Filippo Ficarra: Lab 2 - Feature extraction and Optical flow

fficarra@student.ethz.ch, 22–938–062.

08/10/2023 - 15:39h

1 Introduction

In this lab we aim to extract features from images, such as corners, using Harris detection and match descriptors between two different photos of the same thing.

2 Harris corner detection

The idea of Harris corner detection is to analyze the change of intensity of pixels in a window. Using this idea we can basically identify three different regions:

- flat: there is no change of intensity in all the directions
- edge: there is no change of intensity in the direction of the edge
- corner: large change of intensity in all the directions

To perform this analysis we define a window $\boldsymbol{W}$ and we define the SSD error as:

$$E(u,v) = \sum_{(x,y)\in \boldsymbol{W}} \left[I(x+u, y+v) - I(x,y)\right]^2$$

using Taylor approximation we can approximate the error with:

$$E(\Delta x, \Delta v) \approx [\Delta x, \Delta y]\, M \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix},\ M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Note that I_y and I_y are respectively the partial derivative of the image with respect to x and y.

2.1 Image derivatives

Since images are discrete we need to compute some approximation of the derivative. The approximation used in this case is the following:

$$I_x = \frac{I(x+1,y) - I(x-1,y)}{2},\ I_y = \frac{I(x,y+1) - I(x,y-1)}{2}$$

We can exploit the convolution operations to compute these derivatives, using the following filters:

$$\text{filter}_\text{x} = \begin{bmatrix}[-0.5 & 0 & 0.5]\end{bmatrix},\ \text{filter}_\text{y} = \begin{bmatrix}[-0.5]\\[0]\\[0.5]\end{bmatrix}$$

This convolutions have been done in python in the following way:

Listing 1: Image gradients

```
filter_x = np.array([[1/2, 0, -1/2]])
filter_y = np.array([[1/2],[0],[-1/2]])

I_x = signal.convolve2d(img, filter_x, mode='same')
I_y = signal.convolve2d(img, filter_y, mode='same')
```

2.2 Gaussian blur and local-auto correlation matrix

The next step is to introduce blur in the image to make the detection more robust. Applying a Gaussian Blur filter prior to performing corner detection serves the purpose of decreasing image noise, thereby enhancing the outcome of corner-detection.

To do so we use a gaussian kernel g, thus the matrix M is the following:

$$M = g * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Furthermore to compute the response function C we will need just this 3 elements:

Listing 2: Local auto-correlation matrix elements

```
I_xx_blur = cv2.GaussianBlur(I_x**2,(5,5),sigma,borderType=cv2.BORDER_REPLICATE)

I_yy_blur = cv2.GaussianBlur(I_y**2,(5,5),sigma,borderType=cv2.BORDER_REPLICATE)

I_xy_blur = cv2.GaussianBlur(I_x*I_y,(5,5),sigma,borderType=cv2.BORDER_REPLICATE)

# 3. Compute elements of the local auto-correlation matrix "M"

g_xx = I_xx_blur
g_yy = I_yy_blur
g_xy = I_xy_blur
```

2.3 Harris response and corner detection

The Harris detector uses the following function to score the presence of corners:

$$C = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k(trace(M))^2$$

therefore if

- $\lambda_1 \sim 0$ and $\lambda_2 \sim 0 \implies C \ll 0$ and the region is flat, since the intensity of the pixels doesn't really change in that region
- $\lambda_2 \gg \lambda_1 \implies C < 0$ and we detect an horizontal edge
- $\lambda_1 \gg \lambda_2 \implies C < 0$ and we detect a vertica edge
- $\lambda_1 \sim \lambda_2$ and both large, $\implies C > 0$ and we detect a corner

Alternatively to use $\det(M)$ and $trace(M)$, we can rewrite the response function like this:

$$C = g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - k[g(I_x^2) + g(I_y^2)]^2$$

and then we can reuse the components derived above:

Listing 3: Harris response function

```
C = (g_xx * g_yy - (g_xy**2) - k * (g_xx + g_yy)**2)
```

In order to detect corners we need just to compare every entry of the response with a threshold and perform non-maximum suppression.