Image processing (part 2)

Comparison: Gaussian noise

<table>
<thead>
<tr>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x3</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>5x5</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>7x7</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Comparison: salt and pepper noise

<table>
<thead>
<tr>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x3</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>5x5</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>7x7</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Bilateral filtering

Bilateral filtering is a method to average together nearby samples only if they are similar in value.

\[
g(n) = \frac{1}{C} \sum_{n'} f(n') h_{\sigma_s} (n - n') h_{\sigma_f} (f[n] - f[n'])
\]

\[
C = \sum_{n} h_{\sigma_s} (n - n') h_{\sigma_f} (f[n] - f[n'])
\]
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?
Less than ideal edges

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
```
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?

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.

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:

\[ \delta - \Delta^2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \]