University of Padova
ICT for Internet and Multimedia
Course of Computer Vision - Academic Year 2022-2023

Final Project Report

Computer Vision, Final Project Lab 4:

Keypoints, Descriptors and Matches in order to reconstruct corrupted images

Candidates:

Cecilia Rossi

Student ID: 2091484

Instructors:

Prof. Pietro Zannutigh

Introduction:

The aim of this project was applying different combinations of features description algorithms and matching strategies in order to reconstruct a series of corrupted images, by trying to compute the correct relationship with their missing patches.

This was achieved, firstly, by testing several methods, such as SIFT, SURF and ORB, to separately calculate the keypoints and descriptors on both the corrupted images and their patches, and then use the obtained results to find eventual matches between them.

Once the various matches were computed, they were used to estimate the mapping parameters defining the relationship between images and their patches, assuming that images and patches were linked together by an affine transform, through the RANSAC algorithm.

Finally, I handled the found homography parameters to overlay the patches over the image, in order to fix the corrupted regions and reconstruct it.

All the steps and operations above-mentioned were carried out by exploiting the build-in methods of the C++ extension of OpenCV software library.

Datasets:

The project data consisted of five different directories (*scrovegni*, *pratodellavalle*, *starwars*, *venezia* and *international*), in which the following files were contained:

- *image_to_complete.jpg*: the corrupted image, where some regions were missing, substituted by white areas;
- *patch_i.jpg* (with i = 0,..,3 with the only exception of pratodellavalle, that could count only on 3 patches): the image patches, clear and not subjected to any translation/rotation/flip;
- *patch_t_i.jpg* (for the same i indexes as before): the "challenging" patches, once noise and some transformations (rotations and flip over the axis) were applied to them;
- a file with the name of the corresponding directory, containing the image without corruption;

In the code, I approached the problem from two points of view, letting the user choosing from command line between:

- considering a single dataset at time;
- or analyzing all the images and all the patches together;

This selection would not be presented to the user if, at the beginning of the code, he selected the "challenging" patches. This was done merely for simplification reasons.

In fact, the decision to try to combine together all the datasets was only carried in order to see if the methods, implemented in the code also for the single datasets, would have been

sufficient to detect the correct patches associated to each corrupted image, or if, instead, further automatic recognition steps would have been necessary.

It was found that, if the patches were not associated to the corrupted image, the number of matches found by all the descriptor and matcher combination would have been too low in order to compute an homography, and therefore no correspondence would be identified.

The only exception was represented by the *international* dataset. For that specific corrupted image an unexpected number of matches were found between the image to complete and wrong patches, corresponding instead to different images, making it impossible to correctly reconstruct the image, if we compared it to all the patches at the same time.

Methods:

The code is implemented so that, at the very beginning, the user is asked to select, from command line, if he intends to use the "easy" patches, the ones without any noise or transformation, or the "challenging" ones.

In the first case, the user is later allowed to analyze all the datasets together, or otherwise focusing on a single image and its patches. Then, the user is able to select the type of descriptor algorithms that he intends to use for the computation of keypoints and image descriptors between:

- ***SIFT***
- ***SURF***
- ***ORB***

The keypoints and descriptors obtained with the previously selected method are later given as input to two matching strategies:

- ***BFMatcher*** (Brute Force Matcher);
- ***FLANN-based Matcher*** (a machine learning based matcher, that relies on fast nearest neighbor);

Actually, because of incompatibility between the two techniques, the Flann-based method is not applied if the selected descriptor is ORB, otherwise some software errors occurred.

In the second situation, so if the user decided to make the task more challenging by selecting the transformed patches, he would been able to choose only between SIFT and SURF, and the matches computed from the resulted keypoints would been obtained by testing only the BFMatcher.

Since both SIFT and SURF are scale and rotation invariant methods, they were able to find the right correspondences between the images and its patches even if were using a rotated and noisy version of the patches.

The only issues were meet if the patches were flipped on one or both the axis. For this reason, I inserted a control code that, if the number of matches found was inferior to a

certain threshold (manually fixed to three, which is the minimum number of matches able to compute an homography with the OpenCV implementation of RANSAC), a flip would have been applied to the patch and the keypoints detection would have been repeated.

The flipping operation could be reiterated, if the initial condition endured, for a maximum of 3 times, applying at every iteration a different flip on the patch (the first time along the y-axis, the second time on the x-axis and finally on both).

Finally, in every occasion we refined the matches found by selecting only the matches with a distance inferior to the product between a certain ratio and the minimum distance found among the matches. I used a ratio equal to 0.4 when using SIFT and SURF, while I preferred a higher value (0.85) when applying ORB (otherwise too many matches would have been discarded).

Results, Faults and Problems:

In order to complete the task, I focused on different

1) Results obtained by using different Image Descriptors

The number of matches found for each dataset, with all the combination of descriptors algorithms and matcher, are reported in the following table:

Dataset Name	*Descriptor Name*	*Number of Keypoints*	*Number of Matches*
Scrovegni	SURF	1th patch: 603 2th patch: 158 3th patch: 127 4th patch: 439	1th patch: 365 2th patch: 79 3th patch: 70 4th patch: 240
	ORB	1th patch: 383 2th patch: 251 3th patch: 149 4th patch: 319	1th patch: 9 2th patch: 18 3th patch: 19 4th patch: 58
	SIFT	1th patch: 727 2th patch: 142 3th patch: 187 4th patch: 696	1th patch: 480 2th patch: 90 3th patch: 112 4th patch: 472
Prato della Valle	SURF	1th patch: 549 2th patch: 501 3th patch: 579	1th patch: 270 2th patch: 269 3th patch: 338
	ORB	1th patch: 447 2th patch: 419 3th patch: 424	1th patch: 22 2th patch: 52 3th patch: 21

Starwars	SIFT	1th patch: 402 2th patch: 420 3th patch: 662	1th patch: 211 2th patch: 231 3th patch: 425
	SURF	1th patch: 113 2th patch: 163 3th patch: 316 4th patch: 459	1th patch: 78 2th patch: 87 3th patch: 166 4th patch: 284
	ORB	1th patch: 124 2th patch: 227 3th patch: 410 4th patch: 379	1th patch: 21 2th patch: 17 3th patch: 25 4th patch: 19
Venezia	SIFT	1th patch: 207 2th patch: 172 3th patch: 360 4th patch: 550	1th patch: 121 2th patch: 121 3th patch: 236 4th patch: 354
	SURF	1th patch: 375 2th patch: 627 3th patch: 343 4th patch: 410	1th patch: 194 2th patch: 362 3th patch: 181 4th patch: 177
	ORB	1th patch: 347 2th patch: 408 3th patch: 290 4th patch: 261	1th patch: 14 2th patch: 28 3th patch: 16 4th patch: 17
International	SIFT	1th patch: 535 2th patch: 511 3th patch: 284 4th patch: 258	1th patch: 356 2th patch: 322 3th patch: 193 4th patch: 151
	SURF	1th patch: 269 2th patch: 149 3th patch: 67 4th patch: 18	1th patch: 150 2th patch: 102 3th patch: 33 4th patch: 7
	ORB	1th patch: 278 2th patch: 134 3th patch: 45 4th patch: 12	1th patch: 20 2th patch: 21 3th patch: 35 4th patch: 26
	SIFT	1th patch: 282 2th patch: 152 3th patch: 70 4th patch: 16	1th patch: 178 2th patch: 90 3th patch: 48 4th patch: 37

Table 1: Number of Matches computed with BFMatcher between the corrupted image and its patches (the easy ones), for all the dataset and with all the descriptors algorithms

From the reported results, we can easily notice how SIFT appears to be the overall best feature descriptor, working relatively well also on the most discriminant image, the *international* dataset. But, SURF still returns satisfying outcomes, with much less computational time, with the only exception of the international dataset. For this last dataset, in fact, although able to reconstruct the image in a correct way (we only observe a faintly visible dotted line on the borders of the third patch), it doesn't enable the matcher to detect a so larger number of matches.

By visually inspecting the ORB outcomes, instead, I witnessed its very poor performances. At the beginning, I used the same value for the ratio test applied in order to discard the false matches, but the number of remaining matches was always close to zero. So, I tried to use a higher ratio value (= 0.85), but in this way a larger amount of the matches was still incorrect, leading to a distorted reconstruction of the original image.

2) Results obtained by using different Matching Strategies

In addition to BFMatcher, in the code I also tested a FLANN-based matcher, where Flann stays for Fast Library for Approximate Nearest Neighbors, so an algorithm that implements the matches detection using an optimized version of fast nearest neighbor. I didn't include the results obtained by using this method, since they were very similar to the ones I got by applying the brute force (BFMatcher) approach, with only differences of some units in the number of matches computed.

3) Results obtained by using the "Challenging" Patches

The third angle from which I approached the task was centered on comparing the overall results obtained on the "challenging" patches, with respect to the overcame previously obtained by applying the algorithm to the normal patches.

For simplicity, in the following tab I only reported the results obtained on the transformed patches for every dataset, by applying SURF as the feature descriptor and using the Brute Force Approach in order to detect the matches.

Dataset Name	Number of Keypoints	Number of Matches
Scrovegni	1^{th} patch: 594 2^{th} patch: 167 3^{th} patch: 135 4^{th} patch: 435	1^{th} patch: 354 2^{th} patch: 71 3^{th} patch: 71 4^{th} patch: 228
Prato della Valle	1^{th} patch: 300 2^{th} patch: 692 3^{th} patch: 137	1^{th} patch: 21 2^{th} patch: 692 3^{th} patch: 74
Starwars	1^{th} patch: 237 2^{th} patch: 315 3^{th} patch: 583 4^{th} patch: 904	1^{th} patch: 22 2^{th} patch: 32 3^{th} patch: 74 4^{th} patch: 58
Venezia	1^{th} patch: 52	1^{th} patch: 8

	2th patch: 115 3th patch: 44 4th patch: 58	2th patch: 10 3th patch: 7 4th patch: 5
International	1th patch: 38 2th patch: 47 3th patch: 10 4th patch: 4	1th patch: 20 2th patch: 11 3th patch: 8 4th patch: 3

Table 1: Number of Keypoints and Matches computed for every dataset, by using the challenging patches, with SURF descriptor and BFMatcher strategy

As we could have expected, both the number of keypoints (& descriptors) and the number of matches individuated (after applying the discarding of the wrong matches with the ratio test) decreased. This could be due to the presence of noise affecting the patches, but also to the fact that, even if SURF (and SIFT too) is acknowledged as a scale and rotation invariant method, in the practice there will always be some not ideal factors affecting its functioning.

4) Visual example using the Scrovegni Dataset

I) Corrupted Image and its Patches (the "easy" ones on the first row, while the "challenging" ones on the second row

Figure 1: Corrupted Image and its Patches

II) Matches found using SURF vs SIFT vs ORB

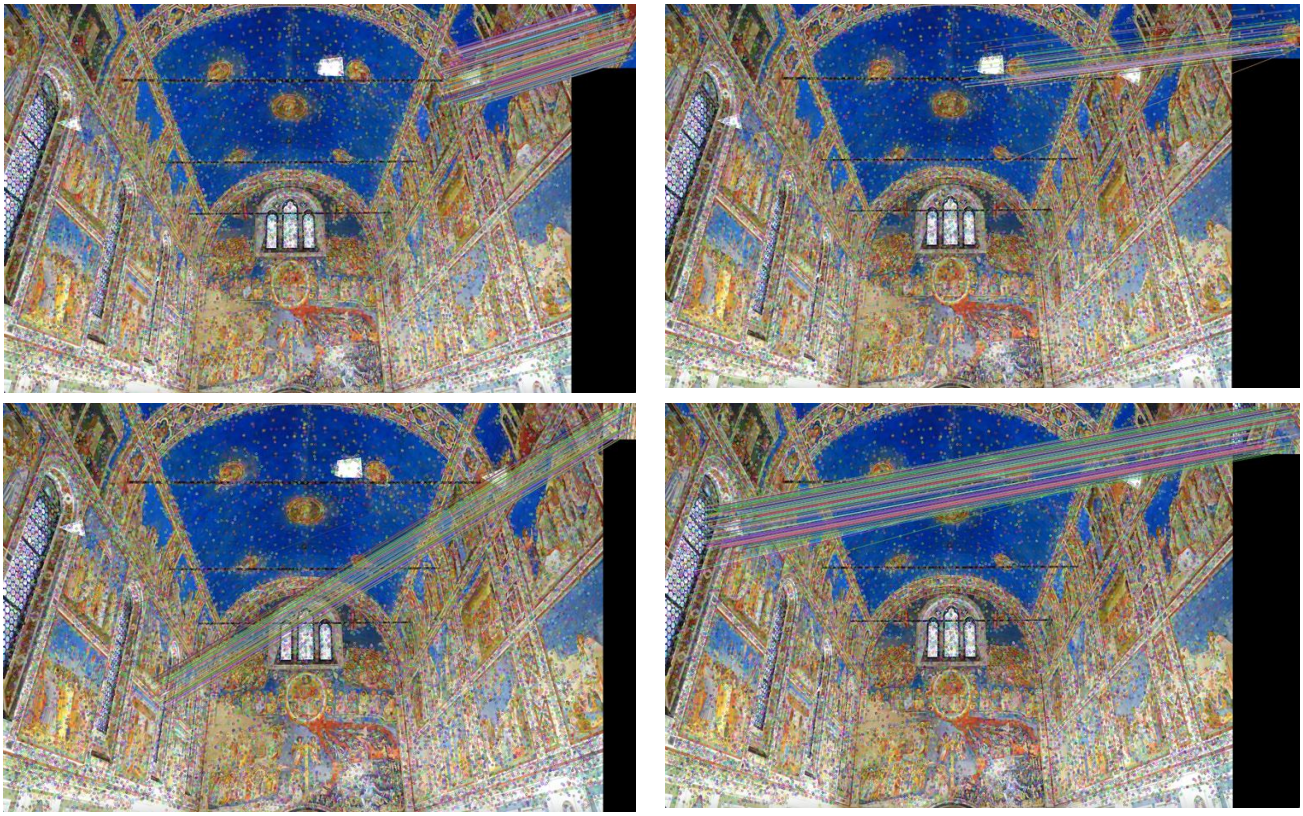

Figure 2: Matches found with SURF

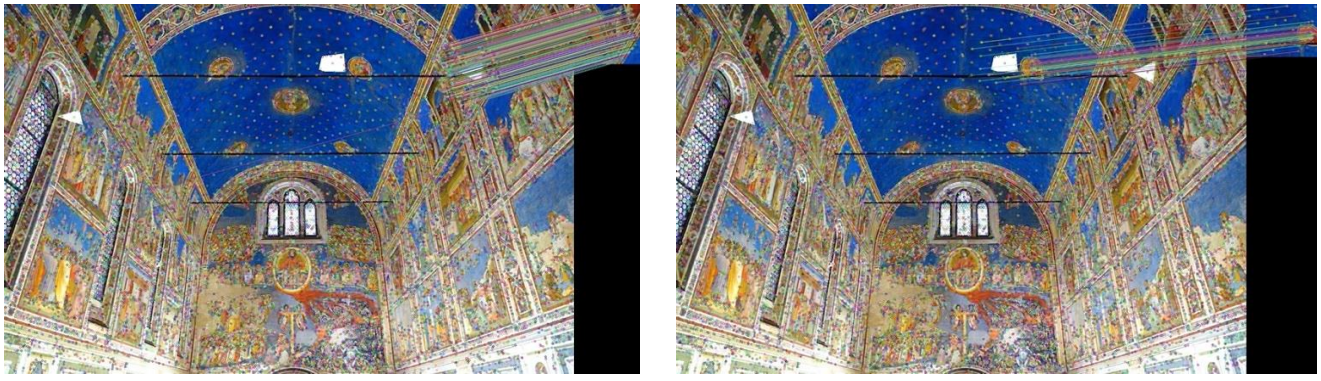

Figure 3: Matches found with SIFT

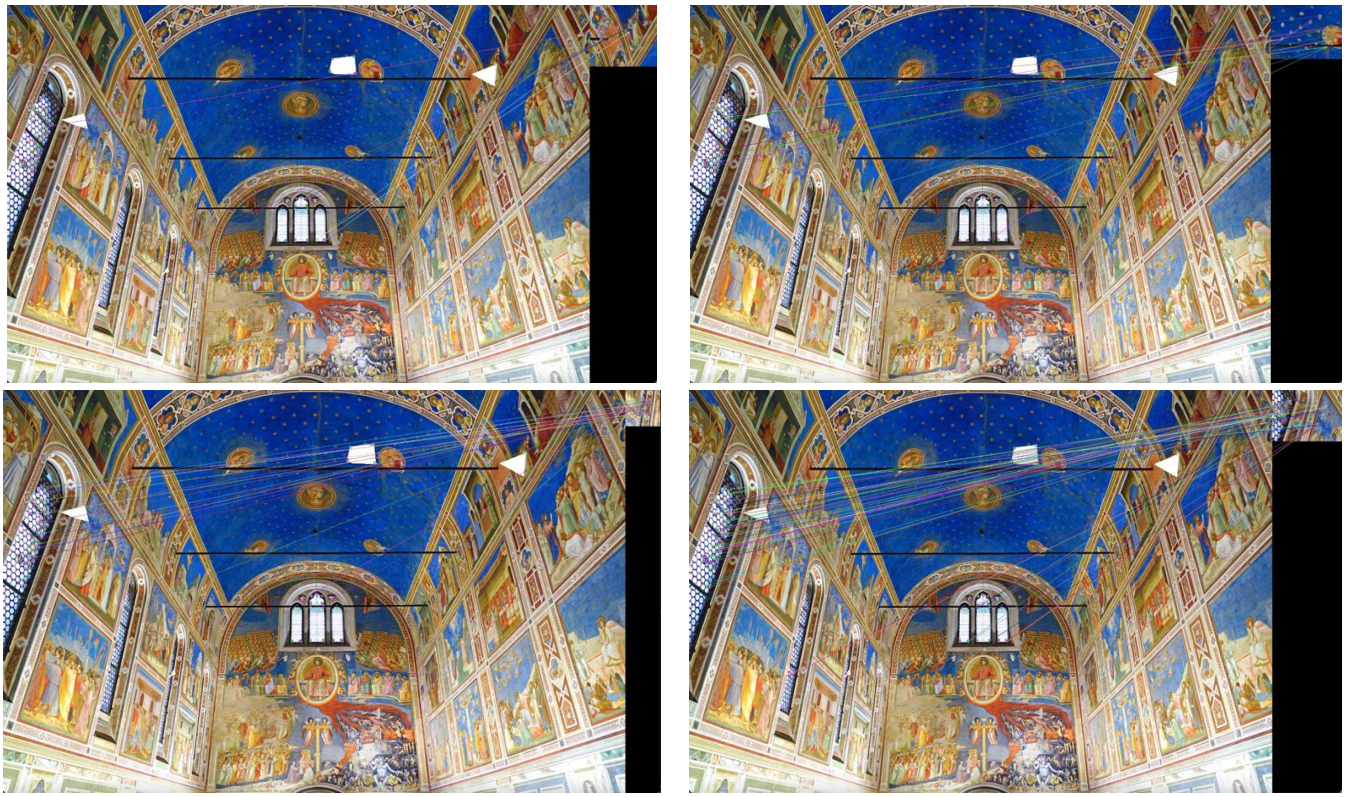

Figure 4: Matches found with ORB

III) Reconstructed Images using SURF, SIFT and ORB

Figure 5: Reconstructed images using, respectively, SURF, SIFT and ORB

IV) Matches found using SURF with the "challenging" patches

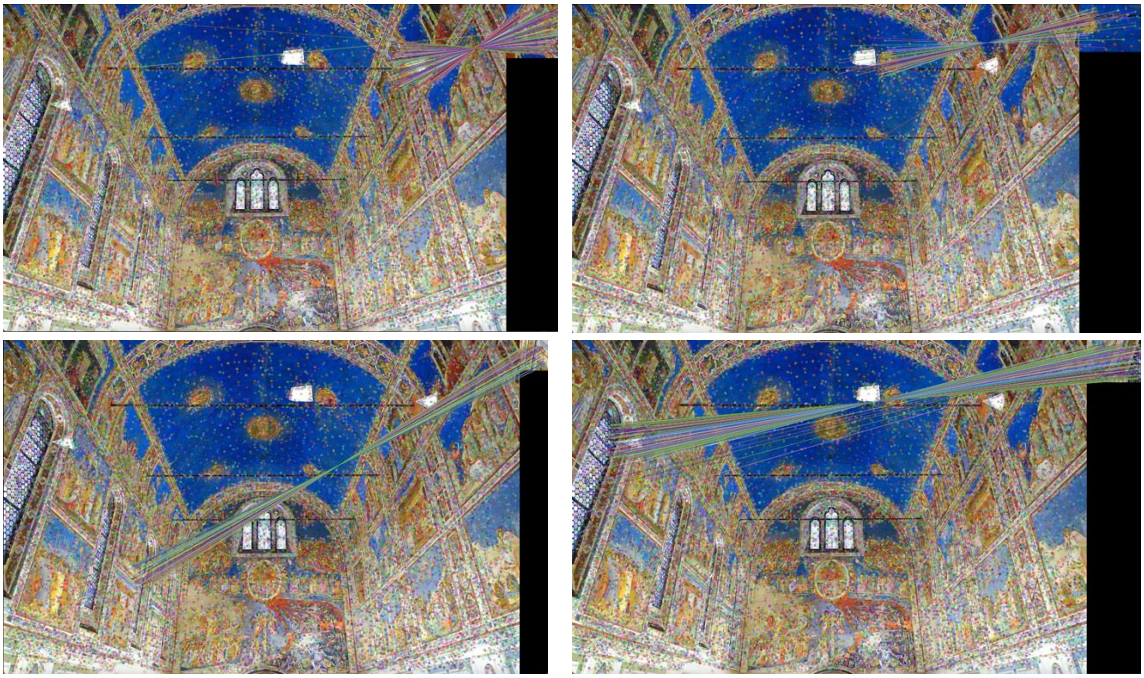

Figure 6: Matches found by applying SURF to the transformed patches

V) Reconstructed Image with SURF and the "challenging" patches

Figure 7: Reconstructed Image using the "challenging" patches

From the reported images, we can appreciate the performance of SURF and SIFT feature descriptors, with respect to the poor outcomes of SIFT, which reconstructed images appears completely distorted and unrecognizable.

In all the cases, the ability of the algorithm to find the matches seems worst in the second patch, located in the ceiling background (which texture with the blue sky and stars is reproduced for a large part of the Capella roof), while in the patches located in regions of the image characterized by bigger shades, texture and angles variations, correlations are easier to be detected.

Conclusions:

This small project provided some insight about the practical application of a feature extractor algorithm and proved how they can be applied not only to a simple task, as the mapping of an image on another one or image completion, but also to more complex scenario like field of view reconstruction and robotic vision that represent some challenges in the advanced computer vision research.