Filtering and Pyramids

CSE P576

Dr. Matthew Brown
Filtering and Pyramids

- Linear filtering (convolution, correlation)
  - Blurring, sharpening, edge detection
- Gaussian and Laplacian Pyramids
  - Multi-scale representations
Linear Operators

• How are photo filters implemented?

original image

blur  sharpen  edge filter
Non-Linear Operators

- How are photo filters implemented?

![Original image](image1)

- edge preserve
- smooth
- median
- canny edges
Correlation Example

<table>
<thead>
<tr>
<th>45</th>
<th>60</th>
<th>98</th>
<th>127</th>
<th>132</th>
<th>133</th>
<th>137</th>
<th>133</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>65</td>
<td>98</td>
<td>123</td>
<td>126</td>
<td>128</td>
<td>131</td>
<td>133</td>
</tr>
<tr>
<td>47</td>
<td>65</td>
<td>96</td>
<td>115</td>
<td>119</td>
<td>123</td>
<td>135</td>
<td>137</td>
</tr>
<tr>
<td>47</td>
<td>63</td>
<td>91</td>
<td>107</td>
<td>113</td>
<td>122</td>
<td>138</td>
<td>134</td>
</tr>
<tr>
<td>50</td>
<td>59</td>
<td>80</td>
<td>97</td>
<td>110</td>
<td>123</td>
<td>133</td>
<td>134</td>
</tr>
<tr>
<td>49</td>
<td>53</td>
<td>68</td>
<td>83</td>
<td>97</td>
<td>113</td>
<td>128</td>
<td>133</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>58</td>
<td>70</td>
<td>84</td>
<td>102</td>
<td>116</td>
<td>126</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>52</td>
<td>58</td>
<td>69</td>
<td>86</td>
<td>101</td>
<td>120</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
45 & 60 & 98 \\
46 & 65 & 98 \\
47 & 63 & 91 \\
50 & 59 & 80 \\
49 & 53 & 68 \\
50 & 50 & 52 \\
\end{array}
\begin{array}{ccc}
127 & 132 & 133 \\
126 & 128 & 131 \\
115 & 119 & 123 \\
113 & 122 & 138 \\
97 & 110 & 123 \\
70 & 84 & 102 \\
\end{array}
\begin{array}{ccc}
137 & 133 & 137 \\
128 & 131 & 133 \\
122 & 138 & 134 \\
102 & 116 & 126 \\
84 & 102 & 113 \\
86 & 101 & 120 \\
\end{array}
\]

\[
\begin{array}{ccc}
0.1 & 0.1 & 0.1 \\
0.1 & 0.2 & 0.1 \\
0.1 & 0.1 & 0.1 \\
\end{array}
\]

\[
\begin{array}{ccc}
69 & 95 & 116 \\
68 & 92 & 110 \\
66 & 86 & 104 \\
62 & 78 & 94 \\
57 & 69 & 83 \\
53 & 60 & 71 \\
\end{array}
\begin{array}{ccc}
125 & 129 & 132 \\
120 & 126 & 132 \\
114 & 124 & 132 \\
108 & 120 & 129 \\
98 & 112 & 124 \\
85 & 100 & 114 \\
\end{array}
\]

\[
\begin{array}{ccc}
65 & 98 & 123 \\
65 & 96 & 115 \\
63 & 91 & 107 \\
\end{array}
\begin{array}{ccc}
0.1 & 0.1 & 0.1 \\
0.1 & 0.2 & 0.1 \\
0.1 & 0.1 & 0.1 \\
\end{array}
\]

\[
0.1 \times 65 + 0.1 \times 98 + 0.1 \times 123 + \\
0.1 \times 65 + 0.2 \times 96 + 0.1 \times 115 + \\
0.1 \times 63 + 0.1 \times 91 + 0.1 \times 107
\]

\[
= 92
\]
Correlation Example

- With colour images, perform the dot products over each band
Correlation

### Figure 3.10
Neighborhood filtering (convolution): The image on the left is convolved with the filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel.

where the sign of the offsets in $f(x,y)$ has been reversed. This is called the convolution operator, $g(x,y) = f(x,y) * h$, (3.15), and $h$ is then called the impulse response function.

The reason for this name is that the kernel function, $h$, convolved with an impulse signal, $(i, j)$ (an image that is 0 everywhere except at the origin) reproduces itself, $h(x,y) = h$, whereas correlation produces the reflected signal. (Try this yourself to verify that it is so.)

In fact, Equation (3.14) can be interpreted as the superposition (addition) of shifted impulse response functions $h(i, j)$ multiplied by the input pixel values $f(k, l)$. Convolution has additional nice properties, e.g., it is both commutative and associative. As well, the Fourier transform of two convolved images is the product of their individual Fourier transforms (Section 3.4).

Both correlation and convolution are linear shift-invariant (LSI) operators, which obey both the superposition principle (3.5),

$$h(f_0 + f_1) = h(f_0) + h(f_1),$$

(3.16)

and the shift invariance principle,

$$g(i, j) = f(i + k, j + l),$$

(3.17)

which means that shifting a signal commutes with applying the operator ($h$ stands for the LSI operator). Another way to think of shift invariance is that the operator "behaves the same everywhere."
Correlation Example

- Centre-surround filter
Correlation Example

- Edge effects

To maintain the image size, we can pad the input by adding boundary pixels.
In this example the input has been zero padded.
Padding

• What happens to pixels that overlap the boundary?

“zero” and “clamp” (also called zero-order hold) are common in vision applications.
Correlation and Convolution

- Correlation
  \[ I(x, y) \text{ corr } k(x, y) = \int_t \int_s I(x + s, y + t)k(s, t) \, ds \, dt \]

- Convolution
  \[ I(x, y) \ast k(x, y) = \int_t \int_s I(x - s, y - t)k(s, t) \, ds \, dt \]

For symmetric kernels, correlation == convolution
Point Spread Function

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\times
\begin{array}{cccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\Rightarrow
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
The point spread function is the correlation kernel rotated by 180° (= the convolution kernel)
Gaussian Blur

- Gaussian kernels are often used for smoothing

1D

\[\ast\]

2D
Gaussian Blur

- 2D Gaussian filter is a product of row and column filters
Edge Filtering

- Gradients can be computed using a finite difference approximation to the derivative, e.g., \( g_x = I_{x+1} - I_x \)
Centre Surround Filter

- Useful for extracting features at a certain **scale**

We can implement a **sharpening** filter by adding a multiple of this high-frequency band back to the image.
Properties of Convolution

- Linear + associative, commutative

2.3
Separable Filtering

• 2D Gaussian blur by horizontal/vertical blur
Separable Filtering

- Several useful filters can be applied as independent row and column operations

\[
\begin{bmatrix}
1 & 1 & \cdots & 1 \\
1 & 1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 1
\end{bmatrix}
\quad \quad
\begin{bmatrix}
1 & 2 & 1 \\
4 & 16 & 24 & 16 & 4 \\
6 & 24 & 36 & 24 & 6 \\
4 & 16 & 24 & 16 & 4 \\
1 & 4 & 6 & 4 & 1
\end{bmatrix}
\quad \quad
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\quad \quad
\begin{bmatrix}
1 & -2 & 1 \\
-2 & 4 & -2 \\
1 & -2 & 1
\end{bmatrix}
\]

(a) box, \( K = 5 \)  
(b) bilinear  
(c) “Gaussian”  
(d) Sobel  
(e) corner
You are now ready to try the **Convolution and Image Filtering** section in Project 1

- **convolve_1d** : Implement 1D convolution. Hint: pad the input with zeros to avoid border cases.
- **convolve_gaussian** : you can transpose a kernel to flip horizontal/vertical, but make sure it is a 2D numpy array - use `np.expand_dims` if not
Image Pyramids

Used in Graphics (Mip-map) and Vision (for multi-scale processing)
Resizing Images

- Naive method: form new image by selecting every $n$th pixel

What is wrong with this method?
Resizing Images

- Improved method: first **blur** the image (low pass filter)

With the correct filter, no information is lost (Nyquist)
Aliasing Example

- Sampling every 5th pixel, with and without low pass filtering

No filtering  
Gaussian Blur $\sigma = 3.0$
Resizing Images

- Note that selecting every 10th pixel ignores the intervening information, whereas the low-pass filter (blur) smoothly combines it.
- If we shifted the original image 1 pixel to the right, the aliased image would look completely different, but the low-pass filtered image would look almost the same.
Blur with a Gaussian kernel, then select every 2nd pixel

\[ I_s(x, y) = I(x, y) \ast g_\sigma(x, y) \]

Often approximations to the Gaussian kernel are used, e.g.,

\[ \frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \]
Blur with a Gaussian kernel, then select every 2nd pixel

\[ I_s(x, y) = I(x, y) \ast g_\sigma(x, y) \]

Often approximations to the Gaussian kernel are used, e.g.,

\[ \frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \]

Gaussian Pyramid
Sampling with Pyramids

Find the level where the sample spacing is between 1 and 2 pixels, apply extra fraction of inter-octave blur as needed.
The diagram illustrates the construction of a Laplacian Pyramid from an input image. The process involves the following steps:

1. **Input Image**: The original image is denoted as $G_1$.
2. **Blurring**: The image is blurred, resulting in $G_2$.
3. **Downsampling**: The blurred image is downsampled by a factor of 2, producing $G_3$.
4. **Laplacian Pyramid**: The difference between the downsampled image and the blurred image is calculated, forming the first level of the Laplacian Pyramid, $L_1$.
5. **Repeating the Process**: This process is repeated for $G_2$ and $G_3$, creating $L_2$ and $L_3$.
6. **Smoothing and Downsampling**: The final image, $G_4$, is downsampled and blurred, completing the Laplacian Pyramid with $L_4$.
Pyramid Blending
Pyramid Blending

\[ I = \alpha F + (1 - \alpha)B \]
Pyramid Blending: blend lower frequency bands over larger spatial ranges
Pyramid Blending

- Smooth low frequencies, whilst preserving high frequency detail
The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid (taken from levels 0, 2, and 4). The left and middle columns show the original apple and orange images weighted by the smooth interpolation functions, while the right column shows the averaged contributions.

Figure 3.42

Laplacian pyramid blending details (Burt and Adelson 1983b)© 1983 ACM.
Alpha blend with sharp fall-off
Alpha blend with gradual fall-off
Non-linear Filtering

• Example: Median filter

Figure 3.18 Median and bilateral filtering: (a) original image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered; (e) original image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered. Note that the bilateral filter fails to remove the shot noise because the noisy pixels are too different from their neighbors.

Figure 3.19 Median and bilateral filtering: (a) median pixel (green); (b) selected -trimmed mean pixels; (c) domain filter (numbers along edge are pixel distances); (d) range filter.
Morphology

- Non-linear binary image operations

original  dilate  erode  majority  open  close

Threshold function in local structuring element

close(.) = erode(dilate(.)) etc., see Szeliski 3.3.2
Binary Operators

- More operators that apply to binary images

original image

dilate
distance transform
connected components
Next Lecture

- Feature Extraction and Matching