Canny Edge Detection

Harshit Varma

October 2020

Let the original image be I

Step 1: Gaussian Blurring

This removes noise, and removes false edges, but it also smoothens true edges at the same time.
Let $G(k, \sigma)$ be a $k \times k$ Gaussian filter with standard deviation σ, then:

$$I_1 = G(k, \sigma) * I$$

This step has two parameters, (k, σ)

Step 2: Gradient Calculation

We use the sobel kernel for gradient approximation.
Let S_x and S_y be the sobel filters.
Then $I_{1_x} = S_x * I_1$ and $I_{1_y} = S_y * I_1$ and,
$I_2 = \sqrt{I_{1_x}^2 + I_{1_y}^2}$ and $\theta = tan^{-1}(\frac{I_{1_y}}{I_{1_x}})$ (Note that θ is the elementwise tan^{-1} of I_{1_y}/I_{1_x} (This is elementwise division))
Other kernels (Roberts cross, Prewitt operator) apart from sobel's can also be used for gradient computations.

Step 3: Non-max Suppression

This is used for thinning the images.
For each pixel in I_2, say at (i, j), find $\theta_{i,j}$.
Now, along $\theta_{i,j}$, compare the intensity values of the nearest neighbours of the pixel, i.e let these neighbours be at (i', j') and (i'', j''), then if $I_2(i', j') < I_2(i, j) > I_2(i'', j'')$, then do nothing, else set $I_2(i, j)$ as 0.
Do this for each pixel.
(Note that, (i', j') and (i'', j'') both $\in \{(i-1, j-1), (i-1, j+1), (i+1, j-1), (i+1, j+1), (i, j-1), (i-1, j), (i, j+1), (i+1, j)\}$)
Let I_3 be the image after non-max suppression.

Step 4: Double Threshold

We use 2 thresholds, $\{t_1, t_2\}, t_2 > t_1$, such that all intensity value above t_2 are assumed to be definitely edges and are set to 1, all intensity values below t_1 are set to 0, all intensity in between t_1 and t_2 could possibly be an edge.

$$\begin{aligned} I_4(i,j) &= 1, \text{ if } (I_3(i,j) > t_2) \\ &= I_3(i,j), \text{ if } (t_2 > I_3(i,j) > t_1) \\ &= 0, \text{ if } (I_3(i,j) < t_1) \end{aligned}$$

The parameters for this step are: t_1, t_2

Step 5: Edge tracking by hysteresis

Based on thresholding's results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around (Let's say the 8 nearest neighbours) the one being processed is a strong one.
(The number of surrounding pixels to be considered can be thought of as a parameter for edge tracking, let's call this n_{et})