Creating Digital Surface Models from satellite imagery using Neural Radiance Fields (NeRF)

Master's Thesis – February 2025

Author

J. Smeets

Supervisors

ir. E. Verbree

ir. R. Voûte

Responsible Professor

Prof.dr.ir. P.J.M. van Oosterom

University

Delft University of Technology

Acknowledgments

There are many people that I would like to thank for their help during the thesis. Firstly, I want to thank Edward Verbree and Robert Voûte, my supervisors from TU Delft and CGI. Without their guidance and supervision, this thesis would not have been possible. Additionally, I would like to thank Peter van Oosterom for his supervision and feedback as a supervising professor. I need to mention Maarten Zeylmans van Emmichhoven who sparked my interest in DSM generation and Neural Radiance Fields. I also want to thank Reinier van Kleef, Steven van Ree and Marijn Brandwijk, who advised and supported me during the office days at CGI. Furthermore, I thank CGI for the opportunity to do my thesis at their company, I enjoyed working in a professional environment and felt welcome from the first day, especially thanks to Robert Voûte. Lastly, I want to thank Tessa van der Wel who supported me during the whole thesis. Without all of you, this thesis would not have been possible and for that I am grateful.

Abstract

Neural Radiance Fields (NeRFs) are increasingly used to generate Digital Surface Models (DSMs) from satellite imagery. However, research is far from mature and questions considering the accuracy in different environments, sensitivity and applicability have not been addressed. This research aims to answer these questions by firstly conducting a thorough literature research, resulting in the selection of a suitable NeRF model called SAT-NGP. The SAT-NGP model is tested on accuracy and runtimes in both urban and rural environments using the Data Fusion Contest 2019 (DFC2019) dataset, both prepared and unprepared. Furthermore, sensitivity analysis on the number of iterations and the ray batch size is conducted. The sensitivity analysis is followed by an applicability research considering the implementation of Superview-1 data of the Netherlands.

This research has found that SAT-NGP, using the DFC2019 dataset, creates more accurate DSMs for urban areas compared to rural areas. Additionally, challenging situations are identified where NeRF produces DSM accuracy errors. These challenging situations include edges of buildings, shadows, flat surfaces and trees. The sensitivity analysis findings indicate that a lower number of iterations can be used with minimal to no effect on the accuracy, reducing runtime significantly by 50%. Altering the ray batch size yielded no improvements concerning accuracy or runtime. Furthermore, the applicability analysis results show that it is complicated to apply SAT-NGP to data different from DFC2019. This is mainly due to the numerous input files that are needed, which all have to be perfectly adjusted and aligned to each other. Through the different analyses and literature research, the findings of this study contribute to the DSM generating NeRF research field. The results show that NeRFs are a promising development in the field of DSM construction from satellite imagery, but many improvements can still be made concerning accuracy methods, sensitivity research and applicability in general.

Contents

List of abbreviations

1. AOI Area Of Interest
2. AHN Algemeen Hoogtebestand Nederland
3. BRDF Bidirectional Reflectance Distribution Function
4. BRDF-NeRF Bidirectional Reflectance Distribution Function Neural Radiance Fields
5. DFC2019 Data Fusion Contest 2019
6. DEM Digital Elevation Model
7. DISK Deep Iterative Subspace Keying
8. DSM Digital Surface Model
9. DTM Digital Terrain Model
10. EO-NeRF Multi-Date Observation Neural Radiance Fields
11. GC-NeRF Geometric Constrained Neural Radiance Fields
12. GT Ground Truth
13. GIS Geographical Information System
14. GPS Global Positioning System
15. LiDAR Light Detection And Ranging
16. Max E Maximum Error
17. MAE Mean Absolute Error
18. ME Mean Error
19. MLP Multi-Layer Perceptron
20. MVS Multi-View Stereo
21. NeRF Neural Radiance Fields
22. RPC Rational Polynomial Camera model
23. RPV Rahman-Pinty-Verstraete reflectance model
24. RS-NeRF Remote Sensing Neural Radiance Fields
25. S-NeRF Shadow Neural Radiance Fields
26. Sat-NeRF Satellite Neural Radiance Fields
27. SAT-NGP Satellite Neural Graphics Primitives
28. SC Solar Correction
29. SfM Structure from motion
30. SGM Semi-Global Matching
31. SIFT Scale-Invariant Feature Transform
32. SpS-NeRF SparseSat Neural Radiance Fields
33. SRTM Shuttle Radar Topography Mission
34. UTM Universal Transverse Mercator coordinate system

1 Introduction

Terabytes of satellite imagery of the Earth are captured each day. These images can be used to extract the elevation of the Earth's surface, producing digital surface models (DSMs) (Derksen & Izzo, 2021). Especially high-resolution satellite images provide a valuable resource for producing accurate 3D Digital Surface Models (DSMs) (Lastilla et al., 2021). The DSMs created can be used for a multitude of applications, including flood risk analysis (McClean et al., 2020), mangrove forest height and biomass mapping (Simard et al., 2006), land cover classification, and 3D change detection (Demarez et al., 2019; R. Qin et al., 2016). Originally, the standard was to create DSMs from images using semi-global dense image matching (Deseilligny and Paparoditis, 2006; Hirschmuller, 2005) (SGM). In addition to SGM, deep learning methods such as hybrid and end-to-end techniques are applied (Chang and Chen, 2018; Hartmann et al., 2017), or alternatively a depth fusion step (Rupnik et al., 2018). Other research has focused on feature extraction and matching using Deep Iterative Subspace Keying (DISK) and LightGlue (Claesen, 2024). Recently, a new method for accurate 3D reconstruction has been developed, called Neural Radiance Fields (NeRF). The method uses input imagery from multiple directions and a neural network to produce accurate scene reconstructions with corresponding depth maps (Mildenhall et al., 2020). After publication, NeRF research sparked a boom of related research adjusting NeRF to a multitude of applications. One of these applications is the field of DSM generation from multi-view satellite imagery.

This research explores the different NeRF variants that have been specifically developed for DSM generation of satellite imagery and uses one of the NeRF models to further test its applicability to different types of research areas. To achieve these objectives, first, a comprehensive literature review of the developed NeRF models is performed.

NeRF has seen little application to DSM generation of environments with increased vegetation, such as a rural environment (L. Zhang and Rupnik, 2023; L. Zhang et al., 2024). In this research, a rural environment is considered a suburb with increased vegetation compared to urban environments. To investigate the performance of NeRF in a rural environment, this research explores the performance of NeRF in such a setting. Little research has been conducted on the optimization of NeRF models that create DSMs from satellite imagery. Marí et al. (2023) have adjusted the number of hidden layers in the MLP network and the number of points sampled for each ray (these factors are further explained in Sections 2.1 and 2.6), but many other parameters remain untested. Additionally, different NeRF models are often run with different numbers of iterations without testing the number of iterations for a specific model. This research tests the influence of the number of iterations and batch size on the accuracy and runtime of a single state-of-the-art NeRF model.

Currently, NeRF models developed for DSM generation require many perfectly adjusted inputs, resulting in difficulties when applying the NeRFs to different datasets. This limits the usability and implementation of NeRF. Therefore, this research will aim to clarify the specific needs of NeRF models and describe how a new dataset can be adjusted and used as input, exploring the applicability of DSM generating NeRFs.

1.1 Problem Statement

Currently, state-of-the-art algorithms and methods for Multi-View Stereo (MVS) and Semi-Global Matching (SGM), both explained in Section 2.3, struggle with several issues. These issues include DSM reconstruction of reflective surfaces, complex topological surfaces, and areas with little texture. The difficulties mentioned often result in unwanted errors (Stathopoulou et al., 2023). In order to mitigate the errors, multiple learning-based methods are applied to address the issues, including feature extraction (Zagoruyko & Komodakis, 2015), multi-view stereopsis (Riegler et al., 2017) and depth fusion (Huang et al., 2018). Other research has focused on improving urban 3D models using multiple datasets and an implicit neural representation (Li, 2024). However, all of the methods only address part of the problem, showing the need for a different, more complete method, such as Neural Radiance Fields (NeRF) (Mildenhall et al., 2020).

Another problem includes that a modern day MVS pipeline requires control over many stages. All of these stages can influence and introduce errors in the final result. Figure 1 shows how a NeRF model (EO-NeRF in this case) eliminates many of the stages of classic MVS, reducing the risk of manual errors. However, it should be noted that a NeRF model requires some pre-processing as opposed to what is shown in Figure 1. This pre-processing includes clipping the data to the research area and adjusting camera reports, for example. Additionally, NeRFs require training a model for each specific scene, which can take much time depending on the hardware available (Derksen and Izzo, 2021; Marí et al., 2022).

State-of-the-art NeRF models, such as EO-NeRF are already outperforming the state-of-the-art classic MVS methods, such as S2P, in terms of vertical accuracy (de Franchis et al., 2014; Marí et al., 2023). Furthermore, NeRF has only been around for a few years, while classic methods are in a more mature research stage (Mildenhall et al., 2020). Because NeRF is still relatively young, many developments can be made. For example, NeRF opened the door for multiple computer vision methods to be integrated, resulting in more accurate results (Billouard et al., 2024; Marí et al., 2022). The application of computer vision methods can solve some of the problems that occur in classic DSM generation techniques mentioned earlier. Currently, vertical accuracies of less than a meter can be achieved from high-resolution pansharpened RGB satellite images with a resolution of 30cm (Marí et al., 2023). These high accuracies are a result of the robustness of NeRF to multiple factors, including lighting changes, transient objects, occlusions and noisy data. The usage of neural radiance fields shows great promise, and it is important to continue research on the topic. Especially concerning accuracy difficulties, sensitivity analysis and applicability. This research aims to identify current difficulties, test sensitivity to certain parameters and describe what is needed in the near future, to improve the accuracy and applicability of NeRF.

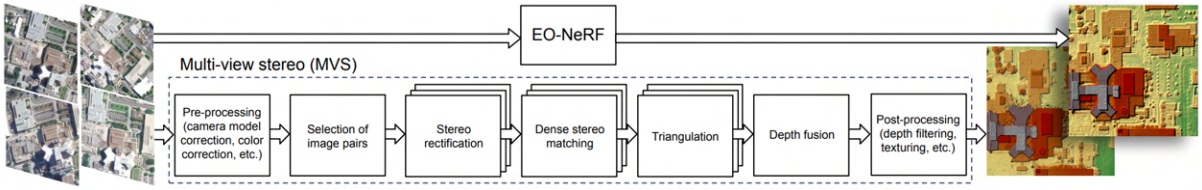


Figure 1: EO-NeRF and NeRF in general eliminate many of the stages that require manual control in modern day MVS pipelines (Marí et al., 2023).

To provide an overview, the advantages of NeRF over classic DSM generation methods are listed below.

1. NeRF eliminates many of the stages involved in classic MVS pipelines.
2. NeRF opens the door for computer vision methods.
3. NeRF is better at handling complex shapes.
4. NeRF produces more smooth and continuous models.
5. NeRF is better at handling different lighting conditions.
6. NeRF is still young and many improvements can be made more easily.

Multiple NeRF models are specifically adjusted to create DSMs from satellite imagery (Billouard et al., 2024; Derksen and Izzo, 2021; Marí et al., 2022; Marí et al., 2023; Wan et al., 2024; Xie et al., 2023; L. Zhang and Rupnik, 2023; T. Zhang et al., 2024; L. Zhang et al., 2024). Their quality is improving every year as the methods become more sophisticated. Furthermore, models are becoming more efficient and computation times are reduced as is the case with SpS-NeRF (L. Zhang and Rupnik, 2023), RS-NeRF (Xie et al., 2023), SAT-NGP (Billouard et al., 2024) and GC-NeRF (Wan et al., 2024).

Almost all models are trained and tested on four specific scenes of the Data Fusion Contest 2019 (DFC2019) dataset, which comprises high-resolution (30cm) pansharpened RGB WorldView-3 satellite imagery of Jacksonville, Florida (Le Saux et al., 2019). SpS-NeRF and BRDF-NeRF are the only variants of NeRF that have been tested on a non-urban environment, comprising vegetation and bare surface (L. Zhang and Rupnik, 2023; L. Zhang et al., 2024). Unfortunately, no ground-truth LiDAR data was available, and the results were compared to a DSM derived from state-of-the-art photogrammetry. The lack of application of NeRF models that use satellite imagery to derive DSMs of environments different from the four specific DFC2019 scenes, shows that there is a research gap. Since all of the models are tested on areas of interest (AOIs) 004, 068, 214, and 260, a bias can be introduced when evaluating the performance of the NeRF models. DSM generating NeRF models are now developed mostly based on the results in these AOIs. Therefore, it is also vital to test the performance on other AOIs. In addition, little research has focused on changing the input parameters and monitoring their effects on the vertical accuracy of the resulting DSM (Marí et al., 2023).

This thesis is executed in collaboration with CGI. The ICT consultancy company CGI also aims to create DSMs from satellite imagery. There is a broad market for DSMs, as detailed height maps are often unavailable for many regions. This provides a business opportunity, CGI can sell the DSMs to the army, where elevation maps are often required to aid various processes. Another application includes the delivery of DSMs to Kadaster, who are interested in having a yearly DSM of certain areas, instead of the Algemeen Hoogtebestand Nederland (AHN), which is developed every few years. The AHN will be discussed further in Section 2.5.1. In addition to aiding Kadaster, other clients can use the DSMs for viewshed analysis or natural hazard risk management (McClean et al., 2020).

Earlier research by Claesen, 2024 has made use of LightGlue and the DISK algorithm to enhance a classic photogrammetry pipeline. His research pointed out that there is still room for improvement considering height accuracy, especially in environments with increased vegetation. The methods and findings of his research are explained further in Section 2.6.1. Parallel to this research on NeRF, another research is conducted involving DSM correction using space-borne LiDAR to improve the shortcomings of the methods used by Claesen, 2024. These shortcomings include reduced feature matching in areas with sparse features resulting in decreased vertical accuracies. Since NeRF does not rely on feature matching, it might be able to produce higher vertical accuracies in these areas.

1.2 Research Objectives and limitations

1.2.1 Main research objective: Creating DSMs using NeRF

The objective of this research is to determine to what extent NeRF can be used to create DSMs from 2D high-resolution RGB satellite imagery of urban and rural environments. In this research, the term rural is used to describe a suburban environment with increased vegetation. An existing state-of-the-art NeRF model is tested on data comprising different environments. The main research objective leads to the following main research question:

- To what extent can a state-of-the-art Neural Radiance Field (NeRF) model be used to create Digital Surface Models (DSMs) of both urban and rural environments from high-resolution RGB satellite imagery?

To answer the main research question, multiple sub research questions are identified. Firstly, a literature research is conducted to identify the optimal state-of-the-art NeRF model for this research. In addition to that, the performance of the NeRF model is evaluated on urban and rural datasets. To determine the impact of the number of iterations and the ray batch size on the vertical accuracy and runtime, a sensitivity analysis is conducted. Lastly, the applicability of NeRF is analyzed regarding the use of a dataset that differs from the DFC2019 data.

1.2.2 Sub-research objectives: Model selection, Accuracy, Sensitivity and Applicability

The following sub-research questions are identified for this research.

1. What is the optimal NeRF model to use for this research, considering available code, computing time and accuracy?
2. What accuracy does NeRF achieve using high-resolution RGB satellite imagery of urban and rural environments?
   (a) What accuracy does NeRF achieve using high-resolution RGB satellite imagery of urban environments?
   (b) What accuracy does NeRF achieve using high-resolution RGB satellite imagery of rural environments?
3. What is the sensitivity of the NeRF model used considering the number of iterations and batch size, in both urban and rural environments?
4. What is the applicability of NeRF and what is needed to use NeRF on data different from DFC2019?

1.3 Research limitations

For this research, the development of a new, optimized NeRF model is out of scope. The development of a new model would be too time-consuming for a thesis project of only 5 months. Additionally, the performance of the NeRF method will not be tested on view synthesis, as the focus of this research lies on creating DSMs. View synthesis is the process of generating accurate novel views from different angles that have not been seen before. Lastly, only NeRF will be used to create DSMs and no other methods such as Gaussian splatting or S2P are explored (de Franchis et al., 2014; Kerbl et al., 2023). Although the methods that are needed to implement data different from DFC2019 are investigated, these methods are not implemented in this research due to time constraints.

1.4 Research methodology

Figure 2 is created to give insight in the structure of this research. The conceptual model shows that the research is divided in three phases. Firstly, the identification phase, starting with a literature research, leads to a research gap, problem statement and the research questions. The identification phase is followed by the execution phase in which results are produced for each sub research question. Lastly, in the evaluation phase, the results of each sub research question are evaluated and a conclusion is drawn, answering the main research question. In addition to the conceptual model, Figure 3 is created to provide an overview of the different steps and processes in this research.

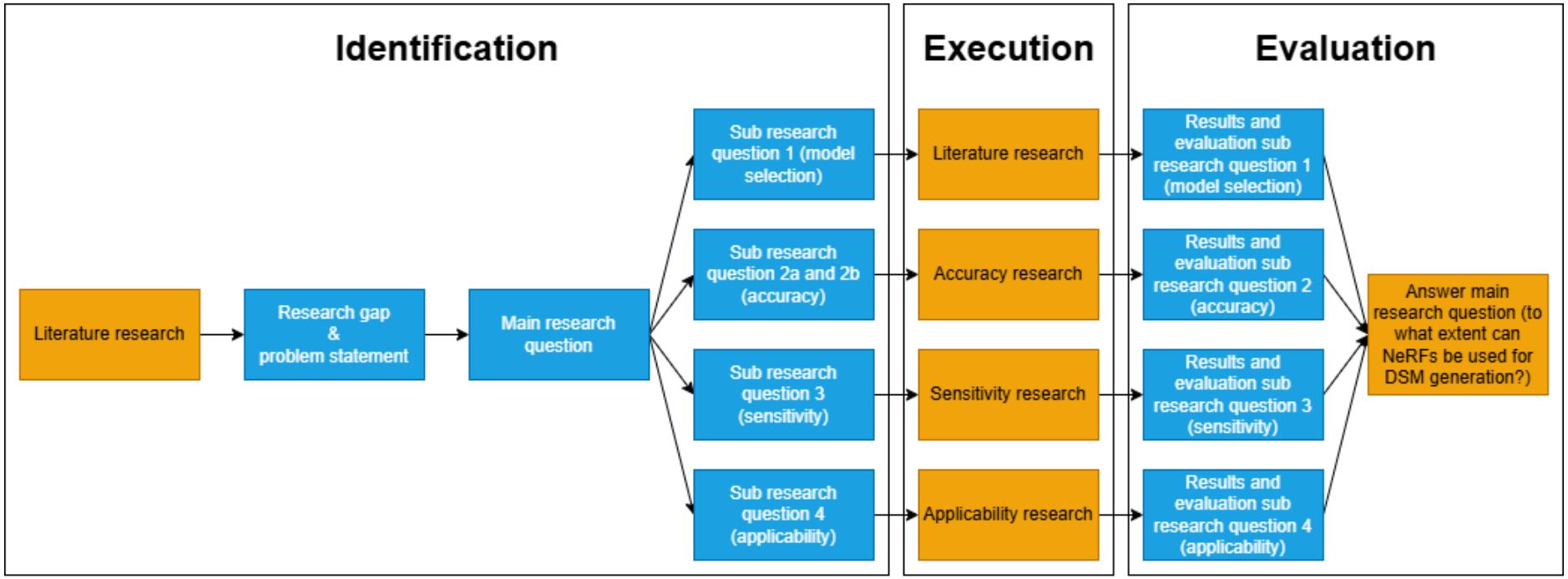


Figure 2: Conceptual model providing an overview of the structure of this research.

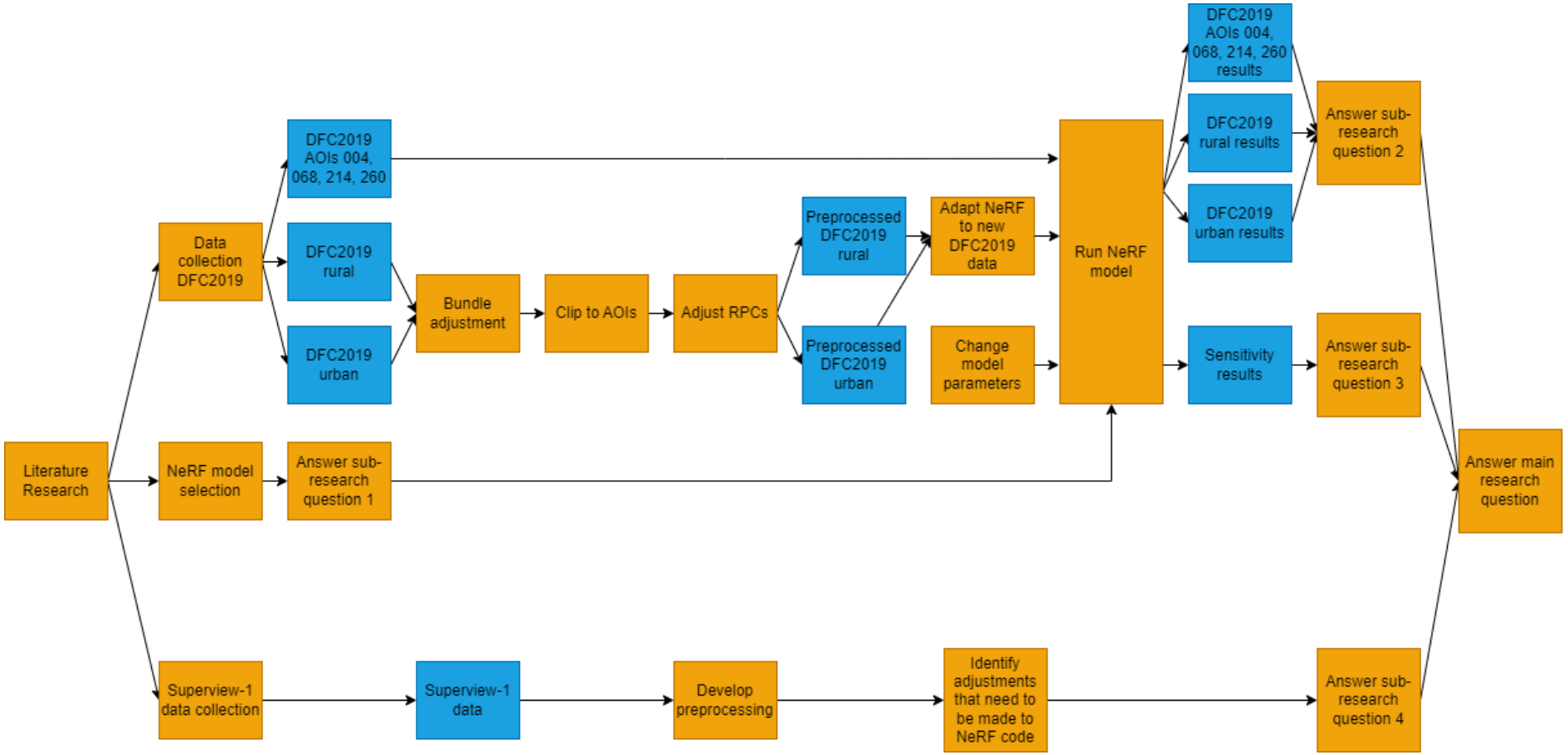


Figure 3: Flowchart providing an overview of the processes and steps that are involved in this research.

2 Related work

This section will first explain how Neural Radiance fields work, followed by background information for DEM reconstruction using satellite imagery. Basic concepts such as the differences between DEMs, DSMs and DTMs are discussed. Furthermore, the concepts of photogrammetry, structure-from-motion, bundle adjustment, multi-view stereo, rational polynomial camera models, LiDAR and the Actueel Hoogtebestand Nederland are covered. Lastly, each NeRF model that produces DSMs from satellite imagery is investigated separately.

2.1 Neural Radiance Fields

Recently, a new method to extract 3D from 2D imagery has been developed called Neural Radiance Fields (Mildenhall et al., 2020) (NeRF). It is based on the principle of learning a 3D object or scene as a continuous function F (Tewari et al., 2020). NeRF uses multiple views of a scene to learn a continuous volumetric representation (i.e., 3D radiance field). Unlike other methods, NeRF is able to include multiple aspects of the physical scene, such as illumination sources and surface radiance as observed in the input images (Mildenhall et al., 2020).

2.1.1 NeRF in a Nutshell

Due to the complicated processes and concepts that are used, NeRF can be difficult to understand. Therefore, this section aims to explain NeRF in a simplified manner.

The initial goal of NeRF is to render or reconstruct realistic 3D scenes from 2D images. The reconstruction allows the 3D scene to be viewed from every angle, even ones that are not covered by the 2D images. In addition to that, NeRF is able to incorporate view-dependent effects, including highlights and reflections. Simply put, the input for NeRF is 2D images that are taken at different angles. After that, rays are cast through the scene and sampled at a certain amount of points. From these rays, the neural network learns the 3D structure and view-dependent color using volume rendering and rendering loss, as shown in Figure 4 (Mildenhall et al., 2020). In this research, modified versions of the original NeRF model are used. These versions are able to extract the surface elevation from satellite imagery input using volume rendering.

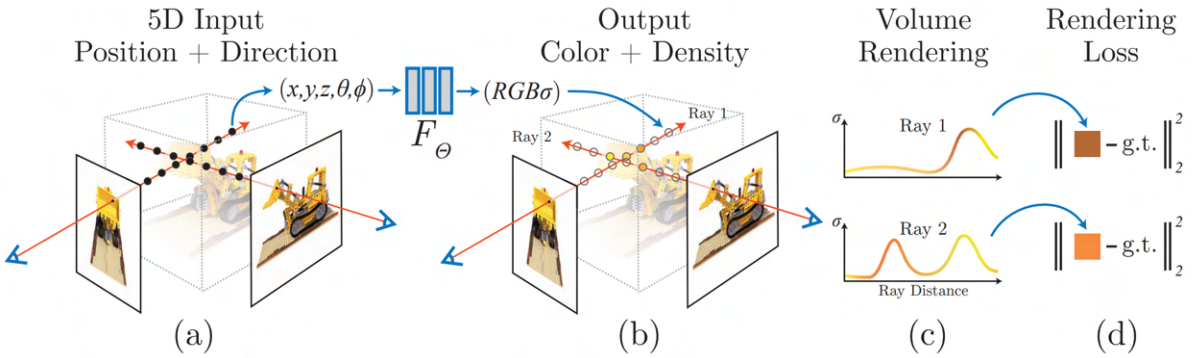


Figure 4: Overview of NeRF. Showing the 5D input (x, y, z, θ, ϕ) from ray sampling (a), the output of the MLP comprising color and volume density (θ, RGB) (b), volume rendering (c) and the rendering loss (d) (Mildenhall et al., 2020).

2.1.2 Positional encoding and Hierarchical sampling

NeRF uses a 5D function (F) to represent a static scene. The 5D function comprises the viewing direction (θ, ϕ) at each point in space (x, y, z). To generate the 5D input for the deep neural network (MultiLayer Perceptron (MLP)), rays are shot (marched) through the 2D images. Positional encoding is used to improve the performance of NeRF in high-frequency colour and geometry variation situations. It is used to map the 5D inputs for the MLP to a higher dimensional space by applying high frequency functions. By using the positional encoding, the MLP is able to better fit data with increased variability. After positional encoding, sampling is performed at a certain amount of points along each ray. A hierarchical sampling method is used to increase rendering efficiency (Mildenhall et al., 2020). In hierarchical sampling, a fine and a coarse network are optimized simultaneously, where the coarse network is used as a guide to sample increasingly at locations where the relevant parts of the volume are located, as shown in Figure 5 (Mildenhall et al., 2020).

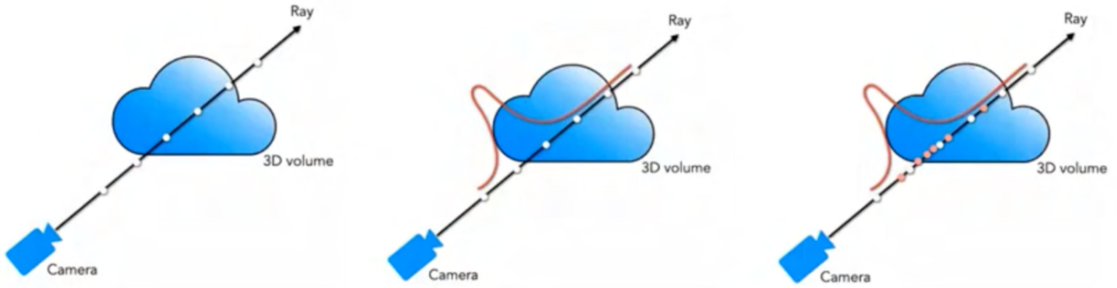


Figure 5: Visualization of the sampling process. The figure shows that coarse sampling is used to detect where sampling should be increased to optimize the finer sampling. On the left, the ray is coarsely sampled with only few points. The middle image shows that it detects where something is approximately located. Finally, the right image shows that the sampling is increased for the approximate location of an object (Kim, 2022).

2.1.3 MLP network

As mentioned before, a deep neural network (MLP) is used to learn the volume density (σ) and view dependent colour (RGB or $\mathbf{c}$) from approximating 5D input (x, y, z, θ, ϕ) of the scene. The MLP is made up of a number of layers that each contain multiple neurons, which are interconnected with all of the neurons from the other layers. For NeRF, the MLP network architecture is made up of 10 layers of which the first 8 intermediate 256-neuron layers have Rectified Linear Unit (ReLU) activation. ReLU activation outputs the input and otherwise 0. At the 5^{th} layer, the input location is concatenated to the layers activation. After the 8^{th} layer, a 9^{th} layer with 256 channels is added to the MLP using no activation. This layer outputs the volume density and a 256-dimensional feature vector. Additionally, the viewing direction is concatenated to the layer's activation. After that, another layer is added with 128 channels and ReLU activation. This layer outputs the RGB value at each position ($\mathbf{x}$) from a certain viewing direction ($\mathbf{d}$) using sigmoid activation. Sigmoid activation outputs a value between 0 and 1 (Mildenhall et al., 2020).

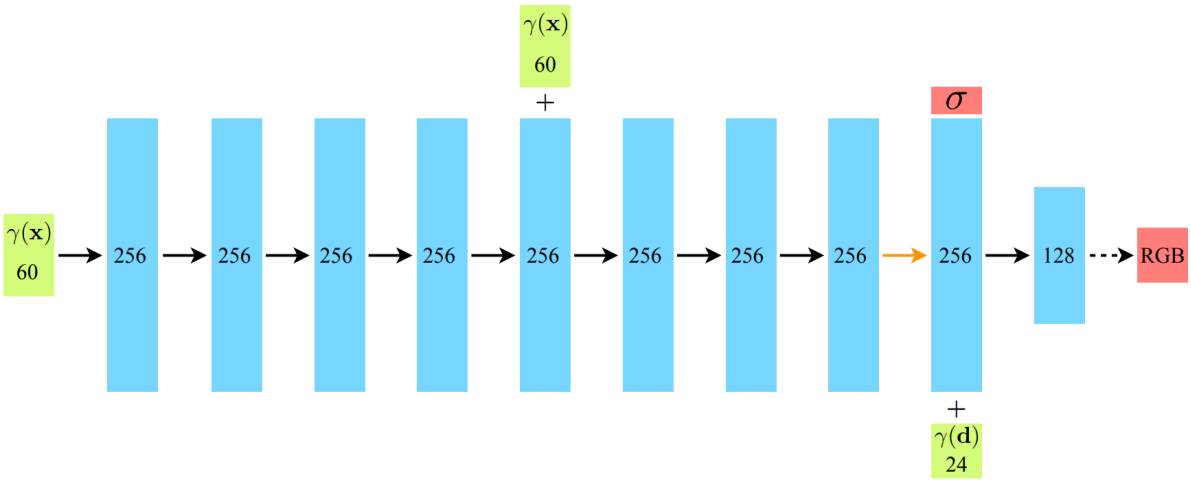


Figure 6: Architectural layout of the MLP network used for NeRF. Input vectors are shown in green, output vectors in red and the MLP layers in blue. The numbers specify the dimensionality of the vectors and the number of channels for the MLP layers. Additionally, black arrows indicate ReLU activations, red arrows no activations and dashed black arrows sigmoid avtivations. The + sign indicates where vectors are concatinated (Mildenhall et al., 2020).

2.1.4 Volume rendering

The output of the MLP is used for volume rendering. Equation (1) shows how the volume rendering is executed. The formula is build out of the following components:

- $C(\mathbf{r})$ The expected color of a camera ray ($\mathbf{r}$).
- $T(t)$ The amount of light that remains at that point. It can be interpreted as the probability that the ray travels from t_n to t_f without hitting another particle.
- $\sigma(\mathbf{r}(t))$ The volume density, which can be interpreted as the differential probability of a ray terminating at an infinitely small particle at location x.
- $\mathbf{c}(\mathbf{r}(\mathrm{t}))$ The color at a certain point from viewing direction $\mathbf{d}$.
- $\mathbf{d}$ The viewing direction (θ, ϕ).

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt, \quad \text{where} \quad T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right) \tag{1}$$

The volume rendering essentially takes the mean of all the points in a ray. To render a certain view, the integral $C(r)$ needs to be estimated for a camera ray through every pixel of a chosen virtual camera. To do that, $T(t)$ is calculated, which stands for the chance that the ray does not hit another particle. In addition to that, stratified sampling is used to sample small intervals instead of fixed points. The stratified sampling allows the discrete set of samples to be evaluated at continuous positions during optimization of the MLP. It enables the scene to be continuously represented. Furthermore, alpha compositing is used to control transparency of the neural radiance fields (Mildenhall et al., 2020).

2.1.5 Rendering loss

After volume rendering, the loss is calculated. This means that the color of the ray is compared to the input point to determine what the difference or loss is. The loss formula, shown as Equation (2) calculates this difference as a squared error for both the coarse and the fine renderings (Mildenhall et al., 2020). The following components make up Equation (2):

- L is the loss which can be interpreted as the difference between the ground truth and the predicted value
- $C(\mathbf{r})$ is the ground truth volume predicted RGB colours
- $\hat{C}_c(\mathbf{r})$ is the course volume predicted RGB colours
- $\hat{C}_f(\mathbf{r})$ is the fine volume predicted RGB colours

$$L = \sum_{\mathbf{r}\in R} \left[\|\hat{C}_c(\mathbf{r}) - C(r)\|_2^2 + \|\hat{C}_f(\mathbf{r}) - C(r)\|_2^2 \right] \tag{2}$$

2.2 DEM, DSM, DTM

In this research, the focus lies on creating height models and therefore, it is important to discuss some terminology. The terms Digital Terrain Model (DTM), Digital Surface Model (DSM) and Digital Elevation Model (DEM) are often used interchangeably. However, there are some differences that should be taken into account when using these terms. Therefore, this section aims to explain the differences between DTM, DSM and DEM.

Firstly, a DTM solely focuses on the bare surface of the Earth without any man-made objects or vegetation. DSMs do take these objects and vegetation into account and can thus be used for different purposes. The normalized DSM (nDSM) provides the height distance between a DTM and a DSM. Lastly, a DEM is an overarching term for DTM and DSMs as it covers the elevation, but there is no specification on what is included. Figure 7 shows the differences between the terms for height models (Ledoux et al., 2024).

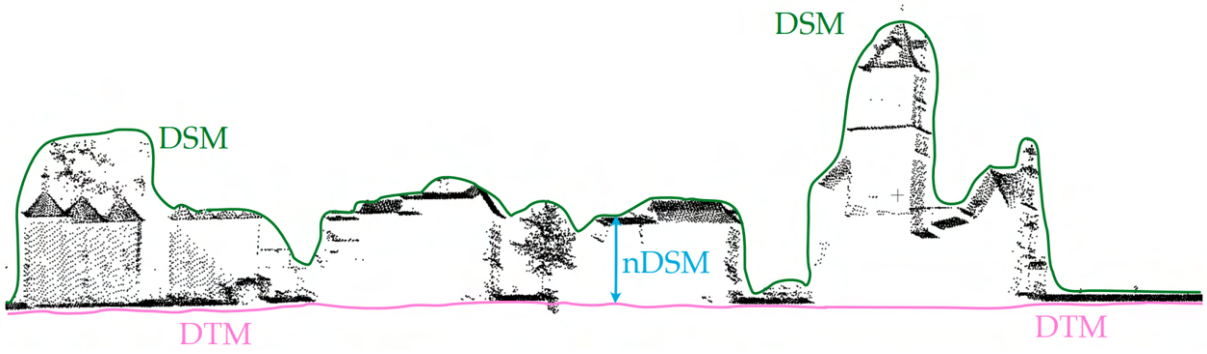


Figure 7: Elevation profile view of a point cloud in Delft showing a DTM in pink, a DSM in green and the nDSM in blue (Ledoux et al., 2024).

In this research, the derived height models are referred to as DSMs because all objects on the Earth are included. However, a distinction must also be made between 2.5D, 2.75D and volumetric modeling (full 3D). Figure 8 illustrates the differences for the types of modeling. 2.5D does not include surfaces under overhanging objects such as trees or balconies and can always be projected on a 2D plane. 2.75D allows for more than 1 z value at each location, while 3D modelling also includes the volumetric modelling of objects (Ledoux et al., 2024). This research only includes 2.5D modeling due to restrictions in the GIS software used.

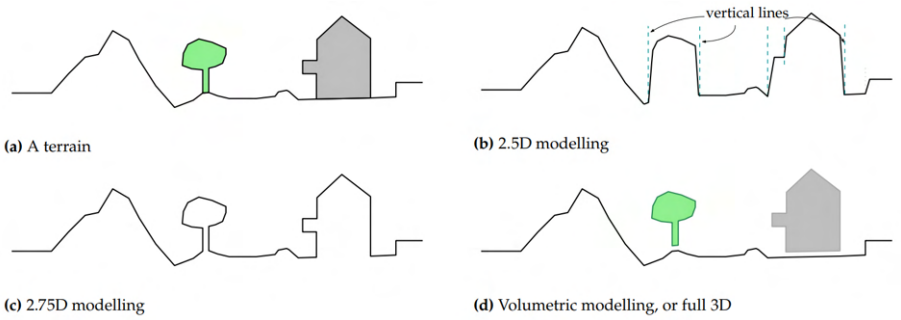


Figure 8: Differences between a terrain (a), 2.5D modelling (b), 2.75D modelling (c), and volumetric modelling or full 3D (d) (Ledoux et al., 2024).

2.3 Photogrammetry

Photogrammetry is defined by Förstner and Wrobel (2016, p.1), as "the science and technology of obtaining information about the physical environment from images, with a focus on applications in mapping, surveying and high-precision metrology". The goal of photogrammetry is to perform these tasks either automatically or semi-automatically, specifically focusing on completeness, reliability and accuracy.

Since the 1930s, humans have been creating topographic maps from aerial imagery. At that time, maps were still drawn by hand on paper. Later, in the 1970s, a new type of data in the form of satellite imagery emerged, some of these satellites were also equipped with laser scanners. Today, unmanned aerial vehicles are widely used as a means to obtain data. Data created from different types of data capture is stored and processed in geoinformation systems (GIS) (Förstner & Wrobel, 2016).

Photogrammetry is based on the principle of triangulation in 3D. Using imagery taken from different angles, semantic information can be derived, such as the elevation of an area. Figure 9 shows the traditional photogrammetry pipeline. Firstly, a task is specified which leads to a flight plan to shoot the imagery. It is important to create a flight plan to ensure full coverage and object identification. Usually, the measurements of the camera pose and orientation are aided by the Global Positioning System (GPS) and inertial measurement units. Cameras are designed so that the perspective model remains accurate up to minor distortions, which are determined by precalibration. Bundle adjustment is used to better estimate orientation parameters and align 3D scene points and 2D scene points (Förstner & Wrobel, 2016). Some common techniques in photogrammetry pipelines are discussed in the following sections.

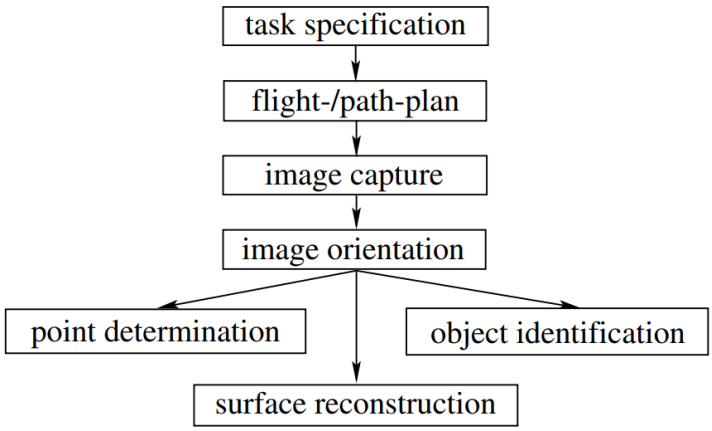


Figure 9: Traditional pipeline of photogrammetry (Förstner & Wrobel, 2016).

2.3.1 Structure-from-motion (SfM)

Structure-from-Motion (SfM) is a method that is often included as a pre-processing step in NeRFs. The method can be used to estimate the camera orientation, which is necessary to cast the rays used in NeRF for example (Marí et al., 2022). Contrary to photogrammetry, SfM does not require calibrated cameras. Additionally, SfM uses feature detection algorithms while photogrammetry uses ground control points or fiducial marks (Schönberger & Frahm, 2016).

SfM is defined by Schönberger and Frahm (2016, p.1), as "the process of reconstructing 3D structure from its projections into a series of images taken from different viewpoints". Basically, a 3D reconstruction is created from 2D imagery. Schönberger and Frahm (2016) propose incremental SfM as a consecutive pipeline including an iterative reconstruction element. The first step is the extraction and matching of features in the imagery. After that, a geometric verification step is executed and a scene graph is produced. The scene graph serves as a base for the reconstruction stage that starts with a two-view reconstruction. More images are added incrementally and triangulation of scene points is performed. Additionally, outliers are filtered and reconstruction is improved using bundle adjustment (Schönberger & Frahm, 2016). Bundle adjustment is further discussed in the next section due to its usage in some of the DSM generating NeRFs.

2.3.2 Bundle Adjustment

Bundle adjustment is described by Triggs et al. (2000, p.1), as "the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates." The name bundle adjustment originates from the bundle of light rays that are reflected by features, which are adjusted according to the feature and camera orientation and location. Everything is adjusted in one bundle, including camera and structure parameters. Reprojection errors are reduced by bundle adjustment, decreasing discrepancies between the location of features in the imagery and the projected positions from camera parameters and 3D points. Figure 10, shows a schematic representation of how bundle adjustment works.

Marí et al. (2021) have developed a bundle adjustment pipeline specifically designed for refining the camera reports of satellite images. The code for the pipeline is publicly available on GitHub and is used in this research to pre-process satellite images of the DFC2019 dataset. The pipeline improves the accuracy of the camera reports, which also improves the final results of the NeRF model.

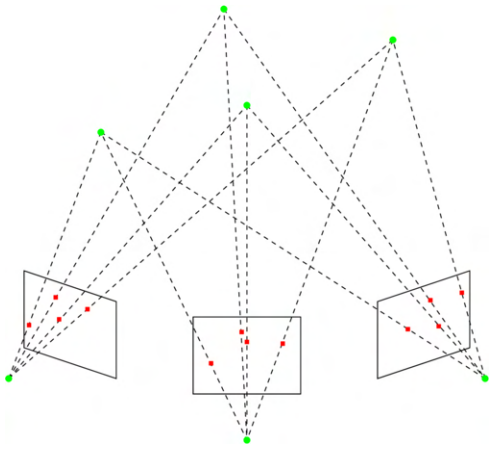


Figure 10: Schematic illustration, showing the process of bundle adjustment (Moons et al., 2009).

2.3.3 Multi-view stereo (MVS) and Semi-Global Matching (SGM)

Where SfM creates a sparse point cloud and is used to estimate camera position and orientation, multi-view stereo (MVS) can use these estimates to create a more dense point cloud and a more accurate 3D reconstruction. The MVS pipeline usually consists of 4 steps involving image collection, camera parameter estimation for each image, 3D geometry reconstruction and optionally material reconstruction of the scene. The results of MVS heavily rely on the quality of the input imagery and camera parameter estimation. Thus, MVS is dependent on the SfM algorithms that are underlying (Furukawa & Hernández, 2015).

Semi-Global Matching (SGM) further improves the final 3D model by performing a smoother and more accurate depth estimation. The method focuses on pixelwise matching of mutual information. Additionally, a global smoothing constraint is approximated by combining multiple 1D constraints. Furthermore, disparity is calculated with sub-pixel accuracy and occlusion detection (Hirschmuller, 2005).

2.4 Rational Polynomial Camera (RPC) models

The Rational Polynomial Camera (RPC) model is defined by Marí et al. (2021, p.345), as "A generic sensor model, which is widely used to describe the acquisition process of optical satellite sensors independently from the specific physical properties of the sensor." The model describes the geometric relationship between image coordinates (pixels) and object-space coordinates (latitude, longitude, and elevation) (G. Zhang & Yuan, 2006). Due to their widespread usage, satellite vendors tend to include an RPC model with the satellite imagery that they sell (K. Zhang et al., 2019). RPC models allow for high-resolution satellite imagery applications due to their increased fitting capability and simplicity. Additionally, RPC models enable photogrammetric interoperability of different satellite imagery as a result of its generalization ability (G. Zhang & Yuan, 2006).

The RPC model usually consists of 78 coefficients and 10 normalization constants. The formulas are shown below in Equation 3 and 4 and have the following components:

- Two dimensional image coordinates u and v.
- Latitude, longitude and altitude x, y, z.
- Ratios of two cubic polynomials parameterized by 39 coefficients g, h.
- Normalization constants $\mu_{x,y,z,u,v}$ and $\sigma_{x,y,z,u,v}$.

$$u = \mu_u + \sigma_u \cdot g\left(\frac{x-\mu_x}{\sigma_x}, \frac{y-\mu_y}{\sigma_y}, \frac{z-\mu_z}{\sigma_z}\right) \tag{3}$$

$$v = \mu_v + \sigma_v \cdot h\left(\frac{x-\mu_x}{\sigma_x}, \frac{y-\mu_y}{\sigma_y}, \frac{z-\mu_z}{\sigma_z}\right) \tag{4}$$

As stated before, these two functions describe the geometric relation between the 2D image coordinates and 3D ground coordinates. The RPC model is an important input in some of the NeRF models that are used to create DSMs from satellite imagery, which will be discussed further in this report.

2.5 Light Detection And Ranging (LiDAR)

LiDAR stands for Light Detection And Ranging, and is a method to measure the distance between an object and the scanner. LiDAR usually consists of a laser, a scanner and a specialized GPS. Often, the equipment is mounted to an aerial vehicle, such as a plane or a helicopter. After that, the vehicle flies over an area and measures the elevation accurately. It measures the distance between the scanner and the ground by evaluating the return time of the laser pulse. LiDAR is used for many applications including archaeology, agriculture, autonomous vehicles, forestry, military, and many more (National Ocean Service & Administration, 2024). In DSM research, a LiDAR elevation model is often considered ground truth due to its accuracy (Derksen and Izzo, 2021; Marí et al., 2022; Xie et al., 2023). However, LiDAR is not without errors, which must be considered when evaluating results (Liu, 2008).

2.5.1 Actueel Hoogtebestand Nederland

The Actueel Hoogtebestand Nederland (AHN) is developed by a collaboration between the water boards, the provinces, and Rijkswaterstaat with the goal of creating a digital elevation dataset of the Netherlands. In this research, the AHN is used as a hypothetical ground truth (gt) dataset to compare the NeRF derived DSMs to. The AHN contains both a DTM and a DSM and has been an ongoing project from 1997 until present. It is provided to the public as open data with no costs. The collection and processing of the elevation data takes multiple years and therefore, a new version of the AHN is distributed every few years. The most recent version is AHN 5, which is not available for the whole of the Netherlands yet (AHN, 2024).

To develop the end product of AHN, multiple steps have to be taken. Firstly, the data is collected through flying flight paths with an airplane. The airplane has a LiDAR scanner on board, which accurately measures the height and creates a point cloud. After the flight paths are flown, the data is checked for completeness and quality to ensure it is of the required standard. Next, the data is processed further and precise Global Navigation Satellite System (GNSS) corrections are applied as well as the alignment of flight paths. The filtering of the data is the most labor intensive part of the pipeline. Filtering allows for distinction between vegetation, buildings and the ground level with their respective classification codes. After filtering, some checking operations are executed on point density, point distribution and the filtering itself. Furthermore, the 3D orientation and location are also examined. If the checking is successful, the data is accepted. When the data is accepted, it can be merged with the existing data of other areas to create a comprehensive elevation dataset of the whole of the Netherlands. Finally, the data is distributed as open data to the public through multiple services (AHN, 2024).

2.6 NeRF models and COLMAP for DSM generation

This section aims to provide an overview of the existing state-of-the-art NeRF models that use satellite imagery to create DSMs. Each NeRF variant adds different methods that focus on optimizing various issues concerning multi-date imagery, accuracy and computing times. Due to differences in settings and hardware, a separate table concerning the accuracy is provided for each NeRF variant. The differences often make the results of the NeRF models difficult to compare. An overview of the NeRF models that are tailored for DSM creation using satellite imagery is shown in Figure 11. Additionally, this section focuses on the DSMs and does not include analysis on novel view synthesis, since that topic is out of scope for this research. Furthermore, the network architecture is not discussed for most models for simplicity reasons. Note that not all of the methods and NeRF models discussed in this section are used in this research. Especially the methods of Sat-Nerf are explained in greater detail because most NeRF models are based on it. Next to Sat-NeRF, increased attention should be paid to SAT-NGP. In addition to the NeRF models, the methodology and results of Claesen (2024) are also briefly explained.

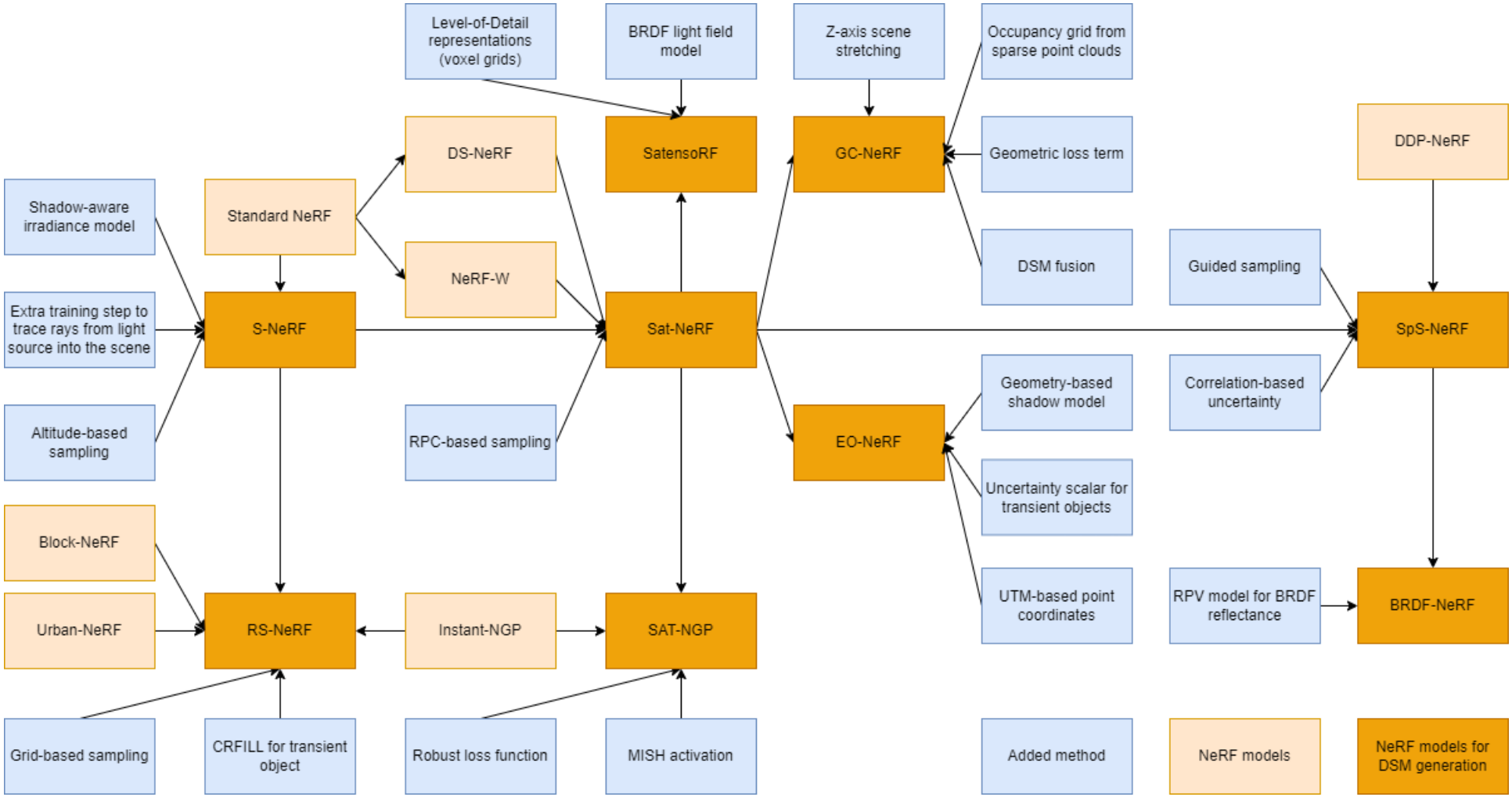


Figure 11: An overview of the NeRF models that are adjusted to create DSMs from satellite imagery, with their respective added methods.

2.6.1 DSM generation using COLMAP

Claesen (2024) aimed to adjust and improve an existing pipeline to generate DSMs from satellite imagery. The original pipeline of K. Zhang et al. (2019) was implemented using COLMAP feature detection and matching using Scale-Invariant Feature Transform (SIFT). COLMAP is a pipeline that uses SfM and MVS with a graphical and command line interface. Additionally, an adjusted pipeline was implemented using Deep Iterative Subspace Keying (DISK) for feature extraction and LightGlue for feature matching (Claesen, 2024). Figure 12 shows an overview of the two different pipelines. Both methods were tested on the Intelligence Advanced Research Projects Activity Multi-View Stereo 3D mapping challenge (Worldview-3 panchromatic data), and Den Haag and Delft datasets (Superview-1 panchromatic data). AHN 4 is used for a ground truth DSM.

The mean difference between the derived DSM and ground truth is not reported on for the Intelligence Advanced Research Projects Activity dataset. However, the mean differences for the original pipeline are 3.72m and 0.88m for Delft and Den Haag respectively. The adjusted pipeline resulted in mean differences of 3.52m and 3.25m for Delft and Den Haag respectively. This means that the vertical accuracy improved for Delft and decreased for Den Haag when applying the adjusted pipeline. Claesen (2024) does not provide an explanation for the decrease in accuracy of the Den Haag dataset. It is interesting to compare the results to the results in this research.

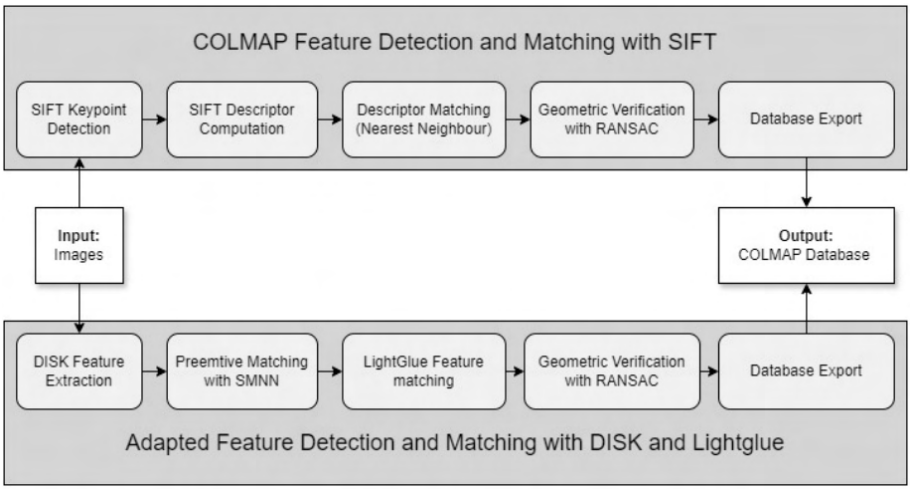


Figure 12: Overview of the two seperate methods that Claesen (2024) has tested including the colmap feature detection using SIFT and the adapted feature detection using DISK and Lightglue (Claesen, 2024).

2.6.2 Shadow Neural Radiance Fields (S-NerRF)

Derksen and Izzo (2021) were the first to develop and optimize NeRF specifically for photogrammetry purposes using high resolution satellite imagery. To create the models, the DFC2019 dataset was used, comprising 10-21 WorldView-3 satellite images of Jacksonville, Florida with a 0.3m resolution and an off-nadir angle below 35° (Le Saux et al., 2019). It must be noted that only the RGB bands were used and that the images are downsampled to 0.6m resolution. The images are downsampled due to the available LiDAR test dataset, which has a resolution of 0.5m. The optimization of NeRF posed different issues compared to computer vision applications. Firstly, the satellite imagery is taken at different timestamps resulting in major non-correlation Additionally, the amount of viewing angles and their distribution is reduced, due to the dataset specifications mentioned earlier. Classic NeRF performs poorly with varying light sources, therefore S-NeRF adds an explicit model that incorporates shadow effects to create more accurate novel-view synthesis and 3D reconstruction. To improve on classic NeRF, S-NeRF introduces three different methods (Derksen & Izzo, 2021).

Firstly, the shadow-aware irradiance model learns the different sources of light for each scene, including a directional white light source (Sun, s) and a diffuse coloured light source (Sky). The light sources are learned in order to render ambient light effects and realistic shadows. The shadow-aware irradiance model is shown in Figure 13, where the outgoing light (c) is derived by multiplying the albedo (a) and the total incoming light (l). Secondly, Derksen and Izzo (2021) use an extra training step to trace rays from the light source into the scene. The additional training step enhances the quality of the learned light source visibility at every point. Lastly, to improve the sampling of the rays, an altitude-based sampling scheme is used. By using altitude-based sampling, the model is better fitted to the relatively "flat" shape of the Earth. The altitude bounds used for the model can be obtained by using an existing LiDAR dataset or a different (coarse) elevation dataset (Derksen & Izzo, 2021).

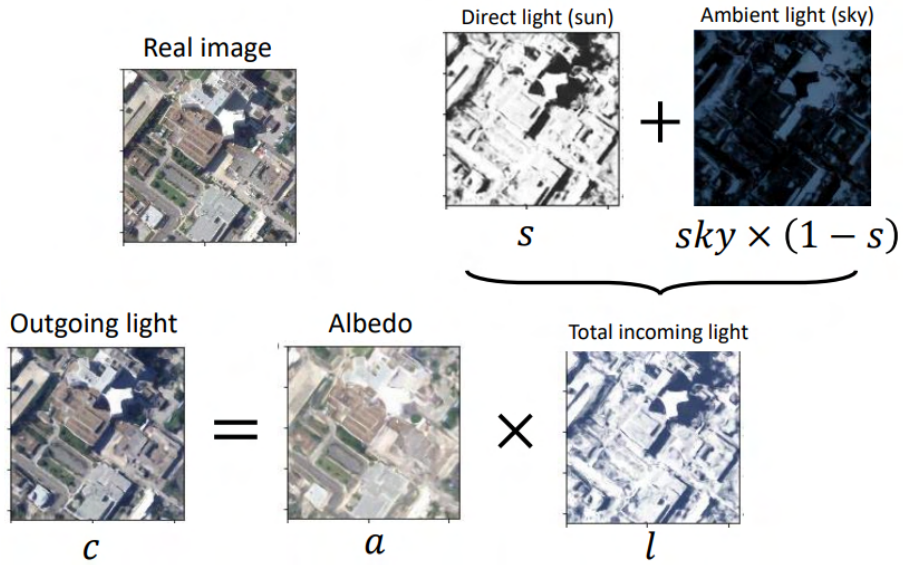


Figure 13: The shadow-aware irradiance model as implemented by Derksen and Izzo (2021)

The model is tested on the study areas 004, 068, 214 and 260 of the DFC2019 dataset, each covering an area of 300m x 300m. For each study area, the Mean Absolute Error (MAE) compared to LiDAR is calculated. This measurement gives the mean deviation from the LiDAR gt DSM, treating negative and positive errors similarly. The MAE is used in every NeRF research as the main DSM accuracy performance indication. Both classic NeRF and S-NeRF without solar correction and with solar correction are compared. The NeRF variants are executed with $N = 64$ sample points per ray and $h = 100$ units for each MLP layer with a batch size of 1024 rays. Training S-NeRF took approximately 8 hours using a GPU with 12GB RAM. The results of Derksen and Izzo, 2021 are shown in Table 1 and show that S-NeRF outperforms classic NeRF for each study area. However, solar correction does not consistently improve the results. In the 004 and 214 study areas, the MAE is lower if solar correction is not included. The code is available for S-NeRF on GitHub (Derksen & Izzo, 2021).

Area index	004	068	214	260
# train images	8	16	21	14
Alt. bounds (m)	-30/0	-30/30	-30/80	-30/30
Altitude MAE (m)				
NeRF	5.607	7.627	8.035	11.97
S-NeRF no SC	**3.342**	4.799	**4.499**	10.18
S-NeRF + SC	4.418	**3.644**	4.829	**7.173**

Table 1: Results of Derksen and Izzo, 2021 showing the MAE(m) for each area and NeRF variation (SC indicates solar correction). The most accurate result for each research area is indicated in bold.

2.6.3 Satellite Neural Radiance Fields (Sat-NeRF)

Sat-NeRF combines the work of S-NeRF, Depth-supervised NeRF (DS-NeRF) and NeRF in the Wild (NeRF-W) to solve satellite-specific issues concerning the uncommon camera models, the large distance between a scene and the camera, and image appearance differences due to multi-date collection (Deng et al., 2022, Derksen and Izzo, 2021, Marí et al., 2022, Martin-Brualla et al., 2021).

NeRF-W is able to learn what is part of the static scene and removes transient objects that are only present in some of the imagery but not all of it. Furthermore, the removal of transient objects also increases the robustness to radiometric variation (Martin-Brualla et al., 2021).

DS-NeRF uses sparse depth supervision to improve learning rates and decrease the number of input images. To achieve this, a depth supervision term is added to the loss function. The sparse depth is generated by a Structure-from-Motion (SfM) pipeline, which is, as mentioned in Section 2.3.1, common in other NeRF variants to estimate camera orientation (Deng et al., 2022).

Sat-NeRF improves on S-NeRF and adopts the shadow-aware irradiance model. Next to that, Sat-NeRF applies a different point sampling strategy. The proposed point sampling strategy is based on the Rational Polynomial Camera (RPC) model that is used for satellite cameras (discussed in Section 2.4. Instead of modeling the satellite camera as a pinhole camera in S-NeRF, Sat-NeRF uses the projection and localization function of the RPC model to cast rays directly in the object space. A benefit of the RPC-based method includes the independence to satellite systems, which is useful when applying NeRF to different datasets (Marí et al., 2022). Furthermore, Marí et al. (2022) use a correction of RPC inconsistencies prior to training, by implementing bundle adjustment as proposed by Marí et al. (2021). As explained in Section 2.3.2, bundle adjustment involves the simultaneous optimization of both the viewing parameters (external and/or internal) of multiple cameras and the 3D positions of the objects they capture. It aims to reduce the reprojection error, which is the difference between the observed image points

and the projected points from the 3D model (Marí et al., 2021).

The Sat-NeRF network architecture that is shown in Figure 14 provides the foundation for multiple NeRFs that use satellite imagery to create DSMs. Due to its importance, it will be discussed in this section. The network architectures of the other NeRFs will not be discussed for simplicity reasons. Firstly, the input coordinates $\mathbf{x}$ are fed into a fully connected 8 layer MLP with SIREN initiation for the first layer (Marí et al., 2022). SIREN stands for Sinusoidal Representation Networks and uses different activation functions compared to ReLU and sigmoid activation. Activation functions influence the output of the neurons and SIREN uses sine functions as outputs. The sine functions are able to capture more details while maintaining efficiency (Sitzmann et al., 2020). After the 8th layer, the volume density is outputted through softplus activation. The softplus activation results in a smoother approximation compared to ReLU. An additional layer is added to the MLP that produces the uncertainty coefficient β together with the transient embedding $\boldsymbol{t}_j$ and another layer with half the neurons and softplus activation. The albedo color $\boldsymbol{c}_a$ is derived through a layer with half the neurons and sigmoid activation. Shading scalar s uses three additional layers with half the neurons, direction of solar rays ω and sigmoid activation. The direction of the solar rays ω is also used for the ambient color $\boldsymbol{a}$ (Marí et al., 2022).

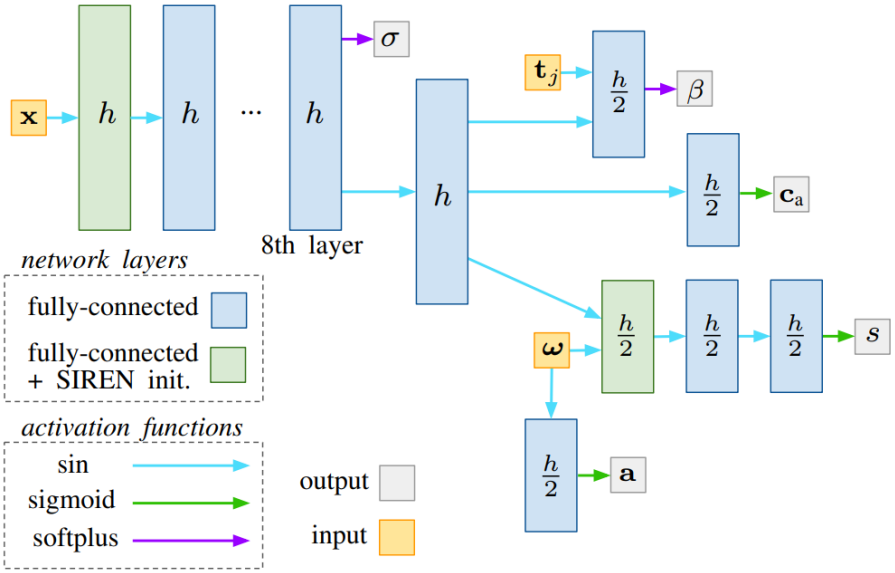


Figure 14: Network architecture of Sat-NeRF, where $\mathbf{x}$ represents the input coordinates, h the number of neurons, σ the volume density, $\boldsymbol{\omega}$ the direction of the solar rays extracted from the azimuth and elevation angle of each input image, $\boldsymbol{t}_j$ the transient embedding vector of image j manually set to $N^{(t)}$=4, β the uncertainty coefficient, $\boldsymbol{c}_a$ the albedo color, s the shading scalar and $\boldsymbol{a}$ the ambient color. Fully connected layers are shown in blue and with optional SIREN initiation in red. Additionally, activation functions are shown in light blue (sin), red (sigmoid) and dark blue (softplus) (Marí et al., 2022).

The same DFC2019 dataset and study areas are used to evaluate Sat-NeRF, except the areas of interest are slightly reduced to 256 x 256 m. The reason for the smaller areas is not stated by (Marí et al., 2022). The NeRF variants are executed with $N = 64$ sample points per ray and $h = 512$ units for each MLP layer with a batch size of 1024 rays. The training of Sat-NeRF takes approximately 10 hours without solar correction or depth supervision and twice as much when they are included. Table 2 shows the MAE results of Sat-NeRF. The results indicate that RPC-based point sampling and bundle adjustment have a positive influence on the MAE values, as the MAE values are lower for NeRF and S-NeRF compared to the values in Derksen and Izzo (2021). Sat-NeRF outperforms S-NeRF and standard NeRF in each scene. The effects of depth supervision and solar correction differ for each scene, as they do not consistently improve the results. Similarly to S-NeRF, the code for Sat-NeRF is available on GitHub (Marí et al., 2022).

Area index	004	068	214	260
# train images	9	17	21	15
Alt. bounds (m)	-24/1	-27/30	-29/73	-30/13
Altitude MAE (m)				
NeRF	3.327	2.593	2.691	3.257
S-NeRF	1.830	1.496	3.687	3.245
S-NeRF + SC	1.472	1.374	2.406	2.299
S-NeRF + SC (no BA)	4.418	3.644	4.829	7.173
Sat-NeRF	1.416	1.275	2.125	2.428
Sat-NeRF + SC	**1.288**	**1.249**	2.009	1.864
Sat-NeRF + SC (no BA)	1.577	1.392	2.176	1.875
Sat-NeRF + DS	1.420	1.298	1.714	**1.624**
Sat-NeRF + DS + SC	1.366	1.277	**1.676**	1.638

Table 2: Results of Marí et al. (2022), showing the MAE(m) for each area and NeRF variation. It must be noted that RPC-based point sampling and bundle adjustment are applied unless specified otherwise. SC and DS stand for solar correction and depth supervision respectively. The most accurate result for each research area is indicated in bold.

2.6.4 Multi-Date Earth Observation NeRF (EO-NeRF)

EO-NeRF builds upon S-NeRF and Sat-NeRF to better fit multi-date satellite data. Multiple improvements are made by Marí et al. (2023) to make the NeRF model more accurate.

The first improvement comprises a different shadow model compared to S-NeRF and Sat-NeRF. S-NeRF and Sat-NeRF try to estimate shadows as a color property determined by the solar direction (Derksen and Izzo, 2021, Marí et al., 2022). However, this method does not perform well when the solar direction differs between the input images. The S-NeRF and Sat-NeRF models use a solar correction term to mitigate the decreased performance, which has some unwanted side effects. EO-NeRF uses a different shadow model that renders shadows based on geometry for every optimization step. This method separates shadows from other transient objects in the scene (Marí et al., 2023).

The second improvement includes the usage of a different network architecture. EO-NeRF combines the loss-functions used in Sat-NeRF and NeRF-W, which introduce an uncertainty scalar that reduces the contribution of camera rays coming from transient objects (Marí et al., 2023, Martin-Brualla et al., 2021).

Furthermore, EO-NeRF adopts Universal Transverse Mercator-based (UTM) point coordinates instead of Earth-centered Earth-fixed, which is used by Sat-NeRF. By using UTM coordinates and altitude to represent 3D points, the properties of the Cartesian system are preserved. Additionally, it is beneficial to have the altitude aligned with the z-axis to better fit the 3D space of the scene. Figure 15 illustrates the improved use of space by the UTM-based method (Marí et al., 2023).

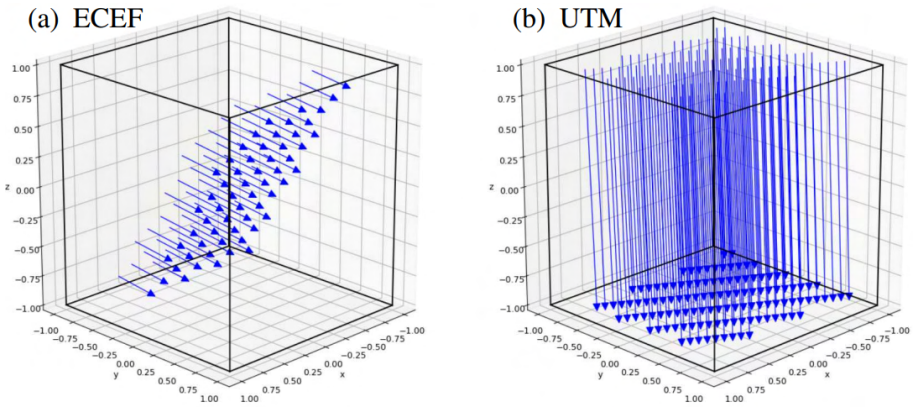


Figure 15: Earth-centered Earth-fixed (ECEF)(a) and a UTM-based(b) 3D space to project 3D point coordinates. The coordinates are normalized between -1 and 1. Both figures use the same RPC camera model to cast rays (Marí et al., 2023).

The final adjustment of EO-NeRF consists of implementing bundle adjustment internally, compared to Sat-NeRF which uses bundle adjustment before the MLP optimization (Marí et al., 2023, Marí et al., 2022). Based on previous research, it is possible to perform bundle adjustment simultaneously with the MLP optimization (Lin et al., 2021). Table 3 shows that simultaneous bundle adjustment yields improved results compared to the external bundle adjustment used in Sat-NeRF.

EO-NeRF is tested on the same DFC2019 dataset as S-NeRF and Sat-NeRF (Le Saux et al., 2019). In addition to the optical RGB imagery, EO-NeRF is able to handle RAW pansharpened imagery. Furthermore, each NeRF variant is executed with $N = 128$ points per ray and $h = 256$ neurons for each MLP layer with a batch size of 1024 rays. Training the model takes between 10 and 20 hours on a 12GB GPU, which is comparable to Sat-NeRF. Table 3 shows that EO-NeRF has lower MAE values compared to Sat-NeRF, indicating improved performance. Especially for the raw pansharpened imagery, EO-NeRF greatly outperforms Sat-NeRF. Furthermore, EO-NeRF is tested with the same external unrefined RPC camera models, showing decreased performance for both types of imagery. However, when internal bundle adjustment is applied, results are similar to standard EO-NeRF. The code for EO-NeRF is not publicly available (Marí et al., 2023).

	True color RGB				Raw pansharpened			
Area index	004	068	214	260	004	068	214	260
# train images	9	17	21	15	9	17	21	15
Alt. bounds (m)	-24/1	-27/30	-29/73	-30/13	-24/1	-27/30	-29/73	-30/13
	Altitude MAE (m)				**Altitude MAE (m)**			
Sat-NeRF	1.48	0.98	2.29	1.76	2.64	1.39	2.38	2.37
EO-NeRF	**1.25**	0.91	1.52	**1.43**	**1.33**	**0.89**	**1.41**	**1.34**
EO-NeRF, ext. RPC	1.39	1.42	1.74	1.95	1.72	1.08	1.58	1.53
EO-NeRF, ext. RPC+ int. BA	1.30	**0.90**	**1.38**	1.48	1.68	0.95	1.44	1.36

Table 3: Results of Marí et al. (2023), showing the MAE(m) for each area, imagery type and NeRF variation. It must be noted that the UTM-based point coordinates are used for every NeRF type. The most accurate result for each research area is indicated in bold.

2.6.5 Remote Sensing Neural Radiance Fields (RS-NeRF)

RS-NeRF is mostly based on S-NeRF, however Xie et al. (2023) also draw inspiration form Instant-NGP (Müller et al., 2022), NeRF (Mildenhall et al., 2020), Block-NeRF (Tancik et al., 2022) and Urban Radiance Fields (Rematas et al., 2022). In addition to improving the accuracy of the DSMs compared to S-NeRF, RS-NeRF also aims to improve the training speed, for which Instant-NGP is used.

Instant-NGP is an acceleration method, which can improve the training times that are required for NeRF. The method uses multi-resolution hash encoding in order to store 3D features, resulting in highly reduced training times. Through the method, GPU parallel efficiency is enhanced and memory access is simplified. The result is a reduction in training times from hours to minutes or seconds for different NeRFs (Müller et al., 2022).

To optimize sampling, RS-NeRF uses a grid-based sampling method. The method divides the 3D space in which the scene is located into 128^3 voxels. After that, model throws away sample points that are located in empty voxels. Empty voxels are either located before or after the ray hits something. Sample points in empty voxels are not used for input in the MLP which speeds up the model. Additionally, rays are sampled simultaneously on custom CUDA kernels, which further improves the training time of the model. CUDA kernels allow models to run processes parallel to each other on GPU cores (Xie et al., 2023).

To handle multi-date satellite imagery with vehicles that are in different locations on different images, CRFILL is implemented. CRFILL is an image-inpainting method that is able to remove vehicles, restoring the vehicle-free appearance of the city, while using a minor amount of computational power. CRFILL only needs to be applied to one image with the least amount of vehicles, because it can be combined with RS-NeRF (Xie et al., 2023). Figure 16 shows the vehicle removal performance of CRFILL and RS-NeRF separately and combined, indicating much better results for the combined option (Xie et al., 2023).

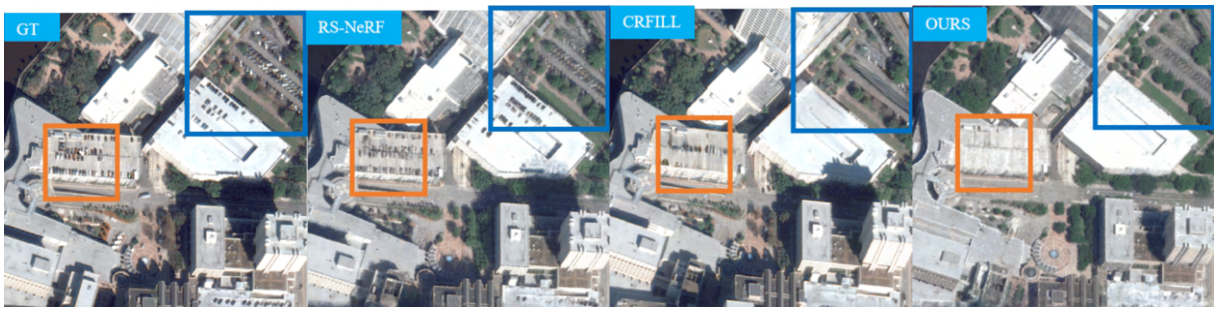


Figure 16: The vehicle removal performance of RS-NeRF, CRFILL and when they are combined (OURS) (Xie et al., 2023).

RS-NeRF is tested on the same DFC2019 dataset as S-NeRF and Sat-Nerf, only using the RGB imagery. A similar batch size of 1024 is used with $N = 1024$ sample points per ray. Furthermore, RS-NeRF only uses 20% of the amount of neurons used in S-NeRF. Training times are reduced to 6 minutes on a NVIDIA GPU with 12GB RAM. Table 4 shows the results of RS-NeRF compared to S-NeRF. RS-NeRF outperforms S-NeRF on each research area while reducing the training times from 8 hours to 6 minutes. Note that Sat-NeRF and EO-NeRF yield more accurate results, however their training times are between 10 and 20 hours. The code for RS-NeRF is not available to the public (Xie et al., 2023).

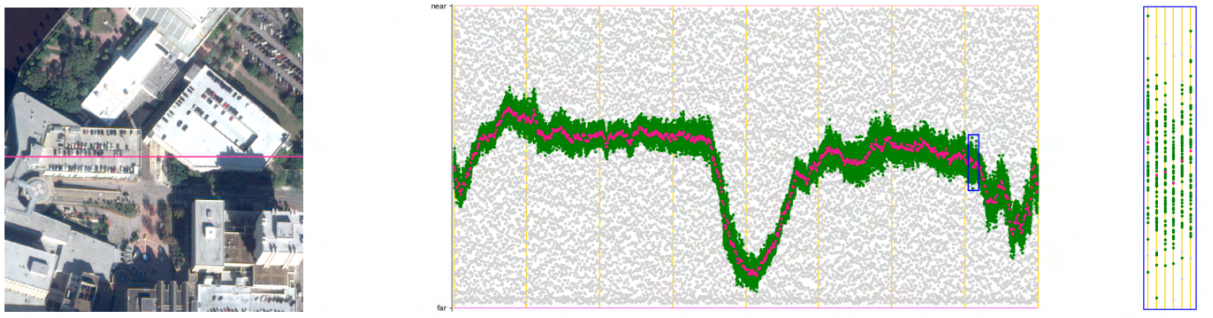
Area index	004	068	214	260
# train images	8	16	21	14
Alt. bounds (m)	-30/0	-30/30	-30/80	-30/30
Altitude MAE (m)				
NeRF	5.607	7.627	8.035	11.97
S-NeRF no SC	3.342	4.799	4.499	10.18
S-NeRF + SC	4.418	3.644	4.829	7.173
RS-NeRF no WS	3.045	2.536	10.677	5.766
RS-NeRF	**1.736**	**1.693**	**2.667**	**2.629**

Table 4: Results of Xie et al. (2023) showing the MAE (m) for each area and NeRF variation. The most accurate result is shown in bold for each research area.

2.6.6 SparseSat-NeRF (SpS-NeRF)

SpS-NeRF focuses on reducing the amount of satellite input imagery that is needed to produce accurate DSMs using NeRF (L. Zhang & Rupnik, 2023). The NeRF model builds on the architecture that is proposed in Sat-NeRF by Marí et al. (2022) and uses the guided sampling strategy proposed by Roessle et al. (2022) in DDP-NeRF.

The guided sampling separates the sampling into two phases. To start, a first batch of sampling finds the approximate location of the surface. The first batch is followed by a second batch which increasingly samples at the approximate location to find the exact location (Roessle et al., 2022). Figure 17 shows how the guided sampling is applied in SpS-NeRF (L. Zhang & Rupnik, 2023).



(a) A selected image row (–) (b) Ray sampling along the selected image row (c) Zoom

Figure 17: Ray sampling as used in SpS-neRF, (a) shows the sampled image row, (b) shows the first batch of sampling in gray with the estimated dense depths in pink and the second batch of sampling in red, (c) zooms in on the sampling of some rays. (L. Zhang & Rupnik, 2023)

Compared to Sat-NeRF, two main changes are made for SpS-NeRF. Firstly, a low resolution depth map is used which is generated through SGM or obtained through the Shuttle Radar Topography Mission (SRTM). The SRTM provides a low resolution (250 x 250 m per pixel) global dataset that can be used for supervision. The depth is included before the optimization

of the network and because it gives the network a rough estimate of the 3D structure, the network requires less input imagery. The second change involves correlation based uncertainty. An uncertainty is measured through the differences of the low resolution depth maps. This increases the robustness of the generated DSMs (L. Zhang & Rupnik, 2023).

It must be considered that SpS-NeRF does not include a method to handle transient objects. Therefore, SpS is only tested on research area 214 of the DFC2019 dataset, which has 3 images that are taken at roughly the same time (Le Saux et al., 2019). Additionally, SpS-NeRF is also tested on a dataset which has Pléiades imagery of a natural environment in Djibouti (Labarre et al., 2019). SpS-NeRF only uses RGB imagery and requires 2-3 images. A batch size of 1024 is used and $N = 64$ samples per ray per batch, the number of neurons for each MLP layer is not specified. However, the network architecture is based on Sat-NeRF so $h = 512$ is assumed. The training of the model took approximately 2 hours on a GPU with 40GB of RAM. It is difficult to compare the training time with other NeRF models due to the large difference in RAM.

The results in Table 5 show that SpS-NeRF outperforms classic NeRF and Sat-NeRF. However, it must be noted that some parts of Sat-NeRF is not optimized for sparse imagery input. Furthermore, the results indicate that SpS-NeRF performs particularly well for the Djibouti dataset with low MAE values. The code to implement SpS-NeRF is publicly available on GitHub (L. Zhang & Rupnik, 2023).

Area index	214 (2 img)	214 (3 img)	Dji (2 img)	Dji (3 img)
Altitude MAE (m)				
NeRF	9.51	6.56	9.72	14.44
Sat-NeRF	5.89	4.63	9.51	10.11
SpS-NeRF	**3.02**	**2.86**	**1.57**	**1.35**

Table 5: Results of L. Zhang and Rupnik (2023), showing the MAE (m) fof the DFC2019 and Djibouti dataset for both 2 and 3 input images.

2.6.7 Satellite Tensorial Radiance Fields (SatensoRF)

SatensoRF focuses on reducing the computational load and therefore also the training times for NeRF. Furthermore, T. Zhang et al. (2024) propose to remove the additional inputs that are needed for S-NeRF and Sat-NeRF, such as sun azimuth and elevation, which are not always present in datasets. Lastly, SatensoRF eliminates the assumption of Lambertian reflectance for the Earth's surface (T. Zhang et al., 2024). A Lambertian reflectance assumes that the surface perfectly diffuses light. In the real world, a bidirectional reflection is often present, especially for surfaces that include water, asphalt roads or vegetation (W. Qin et al., 2001).

To address the first goal of reducing computational load, Level-of-Detail representations are implemented. By creating different levels of detail with 3D voxel grids, the complexity of the scene can be managed better, reducing computing times (T. Zhang et al., 2024). This method is similar to the method used in RS-NeRF (Xie et al., 2023). The same technique is applied to the volume density field of the Earth's surface and the global hue bias. As described earlier, the volume density determines the probability that something is present at a certain location. The global hue bias adjusts the overall color tone of the scene, correcting for the sky as a light source. On top of reducing training times, the proposed implementations optimize SatensoRF for large scale satellite imagery (T. Zhang et al., 2024).

The second and third issues are addressed using a redesigned light field model that includes bidirectional reflection properties. The light field model eliminates the solar information input requirements of S-NeRF and Sat-NeRF. To handle transient objects in multi-date satellite imagery, they are treated as blackbox-noise components. To remove the noise, a total variation regularization is applied, removing transient objects from the scene (T. Zhang et al., 2024).

T. Zhang et al. (2024) report on using the same DFC2019 dataset, however they use different AOIs to evaluate their model. Due to the different usage of the dataset, results are difficult to compare, it is unclear why the MAE was not calculated for the same AOIs that other research uses. Additionally, the batch size was set to 8096 rays for SatensoRF, while it was set to 1024 for S-NeRF and Sat-NeRF. The training time for SatensoRF took 2.8 hours compared to 7.8 and 10.3 hours for S-NeRF and Sat-NeRF respectively. All of the training was performed using a 16GB RAM NVIDIA RTX3090 GPU. The results in Table 6 show that SatensoRF does not produce more accurate results compared to Sat-NeRF. However, computing times are reduced and the model does perform better considering novel view synthesis. The code for SatensoRF is not available to the public (T. Zhang et al., 2024).

Area index	017	159
# train images	9	10
Alt. bounds (m)	-23/4	-29/15
Altitude MAE (m)		
NeRF	3.449	2.628
S-NeRF + SC	2.848	2.318
Sat-NeRF + SC	2.722	1.861
Sat-NeRF + DS + SC	**1.982**	**1.315**
SatensoRF + SC	2.943	2.081
SatensoRF + DS + SC	1.998	1.859

Table 6: Results of T. Zhang et al. (2024), showing the MAE(m) of each NeRF variation for the 017 and 159 areas of the DFC2019 dataset The most accurate results for each research area are shown in bold.

2.6.8 Satellite Neural Graphics Primitives (SAT-NGP)

Similarly to RS-NeRF, SAT-NGP aims to improve computing times for the NeRF model, while maintaining DSM quality (Billouard et al., 2024; Xie et al., 2023). However, RS-NeRF does not produce relighting capabilities and Xie et al. (2023) use a separate inpainting method to process transient objects such as cars.

SAT-NGP uses the foundation of S-NeRF and Sat-NeRF, by implementing a similar shading model and network architecture (Derksen and Izzo, 2021; Marí et al., 2022). Instead of SIREN activation, which is used in Sat-NeRF, the SAT-NGP MLP uses MISH activation. MISH activation can be considered a more flexible and smooth activation compared to ReLU and SIREN. It improves the learning of neural networks by providing a smoother gradient and more complex activations (Misra, 2020). Additionally, the multi-resolution hash encoding method from Instant-NGP is used to speed up the model (Müller et al., 2022).

Another adaptation from SAT-NGP is the adoption of a robust loss function, developed by Sabour et al. (2023). Firstly, an uncertainty image is learned as a network output. Areas with many transient objects result in high uncertainty. After that, the loss function is constrained to only use low uncertainty areas. This process smooths out the areas with transient objects, such as cars, in the resulting DSM (Billouard et al., 2024).

To refine the RPC camera models, bundle adjustment is applied. Additionally, the UTM-based point coordinates are used similar to EO-NeRF (Marí et al., 2023). Again, the DFC2019 dataset comprising RGB satellite photos with a 0.3m pixel size is used. A batch size of 1024 rays is used with N between 4 and 256 samples per ray. The training takes approximately 15 minutes on a 12GB GPU.

Table 7 shows the results of Billouard et al. (2024) compared to S-NeRF, Sat-NeRF and EO-NeRF. While SAT-NGP does not provide the most accurate results, it does speed up the process from hours to minutes. Additionally, for most study areas the accuracy difference is only minor compared to Sat-NeRF and EO-NeRF. The results are also visualized in Figure 18, it must be noted that the MAE values differ for Sat-NeRF (2.009m in Table 7 compared to 4.31m in Figure 18) due to differences in settings (20hrs training time in the table compared to 12hrs mentioned for the figure). The code for SAT-NGP is publicized on GitHub (Billouard et al., 2024).

Area index	004	068	214	260
		Altitude MAE (m)		
S-NeRF	1.472 (10hr)	1.374 (20hr)	2.406 (20hr)	2.299 (20hr)
Sat-NeRF	**1.288** (10hr)	1.249 (20hr)	2.009 (20hr)	1.864 (20hr)
EO-NeRF	2.15 (10hr)	**0.91** (20hr)	**1.52** (20hr)	**1.43** (20hr)
SAT-NGP	1.31 (8min)	2.03 (14min)	2.17 (14min)	1.68 (14min)

Table 7: Results of Billouard et al. (2024), showing the MAE (m) for each research area and NeRF variation. The most accurate result is shown in bold for each research area.

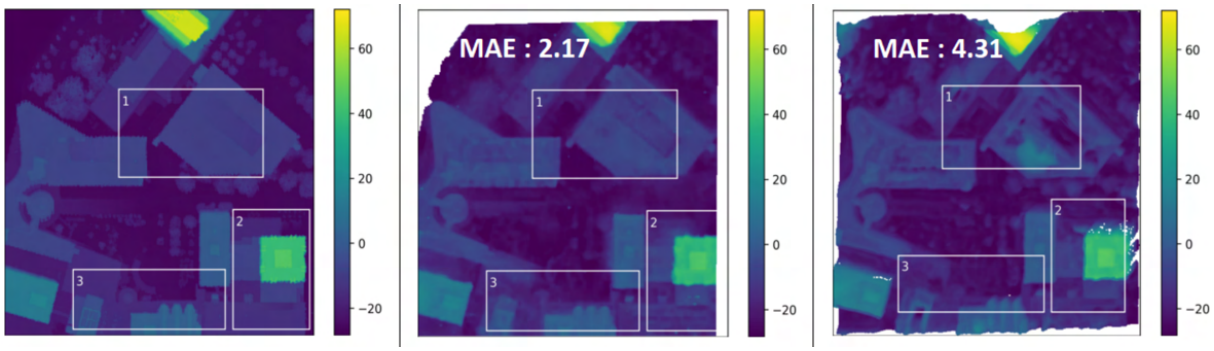


Figure 18: DSMs showing the GT LiDAR (left), SAT-NGP results (middle) and Sat-NeRF results(right). SAT-NGP has a lower MAE in 14 minutes compared to Sat-NeRF in 12 hours for AOI 214 (Billouard et al., 2024).

2.6.9 Geometric Constrained NeRF (GC-NeRF)

GC-NeRF builds on Sat-NeRF and introduces multiple methods to improve DSM accuracy, including z-axis scene stretching, an occupancy grid created from sparse point clouds, a geometric loss term and DSM fusion (Wan et al., 2024).

Z-axis scene stretching is conducted because satellite scenes usually have a larger scale horizontally compared to vertically. Stretching the z-axis of the scene makes the granularity in the z-direction finer and increases the sampling density in the z-direction, thus increasing DSM accuracy. Care must be taken not to overstretch the scene, which can lead to reduced model performance. Figure 19 shows a visual representation of z-axis stretching.

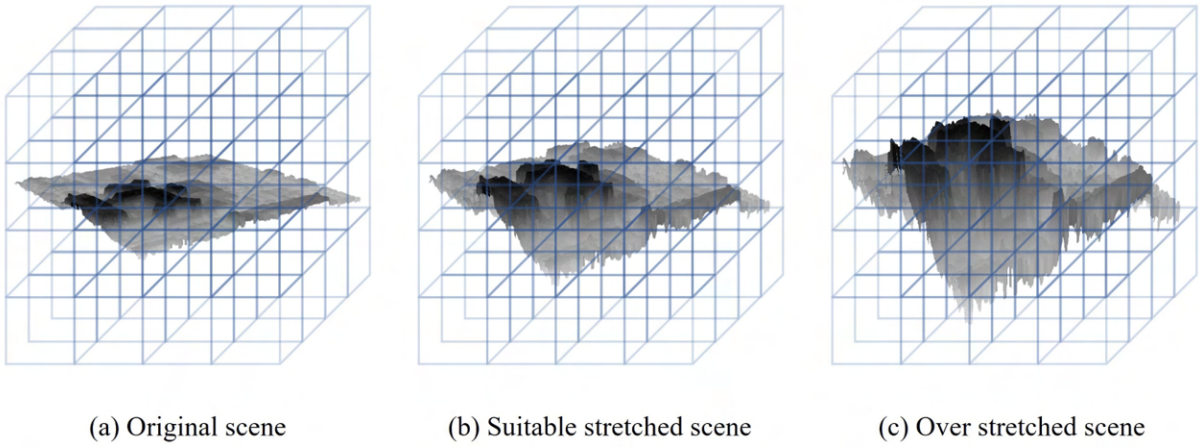


Figure 19: Z-axis stretching showing the flat original scene (a), a suitable stretched scene (b) and an over stretched scene (c) (Wan et al., 2024).

Instead of using altitude bounds to improve sampling, a sparse point cloud generated through bundle adjustment is used to guide sampling for GC-NeRF. The sparse point cloud is converted to an occupancy grid which divides the 3D scene in 128^3 voxels. This allows empty voxels to be skipped and sampling can be increased for the voxels where something is present. The approach is similar to the methods of RS-NeRF and SatensoRF (Wan et al., 2024, Xie et al., 2023, T. Zhang et al., 2024).

The introduced geometric loss term leads to a constrained scene geometry and makes the scene surface thinner. As a result, the accuracy of the DSMs are greatly improved. Additionally, GC-NeRF uses a multi-view DSM fusion strategy, to correct for large DSM errors near tall objects. For this strategy, multiple DSMs are created from different viewpoints, which are then merged into one DSM (Wan et al., 2024).

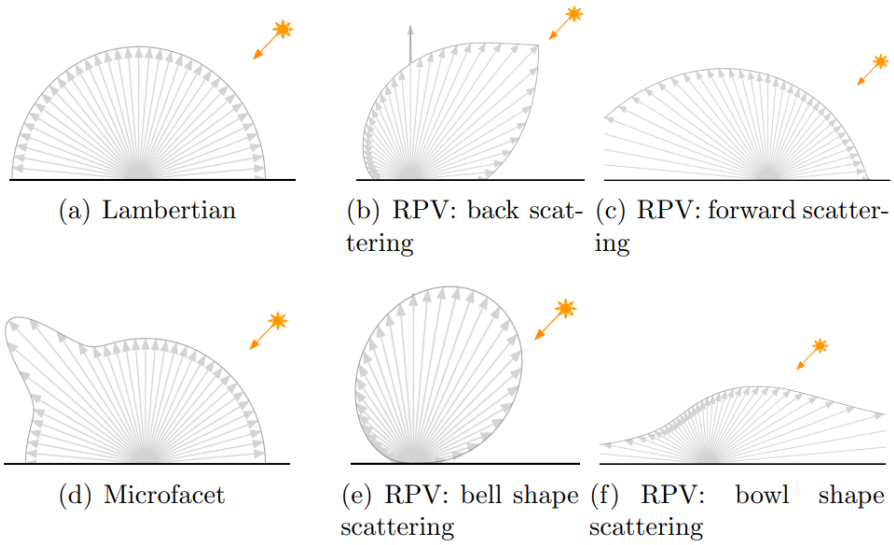
To test GC-NeRF, the same DFC2019 dataset with AOIs 004, 068, 214 and 260 is used (Le Saux et al., 2019). Wan et al. (2024) indicate that $N = 2^{18}$ samples are used for each batch of rays, but do not state the size of the batch. Training the model took 6 minutes on a 12GB NVIDIA RTX 3080 GPU. Table 8 shows that GC-NeRF is the most accurate compared to S-NeRF and Sat-NeRF. The code for GC-NeRF is not available to the public (Wan et al., 2024).

Area index	004	068	214	260
# train images	9	17	22	15
Alt. bounds (m)	-24/0	-27/30	-29/73	-30/13
Altitude MAE (m)				
S-NeRF	4.42	3.64	4.83	7.71
Sat-NeRF	1.37	1.28	1.68	1.64
GC-NeRF	**1.33**	**1.15**	**1.47**	**1.63**

Table 8: Results of Wan et al. (2024), showing the MAE (m) for each research area and NeRF variation. The most accurate result is shown in bold for each research area.

2.6.10 Bidirectional Reflectance Distribution Function NeRF (BRDF-NeRF)

L. Zhang et al. (2024) build on SpS-NeRF, focusing on accurate DSM reconstruction using sparse satellite imagery. Additionally, BRDF-NeRF aims to include the bidirectional reflectance of the surface of the Earth and is specifically designed to perform well on surfaces with anisotropic reflectance characteristics (e.g. vegetation and bare soil). The Bidirectional Reflectance Distribution Function (BRDF) indicates the way that materials reflect light as a result of different lighting conditions and viewing angles. Different scattering patterns are visualized in Figure 20 (L. Zhang et al., 2024). To include the BRDF, a Rahman-Pinty-Verstraete (RPV) model is in-



(a) Lambertian (b) RPV: back scattering (c) RPV: forward scattering

(d) Microfacet (e) RPV: bell shape scattering (f) RPV: bowl shape scattering

Figure 20: Different scattering patterns including, Lambertian (a), RPV (b, c, e, f) and Microfacet (d) scattering (L. Zhang et al., 2024).

cluded in BRDF-NeRF (Rahman et al., 1993). The RPV model is suitable for satellite imagery and uses three physics-based parameters. The parameters include the level of anisotropy, the intensity of the reflectance of the surface cover and the relative amount of forward or backward scattering. The parameter values are obtained through inversion of surface reflectance models on observations (L. Zhang et al., 2024).

To train BRDF-NeRF, a batch size of 1024 is used. A similar sampling method to SpS-NeRF is adopted with $N = 64$ stratified sampling points for each ray followed by $N = 64$ guided sampling points. The model is trained on the Djibouti dataset used in SpS-NeRF and a dataset on Lanzhou (China) consisting of 3 Pleíades 1B and 3 Pleíades 1A images (Labarre et al., 2019). Two areas for each dataset with a size of 1.5 km by 1.5 km were used and training took approximately 10 hours on a 40GB NVIDIA GPU. The results in table 9 show that BRDF-NeRF achieves the most accurate results for each research area. It must be noted that Sat-NeRF is not optimized for sparse satellite imagery input which has led to high MAE values. Code for the BRDF-NeRF model is available to the public on GitHub, however, documentation is very limited (L. Zhang et al., 2024).

Area	Dji-A	Dji-B	Lzh-A	Lzh-B
# train images	3	3	3	3
Altitude MAE (m)				
Sat-NeRF	12.85	18.059	61.299	27.489
SpS-NeRF	1.438	1.761	3.558	3.235
BRDF-NeRF	**1.378**	**1.614**	**3.42**	**2.941**

Table 9: Results of L. Zhang et al. (2024), showing the MAE (m) for each research area and NeRF variation. The most accurate result is shown in bold for each research area.

3 Data

This section covers the datasets that are used in this research. The datasets are covered before the methodology due to their influence on the methodology. Two different types of satellite datasets are used in this research. Firstly, the DFC2019 dataset, consisting of Worldview-3 imagery, is used to answer sub-research questions 2 and 3 on accuracy and sensitivity. However, the dataset is split in two ways, there is a distinction between AOIs that have already been preprocessed and those that have not been preprocessed. Additionally, the DFC2019 data is split into urban scenes comprising little vegetation and rural scenes, which have more vegetation in a suburban environment.

The other satellite dataset consists of three scenes covered by Superview-1 imagery, Delft, Den Haag and Golfcourse "Het Rijk van Margraten" near Maastricht. The Delft and Den Haag datasets have also been used by Claesen (2024). This dataset is used to answer sub-research question 4 on the applicability of NeRF, and determine how NeRF can be applied to satellite data different from DFC2019. An overview of the satellite imagery datasets is provided in Table 10.

Dataset	AOI	Satellite	Data type	Imagery resolution (m)	Scene type	Preprocessed	Number of images
DFC2019	068	Worldview-3	RGB (uint8)	0.3	Urban	Yes	11
	165	Worldview-3	RGB (uint8)	0.3	Urban	No	23
	166	Worldview-3	RGB (uint8)	0.3	Urban	No	24
	167	Worldview-3	RGB (uint8)	0.3	Urban	No	24
	214	Worldview-3	RGB (uint8)	0.3	Urban	Yes	24
	260	Worldview-3	RGB (uint8)	0.3	Urban	Yes	17
	004	Worldview-3	RGB (uint8)	0.3	Rural	Yes	11
	017	Worldview-3	RGB (uint8)	0.3	Rural	No	10
	105	Worldview-3	RGB (uint8)	0.3	Rural	No	21
	359	Worldview-3	RGB (uint8)	0.3	Rural	No	24
	203	Worldview-3	RGB (uint8)	0.3	Rural	No	24
	559	Worldview-3	RGB (uint8)	0.3	Rural	No	23
Delft		Superview-1	Multispectral (uint16)	2.0	Urban	No	21
Den Haag		Superview-1	Multispectral (uint16)	2.0	Urban	No	21
Golfcourse		Superview-1	Multispectral (uint16)	2.0	Rural	No	17

Table 10: Information on the dataset and its corresponding AOI, Satellite, Imagery data type, Imagery resolution, and Number of images.

3.1 The Data Fusion Contest 2019 (DFC2019) dataset

The DFC2019 dataset was released for a data fusion contest in 2019 by the Image Analysis and Data Fusion Technical Committee of the IEEE Geoscience and Remote Sensing Society, the Johns Hopkins University (JHU) and the Intelligence Advanced Research Projects Activity, to boost the application of machine intelligence and deep learning to satellite imagery (Le Saux et al., 2019).

The DFC2019 dataset comprises WorldView-3 panchromatic, 8-band multi-spectral and pan-sharpened RGB images of both Jacksonville, Florida and Omaha, Nebraska. The WorldView-3 satellite is owned by Maxar and is still operational (Le Saux et al., 2019). The satellite was launched on the 13th of August 2014 in a sun-synchronous orbit at an altitude of 617km. Aboard the satellite is the Worldview-110 camera which is able to capture imagery at 0.31m panchromatic, 1.24m for the visible and near-infrared bands, and 3.7m for the short-wave infrared bands. Currently, WorldView-3 captures approximately 680,000km^2 of the Earth every day (ESA, 2025).

For this research, only the Jacksonville dataset is used, which consists of 26 images taken between 2014 and 2016. The imagery is divided into 54 different scenes (AOIs) which all have been given a number. Only the pansharpened RGB images are used as input for the NeRF model, which have a spatial resolution of 0.3m. In this research, a distinction is made between the scenes that are preprocessed and the ones that are not. The scenes 004, 068, 214 and 260 are provided as a preprocessed dataset which means that the imagery is already clipped to a certain AOI, and the RPCs are also enhanced using bundle adjustment. For all of the other scenes, a preprocessing pipeline is used to prepare the data as input for NeRF.

In addition to the imagery, the DFC2019 dataset also includes airborne LiDAR data. The LiDAR data is provided as a DSM with a resolution of 0.5m, and is considered ground truth (gt). Furthermore, a classified raster file is also included, specifying different classes which are listed in Table 11.

ID	Classification
02	Ground
05	Trees
06	Buildings
09	Water
17	Bridge / Elevated road
65	Unlabeled

Table 11: The classification for the classified raster file that comes with the DFC2019 dataset (Le Saux et al., 2019).

The DFC2019 data is downloaded for free from https://ieee-dataport.org/open-access/data-fusion-contest-2019-dfc2019 (Le Saux et al., 2019). To provide a view of what the satellite images look like, sample images of the selected rural and urban AOIs are shown in Figure 21. In total, 12 AOIs are selected, 6 in the urban category and 6 in the rural category.

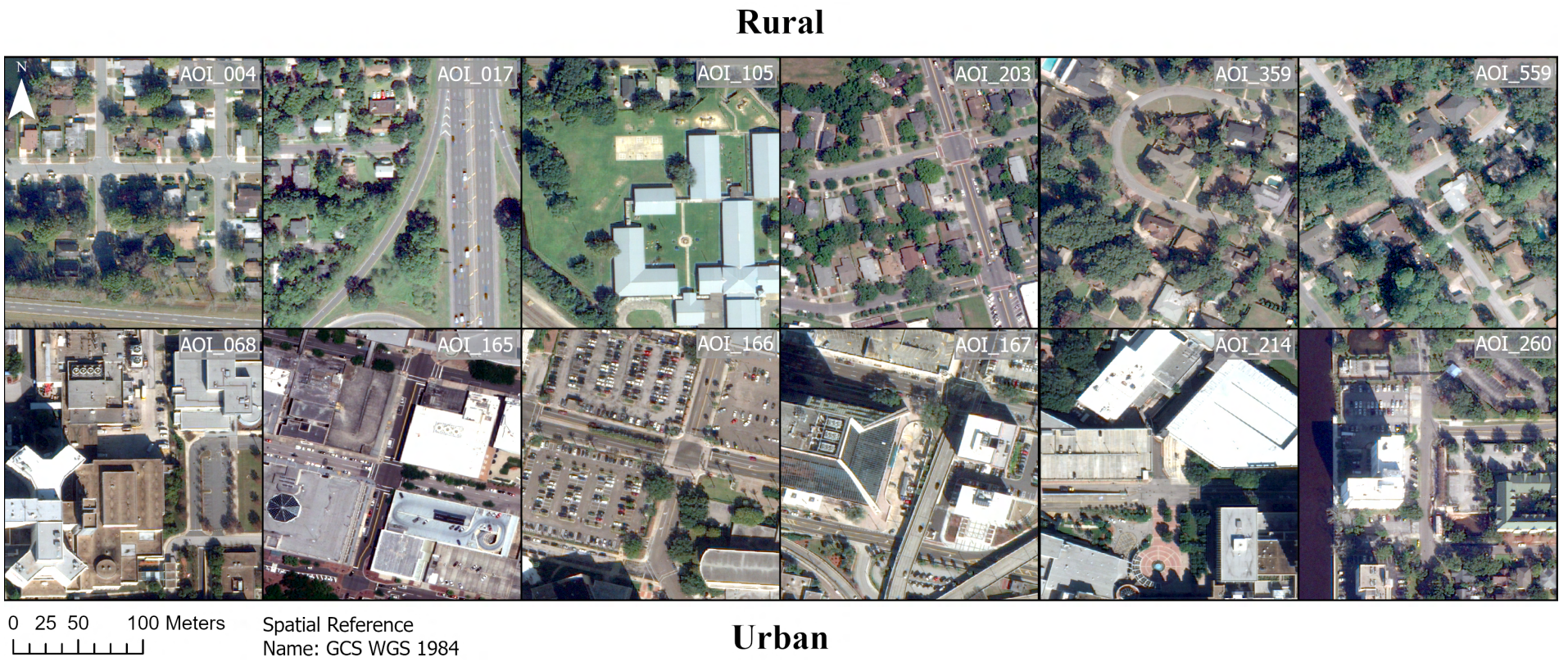


Figure 21: An overview showing the selected urban and rural AOIs of the WorldView-3 satellite imagery from the DFC2019 dataset.

3.2 The Superview-1 dataset

Superview-1, also called the GoaJing-1 constellation, consists of four commercial Chinese satellites that are owned by Beijing Space View Tech Co Ltd. The main goal of the satellites is to capture data that can be used for land and forestry management, high accuracy mapping, maritime research, intelligence and defense. The first 2 satellites were launched in December 2016, while the second 2 satellites were launched in January 2018. The satellites each weigh 560kg and capture approximately 700,000 km^2 of the surface of the Earth every day. They have a sun-synchronous orbit around the Earth at an altitude of 500km phased 90 degrees from one another. The satellites are equipped with two sensors. A panchromatic sensor with a resolution of 0.5m and a multi-spectral sensor that has a resolution of 2m. The multi-spectral sensor captures 4 bands including blue (0.45-0.52μm), green (0.52-0.59μm), red (0.63-0.69μm) and Near-Infrared (0.77-0.89μm). The maximum size of a captured scene is 60km by 70km, and the sensor uses a pushbroom technique (EOS, 2024).

Superview-1 data of the Netherlands is available as open data for the time period 2019-2023 through satellietdataportaal.nl. The data can be downloaded in multiple formats including 2m RGB ortho, 0.5m RGB ortho, 0.5m GIS ready infrared, 0.5 GIS ready RGB and raw imagery. For this research, the raw imagery is downloaded which includes both the panchromatic (0.5m) imagery and RGB imagery (2.0m) in TIF format. However, only the RGB imagery is used to determine the applicability of NeRF. Additionally, an XML file containing metadata for the imagery is provided, as well as an RPC report in an RPB file format. Other datasets available on satellietdataportaal.nl, such as Pleiades-NEO and Superview-NEO were also taken into consideration for this research. However, due to the limited timespan of the available imagery (2023-present), not enough images are available to use for NeRF input. The same problem exists for the mosaic datasets of the Netherlands, which also differ in satellites over time, making them unusable for NeRF (NSO, 2024).

3.3 The AHN dataset

The AHN contains three different endproducts including a LAZ pointcloud, DTM and DSM. The LAZ point cloud is a compressed format of an LAS point cloud and provides the original data. The point cloud includes information on the flight path, point intensity, classification, datetime and more. As described earlier in Section 2.2 the DTM data comprises the elevation of the bare surface of the earth without any buildings or vegetation on it, while DSM does include these objects. Currently, 4 different AHNs are complete, with the 5th version in the making. Table 12 shows the different versions of AHN with their respective collection time, standard error and systematic error. For AHN4 the standard error and systematic error are both 5cm which means that 68.2% of the points have an accuracy of 10cm, 95.4% of the points have an accuracy of 15cm and 99.7% of the points have an accuracy of 20cm. It must be noted that water is excluded from AHN (AHN, 2024). In this research, the AHN 4 dataset is used as the hypothetical LiDAR gt DSM to explore the applicability of NeRF. Although the AHN does not provide a perfect representation of the surface elevation, it is the most accurate data available.

AHN version	Collection time	Standard error (cm)	Systematic error (cm)
AHN1	1996-2002	15	5
AHN2	2007-2012	5	5
AHN3	2014-2019	5	5
AHN4	2020-2022	5	5

Table 12: Table showing the AHN versions, their collection times, standard error and systematic error.

4 Methodology

This section will discuss the methods used to answer the research questions. The methodology chapter is divided into separate sections for each sub research question.

4.1 Optimal NeRF model for DSM generation method (sub research question 1)

To determine the optimal NeRF model to use for this research, a comprehensive literature study is executed. In total, 9 different NeRF models are specifically tailored to create DSMs of satellite imagery. These models are evaluated based on multiple criteria. The most important criterion is the availability of the code. The development of a new NeRF method is out of scope for this research and therefore, an existing model will be used. To use and adjust the existing model, the code has to be available. Secondly, due to limited computing power, computing times are considered for selecting the optimal NeRF model. In general, NeRF is a computer-intensive method and due to the limited timeframe and the limited computing power that is available, it is important to select a NeRF that is efficient and fast. Lastly, the accuracy of the NeRF methods is considered. The desired resulting DSMs should have the highest quality possible for real-life use cases. The selected NeRF model will be used and adjusted to answer the other research questions.

4.2 The accuracy of NeRF method (sub research question 2a and 2b)

The accuracy of the selected NeRF model is tested on both urban and rural environments. The accuracy of NeRF for both environments will be determined using the DFC2019 dataset of Jacksonville, Florida. An overview of the methodology for the accuracy determination is shown in Figure 22.

To derive DSMs of Jacksonville, Florida, the DFC2019 dataset is downloaded from https://ieee-dataport.org/open-access/data-fusion-contest-2019-dfc2019 (Le Saux et al., 2019). As described earlier, the data is made up of a certain number of high-resolution pansharpened RGB Worldview-3 satellite images. Additionally, raw RPCs are obtained which correspond to the full size of the images for each AOI. Lastly, a "Truth" folder is downloaded, the truth folder contains a classified raster, a ground truth (gt) LiDAR raster and a txt file describing the AOI characteristics (number of pixel rows and columns). The AOIs 004, 068, 214 and 260 can be downloaded as a preprocessed dataset, they are already cropped and have bundle adjusted RPCs. The accuracy methodology can be divided into three phases, the data collection, data preprocessing, and the results.

Data collection
In total, 12 AOIs are collected that are all placed in the urban category or the rural category. Four of the AOIs (004, 068, 214 and 260) are preprocessed, while eight are downloaded as unprocessed data. While selecting suitable AOIs, multiple criteria are taken into account. To start, the AOI should fit in the description of an urban scene or a rural scene. The urban category consists of areas with mostly buildings and the rural category consists of suburban areas where more vegetation is present. Secondly, there needs to be a sufficient amount of imagery available, for example, AOI 144 only has 3 images available. Although some NeRFs are optimized to only use a few images for DSM generation (L. Zhang and Rupnik, 2023; L. Zhang et al., 2024), this research required at least 10 images to be available.

Lastly, a visual inspection is performed to check the LiDAR gt DSM and the imagery that is available. It is important to check the imagery to make sure that there are no large transient objects, such as houses that are removed or built during capture. The capture date of the LiDAR gt DSM is unknown and it often shows a scene where buildings are missing although they are present in the imagery, for example.

Data preprocessing
After data collection, the raw DFC2019 data needs to be preprocessed in order to fit the requirements of a NeRF model. To preprocess the data, both the pipeline from bundle adjustment and the pipeline for data preprocessing of SAT-NeRF are used (Marí et al., 2021; Marí et al., 2022). The main code for the process, create_satellite_dataset.py, is shown in Appendix 9.1. Both pipelines are publicly available on GitHub and are optimized to work together. To use the pipelines, multiple python packages need to be installed including: gdal, rpcm, opencv-contrib-python, jupyter, pillow, chardet, matplotlib, numpy, affine, fire, kornia, plyflatten, pyproj, pytorch_lightning, torchmetrics, PyYAML, rasterio, scipy, srtm4, utm, scikit_image and numba. It is important to install the correct version of each package due to dependencies. Sometimes, a package could not be installed using pip, whenever that happens, the alternative conda is used. Additionally, some paths need to be adjusted so that the pipelines can find the right files needed. In addition to changing the paths, adjustments are made to the code to handle data different from the 004, 068, 214, and 260 AOIs of the DFC2019 dataset. The pipelines are executed in the command line using Windows Subsystem for Linux. The first step of the data preprocessing pipeline is to crop the data of each selected scene to a smaller AOI consisting of 512 rows and columns of pixels. The pixels have a size of 0.5m which means that each of the AOIs is 256m x 256m in size. The reduction in size is needed to demand less from the computer hardware. Furthermore, the LiDAR gt DSM and classified raster are only available for the cropped AOIs. After that, the bundle adjustment pipeline is used to refine the RPC models for each AOI. The RPC models are saved in a JSON file combined with image metadata, which includes: the image height and width (number of pixels), sun elevation, sun azimuth, acquisition date, corner coordinates of the AOI, center coordinates of the AOI, minimum altitude and maximum altitude. An example of such a JSON file (AOI 004) is given in Appendix 9.2. After bundle adjustment, the pipeline randomly selects a few images (two or three) for testing and creates a test.txt file listing them. The other images are listed in a train.txt file and are used for training the NeRF model. Altogether, the pipelines produce cropped satellite images as TIF files, JSON files containing metadata and bundle adjusted RPCs, as well as test and train txt files containing lists of the cropped images.

Results

The outputs of the preprocessing pipelines are used as an input to run the NeRF model that is selected during sub research question 1. Additionally, the LiDAR gt DSM and the classified raster file, together with the txt file specifying the AOI are used. To apply the selected NeRF model to data different from the 004, 068, 214 and 260 AOIs, some adjustments need to be made. For example, in the utils.py code, in the predefined_val_ts function, the test images for each AOI need to be added to assign a certain value. The utils.py code is provided in Appendix 9.3. The model is executed through the command line in Windows Subsystem for Linux with a standard configuration, a batch size of 1024 and 60,000 iterations. The batch size stands for the number of rays that are passed through the model in a single optimization step. While the number of iterations refers to the number of optimization steps that the model goes through during training.

Three different outputs are evaluated after the model is run, a DSM, a difference DSM and the error metrics. The error metrics include the Mean Absolute Error (MAE), Mean Error (ME) and Maximum Error (Max E). The DSMs have a pixel resolution of 0.5m and provide insight on how the model performs for different areas in the AOI. The difference DSM provides the elevation deviation of the predicted NeRF DSM compared to the LiDAR gt DSM. Through the difference DSM, areas can be identified where the NeRF model predicts poorly or particularly well. Furthermore, the MAE provides the Mean Absolute Error. This metric gives the mean deviation of the NeRF predicted DSM compared to the LiDAR gt DSM and normalizes negative and positive deviations. The MAE is used as the main performance index to evaluate the urban and rural NeRF predicted DSMs. The other error indexes ME and Max E, provide the average deviation without normalizing positive and negative values, and the maximum deviation respectively. Lastly, using the NeRF DSM and LiDAR gt DSM, elevation profiles are created of two AOIs for both the urban and rural groups. The elevation profiles show both the NeRF predicted DSM and LiDAR gt DSM for each AOI. These evaluations contribute to answering subquestions 2a and 2b concerning the accuracy of NeRF in both urban and rural environments.

Figure 22 shows a schematic overview of the pipeline to answer sub research questions 2a and 2b considering the accuracy of NeRF.

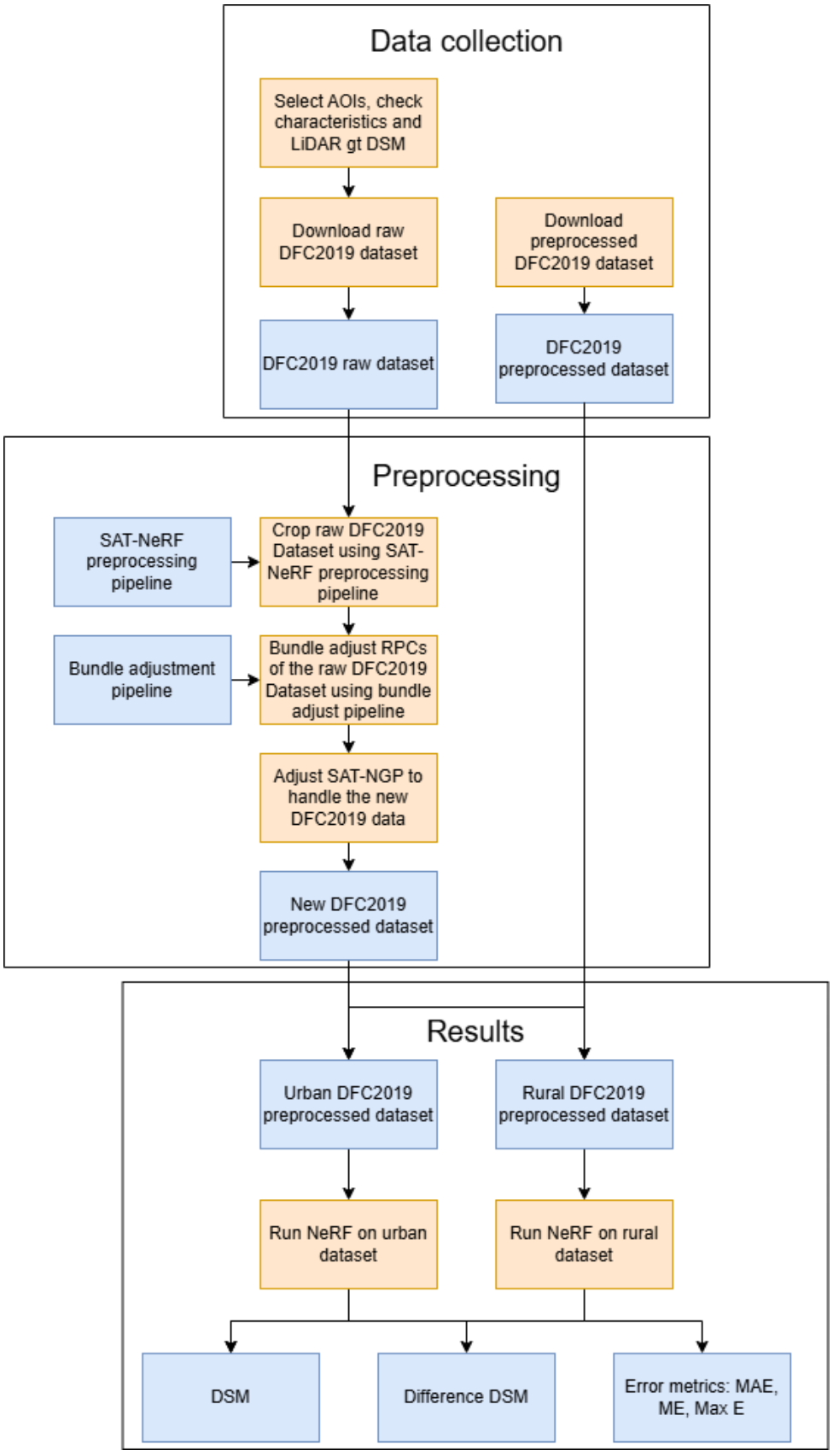


Figure 22: Overview of the methodology considering the accuracy of NeRF (sub research questions 2a and 2b).

4.3 Sensitivity of NeRF method (sub research question 3)

To determine the sensitivity of the selected NeRF model, two parameters are adjusted. The effect of the number of training iterations and the effect of the batch size of rays for each training step are assessed. The exact values of the parameters differ depending on the NeRF model that is selected in sub research question 1. Both parameters are tested with 11 different configurations. For each variation, the other parameters are kept at their defaults and the effects are investigated on MAE values and runtime. The sensitivity is tested on the AOI 068 comprising an urban environment and on AOI 359 comprising a rural environment. Testing on the two different environments allows to investigate sensitivity differences between them. The analysis shows how the adjustment of a parameter influences the accuracy of the derived DSM. A schematic overview of the sensitivity analysis pipeline is shown in Figure 23.

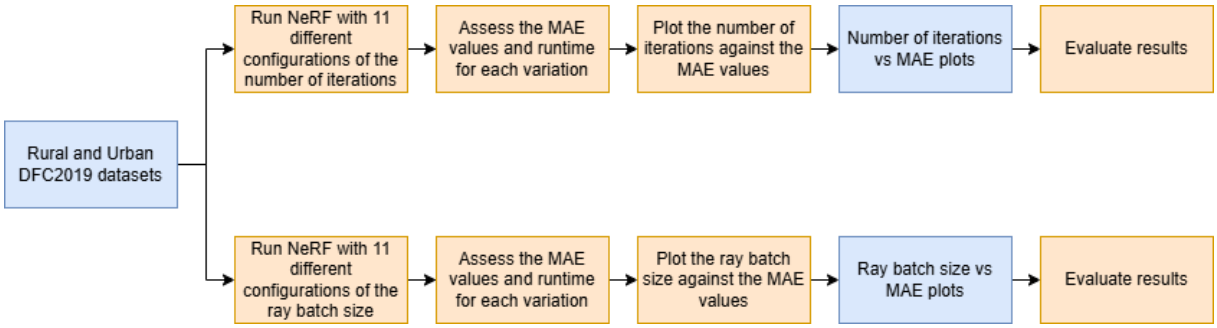


Figure 23: Pipeline to determine the sensitivity of the model considering the amount of rays and number of iterations.

4.4 Applicability of NeRF method (Sub reseach question 4)

To determine the applicability of state-of-the-art NeRF to data different from DFC2019, a study is conducted to determine what is needed for such a model to run. To examine this, three different study areas are selected, Delft, Den Haag and the golfcourse "Het Rijk van Margraten" near Maastricht. Figures 24, 25 and 26 show the research areas of Den Haag, Delft and the golfcourse respectively. The Delft and Den Haag study areas are selected due to their usage in Claesen (2024). While the Golfcourse study area is selected due to its abundance of green, which stays stable over time. For these study areas, Superview-1 satellite imagery is downloaded with the accompanying metadata and RPC files as described in Section 3.1.

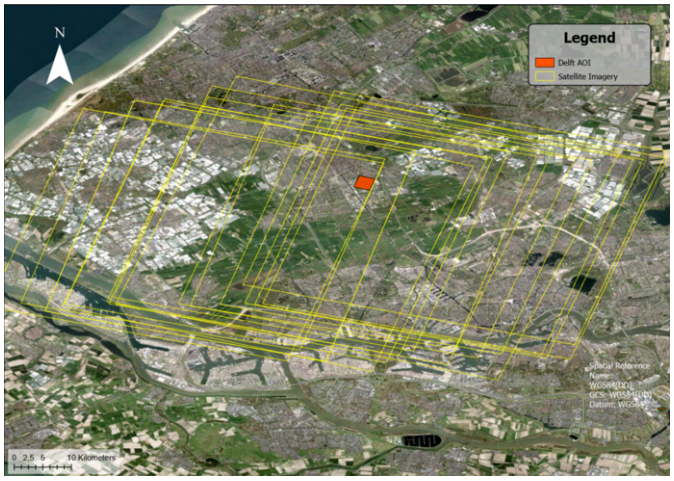


Figure 24: Map showing an overview of the satellite imagery and the location of the Delft AOI.

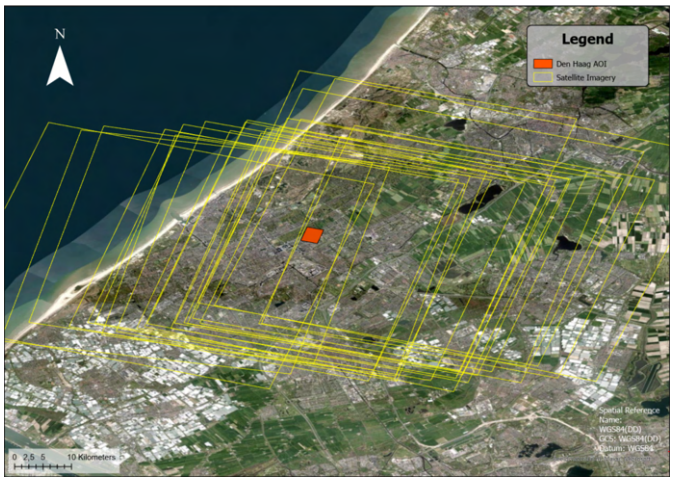


Figure 25: Map showing an overview of the satellite imagery and the location of the Den Haag AOI.

Figure 26: Map showing an overview of the satellite imagery and the location of the Golfcourse AOI.

After downloading, it is investigated what is needed to preprocess the data in such a way that it can be used as an input for NeRF. Additionally, the AHN data is used as the hypothetical LiDAR gt DSM. The first step is to determine how the imagery can be cropped. Next, the RPCs and metadata need to be adjusted and combined into one JSON file for each image. After that, the LiDAR gt DSM, classified raster TIF file and the txt file with the AOI need to be prepared. Through these steps, together with the literature research of research question 1, it is assessed what the applicability of NeRF is, and what is needed to run the model with data different from DFC2019. A schematic overview of the applicability method is shown in Figure 27

Figure 27: Pipeline to determine the applicability of a state-of-the-art NeRF model.

5 Results

This section presents the results achieved in this research. Maps, tables, graphs, and schematic overviews are produced to illustrate the results. The results are discussed in the order of the sub research questions. Starting with the model selection, followed by the results on the accuracy of the selected NeRF model. After that, the sensitivity analysis is presented, and finally, the applicability of NeRF is covered.

5.1 NeRF Model selection

To select the optimal NeRF model to create DSMs from satellite imagery, multiple criteria are considered. Firstly, the code needs to be available, because the time frame does not allow a NeRF model to be built from scratch. This criterion already eliminates some of the NeRF models including RS-NeRF, SatensoRF, GC-NeRF, and EO-NeRF (Marí et al., 2023; Wan et al., 2024; Xie et al., 2023; T. Zhang et al., 2024). In addition to that, the models are evaluated in terms of accuracy and training time. Higher accuracy is preferred to create more accurate models, while lower training times are preferred due to time and hardware constraints of this research.

Through thorough literature analysis, SAT-NGP is identified as the optimal option for this research. SAT-NGP has its code openly available on GitHub. On top of that, the model has a training time of 12 minutes on average for the 004, 068, 214 and 260 AOIs of the DFC2019 dataset, using a 12GB GPU (Billouard et al., 2024). Sat-NGP is only outperformed by RS-NeRF in terms of training times (RS-NeRF does not have code available), while still achieving some of the highest accuracies compared to other NeRFs (Billouard et al., 2024; Xie et al., 2023). As mentioned in Section 2.6.8, SAT-NGP has a lower MAE in 14 minutes compared to Sat-NeRF in 12 hours for AOI 214. The SAT-NGP MAEs are 1.31, 2.03, 2.17m and 1.68m for the AOIs 004, 068, 214 and 260, respectively. Due to the hardware that is available for this research, a T1200 NVIDIA GPU with 4GB of memory, model efficiency is a high priority. Additionally, reduced training times are useful when applying NeRF to create DSMs of larger areas, which might be useful for large-scale case studies. Where other models have shortcomings concerning code availability, computing times and accuracy, SAT-NGP allows for scalability while maintaining a state-of-the-art accuracy. Figure 28 is adjusted from Figure 11 and now indicates the final model that is selected and the models on which it is based. The figure shows that SAT-NGP is built on S-NeRF and Sat-NeRF, adopting their shading model and network architecture. The methods that are added to SAT-NGP include Instant-NGP, a robust loss function and MISH activation. Instant-NGP improves the speed of the model through multi-resolution hash encoding. While the robust loss function removes transient objects from the scene such as cars. Lastly, MISH activation provides improved performance compared to ReLU and SIREN activations. The methods are explained in more detail in Sections 2.6.5 (Instant-NGP), and 2.6.8 (robust loss function and MISH activation).

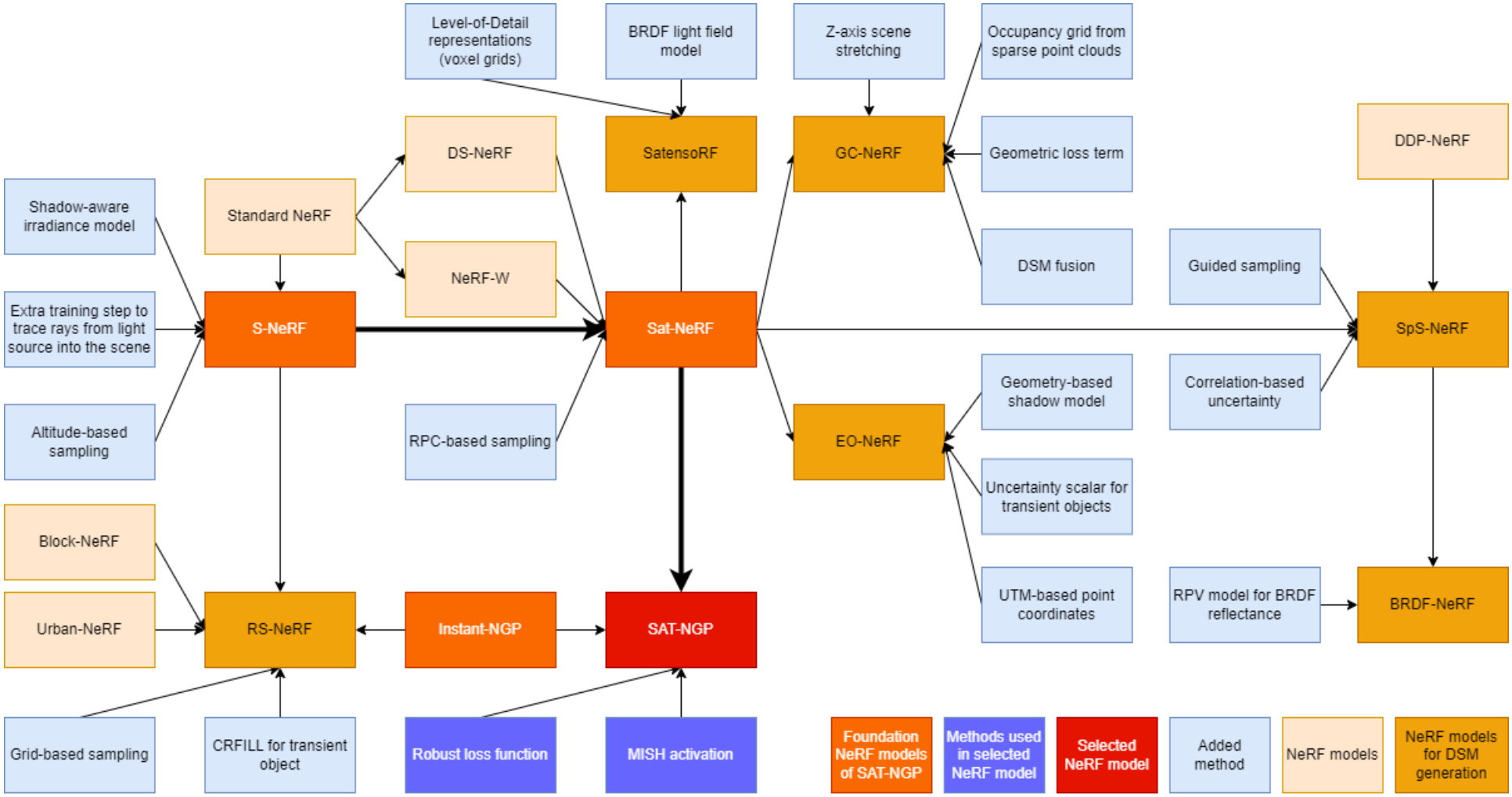


Figure 28: An overview of the NeRF models that are adjusted to create DSMs from satellite imagery, with their respective added methods. SAT-NGP is selected from all of the models and is based on S-NeRF, Sat-NeRF and Instant-NGP, with the robust loss function and MISH activation as added methods. SAT-NGP is the only NeRF model applied in this research, however, some code is borrowed from Sat-NeRF for preprocessing purposes.

5.2 DSM accuracy results

This section discusses the accuracy results of the SAT-NGP model for both urban and rural scenes by comparing MAE, ME and Max E values, in addition to presenting the DSMs and difference DSMs. In addition, elevation profiles are created for two urban and two rural AOIs to highlight the performance of NeRF for different features. For each research area, the standard configuration of SAT-NGP is used. The standard configuration has a batch size of 1024 rays, including 4 to 256 samples for each ray and 60,000 iterations. The model took approximately 45 minutes for each AOI on a laptop with 32GB of RAM and an NVIDIA T1200 4GB graphics card.

5.2.1 Urban accuracy results

Using SAT-NGP, DSMs are created of the urban AOIs 068, 165, 166, 167, 214 and 260 of the DFC2019 dataset. The DSMs are shown in Figure 29. The DSMs show a high level of detail, especially for AOIs 068, 165 and 214. AOI 166 shows two interesting artifacts with increased elevation, which do not seem to be present in the satellite imagery or the LiDAR ground truth (gt) DSM. Note that water features and construction sites are removed from the final DSMs.

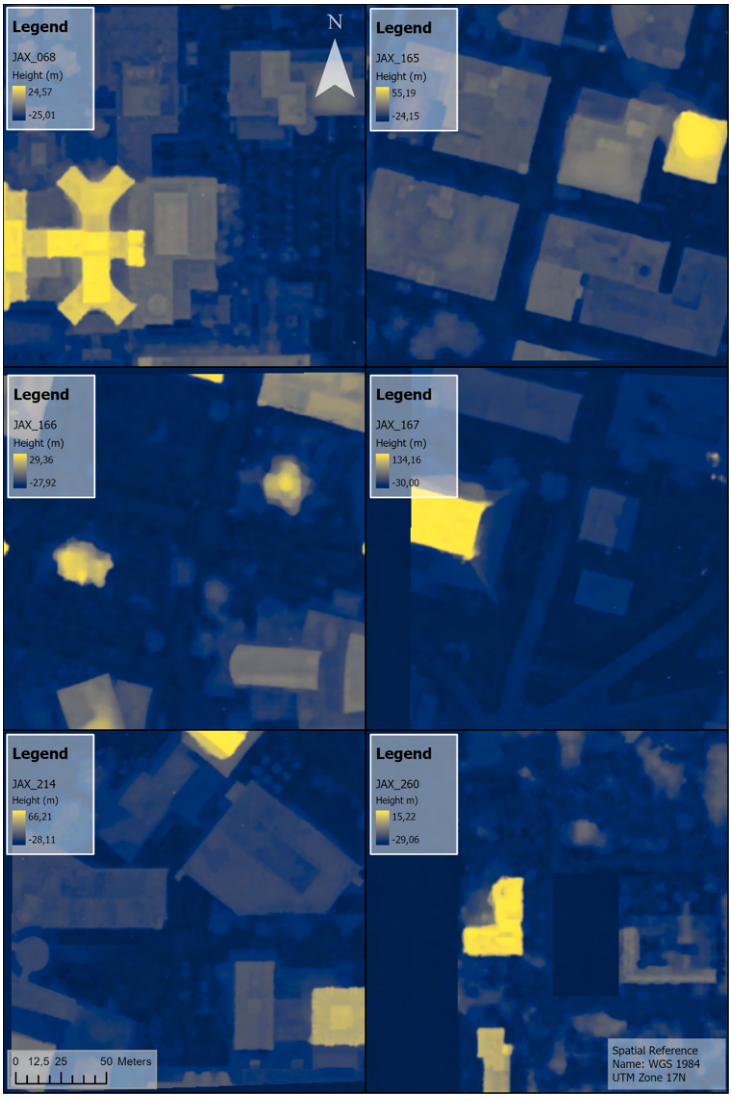


Figure 29: Maps showing the DSMs of AOIs 068, 165, 166, 167, 214 and 260, created using SAT-NGP.

In addition to the DSMs, difference DSMs are created for each AOI, which are shown in Figure 30. The difference DSMs are the result of subtracting the NeRF predicted DSM by the LiDAR gt DSM. For comparison reasons, the AOIs all have the same symbology. It must be noted that some deviations are larger than 50m, however, for the sake of detail, the current symbology is used. Again, the two artifacts in AOI 166 are clearly visible, indicating a large deviation from the LiDAR gt DSM. In general, the other AOIs show that there are some large errors near the edges of buildings, where the height is either underestimated (blue) or overestimated (red).

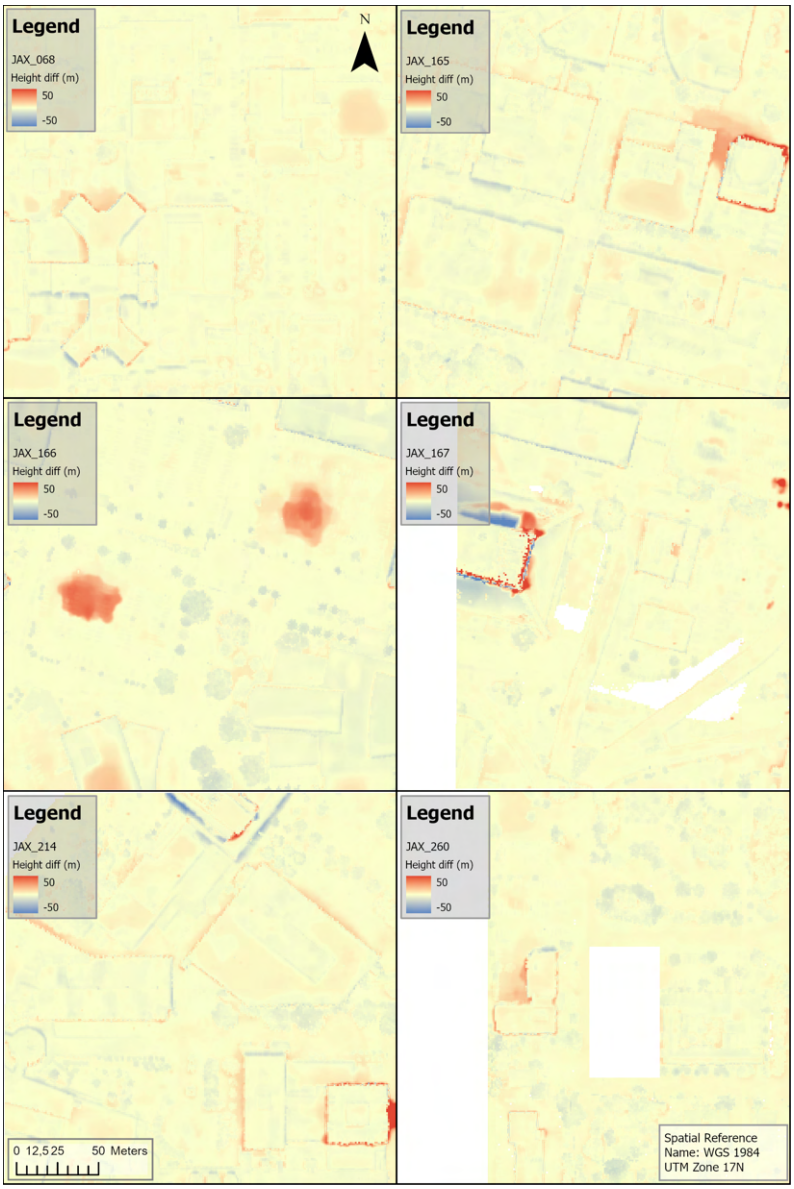


Figure 30: Maps showing the difference DSMs of AOIs 068, 165, 166, 167, 214 and 260.

To provide a clear overview of the imagery, LiDAR gt DSM, NeRF predicted DSM, difference DSM and MAE, Figure 31 is created. For comparison reasons, the height scale is the same for the LiDAR gt DSM and the predicted DSM for each AOI separately. Additionally, the difference DSMs have the same scale throughout all of the AOIs, ranging between -50 and 50m. The figure shows that NeRF has difficulties with edges compared to the LiDAR gt DSM, resulting in errors. Furthermore, earlier mentioned artifacts in AOI 166 are clearly visible.

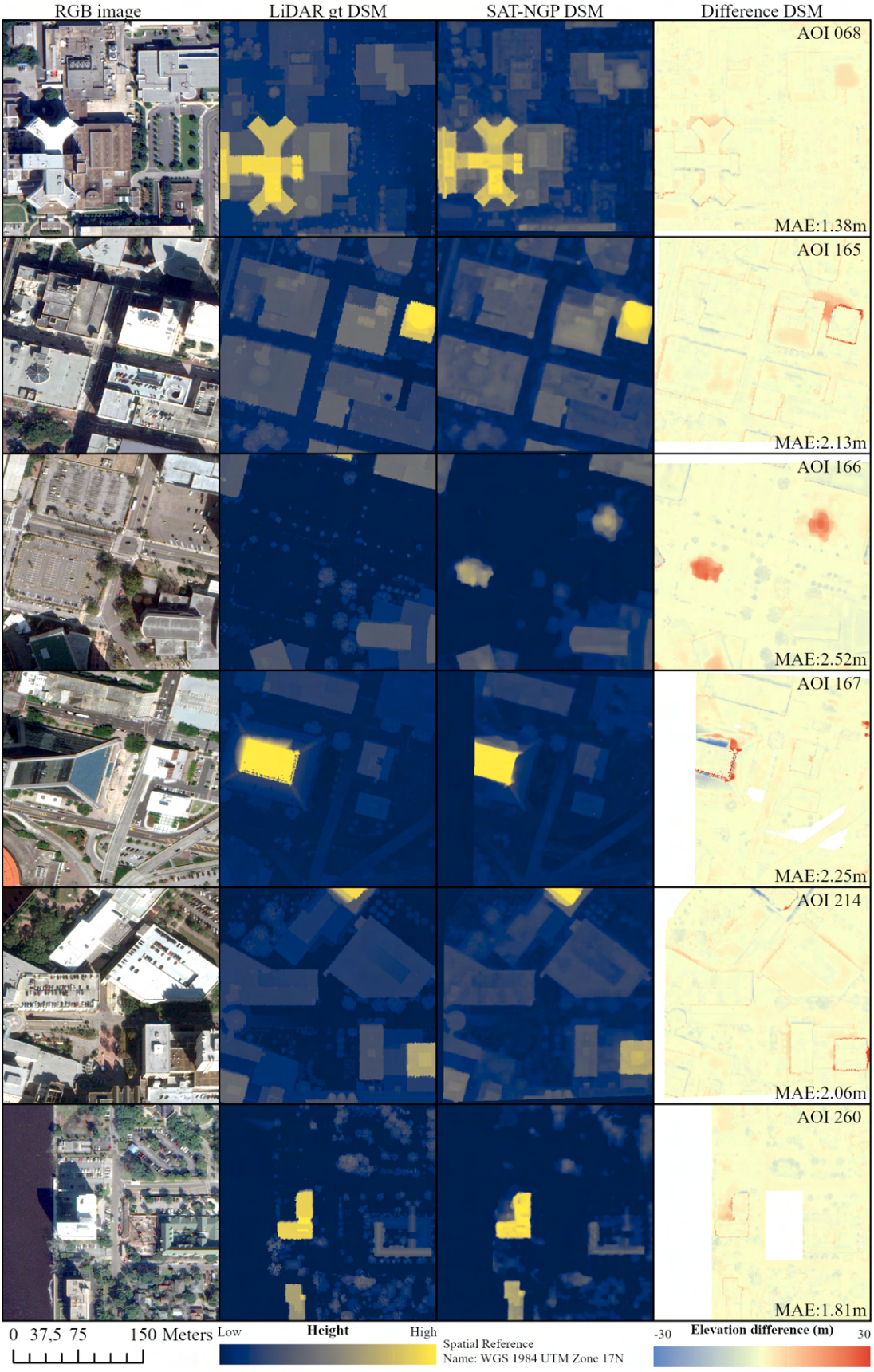


Figure 31: Overview of the urban AOIs, showing the satellite image, LiDAR gt DSM, NeRF predicted DSM, Difference DSM and the MAE in meters.

To highlight the performance of NeRF in different areas, Figure 32 is created, showing elevation profiles of the 068 and 166 AOIs. The elevation profile of the 068 AOI shows that the SAT-NGP DSM is relatively accurate and lies close to the LiDAR gt DSM. However, at a distance of 60m there are some larger deviations. Additionally, there are some small offsets near the edge of the building at 30m distance. The small offset results in high errors due to the height of the building, which is around 30m. The 166 AOI elevation profile shows the large DSM deviations in the parking lot around 50m distance and 200m distance. In general, both elevation profiles also show that the NeRF predicted DSM is smoother compared to the LiDAR gt DSM, which has sharper edges.

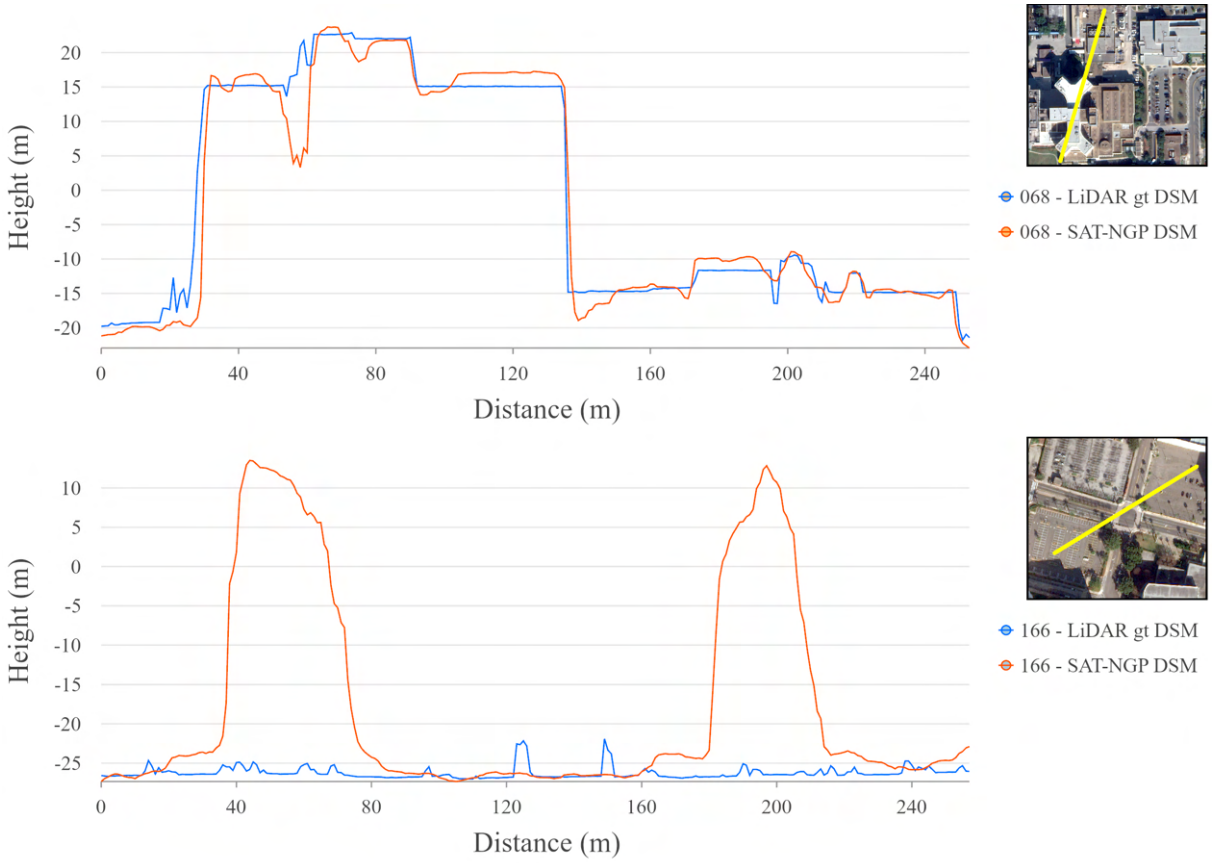


Figure 32: Elevation profiles of the 068 and 166 AOIs. The LiDAR gt DSM is shown in blue and the SAT-NGP DSM is shown in red. Note that the height scale differs between the AOIs.

5.2.2 Rural accuracy

The resulting DSMs of the rural AOIs 004, 017, 105, 203, 359 and 559, are shown in Figure 33. Again, the water features are clipped from the DSM. The rural DSMs look less sharp compared to the urban DSMs. AOI 017 has a highway that runs through the scene, which should be smooth. However, the DSM shows interesting features on the road that should not be there. In general, the elevation of the road is too high. AOI 105 also has some interesting features. Although the fields are flat in the middle left area of the scene, the yellow areas indicate that the fields are elevated in the NeRF predicted DSM compared to the LiDAR gt DSM. Note that the AOIs have different symbology for the height to maintain detail, so they cannot be compared to each other.

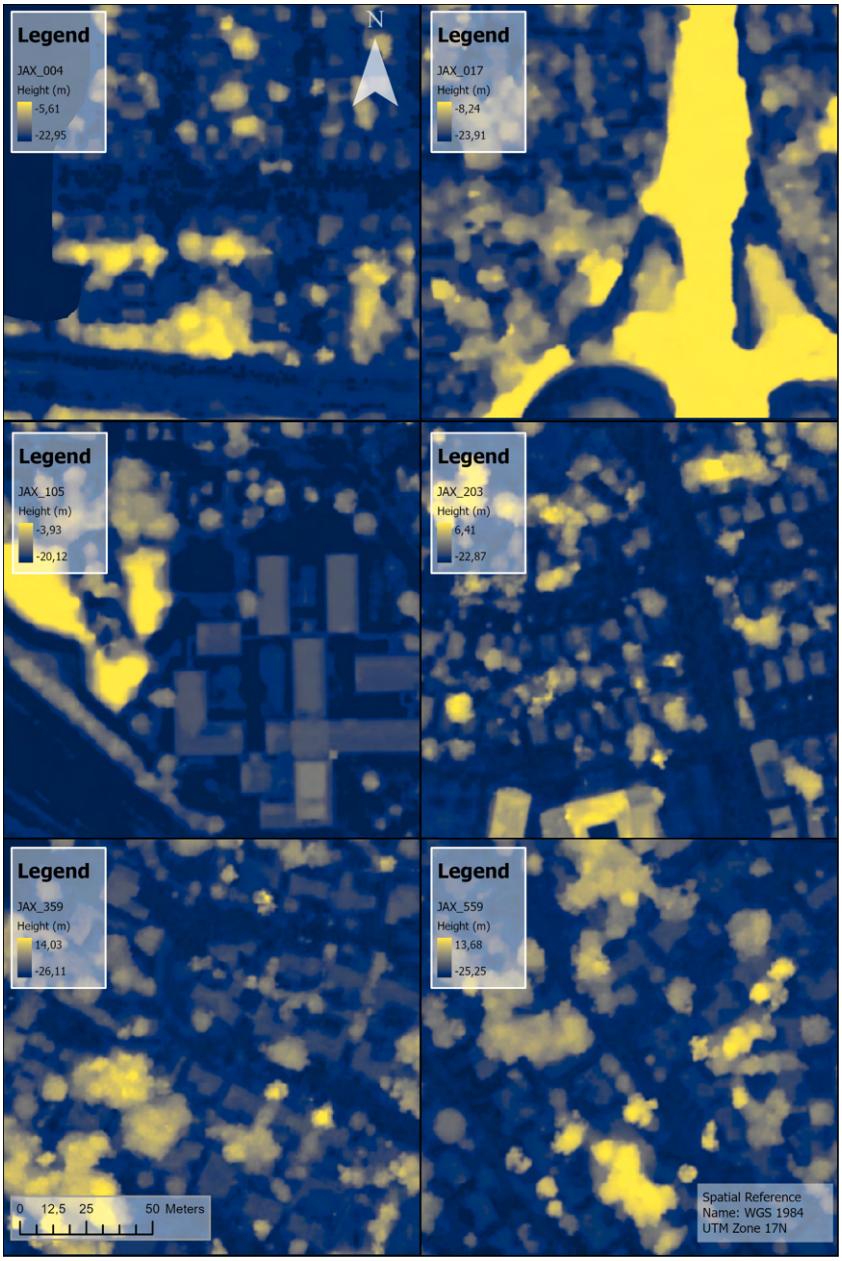


Figure 33: Maps showing the NeRF predicted DSMs of AOIs 004, 017, 105, 203, 359 and 559.

Figure 34 shows the difference DSMs for the rural AOIs. AOIs 017 and 105 stand out with large red areas, indicating an overestimation of the height by the NeRF model. For AOI 017, the entire highway is overestimated, while for AOI 105, the fields are overestimated. In addition to the fields, there is also a train visible in the error figure of AOI 105 in the bottom left (blue line). In general, many trees are visualized in blue, indicating an underestimation of their height. Additionally, the edges of trees are sometimes difficult to estimate for the NeRF model, as can be seen in AOIs 004, 203, 359 and 559 where trees often have a dark red line around them. Lastly, some thin lines of error can be seen in the AOIs 004, 017 and 203. These lines are suspended cables in the air that are ignored by NeRF but not by the LiDAR scanner.

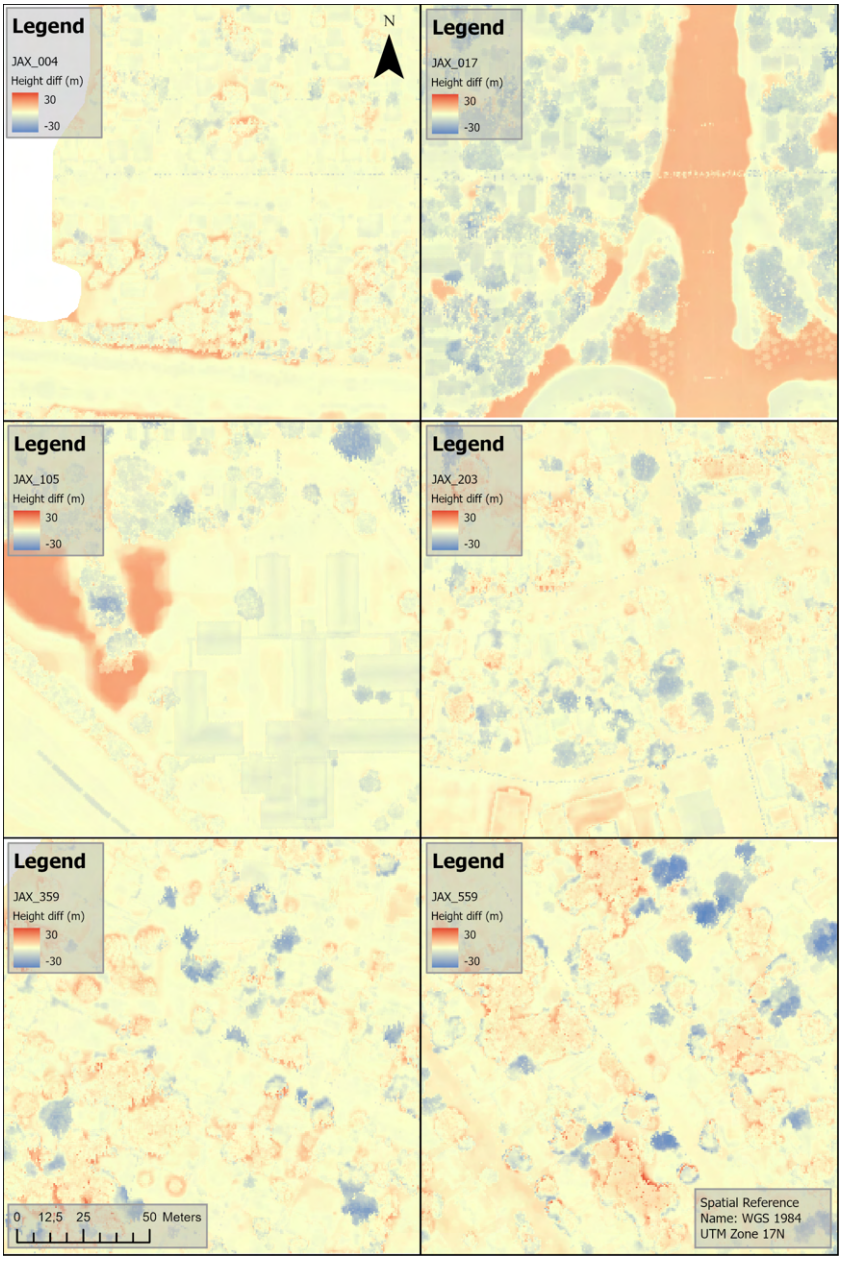


Figure 34: Maps showing the difference DSMs of AOIs 004, 017, 105, 203, 359 and 559.

Figure 35 shows an overview of the rural AOIs, with RGB imagery, LiDAR gt DSM, predicted DSM, difference DSM and MAE values. Again, it must be noted that the difference DSMs have the same scale, while the LiDAR gt DSM and predicted DSM only have the same scale for the same AOIs. The figure shows that the NeRF predicted DSM is often less sharp. The large errors of AOIs 017 and 105 are also clearly visible between the DSMs. Additionally, tree removal can be seen by the reduction of yellow dots in AOIs 004, 359 and 559.

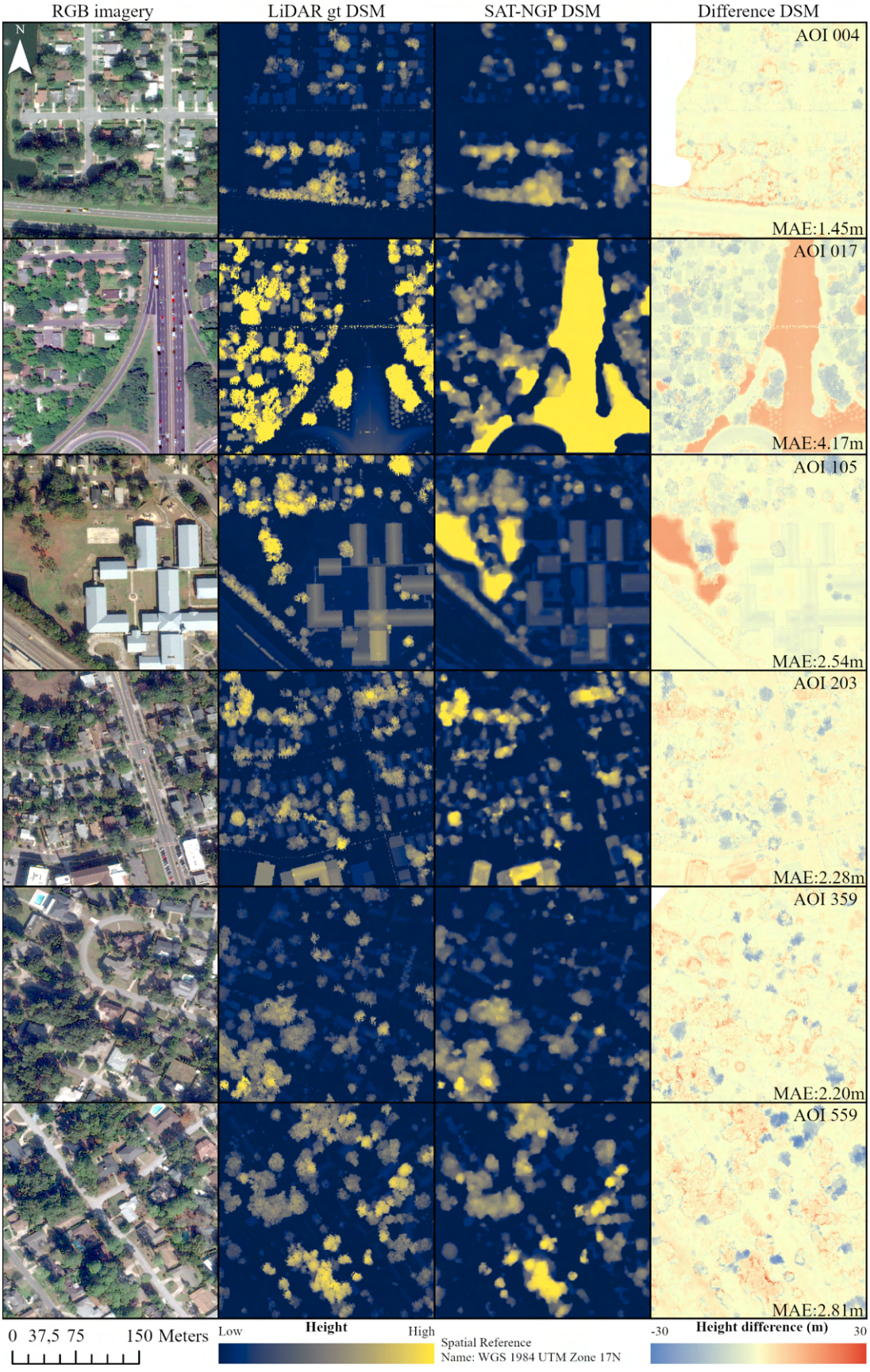


Figure 35: Overview of the rural AOIs, showing the satellite image, LiDAR gt DSM, NeRF predicted DSM, Difference DSM and the MAE in meters.

Figure 36 shows the elevation profiles of AOIs 017 and 105. The 017 AOI has the highest MAE value (4.71) of all AOIs, and the elevation profile clearly visualizes the large errors. The highway between 160 and 200 is correctly interpreted as a flat surface. However, the elevation is estimated to be higher compared to the vegetation on the sides, which is not the case in real life. Additionally, the SAT-NGP model also performs poorly on the left side of the image, where large errors are located. The elevation profile of AOI 105 also clearly shows the errors of the SAT-NGP DSM. The elevation of the flat fields at the distances of 0-40m and 80-140m is severely overestimated. In addition, the removal of trees is shown between 50-80m. Lastly, the SAT-NGP model also fails to predict the angled roofs at 160-180m and 200-220m, where it estimates a flat surface.

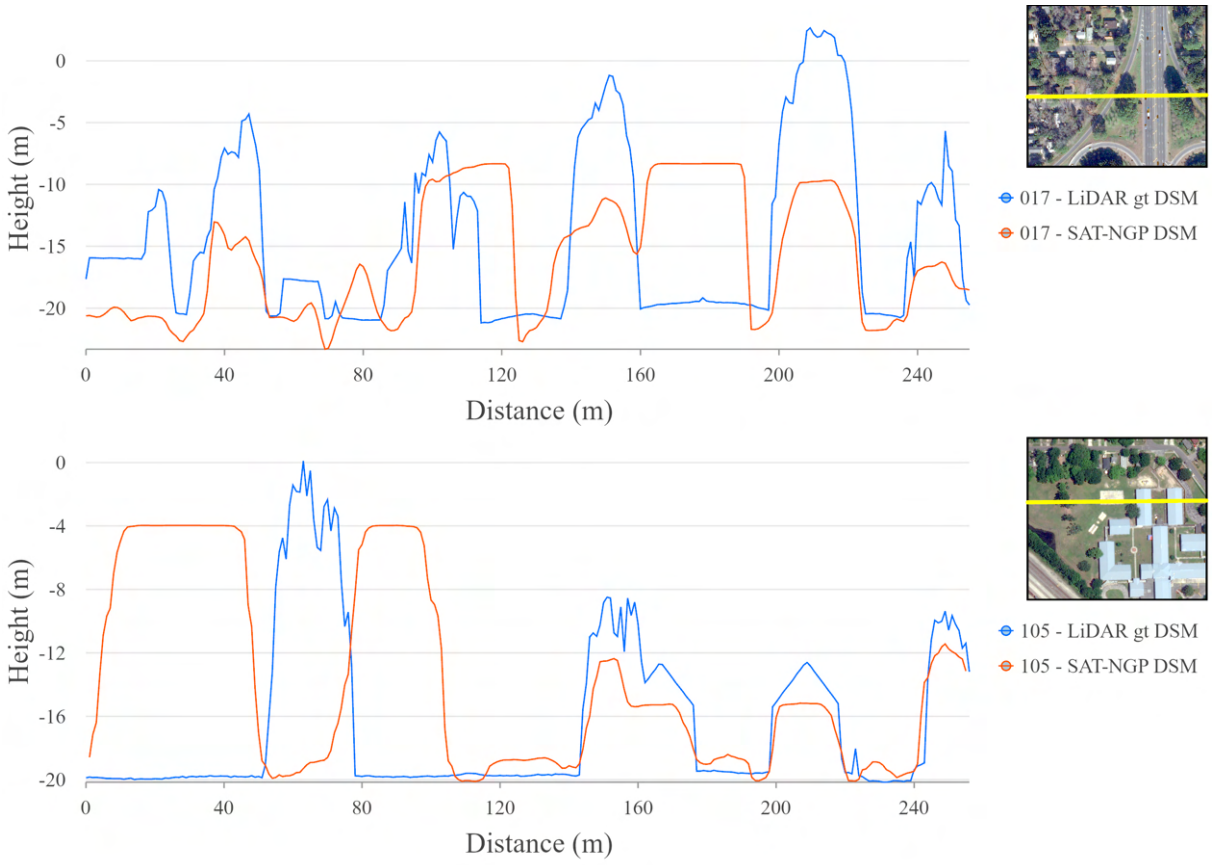


Figure 36: Elevation profiles of the 017 and 105 AOIs. The LiDAR gt DSM is shown in blue and the SAT-NGP DSM is shown in red.

5.2.3 Accuracy comparison of urban and rural scenes

The MAE results are shown in Table 13, where a distinction is made between urban and rural datasets. Additionally, the number of images, the Mean Error (ME) and Maximum Error (Max E) are provided.

Urban Scenes	# images	MAE (m)	ME (m)	Max E (m)	Rural Scenes	# images	MAE (m)	ME (m)	Max E (m)
068	11	**1.38**	1.04	35.27	**004**	11	**1.45**	1.35	16.85
165	23	2.13	9.71	72.15	017	10	4.71	2.94	22.17
166	24	2.52	-4.78	47.12	105	21	2.54	9.57	20.42
167	24	2.25	6.33	126.06	203	24	2.28	1.24	20.83
214	24	2.06	1.80	65.12	**359**	24	**2.20**	3.54	29.10
260	17	**1.81**	-1.20	33.65	559	23	2.81	9.51	31.96

Table 13: MAE, ME and Max E values in meters and the number of images that are used for the urban and rural scenes. The two lowest MAE values are in bold for both the urban and rural scenes.

Table 13 shows that the SAT-NGP model, on average, has a higher MAE value for urban scenes compared to rural scenes. The average MAE for urban scenes amounts to 2.03m, while the average MAE for rural scenes amounts to 2.67m. Another interesting thing to note is that the four lowest MAEs are AOIs 004, 068, 214 and 260. These are the AOIs that are used in almost every research involving DSM creation from satellite imagery using NeRf. The scene with the highest accuracy is the urban 068 scene, while the lowest accuracy is achieved for the rural 017 scene. The number of images is provided to give insight and determine if there is a relation between the number of images and the MAE. Figure 37 shows a plot of the MAE and the corresponding number of images. From the figure, it can be seen that no clear trend is present. Although the MAE values are lower for urban scenes compared to rural scenes, the Max E is much larger. The maximum negative and positive deviations from the ground truth are much higher for urban scenes. 3D visualizations of the DSMs are shown in Appendix 9.4. Furthermore, the ME shows that, for the rural AOIs, the SAT-NGP DSM is higher on average. The urban scenes do not show the same trend, as AOIs 166 and 260 have a negative mean error.

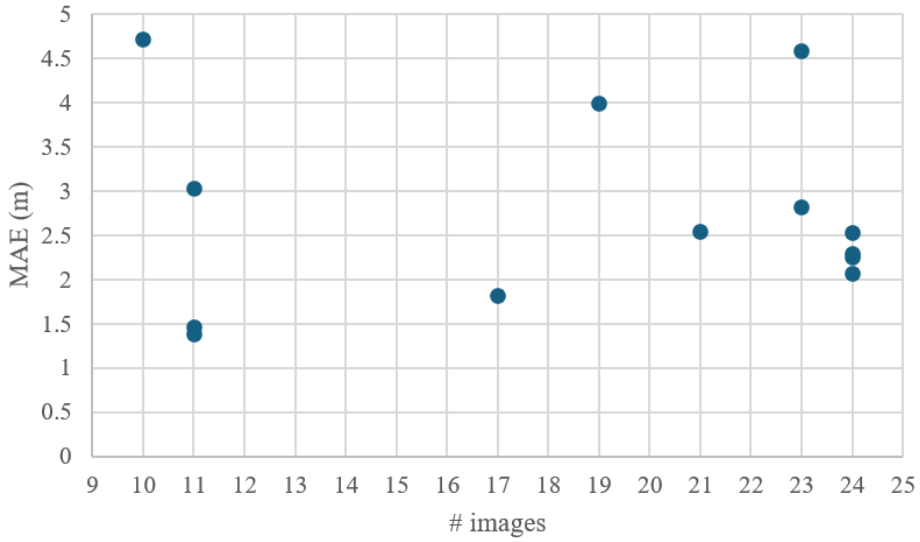


Figure 37: Graph showing the MAE plotted against the number of images.

5.3 NeRF sensitivity results

This section discusses the results of the sensitivity analysis for SAT-NGP, comprising both the number of iterations and the batch size. The sensitivity analysis is performed on both the urban AOI 068 and the rural AOI 359. This allows investigating differences between urban and rural scenes in terms of the sensitivity of SAT-NGP.

5.3.1 Number of iterations sensitivity results

Table 14 shows the number of iterations, the corresponding MAE and the runtime of urban AOI 068 and rural AOI 359. In addition, a graph is provided in Figure 38, showing the number of iterations versus the corresponding MAE on the primary axis (left) and the number of iterations versus the corresponding runtime on the secondary axis (right).

AOI	Iterations	MAE (m)	Runtime (s)
068	1000	1.91	428
068	6000	1.66	421
068	15000	1.44	811
068	30000	1.42	1271
068	45000	1.38	1875
068	60000	1.38	2239
068	75000	1.45	2959
068	90000	1.44	3596
068	105000	1.46	4124
068	120000	1.43	4339
068	135000	1.43	5439
068	150000	1.41	5531
359	1000	2.43	502
359	6000	2.27	523
359	15000	2.16	949
359	30000	2.20	1322
359	45000	2.15	1761
359	60000	2.20	2268
359	75000	2.18	3181
359	90000	2.18	3735
359	105000	2.18	4519
359	120000	2.18	4659
359	135000	2.21	4743
359	150000	2.17	5823

Table 14: The MAEs and runtime of the SAT-NGP model for each configuration of the number of iterations and AOI.

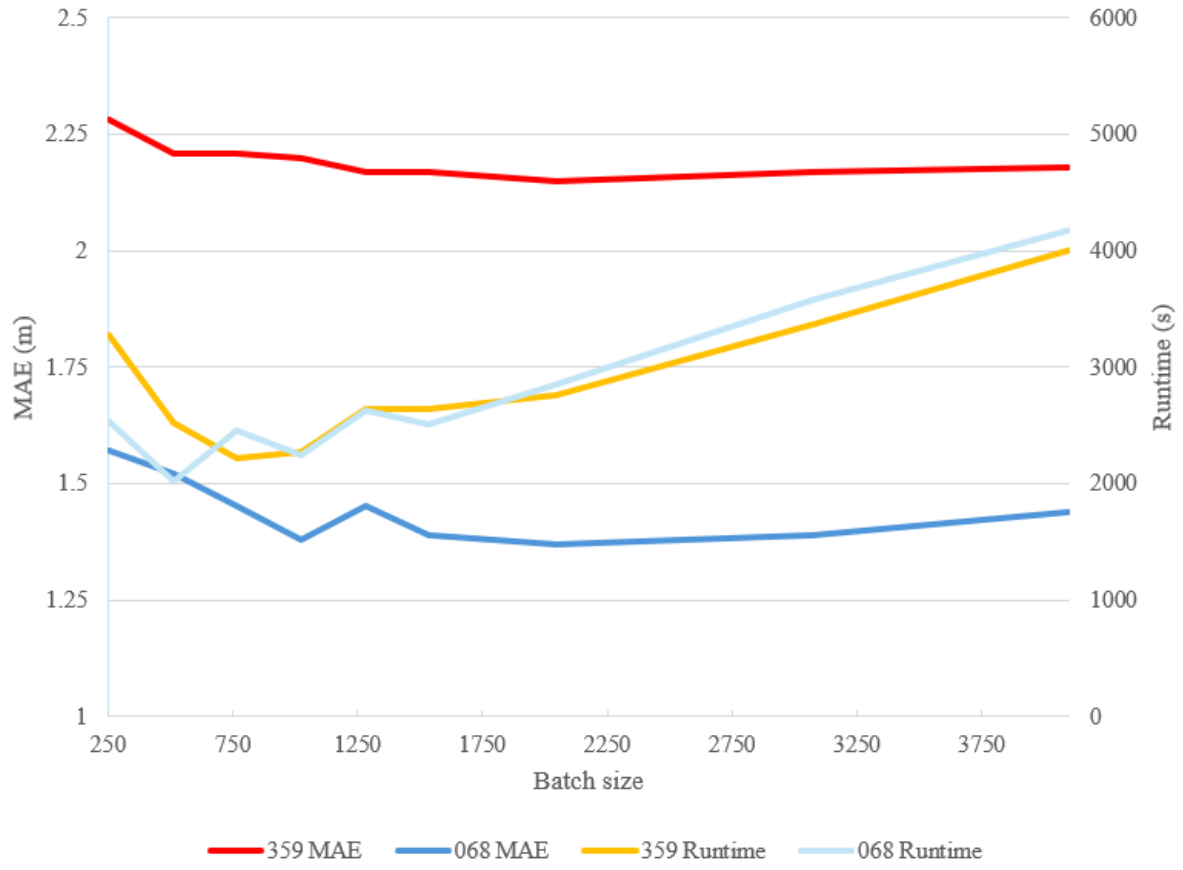


Figure 38: Graph showing the MAE versus the number of iterations on the primary axis (left) and the runtime versus the number of iterations on the secondary axis (right), for both the 068 AOI and the 359 AOI.

Table 14 and Figure 38 show that the MAE decreases when less than 30,000 iterations are used for the SAT-NGP model. This trend is similar between the urban 068 AOI and the rural 359 AOI. When more than 30,000 iterations are used, the MAE does not seem to improve. The highest MAE values are achieved using 45,000 iterations for the rural dataset. While the highest MAE value is the same for 45,000 iterations and 60,000 iterations for the urban dataset. An interesting observation is that the runtime linearly increases with the number of iterations. These results are further evaluated in the Discussion section of this research.

5.3.2 Batch size sensitivity results

The results of the batch size sensitivity analysis are provided in Table 15 and Figure 39. It should be noted that the batch size is capped at 4096 for SAT-NGP. Table 15 shows the AOI, the batch size, and the corresponding MAE and runtime. In addition, Figure 39 shows a plot of the table with the MAE versus the batch size on the primary axis (left) and the runtime versus the batch size on the secondary axis (right).

Area	Batch size	MAE (m)	Runtime (s)
068	256	1.57	2536
068	512	1.52	2020
068	768	1.45	2455
068	1024	1.38	2239
068	1280	1.45	2625
068	1536	1.39	2501
068	2048	1.37	2853
068	2560	1.42	3278
068	3072	1.39	3578
068	3584	1.37	4249
068	4096	1.44	4178
359	256	2.28	3272
359	512	2.21	2515
359	768	2.21	2215
359	1024	2.20	2268
359	1280	2.17	2634
359	1536	2.17	2640
359	2048	2.15	2757
359	2560	2.17	3182
359	3072	2.17	3368
359	3584	2.15	3777
359	4096	2.18	3996

Table 15: The MAEs and runtime of the SAT-NGP model for each batch size and AOI.

Table 15 and Figure 39 show that on average the runtime increases with increasing batch size from 768 upward, for both the 068 and 359 AOIs. However, both AOIs also have an increased run-time when the batch size is lower than 512. Concerning MAE, it seems that there is little change when changing the batch size, as both lines are relatively horizontal. Both AOIs show their lowest MAEs for a batch size of 2048 and the highest for a batch size of 256. The results are further evaluated in the Discussion section.

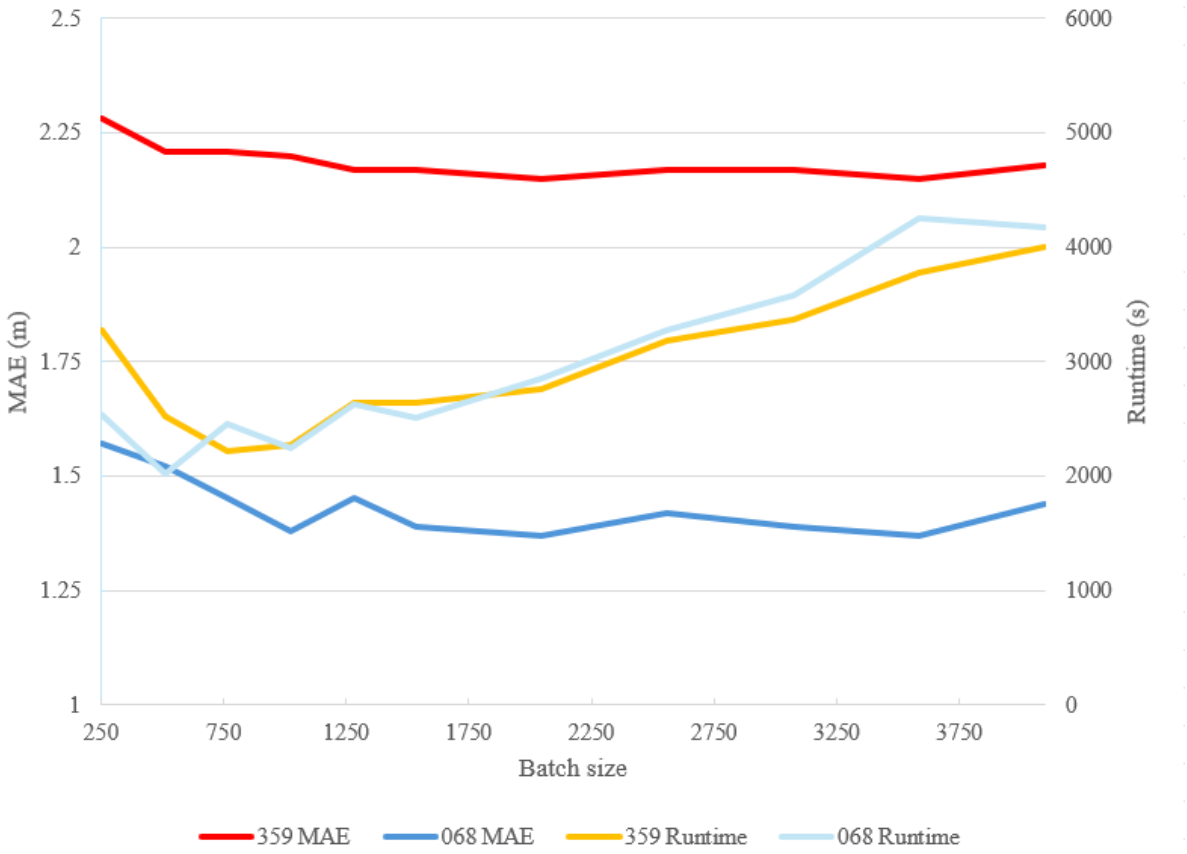


Figure 39: Graph showing the MAE versus the batch size on the primary axis (left) and the runtime versus the batch size on the secondary axis (right), for both the 068 AOI and the 359 AOI.

5.4 NeRF applicability results

This section covers what is needed to apply NeRF to data different from the DFC2019 dataset. For the imagery, it is important that it has RGB bands, preferably with a high resolution, depending on the application. Higher resolutions will provide the NeRF model with more color information, but will also increase runtime. Additionally, an RPC report needs to be available, as well as metadata covering the sun elevation, sun azimuth, and the acquisition data. After downloading the data, the first step is to preprocess the data to fit the input required by NeRF. As a starting point, the input adapter script is used from the Satellite Surface Reconstruction pipeline, which was also used by Claesen (2024), for his thesis (Bullinger et al., 2021).

Claesen (2024), only used the script to crop the panchromatic images of the Superview-1 satellite constellation. However, to use it for NeRF, the script has to be adapted to handle RGB imagery, RPCs and metadata. The first step is to adapt the script to crop the RGB imagery, which has a different spatial resolution (2.0m) compared to the panchromatic imagery (0.5m). The coordinates of the upper left and lower right corners are provided in the pipeline.toml script. In this script, the coordinate system also needs to be set, which is 17N in this case. After that, the script needs to be able to combine each cropped image with the corresponding RPC and metadata. The RPC and metadata are then adjusted to the new cropped image and combined into one JSON file. The JSON file should have the same parameters as the JSON files that are used for the DFC2019 dataset. That includes the minimum and maximum height, the number of rows and columns, sun elevation, sun azimuth, acquisition date, geojson coordinates, feature type (polygon in this case), center coordinate, and the bundle adjusted RPC report. As discussed in Section 2.4, the bundle adjusted RPC report needs to consist of 10 normalization constants and 78 coefficients, which describe the geometric relation between the 2D image coordinates and the 3D ground coordinates.

In addition to the preprocessed imagery with the JSON files, a classified TIF file of the AOI is needed. For this file, each cell has to align exactly with the preprocessed imagery, they need to be in the same coordinate system and have the same cell size. Table 16 shows the classes that the TIF file must have. The literature does not state how these classified files are produced for the DFC2019 dataset (Le Saux et al., 2019). It can be done either manually or through an automated classification pipeline. Next to the classified TIF file, a ground truth DSM is also needed. Again, the pixels of the ground truth DSM need to align exactly with the pixels of the images and thus the final predicted DSM. A resampling step might be needed, as most LiDAR gt datasets have a higher spatial resolution than RGB satellite imagery. The ground truth DSM should also be from approximately the same moment in time as the imagery. This will ensure that there are no features located in the imagery that are not present in the ground truth. Lastly, three txt files need to be created. One txt file specifying the number of rows and columns in the imagery and the cell size, and two specifying the test imagery and the train imagery. This is mainly important for the image reconstruction evaluation of NeRF. For more than 20 images, 3 test images are indicated and for less than 20, 2 test images are indicated. All of the other images are considered training imagery.

ID	Classification
02	Ground
05	Trees
06	Buildings
09	Water
17	Bridge / Elevated road
65	Unlabeled

Table 16: Classification scheme that needs to be used as input for NeRF models.

To run the NeRF model, adjustments need to be made in the code when data different from DFC2019 is used. Several scripts have to be altered including the utils.py, where the predefined_val_ts function needs to be adjusted. All files need to be in the same paths that are used in the code, or the paths need to be adjusted. A simplified overview of the steps that are required to apply NeRF to a dataset different from DFC2019 is shown in Figure 40.

To summarize, a NeRF model needs RGB satellite imagery (TIF), metadata and RPCs (JSON), a ground truth DSM and a classified raster file (TIF). When these files are obtained, some preprocessing needs to be done, followed by some adjustments to the NeRF model. It is of vital importance that all of the files are aligned properly, otherwise the model will not work. If all of the files are preprocessed correctly and are located in the proper paths, the NeRF model can be run on a new dataset.

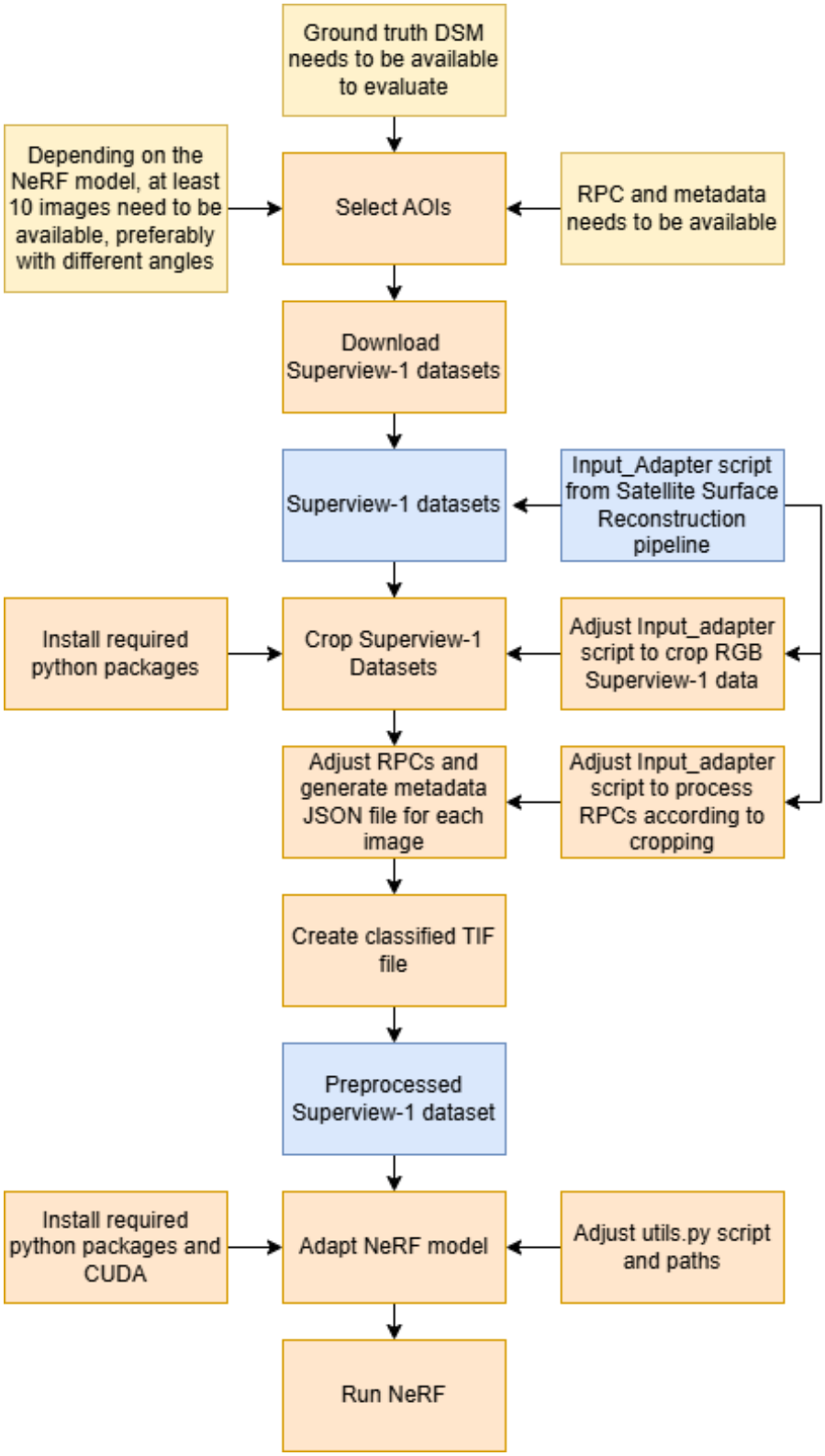


Figure 40: Simplified pipeline showing how NeRF can be applied to a dataset different from DFC2019.

6 Discussion

This section covers the evaluation of the results. The results are explained and their implications are investigated to answer the research questions. Each sub research question is separately covered with a section on the evaluation, followed by a subsection summarizing the takeaways (conclusions) and indicating the research limitations. The answers to the sub research questions lead to the final answer of the main research question in Section 7, the Conclusions.

6.1 Evaluation of the model selection (sub research question 1)

What is the optimal NeRF model to use for this research, considering available code, computing time and accuracy?

The first sub research question involved a literature research to determine the optimal NeRF model for this research. The literature research provided a clear and comprehensive overview of the NeRF models that are currently available for DSM creation from satellite imagery. Through literature research, nine different NeRF models are identified that are optimized to create DSMs from satellite imagery. The models are evaluated on three different factors. Firstly, the code needs to be available, secondly, the low computing times are required and thirdly, the accuracy needs to be as high as possible. Of the nine NeRF models, the SAT-NGP model is selected for this research, due to the code availability, low computing times and high accuracy (Billouard et al., 2024).

The result does not imply that SAT-NGP is the optimal NeRF model for every research considering DSM creation from satellite imagery using NeRF. When computing times are not an issue and sufficient hardware is available, Sat-NeRF would provide a better option. Sat-NeRF reaches MAEs of 1.29, 1.25, 1.68 and 1.624 for the AOIs, 004, 068, 214 and 260 respectively. While SAT-NGP only reaches MAEs of 1.45, 1.38, 2.06 and 1.81 for the same AOIs. Additionally, the Sat-NeRF code is openly available on GitHub with detailed documentation on how to implement the model (Billouard et al., 2024;Marí et al., 2022. Another option could be SpS-NeRF, which can run with only 2 or 3 satellite images. However, SPS-NeRF needs a low-resolution depth map (SRTM for example) and is less accurate (2.86m MAE for 214 AOI). It would be a useful model when there is little available imagery and the focus is on natural areas (L. Zhang & Rupnik, 2023).

Through the addition of Instant-NGPs multi-resolution hash encoding, SAT-NGP is able to decrease computing times from 10+ hours to minutes (depending on the hardware) (Billouard et al., 2024; Müller et al., 2022). According to literature, future versions of NeRF that are optimized to create DSMs from satellite imagery are likely to also use this technique (Billouard et al., 2024; Müller et al., 2022; Xie et al., 2023). The reduced computing times make the NeRF models available to a wider public, improving their applicability to real use cases in the future.

6.1.1 Model selection takeaways and limitations (sub research question 1)

Evaluating the model selection research question, there are some takeaways and limitations. SAT-NGP is considered the optimal model for this specific research based on code availability, training time and accuracy.

However, although the literature research is comprehensive and thorough, there are some shortcomings. More factors could be taken into account, such as ease of implementation, making the literature research more complete. Additionally, limiting the model choice to models with code openly available severely decreases the available options. Some of the models that perform best in terms of accuracy and computing times are excluded, such as EO-NeRF and RS-NeRF (Marí et al., 2023; Xie et al., 2023). Instead of excluding these models beforehand, efforts could be made to contact the scientists directly and ask for the code. Limiting the model selection too much on training times could also be avoided. An option could be to acquire improved hardware through contacts at the university or at CGI, enabling more accurate, computer intensive models to be incorporated.

6.2 Evaluation of the accuracy results (sub research question 2a and 2b)

What accuracy does NeRF achieve using high-resolution RGB satellite imagery of urban and rural environments?

The second sub research question covers the accuracy of NeRF in both an urban and a rural environment. Additionally, a distinction can be made between the standard AOIs (004, 068, 214 and 260) that are used for nearly all NeRF DSM research, and other AOIs from the DFC2019 dataset (Le Saux et al., 2019). DSMs are created for 12 AOIs in total, of which 4 are standard ones used in most research and did not need any preprocessing. The other 8 AOIs are collected as raw data from the same DFC2019 dataset and need to be preprocessed using the bundle adjustment and satellite dataset preparation pipelines (Marí et al., 2021; Marí et al., 2022).

6.2.1 Evaluation of urban accuracy results (sub research question 2a)

The NeRF predicted DSMs of urban areas have an average MAE of 2.03m, varying between 1.38m and 2.52m. Some large deviations are visible in the difference DSMs of Figure 30. The large Max E (Table 13), especially compared to the rural difference DSMs (Figure 34), are partly due to the increased height differences in urban areas. However, some reoccurring errors can be identified. The SAT-NGP model struggles with the sharp edges of buildings, where high errors occur, visualized by dark blue or red pixels. The edges of buildings often have shadows near them, which reduces the information that NeRF can gather from the imagery. The reduced information leads to errors in the predicted DSM. The issue of predicting heights of areas that are shaded is a problem that occurs in all of the NeRF models that have been developed for DSM generation from satellite imagery (Billouard et al., 2024; Derksen and Izzo, 2021; Marí et al., 2022; Marí et al., 2023; Wan et al., 2024; Xie et al., 2023; L. Zhang and Rupnik, 2023; T. Zhang et al., 2024; L. Zhang et al., 2024).

The errors in AOI 166 stand out, because a flat parking lot is present at that location, this is clearly visualized in the elevation profiles shown in Figure 32. SAT-NGP uses a robust loss function to deal with transient objects, such as cars. The model flattens out the DSM if objects are visible in some of the images but not all of them (Billouard et al., 2024). The most likely explanation for the errors is that the imagery is too inconsistent. There is a high variability for these parking lots due to differences in cars, and parking lots cover a large part of the imagery. As a result, NeRF struggled to find the height of these locations. This means that the SAT-NGP model has some flaws and is not always able to predict flat surfaces well, especially when there is high variability in the imagery.

The inability to predict flat surfaces is also common in other NeRF models, including S-NeRF and Sat-NeRF (Derksen and Izzo, 2021; Marí et al., 2022). The urban accuracy results indicate that NeRF does not solve the difficulties experienced by Claesen (2024), concerning the reduced accuracies in areas where limited feature matching is possible. These areas include flat surfaces and shadows.

6.2.2 Evaluation of the rural accuracy results (sub research question 2b)

The MAE results in Table 13 show that SAT-NGP achieved the highest accuracy for the urban AOIs. The rural AOIs have an MAE which is 0.64m higher on average, varying between 1.45m and 4.71m. The result indicates that SAT-NGP struggles more in areas with increased vegetation. This is confirmed in Figure 34, which shows that trees are either underestimated (blue) or their edges are overestimated (red). The underestimated trees can be explained by the robust loss function that is used in SAT-NGP (Billouard et al., 2024). As mentioned before, the robust loss function removes transient objects from the predicted DSMs. The function is designed to remove cars from urban areas, however, in this case, it removed trees. The visual representation of the trees changes through the images due to seasonality and weather conditions. As a result, the model sees it as a transient object and removes some of the trees, resulting in an underestimated height, as can be seen in Figure 36. Interestingly, the ME in Table 13 indicates that there is a general overprediction by SAT-NGP compared to the LiDAR gt DSM. Furthermore, the edges of trees are difficult to model due to shadows, which, as discussed before, reduce the information for the model in certain areas. The errors around the edges occur due to the LiDAR measuring very sharp edges and the NeRF model smoothing the edges. While this study was conducted, another research is published that focused on the impact of solar correction on the performance of NeRF models that create DSMs from satellite imagery. In this research, a land cover study is performed, which indicated that the accuracy of the DSMs was higher for areas with more trees (Chakraborty et al., 2025). This is contradictory compared to this study, which has opposite results. However, this study used the SAT-NGP model while Chakraborty et al. (2025) used standard NeRF, S-NeRF and Sat-NeRF. Additionally, AOIs 260 and 412 are used for the investigation compared to 068 and 359 in this research. Interestingly, Chakraborty et al. (2025) do not mask the construction site from the images of the 260 AOI. Furthermore, some AOIs (017, 359) are included for the MAE analysis, which do not have a LiDAR gt DSM that corresponds to the imagery. The use of different models and different AOIs makes it difficult to compare between the studies, and it indicates that further research is needed.

Continuing on the results of this research, the train that shows up in the bottom left of the difference DSM of AOI 105 in Figure 34 can be attributed to a train being present while the LiDAR was scanning. The train was not consistently present in the imagery and is thus considered a transient object resulting in removal by the robust loss function of SAT-NGP. Figure 33 shows that the model correctly predicts a flat surface at the location of the train. Lastly, it is interesting to look at AOIs 017 and 105 due to the large errors located on flat surfaces. In AOI 017 the highway is predicted to be much higher than the LiDAR gt and AOI 105 shows a similar trend for the flat fields in the middle left of the image. As mentioned before, other papers report on the same issue (Derksen and Izzo, 2021; Marí et al., 2022). The errors indicate that NeRF has difficulties with large flat surfaces such as grass fields and highways.

Another interesting thing to note about the observations is that nearly every NeRF DSM research uses the AOIs, 004, 068, 214 and 260, which have the lowest MAE values (Derksen and Izzo, 2021; Marí et al., 2022; Xie et al., 2023. The usage of these AOIs makes the model performances comparable, however, it can also create a bias. When research only focuses on improving the MAE for these AOIs, it might introduce a bias and overfit the models to perform well for these AOIs specifically. Currently, only EO-NeRF, SpS-NeRF, SatensoRF, BRDF-NeRF and a solar correction impact study, included other research areas as an addition to the standard (Marí et al., 2023; L. Zhang and Rupnik, 2023; T. Zhang et al., 2024; L. Zhang et al., 2024; Chakraborty et al., 2025). The NeRF model accuracies are also plotted against the number of images that are used to investigate if the model performs better or worse with more or less images. The results do not indicate a correlation between the two factors. In fact, it seems that the quality and content of the images, e.g. the consistency, different angles, little shadows, vegetation and sharpness, are more important for the accuracy of the final DSM.

6.2.3 Accuracy research takeaways and limitations (sub research questions 2a and 2b)

Evaluating the accuracy sub research questions, there are some key takeaways. Contrary to other research, the results have shown that NeRF performs better, in terms of accuracy, on urban environments compared to rural environments with increased vegetation (Chakraborty et al., 2025). However, the urban environments have a larger Max E due to the increased height differences as a result of tall buildings. This explains the large maximum error for AOI 167 for example, where a skyscraper is present in the scene (Table 13). The main features that cause errors in the NeRF predicted DSMs are edges of buildings, shadows, trees and flat surfaces. The flat surfaces can include both highways or grass fields (AOIs 105 and 167). Difficulties with shadows and flat surfaces have also been identified in other studies (Chakraborty et al., 2025). The number of images does not affect the performance if it lies between 10 and 24 images, the quality and content of the imagery are the most important.

However, when drawing these conclusions about accuracy, care must be taken and limitations of the research have to be taken into account. For example, the accuracy is only tested on DFC2019 data, and not on other data. The DFC2019 datasets consist of pansharpened Worldview-3 imagery with an RGB resolution of 0.3m. The pansharpening process could influence the performance of the NeRF model, leading to decreased accuracy. Additionally, the model might perform differently when using images that have a higher or lower resolution. A higher resolution could lead to better model performance for both rural and urban environments or improve the performance in only one of these. Therefore, conclusions can only be drawn for this specific dataset. The DFC2019 dataset does not report the date of acquisition of the LiDAR scan, which also causes uncertainty (Le Saux et al., 2019). As stated in Section 4.2, some of the DFC2019 AOIs cannot be used due to differences between the imagery and the LiDAR gt DSM. Undetected discrepancies between the LiDAR gt DSM and the imagery might be present in the other AOIs. As a result, some errors can be introduced that are not caused by the NeRF model, leading to an underestimation of the accuracy of the NeRF model. Furthermore, research has pointed out that LiDAR is not a perfect method in terms of accuracy and the LiDAR gt DSM might not provide a true representation of the real elevation (Liu, 2008). Additionally, the number of AOIs that are used in this research could be increased to create more certainty in determining the performance of the model for both urban and rural environments.

6.3 Evaluation of the sensitivity results (sub research question 3)

What is the sensitivity of the NeRF model used considering the number of iterations and batch size, in both urban and rural environments?

The third sub research question involved the sensitivity of the NeRF model towards the two parameters number of iterations and batch size. As mentioned in Section 1.1, until now, the sensitivity has only been tested considering the number of sample points for each ray and the number of hidden layers in the MLP network (Marí et al., 2023).

6.3.1 Evaluation of the number of iterations sensitivity results

Different NeRF models are often executed with different numbers of iterations. For example, S-NeRF uses 30,000 iterations, while Sat-NeRF uses 300,000 iterations and GC-NeRF uses 15,000 iterations (Derksen and Izzo, 2021; Marí et al., 2022; Wan et al., 2024). For a NeRF model, the number of iterations determines how many times a batch of rays is sampled and passed through the model. After running the model 22 times with 11 different configurations for the number of iterations for both a rural and urban AOI, the results are evaluated. Firstly, the results show an increase in the runtime as the number of iterations is increased. Secondly, the MAE values seem to be nearly stable after 15,000 iterations. If a lower number of iterations is used, the MAE values increase and vertical accuracy decreases. Both the rural and urban AOIs 359 and 068, show similar trends. However, Figure 38 shows that the 068 MAE improves more sharply when the number of iterations is increased between 1,000 and 15,000. From the results, it can be concluded that, for SAT-NGP, a lower number of iterations can be used than the recommended 60,000 while still maintaining vertical accuracy. After 15,000 iterations, the MAE values improve minimally, while the run time is reduced by more than 50%. The reduction in runtime improves the applicability of NeRF in real-life use cases and makes the models available to a wider public with limited hardware. Additionally, it provides computing space for other methods to be added to the model and further improve its accuracy. Furthermore, the sensitivity analysis of iterations shows that the results achieved by Billouard et al. (2024) are even more impressive. Compared to Sat-NERF, which takes 20 hours to run, SAT-NGP improves even more on runtime than is reported if fewer iterations are used (Marí et al., 2022).

6.3.2 Evaluation of the batch size sensitivity results

The batch size refers to the number of rays that are sampled by the model and passed through in a single iteration. Across the different NeRF models, the batch size is mostly kept at 1024, this is the case for S-NeRF, Sat-NeRF, EO-NeRF, RS-NeRF, SpS-NeRF, SAT-NGP and BRDF NeRF (Billouard et al., 2024; Derksen and Izzo, 2021; Marí et al., 2022; Marí et al., 2023; Xie et al., 2023; L. Zhang and Rupnik, 2023; L. Zhang et al., 2024). Notably, SatensoRF is the only other NeRF model that uses a different batch size of 8096, while the authors of GC-NeRF do not report on their batch size (Wan et al., 2024; T. Zhang et al., 2024).

The results of the batch size sensitivity analysis show that the MAE decreases for both the 068 and 359 AOIs from a batch size of 256 until 1024. A batch size of more than 1024 does not seem to improve the performance of NeRF for both AOIs. Investigating the runtime, the results show a decrease between batch sizes 256 and 768 and an increase after 768. From the results, it can be interpreted that a higher batch size after 768 does not lead to improved performance of the SAT-NGP model. It seems that a batch size of 1024 is the most optimal for SAT-NGP considering the runtime and accuracy, which is in line with the recommendations of Billouard et al. (2024) and most other NeRFs. The sensitivity analysis shows little difference between AOIs 068 and 359, indicating no sensitivity differences between urban and rural scenes.

6.3.3 Sensitivity research takeaways and limitations

The sensitivity analysis provides valuable insight into the influence of both the number of iterations and the batch size on the accuracy of SAT-NGP. Increasing the number of iterations only has a noticeable decreasing effect on MAE values up to 15,000 iterations, while the batch size shows that the optimal size is 1024 considering MAE values and runtime. The runtime increased for both parameters from a certain point while no improvement was made on MAE values. Therefore, this sensitivity analysis has shown that for SAT-NGP, training times can be reduced by 50% while maintaining accuracy. This can lead to a wider use of the SAT-NGP model due to reduced hardware requirements. In addition, it improves the applicability of the model for real-life use cases, because larger study areas can be investigated with less computing time.

However, some limitations need to be taken into account when drawing these conclusions. To start, the sensitivity analysis is only conducted on two research areas, which is limited. Including more research areas would strengthen the results, but is not feasible for this research due to time restrictions. Furthermore, results point out little difference between urban and rural areas considering the sensitivity to iterations and the batch size. Including more research areas would also strengthen this conclusion. Although the SAT-NGP model is run 44 times for this sensitivity analysis, more different configurations could also improve the results. For example, different batch sizes and the number of iterations could be tested in combination, such as running NeRF with many iterations and a small batch size. Additionally, the number of iterations could be tested with much higher numbers than 150,000 to see if MAE results could improve. It would also be interesting to investigate the sensitivity for other parameters, such as the number of sample points for each ray, or the number of hidden layers in the MLP. As mentioned before, while conducting this research, another study was published by Chakraborty et al. (2025), who researched the impact of solar correction on NeRFs that create DSMs from satellite imagery. Chakraborty et al. (2025) test the model performance of standard NeRF, S-Nerf and Sat-NeRF considering image reconstruction and DSM accuracy (MAE). Instead of changing parameters, they tested the performance with and without solar correction. Their results show that solar correction improves the results for S-NeRF and Sat-NeRF. It is interesting to note that similar to this research, Chakraborty et al. (2025) use more AOIs than the preprocessed 004, 068, 214 and 260 AOIs. However, they include AOIs such as 017 and 031, which this study has pointed out, do not have a LiDAR gt DSM that corresponds with the imagery. It would be beneficial to conduct sensitivity analysis on other NeRF models, so a conclusion can be drawn on the impact of the number of iterations and the batch size on NeRF models in general.

6.4 Evaluation of the applicability results (sub research question 4)

What is the applicability of NeRF and what is needed to use NeRF on data different from DFC2019?

The fourth sub question investigates the applicability of NeRF and how to use it on data other than the DFC2019. To answer the question, an applicability research is executed on SAT-NGP, the selected model of the first sub research question (Section 6.1). By simulating the application to a different dataset consisting of Superview-1 imagery in the Netherlands, a clear overview is created of what is needed to apply SAT-NGP to different data.

The results show that a NeRF model requires specific inputs, all of which need to be in the perfect format. At least 10 input images are required for SAT-NGP. These images need to have certain metadata available, as well as an RPC report. The RPC report is preferably bundle adjusted and needs to be combined with the metadata in a JSON file for each separate satellite image. Furthermore, a gt DSM is needed for evaluation, which should be created around the same time as the satellite imagery. A classified TIF file specifying a class for each pixel is also needed.

Currently, the combination of these different files and the adjustments that need to be made to them, such as bundle adjusting the RPCs or creating the classified TIF file, shows that it is challenging to apply NeRF to a dataset different from the DFC2019 data. Everything is highly dependent on one another and if something is not exactly configured or aligned the right way, the whole model will fail to produce results. The DFC2019 dataset, especially the prepared 004, 068, 214 and 260 AOIs, provides a "perfect" scenario, which is not the case for most other data. Some NeRF models have been applied to other datasets such as EO-NeRF, SpS-NeRF, BRDF-NeRF (Marí et al., 2023; L. Zhang and Rupnik, 2023; L. Zhang et al., 2024). EO-Nerf is applied to the Intelligence Advanced Research Projects Activity Multi-View Stereo 3D Mapping dataset, however, this is another challenge dataset (similarly to DFC2019) and is similarly considered a "perfect" scenario (Bosch et al., 2016; Marí et al., 2023). The authors of SpS-NeRF and BRDF-NeRF managed to apply their respective NeRF models to Pléiades imagery. Both NeRF models only need 2 or 3 satellite images, which need to be taken at roughly the same time, due to the lack of a transient object removal technique. A downside of the models is that they require a depth input beforehand, in the form of a rough DSM, to guide the sampling of rays. In both SpS-NeRF and BRDF-NeRF, SRTM is used as a rough guidance DSM. Alternatively, SGM can be used to provide a low resolution depth map (L. Zhang and Rupnik, 2023; L. Zhang et al., 2024). The lack of transient objects and need for a rough depth map make the models less usable but more applicable to datasets different from the DFC2019 data. The evaluation shows that there is still room for improvement considering the applicability of NeRFs that create DSMs from satellite imagery in general. Real-life datasets are often incomplete or lack other requirements for a DSM creating NeRF model to work, as is the case for the Superview-1 data for which no classified TIF file was available and many adjustments needed to be made. If DSM creation from NeRFs can be made easier to apply, their promising results will become more accessible and implemented in real-life case studies.

6.4.1 Applicability takeaways and limitations

The applicability study shows that it is still difficult to apply DSM creating NeRFs to other datasets. Most NeRFs have not been applied to other data or to real-life case studies, indicating that it is still hard to do so. Currently, there are too many specific requirements for a NeRF model to work optimally.

It is important to note that this applicability study has only investigated SAT-NGP. Other NeRF models might be easier to implement, although literature indicates that this is not the case (Billouard et al., 2024; Derksen and Izzo, 2021; Marí et al., 2022; Wan et al., 2024; Xie et al., 2023; T. Zhang et al., 2024). Furthermore, the applicability is tested solely on Superview-1 and AHN data. Although some research is done on available datasets for this research, there might be other datasets that are easier to implement for NeRF.

7 Conclusions

In conclusion, this thesis has investigated to what extent NeRFs can be used to create DSMs of both urban and rural environments from satellite imagery. First, the optimal NeRF model for this study was selected based on code availability, runtime, and accuracy. The model selection was followed by an accuracy analysis that included 6 urban and 6 rural AOIs from the DFC2019 dataset. After that, a sensitivity analysis was conducted on the number of iterations and batch size, to determine their impact on MAE values and runtime. Finally, the applicabillity analysis focused on what is needed to apply NeRF to data different from DFC2019.

After a comprehensive literature research, this study has shown that considering the requirements for this research, SAT-NGP is the optimal model to use. Furthermore, the accuracy results show that SAT-NGP produces more accurate results for urban areas compared to rural areas. Multiple challenging situations are identified, including the edges of buildings, shadows, flat surfaces, and trees. Sensitivity analysis indicated that, with a reduction of 45,000 in the number of iterations, the SAT-NGP model can run 50% faster compared to the proposed model configuration, with minimal to no accuracy loss. Changing the proposed ray batch size of 1024 yielded no improvements concerning the accuracy or runtime. Although NeRF shows promising results, the applicability remains low. Numerous different files are required that need to be perfectly adjusted and formatted in order for the model to work. Therefore, this research shows that the extent to which NeRFs can be used to generate DSMs in urban and rural environments can be considered limited. Future research should focus on improving the applicability of NeRF to make it more accessible to the public. Furthermore, the issues concerning edges of buildings, shadows, flat surfaces, and trees should be addressed. It is important to note that this research only used SAT-NGP to derive results concerning the accuracy, sensitivity, and applicability. The findings might not apply to other DSM generating NeRFs, although literature indicates similar trends. Ultimately, NeRF still has a long way to go before it can become a widely used method in both research and applied environments.

8 Future work

This study involving literature research, accuracy analysis, sensitivity analysis and applicability analysis, sparks many new questions concerning NeRFs that create DSMs from satellite imagery. From these questions, some recommendations for future research can be made.

To improve the accuracy of DSMs generated by NeRF, research should focus on incorporating methods to handle shadows as they lead to errors. Research has already produced different shadow models, such as the shadow-aware irradiance model of S-NeRF, improved by EO-NeRF, however, these are still far from perfect (Derksen and Izzo, 2021; Marí et al., 2023). Furthermore, NeRF predicted DSMs are currently too smooth and lack sharp edges, resulting in errors. Research should also focus on incorporating methods to find a solution for this problem. One of the solutions could be edge constrained DSM refinement as proposed by Hu et al. (2025). Their method is able to improve DSM accuracies up to 59.02%. Other issues lie in flat surfaces and trees, these features are difficult for NeRF and other DSM generation methods to handle. For SAT-NGP, it could be beneficial to remove the robust loss function to avoid tree removal and improve performance for natural areas (Billouard et al., 2024).

Next to improving the accuracy, more research should also be conducted on the sensitivity of NeRF to certain parameters. For example, research could include changing the number of samples for each ray, or the number of hidden layers in the MLP network. Furthermore, it is important to perform sensitivity analysis on all of the NeRF models and not just SAT-NGP. Through sensitivity analysis, NeRFs can be better optimized, as this research has shown.

The literature research pointed out that there has been little application of NeRF to datasets different from DFC2019. The DFC2019 dataset consists mostly of urban or rural AOIs, but environments with purely vegetation and bare soil or rock are absent (Le Saux et al., 2019). Since DSMs are used extensively in natural research areas such as flood risk analysis, mangrove forest mapping, and land cover classification, more NeRFs should focus on generating accurate DSMs of these environments (McClean et al., 2020; Simard et al., 2006). Therefore, research should focus on making the DSM creating NeRFs more applicable to other datasets and include environments comprising solely vegetation, soil and rock. This would open up many possibilities and improve the use of NeRF. Currently, only SpS-NeRF and BRDF-NeRF are partly focused on these natural environments (L. Zhang and Rupnik, 2023; L. Zhang et al., 2024). It could be promising to combine the methods of BRDF-NeRF, the most accurate model for natural environments, and EO-NeRF, the most accurate model for urban environments, to create a NeRF capable of handling both (Marí et al., 2023; L. Zhang et al., 2024). However, it could also be wiser to develop separate NeRF models for specific purposes. To enhance the applicability of DSM generating NeRFs, it would be interesting to integrate them into NeRFstudio or a separate framework. NeRFstudio is a software program that lets you choose between different NeRF models for scene reconstruction. You only have to provide the input, which makes it much easier to test different NeRF models (Tancik et al., 2023). Furthermore, it would be greatly beneficial if code is made available for all of the NeRF models, to enhance research and eliminate double work.

It would also be interesting to see if NeRF can be applied using aerial imagery. This type of imagery usually has a higher resolution compared to satellite imagery, which could result in more accurate results. Additionally, the flight path can be adjusted to acquire data from different angles. This results in high-quality datasets that might lead to highly accurate results.

There are number of additions that could have been introduced in this research if more time was available. Firstly, it would have been interesting to implement the applicability research and use SAT-NGP on the Superview-1 data. The results could be compared with DFC2019 results and the performance of SAT-NGP could be investigated in areas that consist solely of vegetation. Additionally, it would be a significant contribution to the scientific field if a pipeline could be developed to apply NeRF to data different from the DFC2019. During the research it became clear that it would take too much time for this thesis. Additionally, as mentioned above, more parameters could be used for the sensitivity analysis such as the number of samples for each ray or the number of layers in the MLP network. Finally, it would have been interesting to try and incorporate different techniques in an existing NeRF model, such as a different solar correction model or robust loss function.

9 Appendix

9.1 Appendix A

```
import rpcm
import glob
import os
import numpy as np
import srtm4
import shutil
import sys
import json
from sat_utils import get_file_id
import rasterio


def rio_open(*args,**kwargs):
    import rasterio
    import warnings

    with warnings.catch_warnings():
        warnings.filterwarnings("ignore", category=UserWarning)
        return rasterio.open(*args,**kwargs)

def get_image_lonlat_aoi(rpc, h, w):
    z = srtm4.srtm4(rpc.lon_offset, rpc.lat_offset)
    cols, rows, alts = [0,w,w,0], [0,0,h,h], [z]*4
    lons, lats = rpc.localization(cols, rows, alts)
    lonlat_coords = np.vstack((lons, lats)).T
    geojson_polygon = {"coordinates": [lonlat_coords.tolist()], "type": "Polygon
        "}
    x_c = lons.min() + (lons.max() - lons.min())/2
    y_c = lats.min() + (lats.max() - lats.min())/2
    geojson_polygon["center"] = [x_c, y_c]
    return geojson_polygon

def run_ba(img_dir, output_dir):

    from bundle_adjust.cam_utils import SatelliteImage
    from bundle_adjust.ba_pipeline import BundleAdjustmentPipeline
    from bundle_adjust import loader

    # load input data
    os.makedirs(output_dir, exist_ok=True)
    print(img_dir)
    myimages = sorted(glob.glob(img_dir + "/*.tif"))
    print(myimages)
    myrpcs = [rpcm.rpc_from_geotiff(p) for p in myimages]
    input_images = [SatelliteImage(fn, rpc) for fn, rpc in zip(myimages, myrpcs)
        ]

    ba_input_data = {}
    ba_input_data['in_dir'] = img_dir
    ba_input_data['out_dir'] = os.path.join(output_dir, "ba_files")
    ba_input_data['images'] = input_images
    print('Input data set!\n')

    # redirect all prints to a bundle adjustment logfile inside the output
        directory
    os.makedirs(ba_input_data['out_dir'], exist_ok=True)
    path_to_log_file = "{}/bundle_adjust.log".format(ba_input_data['out_dir'])
    print("Running bundle adjustment for RPC model refinement ...")
    print("Path to log file: {}".format(path_to_log_file))
```

```
    log_file = open(path_to_log_file, "w+")
    sys.stdout = log_file
    sys.stderr = log_file
    # run bundle adjustment
    #tracks_config = {'FT_reset': True, 'FT_sift_detection': 's2p', '
        FT_sift_matching': 'epipolar_based', "FT_K": 300}
    tracks_config = {'FT_reset': False, 'FT_save': True, 'FT_sift_detection': '
        s2p', 'FT_sift_matching': 'epipolar_based'}
    ba_extra = {"cam_model": "rpc"}
    print('ba_extra', ba_extra)
    ba_pipeline = BundleAdjustmentPipeline(ba_input_data, tracks_config=
        tracks_config, extra_ba_config=ba_extra)
    ba_pipeline.run()
    # close logfile
    sys.stderr = sys.__stderr__
    sys.stdout = sys.__stdout__
    log_file.close()
    print("... done !")
    print("Path to output files: {}".format(ba_input_data['out_dir']))

    # save all bundle adjustment parameters in a temporary directory
    ba_params_dir = os.path.join(ba_pipeline.out_dir, "ba_params")
    os.makedirs(ba_params_dir, exist_ok=True)
    np.save(os.path.join(ba_params_dir, "pts_ind.npy"), ba_pipeline.ba_params.
        pts_ind)
    np.save(os.path.join(ba_params_dir, "cam_ind.npy"), ba_pipeline.ba_params.
        cam_ind)
    np.save(os.path.join(ba_params_dir, "pts3d.npy"), ba_pipeline.ba_params.
        pts3d_ba - ba_pipeline.global_transform)
    np.save(os.path.join(ba_params_dir, "pts2d.npy"), ba_pipeline.ba_params.
        pts2d)
    fnames_in_use = [ba_pipeline.images[idx].geotiff_path for idx in ba_pipeline
        .ba_params.cam_prev_indices]
    loader.save_list_of_paths(os.path.join(ba_params_dir, "geotiff_paths.txt"),
        fnames_in_use)

def create_dataset_from_DFC2019_data(aoi_id, img_dir, dfc_dir, output_dir,
    use_ba=False):

    # create a json file of metadata for each input image
    # contains: h, w, rpc, sun elevation, sun azimuth, acquisition date
    #           + geojson polygon with the aoi of the image
    os.makedirs(output_dir, exist_ok=True)
    path_to_dsm = os.path.join(dfc_dir, "Track3-Truth/{}_DSM.tif".format(aoi_id)
        )
    if aoi_id[:3] == "JAX":
        path_to_msi = "http://138.231.80.166:2334/core3d/Jacksonville/WV3/MSI"
    elif aoi_id[:3] == "OMA":
        path_to_msi = "http://138.231.80.166:2334/core3d/Omaha/WV3/MSI"
    if use_ba:
        from bundle_adjust import loader
        geotiff_paths = loader.load_list_of_paths(os.path.join(output_dir, "
            ba_files/ba_params/geotiff_paths.txt"))
        geotiff_paths = [p.replace("/pan_crops/", "/crops/") for p in
            geotiff_paths]
        geotiff_paths = [p.replace("PAN.tif", "RGB.tif") for p in geotiff_paths]
        ba_geotiff_basenames = [os.path.basename(x) for x in geotiff_paths]
        ba_kps_pts3d_ind = np.load(os.path.join(output_dir, "ba_files/ba_params/
            pts_ind.npy"))
        ba_kps_cam_ind = np.load(os.path.join(output_dir, "ba_files/ba_params/
            cam_ind.npy"))
        ba_kps_pts2d = np.load(os.path.join(output_dir, "ba_files/ba_params/
            pts2d.npy"))
```

```
    else:
        geotiff_paths = sorted(glob.glob(img_dir + "/*.tif"))

    for rgb_p in geotiff_paths:
        d = {}
        d["img"] = os.path.basename(rgb_p)

        src = rio_open(rgb_p)
        d["height"] = int(src.meta["height"])
        d["width"] = int(src.meta["width"])
        original_rpc = rpcm.RPCModel(src.tags(ns='RPC'), dict_format="geotiff")

        img_id = src.tags()["NITF_IID2"].replace("␣", "_")
        msi_p = "{}/{}.NTF".format(path_to_msi, img_id)
        src = rio_open(msi_p)
        d["sun_elevation"] = src.tags()["NITF_USE00A_SUN_EL"]
        d["sun_azimuth"] = src.tags()["NITF_USE00A_SUN_AZ"]
        d["acquisition_date"] = src.tags()['NITF_STDIDC_ACQUISITION_DATE']
        d["geojson"] = get_image_lonlat_aoi(original_rpc, d["height"], d["width"
            ])

        src = rio_open(path_to_dsm)
        dsm = src.read()[0, :, :]
        d["min_alt"] = int(np.round(dsm.min() - 1))
        d["max_alt"] = int(np.round(dsm.max() + 1))

        if use_ba:
            # use corrected rpc
            rpc_path = os.path.join(output_dir, "ba_files/rpcs_adj/{}.rpc_adj".
                format(get_file_id(rgb_p)))
            d["rpc"] = rpcm.rpc_from_rpc_file(rpc_path).__dict__
            #d_out["rpc"] = rpc_rpcm_to_geotiff_format(rpc.__dict__)

            # additional fields for depth supervision
            ba_kps_pts3d_path = os.path.join(output_dir, "ba_files/ba_params/
                pts3d.npy")
            shutil.copyfile(ba_kps_pts3d_path, os.path.join(output_dir, "pts3d.
                npy"))
            cam_idx = ba_geotiff_basenames.index(d["img"])
            d["keypoints"] = {"2d_coordinates": ba_kps_pts2d[ba_kps_cam_ind ==
                cam_idx, :].tolist(),
                              "pts3d_indices": ba_kps_pts3d_ind[ba_kps_cam_ind
                                  == cam_idx].tolist()}
        else:
            # use original rpc
            d["rpc"] = original_rpc.__dict__

        with open(os.path.join(output_dir, "{}.json".format(get_file_id(rgb_p)))
            , "w") as f:
            json.dump(d, f, indent=2)

def create_train_test_splits(input_sample_ids, test_percent=0.15,
    min_test_samples=2):

    def shuffle_array(array):
        import random
        v = array.copy()
        random.shuffle(v)
        return v

    n_samples = len(input_sample_ids)
    input_sample_ids = np.array(input_sample_ids)
    all_indices = shuffle_array(np.arange(n_samples))
```

```
    n_test = max(min_test_samples, int(test_percent * n_samples))
    n_train = n_samples - n_test

    train_indices = all_indices[:n_train]
    test_indices = all_indices[-n_test:]

    train_samples = input_sample_ids[train_indices].tolist()
    test_samples = input_sample_ids[test_indices].tolist()

    return train_samples, test_samples

def read_DFC2019_lonlat_aoi(aoi_id, dfc_dir):
    from bundle_adjust import geo_utils
    if aoi_id[:3] == "OMA":
        zonestring = "15T"
    elif aoi_id[:3] == "JAX":
        zonestring = "17N"
    else:
        raise ValueError("AOI not valid. Expected JAX_(3digits) but received {}"
            .format(aoi_id))
    roi = np.loadtxt(os.path.join(dfc_dir, "Track3-Truth/" + aoi_id + "_DSM.txt"
        ))
    xoff, yoff, xsize, ysize, resolution = roi[0], roi[1], int(roi[2]), int(roi
        [2]), roi[3]
    ulx, uly, lrx, lry = xoff, yoff + ysize * resolution, xoff + xsize *
        resolution, yoff
    xmin, xmax, ymin, ymax = ulx, lrx, uly, lry
    easts = [xmin, xmin, xmax, xmax, xmin]
    norths = [ymin, ymax, ymax, ymin, ymin]
    lons, lats = geo_utils.lonlat_from_utm(easts, norths, zonestring)
    lonlat_bbx = geo_utils.geojson_polygon(np.vstack((lons, lats)).T)
    return lonlat_bbx

def crop_geotiff_lonlat_aoi(geotiff_path, output_path, lonlat_aoi):
    with rasterio.open(geotiff_path, 'r') as src:
        profile = src.profile
        tags = src.tags()
    crop, x, y = rpcm.utils.crop_aoi(geotiff_path, lonlat_aoi)

    rpc = rpcm.rpc_from_geotiff(geotiff_path)
    rpc.row_offset -= y
    rpc.col_offset -= x
    not_pan = len(crop.shape) > 2
    if not_pan:
        profile["height"] = crop.shape[1]
        profile["width"] = crop.shape[2]
    else:
        profile["height"] = crop.shape[0]
        profile["width"] = crop.shape[1]
        profile["count"] = 1
    with rasterio.open(output_path, 'w', **profile) as dst:
        if not_pan:
            dst.write(crop)
        else:
            dst.write(crop, 1)
        dst.update_tags(**tags)
        dst.update_tags(ns='RPC', **rpc.to_geotiff_dict())


def create_satellite_dataset(aoi_id, dfc_dir, output_dir, ba=True, crop_aoi=True
    , splits=False):
    if crop_aoi:
        # prepare crops
```

```
        aoi_lonlat = read_DFC2019_lonlat_aoi(aoi_id, dfc_dir)
        crops_dir = os.path.join(output_dir, "crops")
        os.makedirs(crops_dir, exist_ok=True)
        img_dir = os.path.join(dfc_dir, "Track3-RGB/{}".format(aoi_id))
        print('img_dir', img_dir)
        myimages = sorted(glob.glob(img_dir + "/*.tif"))
        print(myimages)
        pan = True
        if aoi_id in ["JAX_068", "JAX_509"]:
            pan_dir = "/vsicurl/http://138.231.80.166:2332/grss-2019/track_3/
                Track3-MSI-1/"
        elif aoi_id in ["JAX_156", "JAX_165"]:
            pan_dir = "/vsicurl/http://138.231.80.166:2332/grss-2019/track_3/
                Track3-MSI-2/"
        elif aoi_id in ["JAX_359"]:
            pan_dir = "/vsicurl/http://138.231.80.166:2332/grss-2019/track_3/
                Track3-MSI-3/"
        elif aoi_id in ["OMA_212"]:
            pan_dir = "/vsicurl/http://138.231.80.166:2332/grss-2019/track_3/
                Track3-MSI-7/"
        else:
            pan_dir = "/vsicurl/http://138.231.80.166:2332/grss-2019/track_3/
                Track3-MSI-7/"
        for geotiff_path in myimages:
            out_crop_path = os.path.join(crops_dir, os.path.basename(
                geotiff_path))
            crop_geotiff_lonlat_aoi(geotiff_path, out_crop_path, aoi_lonlat)
            if pan:
                pan_crops_dir = os.path.join(output_dir, "pan_crops")
                os.makedirs(pan_crops_dir, exist_ok=True)
                out_crop_path = os.path.join(pan_crops_dir, os.path.basename(
                    geotiff_path))
                geotiff_path = os.path.join(pan_dir, os.path.basename(
                    geotiff_path).replace("RGB.tif", "PAN.tif"))
                crop_geotiff_lonlat_aoi(geotiff_path, out_crop_path, aoi_lonlat)
        img_dir = crops_dir
    else:
        img_dir = os.path.join(dfc_dir, "Track3-RGB/{}".format(aoi_id))
    if ba:
        run_ba(img_dir, output_dir)
    create_dataset_from_DFC2019_data(aoi_id, img_dir, dfc_dir, output_dir,
        use_ba=ba)

    # create train and test splits
    if splits:
        json_files = [os.path.basename(p) for p in glob.glob(os.path.join(
            output_dir, "*.json"))]
        train_samples, test_samples = create_train_test_splits(json_files)
        with open(os.path.join(output_dir, "train.txt"), "w+") as f:
            f.write("\n".join(train_samples))
        with open(os.path.join(output_dir, "test.txt"), "w+") as f:
            f.write("\n".join(test_samples))

    print("done")

if __name__ == '__main__':
    import fire
    fire.Fire(create_satellite_dataset)
```

9.2 Appendix B


```
{
  "img": "JAX_004_006_RGB.tif",
  "height": 813,
  "width": 788,
  "sun_elevation": "+33.5",
  "sun_azimuth": "158.6",
  "acquisition_date": "20141214160402",
  "geojson": {
    "coordinates": [
      [
        [
          -81.70781462153118,
          30.358940042487827
        ],
        [
          -81.70513996047383,
          30.359033380135823
        ],
        [
          -81.70513487087763,
          30.35664441314813
        ],
        [
          -81.70780935131354,
          30.356551109701456
        ]
      ]
    ],
    "type": "Polygon",
    "center": [
      -81.7064747462044,
      30.35779224491864
    ]
  },
  "min_alt": -24,
  "max_alt": 1,
  "rpc": {
    "row_offset": 404.72917331329,
    "col_offset": 394.640297401247,
    "lat_offset": 30.357787163399,
    "lon_offset": -81.706481969014,
    "alt_offset": -21.0,
    "row_scale": 416.501476005362,
    "col_scale": 404.000879344507,
    "lat_scale": 0.00221709962,
    "lon_scale": 0.002439793864,
    "alt_scale": 501.0,
    "row_num": [
      1.3821612e-05,
      0.069553908458,
      -1.811420563723,
      0.742025718701,
      3.247583e-06,
      -2.041298e-06,
      -3.072309e-06,
      -2.5164231e-05,
      -4.358526e-06,
      -5.3558299e-05,
      -2.12126e-07,
      2.131e-09,
      -1.02536e-07,
```

```
      6.2172e-08,
      4.8317e-08,
      3.0404e-08,
      8.311e-08,
      -1.275e-09,
      -2.27536e-07,
      -2.5306e-08
    ],
    "row_den": [
      1.0,
      8.247141e-06,
      1.9326941e-05,
      2.3912253e-05,
      4.8447e-08,
      1.54514e-07,
      -8.2283e-08,
      -2.3215e-08,
      2.39855e-07,
      2.661e-09,
      -8.09e-10,
      1.167e-09,
      7.75e-10,
      1.222e-09,
      1.57e-10,
      5.401e-09,
      2.983e-09,
      -1.91e-09,
      -7.143e-09,
      -6.42e-10
    ],
    "col_num": [
      -0.001225964267,
      1.779131163196,
      0.003508626895,
      -0.778770628026,
      -0.000135167448,
      -5.1215316e-05,
      -0.000314775304,
      0.00090483435,
      -1.7882154e-05,
      6.4791771e-05,
      -2.54656e-07,
      4.34865e-07,
      -4.2573e-08,
      -4.7445e-07,
      -3.43344e-07,
      -8.246e-09,
      7.1166e-08,
      -3.42252e-07,
      4.7512e-08,
      2.16623e-07
    ],
    "col_den": [
      1.0,
      0.000431847146,
      -1.5192884e-05,
      -0.000694560823,
      -1.25983e-07,
      -3.2255e-07,
      -2.3177e-08,
      -6.9578e-08,
      -1.19727e-07,
      -5.43418e-07,
```

```
      -3.34e-10,
      3.106e-09,
      5.83e-10,
      2.692e-09,
      1.23e-10,
      9.6e-11,
      3.73e-10,
      -4.963e-09,
      -3.2e-10,
      -2.065e-09
    ]
  }
```

9.3 Appendix C

```
import os
import random
import rasterio
import numpy as np

import cv2
import matplotlib.pyplot as plt

import torch
import mcubes

from packaging import version as pver
from torchmetrics.functional import structural_similarity_index_measure



def custom_meshgrid(*args):
    # ref: https://pytorch.org/docs/stable/generated/torch..html?highlight=meshgrid#torch.meshgrid
    if pver.parse(torch.__version__) < pver.parse('1.10'):
        return torch.meshgrid(*args)
    else:
        return torch.meshgrid(*args, indexing='ij')

@torch.jit.script
def linear_to_srgb(x):
    return torch.where(x < 0.0031308, 12.92 * x, 1.055 * x ** 0.41666 - 0.055)


@torch.jit.script
def srgb_to_linear(x):
    return torch.where(x < 0.04045, x / 12.92, ((x + 0.055) / 1.055) ** 2.4)


def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True


def visualize_depth(depth):
    """
    depth: (H, W)
    """
    x = depth.cpu().numpy()
    x = np.nan_to_num(x) # change nan to 0
    mi = np.min(x) # get minimum depth
    ma = np.max(x)
    x = (x-mi)/(ma-mi+1e-8) # normalize to 0~1
    x = (255*x).astype(np.uint8)
    x_ = np.clip(x, 0, 255)
    return x_

def save_output_image(input, output_path, source_path):
    """
    input: (D, H, W) where D is the number of channels (3 for rgb, 1 for grayscale)
        can be a pytorch tensor or a numpy array
```

```
    """
    # convert input to numpy array float32
    if len(input.shape) == 3: # rgb
        H, W, D = input.shape
        input = input.view(D, H, W)
    else: # maybe depth
        H, W = input.shape
        input = input.view(1, H, W)
    if torch.is_tensor(input):
        im_np = input.type(torch.FloatTensor).cpu().numpy()
    else:
        im_np = input.astype(np.float32)

    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    with rasterio.open(source_path, 'r') as src:
        profile = src.profile
        profile["dtype"] = rasterio.float32
        profile["height"] = im_np.shape[1]
        profile["width"] = im_np.shape[2]
        profile["count"] = im_np.shape[0]
        with rasterio.open(output_path, 'w', **profile) as dst:
            dst.write(im_np)

def cv2_save_plot(x, path, title, is_depth=False):
    # x: [3, H, W] or [1, H, W] or [H, W]

    if is_depth:
        x = visualize_depth(x)
        plt.imshow(x)
        plt.title(title)
        plt.colorbar()
        plt.savefig(path, bbox_inches='tight', dpi=200)
        plt.clf()
    else:
        plt.imshow(x)
        plt.title(title)
        plt.savefig(path, bbox_inches='tight', dpi=200)
        plt.clf()


def extract_fields(bound_min, bound_max, resolution, query_func, S=128):

    X = torch.linspace(bound_min[0], bound_max[0], resolution).split(S)
    Y = torch.linspace(bound_min[1], bound_max[1], resolution).split(S)
    Z = torch.linspace(bound_min[2], bound_max[2], resolution).split(S)

    u = np.zeros([resolution, resolution, resolution], dtype=np.float32)
    with torch.no_grad():
        for xi, xs in enumerate(X):
            for yi, ys in enumerate(Y):
                for zi, zs in enumerate(Z):
                    xx, yy, zz = custom_meshgrid(xs, ys, zs)
                    pts = torch.cat([xx.reshape(-1, 1), yy.reshape(-1, 1), zz.
                        reshape(-1, 1)], dim=-1) # [S, 3]
                    val = query_func(pts).reshape(len(xs), len(ys), len(zs)).
                        detach().cpu().numpy() # [S, 1] --> [x, y, z]
                    u[xi * S: xi * S + len(xs), yi * S: yi * S + len(ys), zi * S
                        : zi * S + len(zs)] = val
    return u


def extract_geometry(bound_min, bound_max, resolution, threshold, query_func):
    u = extract_fields(bound_min, bound_max, resolution, query_func)
```

```
120
121     vertices, triangles = mcubes.marching_cubes(u, threshold)
122
123     b_max_np = bound_max.detach().cpu().numpy()
124     b_min_np = bound_min.detach().cpu().numpy()
125
126     vertices = vertices / (resolution - 1.0) * (b_max_np - b_min_np)[None, :] +
            b_min_np[None, :]
127     return vertices, triangles
128
129
130 class PSNRMeter:
131     def __init__(self):
132         self.V = 0
133         self.N = 0
134
135     def clear(self):
136         self.V = 0
137         self.N = 0
138
139     def prepare_inputs(self, *inputs):
140         outputs = []
141         for i, inp in enumerate(inputs):
142             if torch.is_tensor(inp):
143                 inp = inp.detach().cpu().numpy()
144             outputs.append(inp)
145
146         return outputs
147
148     def update(self, preds, truths):
149         preds, truths = self.prepare_inputs(preds, truths) # [B, N, 3] or [B, H,
                W, 3], range[0, 1]
150
151         # simplified since max_pixel_value is 1 here.
152         psnr = -10 * np.log10(np.mean((preds - truths) ** 2))
153
154         self.V += psnr
155         self.N += 1
156
157     def measure(self):
158         return self.V / self.N
159
160     def write(self, writer, global_step, prefix=""):
161         writer.add_scalar(os.path.join(prefix, "PSNR"), self.measure(),
                global_step)
162
163     def report(self):
164         return f'PSNR␣=␣{self.measure():.6f}'
165
166
167 class SSIMMeter:
168     def __init__(self, device=None):
169         self.V = 0
170         self.N = 0
171
172         self.device = device if device is not None else torch.device('cuda' if
                torch.cuda.is_available() else 'cpu')
173
174     def clear(self):
175         self.V = 0
176         self.N = 0
177
178     def prepare_inputs(self, *inputs):
```

```
        outputs = []
        for i, inp in enumerate(inputs):
            inp = inp.permute(0, 3, 1, 2).contiguous() # [B, 3, H, W]
            inp = inp.to(self.device)
            outputs.append(inp)
        return outputs

    def update(self, preds, truths):
        preds, truths = self.prepare_inputs(preds, truths) # [B, H, W, 3] --> [B
            , 3, H, W], range in [0, 1]

        ssim = structural_similarity_index_measure(preds, truths)

        self.V += ssim
        self.N += 1

    def measure(self):
        return self.V / self.N

    def write(self, writer, global_step, prefix=""):
        writer.add_scalar(os.path.join(prefix, "SSIM"), self.measure(),
            global_step)

    def report(self):
        return f'SSIM = {self.measure():.6f}'


def predefined_val_ts(img_id):

    aoi_id = img_id[:7]

    if aoi_id == "JAX_068":
        d = {"JAX_068_022_RGB": 8,
             "JAX_068_002_RGB": 8,
             "JAX_068_012_RGB": 1,
             "JAX_068_013_RGB": 1,
             "JAX_068_011_RGB": 1} #3
    elif aoi_id == "JAX_004":
        d = {"JAX_004_002_RGB": 0,
             "JAX_004_015_RGB": 0,
             "JAX_004_014_RGB": 0,
             "JAX_004_009_RGB": 5}
    elif aoi_id == "JAX_017":
        d = {"JAX_017_001_RGB": 8,
             "JAX_017_015_RGB": 0,
             "JAX_017_012_RGB": 10,
             "JAX_017_004_RGB": 5}
    elif aoi_id == "JAX_031":
        d = {"JAX_031_009_RGB": 5,
             "JAX_031_001_RGB": 5,
             "JAX_031_012_RGB": 10,
             "JAX_031_004_RGB": 0}
    elif aoi_id == "JAX_167":
        d = {"JAX_167_005_RGB": 8,
             "JAX_167_003_RGB": 0,
             "JAX_167_001_RGB": 0,
             "JAX_167_004_RGB": 0}
    elif aoi_id == "JAX_105":
        d = {"JAX_105_011_RGB": 5,
             "JAX_105_007_RGB": 5,
             "JAX_105_020_RGB": 0,
             "JAX_105_004_RGB": 0}
    elif aoi_id == "JAX_117":
```

```
        d = {"JAX_117_019_RGB": 5,
             "JAX_117_014_RGB": 8,
             "JAX_117_020_RGB": 0,
             "JAX_117_004_RGB": 0}
    elif aoi_id == "JAX_156":
        d = {"JAX_156_020_RGB": 0,
             "JAX_156_003_RGB": 0,
             "JAX_156_008_RGB": 5,
             "JAX_156_014_RGB": 10}
    elif aoi_id == "JAX_164":
        d = {"JAX_164_021_RGB": 5,
             "JAX_164_011_RGB": 5,
             "JAX_164_005_RGB": 5,
             "JAX_164_014_RGB": 10}
    elif aoi_id == "JAX_165":
        d = {"JAX_165_002_RGB": 10,
             "JAX_165_009_RGB": 8,
             "JAX_165_010_RGB": 8,
             "JAX_165_014_RGB": 10}
    elif aoi_id == "JAX_166":
        d = {"JAX_166_013_RGB": 0,
             "JAX_166_022_RGB": 3,
             "JAX_166_004_RGB": 10,
             "JAX_166_012_RGB": 10}
    elif aoi_id == "JAX_203":
        d = {"JAX_203_002_RGB": 10,
             "JAX_203_019_RGB": 10,
             "JAX_203_026_RGB": 5,
             "JAX_203_004_RGB": 17}
    elif aoi_id == "JAX_214":
        d = {"JAX_214_020_RGB": 0,
             "JAX_214_007_RGB": 8,
             "JAX_214_006_RGB": 8,
             "JAX_214_001_RGB": 18,
             "JAX_214_008_RGB": 2,
             "JAX_214_011_RGB": 17}
    elif aoi_id == "JAX_260":
        d = {"JAX_260_015_RGB": 0,
             "JAX_260_006_RGB": 3,
             "JAX_260_004_RGB": 10,
             "JAX_260_011_RGB": 10}
    elif aoi_id == "JAX_359":
        d = {"JAX_359_011_RGB": 0,
             "JAX_359_012_RGB": 3,
             "JAX_359_009_RGB": 10,
             "JAX_359_002_RGB": 10}
    elif aoi_id == "JAX_559":
        d = {"JAX_559_023_RGB": 8,
             "JAX_559_022_RGB": 8,
             "JAX_559_012_RGB": 0,
             "JAX_559_009_RGB": 5}
    elif aoi_id == "OMA_144":
        d = {"OMA_144_006_RGB": 0,
             "OMA_144_022_RGB": 8,
             "OMA_144_034_RGB": 0,
             "OMA_144_007_RGB": 10}
    elif aoi_id == "OMA_357":
        d = {"OMA_357_041_RGB": 10,
             "OMA_357_042_RGB": 8,
             "OMA_357_034_RGB": 0,
             "OMA_357_007_RGB": 5}
    else:
        return None
```

```
303     return d[img_id]
304
305 def get_parameters(models):
306     """
307     Get all model parameters recursively
308     models can be a list, a dictionary or a single pytorch model
309     """
310     parameters = []
311     if isinstance(models, list):
312         for model in models:
313             parameters += get_parameters(model)
314     elif isinstance(models, dict):
315         for model in models.values():
316             parameters += get_parameters(model)
317     else:
318         # models is actually a single pytorch model
319         parameters += list(models.parameters())
320     return parameters
321
322
323 def _flatten(l):
324     return [item.cpu() for sublist in l for item in sublist]
```

9.4 Appendix D

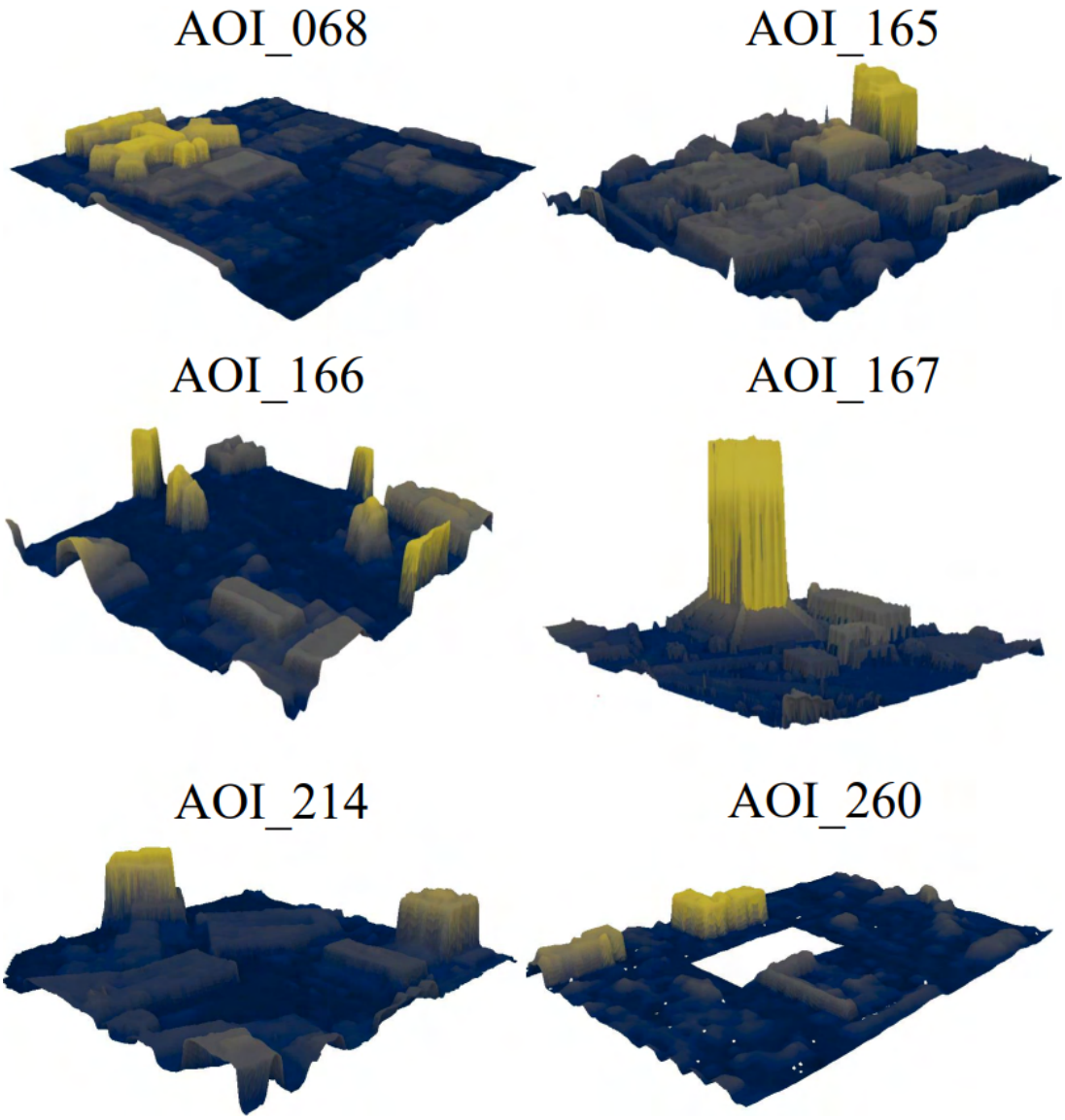


Figure 41: 3D visualizations of the urban AOIs.

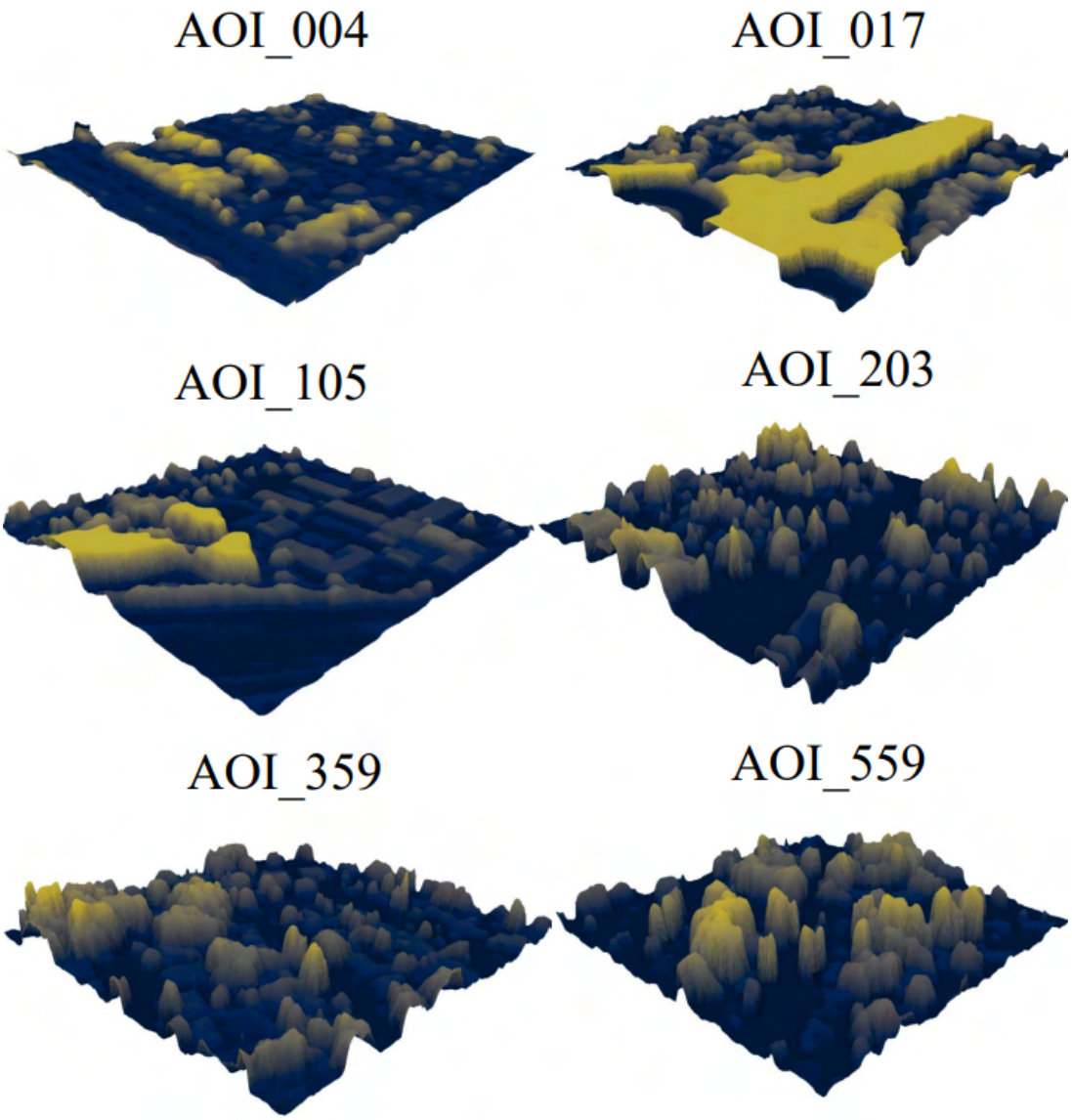


Figure 42: 3D visualizations of the rural AOIs.

List of Figures

List of Tables

10 References

AHN. (2024). *Actueel hoogtebestand nederland*. Retrieved November 6, 2024, from https://www.ahn.nl/

Billouard, C., Derksen, D., Sarrazin, E., & Vallet, B. (2024). Sat-ngp: Unleashing neural graphics primitives for fast relightable transient-free 3d reconstruction from satellite imagery. *arXiv preprint arXiv:2403.18711*. https://doi.org/10.1109/IGARSS53475.2024.10641775

Bosch, M., Kurtz, Z., Hagstrom, S., & Brown, M. (2016). A multiple view stereo benchmark for satellite imagery. *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 1–9. https://doi.org/10.1109/AIPR.2016.8010543

Bullinger, S., Bodensteiner, C., & Arens, M. (2021). 3d surface reconstruction from multi-date satellite images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII-B2-2021*, 313–320. https://doi.org/10.5194/isprs-archives-XLIII-B2-2021-313-2021

Chakraborty, D., Ghosh, K., Sukma, Z., Kim, I. K., Ramaswamy, L., Bhandarkar, S. M., & Mishra, D. R. (2025). An empirical evaluation of the impact of solar correction in nerfs for satellite imagery. In A. Antonacopoulos, S. Chaudhuri, R. Chellappa, C.-L. Liu, S. Bhattacharya, & U. Pal (Eds.), *Pattern recognition* (pp. 161–179). Springer Nature Switzerland. https://doi.org//10.1007/978-3-031-78456-9_11

Chang, J.-R., & Chen, Y.-S. (2018). Pyramid stereo matching network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5410–5418. https://doi.org/10.1109/CVPR.2018.00567

Claesen, R. (2024). *Creating and evaluating digital elevation models from satellite imagery* [Master's thesis, TU Delft]. http://resolver.tudelft.nl/uuid:25109dee-20ee-499f-b35b-4d1fda61a140

de Franchis, C., Meinhardt-Llopis, E., Michel, J., Morel, J.-M., & Facciolo, G. (2014). An automatic and modular stereo pipeline for pushbroom images. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *II-3*, 49–56. https://doi.org/10.5194/isprsannals-II-3-49-2014

Demarez, V., Helen, F., Marais-Sicre, C., & Baup, F. (2019). In-season mapping of irrigated crops using landsat 8 and sentinel-1 time series. *Remote Sensing*, *11*, 118. https://doi.org/10.3390/rs11020118

Deng, K., Liu, A., Zhu, J.-Y., & Ramanan, D. (2022). Depth-supervised nerf: Fewer views and faster training for free. https://doi.org/10.48550/arXiv.2107.02791

Derksen, D., & Izzo, D. (2021). Shadow neural radiance fields for multi-view satellite photogrammetry. https://doi.org/10.48550/arXiv.2104.09877

Deseilligny, M., & Paparoditis, N. (2006). A multiresolution and optimization-based image matching approach: An application to surface reconstruction from spot5-hrs stereo imagery. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *36-1*. https://api.semanticscholar.org/CorpusID:13930506

EOS. (2024). *Superview 1 satellite images*. Retrieved November 15, 2024, from https://eos.com/find-satellite/superview-1/

ESA. (2025). *Worldview-3*. Retrieved February 12, 2025, from https://earth.esa.int/eogateway/missions/worldview-3#instruments-section

Förstner, W., & Wrobel, B. P. (2016). *Photogrammetric computer vision* (Vol. 11). Springer International Publishing. https://doi.org/10.1007/978-3-319-11550-4

Furukawa, Y., & Hernández, C. (2015). Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, *9*(1-2), 1–148. https://doi.org/10.1561/0600000052

Hartmann, W., Galliani, S., Havlena, M., Van Gool, L., & Schindler, K. (2017). Learned multi-patch similarity. *2017 IEEE International Conference on Computer Vision (ICCV)*, 1595–1603. https://doi.org/10.1109/ICCV.2017.176

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *2*, 807–814 vol. 2. https://doi.org/10.1109/CVPR.2005.56

Hu, Z., Zhang, K., & Liu, Y. (2025). Edge constrained dsm refinement based on shading from high resolution multi-view satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 1–12. https://doi.org/10.1109/JSTARS.2025.3526817

Huang, P.-H., Matzen, K., Kopf, J., Ahuja, N., & Huang, J.-B. (2018). Deepmvs: Learning multi-view stereopsis. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2821–2830. https://doi.org/10.48550/arXiv.1804.00650

Kerbl, B., Kopanas, G., Leimkuehler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, *42*(4). https://doi.org/10.1145/3592433

Kim, K. J. (2022). *Nerf: Representing scenes as neural radiance fields for view synthesis*. Retrieved September 23, 2024, from https://velog.io/@kimkj38/NeRF-Representing-Scenes-as-Neural-Radiance-Fields-for-View-Synthesis

Labarre, S., Jacquemoud, S., Ferrari, C., Delorme, A., Derrien, A., Grandin, R., Jalludin, M., Lemaître, F., Métois, M., Pierrot-Deseilligny, M., Rupnik, E., & Tanguy, B. (2019). Retrieving soil surface roughness with the hapke photometric model: Confrontation with the ground truth. *Remote Sensing of Environment*, *225*, 1–15. https://doi.org//10.1016/j.rse.2019.02.014

Lastilla, L., Belloni, V., Ravanelli, R., & Crespi, M. (2021). Dsm generation from single and cross-sensor multi-view satellite images using the new agisoft metashape: The case studies of trento and matera (italy). *Remote Sensing*, *13*(4). https://doi.org/10.3390/rs13040593

Le Saux, B., Yokoya, N., Hänsch, R., & Brown, M. (2019). Data fusion contest 2019 (dfc2019): Large-scale semantic 3d reconstruction. https://doi.org/10.21227/C6TM-VW12

Ledoux, H., Arroyo Ohori, K., Peters, R., & Pronk, M. (2024, May). *Computational modelling of terrains*. Self-published. https://doi.org/10.5281/zenodo.11257197

Li, S. (2024). *Enhancing 3d model for urban area with neural representations* [Master's thesis, TU Delft]. http://resolver.tudelft.nl/uuid:c9211a3c-2b54-4895-8a85-06dcdfe6df3b

Lin, C.-H., Ma, W.-C., Torralba, A., & Lucey, S. (2021). Barf: Bundle-adjusting neural radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 5721–5731. https://doi.org/10.1109/ICCV48922.2021.00569

Liu, X. (2008). Airborne lidar for dem generation: Some critical issues. *Progress in Physical Geography: Earth and Environment*, *32*(1), 31–49. https://doi.org/10.1177/0309133308089496

Marí, R., de Franchis, C., Meinhardt-Llopis, E., Anger, J., & Facciolo, G. (2021). A Generic Bundle Adjustment Methodology for Indirect RPC Model Refinement of Satellite Imagery. *Image Processing On Line*, *11*, 344–373. https://doi.org/10.5201/ipol.2021.352

Marí, R., Facciolo, G., & Ehret, T. (2022). Sat-nerf: Learning multi-view satellite photogrammetry with transient objects and shadow modeling using rpc cameras. https://doi.org/10.48550/arXiv.2203.08896

Marí, R., Facciolo, G., & Ehret, T. (2023). Multi-date earth observation nerf: The detail is in the shadows. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2035–2045. https://doi.org/10.1109/CVPRW59228.2023.00197

Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A., & Duckworth, D. (2021). NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. *CVPR*. https://doi.org/10.48550/arXiv.2008.02268

McClean, F., Dawson, R., & Kilsby, C. (2020). Implications of using global digital elevation models for flood risk analysis in cities. *Water Resources Research*, *56*. https://doi.org/10.1029/2020WR028241

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer vision – eccv 2020* (pp. 405–421). Springer International Publishing.

Misra, D. (2020). Mish: A self regularized non-monotonic activation function. https://doi.org/10.48550/arXiv.1908.08681

Moons, T., van Gool, L., & Vergauwen, M. (2009). *3d reconstruction from multiple images, part 1: Principles*. Now Publishers Inc.

Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, *41*(4). https://doi.org/10.1145/3528223.3530127

National Ocean Service, N. O., & Administration, A. (2024). *What is lidar?* Retrieved January 17, 2025, from https://oceanservice.noaa.gov/facts/lidar.html

NSO. (2024). *Satellietdataportaal.nl netherlands space office*. Retrieved November 18, 2024, from https://viewer.satellietdataportaal.nl/@52.06262,4.691162,8

Qin, R., Tian, J., & Reinartz, P. (2016). 3d change detection – approaches and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, *122*, 41–56. https://doi.org/10.1016/j.isprsjprs.2016.09.013

Qin, W., Herman, J. R., & Ahmad, Z. (2001). A fast, accurate algorithm to account for non-lambertian surface effects on toa radiance. *Journal of Geophysical Research: Atmospheres*, *106*(D19), 22671–22684. https://doi.org//10.1029/2001JD900215

Rahman, H., Pinty, B., & Verstraete, M. M. (1993). Coupled surface-atmosphere reflectance (csar) model: 2. semiempirical surface model usable with noaa advanced very high resolution radiometer data. *Journal of Geophysical Research: Atmospheres*, *98*(D11), 20791–20801. https://doi.org//10.1029/93JD02072

Rematas, K., Liu, A., Srinivasan, P., Barron, J., Tagliasacchi, A., Funkhouser, T., & Ferrari, V. (2022). Urban radiance fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12922–12932. https://doi.org/10.1109/CVPR52688.2022.01259

Riegler, G., Ulusoy, A. O., Bischof, H., & Geiger, A. (2017). Octnetfusion: Learning depth fusion from data. *2017 International Conference on 3D Vision (3DV)*, 57–66. https://doi.org/10.48550/arXiv.1704.01047

Roessle, B., Barron, J. T., Mildenhall, B., Srinivasan, P. P., & Nießner, M. (2022). Dense depth priors for neural radiance fields from sparse input views. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12882–12891. https://doi.org/10.1109/CVPR52688.2022.01255

Rupnik, E., Pierrot-Deseilligny, M., & Delorme, A. (2018). 3d reconstruction from multi-view vhr-satellite images in micmac. *ISPRS Journal of Photogrammetry and Remote Sensing*, *139*, 201–211. https://doi.org/10.1016/j.isprsjprs.2018.03.016

Sabour, S., Vora, S., Duckworth, D., Krasin, I., Fleet, D. J., & Tagliasacchi, A. (2023). Robustnerf: Ignoring distractors with robust losses. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 20626–20636. https://doi.org/10.1109/CVPR52729.2023.01976

Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 4104–4113. https://doi.org/10.1109/CVPR.2016.445

Simard, M., Zhang, K., Rivera-Monroy, V. H., Ross, M. S., Ruiz, P. L., Castañeda-Moya, E., Twilley, R. R., & Rodriguez, E. (2006). Mapping height and biomass of mangrove forests in everglades national park with srtm elevation data. *Photogrammetric Engineering and Remote Sensing*, *72*, 299–311. https://doi.org/10.14358/PERS.72.3.299

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Proceedings of the 34th International Conference on Neural Information Processing Systems*. https://doi.org//10.48550/arXiv.2006.09661

Stathopoulou, E. K., Battisti, R., Cernea, D., Georgopoulos, A., & Remondino, F. (2023). Multiple view stereo with quadtree-guided priors. *ISPRS Journal of Photogrammetry and Remote Sensing*, *196*, 197–209. https://doi.org/10.1016/j.isprsjprs.2022.12.013

Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B. P., Srinivasan, P., Barron, J. T., & Kretzschmar, H. (2022). Block-nerf: Scalable large scene neural view synthesis. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8238–8248. https://doi.org/10.1109/CVPR52688.2022.00807

Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., Mcallister, D., Kerr, J., & Kanazawa, A. (2023). Nerfstudio: A modular framework for neural radiance field development. *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, 1–12. https://doi.org/10.1145/3588432.3591516

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J.-Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., & Zollhöfer, M. (2020). State of the art on neural rendering. https://doi.org/10.48550/arXiv.2004.03805

Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (2000). Bundle adjustment — a modern synthesis. In B. Triggs, A. Zisserman, & R. Szeliski (Eds.), *Vision algorithms: Theory and practice* (pp. 298–372). Springer Berlin Heidelberg.

Wan, Q., Guan, Y., Zhao, Q., Wen, X., & She, J. (2024). Constraining the geometry of nerfs for accurate dsm generation from multi-view satellite images. *ISPRS International Journal of Geo-Information*, *13*, 243. https://doi.org/10.3390/ijgi13070243

Xie, S., Zhang, L., Jeon, G., & Yang, X. (2023). Remote sensing neural radiance fields for multi-view satellite photogrammetry. *Remote Sensing*, *15*, 3808. https://doi.org/10.3390/rs15153808

Zagoruyko, S., & Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4353–4361. https://doi.org/10.48550/arXiv.1504.03641

Zhang, G., & Yuan, X. (2006). On rpc model of satellite imagery. *Geo-spatial Information Science*, *9*, 285–292. https://doi.org/10.1007/BF02826742

Zhang, K., Sun, J., & Snavely, N. (2019). Leveraging vision reconstruction pipelines for satellite imagery. https://doi.org//10.48550/arXiv.1910.02989

Zhang, L., & Rupnik, E. (2023). Sparsesat-nerf: Dense depth supervised neural radiance fields for sparse satellite images. https://doi.org/10.48550/arXiv.2309.00277

Zhang, L., Rupnik, E., Nguyen, T. D., Jacquemoud, S., & Klinger, Y. (2024). Brdf-nerf: Neural radiance fields with optical satellite images and brdf modelling. *arXiv preprint arXiv:2409.12014*. https://doi.org/10.48550/arXiv.2409.12014

Zhang, T., Zhou, Y., Li, Y., & Wei, X. (2024). Satensorf: Fast satellite tensorial radiance field for multidate satellite imagery of large size. *IEEE Transactions on Geoscience and Remote Sensing*, *62*, 1–15. https://doi.org/10.1109/TGRS.2024.3382632