Exercise 6: Simple Features

02504 Computer vision

Morten R. Hannemose, mohan@dtu.dk, DTU Compute

March 10, 2022

Learning objectives

These exercises will introduce you to feature extraction. In this exercise you will write the code for the Harris corner detector as well as try the Canny edge detector.

Programming exercises: Harris corner detector

Image analysis is mostly a computational or numerical task, so we will go directly to the programming exercises today.

You will implement the Harris corner detector and apply it to the image `TestIm1.png` shown in Figure 1 to test that your implementation is correct. Furthermore you should test the Harris corner detector on some of the other images given in the data folder for this exercise to see

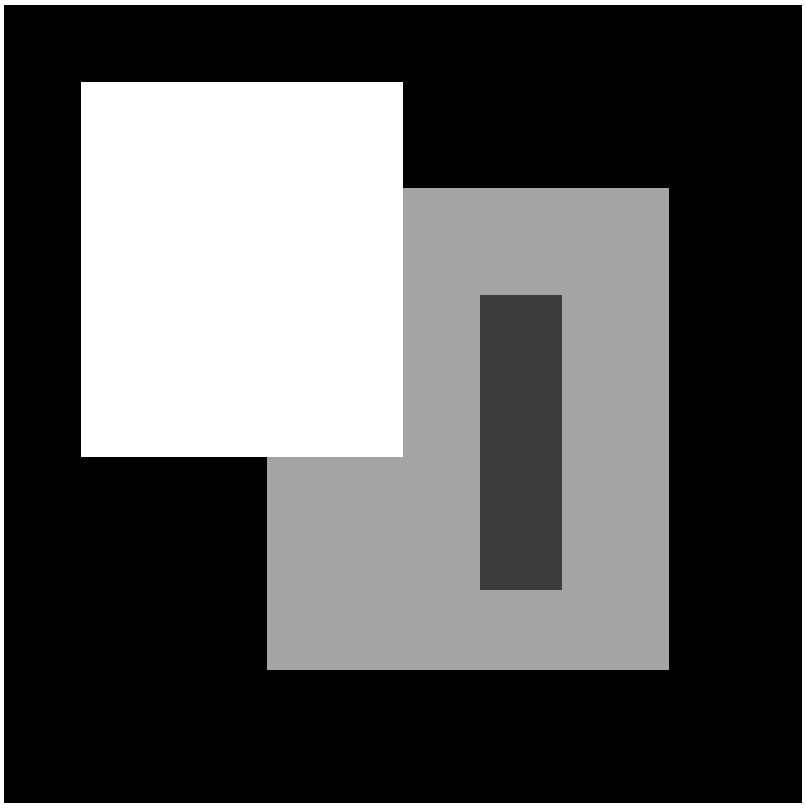

Figure 1: A test image for testing the Harris corner detector.

Exercise 6.1

To start with, create the function `g, gx = gaussian1DKernel(sigma)`, where `g` is the 1D Gaussian kernel, `gx` is the derivative of `g`, and `sigma` is the Gaussian width.

In this function, you have a choice: what length should my Gaussian kernel have? What is gained by setting the length to `sigma`, $2 \cdot$ `sigma`, or $6 \cdot$ `sigma`?

Exercise 6.2

Now create the function `I, Ix, Iy = gaussianSmoothing(im, sigma)`, where `I` is the Gaussian smoothed image of `im`, and `Ix` and `Iy` are the smoothed derivatives of the image `im`. The `im` is the original image and `sigma` is the Gaussian width.

Using the `g, gx = gaussian1DKernel(sigma)` function, how would you do 2D smoothing?
Tip: Using a 1D kernel in one direction e.g. x is independent of kernels in the other directions.

What happens if `sigma` $= 0$? What should the function return if it supported that?

Use the smoothing function on your test image. Do the resulting images look correct?

Exercise 6.3

Now create the function `C = smoothedHessian(im, sigma, epsilon)` where

$$\mathtt{C}(x,y) = \begin{bmatrix} g_\epsilon * \mathtt{Ix}^2(x,y) & g_\epsilon * \mathtt{Ix}(x,y)\mathtt{Iy}(x,y) \\ g_\epsilon * \mathtt{Ix}(x,y)\mathtt{Iy}(x,y) & g_\epsilon * \mathtt{Iy}^2(x,y) \end{bmatrix} \tag{1}$$

and $g_\epsilon * \ldots$ is the convolution of a new Gaussian kernel with width `epsilon`.

Use the smoothed Hessian function on your test image. Do the resulting images still look correct?

We use two Gaussian widths in this function: `sigma` and `epsilon`. The first one `sigma` is used to calculate the derivatives and the second one to calculate the Hessian. Do we need to use both? If you don't know the answer, start on the next exercise and return to this question afterwards.

Exercise 6.4

Create the function `r = harrisMeasure(im, sigma, epsilon, k)` where

$$\mathtt{r}(x,y) = a \cdot b - c^2 - \mathtt{k}\,(a+b)^2 \quad \text{where} \tag{2}$$

$$\mathtt{C}(x,y) = \begin{bmatrix} a & c \\ c & b \end{bmatrix} \tag{3}$$

Now return to the question from last exercise.
Tip: What happens to `r` if you set $\mathtt{sigma} = 0$ or $\mathtt{epsilon} = 0$? Take a look at the lecture notes, equations (4.6) and (4.7) and the equations in between. It is essential that $\mathtt{epsilon} \neq 0$, why is that?

Use the `harrisMeasure` function on your test image. Are there large values near the corners?

Exercise 6.5

Finally, create the function `c = cornerDetector(im, sigma, epsilon, k, tau)` where `c` is a list of points where `r` is the local maximum and larger than some relative threshold i.e.

$$\mathtt{r}(x,y) > \mathtt{tau}.$$

To get local maxima, you should implement non-maximum suppression, see Sec. 4.3.1 in the LN. Non-maximum suppression ensures that $\mathtt{r}(x,y) > \mathtt{r}(x \pm 1, y)$ and $\mathtt{r}(x,y) > \mathtt{r}(x, y \pm 1)$.

Use the corner detector on your test image. Does it find all the corners, or too many corners?

Programming exercises: Canny edge detection

Just like many other imaging operations the Canny edge detector is available in both Matlab and OpenCV. Instead of implementing it ourselves, let us start using someone else's implementation.

Exercise 6.6

Figure out how to run the Canny edge detector in your language and apply it to the two images `TestIm1.png` and `TestIm2.png`

Exercise 6.7

What is the effect of the threshold parameters in the Canny edge detector. Try them out specifically on `TestIm2.png`?