Lecture 4: Image Filtering (continued)

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little and Fred Tung )
Menu for Today (January 15, 2019)

Topics:

- Gaussian and Pillbox filters
- Separability
- The Convolution Theorem
- Non-linear filters

Readings:

- Today’s Lecture: none
- Next Lecture: [Optional] Forsyth & Ponce (2nd ed.) 4.4

Reminders:

- Assignment 1: Image Filtering and Hybrid Images due January 25th
Today’s “fun” Example: Rolling Shutter
Today’s “fun” Example: Rolling Shutter

Rolling shutter effect
I am in class today:

A) True
B) False
Lecture 3: Re-cap

— The **correlation** of $F(X,Y)$ and $I(X,Y)$ is:

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I,J) I(X+i,Y+j)$$

<table>
<thead>
<tr>
<th>output</th>
<th>filter</th>
<th>image (signal)</th>
</tr>
</thead>
</table>

— **Visual interpretation**: Superimpose the filter $F$ on the image $I$ at $(X,Y)$, perform an element-wise multiply, and sum up the values

— **Convolution** is like **correlation** except filter “flipped”

$F(X,Y) = F(-X,-Y)$ then correlation = convolution.
Ways to handle **boundaries**

- **Ignore/discard.** Make the computation undefined for top/bottom k rows and left/right-most k columns
- **Pad with zeros.** Return zero whenever a value of I is required beyond the image bounds
- **Assume periodicity.** Top row wraps around to the bottom row; leftmost column wraps around to rightmost column.

Simple **examples** of filtering:

- copy, shift, smoothing, sharpening

Linear filter **properties:**

- superposition, scaling, shift invariance

**Characterization Theorem:** Any linear, shift-invariant operation can be expressed as a convolution
Example 6: Smoothing with a Gaussian

Idea: Weight contributions of pixels by spatial proximity (nearness)

2D Gaussian (continuous case):

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \]

Forsyth & Ponce (2nd ed.)

Figure 4.2
Example 6: Smoothing with a Gaussian

Idea: Weight contributions of pixels by spatial proximity (nearness)

2D Gaussian (continuous case):

\[
G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)
\]

Forsyth & Ponce (2nd ed.)
Figure 4.2
**Example 6: Smoothing with a Gaussian**

Quantized an truncated $3x3$ **Gaussian** filter:

<table>
<thead>
<tr>
<th>$G_{\sigma}(-1, 1)$</th>
<th>$G_{\sigma}(0, 1)$</th>
<th>$G_{\sigma}(1, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{\sigma}(-1, 0)$</td>
<td>$G_{\sigma}(0, 0)$</td>
<td>$G_{\sigma}(1, 0)$</td>
</tr>
<tr>
<td>$G_{\sigma}(-1, -1)$</td>
<td>$G_{\sigma}(0, -1)$</td>
<td>$G_{\sigma}(1, -1)$</td>
</tr>
</tbody>
</table>
Example 6: Smoothing with a Gaussian

Quantized an truncated **3x3 Gaussian** filter:

<table>
<thead>
<tr>
<th></th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_\sigma(-1, 1)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(0, 1)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(1, 1)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(-1, 0)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(0, 0)$</td>
<td>$\frac{1}{2\pi\sigma^2}$</td>
</tr>
<tr>
<td>$G_\sigma(1, 0)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(-1, -1)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(0, -1)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(1, -1)$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$</td>
</tr>
</tbody>
</table>
## Example 6: Smoothing with a Gaussian

Quantized and truncated 3x3 Gaussian filter:

<table>
<thead>
<tr>
<th></th>
<th>( G_\sigma(-1, 1) )</th>
<th>( G_\sigma(0, 1) )</th>
<th>( G_\sigma(1, 1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G_\sigma(-1, 0) )</td>
<td>( \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} )</td>
<td>( \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} )</td>
<td>( \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} )</td>
</tr>
<tr>
<td>( G_\sigma(-1, -1) )</td>
<td>( \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} )</td>
<td>( \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} )</td>
<td>( \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} )</td>
</tr>
</tbody>
</table>

With \( \sigma = 1 \):

<table>
<thead>
<tr>
<th></th>
<th>0.059</th>
<th>0.097</th>
<th>0.059</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.097</td>
<td>0.159</td>
<td>0.097</td>
</tr>
<tr>
<td></td>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
</tr>
</tbody>
</table>
Example 6: Smoothing with a Gaussian

Quantized an truncated 3x3 Gaussian filter:

\[
G_\sigma(-1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} \quad G_\sigma(0, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_\sigma(1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}
\]

\[
G_\sigma(-1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_\sigma(0, 0) = \frac{1}{2\pi\sigma^2} \quad G_\sigma(1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}
\]

\[
G_\sigma(-1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} \quad G_\sigma(0, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_\sigma(1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}
\]

With \( \sigma = 1 \):

<table>
<thead>
<tr>
<th></th>
<th>0.059</th>
<th>0.097</th>
<th>0.059</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
<td></td>
</tr>
</tbody>
</table>

What happens if \( \sigma \) is larger?
Example 6: Smoothing with a Gaussian

Quantized and truncated 3x3 Gaussian filter:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_\sigma(-1, 1) =$</td>
<td>$G_\sigma(0, 1) =$</td>
<td>$G_\sigma(1, 1) =$</td>
</tr>
<tr>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2}{2\sigma^2}}$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(-1, 0) =$</td>
<td>$G_\sigma(0, 0) =$</td>
<td>$G_\sigma(1, 0) =$</td>
</tr>
<tr>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2}{2\sigma^2}}$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
</tr>
<tr>
<td>$G_\sigma(-1, -1) =$</td>
<td>$G_\sigma(0, -1) =$</td>
<td>$G_\sigma(1, -1) =$</td>
</tr>
<tr>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2}{2\sigma^2}}$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$</td>
<td>$\frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$</td>
</tr>
</tbody>
</table>

With $\sigma = 1$:

```
↑ ↑ ↑
↑ ↑ ↓
↑ ↓ ↑
↑ ↑ ↑
```

What happens if $\sigma$ is larger?

— More blur
Example 6: Smoothing with a Gaussian

Quantized and truncated 3x3 Gaussian filter:

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

$G_\sigma(0, 0) = \frac{1}{2\pi\sigma^2}$

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

$G_\sigma(-1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$

$G_\sigma(0, 0) = \frac{1}{2\pi\sigma^2}$

$G_\sigma(1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$

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

$G_\sigma(0, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$

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

With $\sigma = 1$:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
</tr>
<tr>
<td>0.097</td>
<td>0.159</td>
<td>0.097</td>
</tr>
<tr>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
</tr>
</tbody>
</table>

What happens if $\sigma$ is larger?

What happens if $\sigma$ is smaller?
Example 6: Smoothing with a Gaussian

Quantized and truncated 3x3 Gaussian filter:

\[
G_\sigma(-1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} \quad G_\sigma(0, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_\sigma(1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}
\]

\[
G_\sigma(-1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_\sigma(0, 0) = \frac{1}{2\pi\sigma^2} \quad G_\sigma(1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}
\]

\[
G_\sigma(-1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} \quad G_\sigma(0, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_\sigma(1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}
\]

With \( \sigma = 1 \):

\begin{tabular}{ccc}
\downarrow & \downarrow & \downarrow \\
\downarrow & \uparrow & \downarrow \\
\downarrow & \downarrow & \downarrow \\
\end{tabular}

What happens if \( \sigma \) is larger?

What happens if \( \sigma \) is smaller?

— Less blur
Example 6: Smoothing with a Gaussian

Forsyth & Ponce (2nd ed.) Figure 4.1 (left and right)
Box vs. Gaussian Filter

original

7x7 Gaussian

7x7 box

Slide Credit: Ioannis (Yannis) Gkioulkas (CMU)
Fun: How to get shadow effect?

University of British Columbia

Adopted from: Ioannis (Yannis) Gkioulekas (CMU)
Fun: How to get shadow effect?

University of British Columbia

Blur with a Gaussian kernel, then compose the blurred image with the original (with some offset)

Adopted from: Ioannis (Yannis) Gkioulekas (CMU)
Example 6: Smoothing with a Gaussian

Quantized an truncated 3x3 Gaussian filter:

<table>
<thead>
<tr>
<th>x</th>
<th>Gσ(-1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2}{2\sigma^2}}</th>
<th>Gσ(0, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}</th>
<th>Gσ(1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1, 0</td>
<td>Gσ(-1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}</td>
<td>Gσ(0, 0) = \frac{1}{2\pi\sigma^2}</td>
<td>Gσ(1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}</td>
</tr>
<tr>
<td>-1, -1</td>
<td>Gσ(-1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}</td>
<td>Gσ(0, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}</td>
<td>Gσ(1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}</td>
</tr>
</tbody>
</table>

With σ = 1:

<table>
<thead>
<tr>
<th>Gσ(-1, 1)</th>
<th>Gσ(0, 1)</th>
<th>Gσ(1, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
</tr>
<tr>
<td>0.097</td>
<td>0.159</td>
<td>0.097</td>
</tr>
<tr>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
</tr>
</tbody>
</table>

What is the problem with this filter?
Example 6: Smoothing with a Gaussian

Quantized an truncated 3x3 Gaussian filter:

\[
G_{\sigma}(-1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} \quad G_{\sigma}(0, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_{\sigma}(1, 1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}
\]

\[
G_{\sigma}(-1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_{\sigma}(0, 0) = \frac{1}{2\pi\sigma^2} \quad G_{\sigma}(1, 0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}
\]

\[
G_{\sigma}(-1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}} \quad G_{\sigma}(0, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}} \quad G_{\sigma}(1, -1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}
\]

With \(\sigma = 1\):

\[
\begin{array}{ccc}
0.059 & 0.097 & 0.059 \\
0.097 & 0.159 & 0.097 \\
0.059 & 0.097 & 0.059
\end{array}
\]

What is the problem with this filter?

- does not sum to 1
- truncated too much
Gaussian: Area Under the Curve

- 68% between $-1 \sigma$ and $1 \sigma$
- 95% between $-2 \sigma$ and $2 \sigma$
- 99.7% between $-3 \sigma$ and $3 \sigma$
- 99.99% between $-4 \sigma$ and $4 \sigma$
Example 6: Smoothing with a Gaussian

With $\sigma = 1$:

<table>
<thead>
<tr>
<th></th>
<th>0.059</th>
<th>0.097</th>
<th>0.059</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.097</td>
<td>0.159</td>
<td>0.097</td>
<td></td>
</tr>
<tr>
<td>0.059</td>
<td>0.097</td>
<td>0.059</td>
<td></td>
</tr>
</tbody>
</table>

Better version of the Gaussian filter:

- sums to 1 (normalized)
- captures $\pm 2\sigma$

In general, you want the Gaussian filter to capture $\pm 3\sigma$, for $\sigma = 1 \Rightarrow 7 \times 7$ filter
Efficient Implementation: **Separability**

A 2D function of $x$ and $y$ is **separable** if it can be written as the product of two functions, one a function only of $x$ and the other a function only of $y$.

Both the 2D box filter and the 2D Gaussian filter are separable.

Both can be implemented as two 1D convolutions:
- First, convolve each row with a 1D filter
- Then, convolve each column with a 1D filter
- Aside: or vice versa

The **2D Gaussian** is the only (non trivial) 2D function that is both separable and rotationally invariant.
**Separability**: Box Filter Example

**Standard (3x3)**

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
I(X, Y) = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

\[
F(X, Y) = F(X)F(Y)
\]

**Separable**

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
F(X) = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

\[
F(Y) = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \end{bmatrix}
\]

**Output**

\[
\begin{array}{cccccccccc}
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20 \\
0 & 30 & 50 & 80 & 80 & 80 & 60 & 30 \\
0 & 30 & 50 & 80 & 80 & 80 & 60 & 30 \\
0 & 20 & 30 & 50 & 60 & 60 & 40 & 20 \\
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\
10 & 30 & 10 & 10 & 10 & 10 & 10 & 10 \\
\end{array}
\]

\[
I'(X, Y) = F(X, Y)I(X, Y)
\]
Efficient Implementation: **Separability**

For example, recall the 2D **Gaussian**:

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \]

The 2D Gaussian can be expressed as a product of two functions, one a function of \( x \) and another a function of \( y \)
Efficient Implementation: **Separability**

For example, recall the 2D **Gaussian**:

\[
G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

\[
= \left(\frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \left(\frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right)
\]

The 2D Gaussian can be expressed as a product of two functions, one a function of x and another a function of y.
Efficient Implementation: **Separability**

For example, recall the 2D **Gaussian**:

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \]

\[ = \left( \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \right) \left( \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right) \right) \]

function of x  function of y

The 2D Gaussian can be expressed as a product of two functions, one a function of x and another a function of y.

In this case the two functions are (identical) 1D Gaussians.
Efficient Implementation: Separability

Naive implementation of 2D Gaussian:

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.
Efficient Implementation: **Separability**

Naive implementation of 2D **Gaussian**:

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.

Separable 2D **Gaussian**:
Efficient Implementation: **Separability**

**Naive implementation of 2D Gaussian:**

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

**Separable 2D Gaussian:**

At each pixel, \((X, Y)\), there are \(2m\) multiplications

There are \(n \times n\) pixels in \((X, Y)\)

**Total:** \(2m \times n^2\) multiplications
Example 7: Smoothing with a Pillbox

Let the radius (i.e., half diameter) of the filter be \( r \)

In a contentious domain, a 2D (circular) pillbox filter, \( f(x, y) \), is defined as:

\[
f(x, y) = \frac{1}{\pi r^2} \begin{cases} 
1 & \text{if } x^2 + y^2 \leq r^2 \\ 
0 & \text{otherwise}
\end{cases}
\]

The scaling constant, \( \frac{1}{\pi r^2} \), ensures that the area of the filter is one
Example 7: Smoothing with a Pillbox

Recall that the 2D Gaussian is the only (non trivial) 2D function that is both separable and rotationally invariant.

A 2D pillbox is rotationally invariant but not separable.

There are occasions when we want to convolve an image with a 2D pillbox. Thus, it worth exploring possibilities for efficient implementation.
Example 7: Smoothing with a Pillbox

A 2D box filter can be expressed as the sum of a 2D pillbox and some “extra corner bits”
Example 7: Smoothing with a Pillbox

Therefore, a 2D pillbox filter can be expressed as the difference of a 2D box filter and those same “extra corner bits”
Example 7: Smoothing with a Pillbox

Implementing convolution with a 2D pillbox filter as the difference between convolution with a box filter and convolution with the “extra corner bits” filter allows us to take advantage of the separability of a box filter.

Further, we can postpone scaling the output to a single, final step so that convolution involves filters containing all 0’s and 1’s. — This means the required convolutions can be implemented without any multiplication at all.
Example 7: Smoothing with a Pillbox
Let $z$ be the product of two numbers, $x$ and $y$, that is,

$$z = xy$$
Let $z$ be the product of two numbers, $x$ and $y$, that is,

$$z = xy$$

Taking logarithms of both sides, one obtains

$$\ln z = \ln x + \ln y$$
Let $z$ be the product of two numbers, $x$ and $y$, that is,

$$z = xy$$

Taking logarithms of both sides, one obtains

$$\ln z = \ln x + \ln y$$

Therefore.

$$z = \exp^{\ln z} = \exp^{(\ln x + \ln y)}$$
Speeding Up **Convolution** (The Convolution Theorem)

Let \( z \) be the product of two numbers, \( x \) and \( y \), that is,

\[
z = xy
\]

Taking logarithms of both sides, one obtains

\[
\ln z = \ln x + \ln y
\]

Therefore,

\[
z = \exp^{\ln z} = \exp^{(\ln x + \ln y)}
\]

**Interpretation:** At the expense of two \( \ln() \) and one \( \exp() \) computations, multiplication is reduced to admission.
Speeding Up Rotation

Another analogy: **2D rotation of a point by an angle** $\alpha$ about the origin

The standard approach, in Euclidean coordinates, involves a matrix multiplication

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  \cos \alpha & -\sin \alpha \\
  \sin \alpha & \cos \alpha
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

Suppose we transform to polar coordinates

\[(x, y) \rightarrow (\rho, \theta) \rightarrow (\rho, \theta + \alpha) \rightarrow (x', y')\]

Rotation becomes addition, at expense of one polar coordinate transform and one inverse polar coordinate transform
Speeding Up **Convolution** (The Convolution Theorem)

Similarly, some image processing operations become cheaper in a transform domain.

Gonzales & Woods (3rd ed.) Figure 2.39
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
What follows is for fun
(you will NOT be tested on this)
Fourier Transform (you will NOT be tested on this)

Basic building block:

$$A \sin(\omega x + \phi)$$

Fourier’s claim: Add enough of these to get any periodic signal you want!
Fourier Transform (you will NOT be tested on this)

Basic building block:

\[ A \sin(\omega x + \phi) \]

Fourier’s claim: Add enough of these to get any periodic signal you want!
Fourier Transform (you will NOT be tested on this)

How would you generate this function?

= ? + ?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Fourier Transform (you will NOT be tested on this)**

How would you generate this function?

\[
\text{sin}(2\pi x) = \text{?}
\]
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

\[
\begin{align*}
\text{function} & = \sin(2\pi x) + \frac{1}{3} \sin(2\pi 3x)
\end{align*}
\]
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

\[ f(x) = \sin(2\pi x) + \frac{1}{3} \sin(2\pi 3x) \]

\[ = \sin(2\pi x) + \frac{1}{3} \sin(2\pi 3x) \]

**Slide Credit**: Ioannis (Yannis) Gkioulekas (CMU)
Fourier Transform (you will NOT be tested on this)

How would you generate this function?

\[
\begin{align*}
\text{square wave} & \quad \approx \quad \text{?} \quad + \quad \text{?}
\end{align*}
\]
**Fourier Transform** (you will **NOT** be tested on this)

How would you generate this function?

- Square wave

\[
\approx \quad +
\]

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
How would you generate this function?

\[
\text{square wave} \approx \sum \text{Fourier Transform (you will NOT be tested on this)}
\]

Slide Credit: Ioannis (Yannis) Gkioulakes (CMU)
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Fourier Transform** (you will **NOT** be tested on this)

How would you generate this function?

square wave

How would you express this mathematically?

*Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)*
Fourier Transform (you will NOT be tested on this)

How would you generate this function?

\[
\text{square wave} \quad \Rightarrow \quad \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi k x)
\]

infinite sum of sine waves

Slide Credit: Ioannis (Yannis) Gkioulkas (CMU)
Fourier Transform (you will NOT be tested on this)

Basic building block:

\[ A \sin(\omega x + \phi) \]

Fourier’s claim: Add enough of these to get any periodic signal you want!
Fourier Transform (you will NOT be tested on this)

Image from: Numerical Simulation and Fractal Analysis of Mesoscopic Scale Failure in Shale Using Digital Images
Fourier Transform (you will **NOT** be tested on this)

Forsyth & Ponce (2nd ed.) Figure 4.6
Fourier Transform (you will NOT be tested on this)

Forsyth & Ponce (2nd ed.) Figure 4.6
What preceded was for fun (you will NOT be tested on it)
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: \(\mathcal{O}(n^2)\)
Linear Filters: Properties (recall Lecture 3)

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,

\[
(G \otimes F) \otimes I(X, Y) = (G \otimes F) \otimes I(X, Y)
\]

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

```python
filter = boxfilter(3)
signal.correlate2d(filter, filter, 'full')
```
Example: Two Box Filters

Treat one filter as padded “image”

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

3x3 Box

Output

\[
\frac{1}{9} \times \frac{1}{9} = \frac{1}{81}
\]
Example: Two Box Filters

Treat one filter as padded “image”

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\times\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array} = \begin{array}{cc}
1 & 2 \\
\end{array}
\]

3x3 Box

Output
Example: Two Box Filters

Treat one filter as padded “image”

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[
\frac{1}{9} \times \frac{1}{9} = \frac{1}{81}
\]

Output
**Example:** Two Box Filters

Treat one filter as padded “image”

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\times
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 2 & 3 & 2 & 1 \\
2 & 4 & 6 & & \\
\end{bmatrix}
\]
Example: Two Box Filters

Treat one filter as padded “image”

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\times
\begin{array}{cccc}
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 \\
\end{array}
\times
\begin{array}{cccccccc}
1 & 2 & 3 & 2 & 1 \\
2 & 4 & 6 & 4 & 2 \\
3 & 6 & 9 & 6 & 3 \\
2 & 4 & 6 & 4 & 2 \\
1 & 2 & 3 & 2 & 1 \\
\end{array}
= \frac{1}{81}
\]
Example: Two Box Filters

Treat one filter as padded “image”

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix} \times \frac{1}{9} = \frac{1}{81}
\]

Output
Example: Two Box Filters

```
filter = boxfilter(3)
temp = signal.correlate2d(filter, filter, 'full')
signal.correlate2d(filter, temp, 'full')
```
Example: Separable Gaussian Filter

\[ \frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \times \frac{1}{16} \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \]
Example: Separable Gaussian Filter

\[
\begin{array}{cccccc}
1/16 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\quad \times \quad
\begin{array}{cccc}
1/16 & 1 & 4 & 6 & 4 & 1 \\
1 & 4 & 6 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\quad = \quad
\begin{array}{cccccc}
1/256 & 1 & 4 & 6 & 4 & 1 \\
1 & 4 & 6 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Example: Separable Gaussian Filter

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccccc}
1 & 4 & 6 & 4 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\frac{1}{16} \quad \otimes \quad \frac{1}{16} = \frac{1}{256}
\]

\[
\begin{array}{cccccc}
1 & 4 & 6 & 4 & 1 \\
4 & 16 & & & & \\
6 & & & & & \\
4 & & & & & \\
1 & & & & & \\
\end{array}
\]
Example: Separable Gaussian Filter

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\hline
\frac{1}{16} & 1 & 4 & 6 \\
4 & 6 & 4 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\times
\begin{array}{c}
1 \\
4 \\
6 \\
4 \\
\hline
16 \\
16 \\
24 \\
24 \\
4 \\
4 \\
1 \\
1 \\
4 \\
6 \\
4 \\
1 \\
\end{array}
= \frac{1}{256}
\begin{array}{cccc}
1 & 4 & 6 & 4 \\
4 & 16 & 24 & 16 \\
6 & 24 & 36 & 24 \\
4 & 16 & 24 & 16 \\
1 & 4 & 6 & 4 \\
1 & 4 & 6 & 4 \\
\end{array}
\]
**Example:** Separable Gaussian Filter

\[
\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 4 & 6 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\frac{1}{16}
\end{array} \times \begin{array}{c}
1 \\
4 \\
6 \\
4 \\
1 \\
\hline
\frac{1}{16}
\end{array} = \frac{1}{256}
\begin{array}{cccccc}
1 & 4 & 6 & 4 & 1 \\
4 & 16 & 24 & 16 & 4 \\
6 & 24 & 36 & 24 & 6 \\
4 & 16 & 24 & 16 & 4 \\
1 & 4 & 6 & 4 & 1 \\
\hline
\end{array}
\]
Pre-Convolving Filters

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

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

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 \( G_{\sqrt{2\sigma}}(x) \)
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