3D recovery of urban scenes. Session 5

Guillem Capellera[1], Johnny Núñez[1], and Anna Oliveras[1]

1. Universitat Pompeu Fabra, Barcelona, Spain

0 Introduction

The goal of this last session is to do the 3D reconstruction of uncalibrated images with a stratified method. We will apply Structure from Motion (SfM) pipeline in order to achieve a 3D sparse reconstruction. Then we will get the Dense 3D reconstruction with Multi-View Stereo. In this session, we use the COLMAP tool, introduced in [3].

1 3D mesh reconstruction from a set of images from the Gerrard Hall dataset

After successfully installing Colmap [3] we automatically reconstructed the Gerrard Hall dataset introduced in [2]. The Sparse reconstruction obtained is shown in Figure 1. Then we visualized the dense reconstruction with Meshlab which is shown in Figure 2 and removed some noise manually as shown in Figure 3. We tried to remove the noise with other techniques such as filters from Meshlab but they were not successful.

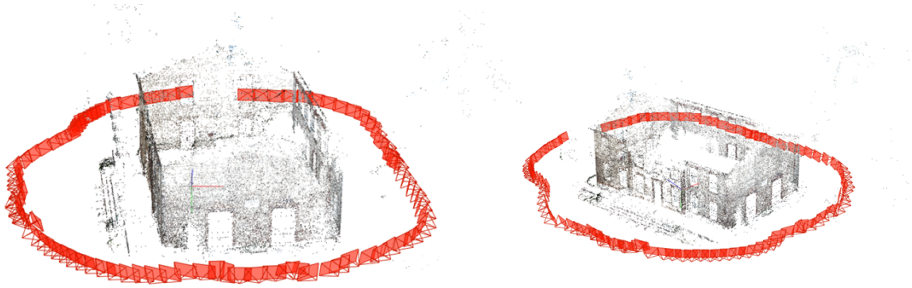

Fig. 1: Automatic Sparse reconstruction of the Gerrard Hall

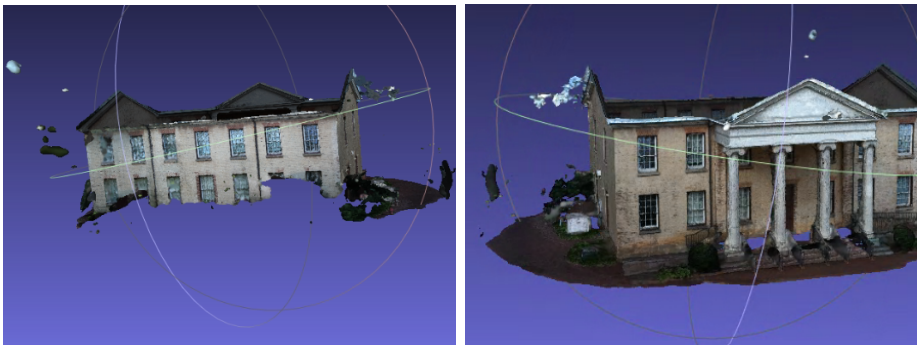

Fig. 2: Automatic Dense reconstruction of the Gerrard Hall

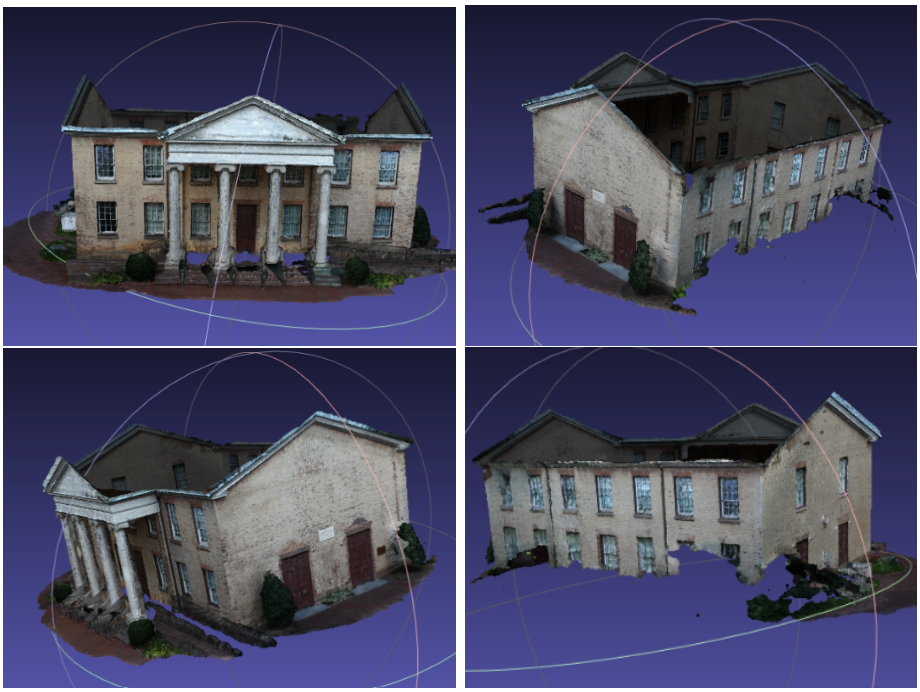

Fig. 3: Automatic Dense reconstruction of the Gerrard Hall after denoise

Note that in the denoised version, we still have some noise for example in the columns or the railing stairs. In addition, we also have some missing parts of the reconstruction due to occlusions.

2 Analyze the reconstructions using python

2.1 Gerrard Hall reconstruction

The reconstruction performed in the first section serves as input to work with the 3D point cloud in python. The model is loaded, displaying three different dictionaries related to images, cameras, and 3D points. The main information for each dictionary is summarized in Table 1.

Information of the dictionaries		
Dictionary	**Information**	**Explanation**
Image	id	Unique identifier of the image
	qvec	Quaternion vector of the image
	tvec	Translation vector of the image
	camera_id	Identifier of the camera used to capture the image
	name	Name of the image file
	xys	2D pixel coordinates of the image points
	point3D_ids	Identifier of the 3D points visible in the image
Camera	id	Unique identifier of the camera
	model	Camera model type
	width	Image width in pixels
	height	Image height in pixels
	params	Camera parameters
Point3D	id	Unique identifier of the 3D point
	xyz	Cartesian coordinates of the 3D point
	rgb	RGB color values of the 3D point
	error	Reprojection error of the 3D point
	image_ids	Identifier of the images in which the 3D point is visible
	point2D_idxs	Indices of the 2D points corresponding to the 3D point in the images

Table 1: Overview of the information and fields contained in the dictionaries (Image, Camera, and Point3D) used to describe the elements in a computer vision system.

The dataset.db, which contains the information of keypoints and matches extracted by Colmap, is then loaded. The keypoints are stored for each image, and we know that their first two coordinates correspond to the pixel location in the image. The matches dictionary contains keys with pairs of images, and for each pair, there is an array of dimensionality (num_matches x 2) that contains the id information between each pair of keypoints that have matched.The code provided also offers the opportunity to visualize the 3D point cloud using a function from open3D.

The first question of the notebook asks to compute the number of keypoints. It's easy to compute using the keypoint dictionary and the len() function for each image. The total number of keypoints is 1068663.

Moving to the second question asks to compute the number of 3D points in the whole point cloud that were generated using keypoints from the first image. Two methods of computing this are performed, and the same result is reached. The first method involves using the point3D dictionary and checking if the first image index appears in the image_ids array for each point. The second method involves using the point3D_ids array contained in the images dictionary and computing the number of non-negative (excluding -1 values) and non-repeated values (we only want different 3D points). With the 8621 keypoints from the first image, 1840 of them generate a total of 1832 different 3D points. Note that here we can see that in few cases, one 3D point has been generated by more than one keypoint.

2.2 Plot the 3D points coloured according to the number of images and error

Once understood the dictionaries structure, we have plotted the 3D points with different colours with respect to the number of images (see Figure 4). Firstly, it is evident that the outlier points that do not belong to the main reconstructed object (the house) have been generated from only one or a few images. These points may be caused by incorrect camera poses, reflections, transparent objects, or other factors that can result in inaccurate triangulation computation. On the other hand, we can see that the regions generated with more images correspond to the walls of the house that are not occluded by the vegetation. This indicates that the presence of vegetation can cause occlusions that affect the accuracy of the 3D reconstruction. In contrast, the walls of the house that are not obstructed by vegetation are likely to have more reliable 3D reconstructions, as they have been generated from multiple images and are less susceptible to occlusions.

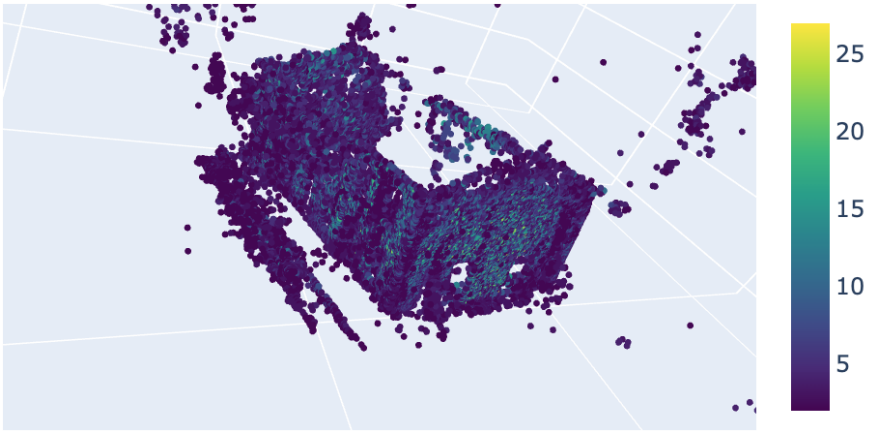

Fig. 4: 3D points coloured according to the number of images.

In Figure 5, the 3D points are plotted with different colors based on their error. The errors in photogrammetry can be caused by various factors, including the lack of overlap between images and the image quality. However, these factors are unlikely to be present in the Gerrard Hall dataset, as it is a complete dataset and the images were taken in good positions and the high-quality reconstruction option in Colmap was used. Another possible source of errors is related to outlier points, which may be generated from images with incorrect camera poses or due to other factors such as reflections or transparent objects. This issue may be present when reconstructing the 3D points corresponding to windows. Finally, the presence of vegetation surrounding the house can also affect the accuracy of the 3D reconstruction by causing occlusions and distorting the triangulation computation, resulting in an increase in error.

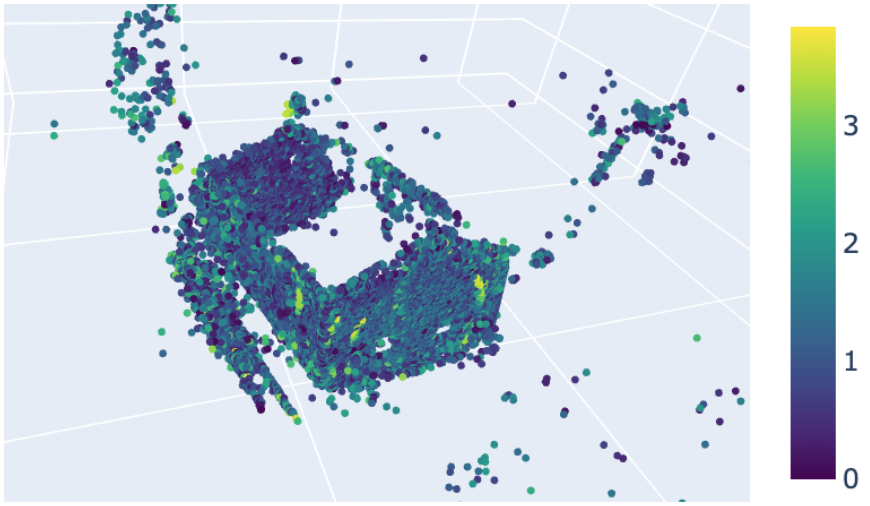

Fig. 5: 3D points coloured according to the error.

2.3 Plot the 3D points that originated from a keypoint in the first image. Also plot the image with the keypoints

In Figure 6, we can observe in the left plot the 3D points generated from the green keypoints in the right image. This supports the conclusions mentioned in the previous section that the vegetation can pose challenges to the reconstruction process by obstructing certain parts that cannot be reconstructed from this first image alone.

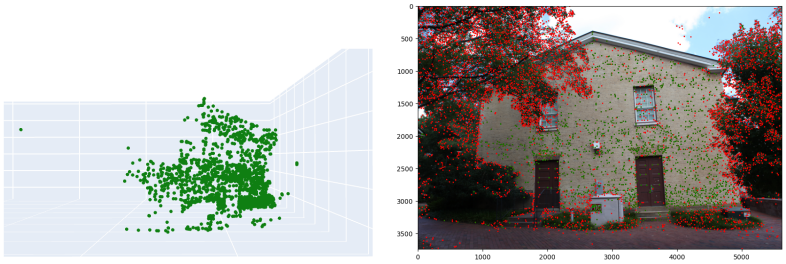

Fig. 6: The left image corresponds to the plot of the 3D points generated by the keypoints of the first image. The right image corresponds to the image with all keypoints. The corresponding ones that generates the 3D points are marked in green color.

2.4 Create a visualization for the number of matches between all images

To analize the number of matches between each pair of images we have plotted this heatmap (see Figure 7) form SEABORN package. The higher values are concentrated on the diagonal, indicating that the images are sorted in a geometrically meaningful way. By examining the heatmap, we can gain an intuition about which range of images depict the "flattened" facade parts of the house, which may correspond to the darker regions.

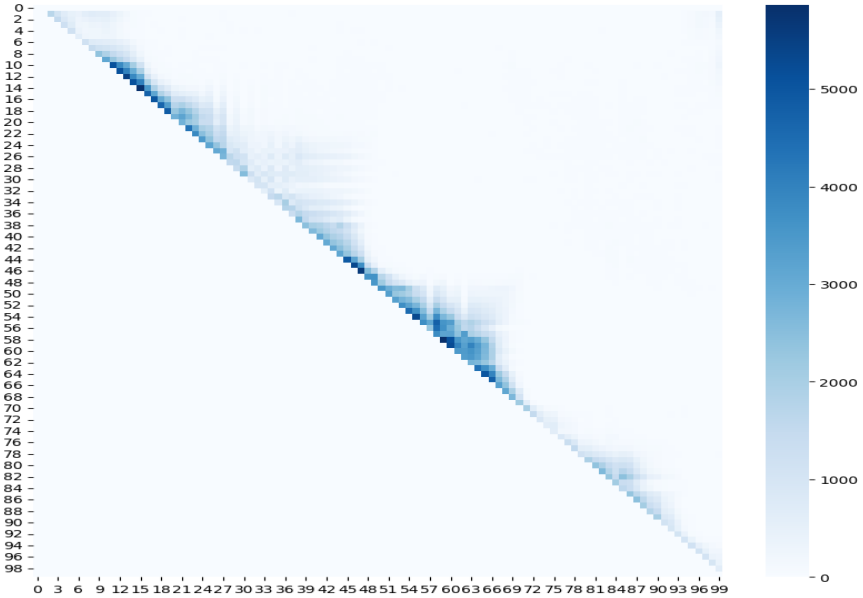

Fig. 7: 100 x 100 matrix that illustrates the number of matches between each pair of images.

2.5 Two images of the Castle dataset

In this case, we did the reconstruction just using the two images from lab 4 extracted from the EPFL Castle P30 dataset introduced in [1]. Then, we repeated the same procedure of the subsection 2.1 to import the reconstruction and generate the model and database in our network. Using database we are able to compute the keypoints of each image and its correspondent matches (see Figure 9.

We realize that there were few errors in matches that could be considered as outliers, so we perform a refinement of this matches to choose only the inlier ones by using only the refined matches provided by COLMAP, that are geometrically verified. This was done with the code provided in the COLMAP's documentation to export the inlier matches (see Figure **??**. After refinement we go from 1693 matches to 1630 inlier matches.

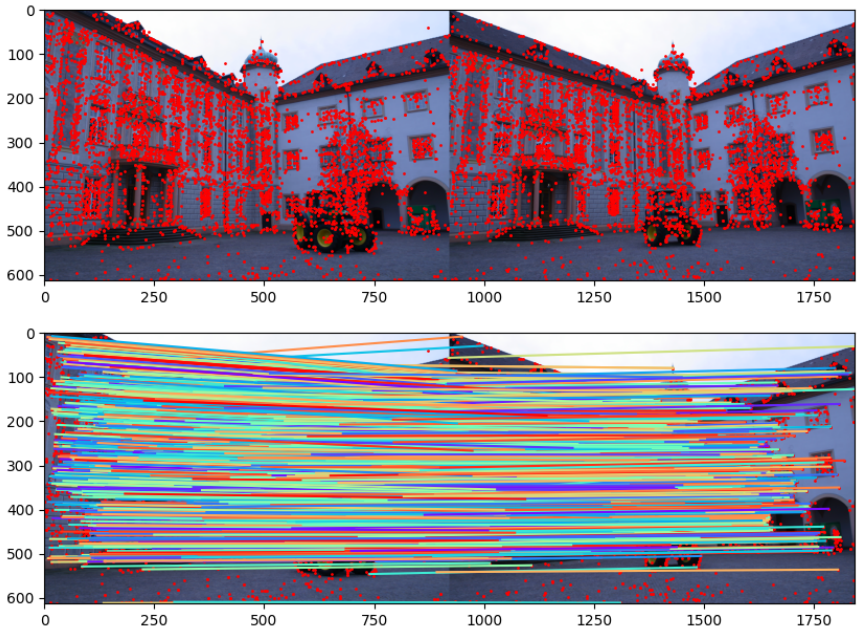

Fig. 8: The first image shows the extracted keypoints of the two images. The second image shows the matches between the keypoints of the two images. Both keypoints and matches are the correspondent results of the COLMAP database.

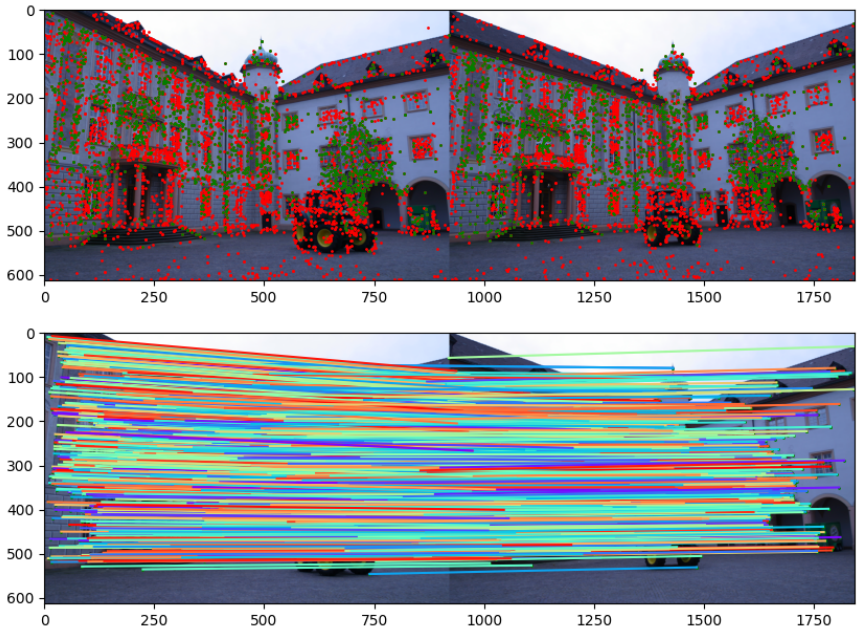

Fig. 9: The first image shows the extracted keypoints of the two images. Green keypoints are the once that are paired in the inlier matches. The second image shows the inlier matches between the green keypoints of the two images.

The matches from lab 4 can be seen in Figure 10

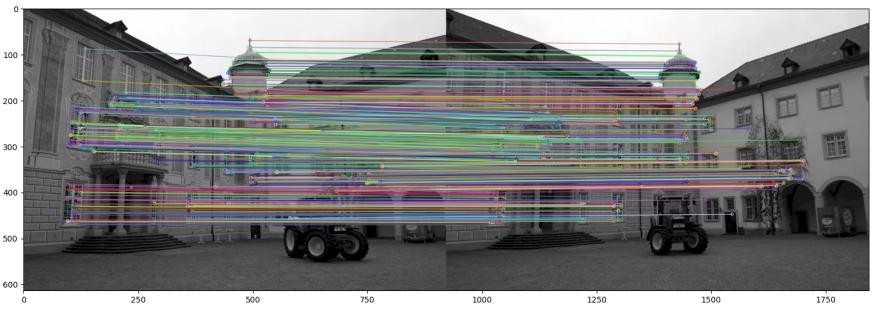

Fig. 10: Inlier matches obtained in lab 4

In Lab4, without using RANSAC, the ORB feature extractor algorithm produced 1290 matches. With RANSAC, the number decreased by approximately half to 681. In Lab5, COLMAP produced more matches than Lab4 before re-

finement, and after refinement, the reduction in matches was not as significant as with RANSAC. Two main differences between Lab4 and Lab5 are the choice of feature extractor, with SIFT used in COLMAP and ORB in Lab4, and the difference in the strictness of the matching algorithms, with Lab4's RANSAC being more restrictive than the refinement used in COLMAP.

2.6 Triangulate and visualize the 3D points from the keypoints extracted using Colmap on the two images used in lab 4

To perform triangulation, we need to first construct the two projection matrices (one for each image). The camera and image dictionaries already contain the necessary information to compute the extrinsic and intrinsic parameters, respectively. Specifically, for each camera, we have the following parameters: params $= [f_x,\ f_y,\ v_0,\ u_0]$, and for each image, we have the translation vector (tvec) and quaternion vector (qvec), which can be transformed into a rotation matrix. Once we have this information, we can use the inlier matches to compute the triangulation between each pair of images.

Figure 11 shows the resulting 3D point clouds obtained by applying the triangulation algorithm to the inlier matches found in both Lab4 and COLMAP, respectively. As discussed in the previous section, COLMAP produced more matches, which led to a larger number of 3D points after triangulation. This resulted in a denser 3D representation of the scene captured in the images. In contrast, because the inlier matches in Lab4 were concentrated mostly in the middle of the two images, the resulting 3D points were also compacted in a single region, losing information about the surrounding areas of the scene. This is an important limitation to keep in mind when working with limited or sparse data.

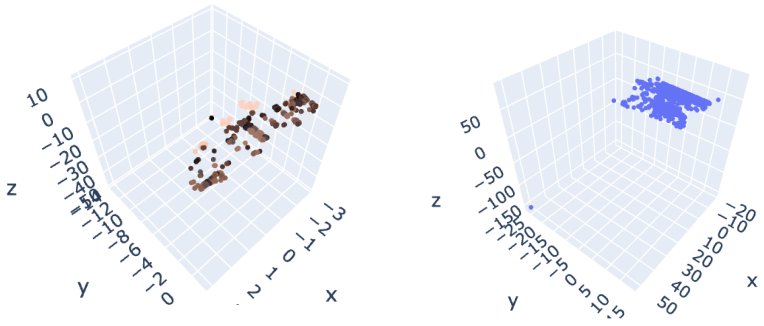

Fig. 11: Both images shows the 3D points computed using the triangulation algorithm. The left image is performed using as input the keypoints of the inlier matches computed in Lab4. The right image is computed using as intput the keypoints of the inlier matches computed with COLMAP.

2.7 Castle reconstruction

To do the castle reconstruction we used the whole dataset from the EPFL Castle P30 introduced in [1].

When there are many photos in the database, COLMAP loads all the extracted data into memory and seeds the reconstruction from two initial images. A scene is incrementally extended as new images are registered and new triangulated points are added. The results of the reconstruction are displayed in real time during the reconstruction process. Colmap utilizes the Structure from Motion (SFM) pipeline. It is an extremely powerful tool that can be used to reconstruct 3D scenes from a large number of photographs. This technique uses the camera's motion across multiple images to estimate the camera's position and orientation in 3D space, as well as the 3D structure of the scene.

On the other hand, we have the case that only we have two images for the castle. In this case, COLMAP attempts to find the best solution by matching the feature points from the two images and using a triangulation algorithm to determine the 3D structure of the castle. A bundle adjustment and other methods are then used to refine the reconstruction process. Finally, the 3D structure is rendered and displayed using a 3D visualization tool.

3 Configuring the Castle reconstruction to improve the results

When doing the automatic reconstruction of the whole castle we achieved the 3D reconstruction shown in Figure 12. See that there is noise and many holes in the walls. Thus, the next step is to configure the reconstruction to get the best possible mesh.

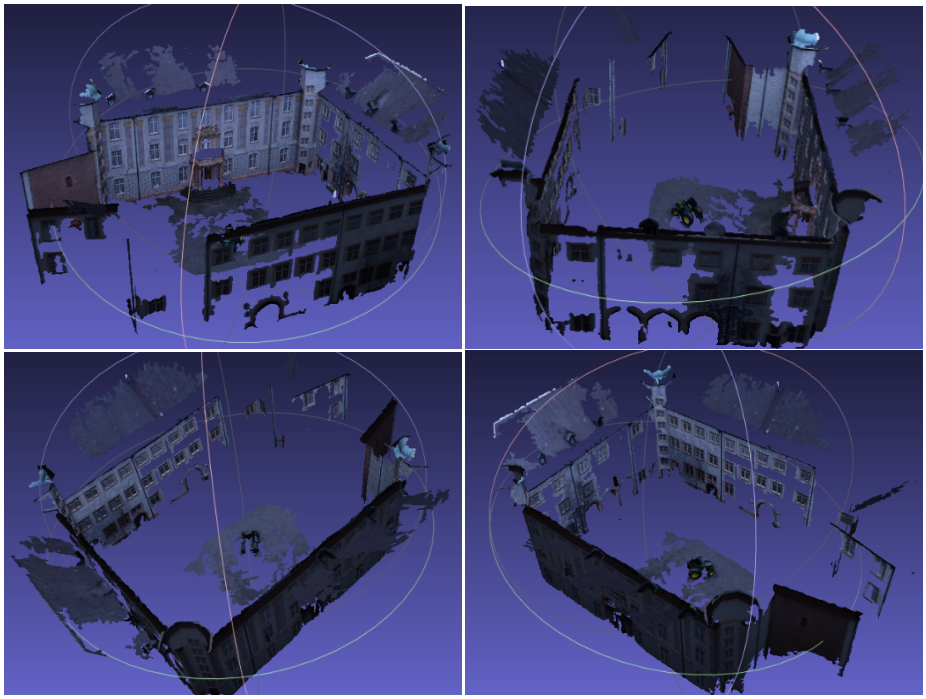

Fig. 12: Automatic reconstruction of the Castle

The first step is the feature extraction. Note that we use the pinhole camera model since we could not find the specifications of the cameras used to capture the EPFL P30 datset. With more time, we could have experimented with other types of camera models and seen which one worked the best like the radial one, if there was radial distortion. The parameters that we change are the maximum number of features which we increase drastically to get as many features as possible to try to fill the holes that we had in the automatic reconstruction. Also, we increase the number of octaves and their resolution to try to obtain more specific features with more details.

Secondly, we do the feature matching. Here we increased the block size since it controls the size of the neighborhood used for local feature detection. In general, with a bigger block size, coarser results are obtained that may lead to a more robust matching of features. We also increased the number of matches to not limit the results. Moreover, we relax some parameters to be less strict like the error, the number of inliers and the confidence. Note that we are trying to have more matches than in the automatic case.

After we do the sparse reconstruction and then we choose the parameters to do the dense reconstruction. In this final step, we change mainly maximum repro-

jection to see if we can achieve a little less accurate solution but with fewer holes.

The results obtained are shown in Figure 13. Note that we achieve a pretty good reconstruction in the zones that have interesting features like the brick wall. Maybe because of the increase in the number of octaves and their resolution, we are obtaining more accurate results than before in the areas where there are enough keypoints. It is important to see that the number of features in the features extraction is not limiting our results, since we set it to a high value.

Fig. 13: Manual reconstruction of the Castle

4 3D mesh reconstruction of images captured by us

In this section, we tried to reconstruct different scenes with photos taken by us. The considerations that we took into account when taking the pictures were the overlap between the pictures ensuring that there was enough information to put the images together. Also, the lighting should be similar across all the images, trying to avoid shadows and reflections.

First, we tried to reconstruct an isolated building. However, we were not able to achieve a good 3D reconstruction maybe because of the bad quality of the

images, taken with a mid-quality smartphone which lead us to poor keypoint matching. The results can be seen in Fig**??**.

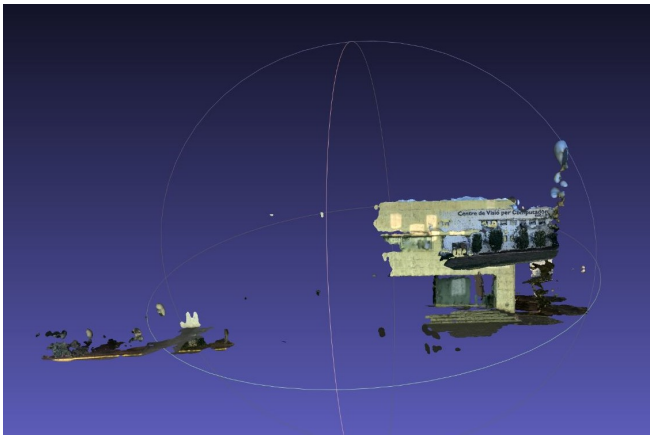

Fig. 14: Failed 3D dense reconstruction

So then, we decided to acquire images with better quality using another device. Moreover, we tried to do the reconstruction of a simpler object to see if we achieved better results. In this case, we were really careful when taking the pictures. The results of the 3D dense reconstruction after removing some noise are shown in Figure 15. To do so we used 30 images, making sure that the object was far enough from the camera. Also, we chose an object with patterns so that we could have more features easily extracted than a plain colour.

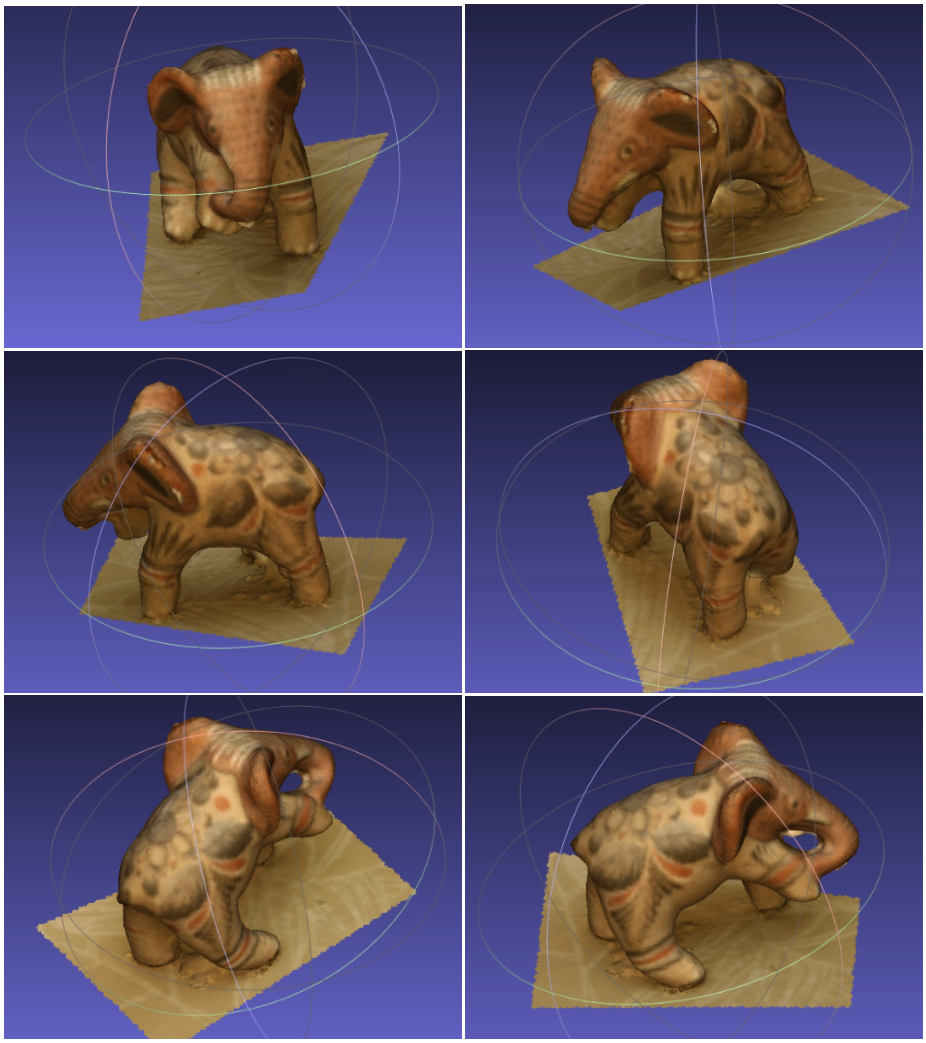

Fig. 15: 3D Reconstruction of an elephant sculpture

5 Conclusions

All in all, in this session we experimented with the COLMAP 3D reconstruction and managed to export the information to later be able to work with it on python. Moreover, we saw that the automatic reconstruction that COLMAP provides is limited and it's better to do a manual reconstruction to find good results. However, this process is time-consuming and the best parameters are difficult to find. Finally, when choosing our own images we saw the importance of capturing properly the images to achieve good results. Also, it is important to

have enough features to extract, so it is essential to avoid plain surfaces without any pattern or roughness and also glasses or materials that produce reflections or transparencies.

References

[1] Bo Li et al. "Worldwide Pose Estimation Using 3D Point Clouds". In: *European Conference on Computer Vision (ECCV)*. 2010.

[2] Torsten Sattler et al. "Image Retrieval for Historical Architectural Research". In: *European Conference on Computer Vision*. Springer. 2012, pp. 453–466.

[3] Johannes L Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.9 (2016), pp. 1814–1830.