What is an image?

We can think of an image as a function, \( f(x, y) \), 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:
  - \( f(x, y) \in [0, 1] \) if a single channel
  - \( f(x, y) \in [0, 255] \) if an 8-bit grayscale channel
  - \( f(x, y) \in [0, 255]^3 \) if an 8-bit RGB channel

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:

$$f(i) = \text{Quantize}(f(i) / \Delta)$$

![Image of digital image samples]

Image processing

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

The simplest operations are those that transform each pixel in isolation. These pixel-to-pixel operations can be written:

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

Examples: threshold, RGB $\rightarrow$ grayscale

Note: a typical choice for mapping to grayscale is to apply the YIQ television matrix and keep the Y.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Pixel movement

Some operations preserve intensities, but move pixels around in the image.

$$g(x, y) = f(\tilde{x}(x, y), \tilde{y}(x, y))$$

Examples: many amusing warps of images

![Image sequence of pixel movement]

Noise

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

![Images of noise types]

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
Ideal noise reduction

Practical noise reduction

How can we "smoothen" away noise in a single image?

Convolution

One of the most common methods for filtering an image is called convolution.

In 1D, convolution is defined as:

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(x') h(x-x') dx'$$

where $\tilde{h}(x) = h(-x)$.

Example:
**Discrete convolution**

For a digital signal, we define **discrete convolution** as:

\[ g[n] = f[n] * h[n] \]
\[ = \sum_{n} f[n'] h[n-n'] \]
\[ = \sum_{\sigma} f[\sigma'] h[\sigma-n] \]

where \( h[n] = h[-n] \).

**Aside:**

One can show that convolution has some convenient properties. Given functions \( a, b, c \):

\[ a*b = b*a \]
\[ (a*b)*c = a*(b*c) \]
\[ a*(b+c) = a*b + a*c \]

We'll make use of these properties later...

---

**Convolution in 2D**

In two dimensions, convolution becomes:

\[ g(x,y) = f(x,y) * h(x,y) \]
\[ = \int \int f(x',y') h(x-x',y-y') \, dx' \, dy' \]
\[ = \int \int f(x',y') \tilde{h}(x-x',y-y') \, dx' \, dy' \]

where \( \tilde{h}(x,y) = h(-x,-y) \).

---

**Discrete convolution in 2D**

Similarly, discrete convolution in 2D is:

\[ g[n,m] = f[n',m'] * h[n,m] \]
\[ = \sum_{n',m'} f[n',m'] h[n-n',m-m'] \]
\[ = \sum_{\sigma_n, \sigma_m} f[\sigma_n', \sigma_m'] h[\sigma_n-n, \sigma_m-m] \]

where \( h[n,m] = h[-n,-m] \).

---

**Convolution representation**

Since \( f \) and \( h \) are defined over finite regions, we can write them out in two-dimensional arrays:

<table>
<thead>
<tr>
<th></th>
<th>128</th>
<th>94</th>
<th>54</th>
<th>78</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>145</td>
<td>98</td>
<td>240</td>
<td>233</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>177</td>
<td>246</td>
<td>228</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>90</td>
<td>255</td>
<td>237</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>111</td>
<td>128</td>
<td>167</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>221</td>
<td>154</td>
<td>97</td>
<td>123</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** This is not matrix multiplication!

**Q:** What happens at the edges?
Mean filters

How can we represent our noise-reducing averaging as a convolution filter (known as a mean filter)?

$$\frac{1}{N^2} \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

Gaussian filters

Gaussian filters weigh pixels based on their distance from the center of the convolution filter. In particular:

$$h[n, m] = \frac{e^{-(n^2 + m^2)}/2\sigma^2}}{C}$$

This does a decent job of blurring noise while preserving features of the image.

What parameter controls the width of the Gaussian? $\sigma$

What happens to the image as the Gaussian filter kernel gets wider? blurrier

What is the constant $C$? What should we set it to?

$$C = \sum_{n,m} e^{-(n^2 + m^2)/2\sigma^2}$$
Median filters

A median filter operates over an image region by selecting the median intensity in the region.

What advantage does a median filter have over a mean filter? [Outlier noise suppression]

Is a median filter a kind of convolution? [No, somewhat edge preserving]

Effect of median filters

Comparison: Gaussian noise

Comparison: Salt and pepper noise
Bilateral filtering is a method to average together nearby samples only if they are similar in value.

Bilateral filter is similar, but includes both range and domain filtering:

$$g[n] = \frac{1}{C} \sum_{n'} f[n'] \cdot h_{\sigma_R}[n - n'] \cdot h_{\sigma_D}[f[n] - f[n']]$$

and you have to normalize as you go:

$$C = \sum_{n'} h_{\sigma_D}[n - n'] \cdot h_{\sigma_D}[f[n] - f[n']]$$

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
What is 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 discrete image?

$$\frac{df}{dx} \approx f[n+1,m] - f[n,m]$$
$$\frac{df}{dy} \approx f[y,n+1] - f[y,n]$$

Steps in edge detection

Edge detection algorithms typically proceed in three or four steps:
- **Filtering:** cut down on noise
- **Enhancement:** amplify the difference between edges and non-edges
- **Detection:** use a threshold operation
- **Localization (optional):** estimate geometry of edges beyond pixels
**Edge enhancement**

A popular gradient filter is the Sobel operator:

\[
\mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}
\]

We can then compute the magnitude of the vector \((\mathbf{G}_x, \mathbf{G}_y)\).

Note that these operators are conveniently "pre-flipped" for convolution, so you can directly slide these across an image without flipping first.

**Results of Sobel edge detection**

- **Original**
- **Smoothed**
- **Sx + 128**
- **Sy + 128**
- Magnitude
- Threshold = 64
- Threshold = 128

**Localization with the Laplacian**

An equivalent measure of the second derivative in 2D is the Laplacian:

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

Using the same arguments we used to compute the gradient filters, we can derive a Laplacian filter to be:

\[
\Delta = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}
\]

(The symbol \(\Delta\) is often used to refer to the discrete Laplacian filter.)

Zero crossings in a Laplacian filtered image can be used to localize edges.

**Second derivative operators**

The Sobel operator can produce thick edges. Ideally, we’re looking for infinitely thin boundaries.

An alternative approach is to look for local extrema in the first derivative: places where the change in the gradient is highest.

**Q:** A peak in the first derivative corresponds to what in the second derivative? **Zero crossing**

**Q:** How might we write this as a convolution filter?
Localization with the Laplacian

Original

Smoothed

Laplacian (+128)

Marching squares

We can convert these signed values into edge contours using a "marching squares" technique.

Sharpening with the Laplacian

Original

Laplacian (+128)

Original + Laplacian

Original - Laplacian

Why does the sign make a difference?

How can you write the filter that makes the sharpened image?

Summary

What you should take away from this lecture:

- The meanings of all the boldfaced terms.
- How noise reduction is done.
- How discrete convolution filtering works.
- The effect of mean, Gaussian, and median filters.
- What an image gradient is and how it can be computed.
- How edge detection is done.
- What the Laplacian image is and how it is used in either edge detection or image sharpening.