# SfM applications for 3D reconstruction from 2D avionics industrial inspection videos

Master Thesis

Thesis report

by

Arbër Demi

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on Wednesday, 17 July, 2024 at 10:30

| | |
|---|---|
| Supervisors: | Dr. J. C. van Gemert |
| | Dr. Y. Lin |
| *Thesis committee*: | |
| Chair: | Dr. J. C. van Gemert (Associate Professor) |
| | Dr. M. Weinmann (Assistant Professor) |
| | Dr. Y. Lin (Post Doc) |
| Project Duration: | November, 2023 - July, 2024 |

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Delft University of Technology · Aiir Innovations

# Preface

This report marks the final stage of my Master's thesis project, titled "SfM applications for 3D reconstruction from 2D avionics industrial inspection videos", conducted at the Delft University of Technology to obtain the degree of Master of Science. The project was conducted within the Computer Vision Lab at TU Delft in collaboration with Aiir Innovations.

I would like to express my gratitude to Dr. J. C. van Gemert for his guidance and supervision during the project. I am also very grateful to Dr. Y. Lin, my daily supervisor, for his constant feedback, flexibility and support. Both significantly helped with the direction and finalization of my work. I also would like to thank Jeroen Delcour and Steve Nowee from Aiir Innovations for their suggestions throughout the thesis.

I want to thank my family, especially my brother Ensar and my mother Arjeta who made this journey possible, my friends who shared a similar journey, and my girlfriend Andini for her support.

# Part I

## Scientific Article

# SfM applications for 3D reconstruction from 2D avionics industrial inspection videos

Arbër Demi, Yancong Lin, Jan C. van Gemert
Delft University of Technology

## Abstract

*Structure-from-Motion (SfM) and Neural Radiance Fields (NeRFs) have significantly advanced 3D reconstruction in multi-view scenarios. Despite their success in handling non-repetitive, texture-rich scenes, applying such techniques to real-world scenarios with texture-less and repetitive structures remains challenging. One such case is in industrial inspection processes, specifically the reconstruction of aircraft engine blades. The goal is to build a 3D model of all the engine blades from a single video. We explore the application of SfM in modeling repetitive and texture-less blades, identify common causes of failure, and improve upon the default SfM by replacing the commonly used exhaustive match with sequential match to handle ambiguity stemming from repetitiveness. Sequential matching enables more precise pose estimation and better 3D reconstruction in our scenario. In addition, we explore the importance of choosing the correct camera model and provide a comparative look at the existing 3D mesh reconstruction solutions, presenting tweaked versions that result in better performance. This work lays the foundation for 3D reconstruction of repetitive and texture-less objects by proposing sequential matching, enabling better 3D modeling of engine blades compared to classic SfM pipelines.*

## 1. Introduction

The inspection process for aircraft engines and their blades is commonly done manually through the use of borescopes, which are optical tools used to view areas that are hard to reach. The use of the borescopes allows for more freedom in camera motion in the tight spaces of aircraft engine interiors, which is highly useful for the inspecting engineers [38]. Footage obtained can be used in more ways than manual inspection, with this paper following the footsteps of BladeNeRF [11] and taking a closer look at 3D reconstruction of the engine blades for automatic defect detection. The data used for this work is obtained through a static camera which is constantly pointing towards the rotating blades at some angle and distance, a similar context to [11]. For our context, we interpret this as our camera rotating while taking footage of static blades, as shown in Fig. 1.

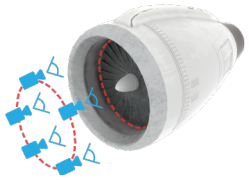

**Figure 1.** *Camera of a borescope following a circular pattern while facing the blades. image from [11].*

The constraints of camera motion, together with the often texture-less surfaces of the engine blades, create a challenging setup for accurate sparse, dense and mesh 3D reconstruction. This setup poses more technical challenges for evaluation due to the lack of ground-truth camera poses.
Our work is concerned with photo-realistic 3D reconstruction of the aircraft engine blades using the footage obtained from the inspection camera, following the main goal of BladeNeRF [11]. Differing from BladeNeRF, the focus of this paper is on the Structure-from-Motion (SfM) techniques [26, 27], which heavily rely on visual features, such as SIFT [19] to calculate camera poses and reconstruct a sparse 3D model.

In [11], the SfM techniques are pushed aside due to the texture-less surface of the aircraft engine blades. This becomes a more challenging problem when considering that most SfM pipelines default to exhaustive matching of features in an attempt to use as much information as possible. In our case, however, exhaustive matching hurts the performance of SfM by matching together distinct blades due to their visual similarity.

There are also many other issues in this setting as shown in [16, 20], one of which is the lack of camera intrinsics. Among those intrinsics is the distortion information. Many borescopes have lenses that introduce some radial distortion. Without the correct camera model to estimate the distortion, or the correct distortion coefficients, the camera pose estimation and quality of sparse reconstruction

suffers, as shown in [20].

In this work, we take a look at some changes that improve the SfM pipeline, allowing it to produce consistent 3D reconstructions of repetitive engine blades. A popular piece of software that provides a full SfM pipeline implementation is COLMAP, which includes work on both sparse and dense reconstruction [25, 26]. Hierarchical Localization [23, 24] builds on top of COLMAP, having code abstractions that lead to easier experimentation of feature extraction and matching, which is why we use Hierarchical Localization as foundation for our work.

The availability of numerous feature extractors and matchers, such as DISK [30] and LightGlue [18], makes SfM a more viable solution for sparse reconstruction and camera pose estimation. We make an attempt at the 3D reconstruction of engine blades using a combination of both DISK and LightGlue.

In addition, we resolve the issues resulting from exhaustive matching by using sequential matching only on a collection of neighboring frames. Despite its simplicity, it proved useful for video footage of engine blades.

Lastly, we test numerous camera models with the goal of finding the best one for accurate reconstruction of the camera path and the shape reconstruction of a blade fan.

On top of the Structure-from-Motion techniques, we need dense and mesh reconstruction techniques to represent the detail required for defect detection. These techniques often require the sparse reconstruction and camera poses as prior information to function. Neural Radiance Fields (NeRF) [21] is a technique that is commonly used for this purpose, specifically for synthesizing novel views of a scene. NeRF overfits a multi-layer perceptron (MLP) to a scene and uses volume rendering techniques with color and density projection for image rendering. A known limitation of NeRF is its performance with limited numbers of input views. Although it can correctly render images that follow alongside the given camera poses during training, any deviations from those poses show errors, and the 3D volume extracted from the model is incorrect. Meanwhile, DS-NeRF [8] makes use of sparse reconstructions coming from techniques such as SfM to add depth supervision to NeRF. This leads to much better results for the 3D geometry of reconstruction. We test NeRF and DSNeRF through their performance in modeling 3D engine blades from a single video.

A different type of dense reconstruction comes in the form of Multi-View Stereo [12, 25], with Poisson Reconstruction [13] allowing for full surface reconstruction from a given point cloud. These approaches can produce very detail rich results, however small issues exits in parts of the pipeline that require more work to be able to produce full visualisations. We also use Poisson Reconstruction to model 3D blades and compare it with NeRF-based reconstruction.

In summary, this work presents these contributions:

- We propose a novel SfM pipeline, featured by sequential matching, that enables 3D reconstruction of repetitive engine blades.
- We test the capabilities of NeRF and its variants in modeling blades from a single video sequence, captured by a front-facing camera and we unveiled the common failure cases of NeRF models.
- We provide a comparative analysis between different dense/mesh reconstruction methods.

## 2. Related works

**Structure from Motion.** SfM has gone through many changes over the years, prompting a couple of main methods to take the forefront. Global SfM [7, 29] was the first approach which followed the original proof of SfM having a solution [31]. This approach attempted to solve the problem by using the 2D coordinate information from the extracted features. These coordinates are then grouped together in one large matrix alongside the camera centers we need to estimate, and solved in a sparse linear system. One recent work did an in-depth study on the effect of different visual features used during the SfM pipeline [2], with the conclusion that SIFT features are still the most reliable. Among the feature matchers and extracters compared is SuperGlue [24], a neural network that matches features by both finding correspondences and rejecting points that don't match, similarly to its lighter version LightGlue [18]. Both networks use SuperPoint [9] for feature extracting. However, the focus in [2] remains on the global SfM method, while assuming that the camera intrinsics used are of good quality, which is not the case for our study. Adding to that, the data used is not similar to our blade data, as it is not texture-less or focusing on secluded objects. For these reasons, we keep using the default from Hierarchical Localization; a combination of DISK [30] and LightGlue [18].

Not long after global approaches were developed, incremental approaches [4, 28] were proposed as well, where images are registered one after the other instead of simultaneously. This approach has often been proven more robust than the global version. Later, there were follow-up developments on error-correcting processes like bundle adjustment, triangulation, or outlier rejection, all of which can be seen in [26].

**Neural Radiance Fields (NeRF).** A well-known technique for recovering 3D scenes from multiple views of images is NeRF [21], which encodes the 3D scene through a fully connected network. Many follow-up works of NeRF [3, 5, 8, 14] are dependent on pre-calculated camera poses to function, while other NeRF works handle the camera pose

estimation process on their own. BaRF [17] uses a fine-to-coarse positional encoding for camera pose optimization, while NeRF– [34] jointly optimizes both the NeRF model and camera parameters by minimizing the photometric reconstruction errors. Unfortunately, NeRF and many of its simpler variants have failed to provide high-quality 3D reconstructions when encountering texture-less scenes [34, 36] or limited numbers of views.

**Depth-aware solutions.** DSNeRF [8] is one of the works that makes use of the sparse reconstruction information that commonly accompanies the camera poses from processes like SfM. This is done by adding a depth loss on top of the photometric loss NeRF uses. NerfingMVS [35] adopts a similar strategy, but uses sparse depth information to train a depth network that provides better adapted depth priors for supervising NeRF. NerfingMVS is using not only the information from the SfM process, but also Multi-View Stereo from COLMAP [25] (MVS). There are also many non-NeRF related solutions for 3D reconstruction, with NeuS [32] trying to solve issues present in NeRF by introducing a new volume rendering method to remove the geometric bias present in NeRF. 3D Gaussian Splatting [14] on the other hand, continues the use of pre-computed sparse points in order to help represent the scene with 3D Gaussian functions. In general, making use of the sparse reconstruction information from SfM has shown to consistently improve performance of these methods. We use DSNeRF and MVS + Poisson Reconstruction to show the importance of depth information for 3D dense and mesh reconstruction of engine blades.

# 3. Methodology

## 3.1. Incremental Structure from Motion

To obtain a sparse reconstruction alongside estimated camera poses from our video, we rely on Incremental Structure-from-Motion (Fig. 2) techniques.

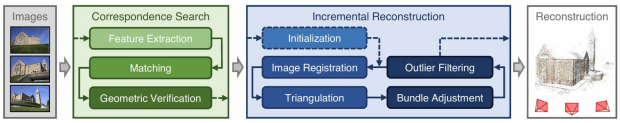


**Figure 2.** *An overall view of incremental structure from motion [26]. After feature extraction and matching, geometric verification assists with quality control of matches. The reconstruction is then initialized with two images, with new images being registered past that. Triangulation, bundle adjustment and outlier filtering follow to obtain the best camera pose estimates and reduce reprojection error. The process repeats until all images are registered.*

We build our model on top of the Hierarchical Localization repository [23], which uses COLMAP for 3D reconstruction through python bindings [10].

Due to the constraints imposed by our data, such as total lack of camera information and texture-less blade surfaces, we end up having a couple of problems:

- **Lack of object texture** - SfM defaults to exhaustive matching, which makes SfM think two distinct blades are the same.
- **Lack of proper camera intrinsics** (mainly distortion) - SfM is sensitive to even small errors in intrinsics, which will botch camera pose estimation and 3D sparse reconstruction.

To handle these problems, we make several changes to incremental SfM. One of the first changes made to the pipeline was ensuring the use of **sequential matching** over exhaustive matching. Regardless of the feature extractor used, blades that are hundreds of frames apart could have a considerable amount of feature overlap between them, leading to incorrect matches when using exhaustive matching. The two main reasons to use sequential over exhaustive matching is to exploit the real-time nature of the video and to mitigate issues caused by the texture-less surfaces of the blades by only matching nearby views.

The second change we made is in handling radial distortion, which is commonly found in borescope footage. The camera models we use are SIMPLE_RADIAL, RADIAL and OPENCV_FISHEYE [1]. The key difference between them is their ability to model radial distortion.

For an isolated look at the effect of modeling the camera distortion wrongly, smaller segments of the video are compared between the distorted and undistorted versions of the data. Distortion is first modeled with SIMPLE_RADIAL, RADIAL and OPENCV_FISHEYE, then the images are undistorted through COLMAP.

We use DISK [30] and LightGlue [18] to extract and match features for the SfM process. After obtaining the sparse reconstruction and camera poses from the SfM process, we continue to dense and mesh reconstruction.

## 3.2. Dense/mesh reconstruction

To obtain the dense and mesh reconstruction, NeRF, DSNeRF, and Multi View Stereo + Poisson Reconstruction are used.

For NeRF and DSNeRF, different data segments are used to test the consistency of meshes extracted from the trained NeRF models. The lengths of the segments are chosen for shorter training times and smaller meshes, while still keeping some growing distance to emphasize differences between results.

A modified version of the NeRF implementation [1] is used as a baseline for this study, while keeping the performance of the base NeRF implementation. To extract meshes

[1] This modified implementation can be found at the NeRF_pl repository [22].

from the NeRF and DSNeRF models, an existing script from the NeRF_pl [22] repository is used, with marching cubes used for the mesh and vertex normals to aid color extraction.

For DSNeRF we test its general performance, and similarly to NeRF, we take a closer look at the effect in the end result of the mesh when using different data segment lengths. We also test the effect of putting a larger and smaller emphasis on the depth loss introduced by DSNeRF by varying its weight.

Multi View Stereo and Poisson Reconstruction are used without any changes from the default COLMAP implementation.

## 4. Experiments

The impact of our proposed changes is tested through simple visual comparisons of the reconstruction results from the following aspects: pre-processing of data, sequential matching, radial distortion, performance of NeRF and DSNeRF and performance of Multi View Stereo alongside Poisson Reconstruction.

The images used below will be a mix of fully colored and solid colored images, as part of the results are obtained in COLMAP or visualized from Hierarchical Localization (fully colored) while some are viewed in MeshLab (solid colored) [6].

### 4.1. Pre-processing data

Given a video as input, we extract frames at a rate of 30 frames per second, resulting in 3632 frames of moving blades. All frames are cropped to 480 x 360 pixels, and the empty black borders around the image are removed.

**Figure 3.** *The original frame extracted from the video. The small black square on the bottom left is replacing a small transparent icon in the full video.*

We split the data into segments of different lengths for 3D reconstruction. At first, a segment of 40 frames is used to obtain early results as it is approximately the amount of frames a blade needs to go out of view while a new blade comes into view (without fully overlapping the previous position). Later, we also use segments of 250, 400, 990, 1300, and 3632 frames for denser and complete reconstruction.

The matching window is set to 15 neighboring frames (15 frames ahead, and if that is not possible, 15 frames behind).

Part of the structure shown in the video does not move alongside the blades. We remove it from the frame information after seeing the effect it had on the reconstruction, as shown in Fig. 4 and Fig. 5. There is a large gap between the blades and the background from the image, which is noticeable in the left section of Fig. 5 and is not present in the image itself. Larger segments tested (250-400 frames) failed reconstruction completely. Therefore this error in depth can affect the rest of the decisions made throughout the SfM process.

**Figure 4.** *On the left is the original frame extracted from the video, on the right is the cropped frame, with the static part blacked out.*

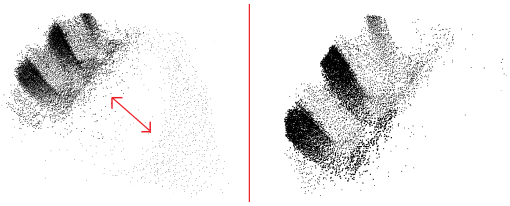

**Figure 5.** *Two 80 frame segments are shown with no color. On the left a sparse reconstruction from images with the static information included, on the right it is removed. The arrow on the left shows the distance between the blades and the static information.*

In addition, much more noise is introduced in the reconstruction, and the time for the reconstructions shown in Fig. 5(using default parameters by COLMAP) is much slower when using the frames with background information, at **7.2 minutes** for reconstruction, compared to **2.1 minutes** without the background information on a single NVIDIA 3070 Ti GPU. This is attributed mainly to bundle adjustment being able to converge faster due to less conflicting information per image.

Lastly, two more important remarks about the data: the blades stop moving for about 100 frames midway, which are removed from the frames used for the experiments, as they do not present any new information. Alongside that, this particular video presents some out-of-plane shifting in several sections of the video. The camera shakes slightly, causing the image to shift inwards or backwards momentarily in a few moments in the video. This may be a cause for some of the issues seen later with spiraling (Fig. 12).

### 4.2. Sequential matching instead of exhaustive matching

The default setup of the SfM sparse reconstruction is exhaustive matching, as creating matches between all possible images is beneficial in most real-world cases. However, for the given setup, it is harmful. This is mainly due to the texture-less and repetitive surface of the blades, which, as mentioned in Section 3, is a problem for matches. These problems are shown in Fig. 6 and Fig. 7.

Given the same data segment, if we instead use sequential matching, the obtained results match more closely our expectations (Fig. 8). These results are achieved with a 15-frame matching window. Notably, similar results can be obtained with values anywhere in the range of 5-20.

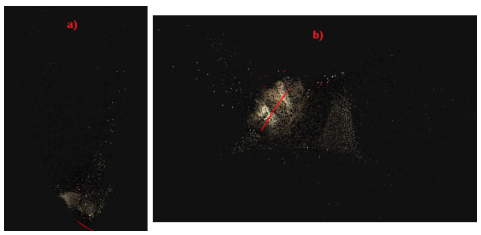


**Figure 6.** *Pictures show different perspectives of the same 400-frame reconstruction using exhaustive matching. A large amount of scattered noise can be seen, with varying colors. On top of that, we see only 2 blades, where 400 frames should show 10 blades.*

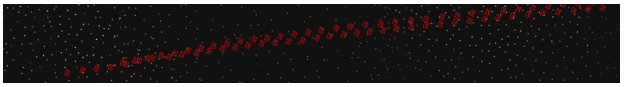

**Figure 7.** *The red frustums represent the estimated camera poses, where there is visible overlap between frames, even though there are no repeated sections in the 80-frame segment used for this reconstruction. Blades overlap in a cycle of approximately 40 frames.*

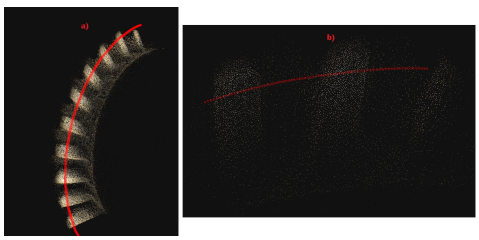


**Figure 8.** *Picture **a)** shows a 400 frame segment sparse reconstruction using sequential matching, where all 10 blades can be seen, and there is little noise. Picture **b)** shows a closer look at the cameras in an 80 frame segment, where the poses are not overlapping and follow a proper line.*

### 4.3. Camera models and camera intrinsics

In pursuit of a full reconstruction, we observe issues due to the lack of proper camera intrinsics.

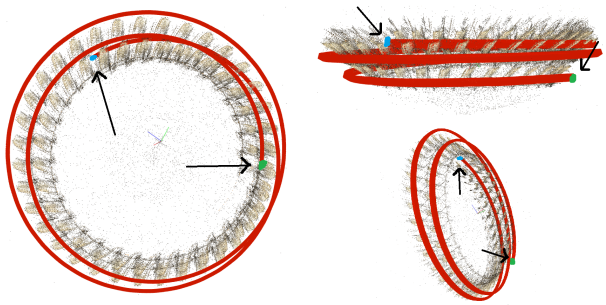

**Figure 9.** *A full reconstruction from all 3624 frames. The camera path and the blades are spiraling out of plane and creating seemingly 2 circles, instead of one closed circle. With blue is marked the starting point of the reconstruction process, and green the end.*

Larger reconstructions (Fig. 9) reveal that there are significant errors propagating throughout the reconstruction process. This can be seen more closely while going through the reconstruction path from the start. At the section near the blue mark there are more consistently angled blades, however as time goes on, they angle more towards the inner side of the circle.

As borescopes are known to have radial distortion, one of the first failure points experimented with is the camera model. Different camera models show us that the distortion level we are working with requires more distortion coefficients to model accurately. If we use the distortion parameters estimated from camera models:

- SIMPLE_RADIAL (has one distortion coefficient, made for simple/low distortion)
- RADIAL (has two distortion coefficients, can model higher levels of distortion)
- OPENCV_FISHEYE (has four distortion coefficients, can model much higher levels of distortion)

we can undistort the images and then feed them back to the sparse reconstruction process, where we see wildly varying results.

Fig. 10 shows 4 reconstructions of a 990 frame segment, which is approximately 98 degrees out of a 360 degree coverage, which we use as our ground truth. Accuracy of results is being measured in angles to compare to our ground truth in Tab. 1.

Fig. 11 shows a reconstruction with distorted data alongside one with undistorted data. Even though there is no actual scale based on real world values, the size within the model is more consistent with the reconstruction using undistorted data.

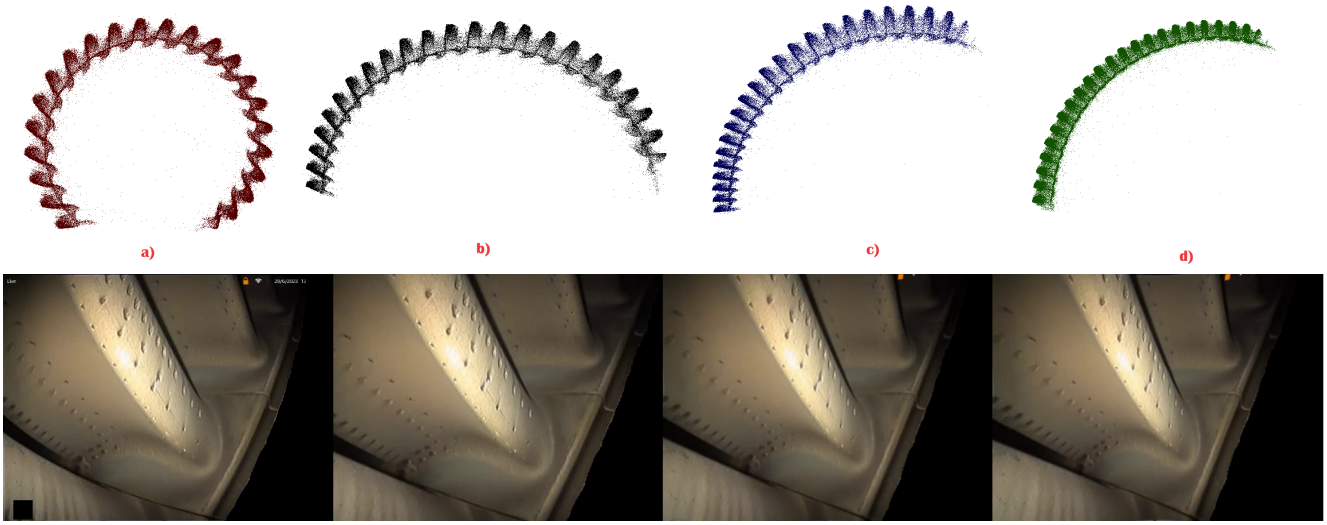


**Figure 10.** *Picture **a)** shows a reconstruction with SIMPLE_RADIAL camera model, using distorted images as they were extracted from the original video. Picture **b)** is also using SIMPLE_RADIAL, however the images have been undistorted using the estimated distortion coefficient from SIMPLE_RADIAL. Following the use of undistorted images, Picture **c)** shows a reconstruction using RADIAL camera model, and Picture **d)** shows a reconstruction using the OPENCV_FISHEYE camera model. Each reconstruction is accompanied with an example of the data used.*

| Reconstruction | Camera model | Angle measured (degrees) ↓ |
|---|---|---|
| Reconstruction **a)** | SIMPLE_RADIAL | 290 |
| Reconstruction **b)** | SIMPLE_RADIAL | 175 |
| Reconstruction **c)** | RADIAL | 107 |
| Reconstruction **d)** | OPENCV_FISHEYE | **102** |
| Ground truth | Unknown | 98 |

**Table 1.** *Comparison of angles between the four different reconstructions in Fig. 10, with **a), b), c) and d)** representing four reconstructions. **a)** is obtained through distorted images, and **b), c) and d)** through undistorted ones with distortion coefficients extracted through SIMPLE_RADIAL, RADIAL and OPENCV_FISHEYE camera models accordingly. Reconstruction **d)** is the most accurate one.*

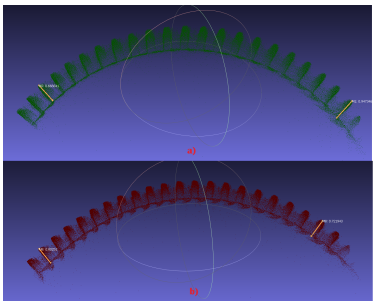

**Figure 11.** *Picture **a)** shows a reconstruction using distorted data while Picture **b)** uses undistorted data. Both reconstructions use OPENCV_FISHEYE camera model. The distance between the two ends in **b)** with **0.080** is much less than in **a)** with **0.2809**.*

Our results show that having the correct camera model to estimate intrinsics is very important for this type of data, and in our case SIMPLE_RADIAL with one coefficient and RADIAL with two coefficients are not enough to correctly model the distortion, while OPENCV_FISHEYE is much better. Unfortunately, using OPENCV_FISHEYE with undistorted data and increasing the amount of frames for full reconstructions does not produce a circular shape (Fig. 12).

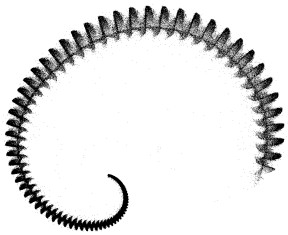

**Figure 12.** *Reconstruction of the full video using OPENCV_FISHEYE camera model with undistorted images. Start of reconstruction is on the right side of the picture, with it degrading by a large amount very quickly around the 1/4th of the way.*

It is worth noting that the angle of the section that looks to be more correct (the right side) is not similar to reconstruction **d)** from Fig. 10, which is not expected behaviour since we achieved much better results with a smaller data segment.

Further experimentation with different 990 frame segments showed that the results were depending on more out-of-view variables. A similar behavior to Fig. 13 can be seen with all camera models, leading to the conclusion that there are still errors propagating throughout the SfM pipeline. These errors create a similar spiraling and size inconsistency effect as was seen in Fig. 12. It is still unclear why

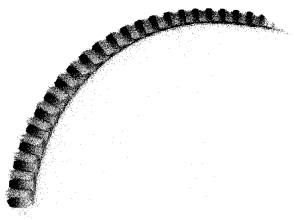

**Figure 13.** *Reconstruction of the second batch of 990 frames from the video using OPENCV_FISHEYE camera model. A clear disparity can be seen between the size of the blades on the left side compared to the right side.*

exactly this inconsistency occurs, however it is likely to do with the camera intrinsics not being estimated properly or perhaps the out-of-plane shift mentioned earlier in 4.1.

We did a short experiment to verify the effect of the out-of-plane shift, by testing two data segments that contain different levels of shift. One has minimal levels, while the other has a considerable amount. Unfortunately, the experiment was inconclusive and had inconsistent results, and therefore we leave this to future work.

Regardless of the issues, between the camera models OPENCV_FISHEYE has less angular errors and size inconsistency.

Although the results have not provided a full correct reconstruction, the SfM pipeline has proved capable of creating consistent sparse reconstructions under the correct conditions. Future work on this topic can likely solve the issues presented here through more testing.

### 4.4. NeRF and Depth-supervised NeRF performance

Assuming we have a correct sparse reconstruction, how good are the options we have for 3D dense/mesh reconstruction? We take NeRF [22] as our baseline. The sparse reconstruction and camera pose information fed to NeRF and DSNeRF [8] are obtained through SIMPLE_RADIAL camera model with distorted images from the original video.

**Implementation details.** Parameters are kept as default for NeRF_pl as the baseline [22]. We use the Adam optimiser [15] and a learning rate of $5 \times 10^{-4}$. A scheduler controls the learning rate, with a $\gamma$ of 0.5 affecting the learning rate every 10 steps, with 30 epochs in total used for training. Every run of NeRF needs around 8 hours of computation time on a single NVIDIA 3070 Ti GPU.

DSNeRF extends the main NeRF implementation through consideration of extra depth supervision through an additional loss variable, as can be seen in figure 14. This mitigates issues in depth with few input views, a known failure point of NeRF models. The information used for this loss comes from the known location of the sparse 3D points from the SfM pipeline.

For DSNeRF, we randomly sample 1024 rays and take 64 samples for each ray. We again use the Adam optimizer, and a learning rate of $5 \times 10^{-4}$. The learning rate decays exponentially every 1000 steps, where we are training for 50k iterations per 40 frames used. Every 50k iterations need around 8 hours of computation time on a single NVIDIA 3070 Ti GPU.

Both NeRF and DSNeRF are tested with segments of 40, 80, 150, 250, and 400 frames length. DSNeRF was further tested by either increasing the weight of the depth loss from 0.1 (default) to 0.4 and 0.8 for more emphasis on depth, or removing it entirely to showcase its importance.

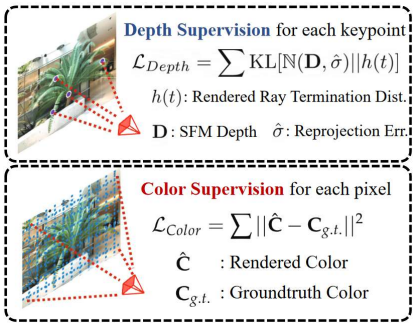


**Figure 14.** *The added depth loss alongside the color loss used in the base NeRF paper [8].*

**Evaluation criteria.** To evaluate the RGB reconstructions, we use PSNR, LPIPS [39] and SSIM [33] on top of manual mesh evaluation.

**Results.** Our results for NeRF are presented in Tab. 2, with example rendered images shown in Fig. 15 and a closer look at the meshes extracted in Fig. 16. Results for DSNeRF are presented in Tab. 3, example rendered image in Fig. 17 and meshes extracted in Fig. 18.

Results Tab. 2 show that the fidelity of the rendered images lowers with more data used to train the base NeRF model. Although results from smaller data segments show good detail Fig. 15, the accompanying meshes extracted Fig. 16 show the weakness of NeRF when there is a lack of different view points. The meshes are not geometrically consistent with what we expect compared to the sparse reconstruction, showing that NeRF is not a viable option for generating 3D meshes from this type of data.

In the meantime, DSNeRF results show less image detail but the depth supervision from the sparse reconstruction has a very noticeable effect on the depth extracted from our DSNeRF model, more clearly seen in the meshes extracted in Fig. 18, where we see results that are more consistent with the sparse reconstructions. The meshes shown can be extracted at higher quality with some tuning of the marching cubes method used, however we are more interested in knowing whether the results match with the sparse reconstruction.

| Data segment | PSNR ↑ | LPIPS ↓ | SSIM ↑ |
|---|---|---|---|
| 40 frames | **36.2** | **0.1321** | **0.9414** |
| 80 frames | 34.4 | 0.1457 | 0.9192 |
| 150 frames | 32.8 | 0.1778 | 0.8985 |
| 250 frames | 31.3 | 0.2857 | 0.8619 |
| 400 frames | 26.9 | 0.4223 | 0.7664 |

**Table 2.** *Data segments used to train NeRF models with their reported PSNR. Models with larger data segments were trained for more iterations to account for the static batch size used. The NeRF model trained with 40 frames performs best in these metrics.*

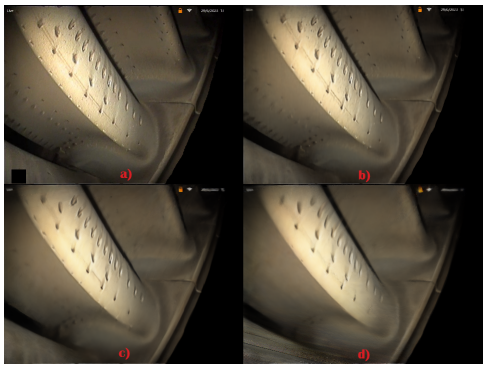


**Figure 15.** *Picture **a)** is the original image extracted from the video. Pictures **c) to d)** are the renders of NeRF models trained on 80, 250 and 400 frames respectively. These renders are from the estimated camera pose by SfM for the frame in Picture **a)**.*

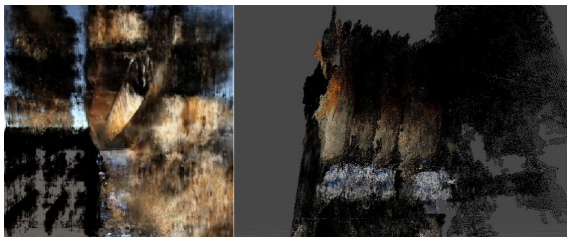

**Figure 16.** *Meshes extracted from trained NeRF models. On the left is a reconstruction from 80 frames, while on the right one from 400 frames. No clear shape can be distinguished that matches the expected results compared to the sparse reconstruction.*

In an attempt to achieve higher fidelity we initialized the base 250 frame model with the weights of the fully trained 40 frame DSNeRF model. This did not show a significant increase in rendered image quality.

There are not very significant differences shown when comparing the meshes extracted from the trained DSNeRF models with a higher focus on depth, however there is a very noticeable difference between the ones using the depth loss and the one without (Fig. 18). With higher resolution parameter tuning for marching cubes, and color added on the mesh, the differences would be more noticeable.

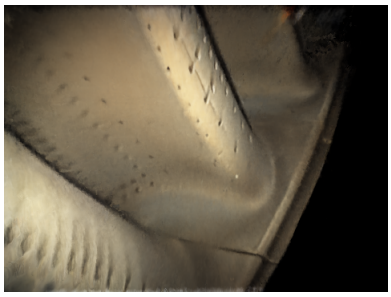

**Figure 17.** *Rendered image from the baseline DSNeRF model trained on 250 frame segment and initialized with weights of a 40 frame trained model. The render is slightly blurry and not very smooth.*

| Data segment | PSNR ↑ | LPIPS ↓ | SSIM ↑ |
|---|---|---|---|
| 250 frames (40 init) | 28.94 | 0.2761 | 0.8331 |
| 250 frames (0.4 depth loss) | 28.45 | 0.3112 | 0.8053 |
| 250 frames (0.8 depth loss) | 27.59 | 0.3565 | 0.7861 |
| 250 frames (0 depth loss) | **30.32** | **0.2096** | **0.8695** |

**Table 3.** *Data segments used to train NeRF models with their reported PSNR, LPIPS and SSIM. 250 frames (40 init) means that the network was initialized with the weights of a 40 frame trained network.The less depth is involved, the better the metrics report, however this does not take into account the quality of the mesh.*

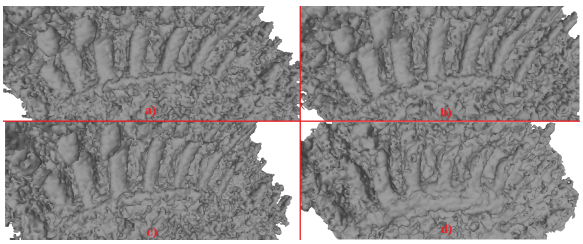


**Figure 18.** *Meshes extracted from trained DSNeRF models, trained on 250 frame data segments. Pictures **a), b) and c)** are using a depth loss weight of 0.1, 0.4 and 0.8 respectively, while **d)** has a weight of 0. The noise around the meshes can be filtered out using sparse reconstruction coordinates in the future. The first three are nearly indistinguishable with this resolution, however there is a clear difference between them and **d)**, where the surface of the blades has gaps and holes in it.*

### 4.5. MVS and Poisson Reconstruction

Similarly to NeRF before, we attempt to see how MVS and Poisson Reconstruction performs under the assumption that we have a correct sparse reconstruction.

MVS is a lengthy process, taking around 157 minutes for a 990 frame data segment on a single NVIDIA 3070 Ti GPU. Meanwhile, after we get a dense reconstruction through MVS, Poisson Reconstruction takes around 11 minutes for the same segment.

The results are shown in Fig. 20 for MVS dense reconstruction and Fig. 21 for Poisson Reconstruction.

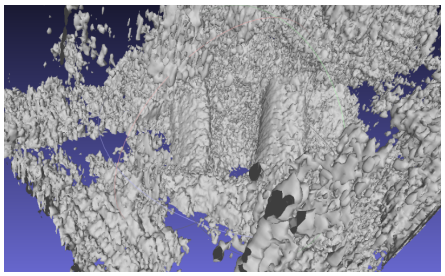

**Figure 19.** *The mesh shown is extracted from a DSNeRF model trained on a 40 frame data segment. Compared to the meshes in Fig. 18, the mesh is very inconsistent and the blades (at the center of the image) are surrounded by too much noise.*

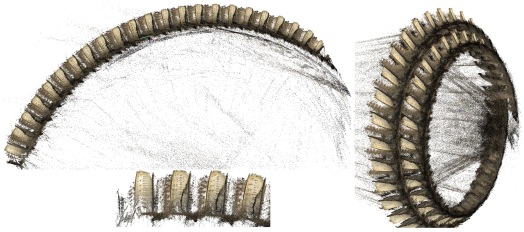

**Figure 20.** *Two separate partial (left) and one other full (right) dense reconstructions are shown. The dense point cloud shows a lot more detail than what was available in the sparse reconstruction, however more noise is introduced as can be seen by the points far away from the reconstruction.*

**Figure 21.** *The left image shows a small Poisson reconstruction which has more detail than the larger reconstruction on the right. The base settings for Poisson reconstruction lead to larger interpolation and less detail, which can be changed for high quality large reconstructions.*

The results for MVS are reasonable considering the level of detail shown, however the point cloud has patches of missing information and is noisy, containing a lot of black dots that were not present in the sparse reconstruction. Looking at the depth maps generated and used by MVS for the dense reconstruction in Fig. 22, a possible source of this noise can be identified in the edges of the image. Attempts to replace the depth maps with externally extracted ones (Fig. 23) that show no errors were unsuccessful due to format mismatch (format used by COLMAP for depth map storage compared to externally extracted depth maps).

Results for the Poisson Reconstruction are very promising with the detail shown, however the patches and noise introduced in the dense reconstruction inevitably appear here as well, since the dense reconstruction is used as a starting point for the mesh creation.

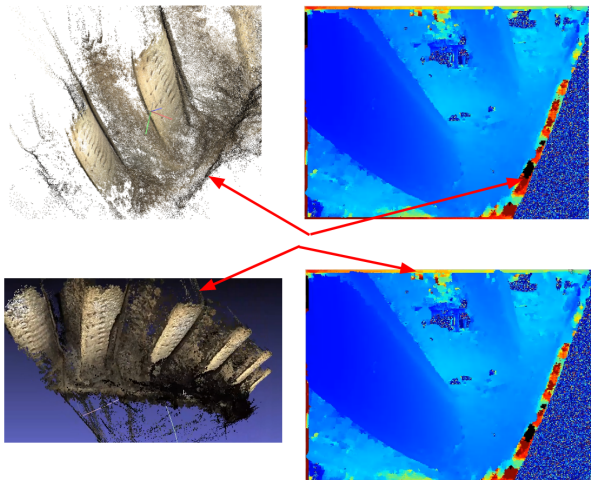

**Figure 22.** *On both image pairs you can see a comparison between abnormalities in the depth maps and noise created in the dense reconstructions. This noise did not appear in the sparse reconstructions.*

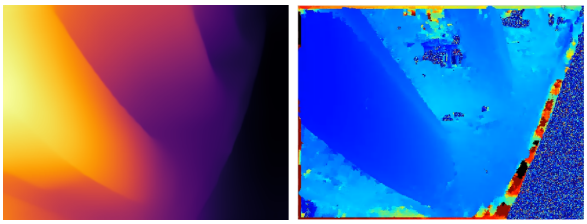

**Figure 23.** *The left image shows a depth map extracted through Depth Anything [37], the right image shows the depth map from COLMAP.*

Both MVS and Poisson Reconstruction show great promise for dense and mesh reconstruction accordingly, however the depth maps generated and used are not appropriate. Further work to adapt external depth maps would very likely solve the noise issues.

## 5. Discussion and Conclusion

We present a novel SfM pipeline for forward-facing aircraft blade inspection footage benefiting mainly from the use of sequential matching with a tight neighbouring matching window. In addition, we show the importance of choosing the correct camera models when estimating the intrinsics in order to achieve results comparable to the inspection video. Although the best performing camera model showed promise with some partial sections, it was not able to fully reconstruct a full 360 degree circle of blades, revealing some remaining errors in the pipeline.

To address this performance, larger scale testing and pa-

rameter tuning with camera models and other videos of different data could prove very insightful and possibly introduce solutions. Testing with different types of data could lead to a more robust camera calibration setup.

Most camera calibration procedures require access to the camera in order to use checkerboard/chessboard like objects for the calibration. As we assume that the camera is not available, a possible route would use the blade-edge lines on the sides and bottom of the blades to assist with the camera calibration process, specifically using the parallel positioning of the lines. Another possible improvement would be adding a new loss in the Structure-from-Motion pipeline, in order to ensure all of the camera poses remain on the same axis. This would be useful in cases where we know beforehand that we are dealing with a video which contains all 360 degrees of the blades.

In addition, we present a comparison between NeRF and DSNeRF for 3D mesh reconstruction alongside results from the MVS dense + Poisson mesh Reconstruction duo. NeRF is shown to follow results previously reported in literature, with issues appearing due to the lack of variety in points of views and the texture-less nature of our data. DSNeRF proved to be much more useful for 3D mesh reconstruction, with the extracted meshes following along the lines of the sparse reconstruction used for supervision. The tuning of the depth loss that is suited best for our data requires further testing with higher resolution meshes, however the importance of the depth loss for our scenario is clearly shown when comparing to the results without it. Future work might follow up with more fine-tuning for DSNeRF and a fully adapted mesh-extraction pipeline, alongside potential results from more recent depth-aware work.

As for MVS + Poisson, we show promising results mainly led by the high graphical fidelity for visualisation, which are held back by erroneous depth maps used for the dense point cloud reconstruction by MVS. Further work in adapting externally extracted depth maps from works such as Depth Anything [37] into the COLMAP MVS + Poisson (or any other MVS + mesh reconstruction combination) could yield great results without the need for training models for scene representation.

In conclusion, our work demonstrates that Structure from Motion is an approach that is worthwhile to pursue further for 3D reconstruction of aircraft engine inspection data, and simultaneously giving an overview and performance analysis of some of the available methods for detailed 3D visualization.

## References


[1] Camera models. https://colmap.github.io/cameras.html. 3

[2] Ayush Baid, John Lambert, Travis Driver, Akshay Krishnan, Hayk Stepanyan, and Frank Dellaert. Distributed global structure-from-motion with a deep front-end, 2023. 2

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 2

[4] Wu Changchang. Visualsfm: A visual structure from motion system. 2

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields, 2022. 2

[6] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 4

[7] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. pages 864–872, 12 2015. 2

[8] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. pages 12872–12881, 06 2022. 2, 3, 7

[9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. pages 337–33712, 06 2018. 2

[10] Mihai Dusmanu, Philipp Lindenberger, John Lambert, and Paul-Edouard Sarlin. Python bindings for colmap, 2014. 3

[11] Nafie El Coudi El Amrani, Yancong Lin, and Jan C. van Gemert. Bladenerf: Exploiting camera constraints for nerf in repetitive texture-less 3d reconstruction, 06 2023. http://resolver.tudelft.nl/uuid:0d76e5f8-929c-4b21-8aa0-03be39250ccf. 1

[12] Michael Goesele, Brian Curless, and Steven Seitz. Multi-view stereo revisited. volume 2, pages 2402–2409, 01 2006. 2

[13] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU, 2006. Eurographics Association. 2

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. 2, 3

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. 7

[16] Devin Lieuw A Soe, Jan C. van Gemert, and Burak Yildiz. Ground truth for evaluating 3d reconstruction of jet engines, 2021. http://resolver.tudelft.nl/uuid:b0d65392-49d6-4d75-9aad-417635938778. 1

[17] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields, 04 2021. 3

[18] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed, 06 2023. 2, 3

[19] David Lowe. Object recognition from local scale-invariant features. *Proceedings of the IEEE International Conference on Computer Vision*, 2, 01 2001. 1

[20] Thomas Markhorst, Jan C. van Gemert, and Burak Yildiz. Performance analysis of simultaneous localization and mapping to reconstruct aircraft engines in 3d, 2021. http://resolver.tudelft.nl/uuid:91a2f96a-eea7-4dbf-bfec-ff3f39999aa2. 1, 2

[21] Ben Mildenhall, Pratul Srinivasan, Matthew Tancik, Jonathan Barron, Ravi Ramamoorthi, and Ren Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*, pages 405–421. 11 2020. 2

[22] Chen Quei-An. Nerfpl: a pytorch-lightning implementation of nerf, 2020. 3, 4, 7

[23] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 2, 3

[24] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2

[25] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 3

[26] Johannes Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. 06 2016. 1, 2, 3

[27] Rajvi Shah, Aditya Deshpande, and P Narayanan. Multistage sfm: A coarse-to-fine approach for 3d reconstruction. 12 2015. 1

[28] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, jul 2006. 2

[29] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9:137–54, 11 1992. 2

[30] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient, 06 2020. 2, 3

[31] Shimon Ullman. The interpretation of structure from motion. 1979. 2

[32] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction, 06 2021. 3

[33] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004. 7

[34] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Prisacariu. Nerf−−: Neural radiance fields without known camera parameters, 02 2021. 3

[35] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo, 09 2021. 3

[36] Yitong Xia, Hao Tang, Radu Timofte, and Luc Gool. Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction, 10 2022. 3

[37] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 9, 10

[38] Zhongda Yuan and Mingguang Liu. Specification for engine borescope inspection report. *IOP Conference Series: Earth and Environmental Science*, 186:012001, 10 2018. 1

[39] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 7

# Part II

## Background information

# 1

# Background

**Incremental SfM Pipeline.** Incremental SfM (Fig. 1.1) starts with the feature extraction and matching process. These distinctive visual features (commonly referred to as keypoints) are crucial for the follow-up steps in the SfM pipeline. There is no one method that is best for visual feature extraction and matching, however SIFT [1] is commonly used as a robust and simple method. Even so, in textureless scenarios like our own data, more complex methods might prove more helpful. Some of these methods include feature extractors such as DISK [2], SuperPoint [3] and feature matchers such as SuperGlue [4].

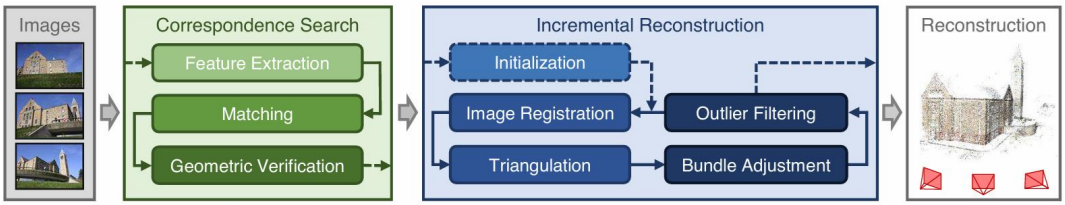


**Figure 1.1:** *An overall view of incremental structure from motion [5]. After feature extraction and matching, geometric verification assists with quality control of matches. The reconstruction is then initialized with two images, with new images being registered past that. Triangulation, bundle adjustment and outlier filtering follow to obtain the best estimates and reduce reproduction error. The process repeats until all images are registered.*

Assume we have a 3D point P and two 2D points:

$$x_1, x_2$$

which are the projections of X in image 1 and image 2, respectively, as seen in Fig. 1.2. The plane in which the 3 points (X, x1, x2) lie, is called the epipolar plane. The relationship between these points can be expressed using a *the Fundamental Matrix* **F**, which can be calculated through:

$$x_1^T F x_2 = 0$$

. A more simplified version derived directly from this formula can then be answered by solving the linear least squares using Singular Value Decomposition (SVD), in the end obtaining an estimate of **F**.

However, all feature extractors come with some noise and outliers regardless of their robustness, especially in our data conditions. In efforts to obtain a better estimate of **F**, some version of the RANSAC [7] algorithm is used select the matrix with the maximum number of inliers.

The **F** estimate can then be used to find the relative camera poses between Image 1 and Image 2. To do that, another matrix called *the Essential Matrix* **E** needs to computed, where

$$E = K^T F K$$

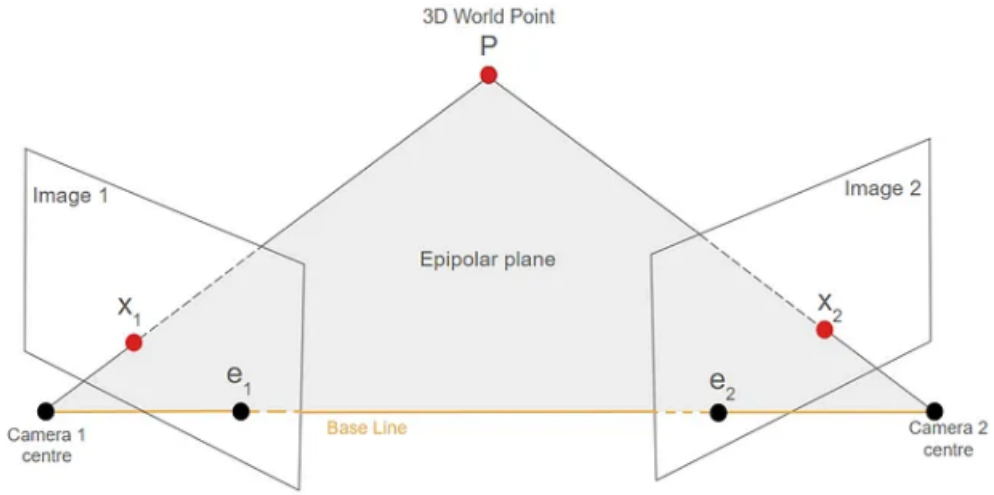


**Figure 1.2:** *A general view of epipolar geometry [6].*

**K** being the camera calibration matrix, which contains information such as the optical centre of the camera and focal length. This information is needed, as **E** assumes that the cameras obey the pinhole camera model and that can be achieved with help from **K**.

**E** can be further decomposed into a matrix **T** corresponding to the translation, and a matrix **R** corresponding to the rotation between views. This is possible once again through SVD. To describe the matrices, we take one of the cameras as a reference point to get projection matrices P1 and P2:

$$P_1 = K_1[I|0]$$

$$P_2 = K_2[R|T]$$

These projection matrices can be used to compute the estimated 3D point coordinates from the image positions of these points in our given views. This process is referred to as triangulation. It is commonly associated with an error minimization step, as the 3D points obtained are usually not fully correct due to ever-present measurement noise from previous steps.

Now that the camera poses and 3D point positions have been computed, we can refine both the poses and points together through a process called **bundle adjustment**. This final stage tries to minimize a cost function related to a weighted sum of squared reprojection errors. Bundle adjustment is often repeated multiple times throughout a SfM process, both locally (taking into account a certain number of views) and globally (taking into account all currently registered views).

For a multi-view application (more than 2 images), the process above repeats for each new image added, however with a new pair. So for figure 1.3, if we started with image **k-1** and **k**, we would next take image **k** and **k+1** together to register image **k+1**.

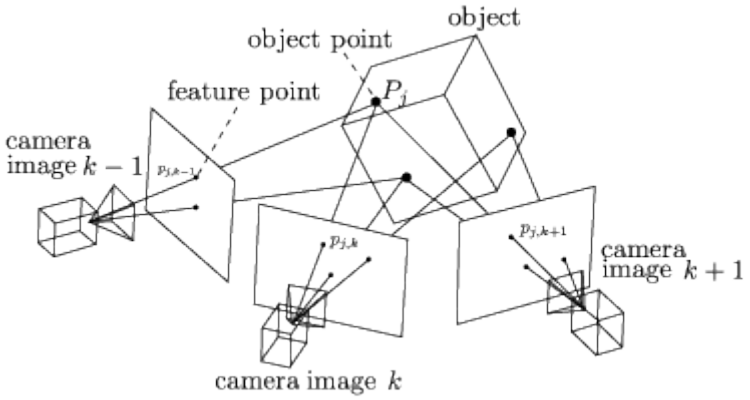


**Figure 1.3:** *Multi-view setup from [8]. All cameras are looking at the same object, however not all can see the same points due to their position.*

# References


[1] David Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *Proceedings of the IEEE International Conference on Computer Vision* 2 (Jan. 2001).

[2] Michał Tyszkiewicz et al. *DISK: Learning local features with policy gradient*. June 2020.

[3] Daniel DeTone et al. “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: June 2018, pp. 337–33712. DOI: 10.1109/CVPRW.2018.00060.

[4] Paul-Edouard Sarlin et al. “SuperGlue: Learning Feature Matching with Graph Neural Networks”. In: *CVPR*. 2020.

[5] Johannes Schönberger et al. “Structure-from-Motion Revisited”. In: June 2016. DOI: 10.1109/CVPR.2016.445.

[6] Teresa Lobo. *Understanding Structure From Motion Algorithms*. https://medium.com/@loboateresa/understanding-structure-from-motion-algorithms-fc034875fd0c. 2023.

[7] Martin A. Fischler et al. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. DOI: 10.1145/358669.358692. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/358669.358692.

[8] Pierre Moulon et al. “Adaptive Structure from Motion with a Contrario Model Estimation”. In: Oct. 2012. DOI: 10.1007/978-3-642-37447-0_20.