UNIVERSITÄT DES SAARLANDES
Prof. Dr.-Ing. Eddy Ilg
Computer Vision and Machine Perception Lab
ilg@cs.uni-saarland.de

November 12, 2023

3D Computer Vision

Assignment 1: Feature Detection, Description and Matching

Submission Deadline: **Dec 1st 8am**

Feature extraction forms the basis of 3D reconstruction and has demonstrated remarkable success across various applications, such as object recognition, image stitching and visual mapping. In this exercise, your initial task is to acquaint yourself with feature extraction methods. You will be implementing SIFT[1] and ORB[2] feature extractors. The total points alloted for these tasks are 100. As a 20 point bonus task, you can compare the descriptors with the deep-learning model SOSNet [3].
Submission Requirement: Please make sure you store all the result outputs in each task directory as provided already in the script templates.
Privacy Note: Please note that you are not allowed to publish the solutions to any of the excercises publicly.
Conventions: Note that torch and numpy are very similar, but not exactly the same. As torch is used in deep learning applications, we implement all exercises in torch.

Part 1 - Detection

(a) Follow the instructions in the `task1a/harris.py` file to implement a Harris keypoint detector.

- Q1 [10 points]: First, You need to finish `compute_score()` to compute the Harris score for a given image. You can use the jupyter notebook to visualize your scores.
- Q2 [10 points]: Then, in `detect_keypoint()` you need to detect the keypoints based on the score and remove adjacent points using non-maximal suppression. Use the jupyter notebook to visualize your results. Finally, run `harris.py` to produce the output file for submission.

(b) Follow the instruction in the `task1b/blob.py` folder to implement a blob detector on a single pyramid level.

- Q1 [10 points] Implement the convolution kernel $\sigma^2 \Delta n_\sigma$ in function `convolution_kernel()` that needs to be applied to the image. You can visualize your kernel with the jupyter notebook. It should look like the one shown in Figure 1.
- Q2 [10 points]: Implement the keypoint detector in the function `detect_keypoint()` for a single pyramid level. Use a non-maximum suppression with window size $3 \times 3 \times 3$ in location and scale space. Finally, run `blob.py` to produce the output file for submission.

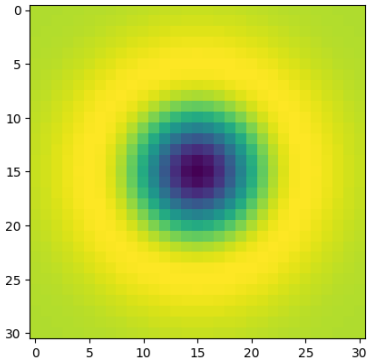

Figure 1: Example convolution kernel for the blob detector for $\sigma = 5$.

Part 2 - Feature Extraction and Matching

(a) To establish correspondences, you will need a matching operation that compares keypoints between two frames and finds the best match.

- Q1 [15 points]: You will find a skeleton implementation for a matching operation in the file `task2a/match.py`. Implement neareast neighbor matching with the first and second Lowe tests and forward-backward consistency check. Use the descriptors from the files. You may visualize your matches with the jupyter python notebook. Finally, run `match.py` to produce the output file for submission.

(b) Implement the rotated BRIEF descriptor in script `task2b/brief.py`. Please use the given keypoints in the folder.

- Q1 [5 points]: Implement the function `pattern()` to produce a type 1 BRIEF pattern for patches of 17×17 size. You can visualize it using the jupyter notebook.
- Q2 [20 points]: Implement the rotated brief descriptor. You can visualize results with the jupyter notebook. Finally, run `brief.py` to produce the output file for submission.

(c) Implement the hog descriptor in `task2c/hog.py`. Please use the given keypoints in the folder.

- Q2 [20 points]: Implement a rotated hog descriptor. You can visualize results with the jupyter notebook. Finally, run `hog.py` to produce the output file for submission.

[Bonus] Part 3 - Comparison of Feature Extractors

(a) Create a benchmark to compare ORB, SIFT and SOSNet [20 Points]:

- The SOSNet model is provided in the `task3` folder. Note that it operates on 32×32 patches. To perform matching with scale and rotation, you will have to extract scaled and rotated patches from the images. Use the SOSNet-notredame model.
- Take the `NotreDame1.jpg` image and rotate it 90 degrees clockwise to produce a second image. Apply some augmentations to simulate illumination changes.
- Use the OpenCV ORB and SIFT implementations. Run ORB, SIFT and SOSNet on the given images (use SIFT keypoints for SOSNet). Create a jupyter notebook to visualize the results of all three methods and compute and compare their accuracy from the known ground truth transformation. Include your jupyter notebook in your submission.

Best of Luck!

References

[1] I. Rey Otero and M. Delbracio, "Anatomy of the SIFT Method," *Image Processing On Line*, vol. 4, pp. 370–396, 2014. `https://doi.org/10.5201/ipol.2014.82`.

[2] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[3] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, "Sosnet: Second order similarity regularization for local descriptor learning," 2019.