Visual Computing Lab 4

Horia Radu
k55592hr

1. Canny has two thresholds that control the edge thresholding process. What is their purpose?

The first (lower) threshold has a purpose that, if a pixel's gradient is lower than the low threshold, then its edge will be rejected by Canny.

The second (upper) threshold has a purpose that, if a pixel's gradient is higher than the high threshold, then its edge will be accepted by Canny.

And if a pixel's gradient is in between those thresholds, it will be accepted only if it is connected to a pixel with a gradient higher than the upper threshold

2. What is the purpose of the aperture parameter? What is the result of changing it from 3 to 5, 7, 9 or greater?

An aperture size of 9 or higher will throw an exception in the Canny function, it needs to be an odd number between 3 and 7, in my code.

This parameter is used in the smoothing phase of the image, to compute the edges of the Canny function.

Increasing this parameter will make the edge detection less sensitive to noise.

The number 3 will use the Sobel kernel for the edge detection, with no blurring and smoothing done, but anything above it will use the Gaussian kernel, and no previous blur on the image will be needed

So, if we use Canny with an aperture parameter of 3 we will need to blur the image before, so that we reduce the noise and get good results, but we can also use a higher aperture parameter, determining the width of the Gaussian kernel used in the process.

Increasing this parameter makes the threshold have a lower impact on the edge detection of images, thus we then will need higher threshold values.

Basically, if we increase this parameter, we will also need to modify the thresholds to get results, but the method will be more sensitive to each edge, making fewer changes based on the threshold.

3. The Hough transform has two parameters that specify the resolution of the accumulator. Their default values are 1 and π/180. What is the effect of increasing the first and reducing the second?

Those 2 parameters are Rho and Theta.

Every line detected by the Hough transformation is calculated using those 2 parameters, with Rho being the perpendicular distance of the line vector and Theta being the angle formed by this line.

Rho and Theta are both accuracies, distance resolution in pixels and respectively angle resolution in radians of the accumulator of the method.

Increasing Rho will make the method detect more lines

Reducing Theta will make the method detect more lines

Thus, increasing the first and reducing the second will make the

accumulator detect more lines, but them being less accurate and a lot of them overlapping each other.

4. The Hough transform has a pair of parameters that determine the minimum length of a line that can be accepted, and the maximum gap between two segments if they are to be considered part of the same line.

What is the effect of changing these values?

The minimum line length parameter will have an impact on what lines Hough transformation will return. It determines the minimum length of a line that will be accepted, the higher it is, less lines will be accepted and the ones below the parameter will be declined. Basically, lines smaller than this will not be discarded by the method.

The maximum gap between two segments relates to how picky the method will be on accepting and declining lines, depending on the gap of the two points that determine the line. The higher this value is, the more lines that are less “collinear” will be accepted. Having this too low might result in a small amount of lines detected, and having this too high will detect too many lines, everywhere, even where there shouldn’t be. Basically, how picky the method is for considering lines.

5. How close are the computed horizons to where you think the horizon should be? What might cause any discrepancy?

For me, I manage to compute all the horizons for the 3 images (horizon1.jpg, horizon2.png, horizon3.jpg) almost perfect to where the horizon actually is.

There are a lot of parameters which affect the detection of those horizons.

First, we need to have a good Canny edge detection for the image, and make sure that the horizon line is visible there.

Then, when drawing the lines on the image, based on the Hough lines transformation, we need to make sure that we are a little picky on those lines, including a minimum length for them.

What might cause a **discrepancy** for the detection of the horizon is the angle of the lines that we take in consideration, and how horizontal and vertical we want them to be, based on how the horizon is in the image.

Finally, I managed to detect the horizon using regression, but for this step we also need to make sure that the horizon line doesn’t get too curved or too straight.

To make it easier to experiment with the images and detection of the horizons, I used 3 thresholds for Canny edge detection, a short lines filter and a filter for how horizontal a line will be accepted as:

The values that I used for the images are:

horizon1.jpg

- Canny Threshold = 94
- Short Lines Filter = 108
- Lines Angle Filter = 10

horizon2.jpg

- Canny Threshold = 81

Short Lines Filter = 105
Lines Angle Filter = 20

horizon3.jpg

Canny Threshold = 52
Short Lines Filter = 10
Lines Angle Filter = 5