Recovery of 3D Urban Scenes:
Structure from Motion for 3D Reconstruction

Álvaro Budria, Alex Carrillo, Sergi Masip and Adrià Molina

Universitat Pompeu Fabra

February 2022

1 Introduction

In this document, we present the results of using COLMAP (please, refer to section 5.1 for a detailed explanation of this pipeline and its paper) in order to create 3D reconstructions from a set of images. Contrary to what we did in the last deliverable, this week we use an SfM pipeline to detect keypoints, compute the matches and perform the reconstruction. First, we will automatically reconstruct the *Gerrard Hall* dataset. Then, we will analyze and visualize the results using Python code. We will also create a reconstruction of the *Castle* set of images, and, finally, we will reconstruct a custom scene with our own images. In addition to that, we will comment on 3 papers on this topic and we will assess their strengths and their weaknesses.

2 3D reconstruction from the Gerrard Hall dataset

In this section, we explain how we reconstruct Gerrard Hall's 3D scene using COLMAP, and we analyze the results.

2.1 Automatic reconstruction using COLMAP

After installing the COLMAP application and downloading the dataset, we proceed to create the reconstruction. To do so, we use the *Reconstruction >Automatic reconstruction* option. Here we can observe several options (see Fig. 1). We set the workspace directory to Gerrard Hall's directory and the images directory to its images subdirectory. Some relevant options in our case are *quality*, which if set too high it may take too long, and the *sparse model* and *dense model*, which we check as we want both models to be created. We also check the *Shared intrinsics* when we use images taken from the same camera.

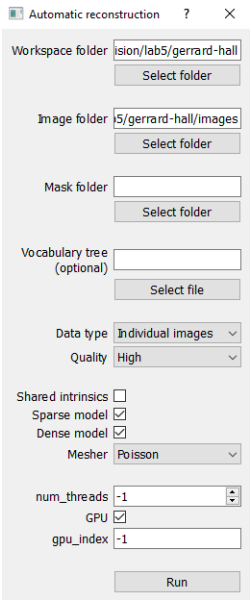


Figure 1: Automatic reconstruction modal.

Once COLMAP finishes the reconstruction, we can visualize the sparse model (see Fig. 2). Around the building, we can observe many noisy points.

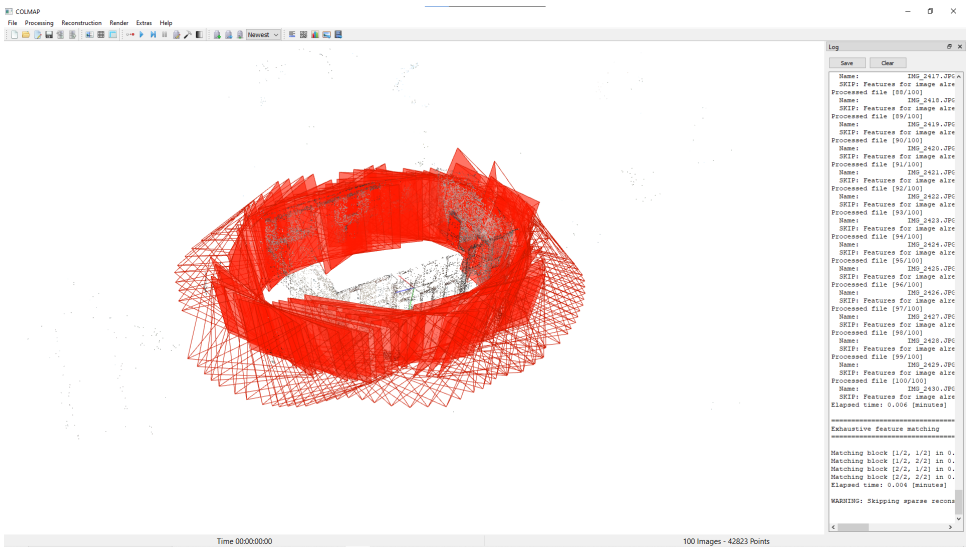


Figure 2: Sparse reconstruction of Gerrard Hall's dataset. The red planes represent the cameras for each frame.

COLMAP also generates a dense point cloud and a mesh reconstruction. For visualizing the dense reconstruction, we load the mesh reconstruction on Meshlab and visualize it (see Fig. 4).

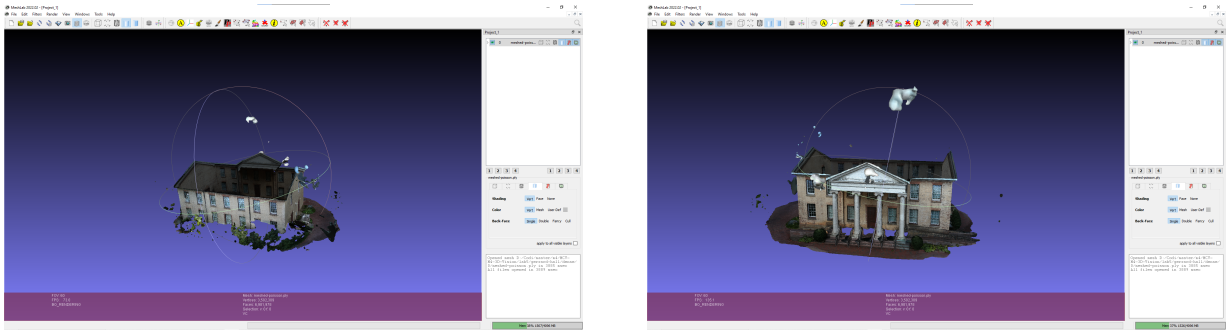


Figure 3: Dense mesh reconstruction of Gerrard Hall's dataset. We load and visualize the meshed polygon model generated by COLMAP onto Meshlab.

As we can see, there are some floating pieces that have been generated from noisy points. We remove these isolated pieces using the *Filters >Cleaning and Repairing >Remove Isolated Pieces (wrt Diameter)*. We set a maximum diameter of 1.01571 units (10%). After doing this, most of the noisy blobs are gone, however, there still remains a blob that is attached to the building, as it is not regarded as an isolated piece.

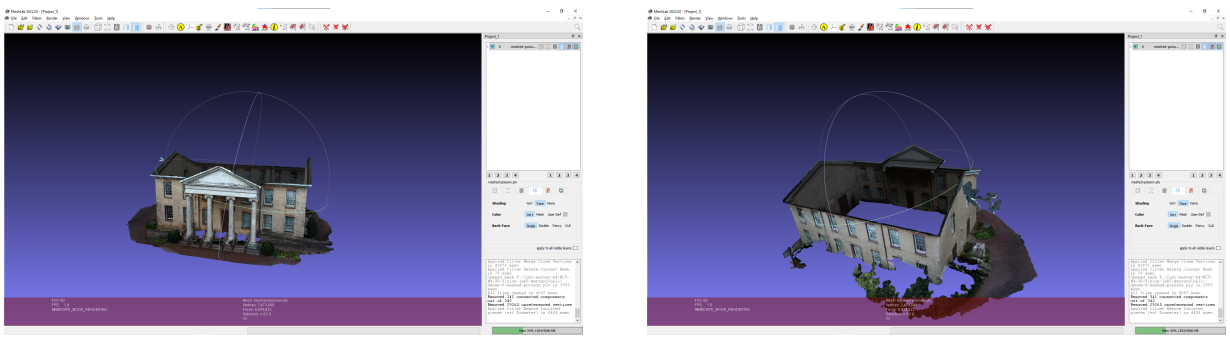


Figure 4: Dense mesh reconstruction of Gerrard Hall's dataset after removing noise.

2.2 Analysis of the reconstruction using Python

2.2.1 Gerrard Hall reconstruction data

COLMAP uses an SQLite database to store the keypoints and the matches corresponding to the steps of *feature extraction* and *feature matching*. Regarding scene reconstruction, it stores the different cameras information, the relationship between each image's keypoints and the 3D points they generate, and the 3D points information in 3 different files (*cameras, images and points3D*) both for the sparse and dense reconstructions. We can easily load this information in Python using *sqlite3* and the model class provided in the practicum's resources. To illustrate this, Figure 5 shows the visualization of the point cloud reconstruction using Python.

After loading the data, we find out that there are 1.061.700 keypoints in total (considering the keypoints of all images). We also check how many 3D points originated from a keypoint in the first image by summing all the 2D points that have a reference to a 3D point and we obtain 2.819. Figure 6 shows a breakdown of the detected 2D keypoints in red and, from all of them, the ones that have a reference to a 3D point in blue. We observe that the number of 3D is considerably low with respect to the number of keypoints, probably due to the difficulty of matching the correspondences and triangulation between so many images, i.e. 100.

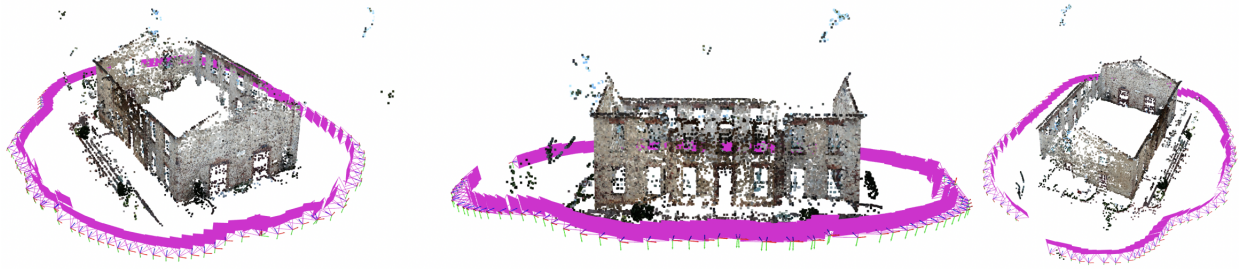


Figure 5: Visualization of the point cloud and cameras of the Gerrard Hall reconstruction (in Python).

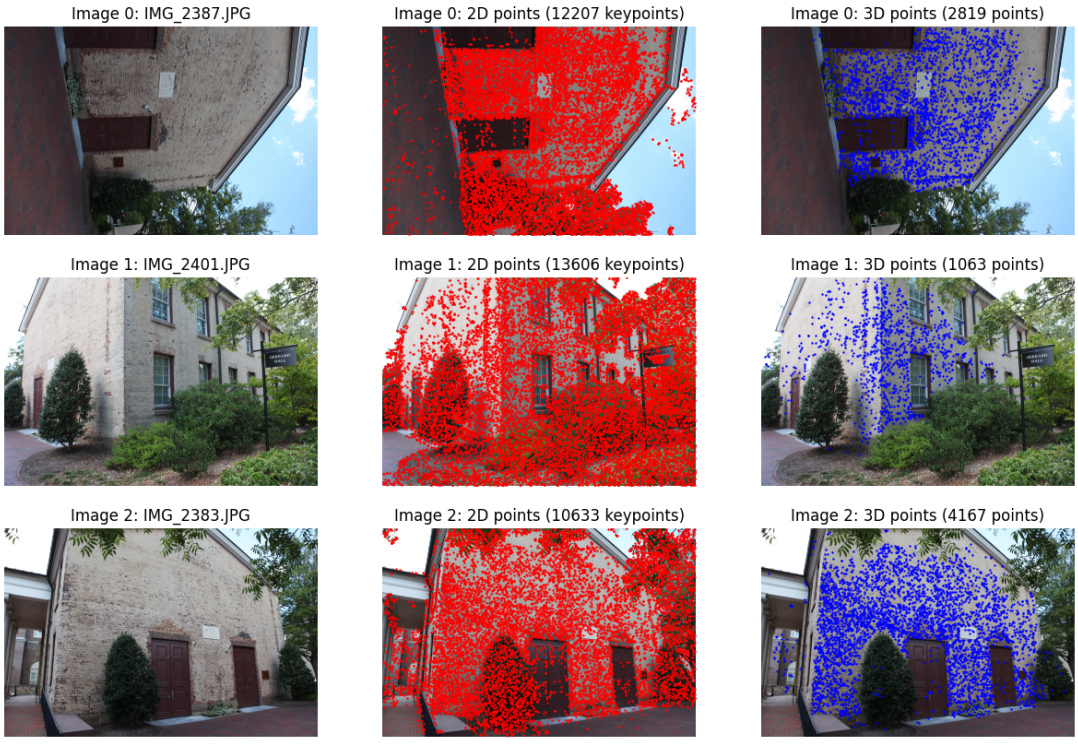


Figure 6: Visualization of detected 2D points (keypoints) and available 3D points of Images 0, 1 and 2 from the database.

2.3 Analysis of the 3D points according to the number of images and the error

In this section, we analyze the point clouds of 3D points of the reconstruction of Gerrard Hall's dataset. By using Python 3D point plots, we can select the `image_ids` for which a keypoint has a 3D reference (not marked as -1) and get its *xyz* coordinates to color each reconstructed point according to the number of images from which it originated, as shown in

Figure 7. As it can be observed, the point cloud mostly shows a packed blue region corresponding to points originated by a low number of images, namely between 1 and 7 images per point. However, some small regions of the building present a greenish-yellowish set of points that indicate that around 10 to 15 images originated them. As it is expected, the number of points originated from the highest number of images (25), i.e. the red points, are very scarce. This is because it is very unlikely to obtain the same exact 3D point of the world from too many different images due to the inherent inaccuracies between the feature detection and extraction, the matching and the reconstruction across all 100 images in the dataset.

Figure 7: Visualization of 3D points coloured according to the number of images from which it originated.

Next, we plot the 3D points coloured according to the reprojection error of each point that one can be obtained from the `point.error` attribute, with the color indicating the magnitude of the error. The reprojection error is a measure of how accurately the 3D points have been reconstructed and projected onto the 2D image plane. A lower reprojection error indicates that the reconstructed 3D points are a good match with the corresponding 2D keypoints in the images, while a higher error suggests that there is some discrepancy between the two. Note that the error is given in pixels of reprojection error and is only updated after global bundle adjustment. Figure 8 illustrates the above and verifies that the practically all points in the reconstruction have a small error of around 0.5 pixels. Also, only a small portion of 3D points are colored in yellow, with a slightly higher reprojection error of 1.5 pixels. In summary, a low error rate in the plot is a positive sign and suggest that the 3D reconstruction has been performed with high accuracy. This could indicate that the calibration of the camera, the quality of the image data, and the accuracy of the reconstruction algorithm used were all good.

Figure 8: Visualization of 3D points coloured according to the error.

2.3.1 Analysis of the 3D points corresponding to the keypoints in the first image

In this section, we analyse the first image of the dataset with the keypoints corresponding to the 3D points and its point cloud. In Figure 9, we compare the geometry of the scene in terms of the spatial distribution and relative position of the objects in the scene. For instance, some shape and orientations of objects can be observed in the point cloud, such as the patterns and structures of the main door, the facade and the roof of the building. Also, by comparing the 3D plot with the image plot, we can asses the accuracy of the 3D reconstruction, and identify any systematic errors or outliers in the data. Furthermore, with this analysis we can perceive how the quality of the image data can also affect the accuracy of the reconstruction. For example, regions such as the trees and vegetation of the bottom of the image lack of keypoints and 3D points due to factors such as lighting, focus or resolution on this part of the image.

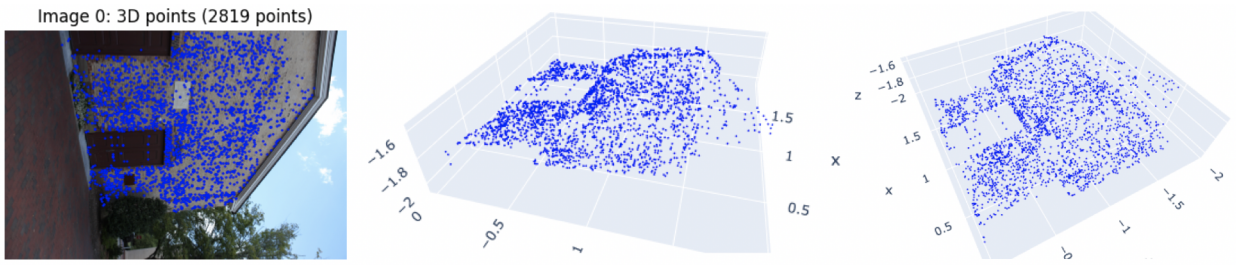


Figure 9: Visualization of the keypoints and corresponding 3D point cloud of Image 1.

2.3.2 Analysis of the number of matches between all images

In this section we perform an analysis on the number of matches between all images. More specifically, we build a heatmap by counting the number of matches and displaying the ids of the corresponding matched images. The visualization of Figure 10 provides information about the degree of overlap between different images. But it also provides details about the redundancy and quality of the image collection, as well as the performance of the feature matching algorithm used to find matches between the images. First, we can see that most of the pairs have a low number of matches: between 0 and 1500, being the mode 500. As many of the cells show low values, this could indicate that the images are not well overlapping, leading to gaps in the coverage of the scene. The heatmap also determines the level of redundancy in the image collection. If many of the cells in the heatmap show high values (it is not the case; very few pairs have more than $3500 - 4000$ matches), this could indicate that there is a high degree of redundancy, which could negatively impact the efficiency of processing the images. On the other hand, a low degree of redundancy can improve the processing efficiency. Additionally, if as the heatmap shows low values for certain image pairs, this could indicate that the images are of low quality or that there are issues with the image data, such as motion blur, poor focus, or low light. Finally, the performance of the feature matching algorithm can also be assessed based on the values in the heatmap. If the majority of the cells show low values, this could indicate that the matching algorithm is nor performing well.

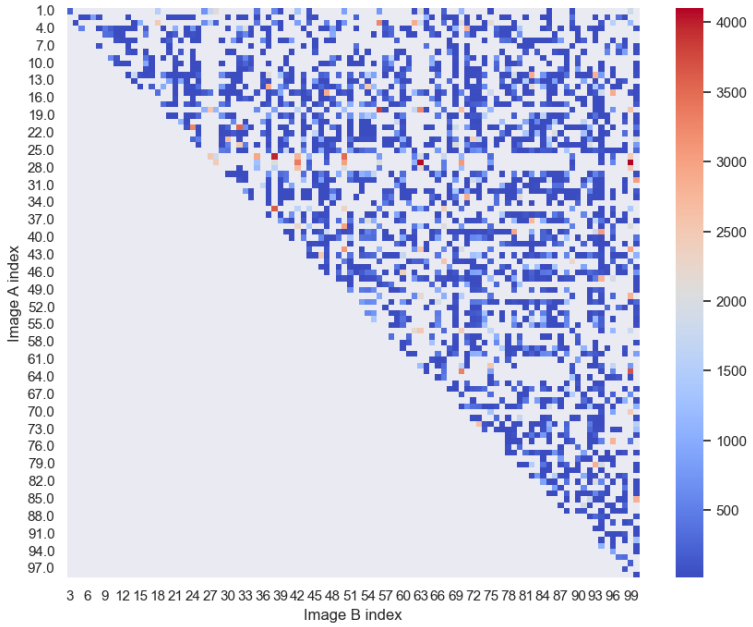


Figure 10: Visualization of the number of matches between all images.

2.3.3 Visualize keypoints and matches between two images of Lab 4 using Colmap

In order to visualize keypoints and matches between two images using Colmap, we followed a set of steps in the GUI. First, we created a new project and a new database, adding the images from the first and second image of the sequence. Next, in the ”Processing” menu, we selected ”Feature extraction” and set the parameters before running the feature extraction process. The program automatically extracted focal length information from the embedded EXIF infromation, which helps to ensure that the correct local length is used in the feature extraction process. Finally, in the ”Processing” menu, we selected ”Feature matching” and set the parameters before running the feature matching process. According to the documentation, the Exhaustive Matching algorithm is the most suited for this task, given that we are only processing two images.

Figure 11: Visualization of the keypoints of two images from COLMAP.

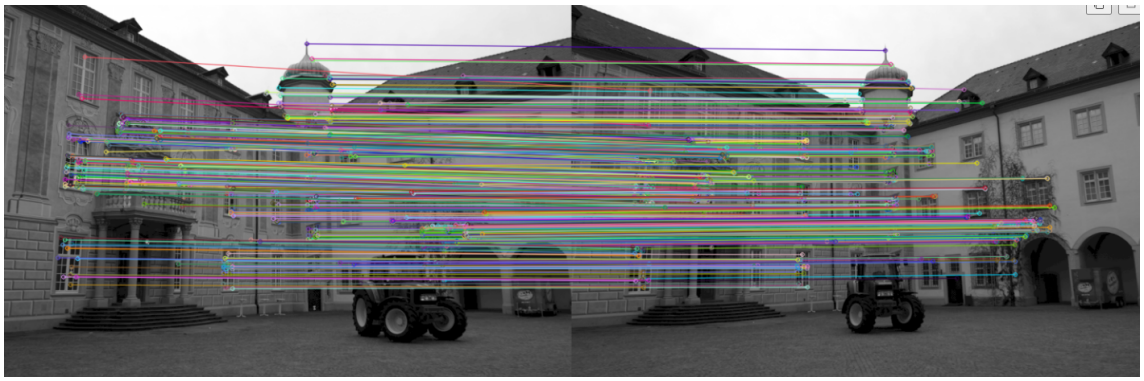


Figure 12: Visualization of the keypoints matches of two images from Lab 4.

Figure 13: Visualization of the keypoints matches of two images from COLMAP.

The most relevant difference we notice is the number of matches found, which in this lab is far larger (see Figs. 12 and 13). This is due to the fact that in the previous lab, we detected up to 3000 keypoints, while COLMAP sets the maximum number of keypoints to more than 8000. Additionally, the use of embedded EXIF information to automatically extract the focal length information is also a useful aspect of the process. This helps to ensure that the correct focal length is used in the feature extraction and matching steps, which can impact the accuracy of the results. In conclusion, there are more keypoints and matches between the two images using Colmap, and they seem to be consistent in comparison with the approaches from last week and suggest that the results are comparable and of good quality.

2.3.4 Triangulation using Colmap's keypoints and matches

Now, we repeat the same procedure we carried out last week to find the cameras and triangulate the points from both images.

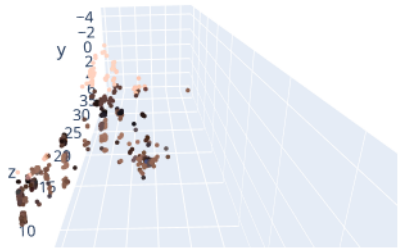


Figure 14: Visualization of the 3D points of the triangulation from lab4 (colors inverted).

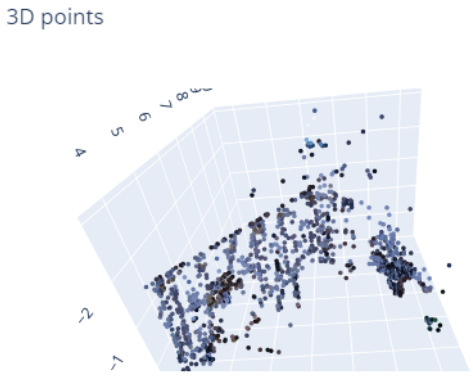


Figure 15: Visualization of the 3D points of the triangulation using the keypoints and matches from COLMAP.

As in the previous section, we notice that there are a lot more 3D points in the triangulation from the keypoints detected in COLMAP (see Fig. 15). However, we also notice that there is more noise in the COLMAP triangulation, while in the lab4 one (see Fig. 14), there are barely noisy points. This indicates that there is room for improvement and tuning the parameters of the feature detection and matching in COLMAP. Also, the COLMAP triangulation has more resolution than the other one, and we can even grasp out some of the shapes in the building.

2.4 Visualize the sparse reconstruction from the Castle images

Finally, we use COLMAP to generate the sparse reconstruction in *Reconstruction >Start reconstruction*. In this case, we do not change the default parameters.

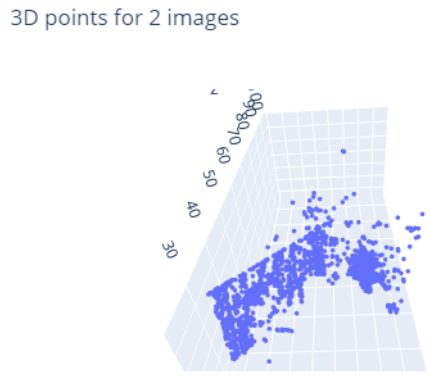


Figure 16: Visualization of the 3D points of the sparse reconstruction using the 2 images from Castle.

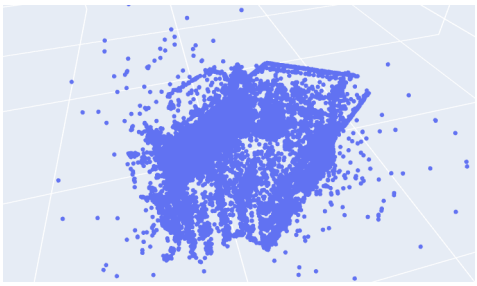


Figure 17: Visualization of the 3D points of the sparse reconstruction using all the images from Castle.

Then, we load the sparse model in Python and plot the 3D points reconstructed by COLMAP. This time, we see that the reconstruction using 2 images (see Fig. 16) is very similar to the triangulation using the technique from last week. This suggests that the selection and number of keypoints may have a significant impact on the final triangulation. However, we would have to further explore the parameters of COLMAP to know the impact of the technique on the final result.

We also made a sparse reconstruction using all the images from the Castle set and we achieved a more detailed and complete reconstruction of it. We can appreciate some shapes such as windows or contours. Nonetheless, there are a lot of outliers floating around that would have to be dealt with.

3 Improving the reconstruction

After creating the sparse reconstruction of the Castle, we move forward to the dense reconstruction. First, we use the *Automatic reconstruction* from COLMAP (medium quality) and we used the result as our baseline to compare against (first image in Fig. 18).

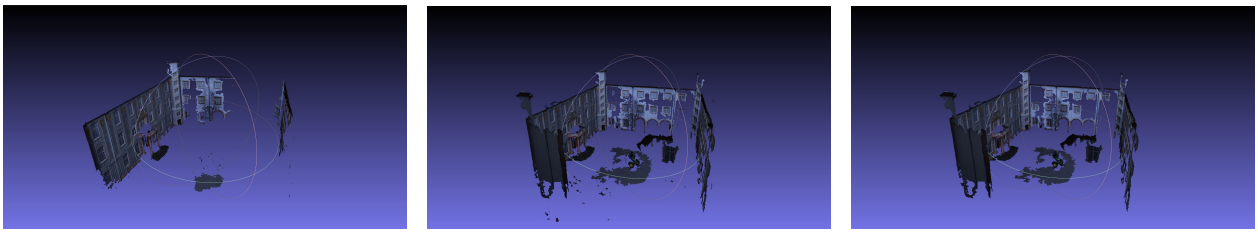


Figure 18: Dense mesh reconstructions of the Castle using COLMAP. The first image corresponds to the automatically generated mesh, the second one to the improved model's mesh and the third one the improved model's mesh without noise.

In order to achieve a better result, we tune some of the parameters from the steps we've been using before (feature extraction, matching and sparse reconstruction). These are the following:

- **SiftExtraction.max_num_features**: The more features, the longer it will take, but the more correspondences and more coherent the reconstruction will be.
- **SiftExtraction.peak_threshold**: Determines how much a keypoint must stand out to not be discarded. If there are noisy keypoints, increasing this value is helpful, but it will result in fewer keypoints.
- **SiftMatching.max_ratio**: The Lowe's ratio test. If there are noisy matches, increasing this value is helpful, but it will result in fewer matches.
- **SiftMatching.min_num_inliers**: The minimum number of inliers needed to relate two images. Increasing this value will result in fewer related images but less noise. If there are few images, it is better to decrease this value.
- **Mapper.min_num_matches**: When performing 3D reconstruction and attempting to relate images (I think), only pairs of images with a minimum number of matches are considered. If there is a sparse point cloud, it may be interesting to decrease this value.
- **Mapper.min_model_size**: The number of images a "sub-scene" must have to be considered. If there are few images, it is better to decrease this value, or a lot of the scene may be discarded.
- **Mapper.tri_min_angle**: Decrease this value if the images are taken from a distance (as is the case).

As the Castle dataset consisted of just 30 images, and because we observed that the floor and the more flat surfaces were problematic, our strategy was tweaking these parameters so we generated more keypoints in spite of the noise this could cause. That is, we wanted to paint all the surfaces we could. Finally, we removed noisy blobs using Meshlab as we did with the Gerrard Hall reconstruction. In the end, we achieve a better result than using automatic reconstruction. However, we were just able to run a few experiments because of the execution speed of the application, so there is still room for improvement by just tweaking these and other parameters.

4 Reconstruction of Our Selection of Images

In this section, we gather several proposals as examples of 3D reconstruction and meshing. First, we propose a reconstruction of museum images taken in the wild Figure 19a. In this matter, `colmap` performs as expected. This is because it's an easy object to navigate circularly without losing any detail and, therefore, obtaining a good global view (Figure 19b) with a lot of intersections which allows the algorithm to effectively perform keypoint matching for even small details (Figure 19c).

We also provide results from images extracted from Google StreetView, more precisely on the Sagrada Familia building Figure 20. At first sight, we observe that both dense and sparse models can't perform properly. This may be due to the lack of variation in the captured images, being the camera static to a point and, therefore, not varying enough the extrinsics. Nevertheless, we also provide results using the software `MeshRoom` [1] (which we find easier to use than `colmap` as it presented fewer installation problems, better performance, compatibility and a way more intuitive interface based on nodes (process) an edge (communication between ongoing data). In Figure 21 we observe the results with `Meshroom` which due to easier experimentation because of all the mentioned advantages, allowed us to better adjust the parameters and obtain better results. In this context, the images from StreetView were enough to reconstruct this scene.

5 Readings

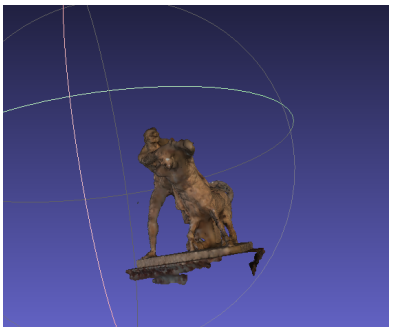
In this section, we comment on three readings having to do with 3D scene modelling from 2D images. For the first two papers, we give a brief overview and comment on the strengths and weaknesses of the proposed method. In the case of the third work, having to do with neural radiance fields, we compare it against the previous two.

5.1 *Structure-from-Motion Revisited*

This paper (Schönberger and Frahm 2016) deals with the structure from motion (SfM) problem, where the goal is to reconstruct a 3D rigid scene from an unordered image collection. The authors propose a multi-step, incremental approach that is more robust, accurate and scalable than prior state of the art. They adopt the typical incremental SfM pipeline, shown in Figure 22.

[1] https://alicevision.org/#meshroom

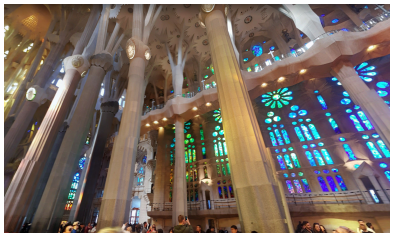
(a) Example of a photograph of Hercules and the centaur Nessus sculpture.



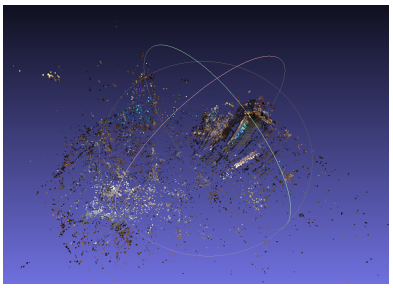
(b) Mesh result for sculpture, we observe that the global structure is very well preserved.

(c) Closer look at the sculpture mesh where we observe a very detailed mesh for the point cloud.

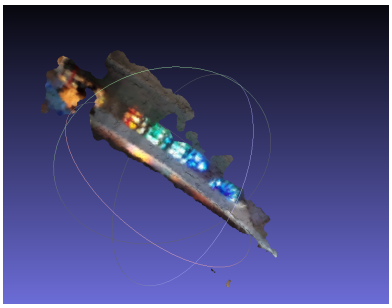
Figure 19: Summary for the sculpture 3D recovery using `colmap` and `meshlab`.



(a) Sample image for the generated dataset of Sagrada Familia from Google StreetView

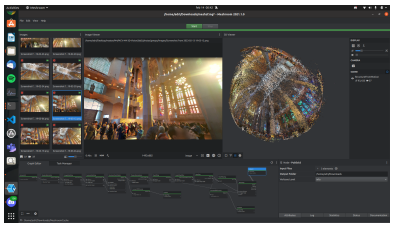


(b) Sparse representation of the data of Sagrada Familia

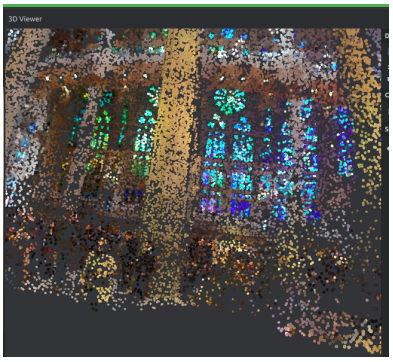


(c) Dense representation for the data of Sagrada Familia

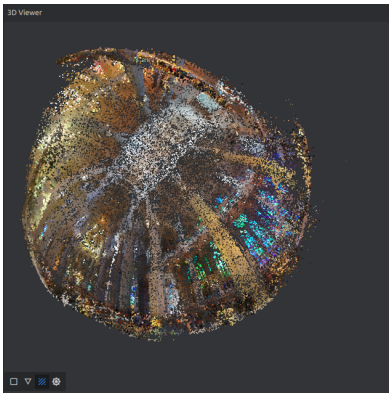
Figure 20: We observe a bad behavior on the reconstruction from the data extracted from google maps. This may be because of the lack of movement, with the caera being a static element with constant extrinsics.



(a) Example of working with `Meshroom`



(b) Inner view of Sagrada Familia reconstruction by `Meshroom`.



(c) Bottom view of Sagrada Familia reconstruction.

Figure 21: Working with `Meshroom` allows us to perform further experimentation in terms of SfM and, therefore, optimizing properly the parameters in order to obtain a good result. It performed faster and allowed us to tweak the pipeline by adjusting the nodes such as SfM, keypoint detection, matching....

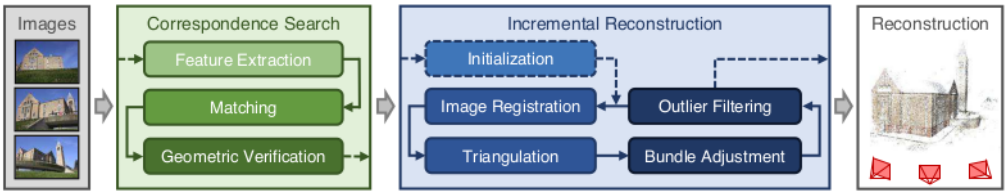


Figure 22: Incremental Structure-from-Motion pipeline. Figure from the original paper.

This pipeline is made up of several components, each of them critical to the final reconstruction result. The paper introduces key improvements in each of the stages, allowing it to outperform prior state of the art, as shown by the experiments. The whole process consists of two main steps: correspondence search and incremental reconstruction. In the former, features are extracted from and matched between images, obtaining a scene graph. This graphs seeds the model with a two-view reconstruction, before incrementally registering images, triangulating scene points, and performing bundle adjustment.

In what follows, we elaborate on the strengths and weaknesses of the method, tracing our analysis along the chain of operations in the SfM pipeline. We then conclude this section with a final discussion.

5.1.1 Correspondence Search

Given a set of input images, the goal is to find projections corresponding to the same scene point. As output, we want a set of geometrically verified image pairs and a graph of image projections per point. Notice that if input images contain little to no overlap, the method will just simply find *few or no* correspondences, rendering the subsequent incremental reconstruction useless.

Feature Extraction. For each image, local features are detected and extracted. These features have to be invariant under radiometric and geometric changes, in order for SfM to uniquely recognize them in multiple images. Thus, the choice of features is critical. SIFT, its derivatives and more recent learned features are proposed in the paper, but it is obvious that if the feature extractor is not suitable to the given scene, the performance will degrade.

Matching. Next, SfM uncovers pairs of overlapping images, leveraging the features. The naive but optimal brute force approach is computationally unfeasible, so a suboptimal feature matching algorithm must be used. Note that both the distance metric for comparing features, and the matching algorithm itself are hyperparameter choices.

Geometric Verification. SfM verifies matches by estimating the transformation that maps feature points between images. If the scene is planar, a homography may be used; otherwise, one can rely on the Essential or the Fundamental matrices. If the algorithm fails to properly decide on the mapping type (e.g. a homography for relating two views is computed for a non-planar scene), then the geometric verification will have failed.

At this stage, another weakness is that in order to decide whether a transformation is appropriate, it must map a sufficient number of features, this threshold being yet another hyperparameter. What is more, the estimation of this mapping is embedded in a RANSAC procedure, which despite eliminating noise, introduces randomness and more compute into the algorithm.

5.1.2 Incremental Reconstruction.

Given the previously computed scene graph, the camera poses and the reconstructed scene points are estimated.

Initialization. Because the method incrementally reconstructs the scene by adding images to the set of considered projections, the initial pair of images has a strong impact on the final result. The authors note that finding a suitable pair is critical, as the reconstruction may never recover from a bad initialization.

In the paper, it is suggested to initialize from a dense location in the image graph, as the higher redundancy of overlapping cameras typically results in more accurate reconstructions. In contrast, though, initializing from sparser locations leads to lower runtimes. Thus, it may be necessary to sacrifice some accuracy to achieve reasonable execution times. Moreover, from the paper, it is not totally clear how to perform this initialization.

Image Registration. Given a starting metric reconstruction, new images can be registered by solving the Perspective-n-Point (PnP) problem. The choice of PnP method may negatively affect performance, and is probably dependent on each particular scene and set of available data.

Estimating pose for calibrated cameras involves RANSAC (which introduces randomness) and a minimal pose solver, which must be chosen based on domain knowledge. Additionally, note that in this incremental scheme, the next image selection matters, and is not trivial. The authors propose a robust method of next best image selection.

Triangulation. At this point, we note that a newly registered image must observe *existing* scene points. If the selected image does not overlap with any of the previously selected ones, it must be discarded. Moreover, at this point is is clear that there can be no discontinuity in the set of input images, otherwise a subset of these images will not be incorporated into the reconstruction, as they are disconnected from the rest. The authors propose a method for triangulation, as they note prior art suffers from limited robustness or high computational cost.

Bundle Adjustment (BA). In this step, the joint non-linear refinement of camera parameters and point parameters minimizing reprojection error is carried out. While efficient direct algorithms exist for up to a few hundred cameras, larger dense collections necessitate indirect algorithms. The authors propose a method for identifying and parameterizing images for efficient BA of dense collections. They incorporate a weighting function ρ_j in the form of the Cauchy function for downweighting outliers. They also provide the option of sharing camera models among any combinations of images, reducing the amount of camera models to be estimated.

5.1.3 Discussion

As can be inferred from our previous analysis, the main weakness of this method is its complex, handcrafted nature. The approach contains several crucial steps, each demanding careful attention. Moreover, in-depth domain knowledge is required to tune the algorithm's hyperparameters, hindering its broader application in industry by non-experts. For example, the choice of the inliers threshold and number of iterations the different RANSAC procedures depends on the scene and available data.

Moreover, it is hard to see how to process multiple scenes in parallel, since at each stage, different scenes will most likely require different runtimes. This hinders the applicability of the method in high-demand scenarios, where multiple users query a single server. For instance, if we were to design a mobile app that allows users to generate a 3D mesh of a scene, we would struggle to jointly process each of the potentially thousands of queries, as each scene requires different runtimes, and perhaps more critically, different hyperparameters. To insist, it is hard to see how to automate the whole process, such that it can be robustly run on many different real scenarios and data regimes, as in-depth knowledge of the system's components is needed to adjust them, and since the method is totally sequential, one badly adjusted step leads to poor performance in the next steps, potentially accumulating errors.

On the other hand, the experiments demonstrate that the method can indeed produce accurate reconstruction in reasonable times, at least when compared to prior state-of-the-art. The testing datasets contain 144,953 unordered photos, distributed over a large area and with highly varying camera density. This is a rather extreme case where each of the pipeline's components are put under stress. The same kind of features (RootSIFT) and matching parameters are employed across all experiments, showing that performance is robust to the type of features and matching. Besides the experiments in the original paper, the discussed approach has been extensively used in academia and industry, which means it has been already tested on many different scenes, proving it effective and reliable.

Regarding the next best view selection, an experiment exemplifies that the proposed method produces more accurate reconstruction by choosing a better ordering of the images. When it comes to triangulation, the authors show that their proposed approach recovers significantly longer tracks, with more track elements. In addition, they prove that the RANSAC-based approach is much faster.

The evaluation of the system as a whole is also presented in the paper. Overall better performance is reported for all datasets, especially for larger models, while retaining competitive runtimes.

Lastly, despite there being several steps along the chain, a human may in principle understand each of them, and relate the challenges in each of them to the final reconstruction results, because they are based on classic techniques with clear mathematical foundations. This means that unlike pure end-to-end deep learning methods, the reviewed approach can be interpreted, both in terms of its final and intermediate outputs.

5.2 *Pixelwise View Selection for Unstructured Multi-View Stereo*

The second work we discuss (Schönberger, Zheng, et al. 2016) introduces a multi-view stereo (MVS) system for dense modeling from unstructured image collections. The method performs MVS with pixelwise view selection for depth/normal estimation and fusion.

MVS leverages multiple views to overcome the inherent occlusion problems of two-view approaches. Accordingly, view selection plays a crucial role in the effectiveness of MVS.

The proposed method simultaneously considers a variety of photometric and geometric priors ameliorating upon a previous depth estimation probabilistic graphical framework (Zheng et al. 2014). Here we do not elaborate on this previous work, and focus solely on the additional contributions. The authors advocate for jointly estimating depth and normal information, as well as carrying out a pixel wise view selection using photometric and geometric priors. Additionally, they introduce a geometric consistency term for simultaneously refining image-based depth and normal fusion.

Keeping the same structure as in the previous section, we comment on the strengths and weaknesses of the method as we present its components, and finalize this section with a general discussion.

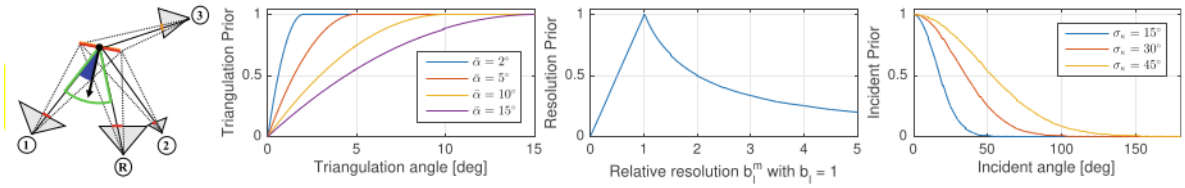


Figure 23: the work is about these priors

5.2.1 Normal Estimation

While (Zheng et al. 2014) map between the reference and source images using frontoparallel homographies, which leads to artifacts for oblique structures, the authors estimate per pixel depth and normals. They incorporate the normals as variables in the probabilistic formulation from (Zheng et al. 2014), giving a strong sense of how they are integrated in the whole model. To avoid having to sample more images due to the increase in variables (as the normals are also computed), they introduce a geometrically based sampling scheme exploiting first-order image smoothness. This scheme has the strengths of being both effective and explainable.

5.2.2 Geometric Priors for View Selection

In contrast to prior work, which decouples inference and per-image geometric priors by preselecting source images, the method integrates geometric priors on a per-pixel basis into the inference. Thus, the priors are given an interpretable probabilistic formulation that directly affects the result.

The motivation for including such priors is that occlusion boundaries, triangulation angles, relative image resolution, and incident angle can vary significantly between a single pair of reference and source images (Fig. 2). The geometric priors leads to a more comprehensive and robust pixel-wise view selection.

Triangulation Prior. The authors note that with a purely photometric, similarity-based image sampling, the most similar images are those having a very similar viewpoint, which coincides with a small baseline. However, smaller baselines do not carry information about depth, because reconstructed points can arbitrarily move along the viewing ray without changing the color similarity. To alleviate this, the authors introduce a measure of stability of a reconstructed form in the form of the angle. This gives an idea of how confident the triangulation is for a given point. Again, this gives a way of probing into the model that deep neural networks do not have.

Resolution Prior. The patches in the reference and source image should have similar size and shape: similar size is favorable as it avoids comparing images captured at different resolutions, and similar shape avoids significantly distorted source patches caused by different viewing directions. Thus, they propose a measure for quantifying the similarity between two patches, and introduce it as a probabilistic it term. We note thoug that this measure will fail for textureless areas, as no information about shape or scale can be inferred there. For instance, a flat blue sky will yield the same patches at multiple locations, and at multiple image scales, rendering the measure is meaningless.

Incident Prior. Here, the authors make use of the insight that, by construction, the camera location can only lie in the positive half-space defined by the plane induced by the depth and normal, while the camera viewing direction must face towards the opposite normal direction. Thus, a likelihood function is introduced, encoding the belief in whether this geometric constraint is satisfied. Something elegant about this is that instead of putting a hard constraint on the problem, the authors soften it and introduce it as an interpretable probabilistic term.

Integration. The work very clearly lays out how to combine the aforementioned priors into the inference. Intuitively, non-occluded images with sufficient baseline, similar resolution, and non-oblique viewing direction are favored in the view selection. We note though that the formulation assumes statistical independence of the priors, which eases optimization, but is not necessarily true.

5.2.3 View Selection Smoothness

To reduce state oscillation during optimization and stripping effects in the final result, a ”temporal” (that is, over the iteration index) smoothing term is introduced. Although the authors do not explicitly mention it, this smoothing could limit the expressivity of the model. Nevertheless, the authors experimentally show that the proposed smoothing reduces view sampling noise.

5.2.4 Photometric Consistency

Although NCC (as employed in (Zheng et al. 2014)) is statistically optimal for Gaussian noise, it is especially vulnerable to producing blurred depth discontinuities. Thus, NCC is adapted to used bilateral weights. Despite not having statistical guarantees, and depending on the statistics of the samples (unlike vanilla NCC), bilateral NCC, is also shown to lead to better reconstructions at occlusion boundaries.

5.2.5 Geometric Consistency

An additional likelihood term for encouraging geometric consistency is added to the joint model. This term is based on the insight that the estimated depths and normals from multiple views are consistent if the reprojection error is small. Here we find at least to downsides to the author's approach. One is that despite most photometric ambiguities being unique to individual views, this is not the case for textureless regions, in which case comparing multiple views does not help. Secondly, the authors do not consider occlusion indicators in the source image for the backward projection, and rely on an approximation instead.

5.2.6 Integration of all Components

We find that all the above priors are elegantly integrated into a single probability function that can be probed into for further insight and understanding about the model. Moreover, the algorithm employed for solving the problem, the Generalized Expectation Maximization (GEM), has well known properties.

5.2.7 Filtering and Fusion

To further eliminate noise and outliers, especially in the case of textureless regions, the authors propose a simple filtering scheme, after which they carry out the fusion of the depths. One aspect we liked about their approach is that a parameter $\bar{\psi}$ allows controlling the resolution of the fused point cloud. Moreover, since the estimated points already have normals, one can easily construct a surface mesh as an optional step.

5.2.8 Discussion

We find the article to be very detailed yet clearly explained. Unlike in the previously reviewed method, there are a few parameters that need to be tuned. This implies that the proposed algorithm could be used by practitioners who may not have much in-depth knowledge about the topic.

One of the inherent strengths of MVS methods, and thus of this one in particular, is that they leverage dense multiple views to overcome the occlusion problem present in two-view approaches. Additionally, depth map fusion methods integrate multiple depth maps into a unified and augmented scene representation while mitigating any inconsistencies among individual estimates.

A particular downside we observe and is reported by the authors themselves is the relatively high runtime of the algorithm. Despite using an NVIDIA Titan X GPU, it takes about 40s to integrate every single view from the *Middlebury* benchmark. This is despite the fact that the algorithm lends itself to massive parallelization in row- and column-wise propagation and the view level. Further improvements are necessary to make the algorithm deployable in real time.

Finally, an extensive ablation study showing the benefit of each term and of the combination of them all is reported. We find that it clearly answers the question of whether each of the individual components is helpful, and whether the whole system improves over prior state-of-the-art. We wonder though if learning-based, data-driven approaches could be introduced as priors, instead of solely handcrafted ones.

5.3 *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*

We end with a comparison of an influential work dealing with a new kind of scene representation, NeRF, with the two previously reviewed methods.

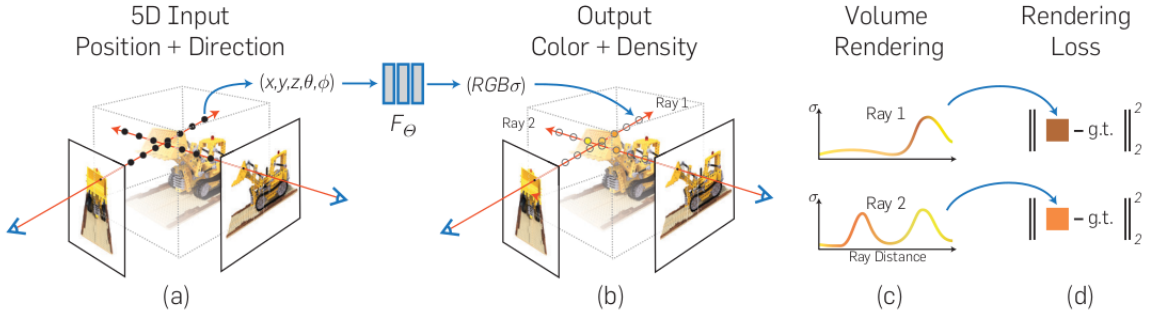


Figure 24: Overview of the neural radiance field representation and volumetric rendering. Images are synthesized by sampling 3D points and viewing directions along camera rays (a), feeding those into an MLP to produce color and volume density (b), and utilizing a volumetric rendering equation to composite these values into an image (c). The whole pipeline is differentiable, and is trained by minimizing the square photometric error (d) between the synthesized and the projected images. Figure from the original paper.

The main fundamental difference between the two methods described above (which we will collectively refer to as COLMAP) and NeRF is that the former deals with the problems of Structure from Motion and Multi View Stereo, whereas NeRF aims at novel view synthesis. Thus, we would use COLMAP to generate a set of 3D scene points, given their projections on multiple images, or to infer depths and normals from a dense collection of images, while we would turn to NeRF for generating novel views. Notice though that both COLMAP and NeRF share a very basic assumption, namely that the underlying 3D scene is rigid. Morover, like COLMAP's MVS approach, NeRF benefits from utilizing multiple overlapping views to disambiguate occlusions, but also colors.

NeRF assumes that the camera parameters are known a priori, while COLMAP is capable of carrying out bundle adjustment on the cameras. In fact, NeRF relies on COLMAP for precomputing camera matrices, if unknown.

Comparing Figures 22 and 24, we can clearly see that NeRF looks much less complex, as it has much fewer intermediate steps, at least at a broad conceptual level. COLMAP relies on classic computer vision techniques, such as feature extraction with SIFT, feature matching, and triangulation. NeRF builds on the representational capacity of neural networks and the strong inductive prior introduced by the choice of inputs (point coordinates and camera viewing directions) and the rendering equation. Note how each step in COLMAP is handcrafted whereas the whole NeRF pipeline is end-to-end differentiable and amenable to gradient descent-based learning.

While COLMAP takes into account several geometric and photometric constraints to drive the reconstruction, NeRF only supervises the pipeline with a photometric loss function in RGB space.

To work properly, both COLMAP and NeRF need resonably good weather and lighting conditions, with enough illumination and without excessive reflections, or transparent objects.

In terms of computational efficiency, COLMAP seems to be much faster. COLMAP needs at most a few hours to process a few hundred images, while for NeRF, training times of over a day are reported for about 30 images. Moreover, even after training NeRF needs a non-negligible amount of time to synthesize novel views, as it must query all directions passing through each pixel of the novel image. In passing, we also remark that COLMAP can be applied in principle to any scene (save perhaps tuning some parameters), while NeRF must be trained specifically to each scene.

As output of the COLMAP pipeline, we get a set of 3D scene points or a depth/normals map. After training a NeRF, we obtain an MLP with a scene compressed in its weights. Thus, while COLMAP deals with a heavier explicit representation of the scene, NeRF can be seen as implicitly representing 3D geometry and lighting via a light-weight implicit neural field.

NeRF does not assume in principle coplanar imaged points, nor a fixed camera rotating about its centre, nor any other particular spatial image pair configuration (though we note that despite not being explicitly mentioned in the original paper, it has been observed that NeRF needs all cameras to be at about the same distance from the central object, and at about the same inclination to work properly). COLMAP may perform SfM with varying spatial image pair configurations, but must explicitly handle this problem and choose the most suitable configuration; and when performing MVS, it assumes that the reference and source images can be related via a homography.

With regards to the implementation, COLMAP is coded in C/C++, in contrast to NeRF, which is programmed in Python (at least until recently).

Finally, we can say that COLMAP belongs to the field of computer vision, as it only performs analysis of the images, while NeRF is at the intersecion of computer vision and graphics, because it deals with the analysis of 2D images, but at the same time incorporates a rendering step.

6 Conclusions

We reconstructed the Gerrard Hall's 3D scene using COLMAP. We first realized that the automatic reconstruction can produce noise and outliers, so a post processing step with MeshLab is necessary. In this scene, we ended up with a low error rate, probably because the views were quite dense, allowing for good calibration and MVS. Interestingly, we realized that most images have no matches, but some do have a few thousands. This indicates that even if most images are not directly connected in the graph, a path exists between them, which allows for the dense 3D reconstruction.

In order to improve the reconstruction on the Castel dataset, we modified some of COLMAP's parameters. Because of the low number of images, our initial reconstruction was having trouble with the floor and flat surfaces, as well as overall sparsity. Thus, we tweaked some hyperparameters leading to more keypoints being generated and matched. This indeed improved ours results.

We also performed SfM on our own set of images. We explored images taken from a budget phone as well as from Google StreetView. We were able to properly reconstruct the detail of a complicated sculpture because our images were dense and encircled the whole object, aiding physical consistency. We managed to properly reconstruct a scene in Sagrada Familia by switching from COLMAP to MeshRoom.

Lastly, we reviewed three highly influential works having to do with the modelling of 3D scenes from 2D data. Unsurprisingly, all three methods assume rigid static scenes, probably because this keeps the complexity of the tasks they address tractable. The first two works, referred to as COLMAP, represent the state-of-the-art for 3D reconstruction with classic handcrafted computer vision techniques. NeRF, on the other hand, represents a novel and exciting new approach for modelling 3D scenes in a learnable fashion with computer graphics techniques. We highlighted some strengths and

weaknesses in COLMAP's approach for SfM and MVS. We saw how the SfM pipeline involves several complex steps, some of which necessitate careful tuning of their parameters, for which in-depth domain knowledge is required. However, the experiments from the original paper and the extensive use of COLMAP in research and industry show that this method is robust across many different scenes. As for the MVS method, a highly interpretable and intuitive probabilistic framework is laid out, though with the downside of particularly high runtimes. In comparison to those two methods, NeRF appears as a highly innovative solution to the problem of novel view synthesis, yet perhaps more interestingly, it offers a powerful means of representing 3D scenes through a neural implicit field that instead of explicitly outlining the scene's contents, can be made to learn a continuous representation of the underlying 3D geometry and appearance.

6.1 Feedback on this lab session

With regard to the optional reading task, we find that the assigned papers are highly relevant to the lab's contents and the course as a whole, and therefore applaud the choice.

With regard to the available software, our experience was hard and rough (even in Linux environments); it is extremely difficult to install, slower than we are used to, and not very intuitive. After trying `MeshRoom`, we realized it allowed us to do everything required for this lab absolutely for free with a more (in our opinion) intuitive UX which allows one to perform modifications at any point in the pipeline, resulting in a way easy experimentation, For example, if we wanted to change something in the keypoint matching, we shouldn't run keypoint detection since it will be repeating work. In this sense, MeshRoom is very nice.

We find the generation of dense reconstructions very insightful and pleasant. Nevertheless, we feel that the pyplot reconstructions are not that motivating.

References

Schönberger, Johannes L. and Jan-Michael Frahm (2016). "Structure-from-Motion Revisited". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4104–4113. DOI: 10.1109/CVPR.2016.445.

Schönberger, Johannes L., Enliang Zheng, et al. (2016). "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, pp. 501–518.

Zheng, Enliang et al. (2014). "PatchMatch Based Joint View Selection and Depthmap Estimation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1510–1517. DOI: 10.1109/CVPR.2014.196.