Introduction to Computer Vision

Alvin Lin

August 2018 - December 2018

Edge Detection

Intuitions guiding edge detection:

- Pixels tend strongly to be like their neighbors. As a consequence, we can estimate a pixel value using its neighbors.
- Smoothing with a Gaussian. This works because pixels look like their neighbors. Positive and negative errors tend to cancel.
- Pixels can differ from their neighbor because they have different albedos, are on different objects, have different surface normals, or have a big difference in shading.
- Pixels that differ from their neighbors are interesting to us because they occur when the gradient is large.

The key idea is that we will suppress image noise by smoothing, and then taking a gradient. An edge is a place of rapid change in the image intensity function, so we can use a derivative to detect this.

Image Gradient

The gradient of an image is defined as:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of the most rapid increase in intensity. Its direction is given by:

$$\theta = \tan^{-1}\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right)$$

The edge strength is given by the gradient magnitude:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Differentiation and Convolution

For a 2D function, the partial derivative is defined as:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$

For discrete data like images, we can approximate this using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

This is linear and shift invariant, and must be the reuslt of a convolution. The associated filter to produce this would be:

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

Other approximations of derivate filters exist:

- Prewitt:

$$M_z = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

- Sobel:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Roberts:

$$M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Noise

Consider plotting a single row or column of the image, if we plot the intensity as a function of position, we get a signal. Derivatives respond strongly to pixels that differ from their neighbors, so noise is heavily amplified if we take the derivative. It is better to smooth first before taking the derivative. With the smoothing kernel h, our convolution output is:

$$\frac{\partial}{\partial x}(h \star f)$$

By the derivative theorem of convolution:

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Because of this, we can filter the image with the derivative of Gaussian filters to get the smoothed gradient.

Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$. By taking the second derivative of the signal, we can consider all the zero-crossings as edges since that is the greatest change in intensity. Again, we can use the derivative theorem:

$$\frac{\partial^2}{\partial x^2}(h \star f) = (\frac{\partial^2}{\partial x^2}h) \star f$$

This is also known as the Laplacian filter (∇^2).

Designing and Edge Detector

An optimal edge detector has:

- Good detection: the optimal detector must have a small number of false positives and false negatives.
- Good localization: the edges detected must be as close as possible to the true edges.
- Single response: the detector must return one point only for each true edge point, minimizing the number of local maxima around the true edge.

2D Edge Detection Filters

Gaussian Filter (for smoothing):

$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{u^2+v^2}{2\sigma^2}}$$

Derivative of Gaussian (for less noisy edge detection):

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$

Laplacian of Gaussian (to find zero-crossings):

$$\nabla^2 h_\sigma(u, v)$$

Canny Edge Detector

The Canny edge detector is probably the most widely used edge detector in computer vision. Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization. Steps:

- Filter the image with the x and y derivatives of the Gaussian
- Find the magnitude and orientation of the gradient
- Non-maximum suppression: thin down multi-pixel wide "ridges" down to a single pixel width
- Thresholding and linking (hysteresis): use a high threshold to start edge curves and a low threshold to continue them

Using a small Gaussian kernel spread size detects fine features while a large spread size detects large scale edges.

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech