CSE 252A: Edge Detection

Lecturer: David Kriegman

Scribed by Owen Jow on October 30, 2018

1 Filtering Again

For most of the contexts in which we apply convolutions (denoising, CNNs, etc.), we're using convolutions because of an assumption that neighboring data values are related in some way. If neighboring values are totally independent, convolution would not be appropriate for these applications.

1.1 Median Filter

Replace each value with the median of its neighborhood. This filter does not create new values; each kernel application produces one of the values that were already there. So it wouldn't be good for smoothing out a halftone image,[1] but it is good at eliminating spike values[2] in an image.

If you apply the median filter too many times, you lose structure (details).

2 Edge Detection

Let's say we want to **segment** the image, or divide it into regions that are semantically meaningful. One way to do this is to detect edges and treat them as boundaries. An **edge** is a discontinuity in brightness as we move in position (a break in space between one intensity and a different intensity).

Edges (as intensity changes, ignoring color) might be image manifestations of object boundaries, surface normal discontinuities, reflectance discontinuities, or lighting discontinuities.

Basically, we want to find discontinuities (jumps). However, due to noise or small variations, we might have jumps everywhere. So we'll want to take image gradients in order to identify discontinuities (edge = "sharp change" = high derivative value), but we'll want to smooth the image first.

Images are sampled signals. To take a horizontal derivative[3] of an image I at (x, y) using central differences, we can simply compute $I(x+1, y) - I(x-1, y)$. We can also divide by 2, which is how a Taylor series expansion will tell us to compute first derivatives.

[1] When we smooth a halftone image, we want each inter-dot value to be an average of the dots around it.

[2] By "spike values," we mean isolated, nontrivially erroneous values. Reconstruction methods often produce these.

[3] Derivatives w.r.t. y work the same way, of course.

2.1 1D Edge Detection

1. Filter out noise by convolving with a Gaussian.
2. Take a derivative by convolving with $[-1, 0, 1]$. (Can combine with the previous step.[4])
3. Find the peaks of the filtered signal. These should be local maxima and sufficiently large.

2.2 2D Edge Detection (Canny)

1. Filter out noise by convolving with a Gaussian G (then "J" $= I * G$).
2. Take a derivative in each direction, combine as gradient magnitude.
   (a) If the gradient ∇J is $(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y})$,[5][6] then the gradient magnitude $\|\nabla J\|$ is $\sqrt{\left(\frac{\partial J}{\partial x}\right)^2 + \left(\frac{\partial J}{\partial y}\right)^2}$.
   (b) Also compute the gradient direction as $\arctan\left(\frac{\partial J}{\partial y} / \frac{\partial J}{\partial x}\right)$.
3. Perform non-maximum suppression (see section 2.2.2).[7]
4. Perform hysteresis thresholding (see section 2.2.3) to track the final edges.

2.2.1 Choice of σ

Note that the gradient magnitude image changes according to the scale σ used to construct the Gaussian filter. A smaller σ means thinner edges which are more sensitive to small variations like noise, while a larger σ means the opposite.

We must take care when setting σ, as there is always a tradeoff between noise and blurred edges! Do we prioritize detection accuracy (making sure we get the right edges) or localization accuracy (making sure our edges are precise)?

Prof. Canny addressed this issue, coming up with a way to optimally assign σ based on the tradeoff between false positives and poor localization. In practice, we usually just pick a σ that works.

2.2.2 Non-Maximum Suppression

In this step, we suppress each gradient magnitude value (set it to 0) *if it is not larger than its two neighbors in the gradient direction at the point.* The gradient direction is orthogonal to the edge represented at the point, so this is thinning out the broad edges from the Gaussian smoothing.

Since the gradient direction probably won't give rise to two exact neighbors in the nearest two rows/columns,[8] we can interpolate to obtain the two neighboring values in the nearest rows/columns.

[4] "derivative of Gaussian filter"

[5] Note that we have a gradient at each location in the image. The overall result is either (Cartesian) an x-derivative image and a y-derivative image, or (polar) a gradient direction image and a gradient magnitude image [reference].

[6] You can also take directional derivatives, i.e. the rate of change in a particular direction as opposed to just the x- or y-direction. This isn't relevant here, but just making a note that these aren't the only derivatives possible.

[7] Unlike the 1D case, we can't just find peaks. A peak, as an isolated value greater than all of the surrounding values, isn't the right notion to use. We want curves (boundaries), not isolated values.

[8] rows if the gradient direction is closer to vertical, columns if the gradient direction is closer to horizontal

2.2.3 Hysteresis Thresholding

Before the previous step, our gradient magnitude image will often depict thick edges (due to the Gaussian filtering and the σ we choose). Since the end result should be edges of minimal width, we ultimately want to trace curves through these thick edges which go through high-magnitude points.

Non-maximum suppression takes care of thinning out edges. **Hysteresis thresholding** takes care of the actual edge tracing part. It involves two thresholds (a "double threshold").[9]

- Any magnitude less than the lower threshold is considered "not an edge" and discarded.
- Any magnitude greater than the higher threshold is considered "definitely an edge" and kept.
- Any magnitude between the thresholds is kept if it is connected[10] to a "definite edge."

To do tracing, we start at each local maximum which is greater than the high threshold and follow the edge in the direction orthogonal to the gradient, stopping when the magnitude drops below the low threshold. (The high threshold starts the edge curve, the low threshold continues it.)

At the end of the day, we are left with only thin, strong edges.

2.2.4 Parameters

Overall, the Canny edge detector involves three parameters: σ, τ_{high}, and τ_{low}, where the latter two parameters are of course the high and low thresholds for hysteresis thresholding and the former parameter is the width of the isotropic Gaussian used for filtering.

[9]If we only have one threshold, we miss out on good edges if it is too high and have bad edges if it is too low.

[10]We determine connectivity based on gradient direction. From a given point, we can predict the next point along the edge curve as the nearest point in the direction orthogonal to the gradient direction. Note that there are *two* directions orthogonal to the gradient direction; we can follow either of them because an edge goes both ways.