UNIVERSITÀ DEGLI STUDI DI VERONA

Automatic 2D Mosaicing

COMPUTER ENGINEERING FOR ROBOTICS AND SMART INDUSTRY

Nicola Marchiotto - VR462317

2023

Contents

1 Introduction

The goal of the project is construct a software application for the generation of automatic 2D mosaic from a set of 2D images related by an homography. The procedure is also known in literature as 2D image stitching. The software must be able to automatic estimate the correspondences between the images, implement a robust estimation of the homography transformation and deal with colour blending.

1.1 Algorithm overview

The implemented algorithm is described by the following steps

- An image is arbitrary chosen as the starting one
- A tree like structures is used to store homography transformations from an image to another image reference frame. This allowed an easy homography concatenation computation. Each node is composed by an image identifier and the homography matrix transformation H which moves the node image in the reference frame of its parent. The starting image is set as the head of the tree.
- Two approaches were used to retrieve the conjugate points matches to compute the homography matrices. They will be analysed later in the report.
  - SIFT
  - LoFTR CNN
- Since the LoFTR CNN requires 640×480 pixel images, these dimension were also used for the SIFT case to have comparable results.
- To build the tree which stores the homography matrices, the following procedure was repeated until no more image to use were left.

  In the following lines, the expression *the image stored in the tree* means the image which corresponds to the image identifier of that node.
  - At every iteration, each image stored in the tree, now called *target_image*, is compared with every image not yet contained in the tree, now called *source_image*.
  - The pair *target_image* - *source_image* with the greatest number of matches is chosen as the best of the iteration. The homography transformation which moves the *source_image* to the *target_image* reference frame is then robustly estimated using RANSAC with the relative point matches.
  - A new node is attached to the tree with image identifier as the one of the source image and the computed homography matrix as the H field of the node. Its parent is set as the node with image identifier equal to the *target_node* image identifier.
  - If at any iteration, the result pair *target_image* - *source_image* produces a number of matches less than 50, the tree building procedure is interrupted
- The mosaic is build in the following way.
  - The starting image is augmented with black contours and then moved to the center of the image, This to allow space were the warped images can be placed
  - Every image is then attached to the starting one by walking the tree and applying the relative homography matrix transformation. If a node has not the tree head as its parent, the homography transformations are iteratively concatenated to retrieve the transformation which moves the node image to the reference frame of the starting one.
  - At each image addition, colour blending operations are performed.

2 Conjugate point extraction

To automatically detect the conjugate points from 2 given images, SIFT technique and LOFTR neural network were exploited.

2.1 SIFT

The Scale Invariant Feature Transform, SIFT, detects salient points using a Laplacian of Gaussian filter to the Scale Space. The method is invariant to scale, rotation, illumination and point of view. The SIFT algorithm is based on two main ideas: *Scale Space* and *Laplacian Pyramid*.

The *Scale Space* of an image is a function $L(x, y)$ that is obtained by a convolution of a Gaussian kernel (at different scales) with the input image:

$$L(x, y, \sigma) = G(x, y) * I(x, y)$$

The idea is that different blur, with a following binarization, shows features of different dimension, according to the Gaussian windows size σ.

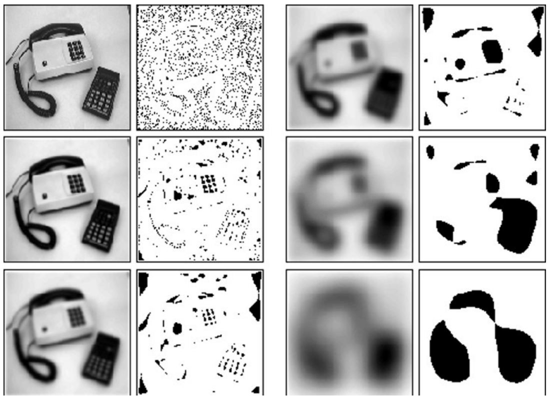

Figure 1: Scale space

Figure 2: Laplacian Pyramid

A *Laplacian pyramid* is a set of images equal to the original one, but at every step the dimension is reduced by half.

The idea is that using images of different size, feature of different dimensions are underlined.

These two approaches are then put together to form the Laplacian of gaussians *LoG*

LoG = Gaussian blur + edge detection with Laplacian

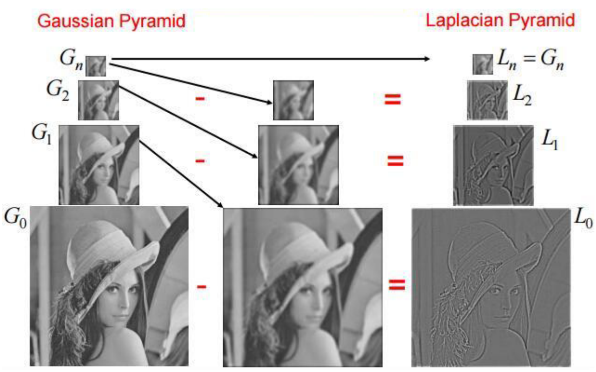

Figure 3: Laplacian Pyramid for edges extraction

The *LoG* kernel is convoluted with the image to find discontinuities i.e. edges, so points which have second derivative equal to zero.

The Difference of Gaussian filter is used to approximate the Laplacian of Gaussian filter:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

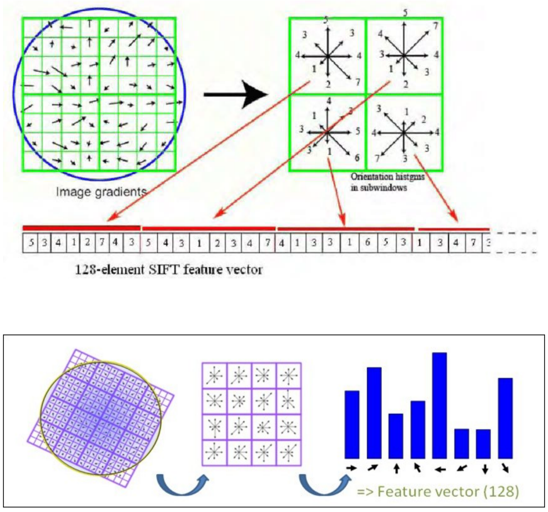

Figure 4: Point Descriptors

Once the salient points have been found, it's necessary to uniquely identify them using a signature.

At each salient point is associated an orientation. This is done centring a $16x16$ window on each salient point, the histogram of gradients is then computed in this region.

The point descriptor, the signature for the point, is then computed. This is a 128-vector obtained from the concatenation of the 16 histograms of gradients!

When in 1.1 the SIFT algorithm is used, the image comparison corresponds to the image descriptors comparison.

Salient points matches were computed using FLANN (Fast Library for Approximate Nearest Neighbors), which allows to perform approximate nearest neighbor searches in high dimensional spaces very easily.

From the various algorithm offered by the library, KNN was chosen to search correspondences between the images.

Figure 5: Point Matching

2.2 LoFTR CNN

LoFTR CNN, Detector-Free Local Feature Matching with Transformers, is a Convolutional Neural Network used for local image feature matching.

Instead of performing image feature detection, description, and matching sequentially, as in the SIFT method, the network establish pixel-wise dense matches at a coarse level at first, and then later refines the good matches at a fine level.
In contrast to dense methods that use a cost volume to search correspondences, self and cross attention layers are used in Transformer to obtain feature descriptors that are conditioned on both images.
The global receptive field provided by Transformer enables the network to produce dense matches in low-texture areas, where feature detectors usually struggle to produce repeatable interest points.

3 Homography Computation

To compute the homography matrix, the opencv library was exploited. This matrix is the perspective transformation H between the source and the destination image, computed by assuming that the two images are related by a common plane.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The homography computation requires only four pairs of points theoretically. Since these points can be affected to error, given that they are automatically extracted, the matrix was computed using RANSAC.

The method iteratively computes the homography using a subset of the extracted pairs to minimize the projection error, until a threshold error or a maximum number of iteration is reached. The following equation represent the projection error to be minimized during the estimation procedure

$$\sum_i \left[\left(x' - \frac{h_{1,1}x_i + h_{1,2}y_i + h_{1,3}}{h_{3,1}x_i + h_{3,2}y_i + h_{3,3}} \right)^2 + \left(y' - \frac{h_{2,1}x_i + h_{2,2}y_i + h_{2,3}}{h_{3,1}x_i + h_{3,2}y_i + h_{3,3}} \right)^2 \right]$$

4 Colour Adjustment Operations

After warping the chosen image to stitch to the starting one, colour correction operations are necessary on the overlapping region, since we can't simply add the intensity level of the two images pixel by pixel.

Figure 6: Without colour blending

The implemented method consists in computing two weights for our images and the apply a weighted sum of these. The formula is the following

$$I_{blend} = \frac{I_1 * w_1 + I_2 * w_2}{w_1 + w_2}$$

w_1 and w_2 are matrix of weights and are computed using the `distance_transform_edt()` function of the python scipy library.

The multiplication $I_i * w_i$ is like applying a Gaussian weight to the image, since the `distance_transform_edt()` return a matrix mask which gives more importance to the pixel in the center of the image and less to the ones at the edges.

Figure 7: With colour blending

5 Results

5.1 Andalo

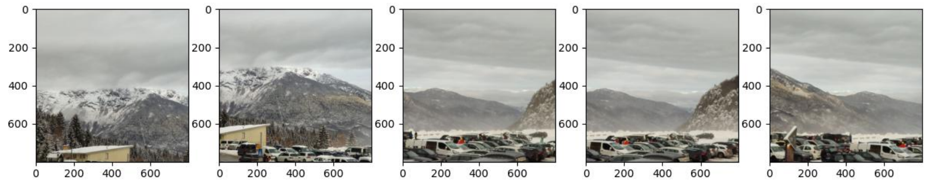

Figure 8: Some of the used images

Figure 9: Result using SIFT

Figure 10: Result using LoFTR

5.2 Arena

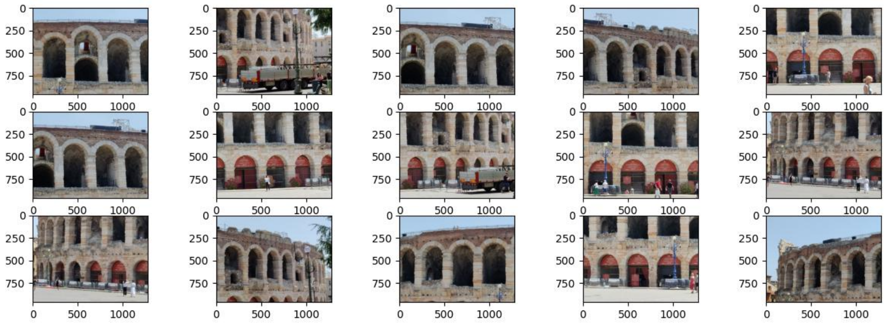

Figure 11: Some of the used images

Figure 12: Result using SIFT

Figure 13: Result using LoFTR

5.3 Mansion

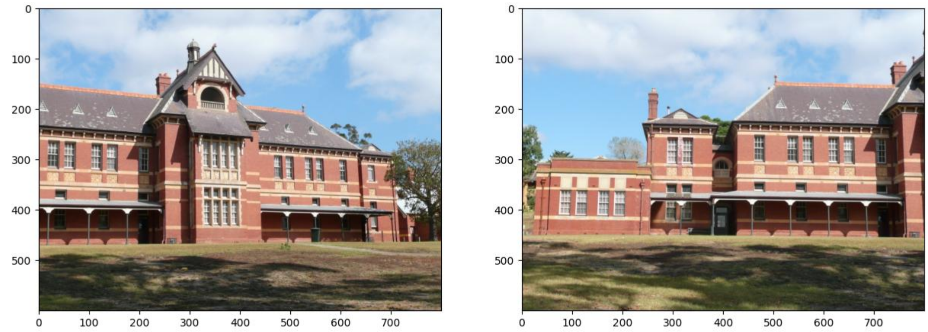

Figure 14: Some of the used images

Figure 15: Result using SIFT

Figure 16: Result using LoFTR

5.4 Bridge

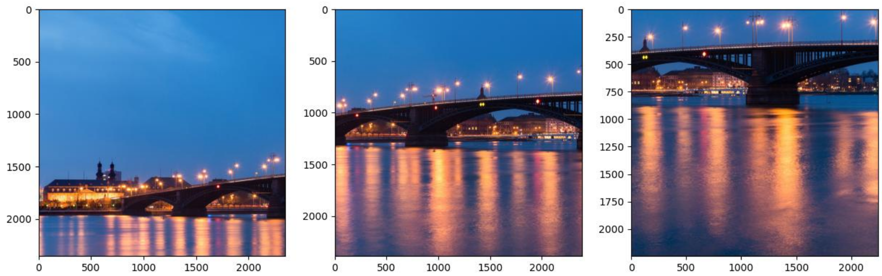

Figure 17: Some of the used images

Figure 18: Result using SIFT

Figure 19: Result using LoFTR

5.5 Building Site

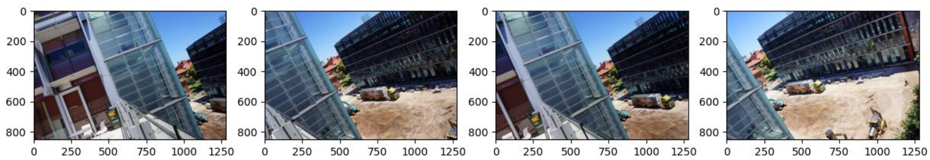

Figure 20: Some of the used images

Figure 21: Result using SIFT

Figure 22: Result using LoFTR

5.6 Fish Bowl

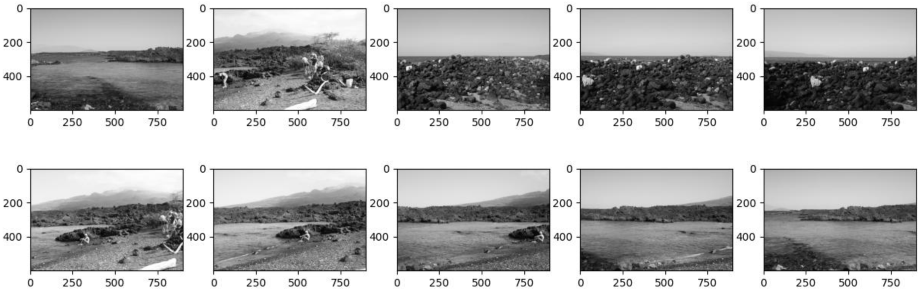

Figure 23: Some of the used images

Figure 24: Result using SIFT

Figure 25: Result using LoFTR

5.7 Golden Gate

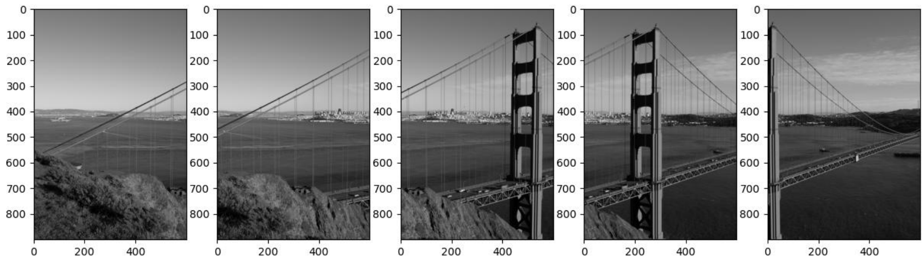

Figure 26: Some of the used images

Figure 27: Result using SIFT

Figure 28: Result using LoFTR

5.8 Half dome

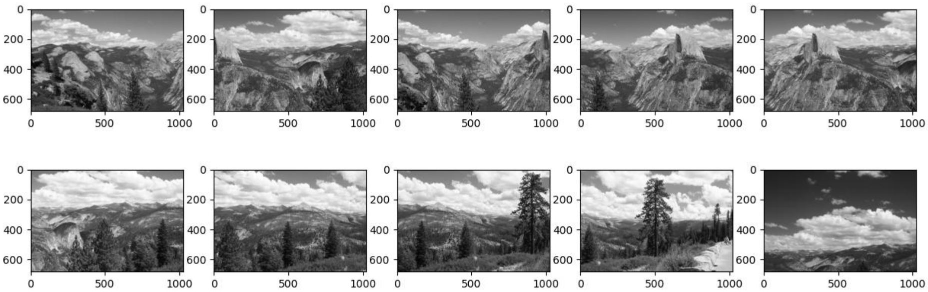

Figure 29: Some of the used images

Figure 30: Result using SIFT

Figure 31: Result using LoFTR

5.9 Beach Hotel

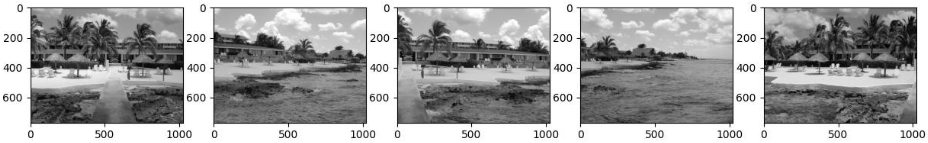

Figure 32: Some of the used images

Figure 33: Result using SIFT

Figure 34: Result using LoFTR

5.10 Mountain Landscape

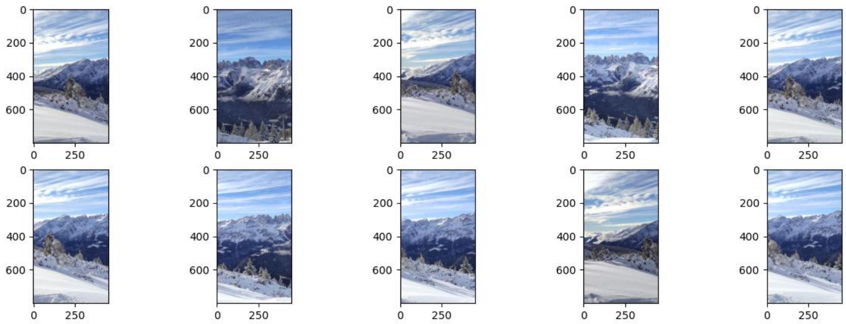

Figure 35: Some of the used images

Figure 36: Result using SIFT

Figure 37: Result using LoFTR

5.11 Office

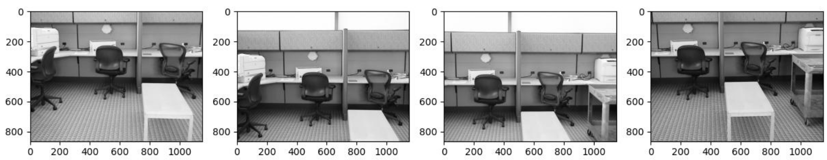

Figure 38: Some of the used images

Figure 39: Result using SIFT

Figure 40: Result using LoFTR

5.12 Ponte Nuovo

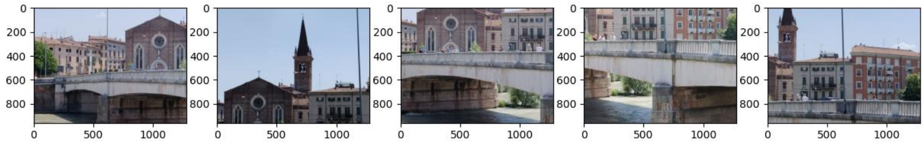

Figure 41: Some of the used images

Figure 42: Result using SIFT

Figure 43: Result using LoFTR

5.13 River

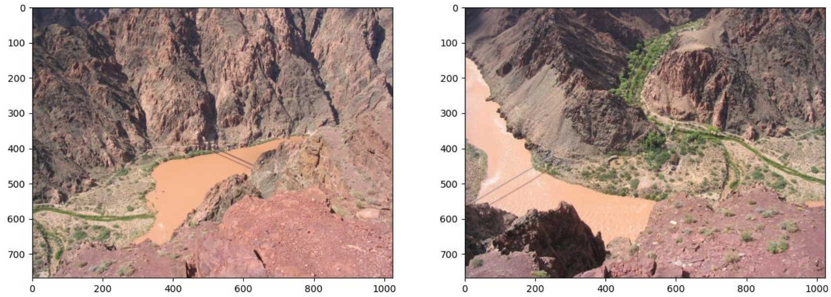

Figure 44: Some of the used images

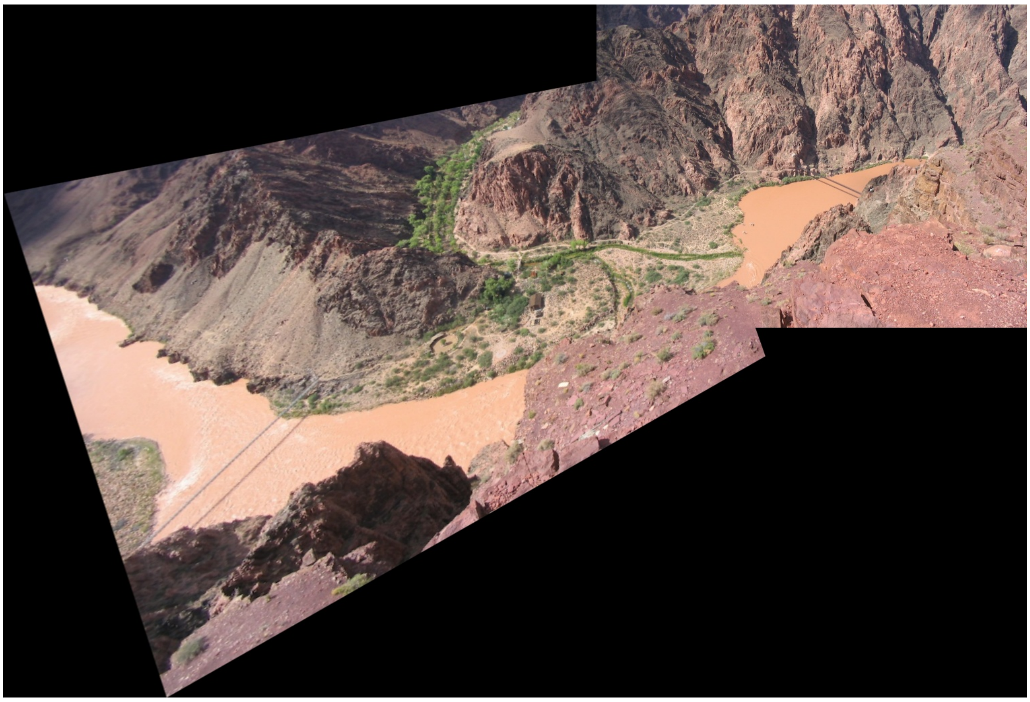

Figure 45: Result using SIFT

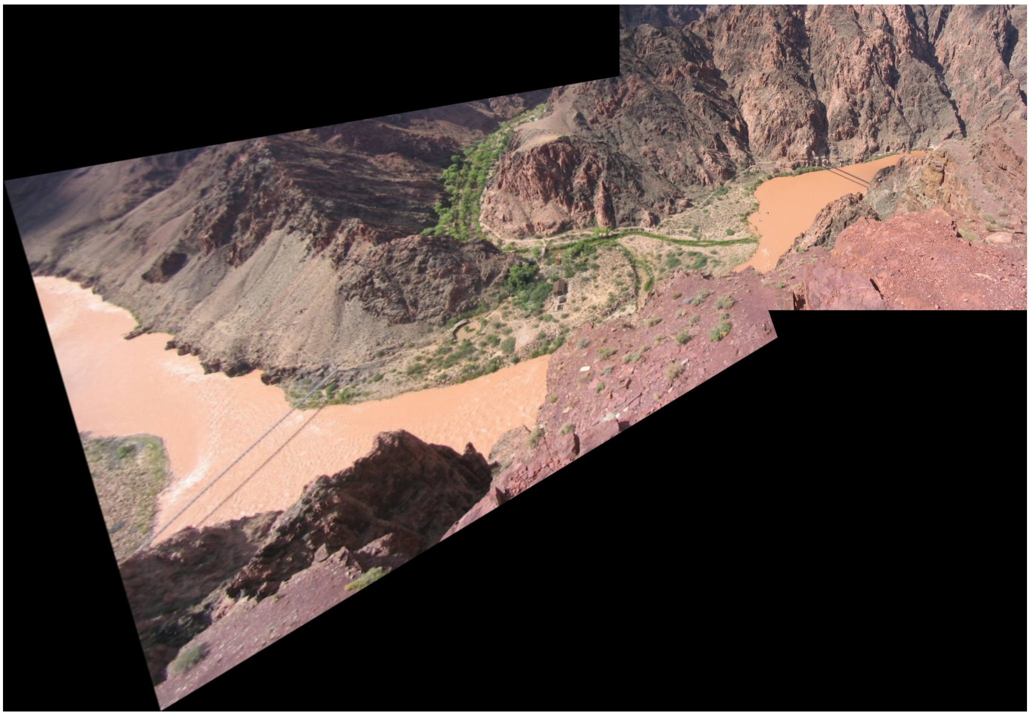

Figure 46: Result using LoFTR

5.14 Roof

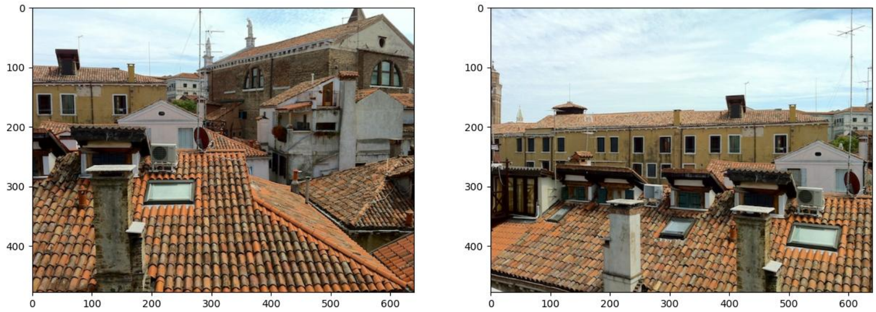

Figure 47: Some of the used images

Figure 48: Result using SIFT

Figure 49: Result using LoFTR

5.15 San Pietro

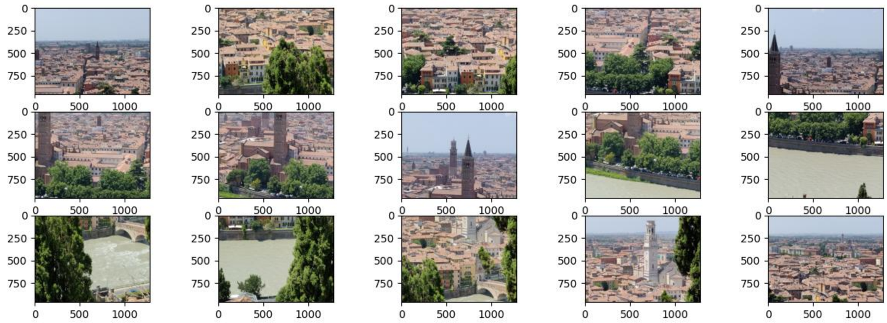

Figure 50: Some of the used images

Figure 51: Result using SIFT

Figure 52: Result using LoFTR

5.16 Shangai

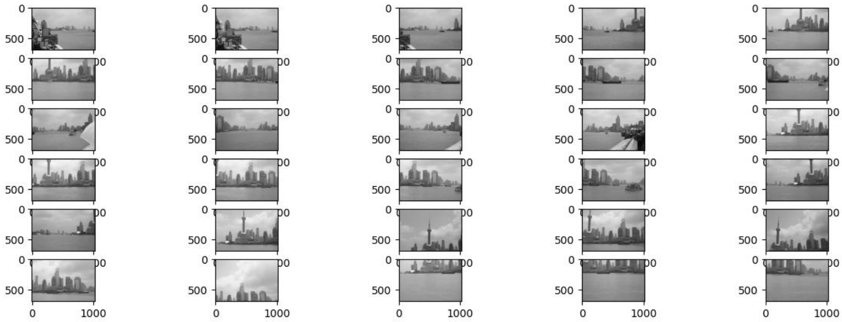

Figure 53: Some of the used images

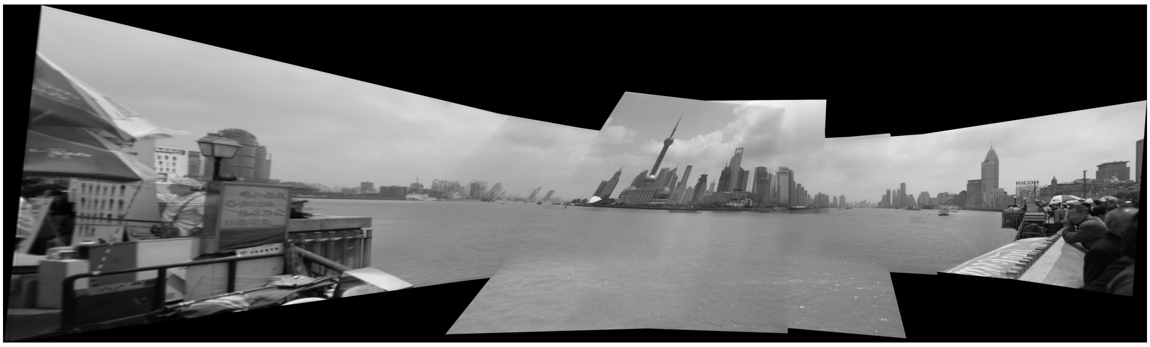

Figure 54: Result using SIFT

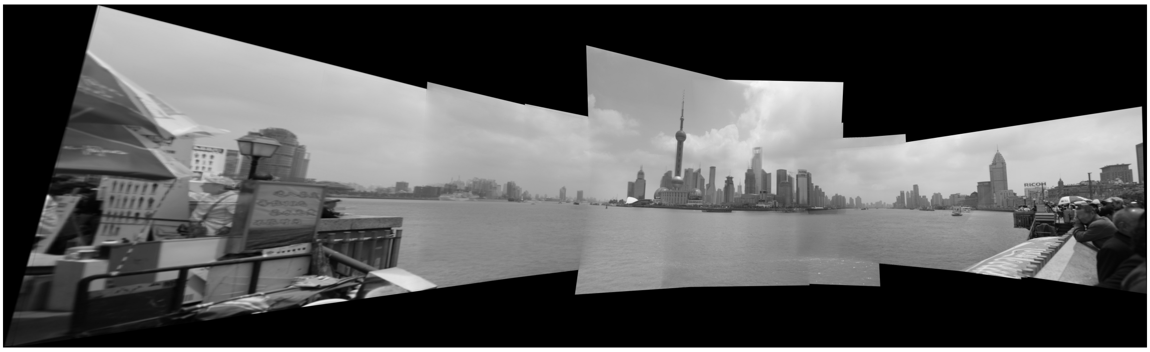

Figure 55: Result using LoFTR

6 References

1. Jiaming Sun, Zehong Shen, Yuang Wang1, Hujun Bao, Xiaowei Zhou. LoFTR: Detector-Free Local Feature Matching with Transformers. In CVPR, 2021