Ground Truth for Evaluating 3D Reconstruction of Jet Engines

Devin Lieuw A Soe[1] , **Jan van Gemert**[1] , **Burak Yildiz**[1]

[1]TU Delft

Abstract

With 3D reconstruction from borescope videos of jet engines, inspections can be made more efficient. Being able to reconstruct 3D models of the jet engines potentially contributes to making the inspections autonomous. Methods that will be used for this reconstruction require to be evaluated to ensure the accuracy of the 3D models that they create. Therefore, this study investigates how a 3D reconstruction method can be quantitatively evaluated using ground truth. From the results can be concluded that 3D models that represent ground truth data can be generated using a combination of manual and algorithmic feature matching between the frames of borescope videos. Furthermore, the Wasserstein distance is found to potentially be a viable measure for quantifying the comparison between 3D models, which is needed for assessment.

1 Introduction

An efficient way to make measurements during an industrial inspection is to create a 3D model of the object in question. This model can be reconstructed using a video of the object. Within the frames of this video, characteristic points referred to as features are located and subsequently a 3D model is constructed by matching the features of different frames. In the ideal case, the output 3D models of a 3D reconstruction method would be compared with a ground truth set when evaluating the method. This allows for quantitative assessment of the reconstruction. Such ground truth data would consist of input-output pairs where the videos represent the input and a set of corresponding 3D models the output. The intention of this research is to investigate how this ground truth data can be created and utilized for quantitative evaluation of 3D reconstruction methods.

Aiir Innovations[1] is a startup company that creates software that automates borescope inspections of jet engines. They aim to make borescope inspections faster and easier to perform without losing inspection quality, and their vision is to make the inspections a fully autonomous process. 3D reconstruction of the jet engines makes it possible to efficiently perform borescope inspections on them.

At the moment, only qualitative evaluations can be done by for example manually looking at the output 3D models and verifying them. However, assessing a reconstruction method with a calculated score will be a more objective way of qualification, since this allows for quantification of the accuracy of the method. Therefore, the main research question is: *How to quantitatively evaluate 3D reconstruction of jet engines with ground truth?* This problem can be divided into two subproblems.

To assess a 3D reconstruction method with a certain accuracy, a reference set that serves as ground truth data is needed. Thus, the question "*How should data that serves as ground truth data be created?*" needs to be answered. For this, different methods of matching features should be considered, one of which being manually annotating the features.

When a sufficient ground truth set is generated, this data will be used to evaluate 3D reconstruction methods. To be able to perform such an evaluation, there should be an efficient measurement method to quantify the errors of the 3D models reconstructed by the system under test. Therefore, answering the question "*How should the difference between two 3D models be measured?*" is needed.

Quantitative assessment of a 3D reconstruction method can be divided into a number of steps, as can be seen in Figure 1.

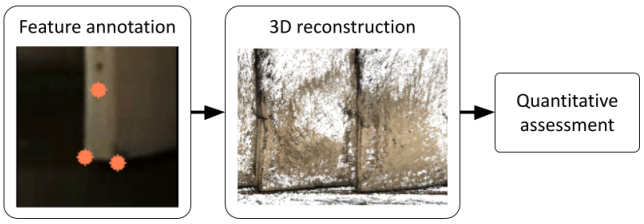


Figure 1: The pipeline for quantitative assessment.

There are three main contributions:

1. Providing a better understanding of the efficiency of manually annotating features.
2. Describing a method on how ground truth data can be created for the evaluation of 3D reconstruction.

[1]Aiir Innovations: https://aiir.nl/

3. Describing a method of quantitatively comparing 3D point clouds that represent 3D models.

The rest of this paper is distributed over 7 Sections. The next section provides a practical problem description. Section 3 discusses related work. After that, Section 4 will give a theoretical explanation of the techniques that will be experimented with and motivates them. The actual experiments are described in Section 5 which also demonstrates the results. The ethical aspects and the reproducibility regarding the outcome of the results are highlighted in Section 6. Section 7 discusses the results from the experiments and covers the limitations of the experiments. The conclusion of this paper is provided in Section 8.

2 Quantitative Evaluation Problem

The aim of this research is to be able to quantitatively evaluate a 3D reconstruction method using a reference set consisting of input-output pairs that serve as ground truth data. This section provides an in-depth description of what needs to be done to achieve this goal.

2.1 Ground truth

Ground truth data is data that is more accurate than the results of the algorithms that will be tested [1]. In this case, the idea is to create 3D models that are more similar to the real objects in terms of dimensions than the results of the 3D reconstruction methods under evaluation. By quantifying the distance between the output 3D models of the algorithm in question and their corresponding ground truth, the accuracy of the 3D reconstruction method can be calculated.

2.2 Feature detection problem

For feature-based 3D reconstruction, features need to be detected within the frames of the input video. Features, also referred to as interest points or keypoints, are locations within an image that stand out. These features can be corners for instance, but also locations where there is a lot of variation in texture [2]. An example of a feature detection algorithm is Scale-Invariant Feature Transform (SIFT), which produces scale-invariant descriptors of local features within an image. It does so by performing keypoint localization and measuring the local image gradients at the location of the keypoints to represent them [3]. Oriented FAST and Rotated BRIEF (ORB) is another example of an algorithm used to detect features. ORB uses FAST for feature detection and BRIEF for the feature descriptors, which makes the algorithm invariant to rotation and relatively fast and insensitive to noise [4].

From the detected features, a 3D model of the object within the video can be constructed by performing feature matching between frames of the video. Structure from Motion (SfM) is an example of such a 3D reconstruction algorithm. With a set of 2D images of an object from different angles, SfM is able to reconstruct a 3D model of the object by calculating the relative 3D positions of the points within the scene. Within this project, an elaborate study on how well SfM works on borescope videos of jet engines is performed by Nonnemaker [5]. Another example of a 3D reconstruction algorithm is Simultaneous Localization and Mapping (SLAM). The idea of SLAM is that it reconstructs a 3D model of the surroundings of the camera in real-time [6]. A variant of SLAM is ORB SLAM where ORB is used to detect the features. While SLAM creates 3D reconstructions from an ordered list of images, SfM does it with an unordered list [7].

Even though there are several 3D reconstruction methods available already, many of them rely on enough texture and features that are easy to find within input images. If videos contain surfaces that are shiny and lack texture, it can be difficult for feature detection algorithms such as SIFT and ORB to locate appropriate keypoints within the frames of the video. In this project, the majority of the input videos that are used are borescope videos of jet engines[2] provided by Aiir. These videos often contain such textureless surfaces. The shiny surfaces reflect light which results in unwanted intensity gradients in the images.

SfM will be utilized for 3D reconstruction in the experiments of this study. To detect and match features when performing SfM, SIFT is most commonly used. Since it is likely that SIFT does not work sufficiently on the borescope videos of the jet engines [8], a more efficient way of localizing and matching features within the frames is needed. This is also a necessity for ensuring that the ground truth 3D models are more precise than the produced models of the reconstruction methods under evaluation. Therefore, SIFT needs to be replaced with better localized and matched set of features.

3 Related Work

This section discusses and criticizes studies that have been done in the area of feature localization and quantitative comparison between 3D point clouds. It also explains how these approaches are related to and differ from this research.

Algorithmic feature detection

For the localization of features, several algorithmic approaches can be used. Examples of algorithms that detect features on a 3D model are HKS [9], 3D Harris [10], Mesh Saliency [11], Salient Points [12], Scale Dependent Corners [13], CGF [14] and SHOT [15]. SuperGlue is a pretrained graph neural network that detects features within images [16]. It also includes a trained optical matching layer that performs feature matching between two images. SuperGlue can be run on a pair of frames of a video to localize and match features between both frames. The neural network has a high performance on borescope videos as is shown by a study performed by Huizer [8]. Feature detection algorithms can contribute to generating the ground truth data, because of the fact that local extrema within the frames can be detected more adequately compared to manual feature annotation.

Manual feature annotation

Another option for localizing features is manual annotation [17]. To obtain the features, humans mark them within the images or on the 3D model in question. Humans have a semantic understanding of the object that they are seeing, but it is more difficult for them to localize local extrema on it. Also,

[2] An example of such a borescope video can be found at https://www.rvi-ltd.com/borescope-inspections-aircraft-engines/

different people will most likely mark different features when annotating a model.

To evaluate the performance of a 3D feature detection algorithm, human-generated features can be used as ground truth data. This idea is presented in [18], where a web page with a user interface is used for collecting the human-generated ground truth. Within the user interface, the user is able to see the 3D models of the set that is used in the experiments, rotate them and annotate the features on them. As a solution to the problem of the variation in feature annotations among users, the annotations of the different participants are merged based on their similarities. Points that are close enough to each other are grouped and outliers are discarded. The resulting features are considered ground truth data and are used to evaluate feature detection algorithms.

However, ground truth data needs to be qualitatively better than the data that is tested and it might be the case that human-generated features are less precise than features that are detected by algorithms. Since humans have more difficulty detecting local extrema within 3D models, algorithms are generally better at localizing features on surfaces or edges. Because of this, evaluating feature detection algorithms with only human-generated ground truth might not be sufficient. In such cases, using a combination of manual and algorithmic feature detection can be more efficient. This idea will be experimented with in this research.

Distance between 3D point clouds

Comparing two 3D point clouds with each other can be difficult. This is partly because of the fact that both point clouds often consist of a different number of points and the order of the points that are included in the clouds can also differ. There are different methods of calculating the difference between 3D point clouds. Examples are the Wasserstein distance [19], the Chamfer distance [20], the Earth Mover's distance [21] and the Hausdorff distance [22].

Being able to quantify the difference between two point clouds that represent 3D models is essential for evaluating a feature-based 3D reconstruction method with ground truth. The Wasserstein distance is an interesting measure for this purpose, because of the fact that differences in order and amount of points in the point clouds have little effect on the distance. For computing the Wasserstein distance, weights are assigned to each of the points. Consequently, the cost is computed of transporting the weight from all points of one point cloud to the points of the other point cloud.

4 Creating Ground Truth Data

Features in the frames of an input video need to be localized and matched to create a ground truth 3D model from the object within the video. This section motivates and describes a number of techniques that are experimented with in this research and that can potentially be used to perform this task. Furthermore, the 3D reconstruction and evaluation of the ground truth models is described.

4.1 Feature annotation

One way of localizing and matching the features within the frames of an input video is by exhaustively annotating them by hand for every frame. When manually annotating the features, it can be ensured that there are no wrong feature matches and that segments of the frames that will not be used for the 3D reconstruction are not annotated. This implies that there is good control over the quality of the final 3D model. While manual annotation provides precise control over which and how many features are localized and matched, there some disadvantages. Firstly, annotating features in every frame of a video often takes relatively long. Also, manual annotation most likely results in a certain amount of noise in the feature localizations. Lastly, it might occur that few features are annotated when doing it by hand. To account for these problems, several algorithmic approaches have been evaluated with experiments. This was done with the intention to not only try to speed up the process, but also improve the accuracy and the amount of the ground truth feature annotations. The main ideas of these algorithmic approaches are described in the following subsections.

4.2 Optical flow

Optical flow is used to make estimations of motion between sequences of images. This sequence can for instance consist of consecutive frames within a video. In this case, the apparent motion in the image plane between for example the first and second frame is determined [23], see Figure 2.

(a) first frame (b) second frame (c) optical flow

Figure 2: A Rubik's cube on a rotating turntable. The motion is represented by the arrows in Figure c.

The Lucas-Kanade approach is an example of an optical flow algorithm [24] and is evaluated with an experiment in this research. An assumption that is made by this algorithm is that the movement of a certain pixel of an image is similar to the movement of the pixels in its neighborhood. The borescope videos of the jet engines contain blades that do not change in form, implying that the neighborhood pixels of a specific pixel will most likely move in a similar direction as the pixel itself. Therefore, the Lucas-Kanade algorithm may be useful for a faster process of feature annotation within the borescope videos.

If the features of a certain frame are annotated, the features of the next frame can be derived by using the Lucas-Kanade algorithm. Each newly annotated feature within the second frame can subsequently be matched to its corresponding feature from the first frame. This would reduce the amount of manual labour needed for feature annotation. The resulting features of the experiment that is done with Lucas-Kanade optical flow are qualitatively evaluated, since they should be at least as accurate as manually annotated features of which

the error is negligible. Verifying the resulting features by comparing them with manual annotation is therefore sufficient.

4.3 Interpolation

Another method that could be used with the intention of saving time in the process of manual feature annotation is interpolating the feature coordinates over frames. If for example frame 3 has a feature on pixel (1, 6), where its x-coordinate is 1 and its y-coordinate is 6, and frame 7 has the same feature located on pixel (5, 4), frames 4, 5 and 6 can be assigned this feature with coordinates (2, 5.5), (3, 5) and (4, 4.5) respectively when interpolating linearly, see Figure 3.

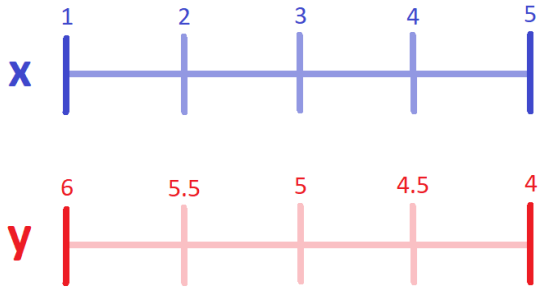


Figure 3: Interpolation between x-coordinates 1 and 5, and between y-coordinates 6 and 4. The intermediate coordinates can be derived.

By applying this method, the feature locations of intermediate frames, frames that are not annotated, are derived by interpolation of the existing feature coordinates of the annotated frames. Therefore, not all frames need to be manually annotated which is beneficial for the amount of time that is needed to produce the annotations of all frames. It might also improve the quality of the annotations in terms of accuracy, since the motion within a borescope video is relatively consistent and there is little change in the direction of the movement of each keypoint over time. Consequently, if the features from each single frame are manually localized, it is likely that the annotations lack consistency and contain noise. Using interpolation on the other hand, the smooth movement of the blades within the videos can be approached better. Just like the Lucas-Kanade experiment, the interpolation experiment are also evaluated by looking at the resulting features.

4.4 Incremental SfM and MVS

There are different ways of performing SfM, one of them being the incremental SfM technique. Incremental SfM initializes the reconstruction of a model with two images after which it registers new images in an incremental fashion. This is done until the full sparse reconstruction of the scene is achieved [5]. The incremental SfM strategy is used for the creation of the sparse point clouds of the ground truth data within this project, since it processes all image pairs that have matching features. Besides, the reconstruction does not need to be done in real-time for the purpose of this study.

Multi-View Stereo (MVS) [25] is a technique that can be used in combination with SfM to construct a dense 3D model out of the obtained sparse point cloud and the image information. MVS will also be used for the experiments of this research, since creating a dense point cloud allows for more efficient qualitative evaluation compared to just using the sparse point clouds obtained from incremental SfM.

4.5 Qualitative Evaluation

Qualitative evaluation is necessary to verify that the reconstructed ground truth 3D models are sufficiently accurate and do not contain too much noise. In practice, a single ground truth 3D model will most likely be utilized within the evaluations of multiple 3D reconstruction algorithms. This ground truth model needs to have a higher quality than the 3D reconstructions that result from all algorithms that are being evaluated. Verification of the ground truth is therefore required.

5 Experiments and Results

Within this project, a number of experiments have been performed with manual feature annotation, optical flow, interpolation, the SuperGlue neural network and the Wasserstein distance. This section describes the setup of these experiments as well as their results.

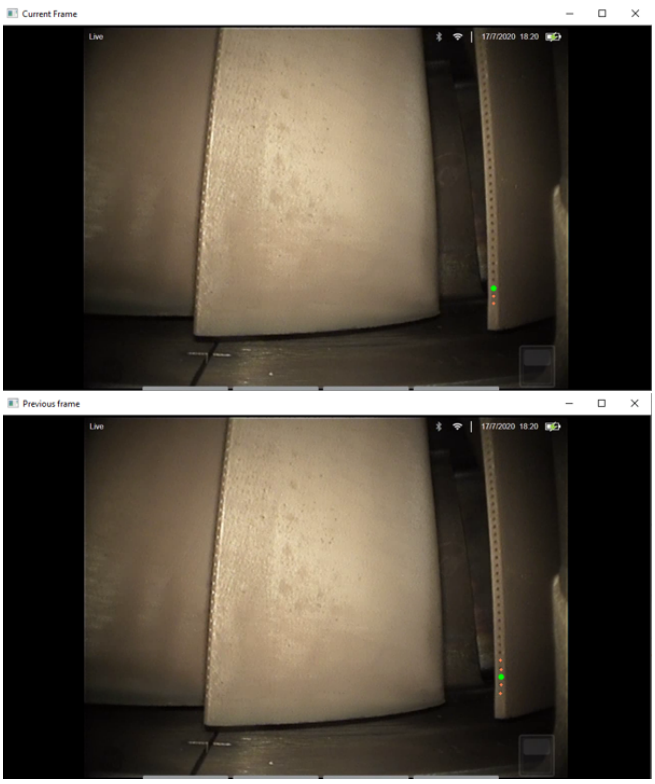
5.1 Feature matching tool

During the evaluation of the different methods of feature annotation, a clear environment was needed that visualizes annotated features and enables for addition and deletion of annotations. A feature matching tool was implemented in Python for this purpose. The *opencv-python* and *numpy* libraries were used for the implementation. In Figure 4 is shown what the user interface looks like when running the tool on a borescope video from Aiir of a jet engine that is labeled as Video 1.

The tool is an efficient way to not only manually annotate the features and matches, but also to combine that with algorithmic methods. When running the tool, the user gets the opportunity to fill in the name of the video file, as well as the name of the output csv-file. Following that, the user is asked to indicate the interval of the frames that will be annotated. If then for example the number 5 is filled in, the tool will only process frames 1, 6, 11, etc. This makes interpolation possible. Lastly, the user can fill in the last frame that needs to be processed. Afterwards, the frames of the video will be loaded and the program will jump to the first frame. The number of frames is displayed in the console, as well as an overview of all commands that can be used to run certain tasks, see Figure 4a. To be able to annotate feature matches within the frames, the current frame and previous frame are displayed. The annotations can only be made in the window of the current frame. Left-clicking creates an annotation at the mouse position and right-clicking deletes the annotation. Features are assigned an id and the user can iterate over them. If two frames both contain a feature with the same id, there is a feature match between the two frames. Already annotated features are colored orange and the feature that is being processed currently is colored green if it is already marked, as can be seen in Figure 4b.

```
Video name: 7.mkv
Output csv-file name: tool/7.csv

Annotate once every ... frames: 1
End frame: 300

Loading the video...
Number of frames: 300

    Press B to go to the previous frame
    Press N to go to the next frame
    Press F to jump to a frame of choice
    Press < to go to the previous feature
    Press > to go to the next feature
    Press G to jump to a feature of choice
    Press I to interpolate the feature annotations for intermediate frames of choice
    Press K to copy the features of choice of the previous frame to the current frame
    Press L to apply the Lucas-Kanade method using the previous frame
    Press T rerun the tool with different parameters
    LEFT CLICK to annotate a feature
    RIGHT CLICK to undo a feature annotation
    Press LSHIFT+W to create a setup for SfM
    Press LSHIFT+P to run SuperGlue (writes to csv)
    Press LSHIFT+C to clear the current frame of features of choice
    Press LSHIFT+R to clear frames of choice of features of choice
    Press LSHIFT+S to save (writes to csv)
    Press LSHIFT+Q to quit and save (writes to csv)
    Type 'x' cancel the current task

Current frame: 1, Current feature: 0
```

(a) The console after the input video has loaded. The number of frames is visible as well as a list of all commands, the current frame and the id of the current feature.



(b) The current frame (upper image) and the previous frame (lower image) are visualized when running the tool. The current feature is colored green and the already annotated features are colored orange.

Figure 4: The user interface of the feature matching tool run on the first 300 frames of Video 1.

SfM and MVS with COLMAP

For the 3D reconstruction, COLMAP[3] is used to run incremental SfM and MVS. COLMAP makes it possible to match features within a video by creating custom descriptor matrices for them and formatting a text file in which all overlapping frame pairs and their corresponding feature matches are listed[4]. In the experiments done in this research, a frame pair that contains at least 10 matches is considered an overlapping frame pair. The feature matching tool offers functionality to generate both the feature descriptor matrices and the text file with the overlapping frame pairs. Using this data, the sparse 3D point cloud is constructed with incremental SfM and the dense point cloud with the built-in MVS functionality. These models can be saved as *.ply* files.

Experiment 1: The same features for all frames

To validate whether inputting custom feature matches in COLMAP works sufficiently, a video of a laptop is used. This video is labeled as the Laptop Video. On each of the frames, the same 16 features are manually annotated and given ids 0 to 15 using the feature matching tool, see Figure 5a. Following that, the descriptor matrices for each of the features are generated and the text file with the overlapping frame pairs is created using the LSHIFT+W command within the tool. The matrices and the text file are used as input in COLMAP and then the 3D point cloud is incrementally constructed with SfM. The *init_min_num_inliers* and *abs_pose_min_num_inliers* parameters within COLMAP are set to 16, since only 16 features are inputted per frame. The default values are maintained for the rest of the parameters. This experiment will demonstrate to what extent the resulting 3D model cohers to the custom feature annotations. In Figure 5b, the resulting 3D point cloud is visualized. The shape of the laptop is accurately reconstructed.

5.2 Lucas-Kanade optical flow

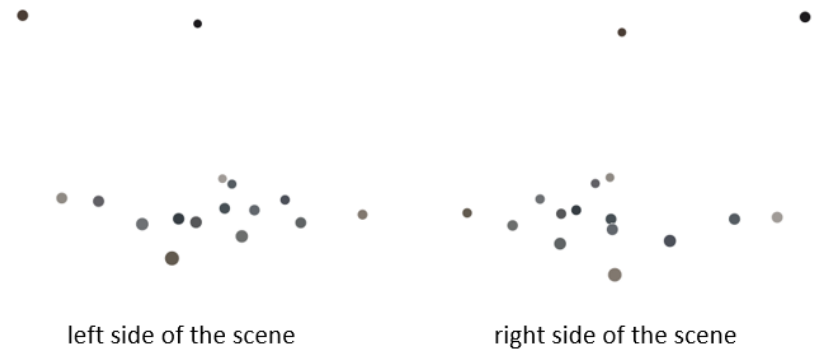
The feature matching tool provides functionality to perform a step of the Lucas-Kanade optical flow algorithm. This step can be executed to derive the features in the current frame from the annotated features in the previous frame. The implementation for this experiment is inspired by the code from OpenCV[5], where features within the frames of a video are detected and tracked throughout the whole video. This is done using the Lucas-Kanade method, which essentially provides the corresponding points in the next frame, and this step is repeated for all frames. For the purpose of the experiment with optical flow in this project, however, the Lucas-Kanade step

[3]https://github.com/colmap/colmap/releases/tag/3.6

[4]urlhttps://colmap.github.io/tutorial.html

[5]https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html/

(a) Two different frames of the Laptop Video annotated with the same 16 frames. For Experiment 1, all frames are annotated with these features.



(b) The resulting 3D model of inputting the 16 feature annotations for every frame in COLMAP. The model is shot from two different viewpoints. The shape of the laptop is clearly observable.

Figure 5: 3D reconstruction of the laptop from the Laptop Video with 16 different features. (**Experiment 1**)

is only performed once when executing it. By using the coordinates of the annotated features in the previous frame as the points that need to be tracked, the corresponding features in the current frame are calculated with the Lucas-Kanade step.

Experiment 2: Lucas-Kanade for feature annotation

For evaluating the effectiveness of this method of using Lucas-Kanade optical flow, an experiment is done with three frames of Video 1. The first frame is manually annotated with features on the clearly distinguishable points. Following that, the Lucas-Kanade step is executed twice to derive the locations of the corresponding features within the frames that follow the first frame. The results of this experiment are visible in Figure 6. It can be observed that there is a noticeable inaccuracy in the localizations of the features.

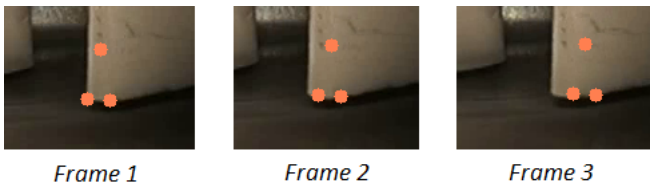


Figure 6: Lucas-Kanade on three consecutive frames of Video 1. The features are not tracked properly. Better results can be obtained with manual annotation. (**Experiment 2**)

5.3 Interpolation

Interpolation of feature annotations is also a functionality that is implemented in the feature matching tool. When running the tool, the user will be asked for an integer input when the line "Annotate once every ... frames: " appears. If for example 10 is filled in, the tool will only iterate over frames 1, 11, 21, etc. The last frame that will be processed depends on the end frame that is filled in by the user afterwards.

When the features within the frames that the tool will iterate over are marked, the user can execute the interpolation by pressing I on the keyboard. This linearly interpolates the features for the intermediate frames, so frames 2-10, 12-20, etc. for the case described in the previous paragraph. The user has the option to indicate the start and end frames of this interpolation, as well as which features are interpolated.

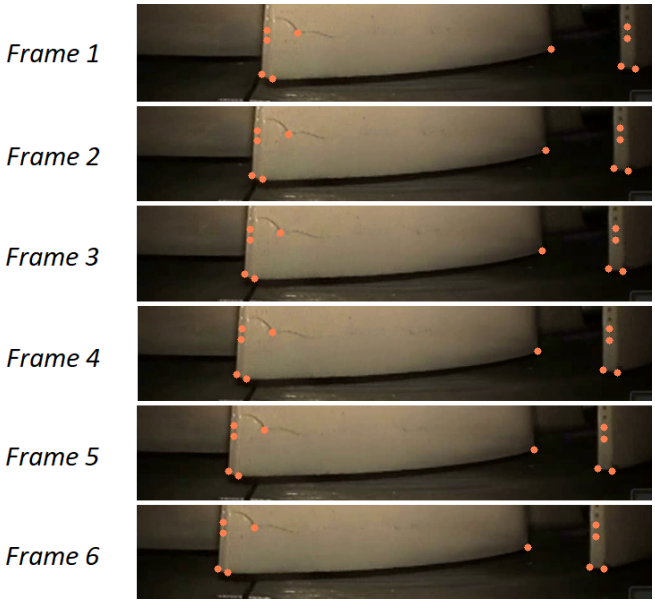


Figure 7: Interpolation of features in Video 1 with $n = 5$. There is no noticeable error in the feature localizations. (**Experiment 3**)

Experiment 3: Interpolating feature annotations
Within this experiment, the quality of using interpolation will be evaluated. Let n be the interpolation interval that is filled in by the user when the line "Annotate once every ... frames: " appeared in the console. In Figure 7, interpolation was done on 10 different features with $n = 5$ to get a sequence of 6 frames that have 10 annotations. The feature annotations are done manually on frames 1 and 6 and the localizations in the intermediate frames are derived by linear interpolation. This resulted in sufficiently accurate feature localizations within the sequence of frames.

If n is too high, the derived feature coordinates in the intermediate frames will not be as accurate as manually annotated features. If for example $n = 30$, the calculated locations of the features in frame 16 have a noticeable error, see Figure 8.

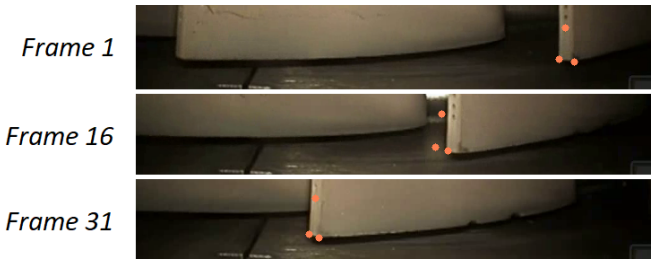


Figure 8: Interpolation of features in Video 1 with $n = 30$. The features in frame 16 are not sufficiently accurate. Better results can be obtained with manual annotation. (**Experiment 3**)

5.4 Constructing ground truth 3D models

For creating the 3D models that will be used as ground truth, a relatively exhaustive approach is used. To produce the feature localizations and matches, the SuperGlue neural network[6] is used in combination with interpolation and algorithmic filtering. As for the parameters, the default values are used and the images are resized to 1280 x 720 pixels.

Experiment 4: Combining SuperGlue with interpolation
This experiment evaluates an approach of creating ground truth 3D models where the features are localized and matched by a combination of SuperGlue and interpolation. In the first iteration SuperGlue is run on all consecutive frame pairs, so on frames 1 and 2, frames 2 and 3, frames 3 and 4, etc. This produces a number of features and matches among the frames. In the next iteration, SuperGlue will process frames 1 and 3, frames 3 and 5, frames 5 and 7, etc. The coordinates of the features of the intermediate frames 2, 4, 6, etc. are derived using linear interpolation in a similar fashion to Experiment 3. Within the same iteration, SuperGlue is run on frames 2 and 4, frames 4 and 6, frames 6 and 8, etc. and the features in the intermediate frames are again derived with interpolation. The second iteration therefore consists of two steps, as opposed to the first iteration which has only one step. The third iteration will in turn consist of three steps where SuperGlue is run on frames 1 and 4, frames 4 and 7, etc. in the first step, frames 2 and 5, frames 5 and 8, etc. in the second step, and frames 3 and 6, frames 6 and 9, etc. in the third step. Again, the intermediate frames are interpolated over. For this experiment, this is repeated until the 5th iteration and the detected and interpolated features from all steps of all iterations are combined into a large set of feature localizations and matches.

To reduce the amount of noise within the point cloud, algorithmic filtering is applied. The intention of this is to filter out points that are not of interest, such as the features that are detected in the background. For Video 1 for example, algorithmic filtering is done such that the movement to the left of all features is between 6 and 30 pixels and the absolute value of the movement along the vertical axis is at most 7 pixels. The resulting set of feature matches is used as input within COLMAP for 3D reconstruction. Apart from the *camera model* being set to *radial*, the default parameters are used in COLMAP. The outcome 3D model of this experiment on Video 1 is visible in Figure 9. This experiment is also done on several other borescope videos of jet engines provided by Aiir, which resulted in similar comparisons between the ground truth and SuperGlue models.

5.5 Wasserstein distance

Calculating the Wasserstein distance is an example of a method of quantifying the difference between two 3D point clouds that represent 3D models. For Experiment 5, the python library *point-cloud-utils*[7] is utilized to perform the comparison.

Experiment 5: Comparing small 3D point clouds
In this experiment, 16 features within the Laptop Video are localized in three different ways. Following that, the three corresponding sparse 3D point clouds are constructed with incremental SfM in COLMAP and they are labeled as Point cloud 1 to 3. These are relatively small point clouds, since computing the Wasserstein distance is computationally expensive and with the available technology, 3D point clouds reconstructed from algorithmic feature detection were too large. The same parameters as those from Experiment 1 are utilized in COLMAP.

- Point cloud 1: The point cloud from Experiment 1.
- Point cloud 2: Reversed order of the ids of the annotated features from Experiment 1. This modification lists the feature matches in reversed order in the text file with overlapping frames, which results in a point cloud where the order of points is reversed.
- Point cloud 3: A point cloud that contains 16 randomly localized 3D points resulting from random annotations within the frames.

Using the *point-cloud-utils* library, the Wasserstein distance is computed between each of the point clouds, which are saved as *.ply* files. The expectation is that Point clouds 1 and 2 have a relatively low Wasserstein distance and that the order of the points within the point cloud does not affect the distance. Furthermore, the distances between Point clouds 1 and 3 as well as between 2 and 3 are expected to be relatively high,

[6] urlhttps://github.com/magicleap/SuperGluePretrainedNetwork

[7] url https://github.com/fwilliams/point-cloud-utils

(a) An example frame of Video 1. (b) Ground truth SfM + MVS reconstruction. (c) SfM + MVS reconstruction with SuperGlue.

Figure 9: Dense SfM + MVS 3D reconstruction on 150 frames of Video 1. The 3D model that results from the ground truth features (b) contains more blades than the model constructed with SuperGlue (c). Also, the shape of the blades is better and there is less noise. (**Experiment 4**)

because of the fact that there is a high difference in the feature localizations. The results of this experiment are visible in Table 1.

Point cloud	Number of points	Compare with	Wasserstein distance
Point cloud 1	16	Point cloud 1	0.0000
		Point cloud 2	11.301
		Point cloud 3	822.59
Point cloud 2	16	Point cloud 1	11.132
		Point cloud 2	0.0000
		Point cloud 3	862.52
Point cloud 3	16	Point cloud 1	912.88
		Point cloud 2	950.20
		Point cloud 3	0.0000

Table 1: Resulting Wasserstein distances of **Experiment 5**. As expected, the order of the points does not affect the distance, while a difference in feature localizations does. The relatively small Wasserstein distance that is computed between Point cloud 1 and 2 results from small variations in 3D reconstruction by incremental SfM.

6 Responsible Research

While this research provides an understanding in how ground truth can be used to quantitatively evaluate 3D reconstruction of jet engines, some responsibility is required when using the introduced methods. The experiments performed within this project show that certain techniques can be combined to produce ground truth 3D models. However, when it comes to the techniques used to localize and match features within input videos, certain algorithmic approaches work better on one video than on another. This implies that the person who is responsible for creating the ground truth data and therefore executing these algorithms should verify feature matches resulting from those algorithms and make manual adjustments where needed. Having inadequate ground truth data causes inaccurate assessments to be made every time it is utilized for evaluation. Thus, the ground truth 3D models should not be carelessly generated by assuming that the algorithmic approaches will always produce sufficient results. Another aspect concerns the quantification of 3D model comparison. While the experiment regarding the Wasserstein distance does show that this method can potentially be used for this purpose, there were some limitations as will be discussed in Section 7. More understanding on the performance of the Wasserstein distance needs to be obtained to verify that the measure is sufficient for the 3D model comparisons that are necessary for assessment. Ensuring the adequacy of the evaluation of jet engines is of major importance for safety when the engine is put into practice.

As for the reproducibility of the experiments done in this research, there are several aspects that need to be taken into account. The self-implemented feature matching tool including the linear interpolation functionality is not publicly available, but can be implemented in Python using publicly available libraries. The algorithms that are used for computing the Wasserstein distance and the Lucas-Kanade optical flow are also available online. Furthermore, COLMAP is publicly downloadable and the configurations of the parameters that are used are explained for each experiment in Section 5. The code for the SuperGlue neural network is also publicly available. The results of the experiments contain features that follow from manual annotation, which cannot be copied. However, they can be approached in a similar way to obtain similar results. Lastly, videos that are used for the experiments are not available, but examples can be found online and similar videos can be created. Considering all these aspects, the methods and results of this study are reproducible by any skilled reader.

7 Discussion

The results and limitations of the experiments in Section 5 are elaborated on in this section, starting with Experiment 1. In this experiment it was shown that the shape of the laptop is observable when inputting 16 features with ids 0 to 15 for every frame of the Laptop Video in COLMAP and running incremental SfM. This implies that accurate 3D reconstructions can be made with incremental SfM by inputting accurately localized features in COLMAP.

From Experiment 2 it can be observed that Lucas-Kanade optical flow does not work sufficiently for speeding up the

process of manual feature annotation. In Figure 6 it is visible that the features cannot be tracked properly throughout the frames. This happens because these points are manually chosen, whereas the OpenCV implementation decides on the points that are tracked using a built-in function *cv.goodFeaturesToTrack()*. Marking the features might lead to bad tracking compared to using the built-in function to decide on them.

Experiment 3 shows that linear interpolation of feature coordinates within annotated frames is an effective way of deriving the locations of the same features within the intermediate frames. It can be observed that there is no noticeable error in the derived features. This implies that the difference in accuracy between manually annotated and interpolated features is negligible. For this technique to provide sufficient results, however, it is important that the interpolation interval (n) is chosen accordingly. If the amount of variation in the speed and direction of the motion within a specific video is relatively high, then values for n that are relatively low produce more accurate feature annotations when interpolating.

Knowing that interpolation can be used to localize features, an evaluation has been done on an iterative approach of creating ground truth 3D models in Experiment 4. The ground truth 3D reconstruction of Video 1, visible in Figure 9b, has significantly more quality than the 3D reconstruction with only SuperGlue for the feature matching, visible in Figure 9c. However, there is still some noise visible in the ground truth reconstruction. This can be prevented by manually deleting incorrect feature detections within the frames. Another way to improve the ground truth model is to manually add accurate feature matches to the already existing matches, as can be deduced from Experiment 1. Due to the fact that manual labour takes relatively long, this is not done in Experiment 4. The experiment does show that a combination of algorithmic feature detection and linear interpolation can be used to create ground truth 3D models, given that the interpolation interval is not too high.

The results of Experiment 5 show that the Wasserstein distance can be used to quantify the difference between two 3D models. The comparisons within this experiment are done between point clouds with low numbers of points. This is because of the fact that calculating the Wasserstein distance has a high computational cost. To compare for example two 3D reconstructions of jet engines, more computational power is needed than available in this project. Also, there are no comparisons done between point clouds containing different amounts of points. However, the effect of inequalities in amount of points can be examined by performing these comparisons.

8 Conclusions

Jet engine inspections can be made more efficient by performing 3D reconstruction from borescope videos of the jet engines. To verify that a certain 3D reconstruction method is valid for this purpose, there needs to be a way of quantitatively evaluating it. Within this study, the main research question was: *How to quantitatively evaluate 3D reconstruction of jet engines with ground truth?*. Several approaches have been evaluated that could potentially be used to generate the ground truth 3D models. By manually annotating features within the frames of a borescope video, high-quality 3D models can be constructed. However, the amount of manual labour that is needed makes the approach inefficient. There are techniques that can be used to speed up the process of manual feature annotation. The experiments of this study show that while Lucas-Kanade optical flow does not perform sufficiently for this task, linear interpolation of feature coordinates does have potential. By deriving feature localizations from interpolation, fewer frames need to be annotated manually. From this research, it can be concluded that the features for the ground truth 3D models can partially be localized with a combination of algorithmic feature detection and interpolation. Since feature detection algorithms often do not perform sufficiently on the borescope videos, filtering out noisy feature matches and manually adding missing matches is also necessary. The 3D reconstruction out of the resulting set of feature localizations and matches can then be created. When the ground truth model is constructed, qualitative evaluation is essential for verifying that the model is similar enough to the real jet engine. If the ground truth model is more accurate than the results of the 3D reconstructions that will be evaluated, it can be utilized for assessment.

As for the quantification of the distance between two point clouds that represent 3D models, the experiment shows that the Wasserstein distance can be used for the comparison. What is still missing, however, is an understanding of the performance of this method when comparing point clouds of different sizes. Also, because of the computational expensiveness of computing the Wasserstein distance, the experiment only considers small point clouds. It might be interesting for future work to investigate how the Wasserstein distance can be used for the comparison of jet engine 3D reconstructions.

References

[1] J. R. Cardosa, L. M. Pereira, M. D. Iversen, and A. L. Ramos, "What is gold standard and what is ground truth?" *Dental Press J. Orthod.*, vol. 19, no. 5, 2014.

[2] I. Laptev, "How to quantitatively evaluate a 3d reconstruction method?" *International Journal of Computer Vision*, vol. 64, pp. 107–123, 2005.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[4] L. Yu, Z. Yu, and Y. Gong, "An improved orb algorithm of extracting and matching features," *International Journal of Signal Processing*, vol. 8, no. 5, pp. 117–126, 2015.

[5] A. M. Nonnemaker, "Evaluating structure-from-motion on shiny and non-textured surfaces in borescope videos," 2021.

[6] Z. Shang and Z. Shen, *Construction Research Congress 2018*, 2017, pp. 305–315.

[7] J. L. Schonberger and J. Frahm, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.

[8] R. M. Huizer, "Performance analysis of interest point detection/matching on shiny and non-textured surfaces," 2021.

[9] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Computer Graphics Forum*, vol. 28, pp. 1383–1392, 2009.

[10] I. Sipiran and B. Bustos, "Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes," *The Visual Computer*, vol. 27, no. 11, 2011.

[11] C. Ha Lee, A. Varshney, and D. W. Jacobs, "Mesh saliency," *ACM transactions on graphics (TOG)*, vol. 24, no. 3, pp. 659–666, 2005.

[12] U. Castellani, M. Cristani, S. Fantoni, and V. Murino, "Sparse points matching by combining 3d mesh saliency with statistical descriptors," *Computer Graphics Forum*, vol. 27, pp. 643–652, 2008.

[13] J. Novatnack and K. Nishino, "Scale-dependent 3d geometric features," *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[14] M. Khoury, Q. Zhou, and V. Koltun, "Learning compact geometric features," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 153–161, 2017.

[15] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," *European conference on computer vision*, pp. 356–369, 2010.

[16] P. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, *SuperGlue: Learning Feature Matching with Graph Neural Networks*, 2020, pp. 4938–4947.

[17] Y. You, Y. Lou, C. Li, L. Li, L. Ma, C. Lu, and W. Wang, "Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13 647–13 656, 2020.

[18] H. Dutagaci, C. P. Cheung, and A. Godil, "Evaluation of 3d interest point detection techniques via human-generated ground truth," *The Visual Computer*, vol. 28, pp. 901–917, 2012.

[19] K. Kawano, S. Koide, and T. Kutsuna, *Learning Wasserstein Isometric Embedding for Point Clouds*, 2020, pp. 473–482.

[20] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, *Learning Representations and Generative Models for 3D Point Clouds*, 2018, vol. 80, pp. 40–49.

[21] H. Fan, H. Su, and L. J. Guibas, *A Point Set Generation Network for 3D Object Reconstruction From a Single Image*, 2017, pp. 605–613.

[22] A. A. Taha and A. Hanbury, "An efficient algorithm for calculating the exact hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2153–2163, 2015.

[23] W. S. P. Fernando, L. Udawatta, and P. Pathirana, "Identification of moving obstacles with pyramidal lucas kanade optical flow and k means clustering," *2007 Third International Conference on Information and Automation for Sustainability*, pp. 111–117, 2007.

[24] N. Sharmin and R. Brad, "Optimal filter estimation for lucas-kanade optical flow," *Sensors*, vol. 12, no. 9, pp. 12 694–12 709, 2012.

[25] J. L. Schönberger, E. Zheng, M. Pollefeys, and J. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.