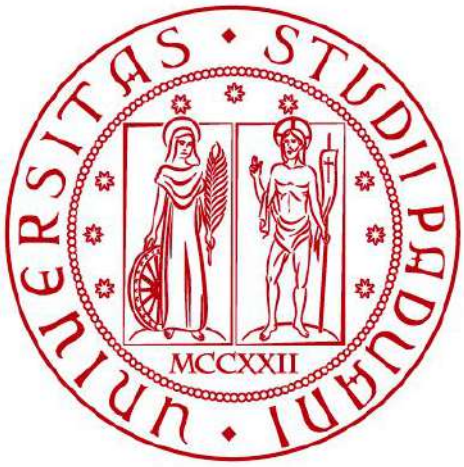


UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITA DEGLI STUDI DI PADOVA

Master Degree of ICT For Internet and Multimedia

Cybersystems Track

Computer Vision

Spring 2023, HOMEWORK #4

Due 14 of June 2023

Ceren Yılmaz Gülten 2080637

1 Keypoints, Descriptors and Matching

The goal of the project, feature extraction from the corrupted image and features, matching these features and fill the corrupted part of the image with the suitable patch. In the main steps, for the feature extraction SIFT is used and for the matching BFMatcher- NORM L2 is used as required. And for the filling and matching the related matches RANSAC algorithm is used. The details of the each code block explained in the code via a comment lines. As a dataset, star wars dataset is chosen from given dataset options.

Figure 1: Corrupted Images with Keypoints

The SIFT method is used as a feature extraction method as specified in the homework. The special function of SIFT for the Open-CV used detect and compute the features in the image. SIFT's primary goal is to identify distinctive and robust local features that can be matched across different images, even under varying scales, rotations, and changes in viewpoint. In the next part of the project, the property of working properly for varying scales, rotations and changes in viewpoint can be shown. To describe each keypoint, SIFT computes a unique feature vector called a descriptor. The descriptor captures the local appearance and orientation information of the image patch surrounding the keypoint. It is robust to changes in illumination, rotation, and viewpoint.

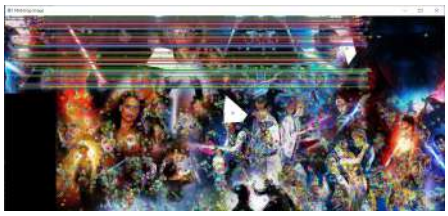
The same procedure of extracting and drawing features (keypoints) are done for the each patch inside of the loop. For the match the patch keypoints and corrupted image keypoints, BFMatcher is used. Brute-Force Matches is take the one feature from the first set and matched with the other images' all keypoints. The Bruce-Force Matches uses the distance calculation and NORM L2 parameter changes the this calculation. For the SIFT operation NORM L2 is the best parameter, for ORB NORM HAMMING is the best. For the each matching knnmatch is used too. knnMatch function from the cv::BFMatcher class performs a k-nearest neighbor search to find the best matches between the descriptors of the patch and the image. It takes the patch descriptors, image descriptors, the matches vector, and k as arguments. In this case, k is set to 2, meaning it will find the two best matches for each patch descriptor. With the

another for loop, we need to check the if the distance is correct for the match. We are doing this with refining matches based on the threshold ratio. The code iterates over the matches and checks if the distance of the first match is smaller than a certain ratio multiplied by the distance of the second match. If it is, the first match is considered a good match and added to the refinedMatches vector.

The other step is RANSAC and fill the corrupted areas with patches. The RANSAC algorithm is used to find the transformation between the patch and the image based on a set of refined matches. Before the RANSAC applied, findHomography function from OpenCV to estimate the homography matrix between patchPoints and imagePoints. The homography represents the transformation that maps points from the patch to the corresponding points in the image. In this case, cv::RANSAC is used as the method for robustly estimating the homography by rejecting outliers. The Open CV function RANSAC implemented directly. After that, with the mask and add operation the fills are completed.

Figure 2: First Patch Keypoints and Match Between Keypoints

Figure 3: First Patch Matched and Fill the Corrupted Area



(a) Second Patch Keypoints and Match Between Keypoints

(b) Third Patch Keypoints and Match Between Keypoints

(c) Fourth Patch Keypoints and Match Between Keypoints

Figure 4: Remain Patches Keypoints and Match Lines Between Keypoints

Figure 5: All Corrupted Areas Filled with Proper Patches

In the normal patches, there is no need to use additional blending or mixing methods. Because, it adds the patches almost perfectly.

In dataset, the transformed version of the patches are working properly without any changes on the code. The reason of this SIFT already, detects the different sizes, alignments and positions of the image. The other one is warpPerspective, the warpPerspective do the geometric transformation of the image. Since, the images are not too noisy. These two already enough to get the correct results for the transformed patches.

Figure 6: First Transformed Patch Keypoints and Match Between Keypoints

Figure 7: All Corrupted Areas Filled with Transformed Proper Patches

The only problem is, the corners of the patches can be observed after the filling the corrupted areas. For this reason, additional blending?mixing operations can be used.

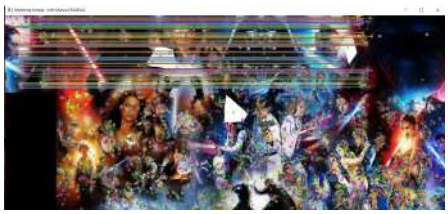
2 Optional Steps

Manual RANSAC and Affine transform part, the concept of these two applied to the code. In the RANSAC part, it is select 3 random features and calculate the best affine matrix based on selected features. If matches are consist of with the selected threshold distance, it counted as inliners. The iteration continues until the selected iteration number. At the end, the highest affine matrix value is selected as best option. In this part, rather than warpPerspective, warpAffine is used.

(a) First Patch Keypoints and Match Between Keypoints with Manual RANSAC

(b) First Patch Matched and Fill the Corrupted Area with Manual RANSAC

Figure 8: Manual RANSAC



(a) Second Patch Keypoints and Match Between Keypoints with Manual RANSAC

(b) Completed Image with Manual RANSAC

Figure 9: Manual RANSAC

With the Manual RANSAC and Affine Transform, the results give the proper results. The implementation is working properly.