4. Edge and contour detection

Abstract
Among other possibilities, edges and contours are important structures in an image since they may facilitate characterising objects in a scene. Therefore, a good detection and representation of this information can be critical for problems such as estimating motion, detecting the presence of certain objects, localising them or even recognising which type of objects they are. In this lab we will see how to detect edges with the Sobel and Canny detectors, and also how to detect straight lines with a voting mechanism, the Hough Transform.

Keywords
Canny and Sobel edge detectors • Hough Transform

Contents

1. Sobel filter

▷ E_1 Run the provided code to apply the Sobel filter to image `cuadros.png`. As you can observe, the result corresponds to the image gradient along a single direction. Now,

- Write down the meaning of the gray level associated with the gradient values in the displayed output.
- Modify `testSobel()` to compute the gradient *only* in the other direction. Observe the result of both gradients (horizontal and vertical) for images `cuadros` and `lena`. Would you know in which direction the gradient has been applied *only* from these visualisations?
- Modify `testSobel()` to additionally compute the gradient *also* in the other direction and return the magnitude of the gradient. Check the result.

Binarise the magnitude of the gradient to get a *binary* edge map. ▷ E_4

Use the variable `bAddNoise` to chech the effect on the edge map of adding (or not) Gaussian noise to the image.

Find out the meaning of the optional paramter `mode` of function `sobel()`, and understand the expected behaviour for each its possible values. You do not have to analyse their respective effects.

2. Detector de Canny

Try now the Canny detector with different values of the standar deviation σ for the Gaussian filter that this detector applies as a first step. Compare the results:

- for images (cuadros and lena);
- with the Sobel fiter; and
- with/without noise (`bAddNoise`).

Write down your observations and interpretation. For instance, is the same σ good for both noisy and noiseless images?

Review the meaning of the two thresholds that Canny detector uses and analyse the result as a function of these parameters. ▷ E_2

3. Hough transform

Try on image cuadros the Hough Transform (HT) for straight lines (TH) as provided in the code. Notice that the HT is applied on the edges, so these have to be obtained first.

The first step in the HT itself consists of computing the accumulator space from a binary edge map. ▷ E_6

1. What is the function that we use to obtain this space, which parameters it has, and what is returned?
2. Identify in the code, which is the variable corresponding to the accumulator space, what its size is, and why.
3. In the displayed figure, what does each axis represent? What are the values in each element in the accumulator space? What characteristic traits of the HT can you notice? What do they correspond to?

Once the accumulator space has been computed, we can proceed to identify its peaks. ▷ E_3, E_5

1. What information do these peaks provide us?
2. What does their higher or lower values depend on?
3. Identify in the code which function we use to locate those peaks, which parameters it has, and what is returned.

Table 1. Four combinations of parameters of the Canny edge detector. The input image and the results of the detection in each case are given in Fig. 1. Your task is to relate them (E_2)

	σ	T_1	T_2
a	1	0.1	0.2
b	1	0.4	0.6
c	3	0.1	0.2
d	3	0.01	0.02

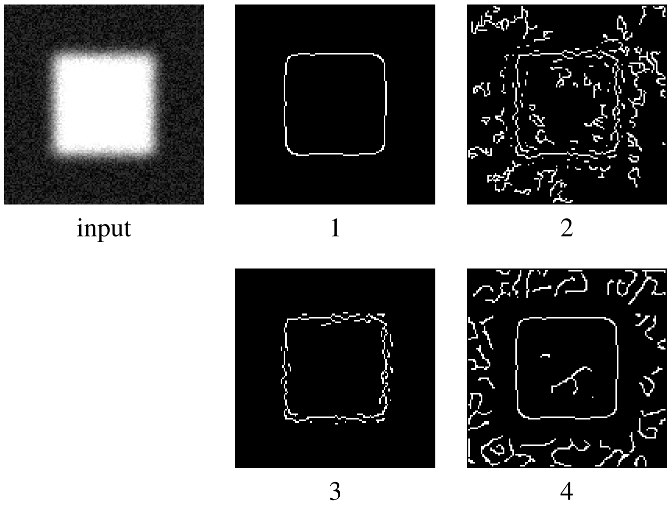

Figure 1. The Canny each detector has been applied with four different parameter combinatios (Table 1) to the input image

4. Study the code that draws the peaks on the accumulator space and that prints on the standard output their values and the corresponding polar coordinates (θ, ρ).

4. Exercises

1. Instead of using the function `sobel()` from module `filters` in `scipy.ndimage`, apply the Sobel detector through the corresponding convolutions. Check visually that the results of both convolutions agree.

2. Given the input image in Fig. 1, of size 128×128, and with gray levels in the (approximate) range $[0,1]$, a Canny filter has been applied, with each of the four combinations of the parameters (σ, T_1 y T_2) given in Table 1. Relate each combination (a, b, c y d) with each of the outputs of the detector (1, 2, 3 y 4) shown in Fig. 1, with a proper argument.

1	2	3	4

The Canny edge detector has been applied with the following call:

```
feature.canny(im, sigma=sigma,
  low_threshold=T1, high_threshold=T2,
  use_quantiles=False)
```

3. Enable the boolean variable `bRotate` to rotate the input image some degrees α. Run the HT for different orientations and check whether the same (number of) peaks are found in the accumulator (Hough) space in all cases. On the other hand, find out whether there is a relationship between the coordinate θ of the peak and the applied rotation α. Why?

4. Histograms of gradients have been used to solve object detection and object recognition problems[1]. Write a function `HOG(im, nbins)` which, given an image `im`, it returns an histogram of `nbins` bins of gradient orientations (*histogram of oriented gradients*) of the gray levels of the image.

5. (Optional) Write a function `displayLines()` to draw the peaks detected with the HT on the input image. Display the result with Matplotlib. Decide on which input parameters this function must have. Then, call it in the intended location within the provided code.

   Hint: The straight line equation used in HT is, as you know, $x\cos\theta + y\sin\theta = \rho$. In this case, we know θ y ρ (the coordinates of each detected and selected peak), and we only need to find out two particular points on the corressponding line, (x_1, y_1) and (x_2, y_2). You can use the ends of the image to set *one* of the coordinates of those points (either $x = 0$ or $y = 0$ or $x = W$ or $y = H$, for a $W \times H$ image), and then get the other coordinate by solving the equation. Once the four coordinates of the two points are known, we can draw the straight line as follows:

   ```
   plt.plot((x1,x2),(y1,y2))
   ```

6. (Optional, advanced) We have seen how to detect straight lines with the Hough transform, but we have not detected the actual *segments*. Think of or find out how to solve this and, if you are up for the challenge, implement and test some simple procedure.

[1] As you can read in this wikipedia entry, a work by Dalal and Triggs in 2005 was a decisive milestone for the development of this concept. Please, bear in mind that this was in the pre-deep-learning area to contextualize this achievement. The HOG descriptor is an illustrative example of how a relatively simple idea can, well applied, turn out to be very effective.