3D augmented reality
a.a. 2020/21

De Toni Alberto - 1236657
Nicola Casagrande - 1242363

Compression strategy using autoencoders & SfM reconstruction Project B.2

University of Padova
Master degree in ICT for Internet & Multimedia

OBJECTIVE:

Design a compression strategy for local SURF descriptors using autoencoders. Training data can be generated using the images of dataset Portello and Castle. Testing must be done on dataset FountainP-11 and Tiso. The reconstructed descriptors (only for the test set) are used to perform a SfM reconstruction using COLMAP (using the two test dataset).

Programming languages: MATLAB/Python/C++.

Data

Datasets were pictures of four architectural elements taken from different perspectives.

Testing

Training

TRAINING

The images from the training set are loaded into a single image datastore.

A loop for each image is designed in order to:

1. Convert the image to grayscale;

2. Detect the SURF features (using the MATLAB function “detectSURFFeatures”);

3. Extract the features (descriptors) using the function “extractFeatures”. In this case they are represented as a vector of 64 columns and N rows;

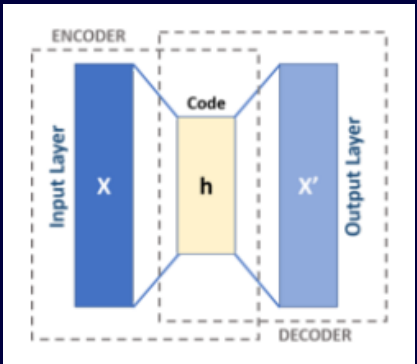
4. Stack vertically the descriptors of each image in a single matrix.

AUTOENCODER

The autoencoder has been built using the MATLAB function “layerGraph”.

We chose the following three-layer structure:

- an input layer that receives features of dimension 64 (using “featureInputLayer(64)”);
- two fully connected layers of dimension 6 and 64 (using “fullyConnectedLayer(6)” and “fullyConnectedLayer(64)”), the second one having dimension 64 as will work as output layer;
- a regression layer that will predict the reconstructed values (using “regressionLayer”).



ADAM (Adaptive Moment Estimation) optimizer
2 Epochs

TESTING

```
MATCHINGS(imdsTiso,
FEATURES(imdsTiso, autoenc));
```

The function **FEATURES** extracts the features from the test dataset and saves the keypoints in a .txt file while the descriptors are fed into the function **MATCHINGS**, which calculates the matchings between each couple of images and saves them in another .txt file.

With autoencoder

matchings - Blocco note di Windows

File Modifica Formato Visualizza ?

```
0000.jpg 0001.jpg
0 1
4 2
5 9868
6 3
9 1936
10 84
12 9
13 21
14 5
15 7
18 26119
21 12
23 11
24 24
25 16
26 5548
27 23
28 11105
29 26
30 29
34 7663
36 8059
37 37
42 48
43 39
45 44
46 27630
47 27
50 22117
53 5854
57 52
60 1215
66 26814
67 203
71 76
76 18609
78 13
80 14613
81 23953
83 24855
87 70
89 6189
```

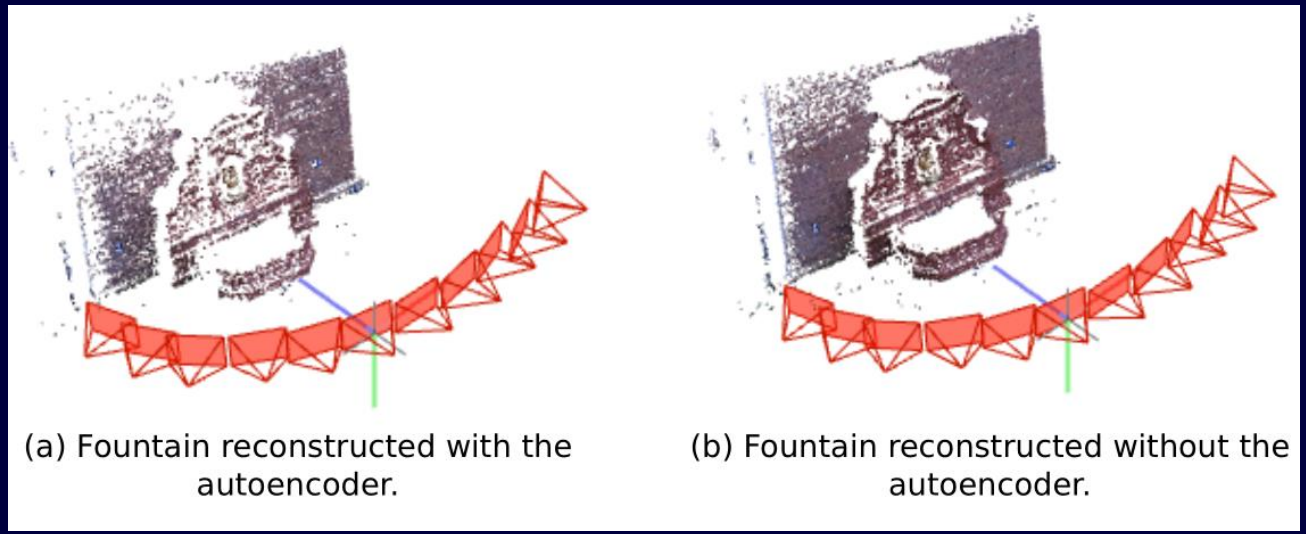
Without autoencoder

matchings_true - Blocco note

File Modifica Formato Visua

```
0000.jpg 0001.jpg
0 1
1 0
3 4
4 2
6 3
8 6
12 9
13 8
14 5
15 7
19 28
21 12
23 11
24 24
25 16
26 14
27 23
29 26
30 29
31 15
33 18
34 20
37 37
39 41
40 56
42 48
44 59
45 44
46 97
47 27
48 30
50 72
52 25
53 5854
54 13
57 52
61 54
64 42
66 131
68 45
71 76
```

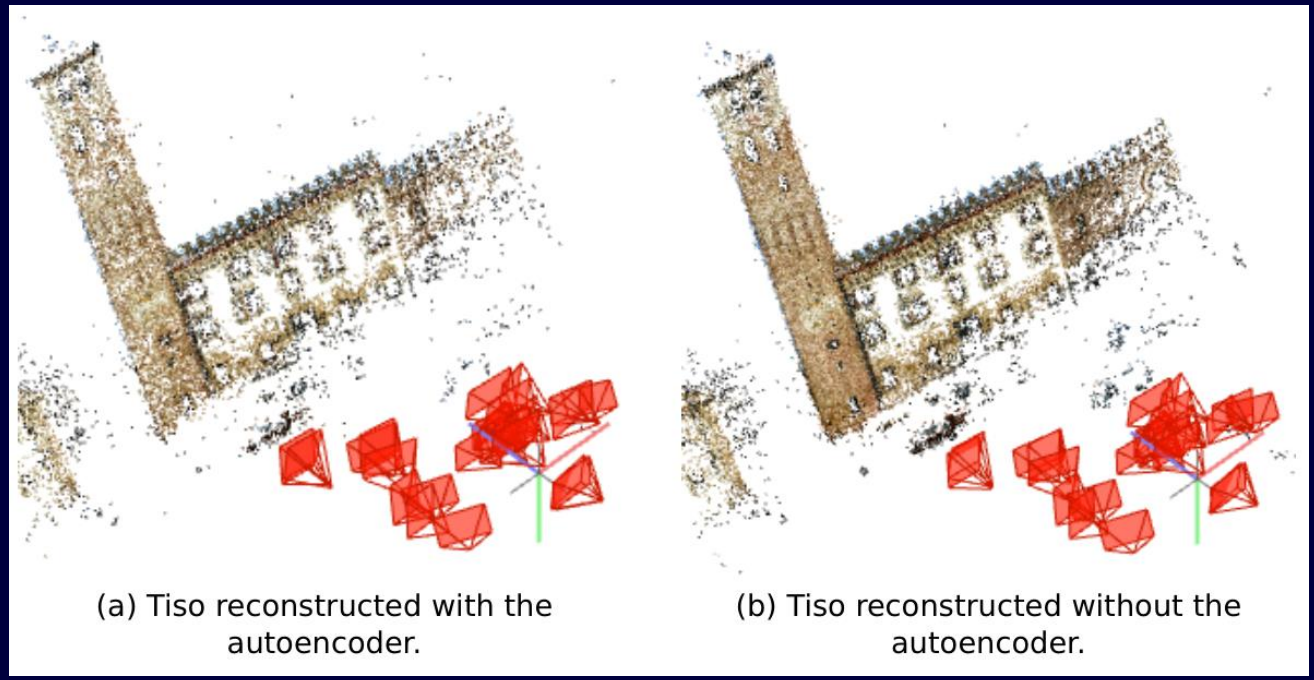
COLMAP RESULTS



(a) Fountain reconstructed with the autoencoder.

(b) Fountain reconstructed without the autoencoder.

COLMAP RESULTS



(a) Tiso reconstructed with the autoencoder.

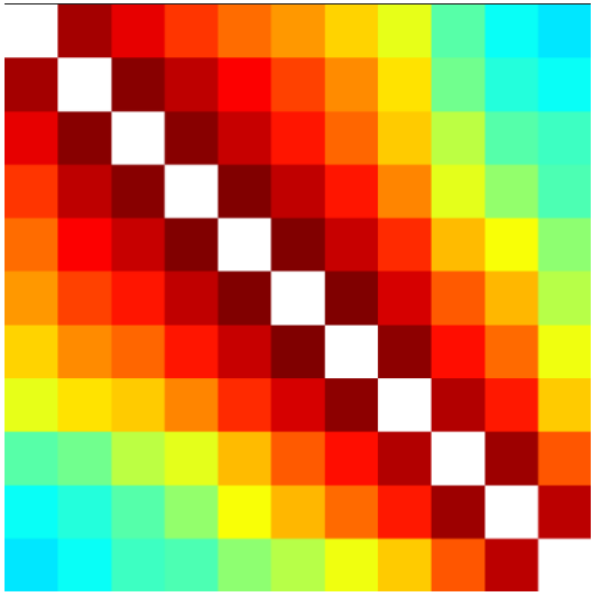
(b) Tiso reconstructed without the autoencoder.

SOME STATS

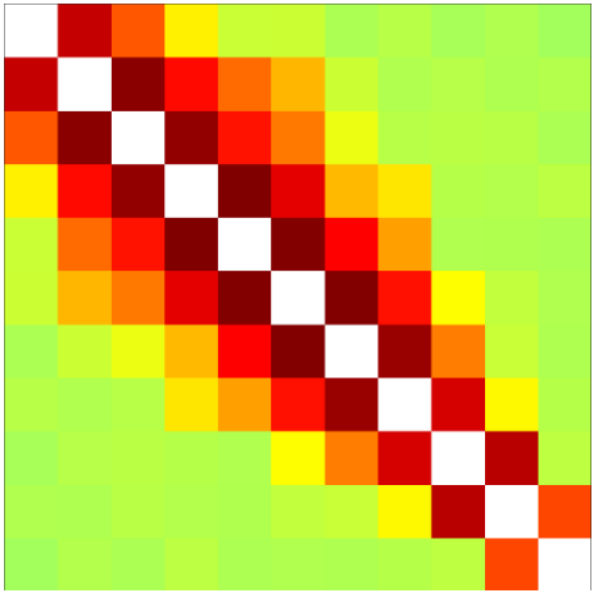
Parameters	No autoencoder	Autoencoder
Cameras	11	11
Images	11	11
Points	30620	14674
Observations	139909	52265
Mean track length	4.5692	3.56174
Mean observations per image	12791	4751.36
Mean reprojection error	0.49785	0.428602

Table 1: Result comparisons for Fountain-P11 between the no-autoencoder and autoencoder strategies.

SOME STATS



(a) Match matrix without the autoencoder.



(b) Match matrix with the autoencoder.

CONCLUSIONS

- This strategy is a **lossy** compression.
- It can be useful in environments where the **quality** of reconstruction is **not important**, for example in object detection.
- The **edges** of the objects are mostly **preserved** while straight planes are “thinned out”, i.e. represented with less points.
- Other few matches can be found in **randomic** places.

Thank you for your attention!