Image processing

Reading


What is an image?

We can think of an image as a function, $f$, from $\mathbb{R}^2$ to $\mathbb{R}$:

- $f(x, y)$ gives the intensity of a channel at position $(x, y)$
- Realistically, we expect the image only to be defined over a rectangle, with a finite range:
  - $f: [a,b] \times [c,d] \to [0,1]

A color image is just three functions pasted together. We can write this as a “vector-valued” function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$
What is a digital image?

In computer graphics, we usually operate on digital (discrete) images:

- **Sample** the space on a regular grid
- **Quantize** each sample (round to nearest integer)

If our samples are $\Delta$ apart, we can write this as:

$$f[i, j] = \text{Quantize}\{f[i \Delta, j \Delta]\}$$

The image can now be represented as a matrix of integer values

\[\begin{array}{cccccccc}
  & 112 & 102 & 113 & 120 & 105 & 6 & 10 \\
 12 & 75 & 91 & 62 & 77 & 80 & 24 & 10 \\
 12 & 81 & 80 & 105 & 86 & 30 & 5 & 80 \\
 12 & 135 & 5 & 101 & 101 & 85 & 6 & 80 \\
 7 & 1 & 1 & 20 & 20 & 87 & 6 & 77 \\
 6 & 18 & 144 & 197 & 197 & 197 & 62 & 200 \\
 20 & 20 & 6 & 106 & 123 & 60 & 0 & 31 \\
 60 & 66 & 507 & 17 & 1 & 0 & 89 & 30
\end{array}\]

Image processing

An image processing operation typically defines a new image $g$ in terms of an existing image $f$.

We can transform either the domain or the range of $f$.

**Range transformation:**

$$g(x, y) = t(f(x, y))$$

What's kinds of operations can this perform?

Image processing

Some operations preserve the range but change the domain of $f$:

$$g(x, y) = f(t_x(x, y), t_y(x, y))$$

What kinds of operations can this perform?

Noise

Image processing is also useful for noise reduction and edge enhancement. We will focus on these applications for the remainder of the lecture...

Common types of noise:

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise**: contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution
Practical noise reduction

How can we “smooth” away noise in a single image?

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>130</td>
<td>110</td>
<td>120</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>110</td>
<td>90</td>
<td>100</td>
<td>90</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
<td>100</td>
<td>90</td>
<td>130</td>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>100</td>
<td>130</td>
<td>110</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>110</td>
<td>80</td>
<td>120</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$F[x, y]$

$G[x, y]$
Mean filtering

\[
F[x, y] = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]

\[
G[x, y] = \begin{bmatrix}
0 & 10 & 20 & 30 & 30 & 30 & 20 & 10 \\
0 & 20 & 40 & 60 & 60 & 60 & 40 & 20 \\
0 & 30 & 60 & 90 & 90 & 90 & 60 & 30 \\
0 & 30 & 60 & 90 & 90 & 90 & 60 & 30 \\
0 & 30 & 60 & 90 & 90 & 90 & 60 & 30 \\
0 & 20 & 30 & 50 & 50 & 50 & 30 & 20 \\
10 & 20 & 30 & 30 & 30 & 30 & 20 & 10 \\
10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix}
\]

Effect of mean filters

Cross-correlation filtering

Let’s write this down as an equation. Assume the averaging window is \((2k+1)\times(2k+1)\):

\[
G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]
\]

We can generalize this idea by allowing different weights for different neighboring pixels:

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

This is called a cross-correlation operation and written:

\[
G = H \otimes F
\]

H is called the “filter,” “kernel,” or “mask.”

The above allows negative filter indices. When you implement need to use: \(H[u+k,v+k]\) instead of \(H[u,v]\)
Mean kernel

What’s the kernel for a $3 \times 3$ mean filter?

\[
H[u, v] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 90 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

\[
F[x, y]
\]

Gaussian Filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

\[
H[u, v] = \begin{bmatrix} 1/16 & 2/4 & 1/2 \\ 2/4 & 4 & 2/4 \\ 1/2 & 2/4 & 1/16 \end{bmatrix}
\]

This kernel is an approximation of a Gaussian function:

\[
h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2 + v^2}{2\sigma^2}}
\]

Mean vs. Gaussian filtering

Convoluption

A convolution operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i-u, j-v]
\]

It is written: \( G = H \ast F \)

Suppose \( H \) is a Gaussian or mean kernel. How does convolution differ from cross-correlation?
Median filters

A Median Filter operates over a window by selecting the median intensity in the window.

What advantage does a median filter have over a mean filter?

Is a median filter a kind of convolution?

Comparison: salt and pepper noise

Comparison: Gaussian noise

Edge detection

One of the most important uses of image processing is edge detection:

- Really easy for humans
- Really difficult for computers
- Fundamental in computer vision
- Important in many graphics applications

How to tell if a pixel is on an edge?
Gradients

The gradient is the 2D equivalent of the derivative:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

Properties of the gradient

- It’s a vector
- Points in the direction of maximum increase of \( f \)
- Magnitude is rate of increase

How can we approximate the gradient in a digital image?

Gradient filters

A popular gradient filter is the Sobel operator:

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{bmatrix}
\]

Results of Sobel edge detection

Laplacian filter

Instead of finding maxima of the first derivative, we can instead look for zero-crossings of the 2nd derivative

A convenient measure of the second derivative in 2D is the Laplacian:

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

A 3x3 Laplacian kernel is:

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{bmatrix}
\]
Edge detection with the Laplacian

Original

Smoothed

Laplacian (+128)

Sharpening with the Laplacian

Original

Laplacian (+128)

Original - Laplacian

What is the kernel that creates the bottom image?

Summary

What you should take away from this lecture:

- The meanings of all the boldfaced terms.
- A richer understanding of the terms “image” and “image processing”
- How noise reduction is done
- How cross-correlation and convolution filtering works
- The effect of mean, Gaussian, and median filters
- What an image gradient is and how it can be computed
- Edge detection with Sobel and Laplacian filters
- Image sharpening with the Laplacian filter