Lecture 4: Image Filtering (continued)

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little and Fred Tung )
Menu for Today (September 13, 2018)

Topics:

- Linear filters
- Linear filter properties
- Correlation / Convolution
- Filter examples: Box, Gaussian, …

Readings:

- Today’s Lecture: Forsyth & Ponce (2nd ed.) 4.1, 4.5
- Next Lecture: none

Reminders:

- Assignment 1: Image Filtering and Hybrid Images is out
- Conveniently, office hours of TAs who are responsible for this assignment are on Thursday (Siddhesh 11-noon) and Friday (Borna 9-10am)
Today’s “fun” Example:
We take a “physics-based” approach to image formation
— Treat camera as an instrument that takes measurements of the 3D world

Basic abstraction is the **pinhole camera**

**Lenses** overcome limitations of the pinhole model while trying to preserve it as a useful abstraction

When **maximum accuracy** required, it is necessary to model additional details of each particular camera (and camera setting)
— Aside: This is called camera calibration
Lecture 3a: Re-cap Snell’s Law

\[ n_1 \sin \alpha_1 = n_2 \sin \alpha_2 \]
Lecture 3a: Re-cap Lenses

Thin lens equation

\[
\frac{1}{z'} - \frac{1}{z} = \frac{1}{f}
\]

characterizes the relationship between \(f\), \(z\) and \(z'\)

Some “aberrations and distortions” persist. For example:
— index of refraction depends on wavelength, \(\lambda\), of light
— vignetting reduces image brightness (gradually) away from the image center

The human eye functions much like a camera
Lecture 3b: Introduction to Filterings

**Point** Operation

**Neighborhood** Operation

point processing

"filtering"

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Let $I(X, Y)$ be an $n \times n$ digital image (for convenience we let width = height)

Let $F(X, Y)$ be another $m \times m$ digital image (our “filter” or “kernel”)

For convenience we will assume $m$ is odd. (Here, $m = 5$)
Linear Filters

Let $k = \left\lfloor \frac{m}{2} \right\rfloor$

Compute a new image, $I'(X, Y)$, as follows

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

**Intuition:** each pixel in the output image is a linear combination of the same pixel and its neighboring pixels in the original image
Linear Filters

For a given $X$ and $Y$, superimpose the filter on the image centered at $(X, Y)$.
Linear Filters

For a given $X$ and $Y$, superimpose the filter on the image centered at $(X, Y)$

Compute the new pixel value, $I'(X, Y)$, as the sum of $m \times m$ values, where each value is the product of the original pixel value in $I(X, Y)$ and the corresponding values in the filter
The computation is repeated for each 

\((X, Y)\)
Linear Filter Example

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

*Image* \[ F(X, Y) \]

*Filter* \[ \frac{1}{9} \]

*Output* \[ I'(X, Y) \]

*Image (signal)*

*Filter*

*Output*

*Slide Credit*: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[
\begin{align*}
F(X, Y) &= \frac{1}{9} \\
I(X, Y) &= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \\
I_0(X, Y) &= kX_j = kX_i = k \\
F'(I, J) &= I(X + i, Y + j) \\
I'(X, Y) &= \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(I, J) I(X + i, Y + j)
\end{align*}
\]

Slide Credit: Ioannis (Yannis) Gkioulakas (CMU)
Linear Filter Example

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

\[
\frac{1}{9}
\]

\[
I(X, Y) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

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

Output

\[
I'(X, Y) = \begin{bmatrix}
0 & \text{output} \\
\end{bmatrix}
\]

Filter

Image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

\[
I(X, Y) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

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

Output

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

\( F(X,Y) \)

Filter

\[ \frac{1}{9} \]

image

\[ I(X,Y) \]

output

\[ I'(X,Y) \]

Filter image (signal)
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

**Image**

\[ I(X, Y) \]

**Filter**

\[ F(X, Y) \]

**Output**

\[ I'(X, Y) \]

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ I(X, Y) \]

\[ F(X, Y) \]

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

Output

Image (signal)

Filter

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Linear Filter Example**

An example of linear filtering is shown below. The input image $I(X, Y)$ is convolved with a filter $F(X, Y)$ to produce the output image $I'(X, Y)$.

The filter $F(X, Y)$ is a 3x3 kernel in this example, with weights given by the values in the kernel.

The output image is computed as:

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

Where $F(I, J)$ represents the filter values and $I(X + i, Y + j)$ are the image pixels at the shifted positions.

**Slide Credit:** Ioannis (Yannis) Gkioulkas (CMU)
Linear Filter Example

$I(X, Y) = I'(X, Y)$

$F(X, Y)$

Filter

$\frac{1}{9}$

$1 \times 1$ filter

$I_0(X, Y)$ = k

Filter image (signal)

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

Output

Image

Output

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[
F(X, Y)
\]

\[
\frac{1}{9}
\]

\[
I(X, Y)
\]

\[
I'(X, Y)
\]

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Linear Filter Example**

The formula for the output of a linear filter is given by:

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

where $F(X, Y)$ represents the filter, $I(X, Y)$ is the input image, and $\text{I'}(X, Y)$ is the output image.

*Slide Credit: Ioannis (Yannis) Gkioulkekas (CMU)*
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
**Linear Filter Example**

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

**Slide Credit:** Ioannis (Yannis) Gkioulekas (CMU)
Linear Filter Example

\[ F(X, Y) \]

\[ I(X, Y) \]

\[ I'(X, Y) = \sum_{i=\pm k} \sum_{j=\pm k} F(I, J) I(X + i, Y + j) \]

Output

Image (signal)

Filter

I0(X, Y) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
0 & 0 & 90 & 90 & 90 \\
\end{bmatrix}

Output

\[ I'(X, Y) \]

\begin{bmatrix}
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 & 50 & 50 & 40 & 20 \\
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 \\
10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 \\
\end{bmatrix}

Slide Credit: Ioannis (Yannis) Gkioulakas (CMU)
Linear Filter Example

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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filters

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

For a given \(X\) and \(Y\), superimpose the filter on the image centered at \((X, Y)\)

Compute the new pixel value, \(I'(X, Y)\), as the sum of \(m \times m\) values, where each value is the product of the original pixel value in \(I(X, Y)\) and the corresponding values in the filter
Linear Filters

Let’s do some accounting ...

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

output

filter

image (signal)

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

When \(m\) is fixed, small constant, this is \(O(n^2)\). But when \(m \approx n\) this is \(O(m^4)\).
Linear Filters: **Boundary Effects**
**Linear Filters: Boundary Effects**

Three standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom $k$ rows and the leftmost and rightmost $k$ columns.
Linear Filters: **Boundary Effects**

Three standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom $k$ rows and the leftmost and rightmost $k$ columns.

2. **Pad the image with zeros:** Return zero whenever a value of $I$ is required at some position outside the defined limits of $X$ and $Y$. 
Linear Filters: **Boundary Effects**
Linear Filters: **Boundary Effects**

Three standard ways to deal with boundaries:

1. **Ignore these locations**: Make the computation undefined for the top and bottom $k$ rows and the leftmost and rightmost $k$ columns.

2. **Pad the image with zeros**: Return zero whenever a value of $I$ is required at some position outside the defined limits of $X$ and $Y$.

3. **Assume periodicity**: The top row wraps around to the bottom row; the leftmost column wraps around to the rightmost column.
Linear Filters: **Boundary Effects**
Linear Filters: **Boundary Effects**
A short exercise ...
Example 1: Warm up

Original

Filter

Result
Example 1: Warm up

Original

Filter

Result
(no change)
Example 2:

Original

Filter

Result

?
Example 2:

Original

Filter

Result

(sift left by 1 pixel)
Example 3:

Original

Filter

Result

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

(filter sums to 1)
Example 3:

Original

Filter
(filter sums to 1)

Result
(blur with a box filter)
Example 4:

Original

Filter
(filter sums to 1)

Result
Example 4:

Original

Filter
(filter sums to 1)

Result
(sharpening)
Example 4: Sharpening

Before

After
Example 4: Sharpening

Before

After

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

\[
I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j) I(X + i, Y + j)
\]
Linear Filters: Correlation vs. Convolution

Definition: Correlation

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

Definition: Convolution

$$I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X - i, Y - j)$$
**Linear Filters**: Correlation vs. Convolution

**Definition: Correlation**

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

\[
= 1a + 2b + 3c + 4d + 5e + 6f + 7g + 8h + 9i
\]
Definition: **Correlation**

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

Definition: **Convolution**

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X - i, Y - j) \]

Filter:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
<td></td>
</tr>
</tbody>
</table>

Image:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Output:

= 9a + 8b + 7c + 6d + 5e + 4f + 3g + 2h + 1i
Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

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

Definition: **Convolution**

\[ I'(X, Y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} F(i, j)I(X - i, Y - j) \]

**Filter** (rotated by 180)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>d</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>h</td>
<td>i</td>
<td>a</td>
</tr>
</tbody>
</table>

**Image**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**Output**

\[ = 9a + 8b + 7c + 6d + 5e + 4f + 3g + 2h + 1i \]
Linear Filters: Correlation vs. Convolution

Definition: Correlation

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

Definition: Convolution

\[ I'(X, Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i, j)I(X - i, Y - j) \]

\[ = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(-i, -j)I(X + i, Y + j) \]

Note: if \( F(X, Y) = F(-X, -Y) \) then correlation = convolution.
Why convolutions are important?

Who has heard of **Convolutional Neural Networks** (CNNs)?

What about **Deep Learning**?

Basic operations in CNNs are convolutions (with learned linear filters) followed by non-linear functions.

**Note:** This results in non-linear filters.
Linear Filters: **Properties**

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

**Superposition:** Let $F_1$ and $F_2$ be digital filters

$$(F_1 + F_2) \otimes I(X, Y) = F_1 \otimes I(X, Y) + F_2 \otimes I(X, Y)$$
Linear Filters: **Properties**

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 a digital filter and let \( k \) be a scalar

\[
(kF) \otimes I(X, Y) = F \otimes (kI(X, Y)) = k(F \otimes I(X, Y))
\]
Linear Filters: Properties

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 a 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)
Linear Filters: Shift Invariance

Output does **not** depend on absolute position
Linear Filters: **Properties**

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 Systems: Characterization Theorem

Any linear, shift invariant operation can be expressed as convolution
Example 5: Smoothing with a Box Filter

Filter has equal positive values that sum up to 1

Replaces each pixel with the average of itself and its local neighborhood

— Box filter is also referred to as *average filter* or *mean filter*
Example 5: Smoothing with a Box Filter

Forsyth & Ponce (2nd ed.) Figure 4.1 (left and middle)
Example 5: Smoothing with a Box Filter

What happens if we increase the width (size) of the box filter?
Example 5: Smoothing with a Box Filter

Gonzales & Woods (3rd ed.) Figure 3.3
Example 6: Smoothing with a Gaussian

Smoothing with a box doesn’t model lens defocus well

— Smoothing with a box filter depends on direction
— Image in which the center point is 1 and every other point is 0
Example 6: Smoothing with a Gaussian

Smoothing with a box doesn’t model lens defocus well
— Smoothing with a box filter depends on direction
— Image in which the center point is 1 and every other point is 0
Example 6: Smoothing with a Gaussian

Smoothing with a box doesn’t model lens defocus well
— Smoothing with a box filter depends on direction
— Image in which the center point is 1 and every other point is 0

<table>
<thead>
<tr>
<th>Filter</th>
<th>Image</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="1,1,1,1" alt="Filter" /></td>
<td><img src="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" alt="Image" /></td>
<td><img src="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" alt="Result" /></td>
</tr>
</tbody>
</table>

\[
\frac{1}{9} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]
Example 6: Smoothing with a Gaussian

Smoothing with a box *doesn’t model lens defocus* well
- Smoothing with a box filter depends on direction
- Image in which the center point is 1 and every other point is 0

Smoothing with a (circular) pillbox is a better model for defocus (in geometric optics)

The **Gaussian** is a good general smoothing model
- for phenomena (that are the sum of other small effects)
- whenever the Central Limit Theorem applies
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
Summary

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

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

— **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”

$$\text{if } F(X,Y) = F(-X, -Y) \text{ then correlation} = \text{convolution.}$$

— **Characterization Theorem**: Any linear, spatially invariant operation can be expressed as a convolution