MSc thesis in Geomatics

Creating a methodology to more objectively measure the performance of reconstruction algorithms for large urban objects generated from low detailed complete ground truth models

Sérénic Monté

June 2024

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Nail Ibrahimli
	Hugo Ledoux
Co-reader:	Shiming Wang

Abstract

In this thesis we present a new idea to objectively assess reconstruction algorithms. Because it is not feasible to completely scan a high-detailed ground truth mesh of large urban objects, the performance of the reconstructed meshes can therefore not be measured objectively. To solve this, we present a new mesh evaluation methodology that can more objectively assess the quality of the generated mesh based on a low detailed ground truth mesh. We achieved this by creating a synthetic dataset based on low-detailed models of large urban buildings and using this as a ground truth mesh and input data for the reconstruction algorithms. Thanks to our new methodology, we were able to compare the output mesh and ground truth mesh using a wireframe model of the meshes. This allows us to give a more objective score to the results without having to look at entire model, which is the usual method. The results of this thesis show that the new methodology has potential to be used for creating a new benchmark, and it opens a new door to using more readily available objects that could not be used before.

Acknowledgements

This work would not have been possible without the help, support and fruitful discussions with my supervisors Nail Ibrahimli, Doctoral student at the 3DUU lab and part of the 3D Geoinformation group at Delft University of Technology, and Hugo Ledoux, associate-professor in the 3D geoinformation research group at the Delft University of Technology. Their example has taught me what a good researcher should be, and I probably could never give them enough credits for this. I would like to thank all of those with whom I had the privilege to work with during this Master thesis. I would also like to thank my family for their never ending love and their unrelenting support.

Contents

List of Figures

List of Tables

Acronyms

NVS	Novel View Synthesis	xiii
NSR	Neural Surface Reconstruction	xiii
MVS	Multi-View Stereo	xiii
SfM	Structure from Motion	5
NeRF	Neural Radiance Fields	2
TSDF	truncated signed distance field	7
SDF	signed distance field	9
MLP	multilayer perceptron	7
dC	standard Chamfer distance	9
PSNR	peak signal-to-noise ratio	xiii
SSIM	structural similarity index measure	xiii
MSSIM	mean structural similarity index measure	11
PMP	Polygon Mesh Processing	22
CGAL	Computational Geometry Algorithms Library	22

1. Introduction

In our society the use of augmented reality and digital twins is playing an evermore important role. For those applications computer vision and 3D model reconstruction techniques are used for instance in: autonomous driving, reconstruction of historical monuments, engineering and digital archives [Farshian et al., 2023; Barron et al., 2023; Li et al., 2023; Mildenhall et al., 2021]. All these applications rely on the use of a digital representation of the environment. The demand for easily reconstructing the environment is increasing. Apart from manual creation of models, there are two major ways to generate models. There is active acquisition like laser scanning and more recently passive acquisition using images to reconstruct objects. To scan a scene or object using active acquisition is a complex task and requires skilled personnel. Passive acquisition on the other hand is a relatively simple process. It only requires images or video of the scene or object. The images or video can be obtained from a camera, mobile phone or captured by drone. The output of active acquisition has a high accuracy and can reach a high level of detail, since the model is acquired by taking active measurements. The passive methods try to reconstruct the environment by estimating the location of the original points, and therefore skipping the expensive process of taking active measurements. At the moment the digital reconstruction of 3D objects is subject of much research, especially to improve the accuracy and details.

Creating a high detailed model from a large urban object using active measurements is prohibitively expensive. Not only is it very expensive to generate a high definition scan but most importantly it is almost impossible to scan an entire building from all angles. It takes a couple of days to scan large objects and keeping the light conditions similar is often difficult, which is helpful for the scan alignment during post-processing. It is nearly impossible to scan a large urban object over several days without any moving interference like humans, animals, or other moving objects. Also, non-movable obstacles like vegetation and buildings create occlusions. Manual creation of a highly detailed 3D model of a large building is near impossible, because it is a too time-consuming task.

To track the quality of the outputs of reconstruction algorithms we need to have a benchmark that compares the reconstruction with the ground truth model for large urban objects. However, for large urban objects there are to the best or our knowledge no benchmarks present that have a highly detailed complete ground truth model. There are several benchmarks have been developed for measuring the quality of reconstruction algorithms, for instance the DTU [Aanæs et al., 2016] and Tanks-And-Tempels [Knapitsch et al., 2017] benchmark. These benchmarks use a ground truth model created using active measurements and are therefore limited to mostly relatively small objects or large parts of the ground truth mesh are missing from the ground truth model.

For this thesis we will look into the following three major categories of reconstruction algorithms: Multi-View Stereo (MVS), Novel View Synthesis (NVS) and Neural Surface Reconstruction (NSR). All three of these methods attempt to render an object based on input images/video, with the output being either a video, a mesh, a point cloud, or images. Progress in the field of reconstruction algorithms is happening at an unprecedented rate. The quality and speed of rendering objects are getting better and faster [Pérez et al., 2023]. This opened the possibility of developing photo realistic 3D models of objects or environments. The example of the progress in speed and quality can be seen in Novel View Synthesis, which contains the well-known Neural Radiance Fields (NeRF) algorithm [Mildenhall et al., 2021]. When NeRF was first used in 2020 it took several hours to render a scene, the fastest method nowadays only takes a fraction of an hour with a significantly improved quality [Barron et al., 2023]. It struck us that we could not find any examples of large urban objects reconstructed with these algorithms with a complete low detailed, or high detailed for that matter, ground truth model.

We are interested if it may be possible to create a methodology to be able to objectively measure the quality of the output with a low detailed complete ground truth model. Therefore, we have to take a different approach in order to be able to measure the various algorithms for large buildings. The current metrics compare the (low detailed) ground truth model with the generated results, but the various rendering algorithms can often display the broad lines well. The quality of the model is on the other hand determined by the extent to which the fine details are displayed. This can be clearly seen in the evaluation of Li et al., see Figure 1.1. Our definition of a low detailed complete ground truth model is a model were only the major shapes of the building is modelled, but the fine details are omitted. The textures and details are shown with images. The structure should be modelled from all above ground angles.

The research question we want to answer in this thesis is: **Can we develop a methodology that allows us to evaluate the quality of reconstruction algorithms, for architectural purposes, without the use of a highly detailed complete ground truth model?**
Our research question lead to the following sub-questions:

1. Are the reconstruction algorithms able to reconstruct a scene from a low resolution model of a large urban object.
2. What are the features that we can extract from a low resolution model, that can be used to test the quality of the generated meshes.
3. Does the width of the region around the extracted feature influence the quality of the evaluation metrics.

In this thesis we show that it is possible to create a methodology for a more objective evaluation of the performance of mesh reconstruction algorithms of large urban objects generated from low detailed complete ground truth models. We have also demonstrated that it is possible to render large urban objects from low detailed manual modelled buildings with photo (realistic) textures. The code and dataset will be available at https://github.com/meshReconstruction/MeshReconstructionEvaluation.git

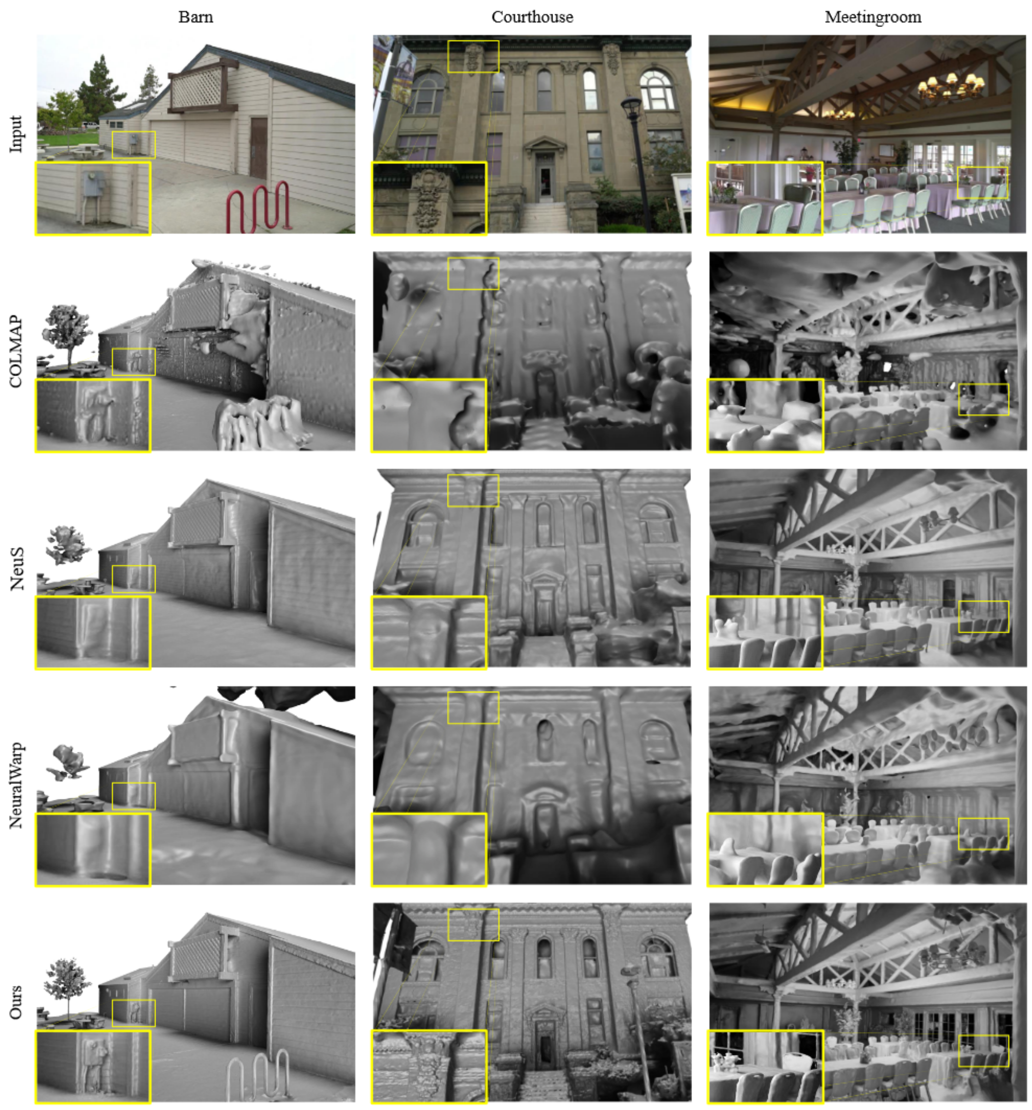


Figure 1.1.: The high quality versus low quality blurred details. Taken from [Li et al., 2023]

2. Related work

In this section we review the scientific research related to this thesis. First we will discuss the existing algorithms, next we will dive into the different evaluation metrics, and lastly we will discuss the different types of existing benchmarks. Reconstructing 3D objects is a computer vision problem that has existed since before the use of image convolution neural networks. Currently, the 3D computer vision for 3D model acquisition research is mainly focused on algorithms using neural networks. The existing deterministic techniques like Multi-View Stereo have matured [Li et al., 2023]. The algorithms we use in this thesis render an object based on input images/video, with the output being a point cloud for MVS, a mesh and video or images for NSR and a video or images for NVS.

2.1. Algorithms

2.1.1. Multi-View Stereo

Multi-View Stereo is used to generate a detailed point cloud from several photos or videos of an object. Because the input consists of 2D photos and videos [Schönberger et al., 2016], MVS techniques have to estimate the most logical 3D point locations of the original object based on the original input images that the object probably had [Schönberger et al., 2016]. The MVS algorithms do this by using various deterministic rules with the use of camera orientation and location [Gu et al., 2020]. If the positions are not known we can use Structure from Motion (SfM) algorithms to estimate the probable camera positions relative to the world origin. MVS algorithms can be subdivided into two main categories, Learning-based and traditional algorithms. For deterministic algorithms we can subdivide it into four sub categories: voxel-base, surface evolution, depth map and patch based algorithms. For learning based algorithms there are two main subcategories, depth map based algorithms and volumetric algorithms.

Traditional

Voxel based algorithms require that a bounding box of the scene is known beforehand. One of the major downsides of these types of algorithms is that the resolution of the grid limits the accuracy of the reconstruction. This can partially be alleviated by using adaptive grids or graphs. Since a bounding box is required for these algorithms, they are often limited to reconstructing objects and not scenes, since it is relatively easy to estimate a bounding box for an object [Sinha et al., 2007].

Surface evolution based algorithms use a model as starting point, for instance a visual hull. The algorithm refines the hull in an optimization based process. Since the creation of the initial model is often difficult for large scenes. They are therefore more suitable for reconstruction of objects and not large scenes [Furukawa and Ponce, 2005].

Depth map based methods do not use a discretized volume with a restriction on the possible depth values corresponding to the predefined accuracy, but calculate a continuous depth for every pixel in the images. The depth can be calculated by sweeping a plane through 3d space parallel to the reference camera and find a corresponding 2D projection for every location on the plane. With the depth maps the scene can be reconstructed by fusing them [Strecha et al., 2006].

Patch based methods first create a set of patches that cover the estimated surfaces of objects. These patches are obtained from pixel level correspondences between images. These patches and their correspondences are then used to create a depth map. Similar as with the depth map based methods these maps can be fused to reconstruct a scene [Furukawa and Ponce, 2010].

Learning based

Depth map based algorithms work similar to the deterministic depth map and patch based algorithms as they also use depth maps to reconstruct the scene. Most of the deep learning based methods in this category use image convolutional neural networks to extract features from the input images. The extracted features in combination with the camera information is used to construct a cost volume which in turn is used to estimate the depth map. This depth map is refined with the use of an image convolutional network [Yao et al., 2018]. As for most of the learning based algorithms it is necessary to pretrain the network on a large set of ground truth data. The pretrained models can be used in reconstructing new scenes [Wang et al., 2021c].

Volumetric based algorithms use the idea of voxel based and surface evolution based algorithms and use a volumetric representation like a voxel grid. The main drawback of these representations is that they are restricted to small-scale problems [Wang et al., 2021a]. The surface would be represented by marking all the voxels that would be on the surface. To better scale the network to larger scenes, different volumetric representations are used like the truncated signed distance field (TSDF) [Wang et al., 2021c].

2.1.2. Novel View Synthesis

Novel View Synthesis are methods to generate new unseen images from a set of input images or video. NVS does not generate a mesh or point cloud but generates RGB images or a video of the object. In 2020 there has been a breakthrough in the NVS field with the algorithm NeRF [Mildenhall et al., 2021]. NeRF made it possible to generate photo realistic unseen images with neural networks [Mildenhall et al., 2021]. The input data for MVS and NVS is identical. Even though the input is the same, the output is very different. NVS is focussed on generating an image or video of the object from unknown view points whereas the main focus of MVS lies in generating a 3d point cloud of the scene.

The main idea behind NeRF is to represent the appearance of the scene as a radiance field. This field is a function of 3D positions and viewing directions and describes for every point in the field how light interacts with the surfaces [Rabby and Zhang, 2024]. 5 dimensional points, consisting of location and viewing direction, are sampled along the camera rays and fed into an multilayer perceptron (MLP) which outputs a colour and density for each position sample. The colour with their density are then aggregated into the final pixel colour. To combine density into colour, a colour must be sampled for each point along the ray. For the NeRF algorithms this continuous integral is estimated by means of quadrature. Quadrature is often used with discrete voxel grids, however using a fixed grid will only limit the MLP because it is sampled at fixed discrete points. The authors of NeRF solved this by using stratified sampling. The ray is divided into evenly spaced bins, from each bin a sample is randomly and uniformly drawn. Not all areas of the ray are equally interesting and sampling many points along the complete ray is inefficient. Therefore, the NeRF algorithm samples in two networks, first a coarse network and afterwards a fine network. The coarse network uses uniform stratified sampling along the ray, the fine network uses the information learned from the coarse sampling and samples with a bias towards the relevant parts. Finally, calculating the colour can be reduced to a traditional alpha compositing problem [Mildenhall et al., 2021].

The original NeRF received much attention in the computer vision community and as a result many algorithms are now based on the original NeRF architecture. The original NeRF algorithm is slow. It takes a few hours up to two days to train the network. Synthesizing new images in a trained network takes between less than a second up to 30 seconds, depending on the resolution. Other methods improve the training and rendering time by optimizing the ray tracing with more aggressive empty space skipping and early ray termination [Tewari et al., 2021]. Other NeRF based methods use different data structures like trees, hashes, grids and sparse-grids to speed up the algorithm. Instant NGP managed to reduce the training time of the algorithm to a couple of seconds by using a multi-resolution hash encoding [Müller et al., 2022]. There are also methods that focus on improving the render quality of NeRF. For instance, Barron et al. [2021] proposed a method (MipNeRF) that does not determine the colour of the pixel by casting a ray but by casting a cone defined by the camera and area surrounding the pixel. MipNeRF takes a similar approach to sampling the cone as NeRF, however since it is casting a cone, the value of each sampled frustum is calculated by computing the integral of the positional encoding in the radiance field. They approximate the frustum as a Gaussian distribution [Barron et al., 2021]. Other works improve the quality of NeRF by focussing on improving the robustness to better handle objects with different lighting conditions [Tewari et al., 2021; Rabby and Zhang, 2024].

In 2023 there was another breakthrough in the field of NVS with Gaussian splatting [Kerbl et al., 2023]. This technique is faster, and the rendering quality is often of higher quality than that of the NeRF based methods. Gaussian splatting represents the scene using 3D Gaussians. Each Gaussian represents a colour and is defined with a position, covariance, alpha and spherical harmonics component. The spherical harmonics component is used to represent the directional component of the radiance field. The initial set of Gaussians might not represent all the objects in the scene well, therefore the algorithm increases the number of Gaussians every 100 iterations according to a fixed set of rules. If a Gaussian does not meet certain criteria such as it is too large, too small, or the alpha value is below a threshold, it can either remove, duplicate, split, move or change the orientation of the Gaussian. To decrease the rendering time the Gaussian splatting algorithm uses a differentiable rasterizer, which sorts the Gaussians for an entire image instead of for every pixel and ignores Gaussians close to the view plane and those far outside the view [Kerbl et al., 2023].

2.1.3. Neural surface Reconstruction

Neural Surface Reconstruction is a technique based on Novel View Synthesis and specifically on the NeRF architecture. NSR uses the same input data as the NeRF algorithms, but the output is focussed on meshes. Due to the nature of the algorithm it is also possible to create synthetic images. Generally the quality of these renders are of lower quality than those of the NVS algorithms [Li et al., 2023]. An upside of the NSR algorithms is that they do not assume that all surfaces are Lambertian, which is typical for the MVS algorithms. A downside of the NVS and NSR algorithms is that they often have difficulty with repetitive patterns and large areas of homogeneous colour [Li et al., 2023]. The main difference between the NVS algorithms and the NSR algorithms is that the latter uses implicit functions instead of the volume density field. Neural implicit functions like occupancy grids or signed distance field (SDF) are better suited for representing the surface of an object [Wang et al., 2021b].

2.2. Evaluation metrics

The most common metrics used to evaluate the meshes and point clouds generated by MVS and NSR are:

- standard Chamfer distance (dC) [Farshian et al., 2023]
- Hausdorff distance (d_{hf}) [Farshian et al., 2023]
- F-score [Farshian et al., 2023]

There are many more metrics that can be used to evaluate the generated models. These methods, EMD/Wasserstein distance, IoU/Jaccard Index, Normal consistency, Jensen Shannon divergence, Coverage, Minimum matching distance, Light field descriptor, are mainly used to evaluate the output generated by MVS [Farshian et al., 2023].

The most commonly used evaluation metrics we use for images are:

- PSNR [Rabby and Zhang, 2024]
- SSIM [Rabby and Zhang, 2024]

Other methods that can be used to evaluate the correspondence between two images are: root mean square error, feature similarity indexing method, information theoretic-based statistic similarity measure, spectral angle mapper, and universal image quality index [Müller et al., 2020].

2.2.1. Mesh and Point cloud evaluation

standard Chamfer distance is widely used to measure the similarity between two point clouds [Lin et al., 2023; Ibrahimli et al., 2023]. The standard Chamfer distance calculates the minimal distance between the points in the two different point clouds. The smaller the average distance, the more the models resemble each other. The standard Chamfer distance as described in [Huang et al., 2023] is calculated as follows, see Equation 2.1.

$$dC = \frac{1}{2}(Comp + ACC) \tag{2.1}$$

$$Comp = \frac{1}{|X_{gt}|} \sum_{x_{gt} \in X_{gt}} min_{x_{pd} \in X_{pd}} \|x_{gt} - x_{pd}\|$$

$$Acc = \frac{1}{|X_{pd}|} \sum_{x_{pd} \in X_{pd}} min_{x_{gt} \in X_{gt}} \|x_{pd} - x_{gt}\|$$

Where X_{gt} the ground truth model is, x_{gt} a sample from the ground truth model, the X_{pd} the predicted model and x_{pd} a sample from the predicted model. [Huang et al., 2023]:

The Hausdorff distance metric is the maximum distance between two models and defined as follows:

$$d_{hf} = max\left[d(X_{pd}, X_{gt}), d(X_{gt}, X_{pd})\right] \tag{2.2}$$

$$d(X, Y) = max_{x \in X} d(x, Y)$$

$$d(x, Y) = min_{y \in Y} \|x - y\|$$

The F-score (Equation 2.3) as described in [Huang et al., 2023; Knapitsch et al., 2017] is the harmonic mean of the precision, Equation 2.4, and recall, Equation 2.5, at a given threshold ϵ:

$$F - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.3}$$

$$Precision = \frac{\left|\left\{x_{pd} \in X_{pd} | min_{x_{gt} \in X_{gt}} \|x_{gt} - x_{pd}\| < \epsilon\right\}\right|}{|X_{pd}|} \tag{2.4}$$

$$Recall = \frac{\left|\left\{x_{gt} \in X_{gt} | min_{x_{pd} \in X_{pd}} \|x_{pd} - x_{gt}\| < \epsilon\right\}\right|}{|X_{gt}|} \tag{2.5}$$

A reconstruction must be complete as well as accurate for a high F-score. [Huang et al., 2023; Knapitsch et al., 2017].

2.2.2. Image evaluations

There are two methods that are commonly used for evaluating images; peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM).

The PSNR is a method to calculate the quality of an output image compared to the corresponding unseen ground truth image. The higher the PSNR, the better the quality of the output image. The Equation 2.6 represents this mathematically.

$$PSNR = 20 \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right) \tag{2.6}$$

$$MSE = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} ||f(i,j) - g(i,j)||^2$$

With f the original image matrix, g the rendered image matrix. Both f and g are of size $m \times n$. $f(i,j)$ returns the pixel value at the ith row and jth column. MAX_f returns the maximum pixel value of the original image matrix.

The second method is *structural similarity index measure*, which is a method that compares images by mimicking the human vision. The method was developed by Wang et al. [2004] based on the idea that human vision is well-developed to extract structural information from scenes. Structure similarity looks at a group of pixels instead of individual pixels such as PSNR. Structure similarity is made up of three different components, luminance (Equation 2.7), contrast (Equation 2.8) and structural information (Equation 2.9). The SSIM (Equation 2.10 and 2.11) can be used to score a complete image, however Wang et al. found that it is more useful to apply the matrix to local regions and taking the mean of all these windows. The average score can be calculated by taking the mean of the SSIM for the different windows (mean structural similarity index measure (MSSIM) Equation 2.12) [Wang et al., 2004].

$$l\left(x,y\right)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1} \tag{2.7}$$

$$C_1=\left(K_1L\right)^2$$

$$c\left(x,y\right)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2} \tag{2.8}$$

$$C_2=\left(K_2L\right)^2$$

$$s\left(x,y\right)=\frac{\sigma_{xy}+C_3}{\sigma_x^2+\sigma_y^2+C_3} \tag{2.9}$$

$$C_3=\frac{C_2}{2}$$

$$SSIM\left(x,y\right)=\left[l\left(x,y\right)\right]^{\alpha}*\left[c\left(x,y\right)\right]^{\beta}*\left[s\left(x,y\right)\right]^{\gamma} \tag{2.10}$$

With both K_1 and $K_2 \ll 1$, the pixel value range is L when using $\alpha=\beta=\gamma=1$

$$SSIM\left(x,y\right)=\frac{\left(2\mu_x\mu_y+C_1\right)\left(2\sigma_{xy}+C_2\right)}{\left(\mu_x^2+\mu_y^2+C_1\right)\left(\sigma_x^2+\sigma_y^2+C_2\right)} \tag{2.11}$$

$$MSSIM\left(X,Y\right)=\frac{1}{M}\sum_{j=1}^{M}SSIM\left(x_j,y_j\right) \tag{2.12}$$

Where M is the number of local windows and x_i is the ith window of image X.

2.3. Existing Benchmarks

Several benchmarks are available for MVS, NVS and NSR. In this section we cover a selection of benchmarks available for the different rendering methods. We look at the individual strengths and weaknesses of the selected benchmarks and their unique features. Because of the architectural focus of this thesis we highlight the challenges with urban datasets/objects.

The research paper from Strecha et al. [2008] is often cited and considered an important dataset. In this thesis a dataset was created in the open air and contains 3 different building facades. A ground truth model was made using a LIDAR scanner. This benchmark was very innovative for its time. It showed that it is very important for MVS to have a detailed ground-truth model. The DTU and Tanks and Temples benchmark based their ground-truth model on these findings of Strecha et al. [2008].

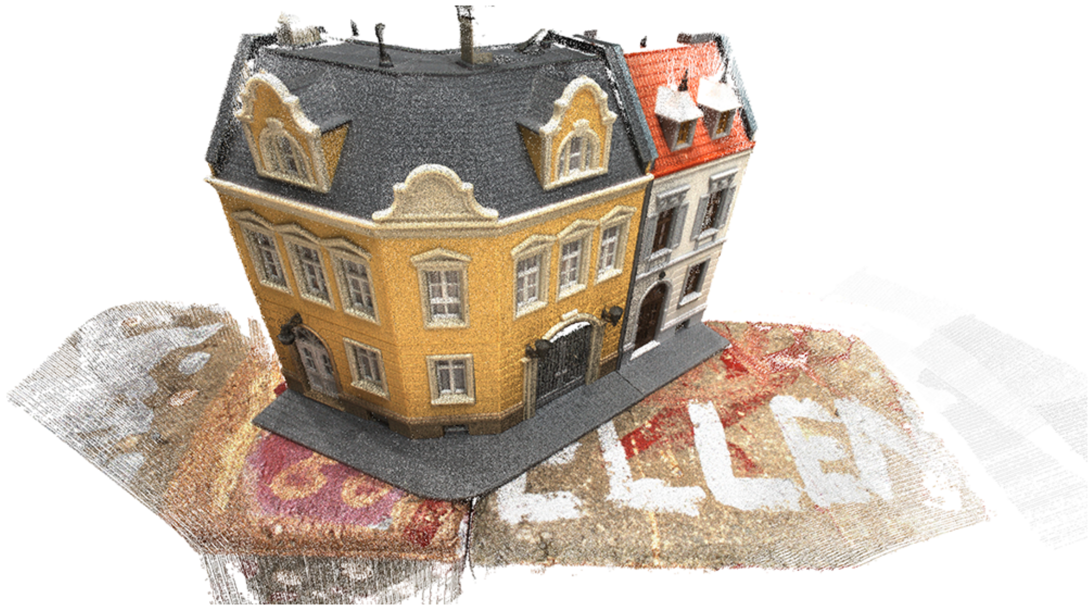


Figure 2.1.: Example image from DTU, courtesy to [Aanæs et al., 2016] small house set 6

The DTU dataset is a commonly used benchmark dataset for MVS, NSR and NVS. The DTU benchmark of Aanæs et al. [2016] consists of scans from 124 different objects. Each scene contains between 49 and 64 different positions and the resolution is 1600 × 1200. The camera parameters are also included. They are of high quality because they have been calibrated with the Matlab calibration tool. The scenes and all the positions were taken in seven different exposures. All objects in this dataset are small objects that fit on a table, like the toy house shown in Figure 2.1 [Aanæs

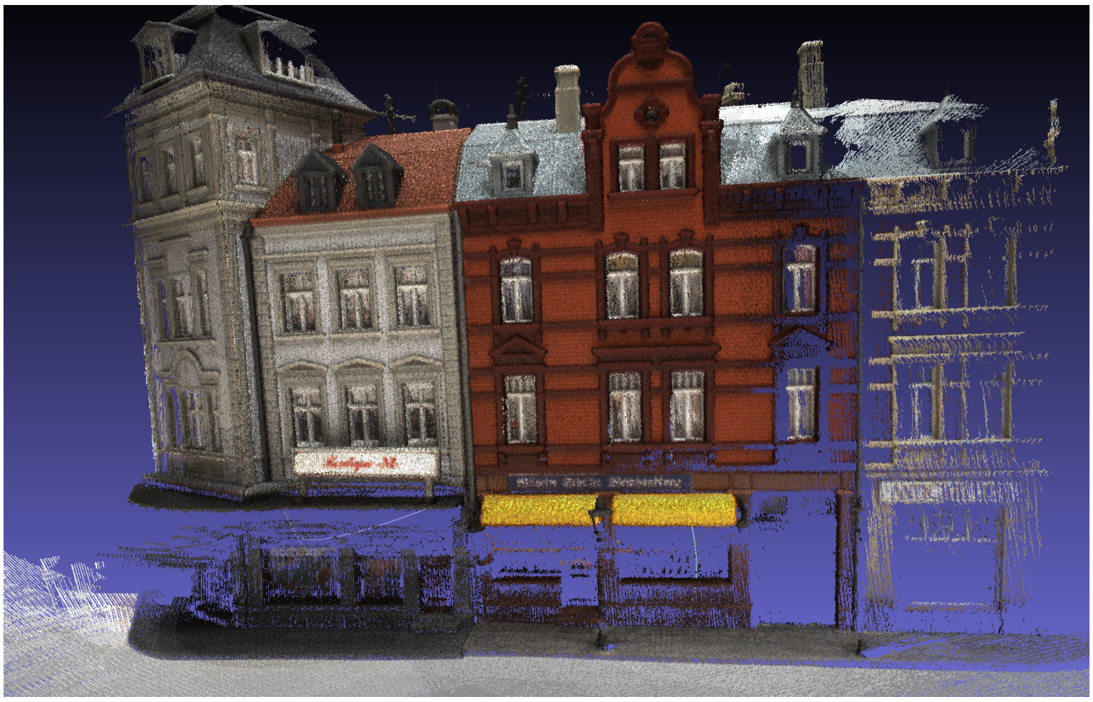


Figure 2.2.: Example of gaps in the DTU dataset

et al., 2016]. The objects have complex shapes and have challenging surface features such as reflections. The recordings were all made indoors, therefore lighting was standardized. The photos were taken using a robotic arm, so that all photos have a standard distance to the object. Unfortunately the scans are not all 360°. This results in gaps in the ground truth mesh, see Figure 2.2. The 360° scans that are available are made with 90° intervals. This resulted in low quality input data and have for that reason been excluded from the benchmark. The lack of 360° input data makes it hard to evaluate the different 3D algorithms because the output objects are 3D objects. The nature of the objects, toys and model buildings, means that the finer details, which are present in live size objects, are not present in the models.

Another important benchmark that exists is the Tanks-and-Temples benchmark made by [Knapitsch et al., 2017]. This benchmark consists of a mix of large objects such as buildings and smaller objects like vehicles. The objects are filmed in high resolution and the ground truth models are made with a at that time state-of-the-art precision laser. These ground truth models are very detailed. This makes it possible to test the algorithms on their ability to represent small details. On the other hand, especially on larger objects the models have major gaps. For instance the courthouse misses almost the entire roof and only part of the dome is included in the ground truth, see Figure 2.3 and Figure 2.4. High detailed ground truth models are important for the MVS and NSR benchmarks because the algorithms have difficulty to render the small

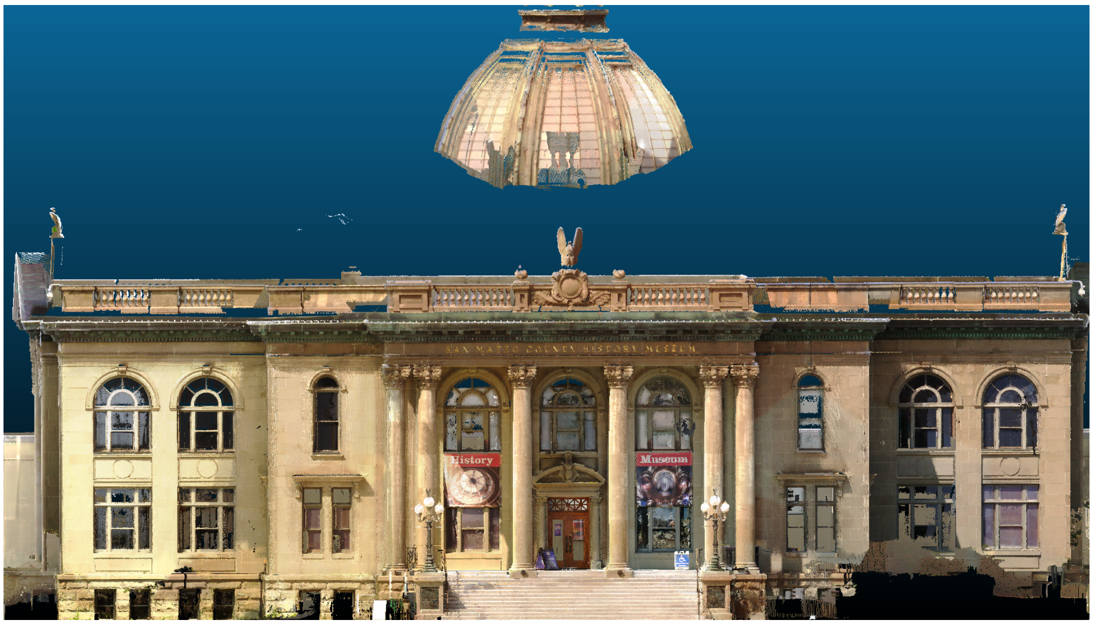


Figure 2.3.: Frontal view of the cleaned courthouse ground truth scan from the Tanks and Temples dataset

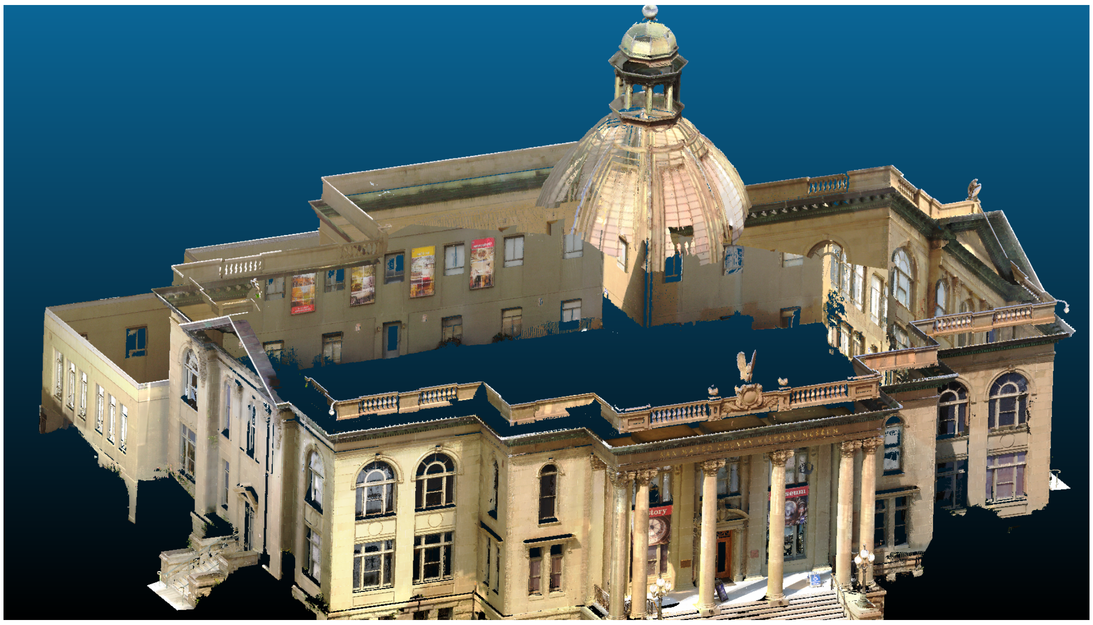


Figure 2.4.: The courthouse objects misses a lot of data especially for the roof

details. The precise scanning with a laser scanner of the object is a lengthy process. Scanning the largest object in the Tanks-and-Temples dataset took two days. These scans have, as always, a high chance of the scan failing for example, due to too many people and vehicles moving through the scene [Knapitsch et al., 2017].

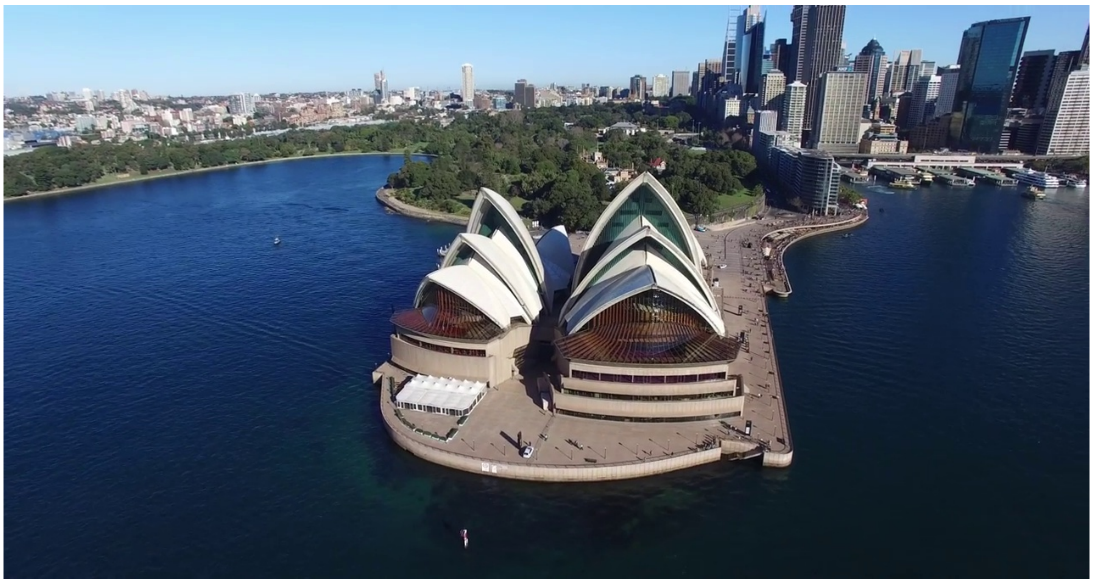


Figure 2.5.: Frame from the OMMO dataset

Various benchmark data sets are available for Novel View Synthesis (NVS). In contrast to the other forenamed benchmarks the NVS benchmarks consist only of video and or images. This makes the creation of large outdoor data sets and benchmarks relatively easy. OMMO is the largest and most resent dataset and benchmark, it contains 33 different objects, see Figure 2.5, [Lu et al., 2023]. All datasets that have high definition images or videos of objects can be used as benchmarks for NVS. Therefore, all the MVS and NSR benchmarks are also used for NVS. To the best of our knowledge there are no benchmark datasets that have highly detailed complete models for large urban objects.

3. Methodology

We can divide the methodology in tree major blocks: Dataset, Algorithm pipeline and Evaluation method.

3.1. Dataset

In Figure 3.1 the schematics of the different steps are visualized.

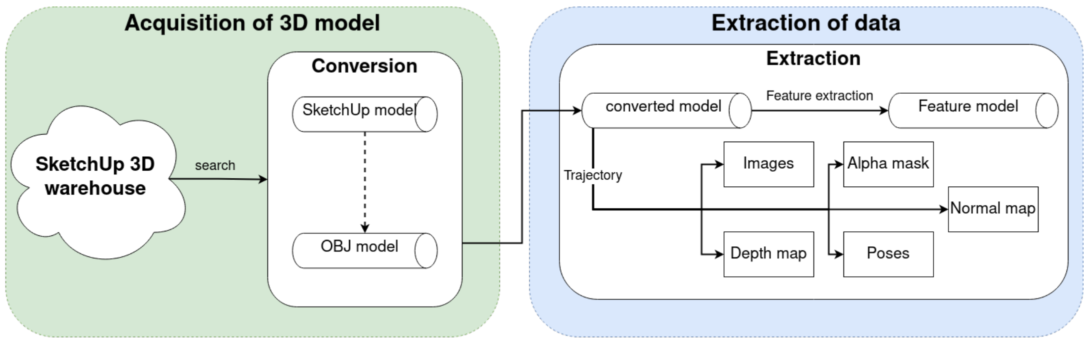


Figure 3.1.: The data pipeline

1. Selecting suitable models in SketchUp warehouse[1].
2. Convert SketchUp model to OBJ format.
3. Load the model into Blender and adjust the image rendering script to ensure that the camera is correctly rotated around the object, to produce the usable images and poses for the algorithms.

3.1.1. Selecting suitable models in SketchUp warehouse

We selected SketchUp Warehouse models that met the following criteria:

1. The model must be complete, i.e. the whole building should be modelled with as much detail as possible.

[1]https://3dwarehouse.sketchup.com/

2. The model must have photo realistic textures.
3. The model must have a correct scale.

SketchUp Warehouse is a public database for SketchUp models, anyone with a SketchUp account can upload their models here and there are no requirements for the uploaded models. As a result, the quality of most models tuned out to be too low to be usable. Therefore, we had to switch to native OBJ models, but most publicly available OBJ models do not contain image based textures like the SketchUp models.

3.1.2. Convert from SketchUp model to OBJ

In order to use the model they have to be loaded into Blender, however Blender does not support SketchUp objects natively. Therefore, we opted to convert the SketchUp models to OBJ, which can be loaded into Blender. We use Rhino 7 [2] to convert the SketchUp models to OBJ since Rhino supports both formats.

3.1.3. Model to input dataset

In Blender, we had to perform the following steps to convert a model to was suitable for input to the algorithms, these are:

1. place OBJ model on axis origin in Blender.
2. move camera target to centre of mass of the model.
3. tweak camera path so that it revolves around the entire model.
4. define the number of frames to generate.
5. for every frame render image.
6. for every frame compute normal map.
7. for every frame compute alpha mask.
8. for every frame compute depth map.
9. for every frame save camera positions (track extrinsic parameters, i.e. rotation and translation).
10. calculate camera intrinsics: focal length, pixel translation to centre of image, image resolution.

[2] https://www.rhino3d.com/7/

We then placed the object with the centre of the footprint on the world axis. The camera tracks to origin of the model, therefore we move the origin of the model to its centre of mass. To create the render the input images around the model. We rotate the camera around the object at a constant distance from the object and oscillate the height using a cosine. Because of the different sizes of the models, we make sure the camera rotates around the object by moving it further or closer to the object, therefore we also adapt the maximum depth value for the depth maps we generate to fully capture the model. For every frame capture not only the image, but also the normal map, alpha mask and depth map. The last step is to calculate the camera intrinsic parameters.

3.2. Algorithm pipelines

In this section we discuss the algorithm pipeline we test on three different types of algorithms, these are:

1. Multi-View Stereo.
2. Neural Surface Reconstruction.
3. Novel View Synthesis.

We do not run the latest state-of-the-art algorithms for the various algorithms because these algorithms are still being developed and are therefore less stable. It is too time-consuming to debug them. This is why we opted for algorithms that are state-of-the-art but not cutting edge.

The MVS algorithms we run are:

- Colmap [Schönberger et al., 2016].
- PatchmatchNet [Wang et al., 2021a].

The NSR algorithms we run are:

- Neus-facto [Yu et al., 2022].
- VolSDF [Yariv et al., 2021].

The NVS algorithms we run are:

- Nerfacto [Tancik et al., 2023].
- Gaussian splatting [Kerbl et al., 2023].

3.2.1. NSR pipeline

The implementation of NSR consists of a set of state-of-the-art algorithms in SDFStudio. SDFStudio is an add-on to NeRFStudio. The different steps are visualized in Figure 3.2.

1. Prepare SDFStudio dataset: images with poses optional alpha mask.
2. Running algorithms and tweak the hyperparameters.
3. Extract features from mesh.
4. Feature mesh scoring with standard Chamfer distance, Hausdorff distance and F-score.

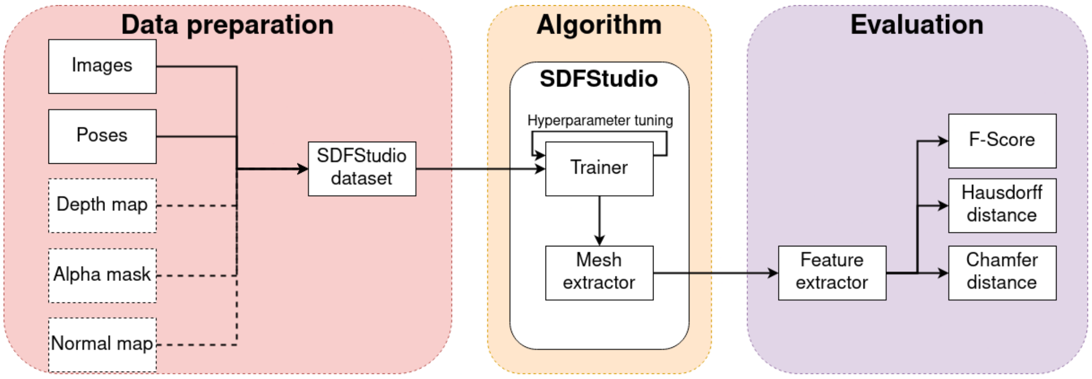


Figure 3.2.: The Neural Surface Reconstruction pipeline

3.2.2. NVS pipeline

NVS has an umbrella implementation (NeRFStudio) the different steps are visualized in Figure 3.3

- Prepare NeRFStudio dataset: images with poses optional alpha mask.
- Running algorithms and tweak the hyperparameters.
- Generating images from the control set.
- Score generated images with corresponding evaluation set images using PSNR and SSIM/MSSIM.

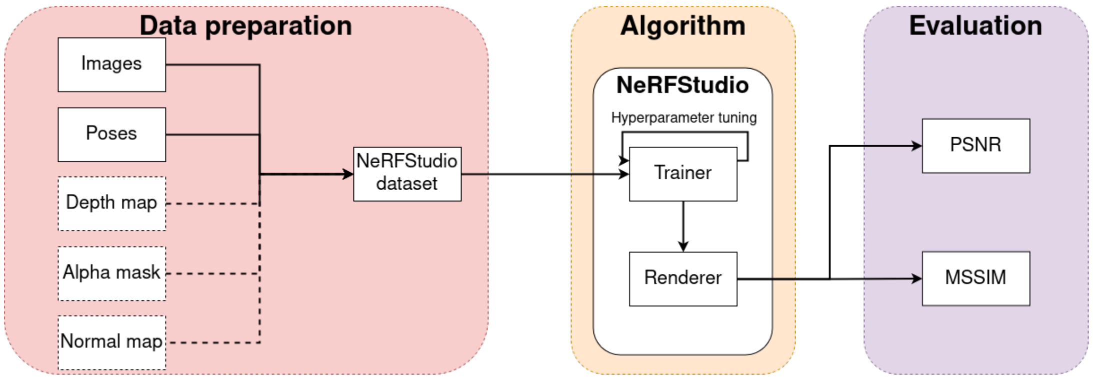


Figure 3.3.: The Novel View Synthesis pipeline

3.2.3. MVS pipeline

MVS does not have an overarching implementation like NeRFStudio and SDFStudio, so for each algorithm the required input must be determined, visualized in Figure 3.4:

1. Prepare input for algorithm.
2. Run algorithms and tweak the hyperparameters.
3. Evaluate Point cloud with standard Chamfer distance, Hausdorff distance and F-score.

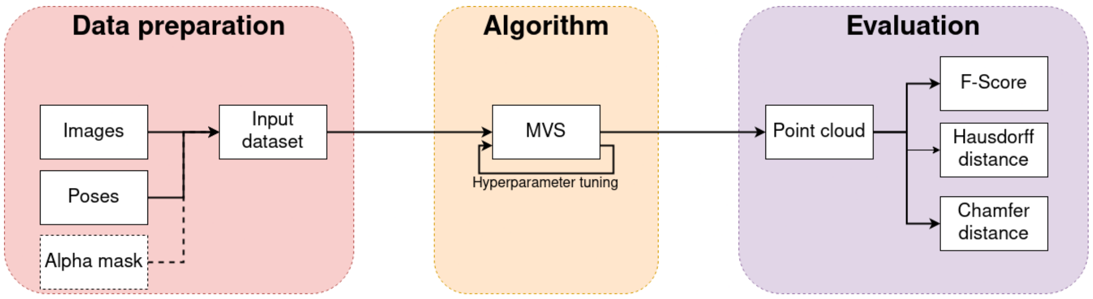


Figure 3.4.: The Multi View Stereo pipeline

3.3. Evaluation

For the evaluation of the algorithm we require a mesh therefore we test this evaluation method on the NSR algorithms. NVS outputs images which we can compare to the generated images and the MVS output is a point cloud. If we were to convert the point cloud to a mesh we needed to use a meshing algorithm, like Poisson reconstruction of Delaunay triangulation. After such a conversion it is unclear what the value of the measurement will be because it is not only evaluating the original output but also the converted output. Therefore, we will only evaluate the MVS output in the traditional manner.

When the reconstruction of a building is evaluated visually most attention is placed on the quality of the finer details. However, when evaluating a reconstruction with a low detailed ground truth mesh the finer details are not evaluated in any way. It struck us that if the algorithm can reconstruct the finer details well, it is also able to reconstruct the edges of the building well. Therefore, we are investigating if evaluating with the sharp features of a mesh is more inline with a visual observation.

To extract the features from the mesh we use the Polygon Mesh Processing (PMP) package in Computational Geometry Algorithms Library (CGAL) [Loriot et al., 2024; The CGAL Project, 2024]. The sharp edges of the input mesh are detected using the detect_sharp_edges function from PMP. We build a wireframe by constructing a mesh from the adjacent faces of the detected features. To determine which edges are sharp the angle between the normals of the adjacent facets is calculated. If the angle is above the specified threshold it will be marked as sharp. Besides varying the threshold we also test with wireframes with only the adjacent facets as well as with wireframes with the two-ring neighbourhood.

The feature meshes can be evaluated with the same metrics as normal meshes. The meshes we will use are the standard Chamfer distance, Hausdorff distance and F-score as described in the previous chapter (2.2.1). We will compare the score for the different feature meshes and the traditional full mesh evaluation with visual observations to determine which combination is most inline with our own visual observation. We use visual observation due to the lack of an objective evaluation method.

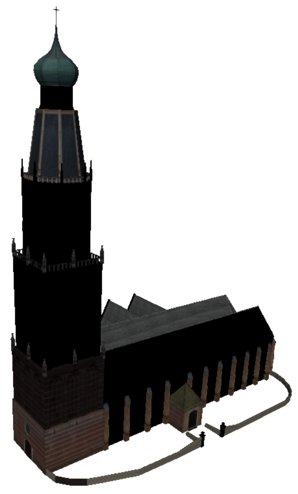
4. Experiments

In this section we discuss the various experiments we have run to investigate whether it is possible to evaluate the quality of neural rendering algorithms, without the use of a highly detailed complete ground truth model. Before we can start with the evaluation of the output of the algorithms, we need to construct a dataset.

4.1. Dataset

We start the dataset creation with selecting models from the SketchUp warehouse that suited our requirements of being complete, correctly scaled and with photo textures. Due to the requirements, many of the models are reconstructions of existing buildings, so images of the buildings were used as textures. These types of textures are more realistic and display fine details that can be found on real buildings. However, many of the textures were missing (Figure 4.1a) and/or misaligned (Figure 4.1b) when converted. Since the textures are created from photos it is a very time-consuming task to repair the models if even possible. We tried to run the algorithms with the models with missing and/or misaligned textures to verify if they were usable. This was however not the case. Therefore, we had to search for models that after conversion had no or minor texture errors that could be corrected in less than a couple of hours. For the conversion of the SketchUp models to OBJ we used Rhino 7.

The next step was generating the input data from the converted OBJ models. We used the following steps to generate the input data: load the model into Blender and tweak the camera path, so it revolves around the model; At fixed intervals save the camera position and orientation, render the image, and compute the normal map, depth map and alpha mask and save the camera intrinsic parameters. With the generated data we could run the algorithms, however with the input data generated with the default settings delivered no usable output when used in the algorithms. Example outputs for MVS can be seen in Figure 4.2a, for NSR can be seen in Figure 4.2b and for NVS can be seen in Figure 4.2c. The NVS output was the most striking example of unusable output. We knew that the implementation of the different algorithms was correct since we tested them with scan 65 (Figure 4.3a) from the DTU dataset [Aanæs et al., 2016].


(a) Model with missing textures

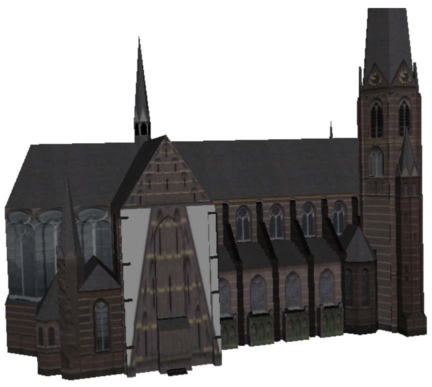
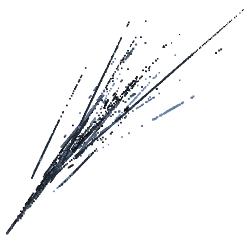
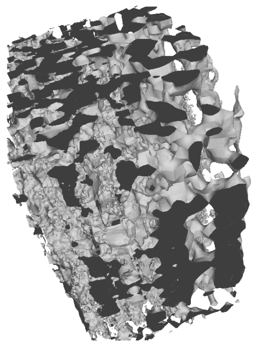

(b) Model with misaligned textures

Figure 4.1.: The examples of meshes that could not be converted


(a) Example of bad MVS output


(b) Example of bad NSR output

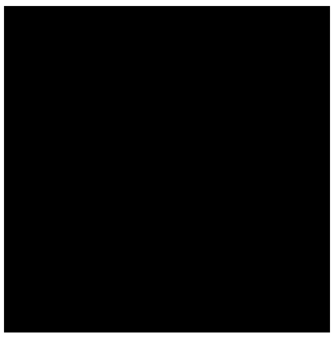

(c) Example of bad NVS output

Figure 4.2.: The examples of unusable output for the three algorithm groups

We were now confronted with the question if the algorithms could not handle large urban objects or if the input data was unusable. First we tried to verify if the algorithm was trained long enough, initially we used 20000 iterations for the NVS and NSR algorithms which took on average six hours to run. So we tried to increase the number of training iterations even up to 200000 to check if longer training would result in meaningful output, this was not the case. Then we tried to run it with fewer iterations, to makes sure the model was not overtrained. This was also not the case. The next question was, can the algorithm handle such models or there something wrong with the input data for these large urban objects. Thus, we searched for examples of large urban objects reconstructed from a low detailed ground truth

(a) Scan 65 from the DTU dataset [Aanæs et al., 2016]

model we were unable to find any examples. Therefore, by trial and error we had to figure out how we had to adapt the input data so that all the algorithms were able to correctly produce output.

With the generation of the input data we could change six settings with which we could experiment in different configurations. To determine the effect of every individual setting we could only change one value and then rerun the different algorithms. This was a very time-consuming and slow process. We started with changing the following settings:

- Alpha masks.
- Normal map.
- Depth map.
- Number of images.
- Distance of camera to object.
- Maximum vertical oscillation.

For example after we increased the number of images, we turned the alpha masks on and off, then we tried it with normal maps turned on or off, etc. Little by little we were able to improve the outcome but still the results were in a usable state. So we started to experiment with settings that were outside the ones we defined in our methodology.

By reverse engineering small object, we tried to determine if there were maybe other settings we could change in order to improve the results. One of the things we tried was: if we were to use the poses from SfM pipeline of COLMAP for small objects, are they different from the poses we would produce with the same small object. Then we figured out that the intrinsic camera parameters we were generating, were different from the COLMAP output. We could improve the quality of the input data considerably. We also tried to change the lighting conditions in order to mimic the original photo textures better. We added directional light sources of various intensities in combination with changes to the original settings. Eventually we also added environmental lighting textures. With the addition of COLMAP poses, directional light, light intensity and environmental lighting, we had ten different settings we could try in different configurations in order to improve output of all three algorithms at the same time. Sometimes a setting would work in one algorithm but not in another.

This meant we made a significant change to the way we originally anticipated our methodology. We anticipated a linear process, but we ended up with a long iterative process to optimize the ten different settings before we could start with evaluating the usability of our research topic. When we first managed to successfully render the first object we were then able to increase the number of usable objects in a relatively easy manner.

This resulted in the ten models displayed in Figure 4.4. We aimed for different types of buildings. We strived for models with different sizes, orientations and geometric complexity. We can divide the models into three different size categories: large, medium and small. Large consists of EWI, Stieltjesweg and Winchester Cathedral, medium include: Apartment, Flat, Neo, Speerhall, Stoommachine. The small models are: House and Villa.
For orientation, we have vertical, horizontal and square buildings. With Apartment, EWI and Neo in the vertical category. The horizontal category consists of: Stieltjweg, House, Speerhall, Villa and Winchester Cathedral. The square category contains Stoommachine and Flat.
The geometric complexity we subdivided the models into two categories: simple and complex. The simple models are models where fewer details are modelled: Stoommachine, EWI, and House. The more complex models are: Stieltjesweg, Apartment, Flat, Neo, Speerhall, Villa and Winchester Cathedral.

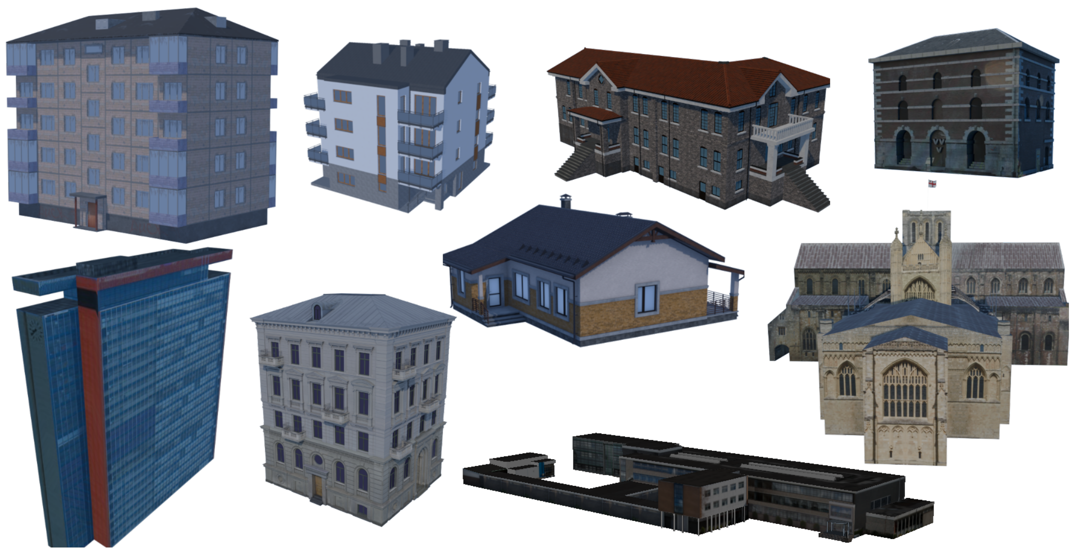


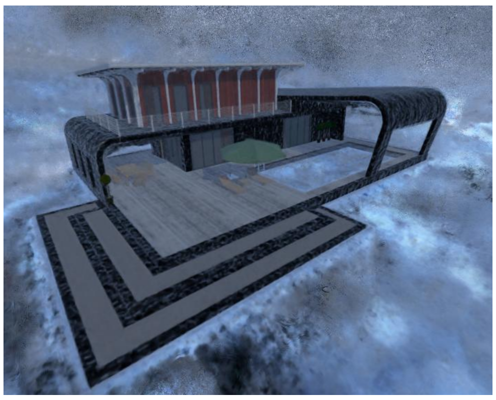
Figure 4.4.: The ten models. From left to right top to bottom: Apartment, Flat, Speerhall, Stoommachine, EWI, Neo, House, Winchester Cathedral, Stieltjesweg

4.2. Algorithms

With the input data now working we can investigate whether it is possible evaluate the quality of the reconstruction algorithms with a low detailed complete ground truth model. We will first discuss the results for NVS. The models we selected have photo realistic textures and therefore the algorithm can be evaluated in a standard manner. Next we will discuss MVS, we are generating a point cloud for with these algorithms. If we would use a meshing algorithm on the point cloud it would no longer be clear if we would measure the quality of the MVS algorithm or of the meshing algorithm. Therefore, we opted to evaluate the output of MVS without feature extraction from an externally generated mesh. The NSR algorithms produce a mesh and the evaluation with feature extraction is tested in this category.

4.2.1. NVS

First we discuss the results of the NVS algorithms. The algorithms we have run for NVS are Nerfacto and Splatfacto. With Nerfacto a NeRF based algorithm and Splatfacto an open source implementation of Gausssian splatting. In table 4.1 the PSNR and SSIM scores are shown for the two algorithms for the ten models. Both Nerfacto and Splatfacto have difficulty rendering EWI (Figure 4.6) and villa (Figure 4.5) as their scores are lower than the other models. EWI 4.6a is the tallest building in our dataset, this may be the reason that the algorithms have difficulty rendering this object since it is zoomed out to fully capture the model. Villa is a building with many small details, such as the railings on the balcony. This fine details might be the reason the algorithms have difficulty with this building.


(a) The Nerfacto rendering for the object Villa

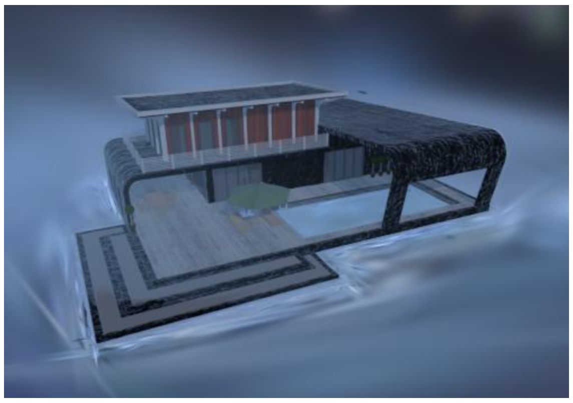
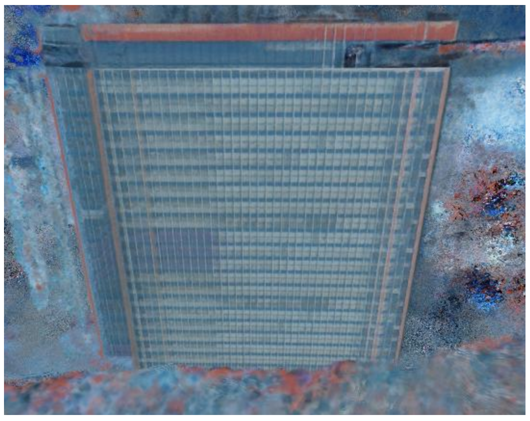

(b) The Splatfacto rendering for the object Villa

Figure 4.5.: The Nerfacto and Splatfacto renderings of the Villa model


(a) The Nerfacto rendering for the object EWI

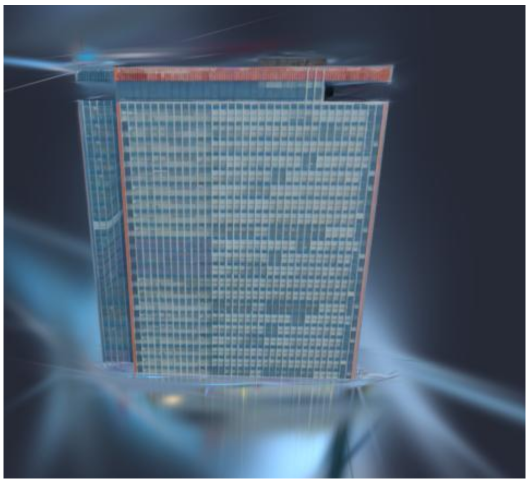

(b) The Splatfacto rendering for the object EWI

Figure 4.6.: The Nerfacto and Splatfacto renderings of the EWI model

mesh	algorithm	PSNR↑	SSIM↑
Apartment	Nerfacto	27.633	0.801
	Splatfacto	18.256	0.931
Ewi	Nerfacto	18.605	0.474
	Splatfacto	14.832	0.912
Flat	Nerfacto	27.892	0.866
	Splatfacto	15.532	0.904
House	Nerfacto	24.591	0.730
	Splatfacto	21.070	0.934
Neo	Nerfacto	27.227	0.735
	Splatfacto	14.102	0.878
Speerhall	Nerfacto	23.698	0.725
	Splatfacto	17.503	0.904
Stoommachine	Nerfacto	24.189	0.754
	Splatfacto	14.212	0.907
Stieltjesweg	Nerfacto	25.842	0.781
	Splatfacto	17.803	0.889
Villa	Nerfacto	18.955	0.552
	Splatfacto	15.558	0.860
Winchester Cathedral	Nerfacto	29.584	0.805
	Splatfacto	17.685	0.925

Table 4.1.: PSNR and SSIM for NVS

4.2.2. MVS

We have run two algorithms for MVS, the first algorithm is COLMAP and the second one is PatchmatchNet. Both algorithms use the poses structure from COLMAP which are quaternions, but he poses we generate are a rotation translation matrix. Normally COLMAP poses are translated to rotation translation matrices for the use in NVS and NSR algorithms. We tried to reverse engineer our rotation translation poses to COLMAP quaternion poses under the assumption that the translation would be symmetrical. Unfortunately we have been unable to reverse engineer it successfully, see examples in Figure 4.7. Therefore, we are using the SfM algorithm of COLMAP to generate poses for the two algorithms.

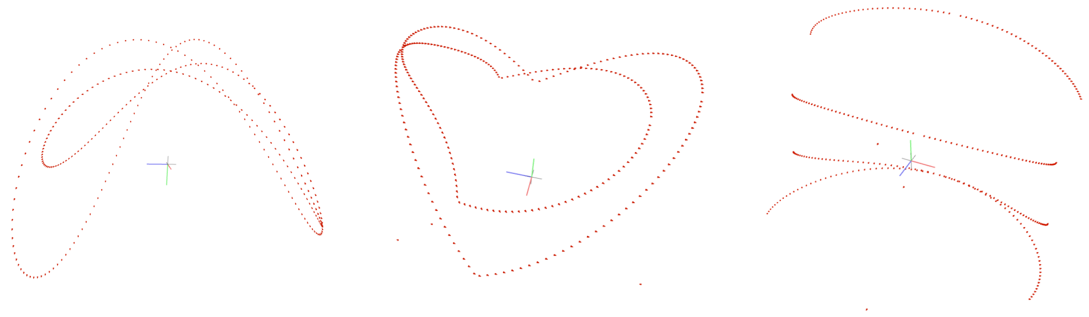


Figure 4.7.: Examples of wrong conversions from rotation translation matrix to COLMAP poses

Both COLMAP and PatchmatchNet had more trouble with reconstructing the buildings than the algorithms in the other two categories. COLMAP performed better than PatchmatchNet on all our models, which can be seen in Figure4.8. This is also reflected in most of the values from table 4.2, especially the F-score shows this. COLMAP was able to reconstruct some models better than others. This is mostly due to the symmetry in some models which the SfM algorithm has difficulty with. An example of a symmetric model is Neo, in Figure 4.9. The COLMAP reconstruction from the front and back of the building is shown, but we can see that the backside is missing.

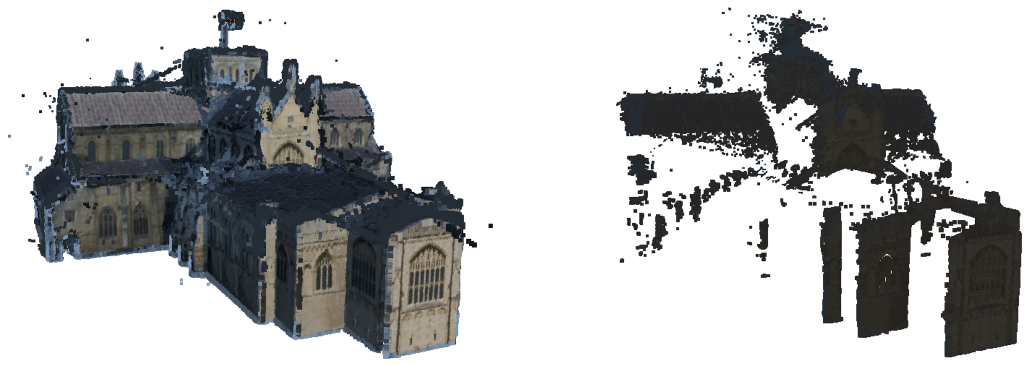


Figure 4.8.: Left COLMAP, right PatchmatchNet output for Winchester cathedral

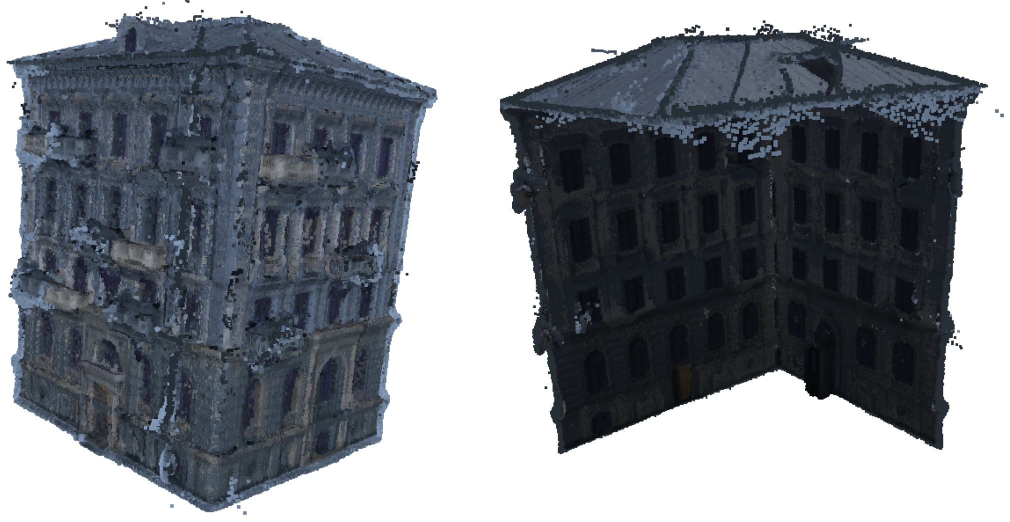


Figure 4.9.: COLMAP reconstruction of the Neo model. Left from the front, right from the back

	COLMAP			PatchmatchNet		
	Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
Apartment	7.264	0.688	0.030	6.707	0.410	0.030
Ewi	17.671	1.038	0.644	18.839	2.071	0.009
Flat	3.902	0.349	0.235	8.104	0.819	0.041
House	5.088	0.411	0.015	3.087	0.060	0.518
Neo	13.299	1.091	0.428	13.774	1.377	0.021
Speerhall	7.622	0.077	0.488	13.061	1.838	0.005
Stieltjesweg	54.983	0.711	0.420	30.048	3.236	0.001
Stoommachine	9.142	0.802	0.585	8.625	1.132	0.001
Villa	15.660	0.134	0.464	10.564	0.759	0.002
Winchester	62.509	0.211	0.426	30.321	4.483	0.000

Table 4.2.: Results MVS

4.2.3. NSR

In the NSR category we used the Neus-faco and VolSDF algorithm to reconstruct the models. These algorithms produce meshes that enable us to extract features. For every model we first do a visual inspection of the reconstruction to see which object is reconstructed better. Next we evaluate the reconstructions using the full mesh for both the reconstruction and the ground truth model. This is the traditional way of evaluating and serves as a baseline. For the wire models we tested six configurations. Three wire models where only the adjacent facets to the detected sharp edges were selected, these are marked as small models. The three wire models where the two ring neighbours around the adjacent facets were also selected are marked as the large models. For both categories we tested with an angle threshold of 25°, 40° and 60°. The analysis per model can be found in Appendix A. Table 4.3 shows the percentage where the quality prediction of the metric is inline with the visual observation aggregated over all the results.

		Hausdorff	chamfer	F-score	combined
large	25	50%	50%	40%	46.7%
	40	50%	70%	30%	50%
	60	50%	70%	40%	53.3%
small	25	60%	60%	40%	53.3%
	40	60%	70%	40%	56.7%
	60	50%	70%	50%	56.7%
	full	50%	40%	50%	46.7%

Table 4.3.: Percentage where metric corresponds with visual analysis

5. Conclusion

Due to the complexity of creating a high detailed ground truth model, through laser scanning or manually modelling, of large urban objects, there is no way to objectively measure the performance of the passive reconstruction algorithms with the current metrics. In this thesis we wanted to investigate if it is possible to create a methodology to quantify the performance of passive reconstruction algorithms for large urban objects with a low detailed but complete ground truth model.

The idea was based on the visual observation that, if an algorithm is able to reconstruct the finer details, the larger features of the building are also reconstructed better. Unfortunately there are no datasets available consisting of large urban objects with a complete ground truth model. Only partially scanned buildings where large parts are missing are currently publicly available.

Therefore, the first question we had to answer is: can the algorithms reconstruct a large urban object based on a low detailed complete ground truth model. To answer this question we needed to create a new dataset. Out of the box the low detailed models were not suitable to be used in the algorithms. With the right settings of the parameters and hyperparameters it is possible to reconstruct large urban objects with algorithms from MVS, NVS and NSR from low detailed models.

With these outcomes we were now able to measure the performance of the NSR algorithms based on feature extraction from the generated meshes. We experimented with three different metrics (Hausdorff distance, standard chamfer distance, and F-score) and six combinations of wire meshes. We found that, based on our ten models, the standard chamfer distance with a wire mesh with an angle between 40 and 60 degrees for both small and large gives a better prediction of the quality of the algorithms than the traditional evaluation on the full mesh, based on our visual analysis. In table 4.3 we can see that the traditional standard chamfer distance based on the full mesh, only 40% of the time was inline with our visual analysis. Whereas the standard chamfer distance based on the feature extraction was for 70% inline with our visual analysis.

Measuring the performance of mesh reconstruction algorithms based on feature extraction of the ten models has shown potential for the creation of a benchmark for large urban objects generated from low detailed complete ground truth models. For future research we are interested if a benchmark can be created where a large number of low detailed complete ground truth models are scored with feature extraction.

A. Evaluation per model for NSR

For every model we show the ground truth model, the reconstruction of the NeuS-facto, the reconstruction of VolSDF, the quantitive evaluation for the different wire-frames and full mesh. With a visual inspection we indicate which reconstruction is better. In the table we indicate for every metric and evaluation mesh which algorithm is scoring better with a green colour.

A.1. Apartment

For the visual inspection of the apartment model, the reconstructions are shown in Figure A.1). For this object we choose the VolSDF object because the general shape of the object is better modelled. The corners of the building are also better rendered. Even though the details in the NeuS-facto renderings are sharper, there is too much noise in the reconstruction. The results of the metrics are shown in Table A.1.

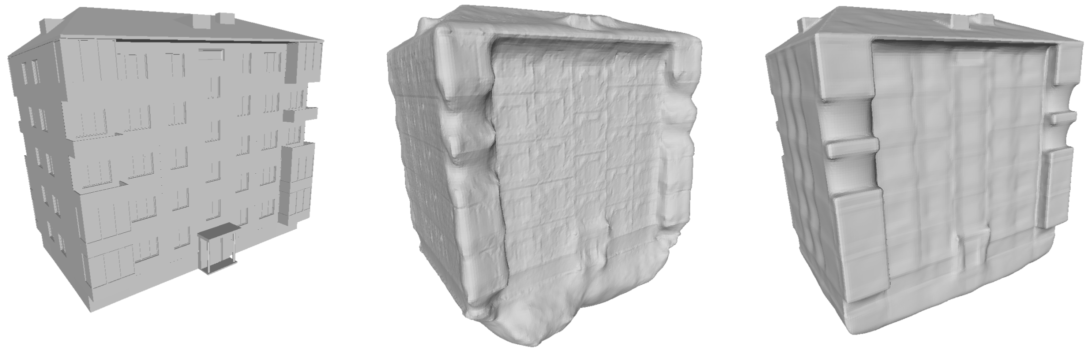


Figure A.1.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the apartment model

A.2. EWI

The reconstructions are shown in Figure A.2). We chose the NeuS-facto model for the visual inspection because the sides of the object are better modelled, the stepped design of the building is clearly visible here, while with VolSDF this has become a

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
	25	4.750	0.454	0.562	4.764	0.606	0.402
large	40	4.750	0.858	1.149	4.805	0.706	0.414
	60	5.980	1.580	0.623	6.759	1.485	0.386
	25	4.672	0.512	0.533	4.864	0.631	0.397
small	40	4.672	0.896	0.535	4.873	0.734	0.432
	60	6.003	1.650	0.625	6.818	1.519	0.387
	full	6.043	0.496	0.602	5.259	0.383	0.582

Table A.1.: Results apartment for NSR

large curve. In addition, more detail is shown in the windows in the facade. The quantified results are shown in Table A.2.

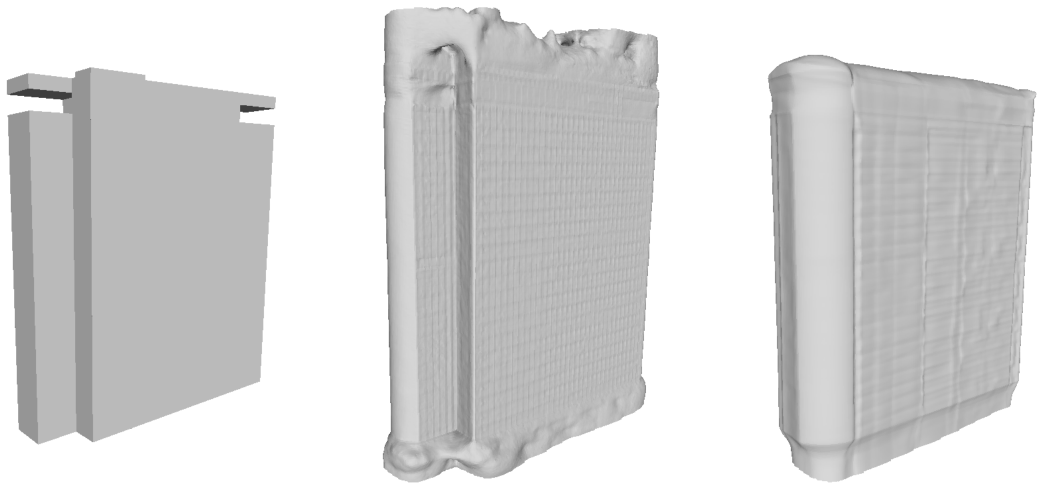


Figure A.2.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the Ewi model

A.3. Flat

The reconstructions are shown in Figure A.3).We choose the NeuS-facto result for the visual inspection because the balconies on the side of the building are better modelled. This shows that there are raised edges on the balcony. The various windows have also been modelled in higher detail. The quantified results are shown in Table A.3.

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
	25	22.570	5.097	0.188	22.135	5.047	0.457
large	40	20.931	6.871	0.240	35.061	8.986	0.402
	60	27.048	9.561	0.234	39.463	13.761	0.084
	25	22.998	4.997	0.185	22.579	5.556	0.473
small	40	20.980	6.814	0.257	35.293	10.165	0.435
	60	27.277	9.774	0.233	39.870	14.245	0.049
	full	17.407	1.801	0.397	11.907	1.159	0.541

Table A.2.: Results Ewi for NSR

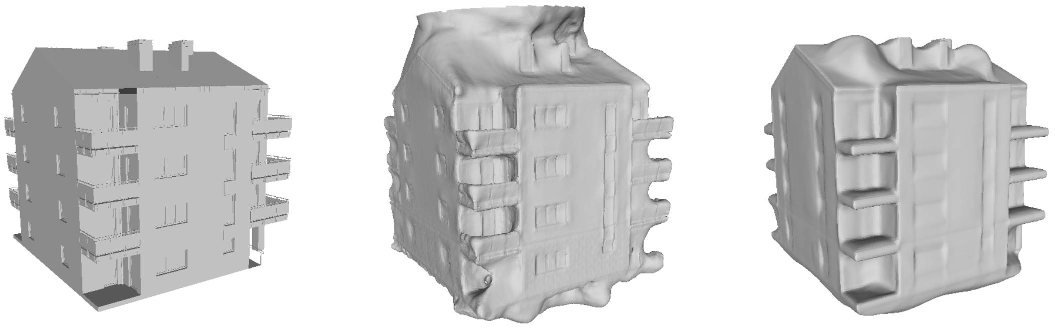


Figure A.3.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the flat model

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
	25	6.154	0.501	0.563	6.975	0.871	0.552
large	40	5.919	0.703	0.557	6.975	1.476	0.585
	60	5.724	1.362	0.565	10.400	3.462	0.666
	25	6.134	0.504	0.536	6.921	0.905	0.546
small	40	5.908	0.743	0.545	6.887	1.520	0.593
	60	5.645	1.436	0.574	10.453	2.111	0.624
	full	6.239	0.613	0.602	7.019	0.625	0.528

Table A.3.: Results flat for NSR

A.4. House

The reconsturction of the House model can be seen in Figure A.4. The house that is visually better is the VolSDF model because the NeuS-facto model has a lot of noise around the object. The VolSDF object has slightly less detailed objects in the facades than that of NeuS-facto. But overall, VolSDF's result looks better. The quantified results are shown in Table A.4.

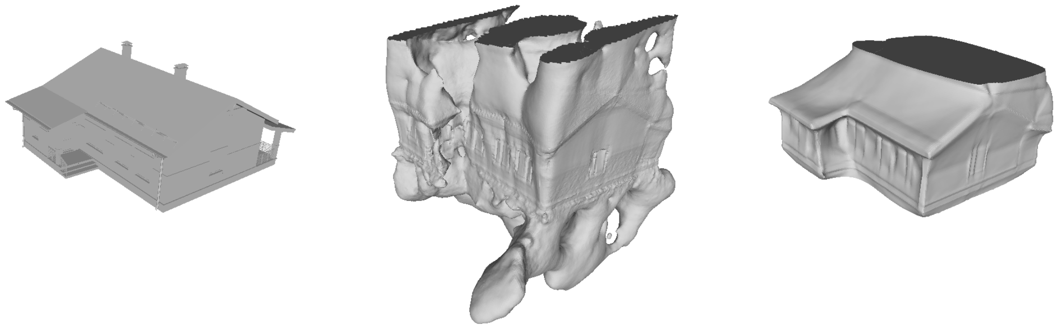


Figure A.4.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the house model

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
large	25	3.421	0.507	0.421	4.539	0.544	0.420
	40	3.421	0.662	0.429	4.565	0.923	0.460
	60	3.421	1.026	0.449	4.565	1.318	0.445
small	25	3.532	0.550	0.414	4.553	0.581	0.425
	40	3.532	0.550	0.414	4.590	0.999	0.461
	60	4.553	0.581	0.425	4.590	1.380	0.449
	full	3.421	0.459	0.383	4.740	0.507	0.285

Table A.4.: Results house for NSR

A.5. Neo

The reconstruction of the Neo model can be found in Figure A.5. The NeuS-facto results looks better than that of VolSDF because the details are better modelled. With VolSDF the details are more smoothly detailed, while with NeuS-facto the details are modelled sharper. The quantified results are shown in Table A.5.

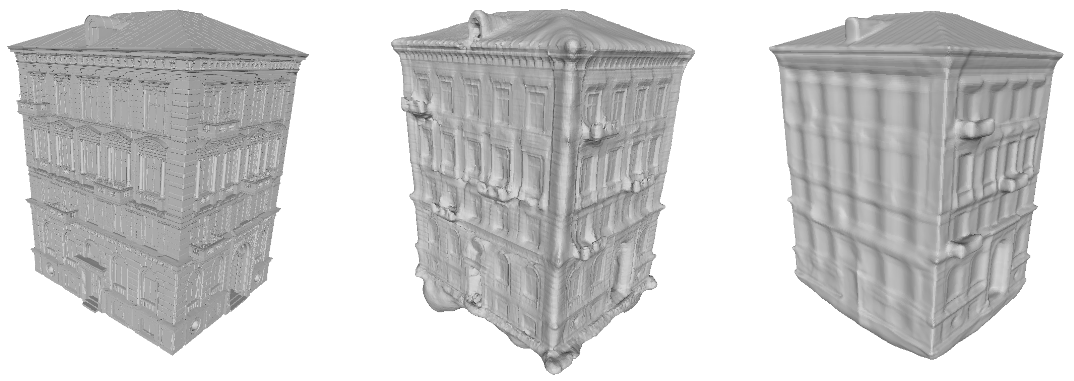


Figure A.5.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the Neo model

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
	25	5.176	0.377	0.433	6.971	0.820	0.483
large	40	5.072	0.542	0.402	8.639	1.414	0.506
	60	4.878	0.834	0.377	14.554	3.013	0.423
	25	5.076	0.398	0.432	7.061	0.889	0.546
small	40	4.973	0.574	0.400	8.732	1.490	0.567
	60	4.812	0.887	0.373	14.638	3.037	0.371
	full	6.188	0.553	0.484	5.677	0.437	0.439

Table A.5.: Results Neo for NSR

A.6. Speerhall

The reconstruction is shown in Figure A.6. We chose the NeuS-facto model because this model represents the details much better than the VolSDF model. The details are sharper. With the full SDF model the details are very smooth. The quantified results are shown in Table A.6.

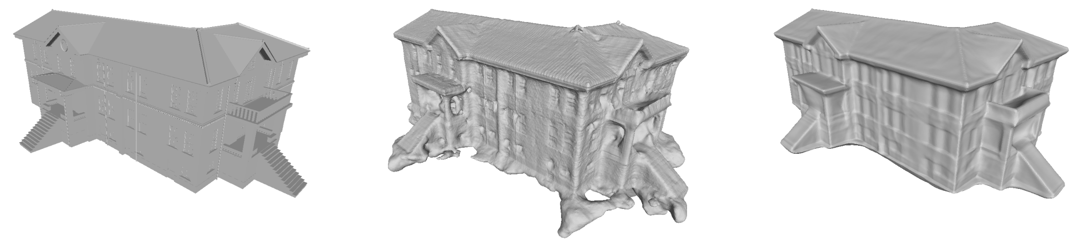


Figure A.6.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the Speerhall model

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
large	25	6.612	0.415	0.543	5.849	0.364	0.433
	40	6.612	0.519	0.556	8.060	1.174	0.372
	60	6.612	0.782	0.544	13.265	2.866	0.193
small	25	6.481	0.439	0.522	5.955	0.400	0.436
	40	6.481	0.561	0.512	8.195	1.216	0.366
	60	6.481	0.814	0.513	13.419	2.867	0.134
	full	7.188	0.751	0.580	7.443	0.434	0.511

Table A.6.: Results Speerhall for NSR

A.7. Stoommachine

The reconstructed meshes are displayed in Figure A.7. For Stoommachine it is a difficult choice between NeuS-facto and VolSDF because the shapes of both objects are well rendered. The difference is mainly in the modelled details, where the NeuS-facto object is better detailed than the VolSDF object. That is why we determined the NeuS-facto object is better modelled. The quantified results are shown in Table A.7.

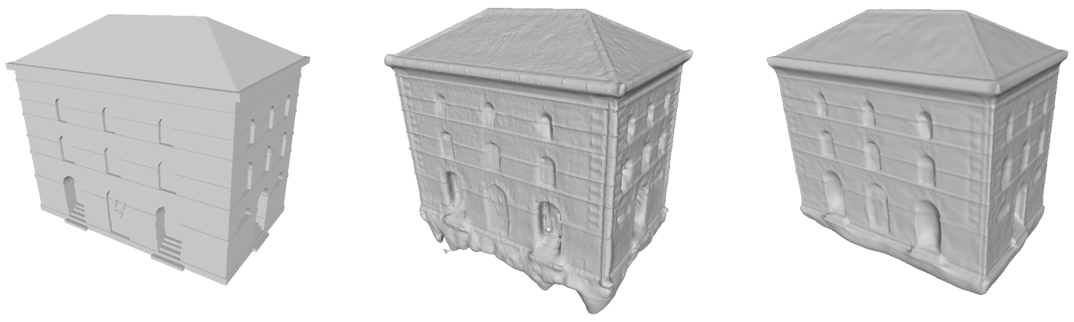


Figure A.7.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the Stoommachine model

A.8. Stieltjesweg

For Stieltjesweg the reconstructions are shown in Figure A.8. The NeuS-facto object is better modelled because the overall shape of the object is better and more details have been modelled. There is some extra noise on the left side of the object, but this

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
	25	4.676	0.473	0.538	5.907	0.421	0.449
large	40	4.676	0.680	0.564	6.940	1.081	0.414
	60	4.847	1.057	0.695	13.279	2.624	0.542
	25	4.676	0.533	0.515	5.997	0.460	0.465
small	40	4.676	0.758	0.577	7.032	1.106	0.531
	60	4.927	1.169	0.699	13.321	2.628	0.636
	full	4.676	0.427	0.581	6.287	0.409	0.536

Table A.7.: Results Stoommachine for NSR

does not outweigh the details and the shape. The quantified results are shown in Table A.8.

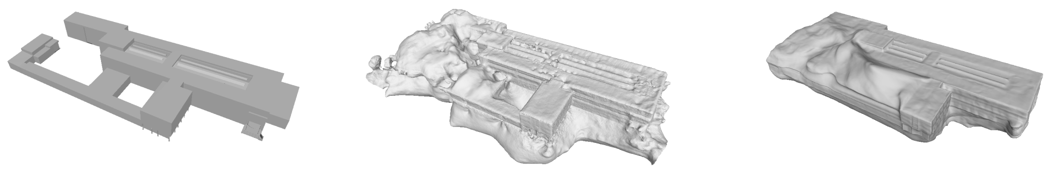


Figure A.8.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the Stieltjesweg model

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
	25	28.199	2.569	0.379	19.719	2.809	0.475
large	40	37.181	3.789	0.391	22.797	4.304	0.475
	60	37.181	6.031	0.353	34.320	7.918	0.373
	25	28.428	2.688	0.386	19.765	2.888	0.499
small	40	37.181	3.970	0.387	22.955	4.334	0.491
	60	37.181	6.648	0.347	34.317	8.004	0.397
	full	26.902	5.622	0.232	16.533	4.498	0.265

Table A.8.: Results Stieltjesweg for NSR

A.9. Villa

The reconstruction for the villa model can be found in Figure A.9. For the visual comparison of the villa model, we find the result of the VolSDF model is better since the NeuS-Facto model contains more noise than the VolSDF model. The quantified results are shown in Table A.9.

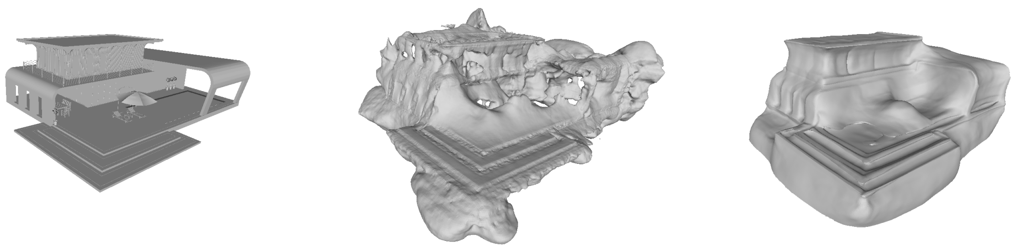


Figure A.9.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the villa model

		neusfacto			volsdf		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
large	25	9.811	0.838	0.274	5.262	1.353	0.412
	40	9.667	1.076	0.284	5.362	1.507	0.416
	60	9.667	1.293	0.338	7.179	2.214	0.571
small	25	9.811	0.886	0.257	5.321	1.449	0.424
	40	9.759	1.176	0.270	5.456	1.609	0.425
	60	9.741	1.418	0.273	7.030	2.332	0.575
	full	9.811	0.928	0.177	5.063	1.170	0.218

Table A.9.: Results villa for NSR, results pending

A.10. Winchester cathedral

For the visual comparison, the reconstructions are shown in Figure A.10, we determined the NeuS-facto result is better because the details of the object are better modelled. There is more noise around the object, but the complex details such as the pillars are modelled more sharply. The quantified results are shown in Table A.10.

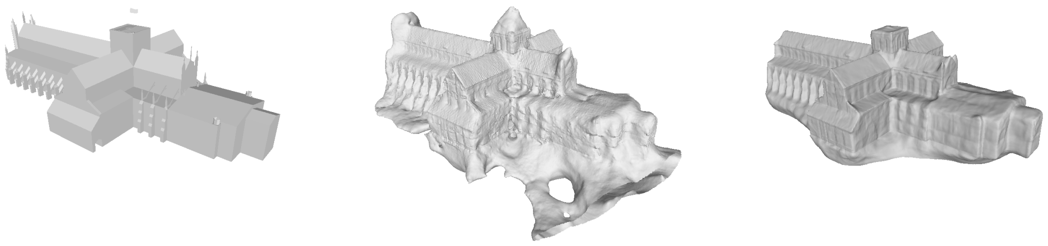


Figure A.10.: Left ground truth mesh, middle NeuS-facto output, right VolSDF output for the Winchester cathedral model

		NeuS-facto			VolSDF		
		Hausdorff↓	chamfer↓	F-score↑	Hausdorff↓	chamfer↓	F-score↑
large	25	32.283	2.562	0.298	15.960	4.475	0.286
	40	32.283	3.750	0.358	22.169	7.698	0.367
	60	32.283	6.823	0.417	32.077	13.041	0.410
small	25	32.012	2.835	0.304	15.960	4.711	0.285
	40	32.012	4.178	0.364	22.321	8.299	0.327
	60	32.012	7.696	0.384	32.442	13.559	0.406
	full	34.471	5.451	0.287	15.960	2.678	0.275

Table A.10.: Results Winchester cathedral for NSR

Bibliography

Aanæs, H., Jensen, R. R., Vogiatzis, G., Tola, E., and Dahl, A. B. (2016). Large-Scale Data for Multiple-View Stereopsis. *International Journal of Computer Vision*, 120(2):153–168.

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. (2021). Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844. ISSN: 2380-7504.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2023). Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. pages 19697–19705.

Farshian, A., Götz, M., Cavallaro, G., Debus, C., Nießner, M., Benediktsson, J. A., and Streit, A. (2023). Deep-Learning-Based 3-D Surface Reconstruction—A Survey. *Proceedings of the IEEE*, 111(11):1464–1501. Conference Name: Proceedings of the IEEE.

Furukawa, Y. and Ponce, J. (2005). Carved visual hulls for high-accuracy image-based modeling. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, pages 146–es, New York, NY, USA. Association for Computing Machinery.

Furukawa, Y. and Ponce, J. (2010). Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., and Tan, P. (2020). Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2492–2501. ISSN: 2575-7075.

Huang, J., Gojcic, Z., Atzmon, M., Litany, O., Fidler, S., and Williams, F. (2023). Neural Kernel Surface Reconstruction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4369–4379. ISSN: 2575-7075.

Ibrahimli, N., Ledoux, H., Kooij, J. F. P., and Nan, L. (2023). DDL-MVS: Depth Discontinuity Learning for Multi-View Stereo Networks. *Remote Sensing*, 15(12):2970. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

Kerbl, B., Kopanas, G., Leimkuehler, T., and Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14.

Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):78:1–78:13.

Li, Z., Müller, T., Evans, A., Taylor, R. H., Unberath, M., Liu, M.-Y., and Lin, C.-H. (2023). Neuralangelo: High-Fidelity Neural Surface Reconstruction. arXiv:2306.03092 [cs].

Lin, F., Yue, Y., Hou, S., Yu, X., Xu, Y., Yamada, K. D., and Zhang, Z. (2023). Hyperbolic Chamfer Distance for Point Cloud Completion. pages 14595–14606.

Loriot, S., Rouxel-Labbé, M., Tournois, J., and Yaz, I. O. (2024). Polygon Mesh Processing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition.

Lu, C., Yin, F., Chen, X., Liu, W., Chen, T., Yu, G., and Fan, J. (2023). A Large-Scale Outdoor Multi-Modal Dataset and Benchmark for Novel View Synthesis and Implicit Scene Reconstruction. pages 7557–7567.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2021). NeRF: representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106.

Müller, M. U., Ekhtiari, N., Almeida, R. M., and Rieke, C. (2020). SUPER-RESOLUTION OF MULTISPECTRAL SATELLITE IMAGES USING CONVOLUTIONAL NEURAL NETWORKS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-1-2020:33–40. Conference Name: XXIV ISPRS Congress, Commission I (Volume V-1-2020) - 2020 edition Publisher: Copernicus GmbH.

Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15.

Pérez, J., Rojas, S., Zarzar, J., and Ghanem, B. (2023). *Enhancing Neural Rendering Methods with Image Augmentations*.

Rabby, A. S. A. and Zhang, C. (2024). BeyondPixels: A Comprehensive Review of the Evolution of Neural Radiance Fields. arXiv:2306.03000 [cs].

Schönberger, J. L., Zheng, E., Frahm, J.-M., and Pollefeys, M. (2016). Pixelwise View Selection for Unstructured Multi-View Stereo. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 501–518, Cham. Springer International Publishing.

Sinha, S. N., Mordohai, P., and Pollefeys, M. (2007). Multi-View Stereo via Graph Cuts on the Dual of an Adaptive Tetrahedral Mesh. pages 1–8. IEEE Computer Society.

Strecha, C., Fransens, R., and Van Gool, L. (2006). Combined Depth and Outlier Estimation in Multi-View Stereo. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2394–2401. ISSN: 1063-6919.

Strecha, C., von Hansen, W., Van Gool, L., Fua, P., and Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. ISSN: 1063-6919.

Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., Mcallister, D., Kerr, J., and Kanazawa, A. (2023). Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, pages 1–12, New York, NY, USA. Association for Computing Machinery.

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Xu, Z., Simon, T., Nießner, M., Tretschk, E., Liu, L., Mildenhall, B., Srinivasan, P., Pandey, R., Orts-Escolano, S., Fanello, S., Guo, M., Wetzstein, G., Zhu, J.-Y., Theobalt, C., Agrawala, M., Goldman, D. B., and Zollhöfer, M. (2021). Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, SIGGRAPH '21, pages 1–320, New York, NY, USA. Association for Computing Machinery.

The CGAL Project (2024). *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition.

Wang, F., Galliani, S., Vogel, C., Speciale, P., and Pollefeys, M. (2021a). Patchmatch-Net: Learned Multi-View Patchmatch Stereo. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14189–14198. ISSN: 2575-7075.

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021b). NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. S., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27171–27183. Curran Associates, Inc.

Wang, X., Wang, C., Liu, B., Zhou, X., Zhang, L., Zheng, J., and Bai, X. (2021c). Multi-view stereo in the Deep Learning Era: A comprehensive review. *Displays*, 70:102102.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612. Conference Name: IEEE Transactions on Image Processing.

Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. (2018). MVSNet: Depth Inference for Unstructured Multi-view Stereo. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 785–801, Cham. Springer International Publishing.

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021). Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Yu, Z., Chen, A., Antic, B., Peng, S., Bhattacharyya, A., Niemeyer, M., Tang, S., Sattler, T., and Geiger, A. (2022). SDFStudio: A Unified Framework for Surface Reconstruction.

Colophon

This document was typeset using $\LaTeX$, using the KOMA-Script class `scrbook`. The main font is Palatino.