Math 142 Reading Week 10

Abhi Uppal

November 22nd, 2021

1 4.2 Edges

Edge points are often more important than interest points – partially because they are more plentiful, and mostly because they correspond to occlusion in 3d events, they represent the boundaries of objects, and are usually delineated by visual contours.

1.1 4.2.1 Edge Detection

The big question here is: given an image, how do we find all of the salient edges?

Qualitatively, we can describe an edge to be something that occurs at boundaries between regions of different colors, intensities, or textures. However, it is unfortunately difficult to segment an image into coherent regions. Rather, we usually want to detect edges based on local features.

So, a reasonable approach would be to think of an edge as an area of rapid intensity variation. Think of an image as a height field, where the height corresponds to intensity – an edge would then correspond to a cliff or a steep slope. Graphically, this looks like a region of closely packed contour lines on a topographic map. Mathematically, we represent it with a *gradient*:

$$J(x) = \nabla I(x) = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})(x)$$

This local gradient function points in the direction of the steepest increase in the gradient function. The magnitude indicates the strength of this variation, and it points perpendicular to a local contour.

Taking image derivatives tends to amplify noise, since the noise to signal ratio is larger at high frequencies. It's therefore a good idea to send an image through a low-pass filter as a preprocessing step to comptuing a gradient. We use a Gaussian smoothing filter since it is the only clearly separable and circularly symmetric smoothing filter (circular symmetry is important since we want edge detection to be orientation-independent).

Differentiation is a linear operator, so it commutes with other linear filters. With this in mind, we can write the gradient of the smoothed image as:

$$J_\sigma(x) = \nabla[G_\sigma(x) * I(x)] = [\nabla G_\sigma](x) * I(x),$$

where G_σ represents the Gaussian kernel function and $*$ represents the convolution operator.

If we want to determine isolated edges (single pixels at discrete locations on edge contours), we can do this by selecting the maxima of the gradient magnitude in a direction perpendicular to the contour (along the gradient direction)

Finding this maximum corresponds to taking a directional derivative in the direction of the gradient and then looking for zero crossings. The desired directional derivative is:

$$S_\sigma(x) = \nabla \cdot J_\sigma(x) = [\nabla^2 G_\sigma](x) * I(x)$$

The gradient inside this expression is called the *Laplacian* – in this case it is called the *Laplacian of Gaussian (LoG)*.