Image processing
Reading

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:

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

- 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

[Show image sequence.]
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
Ideal noise reduction
Ideal noise reduction
Practical noise reduction

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

- Is there a more abstract way to represent this sort of operation? *Of course there is!*
Discrete convolution

For a digital signal, we define \textbf{discrete convolution} as:

\[ g[i] = f[i] \ast h[i] \]

\[ = \sum_{i'} f[i']h[i - i'] \]

\[ = \sum_{i'} f[i']\hat{h}[i' - i] \]

where \[ \hat{h}[i] = h[-i] \]
Discrete convolution in 2D

Similarly, discrete convolution in 2D becomes:

\[
g[i, j] = f[i, j] * h[i, j] \\
= \sum_{i'} \sum_{j'} f[i', j'] h[i - i', j - j'] \\
= \sum_{i'} \sum_{j'} f[i', j'] \hat{h}[i' - i, j' - j]
\]

where \( \hat{h}[i, j] = h[-i, -j] \)
Convolution representation

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

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>79</td>
<td>23</td>
<td>119</td>
<td>120</td>
<td>105</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>9</td>
<td>62</td>
<td>12</td>
<td>78</td>
<td>34</td>
</tr>
<tr>
<td>10</td>
<td>58</td>
<td>197</td>
<td>46</td>
<td>46</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>176</td>
<td>135</td>
<td>5</td>
<td>138</td>
<td>191</td>
<td>68</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>26</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>89</td>
<td>144</td>
<td>147</td>
<td>187</td>
<td>102</td>
<td>62</td>
</tr>
<tr>
<td>265</td>
<td>252</td>
<td>0</td>
<td>166</td>
<td>123</td>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>166</td>
<td>63</td>
<td>127</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>99</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 filter as a convolution diagram (known as a mean filter)?
Effect of mean filters

<table>
<thead>
<tr>
<th>Gaussian noise</th>
<th>Salt and pepper noise</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="3x3 Gaussian noise" /></td>
<td><img src="image2" alt="3x3 Salt and pepper noise" /></td>
</tr>
<tr>
<td><img src="image3" alt="5x5 Gaussian noise" /></td>
<td><img src="image4" alt="5x5 Salt and pepper noise" /></td>
</tr>
<tr>
<td><img src="image5" alt="7x7 Gaussian noise" /></td>
<td><img src="image6" alt="7x7 Salt and pepper noise" /></td>
</tr>
</tbody>
</table>
Gaussian filters

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

\[ h[i, j] = \frac{e^{-(i^2+j^2)/(2\sigma^2)}}{C} \]

- This does a decent job of blurring noise while preserving features of the image.
- What parameter controls the width of the Gaussian?
- What happens to the image as the Gaussian filter kernel gets wider?
- What is the constant \( C \)? What should we set it to?
Effect of Gaussian filters

Gaussian noise  Salt and pepper noise

3x3

5x5

7x7
Median filters

- A **median filter** operates over an \( m \times m \) 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?
Effect of median filters

3x3

5x5

7x7

Gaussian noise

Salt and pepper noise
Comparison: Gaussian noise

Mean  Gaussian  Median

3x3

5x5

7x7
Comparison: salt and pepper noise
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?

**Q:** How might you detect an edge in 1D?
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?
Less than ideal edges

![Graph showing pixel distribution](image)
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 magnitude computation is the Sobel operator:

$$s_x = \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}$$

$$s_y = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{bmatrix}$$

We can then compute the magnitude of the vector \((s_x, s_y)\).
Results of Sobel edge detection

Original
Smoothed
Sx + 128
Sy + 128
Magnitude
Threshold = 64
Threshold = 128
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?

Q: How might we write this as a convolution filter?
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.
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

Why does the sign make a difference?
How can you write each filter that makes each bottom image?
Spectral impact of sharpening

We can look at the impact of sharpening on the Fourier spectrum:

\[
\delta - \Delta^2 = \begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0 
\end{bmatrix}
\]
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
Next time: Affine Transformations

■ Topic:
■ How do we represent the rotations, translations, etc. needed to build a complex scene from simpler objects?

■ Read:
  • Watt, Section 1.1.

Optional:
  • Foley, et al, Chapter 5.1-5.5.