Image Processing

Definitions

- Many graphics techniques that operate only on images
- **Image processing**: operations that take images as input, produce images as output
- In its most general form, an **image** is a function $f$ from $\mathbb{R}^2$ to $\mathbb{R}$
  - $f(x, y)$ gives the intensity of a channel at position $(x, y)$ defined over a rectangle, with a finite range:
    $$f: [a,b] \times [c,d] \rightarrow [0,1]$$
  - A color image is just three functions pasted together:
    $$f(x, y) = (f_r(x, y), f_g(x, y), f_b(x, y))$$

Images as Functions
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 \( d \) apart, we can write this as:

\[
 f'(i, j) = \text{Quantize}(f(i \cdot d, j \cdot d))
\]

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:

\[
g(x, y) = t(f(x, y))
\]

- Example: threshold, RGB \( \rightarrow \) grayscale

Pixel Movement

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

\[
g(x, y) = f(u(x, y), v(x, y))
\]

- Examples: many amusing warps of images
Multiple input images

- Some operations define a new image $g$ in terms of $n$ existing images ($f_1, f_2, \ldots, f_n$), where $n$ is greater than 1.
- Example: cross-dissolve between 2 input images.

Noise

- 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.

Noise Examples

- Ideal noise reduction

- Original
- Salt and pepper noise
- Impulse noise
- Gaussian noise
**Ideal noise reduction**

**Practical noise reduction**

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

**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 3x3 mean filter?

\[
F[x, y]
\]
Mean Filters

Gaussian Filtering

- A Gaussian kernel gives less weight to pixels further from the center of the window.
- This kernel is an approximation of a Gaussian function:
  \[ h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2 + v^2}{2\sigma^2}} \]

Gaussian Filters

- Gaussian filters weigh pixels based on their distance to the location of convolution.
  \[ h[i, j] = e^{-\frac{(i^2 + j^2)}{2\sigma^2}} \]
- Blurring noise while preserving features of the image
- Smoothing the same in all directions
- More significance to neighboring pixels
- Width parameterized by \( \sigma \)
- Gaussian functions are separable
- Convolving with multiple Gaussian filters results in a single Gaussian filter

Convolution

- 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 $k \times k$ region by selecting the median intensity in the region.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?
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?**

**Gradient**

- 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?
Edge Detection Algorithms
- 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 magnitude computation is the Sobel operator:
  \[
  s_x = \begin{bmatrix}
  -1 & 0 & 1 \\
  -2 & 0 & 2 \\
  -1 & 0 & 1 
  \end{bmatrix}
  \]

- We can then compute the magnitude of the vector \((s_x, s_y)\):

Sobel Operator
- The Sobel operator is applied to an image to detect edges. The input image is shown in the left column, and the Sobel operators are applied to the image to produce the result shown in the right column.
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?

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:
  \[
  \Delta^2 = \begin{bmatrix}
  0 & 1 & 0 \\
  1 & -4 & 1 \\
  0 & 1 & 0
  \end{bmatrix}
  \]
- Zero crossings of this filter correspond to positions of maximum gradient. These zero crossings can be used to localize edges.

Laplacian alternatives

<table>
<thead>
<tr>
<th>0  1  0</th>
<th>1  1  1</th>
<th>-1  2  -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 -4  1</td>
<td>1 -8  1</td>
<td>2 -4  2</td>
</tr>
<tr>
<td>0  1  0</td>
<td>1  1  1</td>
<td>-1  2  -1</td>
</tr>
</tbody>
</table>

Localization with the Laplacian

Original Smoothed

Laplacian (+128)
**Summary**

- Formal definitions of image and image processing
- Kinds of image processing: pixel-to-pixel, pixel movement, convolution, others
- Types of noise and strategies for noise reduction
- Definition of convolution and how discrete convolution works
- The effects of mean, median and Gaussian filtering
- How edge detection is done
- Gradients and discrete approximations