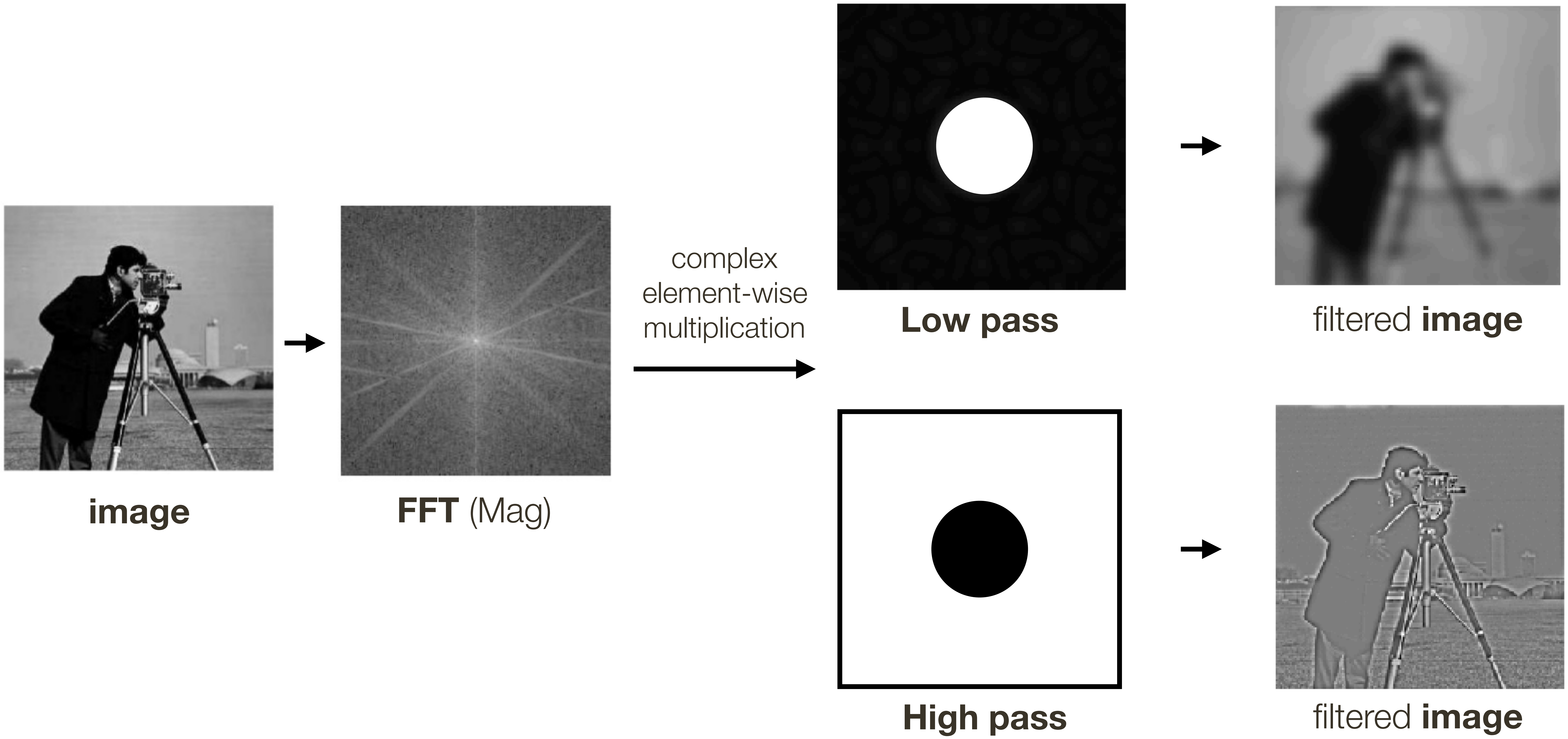
filtered **image**



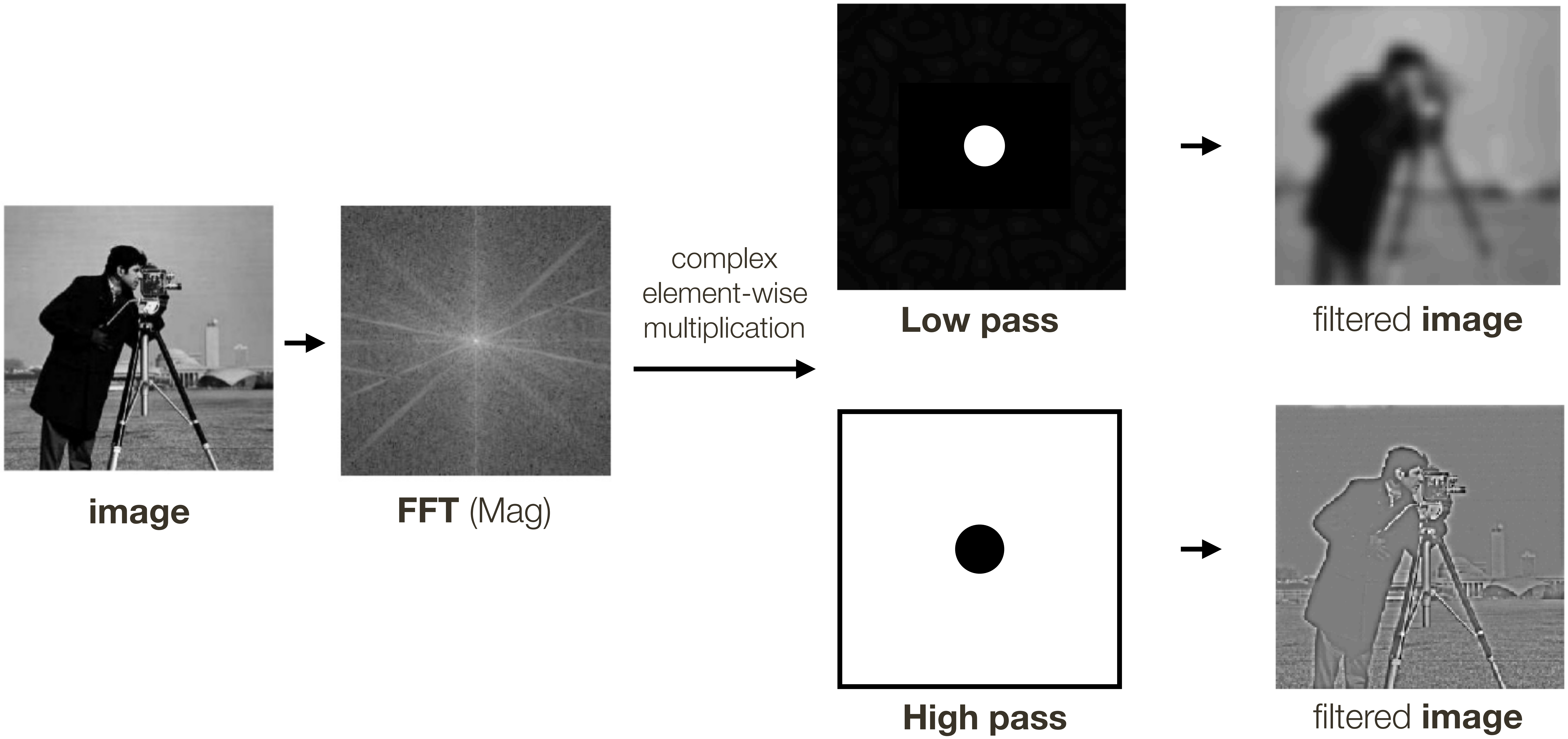
High pass

filtered **image**

Perfect **Low-pass / High-pass** Filtering



Perfect **Low-pass / High-pass** Filtering

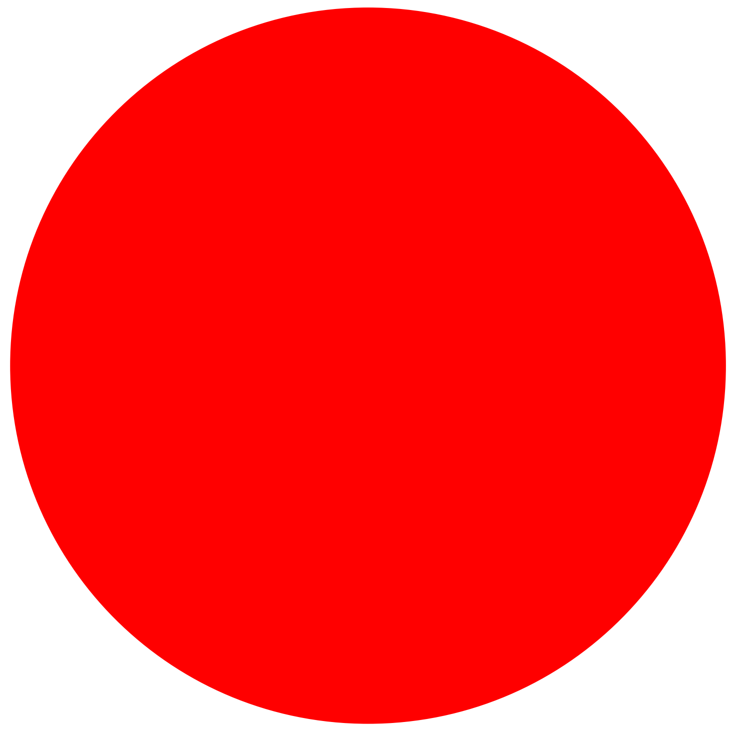


Low-pass Filtering = "Smoothing"?

Box Filter

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

Pillbox Filter



Gaussian Filter

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

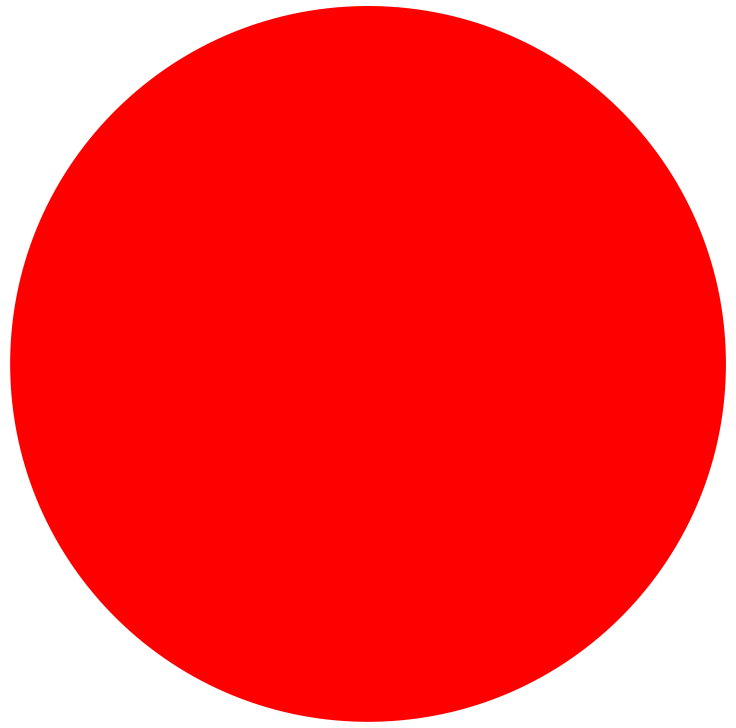
Are all of these **low-pass** filters?

Low-pass Filtering = “Smoothing”

Box Filter

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

Pillbox Filter



Gaussian Filter

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

Are all of these **low-pass** filters?

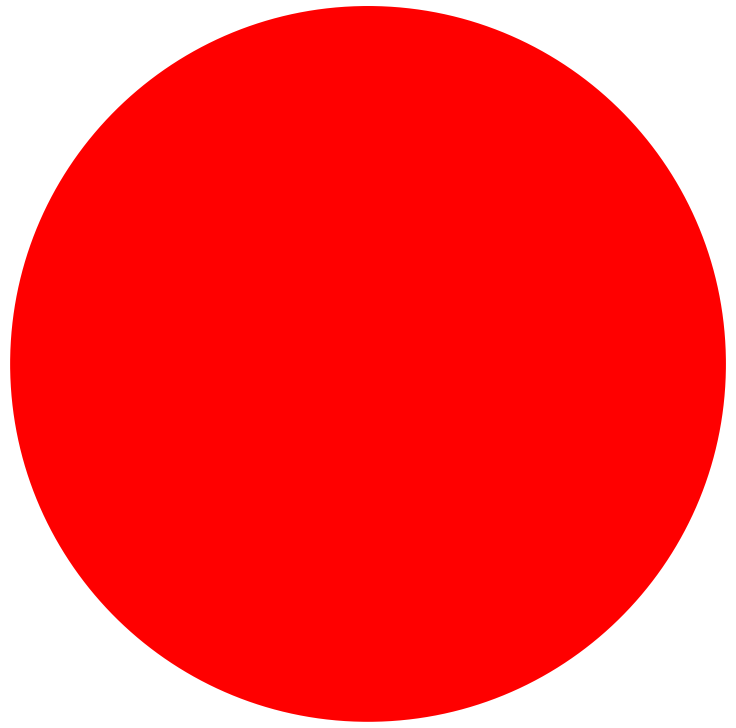
Low-pass filter: Low pass filter filters out all of the high frequency content of the image, only low frequencies remain

Low-pass Filtering = "Smoothing"

Box Filter

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

Pillbox Filter



Gaussian Filter

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

$$\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Image

Are all of these **low-pass** filters?

Low-pass filter: Low pass filter filters out all of the high frequency content of the image, only low frequencies remain

After long detour …

lets go back to **efficiency**

Speeding Up **Convolution** (The Convolution Theorem)

Convolution **Theorem**:

Let $$i'(x,y) = f(x,y) \otimes i(x,y)$$

then $$\mathcal{I}'(w_x, w_y) = \mathcal{F}(w_x, w_y)\, \mathcal{I}(w_x, w_y)$$

where $\mathcal{I}'(w_x, w_y)$, $\mathcal{F}(w_x, w_y)$, and $\mathcal{I}(w_x, w_y)$ are Fourier transforms of $i'(x,y)$, $f(x,y)$ and $i(x,y)$

At the expense of two **Fourier** transforms and one inverse Fourier transform, convolution can be reduced to (complex) multiplication

Speeding Up **Convolution** (The Convolution Theorem)

General implementation of **convolution**:

At each pixel, (X, Y), there are	$m \times m$	multiplications
There are	$n \times n$	pixels in (X, Y)
Total:	$m^2 \times n^2$	multiplications

Convolution if FFT space:

Cost of FFT/IFFT for image: $\mathcal{O}(n^2 \log n)$

Cost of FFT/IFFT for filter: $\mathcal{O}(m^2 \log m)$

Cost of convolution: $\mathcal{O}(n^2)$

Note: not a function of filter size !!!

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 $G \otimes F = F \otimes G$

Note: Correlation, in general, is **not associative**.

Example: Two Box Filters

filter = boxfilter(3)

signal.correlate2d(filter, filter,′ full′)

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} = \frac{1}{81}\begin{bmatrix}1 & 2 & 3 & 2 & 1\\2 & 4 & 6 & 4 & 2\\3 & 6 & 9 & 6 & 3\\2 & 4 & 6 & 4 & 2\\1 & 2 & 3 & 2 & 1\end{bmatrix}$$

3x3 **Box** 3x3 **Box**

Example: Two Box Filters

Treat one filter as padded "image"

Note, in this case you have to pad maximally until two filters no longer overlap

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

3x3 **Box** 3x3 **Box** **Output**

Example: Two Box Filters

Treat one filter as padded “image”

$$
\frac{1}{9}\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}
\otimes
\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}
=
\frac{1}{81}\begin{bmatrix}\ &\ &\ &\ &\ &\ &\ \\ &1&\boxed{2}& & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{bmatrix}
$$

3x3 **Box** 3x3 **Box** **Output**

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9}\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \frac{1}{81}\begin{bmatrix}&&&&&&\\&1&2&\boxed{3}&&&\\&&&&&&\\&&&&&&\\&&&&&&\\&&&&&&\\&&&&&&\end{bmatrix}$$

3x3 **Box**   3x3 **Box**   **Output**

Example: Two Box Filters

Treat one filter as padded "image"

$$\frac{1}{9}\begin{array}{|c|c|c|c|c|c|c|}\hline 0&0&0&0&0&0&0\\\hline 0&0&0&0&0&0&0\\\hline 0&0&1&1&1&0&0\\\hline 0&0&1&1&1&0&0\\\hline 0&0&1&1&1&0&0\\\hline 0&0&0&0&0&0&0\\\hline 0&0&0&0&0&0&0\\\hline\end{array} \otimes \frac{1}{9}\begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&1&1\\\hline 1&1&1\\\hline\end{array} = \frac{1}{81}\begin{array}{|c|c|c|c|c|c|c|}\hline &&&&&&\\\hline &1&2&3&2&1&\\\hline &2&4&\boxed{6}&&&\\\hline &&&&&&\\\hline &&&&&&\\\hline &&&&&&\\\hline &&&&&&\\\hline\end{array}$$

3x3 **Box** ⊗ 3x3 **Box** = **Output**

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9}\begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix} 1&1&1\\ 1&1&1\\ 1&1&1 \end{bmatrix} = \frac{1}{81}\begin{bmatrix} &&&&&& \\ &1&2&3&2&1& \\ &2&4&6&4&2& \\ &3&6&9&6&3& \\ &2&4&6&4&2& \\ &1&2&3&2&1& \\ &&&&&& \end{bmatrix}$$

3x3 **Box** | 3x3 **Box** | **Output**

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9}\begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix} 1&1&1\\ 1&1&1\\ 1&1&1 \end{bmatrix} = \frac{1}{81}\begin{bmatrix} 1&2&3&2&1\\ 2&4&6&4&2\\ 3&6&9&6&3\\ 2&4&6&4&2\\ 1&2&3&2&1 \end{bmatrix}$$

3x3 **Box** | 3x3 **Box** | **Output**

Example: Two Box Filters

```
filter = boxfilter(3)
temp = signal.correlate2d(filter, filter,' full')
signal.correlate2d(filter, temp,' full')
```

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \frac{1}{729}\begin{bmatrix}1&3&6&7&6&3&1\\3&9&18&21&18&9&3\\6&18&36&42&36&18&6\\7&21&42&49&42&21&7\\6&18&36&42&36&18&6\\3&9&18&21&18&9&3\\1&3&6&7&6&3&1\end{bmatrix}$$

3x3 **Box** 3x3 **Box** 3x3 **Box**

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

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

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

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

Pre-Convolving 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 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**.

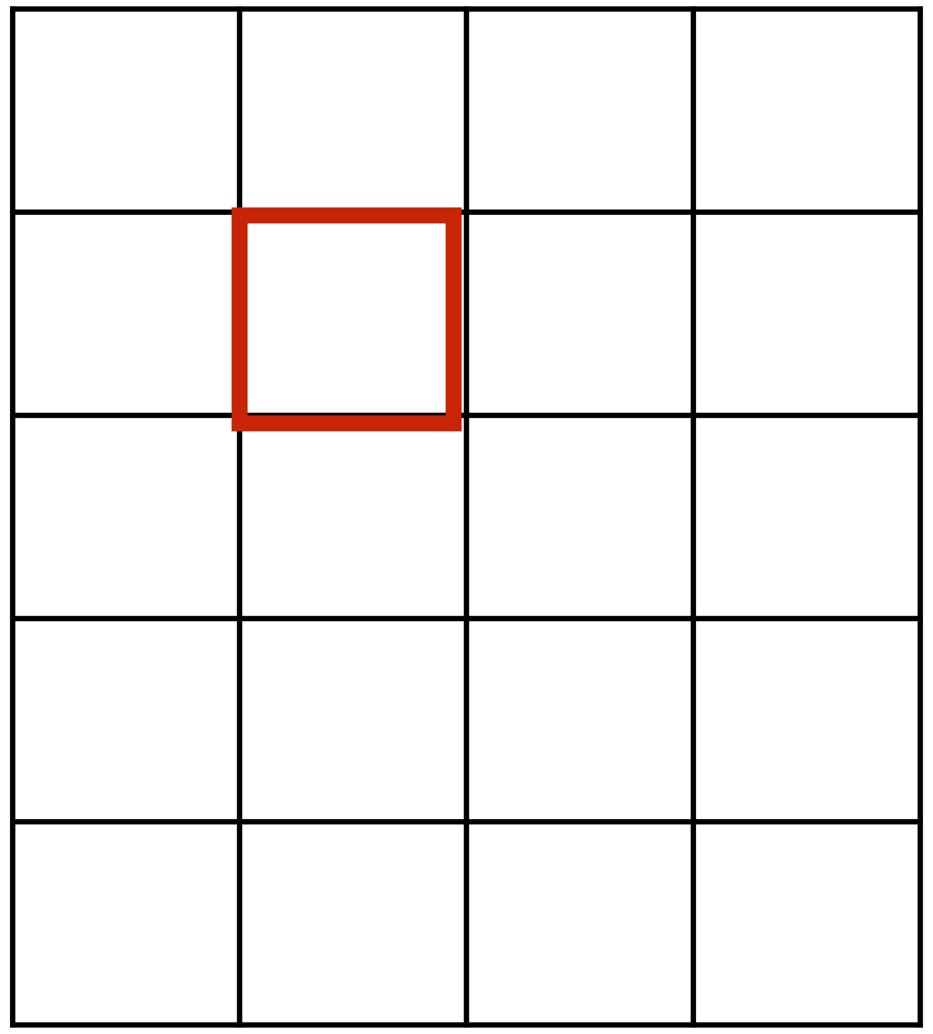
For example, the median filter (which is a very effective de-noising / smoothing filter) selects the **median** value from each pixel’s neighborhood.

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

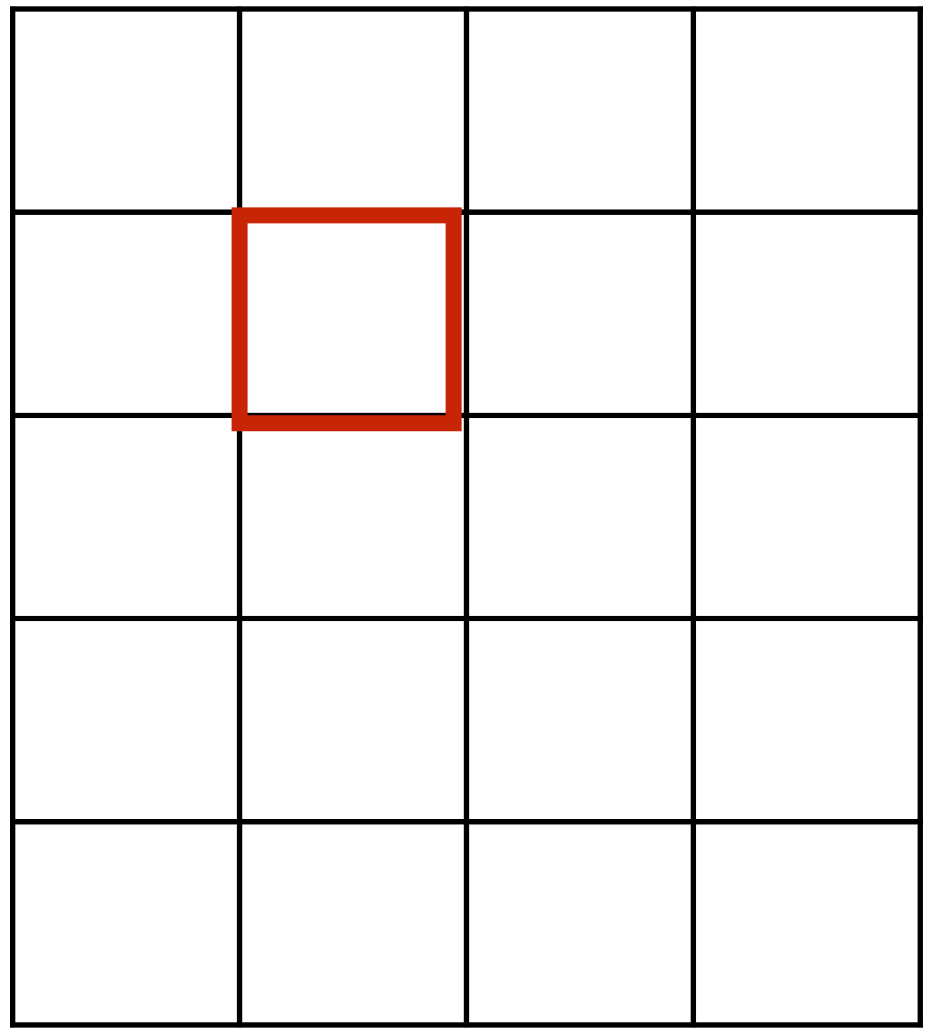
Median Filter

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:

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

$\longrightarrow$

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

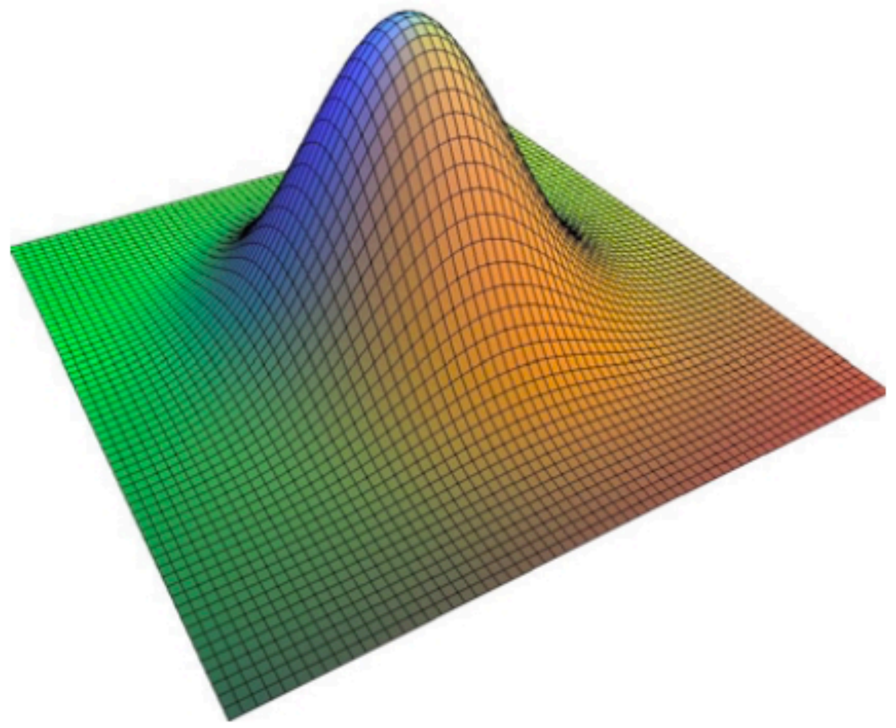
Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

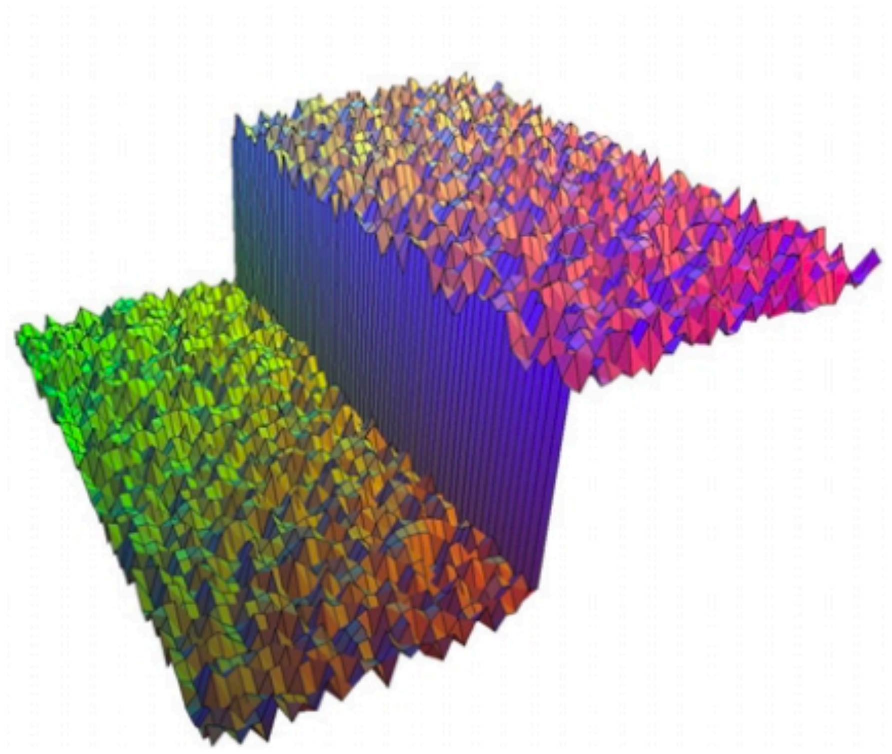
$$\sum \begin{bmatrix} 0.11 & 0.16 & 0.03 \\ 0.16 & 0.26 & 0.01 \\ 0.11 & 0.16 & 0.01 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.9 \\ 0.1 & 0.1 & 1 \\ 0 & 0.1 & 1 \end{bmatrix} = 0.1$$

Bilateral Filter

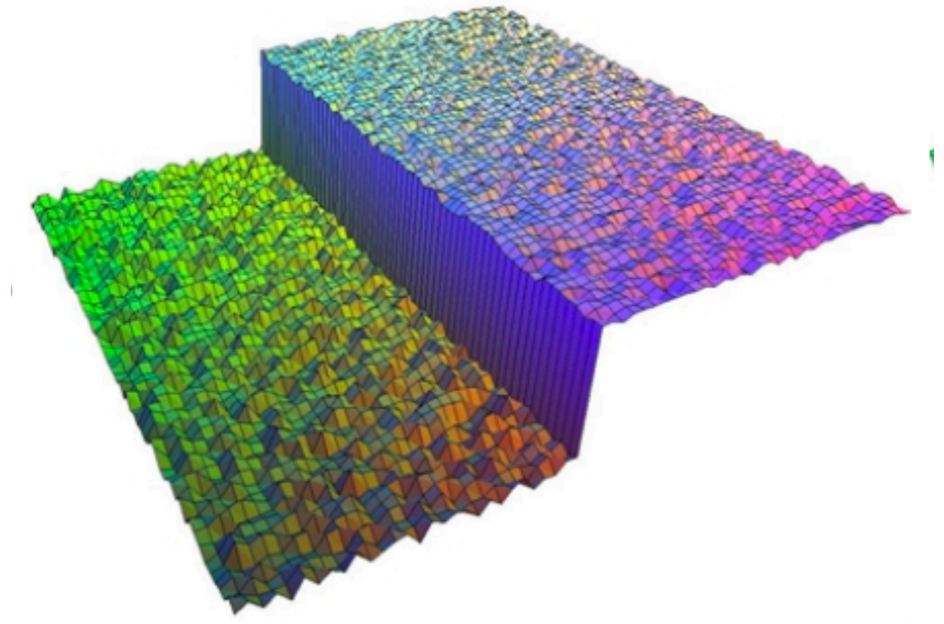
Bilateral Filter



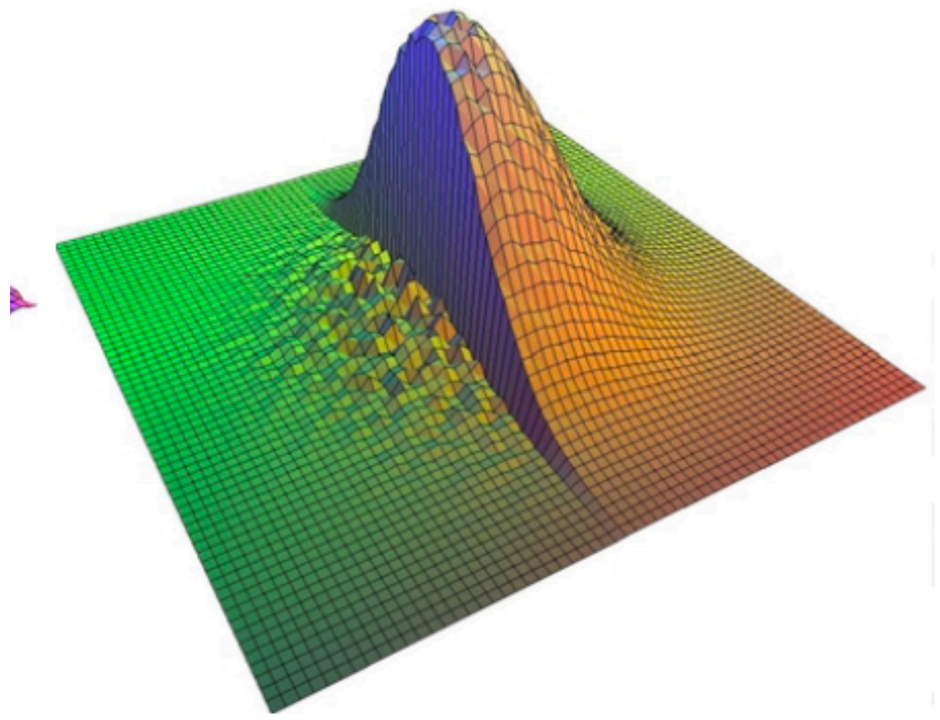
Domain Kernel



Input

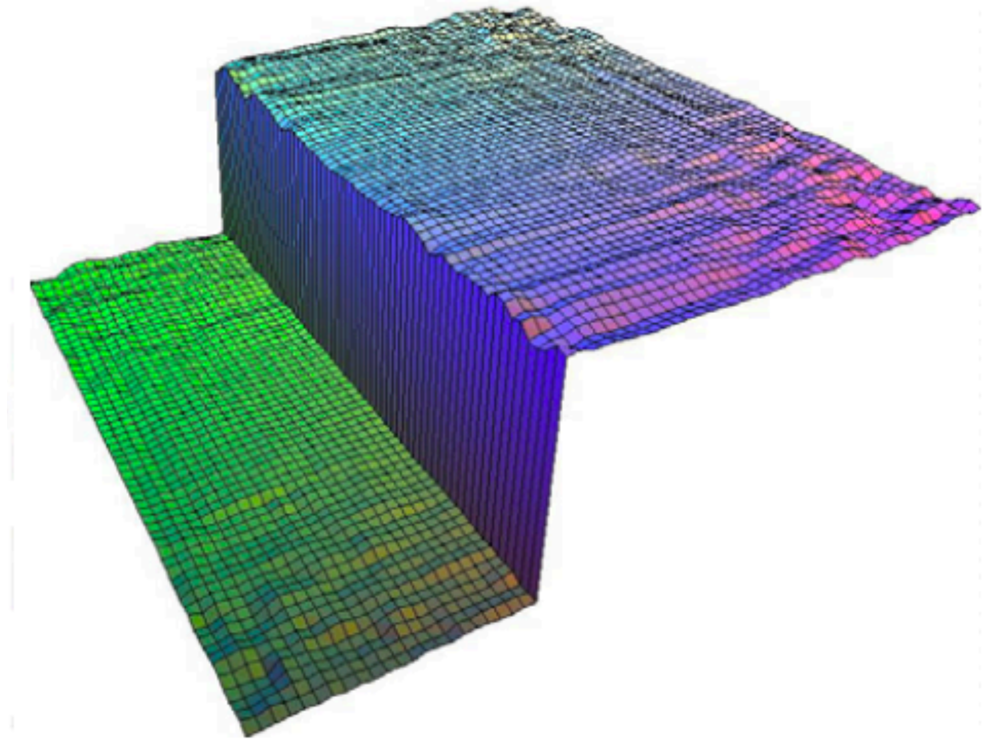


Range Kernel Influence



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

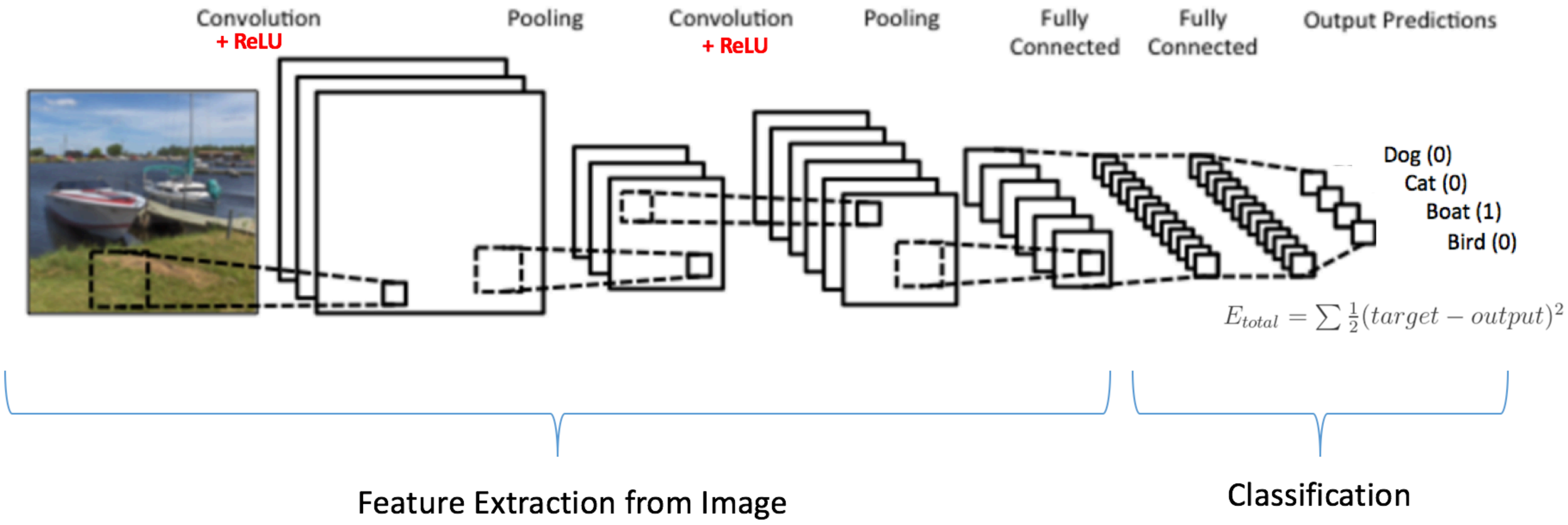
System using ‘joint’ or ‘cross’ bilateral filtering:

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