Middle East Technical University | Department of Computer Engineering

CENG 483

Introduction to Computer Vision

Fall 2023-2024

Take-Home Exam 2

Corner Detection with Harris Corner Detector

Due date: **12 December 2023, 23:59**

1 Objective

This assignment is designed to provide you with hands-on experience in key point detection using the Harris Corner Detector. You will be tasked with implementing various versions of the Harris Corner Detector and visualizing the detected keypoints in given images. The implementation process will be divided into distinct tasks, and you are required to present your code and discussions in a Jupyter notebook for each task.

2 Harris Corner Detector

The Harris Corner Detector identifies corners or keypoints by analyzing the change in intensity within an image patch after shifting that patch in eight different directions. Corners represent regions where any shift in any direction results in a significant change in intensity.

2.1 Variations

You will implement different versions of the Harris Corner Detector given in the list below:

1. Naive Formula + Uniform Weighting (without non-maximum suppression)
2. Naive Formula + Uniform Weighting
3. Taylor's Approximation + Uniform Weighting
4. Fast Implementation + Uniform Weighting
5. Fast Implementation + Circular Weighting
6. Fast Implementation + Gaussian Weighting
7. Fast Implementation + Gaussian Weighting + RGB Split

Preliminary information is available in `07_harris.pdf`. Naive formula is given in slide 24, Taylor's approximation in 29, and fast implementation in 51-52.

The *circular weighting* is the same as uniform weighting, but the pixels that don't satisfy $x^2 + y^2 \leq r^2$ are ignored, where x and y denote the location of the pixel within the filter (taking the center of the filter as origin) and $2r + 1 = \text{window\_size}$. In a 3×3 filter, this corresponds to a plus sign.

In *RGB split*, the color channels of the image (RGB) will be split, and the corner detector will be run on each channel separately until the non-maximum suppression part. The non-maximum suppression and the selection of top-10 corners will be run among all channels.

2.2 Jupyter Notebook Instructions

For each task in the Jupyter notebook, you are expected to do the following:

- Display all input images (3 original, 3 rotated) along with their top-10 corners that achieve the highest score (after applying non-maximum suppression unless otherwise specified).
- Print the average time to process one image.
- Include a brief discussion about the pros and cons of the given approach.

3 Restrictions and Tips

- Your implementation must be in Python 3.
- You must use `numpy`. However, you can use other tools to convert images into `numpy` arrays and to apply convolution (e.g. `scipy convolve2d`).
- The implementation of the corner detector must be your own; the usage of functions such as `cv2.cornerHarris` is **forbidden**.
- Keep in mind that your code will be subject to manual inspection. Ensure that your code is readable and adequately commented.
- The correctness of your implementation is crucial. Your implementation may be tested with random configurations.

4 Submission

- **Late Submission:** As in the syllabus.
- Both the code and discussions should be in the given Jupyter notebook, which should be uploaded to ODTUClass in a compressed archive file whose name is `notebook.tar.xz`. The archive should contain `notebook.ipynb` at its root directory. You can use the following command to create this archive: `tar cJf notebook.tar.xz notebook.ipynb`

5 Regulations

1. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.
2. **Newsgroup:** You must follow the course web page and ODTUClass (odtuclass.metu.edu.tr) for discussions and possible updates on a daily basis.