The deadline for this exercise is on Sunday 06.05.2018, 23:59

Harris Corner and Edge Detection

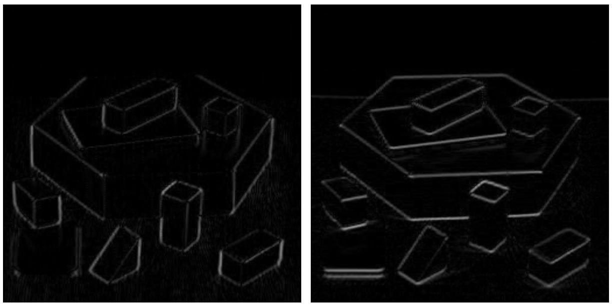
In this exercise, you are will implement a corner and edge detection algorithm based on the paper `A Combined Corner and Edge Detector` by Harris from 1988.

When you are finished with this exercise, compress the complete directory into a ZIP and upload it to StudOn. For groups of two, one member uploads the submissions and adds his/her partner to the exercise. In the week after deadline, you have to attend the exercise session in Huber-CIP and show your solution to one of tutors.

1 Harris Corner Response [5 Points]

At first the Harris Corner Response is computed for every pixel of the input image. The value of this function is then used in task 2 and 3 to extract corners and edges. Everything for this task has to be implemented in the function `harrisResponseImage` of `main.cpp`.

1. Compute an approximation of first spatial derivative in x and y direction (I_x and I_y respectively) using filters and store the results in `dIdx` and `dIdy`. You can use the OpenCV function `Sobel` or `getderivkernels` in combination with `filter2d`.



2. Compute the mixed products I_{xx}, I_{yy}, I_{xy} for auto-correlation with

$$I_{xx} = I_x^2$$

$$I_{yy} = I_y^2$$

$$I_{xy} = I_x I_y$$

   and store the results in `Ixx`, `Iyy`, and `Ixy`.

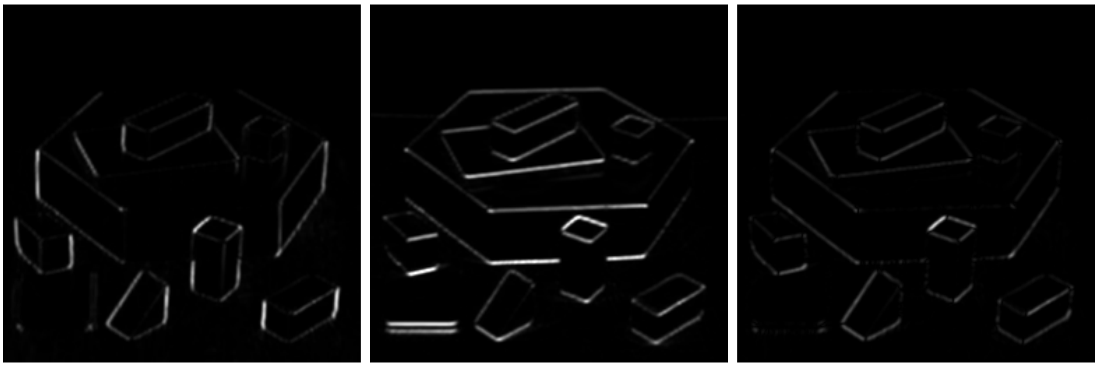
3. Convolve the mixed products with a zero mean Gaussian with $\sigma = 1$

$$A = I_{xx} \circledast G$$

$$B = I_{yy} \circledast G$$

$$C = I_{xy} \circledast G$$

   Use the OpenCV function `gaussianblur` and store the result in `A`, `B`, and `C`.



4. For each pixel, construct the structure tensor T and compute the Harris response R with

$$T = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$R = \text{Det}(T) - k\ \text{Trace}(T)^2$$

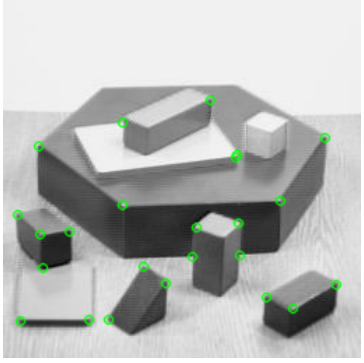
2 Corner Detection [5 Points]

Given the Harris response function R, stable corner points can be extracted by searching for local maximas and thresholding. Implement the function `harrisKeypoints` that creates a new keypoint for a pixel (x, y), if the following two conditions are met:

- $R(x, y) > t_h$, with $t_h = 0.1$
- $R(x, y)$ is a local maximum of R in the 1-neighborhood of (x, y) (8 checks in total).

New keypoints can be created and added to the result array like this:

```
KeyPoint kp(x,y,1);
points.push_back(kp);
```



3 Edge Detection [5 Points]

Similar to the corner detection, implement the function `harrisEdges` to identify edge points by checking the following conditions:

- $R(x, y) < t_e$, with $t_e = -0.01$.
- $R(x, y)$ is a local minimum in x **or** y direction.

If a points (x, y) is marked as edge, paint the result red:

```
res.at<Vec3b>(y,x) = Vec3b(0,0,255);
```