Exercise 2: Feature Extraction

Due: 23.11.2017

Please note that exercise 2 is long and may require all four sessions to complete. Don’t leave it all to the last day and don’t be afraid to ask for help from the assistants!

Visual features are attributes or characteristics of an image. Feature extraction is a common step used in many computer vision routines such as object detection, stereo correspondence, motion extraction, etc. In this exercise, you will explore three types of low-level features: edges, lines and corners.

1 Harris corner detection

The Harris corner detector finds corners in an image. Conceptually, corner detection can be thought of as an auto-correlation of an image patch. Consider a window which slides over an image patch. If the image patch is constant or "flat", then there will be little to no intensity changes in the window. If the image patch has an edge, then there will be no intensity changes in the window along the direction of the edge. If a corner is present, however, then there will be a strong intensity changes in the window regardless of the direction.

Implement a Harris corner detector. A code template is provided (harris.ipynb). Test your implementation on ~cvcourse/pics/CircleLineRect.png and ~cvcourse/pics/additional/zureth.png.

- **Derivatives**: Approximate the image gradients $f_x$ and $f_y$ by using either a Sobel operator or the first derivative of a Gaussian combined with smoothing. For convolution, use the built-in convolve function in the CImg library.

- **Second order moments**: Use the gradients computed above to determine the second-order moments $f_x^2$, $f_y^2$ and $f_x * f_y$.

- **Harris response**: Smooth your second order moments (i.e. with Gaussian smoothing) and compute the determinant, trace and finally the Harris response. Write a function to extract the local maxima of the Harris response (see code template) and to visualize your detected corners.

- **Rotation and scaling invariance**: Try different values for the rotate parameter in the main function provided in the code template to rotate the input image. Apply the Harris corner detector again and observe. Is the Harris detector invariant for scale changes?

Task: Apply your implementation of the Harris corner detector to the image of the two cuboids to find the corners.

Questions

- Is the Harris corner detector robust with respect to intensity changes in the image? Why or why not?

- Is the Harris corner detector robust with respect to rotation? Why or why not?
2 Canny Filter

The Canny filter determines edges based on searching for local minima or maxima in the approximated first-order derivative of a signal.

2.1 Theoretical Questions

The final version of the Canny filter is as follows:

\[
\frac{d}{dx} G(x) = \frac{d}{dx} \left( \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{x}{\sigma})^2} \right)
\]

a) Apply the Canny-Filter to the following profile in order to detect the edges. What is the magnitude of the filter response and the polarity at the edges? (draw a sketch).

\[
\text{mirrored } \frac{d}{dx} G\text{-Filter}
\]

b) Apply the Canny-Filter with non-maximum suppression for different values for \( \sigma \) (width of the filter). For the following profiles, draw the positions of the detected edges as a function of \( \sigma \).

c) Is it possible to detect lines (\( L|h|L \)) and edges (\( L|Hh \)) of different width and shape with a linear filter? Justify your answer.
2.2 Practical Exercise

Having seen how the Canny filter works in the 1-D case in the above theoretical exercise, we now move onto the 2-D case. In 2-D, the gradient magnitude is a measure for the strength of an edge, while direction of the gradient vector gives edge orientation. Edge detection can be done by thresholding the gradient magnitude, though this generally results in thick contours which are poorly localized. To thin the contours into single-pixel-wide lines, one should use non-maximum suppression. Pixels in the contour are kept as edge pixels only if its gradient magnitude is greater than that of its two neighboring pixels in the direction orthogonal to the edge, i.e. where the gradient is a maximum. A code template is provided at `canny.ipynb`

![Diagram](image)

Implement this edge detection approach in 2-D. You should produce the following images:

- **Gaussian Smooth Image:** Use the code provided in `canny.ipynb`.

- **Gradient Image:** Normalize the gradient values in order to cover the full range of grey values.

- **Orientation Map:** At each pixel, the orientation of an edge can be obtained from the gradient. Note that edge directions $\alpha$ and $\alpha + \pi$ are equivalent. The resulting directions can be coded as grey values.

- **Edge image according to the threshold criteria:** Each point whose absolute value of the gradient reaches a given threshold will be displayed.

- **Edge image after the Non-Maximum-Suppression:** Each point, which is fulfilling the threshold criteria and has a larger edge strength than its direct neighboring pixels in the direction orthogonal to the edge, will be displayed. Note that this is not the same as the non-maximum suppression in Harris corner detection!

Extract features from the image `~cvcourse/pics/zurlim.png` with your implementation. Appropriate parameters are suggested in the code template.

**Task:** Apply your implementation to the two cuboids from the previous filtering step.
3 Hough Transform for detecting straight lines

The Hough Transform is described in the script. Write a program to perform the Hough Transform. The program should be able to detect (straight) lines in a gray-scale picture.

**Input data:** To help you with the development, use `~/cvcourse/pics/additional/line2.png`. Later, the program should also be used with the picture `~/cvcourse/pics/houghtransform/circles_and_lines.png`.

**Computation of the transformation:** Choose a parameter space suited for description of lines. All the pixels in the input picture whose intensities lie in a certain interval should be taken into account for the transformation into the parameter space. Create an accumulation matrix to count the score for each point in the parameter space.

**Non-maximum suppression:** Re-use your non-maximum suppression code from Harris corner detection to extract the local maxima in the accumulation matrix.

**Back (Inverse)-transformation:** Find the points with maximum values in the accumulation matrix (e.g., through thresholding). These points of maximum should be taken into account for the back-transformation.

**Task:** Apply your implementation of the Hough transform to your Canny edge outputs of the two cuboids to find the straight lines.

**Questions**

- **Circles:** What would be a good parameter space to choose if we were detecting circles instead of straight lines? Assume that the radius of the circle is already known.