Lecture notes: Edge detection continued

The most famous edge detector is due to Canny. Over the years, hundreds (or thousands) of edge detection algorithms and ideas have been published. Canny put together a lot of ideas that we have discussed, along with one or two new ones (see below). Since his paper (mid 80's), no one has made any further substantial progress. Therefore his paper has become a sort-of standard against which progress (or lack thereof) is measured.

Canny's algorithm has the following steps (see Sonka, page 92):

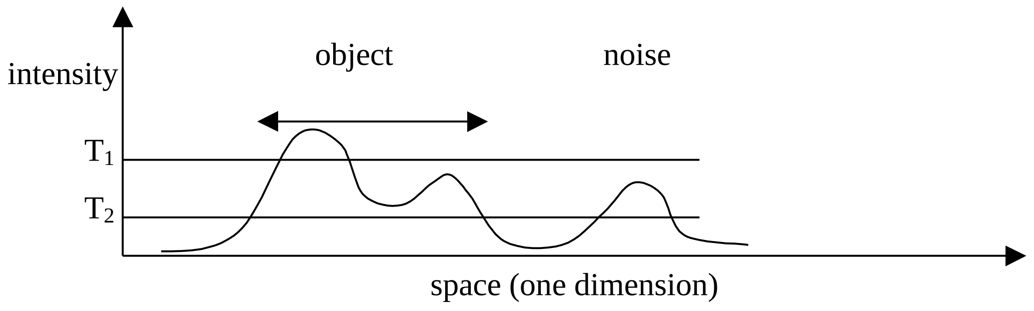
1. Smooth image (convolve with Gaussian of set scale sigma).

2. Estimate local edge direction at each pixel (see text for exact formula). This lets the actual gradient magnitude of the edge be computed along a specific direction. It also lets the zero-crossing be detected in a precise direction, giving better localization.

3. Estimate edge location at each pixel (find zero crossing). Canny calls this "non-maximal supression". This method of localizing edges using direction first is considered a breakthrough, but I personally don't think it makes that big a difference.

4. Estimate edge magnitude at each pixel.

5. Threshold the gradient magnitude image using hysteresis. A pixel must have a response $>T_1$ or be connected to a pixel having a response $>T_1$ and itself have a response $>T_2$. This can be illustrated in 1D:



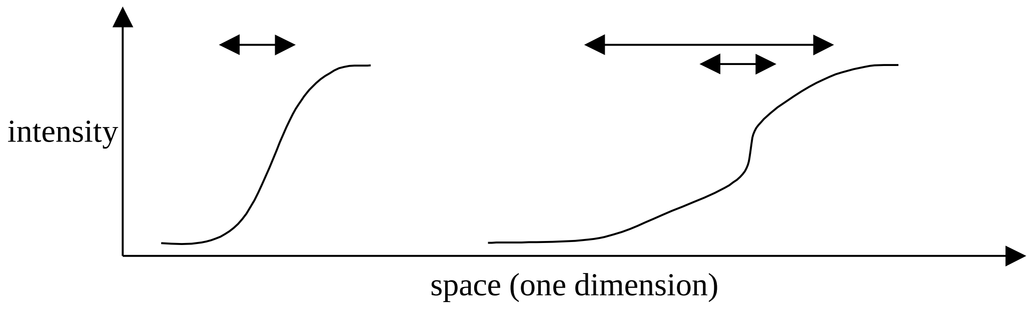
   In this example using hysteresis thresholds the entire object, whereas a single threshold T_1 would miss half the object, and a lower single threshold T_2 would also pass a spurious response.

   I personally believe this was Canny's biggest contribution. In the example from his paper, you can see how hysteresis gives more complete boundaries of the objects without the additional spurious edges that would be found using a single lower threshold.

6. Repeat steps 1-5 for ascending values of sigma (just like scale-space filtering).

7. Aggregate responses using "feature synthesis". A response at a small value of sigma is used to predict how the response should look at a larger value of sigma. If the prediction matches, then the response at the smaller value is kept. If the response grows, then the edge

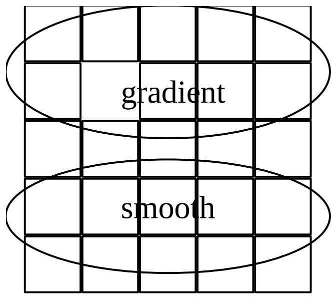
is assumed to be wider and the response at a larger sigma is taken. The following example demonstrates in 1D:



In the example on the left, a small value of sigma finds a strong edge that is verified through weaker responses at larger values of sigma. In the example on the right, a small value of sigma finds a strong edge but larger values of sigma find even stronger edges, so that the edge is better located using a larger sigma.

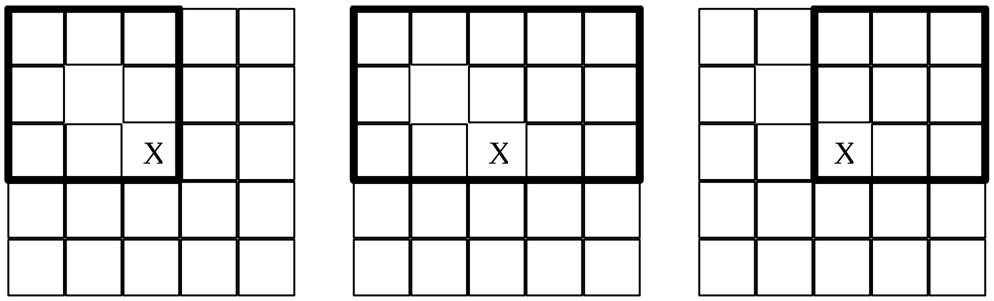
In recent years there has been an edge detector evaluation study, comparing how people perceive the results of different algorithms, and how machine vision systems perform using the results of different algorithms. The results of these studies support the idea that there has been little-to-no progress since Canny's paper.

Changing topics slightly, there is another older idea on how to solve the smoothing vs. gradient detection battle. It is called **edge preserving smoothing.** The idea is to perform smoothing only in the part of the window in which the gradient is weakest:



In this example, the strongest gradient is horizontal on the top side, so smoothing is only done in the bottom half of the window.

Edge preserving smoothing is implemented using a set of **sub-masks.** The following figure shows three example sub-masks of a 5x5 window around a pixel "X" (the number of sub-masks and their shapes can vary).



The variance in intensity is computed in each sub-mask. Smoothing is done in the sub-mask with the lowest variance, or in the entire window if the total variance is below some threshold (meaning there is no substantial gradient in the window).