SCHOOL OF ENGINEERING AND DESIGN

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Inpainting of unseen façade objects using deep learning methods

Thomas Fröch

SCHOOL OF ENGINEERING AND DESIGN

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Inpainting of unseen façade objects using deep learning methods

Author:	Thomas Fröch
Supervisor:	Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe
Advisor:	M.Sc. Olaf Wysocki, Dr. Yan Xia, M.Sc. M.Sc. Benedikt Schwab
Submission Date:	30.11.2023

Slightly updated version with spelling and grammatical corrections.

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 30.11.2023 Thomas Fröch

Acknowledgments

I would like to express my deep gratitude to my Master's thesis mentors: M.Sc. Olaf Wysocki, Dr.-Ing. Yan Xia, M.Sc. M.Sc. Benedikt Schwab, and Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe. Their extensive expertise, profound knowledge, and invaluable assistance were indispensable for the realization of this thesis in its current form. The success of this thesis owes much to the significant contribution of Prof. Dr. Daniel Cremers, who graciously provided the computational resources essential for my work. I am profoundly grateful for his generosity and kindness, which played a pivotal role in realizing my research. I also want to extend my appreciation to M.Sc. Philip-Roman Hirt and Quirin Lohr for their kindness and unwavering support. Finally, I express my heartfelt gratitude to everyone who stood with me during the six months of my thesis. Without your support, this thesis would not have been possible.

Abstract

In the realm of 3D reconstruction pipelines, 2D conflict maps that indicate the presence openings in façades such as windows and doors, represent an intermediate output [1]. However, these maps often fall short of completeness due to insufficient coverage or occlusions caused by objects such as vegetation. This research delves into the exploration of deep learning strategies to address this limitation by inpainting unseen façade objects into the 2D conflict maps. The central focus of this study revolves around deploying the Stable Diffusion inpainting model [2, 3] and the LaMa GAN [4] for this purpose. Specifically, I investigate in the potential of personalizing a pre-trained Stable Diffusion inpainting model with Dreambooth [5] to facilitate its application in inpainting unseen façade objects within 2D conflict maps. Simultaneously, I undergo training for the LaMa GAN with a similar objective. By utilizing synthetic conflict maps that are derived from randomly generated semantic city models [6] and such that are derived from databases of annotated optical façade images [7] as data for training the LaMa GAN and personalisation a pre-trained Stable Diffusion with Dreambooth, I investigate if such data sources can facilitate the deployment of deep learning models. My results demonstrate the general capability of deep learning based methods for inpainting unseen objects into the 2D conflict maps. I find that the personalisation with Dreambooth yields improvements regarding the behaviour of diffusion based models when considering tree-shaped masks. This, and the successful training of the LaMa GAN demonstrate the utility of synthetic conflict maps and such derived from annotated images. The insights gained in this paper can be applied to existing pipelines for the reconstruction of LOD3 models. This way my work contributes to improving the reconstruction accuracy of such approaches. It also serves as the basis for further exploration.

Keywords: machine learning, image inpainting, façade reconstruction, LOD3 reconstruction, mobile laser scanning, Stable Diffusion, Dreambooth, LaMa GAN

Contents

1 Introduction

Driven by pressing issues such as global climate change and the transport transition in Germany, there is a growing interest in using detailed semantic 3D building models for various applications such as façade solar potential estimation and testing automated driving functions [8, 9]. However, the current demand for such models can only be partially met due to the complex reconstruction process involved. Leveraging point clouds from Mobile Laser Scanning (MLS) offers a promising way to reconstruct façade details [1, 8, 10, 11]. An intermediate result in the reconstruction process is a 2D-conflict map. It indicates semantic conflicts between the façade-surface of an LOD2 model and an MLS point cloud that represents the respective façade. Such conflicts commonly result from the presence of openings such as windows, doors, and underpasses. This information can be used for LOD3 model reconstruction [1]. A challenge remains in dealing with incomplete data, for example, caused by occlusions due to the presence of vegetation.

Semantic image inpainting describes the process of using semantic information within an image to artificially fill in missing regions of an image [12]. The rapid developments in the field of machine learning have made a large number of models efficiently applicable for a variety of purposes, including semantic image inpainting [2, 4]. However, filling regions with semantically meaningful content and maintaining consistency with the overall structure of an image remains challenging [13, 14].

Within the framework of this thesis, I aim to contribute to improving the reconstruction accuracy of semantic 3D building models at level of detail 3 (LOD3) by investigating strategies for the semantic completion of 2D-conflict maps. This work addresses the following research questions:

- Can a deep-learning-based method be utilized for the semantic inpainting of unseen façade objects in 2D conflict maps to obtain a completed conflict map of a façade?
- Can synthetic semantic city models and the ubiquity of annotated photographic image databases be leveraged to facilitate the training of deep-learning models for the completion of 2D conflict maps?

- Can the personalization of a Diffusion Probabilistic Model (DM) with Dreambooth facilitate its deployment for the inpainting of unseen façade objects into 2D conflict maps?

My methodology introduces an approach for determining 2D-conflict maps by combining semantic LOD2 building models and corresponding MLS point clouds. I personalize a pre-trained Diffusion model with Dreambooth and train the LaMa GAN on a dataset consisting of synthetic conflict maps and those derived from annotated images. The results, findings, and insights of this work may be used within pipelines for the reconstruction of LOD3 building models to facilitate their accuracy. The implementation and the digital appendix are available in a designated GitHub repository[1].

[1]https://github.com/ThomasFroech/InpaintingofUnseenFacadeObjects

2 Theoretical background

2.1 Semantic city models

Besides the ability to provide geometric and visual information on topographic objects, semantic city models also provide information on structures, taxonomies, and aggregations. Such semantic information distinguishes semantic city models from models designed solely for visualization purposes [15]. Semantic city models focus on the scale of city quarters, cities, and even complete regions. This focus on scale distinguishes semantic city models from other semantic 3D models of the built environment, such as Building Information Modeling (BIM) [16].

2.1.1 CityGML

CityGML is an open data model that is intended for the representation and exchange of virtual semantic 3D city models [16]. It can be comprehended as the outcome of an intensive process aimed at achieving uniform definitions and a unified understanding of the various components within a 3D city model. CityGML, established by the Open Geospatial Consortium (OGC), has held the status of an open international OGC standard since 2008 [16, 17]. Subsequently, three versions of CityGML have been released: 1.0, 2.0, and the most recent version, 3.0 [18]. The foundation of CityGML's data model is the ISO 19100 standards family framework for modeling geographic features. The implementation is realized as an application schema for Geography Markup Language (GML), based on the Extensible Markup Language (XML). Over the years since its initial publication, CityGML has continued to gain popularity, and CityGML models are now widely adopted [9].

2.1.2 Representation of buildings with CityGML

Multiscale representation

CityGML facilitates the representation of buildings at various semantic levels of Detail (LOD). In CityGML 2.0, these levels span from depictions only encompassing building footprints (LOD 0) to highly intricate representations that encompass both exterior and interior details (LOD 4). There are five progressively refined LOD levels in CitygML

LOD 0	Only footprint of a building
LOD 1	Cuboid without detailed roof or façade structures
LOD 2	Cuboid with a detailed roof structure.
LOD 3	Facade details augment the LOD 2 structure
LOD 4	Interior augments the LOD 3 structure

Table 2.1: Primary characteristics of the different LOD levels [16]

2.0. Table 2.1 provides a comprehensive description of each LOD level, while Figure 2.1 illustrates the five different LOD levels.

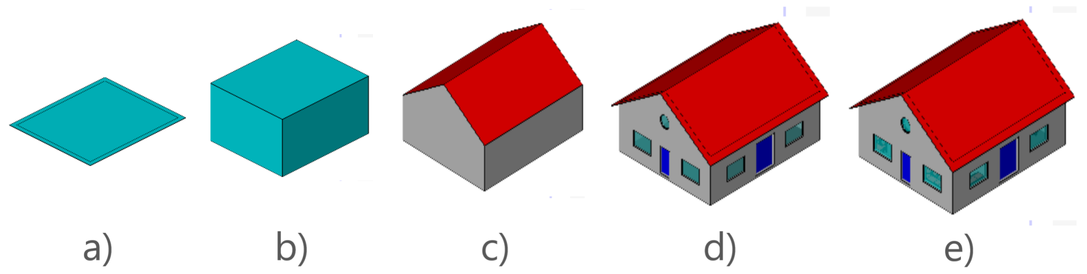


Figure 2.1: FZK-House in different LOD levels: a) LOD 0, b) LOD 1 c) LOD 2, d) LOD 3, e) LOD 4, Figures based on [19]

The latest version, CityGML 3.0, follows a new LOD concept that comprises only four LOD levels. The interior and exterior of buildings can be modeled at all of these four levels. The LOD level of the exterior does not need to match that of the interior.

Another possibility for the specification of multi-scale representations are the 16 geometric LODs summarized in Figure 2.2. These levels, in my work referred to as Delft LODs, represent a finer gradation regarding the details that featured details than the regular LOD levels.

Modeling of geometry

CityGML generally follows the Boundary Representation (B-Rep) model, where individual objects are represented by their respective boundaries. A central concept of representing the built environment with CityGML is the decomposition into meaningful objects with precise semantics [16]. This also applies to the representation of individual buildings within a semantic city model. In CityGML 2, a building comprises a set of different objects such as, among others, *WallSurface* and *RoofSurface*. According to the B-Rep model, these objects are represented by their boundaries. This decomposition

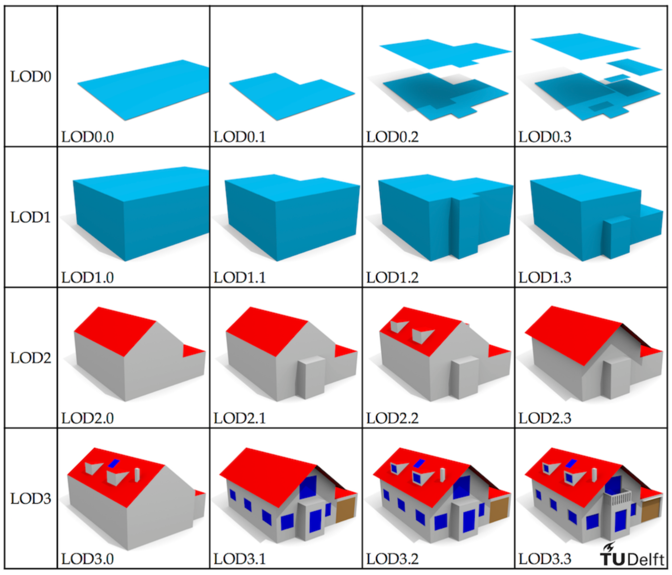


Figure 2.2: Overview of the 16 Delft LOD levels that can be specified. Figure from [20]

finally leads to linear objects such as interior and exterior rings represented by the coordinates of the individual points they comprise [21].

2.1.3 Applications of semantic city models

Semantic city models exhibit a broad spectrum of applications in diverse domains, such as urban planning, traffic management, environmental engineering, and numerous others [9]. Specifically, the need for LOD3 building models, distinguished by their detailed façade representations, is often indispensable. Disciplines that require such detailed façade representations include testing automated driving functions or estimating the solar potential of façades [22]. While LOD2 models are almost ubiquitous today, LOD3 models remain scarce [8].

2.2 Mobile Laser Scanning (MLS)

2.2.1 Basic information

Mobile Laser Scanning (MLS) is a method for the fast and accurate acquisition of 3D point cloud data [23]. A mobile laser scanner is mounted on a mobile platform such as a vehicle or an aircraft. The entire system typically includes the following three subsystems [23]:

- Global Positioning System (GPS) / Inertial Navigation System (INS)
- Mobile laser scanner
- Optical camera system

The GPS/INS unit enables geo-referencing of the acquired point data while the laser scanner acquires the point data itself. The camera system captures images of the scenery around the systems simultaneously with the GPS/INS and laser scanner measurements [23].

2.2.2 Characteristic data

The large number of points in MLS point clouds comes at the cost of large file sizes, imposing significant requisites regarding of storage capacity, computational resources for processing, and the concomitant complexities associated with managing such datasets.

An important aspect regarding point density is the distance of the sensor to an object. An increase in distance is accompanied by a decrease in point density. This circumstance imposes challenges regarding building façades. The accuracy and point density in the upper floors of buildings will generally be lower than in the lower floors. Particularly with 3D reconstruction methods that rely on the points directly, this can cause difficulties.

The environment in which MLS point clouds are captured typically encompasses elements such as vehicles, trees, and assorted obstacles that frequently obscure the view of building façades. Figure 2.3 shows a real example of such occlusions. Trees obstruct the line of sight to the building façade, resulting in areas of the façade where data is missing, corresponding to the contours of the obstructing trees.

2.3 3D reconstruction

2.3.1 Concept of 3D reconstruction

The challenges of reconstructing state-of-the-art LOD 3 models, coupled with the consequent scarcity, ensure great research interest in this field of research [10, 22, 24–27]. Different data sources, such as optical images, MLS point clouds, and oblique Airborne Laser Scanning (ALS) point clouds, are utilized for detailed 3D façade reconstruction [25, 28, 29]. The use of combinations of different data sources within a reconstruction pipeline has also been explored. An example is the study of Wysocki *et al.*, where MLS point clouds and optical images are combined using Bayesian networks. Their

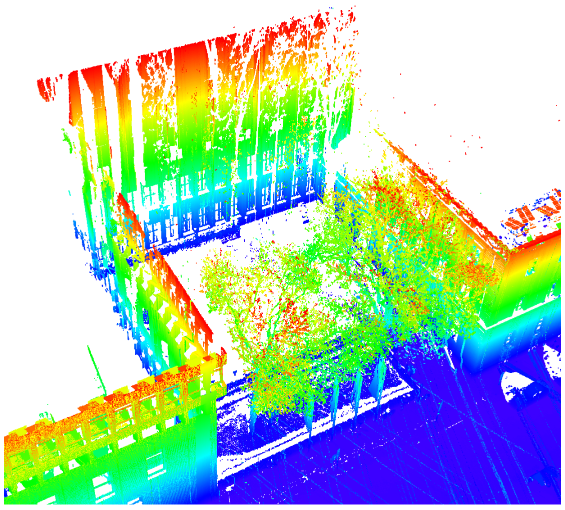


Figure 2.3: Exemplary occlusions by trees in the MF point cloud (height color-coded)[1]

reconstruction pipeline represents a powerful and sophisticated approach for refining LOD2 semantic building models [10].

2.3.2 Complexity of façade details

The complexity of façade details represents one of the significant challenges imposed by this subject of 3D reconstruction. Windows, for example, can have an almost unlimited variety of shapes. Standard methods for circumventing this issue are the introduction of rectangularity assumptions or the use of bounding boxes [30]. An example of such an assumption of rectangularity is the study of Hoegner and Gleixner. While they demonstrate good results with a detection rate of 86%, they strictly simplify windows as rectangles [29]. Recently, an approach for 3D reconstruction that does not require such assumptions has been introduced. The method incorporates semi-global information in binary images of projected façade details into a modified Bag-of-Words approach to adapt to individual shapes. However, further development is required before it becomes applicable to larger scales [11].

2.3.3 3D reconstruction and standards

Incorporating reconstructed 3D façade details into existing standards for semantic city models such as CityGML represents another challenge in 3D reconstruction. It is crucial to ensure the seamless integration of features, identified within the framework of a reconstruction pipeline, into existing building models without affecting their applications. According to Wysocki *et al.*, the positions of specific components present in the LOD 2 model (Ground Surface, Roof Surface, Wall Surface) should, in general, not be altered during the refinement process. Such an action could impose inconsistencies in the resulting refined dataset [10]. Confidence information for every reconstructed feature could be incorporated into the refined model. Such additional information could be valuable for decision-making process, for example, in navigation applications [10].

2.4 Generative Adversarial Networks

2.4.1 General properties of GANs

Generative Adversarial Network (GAN)s are a sub-type of generative artificial neural networks [31, 32]. They aim at implicitly modeling a target data distribution to artificially generate new samples of this target distribution [31]. Generally, GANs consist of two individual competing networks (adversaries): the generator and the discriminator. Training GANs is an adversarial process [33] that can be described as a "zero-sum game" between both networks [34]. Figure 2.4 provides a very coarse example of a very basic GAN. The generator tries to create new samples, while the discriminator tries to determine whether a given sample is real or artificially generated [31]. The training objective of the discriminator is to optimize its classification accuracy to differentiate between authentic samples and samples that are artificially generated [31]. In contrast, the generator is trained to generate data samples so that the discriminator cannot distinguish artificial samples from real ones [31]. The underlying principle of the training of GANs is described by the value function (equation 2.1). With D(...) being the output of the discriminator given an arbitrary input sample and G(z) being a sample generated by the generator according to a random input vector z [33].

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

Ideally, the training continues until neither of the two networks can improve on its goal. This theoretical condition is called the Nash equilibrium [34]. Reaching this point is exceedingly difficult because of the often unstable behavior of GANs summarized in the next section. This renders the Nash equilibrium infeasible [35].

2.4.2 Training GANs

Training GANs has proven to be a difficult task due to various reasons. In particular, the discriminator network is known for its problematic behavior and instability during training [36]. The following problems are commonly faced in this context:

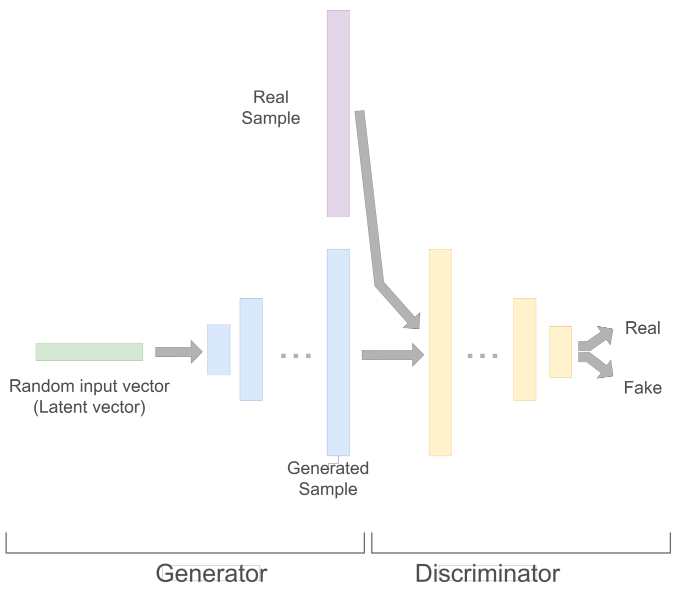


Figure 2.4: Schema of a basic GAN

- Necessity of large amounts of training data [37]
- Mode collapse [38, 39].
- Non-convergent training [39]
- Gradient vanishing or explosion [39]
- Uninformative gradients [40]
- Underfitting [41]
- Overfitting [41]

Great effort has been put into improving the training of GANs. As an example, progress has been made by the introduction of Wasserstein GAN (WGAN)s by Arjovsky *et al.* in 2017 as an alternative way of training GANs in a more stable way [42]. As another example, Miyato *et al.* introduce spectral normalization as a regularization technique to stabilize the training process by controlling the Lipschitz constant of a network. The latter limits the amount of change allowed in a network's output when its input is altered [36, 43]. Besides these two examples, various different techniques to stabilize the training of GANs exist [35, 39].

2.5 Diffusion Probabilistic Models

Diffusion Probabilistic Models, often referred to as Diffusion Model (DM), represent another class of generative models [2, 32]. The underlying principle is the gradual transformation of a given probability distribution into another. This idea originates from the domain of nonequilibrium statistical physics [2, 44, 45]. DMs are trained to learn a probability distribution by applying this concept of probability distribution transformations. A given input, the starting state, is gradually transformed into a known and well-behaved probability distribution, the prior state [32]. The Gaussian distribution is often chosen as a prior state [32]. This transformation process, referred to as the diffusion process, forward process, or forward trajectory, can be interpreted as the destruction of any structure within the given data by the subsequent addition of noise. The forward process is inverted by applying an iterative denoising process, thus converting the prior back to the starting state. This process is commonly referred to as the reverse process, reverse trajectory, or denoising process [2, 32, 45]. Figure 2.5 illustrates the general concept of DMs, with the starting state x , the prior state z, the reconstructed starting state $\tilde{x}$, the forward process $q(x_t|x_{t-1})$ and the reverse process $p_\theta(x_{t-1}|x_t)$.

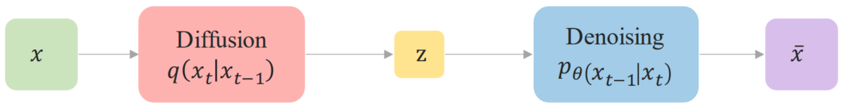


Figure 2.5: Schema: General concept of DMs. Figure from: [32]

2.6 Image inpainting

2.6.1 Taxonomy of image inpainting strategies

Image inpainting has been a challenging discipline with diverse applications such as image restoration, object removal, or image completion for a long time [13, 37]. Existing image inpainting approaches can be classified according to the schema presented in Figure 2.6. Traditional methods include many different approaches, such as diffusion-based methods, example-based texture and structure synthesis, and sparse representation methods [46]. Often, such traditional methods rely on the solving of Partial Differential Equations (PDEs). Examples of such approaches are the inpainting method by Telea and inpainting based on the Navier-Stokes PDEs [47, 48]. Many studies have used different varieties of GANs for image inpainting [46].

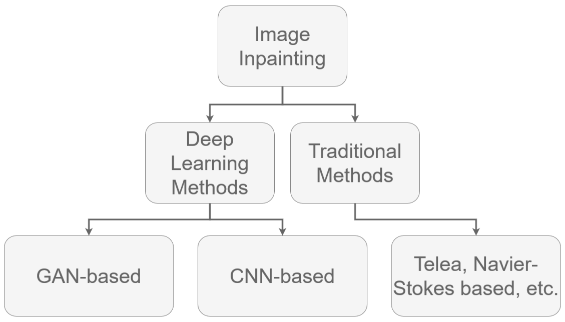


Figure 2.6: Hierarchy of image inpainting strategies. Figure adapted from [46]

2.6.2 Key challenges in semantic image inpainting

The key challenges in semantic image inpainting are to complete regions with semantically meaningful content [13] and to ensure consistency with the overall image structure [14]. Incorporating information that is far away from the region that should be filled has proven to be difficult, as existing methods often produce blurry or overly smooth results [49, 50]. Generalizability is also a desirable property of an image inpainting method [46]. The shape of the missing regions in an image can make image inpainting more difficult, too [13]. Many studies focus on the filling of missing regions of regular shape [13]. Existing approaches are often specialized for distinct purposes, such as the completion of faces [46].

3 Related work

3.1 Conflict maps

Wysocki *et al.* introduce a method for the reconstruction of underpasses in semantic LOD2 city models from co-registered MLS point clouds [8, 22]. As a part of their proposed processing pipeline, they introduce a probabilistic approach for identifying conflicts between the semantic LOD2 city models and the MLS point clouds. Such conflicts are characterized by a mismatch between the semantic city model and the corresponding MLS point cloud. Their methodology is based on an occupancy grid realized as an octree structure, where the size of the voxels corresponds to the combined uncertainty of the MLS measurements and the semantic city model. Ray casting is utilized for the identification of the conflicts. Figure 3.1 provides a visual impression of their concept. The rays are defined by the sensor point (s_i) as the shared origin and a direction that is defined by the direction to the measured point (p_i) in the MLS point cloud. According to the evaluation of this step, the voxels are classified as empty, occupied, unknown, confirmed, or conflicted. Lastly, a texture map, as it is exemplarily depicted in Figure 3.2, is established from the results of this analysis. Conflicting areas are indicated in red, confirming areas in green. Occluded areas without information are represented in grey [8, 22]. In this work, we refer to a texture map that indicates conflicts between an LOD2 model and a MLS point cloud as a conflict map.

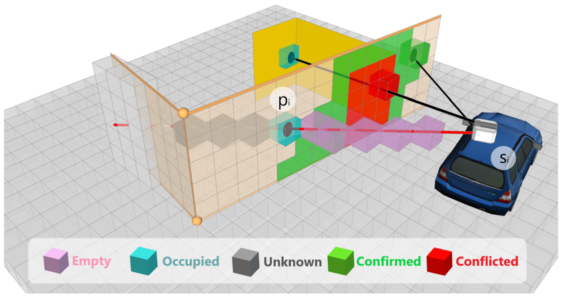


Figure 3.1: Ray casting on a 3D octree grid, Figure from: [22]

As already implicitly suggested in Figure 3.1, significant conflicts can predominantly be attributed to the presence of openings in the facade, such as windows or underpasses.

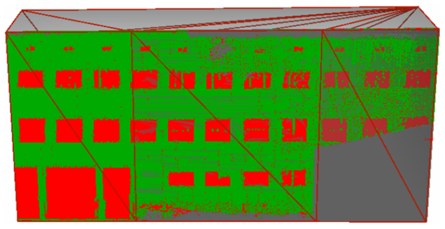


Figure 3.2: Exemplary conflict map of a building from the TUM-Façade dataset. Green: confirming areas, red: conflicting areas, grey: areas without information (occluded areas) Figure from: [22]

The permeability of glass regarding laser pulses is of particular significance in this context [25]. This phenomenon can be leveraged for the 3D reconstruction of LOD3 semantic building models from MLS point clouds.

Information on conflicts between a semantic city model and a corresponding point cloud can also be used directly at the level of the individual points without further processing steps. In the study of Yahya, conflicting points are identified to selectively extract points from a point cloud exclusively representing the facade structure [51].

3.2 Synthetic semantic city model generation

To circumvent the issue of the scarcity of LOD3 semantic city models and to facilitate the development of new applications for such models, Biljecki *et al.* introduce Random3Dcity, a method for the automatic generation of random semantic 3D city models [6]. Their application offers the possibility of generating synthetic CityGML data in Delft LOD levels. For generating the synthetic semantic city models at any Delft LOD, Biljecki *et al.* make use of a procedual approach, which represents a standard method for obtaining design models. Such an approach is based on the definition of rules and guidelines regarding the properties of the output. In the context of the implementation of Biljecki *et al.*, it offers a high degree of flexibility as it allows the incorporation and modification of various specifications regarding the semantic city model to be produced. Their workflow comprises two distinct modules that operate independently, the procedual modeler and the 3D data realization. First, an XML document that contains the configurations of the random 3D semantic model is obtained from the procedual modeler. Each building is characterized by parameters such as width or roof height. These are determined in a top-down approach from coarse to fine, considering a set of rules for the different features. Doors, for example, are only allowed to be placed on the ground floor, and the height of a window must not exceed the height of the floor it is located on. In the subsequent step, the architecture that is specified in the XML document is transferred into the final CityGML dataset [6, 20]. Figure 3.3 displays an exemplary synthetic semantic city model generated with the Random3Dcity application. The whole dataset comprises 25 buildings at the Delft LOD 3.3 level.

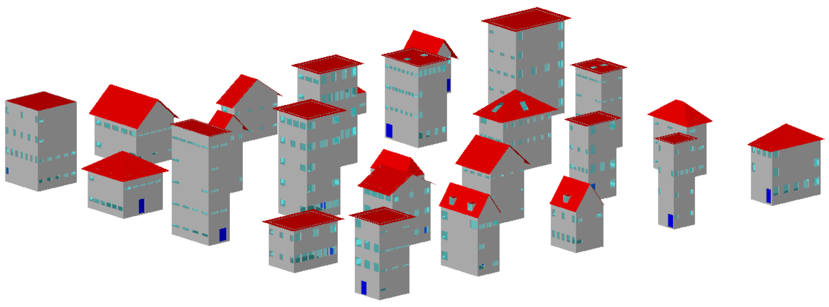


Figure 3.3: Screenshot of an exemplary synthetic semantic city model consisting of 25 buildings at the Delft LOD 3.3 level that has been generated with the Random3Dcity application

3.3 LaMa image inpainting

3.3.1 Inpainting model

Large mask inpainting (LaMA) is a state-of-the-art image inpainting method, introduced by Suvorov *et al.* in 2021 [4]. In their work, they tackle one of the key issues in semantic image inpainting, the limited receptive field of the model, by incorporating the Fast Fourier Convolution (FFC) operator [52], a high receptive field perceptual loss, and large training masks. The LaMa model is based on the GAN architecture. It is generally structured similar to a feed-forward ResNet inpainting network. In this work, I utilize a pre-trained LaMa model to compare and evaluate different inpainting strategies.

Fast Fourier Convolution (FFC) operator

One of the essential features of the LaMa model architecture is the FFC operator[4]. It has first been introduced by Chi *et al.* in 2020 [52]. It is responsible for providing a large receptive field that covers the entire image, thus allowing to incorporate global information in early layers of the network [4].

3.3.2 Generating random masks

Besides providing a powerful and easy-to-use (pre-trained) model for image inpainting, the work of Suvorov *et al.* is also known for introducing a sophisticated routine for generating random masks [4]. The properties of the masks used during the training of an inpainting model greatly influence the performance of the resulting inpainting model [4]. Their methodology has also been adapted in other studies, such as that of Rombach et al. [2]. Figure 3.4 displays a collection of six masks created with the mask generation routine by Suvorov *et al.*. The black areas represent the areas that an inpainting model should not change, while the white areas indicate missing regions within an image.

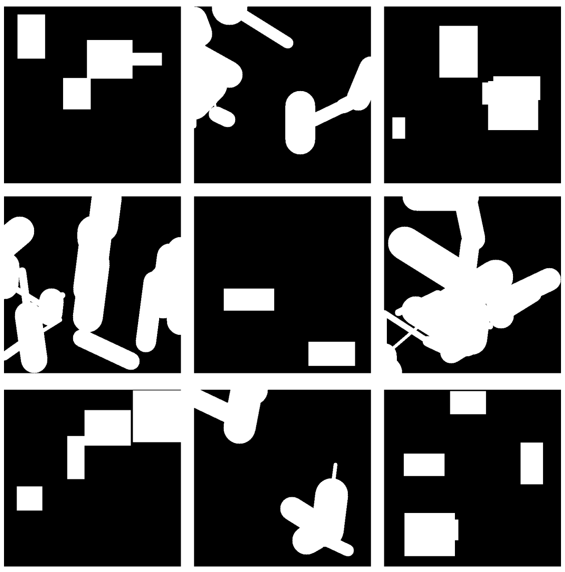


Figure 3.4: Collection of six masks that have been generated with the LaMa mask generation routine. Original size of each mask: 512 x 512 pixels.

3.4 Stable Diffusion

With the introduction of latent diffusion models, such as Stable Diffusion, Rombach *et al.* have made a significant step towards the democratization of large diffusion models [2]. The required computational resources are reduced by mapping the input features into a latent feature space with a pre-trained auto-encoder network. Their sophisticated, diffusion-based text-to-image model offers tremendous flexibility. Conditioning mechanisms, supported by cross-attention layers in the model architecture, offer the possibility of incorporating additional information. For example, adding five specialized input channels and a specialized training procedure makes it possible to apply the model to image inpainting purposes. Unlike most inpainting strategies, applying a Stable Diffusion model to image inpainting requires providing a text prompt. Various different pre-trained models and their various training checkpoints are publicly available.

3.5 Dreambooth

Dreambooth represents a method for personalizing large text-to-image diffusion models [5]. It offers the possibility to perform subject-driven fine-tuning so that the network can produce a variety of new examples of a learned instance of an object type. A unique identifier that consists of an arbitrary combination of characters is bound to the specific instance. This identifier is used in the text prompt to refer to the learned instance during inference. A specialized prior-preservation loss can be applied during training to create a larger variety of results by enhancing property modification, re-contextualization, or novel view synthesis. This loss supervises the model with its own generated images, thus allowing it to maintain the diversity of an object class learned during the training of the pre-trained model [5].

A considerable advantage of Dreambooth is the relatively low number of required training data. Only approx. four to five images are necessary to perform a successful subject-driven fine-tuning [5]. This enables the application in domains where the number of available training samples is comparatively low. A disadvantage of Dreambooth is the sensitivity towards hyper-parameter settings, such as the number of training steps and the learning rate. Unfortunate choices can often lead to overfitting [53].

Dreambooth can be applied to any diffusion-based image inpainting model [5]. It is also available for the variant of Stable Diffusion that has been modified for image inpainting purposes [54].

3.6 Traditional inpainting strategies

3.6.1 Navier-Stokes based inpainting

A method based on the principles of fluid dynamics, specifically the Navier-Stokes partial differential equations, represents an example of a traditional inpainting approach. In this method, an image is conceptualized as a dynamic fluid system. The core concept is to guide the flow of information into the missing regions, much like how fluids naturally fill empty spaces [48]. An implementation is available, for example, in OpenCV [55].

3.6.2 Telea inpainting

The approach of Telea is another example of a traditional inpainting strategy. It is based on the Fast Marching Method (FMM) [56]. Pixels around a missing area are analyzed and used to predict the pixels in the gap. Telea's method involves solving a PDE for each pixel in the inpainting region. These PDEs ensure that the newly generated pixel values are visually coherent and smooth concerning their surroundings [47]. An implementation is available, for example, in OpenCV [55].

3.7 RealFill

Tang *et al.* introduced the generative inpainting model RealFill in September 2023 [57]. The development of this model took place with different objectives than most other inpainting strategies. For example, the Stable Diffusion inpainting model has been developed to generate new content not present in the image before the inpainting procedure. Therefore, the content that the masked regions in the image are filled with can be vastly different from the content of the unmasked parts. The objective of RealFill is to complete the image in a semantically more meaningful manner. The content used to fill in the missing parts of the image should align with expectations rather than introducing something entirely different. This problem setup is referred to as authentic image completion. Given a small set of reference images, the target image is completed according to the content of these reference images [57].

The central concept of RealFill is fine-tuning a pre-trained Stable Diffusion model. In their study, Tang *et al.* combine Dreambooth with Low Rank Adaptions (LoRA) to increase memory efficiency by not fine-tuning all weights in the network [57, 58].

4 Method

4.1 Overview

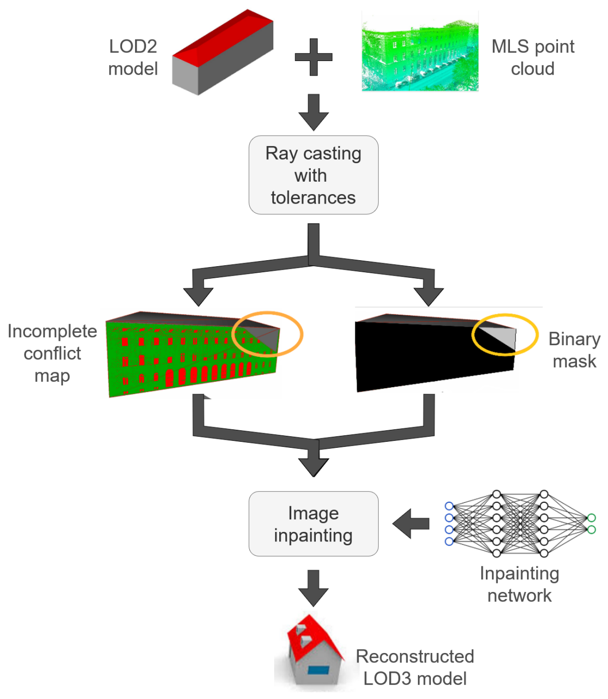


Figure 4.1: Schematic diagram illustrating the overall concept that is applied in this work

Figure 4.1 provides an overview of the general workflow applied in this work. The central concept is the semantic completion of an incomplete conflict map using a deep learning approach. I obtain conflict maps and corresponding binary masks that indicate missing areas from combining LOD2 models and MLS point cloud data. For this purpose, I apply a ray casting approach with tolerances to identify conflicts between the LOD2 model and the MLS point cloud. To facilitate its deployment for the completion of incomplete conflict maps, I personalize a pre-trained Stable Diffusion model[2, 3] with Dreambooth [5], utilizing datasets consisting of synthetic conflict maps and such that are derived from a database of annotated images. I derive synthetic conflict maps from semantic building models that I randomly generate with the Random3Dcity engine[6]. As an alternative, I explore training the LaMa GAN [4] using a similar dataset composed of 20,000 conflict maps obtained from the same data sources.

4.2 Determining conflict maps from semantic 3D building models and MLS point clouds

In contrast to the probabilistic approach used in the studies by Wysocki *et al.* and Yahya [22, 51], I obtain conflict maps by applying a deterministic approach. I chose this method with the intention of obtaining only high-probability conflicts that serve as a suitable basis for evaluating inpainting methods. Figure 4.2 gives an overview of the steps conducted in this work to obtain a conflict map. I apply the NumPy Python library in the implementation of my methodology [59].

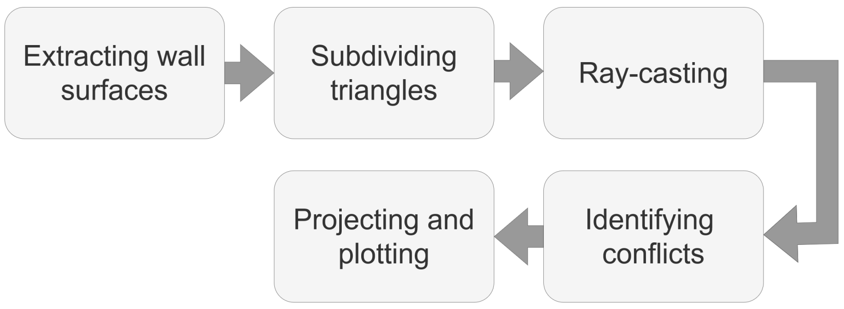


Figure 4.2: Schematic workflow for the deterministic generation of conflict maps from CityGML and MLS data

4.2.1 Extracting WallSurfaces

As a first step, I extract all wall surfaces from a given CityGML dataset. While this step is not generally necessary, it simplifies further processing steps. Since CityGML is very flexible, different possibilities of geometric representation have to be considered when extracting individual objects without loss of information. To ensure the extraction of all points, regardless of the individual representation, I apply functionalities implemented by Biljecki *et al.* within the framework of the CityGML2OBJs application [60]. It must be mentioned that the approach for extracting wall surfaces that I use in this work can only be applied to CityGML datasets with version 2.0. CityGML 3.0 still needs to be supported by my work.

As an alternative approach for extracting the Wall Surfaces and to test my approach for correctness, I also apply a filtering functionality offered in Feature Manipulation Engine (FME).

4.2.2 Subdividing triangles

I subdivide the triangles that the surface consists of into smaller ones, obtaining a more fine-grained representation of the surface. In the following steps, this is going to enable the accumulation of triangles during the ray casting. For the triangle subdivision, I make use of the respective Open3D functionality [61]. I iteratively subdivide triangles in five to eight iterations.

The geometric resolution of the conflict maps I obtain from the process depicted in Figure 4.2 largely depends on the size of the triangles in the meshes representing the façade surfaces. With decreasing size of the triangles, the geometric resolution increases, and vice versa. However, a smaller triangle size results in a larger number of individual triangles and thus, a more extensive required computational effort in the subsequent steps. Hence, I aim to reduce the size of the triangles as much as possible while keeping the computational effort within an acceptable range.

Figure 4.3 depicts an example of the triangle subdivision. The illustration shows that the final size of the triangles depends on the original size of the triangles. This implies that in the final conflict maps resulting from the process illustrated in Figure 4.2, different geometric resolutions can be present within the same façade simultaneously. Another observation that can be obtained from Figure 4.3 is that the triangles resulting from the subdivision process are not necessarily equilateral. Hence, the geometric resolution can also vary according to direction.

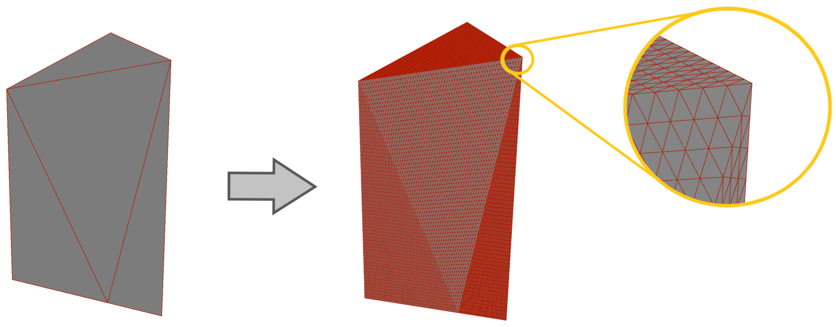


Figure 4.3: Exemplary façade before and after triangle subdivision with six iterations

4.2.3 Ray casting

I identify areas in the façades that have been scanned by the MLS system by performing a ray casting. In this work, I apply the ray casting functionality implemented in Open3D [61]. In defining the rays, I follow the approach that is similar to that of Wysocki *et al.* [8, 22]. I consider each viewpoint of the MLS scanner as an origin of a set of rays, while the directions to the points that are acquired from the respective viewpoint are considered as the direction of the individual rays in the set. This definition is reflected in Equation 4.1; with the ray $\mathbf{r}_p$ from the viewpoint $\mathbf{v}_p$ to the MLS point $\mathbf{p}$.

$$\mathbf{r}_p = \mathbf{v}_p + (\mathbf{p} - \mathbf{v}_p) \tag{4.1}$$

From the Open3D functionality, I obtain a list of triangles that are intersected by these rays.

It is of utmost importance that the ray casting is performed simultaneously for all façades that are considered. Subsequent façade-wise processing could result in the calculation of intersection points that do not exist. Such intersection points would corrupt the generation of meaningful conflict maps since the distance to the false intersection point would be larger than the distance to the measured point, thus indicating a conflict that does not exist in reality.

4.2.4 Identifying conflicts

The identification of conflicts between the LOD2 model and the MLS point cloud is related to the analysis of the geometric distance between the points in the MLS point cloud and the surface of the LOD2 model. Points that show a large geometric distance to a surface are more likely to conflict with the LOD2 model than points that show a smaller distance to a surface.

While Wysocki *et al.* and Yahya use a probabilistic approach identifying conflicts [8, 22, 51], I adopt a deterministic approach. I apply ray casting analysis with a set of tolerances to classify measurements as conflicted or confirming. My motivation to use this approach is to obtain only conflicts with a high probability.

As illustrated in Figure 4.4, I evaluate the intersection distance for each triangle hit by at least one of the rays. In the subsequent step, I compare this distance to the distance to the respective point in the MLS point cloud. If the difference between these two distances exceeds a tolerance, I classify the measurement as conflicted or unknown.

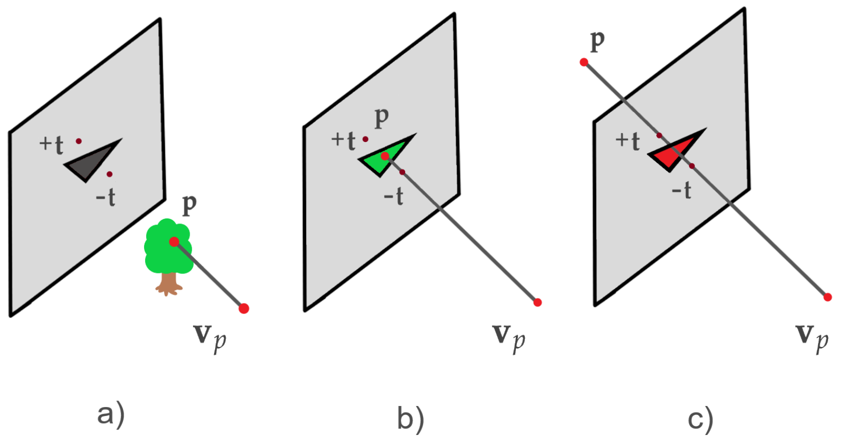


Figure 4.4: Schematic illustration of the conflict identification. a) unknown (dark-grey) b) confirming (green) c) conflicting (red). $\mathbf{v}_p$ corresponds to the viewpoint, $\mathbf{p}$ to a measured point that is part of the point cloud. $+t$ and $-t$ are the positive and negative tolerances.

4.2.5 Projecting and plotting

The façades are represented as 3D data, but the resulting conflict maps are required to be 2D raster data. This necessitates the utilization of a projection. The implementation for the projection that I use in this work is based on the work of Biljecki *et al.* [60]. I project every triangle within a façade into 2D, ensuring its frontal view. In the subsequent step, I plot the triangles using the Matplotlib Python library [62]. Figure 4.5 displays a conflict map that has been generated according to the method used in this work. Green indicates confirming areas, while black and red represent areas where a conflict is present. Areas where the evaluated intersection distance is shorter than the distance to the point in the MLS point cloud are represented in red. Those where the evaluated intersection distance is longer than the distance to the point in the MLS point cloud are represented in black. A tolerance of 30 cm has been used during the generation process. It has to be mentioned that the conflict map in Figure 4.5 is resampled due to the use of the Matplotlib Python library [62].

4.3 Synthetic conflict maps from random city models

I utilize the Random3Dcity application for the generation of random semantic city models [6, 20] that I use to produce artificial conflict maps. The generation process of the synthetic conflict maps is simplified since semantic information is already available

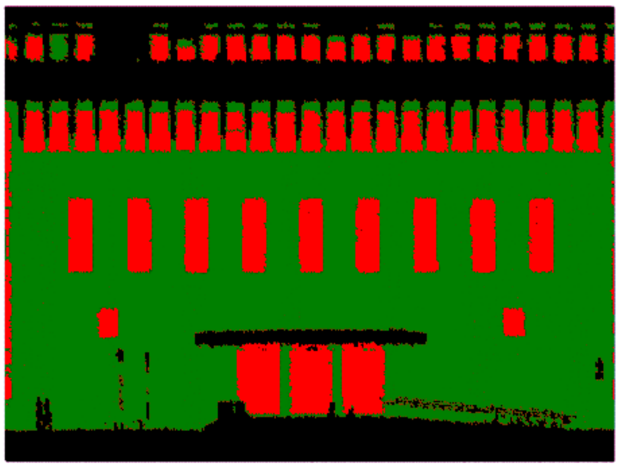
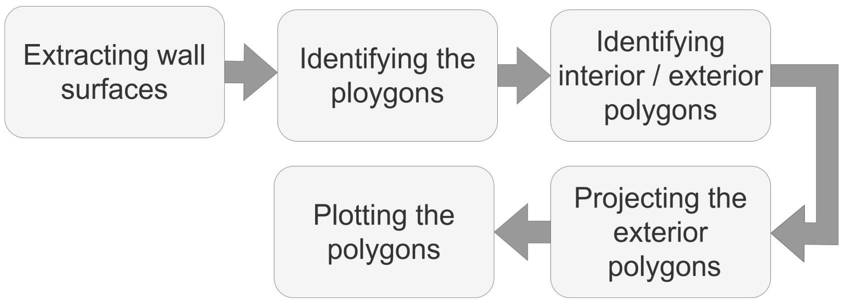


Figure 4.5: Exemplary conflict map of the eastern façade of the main entrance at the TUM City Campus, green: confirmed, red: conflicting (larger distance), black: conflicting (shorter distance). The triangles have been subdivided in eight iterations for this conflict map. The tolerance was set to 30 cm.

in the artificially generated CityGML datasets. Areas that would cause conflicts in real scenarios are identified as holes within the façade surfaces. For simplicity, I only consider windows and doors within this work. Extruded façade objects such as balconies are not taken into account. I directly project and plot the façade surfaces in the according colour. Figure 4.6 gives an overview of the applied workflow. I apply the same projection method that I use to create conflict maps from CityGML LOD2 models and MLS point clouds. A collection of four exemplary artificially generated conflict maps is provided in Figure 4.7. White areas are considered to be conflicted, while black areas are considered to be confirming areas.


Extracting wall surfaces
Identifying the ploygons
Identifying interior / exterior polygons
Projecting the exterior polygons
Plotting the polygons
Figure 4.6: Schematic workflow for the generation of artificial conflict maps from random CityGML datasets

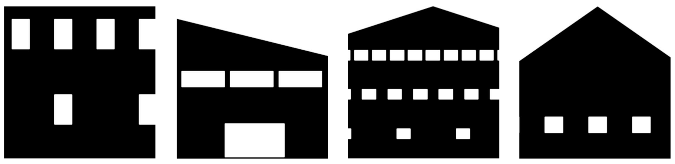


Figure 4.7: Four exemplary artificially generated conflict maps

4.4 Obtaining ground truth data from LOD3 models

I obtain ground-truth conflict maps from LOD3 models by applying the workflow illustrated in Figure 4.8. In contrast to the workflow I apply for generating synthetic conflict maps from randomly generated semantic building models, extruded façade objects are considered here.

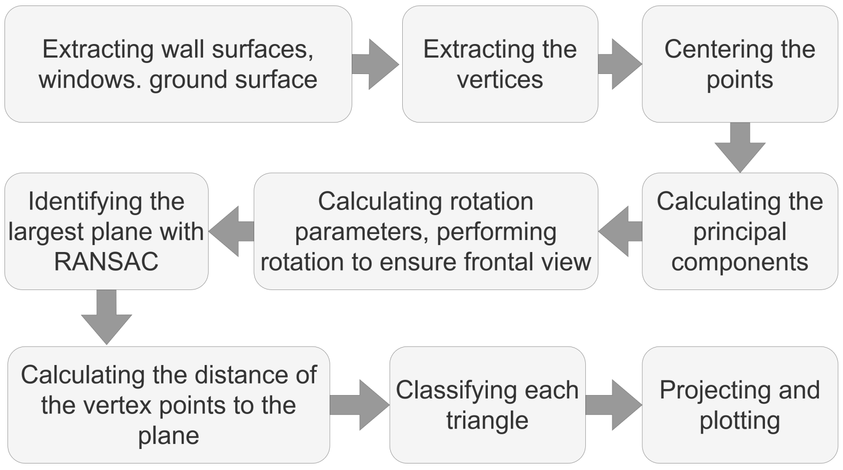

Extracting wall surfaces, windows. ground surface
Extracting the vertices
Centering the points
Calculating the principal components
Calculating rotation parameters, performing rotation to ensure frontal view
Identifying the largest plane with RANSAC
Calculating the distance of the vertex points to the plane
Classifying each triangle
Projecting and plotting

Figure 4.8: Schematic workflow for the generation of ground truth conflict maps from LOD3 semantic building models

4.4.1 Principal components and rotation parameters

I ensure that each façade is adequately aligned with the coordinate axes for its frontal view. This alignment involves a rotation around the z-axis since all façades are vertical. To determine the necessary rotation, I first evaluate the principal components of the vertex point distribution of each centered LOD3 façade. I obtain the principal components by performing a singular value decomposition of the vertex point matrix. For this purpose, I apply the corresponding scipy.linarg functionality [63]. After identifying the largest principal components, I calculate the required rotation angle to align the façade with the x-axis of the reference system. This process corresponds to rotating the façade in so that the covariance of the vertex points becomes maximal in the x-direction.

4.4.2 Plane identification with RANSAC

Assuming that most of the vertex points in the LOD3 façade lie in one plane, I apply the Random Sample Consensus (RANSAC) algorithm to identify the plane that contains the largest number of inliers. I set a threshold of 0.005m and 10 as the minimal number of inliers and perform a maximal number of 1000 iterations. I utilize the implementation in pyRANSAC-3D [64].

4.4.3 Determinig conflicts

I assume that the identified plane is equal to the façade, as it would be represented in a corresponding LOD3 semantic building model. Therefore, I identify all triangles that lie in this plane as confirming. All triangles that deviate from this plane are classified as conflicting. I manually remove all window glass from the building model. The remaining holes are automatically classified as conflicting. The pulses of a laser scanner usually penetrate window glass [25]. This circumstance justifies my approach.

4.5 Mask generation

4.5.1 Determining masks from intersection distance analysis

The conflict maps I produce in this work can show incompleteness for various reasons, such as occlusion caused by vegetation or insufficient coverage. Completing the missing regions in the conflict maps requires the identification of occluded or unknown regions. Such information can be provided in the form of binary masks. My approach involves analyzing the intersection distances derived from the ray-casting process, allowing for the simultaneous generation of such masks alongside the conflict maps. I define a threshold t that I use to distinguish conflicts originating from façade elements and those originating from occluding objects. In Figure 4.9, I illustrate the identification of occluding vegetation within a conflict map. I have chosen a threshold value of $t = 3\,\mathrm{m}$ for this demonstration.

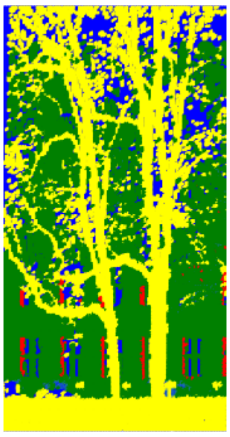


Figure 4.9: Exemplary identification of occluding vegetation in a conflict map with lower resolution. A threshold of 3 m has been used. The identified vegetation is colour coded in yellow. It could be used as a mask for inpainting.

4.5.2 Random masks

I create a set of approx. 200 random masks that I use to test models on exemplary datasets by applying the random mask generation routine introduced by Suvorov *et al.* [4]. I consistently use the same mask dataset to test different models to enhance the comparability of the results. Exemplary random masks are depicted in Figure 4.10.

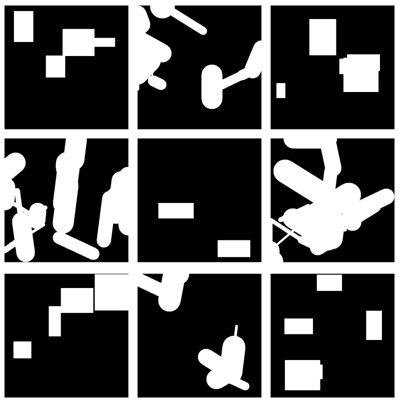


Figure 4.10: Collection of nine random masks that have been generated using the routine introduced by [4]

4.5.3 Masks from laser scanning point clouds

I use a point cloud representing a tree to create a mask for evaluating the completion of real conflict maps. I apply the workflow illustrated in the diagram in Figure 4.11. As a first step, I normalize the point cloud concerning translation and scale invariance. Subsequently, I project the point cloud to 2D and sample it on a grid representing the resolution of the output mask. Lastly, I convert the grid into a binary image. I apply the Pillow Python library [65].

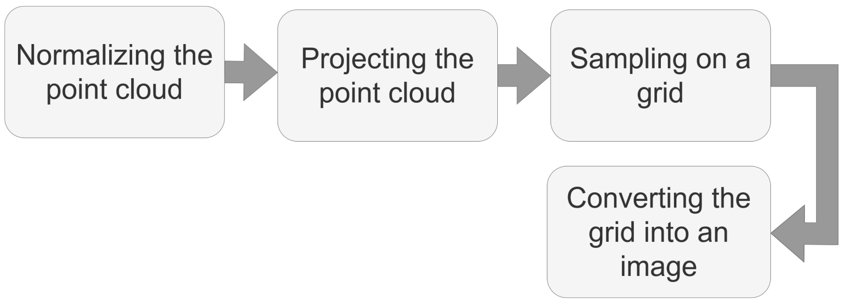

Normalizing the point cloud
Projecting the point cloud
Sampling on a grid
Converting the grid into an image

Figure 4.11: Schematic diagram of the workflow that I apply for the generation of a mask from a point cloud

4.6 Finetuning with Dreambooth

I utilize Dreambooth [5] to personalize a pre-trained Stable Diffusion inpainting model. The pre-trained inpainting model that I apply is based on Stable Diffusion v-1-2, which has been trained on the laion-improved-aesthetics dataset. It has undergone additional inpainting training on the laion-aesthetics v2 5+ dataset [2, 3]. I leverage synthetic conflict maps and conflict maps derived from the CMP dataset [66] as training data. I vary the number of training samples in my different sets of experiments. With a maximum of only a few hundred individual samples, this number remains comparatively low to a sophisticated training dataset that could be used for training a diffusion network.

4.6.1 Training datasets

I perform three fine tunings, each with a different fine-tuning dataset. The specifications of these three datasets are summarized in Table 4.1.

For simplicity, I consider the conflict maps as binary images in my experiments. Black pixels indicate conformance, while white pixels indicate the presence of a conflict.

Table 4.1: Specifications of the datasets applied for the personalization of the Stable Diffusion inpainting model with Dreambooth

Datset	Specification
1	a total number of five synthetically generated conflict maps
2	a total number of 192 synthetically generated conflict maps
3	a total number of 228 conflict map that are derived from annotated images

4.6.2 Hyperparameter settings

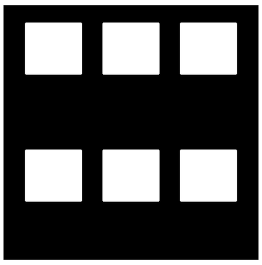
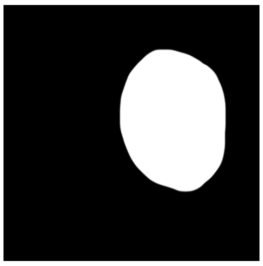
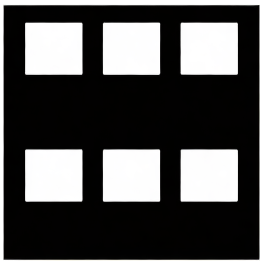
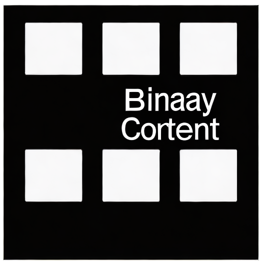
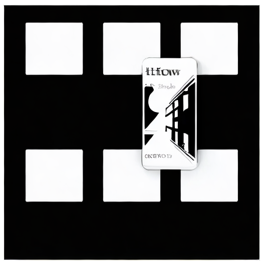
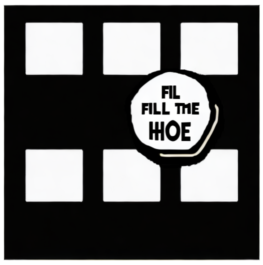
Since Dreambooth is sensitive to the settings of the hyperparameters, I make sure to use the same values in all experiments to ensure comparability. I choose these settings because experiments have proven them to be suitable [5]. The specific settings that I use are summarized in Table 4.2.

Table 4.2: Specifications of the training parameters applied for the personalization of the Stable Diffusion inpainting model with Dreambooth

Hyperparameter	Setting
Resolution	$512 \cdot 512$
Training batch size	1
Number of gradient accumulation steps	1
Learning rate	5×10^{-6}
Number of warmup steps	0
Maximal number of training steps	400
Unique identifier	"sks"

As a text prompt, I consistently apply the string *"a black background with white patches"*. I choose this text prompt by manually constructing text prompts and qualitatively analyzing the corresponding inpainting results. In these experiments, I noticed the strong dependence of the inpainting results on the used text prompt. Table 4.3 illustrates this dependence with three examples of synthetic data. I achieve the best results with a text prompt that describes the content that the missing region should be filled with as closely as possible.

Table 4.3: Examplary inpainting results demonstrating the dependence towards the text-prompt

Original image	Mask	*"black background with white surfaces"*
		
"binary content"	Empty text-prompt	*"fill the hole consistently"*
		

4.7 Training of the LaMa-GAN

I train the LaMa inpainting GAN [4] from its foundation as an alternative deep learning approach for image inpainting and comparison to the Stable Diffusion models. The number of annotated images is, with several hundred, too small to use them as the only source for training data. In addition, state-of-the-art LOD3 building models, which could be used to derive conflict maps, are scarce. I circumvent this problem by constructing a dataset of 20.000 synthetically generated conflict maps. I augment this dataset by adding a smaller number of conflict maps derived from annotated images of façades and split it into three subsets, the training, testing, and validation datasets. The structure of these partitions is summarised in Table 4.4.

Table 4.4: LaMa training, testing and validation dataset

Partition	total number of conflict maps	number of synthetic conflict maps	number of conflict maps derived from annotated images
Training	16,229	16,558	329
Evaluation	2,024	2,001	23
Validation	2,023	2,000	23
Testing	102	100	2

Compared to personalizing Stable Diffusion models, an advantage of using the LaMa GAN is the smaller amount of required computational resources. When accepting a loss in speed, the LaMa GAN can be deployed with a regular CPU instead of a graphics card GPU. This makes the LaMA GAN available for a broader range of potential users.

4.8 Substituting the manual text prompt specification

The Stable Diffusion inpainting model requires providing a text prompt. Ideally, such text prompts are specifically designed according to the properties of each image. Manually modeling the text prompts is infeasible for a more significant number of images. I introduce a method for automatically constructing of a text prompt according to low-level properties of the input images. In contrast to using a static text prompt, my method allows for a certain flexibility.

I construct the text prompt forwarded to the Stable Diffusion inpainting model from pre-defined substrings by combining these according to the following low-level features:

- Histogram

- Symmetry
- Fragmentation

4.8.1 Histogram analysis

I identify the background color of the image by analyzing the binary histogram of the input image. The predominant color, characterized by a higher number of occurrences, is considered the background color.

4.8.2 Symmetry evaluation

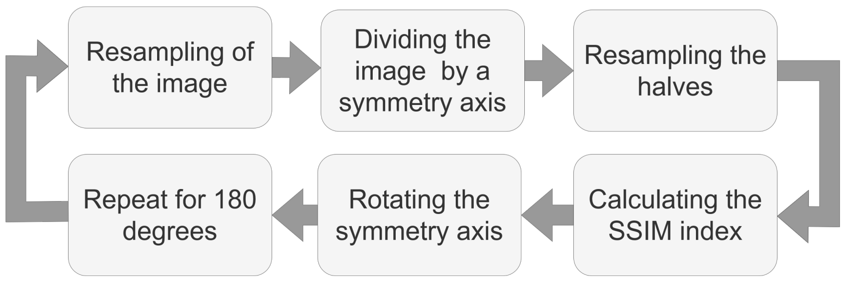

Resampling of the image
Dividing the image by a symmetry axis
Resampling the halves
Calculating the SSIM index
Rotating the symmetry axis
Repeat for 180 degrees

Figure 4.12: Schematic diagram of the workflow that I apply for the symmetry evaluation

Axis identification

I evaluate the axis symmetry in the images to modify the text prompt accordingly. The workflow that I apply for this purpose is displayed in Figure 4.12. In the first step, I define a symmetry axis that intersects with the center of the image. This axis divides the image into two halves of equal size. After resampling, I compare the two halves by evaluating the Structural Similarity Index (SSIM). I utilize the implementation of the SSIM available in the scikit-image Python library [67]. By iteratively rotating the axis and repeating the previous two steps, I approximate the best-fitting symmetry axis that intersects with the center of the image.

Evaluating symmetry

I establish a criterion for the determination of the presence of symmetry. This criterion, summarized in equation 4.2, is derived from the rotational angle-dependent function of the SSIM.

$$\text{Symmetry} = \begin{cases} \text{True}, & \text{if } (I_{\max} - \bar{I}) > \sigma_I \ \wedge \ I_{\max} > t \\ \text{True}, & \text{if } (I_{\max} - \bar{I}) > 3 \cdot \sigma_I \ \wedge \ I_{\max} < t \\ \text{False}, & \text{otherwise} \end{cases} \tag{4.2}$$

With the maximal SSIM score $I_{\max}$, the mean value of the SSIM scores $\bar{I}$, the standard deviation of the SSIM scores σ_I and the arbitrary threshold t.

Figure 4.13 illustrates my approach for an exemplary conflict map. The two prominent peaks in the diagram align with the superimposed conflict map. These peaks, both associated with the same symmetry axis, surpass the standard deviation that serves as the decision criterion in this case. Consequently, this conflict map would be classified as symmetric.

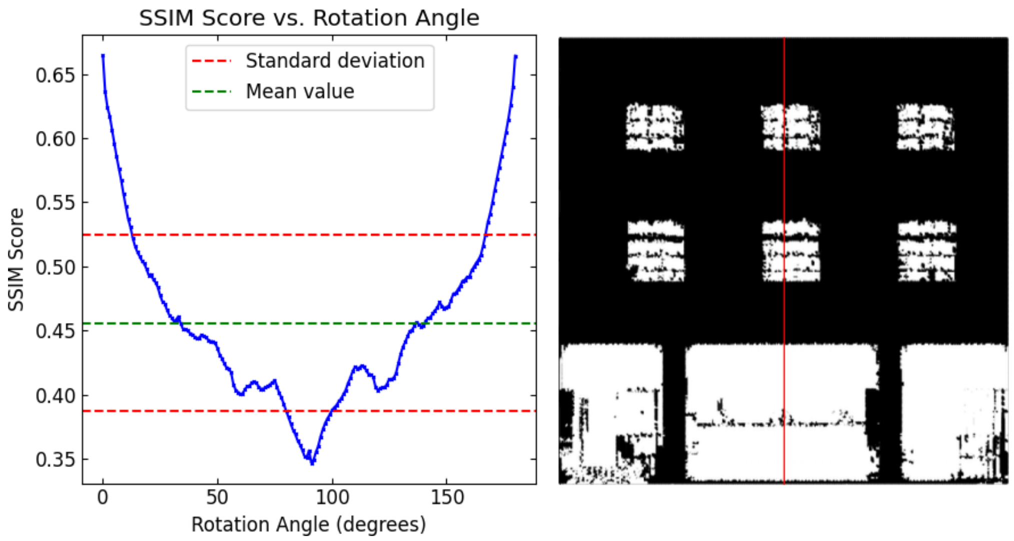


Figure 4.13: Evaluation of the symmetry properties. left side: Graph representing the evaluated SSIM scores against the rotation angle. right side: Corresponding conflict map with the most prominent identified symmetry axis superimposed

4.8.3 Analysing fragmentation

To analyze the fragmentation of a conflict map, I first identify the contours with the corresponding OpenCV function [55] that implements the contour detection algorithm of [68]. In the subsequent step, I calculate the average and cumulative contour area and determine the fragmentation score according to equation 4.3.

$$f = \frac{\bar{A}}{(\sum_{n=0}^{N} A_n)} \tag{4.3}$$

With the fragmentation score f, the mean contour area $\bar{A}$, the number of identified contours N, and the surface area of the n^{th} contour A_n.

4.8.4 Constructing the text prompt

I construct the text prompt from a set of pre-defined strings according to the results of the previously discussed analyses. These initial strings have to be selected with much care. To decide on which pre-defined strings to use, different thresholds have to be defined. I choose the pre-defined strings, and the thresholds according to the results of a trial and error approach on a small scale.

5 Experiments

5.1 Choosing test datasets

5.1.1 MLS point clouds

I utilize the proprietary "MoFa3D - Mobile Erfassung von Fassaden mittels 3D Punktwolken" (MF) point cloud, captured by 3D Mapping Solutions in 2023. It covers the area of the TUM City Campus and parts of the Pfisterstraße in Munich. An overview of the spatial extent of the parts of the MF point cloud used in this work is provided in the map in Figure 5.1. Wysocki *et al.* have used the MF point cloud for experiments on 3D façade reconstruction [10]. The point cloud is georeferenced in a local coordinate reference system. This ensures that the absolute values of the point coordinates are sufficiently small to not cause numerical instabilities during computation.

5.1.2 Semantic city models

LOD2 models

I obtained an official semantic LOD2 city model from the Bavarian Surveying Administration [69]. More specifically I acquired the $2\,\text{km} * 2\,\text{km}$ tile with the identifier 690_5336. This tile partially covers the area of the TUM City Campus. I manually filtered this dataset to collect all buildings within the TUM City Campus. Although unnecessary, this filtering step reduces the computational effort required for generating conflict maps. I apply the FME filtering transformer for this semantic filtering. The European Terrestrial Reference System 1989 (ETRS89) in the Universal Transverse Mercator (UTM) Zone 32 is used for position information of the dataset. The height information is provided in the German Main Height Network 2016 (DE DHHN2016 NH). Since the MLS point clouds are referenced in a local coordinate reference system, a transformation of the semantic city model into this local coordinate system is required. For this purpose I applied a simple translation in FME. Figure 5.2 gives an overview of the LOD2 building models representing the TUM Main Campus and some of its surrounding buildings. In addition I also acquired the tile with the identifier 692_5334 that covers the Pfisterstraße in Munich.

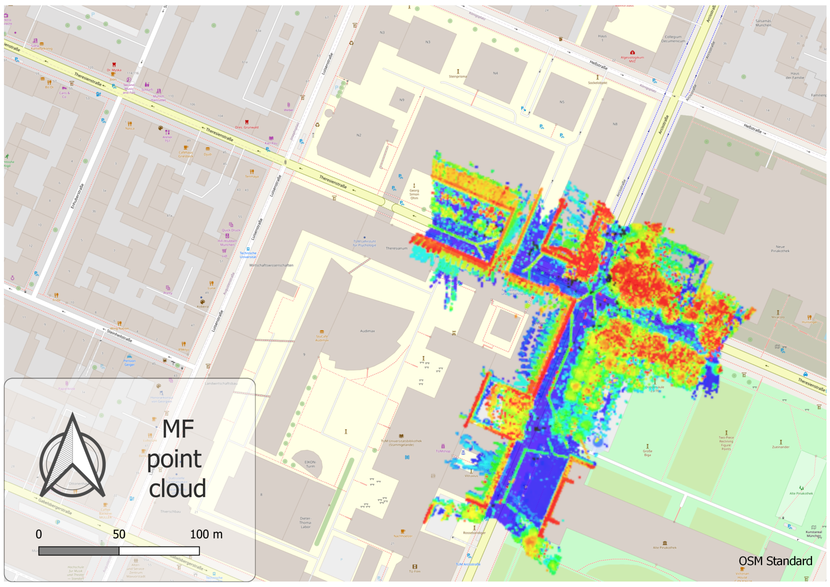


Figure 5.1: Map displaying the spatial extent of the parts of the MF point cloud that are used in this work

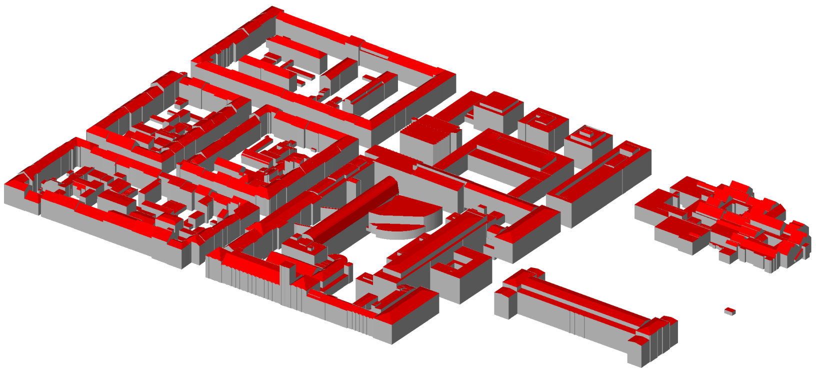


Figure 5.2: Semantic LOD2 building models representing the TUM Main Campus and its surrounding buildings

LOD3 models

I obtained semantic LOD3 building models representing the buildings of the TUM main campus from the tum2twin GitHub repository [70]. The data provided is in a beta version due to ongoing construction activities at the site and unfinished modeling works. I used these models as ground truth data to evaluate conflict maps and inpainting results. Figure 5.3 exemplarily displays the model representing building No. 57.

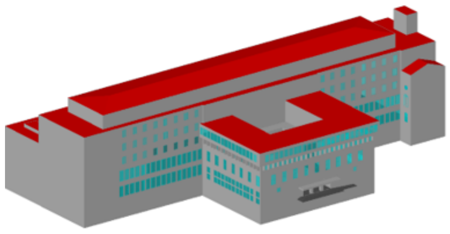


Figure 5.3: LOD3 model of building No. 57

5.1.3 Point clouds for mask generation

I used a point cloud acquired by stop-and-go scanning with a Riegl VZ-400i terrestrial laser scanner mounted on a passenger car. It is part of the TreeML-Data collection comprising more than 3,775 leaf-off point clouds [71]. The point cloud I use is depicted in Figure 5.4. Figure 5.5 shows the inverted mask I generate from this point cloud.

5.1.4 CMP datasets of annotated images

I made use of the CMP database of annotated images provided by the Center for Machine Perception in Prague [66]. The dataset is divided into two partitions, totaling 606 images. The CMP-base dataset consists of 378 images, while the CMP-extended dataset includes 228 additional images. The dataset provides three components for each façade: an RGB image, a corresponding annotation, and an XML document. A total of 12 classes is used for classifying different façade elements, with the annotations exclusively relating to façades. In particular, vegetation and occluding objects such as cars or pedestrians are not included in the set of classes. The background class represents the only exception [66]. Figure 5.6 exemplarily displays the RGB image and the corresponding annotation image for a façade from the CMP-base dataset.

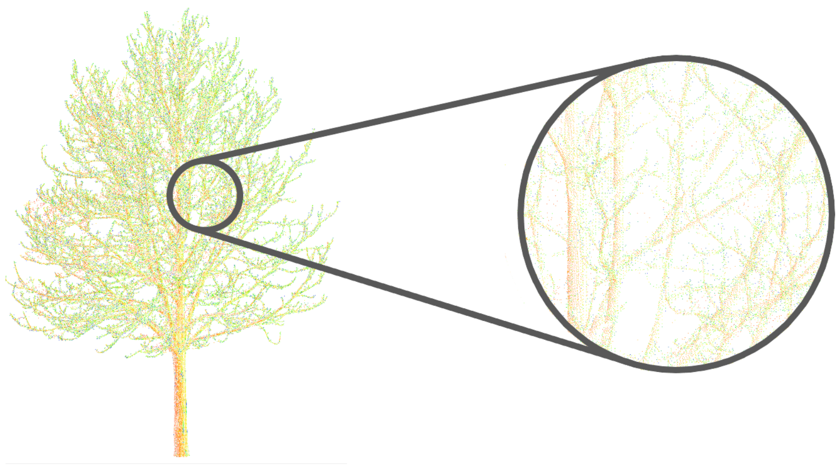


Figure 5.4: Point cloud of a tree acquired by stop-and-go scanning with a Riegl VZ-400i terrestrial laser scanner [71]

5.2 Evaluation metrics

To evaluate the quality of the conflict maps and the inpainting performance, I compared the results to ground truth information derived from either already existing LOD3 models or was readily available, as I artificially masked complete images during specific experiments. The inpainting result should resemble this ground truth information as closely as possible.

Various methods for quantitatively evaluating the similarity between two images are available [72]. I quantitatively evaluated the results with three different metrics.

5.2.1 Structural Similarity Index (SSIM)

The SSIM is a statistical measure developed to provide a tool to quantitatively evaluate the objective similarity of an image concerning a reference image [73]. For two images A and B of the same size, it is calculated according to Equation 5.1 [73, 74]:

$$\text{SSIM}(A,B) = \frac{(2\mu_A\mu_B + C_1)(2\sigma_{AB} + C_2)}{(\mu_A^2 + \mu_B^2 + C_1)(\sigma_A^2 + \sigma_B^2 + C_2)} \tag{5.1}$$

With μ_A and μ_B being the mean, and σ_A^2 and σ_B^2 being the variances of the intensity values of the images A and B. σ_{AB} is the covariance between the pixel values in both images. The constants C_1 and C_2 are used to increase the numerical stability in the case of a denominator that is close to zero. The SSIM score takes values between

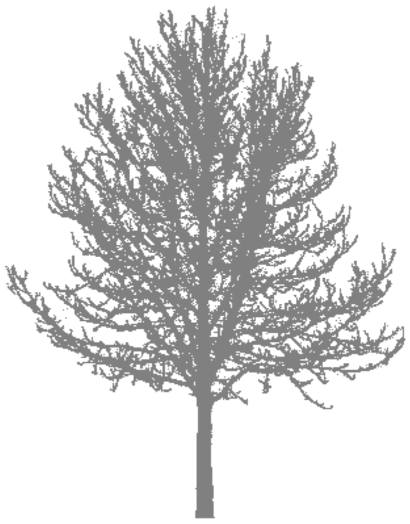


Figure 5.5: Mask generated from the point cloud of a tree acquired by stop-and-go scanning with a Riegl VZ-400i terrestrial laser scanner

zero and one, with a higher value corresponding to a larger similarity [73, 74]. In my experiments, I applied the implementation of the SSIM that is available in the scikit-image Python library [67]. I use the default setting of 11 pixels as a sliding window size.

5.2.2 Jaccard index

The Jaccard index, also often referred to as Intersection over Union (IoU), is a popular evaluation metric applied, for example, in object detection benchmarks. It is used to quantify the similarity of two sets. As demonstrated by Equation 5.2, the IoU is defined as the ratio of the intersection of two sets to the union of both sets [75]. This general definition makes the metric applicable in a broader range of scenarios. An advantage of the Jaccard index is that it can be easily extended to an arbitrary number of dimensions. In theory, this would make applications directly in the 3D domain possible.

$$\mathrm{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{5.2}$$

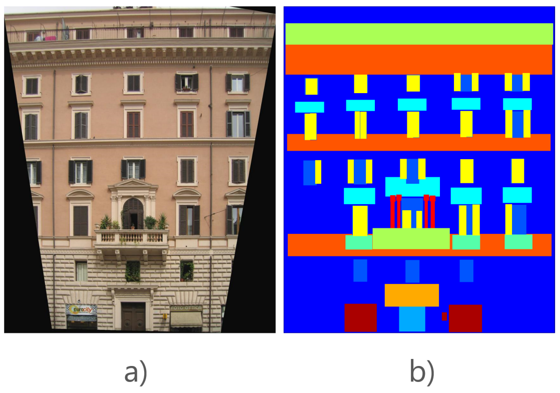


Figure 5.6: A façade from the CMP-base dataset. a) rectified RGB image b) corresponding annotation [66]

5.2.3 Deep-learning-based features

As a third performance evaluation method, I employed a deep learning approach. I utilized a pre-trained ResNet50 architecture to extract features from the conflict maps. The model I used has been trained on the ImageNet dataset [76] and is accessible through the TensorFlow Python library [77, 78]. I evaluated the similarity of the obtained feature vectors by employing the cosine similarity as it is defined in Equation 5.3 [67].

$$C(\mathbf{X}, \mathbf{Y}) = \frac{\mathbf{X} \cdot \mathbf{Y}^T}{\|\mathbf{X}\| \cdot \|\mathbf{Y}\|} \tag{5.3}$$

With the feature vectors **X** and **Y**. An implementation of this similarity measure is available in the scikit-image Python library [67].

5.3 Generation of conflict maps

5.3.1 Preprocessing

I generated conflict maps using the official LOD2 models and the MF point cloud. To conduct the steps displayed in Figure 4.2, the surfaces in the LOD2 semantic city model had to be converted into a mesh data type. Such a conversion has previously been addressed in the*CityGML2OBJs* project at the TU Delft and in the *CityGML2OBJs 2.0* project at the TUM Chair for Geoinformatics at the Technical University Munich [60, 79].

While CityGML2OBJs offers an open application for converting CityGML 1.0 and 2.0 datasets into the Wavefront Object Format (OBJ) data format, it comes with the drawback of an unsolved issue at the time. For yet unidentified reasons, geometric distortions appear for specific CityGML datasets. The conversion fails for some polygons, resulting in the lack of the respective polygon in the resulting OBJ file. Due to these unsolved issues, I chose to use FME to convert the CityGML data into the OBJ format.

5.3.2 Parameter settings

I determined conflict maps with different tolerance settings and the number of iterations in the triangle subdivision process. In addition, I conducted an experiment where I included an additional second tolerance, as summarized in Equation 5.4.

$$\text{colour} = \begin{cases} \text{red} & \text{if } d > t_1 \ \wedge \ d < t_2 \\ \text{black} & \text{if } d < -t_1 \ \wedge \ d > -t_2 \\ \text{green} & \text{if } |d| \leq t_1 \ \wedge \ |d| < t_2 \\ \text{yellow} & \text{if } d > t_2 \\ \text{white} & \text{if } d < -t_2 \end{cases} \tag{5.4}$$

5.4 Personalization with Dreambooth

I utilized the computational resources kindly provided by the TUM Chair of Computer Vision & Artificial Intelligence. Detailed information on these resources and the durations of the experiments are summarized in Table 5.1.

Table 5.1: Details on the computational resources and the runtime performance

Dataset	Technical details	Runtime information
1	NVIDIA RTX 8000 GPU 48 GB VRAM (only 32 GB used) CUDA Version 7.5.	Duration: 00:09:33
2	NVIDIA RTX 8000 GPU 48 GB VRAM (only 32 GB used) CUDA Version 7.5.	Duration: 00:08:25
3	NVIDIA A40 48 GB VRAM (only 32 GB used) CUDA Version 8.6.	Duration: 00:06:37

5.5 Substituting the manual text prompt specification

For substituting the manual text prompt specification, I make use of the specifications that are summarized in Equations 5.5, 5.6 and 5.7. The parameters n_{black} and n_{white} represent the percentage of black and white pixels in the image. The fragmentation score f is assessed against a threshold t_f to discriminate the patch size. I combine the three strings into the final automatically generated text prompt. I use a value of 0.98 for t_f. The symmetry of the image is evaluated according to the criterion established in Equation 4.2. I set the arbitrary threshold t to 0.5.

$$\text{string } 1 = \begin{cases} \textit{"black background"} & \text{if } n_{\text{black}} > 50\% \\ \textit{"white background"} & \text{if } n_{\text{white}} > 50\% \end{cases} \tag{5.5}$$

$$\text{string } 2 = \begin{cases} \textit{"with large black patches"} & \text{if } f > t_f \ \wedge \ n_{\text{white}} > 50\% \\ \textit{"with large white patches"} & \text{if } f > t_f \ \wedge \ n_{\text{black}} > 50\% \\ \textit{"with small black patches"} & \text{if } f < t_f \ \wedge \ n_{\text{white}} > 50\% \\ \textit{"with small white patches"} & \text{if } f < t_f \ \wedge \ n_{\text{black}} > 50\% \end{cases} \tag{5.6}$$

$$\text{string } 3 = \begin{cases} \textit{"that are consistent and symmetric} & \text{if symmetry} = \textit{True} \\ \textit{with the rest of the image"} & \\ \textit{""} & \text{if symmetry} = \textit{False} \end{cases} \tag{5.7}$$

5.6 Training the LaMa GAN

I trained the LaMa GAN for 25 epochs. During each epoch, the model was evaluated on the evaluation partition by computing the mean value of the SSIM and the Frechet Inception Distance (FID). Figure 5.7 provides a graphical representation of the results of these evaluations. I find the best performance in Epoch 17 with a total mean value of SSIM and FID of 0.95.

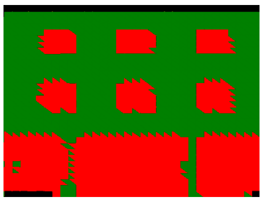
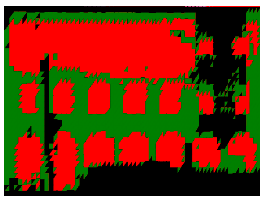
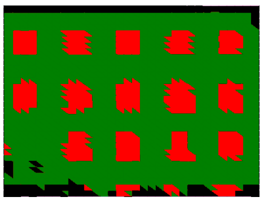
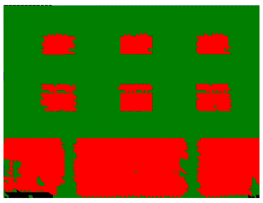
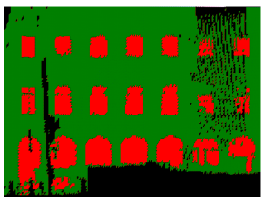
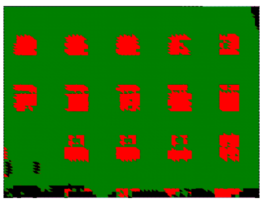
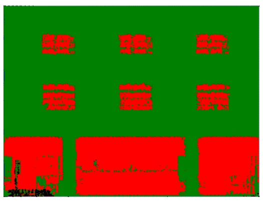
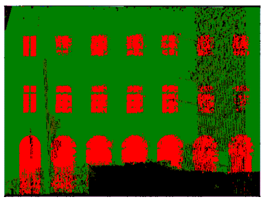
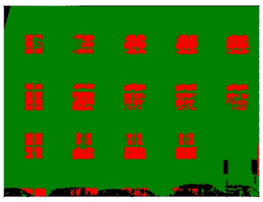
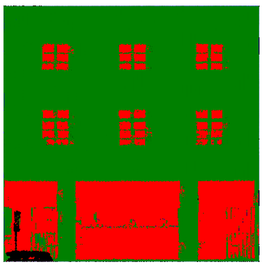
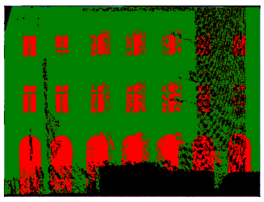
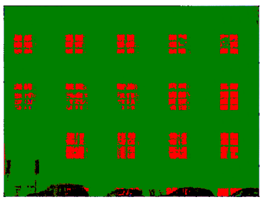
5.7 Evaluation

5.7.1 Qualitative evaluation

Evaluation of the generated conflict maps

Table 5.2 contains a selection of examples I obtained for conflict maps. The parameter t refers to the used tolerance in meters, while n represents the number of iterations in the triangle subdivision process.

Table 5.2: Exemplary conflict maps created from the official LOD2 semantic building models and the MF point cloud

Parameters	Building 58, northern façade	Building 22, eastern façade	Building 60, eastern façade
$t = 0.3$, $n = 5$			
$t = 0.7$, $n = 6$			
$t = 0.7$, $n = 8$			
$t = 0.8$, $n = 8$			

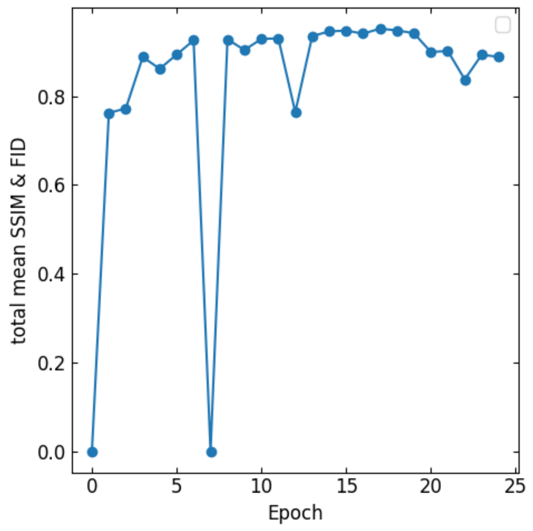


Figure 5.7: Total mean value of SSIM and FID for each training epoch

I draw various insights from these conflict maps. The geometric resolution and detail level are directly linked to the number of iterations in the triangle subdivision process. As I increase this iteration count, I consistently witness an improvement in the geometric resolution across all conflict maps. The threshold also influences the details that can be observed. I observe that small thresholds of approx. 30cm can lead to problems. In such cases, the difference between the wall in the LOD2 model and the actual building can be larger than the threshold, resulting in the loss of some façade details. For instance, such a case can be observed in the eastern façade of building 22, where parts of the second floor are lost.

I obtained more information by including additional tolerance thresholds in the distance analysis of intersecting rays, as described in Equation 5.4. In Figure 5.8, more details are visible, particularly in the arched windows on the ground floor. I used $t_1 = 0.7$, $t_2 = 3.0$, and eight triangle subdivision iterations for this conflict map.

Inpainting of structure similar to vegetation

I inferred the models I obtained by personalizing the Stable Diffusion model with Dreambooth on different datasets. I observe several noteworthy findings by visually analyzing the inpainting results. In addition, I also evaluated the performance of the LaMa GAN and two traditional inpainting approaches.

I notice that the inpainting results display a significant dependence on the properties

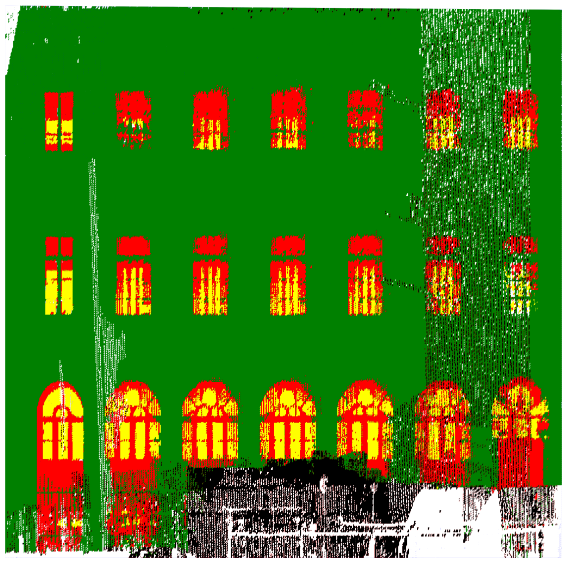


Figure 5.8: Conflict map with additional tolerance thresholds to obtain more detailed façade information. Color-coded according to Equation 5.4

of the mask. Specifically, investigation reveals that the Stable Diffusion networks tend to inpaint trees when the masks resemble the shape of a tree. Table 5.3 and 5.4 illustrate this behavior. When considering tree-shaped masks, the inpainting of structures similar to trees can be observed most prominently for the pre-trained Stable Diffusion model. I find that the personalization of this model with Dreambooth mitigates this phenomenon. In Table 5.3, I observe less prominent inpainting of structures similar to trees for the first and the second of these models.

In general, this represents an issue, as the main objective is to eliminate occluding elements, such as vegetation, from the conflict maps. Nevertheless, as depicted in Figure 5.9, it becomes qualitatively evident that applying inpainting, even in cases where inpainted content resembles vegetation, can result in noticeable enhancements.

In contrast to the Stable Diffusion models, the pre-trained LaMa GAN displays no such behavior. Notably, as demonstrated in Table 5.4, the phenomenon can not be observed in the case of randomly generated masks. Therefore, I attribute the inpainting of structures similar to vegetation to the shape of the mask rather than to the model itself.

Table 5.3: Exemplary inpainting results on real conflict maps with tree-shaped masks demonstrating the influence of the mask properties

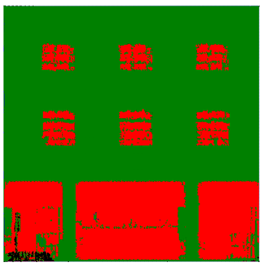
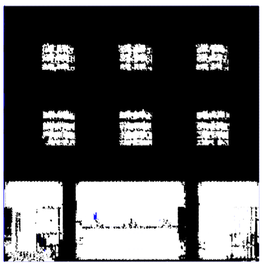
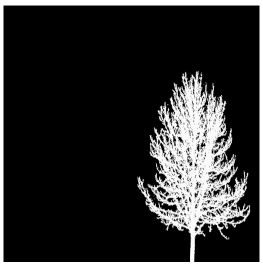
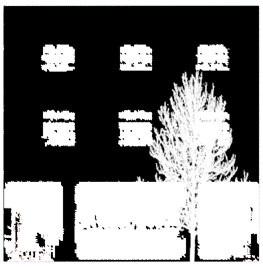
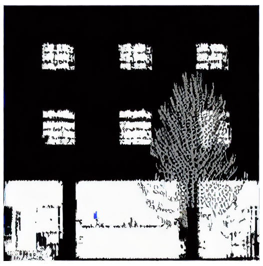
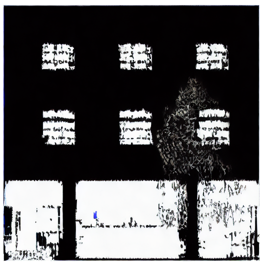
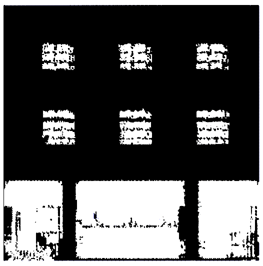
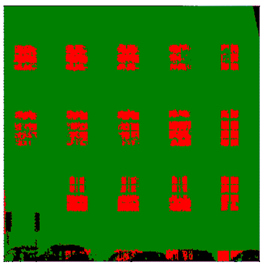
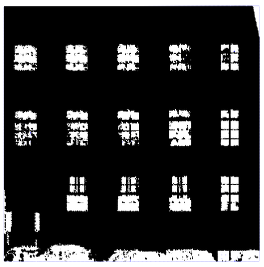
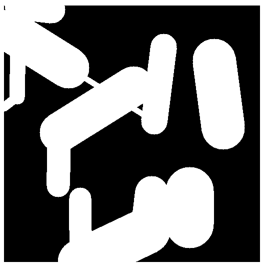
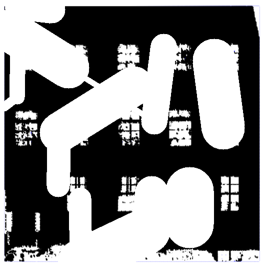
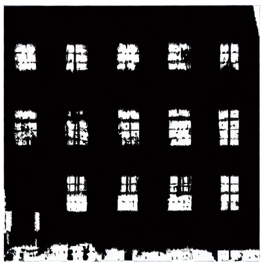
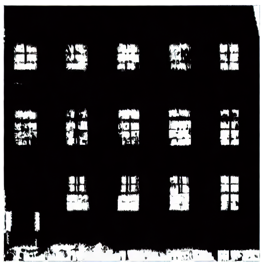
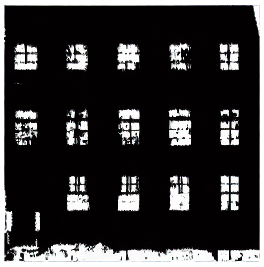
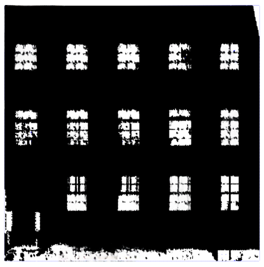
Original conflict map	Binary conflict map	Mask	Masked binary conflict map
			
Stable-Diffusion (pre-trained)	**Stable Diffusion (personalization 1)**	**Stable Diffusion (personalization 2)**	**LaMa GAN**
			

Table 5.4: Exemplary inpainting results on real conflict maps with randomly generated masks demonstrating the influence of the mask properties

Original conflict map	Binary conflict map	Mask	Masked binary conflict map
			
Stable Diffusion (pre-trained)	**Stable Diffusion (personalization 1)**	**Stable Diffusion (personalization 2)**	**LaMa GAN**
			

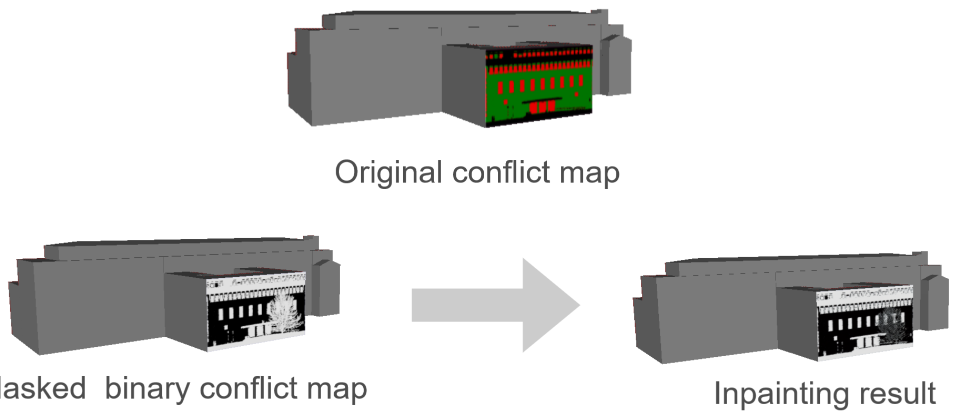


Figure 5.9: Exemplary demonstration of the inpainting approach with the third personalised Stable Diffusion model. The original conflict map, as well as the masked binary conflict map and the inpainting result map are incorporated into the LOD 2 model as textures.

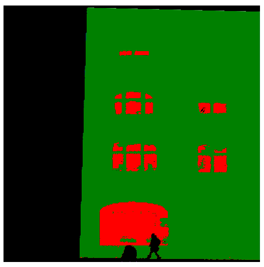
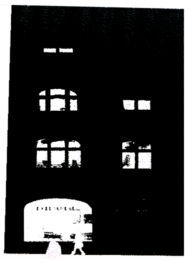
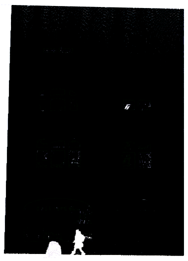
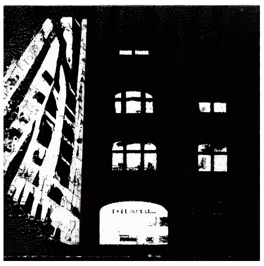
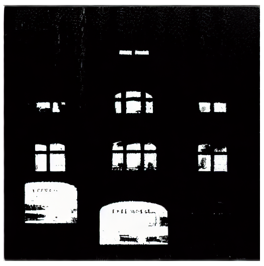
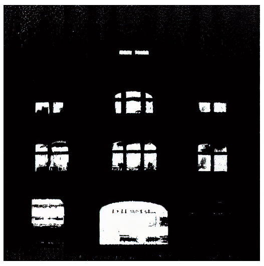
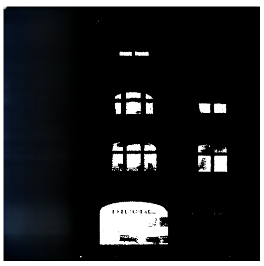
Inpainting large areas

I find that the inpainting of large masked areas is challenging. In particular, the LaMa GAN meets its boundaries here, as indicated in Table 5.5. In this example, approximately the left third of the façade is missing and has to be completed. An axis symmetry concerning a central, vertical axis can be assumed for this particular conflict map.

I observe that the LaMa GAN at epoch 17 predominantly fills in a black background, lacking any detailed façade structures. This trend is consistent with the behavior observed in the pre-trained LaMA GAN, which is not explicitly presented here. Notably, there seems to be an absence of symmetry consideration in this example. In contrast, the diffusion models exhibit an apparent recognition of symmetric properties within images, demonstrating a capacity for generating diverse content with increased variability. However, the results from the pre-trained model reveal that the generated content does not necessarily align with the overall image context.

When evaluating relatively small masks, such as those applied to pedestrians in the examples presented in Table 5.5, consistent and high-quality inpainting results are observed across all models.

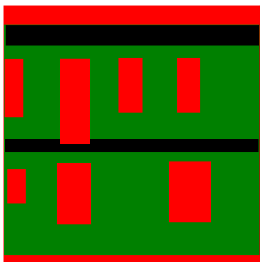
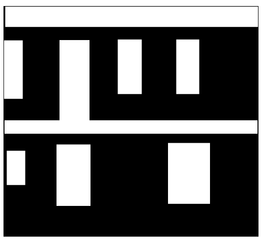
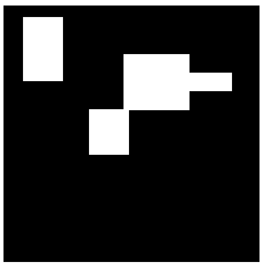
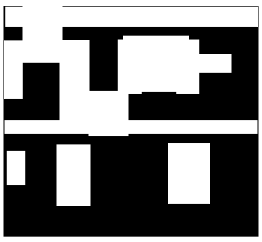
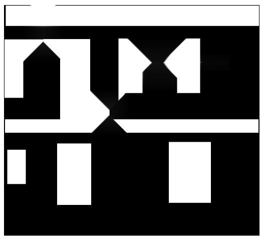
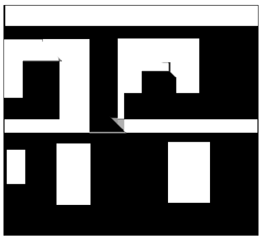
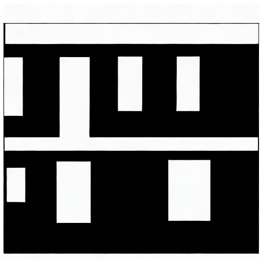
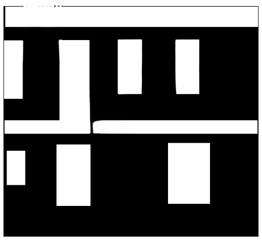
Table 5.5: Exemplary inpainting results on real conflict maps demonstrating the variability of the inpainting results when considering large masks

Original conflict map	Binary conflict map	Mask	
			
Stable Diffusion (pre-trained)	**Stable Diffusion (personalization 1)**	**Stable Diffusion (personalization 2)**	**LaMa GAN (epoch 17)**
			

Comparison to traditional inpainting approaches

I find that deep learning approaches perform better than the traditional inpainting strategies. The deficiencies of these traditional methods are evident in the exemplary results presented in Table 5.6. I tested Navier-Stokes-based inpainting and the method by Telea on a conflict map derived from an annotated image from the CMP-extended database. I used the same randomly generated mask displayed in Table 5.6 for all of these experiments. I compared the results to those obtained with the LaMa GAN at epoch 17 and the first personalized Stable Diffusion model. The results obtained with deep learning-based approaches exhibit a higher visual resemblance to the original conflict map compared to traditional inpainting methods. I notice significant semantic inconsistencies in the results obtained by traditional inpainting methods. I attribute this primarily to their limited capacity to consider global image content effectively. In contrast, deep learning strategies excel in capturing and incorporating global context, leading to more consistent inpainting results.

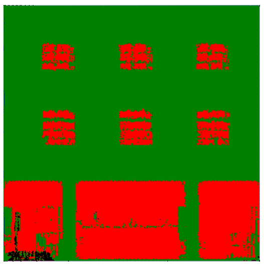
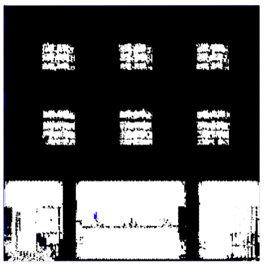
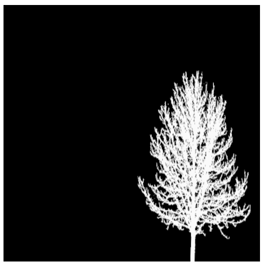
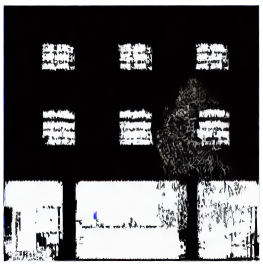
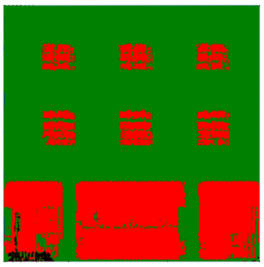
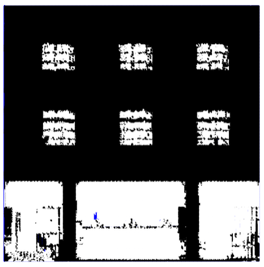
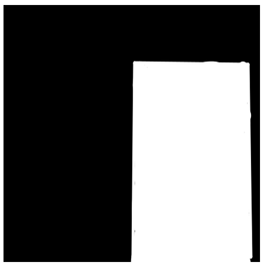
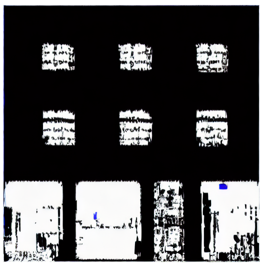
Table 5.6: Exemplary inpainting results on conflict maps derived from the CMP-extended dataset demonstrating the weaknesses of traditional inpainting strategies

Original conflict map	Binary conflict map	Mask	Masked binary conflict map
			
Telea Inpainting	**Navier-Stokes based Inpainting**	**Stable Diffusion (personalization 1)**	**LaMa GAN (epoch 17)**
			

Mitigation of vegetation-like inpainting by replacing masks

By substituting the intricate tree-shaped mask with a simple rectangle according to the bounding box, I attempted to mitigate the tendency of the Stable Diffusion models to inpaint structures similar to trees. The central concept is that I discourage the Stable Diffusion networks from inpainting such structures by eliminating the tree-shaped mask. This way I aimed to improve the results even when ground-truth information was lost. Table 5.7 demonstrates that substituting the mask can prevent the inpainting of vegetation-like structures. However, the loss of ground-truth information results in the inpainting of structures lacking semantic consistency with the original image.

Table 5.7: Exemplary inpainting results on real conflict maps demonstrating the effect of replacing the mask

Original conflict map	Binary conlict map	Mask	Stable Diffusion (personalization 1)
			
Original conflict map	Binary conflict map	Mask (tree structure replaced)	Stable Diffusion (personalization 1)
			

Comparison to ground-truth LOD3 building models

I compared the inpainting results on real conflict maps with the conflict maps derived from corresponding LOD3 models. Table 5.8 displays a collection of four selected examples. Despite less than flawless outcomes, the networks demonstrate a general capability to inpaint semantically meaningful content into missing regions. I observe a consideration of symmetry properties, and a continuation of existing image structures. These findings, along with the results presented in Tables 5.3 and 5.5, demonstrate the potential of deep-learning networks for completing incomplete conflict maps.

I notice that the content inpainted into missing regions by the Stable Diffusion models often resembles the remaining content of the image. This phenomenon can be observed in the inpainting results on real conflict maps in Table 5.3, 5.5, and 5.8. It includes image structures without clear semantic meaning or those representing noise.

5.7.2 Quantitative evaluation

Evaluation of the conflict maps

I assessed the quality of conflict maps by comparing them to ground truth information derived from corresponding LOD3 models. As shown in Table 5.8 and 5.9, there is a noticeable visual correspondence between the conflict maps and the ground truth information derived from the corresponding LOD3 models. However, the resulting similarity values reveal discrepancies between the LOD3 model and the conflict maps. For example the SSIM score 0.51 for the conflict map with the underpass. There are two possible explanations for this. The conflict maps either contain incorrect information, or the LOD3 building model does not correspond to the actual building façade close enough. Determining the cause of these discrepancies would involve a manual assessment of the LOD3 building model.

Evaluation of the inpainting approaches

For the quantitative evaluation of the inpainting models, I inferred all the models on the conflict maps derived from the CMP-extended dataset, which consists of 228 annotated images [66]. For each of the models, I conducted two experiments. For the first experiment, I utilized a set of randomly generated masks that I obtained with the random mask generation routine by Suvorov *et al.* [4]. For the second one, I applied a tree mask that I generated from a point cloud representing a tree. I evaluated the similarity of the images in the original dataset to the inpainting results.

Effect of the personalisation with Dreambooth

I make several observations based on this data. It is evident that, when using randomly generated masks, the personalization of pre-trained models with Dreambooth yields marginal to negligible enhancements in performance. In contrast, a slight improvement

Table 5.8: Exemplary inpainting results on real conflict maps demonstrating the potential of deep-learning models for the completion of 2D conflict maps

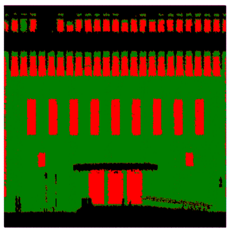
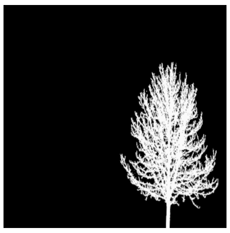
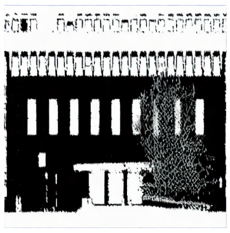
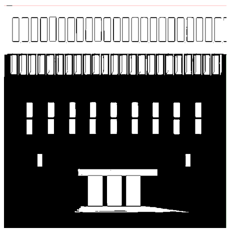
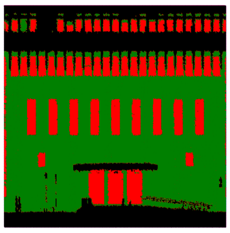
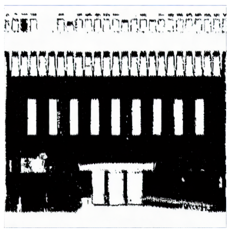
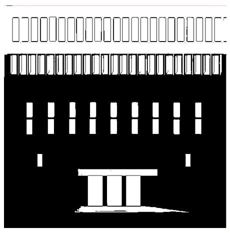
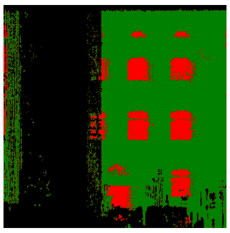
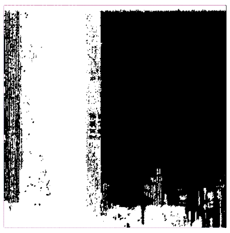
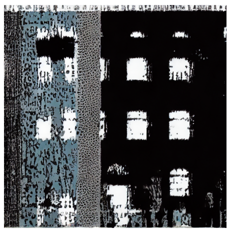
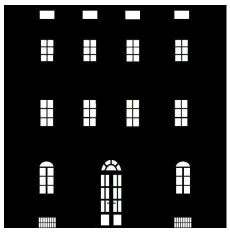
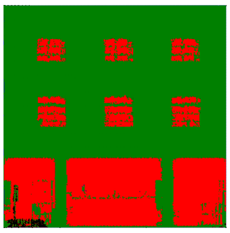
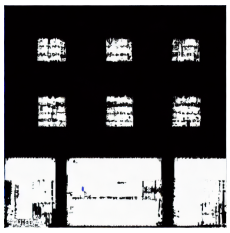
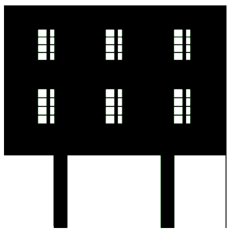
Original conflict map	Mask derived form tree point cloud	Stable Diffusion (personalization 1)	Ground-truth from LOD3
			
Original conflict map	Randomly generated mask	Stable Diffusion (personalization) 3	Ground-truth from LOD3
			
Original conflict map	Mask derived from distance analysis	Stable Diffusion (personalization 2)	Ground-truth from LOD3
			
Original conflict map	Randomly generated mask	Stable Diffusion (personalization 1)	Ground-truth from LOD3
			

Table 5.9: Evaluation of real conflict maps with ground truth information from LOD3 building models

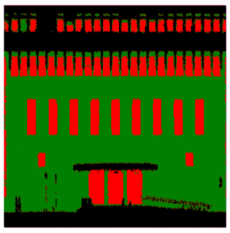
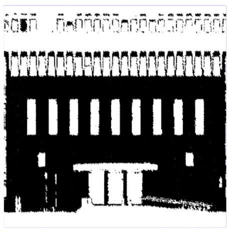
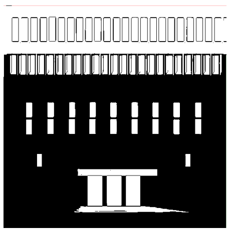
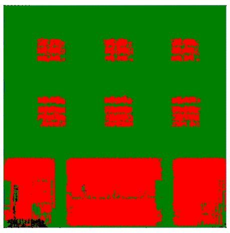
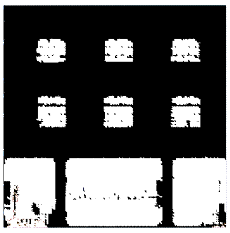
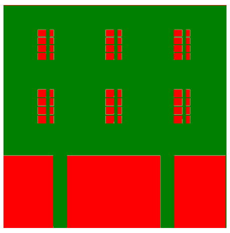
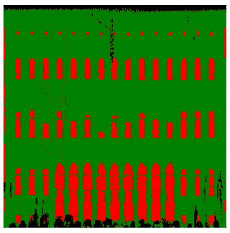
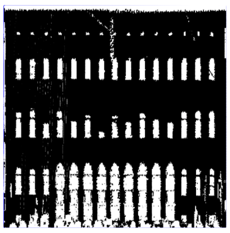
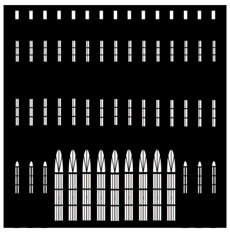
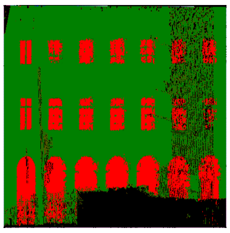
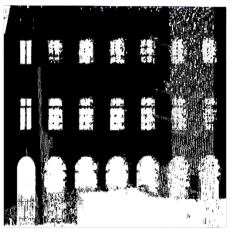
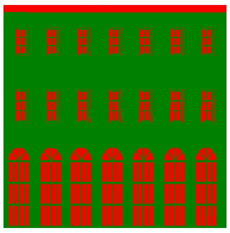
Original conflict map	Binary conflict map	Ground truth from LOD3	Similarity values
			SSIM: 0.51 IoU: 0.58 DL: 0.85
			SSIM: 0.78 IoU: 0.25 DL: 0.89
			SSIM: 0.46 IoU: 0.27 DL: 0.86
			SSIM: 0.35 IoU: 0.34 DL: 0.74

Table 5.10: Quantitative evaluation of the inpainting results with the SSIM

Model / Method	mean SSIM (random masks)	mean SSIM (tree mask)
S.D. pre-trained	0.90	0.75
S.D. Pers. 1	0.91	0.78
S.D. Pers. 2	0.90	0.81
S.D. Pers. 3	0.88	0.74
S.D. pre-trained (text-prompt subst.)	0.90	0.75
S.D. Pers. 1 (text-prompt subst.)	0.90	0.84
S.D. Pers. 2 (text-prompt subst.)	0.90	0.80
S.D. Pers. 3 (text-prompt subst.)	0.88	0.73
LaMa pre-trained	0.95	0.98
Lama epoch 17	0.96	0.96
Navier-Stokes	0.90	0.95
Telea	0.87	0.94

Table 5.11: Quantitative evaluation of the inpainting results with the Jaccard index

Model / Method	mean IoU (random masks)	mean IoU (tree mask)
S.D. pre-trained	0.72	0.66
S.D. Pers. 1	0.73	0.66
S.D. Pers. 2	0.72	0.66
S.D. Pers. 3	0.70	0.65
S.D. pre-trained (text-prompt subst.)	0.72	0.66
S.D. Pers. 1 (text-prompt subst.)	0.71	0.66
S.D. Pers. 2 (text-prompt subst.)	0.72	0.66
S.D. Pers. 3 (text-prompt subst.)	0.70	0.64
LaMa pre-trained	0.95	0.98
LaMa epoch 17	0.90	0.94
Navier-Stokes	0.92	0.97
Telea	0.89	0.97

Table 5.12: Quantitative evaluation of the inpainting results with deep-learning-based features

Model / Method	mean DL (random masks)	mean DL (tree mask)
S.D. pre-trained	0.97	0.89
S.D. Pers. 1	0.97	0.90
S.D. Pers. 2	0.97	0.92
S.D. Pers. 3	0.96	0.89
S.D. pre-trained (text-prompt subst.)	0.97	0.89
S.D. Pers. 1 (text-prompt subst.)	0.97	0.94
S.D. Pers. 2 (text-prompt subst.)	0.97	0.91
S.D. Pers. 3 (text-prompt subst.)	0.96	0.89
LaMa pre-trained	0.98	0.99
LaMa epoch 17	0.93	0.97
Navier-Stokes	0.95	0.96
Telea	0.89	0.95

in performance compared to the pre-trained model can be observed in the case of tree-shaped masks. This is most noticeable when evaluating the SSIM score. Here, an improvement from 0.75 (pre-trained model) to 0.81 (second personalized model) can be observed. This corresponds to findings from the qualitative analysis, indicating that personalizing the Stable Diffusion model with Dreambooth can address the issue of inpainting structures resembling trees.

Effects of the tree-shaped masks on diffusion models

Notably, the performance of all examined diffusion models decreases when subjected to tree-shaped masks instead of randomly generated masks. As an example, the Jaccard index of the pre-trained Stable Diffusion model drops from 0.72 to 0.66 in this scenario. I attribute this behavior to the inpainting of structures similar to trees when the mask resembles the shape of a tree. This phenomenon is evident in qualitative analysis, as illustrated in Table 5.3. When focusing solely on the personalized Stable Diffusion models with tree-shaped masks, it becomes evident that the second personalization proves to be the most successful, while the third one shows no substantial improvements. For instance, when considering the Jaccard index, the second personalized model achieves a score of 0.81, while the first model scores 0.78 and the third model only attains 0.74. This trend remains consistent across all the evaluation metrics employed.

Performance of the LaMa GAN

The LaMa GAN and the traditional inpainting approaches display different behavior. When exposed to a tree-shaped mask, an increasing performance can be observed. In the case of the LaMa GAN this is demonstrated by an increase in the SSIM score from 0.95 to 0.98.

I attribute this behavior to the size and distribution of the masked area. On average, the randomly generated masks I use cover a larger area than the specific tree-shaped mask utilized in my experiments. Unlike a simple, connected mask without holes, a tree-shaped mask is a more intricate structure, allowing for the preservation of some background information within the superstructure of the general tree shape. The observed increase in performance also corresponds to the observations I made in the qualitative analysis. While the diffusion models tended to inpaint structures similar to trees, I observe no such behavior for the LaMa GAN.

I observe that, in general, the LaMa GAN achieves better performances than the Stable Diffusion models. In general, the performance of the LaMa GAN at training epoch 17 is slightly less good than that of the pre-trained model. When considering random masks as an example, the Jaccard index of the pre-trained model is 0.95 while it is 9.90 for the model trained on my dataset.

Synthetic training data

Tables 5.10, 5.11, and 5.12 suggest that personalization with annotated data yields better results than with synthetic data. Across all evaluation metrics employed, the third personalized model displays lower performance than the first and second.

The comparison of the performance of the pre-trained LaMa GAN and the trained model at epoch 17 indicates a different behavior. The pre-trained model performs better than the model trained on mostly synthetic data.

However, it is crucial to interpret these results with caution, considering the influence of various factors, such as the size of the training dataset or the settings of hyperparameters. Further experiments are necessary to draw a definitive conclusion.

Effect of substituting the text prompt

Across all evaluation metrics deployed, I observe no improvement in performance by substituting the text prompt as it has been done in this work. The comprehensive performance remains consistent with the models without text-prompt substitution across all types of masks. I observe the same performance patterns in performance as before. There are several possible explanations for this behavior. The variability of the generated text prompts could be too small to address the variety of the images. The measures that I use to discriminate strings in the text prompt generation process could be insufficient for sensible discrimination.

Comparison to traditional inpainting methods

High performances can be observed for both traditional inpainting strategies across all evaluation metrics employed. For example, the Jaccard index for the Navier-Stokes-based inpainting approach reaches a value of 0.97 in the case of the tree-shaped mask. Similar to the behavior of the LaMa GAN, I observe an increasing performance for the tree-shaped mask compared to the randomly generated masks for both traditional methods. Again, I attribute this to the size and distribution of the masked area.

This observation is conflicting with findings from the qualitative analysis. In Table 5.6, I notice significant semantic inconsistencies in the results obtained with these approaches. I attribute the high similarity scores that the traditional methods achieve to an inherent bias in the similarity measures, to possible counterintuitive results of evaluation metrics, and to a possible weakness of the evaluation metrics in measuring semantic consistency.

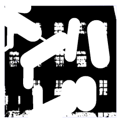
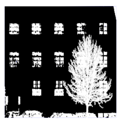
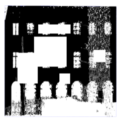
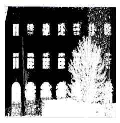
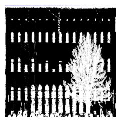
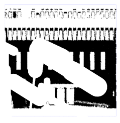
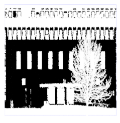
Quantitative evaluation of the inference on real conflict maps

I quantitatively evaluated the inpainting results on real conflict maps by masking the conflict maps with randomly generated, and tree-shaped masks. Using the unmasked images as ground truth information, I quantified the inpainting performance in a more reliably as if I used conflict maps derived from corresponding LOD3 models.

In the case of randomly generated masks, I observe an increasing similarity to the ground truth information for the inpainting results. This is particularly evident when examining the SSIM scores and the cosine similarity of deep-learning features. I observe an average increase of 0.31 in the SSIM values and an average increase of 0.17 in the cosine similarity of the deep learning features. The initial average for the latter is 0.81 pre-inpainting, which significantly improves to an average of 0.98 across all models after the inpainting. This underscores the effectiveness of inpainting with pre-trained and personalized Stable Diffusion models on real conflict maps, leading to substantial enhancements in similarity with ground truth information.

In the case of tree-shaped masks, the improvement in similarity by inpainting is less prominent than for the random masks. I find only a small improvement of at average 0.07 in the corresponding SSIM scores. Almost no change (0.004) is evident in the average similarity of the deep-learning-based features. The initial average is 0.93 pre-inpainting, 0.94 across all models after the inpainting. This corresponds to my

Table 5.13: Quantitative evaluation of the inpainting results on real conflict maps

Binary conflict map superimposed with mask	Similarity to Ground truth (unmasked conflict map)	Stable Diffusion pre-trained	Stable Diffusion (pers. 1)	Stable Diffusion (pers. 2)	Stable Diffusion (pers. 3)
	SSIM: 0.28 IoU: **0.39** DL : 0.70	SSIM: 0.82 IoU: 0.27 **DL: 0.98**	SSIM: 0.83 IoU: 0.24 **DL: 0.98**	**SSIM: 0.85** IoU: 0.26 **DL: 0.98**	SSIM: 0.81 IoU: 0.24 DL: 0.97
	SSIM: 0.71 **IoU: 0.62** DL : 0.90	SSIM: 0.76 IoU: 0.25 DL: 0.89	SSIM: 0.75 IoU: 0.27 DL: 0.93	**SSIM: 0.77** IoU: 0.22 **DL: 0.94**	SSIM: 0.76 IoU: 0.26 Dl: 0.90
	SSIM: 0.68 **IoU: 0.83** DL : 0.87	SSIM: 0.85 IoU: 0.61 **DL: 0.98**	**SSIM: 0.87** IoU: 0.67 **DL: 0.98**	SSIM: 0.85 IoU: 0.61 **DL: 0.98**	SSIM: 0.84 IoU: 0.63 **DL: 0.98**
	SSIM: 0.78 **IoU: 0.91** DL : 0.96	SSIM: 0.79 IoU: 0.58 **DL: 0.97**	**SSIM: 0.81** IoU: 0.59 **DL: 0.97**	SSIM: 0.8 IoU: 0.55 DL: 0.96	SSIM: 0.77 IoU: 0.55 DL: 0.96
	SSIM: 0.67 **IoU: 0.70** DL : 0.86	SSIM: 0.92 IoU: 0.47 **DL: 0.99**	SSIM: 0.92 IoU: 0.51 **DL: 0.99**	SSIM: 0.91 IoU: 0.46 DL: 0.98	**SSIM: 0.93** IoU: 0.45 **DL: 0.99**
	SSIM: 0.76 **IoU: 0.75** DL : 0.94	SSIM: 0.81 IoU: 0.40 DL: 0.95	SSIM: 0.83 IoU: 0.40 DL: 0.95	**SSIM: 0.84** IoU: 0.46 **DL: 0.96**	SSIM: 0.82 IoU: 0.41 Dl: 0.95
	SSIM: 0.56 **IoU: 0.74** DL : 0.80	SSIM: 0.81 **IoU: 0.74** DL: 0.96	**SSIM: 0.85** IoU: 0.70 DL: 0.97	**SSIM: 0.85** IoU: 0.71 **DL: 0.98**	SSIM: 0.84 IoU: 0.71 DL: 0.97
	SSIM: 0.77 **IoU: 0.83** **DL : 0.93**	SSIM: 0.81 IoU: 0.68 DL: 0.90	**SSIM: 0.83** IoU: 0.66 **DL: 0.93**	SSIM: 0.82 IoU: 0.65 DL: 0.92	**SSIM: 0.83** IoU: 0.66 DL: 0.91

expectations because the Stable Diffusion models tend to inpaint structures similar to trees, as illustrated in Table 5.3.

In contrast to the SSIM and deep-learning-based features, I observe that the IoU shows no improvement by the inpainting. More specifically, I observe a consistent decrease of -0.24 in similarity to the ground truth across all models and masks for this particular evaluation metric. This result conflicts with my observations from the qualitative evaluation of the inpainting results and the quantitative evaluation of SSIM score and the deep-learning-based features, where I observe an improvement in similarity to the ground truth data.

6 Discussion

6.1 Settings of hyperparameters

As demonstrated in Table 5.2, finding suitable hyperparameters is not a trivial task. The suitability of specific settings depends on various factors discussed in the following sections.

6.1.1 Thresholds

The minimal detectable deviation of the MLS data from the LOD2 model, set with the parameter t_1 in Equation 5.4, is limited by the deviation of the LOD2 model from the building in reality. This deviation can vary within each façade. Apart from the overall geometric accuracy of the building model, such information is commonly unavailable. An optimal set of additional thresholds to maximize the extracted information content depends on the individual structure of each building and has to be identified experimentally.

6.1.2 Triangle subdivision iterations

As illustrated in Table 5.2 and Figure 4.3, the geometric resolution largely depends on the number of iterations during the triangle subdivision process. The computational effort that is required for the generation is also strongly influenced by this number. Since the outcome depends on the properties of the initial façade, such as the overall size or the geometric shape, no simple general rule that could be applied in any scenario can be established. When considering official LOD2 data provided by the Bavarian Surveying Administration, I find that using eight iterations in the triangle subdivision process yields a good compromise between geometric accuracy and computational effort.

6.2 Probabilistic information for the generation of conflict maps

Incorporating probabilistic information into the generation process of the conflict maps offers a possibility to mitigate some of the issues discussed previously. Prior knowledge about the uncertainty of the LOD2 building model and the MLS point cloud could be utilized in a similar approach as that of Wysocki *et al.* [8, 22]. It could also be used to predict suitable settings of hyperparameters. Further information, such as the height of a particular triangle above the ground, could also be included similarly.

6.3 Randomly generated masks

6.3.1 Suitability of randomly generated masks

The randomly generated masks were used as a proof of concept, which might not have an application in real scenarios. In real application scenarios, masks have the shape of occluding objects such as vegetation, cars, or pedestrians. I assume that objects resembling the shape of random masks are very rare. Therefore, a Deep Learning model is trained to complete images according to masks with very rare shapes in my work. Since the shape of the mask can largely influence the inpainting outcomes, a resulting model might not be well specialized for completing conflict maps, even if trained with actual façade data.

6.3.2 Incorporating object databases

Therefore, to improve the inpainting performance, it could be beneficial to replace the random masks that are used for training and personalization with masks that represent such real objects. Since a significant number of masks is required, an extensive database of occluding objects would be necessary. More realistic masks could be derived from such a database using additional augmentation methods [80].

6.4 Suitability of the evaluation metrics

When analyzing and interpreting the results of the quantitative evaluation, the suitability of the evaluation metrics has to be considered. Several factors can compromise the evaluation results.

6.4.1 Evaluation with unrealistic masks

Using unrealistic masks introduces a potential divergence between evaluation results and the actual performance of the model. The evaluation outcomes may not accurately reflect the real-world capabilities of the models in such instances. Conclusions drawn from such evaluations can fail to provide meaningful insights for improving performance in real-world scenarios.

6.4.2 Inherent bias in the similarity measures

I assume a potential inherent bias in the similarity measures favoring higher scores within my dataset. All conflict maps that I use in my experiments share the characteristic of being binary images. This shared property may introduce a positive similarity bias, resulting in consistently high measurement values, even when conflict maps exhibit observable semantic differences discernible to the human eye. To validate this assumption, further experiments would be required.

6.4.3 Suitability of the SSIM score

Nilsson and Akenine-Möller have pointed out that the SSIM sometimes yields counter-intuitive results that do not necessarily align with human perception [74]. I assume that such behavior could be part of an explanation for the traditional inpainting strategies achieving comparative high values of the SSIM scores, even with weaknesses in recognition of global semantic information as illustrated in Table 5.6.

6.4.4 Suitability of the Jaccard index

The results from the quantitative analysis using the Jaccard index contradict those obtained from assessments using SSIM and deep learning features, as well as the findings from the qualitative analysis. The level of fragmentation in the conflict maps may serve as an explanation for this phenomenon. High levels because of noise and complex scene structures are observable, for example, in Table 5.2. Greater overlap due to larger continuous areas within the images may positively impact the Jaccard index. However, further experiments would be necessary to verify this assumption.

6.4.5 Measuring semantic information

All the evaluation metrics I deploy in this work measure the similarity of two images. The semantic correctness of the unseen façade details inpainted into the missing regions is related to this visual similarity to the ground truth information, but not identical. I interpret the perceived similarity of the completed conflict map to an image representing ground truth information as a good approximation of the semantic similarity. The evaluation metrics that I use are incapable of discriminating between semantically meaningful image content, such as structures that belong to a window, and irrelevant features, such as noise. Nor do they take possible symmetric properties into account. To accurately measure the semantic similarity, high-level information on the conflict maps has to be quantified. A solution to this challenging task might be developing and deploying a designated deep learning approach.

6.5 Substituting the text-prompt

6.5.1 Precision of description

As illustrated in Table 4.3, describing the image content that a missing region should be filled with can be challenging. In the tables 5.10, 5.11 and 5.12, it becomes evident that my method for the substitution of manual text prompt selection does not positively affect the performance of the Stable Diffusion models. An explanation might be that the automatically constructed text prompts do not describe the content that should be inpainted into the missing parts of the conflict map with sufficient precision.

6.5.2 Suitability of the used metrics

I utilize certain features to automatically determine a text prompt. However, these criteria may not sufficiently identify a favorable combination of individual strings to create an appropriate output. The overall quality of a final inpainting result could rely on image properties beyond those I have explored. Additional experiments are required to ascertain the adequacy of the selected features and identify the effectiveness of alternative features.

6.6 Post-Processing of conflict maps

6.6.1 Resampling

Since some Deep Learning Networks have fixed sizes of input and output images, a resampling step can be required for further applications of the completed conflict maps. This way, they could be used as a texture or be combined with additional geo-referenced information. The original coordinates of the façade corners would be required for this resampling.

6.6.2 Quality enhancement

I find that in many cases, the inpainted areas, as well as the conflict maps themselves, contain disturbances. This is, for example, illustrated in Table 5.8. For example, within a 3D-reconstruction pipeline, post-processing steps could be used to enhance the image quality to make use of the conflict maps. Typical noise filters could be used for this purpose.

6.7 Substituting vegetation-shapes in masks

6.7.1 General considerations

As pointed out in Table 5.7, replacing masks with simpler shapes could yield improvements in some instances. A balance must be struck between the loss of ground truth information with possible resulting semantic inconsistencies, and the benefit of avoiding the inpainting of vegetation-like structures. Establishing such a balance is non-trivial and requires in-depth knowledge about the specific use case scenario.

6.7.2 Identifying vegetation

In my experiments, I manually substituted the tree-shaped mask with a rectangle. Automatically identifying vegetation-like structures in binary masks and assessing their bounding boxes for replacement with rectangles is challenging. In real application scenarios, trees might merge into one another. Incompleteness and unfavorable shapes that are not typical for vegetation might contribute to this difficulty.

7 Limitations

7.1 Limitations in applicability

7.1.1 Transparent façades

Though my method effectively addresses issues with conventional building façades, I anticipate challenges when deploying it on façades predominantly composed of glass. In such a scenario, only the interior of the building would be captured by the laser scanner. In the most unfavorable circumstances, the entire façade might be deemed conflicting, even when pertinent structures for 3D reconstruction exist, such as extruded façade objects made of glass.

7.1.2 Façade boundaries

Regarding extruded façade elements such as balconies that wrap around building corners, caution is advised. In these instances, parts of such elements may not be represented in any conflict map. This problem is illustrated in Figure 7.1. In this example, façade a and b would be represented by two individual conflict maps. The green lines represent the parts of the façades considered as confirming while the red lines represent those considered conflicting. The blue lines represent areas that are not covered by any of the two conflict maps. The lost information about the balcony could lead to an erroneous 3D reconstruction of the LOD 3 building model.

7.2 Scalability

As discussed earlier, the maximum geometric resolution is partly dependent on the number of triangle subdivisions. In certain scenarios, the required geometric resolution may necessitate an excessively high number of triangle subdivisions, potentially leading to memory issues. This situation may arise when dealing with point clouds characterized by a very high point density or generally large datasets. Consequently, the processing of large datasets might require a reduced geometric resolution.

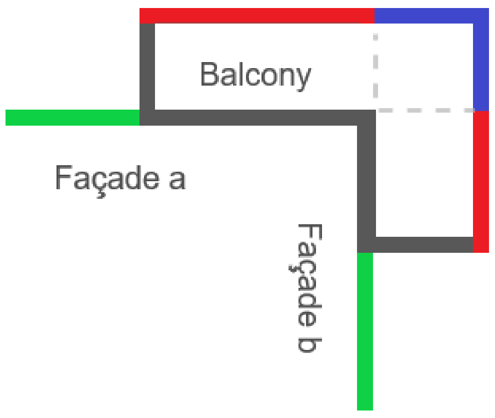


Figure 7.1: Schematic illustration of the coverage problem for extruded façade objects that wrap around corners of building.

7.3 Absence of probabilistic information

My approach does not incorporate information regarding the uncertainty of the MLS points and the LOD 2 building models. As demonstrated in Table 5.2, the absence of such information may result in the loss of relevant façade details. The optimization of hyperparameters to minimize information loss remains an unresolved challenge. Under the unfavorable circumstance that a threshold is smaller than the combined uncertainty of the MLS points and the LOD 2 building models, the consequent information loss could lead to an erroneous 3D reconstruction result.

7.4 Semantic Information

The information in the conflict maps relates to openings in the façade, such as windows. Extruded façade objects are also considered. When passing the conflict maps to the deep learning models, we only consider the conflict map as a binary image without further semantic information. More experiments are necessary to discern to what extent the deep learning models can understand the binary conflict maps as information about the façades of buildings.

8 Conclusion

This work demonstrates that deep-learning-based methods can be utilized for the semantic inpainting of unseen façade objects into 2D conflict maps. This is underlined by the average increase of 0.31 in the SSIM values and an average increase of 0.17 in the cosine similarity of the deep learning features that is evident in Table 5.13. Both, Generative Adversarial Networks (GANs) and DMs prove to be effective in this context. The LaMa GAN encounters challenges when handling relatively large masks. Overall, the inpainting of smaller masks is qualitatively observed to be more successful compared to larger masks.

For personalizing the pre-trained Stable Diffusion with Dreambooth and training the LaMa GAN, I utilized conflict maps derived from synthetic semantic city models and such that I derived from annotated images. Despite the slightly diminished performance of the LaMa GAN trained in this manner compared to the pre-trained version, particularly the personalization experiments with Dreambooth underscore the applicability of such data to facilitate the use of deep learning models for inpainting unseen façade objects into 2D conflict maps.

Notably, when using masks shaped like trees, the diffusion networks tend to inpaint structures similar to trees in my experiments. This behavior is not evident in the case of randomly generated masks. This is, for example, reflected in the Jaccard index of the pre-trained Stable Diffusion model in Table 5.11. This measure drops from 0.72 to 0.66 if exposed to the tree-shaped mask instead of randomly generated masks. The LaMa GAN does not demonstrate such inpainting tendencies. Although in my experiments, the personalization with Dreambooth does not lead to a significant performance increase in diffusion models compared to their pre-trained counterparts, it helps to mitigate their tendency to inpaint structures similar to trees. Although it does not eliminate the issue, the improvements by this personalization can contribute to enhanced accuracy in 3D reconstruction.

Future work could focus on substituting the text prompt for the Stable Diffusion models more sophisticatedly. My approach may find application by integrating it into existing pipelines for 3D reconstruction. Additionally, there is the possibility to deploy my approach to conflict maps, acquired through the probabilistic methodology introduced by Wysocki *et al.* [22]. Another avenue for future exploration is refining synthetic training data generation to achieve higher realism. Increasing the computational efficiency of the conflict map generation and thus improving the scalability of my approach could represent another objective of future work.

Glossary

ALS Airborne Laser Scanning. 6

B-Rep Boundary Representation. 4

BIM Building Information Modeling. 3

DE DHHN2016 NH German Main Height Network 2016. 34

DM Diffusion Model. 10, 67, 69

ETRS89 European Terrestrial Reference System 1989. 34

FFC Fast Fourier Convolution. 14

FID Frechet Inception Distance. 41

FME Feature Manipulation Engine. 20, 34, 40

FMM Fast Marching Method. 17

GAN Generative Adversarial Network. 8–10, 14, 30, 41, 43, 44, 47, 49, 56, 57, 69

GML Geography Markup Language. 3

GPS Global Positioning System. 6

INS Inertial Navigation System. 6

IoU Intersection over Union. 38, 59

LOD levels of Detail. 3–8, 12–14, 18, 21, 23–25, 30, 34–37, 39, 42, 43, 50–53, 57, 60, 61, 65, 69, 70, 72

LoRA Low Rank Adaptions. 17

MLS Mobile Laser Scanning. 5, 6, 12, 13, 18, 19, 21–23, 34, 60, 61, 69

OBJ Wavefront Object Format. 40

OGC Open Geospatial Consortium. 3

PDEs Partial Differential Equations. 10, 17

RANSAC Random Sample Consensus. 25

SSIM Structural Similarity Index. 31, 32, 37, 38, 41, 51, 55–57, 59, 62, 67

UTM Universal Transverse Mercator. 34

WGAN Wasserstein GAN. 9

XML Extensible Markup Language. 3, 13, 14, 36

List of Figures

List of Tables

Bibliography

[1] Olaf Wysocki, Yan Xia, Magdalena Wysocki, Eleonore Grilli, Ludwig Hoegner, Daniel Cremers, and Uwe Stilla. Scan2LoD3: reconstructing semantic 3D building models at LoD3 using ray casting and Bayesian networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 6548–6558. arXiv, May 2023. doi: 10.48550/ARXIV.2305.06314.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] Robin Rombach and Patrick Esser. runwayml/stable-diffusion-inpainting. Online, 2023. URL `https://huggingface.co/runwayml/stable-diffusion-inpainting`. Acessed: 01.07-2023.

[4] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2149–2159. arXiv, September 2022. doi: 10.48550/ARXIV.2109.07161.

[5] Nataniel Ruiz, Yuanzhen Li, Varun Jampan, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In Lisa O'Conner, editor, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510, 2023.

[6] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. Generation of multi-LOD 3D city models in CityGML with the procedual modelling engine Random3Dcity. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W1:51–59, sep 2016. doi: 10.5194/isprs-annals-iv-4-w1-51-2016.

[7] Radim Tylecek. The CMP facade database. *Research Reports of CMP, Czech Technical University in Prague*, (24), 2013. ISSN 1213-2365.

[8] Olaf Wysocki, Ludwig Hoegner, and Uwe Stilla. Refinement of semantic 3D building models by reconstructing underpasses from MLS point clouds. *International Journal of Applied Earth Observation and Geoinformation*, 111:102841, jul 2022. doi: 10.1016/j.jag.2022.102841.

[9] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. Applications of 3D city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, dec 2015. doi: 10.3390/ijgi4042842.

[10] Olaf Wysocki, Ludwig Hoegner, and Uwe Stilla. MLS2LoD3: refining low LoDs building models with MLS point clouds to reconstruct semantic LoD3 building models. In Thomas H. Kolbe, Andreas Donaubauer, and Christof Beil, editors, *Recent Advances in 3D Geoinformation Science: Proceedings of the 18th 3D GeoInfo Conference*, 2023.

[11] Thomas Fröch, Olaf Wysocki, Ludwig Hoegner, and Uwe Stilla. Reconstructing façade details using MLS pointclouds and Bag-of-Words approach. In Thomas H. Kolbe, Andreas Donaubauer, and Christof Beil, editors, *Recent Advances in 3D Geoinformation Science: Proceedings of the 18th 3D GeoInfo Conference*, 2023.

[12] Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5485–5493. IEEE, July 2017. URL `https://openaccess.thecvf.com/content_cvpr_2017/html/Yeh_Semantic_Image_Inpainting_CVPR_2017_paper.html`.

[13] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In Vittorio Ferrari, Martial Herbert, Cristian Sminchiescu, and Yair Weiss, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, page 85–100. NVIDIA Corporation, September 2018. URL `https://openaccess.thecvf.com/content_ECCV_2018/papers/Guilin_Liu_Image_Inpainting_for_ECCV_2018_paper.pdf`.

[14] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):1–14, jul 2017. doi: 10.1145/3072959.3073659.

[15] Gerhard Gröger and Lutz Plümer. CityGML–interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, 2012.

[16] Thomas H. Kolbe. *Representing and exchanging 3D city models with CityGML*, chapter 3D Geo-Information Sciences. Lecture Notes in Geoinformation and Cartography, pages 15–31. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-87395-2. doi: https://doi.org/10.1007/978-3-540-87395-2_2.

[17] Gerhard Groger, Thomas H. Kolbe, Angela Czerwinski, and Claus Nagel. OpenGIS® City Geography Markup Language (CityGML) Encoding Standard, Version 1.0.0., 2008.

[18] Thomas H. Kolbe, Tatjana Kutzner, Carl Stephen Smyth, Claus Nagel, Carsten Roensdorf, and Charles Heazel. Ogc city geography markup language (citygml) part 1: Conceptual model standard, 2021. URL `http://www.opengis.net/doc/IS/CityGML-1/3.0`.

[19] Karlsruhe Institute of Technology (KIT) Institute for Applied Computer Science (Campus North). CityGML example FZK-Haus. Online, 2021. URL `https://www.citygmlwiki.org/index.php?title=FZK_Haus`. Accessed: 10.07.2023.

[20] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37, sep 2016. doi: 10.1016/j.compenvurbsys.2016.04.005.

[21] Thilo Brüggemann and Petra von Both. *3D-stadtmodellierung: CityGML*, chapter 10, pages 177–192. Springer Fachmedien Wiesbaden, 2015.

[22] Olaf Wysocki, Eleonore Grilli, Ludwig Hoegner, and Uwe Stilla. Combining visibility analysis and deep learning fror refinement of semantic 3D building models by conflict classification. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, X-4/W2-2022:289–296, oct 2023. doi: 10.5194/isprs-annals-x-4-w2-2022-289-2022.

[23] Iván Puente, H. González-Jorge, Pedro Arias, and Julia Armesto. Land-based mobile laser scanning systems: A review. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W12:163–168, sep 2012. doi: 10.5194/isprsarchives-xxxviii-5-w12-163-2011.

[24] H. Huang, M. Michelini, M. Schmitz, L. Roth, and H. Mayer. LOD3 building reconstruction from multi-source images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2020:427–434, aug 2020. doi: 10.5194/isprs-archives-xliii-b2-2020-427-2020.

[25] Sebastian Tuttas and Uwe Stilla. Reconstruction of façades in point clouds from multi aspect oblique ALS. *ISPRS Annals of the Photogrammetry, Remote*

Sensing and Spatial Information Sciences, II-3/W3:91–96, oct 2013. doi: 10.5194/isprsannals-ii-3-w3-91-2013.

[26] Bryan G. Pantoja-Rosero, Radhakrishna Achanta, Mateusz Kozinski, Pascal Fua, Fernando Perez-Cruz, and Katrin Beyer. Generating LoD3 building models from structure-from-motion and semantic segmentation. *Automation in Construction*, 141 (104430), 2022. doi: https://doi.org/10.1016/j.autcon.2022.104430.

[27] Nora Ripperda. *Rekonstruktion von fassadenstrukturen mittels formaler grammatiken und reversible jump markov chain monte carlo sampling*. PhD thesis, Leibnitz Universität Hannover, 2010. Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover.

[28] Hongchao Fan, Gefei Kong, and Chaoquan Zhang. An interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network. *Big Earth Data*, 5(1):49–65, January 2021. doi: 10.1080/20964471.2021.1886391.

[29] Ludwig Hoegner and Georg Gleixner. Automatic extraction of facades and windows from MLS point clouds using voxelspace and visibility analysis. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2022:387–394, May 2022. doi: 10.5194/isprs-archives-xliii-b2-2022-387-2022.

[30] S. Hensel, S. Goebbels, and M. Kada. Facade reconstruction for textured LOD2 CityGML models based on deep learning and mixed integer linear programming. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:37–44, may 2019. doi: 10.5194/isprs-annals-iv-2-w5-37-2019.

[31] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, January 2018. doi: 10.1109/MSP.2017.2765202. URL `https://ieeexplore.ieee.org/abstract/document/8253599`.

[32] Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z. Li. A survey on generative diffusion model, September 2022.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Gharamani, Max Welling, Corinna Cortes, neil D. Lawrence, and Kilian Q. Winberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 1–9, 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf`.

[34] Farzan Farnia and Asuman Ozdaglar. Do GANs always have nash equilibria? In Hal 3 Daumé and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3029–3039, 2020.

[35] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, Barcelona, Spain., 2016.

[36] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL `https://arxiv.org/abs/1802.05957`.

[37] Omar Elharrouss, Noor Almaadeed, Somaya Al-Maadeed, and Younes Akbari. Image inpainting: A review. *Neural Processing Letters*, 51(2):2007–2028, dec 2020. doi: 10.1007/s11063-019-10163-0.

[38] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. January 2017. doi: 10.48550/ARXIV.1701.04862.

[39] Maciej Wiatrak, Stefano V. Albrecht, and Andrew Nystrom. Stabilizing generative adversarial networks: A survey. *arXiv preprints*, September 2019. doi: 10.48550/ARXIV.1910.00927.

[40] Zhiming Zhou, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang. Lipschitz generative adversarial nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, PMLR*, number 97, pages 7584–7593, 2019.

[41] Shichang Tang. Lessons learned from the training of GANs on artificial datasets. *IEEE Access*, 8:165044–165055, 2020. ISSN 2169-3536. doi: 10.1109/access.2020.3022820.

[42] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223, 2017. URL `http://proceedings.mlr.press/v70/arjovsky17a/arjovsky17a.pdf`.

[43] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 5562–5571, 2012. URL `https://proceedings.mlr.press/v139/`.

[44] Christopher Jarzynski. Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Physical Review*, 56(5):5019–5035, 1997.

[45] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, PMLR*, volume 37, pages 2256–2265, 2015.

[46] Jireh Jam, Connah Kendrick, Kevin Walker, Vincent Drouard, Jison Gee-Sern Hsu, and Moi Hoon Yap. A comprehensive review of past and present image inpainting methods. *Computer Vision and Image Understanding*, 203:103147, feb 2021. doi: 10.1016/j.cviu.2020.103147.

[47] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of Geographics Tools*, 6(1):23–34, April 2004. doi: https://doi.org/10.1080/10867651.2004.10487596.

[48] M. Bertalmio, A.L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. IEEE Comput. Soc, 2001. doi: 10.1109/cvpr.2001.990497.

[49] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5505–5514. IEEE, 2018. URL `https://openaccess.thecvf.com/content_cvpr_2018/html/Yu_Generative_Image_Inpainting_CVPR_2018_paper.html`.

[50] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. EdgeConnect: Generative image inpainting with adversarial edge learning. *arXiv preprint*, January 2019. doi: 10.48550/ARXIV.1901.00212.

[51] Khairil Ariffin Bin Yahya. Facade segmentation using neural networks and building model conflicts. Master's thesis, Technische Universität München, School of Engineering and Design, 2023.

[52] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4479–4488. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/2fd5d41ec6cfab47e32164d5624269b1-Paper.pdf`.

[53] Suraj Patil, Pedro Cuenca, and Valentine Kozin. Training stable diffusion with Dreambooth using diffusers. Online, July 2022. URL `https://huggingface.co/blog/dreambooth`.

[54] Unknown. Dreambooth for the inpainting model. Online, July 2023. URL `https://github.com/huggingface/diffusers/tree/main/examples/research_projects/dreambooth_inpaint`. Acessed 03.11.2023.

[55] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[56] J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, jan 1999. doi: 10.1137/s0036144598347059.

[57] Luming Tang, Nataniel Ruiz, Qinghao Chu, Yuanzhen Li, Aleksander Holynski, David E. Jacobs, Bharath Hariharan, Yael Pritch, Neal Wadhwa, Kfir Aberman, and Michael Rubinstein. RealFill: reference-driven generation for authentic image completion. *arXiv preprint*, 2023. doi: 10.48550/ARXIV.2309.16668.

[58] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint*, August 2023. doi: 10.48550/ARXIV.2308.03303.

[59] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[60] Filip Biljecki and Ken Arroyo Ohori. Automatic Semantic-preserving Conversion Between OBJ and CityGML. In *Eurographics Workshop on Urban Data Modelling and Visualisation 2015*, pages 25–30, Delft, Netherlands, November 2015. doi: 10.2312/udmv.20151345.

[61] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprints*, January 2018. doi: 10.48550/ARXIV.1801.09847.

[62] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[63] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[64] Leonardo Mariga. pyRANSAC-3D, 10 2022. URL `https://github.com/leomariga/pyRANSAC-3D`.

[65] Alex Clark. Pillow (pil fork) documentation, 2015. URL `https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf`. Acessed: 30.11.2023.

[66] Tyleček Radim and Šára Radim. Spatial pattern templates for recognition of objects with regular structure. In Joachim Weickert, Matthias Hein, and Bernt Schiele, editors, *Pattern Recognition*, volume 8142, pages 364–374, Berlin, Heidelberg, 2013. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-40602-7_39.

[67] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. ISSN 2167-8359. doi: 10.7717/peerj.453. URL `https://doi.org/10.7717/peerj.453`.

[68] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1): 32–46, apr 1985. doi: 10.1016/0734-189x(85)90016-7.

[69] Bayerische Vermessungsverwaltung. 3D-Gebäudemodelle (LoD2). Online, 2023. URL `https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=lod2`. Accessed: 30.11.2023.

[70] Benedikt Schwab, Olaf Wysocki, Manoj Biswanath, and Julia Barbosa. tum2twin. Online, 2023. URL `https://github.com/tum-gis/tum2twin`. Accessed: 30.11.2023.

[71] Hadi Yazdi, Qiguan Shu, Thomas Rötzer, Frank Petzold, and Ferdinand Ludwig. TreeML-Data a multidisciplinary and multilayer urban tree dataset. sep 2023. doi: 10.21203/rs.3.rs-3358921/v1.

[72] Gintautas Palubinskas. Image similarity/distance measures: what is really behind MSE and SSIM? *International Journal of Image and Data Fusion*, 8(1):32–53, dec 2016. doi: 10.1080/19479832.2016.1273259.

[73] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Ero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, apr 2004. doi: 10.1109/tip.2003.819861.

[74] Jim Nilsson and Tomas Akenine-Möller. Understanding SSIM. *arXiv preprints*, June 2020. doi: 10.48550/ARXIV.2006.13846.

[75] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, 2019.

[76] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2009. doi: 10.1109/cvpr.2009.5206848.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[78] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[79] Thomas Fröch, Olaf Wysocki, and Benedikt Schwab. CityGML2OBJ 2.0. Online, 2023. URL `https://github.com/tum-gis/CityGML2OBJv2`. Accessed: 30.11.2023.

[80] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, jul 2019. doi: 10.1186/s40537-019-0197-0.