DOI:10.15774/PPKE.ITK.2023.007

Deep Learning Based Methods on 2D Image and 3D Structure Inpainting

A thesis submitted for the degree of
Doctor of Philosophy

Yahya Ibrahim

Scientific adviser:
Csaba Benedek, DSc

Roska Tamás Doctoral School of Sciences and Technology
Pázmány Péter Catholic University

Budapest, 2023

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

I would like to dedicate this thesis to my parents

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Acknowledgements

First of all, I owe my deepest gratitude to my supervisor *Prof. Csaba Benedek* for his continuous support, motivation and patience during my Ph.D. study and work.

I would like to express my sincere gratitude to all the Pazmany Peter Catholic University (PPCU) members, thanks to *Prof. Péter Szolgay* who provided me the opportunity to study here, and *Prof. Gábor Szederkényi* for providing me with all the necessary facilities for the research, thanks to *Mrs. Vida Tivadarné* for her continuous assistance with administrative concerns. I deeply thank *Prof. Árpád Csurgay* for the motivation and encouragement. I would like to thank all colleagues at PPCU with whom I spent these past few years *Sam Khozama*, *Ward Fadel*, *Jalal Alafadi*, *Nawar Al.hemeary*. Special thanks go to *Marwan Hassan* for his tremendous support since the first day of my international journey.

I thank the reviewers of my thesis for their work and valuable comments.

The support of the Institute for Computer Science and Control (SZTAKI) is also gratefully acknowledged for employing me to proceed with my research. My sincere thanks go to *Prof. Támas Szirányi* for offering me the job. I thank my closest colleagues from SZTAKI, Machine Perception Research Laboratory for their help: *Lóránt Kovács*, *Balázs Nagy*, *Örkény Zováthi*, *Zsolt Jankó*, *Marcell Kégl*.

For further financial support, my research work was supported by various projects and grants: by Stipendium Hungaricum scholarship program; by the National Research, Development and Innovation (NRDI)

DOI:10.15774/PPKE.ITK.2023.007

Office of Hungary within the frameworks of the National Laboratory for Autonomous Systems (NLAS), and the Artificial Intelligence National Laboratory (MILAB); by the NRDI Fund (OTKA) grants K-120233, KH-125681 and K-143274; by the Hungarian R&D grants EFOP-3.6.2-16-2017-00013, and TKP2021-NVA-01.

To all my friends, thank you for your kindness and assistance throughout my times of need. Your friendship strengthens me to face difficulties. I cannot list all the names here, but you are always on my mind.

Last but not least, I would like to express my sincere gratitude To: the kind, loving, simple people of my small village "*Balghounes*", my big family *"Ibrahim"* and *"Omran"*. To my first mentors and forever guardian angels my grandfathers *Yahya* and *Mohammed*. To my grandmothers *Radiah*, *Manera*, and *Hajar* for their neverending blessings. To my godfather *Nasr* for his kind heart. To my father *Ahmed*, whose wise words have guided me through the hard times. To my mother *Zainab* and her unconditional love that I will never be able to pay back even if I live for a thousand years. To my brothers: *Mohammed* whose feelings were always beside me, *Yousef* who has always been here for me whenever and wherever needed, and to the hope of our small family *Sharaf* who has a bright future ahead of him. To *Ghaith* my brother from another mother.

To those whom I may have not mentioned by name but who supported me directly or indirectly in accomplishing my Ph.D. research.

DOI:10.15774/PPKE.ITK.2023.007

Abstract

In this thesis, novel deep learning-based models for automatic analysis and inpainting 2D images and 3D structures are proposed. The proposed methods are primarily focused toward two specific use cases: (i) Inpainting 2D images of masonry walls in archaeology, and civil engineering applications, (ii) Filling up the missing regions in point clouds of 3D models acquired using Mobile Laser Scanning (MLS) systems. The proposed methods have been evaluated on large databases containing a variety of synthetic and real-world scenarios. We have compared the proposed approaches to state-of-the-art methods, and the results demonstrate significant advantages over recent state-of-the-art approaches.

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Contents

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

List of Figures

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

List of Tables

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Chapter 1

Introduction

Over the past ten years, deep learning networks have yielded exceptional success in a variety of computer vision tasks (segmentation, classification, object detections, etc.) across a wide range of fields, including manufacturing, autonomous driving, healthcare, archaeology, and civil engineering. However, currently available techniques are still far away from human-level performance [8], due to a variety of reasons, including the low resolution of the used sensors, occlusions, and the limited number of viewpoints used during scanning.

Occlusions may occur under several conditions in machine perception and computer vision applications, whether Lidars, cameras, or multiple sensor technologies are employed. Given that the camera/Lidar – from a fixed viewpoint – can only observe one side of the object being investigated, self-occlusions can occur even in situations with a single object. Moreover, occlusions may be found in a variety of complicated circumstances, for example if a part of an object is outside the field of view or when one or multiple objects occlude each other in the scene.

In object recognition applications, deep neural networks [9, 10] are still behind humans in the presence of occlusion [11, 12], as detection failure rate increases with the increase of the occlusion level. When the degree of occlusion exceeds 50%, objects can hardly be detected [13]. On the other hand, human minds are adept at predicting the invisible components of the scene by observing the visible ones.

In real-life complex situations, occlusion is a major challenge for many sorts

DOI:10.15774/PPKE.ITK.2023.007

of image processing tasks. We can here mention applications like object detection in Advanced Driving Assistance Systems (ADAS), where it is challenging to accurately detect pedestrians or cars when they are partially occluded, especially in crowded scenes. In remote sensing images, the presence of clouds and their shadows can affect the quality of processing these images in several applications [14, 15]. Hence, to make good use of such images, assuming that the background of the occluded parts follows the same pattern as the visible parts of the image, inpainting algorithms can be trained in such cases to remove the occluded regions and fill them with the expected elements (such as road network, vegetation and built-in structures).

Similar concerns can be seen in the context of archaeology and civil engineering applications, where investigating masonry buildings necessitates the precise outlining of their structural components, which are frequently covered up by objects like decorative elements, wall sculptures, or vegetation.

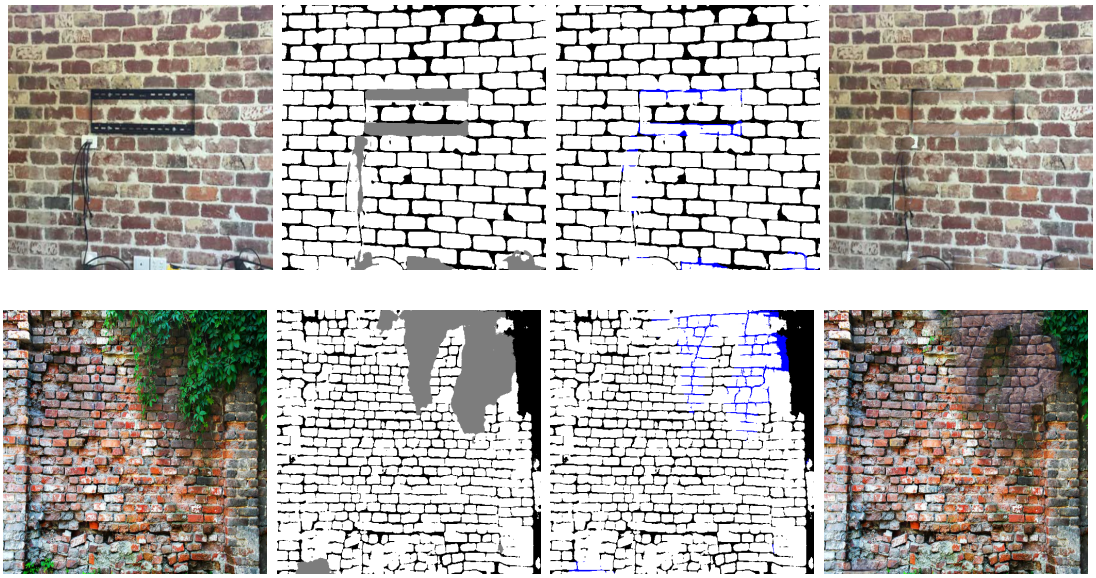
Therefore, occlusion-aware networks have been thoroughly investigated in a variety of fields, including pedestrian detection [16], object tracking [17], face detection [18], and car detection [19].

In contrast to optical cameras, 3D sensors are less restricted by lighting and illumination, and can get accurate 3D geometric data from the scene. Hence, their usage in environmental perception is expanding rapidly.

However, considering both indoor and outdoor mobile mapping platforms, the scanning platforms equipped with 3D sensors cannot access specific locations to scan certain objects from all sides, resulting in incomplete point cloud representations. Therefore, point cloud completion – the estimation of an object's full shape from point sets that only partially describe its geometry – becomes a fundamental key challenge in numerous computer vision and robotic tasks, such as virtual reality (VR)/ augmented reality (AR) applications, and simultaneous localization and mapping (SLAM).

This thesis deals with two selected tasks from the problem family, in which automated occlusion or missing region detection is applied to either 2D images or 3D point clouds, and inpainting networks are then used to reconstruct the occluded/missing portions based on the observable information from the scene.

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) Segmentation image (c) Inpainted structure (d) Inpainted Image

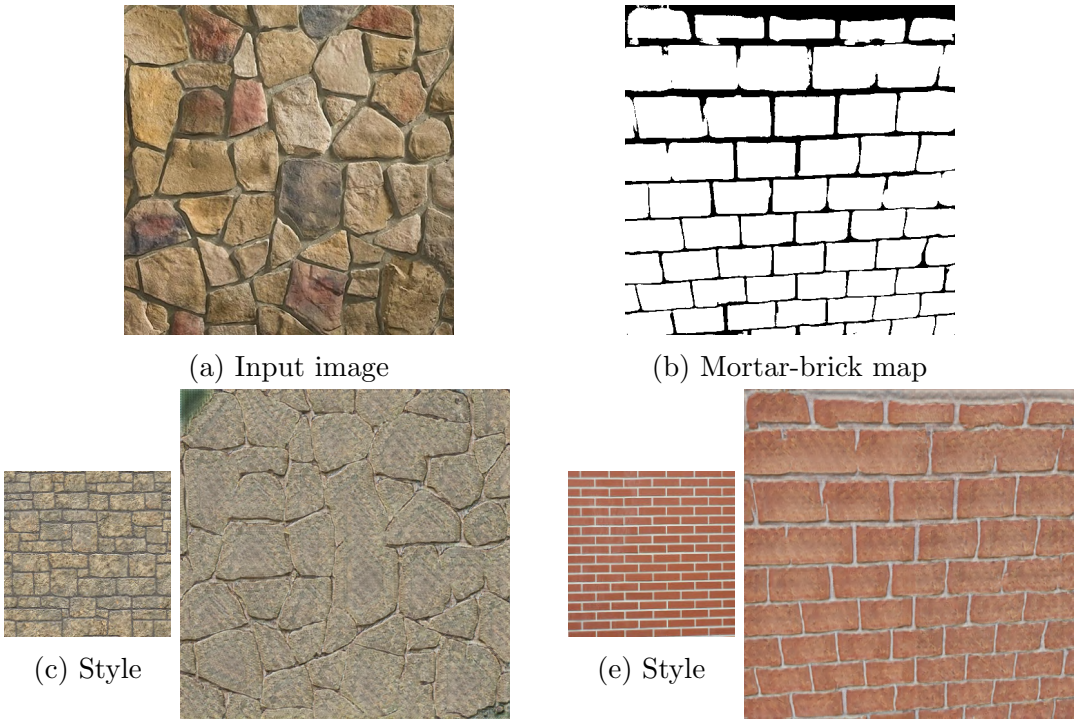
Figure 1.1: Results of the proposed method [1] on two selected samples (Topic 1)

In the following, I give a brief description of the investigated research problems which will be explained in more details in the following chapters of the thesis.

- **Topic 1 - Deep learning-based masonry wall image analysis focusing on inpainting the occluded regions:**

Image-based analysis and documentation of man-built structures is a core step of many applications, including archeology, cultural heritage preservation, architecture and civil engineering. The surveyors in these processes need to extract comprehensive information about the studied sites, among others about the current conditions, types of appropriate treatment, and the expected consequences of any intervention. During the investigation of building masonry, accurate detection and outlining of their structural components is a key initial step of the documentation process. However in a real-world scenario, the wall structure is invisible in some image segments due to occlusion by various objects such as vegetation (see examples in Figure. 1.1(a)), or in damaged wall parts in archaeological applications. A possible way for virtually filling in the missing part of the

DOI:10.15774/PPKE.ITK.2023.007



(a) Input image (b) Mortar-brick map

(c) Style (d) Our output

(e) Style (f) Our output

Figure 1.2: Wall-to-wall style transfer samples, the first row shows the inputs: (a) wall image. (b) Mortar-brick map. The second row represents the style image and our output side by side (Topic 1).

image is the application of image inpainting techniques. The proposed method provides a blind inpainting algorithm for masonry wall images, performing the automatic detection and virtual completion of occluded or damaged wall regions, and a brick segmentation leading to an accurate model of the wall structure.

Our method robustly deals with a wide range of the wall structural components and accurately detects various kinds of possible occlusion or damage effects.

For this purpose, we propose a three-stage deep neural network that comprises a U-Net-based sub-network for wall segmentation into brick, mortar and occluded regions, which is followed by a two-stage adversarial inpainting model. The first adversarial network predicts the schematic mortar-brick pattern of the occluded areas based on the observed wall structure, giving structural information that is essential for archeological and architectural applications Finally, the second adversarial network predicts the RGB pixel values yielding a realistic vi-

DOI:10.15774/PPKE.ITK.2023.007

sual experience for the observer. The proposed algorithm works in an end-to-end manner without assuming any user interaction and it does not require any specific pre-information about the wall structure or the occluding objects.

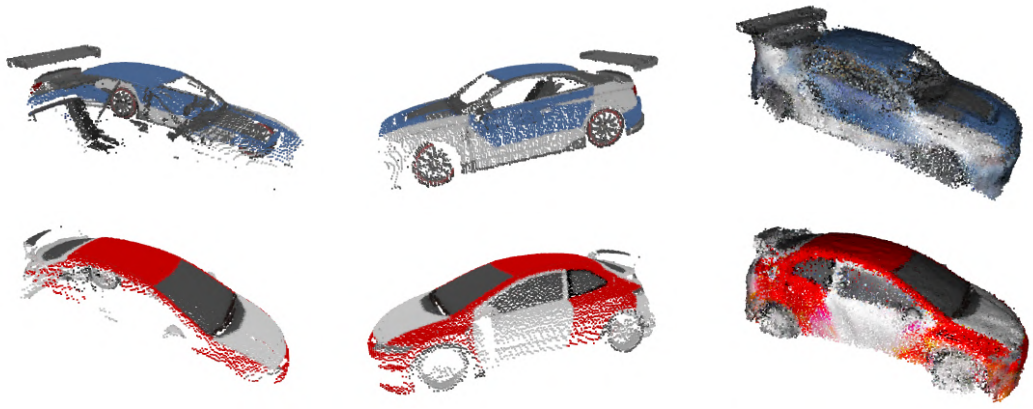
The outputs of our algorithm (see Figure. 1.1(b–d)) allow archaeologists to comprehend the wall structure of the unseen areas and give a more accurate representation that may help in reassembling wall fragments. Moreover, the technique may be utilized in a variety of scenarios in which archaeologists need to change portions of a degraded wall that have been dirty or have lost their original color over time, using style characteristics derived from undamaged wall parts. The algorithm can be used when archaeologists want to build virtual representations of destroyed historical sites, and aim to transfer a selected wall pattern onto a manually sketched wall layout (see Figure. 1.2).

- **Topic 2 - Multi-view based point cloud completion network for mobile laser scanning data:**

Recently, point clouds acquired by 3D scanning systems are widely used in industrial applications. Yet, due to factors like occlusions, light reflections, and limited viewing angles, scanned point clouds of several objects generally lack sufficient detail to adequately represent their full shapes. Therefore, completing the missing information is an essential step in several machine perception and computer vision applications. For instance, creating a 3D point cloud of the complete environment facilitates route planning and decision making in robot applications. In addition, a full 3D representation of the underground mining areas is needed for accurate safety monitoring [20]. When it comes to autonomous driving, a high-resolution 3D representation of the surroundings is required for an accurate and reliable self-localization.

In this topic, we propose a multi-view based approach for completing vehicle point clouds extracted from Riegl VMX-450 MLS scanner measurements. The proposed method utilizes as input preliminary segmented object point clouds acquired by mobile laser scanning. We introduce a new representation for the input point cloud obtained by projections from several viewpoints; in which the geometry and color information is stored in a six-channel, 2D images for each viewpoint. Then, we fill in the blanks areas using a 2D convolutional neural network, The

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (right side view) (b) Input (left side view) (c) Our result

Figure 1.3: Results of the proposed method [2] on two partial point cloud samples (Topic 2).

completed images are then reprojected into the 3D space in order to build a 3D point cloud depicting the whole shape of the object under study (see Figure 1.3). We have tested our proposed method both on a new synthetic benchmark set derived from ShapNet [114] which is a widely used public dataset, and on real-world MLS measurements. We have shown the efficiency and robustness of the method by comparing the model against different state-of-the-art methods.

In the following, we present the outline of the thesis. In Chapter 2 we propose a novel deep learning based approach for inpainting 2D images of masonry walls. The Chapter is organized as follows: In Section 2.1, we introduce the topic by focusing on the problem statement and the objective of the proposed methodology. Section 2.2 summarizes previous works related to state-of-the-art wall image delineation, image inpainting and style transfer algorithms; Section 2.3 shows the data the we worked on and details about the way we generate and augment it; Section 2.4 introduces our proposed method in details; Section 2.5 presents the experimental results and a details discussion about it is presented; Section 2.6 provides an ablation study of the adopted technique. finally, Section 2.7 summarizes the conclusions of our chapter. In Chapter 3, we propose a novel multi-view based network for 3D point cloud completion. The organization of this chapter is as follows: In Section 3.1 we address the topic by concentrating on the explanation of the problem and the purpose of the proposed approach. Section

DOI:10.15774/PPKE.ITK.2023.007

3.2 provides a summary of prior work on state-of-the-art 3D shape completion methods, and multi-view based approaches; Section 3.3 demonstrates data generation technique; In Section 3.4, we introduce our proposed method in technical details; Section 3.5 displays the experimental results; Section 3.6 provides an ablation study of the proposed method, and Section 3.7 concludes the chapter with a summary of its study results. An overview and last thoughts are included in the Conclusions section of the thesis. Appendix A begins with an overview of Lidar technology, followed by some additional technical and implementation details about the deep learning models used in this thesis. while Appendix B summarizes the used abbreviations.

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Chapter 2

Deep Learning-Based Masonry Wall Image Analysis

This chapter introduces a novel deep learning-based fully automatic approach for the semantic analysis and documentation of masonry wall images, performing in parallel automatic detection and virtual completion of occluded or damaged wall regions, and brick segmentation leading to an accurate model of the wall structure. The network predicts the schematic mortar-brick pattern of the occluded areas based on the observed wall structure, providing in itself valuable structural information for archaeological and architectural applications. Moreover, it predicts the pixels' color values yielding a realistic visual experience for the observer. Furthermore, the network can be used for texture transfer: one can change the texture style of a given wall image, based on another wall image. For training and testing the network, a new manually annotated dataset for masonry wall analysis is also introduced in this chapter, which is used to evaluate the proposed approach, and to compare our solution to various reference techniques proposed for the same objectives.

DOI:10.15774/PPKE.ITK.2023.007

2.1 Introduction

Over the past years, we have observed a significant shift to digital technologies in Cultural Heritage (CH) preservation applications. The representation of CH sites and artifacts by 2D images, 3D point clouds or mesh models allows the specialists to use Machine learning (ML) and Deep learning (DL) algorithms to solve several relevant tasks at a high level in a short time. Application examples range from semantic segmentation of 3D point clouds for recognizing historical architectural elements [21], to the detection of historic mining pits [22], reassembling ancient decorated fragments [23], or classification of various segments of monuments [24].

Although many recent cultural heritage documentation and visualization techniques rely on 3D point cloud data, 2D image-based documentation has still a particular significance, due to the simplicity of data acquisition, and relatively low cost of large-scale dataset generation for deep learning-based methods. Image-based analysis has recently shown a great potential in studying architectural structures, ancient artifacts, and archaeological sites: we can find several approaches on the classification of biface artifacts, architectural features, and historical items [51, 52, 53], and also on digital object reconstruction of historical monuments and materials [54, 55].

2.1.1 Problem Statement

The maintenance of walls is a critical step for the preservation of buildings. Surveyors usually separate and categorize firstly the composing materials of walls either as main components (regular or irregular stones, bricks, concrete blocks, ashlars, etc.) or as joints (mortars, chalk, clay, etc.). In this chapter, for simpler discussion the term *brick* is used henceforward to describe the primary units of any type, and the term *mortar* is used to describe the joints. By investigating masonry walls of buildings, bricks are considered as the vital components of the masonry structures. Accurate detection and separation of bricks is a crucial initial step in various applications, such as stability analysis for civil engineering [58, 59], brick reconstruction [82], managing the damage in architectural buildings [56, 57], and in the fields of heritage restoration and maintenance [60].

DOI:10.15774/PPKE.ITK.2023.007

Various ancient heritage sites have experienced several conversions over time, and many of their archaeologically relevant parts have been ruined or occluded by various objects such as decorative elements, wall sculptures, and vegetation. Reassembling these damaged/occluded areas is an essential task for archaeologists studying and preserving these ancient monuments. Such structure estimation should be based on a deep analysis of the pattern of the visible wall segments.

2.1.2 Aim of the Chapter

To address the above challenges, we propose an end-to-end deep learning-based algorithm for masonry wall image analysis and virtual structure recovery. More specifically, given as input an image of a wall partially occluded by various irregular objects, our algorithm solves the following tasks in a fully automatic way (see also Figure 2.1):

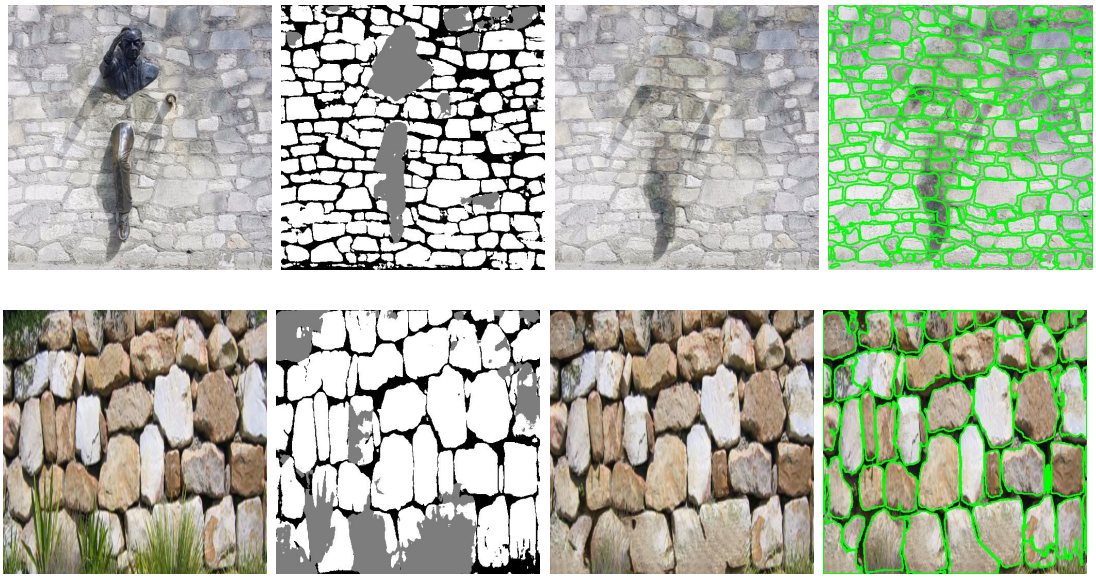
1. We detect the regions of the occluding objects, or other irregular wall components, such as holes, windows, damaged parts.

2. We predict the brick-mortar pattern and the wall color texture in the hidden regions, based on the observable global wall texture.

3. We extract the accurate brick contours both in the originally visible and in the artificially inpainted regions, leading to a strong structural representation of the wall.

First for the input wall images (Figure 2.1a) a U-Net [25] based algorithm is applied in a pre-processing step, whose output is a segmentation map with three classes, corresponding to brick, mortar, and irregular/occluded regions (see Figure 2.1b). This step is followed by an inpainting (completion) stage where two Generative Adversarial Networks (GANs) are adopted consecutively. The first GAN completes the missing/broken mortar segments, yielding a fully connected mortar structure. Thereafter, the second GAN estimates the RGB color values of all pixels corresponding to the wall (Figure 2.1c). Finally, we segment the inpainted wall image using the watershed algorithm, yielding the accurate outlines of the individual brick instances (Figure 2.1d).

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) U-Net output (c) Inpainted im. (d) Seg. output

Figure 2.1: Results of our proposed method on two selected images: (a) Input: wall image occluded by irregular objects (b) Result of preliminary brick-mortar-occluded region separation (c) Generated inpainted image (d) Final segmentation result for the inpainted image.

Furthermore, the model has been used to deal with style transfer and artificial coloring of schematic wall sketch maps. In particular, one can replace the coloring style (e.g. color and texture pattern of the bricks) of a wall image, with another wall's style, while maintaining the wall's original structural integrity. Given two images as inputs: a *content image* which is a color wall image or a binary image for the wall structure, and a *style image* which is a different wall image, the goal is to create a new image that incorporates both the structure of the *content image* and the texture style of the *style image* (see Fig. 2.18 and Fig. 2.19). For both use-cases, our proposed algorithms are freely available for testing on our laboratory's public website[1].

Due to the lack of existing masonry wall datasets usable for our purpose, we collected various images from modern and ancient walls and created our dataset in order to train the networks and validate our proposed approach.

[1]`http://imgproc.mplab.sztaki.hu/masonrydemo.`

DOI:10.15774/PPKE.ITK.2023.007

The motivation of the research work presented in this chapter was on one hand to develop a robust wall segmentation algorithm that can deal with different structural elements (stones, bricks ashlar, etc.), and accurately detecting various kinds of possible occlusion or damage effects (cement plaster, biofilms [81]). The output of our segmentation step can support civil engineers and archaeologists while studying the condition and stability of the wall during maintenance, documentation or estimation of the necessary degree of protection. On the other hand, our proposed inpainting step can support the planning of the reconstruction or renovation work (both in virtual or real applications), by predicting and visualizing the possible wall segments in damaged or invisible regions.

2.2 Related Work

Nowadays, there is an increasing need for efficient automated processing in cultural heritage applications, since manual documentation is time and resource consuming, and the accuracy is highly dependent on the experience of the experts. Developing an automatic method to analyze widely diverse masonry wall images is considered as a challenging task for various reasons [62]: issues should be addressed related to occlusions, lighting and shadows effects, we must expect a significant variety of possible shape, texture, and size parameters of the bricks and surrounding mortar regions, moreover, the texture can even vary within connected wall fragments, if it consists of multiple components built in different centuries from different materials .

In this section, we highlight three essential challenges linked to the discussed research topic, and provide a brief summary of recent image segmentation methods utilized for masonry wall delineation. Following that, we give an overview on the state-of-the-art in image inpainting techniques, focusing on their suitability for the task of completing the wall images. Finally, we discuss recent research on style transfer methods.

2.2.1 Masonry Wall Delineation

The literature presents several wall image delineation algorithms, that achieved good results for specific wall types. In an earlier study [67] adopt various pixel-

DOI:10.15774/PPKE.ITK.2023.007

based and object-oriented image processing technologies for detecting and characterizing the structural damage in historical buildings based on multi-spectral measurements. Oses et. al. [65] focus on the classification of built heritage masonry for determining the necessary degree of protection in different buildings, using an automatic image-based delineation method. Riveiro et. al. [61] present an automatic color-based algorithm for segmenting masonry structures, based on an improved marker-controlled Watershed. However, this later algorithm [61] purely focuses on the morphological analysis of quasi-periodic masonry walls, where the geometry of masonry courses follows horizontal rows, which condition does not hold very often - especially for ancient walls. Sithole et. al. [62] propose a semi-automatic segmentation algorithm to detect the bricks in masonry walls, working on *3D point cloud data* obtained by laser scanning. Although using such data sources becomes widespread in archeology and architecture nowadays, the 2D image based investigation addressed in this paper still has significance in particular for processing archive measurements, or in situations when long scanning surveys are not feasible. Since the method of [62] is based on the 3D triangulation of the 3D point cloud, reflectance, and RGB triplets, it cannot be suit to 2D data in a straightforward way. Similar issues appear by the work of Bosché et. al. [66], who introduce a method that simultaneously considers 3D information both in global and local levels for segmenting the walls into regions corresponding to bricks and mortar joints. Kajatin et. al. [63] suggested -based on a combination of machine learning techniques- a method that employs a weighted voting mechanism to combine the separate segmentation outputs of eight classifiers before executing a threshold operation to produce the final binary segmentation. Idjaton et. al. [64] presents a method for automatically doing stone-by-stone segmentation on large cultural heritage structures. On an image dataset of two Renaissance-style castles, two convolutional neural networks are compared with traditional edge detection or thresholding techniques.

While deep learning (DL) algorithms have achieved remarkable success in various computer vision applications (segmentation, classification, detections, etc.) in a wide range of fields from medical images [70], scene understandings [71] or autonomous driving [74], we only find a few references yet for application of DL methods in architecture or cultural heritage documentation. In addition, existing

DOI:10.15774/PPKE.ITK.2023.007

methods [68, 69] use deep learning rather for classification of the available images of the architectural heritage, instead of segmentation and feature extraction for detailed analysis.

2.2.2 Inpainting Algorithms

Image inpainting is a process where damaged, noisy, or missing parts of a picture are filled in to present a complete image. Existing image inpainting methods can be divided into two main groups called *non-blind* and *blind* techniques. In *non-blind* inpainting approaches the masks of the missing regions are given in advance to the algorithm, which requires cumbersome user intervention in complex scenes. In *blind* inpainting techniques the algorithm has to automatically define the pixels that are corrupted and have to be completed, i.e., the mask is also automatically extracted. The state-of-the-art blind inpainting algorithms [29, 32, 33] use strong assumptions regarding the shape and source of the occluded regions: they focus for example on the removal of scratches or texts which can be efficiently detected through low-level image features. However these techniques often fail in cases of multiple occluding objects with different shapes and textures appearing in various positions inside the image.

Our selected problem needs a significantly different blind inpainting approach from the existing ones: the occluded or damaged image regions may exhibit a high variety, however, we can assume that the *background*, which is a wall image, follows a regular pattern. In real-world wall images, the bricks and mortars form typical (but for each wall possibly different) texture patterns, and the occluded/damaged parts appear as *irregular* regions in the input images.

Diffusion-based methods [37, 38] and patch-based techniques [35, 36, 80] are traditional methods that use local image features or descriptors to inpaint the missing parts. However, such techniques lack the accuracy when the occluded parts are large, and they may produce miss-alignments for thin structures such as wall mortar (see Figure 2.2b–d, the results of a patch-based technique [80]).

Large datasets are needed to train the recently proposed deep learning models [27, 30, 26], which inpaint the images after training and learning a hidden feature representation. A generative multi-column convolutional neural network

DOI:10.15774/PPKE.ITK.2023.007

(a) Input 1 (b) Output 1 (c) Input 2 (d) Output 2

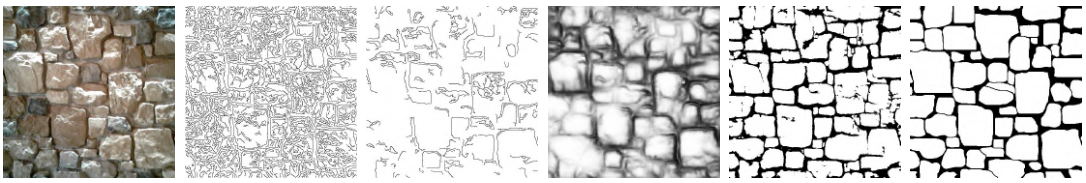
Figure 2.2: Patch-based technique [80] results for two wall samples, (**a**,**c**) inputs photos with holes that should be inpainted, (**b**,**d**) the results, we can see the miss-alignments problem of the mortar lines.

(GMCNN) model [76] was proposed with three sub-networks, namely the generator, the global and local discriminator, and a pre-trained VGG model [44]. This approach deals in parallel with global texture properties and local details by using a confidence driven reconstruction loss and diversified Markov random field regularization, respectively. A multi-output approach [31] is introduced to generate multiple image inpainting solutions with two GAN-based [39] parallel paths: the first path is reconstructive aiming at inpainting the image according to the Ground Truth map, while the other path is generative, which generates multiple solutions.

The EdgeConnect [28] approach uses a two-stage adversarial network that completes the missing regions according to an edge map generated in the first stage. Initial edge information should be provided to this algorithm which is based on either the Canny detector [46] or the Holistically-Nested Edge Detection algorithm (HED) [45]. However, if we apply this method for the masonry wall inpainting application, non of these two edges generators [46, 45] can provide semantic structural information about the walls, since their performance is sensitive to the internal edges inside the brick or mortar regions, creating several false edges which are not related to the pattern of the wall structure. Moreover, the outputs of both edge generators [46, 45] strongly depend on some threshold values, which should be adaptively adjusted over a wall image. To demonstrate these limitations, we show the outputs of the Canny edge detector (applied with two different threshold values), and the HED algorithm in Figure 2.3b–d for a

DOI:10.15774/PPKE.ITK.2023.007

selected masonry wall sample image. As explained later in details, one of the key components of our proposed method will be using an efficient, U-Net based delineation algorithm as a preprocessing step, whose output will approximate the manually segmented binary brick-mortar mask of the wall (see Figure 2.3e,f).



(a) Input Image | (b) Canny $\sigma = 0.1$ | (c) Canny $\sigma = 0.4$ | (d) HED Output | (e) Ours Output | (f) Expected Output

Figure 2.3: Wall delineation: comparison of state-of-the-art edge detectors to our proposed U-Net based approach, and to the Ground Truth (GT).

2.2.3 Style Transfer Algorithms

Style transfer approaches combine two images (a *content image* and a *style image*) so that the resulting output image keeps the content image's fundamental elements while appearing to be *painted* in the style of the reference *style image*. Li and Wand proposed a patch-based non-parametric Markov Random Field (MRF) approach [83], which has been proved to be effective in preserving coherent textures in complex images, however it worked less efficiently with non-textured styles. [84] combines independent network components to learn the corresponding content and style information, and uses a *StyleBank* layer which comprises mid-level convolutional filters to individually learn different styles, where style is coupled to a set of parameters. This algorithm provides the advantage of learning a new style as well as a flexible control over style fusion. However, it has a number of weaknesses, including the lack of details in the result images. Li et al. [85] attempt to use a sequence of feature transformations to transfer arbitrary styles without the need for style learning. However, when the used feature vector has a large dimension, the expensive matrix calculation in whitening and colouring modifications becomes a limitation. In contrast to the existing state-of-the-art algorithms, our solution uses a preliminary phase to extract the *content wall im-*

DOI:10.15774/PPKE.ITK.2023.007

age's dominant structure, then we add the target wall image style to the model in a direct manner, as it will be detailed later in Sec. 2.4.4.

2.3 Dataset Generation

We have constructed a new manually annotated dataset for training and evaluating the proposed algorithm for the aforementioned problems. The dataset contains images of facades and masonry walls, either from historical or contemporary buildings. As demonstrated in Figure 2.4, several different types of wall structures are included, for example three types of rubble masonry (Random, Square, Dry), and two types of ashlar (Fine, Rough). Even within a given class, the different samples may show highly variable characteristics, both regarding the parameters of the bricks (shape, size and color), and the mortar regions (thin, thick, or completely missing). In addition, the images are taken from different viewpoints in different lighting conditions, and contain shadow effects in some cases.

Using a large dataset plays an essential role to achieve efficient performance and generalization ability of deep learning methods. On the other hand, dataset generation is in our case particularly costly. First, for both training and quantitative evaluation of the brick-mortar segmentation step, we need to manually prepare a binary delineation mask (denoted henceforward by $\mathrm{I}_{\mathrm{wall\_ftr}}$) for each wall image of the dataset, where as shown in the bottom row of Figure 2.4, background (black pixels) represents the regions of mortar, and foreground (white pixels) represents the bricks. In addition, for training and numerical quality investigation of the inpainting step, we also need to be aware of the accurate occlusion mask, and we have to know the original wall pattern, which could be observable without the occluding object. However, such sort of information is not available, when working with natural images with real occlusions or damaged wall parts, as shown in Figure 2.1.

For ensuring in parallel the efficiency and generality of the proposed approach, and the tractable cost of the training data generation process, we have used a combination of real and semi-synthetic images, and we took the advantages of various sorts of data augmentation techniques.

DOI:10.15774/PPKE.ITK.2023.007

Our dataset is based on 532 different wall images of size 512×512, which are divided among the training set (310 images), and three test sets (222 images).

The training dataset is based on 310 occlusion-free wall images, where for each image we also prepared a manually segmented I_{wall_ftr} binary brick-mortar mask (see examples in Figure 2.4). For training, the occlusions are simulated using synthetic objects during data augmentation, so that the augmentation process consists of two phases:

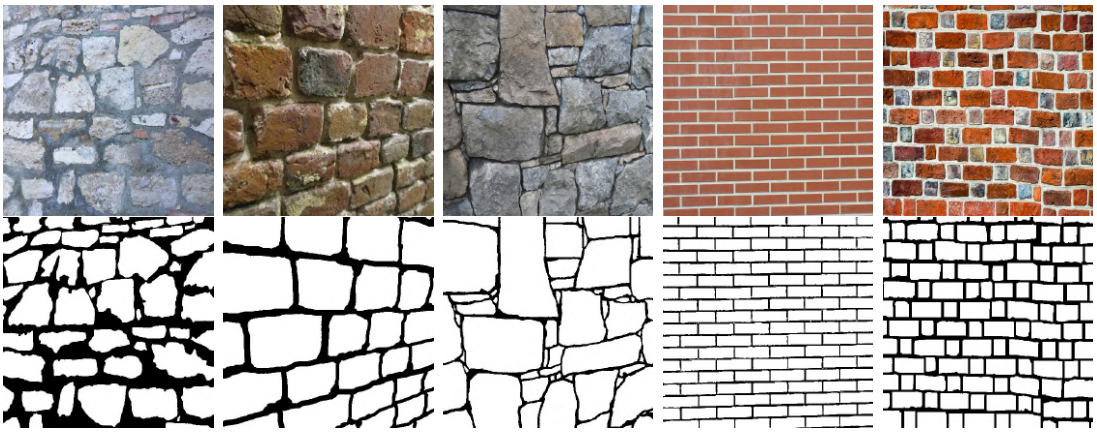
(i) Offline augmentation: 7 modified images are created for each base image using the following transformations: (a) horizontal flip, (b) vertical flip, (c) both vertical and horizontal flips, (d) adding Gaussian noise (e) randomly increasing the average brightness, (f) randomly decreasing the average brightness, and (g) adding random shadows.

(ii) Online synthetic occlusion and data augmentation [41]: each image produced in the previous step is augmented online (during the training) by occluding the wall image I_{wall} with a random number (from 1 and 8) of objects (represented by I_{oclud}), chosen from a set of 2490 non_human samples of the Pascal VOC dataset [42]. With these parameter settings, we generated both moderately, and heavily occluded data samples. As demonstrated in Figure 2.5 (a,b) the result of this synthetic occlusion process is the I_{in} image, which is used as the input of the proposed approach. We also generate a target (Ground Truth) image for training and testing the image segmentation step, so that we combine the binary delineation map (I_{wall_ftr}) and the mask of the chosen occluding objects in a three-class image $I_{wall\_ftr\_oclud}$ (see Figure 2.5b), which has the labels mortar (black), brick (white), and occlusion (gray) . Afterward, we used the Imgaug library [43] to apply Gaussian noise, cropping/padding, perspective transformations, contrast changes, and affine transformations for each image.

Our test image collection is composed of three subsets:

- Test Set (1): We used 123 occlusion-free wall images and simulated occlusions by synthetic objects, as detailed later in Section 2.5. We only remark that here both the wall images and the synthetic objects were different

DOI:10.15774/PPKE.ITK.2023.007



(a) Random (b) Square (c) Dry (d) Fine (e) Rough

Figure 2.4: Sample base images from our dataset (**top row**), and the Ground Truth brick-mortar delineation maps (bottom row). Wall types: (**a**) Random rubble masonry, (**b**) Square rubble masonry, (**c**) Dry rubble masonry, (**d**) Ashlar fine, (**e**) Ashlar rough.

from the ones used to augment the training samples. A few synthesized test examples are shown in the first rows of Figures 2.8 and 2.9.

- Test Set (2): We took 47 additional wall images with real occlusions and with available Ground Truth (GT) data in ancient sites of Budapest, Hungary. In these experiments, we mounted our camera to a fixed tripod, and captured image pairs of walls from the same positions with and without the presence of occluding objects (see the second rows in Figures 2.8 and 2.9).

- Test Set (3): For demonstrating the usability of the proposed techniques in real-life, we used 52 wall pictures:

  – 25 photos of various walls with real occlusions or damaged segments are downloaded from the web (see examples in Figure 2.1).

  – 27 masonry wall images from ancient archaeological sites in Syria are used to evaluate the inpainting and the segmentation steps of our proposed algorithm.

Note that since Test Set (1) and Test Set (2) contain both occluded and occlusion free images from the same site, they can be used in numerical evaluation of the proposed algorithm. On the other hand, Test Set (3) can be only used in

DOI:10.15774/PPKE.ITK.2023.007

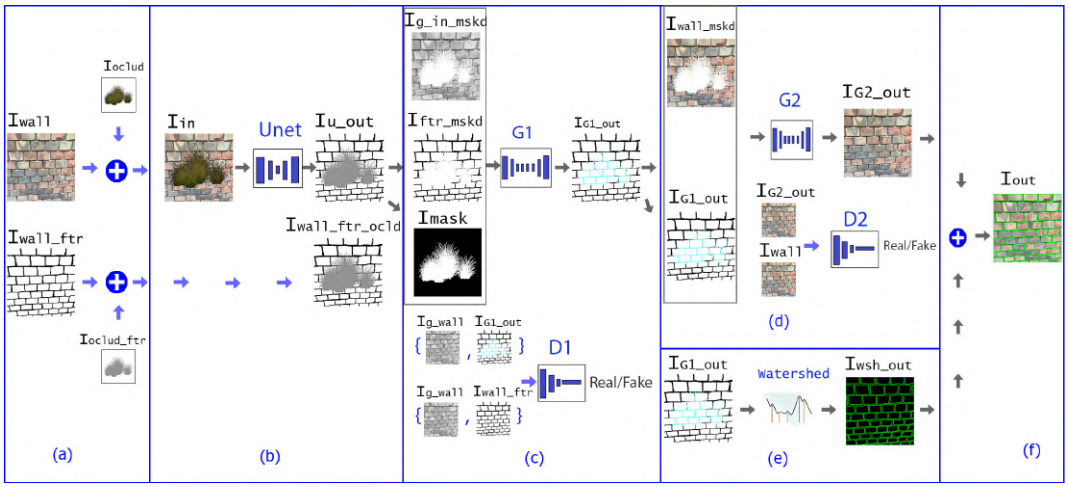


Figure 2.5: Dataflow of our algorithm. (**a**) Data augmentation by adding synthetic occlusion (**b**) Pre-processing Stage (U-Net network): the input I_{in} contains a wall image with occluding objects, the output I_{u_out} is expected to be similar to the target $I_{wall\_ftr\_oclud}$ (the delineation map of the wall occluded by various objects). (**c**) Inpainting Stage - Hidden Feature Generator $G1$: the input consist of the masked image of I_{in}, the masked extracted delineation map, and the mask of the occluding objects. The output is the predicted delineation map of the complete wall structure. (**d**) Inpainting Stage - Image Completion $G2$: the inputs are the color image of the wall, the occlusion mask and the predicted delineation map; the output is the inpainted image. (**e**) Segmentation Stage (Watershed Transform): the input is the predicted delineation map, the output is the bricks' instance level segmentation map. (**f**) Brick segmentation map superimposed to the inpainted image output (I_{out}).

qualitative surveys, however these images may demonstrate well the strength of the technique in real applications.

The wall images and the manually annotated masks used in the dataset are available at the website of the authors (`http://mplab.sztaki.hu/geocomp/masonryWallAnalysis`).

2.4 Proposed Approach

The main goals of the proposed method are (i) to automatically extract the individual brick instances of masonry wall images, (ii) to detect occluded or damaged wall segments, and (iii) to fill in the occluded/missing regions by a realistic prediction of the brick-mortar pattern.

DOI:10.15774/PPKE.ITK.2023.007

The complete workflow is demonstrated in Figure 2.5. We assume that the input of the algorithm is a masonry wall image I_{in} which is partially occluded by one or several foreground objects. Our method provides multiple outputs: a color image of the reconstructed wall I_{G2_out}, the binary brick-mortar delination map I_{wsh_out} of I_{G1_out}, and the contours of the individual bricks. These outputs can be superimposed yielding a segmented color image I_{out} of Figure 2.5f.

The proposed approach is composed of three main stages as shown in Figure 2.5:

1. Pre-processing stage (Figure 2.5b): taking an input image I_{in} two outputs are extracted: the delineation map of the observed visible bricks, and the mask of the occluding objects or damaged wall regions.

2. Inpainting: this stage has two steps, the first one completes the delineation map in the occluded image parts (Figure 2.5c), and the second one predicts RGB color values in these regions (Figure 2.5d).

3. Segmentation (Figure 2.5e): this stage aims to separate the mortar from the brick regions in the inpainted wall image, and extract the accurate brick contours.

The inpainting stage has been modified to fit two wall images and the objective here is to transform the style form an image to another. In the next sections, we discuss the main three stages of the proposed method and the style transfer network.

2.4.1 Pre-Processing Stage

The goal of this step is twofold: extracting the delineation structure by separation of the bricks from the mortar regions, and detecting the masks of possibly occluded wall parts which should be inpainted in the consecutive steps.

As shown in Figure 2.5b, the segmentation step is realized by a U-Net deep neural network [25], which consists of two main parts. The encoder part encodes the input image I_{in} into a compact feature representation, and the decoder part decodes these features into an image I_{u_out} that represents the predicted classes. In

DOI:10.15774/PPKE.ITK.2023.007

our implementation, a pre-trained Resnet50 [9] encoder is adopted in the encoder step, and a three-class image $\mathrm{I}_{\mathrm{wall\_ftr\_oclud}}$ is used as the target of this network, whose labels represent brick (white), mortar (black) and occluded (gray) regions. From the $\mathrm{I}_{\mathrm{u\_out}}$ output, we derive two binary auxiliary images: $\mathrm{I}_{\mathrm{mask}}$ is the binary mask of the occluded pixels according to the prediction; and $\mathrm{I}_{\mathrm{ftr\_mskd}}$ is a mask of the predicted bricks in the observed wall regions, where both the occluded and the brick pixels receive white labels, while the mortar pixels are black.

The network is trained using the Adam optimizer [40] over a joint loss ℓ_{u_net} that consists of a cross-entropy loss and a feature-matching loss term. The cross-entropy loss ℓ_{Cro} is applied to measure the difference between the probability distributions of the classes of the U-Net output $\mathrm{I}_{\mathrm{u\_out}}$ and the target $\mathrm{I}_{\mathrm{wall\_ftr\_oclud}}$, while the feature-matching loss ℓ_{ftr_mat} contributes to the training of the Inpainting Stage as described in Section 2.4.2.1:

$$\ell_{u\_net} = \lambda_1^1 \, \ell_{Cro} + \lambda_2^1 \, \ell_{ftr\_mat}, \tag{2.1}$$

where λ_1^1 and λ_2^1 are regularization parameters which are modified during the training process.

During the training phase, the $F1_{u\_net}$ value is calculated as the average of the F_1-scores for the three classes (Brick, mortar and the occluded objects), which is used as a confidence number for modifying the weights in the subsequent network component of the Inpainting Stage.In our experiment, we choose for the first epochs $\lambda_1^1 = 1$, $\lambda_2^1 = 0$, and for the last epochs $\lambda_1^1 = 1$, $\lambda_2^1 = 0.1$.

2.4.2 Inpainting Stage

This stage is composed of two main sub-steps. First, the Hidden Feature Generator completes the mortar pixels in the area covered by the white regions of the predicted $\mathrm{I}_{\mathrm{mask}}$ image, based on the mortar pattern observed in the $\mathrm{I}_{\mathrm{ftr\_mskd}}$ mask. Second, the Image Completion step predicts the RGB color values of the pixels marked as occluded in the $\mathrm{I}_{\mathrm{mask}}$ image, depending on the color information of the $\mathrm{I}_{\mathrm{wall\_mskd}}$ image restricted to the non-occluded regions, and the structural wall features extracted from the $\mathrm{I}_{\mathrm{G1\_out}}$ map within the occluded regions.

The inpainting stage is based on two different generative adversarial networks (GAN), where each one consists of generator/discriminator components. The

DOI:10.15774/PPKE.ITK.2023.007

generator architecture follows the model proposed by [49], which achieved impressive results in image-to-image translation [34] and in image inpainting [28]. The generator is comprised of encoders which down-sample the inputs twice, followed by residual blocks and decoders that upsample the results back to the original size. The discriminator architecture follows the 70×70 PatchGAN [34], which determines whether overlapping image patches of size 70×70 are real or not. In Figure 2.5, G_1 and D_1 represent the Generator and Discriminator of the Hidden Feature Generator stage, and G_2 and D_2 represent the Generator and Discriminator of the Image Completion stage.

2.4.2.1 Hidden Feature Generator

For creating the inputs of the first generator (G_1), three images are used: the two output images of the pre-processing stage ($\mathrm{I_{mask}}$, $\mathrm{I_{ftr\_mskd}}$), and $\mathrm{I_{g\_in\_mskd}}$, which is the masked grayscale version of the input image. The output of G_1, $\mathrm{I_{G1\_out}}$ represents the predicted wall structure, completing $\mathrm{I_{ftr\_mskd}}$ with brick outlines under the occlusion mask regions. The first discriminator D_1 predicts whether the predicted wall features are real or not, based on the inputs $\mathrm{I_{G1\_out}}$ and $\mathrm{I_{wall\_ftr}}$, both ones conditioned on the grayscale wall image $\mathrm{I_{g\_wall}}$.

The network is trained by the adversarial loss and the feature-matching loss, which are both weighted by the $F1_{u\_net}$-score that reveals the accuracy of the pre-processing stage:

$$\min_{G1} \max_{D1} \ell_{G1} = \min_{G1} \big(F1_{u\_net} (\lambda_1^2 \max_{D1} \ell_{adv_1} + \lambda_2^2 \, \ell_{ftr\_mat}) \big). \tag{2.2}$$

The adversarial loss ℓ_{adv_1} is formulated as a two-player zero-sum game between the Generator network G_1 and the Discriminator network D_1 where the generator attempts to minimize the Eq. (3.3), while the discriminator tries to maximize it:

$$\begin{aligned} \ell_{adv_1} = \mathbb{E}_{(\mathrm{I_{wall\_ftr}}, \mathrm{I_{g\_wall}})} \big[log\big(D_1(I_{\mathrm{wall\_ftr}}, I_{\mathrm{g\_wall}})\big)\big] \\ + \mathbb{E}_{(\mathrm{I_{G1\_out}}, \mathrm{I_{g\_wall}})} \big[log\big(1 - D_1(I_{\mathrm{G1\_out}}, I_{\mathrm{g\_wall}})\big)\big] \end{aligned} \tag{2.3}$$

The feature matching loss ℓ_{ftr_mat} [28] is used to make the training process stable by forcing the generator to produce results with representations as close as possible to the real images. To increase the stability of the training stage, the Spectral

DOI:10.15774/PPKE.ITK.2023.007

Normalization (SN) [47] is applied for both the generator and the discriminator, scaling down the weight matrices by their respective largest singular values. In our experiment, we choose: $\lambda_1^2 = 1$, $\lambda_2^2 = 10$.

2.4.2.2 Image Completion

The input of the second generator, G_2 combines both the masked wall image ($\mathrm{I}_{\mathrm{wall\_mskd}}$) and the output of the first generator ($\mathrm{I}_{\mathrm{G1\_out}}$). The aim of the G_2 generator is creating an output image $\mathrm{I}_{\mathrm{G2\_out}}$ which is filled with predicted color information in the masked regions (see Figure 2.5d). The inputs of the discriminator D_2 are $\mathrm{I}_{\mathrm{G2\_out}}$ and $\mathrm{I}_{\mathrm{wall}}$ and its goal is to predict whether the color intensity is true or not. The network is trained over a joint loss that consists of L_1 loss, adversarial loss, perceptual loss, and style loss.

The adversarial loss ℓ_{adv2} has the same function as described in Section 2.4.2.1. The L_1 loss ℓ_{L_1} is applied to measure the distance between the algorithm output $\mathrm{I}_{\mathrm{out}}$ and the target $\mathrm{I}_{\mathrm{wall}}$, which is applied in the non-masked area ℓ_{L_1} and in the masked area $\ell_{L_1 hole}$, respectively. While the perceptual loss ℓ_{perc} [28, 49] is applied to make the results perceptually more similar to the input images, the style loss ℓ_{sty} [28, 48] is used to transfer the input style onto the output. The TV loss ℓ_{TV} is used to smooth the output in the border areas of the mask. Each loss is weighted by a regularization parameter as described by the following equation:

$$\ell_{G2} = \lambda_1^3 \, \ell_{L_1} + \lambda_2^3 \, \ell_{L_1 hole} + \lambda_3^3 \, \ell_{adv2} + \lambda_4^3 \, \ell_{perc} + \lambda_5^3 \, \ell_{sty} + \lambda_6^3 \, \ell_{TV}. \tag{2.4}$$

In our experiment, we choose: $\lambda_1^3 = 1$, $\lambda_2^3 = 5$, $\lambda_3^3 = 0.1$, $\lambda_4^3 = 0.1$, $\lambda_5^3 = 250$, $\lambda_6^3 = 0.1/S$, where S refers to the image size (here $S = 256 \times 256$), the first fifth parameters were chosen based on [28], and the last one parameters were chosen to balance the impact of each loss.Appendix A includes more information on the model's architecture, loss functions, and training strategy.

2.4.3 Segmentation Stage

The last stage of the proposed algorithm is responsible for extracting the accurate outlines of the brick instances, relying on the previously obtained Hidden Feature Generator delineation map $\mathrm{I}_{\mathrm{G1\_out}}$. Although the delineation maps are quite

DOI:10.15774/PPKE.ITK.2023.007

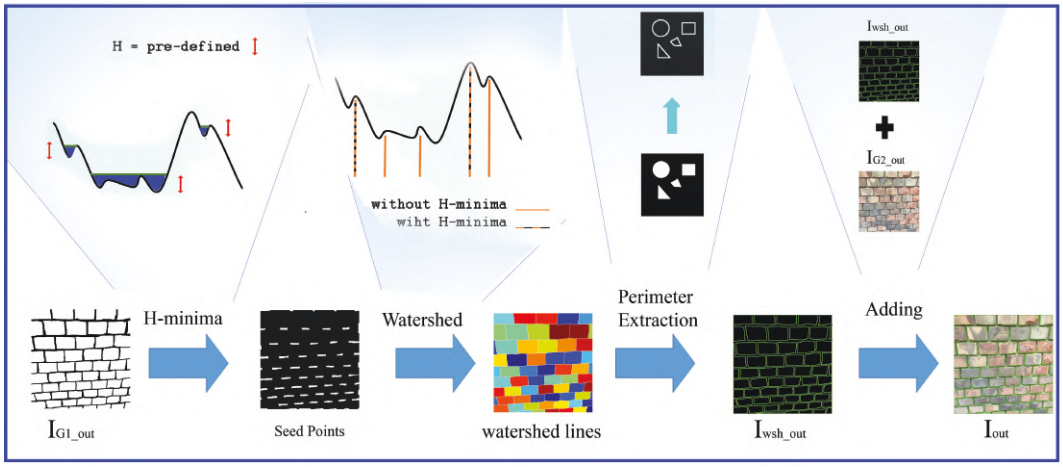


Figure 2.6: Dataflow of the Segmentation Stage, which extracts the accurate brick contours on the inpainted wall images.

reliable, they might be noisy near to the brick boundaries, and we can observe that some bricks are contacting, making simple connected component analysis (CCA) based separation prone to errors. To overcome these problems, we apply a marker based watershed [75] segmentation for robust brick separation.

The dataflow of the segmentation stage is described at Figure 2.6, where first, we calculate the inverse distance transform (IDT) map of the obtained delineation map (I_{G1_out}). Our aim is to extract a single compact seed region within each brick instance, which can be used as internal marker for the watershed algorithm. Since the IDT map may have several false local minima, we apply the H-minima transform [73], which suppresses all minima under a given H-value. Then, we apply flooding from the obtained H-minima regions, so that we consider the inverse of I_{G1_out} (i.e., all mortar or non-wall pixels) as an external marker map, whose pixels cannot be assigned to any bricks. Finally the obtained brick contours can be displayed over the Image Completion output (see I_{out}).

2.4.4 Style Transfer

The goal of the style transfer component is to fill or modify the wall's texture style based on an image of another wall or wall segment. The workflow of this procedure is presented in Figure 2.7. The style of the input image I_{in} is changed

DOI:10.15774/PPKE.ITK.2023.007

to match the style of another wall image sample called the *style image* $\mathrm{I}_{\mathrm{style}}$. Archaeologists can use this algorithm to modify some degraded segments of the studied wall that have become soiled or lost its original color over time due to environmental factors, relying in various style features extracted from the intact wall regions.

The proposed style transfer procedure uses the same G_1 and G_2 network components, that were previously introduced in the wall analysis section (Sec. 2.4.2). Here the *image completion* network G_2 has two inputs: the first one is the G_1 generator's output $\mathrm{I}_{\mathrm{G1\_out}}$, which is a brick-mortar separation map that includes the predicted complete wall structure for the processed wall image I_{in} (including both the originally visible and occluded wall components). The second input is the style image $\mathrm{I}_{\mathrm{style}}$, which is represented by an occlusion-free colored wall image with a different texture style. The network extracts a representation for the texture style of $\mathrm{I}_{\mathrm{style}}$ and transfers it to the predicted brick-mortar separation map $\mathrm{I}_{\mathrm{G1\_out}}$, resulting in a new image $\mathrm{I}_{\mathrm{G2\_out}}$, that is a color image reflecting the structure of I_{in} and the style of $\mathrm{I}_{\mathrm{style}}$.

Note that apart from changing the style of existing wall images (see Figure 2.18), we can also directly feed in a binary brick-mortar map to the network for applications where we initially have a binary sketch of the wall only, and intend to paint it using the style from another wall image (see Figure 2.19).

To use the same trained network as in the previous wall analysis application and to avoid the need for any additional training for the G_2 generator, which is originally trained to paint exclusively the *occluded* (i.e. masked) regions of its input image (see Figure 2.5), we add here virtual masks to the style images. The joint usage of two binary virtual masks - which are inverted variants of each other - ensures that the entire target image will be filled with the texture of the style image, so that the style image is fed twice to the network, each time with one of the masks.

2.5 Experimental Results and Evaluation

We evaluated the proposed algorithm step by step, both via quantitative tests and an extensive qualitative survey, using the test data collection introduced in

DOI:10.15774/PPKE.ITK.2023.007

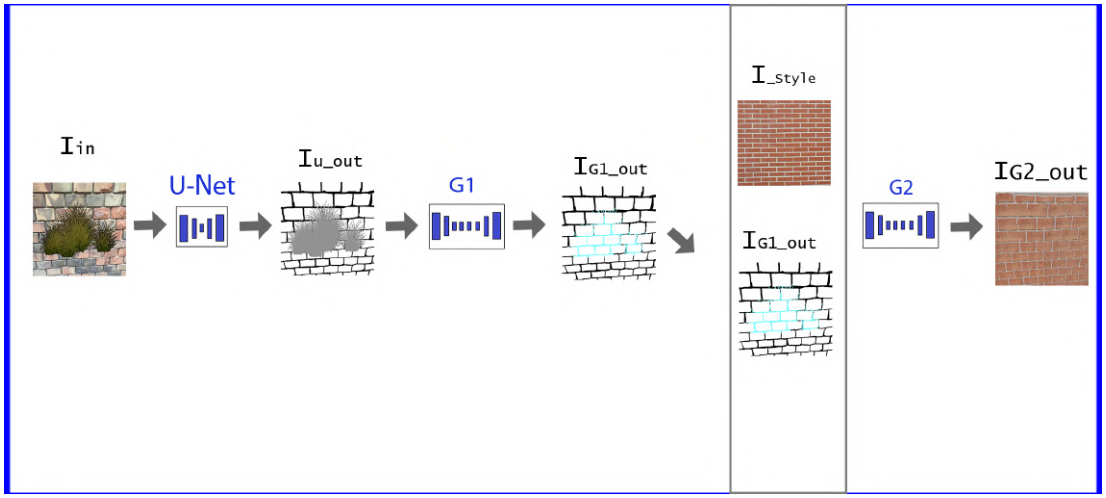


Figure 2.7: Dataflow of the Style Transfer application

Section 2.3. Our test data collection consists of three test sets, where the first two sets can be used in quantitative evaluation.

As mentioned in Section 2.3, test Set (1) was constructed by simulating occlusions on 123 different (occlusion-free) wall images, so that neither the wall images nor the occluding objects have been used during the training stage. More specifically, as occluding objects, we used a set of 545 different person images from the Pascal VOC dataset [42], and by random scaling we ensured that each one of the generated synthetic objects covers from 2% to 10% of the area of the wall image (see examples in the first rows of Figures 2.8 and 2.9). For each base wall image of this test subset, we generated five test images (I_{in}), so that the number of the occluding objects varies from 1 to 5 in a given test image. As a result, the augmented image set of Test Set (1) was divided into five groups depending on the number of the occluding objects.

Test Set (2) contains image pairs of walls taken from the same camera position, where the first image contains real occluding objects, while the second image shows only the pure wall without these objects (see the second rows in Figures 2.8 and 2.9). In this way the second image can be considered as a Ground Truth for the inpainting technique. We recall here that the process for capturing these images was discussed in Section 2.3.

DOI:10.15774/PPKE.ITK.2023.007

In the following sections, we present evaluation results for each stage of the algorithm.

2.5.1 Occlusion Detection in the Pre-Processing Stage

The pre-processing stage classifies the pixels of the input image into three classes: brick, mortar, and regions of occluding object. Since brick-mortar separation is also part of the final segmentation stage, we will detail that issue later during the discussion of Section 2.5.3, and here we only focus on the accuracy of occlusion mask extraction.

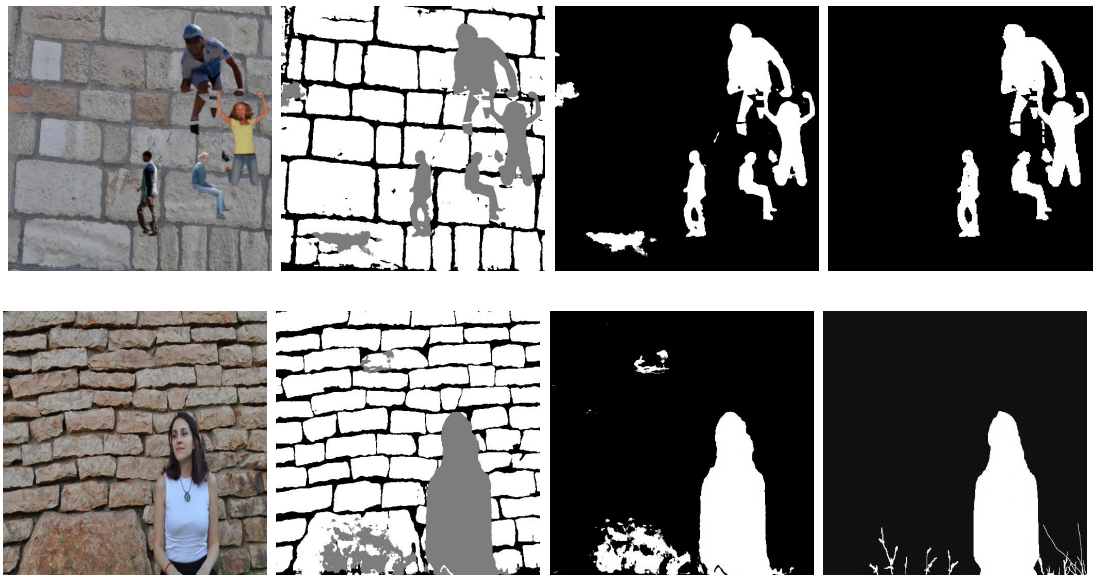
Both for Test Set (1) and Test Set (2), we compare the occlusion mask estimation provided by the U-Net (i.e., gray pixels in $\mathrm{I}_{\mathrm{u\_out}}$) to the corresponding Ground Truth (GT) mask derived from the reference map $\mathrm{I}_{\mathrm{wall\_ftr\_oclud}}$, as demonstrated in Figure 2.5b. Note that for images from Test Set (1), this mask is automatically derived during synthetic data augmentation in the same manner as described for training data generation in Section 2.3 (ii). On the other hand, for Test Set (2) the GT masks of the occluding (real) objects are manually segmented.

Taking the GT masks as reference, we calculate the pixel-level Precision (Pr) and Recall (Rc) values for the detected occlusion masks, and derive the F_1-score as the harmonic mean of Pr and Rc.

The results are reported in Table 2.1, where for Test Set (1) (Synthetic) we also present separately the results within the five sub-groups depending on the number of occluding objects. The last row (Real) corresponds to the results in Test Set (2). For qualitative demonstration of this experiment, in Figure 2.8 we can compare the silhouette masks of the occluding objects extracted from the U-Net network output, and the Ground Truth occlusion mask (see a sample image from the Synthetic Test Set (1) in the first row, and one from the Real Test Set (2) in the second row).

The numerical results in Table 2.1 confirm the efficiency of our algorithm in detecting the occluded regions of the images. For all considered configurations, the recall rates of the occluded areas surpass 83%, ensuring that the real outlier regions (which should be inpainted) are found with a high confidence rate. Since the network also extracts some false occlusions, the measured F_1-scores are

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) U-Net Output (c) Extracted Mask (d) GT

Figure 2.8: Pre-processing Stage results: (**a**) Input: wall image occluded by irregular objects (first row: synthetic occluding objects, second row: real occluding objects), (**b**) U-Net output with brick, mortar and occluded classes (**c**) automatically estimated occlusion masks, (**d**) Ground Truth (GT).

Table 2.1: Evaluation of occlusion mask extraction in the Pre-processing Stage for Test Set (1) and (2).

Test set	**num. of obj.**	**Precision**	**Recall**	F_1-score
(1) - Synthetic	**one**	**52.84**	**83.90**	**58.99**
	two	**70.15**	**88.38**	**75.06**
	three	**77.36**	**89.58**	**81.49**
	four	**80.51**	**89.48**	**83.38**
	five	**83.93**	**89.71**	**85.88**
	average	**72.95**	**88.22**	**76.96**
(2) - Real	**-**	**78.32**	**92.87**	**83.63**

DOI:10.15774/PPKE.ITK.2023.007

between 58% and 85%. However, the observed relatively lower precision does not cause critical problems in our application, since false positives only implicate some unnecessarily inpainted image regions, but during the qualitative tests of the upcoming inpainting step, we usually did not notice visual errors in these regions (see the discussion in Section 2.5.4).

We can also notice from Table 2.1, that by increasing the number of occluding objects, the precision (thus also the F_1-score) increases due to less false alarms. Based on further qualitative analysis (see also the discussion in Section 2.5.4) we can confirm, that the U-Net network can detect the occluding objects with high accuracy as nearly all of the virtually added irregular objects or the real occluding objects have been correctly detected.

2.5.2 Quantitative Evaluation of the Inpainting Stage

In this stage, two output maps are extracted sequentially, the hidden feature generator output, and the image completion output, each will be addressed individually in the following sections:

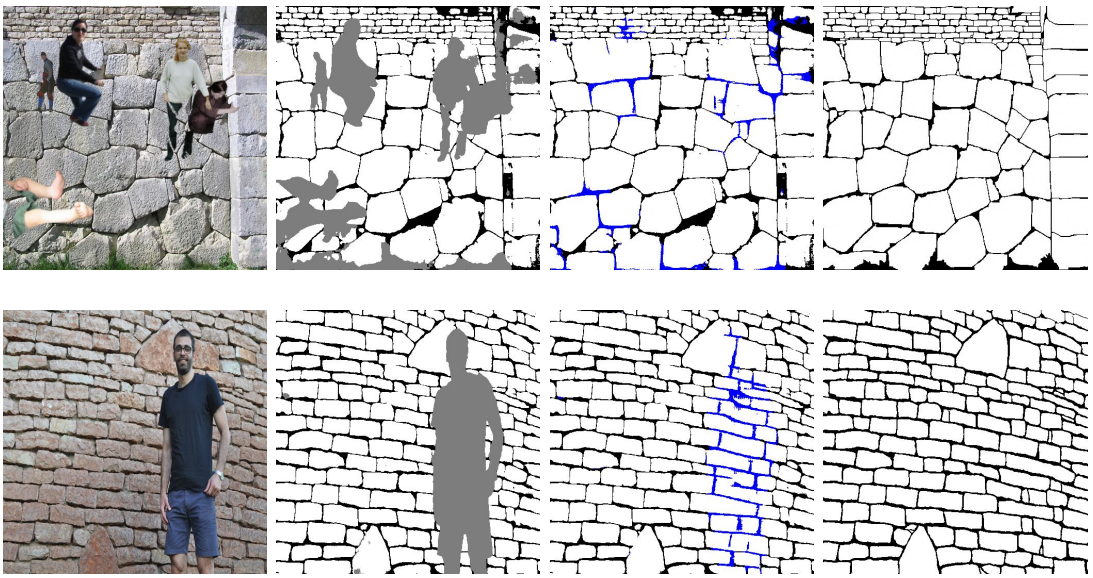
2.5.2.1 Hidden Feature Generator Output

The first result of the *Inpainting Stage* is the G_1 output which is a binary classification mask showing the prediction of the mortar and the brick regions in the occluded areas. Figure 2.9c shows the output of G_1 where the blue pixels indicate the predicted mortar pixels in the masked region, while the Ground Truth mortar regions are provided as a reference in Figure 2.9d. Since the result of this step is a structure delineation mask, which directly affects the brick segmentation stage, corresponding numerical results will be presented in Section 2.5.3.

2.5.2.2 Image Completion Output

The second result of the *Inpainting Stage* is the G_2 output, which is an inpainted color image $\mathrm{I}_{\mathrm{G2\_out}}$ that should be compared to the original occlusion-free wall image $\mathrm{I}_{\mathrm{wall}}$. For evaluating this step, we compared our method to state-of-the-art inpainting algorithms. Since (as mentioned in Section 2.2.2) existing blind image inpainting algorithms have significantly different purposes from our approach

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) U-Net Output (c) G_1 Output (d) GT

Figure 2.9: Structure inpainting results (first *GAN*): (**a**) Input: wall image occluded by irregular objects (first row: synthetic occluding objects, second row: real occluding objects), (**b**) U-Net output, (**c**) G_1 output: inpainted mortar-brick map (predicted mortar pixels in blue), (**d**) Ground Truth (GT).

DOI:10.15774/PPKE.ITK.2023.007

(such as text or scratch removal), we found a direct comparison with them less relevant. Therefore, in this specific comparative experiment, we used our approach in a non-blind manner and compared it to three non-blind algorithms: the Generative Multi-column Convolutional Neural Network (GMCNN) [76], the Pluralistic Image Completion [31], and the Canny based EdgeConnect [28].

In the training stage, for a fair comparison, an offline augmentation step was used in order to train all algorithms on the same dataset. We used the Augmentor library [50] for randomly rotating the images up to $\pm 25^{\circ}$, and randomly applying horizontal and vertical flipping transforms, zooming up to 15% of the image size, and changing the contrast of the images, yielding altogether 10,000 augmented images. Moreover, during the training phase, for simulating the missing pixels (i.e., holes or occluded regions that should be inpainted in the images), we used the same masks for all inpainting algorithms. These masks are derived from two different sources: (i) Irregular external masks from a publicly available inpainting mask set [27] which contains masks with and without holes close to the image borders, with a varying hole-to-image area ratios from 0% to 60%. (ii) Regular square shaped masks of fixed sizes (25% of total image pixels) centered at a random location within the image.

In the evaluation stage, randomly chosen external masks were adopted from the mask set proposed by [27]. More specifically, we chose 5 subsets of masks according to their hole-to-image area ratios: ((0%, 10%], (10%, 20%], (20%, 30%], (30%, 40%], (40%, 50%]). Thereafter, we overlaid occlusion-free wall images from the Test Set (1) with the selected/generated masks, to obtain the input images of all inpainting algorithms during this comparison (see also Figure 2.10, first row).

As mentioned in [27], there is no straightforward numerical metric to evaluate the image inpainting results due to the existence of many possible solutions. Nevertheless, we calculated the same evaluation metrics as usually applied in the literature [27, 28]:

(1) Peak Signal-to-Noise Ratio (PSNR).

(2) Structural Similarity Index (SSIM) [78] with a window size of 11.

(3) Relative L_1 error.

DOI:10.15774/PPKE.ITK.2023.007

(4) FID (Fechet Inception Distance) Score [77]: shows usually a high correlation with the human visual judgment, and used to evaluate GANs algorithms.

Table 2.2: Comparison of our inpainting results to three state-of-the-art non-blind inpainting algorithms. Notation: ↑ Higher is better. ↓ Lower is better.

Metrics	Mask	GMCNN[76]	Pluralistic [31]	EdgeConnect [28]	Our
PSNR ↑	0-10%	23.59	25.88	32.19	31.62
	10-20%	22.25	23.92	27.73	27.16
	20-30%	21.08	22.38	24.95	24.34
	30-40%	20.11	21.23	23.09	22.46
	40-50%	19.26	20.19	21.50	20.91
SSIM ↑	0-10%	82.01	86.81	96.62	96.25
	10-20%	75.46	79.73	91.41	90.52
	20-30%	67.73	71.26	84.01	82.46
	30-40%	60.03	62.48	75.29	73.44
	40-50%	50.98	51.85	64.72	62.20
L1 ↓	0-10%	5.12	3.61	0.56	0.57
	10-20%	5.77	4.34	1.33	1.37
	20-30%	6.58	5.20	2.39	2.47
	30-40%	7.37	6.03	3.49	3.59
	40-50%	8.25	7.00	4.73	4.91
FID-score ↓	0-10%	66.65	43.51	8.88	8.42
	10-20%	84.69	56.23	21.56	21.49
	20-30%	107.64	75.90	42.32	41.87
	30-40%	134.82	87.23	62.89	60.68
	40-50%	157.12	108.58	88.10	86.20

Table 2.2 shows the numerical results of the inpainting algorithms for each mask subset, and in Figure 2.10 we can visually compare the results of the considered GMCNN, Pluralistic and EdgeConnect methods to the proposed approach and to the Ground Truth.

Both the qualitative and quantitative results demonstrate that our approach outperforms the GMCNN and Pluralistic techniques, while the numerical evaluation rates of the proposed approach are very similar to EdgeConnect. While our FID-Scores (which is considered as the most important parameter) are slightly better, yet we found that these results do not provide a reliable basis for comparison.

DOI:10.15774/PPKE.ITK.2023.007

For this reason, we have also performed a qualitative comparative survey between EdgeConnect and the proposed method so that for 25 different input images we showed the output pairs in a random order to 93 independent observes and asked them to choose the better ones without receiving any information about the generating method, and without using any time limitation. The participants of the survey were selected having different job backgrounds, but most of them were working in the fields of architecture or archaeology (Figure 2.11a displays the distribution of the different fields of expertize among the participants). Figure 2.11b shows for each test image the percentage of the answers, which marked our proposed algorithm's output better than EdgeConnect's result. This survey proved the clear superiority of our proposed method, since in 18 images out of 25 our approach received more votes than the reference method, and even in cases when our technique lost, the margin was quite small, which means that both approaches were judged as almost equally good. By summarizing all votes on all images, our method outperformed the Canny-based EdgeConnect with a significant margin: 58.59% vs. 41.41%.

The above tendencies are also demonstrated in Figure 2.12 for two selected images. On one hand, the numerical results of EdgeConnect are very close to our figures (EdgeCon/Ours PSNR: 27.02/25.95, SSIM: 86.67/85.16, L1: 1.46/1.67). On the other hand, we can visually notice that our algorithm reconstructs a significantly better mortar network, which is a key advantage for archaeologist aiming to understand how the wall was built. The superiority of our algorithm over the Canny-based EdgeConnect approach may be largely originated from the fact, that our method is trained to make a clear separation between the mortar and the bricks in the pre-processing stage, even in difficult situations such as densely textured wall images or walls with shadows effects. In such cases the Canny detector provides notably noisy and incomplete edge maps which fools the consecutive inpainting step.

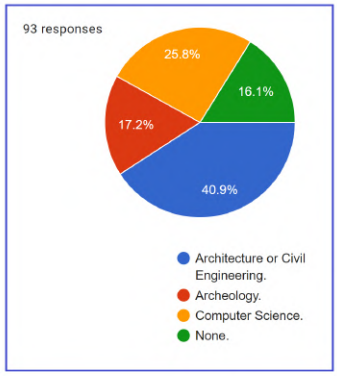
2.5.3 Quantitative Evaluation of the Segmentation Stage

We evaluated the results of the brick segmentation step for the images of Test Set (1) (synthetic—123 images), and Test Set (2) (real—47 images), since in these

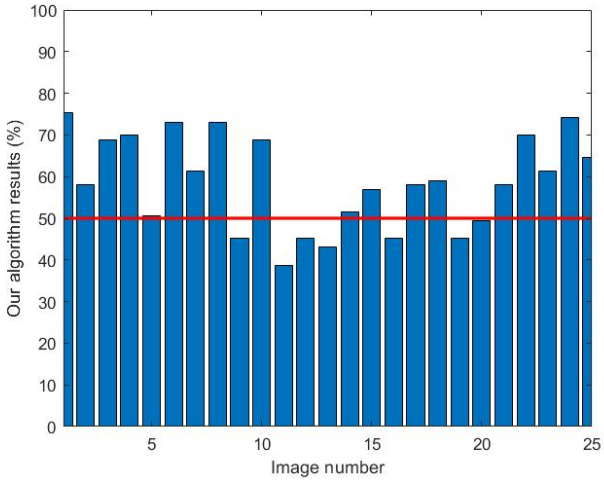
DOI:10.15774/PPKE.ITK.2023.007

Figure 2.10: Comparing our method to three non-blind state-of-the-art inpainting algorithms for four samples in the test dataset shown in the different columns. First row: input images with synthetic occlusions, Second row: GMCNN output [76], Third row: Pluralistic Image Inpainting output [31], Fourth row: Canny based EdgeConnect output [28], Fifth row: output of our proposed approach, Sixth row: Ground Truth.

DOI:10.15774/PPKE.ITK.2023.007

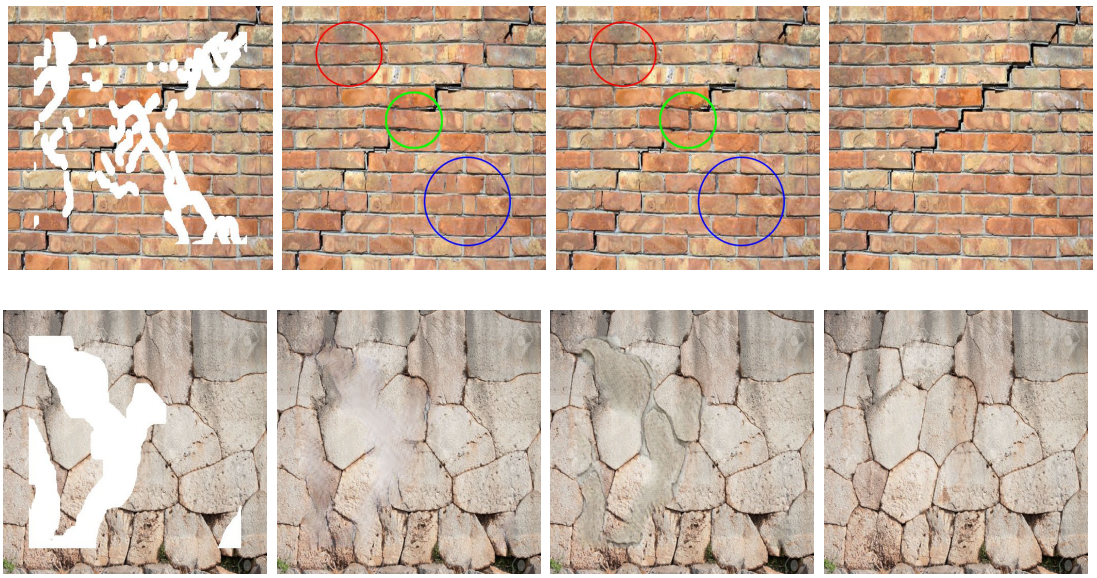


(a) Participants by experience.



(b) Our algorithm's winning % for each image.

Figure 2.11: Qualitative comparison results of the EdgeConnect and the proposed method using 25 selected test images. In the survey, 93 people participated with different background (see (**a**,**b**) shows for each image the percentage of the answers, which marked the proposed algorithm's output better than EdgeConnect's result.

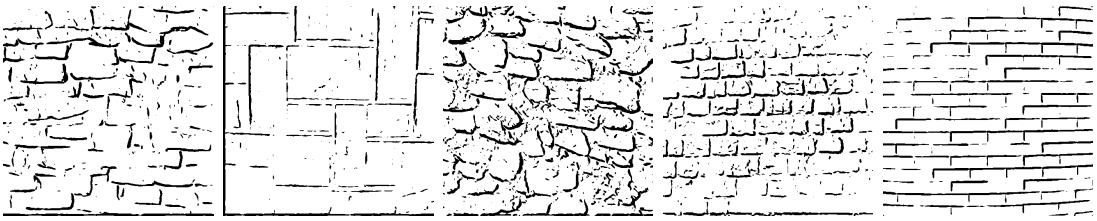


(a) Masked Input (b) EdgeConnect (c) Ours (d) GT

Figure 2.12: Comparison of the Canny based EdgeConnect algorithm [28] to the results of our proposed method and to the reference GT. Circles highlight differences between the two methods' efficiency in reconstructing realistic brick outlines.

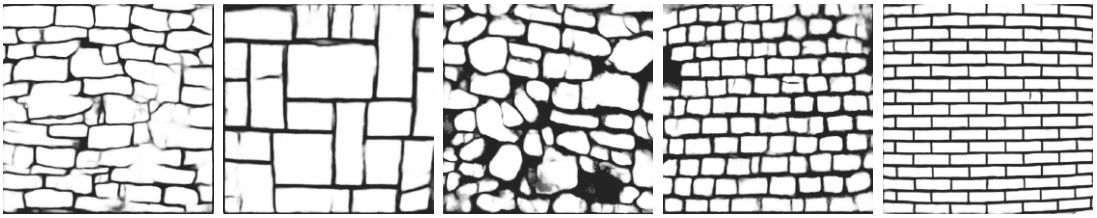
DOI:10.15774/PPKE.ITK.2023.007

(a) Raw input images

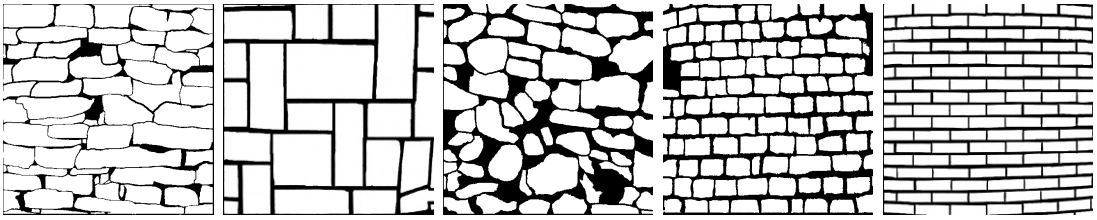


(b) Results of Riveiro's delineation method

(c) Results of Oses' delineation method



(d) Results of the proposed U-Net-based delineation



(e) The ground truth binary brick vs. mortar mask

Figure 2.13: Comparison between the-state-of-the-art delineation methods and our method; (a) images; (b) Riveiro's method; (c) Oses' method; (d) Our U-Net based delineation output; (e) Ground truth

DOI:10.15774/PPKE.ITK.2023.007

test sets, the background wall images are also available. we separately analyze the efficiency of the delineation step, and the final brick segmentation results.

2.5.3.1 Delineation Step Output

First, we start with the discussion of the delineation step, where we compare pixel-wise the output map of our U-Net component and other state-of-the-art methods [61, 65] (see Figure 2.13(b)-(d)) to the expected Ground Truth masks (Figure2.13(e)). Since the correctness of the structure can be better described by the pixel-level accuracy of the thinner mortar regions, we calculate pixel-level Precision (Pr) and Recall (Rc) values from the viewpoints of the mortar pixels, and take the F1-score as the harmonic mean of Pr and Rc.

Demonstrative sample results of using the state-of-the-art delineation methods and our method are shown in Figure 2.13, and the corresponding quantitative evaluation values are provided in Table. 2.3. We can confirm, that the proposed U-Net based method can detect the outlines of the bricks with high accuracy (above 80%) for any types of walls, significantly surpassing the reference methods, which suffer both from false detection and misdetection effects. While the reference techniques show better results by processing photos of walls with regularly shaped and aligned bricks, their performance is drastically degraded for the irregularly structured stone walls. In summary, we found neither of the two reference methods [65], [61] capable for providing efficient markers for the Watershed process.

Table 2.3: Evaluation of the delineation step. Comparison of state-of-the-art methods and our proposed U-Net-based approach.

Method	F1-score (%)	Precision (%)	Recall(%)
Riveiro method [61]	23.65	37.04	17.71
Oses method [65]	22.58	39.57	16.91
Proposed method	**81.57**	**81.16**	**82.14**

DOI:10.15774/PPKE.ITK.2023.007

2.5.3.2 Segmentation Stage Output

Second, we measure the segmentation accuracy for the occlusion-free wall images. In this evaluation step, we measure the algorithm's performance in two different ways: we investigate how many bricks have been correctly (or erroneously) detected, and also assess the accuracy of the extracted brick outlines. For this reason, we calculate both (i) object level and (ii) pixel level metrics. First of all, an unambiguous assignment is taken between the detected bricks (DB) and the Ground Truth (GT) bricks (i.e., every GT object is matched to at most one DB candidate). To find the optimal assignment we use the Hungarian algorithm [72], where for a given DB and GT object pair the quality of matching is proportional to the intersection of union (IOU) between them, and we only take into account the pairs that have IOU higher than a pre-defined threshold 0.5. Thereafter, object and pixel level matching rates are calculated as follows:

(i) At object (brick) level, we count the number of True Positive (TP), False Positive (FP) and False Negative (FN) hits, and compute the object level precision, recall and F1-score values. Here TP corresponds to the number detected bricks (DBs) which are correctly matched to the corresponding GT objects, FP refers to DBs which do not have GT pair, while FN includes GT objects without any matches among the DBs.

(ii) At pixel level, for each correctly matched DB-GT object pair, we consider the pixels of their intersection as True Positive (TP) hits, the pixels that are in the predicted brick but not in the GT as False Positive (FP), and pixels of the GT object missing from the DB as False Negative (FN). Thereafter we compute the pixel level evaluation metrics precision, recall and F1-score and the intersection of unions (IOU). Finally the evaluation metric values for the individual objects are averaged over all bricks, by weighting each brick with its total area.

Table. 2.4 shows the quantitative object and pixel level results of the complete workflow for different wall categories. We can conclude that our algorithm provides high quality output for all categories of the test dataset, as the brick and pixel level F1-scores are measured in almost every cases between 76% and

DOI:10.15774/PPKE.ITK.2023.007

97%. The best results are naturally observed for the ashlar fine subset, which contains high contrasted photos of modern buildings with simple and regular brick layouts. However, as both qualitative and quantitative examples shows, the proposed method can generally handle very different masonry types, and it shows graceful degradation in cases of more challenging samples, such as random masonry.

Table 2.4: Evaluation of brick segmentation. Object (brick) and pixel level precision, recall, F1-score and IOU values for the *augmented* test dataset.

Type	No. of (augm) images	Recall(%)		Precision(%)		F1-score(%)		IOU(%)
		Brick level	Pixel level	Brick level	Pixel level	Brick level	Pixel level	Pixel level
Random	304	83.80	82.08	77.69	83.03	79.93	81.87	71.31
Square	411	85.23	78.35	69.87	86.10	75.56	78.85	69.91
Dry	375	84.97	85.04	73.54	87.47	77.74	85.36	76.66
Fine	268	**97.53**	**96.43**	**97.67**	**92.18**	**97.58**	**94.19**	**89.12**
Rough	244	81.47	84.19	79.57	78.34	79.87	81.92	69.38
Average	1602	86.38	84.53	78.34	85.67	81.23	83.98	74.88

Thereafter, we also evaluate the accuracy of brick segment prediction in the occluded regions, by investigating the accuracy of the segmentation module in these regions. For the synthetic test set, we investigate the performance as a function of the number of occluding objects, moreover, we test the algorithm for the images with real occlusions (Test Set (2)).

Table 2.5 shows the quantitative object and pixel level results of the segmentation stage. The results confirm the efficiency of our algorithm for brick prediction in the occluded regions, as the F1-scores at both brick and pixel levels are in each case above 67%. Note that the observed results are slightly weaker for the images with real occlusions (Test Set (2)), which is caused by two factors. First, in these images the area of the occluding objects is relatively large (more than 30%). Second, the occluding objects form large connected regions, which yields a more challenging scenario than adding 5 smaller synthetic objects to different parts of the wall image, as we did in the experiment with artificial occlusions.

It is important to notice that the Ground Truth is not the only reference that our result could be compared to, as there might be many valid realistic solutions.

DOI:10.15774/PPKE.ITK.2023.007

Table 2.5: Evaluation of brick segmentation. Object and pixel level precision, recall, F1-score and IOU values for the *Test Set (1)* using different numbers (0-5) of synthetic occluding objects; and results on *Test Set (2)* with real occlusions.

Test	Num. of obj.	Recall (%)		Precision (%)		F1-score (%)		IOU (%)
		Brick	Pixel	Brick	Pixel	Birck	Pixel	Pixel
Set (1)	no occl.	82.79	76.58	73.11	85.14	76.62	78.20	70.24
	one	83.85	76.54	73.39	85.42	77.25	78.14	70.49
	two	82.89	76.11	72.91	85.12	76.56	77.70	69.54
	three	83.35	76.20	73.08	85.27	76.84	77.84	69.19
	four	82.28	75.15	71.65	84.15	75.63	77.00	68.21
	five	81.47	74.69	71.93	82.85	75.47	76.26	67.01
Set (2)	no occl.	79.76	72.78	70.73	80.57	74.44	74.18	63.78
	with Obj	67.56	66.25	72.26	71.84	69.22	67.44	54.04

However, both the quantitative results and the qualitative examples show that the proposed method can generally handle very different masonry types with high efficiency.

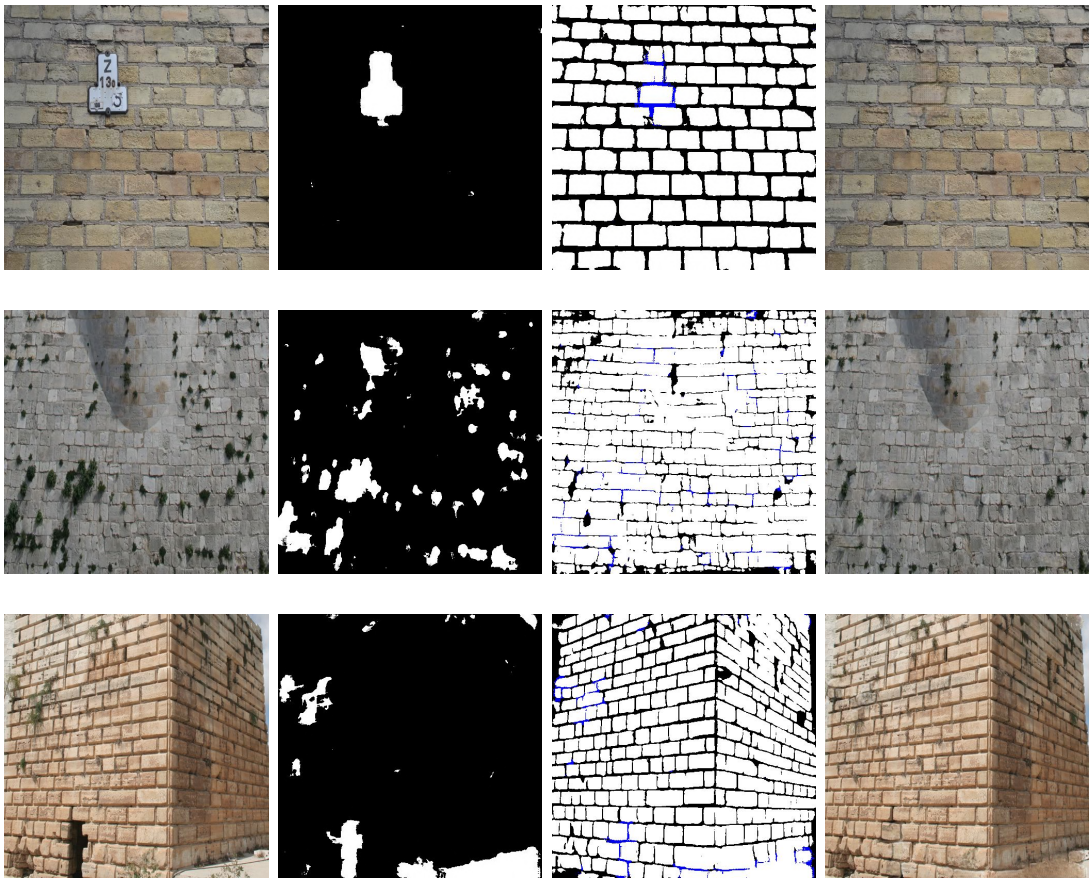
The prediction results show graceful degradation in cases of more challenging samples, such as random masonry, as the wall could take any pattern and all the predicting patterns in the occluding regions can be considered as a real pattern (see Figure 2.9). Expected outputs are more definite when we are dealing with square rubble masonry or ashlar masonry categories where the pattern is specific and well predictable (see the first and third rows in Figure 2.14).

2.5.4 Qualitative Results

In this section, we discuss further qualitative results demonstrating the usability of the proposed approach in various real application environments.

Figure 2.14 shows the results of occlusion detection, brick-mortar structure segmentation and prediction, and inpainting for real images from Test Set (3). The first image displays a wall from Budapest, where a street plate occludes a few bricks. Here exploiting the quite regular wall texture and the high image contrast, all the three demonstrated steps proved to be notably efficient. The second and third images contain ancient masonry walls from Syria occluded by some vegetation. We can observe that several plants were detected as occluded

DOI:10.15774/PPKE.ITK.2023.007

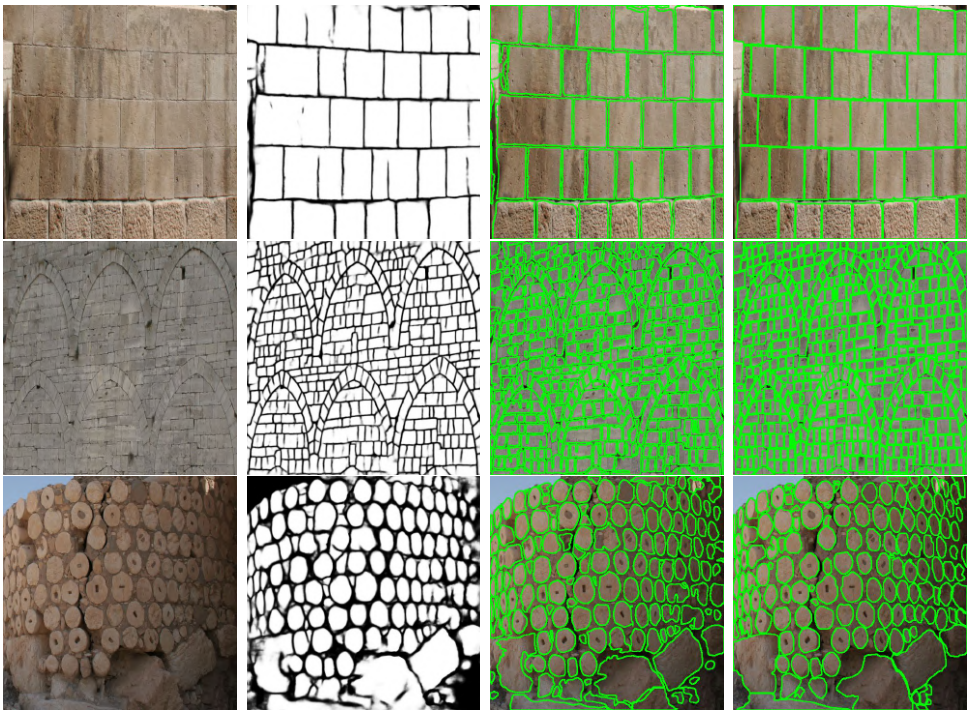


(a) Input (b) Mask (c) $G1$ Output (d) Inpainting result

Figure 2.14: Results of the proposed algorithm (inpainting step) on real scenes with real occlusions (first row: a wall in Budapest, last two rows: ancient walls in Syria).

regions and appropriately inpainted, while the wall's structure has also been correctly delineated and reconstructed. Note that such delineation results (i.e., outputs of the G1 component) can be directly used in many engineering and archeology applications, such as analysis of excavation sites or maintenance of buildings. On the other hand, we can also notice a number of further issues in these images, which are worth for attention. First, due to the low contrast of the second image (middle row), not all mortar regions could have been extracted, but this fact does not cause significant degradation in the inpainting results. Second, some vegetation regions are left unchanged, which artifact can only be eliminated with involving further features during the training. Third, in the last row, the

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) Mask (c) Ours (d) GT

Figure 2.15: Results of the brick segmentation step of the proposed algorithm on real scenes for ancient walls in Syria.

algorithm also filled in a door, which was identified as an occluded or damaged wall region. To avoid such phenomena, special elements, such as doors, windows or ledges could be recognized by other external object recognizer modules [79], and masked during the inpainting process.

Figure 2.15 shows further brick segmentation results for some occlusion-free archeological wall images (also from Test Set (3)), which have challenging structures with diverse brick shapes and special layouts. Moreover the first and second images also have some low contrasted regions which makes brick separation notably difficult. First we can observe that the delineation step (second column) is highly successful even under these challenging circumstances. Moreover, even in cases where touching bricks were merged into a single connected blob in the delineation map, the watershed based segmentation step managed to separate them as shown in the final output (third column), which is notably close to the manually marked Ground Truth (fourth column).

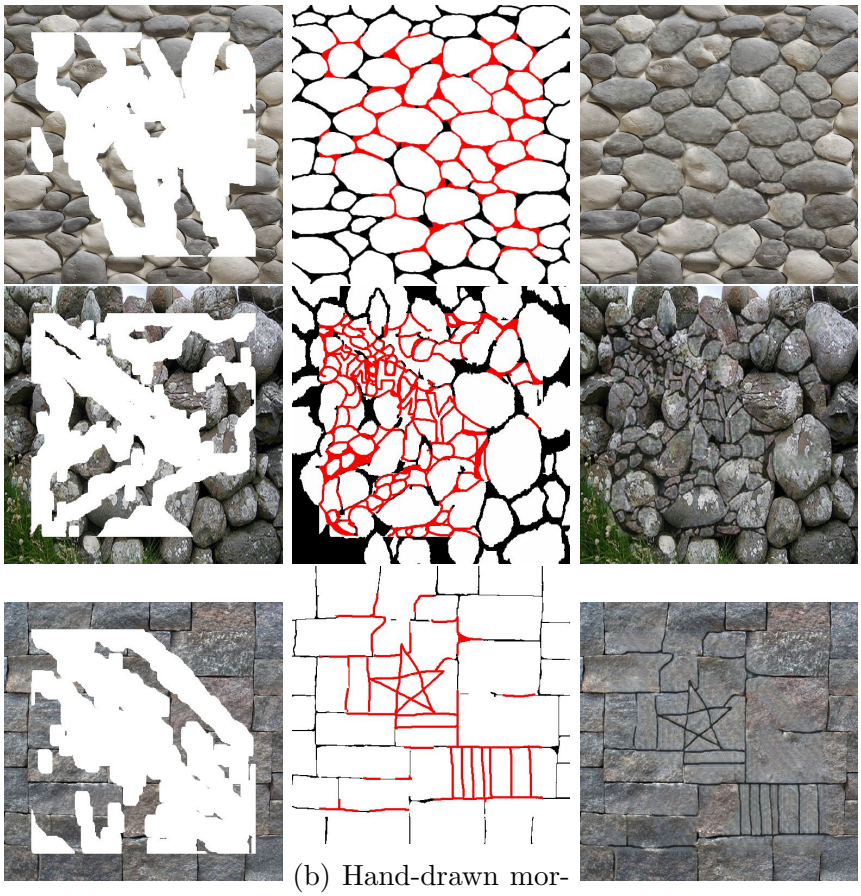
DOI:10.15774/PPKE.ITK.2023.007

Next we introduce a possible extension of our algorithm which offers an additional feature, making our approach unique among the existing inpainting techniques. We give the users the flexibility to draw and design their own perspective of the mortar-brick pattern in the hidden (occluded) areas via simple sketch drawings. Then the inpainting stage uses this manually completed delineation map as input to produce the colored version of the reconstructed wall image, whose texture style follows the style of the unoccluded wall segments, meantime the pattern defined by the sketch drawing also appears in the inpainted region. This feature allows the archaeologists to easily imagine the wall structure of the hidden parts which might also help in reassembling the wall elements through a better visualization. Figure 2.16 demonstrates the above described process. First, we assume that some regions of the wall are hidden (see white pixels in Figure 2.16a). Thereafter, we allow the user to modify the delineation map by hand, adding new mortar regions which are shown in red in Figure 2.16b. Finally the algorithm completes the wall elements with realistic results (Figure 2.16c), which works even if the user draws unexpected or non-realistic wall structures as shown in the second and third rows of the figure.

For a brief summary, Figure 2.17 demonstrates the results of our proposed algorithm step by step. The first row shows selected input wall images which contain various occluding objects (synthetic occluding objects in the first two columns, and real ones in the last two columns). The second row displays the output of the pre-processing stage where occluding objects are shown in gray, and bricks in white. The third row is the $G1$ output where the predicted mortar pixels under the occluded regions are shown with blue color, and the forth row displays the output of the inpainting stage which is an inpainted color image. The fifth row shows the output of the brick segmentation stage, which is followed by the Ground Truth, i.e., the original wall image without any occluding objects.

In the first example (see the first column), the pre-processing step predicts some segments of the yellow line as occluding objects, due to their outlier color values compared to other segments of the wall. The false positive occlusion predictions naturally affect the final results, however, as shown by this example, the inpainting result is yet realistic, and the brick segmentation output is almost perfect here. In the second and third columns, the bricks of the walls follow

DOI:10.15774/PPKE.ITK.2023.007



(a) Input images (b) Hand-drawn mortars (c) Output

Figure 2.16: Inpainting results with utilizing simple sketch drawings (shown by red in the middle column) created by experts for mortar structure estimation in the occluded regions.

less regular patterns, but even in these complex cases, the proposed algorithm predicts efficient mortar-bricks maps and realistic inpainted color images of the walls. Moreover, the brick segmentation results are of high quality in the non-occluded regions, despite the high variety in the shapes of the brick components. The wall in the fourth (last) column follows a periodic and highly contrasted pattern, thus the pre-processing stage is able to separate the mortar, brick and occlusion regions very efficiently. Some minor mistakes appear only in the bottom-right corner, where the color of the occluding person's lower leg is quite similar the one of the frequent brick colors in the image.

DOI:10.15774/PPKE.ITK.2023.007

Based on the above discussed qualitative and quantitative results, we can conclude that our algorithm provides high quality inpainting and segmentation outputs for different wall patterns and structures, in cases of various possible occluding objects.

2.5.5 Style Transfer Results

In Figure 2.18, we show results for the *style transfer* application, where we present two different *content image* samples in the first row, and each of them is transformed in the process using two different *style images*. (The style images and the corresponding style transfer results are displayed side by side in the second and third rows.)

Next, we present the results of transforming a binary brick-mortar map to a wall image. Figure 2.19 displays two brick-mortar sketch maps, which can be drawn even by hand. Each of them is transformed to different colored wall images using two distinct *style images*, and the results are shown in the last two rows.

We can observe in both Figure 2.18 and 2.19 how the algorithm manged to paint the style texture of the style image onto the brick-mortar pattern of the processed image or the hand drawn sketch map, so that the mortar regions efficiently match the color of the mortar, and the appearances of the brick regions match the brick texture of the style image. However there are a few error cases when some neighboring bricks are merged into a large brick object in the image as shown in the center part of Figure 2.18(d).

In image inpainting tasks there are usually many alternative solutions. Therefore, similarly to our user survey regarding the image inpainting in Sec. 2.5.2.2, we conducted a user survey to assess our proposed style transfer approach. Using 15 different style images and 15 content images (9 color wall images and 6 brick-mortar maps), we generated 15 stylized images by our network. We showed the outputs to 28 test subjects displaying the images side-by-side with the style images, and asked them to vote whether they find the result of our style transfer network visually appropriate or not. 18 out of 28 participants said that more than 66% of the seen images were transformed appropriately; 10 out of 15 images received more than 66% votes that it has been transferred appropriately. By

DOI:10.15774/PPKE.ITK.2023.007

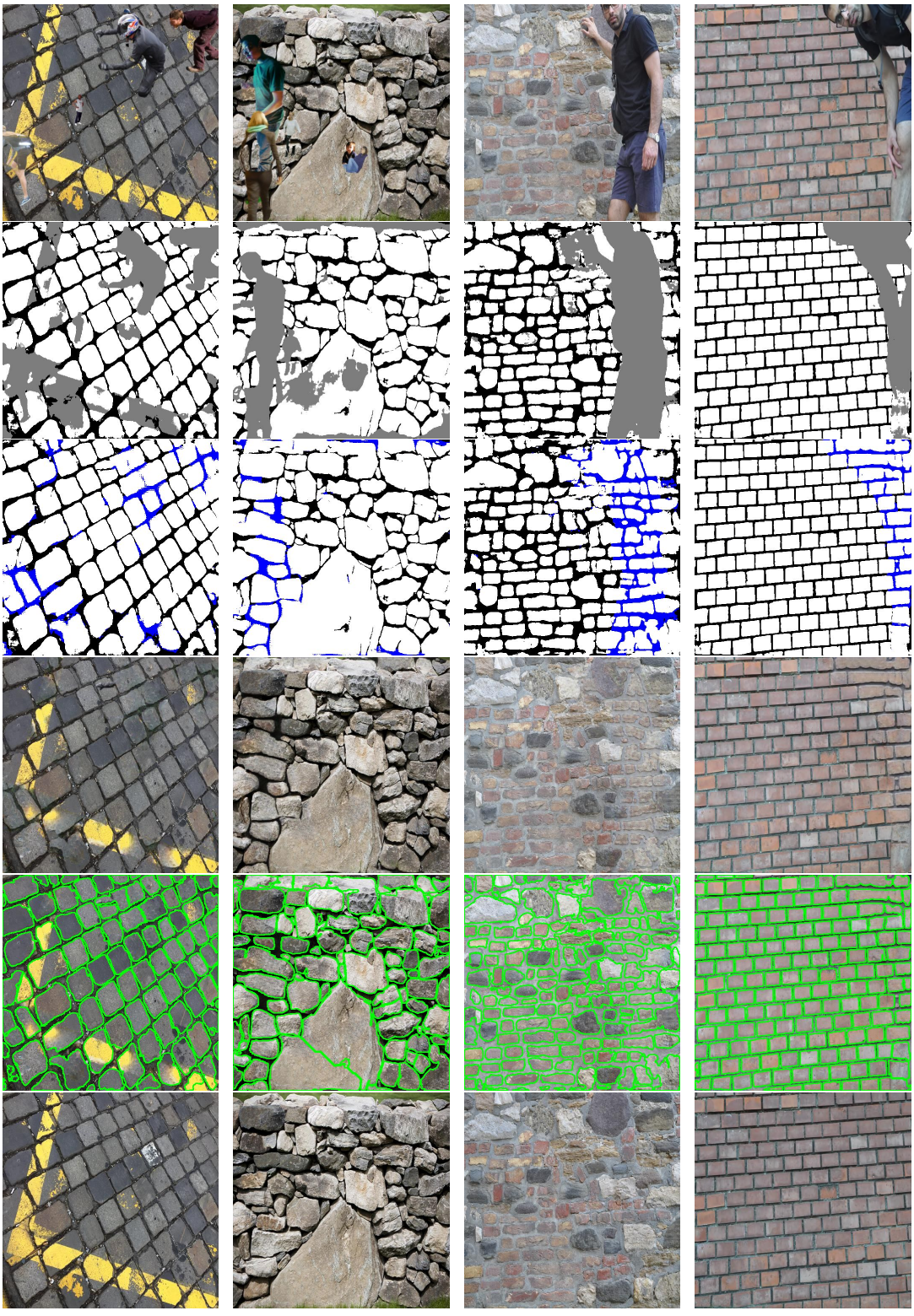
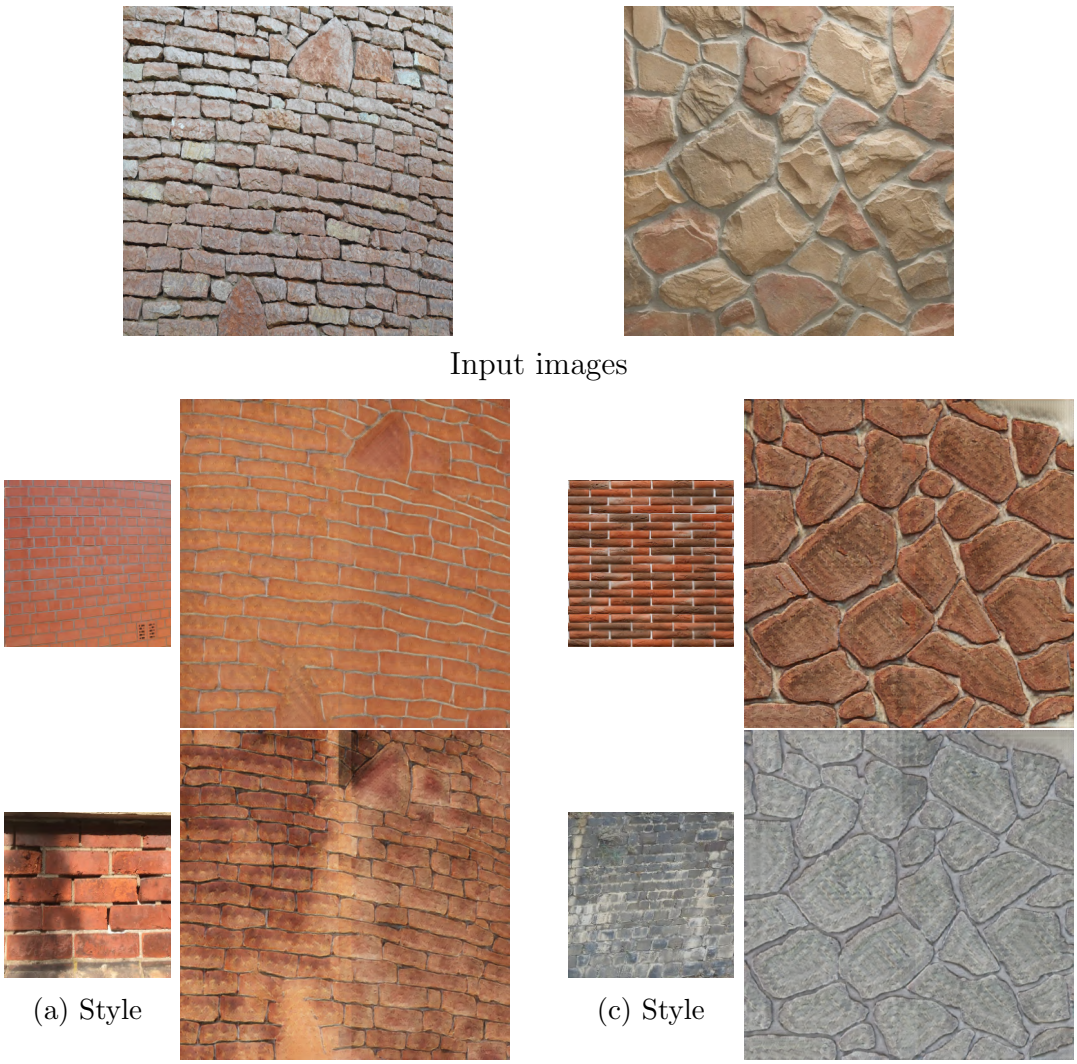


Figure 2.17: Results of the proposed algorithm step by step for four samples in the test dataset, first row: input images with occluding objects (first two images: synthetic occlusions, last two images: real occlusions), second row: U-net output, third row: I_{G1_out} output, fourth row: segmentation output, fifth row: GT.

DOI:10.15774/PPKE.ITK.2023.007

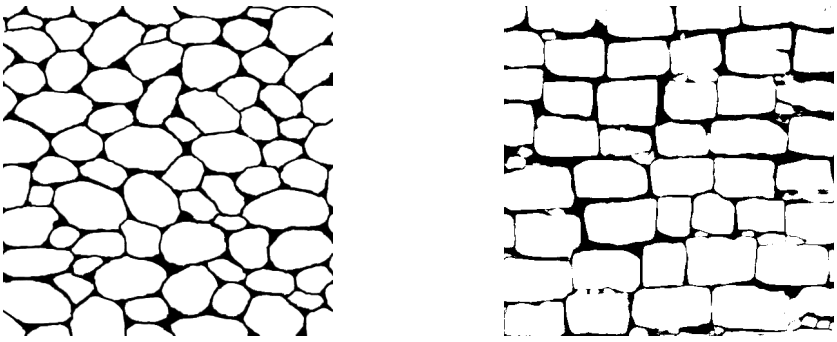


Input images

(a) Style (b) Our output (c) Style (d) Our output

Figure 2.18: Wall to wall style transfer samples, first row shows the input images, second and third rows represent the style image and our output side by side.

DOI:10.15774/PPKE.ITK.2023.007

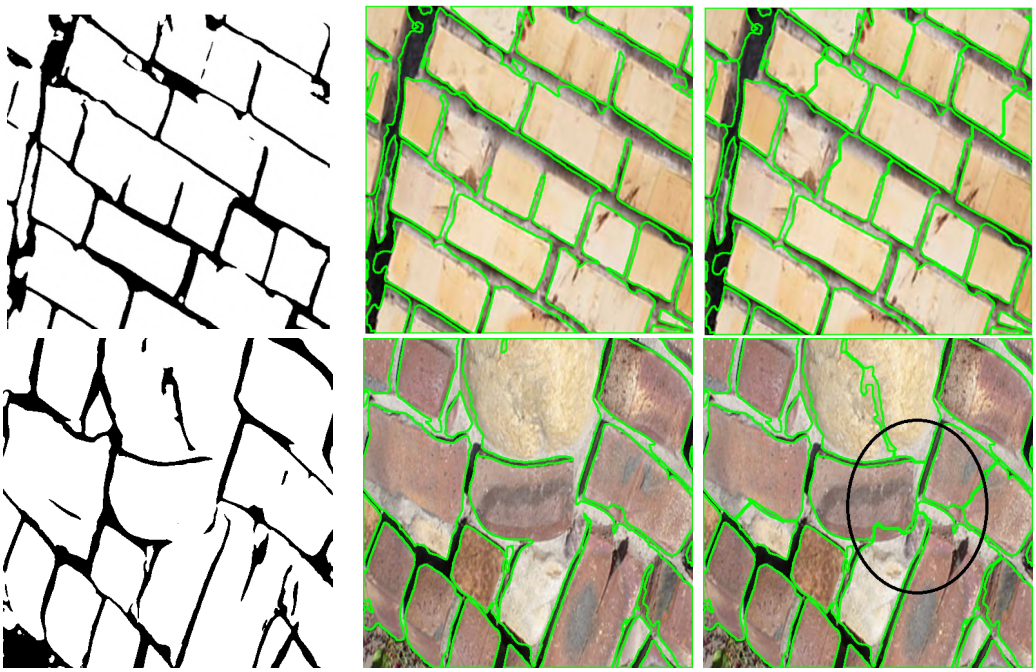


Input brick-mortar delineation maps

(a) Style (b) Our output (c) Style (d) Our output

Figure 2.19: brick-mortar map to a wall style transfer samples.

DOI:10.15774/PPKE.ITK.2023.007



(a) U-Net based mask (b) CCA labeling (c) Prop. Watershed

Figure 2.20: Comparison of the brick segmentation results with connected component analysis (CCA) and the proposed Watershed technique based on the same U-Net mask.

summarizing all votes on all images, 70.47% (296/420) images were considered being transferred correctly.

2.6 Ablation Studies

In this section, we present ablation studies to support our decisions regarding model construction and to provide further information about the employed method.

2.6.1 Watershed as a Post-processing Step

The necessity of applying the marker based Watershed process instead of using a simple connected component analysis (CCA) approach becomes evident by checking Figure 2.20 and Table 2.5. Figure 2.20 displays for a sample region the brick segmentation result by CCA and by the proposed Watershed algorithm in

DOI:10.15774/PPKE.ITK.2023.007

parallel. As shown, if some mortar sections are missing or misdetected, neighboring bricks can be erroneously merged into the same object by CCA, while the Watershed approach efficiently handles these situations. Table 2.6 confirms that such effects may also cause notable differences in quantitative performance parameters, especially for *rough ashlar* walls.

Table 2.6: Object (brick) level F1-scores of connected component analysis (CCA) and the proposed Watershed technique for brick segmentation using in both cases our U-Net based delineation maps as input.

The method	Random	Square	Dry	Fine	Rough	Average
CCA	77.46	**77.44**	76.23	95.76	67.02	78.63
Prop. Watershed	**79.93**	75.56	**77.74**	**97.58**	**79.87**	**81.23**

2.6.2 Hidden Feature Generator Impact

The main goal of the G_1 Hidden Feature Generator is to generate mortar and brick regions with a similar pattern to the observable wall segments. Here in this section, we compare our reconstructed delineation map $\mathrm{I}_{\mathrm{G1\_out}}$, and (as reference) the generated edge map of the Canny based EdgeConnect algorithm, to the GT delineation map $\mathrm{I}_{\mathrm{wall\_ftr}}$. It should be noted, that since EdgeConnect is a non-blind method, by this comparison we used our U-Net based occlusion masks for both EdgeConnect and our proposed technique. Qualitative test results are shown in Figure 2.21, which clearly demonstrates the superiority of our proposed method over the Canny based reference approach. Moreover, we can see here two limitations of EdgeConnect: in the first row, we can find a densely textured image of a stone wall, whose edge map is notably noisy leading to a poor delineation result. In the second row the shadow causes both false and missing Canny edges, which effect fools the consecutive inpainting step. On the other hand, our method provides in both cases a clear distinction between the bricks and mortar regions, since the model has been trained for such cases as well.

We also performed quantitative comparison at this stage, by calculating the pixel-level Precision (Pr), Recall (Rc) and F_1-score values for the predicted mortar regions in the area covered by the occlusion mask. Here we compared both

DOI:10.15774/PPKE.ITK.2023.007

Table 2.7: Numerical comparison of the Canny-based delineation algorithm of the EdgeConnect approach, and the Hidden Feature generator of our method for wall structure prediction

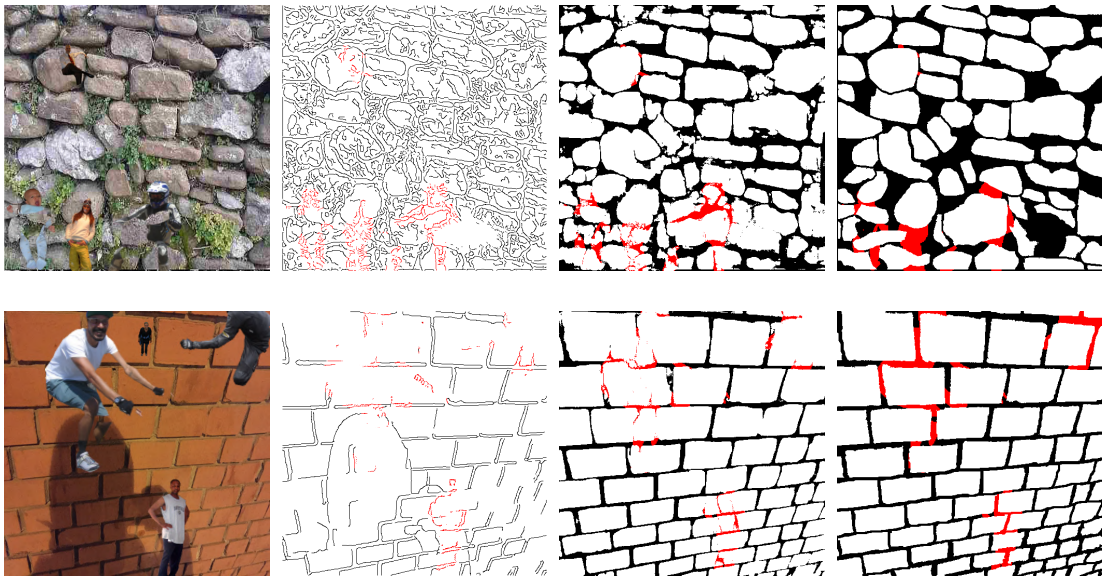
obj.	**Precision (%)**		**Recal(%)**		**F1-score(%)**	
	EdgeCon.	**Ours**	**EdgeCon.**	**Ours**	**EdgeCon.**	**Ours**
Synthetic	34.02	**59.57**	12.89	**52.06**	16.60	**53.37**
Real	27.89	**40.11**	5.85	**38.40**	8.29	**37.83**

EdgeConnect's and our results to the GT delineation map I_{wall_ftr}. Table 2.7. shows the average of the obtained Pr, Rc, and F_1-scores in the test data set, considering both the synthetic and the real test samples. Note that due to multiple possible solutions for structure inpainting even our outputs do not exactly match the GT (38-53% F_1-score), however they are significantly more correlated with the hand-labeled mask, than the maps of the Canny based technique (8-17%). Note that to fairly compare our technique to EdgeConnect, we had to separately train the U-Net component from the remaining parts of the network. When we switched back to using our end-to-end approach, we observed that the results have slightly improved: for example the F_1-scores measured in Table 2.7 for real images increased from 37.83% to 39.1%. Apart from these numerical measures, the visual results of Figure 2.21 confirm that our approach is able to predict robustly and efficiently the mortar-brick structure in the occluded areas. It is important to emphasize, the accuracy of predicting the mortar in missing areas largely influences the generation of realistic color images in the subsequent *Image Completion* step. For these reasons, we regard the efficiency of the *Hidden Feature Generator* as a significant by-product of our research work.

2.7 Conclusions of the Chapter

This chapter introduced a new end-to-end wall image analysis and style transfer from one wall to another. Our network has two different ways of utilization: in the first one, the algorithm detects the occluded or damaged wall parts based on a single image, and inpaints the missing segment with the expected wall elements. In addition, the method also provides an instance level brick segmentation output

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) EdgeCon. (c) Our G_1 (d) GT

Figure 2.21: Comparison of the delineation mask of EdgeConnect to our method (I_{G1_out}) and to the Ground Truth (GT, I_{wall_ftr}).

for the inpainted wall image. Our method consists of three stages. The first, preprocessing stage adopts a U-Net based module, which separates the brick, mortar and occluded regions of the input image. This preliminary segmentation serves as input of the inpainting stage, which consists of two GAN based networks: the first one is responsible for wall structure compilation; while the second one for the color image generation task. The last stage uses the watershed algorithm which fulfills accurate brick segmentation for the complete wall. The algorithm's second application involves altering the second GAN's inputs to fit two different wall images.

We have shown by various quantitative and qualitative experiments that for the selected problem the proposed approach significantly surpasses the state-of-the-art general inpainting algorithms, moreover, the segmentation process is highly general and largely robust against various artifacts appearing in real-life applications.

DOI:10.15774/PPKE.ITK.2023.007

Chapter 3

MVPCC-Net: Multi-View Based Point Cloud Completion Network for MLS Data

This chapter introduces a novel multi view-based method for completing high-resolution 3D point clouds of partial object shapes obtained by mobile laser scanning (MLS) platforms. The proposed approach estimates both the geometry and color cues of the missing or incomplete object segments, by projecting the 3D input point cloud by multiple virtual cameras, and performing 2D inpainting in the image domains of the different views. In contrast to existing state-of-the-art methods, the proposed method can generate point clouds consisting of a variable number of points, depending on the detailedness of the input measurement, which property highly facilitates the efficient processing of MLS data with inhomogeneous point density. For training and quantitative evaluation of the proposed method, a new point cloud dataset is introduced, which consists of both synthetic point clouds of four different street objects with accurate ground truth, and real MLS measurements of partially or fully scanned vehicles.

DOI:10.15774/PPKE.ITK.2023.007

3.1 Introduction

As a result of the rapid advancement of 3D data acquisition technology and the decreasing prices of 3D sensors, point clouds have become widely available formats for representing the three-dimensional environment in various robotics, surveillance, and autonomous driving applications. On the other hand, point clouds collected by real scanners frequently provide only partial shapes of the scanned items due to low sensor resolution, occlusions, and the limited number of viewpoints used during scanning. Therefore point cloud completion, i.e. the estimation of an object's full shape from point sets which only partially describe its geometry, is a fundamental key challenge in numerous computer vision and robotic tasks, such as virtual reality (VR)/ augmented reality (AR) applications [88], object tracking and simultaneous localization and mapping (SLAM) [89, 119].

3.1.1 Problem Statement

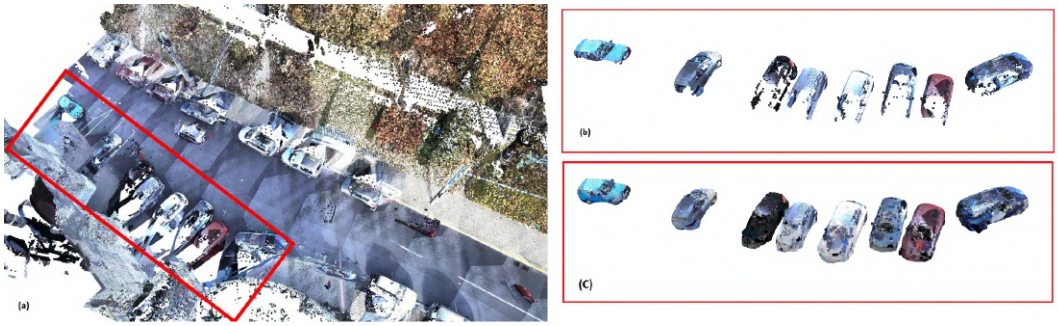


Figure 3.1: MLS data and our results on vehicle shape completion: (a) MLS data from a Budapest street showing numerous incomplete car shapes, (b) Partial car shapes obtained by semantic point cloud segmentation [110] from the MLS scene, (c) Our results for completing the car shapes.

Mobile laser scanning (MLS) platforms equipped with time synchronized Lidar sensors and navigation units can produce very dense and feature-rich point clouds for urban areas, as shown in Figure 3.1(a). However, due to the fact that the scanning vehicle can only move on the road, the point cloud models collected for many field objects have incomplete shapes. For example, the sidewalk sides of the

DOI:10.15774/PPKE.ITK.2023.007

parking vehicles are typically missing from the scanned scene models (Figure 3.1). Exploiting that available semantic point cloud segmentation methods [110, 115] can efficiently separate regions of different object classes in an MLS scene, we address in this paper the task of filling the missing regions of selected MLS object shapes to create a more realistic representation of the real environment. For example, from vehicle regions segmented from the raw MLS point cloud (Figure 3.1(b)), we aim to derive completed vehicle point cloud models (Figure 3.1(c)).

Existing point cloud completion methods apply various approaches to address the underlying unstructured nature of the point clouds, such as voxelization [99], intermediary 3D grids [102], or directly processing the point cloud [112] with the PointNet encoder [115]. These techniques have achieved remarkable success in terms of estimating complete geometric models of various object shapes. However, the obtained 3D point cloud models are often only roughly detailed, because the above methods are restricted to provide outputs with a fixed constant number of points, regardless of the size, shape complexity, or resolution of the input point cloud measurement segment corresponding to a given object.

In order to apply the aforementioned techniques [99, 102, 112] to MLS data, the input point cloud should be spatially downsampled, yielding as output simplified object shape models with significantly lower point density and less geometric detailedness compared to the genuine Lidar measurements. Moreover, a typically featured test scenario of these methods focuses on generating realistic object shapes from only a few (e.g. less than 15) measurement points, which task has a large degree of freedom in terms of the possible acceptable solutions: these use-cases are more connected to shape generation than shape completion.

3.1.2 Aim of the Chapter

In this chapter, we introduce a point cloud completion algorithm that can handle the unstructured nature, and maintain the high resolution of the MLS measurement data. To address the above challenges, we propose the *Multi-View Based Point Cloud Completion Network* (MVPCC-Net), a network designed to generate dense and detailed 3D object models from partial point cloud measurements.

DOI:10.15774/PPKE.ITK.2023.007

The algorithm encodes the input 3D point cloud by a set of sparsely filled multi-channel images representing both geometry and color information available from the sensor data. This approach allows us to use 2D Convolutional Neural Networks (CNNs) for filling in the missing structural and color information in the 2D image domain. Thereafter, the inpainted multi-view grid maps are fused to produce dense 3D point clouds representing the completed object shapes.

In addition, we present here a novel point cloud dataset which consists in part of synthetic data, and in part of real colored MLS point clouds of partially or fully scanned vehicles, captured by a car-mounted mobile laser scanning system in Budapest, Hungary. Since by processing real MLS street measurements, we cannot rely on accurate ground truth models, our method was trained on the synthetic part of the dataset derived from ShapeNet [114] for four street object classes. The ShapeNet-based point clouds are used as well for quantitative comparison of our technique versus various state-of-the-art methods. On the other hand, we also demonstrate that with using an additional network component for orientation adjustment of the input point clouds, the MVPCC-Net trained on purely synthetic data can be directly applied for completing real MLS object samples without the need of any additional fine-tuning step.

3.2 Related Work

Numerous deep learning approaches have been developed for 3D point cloud processing. In volumetric approaches, point clouds are *voxelized* using a 3D grid which is taken as input of a three-dimensional convolutional neural network. Multi-view techniques project 3D point clouds to several planes from various perspectives and extract view-wise information. The PointNet method [115] and its extensions [103] directly process the point clouds using a symmetric function, which allows the network to tolerate uncertainty in the order of points, and accurately captures both global and local properties of a point cloud.

In this section we provide a methodological review on state-of-the-art algorithms used for 3D shape completion and on multi-view approaches for point cloud processing.

DOI:10.15774/PPKE.ITK.2023.007

3.2.1 3D Shape Completion Methods

3D shape completion methods can be categorized into three main groups: Geometry-based approaches [122] have effectively been utilized to repair small holes on point clouds, using geometric restrictions such as local surface or volumetric smoothness. Template-based approaches [123] deform or reconstruct 3D point clouds that correspond to the most similar templates detected in a 3D shape database.

Learning-based approaches have been widely adopted by 3D point cloud completion techniques due to the availability of synthetic public datasets like ShapeNet. PointNet is utilized as an encoder in multiple state-of-the-art techniques [112, 118, 113, 101] with various types of decoders: FoldingNet's decoder [101] warps a predefined 2D grid so that it fits the input point cloud, by using two successive three-layer perceptrons. The Point Completion Network (PCN) [112] employs two-stage decoders that combine the advantages of fully-connected and folding-based decoders. Extending the PCN network, the Vehicle Points Completion-Net (VPC-Net) [113] combines the partial inputs with the PCN's decoder outputs to construct more homogeneous point clouds with finer-grained information. TopNet [118] includes a decoder that generates point clouds in a hierarchical structure, where each point operates as a branch of a tree. SoftPoolNet [100] provides a two-stage, multi-resolution architecture for completing point clouds by substituting max pooling with softmax function in order to retain local information. The 3D-Encoder-Predictor Network (3D-EPN) uses directly a volumetric representation of 3D point clouds [99]. However, converting point clouds to 3D volumes yields data quantization, which step can remove many fine-grained details. For this reason, in the Gridding Residual Network (GRNet) [102] 3D grids are proposed to regularize unstructured point clouds while keeping their context and structure detailed. The recent PoinTr [106] method and its extension called AdaPoinTr [107] consider the point cloud completion issue as a set-to-set translation problem, so they transform the point clouds into a sequence of point proxies, then they use a Transformer encoder-decoder architecture for point cloud completion. A topology-aware method called LAKeNet [104] fills in the missing parts of the 3D point cloud's structure by using three steps: aligned keypoint localization, surface skeleton generation, and shape refinement. SeedFormer [109] introduces

DOI:10.15774/PPKE.ITK.2023.007

a novel Upsample Transformer with a new shape representation (Patch Seeds) to preserve both regional information and global structures. Snowflake Point Deconvolution (SPD) is an approach developed by SnowflakeNet [108] to complete the shape of the point cloud, where child points are generated progressively from selected parent points.

However, all of the above methods use Chamfer Distance (CD) as training loss, thus that they minimize the mean of local point-to-point distances between the predicted and the ground truth point clouds, which process does not guarantee the effective characterization of shape similarity [102].

3.2.2 Multi-view based Approaches

Multi-view based approaches have recently shown their efficiency in several tasks, including classification [98, 120] and segmentation [96, 97, 121], moreover, reconstructing 3D shapes from a single image or a series of images is an active research area with different applications in robotics, and in virtual/augmented reality. Existing approaches adopt various output representations. A voxel-based output is provided by [95], where a 2D convolutional neural network encodes 2D images into a latent representation, which is subsequently decoded into 3D object shapes by a 3D convolutional neural network. The Point Set Generation Network [93] derives a point-based output, so that a set of unordered points is directly extracted from a single image. Lin et al. [94] present pseudo-rendered depth images as output and construct dense 3D objects by re-projecting them.

Image and point cloud fusion-based techniques have also been proposed for 3D shape completion recently. The View-guided Point Cloud completion method (ViPC) [91] relies on auxiliary RGB image data for point cloud completion, assuming that the input image contains structural information for the missing shape part. Similarly, the Cross-modal Shape-transfer model (CSDN) [92] combines the image and point cloud information in expressing a full shape.

3.3 Data Generation

While supervised deep learning-based algorithms require extensive training data, collecting proper 3D point cloud measurements for our method from real-world

DOI:10.15774/PPKE.ITK.2023.007

environments – including both incomplete and complete or manually completed object shapes – is highly challenging and resource intensive. To circumvent these limitations, we trained and quantitatively evaluated our model using synthetic data, which consists of pairs of partial and complete point cloud models of various 3D object shapes, so that the incomplete point clouds can be used as input of our method, while the complete shapes of the same objects as ground truth. In addition, we extensively tested our trained shape completion network on real-world (incomplete) Mobile Laser Scanning (MLS) measurements. We provide access to both the synthetic and real data used in our tests in a novel public dataset for the scientific community.

3.3.1 Synthetic Data

Recent works [112, 118, 113] utilized ShapeNet [114], a large-scale 3D synthetic dataset to construct the training data, deriving point clouds from meshes available in ShapeNet by sampling a predefined number of points. However, point cloud models generated in this way often cannot be considered as efficient references for real MLS data, since, for example, they may also contain various internal object structures which are occluded during an outdoor scanning process.

To address the above issue, we used the approach described by [90] to generate the (Ground Truth) 3D point clouds of objects from their complete mesh models. First we render a model based on projections of the mesh to discrete 2D lattices from distinct viewpoints, which are re-projected in the next step to the object's coordinate system. We set the resolution of the lattice of projection so that for each object we produce a dense, colored point cloud that accurately depicts even fine visible surfaces. Henceforward we refer to this point cloud as *fine GT*, where the different object models may consist of a variable number of points depending on the detailedness of the shape's mesh model.

Next we generate incomplete point cloud models for the objects of interest, which can be used as input of our algorithm during training and also in the test phase for quantitative evaluation. Here the previously created point clouds models of the complete object shapes are projected by a *subset* of the above defined virtual cameras, and the views of the selected cameras are re-projected in

DOI:10.15774/PPKE.ITK.2023.007

the 3D space. In this way we can synthesize partial object point cloud samples, that represent only points visible from the selected virtual cameras.

As processing urban MLS data is the primary goal of our approach, we selected four object categories from ShapeNet, that are relevant for street scenarios: car, bus, motorcycle, and train. Since a MLS system captures a scene from the top of a scanning vehicle, it often cannot capture the bottom part of street objects. Therefore by simulated MLS point cloud generation, we did not place upward looking virtual cameras to the ground plane, we applied instead several side-view and downward facing cameras around the object.

Our synthetic dataset contains in total 4918 distinct models, of which 4580 objects are used to train our model and 338 ones are used for evaluation. Twenty partial samples were generated for each complete object shape using twenty distinct perspectives, yielding a training set of 91,600 objects and a test set of 6760 samples. Some partial object samples of the new dataset are shown in the first column of Figure 3.3, while the last column of the same figure demonstrates the corresponding complete point clouds used as ground truth.

Technically, our deep neural network is trained using a set of 2D six-channel images, which are derived from the partial and complete point cloud samples by projections in advance. To prepare the data, in the preprocessing phase, we generate twenty images from twenty distinct perspectives, while in the training phase, we follow a preliminary fixed view selection strategy - presented in Sec. 3.6.1 - which ensures that the selected views evenly surround all sides of the object.

3.3.2 MLS Data

Apart from synthetic training and test data generation, we also created a real-world test set that consists of (mostly partial) vehicle point clouds extracted from measurements of a Riegl VMX-450 MLS scanner. The raw MLS test data was provided by the City Council's Road Management Department (Budapest Közút Zrt.) in Budapest, Hungary. For ensuring accurately segmented vehicles in the new test set, we utilized a user friendly 3D point cloud annotator tool described in

DOI:10.15774/PPKE.ITK.2023.007

[110]. In the preprocessing phase, each object sample was scaled and transformed to fit within a 3D box with coordinates between -0.5 and 0.5.

Our real MLS data collection consists of 424 object samples in total. On one hand, 370 point clouds represent partial vehicle shapes, where the scans of the complete objects are not available in the MLS data, thus they can only be used for qualitative analysis of the proposed technique (see Fig. 3.5). On the other hand, 54 samples depict almost entire vehicle shapes (see Figure 3.4 (g)), which can be also used as ground truth similarly to the synthetic models presented in Sec. 3.3.1. Here we generated four partial point cloud samples from each complete MLS vehicle shape, each one was created by reprojecting an image created from a single virtual camera position, which was located in the front, behind, to the right, or to the left of the selected object of interest (see Figure 3.4 (a)). Note that while the synthetic dataset has a quite homogeneous point cloud characteristic, we can observe notable point density variations within the set of real MLS samples. In our dataset, an input MLS point cloud representing an incomplete car shape consists of – in average – 31K points, and their size range varies between 2K and 130K points, while the complete car point clouds contain 35K points in average, spanning the range of 15K–72K points.

3.4 Proposed Approach

The main goal of the proposed 3D point cloud completion approach is to transform a point cloud segment representing only a portion of an object into a point cloud describing the entire object shape. For uniform treatment, we initially ensure that the incomplete input point cloud segments are scaled and reshaped to fit inside a 3D unit bounding box with point coordinates in the range of [-0.5, 0.5].

As shown in Figure 3.2, our method implements three sequential steps: (a) Calculation of a multi-view image-based representation of the incomplete point cloud measurement, (b) Shape and color completion in the 2D image domains using an inpainting network, and (c) Re-projection of the inpainted multi-view images to obtain a completed 3D point cloud model of the object of interest. The three main steps are introduced in the following subsections in detail.

DOI:10.15774/PPKE.ITK.2023.007

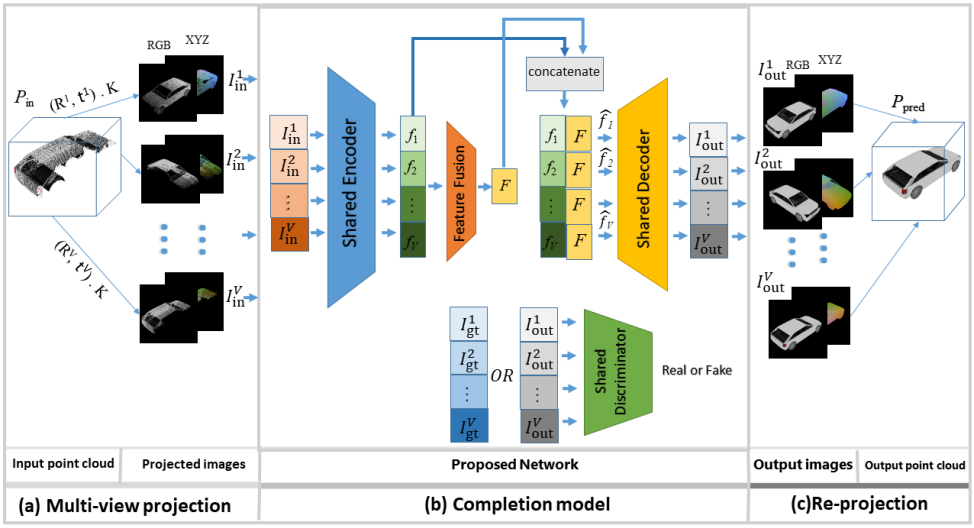


Figure 3.2: Dataflow of our algorithm: (a) Multi-view projection: incomplete point cloud measurement is represented as multi-view images, each view is recorded as a six-channel image with RGB color information and XYZ geometry information; (b) Completion model: it completes the shape and color information in the 2D image domain; (c) Re-projection: the inpainted multi-view images are reprojected into the 3D space to generate a completed 3D point cloud model of the object.

3.4.1 Multi-view 3D Representation

Let us denote by $\mathcal{P}_{\text{in}}$ the input point cloud of N_{in} points, representing a single, partially scanned scene object:

$$\mathcal{P}_{\text{in}} = \{p_n\}_{n=1}^{N_{\text{in}}},$$

where each point p_n is associated to a 6D descriptor comprising three location coordinates (XYZ) in the point cloud's local Descartes coordinate system, and three color coordinates (RGB).

In the first step, the point cloud $\mathcal{P}_{\text{in}}$ is mapped to different 2D grids from a variety of perspectives: multi-view 2D images are captured by a set of V virtual cameras located at predetermined positions around the object's 3D bounding box. Henceforward, $v \in \{1, \ldots, V\}$ refers to the index of a selected view. Assuming that the position and orientation of each virtual camera are known, the view-transformation for each camera v can be described by a R^v rotation matrix and a t^v translation matrix, which can be analytically calculated. Thereafter, the input

DOI:10.15774/PPKE.ITK.2023.007

point cloud $\mathcal{P}_{\text{in}}$ can be transformed to the reference coordinate system of camera v as follows:

$$q_n^v = R^v \cdot p_n + t^v, \quad n = 1, \ldots, N_{\text{in}}; \quad v = 1, \ldots, V \tag{3.1}$$

where q_n^v denotes vth the camera coordinates corresponding to point $p_n \in \mathcal{P}_{\text{in}}$. Next, each point is projected onto the camera plane using the virtual cameras' projection matrix K, which is determined by the intrinsic camera parameters:

$$(x_n^v, y_n^v) = K \cdot q_n^v \quad n = 1, \ldots, N_{\text{in}}; \quad v = 1, \ldots, V \tag{3.2}$$

The intrinsic settings of the cameras are adjusted to ensure that the objects of the training dataset are entirely contained within the considered image windows, while the coverage rate of the projected regions in the images is maximized. The result of Eq. (3.2) is a 2D pixel position (x_n^v, y_n^v) on the image plane of camera v. To keep the genuine 3D geometric (XYZ) and 3D color (RGB) information from in the original point cloud $\mathcal{P}_{\text{in}}$, we store the point projections in a six-channel image I_{in}^v associated with each view v. More specifically, if point p_n is projected to pixel (x_n^v, y_n^v) in view v, the values of the different image channels at the given pixel of I_{in}^v are equal to the concerning (XYZRGB) coordinates of p_n.

Note that by projecting 3D point clouds to 2D images, multiple points might be projected to the same pixel of a given image lattice. We handle this issue by sorting these points by their distances from the given camera, and we only retain the closest points, keeping only the exposed portion of the object from the camera perspective. The presence of multiple cameras from different directions will ensure that the object points visible from any viewpoint will be represented by one or multiple projected images.

On the other hand, following the above procedure, the different I_{in}^v camera images will be only sparsely filled, containing several pixels without any point projections: by these pixels we set zero values for all channels.

In summary, as shown in Figure 3.2(a), this step derives a collection of multichannel images, that can be directly processed by 2D CNN architectures (see Sec. 3.4.2). Each camera records a six-channel image, comprising geometry (XYZ channels) and color (RGB channels) information, however the geometry is directly

DOI:10.15774/PPKE.ITK.2023.007

stored in the point cloud's original Descartes coordinat system. As a result, a point projected to multiple virtual camera planes will have the same geometrical coordinates in each view image. We demonstrate later in Sec. 3.4.3 that this property is highly beneficial during the 3D point cloud re-projection phase of the process.

3.4.2 Completion Model

The second main step of the proposed approach (see Figure 3.2(b)) performs structure and color inpainting of the missing object regions in the domain of the six-channel I^v_{in} images generated from different viewpoints in the previous phase. The resulting inpainted images will later guide the generation of the final 3D point cloud in the last step (Sec. 3.4.3).

Our method uses a Generative Adversarial Network (GAN) [39] architecture, that consists of the *Generator* (G) and *Discriminator* (D) networks.

Our *Generator* implements three subsequent steps:

(i) First, we use a *shared encoder* for all views, which applies two-stage down-sampling followed by using eight residual blocks to separately encode each of the six-channel images I^v_{in} associated with view v to its own view-level latent feature cube f_v of size $64 \times 64 \times 256$, for $v \in \{1, \ldots, V\}$.

(ii) The intermediate *feature fusion* phase employs (3×3) convolutions, followed by a ReLU activation function, to fuse all the view-level features denoted by $[f_1, f_2, \ldots, f_V]$ into a single global feature cube F of the same size as each f_v. The global feature is expected to facilitate the transmission of shared characteristics between distinct viewpoints.

(iii) In the third step, we take each view-level latent feature f_v, and concatenate it to the global feature F calculated in the previous phase, obtaining a $\hat{f}_v = [f_v, F]$ extended feature cube of size of size $64 \times 64 \times 512$ for each view $v \in \{1, \ldots, V\}$. Next, the *shared decoder* processes the different view's $\hat{f}_v$ features sequentially, so that each $\hat{f}_v$ is upsampled to the size of the original I^v_{in} image using dilated convolutions, with a dilation factor of two. The output of the decoder is an inpainted I^v_{out} image, with the same size and

DOI:10.15774/PPKE.ITK.2023.007

six-channel format (i.e. XYZRGB channels) as the encoder input I^v_{in}. Let us observe, that although the decoder generates the different I^v_{out} images separately for the different views, the decoder's input features contain information from all views via the F global feature component of $\hat{f}_v$, thus cross-view fusion is implicitly implemented at this step.

Next, the six-channel I^v_{out} images predicted by the *Generator* are presented to the *Shared Discriminator*, whose task is to decide whether they are real or fake. The discriminator architecture is based on the 70×70 PatchGAN [34], which determines whether overlapping image patches of size 70×70 are real or not.

For model training, we follow a supervised approach using Ground Truth (GT) images projected from complete object models, as detailed in Sec. 3.3. In the training phase, we attempt to ensure that for each view v the six-channel image $I^v_{\text{out}} = G(I^v_{\text{in}})$ predicted by the *Generator*, becomes as similar to the GT image I^v_{gt} as feasible. The network is trained using a combined loss function consisting of six subterms: smooth $L1$ loss, adversarial loss, perceptual loss [49], style loss [48], binary cross entropy loss and Total Variation loss [117] as detailed in the following.

The *smooth* $L1$ *loss* term ℓ_{L1} is used to keep the distance low between the algorithm's six-channel image output and the corresponding (GT) target image. Smooth $L1$ loss combines the benefits of $L1$-loss and $L2$-loss by providing stable gradients when the distance is high, while it reduces the oscillations when the distance is small [86].

The adversarial loss ℓ_{adv} is also applied to all channels of the generated I^v_{out} image. It is presented as a zero-sum competition between the generator and discriminator networks so that the generator attempts to minimize the value defined by Eq. (3.3), while the discriminator attempts to increase it:

$$\ell_{\text{adv}} = \mathbb{E}_{\{I^v_{\text{gt}}\}}\big[\log\big(D(\sum_{v=0}^{V} I^v_{\text{gt}})\big)\big] + \mathbb{E}_{\{I^v_{\text{out}}\}}\big[\log\big(1 - D(G(\sum_{v=0}^{V} I^v_{\text{in}}))\big)\big] \tag{3.3}$$

The *perceptual loss* ℓ_{perc} and *style loss* ℓ_{sty} terms are calculated exclusively for the RGB color image channels to make them perceptually and stylistically more similar to the GT's coloring. The *binary cross entropy* ℓ_{Lcro} measures the

DOI:10.15774/PPKE.ITK.2023.007

difference between two binary masks defined by non-zero pixels in the output image I^v_{out} and the ground truth I^v_{gt}, respectively. Finally, the *Total Variation* (TV) loss ℓ_{TV} - which promotes spatially smooth output images - is utilized to smooth the geometric output channels. The *Generator*'s combined loss function ℓ_G is derived from the above defined six subterms, with using $\lambda_1, \ldots, \lambda_6$ regularization parameters:

$$\ell_G = \lambda_1 \ell_{L_1} + \lambda_2 \ell_{\text{adv}} + \lambda_3 \ell_{\text{perc}} + \lambda_4 \ell_{\text{sty}} + \lambda_5 \ell_{\text{Lcro}} + \lambda_6 \ell_{TV} \tag{3.4}$$

For our experiments, the following regularization hyperparameters of the loss function are used: $\lambda_1 = 50$, $\lambda_2 = 0.1$, $\lambda_3 = 250$, $\lambda_4 = 0.1$, $\lambda_5 = 0.1$, $\lambda_6 = 0.1/S$, where S refers to the image size (here $S = 256 \times 256$). The first three parameters were chosen based on [28], and the last three parameters were chosen to balance the impact of each loss. More details about the model structure, the used loss functions, and the training strategy are presented in Appendix A

3.4.3 Re-projection

As a result of the previous step, a set of inpainted six-channel images are available, which represent the projections of the object shape from various viewpoints. Each non-zero pixel in each view image encodes a 3D point in the object's coordinate system, where the geometry channels determine the given point's normalized XYZ Descartes coordinates, and the color channels define the associated RGB value. Consequently, the completed point cloud of the object can be derived in a straightforward way by re-projecting all points stored in the V different views one after another to the same 3D space (see Figure 3.2(c)). Let us observe that this process admits generating output point clouds that consist of a variable number of points. More specifically, the number of points added to the incomplete input point cloud is determined by the total number of inpainted pixels in the view images.

During this stage of the algorithm, two post-processing steps are applied for enhancing the quality of the generated point cloud. First, since we observed that re-projecting the boundary pixels of the objects from the different image views results in noisy points surrounding the 3D shape, we slightly erode the foreground

DOI:10.15774/PPKE.ITK.2023.007

regions of the inpainted images before executing the re-projection. Second, we also employ a statistical outlier filter [111] to eliminate further outliers from the final point cloud.

3.5 Experimental Results and Evaluation

We have trained and evaluated the proposed technique using our new dataset introduced in Sec. 3.3, which consists of both synthetic and real MLS object point cloud samples. In this section, we present a detailed quantitative and qualitative performance analysis, and comparison versus various state-of-the-art reference methods.

3.5.1 Evaluation Methodology

Since the output of the proposed method is a colored point cloud, we separately evaluate the quality of object geometry prediction and the realistic nature of RGB color estimation.

We perform the *geometric assessment* of the object shapes against various state-of-the-art 3D point cloud completion approaches in the point cloud's local 3D coordinate system, so that we compare each predicted point cloud ($\mathcal{P}_{\text{pred}}$) to the corresponding ground-truth point cloud ($\mathcal{P}_{\text{gt}}$). There is no good numerical metric to evaluate such completion 3D point completion results due to the existence of many possible solutions, Nevertheless for quantitative analysis, we rely on the Chamfer Distance (CD) and the F1-score, which are two frequently used measures for comparing the similarity of two sets of points [100, 102].The Chamfer Distance is calculated by searching for the closest point pairs between the predicted point cloud $\mathcal{P}_{\text{pred}}$ and the corresponding ground truth $\mathcal{P}_{\text{gt}}$ in two directions, as described by Eq. (3.5).

$$D_{\text{CD}}(\mathcal{P}_{\text{pred}}, \mathcal{P}_{\text{gt}}) = \frac{1}{2N_{\text{pred}}} \sum_{p \in \mathcal{P}_{\text{pred}}} \min_{q \in \mathcal{P}_{\text{gt}}} ||p - q||_2^2 + \frac{1}{2N_{\text{gt}}} \sum_{q \in \mathcal{P}_{\text{gt}}} \min_{p \in \mathcal{P}_{\text{pred}}} ||p - q||_2^2 \quad (3.5)$$

where N_{pred} and N_{gt} denote the number of points in $\mathcal{P}_{\text{pred}}$ and in $\mathcal{P}_{\text{gt}}$, respectively; and $||p - q||_2^2$ stands for the Euclidean distance between the locations of points p and q.

DOI:10.15774/PPKE.ITK.2023.007

For considering an alternative geometric accuracy measure of point cloud completion, we also calculate the F1-score, which considers as a match any pair of points whose distance is less than a given distance threshold τ. The F1-score (F1) as a function of τ is computed as follows:

$$\mathrm{F1}(\tau) = \frac{2 \cdot \mathrm{Pr}_\tau \cdot \mathrm{Rc}_\tau}{\mathrm{Pr}_\tau + \mathrm{Rc}_\tau} \tag{3.6}$$

where Pr_τ and Rc_τ stand for Precision and Recall, for a given threshold τ:

$$\mathrm{Pr}_\tau = \frac{1}{N_{\mathrm{pred}}} \sum_{p \in \mathcal{P}_{\mathrm{pred}}} \big[\min_{q \in \mathcal{P}_{\mathrm{gt}}} ||p - q|| < \tau \big] \tag{3.7}$$

$$\mathrm{Rc}_\tau = \frac{1}{N_{\mathrm{gt}}} \sum_{q \in \mathcal{P}_{\mathrm{gt}}} \big[\min_{p \in \mathcal{P}_{\mathrm{pred}}} ||p - q|| < \tau \big], \tag{3.8}$$

where based on [102] we adopted the threshold $\tau = 0,01$, commonly used for normalized point coordinates.

In contrast to 3D geometry analysis, we can mainly rely on qualitative tests for evaluating the *color prediction* of the proposed method, which can be performed either in the 3D point cloud space (Sec. 3.5.3), or in the 2D image domain of the individual views (Sec. 3.6). Since the considered reference methods only deal with geometry completion, they cannot be involved here in the comparative tests.

Generally, the assessment of RGB image inpainting is regarded as a highly subjective process, where we cannot find any straightforward numerical metric for evaluation of the results [27]. However, there are a number of standard evaluation metrics used in literature which we also adopt here, including the Peak Signal-to-Noise Ratio (PSNR), the Structural Similarity Index (SSIM) [78], and the Relative $L1$ error [27]. We will calculate the later measures in the ablation experiments (Sec. 3.6), where we analyze their dependency on various settings of the proposed model.

3.5.2 Comparative Evaluation on Synthetic Data

In the first part of the evaluation process, we use the synthetic dataset presented in Sec. 3.3.1 for testing our method, and for comparing it to four recent state-of-the-art 3D point cloud completion algorithms: TopNet [118], GRNet [102], PCN [112] and VPC-Net [113].

DOI:10.15774/PPKE.ITK.2023.007

For providing a fair comparison, a number of careful considerations should be taken. First, while our proposed model is able to process and generate object point clouds that consist of a variable number of points, the above mentioned reference methods are limited to produce point cloud outputs with a fixed size of 16,384 points. For this reason, apart from using the high density *fine GT* described in Sec. 3.3.1 for training our proposed MVPCC-Net, we also generated a downsampled version of each object's ground truth point cloud using the Farthest Point Sampling technique [116]. The downsampled point clouds - referred henceforward as *coarse GT* samples – consist of exactly 16,384 points, thus they can be used for training the reference approaches.

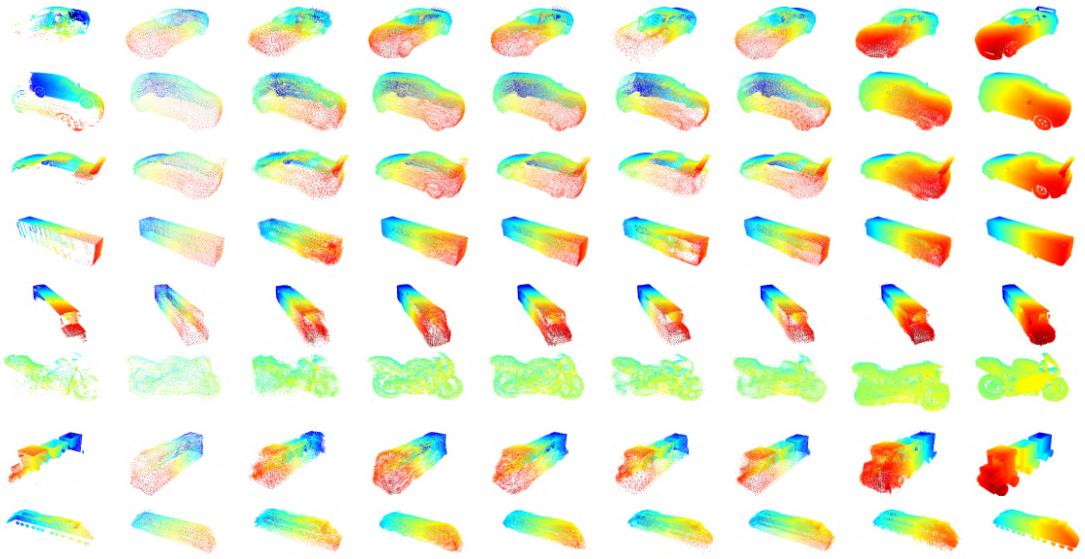
Second, since the reference methods exclusively deal with geometry information, in these experiments we also limited our model's training and evaluation only considering the XYZ channels. For this reason, during this comparison, we ignored the RGB channels, and we used the perceptual loss and the style loss terms exceptionally for the XYZ channels (with a learning rate of 10^{-4}).

The qualities of the completed object shapes are characterized by the geometric evaluation parameters defined in Section 3.5.1. Comparative results are provided in Table 3.1 and Table 3.2, using as quality measures the mean Chamfer Distance and the mean F1-score over the test set, respectively. Here we used the proposed MVPCC-Net model with four views ($V = 4$), which proved to be the most efficient settings in our ablation experiments, as detailed later in Sec. 3.6.1. To demonstrate that the relative performances of the considered techniques are fairly stable regardless of how detailed reference point clouds are used as ground truth, we calculated the geometric evaluation metrics for both the *fine GT* and *coarse GT* samples, which results are shown side by side in Tables 3.1 and 3.2. Given that the denser *fine GT* samples have more points than the *coarse GT* objects, it is evident that the CD error rates associated with the *fine GT* are generally lower (and the F1-scores are higher) than the results connected to the *coarse GT* regarding each method.

The superiority of our method's *coarse GT*-related results demonstrates its efficacy in recreating the global structure of the objects under study (i.e, size and shape of their main components). On the other hand, our MVPCC-Net approach is also superior at reconstructing local features and fine geometric structures

DOI:10.15774/PPKE.ITK.2023.007

appearing only in denser point clouds: This observation is supported by our efficient *fine GT*-related numerical rates, and also by comparative qualitative results shown in Figure 3.3, which displays for various sample objects the input partial point clouds, the results of all considered techniques, and the *fine GT* as a reference.



(a) Input (b) TopNet (c) GRNet (d) PCN (e) VPC (f) Seed (g) Snow (h) Ours (i) GT

Figure 3.3: Qualitative results on the synthetic dataset, where we present the input partial point cloud, the results of the references methods, PCN, TopNet, GRNet, VPC-Net, SeedFormer, SnowflakeNet,Our results, and the GT stands for the complete 3D object.

3.5.3 Comparative Evaluation on Real MLS data

In the second phase of the experiments, we evaluate the performance of the proposed approach and the reference techniques on real MLS point cloud samples, presented in Sec. 3.3.2. Due to the lack of sufficient number of (complete) training samples among the available MLS object point clouds, we use here the MVPCC-Net with weight parameters trained on synthetic data in the previous test phase.

However, by replacing synthetic point cloud inputs with real MLS measurements we have to deal with a practical issue: While in synthetic datasets stan-

DOI:10.15774/PPKE.ITK.2023.007

Table 3.1: Evaluation of our algorithm's geometric accuracy compared to the state-of-the-art algorithms on the synthetic dataset using **Chamfer Distance** ($\times 10^{-3}$) $\downarrow$. By each object category, the first column refers to the comparison with the coarse GT (16384 points), while the second column represents the comparison results to the fine GT, the best results are highlighted in bold.

Method	Buses 1440 samples		Cars 4680 samples		Motorcycles 180 samples		Trains 480 samples		Overall 6780 samples	
	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT
TopNet [118]	7.254	6.514	10.724	8.189	19.362	18.976	10.264	9.919	10.184	8.242
GRNeT [102]	7.776	7.153	8.596	7.807	9.077	8.675	7.149	7.544	8.332	7.672
PCN [112]	5.870	5.130	7.549	6.107	13.326	12.869	7.257	6.823	7.325	6.129
VPC-Net [113]	**4.688**	**3.965**	6.666	5.604	9.945	9.531	5.935	5.512	6.281	5.353
SeedFormer [109]	6.592	5.922	7.609	6.685	8.524	8.074	6.737	6.243	7.355	6.528
SnowflakeNet [108]	4.94	4.202	6.101	5.274	**8.232**	**7.883**	**5.516**	**5.102**	5.869	5.103
Ours	5.113	4.333	**5.813**	**4.702**	8.448	7.928	5.853	5.309	**5.737**	**4.752**

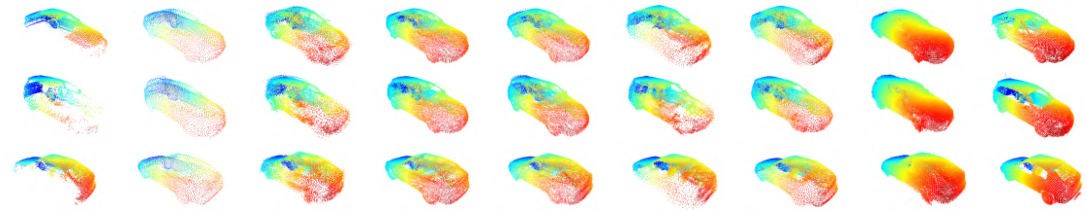
Table 3.2: Evaluation of our algorithm's geometric accuracy compared to the state-of-the-art algorithms on the synthetic testing dataset, **F1-score** (%) $\uparrow$, The first number is a comparison with the coarse GT (16384 points), while the second number represents a comparison with the fine GT, the best results are highlighted in bold.

Method	Buses 1440 samples		Cars 4680 samples		Motorcycles 180 samples		Trains 480 samples		Overall 6780 samples	
	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT
TopNet [118]	82.06	82.85	69.56	73.31	24.85	25.95	64.70	64.91	70.68	73.48
GRNeT [102]	76.73	78.42	72.25	75.37	69.20	70.75	77.68	78.70	73.50	76.13
PCN [112]	89.74	90.96	83.79	86.99	49.35	51.17	80.49	81.29	83.91	86.47
VPC-Net [113]	**95.50**	**96.22**	87.63	89.57	66.89	68.23	87.56	88.12	88.75	90.31
SeedFormer [109]	85.62	86.80	81.03	84.28	75.22	76.86	84.71	85.59	82.11	84.71
SnowflakeNet [108]	94.01	94.83	88.34	90.23	74.85	75.89	**89.79**	**90.12**	89.28	90.77
Ours	92.64	93.70	**89.61**	**91.89**	**76.46**	**77.58**	88.22	88.99	**89.81**	**91.68**

Table 3.3: Evaluation of our algorithm's geometric accuracy compared to the state-of-the-art algorithms on the real MLS object samples, using **Chamfer Distance (CD,** $\times 10^{-3}$)$\downarrow$, and **F1-score** (%)$\uparrow$

Method	$\downarrow$**CD** ($\times 10^{-3}$)	$\uparrow$**F1-score** (%) $\tau = 0.01$	$\uparrow$**F1-score** (%) $\tau = 0.005$
TopNet [118]	12.725	52.87	14.10
GRNet [102]	10.185	67.73	31.47
PCN [112]	10.410	67.25	31.72
VPC-Net [113]	7.563	**80.77**	51.62
SeedFormer [109]	8.248	77.61	36.11
SnowflakeNet [108]	8.487	79,23	46.28
Ours	**6.962**	80.75	**58.28**

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) TopNet (c) GRNet (d) PCN (e) VPC (f) Seed (g) Snow (h) Ours (i) GT

Figure 3.4: Qualitative results on Real-world dataset, where we present the input partial point cloud, and the results of the references methods, PCN, TopNet, VPC-Net, SeedFormer, SnowflakeNet, Our results, and the GT stands for the complete 3D object

dardized objects alignments can be ensured (e.g. the forward direction of vehicles is equal to one of the axes in their local Descartes coordinate system), object fragments extracted from real MLS data may have arbitrary orientations. For this reason, we proposed an additional network component for re-aligning the input MLS point cloud segments, enabling the direct application of our model for real measurements without the need for any additional fine-tuning, although the model had been trained on purely synthetic data. For this purpose, we adopted a spatial transformer network (STN) [87], which provides as output a 3×3 rotation matrix, that can be used to rotate an input point cloud sample around its vertical axis, so that it is transformed into a canonical orientation defined by ground truth samples in the training phase. The STN network segment was trained independently of the other components of the MVPCC-Net with synthetic vehicle shape models (described in Sec. 3.3.1). The input of this training phase consists of partial object point clouds rotated with randomly chosen angles around their vertical axes, and its reference outputs are the same point cloud segments with standard orientations, facing in the direction of the x-axes of their local coordinate system. The aim of this training phase is to learn the transform providing an appropriate rotation matrix for new object samples with arbitrary initial orientations.

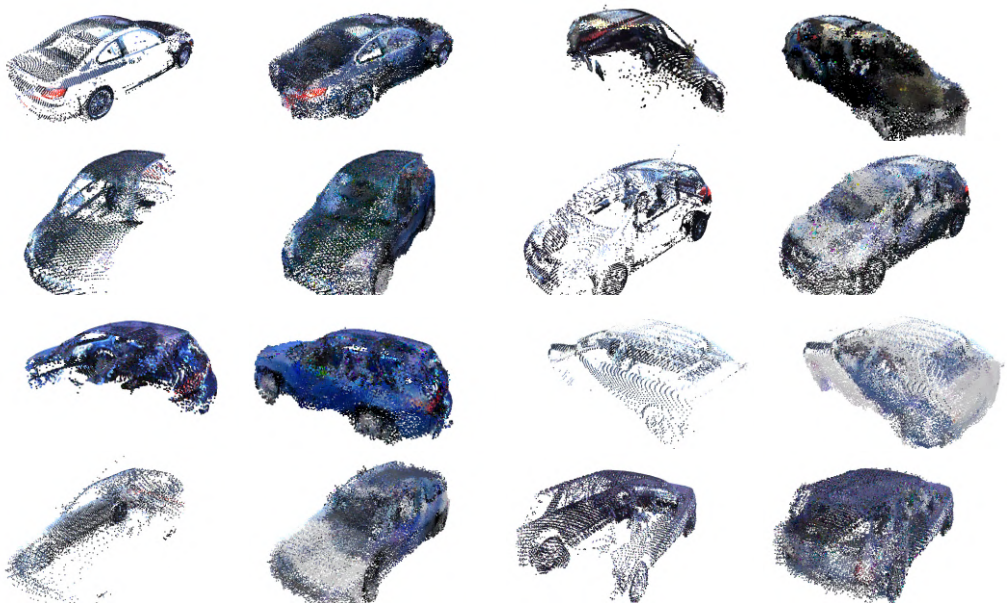
After orientation adjustment, we tested the proposed MVPCC-Net and the reference techniques on real MLS object samples, so that neither our model nor other models were fine-tuned for the MLS measurements. As mentioned in Sec. 3.3.2, we have 54 completely scanned vehicle models in our MLS dataset which can be used for quantitative evaluation in a similar manner to the synthetic data,

DOI:10.15774/PPKE.ITK.2023.007

while the remaining 370 MLS objects enable us to perform a qualitative study on a widely diverse set of real vehicle point cloud measurements. Numerical and qualitative evaluation results are shown in Table 3.3 and in Fig. 3.4 respectively, which confirm that proposed MVPCC-Net outperforms the reference methods, and it is capable of efficiently processing MLS measurements. As demonstrated in Table 3.3, our model is clearly better than the other techniques regarding the Chamfer Distance (2nd column of the table), however, the F-scores of VPC-Net and MVPCC-Net are nearly identical with the standard distance threshold settings $\tau = 0.01$ (see the 3rd column). For this reason, we also calculated the F-score values with a more strict threshold selection $\tau = 0.005$, which choice yielded already a clear advantage of the proposed method (4th column). We can also conclude based on Figure 3.4 that our method is more capable of producing dense and finely detailed point clouds which property is highly advantageous by processing dense MLS data.

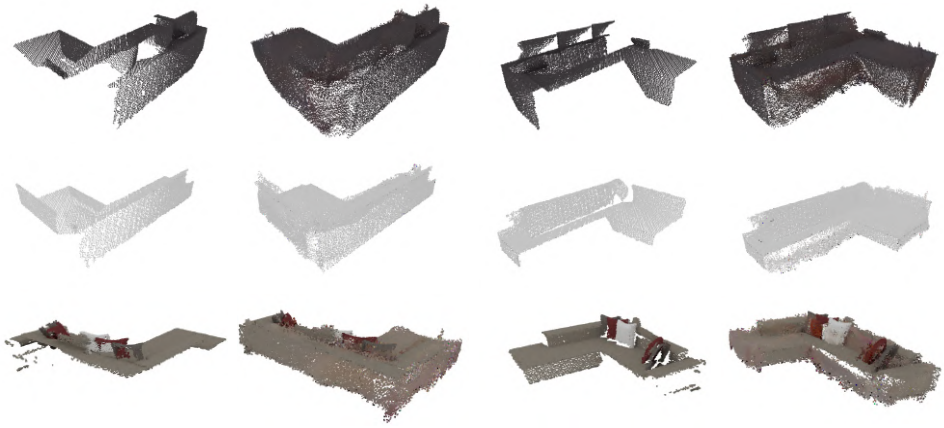
Although in Figure 3.3 and 3.4 we only visualized the geometry of the generated object point clouds, the MVPCC-Net method can also estimate RGB color value for each point as described in Sec. 3.4. For selected MLS objects, the input-output pairs of the proposed model are shown as colored point clouds in Figure 3.5. These qualitative results confirm, that in many different situations, both the vehicle's global shape and its color schema can be predicted in a realistic manner, and the generated object segments fit well with the captured partial MLS measurements both in geometry and in color. Exploiting that vehicles have in general symmetric shapes, many components such as tail lamps or door textures are efficiently transformed from one side to the other one. On the other hand, the proposed method is also successful in predicting completely missing frontal or back regions, where symmetry-based shape completion cannot be performed. As a limitation, some erroneously textured areas may appear in different vehicle regions, which phenomenon is in part a consequence of texture errors in the raw MLS measurements.

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) Our result (c) Input (d) Our result

Figure 3.5: Results of the proposed method with MLS point clouds acquired using a Riegl VMX Mobile Laser Scanner.



(a) Input#1 (b) Output#1 (c) Input#2 (d) Output#2

Figure 3.6: Results of the proposed approach on the Shapenet datasetâs asymmetrical sofa objects. For the same item, each row shows the input and our method's output from two viewing angles.

DOI:10.15774/PPKE.ITK.2023.007

3.5.4 Completion Results for Asymmetric Objects

While the objects investigated in the previous sections have symmetric geometry, the proposed method is not limited to dealing with such object shapes. Since asymmetrical objects are usually considered more challenging for shape completion methods, we also investigated how our MVPCC-Net is able to complete partial point clouds of sofa objects. The sofa dataset was generated from ShapeNet in a similar manner as described in Sec. 3.3.1 for vehicles. For this experiment, we used 2930 training objects and 43 test samples for validation.

The results of shape completion for three sample sofa objects are displayed in Figure 3.6, where each row shows the same input point cloud from two different perspectives, alongside the predicted object shapes. The results confirm at a proof-of-concept level that our model can also generate realistic results for asymmetrically shaped items.

3.5.5 Computational Time

In this section, we present an experimental study about the execution time of each individual step of the proposed algorithm. Our experiments were performed on a personal computer (PC) with AMD Ryzen 9 5900X 12-Core Processor, 32-GB RAM and a NVIDIA GeForce RTX 3060 Ti GPU. We run the proposed MVPCC-Net model with four views ($V = 4$) on 216 real MLS samples described in Sec. 3.3.2. The completion model part was executed on the GPU, while the remaining steps have been implemented for CPU. Table 3.4 presents the measured average execution time per object in milliseconds (ms), for the consecutive steps of our proposed shape prediction workflow. As the results show, with our method the mean total computing time for a sample object was around 55 ms. As for the reference techniques, we tested two methods [109, 108] on the same PC configuration as our model, and their execution time varied between 28–48 ms for the different MLS object samples. Note that as shown in Table 3.4, almost half the processing efforts in our model correspond to the statistical outlier filter [111], which step can be significantly accelerated further by using a GPU-based implementation [105]. The remaining considered reference methods were

DOI:10.15774/PPKE.ITK.2023.007

tested on slightly different hardware platforms, nevertheless the experienced running times were largely similar to our model. We should also emphasize, that in the targeted MLS data processing application, real-time operation is usually not a strict requirement, thus we regard the speed of our algorithm adequate for applicability.

Table 3.4: Execution Time for each step of our algorithm in milliseconds (ms). The time is measured on an NVIDIA GeForce RTX 3060 Ti GPU with batch size of 1.

NO. views	Projection	Completion model			Re-projection	Filter	Overall
		Encoder	Fusion	Decoder			
4 views	6.311	10.493	0.132	4.514	7.933	25.621	55.004

3.6 Ablation Studies

The proposed MVPCC-Net model consists of various components and it contains a number of hyperparameters which influence its performance. To support our decisions regarding model design, and provide more information about parameter settings, we present ablation studies in this section, wherein each experiment we train our model with the car shape training dataset until the convergence (using a learning rate of 10^{-4}), and we validate the model performance on the corresponding test set.

3.6.1 Optimal Settings of the Number of Views

In this section, we examine further the performance of the proposed method on colored incomplete point cloud inputs, and study the impact of changing the number of views with respect to the geometry and coloring of the completed point cloud output.

Figure 3.7 displays the results per view as images for a sample vehicle object, using a total of $V = 10$ views in the proposed model. Each row displays the input, our model's output, and the ground truth for the corresponding view. As shown in the first three columns of Figure 3.7, the results depict realistic color predictions, with color characteristics matching on both sides of the vehicle (see rows 9 and 10), while realistic tail lamps can be observed on the vehicle's rear end (see in rows 1, 2, and 5), and the texturing style of the wheels also looks real

DOI:10.15774/PPKE.ITK.2023.007

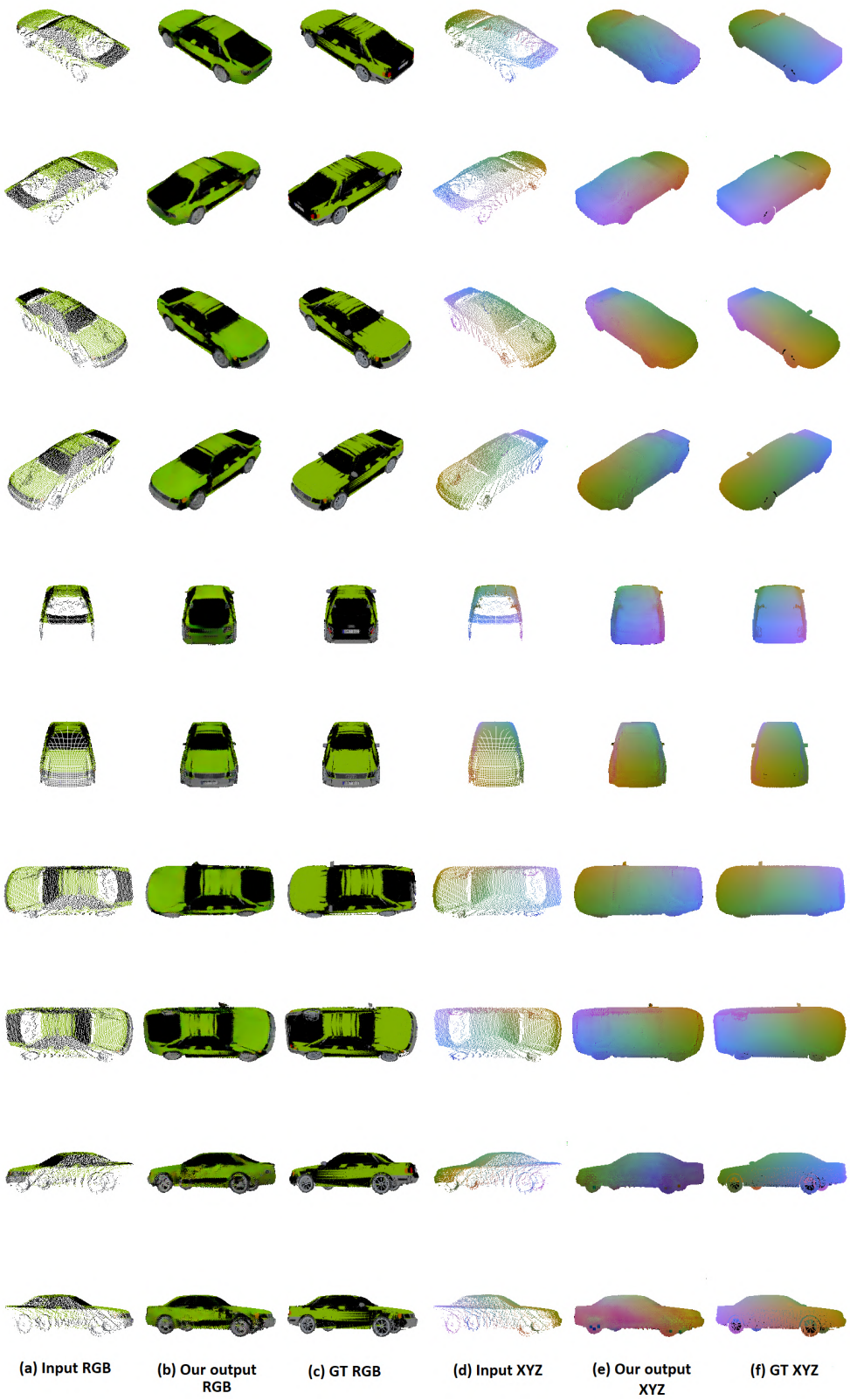


Figure 3.7: Result of the XYZRGB channels from ten views for a partial input point cloud. For each row: (a) RGB input, (b) RGB output, (c) RGB GT, (d) XYZ input, (e) XYZ output, (f) XYZ GT.

DOI:10.15774/PPKE.ITK.2023.007

Table 3.5: Effects of view aggregation. Results on a test set of synthetic data (car shape), **PSNR**↑, **SSIM**↑, **MAE**↓ on color channels, **Chamfer Distance** (10^{-3}) ↓, **F1-score** (%) ↑ on geometric accuracy.

No. views	**2D Color Images**			**3D Geometry**	
	PSNR ↑	**SSIM** ↑	**MAE** ↓	**CD** ($\times 10^{-3}$) ↓	**F1-score** (%) ↑
3 views	25.03	0.9013	0.0860	6.937	81.31
4 views	24.95	0.8969	0.0892	**6.585**	**83.07**
5 views	22.67	0.7803	0.1768	7.45	80.67
6 views	21.44	0.7063	0.2340	8.28	76.03
7 views	20.11	0.6367	0.2768	11.214	65.08
8 views	19.42	0.5908	0.2968	9.19	73.16
10 views	18.40	0.5342	0.3689	13.697	63.54

(see rows 1-4). The color images of the last three columns of Figure 3.7 visualize the shape geometry, so that the normalized XYZ coordinate values stored in the geometry channels are displayed as *pseudo geometry images*. These results demonstrate that the model is able to accurately predict the shape of the car, and even details such as predicted wheels and mirrors are visible in the output.

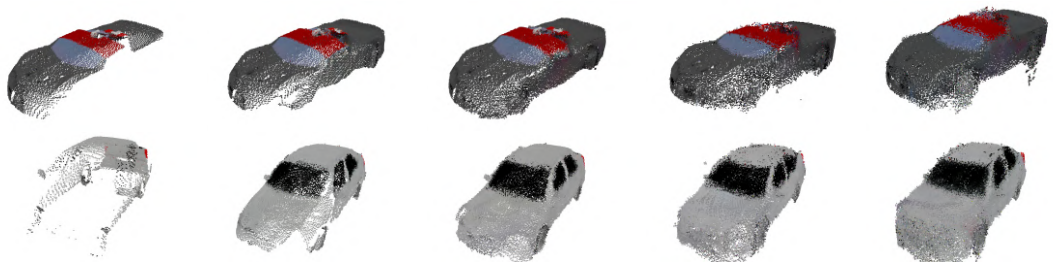
In the next phase of the analysis, we trained and tested the proposed model with increasing the number of views one by one from three to ten. The views are chosen in the order indicated by the consecutive rows in Figure 3.7, for example, the four-view configuration uses the images shown in the first four rows.

Table 3.5 presents quantitative results of our model with different numbers of views for the whole car shape test dataset, using various metrics for geometry and color evaluation defined in Sec. 3.5.1. These experiments confirm that we can obtain the most accurate 3D geometry prediction by using in total four different viewpoints (following the settings of the first four views in Figure 3.7). While additional views yield a higher point density output point cloud, they also cause additional noisy points surrounding the object shape, reducing the accuracy of shape estimation. This phenomenon is shown in Table 3.5, where the Chamfer Distance increases (and the F-score decreases) in cases of more than four viewpoints.

We can observe similar tendencies regarding color estimation based on Table 3.5 and Figure 3.8: Using more than four views yields weaker performance rates, and we can also see noisier texture by six or eight views via visual verification

DOI:10.15774/PPKE.ITK.2023.007

(Figure 3.8(d),(e)). Note that although according to Table 3.5 the best color-based evaluation rates are obtained by three views, such a configuration still generates incomplete object shapes (Figure 3.8(b)), leading to larger geometric error values (see CD and F1-score rates in Table 3.5).



(a) Input (b) 3 views (c) 4 views (d) 6 views (e) 8 views

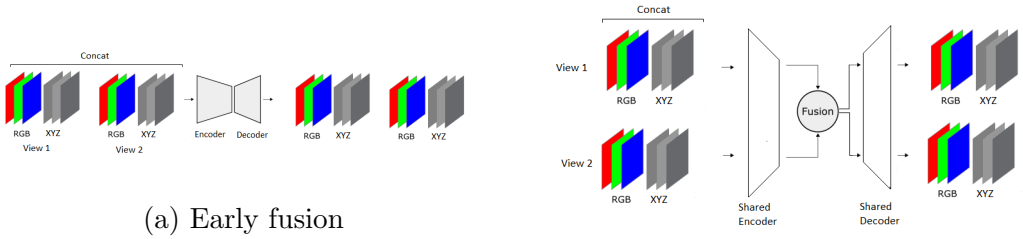
Figure 3.8: Effects of view aggregation. Results of the proposed method with different number of views: (a) Input, our model results using three, four, six, and eight views are shown in (b)-(e) respectively.

3.6.2 View Fusion Strategies

In this section, we compare the view fusion strategy introduced in Sec. 3.4.2 to a straightforward early fusion technique used as baseline model, where the I_{in}^{v} images of all v views are concatenated into a single input data cube for the encoder, while the decoder generates all output views in one step. On the contrary, our method implements a late fusion approach, which generates first a separate feature representation f_v for each view v by a shared encoder, thereafter a *Feature Fusion* network component generates a global feature F from the view-level features which is used by a shared decoder to produce the inpainted images per view in a sequential process. The block diagrams of the above defined early and late fusion approaches are shown in Figure 3.9.

Next, we conducted experiments for comparing the efficiency of the early fusion to the late fusion strategies in the MVPCC-Net model. As input, we considered on the one hand measurements with color and geometry channels (XYZRGB), and on the other hand, pure shape data containing geometry channels only (XYZ). The obtained results regarding 3D geometric accuracy over the car shape set are presented in Table 3.6, while Figure 3.10 shows the results of

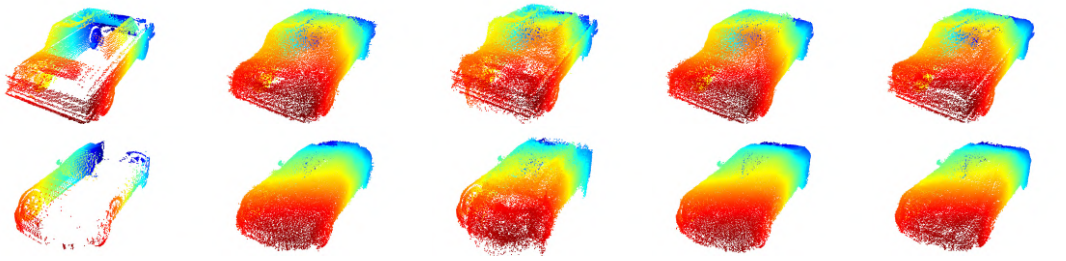
DOI:10.15774/PPKE.ITK.2023.007



(a) Early fusion

(b) Late fusion

Figure 3.9: Types of fusions, (a) Early fusion method, (b) Late fusion method.



(a) Input (b) Early (XYZ) (c) Early XYZRGB (d) Late XYZ (e) Late XYZRGB

Figure 3.10: Effects of Fusion strategies. Results of the proposed method with different fusion strategies: (a) Input, our model results using (b) early fusion method on geometry channels, (c) early fusion method on color and geometry channels, (d) late fusion method on geometry channels, (e) late fusion method on color and geometry channels

the different fusion strategies for two sample objects, where we present the point cloud results without displaying RGB color for enabling better visual comparison of the object geometries. The quantitative and qualitative results confirm that the proposed late fusion approach outperforms the early fusion baseline technique for both types of input data (XYZ and XYZRGB). Regarding the geometric parameters, the models purely considering XYZ channels yield in general better results, which fact indicates that generating colored point clouds comes with some degradation of the shape geometry. On the other hand, adding RGB information to the early fusion approach yields a significantly larger reduction of the geometric accuracy, than by using the late fusion technique, where we can only observe a slight negative effect on the accuracy of the predicted object shape.

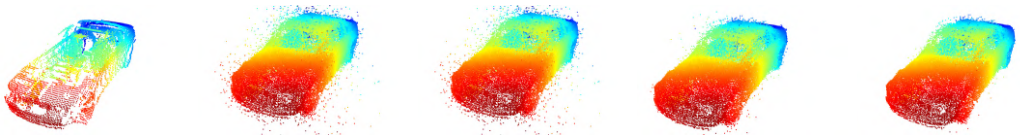
DOI:10.15774/PPKE.ITK.2023.007

Table 3.6: Effect of various fusion strategies, Results on a testing set of synthetic data (car shape), **Chamfer Distance** ($\times 10^{-3}$) $\downarrow$, **F1-score** (%) $\uparrow$ on geometric accuracy.

Fusion method	**CD** ($\times 10^{-3}$) $\downarrow$	**F1-score** (%) $\uparrow$
Early fusion (XYZ)	5.735	87.33
Early fusion (XYZRGB)	11.792	69.29
Late fusion (XYZ)	**5.315**	**89.04**
Late fusion (XYZRGB)	6.585	83.07

3.6.3 Re-projection Filter Parameters

This section demonstrates the significance of applying the erosion and outlier filters as post processing steps of the view re-projection phase (Sec. 3.4.3), and we also justify here the filters' parameter selection strategy.



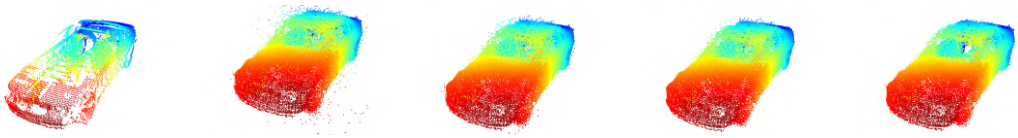
(a) Input (b) Not used (c) 1×1 (d) 3×3 (e) 5×5

Figure 3.11: Analysis of the effect of using the erosion operation in post-processing: (a) input point cloud, (b) result without using the erosion operation, (c)-(e) results using erosion operation with kernel sizes 1×1, 3×3 and 5×5, respectively.

In the following experiment, we test the proposed MVPCC-Net model with four views, and purely geometric (XYZ) input data, so that we calculate the geometric CD and F1-score values for the output point clouds provided by the re-projection step with various filter parameters.

We start with the analysis of using the erosion operator, where we evaluate the results in four configurations: first without using the erosion step, thereafter with implementing the erosion with square shaped kernels of sizes 1×1, 3×3, and 5×5, respectively. Based on the results presented in Table 3.7 and Figure 3.11, we can conclude that by applying the erosion operator we can get a smoothed output point cloud, where several noisy points around the investigated object's shape are removed. Since by increasing the kernel size to 5×5, we can often observe the removal of some real object components, we decided to use the 3×3 kernel, which choice also corresponds to the best evaluation rates in Table 3.7.

DOI:10.15774/PPKE.ITK.2023.007



(a) Input (b) No SOF (c) σ=8 (d) σ=5 (e) σ=2

Figure 3.12: Analysis of the effect of using Statistical Outlier Filter (SOF) as a post-processing step: (a) input point cloud, (b) results without using SOF, (c)-(e) results using SOF with standard deviation rates $\sigma = 8$, 5 and 2, respectively.

Table 3.7: Analysis of the effect of using erosion operation as a post-processing step with different kernel sizes

Kernel size	**CD** ($\times 10^{-3}$) $\downarrow$	**F1-score** (%) $\uparrow$
Not used	5.583	88.69
(1×1)	5.335	88.89
(3×3)	**5.315**	**89.04**
(5×5)	5.371	88.84

Next, we demonstrate the significance of the Statistical Outlier Filter (SOF). Figure 3.12(b) shows the results without SOF, while Figure 3.12(c)-(e) display the output point clouds using the statistical outlier filter with its standard deviation parameter σ equal to 8, 5, and 2, respectively. We can see that the filter can remove several noisy points around the object boundary, while the number of the filtered points increases by decreasing the standard deviation rate.

Applying the statistical outlier filter as the last step of our algorithm has a significant impact on the geometric accuracy of the final results: according to Table 3.8, both the CD ad F1-score rates can be improved by around 2%. We can also observe that using SOF with a standard deviation of $\sigma = 8$ decreases the CD by removing a large number of noisy points, whereas selecting a $\sigma = 5$ improves the F1-score while it maintains the CD nearly unchanged compared to the case of $\sigma = 8$. Using a standard deviation of $\sigma = 2$ we experience a minor rise in the F1-score at the expense of a significant decrease in CD. Based on the above experiments, in our final model we applied the statistical outlier filter with a standard deviation of $\sigma = 5$ in order to obtain a reasonable balance between the CD score and the F1-score evaluation rates.

DOI:10.15774/PPKE.ITK.2023.007

Table 3.8: Analysis of the effect of using Statistical Outlier Filter (SOF) as a post-processing step with different standard deviation (σ) parameters

Settings	**CD** ($\times 10^{-3}$) $\downarrow$	**F1-score** (%) $\uparrow$
No SOF	7.076	86.86
$\sigma = 8$	**5.311**	88.99
$\sigma = 5$	5.315	89.04
$\sigma = 2$	5.349	**89.05**

3.7 Conclusion of the Chapter

In this chapter, we proposed a novel method for completing colored 3D point clouds representing various incomplete object shapes. We focus on generating models of street objects based on Mobile Laser Scanning measurements, where a key requirement is to keep the high detailness of the input data. Our method generates first multiple projections of the incomplete input point cloud by virtual cameras positioned around the object of interest. Thereafter the sparsely filled multichannel view images are completed in the 2D domain, and they are reassembled in the 3D space, resulting in dense point clouds of the whole object. It has been demonstrated by quantitative and qualitative experiments both on synthetic and on real-world MLS data that the new method is applicable and it outperforms various state-of-the-art techniques in terms of geometric shape accuracy, realistic RGB coloring, and preserving high resolution.

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Chapter 4

Conclusions of the Thesis

This dissertation introduces novel approaches for inpainting occluded regions in 2D images and 3D structures in real machine perception and computer vision applications. We studied two different research problems: inapinting and segmenting wall images that may include occluded areas, and completing the 3D shape of partially Lidar-scanned objects. Models based on deep convolutional neural networks and a novel data format have been developed to overcome the presented difficulties. It has been shown quantitatively and qualitatively that the proposed approaches outperform state-of-the-art techniques, as well as performing properly in real world applications.

4.1 New Scientific Results

1. Thesis: I have proposed a novel masonry wall image analysis and virtual structure recovery technique. The introduced approach automatically segments the wall structure and inpaints possible wall segments in the observed occluded/damaged regions. I have experimentally demonstrated that the proposed technique outperforms recently published models with the same objectives in terms of various wall structure and visual color metrics.

Published in [1][3][4][6]

This thesis deals with three selected tasks: First robust wall segmentation algorithm is performed to separate various sorts of structural elements (stones, bricks, ashlar, etc.) from the mortar regions, and it can also detect the occluded

DOI:10.15774/PPKE.ITK.2023.007

and damaged wall regions. The second task is predicting and visualizing potential wall segments in the occluded/damaged wall regions. In the third task, a new style transfer technique is proposed between two wall images by filling or modifying the texture style of one wall based on another wall.

All of the aforementioned procedures are publicly available for testing on the following user-friendly website: `http://imgproc.mplab.sztaki.hu/masonrydemo`

1.1. I have developed a technique for separating the bricks from the mortar in masonry wall images and obtaining accurate brick structures. The proposed method uses the U-Net-based delineation output as robust markers for the Watershed algorithm. I have shown the importance of employing the marker-based Watershed process rather than a basic connected component analysis (CCA) approach. Moreover, I have experimentally demonstrated that the proposed technique surpasses the most recent wall segmentation techniques.

Several segmentation approaches exist in the literature [61, 65, 67]; however, the majority of them solely focus on the morphological analysis of quasi-periodic masonry walls, where the geometry of masonry courses follows horizontal rows, a condition that does not hold very often, particularly for ancient walls. I have designed a method that is adaptable to a large variety of wall structures, including both historic wall structures and modern building facades. I have shown that the proposed approach significantly surpasses earlier available solutions, and it is largely robust against various noise effects, different illumination conditions, changes in viewpoints, and varying masonry types. In addition, I have shown, using both quantitative and qualitative experiments, that the proposed technique can identify a wide variety of possible occluding objects with high accuracy.

To train and test the proposed network, I have created a new annotated dataset based on 532 different wall images. I have made the dataset publicly available in the following website:

`http://mplab.sztaki.hu/geocomp/masonryWallAnalysis`

1.2. I have proposed a novel blind masonry wall image inpainting technique, performing the automatic detection and virtual completion of occluded or damaged wall regions. The proposed method works in an end-to-end manner, starting with a segmentation step that detects the occluded regions and the wall structure in the visible areas, it then proceeds by two consecutive inpainting stages: wall feature completion, and color image completion. I have

DOI:10.15774/PPKE.ITK.2023.007

demonstrated the advantages of using domain specific semantic segmentation information (mortar-brick features) over low feature information (such as Canny-based edge information) as a preprocessing step of the inpainting stage for obtaining realistic wall structures in the occluded areas. Moreover, I have experimentally shown that the results of the proposed technique significantly suppress the state-of-the-art inpainting algorithms in terms of FID-score and human visual judgment for masonry wall image inpainting applications.

The main goal of the proposed algorithm is to efficiently complete missing wall regions with realistic mortar-brick structure and color information. The proposed automatic method works in an end-to-end manner with a U-Net based model for detecting of the occluding segments, followed by two Generative Adversarial Networks (GANs) applied consecutively. The first GAN utilizes as input the segmentation results presented in Thesis 1.1, and completes the missing/broken brick and mortar segments yielding a complete wall structure mask with a connected mortar network. Thereafter, the second GAN estimates the RGB color values of the pixels in the predicted mortar-brick regions. I have demonstrated, through quantitative and qualitative comparisons, that the method is superior to other state-of-the-art techniques for inpainting realistic wall structures and color textures in terms of FID-score and human visual judgment.

1.3. I have proposed a new technique for style transfer between two different walls. The algorithm replaces the coloring style of a wall image, with another wall's style, while maintaining the wall's original structural integrity. I have provided a comprehensive qualitative evaluation to demonstrate the benefits of our approach in wall-to-wall style transfer.

I have modified the second GAN proposed in Thesis 1.2 and adapted it for the wall image style transfer tasks in order to transfer the texture and color style from one image to another wall structure. This approach requires two images as inputs: the first one is the content image which is a color wall image or a binary image for the wall structure, and the second image is the style image which is a different wall image. The goal is to create a new image that incorporates both the structure of the content image and the texture style of the style image. I have demonstrated, through a number of qualitative experiments, that the proposed method is significantly robust under different circumstances, in various

DOI:10.15774/PPKE.ITK.2023.007

applications.

2. Thesis: I have proposed a novel Multi-View Based Point Cloud Completion Network (MVPCC-Net) for completing colored 3D point clouds representing various incomplete object shapes. I have demonstrated by quantitative and qualitative experiments both on synthetic and real-world MLS data that the proposed method outperforms various state-of-the-art 3D point cloud completion techniques.

Published in [2][5]

Mobile laser scanning (MLS) is an emerging technology for generating extremely dense and very precise 3D point clouds for urban environments; yet, because the scanning vehicle can only operate on roads, many point clouds of field items have incomplete shapes.

State-of-the-art point cloud completion methods [99, 102, 112, 118] have demonstrated success in estimating full geometric models of various object shapes. However, 3D point cloud models obtained by previous approaches represent only coarsely detailed object shapes, because the aforementioned methods are limited to providing outputs with a constant fixed number of points.

I have introduced a novel multi view-based approach for completing high-resolution 3D point clouds of partial object shapes obtained by MLS platforms, which is able to preserve the genuine high level of detailedness of partial point cloud shapes obtained by real MLS systems. I have proved through quantitative and qualitative experiments on the provided dataset that our method outperforms state-of-the-art techniques in reconstructing the local fine geometric structures and predicting the overall shape of the objects.

2.1. I have proposed a new approach for encoding the unstructured 3D point cloud data from several surrounding perspectives into a set of regular multi-channel 2D images comprising both geometry and color information. This representation permits the use of 2D Convolutional Neural Networks (CNNs) to fill in the missing structural and color information in the image domain, and afterward it enables the creation of a dense colored 3D point cloud representing the full object's shape.To experimentally validate the proposed approach I have constructed a new database that consists of both synthetic and real MLS data and I have made it publicly available.

Previous point cloud completion methods employ various techniques to deal

DOI:10.15774/PPKE.ITK.2023.007

with the fundamental unorganized character of the point clouds, such as voxelization [99], intermediary 3D grids [102], or directly processing the point cloud [112, 118], with the PointNet encoder [115]. In order to apply the aforementioned data representations with feasible computational requirements to MLS data, the input point cloud must be spatially downsampled, resulting in simplified object shape models with substantially lower point density and less geometric information than the original MLS measurements. To complete the missing regions of MLS measurement data and to keep its high resolution, I have proposed a new representation of the point cloud data in which the input point cloud is represented as a collection of sparsely filled multi-channel images that capture from multiple angles and convey the geometry and color information. This representation facilitates the usage of 2D Convolutional Neural Networks (CNNs) and simplifies the fusion of color and geometry information. The proposed deep neural network is trained on this new data representation and the results are then reprojected to a 3D point cloud. I have demonstrated the effectiveness of this representation by presenting quantitative and qualitative results that verify the accuracy and density of the output point cloud.

For training and quantitative evaluation of the proposed method, I have provided a new point cloud dataset consisting of both synthetic point clouds of four different street object classes with accurate ground truth, and real MLS measurements of partially or fully scanned vehicles. I have demonstrated by quantitative and qualitative experiments both on synthetic and real-world MLS data that the proposed method is applicable and it outperforms various state-of-the-art techniques in terms of geometric shape accuracy, realistic RGB coloring, and preserving high resolution.

> *2.2. I have proposed a late fusion-based technique for fusing the view-level features generated from several perspectives around the object. The proposed method provides a robust global feature for transmitting shared characteristics between distinct viewpoints. I have demonstrated by quantitative and qualitative results that the proposed approach outperforms the early fusion baseline technique both for pure geometric data samples (XYZ), and for colored point clouds (XYZRGB).*

Existing multi-view based methods use an early fusion technique, where all projected images from all views are concatenated into a single input data for the

DOI:10.15774/PPKE.ITK.2023.007

encoder, while the decoder generates all output views in one step. On the contrary, my proposed method uses a late fusion strategy, whereby a shared encoder is used to build a view-level feature representation for each view separately, and then a feature fusion network component is used to create a global feature from the view-level features. The shared decoder receives the global feature and the view-level features in a certain order to generate the view-level output images that constitute the whole 3D point cloud.

I have quantitatively and qualitatively demonstrated that adding RGB information to the early fusion based method reduces the geometric accuracy notably more than the late fusion based method, where we notice just a minor decrease in the accuracy of the predicted object shape.

4.2 Application of the Results

All the developed algorithms can be used by various up-to-date or future computer vision systems. The first thesis can be applied to various image-based documentation and survey applications in archeology, architecture, or civil engineering, where brick segmentation is considered as an important initial step in the analysis of masonry wall images. Image-based analysis of man-built structures is considered as a core step of many applications, such as stability analysis in civil engineering, condition estimation and damage detection of buildings in architecture, digital documentation in archeology or maintenance and restoration in cultural heritage preservation.

The 3D point cloud completion method presented in the second thesis is useful in a wide variety of computer vision and robotic activities where full scene representation is needed for improved scene visualization, such as VR/AR applications, self-driving, and surveillance applications.

DOI:10.15774/PPKE.ITK.2023.007

4.3 Implementation Details

The main platform for point cloud handling and processing was implemented in Python3 and Open3D while the neural network models were implemented and trained in Python3 with Pytorch/ Keras frameworks. The hardware set up for training contains two Nvidia Geforce RTX 3060 Ti GPU with 16 GB device memory and 64 GB main memory.

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Appendix A

Supplementary Material

A.1 Lidar Technology

Lidar – Light Detection and Ranging – is a laser based remote sensing technology, which is increasingly used for several applications, such as autonomous vehicles, aerial inspection, forestry and land management and robotics. The technique is based on the remote measurement of an item utilizing a light beam transmitted to the object and reflected to the transmitter, and the distance between the Lidar system and the target is determined by the duration of flight of the light produced. The output of the measurement is a highly accurate 3D point cloud with coordinates in a local or global coordinate system, depending on the kind of Lidar device and the application domain.

Figure A.1: Riegl VMX-450 mobile mapping system.

Riegl VMX-450 (Figure A.1) was used in Chapter 3 for data acquisition in this thesis, the Lidar mounted on the roof of a moving vehicle can provide highly detailed, precise, and feature-rich information about the 3D environment even

DOI:10.15774/PPKE.ITK.2023.007

Notation	Definition
Qk-s	(1×1) Convolution-BatchNorm-ReLU layer
Wk-s	(3×3) Convolution-BatchNorm-ReLU layer
Tk-s	(7×7) Convolution-BatchNorm-ReLU layer
Mk-s	(3×3) Convolution-InstanceNorm-ReLU layer
Dk-s	(4×4) Convolution-InstanceNorm-ReLU layer
Ck-s	(7×7) Convolution-InstanceNorm-ReLU layer with reflection padding
Rk-s	(3×3) Convolution-SpectralNorm-InstanceNorm-ReLU layer with residual block across all layers and dilated convolution in the first layer
Uk-s	(4×4) Convolution-InstanceNorm-ReLU layer with transpose convolution for up-sampling
Vk-s	(4×4) Convolution-SpectralNorm-LeakyReLU layer

Table A.1: Notation of the used layers in the thesis

at high speeds. The system incorporates two RIEGL VQ-450 laser scanners, an inertial measurement unit (IMU), a global navigation satellite system (GNSS), and up to six digital cameras. The system offers rapid 3D data collection with high accuracy and high resolution, which may serve as a foundation for various applications including mapping of transportation infrastructure, city modeling, fast mapping of construction sites, surveying of mining, and network planning.

A.2 Deep Learning Network Parameters

In this section, we provide additional technical information regarding the networks utilized in the thesis, including the structure of the networks, the loss functions used, and the training technique.

A.2.1 Network Architectures

We provide the architectures of the networks utilized in the thesis following the naming convention described in [28, 49].

DOI:10.15774/PPKE.ITK.2023.007

Table A.1 represents the notations used to describe the used layers, where k stands for the number of filters and s stands for the used stride. Each network will be addressed individually in the following sections:

A.2.1.1 Pre-Processing Stage

The network used in Section 2.4.1 for the pre-processing stage is a U-Net based network, It is an encoder-decoder-based model including skip connection (i.e. the layers in the encoder part are skip connected and concatenated with layers in the decoder part). The encoder structure follows a Resnet50 [9] encoder structure which can be described as follows:

T64-2, (*(3 × 3) Max Pool, s=2*), 3× (*Q64-2*, *W64-1*, *Q256-1*), 4× (*Q128-2*, *W128-1*, *Q512-1*), 6× (*Q256-2*, *W256-1*, *Q1024-1*), 3× (*Q512-2*, *W512-1*, *Q2048-1*).

The decoder structure follows an inverse of the encoder structure with transpose convolution for up-sampling.

A.2.1.2 Generators

The architecture of the used Generators in Section 2.4.2 are adopted from the model proposed by [49], which follows the following structure:

C64-1, *D128-2*, *D256-2*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *U128-2*, *U64-2*, *Ck-1.*

The final layer *Ck-1* varies depending on the generator. In the *Hidden Feature Generator* $G1$ described in Section 2.4.2.1, *Ck-1* has a channel size of 1 (k = 1) with *sigmoid* activation. In the *Image Completion Generator* $G2$ described in Section 2.4.2.2, *Ck-1* has a channel size of 3 (k =3) with *tanh* activation for the prediction of RGB pixel intensities.

In addition, we add Spectral Normalization for all layers of *G1.* Spectral normalization (SN) [47] further stabilizes training by scaling down weight matrices by their respective largest singular values, effectively restricting the Lipschitz constant of the network to one. Although this was originally proposed to be used only on the discriminator, recent work [129] suggests that the generator can also benefit from SN by suppressing sudden changes in parameter and gradient values. Therefore, we apply SN to both the generator and discriminator.

DOI:10.15774/PPKE.ITK.2023.007

The generator used in Chapter 3 is related to the number of views selected to encode the input point cloud, and the architecture of it as follows:

$N_{views}\times$ (*C64-1*, *D128-2*, *D256-2*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*, *R256-1*,), *M256-1*, $N_{views}\times$ (*U128-2*, *U64-2*, *C6-1.*).

Where the N_{views} is the number of views. The final channel has 6 channels which are the XYZ geometry information, and the RGB color information for each view.

A.2.1.3 Discriminators

The discriminators used in the thesis have the same architecture which is based on the 70×70 PatchGAN [34, 126], the architecture is as follows:

V64-2, *V128-2*, *V256-2*, *V512-1*, *V1-1*

The final convolution layer produces scores predicting whether 70×70 overlapping image patches are real or fake. LeakyReLU [124] is employed with slope 0.2.

A.2.2 Loss Functions

The Perceptual loss [49] measures high-level perceptual and semantic differences between images by defining a distance measure between activation maps of a pre-defined network, which are presented in Equation A.1.

$$\ell_{\text{perc}} = \mathbb{E}\big[\sum_{i} \frac{1}{N_{\text{i}}} ||\phi_i(I_{\text{wall}}) - \phi_i(I_{\text{G2\_out}})||\big] \tag{A.1}$$

where ϕ_i is the activation map of the i_{th} layer of a pre-trained network. In this thesis, we used the activation map of the VGG19 network pre-trained on the ImageNet dataset [125] from layers *relu1_1*, *relu2_1*, *relu3_1*, *relu4_1* and *relu5_1*. These activation maps are also used to compute the style loss [48] which measures the differences between the covariances of the activation maps.

The style loss is defined as

$$\ell_{\text{sty}} = \mathbb{E}_j\big[||G_j^{\phi}(I_{\text{wall}}) - G_j^{\phi}(I_{\text{G2\_out}})||\big] \tag{A.2}$$

where G_j^{ϕ} is a $C_j \times C_j$ Gram matrix constructed from activation maps ϕ_j

DOI:10.15774/PPKE.ITK.2023.007

It was determined that outputs were severely pixelated and noisy when just perceptual and style losses were optimized. To address this problem, the Total variation loss was developed, which maintains spatial continuity and smoothness in the generated image to prevent noisy and overly-pixelated results. The Total variation loss is the total of the absolute differences between nearby pixel values in the input images; it quantifies the amount of noise in images. as defined in the following:

$$\ell_{\text{tv}} = \sum_{i}^{n-1} |(I_{(\text{i+1,j})}) - (I_{(\text{i,j})}) + \sum_{i}^{n-1} |(I_{(\text{i,j+1})}) - (I_{(\text{i,j})})| \tag{A.3}$$

In the Hidden Feature Generator step, the perceptual loss [49] - where activation maps are compared with those from the pre-trained VGG network can not be used, since the VGG network is not trained to produce two classes (mortar and bricks) images, It doesn't provide the result we were looking for this stage. Therefore, the feature-matching loss ℓ_{FM} is used so that it compares the activation maps in the intermediate layer of the discriminator. Using the feature-matching loss stabilizes the training process by forcing the generator to produce results with representations that are similar to real images.

The feature matching loss ℓ_{FM} is defined as

$$\ell_{\text{FM}} = \sum_{i=1}^{L} \frac{1}{N_i} ||D_1^{(i)}(I_{\text{wall\_ftr}}) - D_1^{(i)}(I_{\text{G1\_out}})|| \tag{A.4}$$

where L is the final convolution layer of the discriminator, N_i is the number of elements in the i'th activation layer, and $D_1^{(i)}$ is the activation in the i'th layer of the discriminator.

A.2.3 Normalization Strategy

This section highlights the normalization techniques employed while training the networks presented in the thesis. Normalization refers to a scaling technique method in which data points x are *shifted* and *re-scaled* by its *mean* μ and *standard deviation* σ respectively.

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{A.5}$$

DOI:10.15774/PPKE.ITK.2023.007

A.2.3.1 Batch Normalization

Given a feature map of dimensions $N \times C \times H \times W$, where N = Batch size, C = Number of filters (channels) in the layer, H = Height of the activation map, W = Width of the activation map. By using the batch normalization technique [127], the *mean* and *variance* are computed for each individual channel over all samples and both spatial dimensions, as presented in the Equations A.6, A.7 respectively. Each network layer is normalized based on these computed values as provided in Equation A.8, where ϵ is an arbitrarily small constant added to the denominator for numerical stability.

$$\mu_c = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{icjk} \tag{A.6}$$

$$\sigma_c^2 = \frac{1}{NHW} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{icjk} - \mu_c)^2 \tag{A.7}$$

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \tag{A.8}$$

A.2.3.2 Instance Normalization

By using the instance normalization technique [128], the *mean* and *variance* are calculated for each individual channel for each individual sample across both spatial dimensions, as presented in the Equations A.9, A.10 respectively, and the layers of the networks are re-centered and re-scaled based on Equation A.11.

$$\mu_{nc} = \frac{1}{HW} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{ncjk} \tag{A.9}$$

$$\sigma_{nc}^2 = \frac{1}{HW} \sum_{j=1}^{H} \sum_{k=1}^{W} (x_{icjk} - \mu_{nc})^2 \tag{A.10}$$

$$\hat{x} = \frac{x - \mu_{nc}}{\sqrt{\sigma_{nc}^2 + \epsilon}} \tag{A.11}$$

DOI:10.15774/PPKE.ITK.2023.007

A.2.4 Training Setup and Strategy

In this section, we present the training methodology and the optimization strategy of the networks presented in the thesis

A.2.4.1 Training Setup of the Networks used in Chapter 2

For the training phase, the three sub-networks (*U-Net*, G_1, G_2) can interact through shared dependencies in their loss functions. However, this model leads to a highly complex optimization problem, with large computational and memory requirements. Particularly, we had to manage to fit into the GPU memory during the training process. For these reasons, in the initial training phase, the three networks were separately trained, providing an efficient initial weight-set for the upcoming joint optimization steps. Thereafter the first two networks (*U-Net*, G_1) were trained together, so that the the $F1_{u\_net}$ F_1-score of the *U-Net* output was used as a confidence value to weight the loss function of G_1 (see Equation (2.2) in Section 2.4.2.1), and the feature matching loss ℓ_{ftr_mat} of G_1 was used to update the U-Net weights (see Equation (2.1) in Section 2.4.1). For all stages in the network, the Adam optimizer parameters were set as $\beta_1 = 0$ and $\beta_2 = 0.9$.

A.2.4.2 Training Setup of the Networks used in Chapter 3

PyTorch is used to implement the proposed completion network, which is trained on 256×256 images. As an optimization algorithm, we employ the Adam optimizer [40] with the settings $\beta_1 = 0$ and $\beta_2 = 0.9$. The default batch size is 4, but when the number of selected views is greater than five, the batch size is reduced to 1, so that the computation can be completed on the GPU, and the weights are only updated in every four iterations. The model is trained in three sessions in which the learning rate parameter, which determines the step size at each iteration while moving toward a minimum of the loss function, is gradually decreased: in each section, we train the model until converges, while using learning rates 10^{-4}, 10^{-5}, and 10^{-6} respectively. In the loss function, the TV loss term is only used in the last two sections.

DOI:10.15774/PPKE.ITK.2023.007

DOI:10.15774/PPKE.ITK.2023.007

Appendix B

Summary of Abbreviations

Chapter 1	
ADAS	Advanced Driving Assistance Systems
VR	Virtual Reality
AR	Augmented Reality
SLAM	Simultaneous Localization And Mapping
Chapter 2	
CH	Cultural Heritage
DCH	Digital Cultural Heritage
DL	Deep Learning
ML	Machine Learning
GAN	Generative Adversarial Networks
HED	Holistically-Nested Edge Detection
IDT	inverse distance transform
GT	Ground Turth
Pr	Precision
Rc	Recall
TP	True Positive
FP	False Positive
FN	False Negative
IOU	intersection of union
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index
FID	Frechet Inception Distance
Chapter 3	
MLS	Mobile Laser Scanning
VR	Virtual Reality

DOI:10.15774/PPKE.ITK.2023.007

AR	Augmented Reality
SLAM	Simultaneous Localization And Mapping
CNNs	Convolutional Neural Networks
PCN	Point Completion Network
VPC-Net	Vehicle Points Completion-Net
3D-EPN	3D-Encoder-Predictor Network
GRNet	Gridding Residual Network
CD	Chamefer Distance
ViPC	View-guided Point Cloud
CSDN	Cross-model Shape-transfer model Distance
GT	Ground Truth
TV	Total Variation
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index
STN	Spatial Transformer Network
SOF	Statistical Outlier Filter
Chapter 4	
CCA	Connected Component Analysis
GAN	Generative Adversarial Networks
VR/AR	Virtual Reality/Augmented Reality
GPU	Graphics Processing Unit
Appendix A	
IMU	Inertial Measurement Unit
GNSS	Global Navigation Satellite System
SN	Spectral Normalization

DOI:10.15774/PPKE.ITK.2023.007

References

The author's journal publications

[1] **Y. Ibrahim**, B. Nagy, and C. Benedek, "Deep Learning-Based Masonry Wall Image Analysis," *Remote Sensing*, vol. 12, no. 23, 2020. IF=4,8, Scimago Q1. (document), 1.1, 4.1

[2] **Y. Ibrahim** and C. Benedek, "MVPCC-Net: Multi-View Based Point Cloud Completion Network for MLS Data," *Image and Vision Computing*, vol. 134, no. article 104675, 2023. IF=3.860, Scimago Q1. (document), 1.3, 4.1

The author's international conference publications

[3] **Y. Ibrahim**, B. Nagy, and C. Benedek, "CNN-Based Watershed Marker Extraction for Brick Segmentation in Masonry Walls," in *16th International Conference on Image Analysis and Recognition*, (Cham), pp. 332–344, Springer International Publishing, 2019. 4.1

[4] **Y. Ibrahim**, B. Nagy, and C. Benedek, "A GAN-based Blind Inpainting Method for Masonry Wall Images," in *25th International Conference on Pattern Recognition (ICPR)*, pp. 3178–3185, 2021. 4.1

[5] **Y. Ibrahim**, B. Nagy, and C. Benedek, "Multi-view Based 3D Point Cloud Completion Algorithm for Vehicles," in *26th International Conference on Pattern Recognition (ICPR)*, pp. 2121–2127, 2022. 4.1

DOI:10.15774/PPKE.ITK.2023.007

[6] **Y. Ibrahim**, P. Szulovszky, and C. Benedek, "Masonry Structure Analysis, Completion and Style Transfer Using a Deep Neural Network," in *International Workshop on Pattern Recognition for Cultural Heritage, to appear in the ICPR 2022 Workshop Proceedings, Lecture Notes in Computer Science*, (Montreal, Canada), Springer, August 21, 2022. 4.1

[7] W. Fadel, C. Kollod, M. Wahdow, **Y. Ibrahim**, and I. Ulbert, "Multi-Class Classification of Motor Imagery EEG Signals Using Image-Based Deep Recurrent Convolutional Neural Network," in *8th International Winter Conference on Brain-Computer Interface (BCI)*, pp. 1–4, 2020.

Publications connected to the dissertation

[8] K. Saleh, S. Szenasi, and Z. Vamossy, "Occlusion handling in generic object detection: A review," pp. 000477–000484, 01 2021. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 770–778, IEEE Computer Society, jun 2016. 1, 2.4.1, A.2.1.1

[10] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734, 2015. 1

[11] H. Zhu, P. Tang, and A. L. Yuille, "Robustness of Object Recognition under Extreme Occlusion in Humans and Computational Models," in *Annual Meeting of the Cognitive Science Society*, 2019. 1

[12] L. S. Huber, R. Geirhos, and F. A. Wichmann, "A four-year-old can outperform resnet-50: Out-of-distribution robustness may not require large-scale experience," in *SVRHM 2021 Workshop @ NeurIPS*, 2021. 1

[13] C. Ning, L. Menglu, Y. Hao, X. Su, and L. Yunhong, "Survey of pedestrian detection with occlusion," *Complex Intelligent Systems*, vol. 7, pp. 1–11, 10 2020. 1

DOI:10.15774/PPKE.ITK.2023.007

[14] C. Duan, J. Pan, and R. Li, "Thick cloud removal of remote sensing images using temporal smoothness and sparsity regularized tensor optimization," *Remote Sensing*, vol. 12, no. 20, 2020. 1

[15] P. Dai, S. Ji, and Y. Zhang, "Gated convolutional networks for cloud removal from bi-temporal remote sensing images," *Remote Sensing*, vol. 12, no. 20, 2020. 1

[16] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Occlusion-Aware R-CNN: Detecting Pedestrians in a Crowd," in *European Conference on Computer Vision (ECCV)* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 657–674, Springer International Publishing, 2018. 1

[17] Y. Liu, X.-Y. Jing, J. Nie, H. Gao, J. Liu, and G.-P. Jiang, "Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in rgb-d videos," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 664–677, 2019. 1

[18] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: Deep Hypersphere Embedding for Face Recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[19] N. D. Reddy, M. Vo, and S. G. Narasimhan, "Occlusion-net: 2d/3d occluded keypoint localization using graph networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7318–7327, 2019. 1

[20] F. Lin, Y. Xu, Z. Zhang, C. Gao, and K. D. Yamada, "Cosmos propagation network: Deep learning model for point cloud completion," *Neurocomputing*, vol. 507, pp. 221–234, 2022. 1

[21] R. Pierdicca, M. Paolanti, F. Matrone, M. Martini, C. Morbidoni, E. S. Malinverni, E. Frontoni, and A. M. Lingua, "Point Cloud Semantic Segmentation Using a Deep Learning Framework for Cultural Heritage," *Remote Sensing*, vol. 12, p. 1005, Mar 2020. 2.1

DOI:10.15774/PPKE.ITK.2023.007

[22] Gallwey, Eyre, Tonkins, and Coggan, "Bringing Lunar LiDAR Back Down to Earth: Mapping Our Industrial Heritage through Deep Transfer Learning," *Remote Sensing*, vol. 11, p. 1994, Aug 2019. 2.1

[23] R. de Lima-Hernandez and M. Vergauwen, "A Hybrid Approach to Reassemble Ancient Decorated Block Fragments through a 3D Puzzling Engine," *Remote Sensing*, vol. 12, p. 2526, Aug 2020. 2.1

[24] S. Teruggi, E. Grilli, M. Russo, F. Fassi, and F. Remondino, "A Hierarchical Machine Learning Approach for Multi-Level and Multi-Resolution 3D Point Cloud Classification," *Remote Sensing*, vol. 12, p. 2598, Aug 2020. 2.1

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, vol. 9351 of *LNCS*, pp. 234–241, 2015. 2.1.2, 2.4.1

[26] D. Pathak, P. Krähenbl̈, J. Donahue, T. Darrell, and A. A. Efros, "Context Encoders: Feature Learning by Inpainting," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, 2016. 2.2.2

[27] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image Inpainting for Irregular Holes Using Partial Convolutions," in *European Conference on Computer Vision (ECCV)*, 2018. 2.2.2, 2.5.2.2, 3.5.1

[28] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning," in *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3265–3274, 2019. (document), 2.2.2, 2.4.2, 2.4.2.1, 2.4.2.2, 2.4.2.2, 2.5.2.2, 2.2, 2.10, 2.12, 3.4.2, A.2.1

[29] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," in *Advances in Neural Information Processing Systems 25 (NIPS)*, pp. 341–349, 2012. 2.2.2

DOI:10.15774/PPKE.ITK.2023.007

[30] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-Form Image Inpainting With Gated Convolution," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4470–4479, 2019. 2.2.2

[31] C. Zheng, T. Cham, and J. Cai, "Pluralistic Image Completion," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1438–1447, 2019. (document), 2.2.2, 2.5.2.2, 2.2, 2.10

[32] Y. Liu, J. Pan, and Z. Su, "Deep Blind Image Inpainting," in *Intelligence Science and Big Data Engineering. Visual Data Engineering*, pp. 128–141, 2019. 2.2.2

[33] R. Köhler, C. Schuler, B. Schölkopf, and S. Harmeling, "Mask-specific inpainting with deep neural networks," in *German Conference on Pattern Recognition, LNCS*, pp. 523–534, 2014. 2.2.2

[34] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017. 2.4.2, 3.4.2, A.2.1.3

[35] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Transactions on Graphics*, vol. 28, Aug. 2009. 2.2.2

[36] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image Melding: Combining inconsistent images using patch-based synthesis," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 82:1–82:10, 2012. 2.2.2

[37] S. Esedoglu and J. Shen, "Digital inpainting based on the Mumford-Shah-Euler image model," *European Journal of Applied Mathematics*, vol. 13, pp. 353–370, 2002. 2.2.2

[38] D. Liu, X. Sun, F. Wu, S. Li, and Y. Zhang, "Image Compression With Edge-Based Inpainting," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, pp. 1273–1287, Oct 2007. 2.2.2

DOI:10.15774/PPKE.ITK.2023.007

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014. 2.2.2, 3.4.2

[40] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2014. 2.4.1, A.2.4.2

[41] I. Sárándi, T. Linder, K. O. Arras, and B. Leibe, "How Robust is 3D Human Pose Estimation to Occlusion?," in *IEEE/RSJ International Conference Intelligent Robots and Systems Workshop (IROSWS)*, 2018. 2.3

[42] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results." 2.3, 2.5

[43] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. FernĂAndez, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, *et al.*, "imgaug." `https://github.com/aleju/imgaug`, 2020. 2.3

[44] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014. 2.2.2

[45] S. Xie and Z. Tu, "Holistically-nested edge detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1395–1403, 2015. 2.2.2

[46] J. Canny, "A computational approach to edge-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986. 2.2.2

[47] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *International Conference on Learning Representation*, 2018. 2.4.2.1, A.2.1.2

DOI:10.15774/PPKE.ITK.2023.007

[48] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016. 2.4.2.2, 3.4.2, A.2.2

[49] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision (ECCV)*, 2016. 2.4.2, 2.4.2.2, 3.4.2, A.2.1, A.2.1.2, A.2.2, A.2.2

[50] M. D. Bloice, P. M. Roth, and A. Holzinger, "Biomedical image augmentation using Augmentor," *Bioinformatics*, vol. 35, pp. 4522–4524, 04 2019. 2.5.2.2

[51] M. G. Eramian, E. Walia, C. Power, P. A. Cairns, and A. P. Lewis, "Image-based search and retrieval for biface artefacts using features capturing archaeologically significant characteristics," *Machine Vision and Applications*, vol. 28, pp. 201 – 218, 2016. 2.1

[52] S. Prasomphan and J. E. Jung, "Mobile Application for Archaeological Site Image Content Retrieval and Automated Generating Image Descriptions with Neural Network," *Mobile Networks and Applications*, vol. 22, p. 642â649, 2017. 2.1

[53] L. van der Maaten, P. Boon, G. Lange, J. Paijmans, and E. Postma, "Computer vision and machine learning for archaeology," in *Proceedings of CAA-2006*, p. online, Unknown Publisher, 2006. 2.1

[54] N. A. Rasheed and M. J. Nordin, "Classification and reconstruction algorithms for the archaeological fragments," *Journal of King Saud University - Computer and Information Sciences*, 2018. 2.1

[55] A. Hadjiprocopis, M. Ioannides, K. Wenzel, M. Rothermel, P. S. Johnsons, D. Fritsch, A. Doulamis, E. Protopapadakis, G. Kyriakaki, K. Makantasis, G. Weinlinger, M. Klein, D. Fellner, , A. Stork, and P. Santos, "4D reconstruction of the past: the image retrieval and 3D model construction pipeline," in *Second International Conference on Remote Sensing and*

DOI:10.15774/PPKE.ITK.2023.007

Geoinformation of the Environment (RSCy) (D. G. Hadjimitsis, K. Themistocleous, S. Michaelides, and G. Papadavid, eds.), vol. 9229, pp. 331 – 340, International Society for Optics and Photonics, SPIE, 2014. 2.1

[56] M. Bolhassani, S. Rajaram, A. A. Hamid, A. Kontsos, and I. Bartoli, "Damage detection of concrete masonry structures by enhancing deformation measurement using DIC," in *Nondestructive Characterization and Monitoring of Advanced Materials, Aerospace, and Civil Infrastructure 2016* (T. Yu, A. L. Gyekenyesi, P. J. Shull, and H. F. Wu, eds.), vol. 9804, pp. 227 – 240, International Society for Optics and Photonics, SPIE, 2016. 2.1.1

[57] L. Ali, W. Khan, and K. Chaiyasarn, "Damage Detection and Localization in Masonry Structure Using Faster Region Convolutional Networks," *International Journal of GEOMATE*, vol. 17, pp. 98–105, 2019. 2.1.1

[58] R. Hashimoto, T. Koyama, M. Kikumoto, T. Saito, and M. Mimura, "Stability Analysis of Masonry Structure in Angkor Ruin Considering the Construction Quality of the Foundation," 2014. 2.1.1

[59] R. Kılıç Demircan and A. İ. Ünay, "Structural Stability Analysis of Large-Scale Masonry Historic City Walls," in *Proceedings of 3rd International Sustainable Buildings Symposium (ISBS)* (S. Fırat, J. Kinuthia, and A. Abu-Tair, eds.), (Cham), pp. 384–395, Springer International Publishing, 2018. 2.1.1

[60] E. Valero, F. Bosché, A. Forster, and E. Hyslop, "Historic Digital Survey: Reality Capture and Automatic Data Processing for the Interpretation and Analysis of Historic Architectural Rubble Masonry," in *Structural Analysis of Historical Constructions* (R. Aguilar, D. Torrealva, S. Moreira, M. A. Pando, and L. F. Ramos, eds.), (Cham), pp. 388–396, Springer International Publishing, 2019. 2.1.1

[61] B. Riveiro, B. Conde, H. Gonzalez, P. Arias, and J. CaamaĂao, "Automatic creation of structural models from point cloud data: the case of masonry structures," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-3/W5, pp. 3–9, 8 2015. 2.2.1, 2.5.3.1, 2.3, 4.1

DOI:10.15774/PPKE.ITK.2023.007

[62] G. Sithole, "Detection of bricks in a masonry wall," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, vol. XXXVII, pp. 567–572, 2008. 2.2, 2.2.1

[63] R. Kajatin and L. Nalpantidis, "Image segmentation of bricks in masonry wall using a fusion of machine learning algorithms," in *Pattern Recognition. ICPR International Workshops and Challenges* (A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, eds.), (Cham), pp. 446–461, Springer International Publishing, 2021. 2.2.1

[64] K. Idjaton, X. Desquesnes, S. Treuillet, and X. Brunetaud, "Stone-by-stone segmentation for monitoring large historical monuments using deep neural networks," in *Pattern Recognition. ICPR International Workshops and Challenges* (A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, eds.), (Cham), pp. 235–248, Springer International Publishing, 2021. 2.2.1

[65] N. Oses, F. Dornaika, and A. Moujahid, "Image-based delineation and classification of built heritage masonry," *Remote Sensing*, vol. 6, p. 1863â1889, Feb 2014. 2.2.1, 2.5.3.1, 2.3, 4.1

[66] F. BoschĂŠ, E. Valero, A. Forster, L. Wilson, and A. Leslie, *Evaluation of historic masonry substrates: towards greater objectivity and efficiency*. 06 2016. 2.2.1

[67] M. Hemmleb, F. Weritz A, A. Schiemenz B, A. Grote C, and C. Maierhofer, "Multi-spectral data acquisition and processing techniques for damage detection on building surfaces," in *ISPRS Commission V Symposium*, pp. 1–6, 1 2006. 2.2.1, 4.1

[68] J. L. Fernández, P. M. Lerones, E. Z. Casanova, and J. G. García-Bermejo, "Applying deep learning techniques to cultural heritage images within the inception project," in *EuroMed*, 2016. 2.2.1

DOI:10.15774/PPKE.ITK.2023.007

[69] J. Llamas, P. M. Lerones, R. Medina, E. Zalama, and J. GĂłmez-GarcĂa-Bermejo, "Classification of Architectural Heritage Images Using Deep Learning Techniques," *Applied Sciences*, vol. 7, p. 992, 09 2017. 2.2.1

[70] V. Zyuzin, P. Sergey, A. Mukhtarov, T. Chumarnaya, O. Solovyova, A. Bobkova, and V. Myasnikov, "Identification of the left ventricle endocardial border on two-dimensional ultrasound images using the convolutional neural network Unet," in *Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 76–78, May 2018. 2.2.1

[71] Y. Tao, P. Palasek, Z. Ling, and I. Patras, "Background modelling based on generative unet," in *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, Aug 2017. 2.2.1

[72] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955. 2.5.3.2

[73] X. MuĂaoz, J. Freixenet, X. Cufi, and J. Marti, "Strategies for image segmentation combining region and boundary information," *Pattern Recognition Letters*, vol. 24, pp. 375–392, 01 2003. 2.4.3

[74] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Honolulu, Hawaii), pp. 5221–5229, 2017. 2.2.1

[75] J. B. Roerdink and A. Meijster, "The Watershed transform: Definitions, algorithms and parallelization strategies," *Fundam. Inf.*, vol. 41, pp. 187–228, Apr. 2000. 2.4.3

[76] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, "Image Inpainting via Generative Multi-column Convolutional Neural Networks," *CoRR*, vol. abs/1810.08771, 2018. (document), 2.2.2, 2.5.2.2, 2.2, 2.10

[77] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium," *CoRR*, vol. abs/1706.08500, 2017. 2.5.2.2

DOI:10.15774/PPKE.ITK.2023.007

[78] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. 2.5.2.2, 3.5.1

[79] J. Femiani, W. R. Para, N. J. Mitra, and P. Wonka, "Facade segmentation in the wild," *CoRR*, vol. abs/1805.08634, 2018. 2.5.4

[80] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 416–423, July 2001. (document), 2.2.2, 2.2

[81] C. Suchocki, M. D.-S. OrcID, J. Katzer, J. Janicka, J. RapiĹski, and P. StaĹowska, " Remote Detection of Moisture and Bio-Deterioration of Building Walls by Time-Of-Flight and Phase-Shift Terrestrial Laser Scanners ," *Remote Sensing*, vol. 12, p. 1708, MAY 2020. 2.1.2

[82] Y. Shen, R. Lindenbergh, J. W. , and V. G. Ferreira, "Extracting Individual Bricks from a Laser Scan Point Cloud of an Unorganized Pile of Bricks," *Remote Sensing*, vol. 10, p. 1708, Oct 2018. 2.1.1

[83] C. Li and M. Wand, "Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks," in *European Conference on Computer Vision (ECCV)* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 702–716, Springer International Publishing, 2016. 2.2.3

[84] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "StyleBank: An Explicit Representation for Neural Image Style Transfer," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2770–2779, 2017. 2.2.3

[85] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017. 2.2.3

DOI:10.15774/PPKE.ITK.2023.007

[86] Z. Liu, J. Cheng, Q. Wang, and L. Xian, "Improved Design Based on IoU Loss Functions for Bounding Box Regression," in *IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pp. 452–458, 2022. 3.4.2

[87] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial Transformer Networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015. 3.5.3

[88] D. Bolkas, J. Chiampi, J. Chapman, and V. Pavill, "Creating a virtual reality environment with a fusion of sUAS and TLS point-clouds," *International Journal of Image and Data Fusion*, vol. 11, pp. 1–26, 01 2020. 3.1

[89] F. Hariz, H. Souifi, R. Leblanc, Y. Bouslimani, M. Ghribi, E. Langin, and D. Mccarthy, "Direct Georeferencing 3D Points Cloud Map Based on SLAM and Robot Operating System," in *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pp. 1–6, 2021. 3.1

[90] Y. Nie, Y. Lin, X. Han, S. Guo, J. Chang, S. Cui, and J. Zhang, "Skeleton-bridged Point Completion: From Global Inference to Local Adjustment," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 16119–16130, 2020. 3.3.1

[91] X. Zhang, Y. Feng, S. Li, C. Zou, H. Wan, X. Zhao, Y. Guo, and Y. Gao, "View-Guided Point Cloud Completion," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15890–15899, 2021. 3.2.2

[92] Z. Zhu, L. Nan, H. Xie, H. Chen, M. Wei, J. Wang, and J. Qin, "CSDN: Cross-modal Shape-transfer Dual-refinement Network for Point Cloud Completion," 2022. 3.2.2

[93] H. Fan, H. Su, and L. Guibas, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 2463–2471, 2017. 3.2.2

DOI:10.15774/PPKE.ITK.2023.007

[94] C.-H. Lin, C. Kong, and S. Lucey, "Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 3.2.2

[95] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction," in *European Conference on Computer Vision (ECCV)*, pp. 628–644, Springer International Publishing, 2016. 3.2.2

[96] M. Jaritz, J. Gu, and H. Su, "Multi-View PointNet for 3D Scene Understanding," in *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3995–4003, 2019. 3.2.2

[97] A. Boulch, B. L. Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *Eurographics Workshop on 3D Object Retrieval*, vol. 2, 2017. 3.2.2

[98] J. Zhang, D. Zhou, Y. Zhao, W. Nie, and Y. Su, "MV-LFN: Multi-view based local information fusion network for 3D shape recognition," *Visual Informatics*, vol. 5, no. 3, pp. 114–119, 2021. 3.2.2

[99] A. Dai, C. R. Qi, and M. Nießner, "Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3.1.1, 3.2.1, 4.1, 4.1

[100] Y. Wang, D. J. Tan, N. Navab, and F. Tombari, "SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification," in *European Conference on Computer Vision (ECCV)*, pp. 70–85, Springer International Publishing, 2020. 3.2.1, 3.5.1

[101] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 206–215, 2018. 3.2.1

DOI:10.15774/PPKE.ITK.2023.007

[102] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "GRNet: Gridding Residual Network for Dense Point Cloud Completion," in *European Conference on Computer Vision (ECCV)*, pp. 365–381, Springer International Publishing, 2020. 3.1.1, 3.2.1, 3.5.1, 3.5.1, 3.5.2, 3.1, 3.2, 3.3, 4.1, 4.1

[103] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. 3.2

[104] J. Tang, Z. Gong, R. Yi, Y. Xie, and L. Ma, "Lake-net: Topology-aware point cloud completion by localizing aligned keypoints," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1726–1735, 2022. 3.2.1

[105] N. A. Hadi, S. A. Halim, and N. Alias, "Statistical Filtering on 3D Cloud Data Points on the CPU-GPU Platform," *Journal of Physics: Conference Series*, vol. 1770, p. 012006, mar 2021. 3.5.5

[106] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "Pointr: Diverse point cloud completion with geometry-aware transformers," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3.2.1

[107] X. Yu, Y. Rao, Z. Wang, J. Lu, and J. Zhou, "Adapointr: Diverse point cloud completion with adaptive geometry-aware transformers," 2023. 3.2.1

[108] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han, "Snowflake point deconvolution for point cloud completion and generation with skip-transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2022. 3.2.1, 3.1, 3.2, 3.3, 3.5.5

[109] H. Zhou, Y. Cao, W. Chu, J. Zhu, T. Lu, Y. Tai, and C. Wang, "Seedformer: Patch seeds based point cloud completion with upsample transformer," in *European Conference on Computer Vision (ECCV)*, (Berlin, Heidelberg), p. 416â432, Springer-Verlag, 2022. 3.2.1, 3.1, 3.2, 3.3, 3.5.5

DOI:10.15774/PPKE.ITK.2023.007

[110] B. Nagy and C. Benedek, "3D CNN-Based Semantic Labeling Approach for Mobile Laser Scanning Data," *IEEE Sensors Journal*, vol. 19, no. 21, pp. 10034–10045, 2019. (document), 3.1, 3.1.1, 3.3.2

[111] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018. 3.4.3, 3.5.5

[112] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point Completion Network," in *International Conference on 3D Vision (3DV)*, pp. 728–737, 2018. 3.1.1, 3.2.1, 3.3.1, 3.5.2, 3.1, 3.2, 3.3, 4.1, 4.1

[113] Y. Xia, Y. Xu, C. Wang, and U. Stilla, "VPC-Net: Completion of 3D vehicles from MLS point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 174, pp. 166–181, 2021. 3.2.1, 3.3.1, 3.5.2, 3.1, 3.2, 3.3

[114] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1, 3.1.2, 3.3.1

[115] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017. 3.1.1, 3.2, 4.1

[116] C. Moenning and N. A. Dodgson, "Fast Marching farthest point sampling," in *Eurographics - Posters*, Eurographics Association, 2003. 3.5.2

[117] K. Vanjigounder, K. Narayanankutty, and S. Veni, "Performance Comparison of Total Variation based Image Regularization Algorithms," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 4, pp. 419–425, 2016. 3.4.2

DOI:10.15774/PPKE.ITK.2023.007

[118] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "TopNet: Structural Point Cloud Decoder," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 383–392, 2019. 3.2.1, 3.3.1, 3.5.2, 3.1, 3.2, 3.3, 4.1, 4.1

[119] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. 3.1

[120] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 945–953, 2015. 3.2.2

[121] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep Projective 3D Semantic Segmentation," in *Computer Analysis of Images and Patterns*, pp. 95–107, Springer International Publishing, 2017. 3.2.2

[122] A. Tagliasacchi, M. Olson, H. Zhang, G. Hamarneh, and D. Cohen-Or, "VASE: Volume-Aware Surface Evolution for Surface Reconstruction from Incomplete Point Clouds," *Computer Graphics Forum*, 2011. 3.2.1

[123] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem, "Completing 3D object shape from one depth image," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2484–2493, 2015. 3.2.1

[124] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning*, (Atlanta, Georgia), 2013. A.2.1.3

[125] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, Apr 2015. A.2.2

DOI:10.15774/PPKE.ITK.2023.007

[126] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017. A.2.1.3

[127] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448â456, JMLR.org, 2015. A.2.3.1

[128] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016. A.2.3.2

[129] A. Odena, J. Buckman, C. Olsson, T. Brown, C. Olah, C. Raffel, and I. Goodfellow, "Is generator conditioning causally related to GAN performance?," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3849–3858, PMLR, 10–15 Jul 2018. A.2.1.2