COMPUTER VISION - LAB 4

Computer Vision 2023,
P. Zanuttigh, G. Rizzoli, D. Shenaj

Topics: Keypoints, Descriptors and Matching
Goal: Fix the corrupted regions in the provided image using data from patches.

Write a C++ software which includes methods that:

1. Load the corrupted image and the patches from one of the provided datasets.
    a) Different sets have different level of complexity, no need to solve all of them.
    b) The "_t" patches have been subject to some transformations that make the problem more challenging.
2. Extract SIFT features from the image (SIFT features requires OpenCV≥4.4)
3. Extract SIFT features from the patches (SIFT features requires OpenCV≥4.4)
4. For each patch
    a. Compute the match between the image and patch features extracted in (2). For this, OpenCV offers you the `cv::BFMatcher` class. Remember to use L2 distance for SIFT (the Hamming distance is good for ORB).
    b. Refine the matches found above by selecting the matches with distance less than *ratio * min_distance*, where *ratio* is a user-defined threshold and *min_distance* is the minimum distance found among the matches.
5. You can assume the images and patches are linked together by an affine transform, using the refined matches, find the transformation between the images. To this end, you can use the RANSAC algorithm. The set of inliers can be computed by using the `findHomography()` function, with `CV_RANSAC` as the third parameter (hint: the inliers can be retrieved by using the mask argument).
6. Using the found homographies overlay the patches over the image in order to fix the corrupted regions.
7. This is the baseline assignment, see the second page for additional suggestions for extra features allowing to improve your mark.

Optional steps

1. Manually implement a simplified RANSAC and affine transform estimation following the trace on the slides.
2. Acquire your own images (you can take an image, delete some areas and extract patches covering them but larger to allow for the matching, you can use any image editing software, e.g., GIMP).
3. Try different feature descriptors from OpenCV or other libraries (e.g., ORB).
4. Use some blending/mixing techniques to better fill the regions.
5. Try to use multiple images and all the patches together and automatically guess which images are linked to which patches.
6. You can also propose any algorithm you can figure out different than the proposed one to perform the task.
7. Use other techniques for the matching, e.g., template matching or machine learning based approaches.
8. Any additional idea is welcome!