REndering Alternative Deep Image-based Sanssouci (REAL-DIBS)

Alberto Pennino

Master Thesis
March 2020

Dr. Hayko Riemenschneider
Dr. Christopher Schroers
Prof. Dr. Markus Gross

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

RENDERING ALTERNATIVE DEEP IMAGE-BASED SANSSOUCI (REAL-DIBS)

Authored by (in block letters):

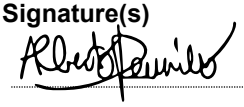
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
PENNINO	ALBERTO

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 30/03/2020	

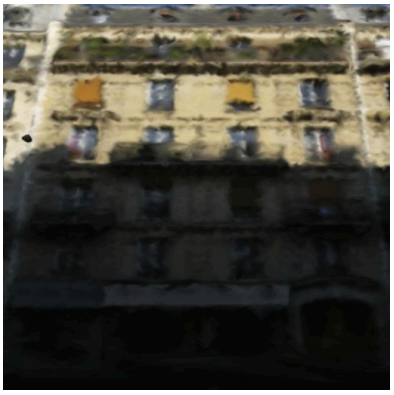
For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Disney RESEARCH STUDIOS

Bachelor/Master/Semester Thesis

REndering ALternative - Deep Image-Based Sanssouci



Introduction

The modern smart devices collect a lot of information for understanding the real environment. Smartphones and Augmented Reality devices capture photos and create 3D reconstructions. However these 3D reconstructions are by nature at inverse complex problem with various limitations. One of these is their realism and this work aims to understand and provide an alternative to current rendering and visualization of these 3D reconstructions.

Task Description

This thesis works on the rendering quality of coarse image-based 3D reconstructions. 3D reconstructions created through photogrammetry, surface meshing and coloring by texture atlas or simple vertex/face colors involves various artifacts and some artifacts degrade the final rendered image quality with respect to geometric and photometric accuracy. The goal of this project is to identify simple and worry-free upgrade strategies and models to make a rendering more realistically reflect input images. In this work we will employ and compare state-of-the-art generative models like CycleGAN, StackGAN, StyleGAN, BigGAN etc, along with other newly-developed techniques to establish baseline models and make an attempt to build upon these models to achieve significant improvements.

Skills

- Background in Computer Vision and /or Computer Graphics and/or Machine Learning
- Good programming skills in C++ and/or Matlab and/or Python
- Experience with Tensorflow and Tensorboard.

Remarks

A written report and an oral presentation conclude the thesis. The thesis is will be overseen by Prof. Markus Gross and supervised by Dr. Christopher Schroers (Disney Research) and Dr. Hayko Riemenschneider (Disney Research).

Contact

For further information, please contact the thesis coordinator.

Abstract

Estimating 3D reconstructions from a collection of images is crucial for a vast variety of applications and thus has received a large amount of attention in recent years. Although classical methods exist which can produce fairly high quality 3D reconstructions, images directly rendered from this representation typically still suffer from a variety of visual artifacts.

In this thesis, we build upon new advances in neural networks and differentiable rendering to synthesize novel views based on an initial 3D reconstruction *and* the original input views. More specifically, we explore the value of Generative Adversarial Networks for our application and analyze the potential of neural rendering, per vertex geometric optimisation, and Geometric Deep Learning.

Based on the insights from these explorations, we finally propose a novel hybrid method called Morph-GAN that combines advantages through per vertex geometry refinement with the texture generation capabilities of GAN models within a differentiable rendering framework.

Our experiments show that the proposed model is capable of significantly improving image quality and the overall realism of generated images.

Contents

List of Figures

1

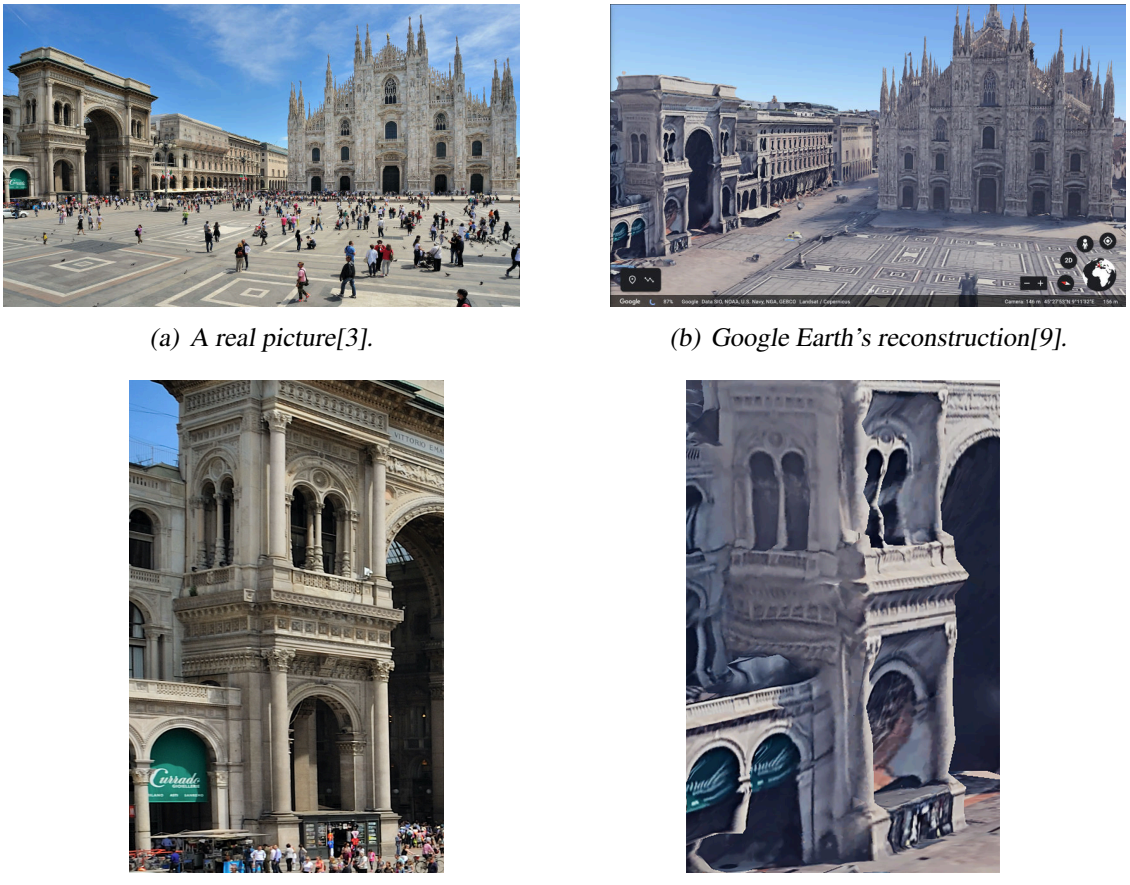
Introduction

In this work we address the problem of Total Scene Capture and Image Based Rendering (IBR). The general problem can be described as obtaining a realistic and detailed total scene reconstruction given a deterministic set of images.
Ideally the ultimate application of this field of research would be to be able to perfectly represent the real world into a computer simulated reality. The problem is complex by nature and involves various layers of modeling: 3D shape reconstruction, illumination modeling, materials modeling and complex physical simulation.
Practically this broad task has to be tackled in smaller problems and can be addressed from various perspectives. Ranging from rendering a scene under different appearance conditions (light, time of the day or season) to scene editing and 3D scene modeling.
More specifically, in the scope of this work, we operate within the novel view synthesis domain over urban environments.
Given a set of pictures of a specific place (a city or a building), our goal is to generate realistic and detailed images of the same scene at any given new viewpoint. Ideally the generated images would be indistinguishable for the real ones.
This challenging task is to this day object of prominent research interest and multiple approaches have been proposed.
In this work we perform this task by proposing a new approach based on state-of-the-art generative deep learning models, geometric deep learning and neural rendering.

1.1 Applications

Recently an increasing demand for realistic and reliable 3D content has emerged in various applications. Moreover novel view synthesis is object of interest in numerous domains.

Mapping. Novel view synthesis is a key issue in modern map making. Combining information from satellite imagery, user taken pictures and street view videos still represents an open issue within the most popular maps platforms. Even though incredible progress has been made in recent years, realistic 3D models of cities and urban environments are still far from being achieved.
An example of this limitations can be observed in Fig 1.1. By comparing a real picture to Google Earth's reconstruction [9], some visual artifacts appear evident. Looking at the details of the gallery's entrace, for example, some very strong geometrical inaccuracies can be noted. Straight lines are not preserved, columns and planes appear distorted. Given the quality of the reconstruction, the user is able to clearly recognise the specific place but, despite the impressive results offered by Google's platforms, can not yet be fooled into believing that the reconstructed images are real.
The user is thrown off by various visual artifacts and, the desired "full immersion" feeling, is yet to be achieved.



(a) A real picture[3]. (b) Google Earth's reconstruction[9].

(c) Detail of (a) (d) Detail of (b)

Figure 1.1: *A comparison between a real image and Google Earth's reconstruction.*

Virtual Reality. Being able to model real objects in a 3D simulated environment is essential in the field of virtual and augmented reality. Every aspect of virtual reality revolves around making the simulation seem as realistic and immersive as possible. The goal is to produce an immersive experience and to "trick" the user into believing everything he/she sees, hears or feels, is real.
During the recent years multiple devices have been developed in order to provide virtual reality to users. The majority of these devices are virtual reality goggles. They comprise a stereoscopic head-mounted display (providing separate images for each eye), stereo sound, and head motion tracking sensors. Inherently from the stereoscopic nature of such devices, realistic image rendering ad 3D re-projection become crucial for this application.
Generating extremely realistic re-projections of the 3D objects over the stereoscopic display is key to give the user the overall visual illusion of a virtual reality. At the moment most of the 3D scenery utilised in virtual reality applications is hand-designed by professionals which model every geometry, texture and material by hand.
This industry field could really benefit from novel view synthesis over urban environments. Imagine wanting to create a simulation in a know place. Rather than having to manually model the given environment, some could simply provide a series of images of the given place and produce directly a 3D model. Such model could be immediately projected onto the virtual reality device, and the user could experience a simulation of the chosen place.

Video-games. A similar application to virtual reality is the world of video-games. Video-games have the same premise as virtual reality: give the user an immersive and realistic experience. Being able to produce high quality rendering and faithful imagery is key feature in the modern entertainment industry.

Movies and CGI. Similarly, in recent years, the movie industry has seen an exponential use and development of modern technology. By taking for example recent colossal movies like Marvel's Avengers, we can observe in Fig 1.2, how actors are nowadays acting in a complete green screen environment. During post-production, the scenery is edited and added in using CGI (Computer Generated Imagery). A faithful and realistic CGI is key to the enjoyment of the viewer. Realistic image rendering represents then a key feature for the development of these applications.

(a) Green screen scene during filming. *(b) Final movie shot with CGI post production.*

Figure 1.2: *A comparison between the same shot during filming and after CGI post-production. Images taken from the Marvel's colossal movie "Avengers: Endgame". Refer to [8] for a more in depth analysis of VFX in the movie.*

1.2 Overview

Image based rendering and novel view synthesis are complex ill-posed problem where we want to estimate a model whose number of parameters is much larger than the effective available measurements.
All the already existing approaches show one or more notorious visual artifacts, Fig 1.3: eroded or floating geometries, texture inaccuracies, lighting issues, wobbly lines and bumpy walls are well-know issues in the context of image based rendering.

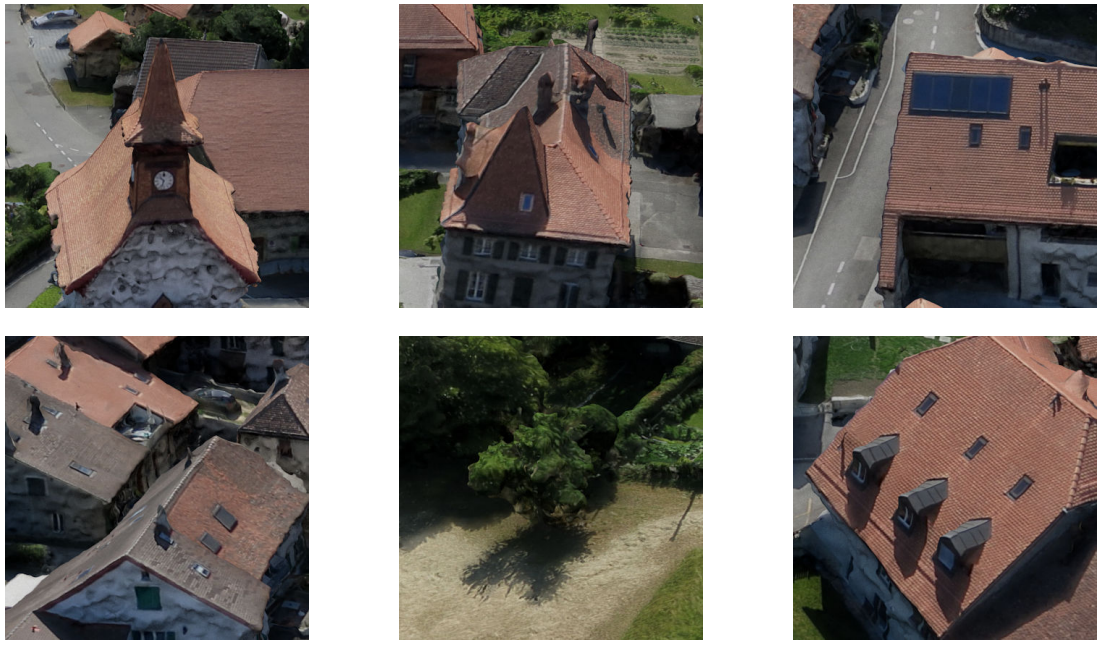


Figure 1.3: *Some examples of the visual artifacts that our models should prevent.*

Throughout the realisation of this work, we develop various methods aiming to improve image rendering quality over urban environments. Inspired by the recent developments of GAN models [20] [46] [33] and neural rendering applications [29] [39], we investigate these very successful areas of research, and we propose a new neural model which, from a set of images and a 3D textured mesh, is capable of significantly improving image quality and the overall realism of generated images.

We firstly explore the field of GAN models. During our experiments we propose a new rendering pipeline that directly operates over imperfect rendered images to produce new generated images. We show great results in terms of generated image quality and improvements in overall image realism. Specifically, the GAN approach excels in texture and color improvements but still suffers from geometrical imperfections.

We then move our interest to geometrical optimisation. We deploy a differentiable rendering framework and explore per vertex optimisation over vertex color and location, and shape aware geometric optimisation using Geometrical Deep Learning techniques. We show that our approach produces impressive results in geometry refinement, lacking although high-frequency texture details in the generated images.

Finally we introduce Morph-GAN, a hybrid model that combines advantages of both the GAN and the geometry refinement approaches. We present results over multiple image patches featuring common visual artifacts showing how our model, Morph-GAN, is able to generate artifacts-free images, Fig 1.4.

Figure 1.4: *Some results of the Morph-GAN model over the same image patches of Fig 1.3.*

2

Background

2.1 Dataset generation

In the context of our application, we work with some small urban scenery. The approaches proposed in this work can be easily extended to greater scale environments given enough computational power. Specifically we worked with four different datasets:

- **Sullens**. This dataset is build by a singular drone flight over the small village of Sullens, Switzerland. This dataset features 37 images with a resolution of 3000x4000 pixels. For additional details on the drone refer to [35] and for more details about the dataset refer to [34]. Some images of this dataset can be found in Fig 2.1.
- **Trinity**. Similarly to the Sullens dataset, this dataset is also generated by a drone flight over the Trinity university campus in Dublin, Ireland. This dataset features 800 pictures with a resolution of 3000x4000 pixels. For additional details on the drone refer to [6] and for more details about the dataset refer to [21]. Some images of this dataset can be found in Fig 2.2.
- **Rue Monge**. This dataset features 428 pictures taken from a hand-held camera during a walk through Rue Monge, Paris, France. Each picture has a resolution of 4000x3000 pixels. Refer to [15] for additional details over this dataset. Some images of this dataset can be found in Fig 2.3.
- **Merton**. This dataset only features 3 pictures with a resolution of 1024x768 pixels. The pictures portray some historical buildings in the Merton's College, Oxford, United Kingdom. Refer to [36] for additional details over this dataset. Some images of this dataset can be found in Fig 2.4.

Some examples of pictures taken from each dataset can be seen in the following figures.

Figure 2.1: *Two pictures from the Sullens Dataset.*

Figure 2.2: *Two pictures from the Trinity Dataset.*

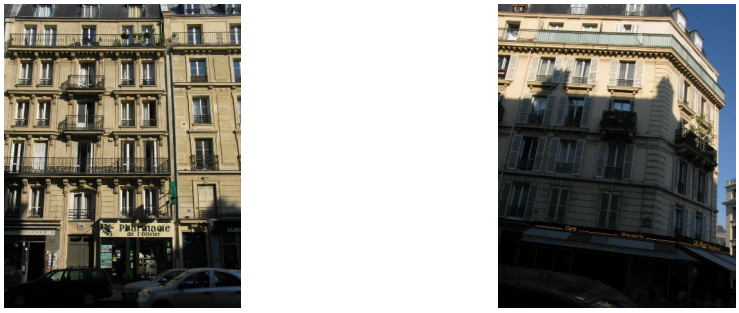


Figure 2.3: *Two pictures from the Rue Monge Dataset.*

Figure 2.4: *Two pictures from the Merton Dataset.*

In order to be able to synthesise some novel views given the set of pictures, we firstly need to reconstruct a 3D model for each of the given datasets. We can then render new views from the obtained 3D models.
We deploy a classical computer vision pipeline. We derive a point-cloud from the pictures using Structure From Motion (SfM). Subsequently we perform triangular meshing and texturing over the obtained point-cloud.

2.1.1 Structure from Motion

Structure From Motion (SfM) is a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences. SfM is a classical procedure developed in the fields of computer vision and visual perception.

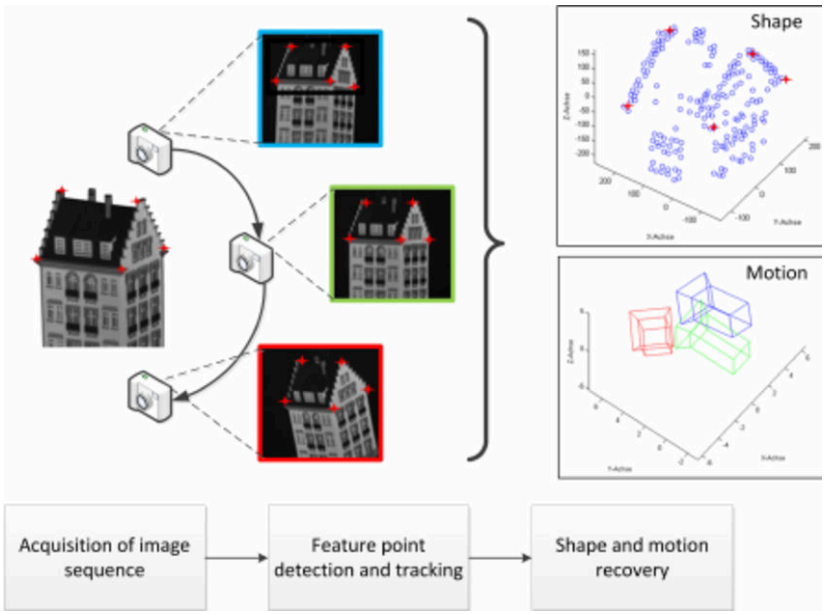


Figure 2.5: *Structure from Motion pipeline as in [5].*

SfM is built around the phenomenon of motion parallax and basic human perception. Humans obtain most of the information about their surrounding by moving around and exploring. While moving around, the surrounding objects move with different amounts depending on their distance from the observer. This is motion parallax. These variations in distance offer depth information which can then be used to generate an accurate 3D representation of the surrounding world.

Finding structure from motion presents a similar problem to finding structure from stereo vision. In both scenarios, the correspondence between images and the reconstruction of 3D object needs to be found.

To find correspondence between images we need to extract some salient features in each image. Such features usually reside in corner points which are tracked from one image to the next. One of the most widely used feature detectors in the field of computer vision is the scale-invariant

feature transform (SIFT) descriptor. The SIFT features detected from all the images are then matched. Given all the extracted features and the respective matches, it is then possible to triangulate 3D points with respect to camera coordinates. The result of applying SfM to set of images is then a collection of 3D point called point-cloud Fig 2.6.

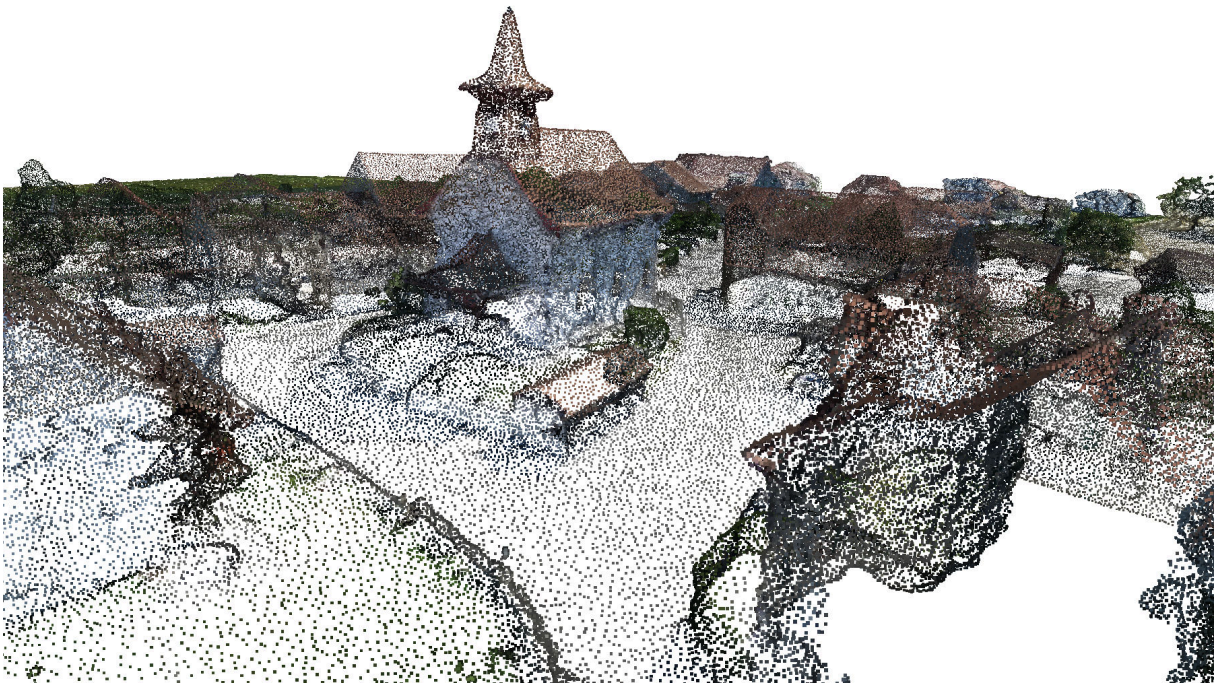


Figure 2.6: *Example of the Point-cloud obtained by applying the SfM algorithm to the Sullens dataset.*

2.1.2 Triangular Meshing

Once a point-cloud representation of the scene is obtained through SfM, we can convert the point-sampled model to a textured mesh model with an approach described by [43] . Triangular textured 3D meshes are some of the most popular representation methods for three-dimensional data in the field of computer graphics. A triangle mesh comprises a set of triangles that are connected by their common edges or corners. The mesh components are vertices, edges, and triangles.

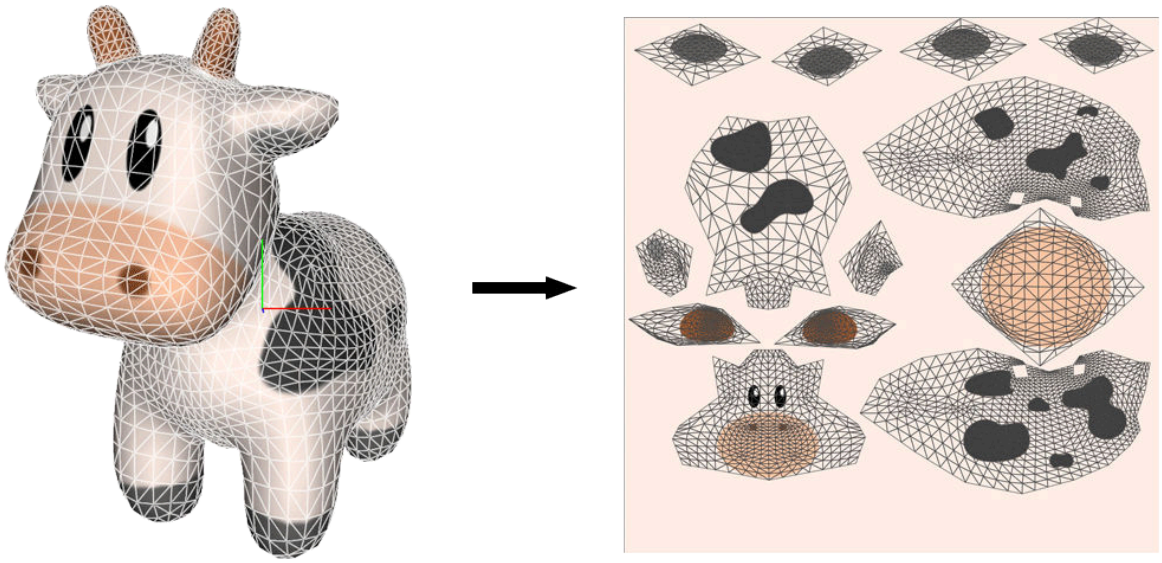


Figure 2.7: *Mesh texture mapping using UV map [30].*

The triangular mesh is produced jointly with a texture image. The mesh contains all the geometrical information of the scene while the texture contains all the visual information. Each triangular face of the 3D mesh is mapped to the produced texture via a UV map like in Fig 2.7, as is standard procedure in computer graphics application. In other words the UV map is used to "wrap" the texture around the 3D geometry. An example of this process can be seen in Fig 2.8 over the Sullens dataset mesh model.

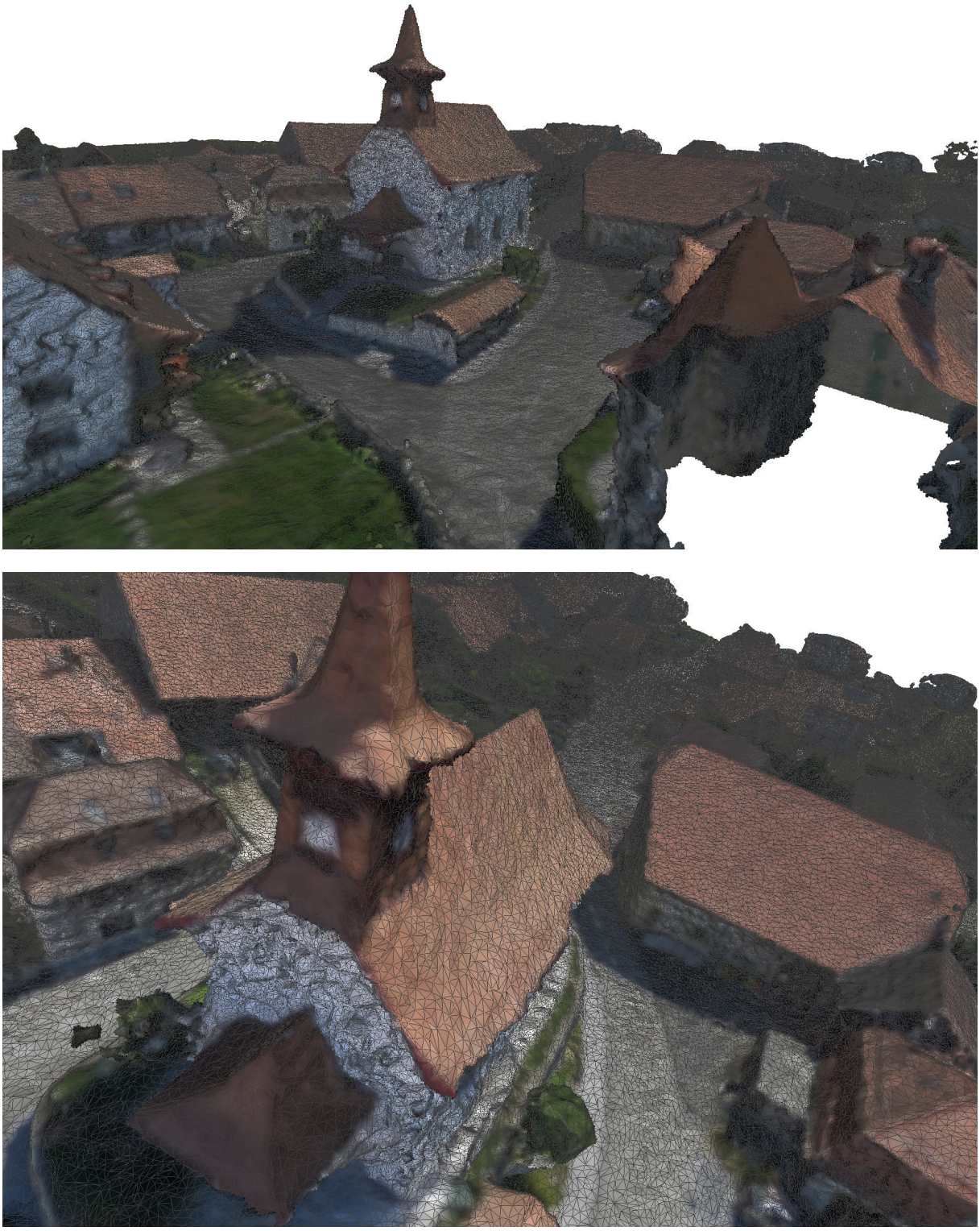


Figure 2.8: *Textured mesh obtained from the point-cloud of the Sullens dataset.*

2.2 Classical Rendering Pipeline

By using the SfM and meshing pipeline previously described, we can obtain a 3D model of any given scene described by a set of images.

Given the textured 3D model, we can now render any novel view by using a classical computer graphics **rendering** pipeline.

Rendering or image synthesis is the automatic process of generating a photorealistic image from a 2D or 3D model given a scene representation containing geometry, viewpoint, texture, lighting, and shading information as a description.
Rendering is one of the major sub-topics of 3D computer graphics. Over time, a number of different rendering techniques have been developed.

Rasterization. It is one of the earliest methods for rendering and works by treating the model as a mesh of polygons. Each of the mesh's vertices, which contains information about his position and color, is projected onto a plane normal to the camera. Using the vertices as borders, the remaining pixels of the projected image are filled with colors obtained by bi-linear interpolation. Rasterization is a fast form of rendering and is widely used to this day for real-time rendering applications. One of the main issues with rasterization are aliasing and low-resolution. Such problems have been addressed by anti-aliasing, a process used to smoothen the edges of objects and blend them into surrounding pixels. The main limitation of rasterization is overlapping of neighbouring objects. If surfaces overlap, the last one drawn will be reflected in the render, causing the wrong object to be rendered. To solve this, the concept of a Z-buffer was developed for rasterization. This involves a depth sensor to indicate which surface is under or over in a particular point of view.

Ray Casting. In order to address the problem of overlapping surfaces, ray casting was introduced. In Ray casting, as its name implies, rays are cast onto the model from the camera's point of view.
The rays are drawn out to every pixel on the image plane. The surface it hits first will be shown in the render, and any other intersection after a first surface will not be rendered.

Ray Tracing. Despite the advantages brought by ray casting, the technique still lacked the ability to properly simulate shadows, reflections, and refractions. In order to simulate these important physical elements, ray tracing was developed. Ray tracing works similar to ray casting. Essentially, like in ray casting, rays are cast from the camera's point of view onto the models. Unlike ray casting though, after hitting a point in the model, the original rays emit secondary rays: shadow rays, reflection rays, or refraction rays , depending on the surface's properties. A shadow is generated on another surface if the path of the shadow ray to the light source is hindered by the surface. If the surface is a reflective one, the resulting reflection ray will be emitted at an angle and will illuminate any other surface it hits, which will further emit another set of rays. For a transparent surface, a refractive ray is emitted once the surface is hit by the secondary ray.

Rendering Equation. A full physical simulation of the rendering pipeline can be found in [4] where the total rendering equation is explained and contextualised.

2.3 Visual Artifacts

By applying the classical rendering pipeline to our datasets, we obtain very good results that can be observed in Fig 2.9.
The rendered images are very impressive in realism and fidelity to the original images. Despite this being a difficult starting point given the already excellent quality of the generated results, there is still room for improvement.

(a) Original Image taken from drone.

(b) Output of the full rendering pipeline.

Figure 2.9: *A comparison between the original input image and its reconstruction after the execution of the full rendering pipeline.*

Multiple visual artifacts appear evident in the rendered image. We can obtain some insightful information regarding these artifacts by looking at specific patches of Fig 2.9 marked by red squares. The patches, shown in Fig 2.10, illustrate more clearly how visual artifacts reduce the level of realism of the produced images, causing the observer to doubt their originality.

Rendered image. Target Image.

Rendered image. Target Image.

Figure 2.10: *Various visual artifacts that can be observed in the produced rendered images.*

As shown in Fig 2.10 the visual artifacts are a mixture of geometrical inaccuracies, wrong texture generation and mapping, color and lighting issues and occlusion induced errors.

Most of the geometrical artifacts can be attributed to initial errors in the drone's odometry. An even small error in the a picture's camera position can be amplified during the SfM pipeline and can cause important geometrical inaccuracies later on in the final rendered image. An example of these kind of artifacts can be seen when looking at the church's roof. In this example it is possible to observe inconsistent roof edges, with holes and wobbly lines as edges. These are main example of geometrical inaccuracies that can be seen in multiple areas of the rendered image.

Flat uniform areas such as walls are difficult areas to match between various images. The SIFT descriptors of these areas are in fact very similar and matching them is a non-trivial operation. Therefore, errors in the matching step produce artifacts such as bumpy walls. Looking at the church's front wall and at most of the other buildings' walls we can notice how walls that should appear uniform and white, actually appear as bumpy walls. This effect is aggravated by lighting conditions.

Moreover the mesh smoothing operation performed before rendering can cause some artifacts such as geometry erosion or disappearance. This phenomenon can even be cause for "floating geometries". An example of these can be seen by looking at chimneys. Being small geometric protuberances, they are often cut off leaving unrealistic geometrical shapes over the roof which sometimes remain floating.

Texture inaccuracies can be seen when looking for example at vegetation.
In addition the whole image generally suffers from imperfect lighting and wrong color.

Most areas of the generated images suffer from one or more of these visual artifacts. You can find a more detailed analysis of the visual artifacts present in each of the presented patches in the following Table 2.1.
Please note that these patches will be object of tests for all future models presented in this work.

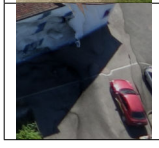
Image Patch	Description	Eroded or Floating Geometries	Texture Inaccuracies	Lighting or Color Issues	Wobbly Lines	Bumpy Walls
	A house with chimneys	X		X	X	X
	A church	X		X	X	X
	Various rooftops	X		X	X	X
	A roof with windows	X		X	X	X
	A roof with vegetation		X	X	X	
	A roof with solar panels			X	X	X
	Some far away roofs	X	X	X	X	X
	A small house		X	X	X	
	A small tree		X	X		
	A very close building and a car			X		X

Table 2.1: *In depth visual artifacts analysis of Sullens image patches. Each X marks the presence of a specific visual artifact.*

2.4 Existing Techniques

Various techniques have been developed in recent years to address visual issues caused by classical rendering and obtain a satisfactory level of realism in the produced images.
The problem is vast and complex by nature and multiple path for improvements are available. One could improve each step of the classical pipeline singularly and produce overall better results. Another could focus on a specific issues produced by the pipeline and tackle that in an isolated manner.
More importantly though, the goal is to produce an overall realistic looking image regardless of the method chosen.
Recently, given the unprecedented advancements in the field of deep learning and computer vision, numerous researchers have applied neural network based techniques to various image based rendering applications.

M. Meshry et al. perform total scene capture in their recent work [29], demonstrating realistic manipulation of image view-point, appearance and semantic labeling.
Their work is primarily focused on capturing tourist landmark around the world using publicly available community photos as sole input. Due to the lack of camera position data, their approach focuses on two main components: the production of an input images factorised representation which separates viewpoint, appearance conditions and transient objects and the rendering of realistic images from the produced factorised representation.
By deploying a very interesting pipeline featuring a neural encoder network for the factorised representation and a neural renderer for image rendering, they show incredible results in terms of image manipulation over appearance conditions.
Although their work focuses mainly on image manipulation over multiple atmospheric, seasonal or time-of-the-day conditions over a much greater set of images, they show good potential in the application of GAN models and neural renderers in a total scene capture setting.

Similarly J. Thies, M. Zollhöfer and M. Niebner use neural renderers and neural auto-encoder models to perform image synthesis in their recent work [38]. Not tailored to urban scene reconstruction, this work proposes a novel approach to scene rendering and texture optimisation. The introduction of learnable and interpretable neural textures depicts a new interesting way to represent realistic 3D content untangling from the constraints of the classical computer graphics rendering techniques.

We take inspiration from both these articles in future chapters of our work: Chapter 3 and Chapter 4.

In the context of image relighting, J.Philip et al. in [31] propose the first learning-based algorithm that can relight images in a realistic and controllable matter given multiple views of an outdoor scene. They introduce a geometry aware neural network which receives normal maps, specular directions and other geometrical cues, jointly with target and source shadow masks.
The proposed approach expects the deployment of three neural models: one for source shadow mask refinement, one for target shadow mask refinements and a third model for final image relighting.

While most of these works aim at fixing texture, color and lighting or change the scene appearance conditions, they do not optimise the 3D geometries.

Alternatively, as shown in the work of P. Hedman et al. in [18], a perfect 3D model representation is not required in order to perform image rendering obtaining limited geometrical artifacts. Given the numerous visual artifacts caused by geometrical modeling and mesh reconstruction, they propose a new method that bypasses the need for a good geometrical representation by blending together warped images obtained from various viewpoints.
Similarly to the work of [11], after applying different multi-view-stereo approaches to carry out per view geometry refinement, they train a neural network to blend in the multiple imperfect rendered images into one. The model learns blending weights to combine the input images.

Finally an interesting approach is proposed by J. Flynn et al. in [10], where the use of multi-plane-images (MPI) as in [45] is proposed alternatively to 3D geometries. The model learns MPIs from the set of input images, and then overlaps them to produce novel views. This approach is proved to be very effective for small changes in view-point locations.

3

Generative models

3.1 Introduction

In recent years an increase in popularity among researchers can be seen in the use of neural generative models. Generative models are a class of statistical methods whose goal is to estimate the joint probability distribution of data $P(X, Y)$ between the observed data X and corresponding labels Y.
Thanks to the developments in Convolution Neural Networks (CNN) following the works of [24], neural generative models gained more and more popularity among researchers.
In particular, a specific subset of generative models have been used extensively in a huge variety of fields of research: Generative Adversarial Networks (GAN) models.

Firstly introduced by Ian Goodfellow et al. in [14], GAN models exploit an adversarial process to train two separate multi-layer perceptrons: a generative model G, called **generator**, that captures the data distribution, and a second model D, called **discriminator** whose goal is to estimate the probability that a given sample $\widetilde{x}$ came from the data distribution $P(X)$ rather than G.

The proposed framework consists in a two-player min-max game between **generator** and **discriminator** with the following value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log (1 - D(G(z)))]. \tag{3.1}$$

Where in order to learn the generator's distribution p_g over data x, a prior on input noise variable $p_z(z)$ is defined. Then a differentiable function G is defined to represent a mapping from noise z to data space: $G(z; \theta_g)$ parametrised by θ_g. Similarly a second differentiable function D is defined as $D(x; \theta_d)$ where $D(x)$ represents the predicted probability of $x \sim P(X)$.
The objective of the generator G is to produce an output $G(z)$ which will induce the discrimi-

nator D to produce a wrong classification. At convergence, the generator has learned to sample from z and produce new elements $\widetilde{x} \in G(z)$ which are indistinguishable from $x \in P(X)$.

3.2 Related work

Since the publication of [14], GAN models became some of the most widely used models in a majority of relevant research fields.
In particular GAN models found immense usability in the fields of image style transfer and image to image translation, furthermore representing a very useful tool for data augmentation in a much wider range of applications.

Image style transfer is a specific area of research that features models whose goal is to create artistic images of high perceptual quality.
Firstly introduced in [13], artistic style transfer gained more an more popularity in the world of image generation. Gatys et al. provide a model that uses neural representations to separate and recombine content and style of arbitrary images.
The intuition of the authors comes from a deeper analysis of CNN networks and their level of information abstraction throughout the network. They show how the generic feature representations learned by high-performing Convolutional Neural Networks can be used to independently process and manipulate the content and the style of natural images.
As reported by the authors, CNNs trained on object recognition, develop a representation of the image that makes object information increasingly explicit along the processing hierarchy.
In other words, the deeper into the network, the more low-frequency information is represented. The first convolutions of a network capture very high-frequency information, pixel level information. The final layers, whose output is usually used to perform image classification, have to carry information regarding the full image.
In this work they propose a continuous image optimisation over a loss computed at multiple network layers: the produced image has to have an high-frequency representation similar to the artist's painting (Style loss), and a low-frequency representation similar to the original input picture (Content loss). By enforcing this constraints, they guarantee that the produced image will preserve the original content of the input image but will assume the style of the artist's paintings.
Additional improvements to Gatys' technique can be seen in [42] where additional losses are implemented in order to capture small and complex textures. While this is not per se a GAN model, the work proposed by Gatys et al. has inspired various other researchers and evolved into a core GAN application. The concepts of style and content loss are key in the developments of a multitude of future works.

Among the others, a very promising evolution in the area of the style transfer, is presented by A. Sanakoyeu et al. in [33]. The proposed model called ArtGAN is a GAN evolution of the Gatys approach.
The image that need to be transformed is given to a generator auto-encoder model. The encoder part of the generator encodes the input image into a latent space. The decoder then reconstructs an image from the given latent space. The produced image is given to a discriminator classifier which has to decide if the produce image is a real painting of the artist or a fake.

While Gatys' approach produces great results, it remains a singular image optimisation technique; the style of the artist is not actually learned. With the ArtGAN model, the generator, with the help of the discriminator, learns how to reproduce the artist's style and is then able to transform any image with the learned style.
In this application the encoder learns to extract content information, while the decoder learns to transform the content and add proper style information. A style aware content loss is introduced to preserve the content information while at the same time reproduce the artist's style.

A similar approach to style transfer can be seen in an application closer to our own in [27]. In this work style transfer is used over various images, mainly urban environments, to modify the time-of-the-day/lighting conditions.

Alongside style transfer, another very popular research area for GAN models is image-to-image translation.
Many problems in image processing, computer graphics and computer vision can be posed as "translating" an input image to a corresponding output image.
One of the most innovative works in this field is the Pix2pix model of [20]. The Pix2pix model proposed by P. Isola et al. represented a huge turning point in GAN models evolution and applicability.
In their work the authors propose the use of a GAN model to learn the "translation" transformation needed to go from a group of input images A to an group of output images B. The generator receives the input image and generates an output image. The output image is passed to the discriminator which has to evaluate if the generated image is a fake or if it actually belongs to the group of images B.
Isola et al. work with paired image data: for each input image they also have the corresponding target output image. Having paired data they can then take advantage of a powerful loss function to accompany the classical GAN loss: the L1 loss.
As impressive as the results of [20] are, they suffer from a major downfalls of the L1 loss: the production of blurry artifacts. Moreover the approach is limited to paired image data which is in practice rarely available.
The pix2pix model has been subsequently improved by Wang et al. in [41] with the introduction of a deeper and more complex network which features res-blocks [17], multi-scale discriminators, coarse-to-fine generator and improved adversarial loss.

Motivated by the limitations of the pix2pix model, Zhu et al. in their work [46], propose the CycleGAN model.
The Cycle-GAN works with unpaired image data and exploits a clever trick to avoid the use of the L1 loss. The model's goal is to learn a mapping function $G : X \rightarrow Y$ such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, they couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a cycle consistency loss to enforce $F(G(X)) \approx X$ (and vice versa).

Inspired by the incredible results obtained by these models we deploy some of these models in the context of our application.
An more in depth analysis of the previously referred losses can be found in the following Fig 3.2.

3.3 Approach

Motivated by the impressive results obtained by GAN models in the domains of image-to-image translation and style transfer, we want to exploit their properties in the context of our application, image based rendering and novel view synthesis, in order to generate more realistic images. We propose the following rendering pipeline as shown in Fig 3.1.

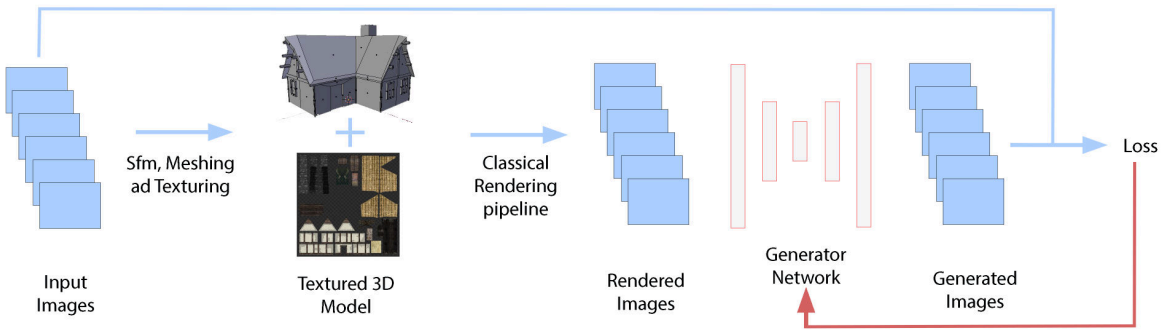


Figure 3.1: *New proposed rendering pipeline with GAN integration. Images from [2].*

We deploy the original rendering pipeline to produce a dataset of rendered images. The datasets are then a collection of image couples. The paired data looks as follows: rendered images as input and original images as target. We split this new produced datasets into train and test sets. Our models during training receive the set of rendered images as input and the real pictures as targets in the loss function.

If we consider our problem under an image-to-image translation perspective, we want to learn a mapping $G : X \rightarrow Y$ where X represents the set of rendered images coming from the classical rendering pipeline, and Y represents the set of target images which are the real images.
Ideally we would like the neural model to translate the input rendered image into a more realistic image. The generator of the GAN model should detect the inaccuracies and artifacts of the input rendered image and learn to fix such issues to produce a more realistic image that the discriminator could not tell apart from a real image.

Alternatively we can look at our problem as a style transfer application. Similarly to various style transfer approaches [13] [42] [33], where a famous painter style such as Van Gogh or Monet is applied to real images to produce new paintings, we would like to apply an "ultra-realistic" painting style to our rendered images to produce "new" realistic images. Equally to how the neural model learn how to simulate an artist brush and technique over a real image, we want our model to learn how to simulate a realistic technique over an imperfect input image.

With these goals in mind, we explored a variety of GAN models in various domain and applications.

Every GAN model has by definition two main components: the generator and the discriminator. Both the generator and discriminator are multi-layer perceptrons. In various works multiple architectures for these models have been proposed.

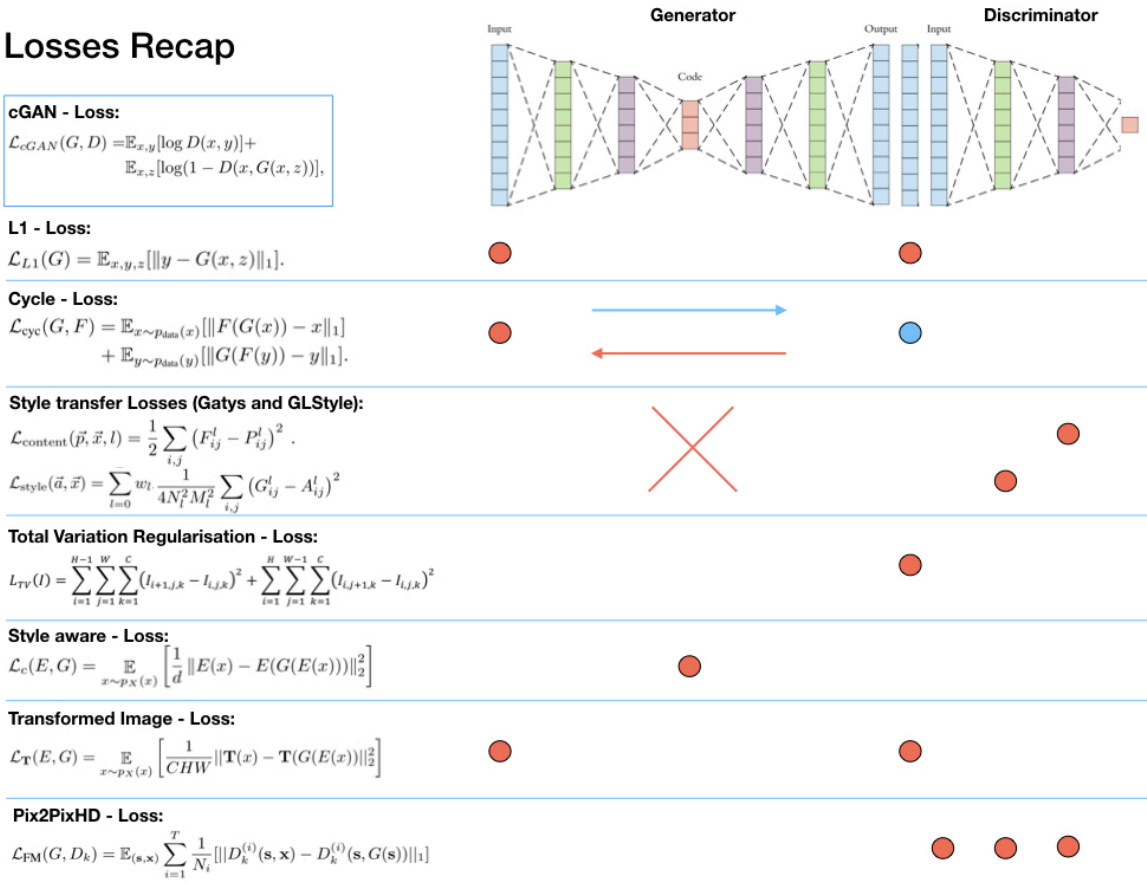


Figure 3.2: *Comparison of various GAN losses. Specifically to where they act in the GAN architecture.*

Among the different architectures, the U-net as proposed by Olaf Ronneberger et al. in [32], represents the most popular architecture used to build the generator. Concerning the discriminator, any multiple multi-layer convolutional networks can be used.

Throughout our experiments we deploy a U-net architecture for the generator, and a simple convolutional network as discriminator.

Although the generator's and discriminator's architectures are important components of any work involving GAN models, they are not the main cause of performance enhancements.

The main component of any GAN model, one could say generally of any machine learning application, is the choice of the loss function. The choice of the loss function is in fact the key element that determines the behaviour of the model. Having the same generator-discriminator structure with different losses changes drastically the outcome of the generated images. We present here in Fig 3.2 a comparison between various GAN losses as proposed in the aforementioned papers of Section 3.2. All the presented losses accompany the original GAN Loss and are tailored additions aimed at fixing specific problems in image generation. The majority of the proposed losses act directly on the output image of the generator. These losses tackle directly the behaviour of the generator network enforcing some properties in the resulting generated images. Some others, mainly in style transfer applications, act on the intermediate layers of the discriminator network. It is shown in previous work by Gatys et al. [13], how any con-

volutional classifier detects low level information in its first and less deep layers. The deeper into the network the more complex and high-level the information becomes. Multiple losses exploit these properties by enforcing the discriminator to penalise differences in either high or low level information. An example of such technique is noticeable in [42] [33].

During our experiments we train three different baseline GAN models over their respective losses: Pix2Pix [20], CycleGAN [46] and ArtGAN [33].

Pix2pix. Our implementation of the Pix2pix model preserves the original architecture of [20]. Pix2pix minimizes both the GAN Loss and the L1 loss over target images. By minimizing this loss, the generator is forced to produce "near the ground truth" output in a L1 sense. Known issues of this loss are the production of blurry images: the loss fails to encourage high-frequency crispness but accurately captures low-frequency details.
In other words this loss allows the generator model to produce good images in terms of content but doesn't include enough information to produce very specific details and textures. In order to improve the generation of high-frequency information it is beneficial to restrict attention to local patches: a Patch GAN approach is implemented penalising structure at scale of patches.

We train our model over random 768x768px patches of the original images. Horizontal flipping is carried out randomly as well. We deploy this data augmentation pipeline to make the model more robust and to augment the size of our training sets. The model is trained for 10000 epochs.

CycleGAN. Our implementation of the CycleGAN model preserves the original architecture of [46].
The objective of any GAN model is to produce $G : X \rightarrow Y$ such that $\hat{y} = G(x)$ is indistinguishable form $y \in Y$. In theory this objective can induce an output distribution over $\hat{y}$ that matches the empirical distribution $p_{data}(y)$. An issues with this assumption relies in that fact that G does not guarantee that x and y are paired in a meaningful way. There are infinitely many other G functions that can produce $\hat{y}$.
Whilst this issue is addressed by Pix2pix by optimizing over the L1 loss, in addition to the standard GAN loss, in the CycleGAN application paired image data is not available and the L1 loss is therefore inapplicable.
Ingeniously the authors of [46], came up with a trick to bypass this issue: the introduction of a forward cycle $G : X \rightarrow Y$ and a backwards cycle $F : Y \rightarrow X$. L1 loss can then be computed between an input image $x \in X$ and $\hat{x} = F(G(x))$. The second generator F also ensures that the mapping function $G : X \rightarrow Y$ is bijective. Therefore in the CycleGAN model two different GAN losses are computed: one for the forward cycle and one for the backwards cycle.
As suggested in [33], some known issues of this loss are the production of images where either the content is preserved and the produced image is not sufficiently different in content (no geometrical changes) or the stylised image has a suitable degree of abstractness and a pixel-based comparison with the original image has to fail.

Training details are identical to Pix2pix.

ArtGAN. Our implementation of the ArtGAN model preserves the original architecture of [33]. We deploy the model proposed by A. Sanakoyeu et al. to test the applicability of state of the art style transfer techniques in the domain of our work.
As described in the paper, ArtGAN minimises three different losses: the GAN loss, the Transformer loss over the input image and a Style Aware loss. The Style Aware loss is an L1 loss

between the encoded latent spaces of the input image x and the generated image $G(x)$. By minimising this loss we ensure that the generator model during encoding preserved the content information necessary to reconstruct the image. The encoder extract information regarding the general "content" of the image whilst the decoder fills-in the "style" information such as texture and color. The transformer loss is a weak form of L1 loss whose goal is not to penalise changes in content to a certain extent: a small transformation is applied to both the input and the target image which are then compared in a L1 fashion. Information of target images is passed to the model only in the discriminator phase, therefore is accounted for during optimisation within the GAN loss.

The model's architecture is not a classical U-net architecture, the generator features multiple res-blocks [17] at latent space level and there are no skip connections. Input images are resized to a fixed dimension of 1200x1800px, subsequently random cropping of 768x768px patches is carried out. Horizontal flipping is carried out randomly as well. The model is also trained for 10000 epochs.

3.4 Results

We train the models separately over each one of our previously described datasets. We now consider some examples from the test sets of the Sullens dataset and the Rue Monge dataset.

The image generation power of the Pix2pix model can be witnessed clearly on examples of the Rue Monge dataset in Fig 3.3 where very high-frequency details are filled in by the generator over multiple facades.

(a) Input image.

(b) Target image.

(c) CycleGAN image.

(d) Pix2pix image.

Figure 3.3: *Results on a test image of the Rue Monge Dataset.*

Moreover we can observe the results of Pix2pix Fig 3.5, CycleGAN Fig 3.6 and ArtGAN Fig 3.7 on a test image of the Sullens dataset.

We can observe how Pix2pix achieves quite impressive results. Visual artifacts such as bumpy walls are repaired: flat areas appear uniform and lighting issues are fixed. The colors of the generated images match the colors of the target images. Moreover the texture of numerous buildings and plants seems to be improved compared to the input images.
On the other hand we can also observe various limitations of the Pix2pix model. The most evident visual artifact introduced by Pix2pix is the aforementioned blurriness of some areas of the generated pictures. Moreover we can also notice how the Pix2pix model is not able to fix **geometrical issues** of the input image.
Visual artifacts such as wobbly lines, holes in walls and floating geometries remain evident and the model is not able to fix them.

Concerning CycleGAN, we can observe in the produced images how the model does not learn how to apply major noticeable changes to the input images. We hypothesise that this behaviour happens as a cause to the lack of paired images. The model in order to be able to re-create the original image x from $G(x)$ applies little to none modifications to the original input. Similarly on the Rue Monge and Trinity datasets we can see no real improvement in the quality of the generated images. At times the model introduces some high-frequency information but also introduces heavy granular deformation artifacts.

Figure 3.4: *Input image.*

Figure 3.5: *Pix2pix generated image.*

Figure 3.6: *CycleGAN generated image.*

Figure 3.7: *ArtGAN generated image.*

Figure 3.8: *Target image.*

Interesting results can be observed in the ArtGAN application. The ArtGAN model similarly to Pix2pix, menages to fix the building's walls and the overall color range of the image. Interestingly it appears that the model is able to fix some of the geometrical errors that Pix2pix fails to address.
Unfortunately the generated images look warped and present multiple visual artifacts that make the generated image more similar to an artist's painting rather than a real image.
We suspect that these artifacts are a counter-effect of the same properties that make this model very suited to learn artistic styles. The model needs to extract very basic information from the input image during encoding, the decoding phase is aimed at reproducing the artist's style and transform the "content" captured in the latent space into final generated picture resembling the artist's work. In order to not be affected from the input image as much, the model presents no skip connection and many details of the input image are lost in the encoding process. The model tends to lose a lot of texture and high-frequency information because such information has to be overwritten by the artist's style. On the other hand, in our application we would like the model to preserve some style aspects and discard some of the content lines.. Moreover the transformer loss, necessary to allow geometrical modifications, is cause of multiple warping phenomenons.

Generally all models suffer from the very high resolution of our input images. Generating a very detailed 12Mpx image is a non trivial task which probably requires a greater number of training images and deeper networks.

Some more in depth details of the generated visual artifacts and model behaviours can be found in Fig 3.9 where we isolate image patches representing specific visual artifacts of a test input image.

Also we can observe in the following Table 3.1, how each model behaves with regards to known visual artifacts. Please note that new visual artifacts introduced by the respective models are not taken into account in this table.

Model Name	Eroded or Floating Geometries	Texture Inaccuracies	Lighting or Color Issues	Wobbly Lines	Bumpy Walls
Input	X	X	X	X	X
Pix2pix	X			X	
CycleGAN	X			X	X
ArtGAN	too deformed			X	

Table 3.1: *Comparison between Pix2pix, CycleGAN and ArtGAN over the presence of known visual artifacts. Each X marks the presence of a specific visual artifact.*

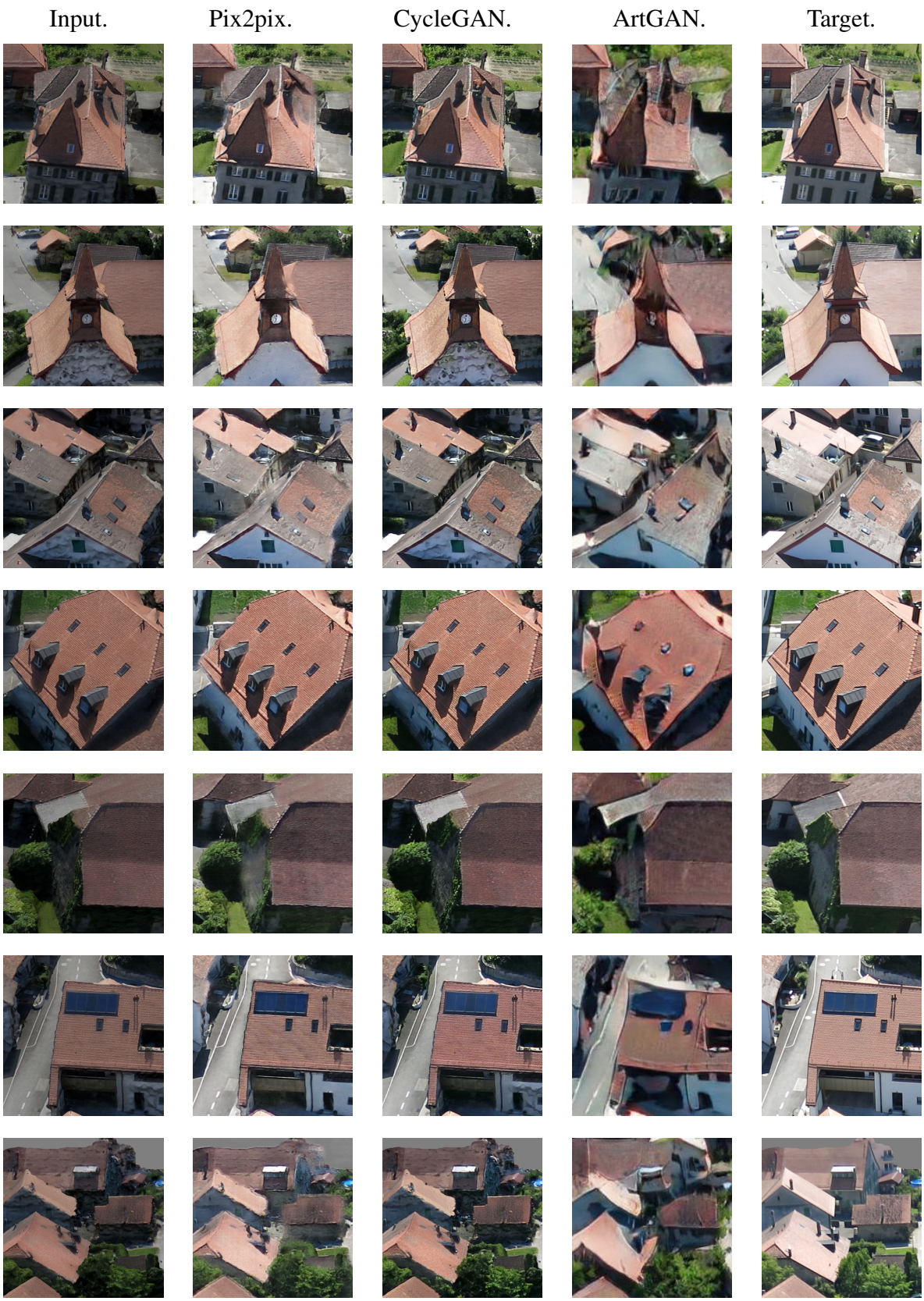

Input.
Pix2pix.
CycleGAN.
ArtGAN.
Target.

Figure 3.9: *Comparison of image details between Pix2pix, CycleGAN and ArtGAN models.*

3.5 Conclusions

The presented experiments, mainly the use of the Pix2pix model, show a very good potential in the deployment of GAN models in the context of image based rendering and novel view synthesis.
We notice the importance and the advantaged brought by the use of paired training data. The L1 loss, if target images are available, brings great advantages in image generation despite the introduction of some blurry artifacts and the lack of very high-frequency information.
Great room for improvement over these issues can be seen in the deployment of deeper and more complex models such as Pix2pixHD [41] which are able to learn and interpolate very detail-rich information.

Overall the application of GAN models to imperfect rendered images has shown promising results mainly in the reconstruction of textures and colors.

4

Neural Rendering and Geometry Refinement

4.1 Introduction

Given the promising results of the application of GAN models to imperfect rendered images, we move our interest to where that approach fails.
By looking at the very good results obtained by the Pix2pix model, we can notice how the most evident visual artifacts still present in the generated images are geometrical inaccuracies. As previously stated, even though Pix2pix menages to fix texture, color and add some visual details in the generated images, it cannot fix geometrical issues.
In fact we can observe how, wherever there are some geometrical imperfections, the Pix2pix model tries to intervene and fix the issues by generating blurry areas. This is a direct effect of optimising over the L1 loss, and represents a key failure of the original model.
A detailed example of this behaviour can be seen in the following Fig 4.1.

Figure 4.1: *Blurry areas around the roof's edges appear evident in the image generated by Pix2pix model.*

With the goal of producing overall more realistic images, we propose a new approach to fix and refine these imperfect geometries therefore tackling directly one of the greatest failure cases of the GAN approach.

4.2 Neural Renderers

In order to be able to enhance and fix the initial 3D model's geometry, we need to intervene before the rendering step happens. Ideally if one could fix the geometry and the texture of the 3D model, the rendering results would be incredibly realistic given the performances of state of the art rendering models.
Unfortunately, in our application we don't have any information regarding the original geometrical shape of the scenery. The only information accessible is the collection of target images from which the 3D model was generated.

We circumvent this issues by deploying a newly proposed tool in the field of computer graphics and vision: **neural renderers**.
Neural renderers are rendering models that are completely differentiable. These models can perform rasterization or ray tracing in a fully differentiable manner.
We want to use these models to bridge the gap between the two separate sections of the pipeline: the computer graphics side (rendering) and the computer vision side (neural model). An example of use of neural renderer can be seen in Fig 4.2.

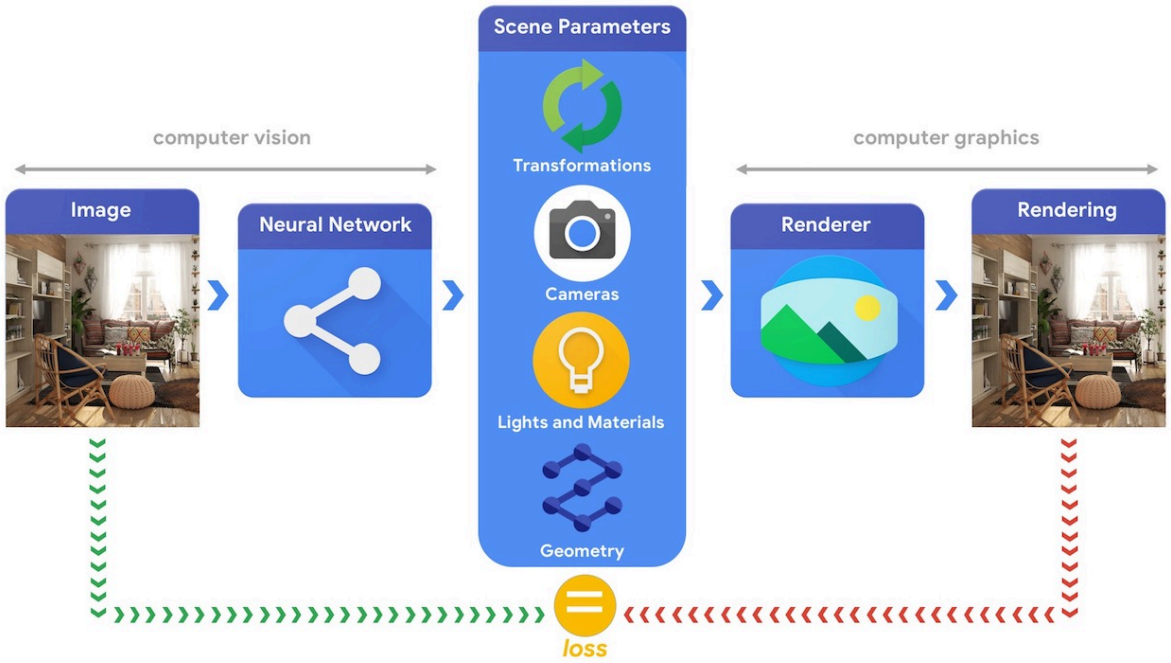


Figure 4.2: *Example of use of a differentiable renderer as shown in [37].*

In practice neural renderer allow gradients to flow through the rendering step during optimization and directly affect the geometry and texture of the 3D model.

The recent introduction of differentiable renderers in the fields of computer graphics and computer vision, opened up endless new research possibilities in both fields.

J. Thies et al. used a neural renderer in their recent work [38]. The authors improve upon the work of [39] by proposing an innovative method of rendering that features fixed 3D geometry and learnable neural texture.
Given the imperfect 3D geometry of the scene they build a network that learns iteratively the scene texture. The texture is not a three channel image anymore, it is a multidimensional array of learnable parameters.
Whenever a new view has to be rendered, given the respective UV, they sample the respective values from the neural texture obtaining, via Z-buffer rasterization a multidimensional image. The sampled image is then passed to a neural network whose role is to interpret the details of the image over the various channel producing at the end a three channel image.
We take inspiration from this work and learn how useful a neural renderer can be. The renderer is in fact the key element of this work, without a differentiable bridge between the texture and the final generated images, neural texture learning and optimisation would not have been possible.

In the filed of Image based rendering, applications of neural renderers can be seen in [29] and [28] where rendered images from a proxy 3D point-cloud are interpreted by a network alongside per pixel albedo, normal, depth and any derivative information.
The main goal of [29] is to render touristic landmarks under multiple conditions. Initially they train an appearance encoder E^a whose goal is to encode the appearance information of a target photo in a latent space $l_{appearance}$. The differentiable renderer is built as a GAN model: the generator is an auto-encoder network which received view specific depth information as input. The encoder encodes the view specific information in a latent space $l_{encoder}$. The two latent spaces $l_{appearance}$ and $l_{encoder}$ are stacked together and finally the decoder interprets the information regarding view and appearance and produces a final image.

An interesting use of differentiable renderers can be seen in [26]. In this work by H. Liu et al., the propose a differentiable triangular mesh renderer that allows users to edit input 3D surface by supplying image processing filters.
During the realisation of our work we use a similar technique for vertex location optimisation.

In our work we use two different neural renderer models called Dirt [19] and Redner [25].

4.3 Per Vertex Optimisation of Position and Color

Within our specific application, we can exploit neural renderers to propagate the computed loss between the rendered images and the target images, directly to the original geometry.

As a first step in order to improve the quality of our 3D model geometry, we define a novel per vertex direct mesh optimisation, or mesh morphing, approach over vertex position and color. A visualisation of this can be seen in Fig 4.3.

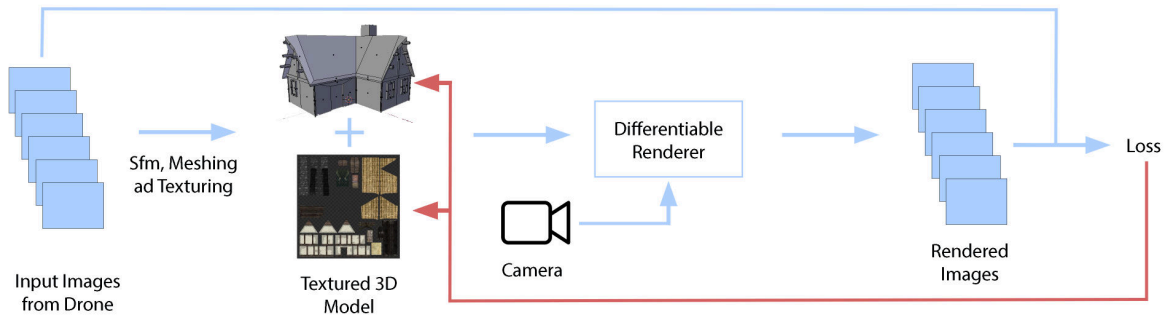


Figure 4.3: *Proposed rendering pipeline with neural renderer integration.*

We consider a textured 3D mesh defined by a set of n vertices $V = (v_1, v_2, ..., v_n)$ and m faces $F = (f_1, f_2, ..., f_m)$. Every vertex is defined by its XYZ spatial coordinates and RGB vertex color coordinates: $\forall v_i \in V, v_i = (x_i, y_i, z_i, r_i, g_i, b_i)$.

We define a set of k cameras $C = (c_1, c_2, ..., c_k)$ where $\forall c_j \in C, c_j = (x_j, y_j, z_j, \theta_j, \gamma_j, \phi_j)$, XYZ being camera spatial coordinates and $\theta\gamma\phi$ being the camera's angles with respect to the world's axis system. Each camera $c_j \in C$ is associated to an image $i_j \in I$ of the considered dataset of target images.

Lastly we define a differentiable renderer R, which performs a differentiable rasterization function. We choose to perform rendering via rasterization considering the already elevated complexity of our task: we want to address basic issues without the additional bias and complexity derived by the addition of lighting, shading or reflection.

The proposed differentiable renderer receives information of the full geometrical model plus a specific camera. The model renders then a view of the model from the specified camera:

$$R(V, F, c_j) = r_j \tag{4.1}$$

where r_j is a rendered image of the 3D model defined by V and F from the perspective of camera c_j. Given the rendered image r_j of camera c_j, we can then compute an L1 loss over the respective target image i_j:

$$L(r_j, i_j) = |i_j - r_j| \tag{4.2}$$

Finally we can optimise the produced loss with respect to the original vertices positions and colors. This step in only possible by exploiting the differentiable properties of the differentiable renderer R. We iteratively modify the original 3d geometry model by changing its vertices positions and colors. The update step for vertex v_i at time t looks as follows:

$$v_i^t \leftarrow v_i^{t-1} + \lambda * \frac{\partial L(r_j, i_j)}{\partial v_i} \tag{4.3}$$

We perform such optimisation for 10000 epochs with a batch size of 3 images per step and a learning rate $\lambda = 0.002$. We stop the optimisation of the model after approximately 500 epochs when the convergence is reached. Convergence varies between various datasets.

Given the nature of our application, we highlight the importance of having a batch size greater than one. We are in fact computing a L1 loss in image space and optimizing over vertices positions and colors in 3D space. If a batch size of one would be chosen, every vertex at each optimisation step could be moved alongside a 3D ray originated from the camera origin to the respective pixel in the image space. This would cause inaccuracies in the optimisation of the vertex 3D position. We therefore use a more comprehensive batch size that allows for a more stable optimisation following the principles of standard triangulation of a point in 3D space.

We apply this method to the original 3D model geometry defined by vertices positions and colors. Texture information is ignored. Results can be seen in the following Fig 4.6 where per vertex optimisation is applied to the Sullens dataset.

Figure 4.4: *Input image.*

Figure 4.5: *Target image.*

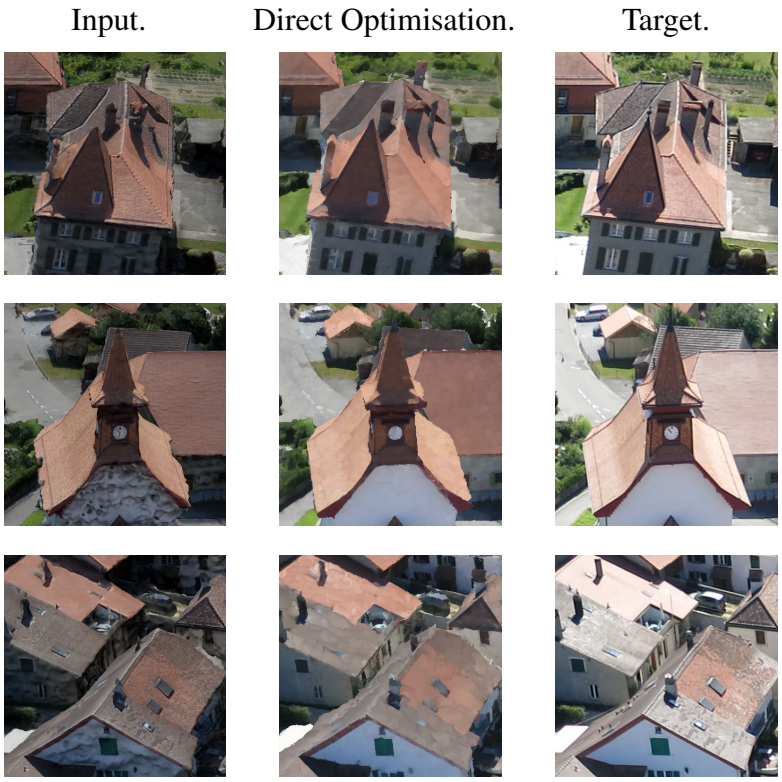
Figure 4.6: *Image rendered over optimised geometry and color.*

The images rendered from the morphed 3D mesh show some incredible geometrical improvements. The model refines the 3D geometry to render images that will be as similar as possible to the target images. Not having any information about the texture, the optimisation is carried out exclusively over vertex position and color.

We notice in Fig 4.7 how all geometrical visual artifacts present in previous models such as floating geometries, wobbly lines and bumpy walls, have now been fixed. The model optimises the geometry by changing the position of various vertices and straight lines are generated naturally from the faces triangular shapes.
The produced images look sharp and geometrically accurate. The color range has been adapted to that of the target images as well.

We can have a better understanding of the actual effects of this model by observing some image details in Fig 4.7. We notice how floating or disappearing geometries such as chimneys are reconstructed. Moreover roof edges are recovered.
We can also observe in multiple patches how wobbly lines have been fixed. Every roof and building border line look smooth and straight. Furthermore walls appear flat and uniform.

Obviously, due to the lack of texture, the images do not look realistic: the high-frequency information (introduced by the texture) is missing. We can summarise the effects of this approach as a geometrical low-frequency optimisation over a set of target images.



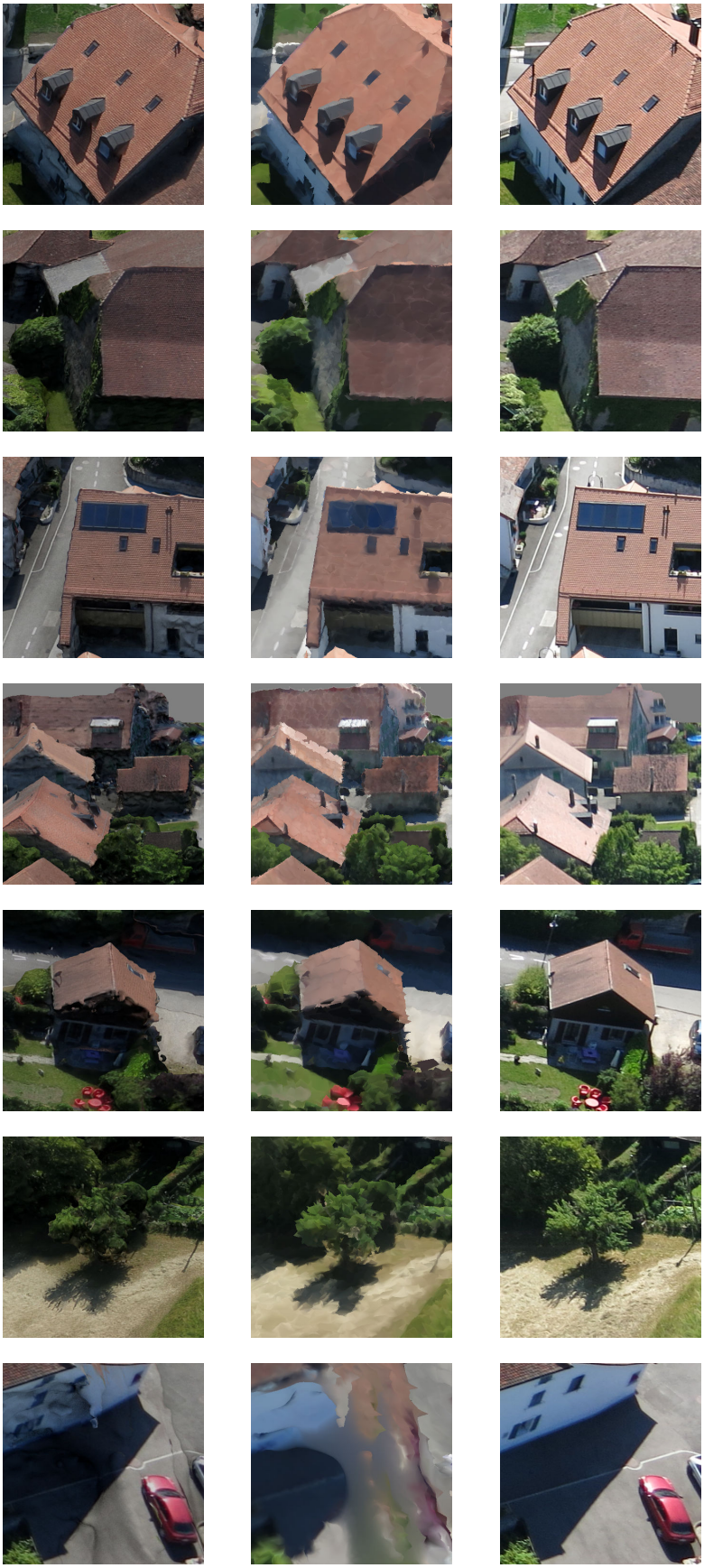


Figure 4.7: *Comparison of image details between input image, per vertex optimisation rendered image and target image.*

Although the results in image space appear to be exceptionally good, what exactly happened to the actual 3D geometry during the optimisation?
In order to have a better understanding of how exactly the geometrical model is being optimised, we have to look directly at the produced 3D mesh rather than at its image projections. We can have a better look over the final optimised geometry in Fig 4.8 .

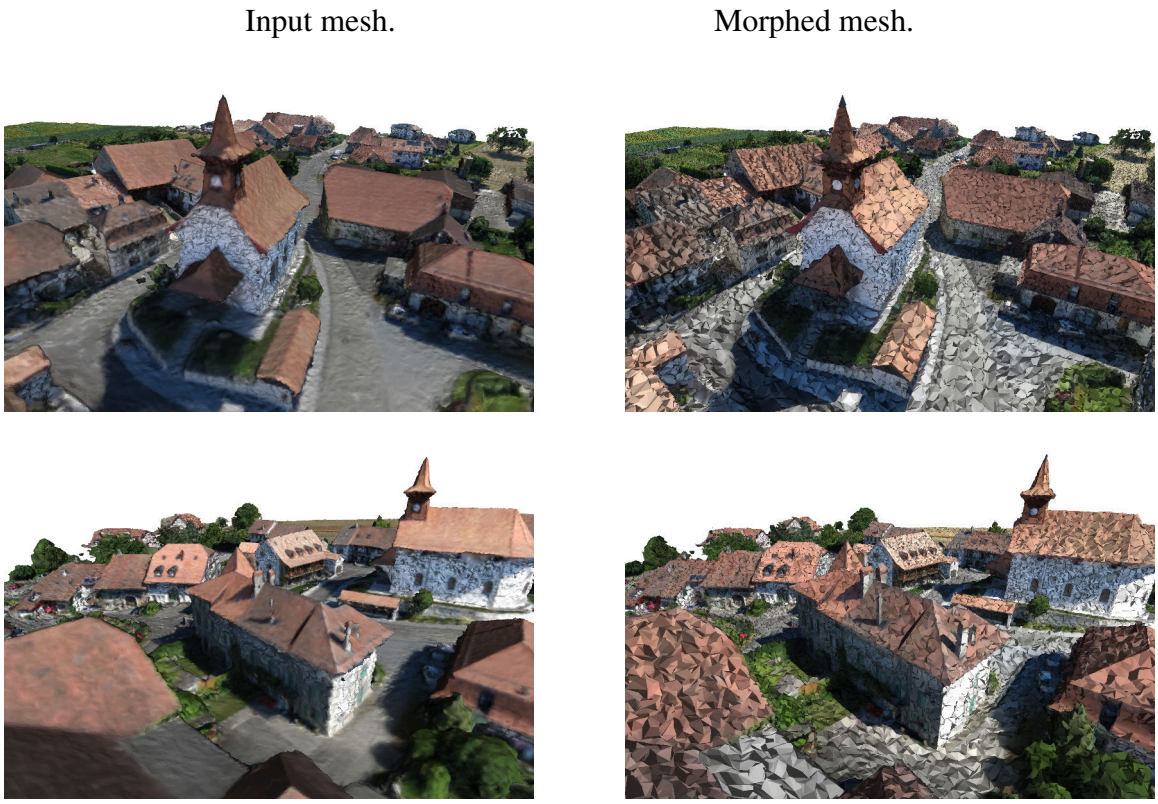


Figure 4.8: *Multiple views over morphed mesh of the Sullens 3D model.*

Overall the geometry is enriched and refined: chimneys are rebuilt and roof edges are straightened. The geometry looks crispier and resembles much more a realistic urban geometry compared to the original one.
It is fairly easy to notice how, during optimisation, bigger triangles are produced. The model produces wider triangles to tackle the issue of imperfect lines and borders: a bigger triangle whose side is aligned with the side of a roof can in fact be visualised as a straight line, fixing in fact the wobbly lines artifacts.
The lack of texture information allows the model to perform relatively big movements over vertex positions.

We can go as far as saying that the model is "cheating" the computed L1 loss. Although the visual results are quite astounding in terms of refinement of geometrical artifacts, the actual geometrical model is not yet perfectly fixed by the optimisation.

The loss function is in fact computed over images and acts on the image space. This causes the optimisation to be almost entirely free in the geometrical world. As long as the projected results on the image space improve the loss function, there are no restrictions on what happens to the

geometry during the optimisation.
A great number of vertices is therefore covered by triangular surfaces. Numerous spikes are forming and the overall geometry in not entirely realistic.
All of this issues are not actually captured by the L1 loss because these artifacts are not evident in the rendered images.

A major role in this issues is played by the limited amount of camera view-points available in the dataset. The more cameras available, the less evident these geometrical imperfections would be. Some geometrical inaccuracies would in fact be noticeable on the image space if some specific view-points were available as target images. If those images were to be available, the optimisation would adjust the geometry to produce correct looking images for that novel view-point as well.

With the addition of texture information, the L1 loss alone is not sufficient to produce realistic results anymore. Due to the introduction of texture, vertex freedom of movement is drastically reduced and the "big triangle" occlusion trick is not possible anymore. There is simply too much information to be processed and a simple derivation over vertices positions and colors is not powerful full enough to fix any visual artifacts.

Taken into account the issues of the per vertex direct optimisation approach we experiment with a novel approach that aims at producing good results on the image space alongside fixing the inaccuracies of the 3D model geometry.
This approach requires the introduction of a new exciting field of research: Geometrical Deep Learning.

4.4 Shape Aware Optimisation via Geometric Deep Learning

Geometrical Deep Learning (GDP) is a relative new field of research whose goal is to apply deep learning techniques to irregular data structures such as graphs. 3D meshes are easily representable as graphs allowing us to freely use GDP techniques.
Similarly to how convolutions are essential elements for image understanding in the field of computer vision, we want to take advantage of graph convolutions to gain a better understanding of geometries.

As shown in Fig 4.9, we propose to build a deep geometric neural model that works directly on the 3D mesh. Such model is be equivalent to a CNN network but works with geometries instead.

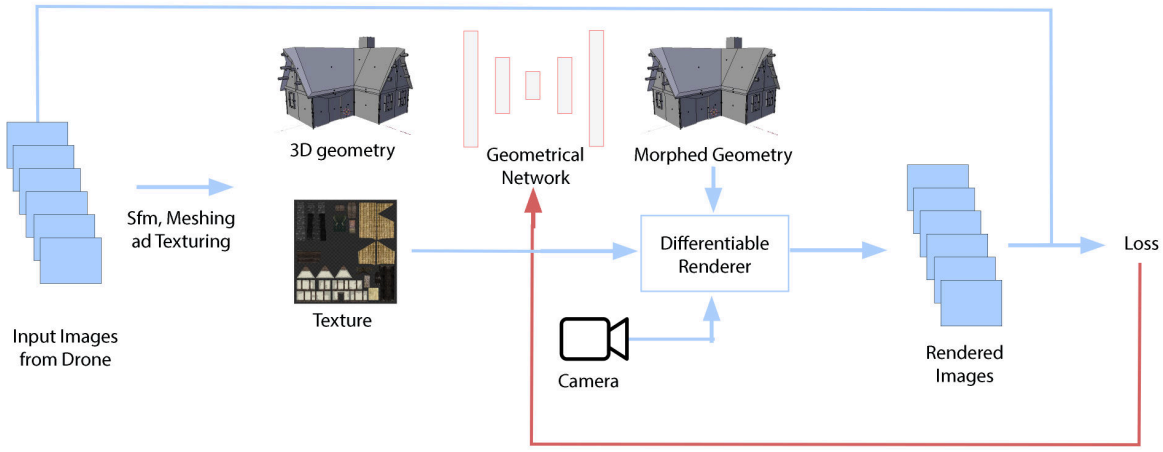


Figure 4.9: *Proposed rendering pipeline with neural renderer and deep geometric network integration.*

With the introduction of a geometrical network, we would like to produce more realistic geometries.
We hope that the introduction of a convolutional model will allow the model to escape the previous per vertex optimisation limitations and transition into a shape-aware geometrical optimisation approach. Ideally the model would learn to analyse groups of vertices and faces at the same time, gradually understanding the features of the geometrical shape, learning therefore to refine them cohesively in order to produce visually correct results.
Similarly to how convolutions on images are used to aggregate information of the surrounding areas of a pixel, we want to use geometric convolutions to aggregate the information of the neighbors of a vertex.

The proposed approach is very similar to the one of per vertex optimisation introduced in Section 4.3, with the following difference: the original geometry is transformed by a geometrical network before being rendered.

Geometric convolutions have been introduced very recently and are object of extensive research in a multitude of fields. Geometric Convolutional Networks are in fact not specific to mesh optimisation or 3D object manipulation, they can work over any kind of data that can be represented as a graph structure making them very useful and applicable tools.
Recently, in the context of 3D mesh classification and segmentation, a very interesting model called MeshCNN has been proposed by R.Hanocka et al. [16]. In this work the authors build a novel geometrical deep learning architecture for shape classification and segmentation based on edge-collapsing pooling layers and edge convolutions.
Previously N. Verma et al. [40] proposed a similar technique for shape classification and segmentation using dynamical filter graph convolutions and Graclus pooling [7].
Unfortunately both methods are not applicable to our work. We want to create a network that learns geometric correlations and, starting from imperfect geometries, is able to refine them. The previously described approaches use graph convolutions to converge shape information in progressively smaller latent spaces which are then subject to classification or segmentation procedures. Although shape information is learned and preserved throughout the network, specific geometric information such as exact vertex position is lost.

For this reason we use various geometrical models built with re-adapted geometrical convolutional layers suited for our applications.

Consider the textured 3D mesh M defined by a set of n vertices $V = (v_1, v_2, ..., v_n)$ and m faces $F = (f_1, f_2, ..., f_m)$. We can define a set of edges $E = (e_{12}, e_{13}, ..., e_{nk})$ where $e_{ij} \in E$ shows the existence of an edge connecting vertex v_i to vertex v_j. The triangular mesh M can also be visualised as an undirected graph G. The graph is defined by a set nodes $N = (n_1, n_2, ..., n_n)$ where $N = V$ set of vertices, and the same set of edges E.
Alternatively the graph connectivity can be fully represented by an adjacency matrix $A^{n \times n}$ where if $a_{ij} = 1$ then vertices v_i and v_j are connected by and edge e_{ij} and if $a_{ij} = 0$ then vertices v_i and v_j are not connected by any edge. Each node of the graph contains information regarding the XYZ coordinates of the respective vertex: $n_i = (x_i, y_i, z_i)$.

We build a geometrical network that receives as input the original graph G representing the original mesh model, and return as output a novel graph G' which can be interpreted as a novel geometrical mesh M'.

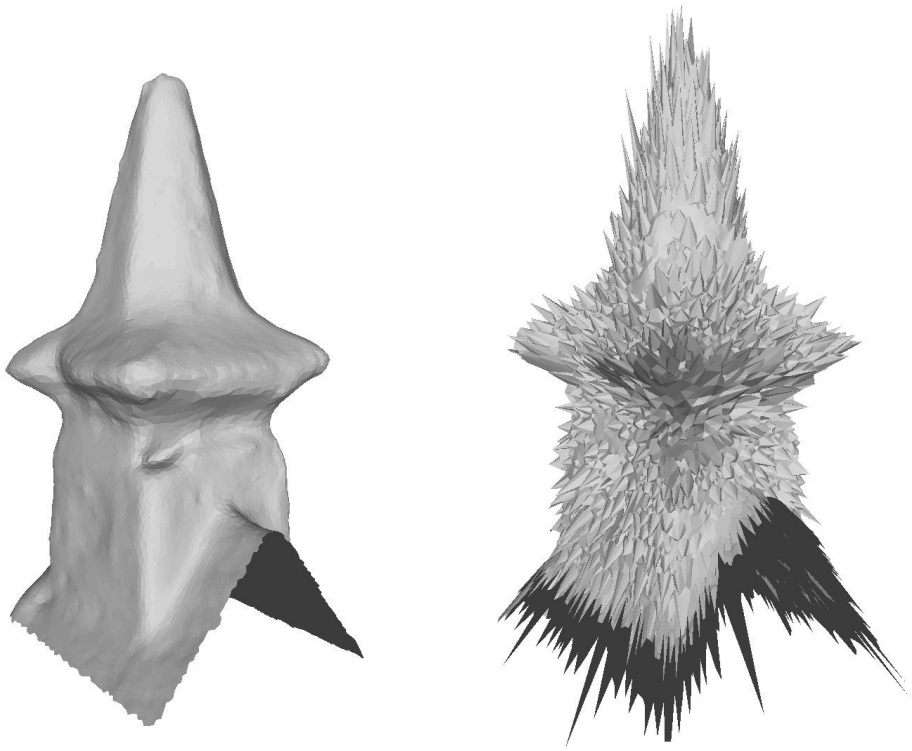
We build our geometrical networks with Graph Convolutions as introduced by T. Kipf and M. Welling in [23]. The introduced graph convolutions build a multi-layer Network (GCN) which performs a function $f(V, A)$, with the following layer-wise propagation rule:

$$H^{(l+1)} = \rho(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}H^{(l)}W^{(l)}) \tag{4.4}$$

Where $A = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections. I_N is the identity matrix, $\widetilde{D}_{ii} = \sum_j A_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\rho(x)$ denotes an activation function, such as the $ReLU(x) = max(0, x)$. $H^{(l)}R^{ND}$ is the matrix of activations in the l-th layer; $H(0) = V$.
In other words, applying a graph convolution over a node, will produce a new node obtained by a weighted linear combination of the the connected neighboring nodes (and itself due to self loops in A) plus a non-linearity operation.

At first, to test the applicability of the proposed network over our geometrical applications, we

train each model over an identity loss. We want the output geometry to be identical to the input geometry.
Due to the complexity of these models, we can not use the mesh geometry in its entirety and, during our experiments, we will consider only the church tower of the Sullens dataset.
As shown in Fig. 4.10, the results are not ideal. The graph convolutions produce spikes over the whole geometry.



(a) Original Geometry. (b) GCN Network Output.

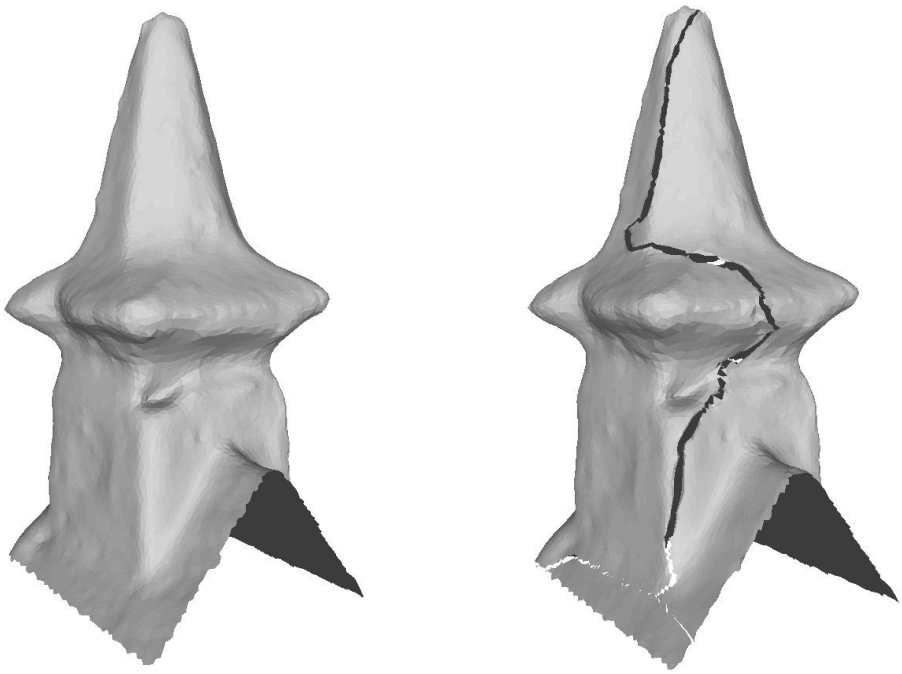
Figure 4.10: *Comparison between original geometry and results of the GCN model trained over identity loss.*

In order to fix this issue, we propose a modification of the original graph convolutions of [23]. The spikes are generated due to an intrinsic issue of graph convolutions. At each convolution each vertex is shifted by a linear combination of its neighboring vertices. The combination is guided by the adjacency matrix A: the new X coordinate of a vertex will be the sum of all neighboring vertices X coordinates normalised by the number of vertices. Once obtained the new XYZ coordinates they are modified via a weighted linear combination to produce the final XYZ coordinates.
On the other hand, when applying convolutions over images, each pixel is is affected by a weighted combination of the neighboring pixels. Similarly we introduce a learnable matrix of weights $M^{n\times n}$ such that:

$$H^{(l+1)} = \rho(\widetilde{D}^{-1/2}(\widetilde{A} \circ M)\widetilde{D}^{-1/2}H^{(l)}W^{(l)}) \tag{4.5}$$

This matrix can be interpreted as a learnable edge weight matrix that regulates the impact of each vertex over its neighbors.

Results of the identity loss optimisation after the introduction of (4.5) can be seen in the following Fig 4.11.



(a) Original Geometry. (b) GCN Network Output.

Figure 4.11: *Comparison between original geometry and results of the GCN model trained over identity loss after the introduction of edge weight matrix M.*

The visible cracks in the geometry are caused by a specific mesh representation chosen in our experiments where the mesh is split into multiple sub-meshes with respect to areas of the texture map.

The model shows good results in minimising the identity loss. We perform ulterior tests by introducing random noise over the vertex positions of the input geometry. The noisy geometries are given as input to the already trained GCN model and the results are shown in Table 4.1.

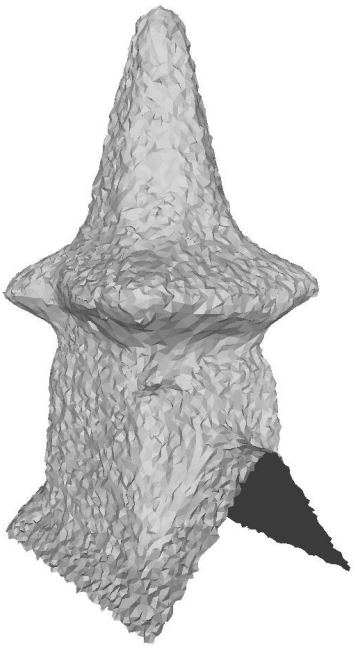
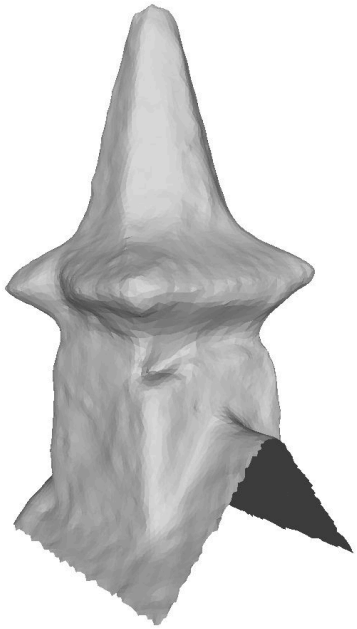
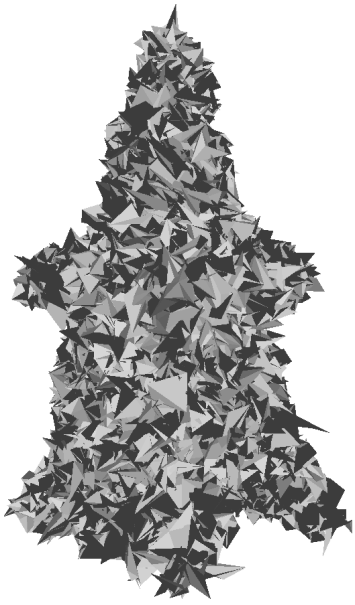
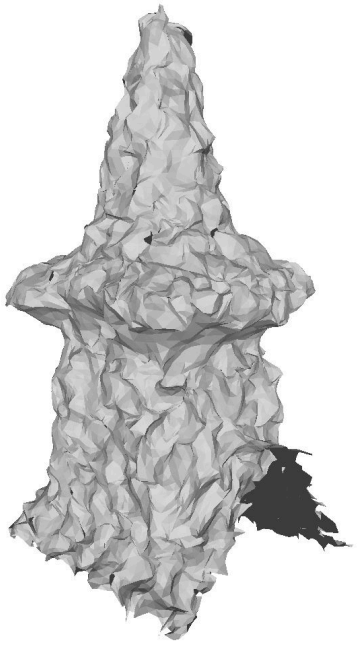
Noise Level	Input Noisy Geometry	GCN Output
$\sigma = 0.002$		
$\sigma = 0.02$		

Table 4.1: *GCN response to noisy input geometries.*

Given the very good results of the modified GCN model over identity loss optimisation and robust denoise testing, we can move forward and deploy the model in the previously described pipeline of Fig. 4.9.

We initialise the model with the weights obtained from identity loss optimisation, therefore making sure that in the first iterations the model generates the original geometry.

We then begin the training procedure: the original geometry is passed to the GCN model which produces a new output geometry. The new geometry is rendered via a differentiable renderer and the rendered image is compared to the target image. The computed L1 loss is back-propagated to the GCN weights.

We show results of the traning in the following Table 4.2.

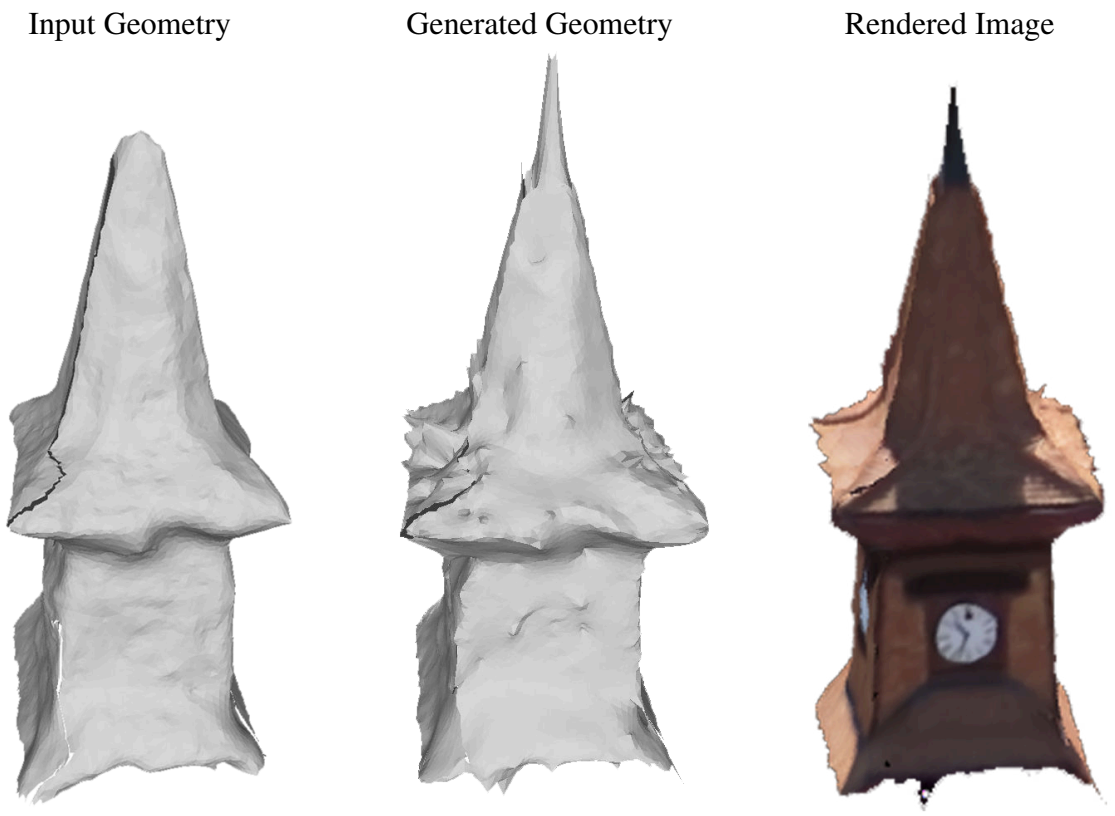


Table 4.2: *Results of shape aware geometry optimisation pipeline using GCN .*

We perform the same experiments over other Graph Convolutional Networks such as SGC as introduced in [44] and Graph U-net an auto-encoder network proposed in [12] which, in addition to graph convolutions, features pooling and un-pooling layers. The results are similar to those of GCN.
By observing the results produced by GCN in the geometry refinement pipeline, we can notice how most changes in geometry are not reflected into direct noticeable changes in the rendered images. Multiple spikes are generated all over the geometry, these spikes can be noticed clearly in the geometry but are not completely noticeable in the rendered image. In fact they are flattened onto the roof surface during the rendering step and are only visible over the geometry

edges. Most spikes are generated to produce homogeneous edges when rendering specific images from certain view-points.

Given the results of all the performed experiments with the GDP approach, we conclude the following: in order to perform shape aware geometrical optimisation, it is not sufficient to optimise a loss computed over the projected image spaces.
Too much information is lost during the projection from 3D space to the image plane. The lost information is crucial for the reconstruction of the geometry and cannot be ignored.
Various elements of depth and occlusion are essential for geometrical optimisation and can only be included when optimising over losses computed directly onto geometrical objects.
It is not surprising to notice how the previously described GDP models perform exceptionally well when trained over an identity loss calculated directly over geometrical objects. The same models deteriorate in performances as soon as image based losses are introduced.
We believe that geometrical deep learning for geometry optimisation has incredible potential and we see room for future research.

5

A Hybrid Model: The Morph-GAN

5.1 Model Description

Taken into account the insightful results of all the previous experiments, we need to take a step back and observe the overall results obtained to this point.

At first we approached our novel view synthesis and image based rendering problem with the introduction of a GAN model as shown in Fig 3.1.
Such models, particularly the Pix2pix one, offered very good results when dealing with color and texture issues lacking although the capability of fixing geometrical inaccuracies.

We then introduced a per vertex geometry optimisation procedure over vertices locations and colors as shown in Fig 4.3. The direct optimisation operation showed incredibly positive results in terms of geometrical refinements to the original geometry but lacked entirely fidelity over the images high-frequency details.

Lastly we addressed a full geometry and texture reconstruction scenario with GDL, realising that realistic geometry optimisation is a non trivial operation when working only with image data.

Taking into account the observations and insights gained from all these previous experiments, we propose a final hybrid model which we call **Morph-GAN** where we combine both the geometrical optimisation and GAN approaches.
The proposed pipeline is shown in the following Fig 5.1.

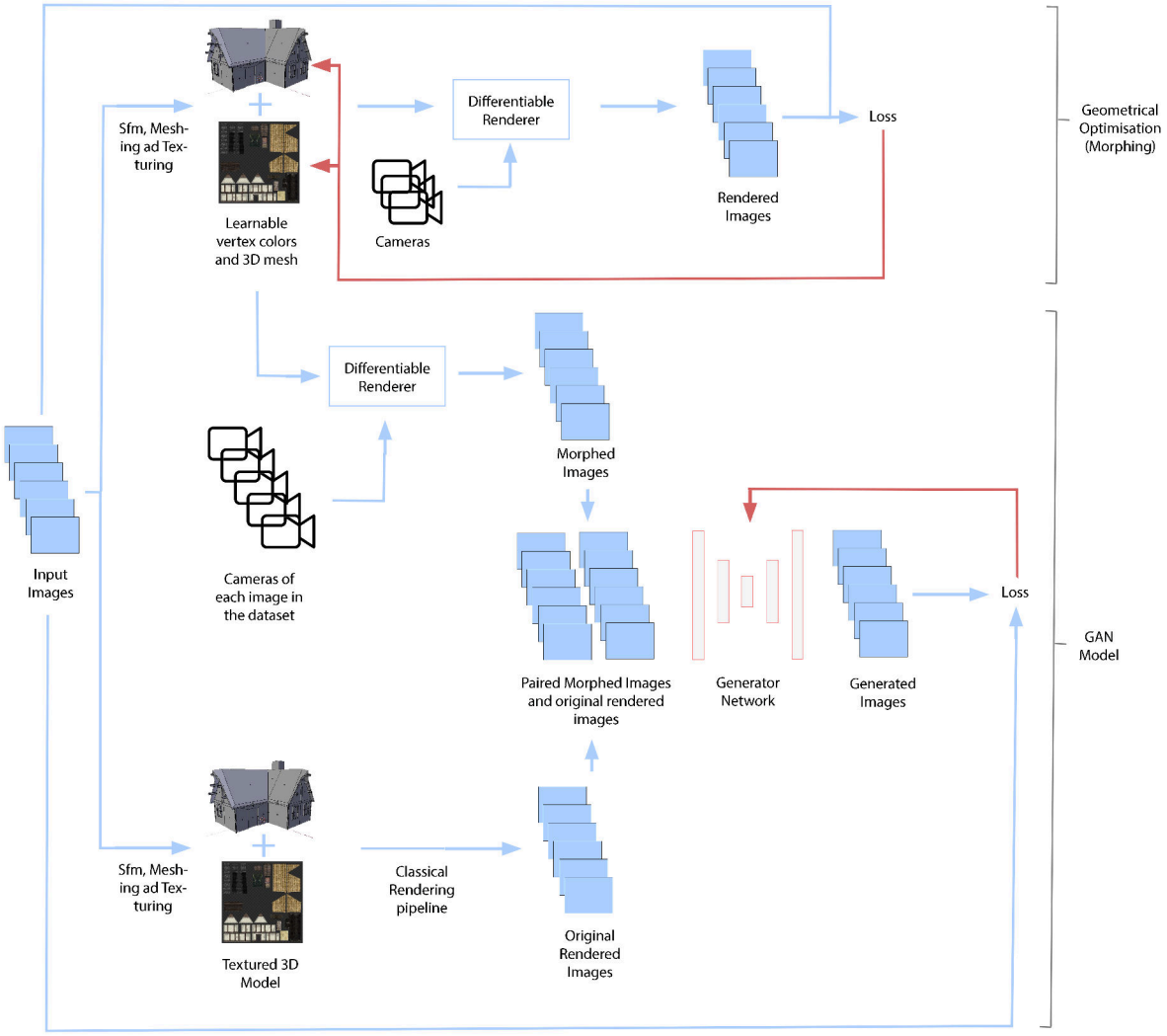


Figure 5.1: *Morph-GAN pipeline.*

Consider a specific dataset of target images I_{target} associated with a set of n cameras C, a 3D mesh M defined by a set of n vertices $V = (v_1, v_2, ..., v_n)$ and m faces $F = (f_1, f_2, ..., f_m)$ with its associated texture T and a differentiable renderer R.

We firstly perform per vertex optimisation or "morphing" over vertices locations and colors as previously described in Section 4.3.
We progressively morph the original mesh M optimising over an L1 loss between the rendered images $r_j = R(V, F, c_j)$ and the target images $i_j \in I_{target}$, updating gradually the vertices positions and colors as shown in equation (4.3).

Once convergence is reached, we stop the morphing procedure. We therefore obtained a new optimised mesh M' defined by a new set of n vertices $V' = (v'_1, v'_2, ..., v'_n)$ and the same set of m faces F.

We proceed to render via the differentiable renderer R each camera view c_j for all cameras in C. By doing so, we produce a new version of the original dataset of morphed images $I_{morphed}$.

$$I_{morphed} = \{m_j | m_j = R(V', F, c_j), \forall c_j \in C\} \tag{5.1}$$

The new dataset is simply a reconstruction of the original dataset over the optimised mesh M'. It is also identically distributed into training and test sets.
Each image $m_j \in I_{morphed}$ is associated to the same camera $c_j \in C$ as $i_j \in I_{target}$.

Given the three sets of images: the set of target images I_{target}, the set of original rendered images I_{render} and the newly generated set of morphed images $I_{morphed}$, we can finally produce a novel dataset of paired images I_{paired} such that:

$$I_{paired} = \{p_j | p_j = r_j \parallel m_j, \forall j \in card(C)\} \tag{5.2}$$

Each image $p_j \in I_{paired}$ is a six channel image obtained by the concatenation of the original rendered image $r_j \in I_{render}$ and the morphed image $m_j \in I_{morphed}$.

Given this new dataset we can finally train a new Pix2pix GAN model which will take as input the set of images I_{paired} and as target the set of images I_{target}.
The GAN model is identical to the one used in Section 3.3.

5.2 Results

The model is trained for 10000 epochs with a learning rate of $\lambda = 0.0002$ and batch size of 3. As shown in the following figures 5.2, the model converges after approximately 1000 epochs.

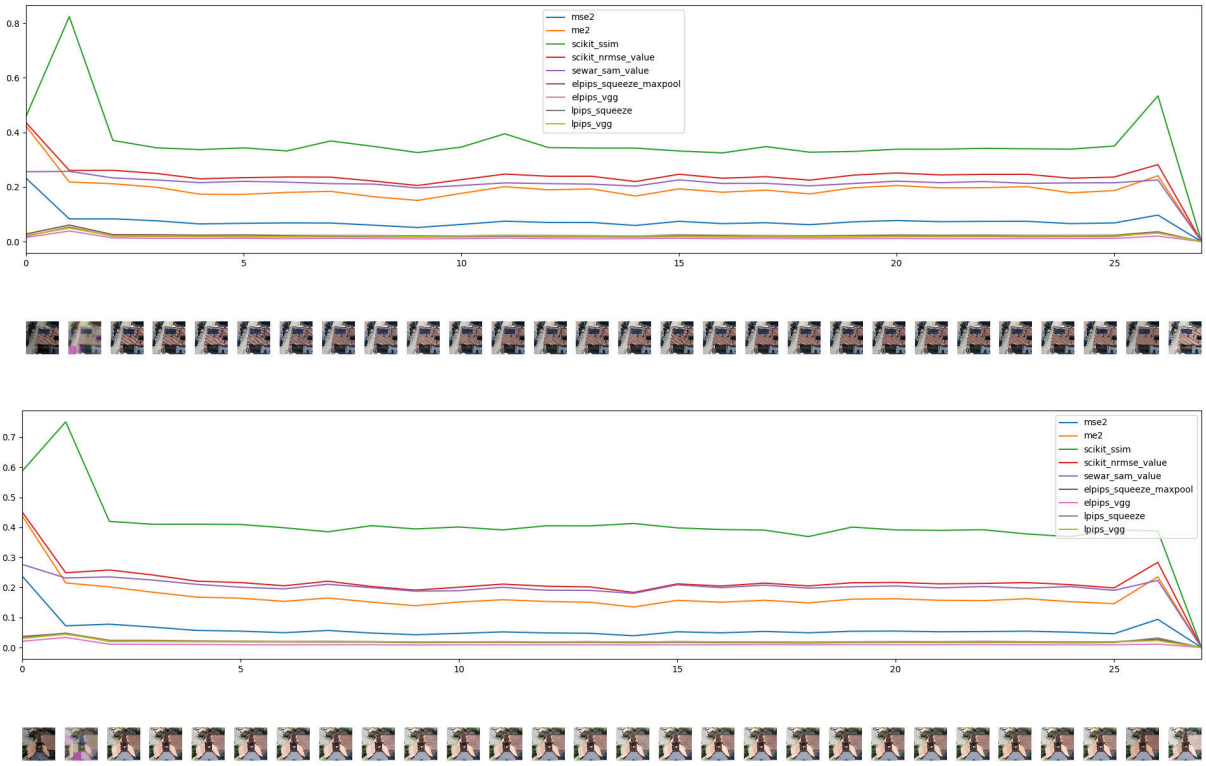


Figure 5.2: *Morph-GAN convergence during training with respect to various metrics. Last two data-points are respectively: input morphed image and target image.*

In this section we compare results of three models:

- The original Pix2pix model developed in Section 3.3 which receives as input the set of original rendered images I_{render}.
- A variation of the Morph-GAN model which receives as input only the set of morphed images $I_{morphed}$. This version should be called partial Morph-GAN.
- The complete Morph-GAN model which receives as input both the set of morphed images $I_{morphed}$ and the set of original rendered images I_{render}.

We can observe the results of all the aforementioned models over a test image of the Sullens dataset in the following figures.

Figure 5.3: *Input original rendered image.*

Figure 5.4: *Input Morphed image.*

Figure 5.5: *Image generated by the Pix2pix model.*

Figure 5.6: *Image generated by the partial Morph-GAN model.*

Figure 5.7: *Image generated by the complete Morph-GAN model.*

Figure 5.8: *Target image.*

We can notice the already very good results of the partial Morph-GAN model shown in Fig 5.6. Most of the original visual artifacts have been fixed. The majority of geometrical issues have been resolved: chimneys are reconstructed, walls are flattened and roof and building edges are straight lines. We can notice tremendous improvements when compared to the input morphed image shown in Fig 5.4.
The GAN model which works on top of the input morphed image menages to fill in successfully most of the texture information. Walls and roofs do not appear as flat unrealistic surfaces anymore but actually show high-frequency details and look much more realistic. These results show how this model gained benefits of both the geometric optimisation and the GAN model.

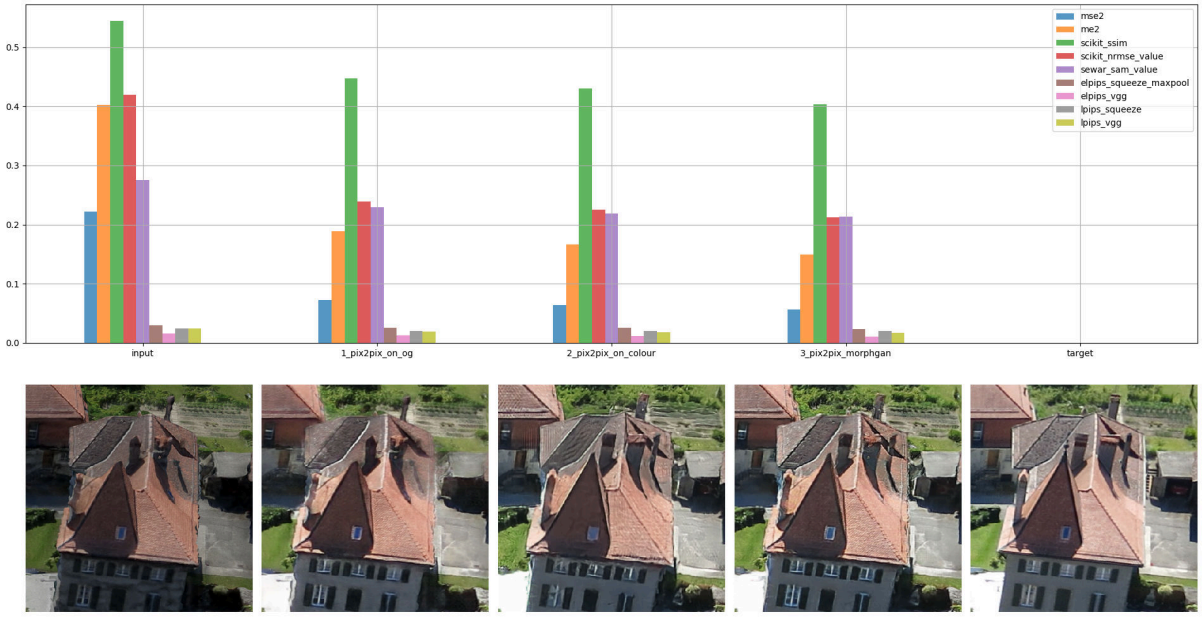
The results of the complete Morph-GAN model shown in Fig 5.7 are even better.
Differently from his incomplete version, the complete Morph-GAN receives as input both the original rendered images and the morphed images. The original rendered input images already provides the model with a great amount of high-frequency information: in fact the original rendered images already have texture information embedded. While the partial Morph-GAN has to learn how to add high-frequency details only over the loss function, the complete Morph-GAN already receives that information as input.
The results of the complete Morph-GAN show an astounding improvement when compared to both the original rendered images and the previous Pix2pix results. All geometrical artifacts are fixed and the high-frequency details are reconstructed.
The model is a perfect blend between the positive aspects of the morphing procedure of Section 4.3 and the advantages of the GAN models of Section 3.3.

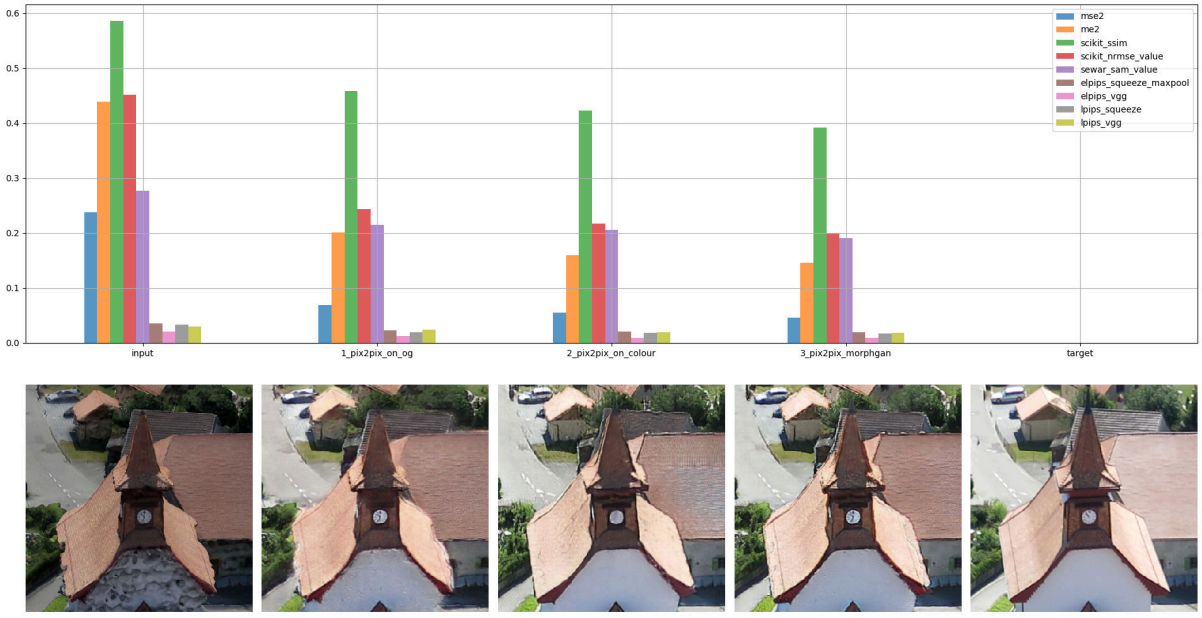
We show more proof of improvement by performing metric analysis over some critical image patches with respect to the following popular metrics:

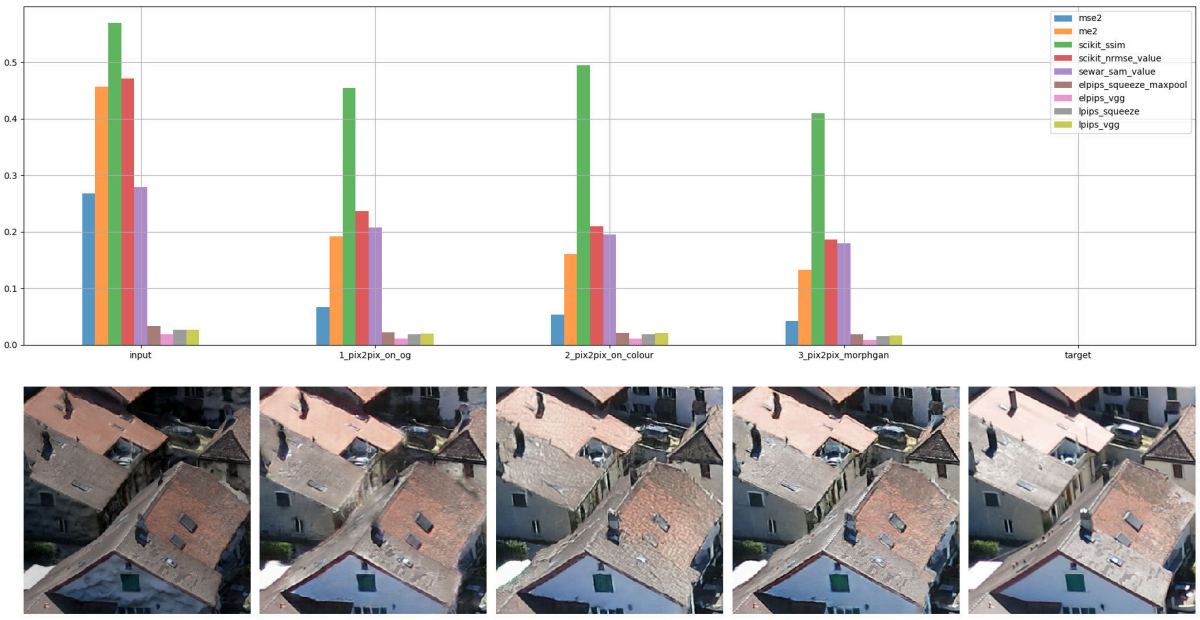
- MSE: Mean Squared Error.
- ME: Mean Error.
- SSIM: Structural Similarity Index (shown as 1-SSIM)
- NRMSE: Normalised Root Mean Squared Error.
- SAM: Spectral Angle Mapper.
- E-LPIPS: Robust Perceptual Image Similarity via Random Transformation Ensembles [22].

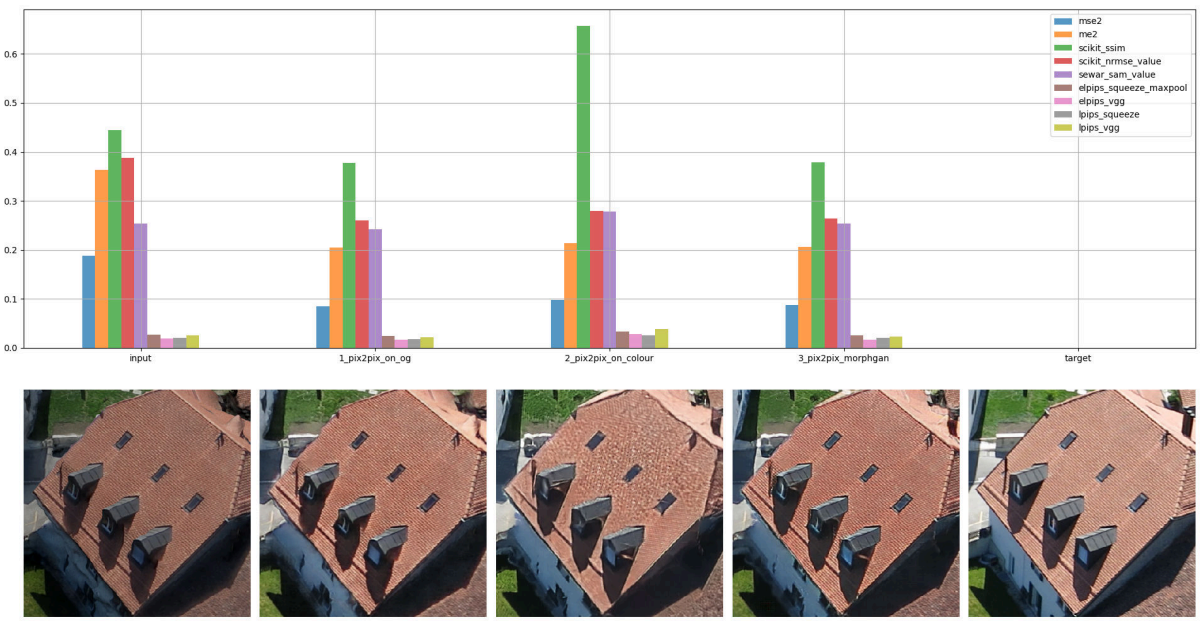
The metric analysis shows a net improvement of the Morph-GAN model over all the other considered models and a drastic improvement compared to the original rendered image.

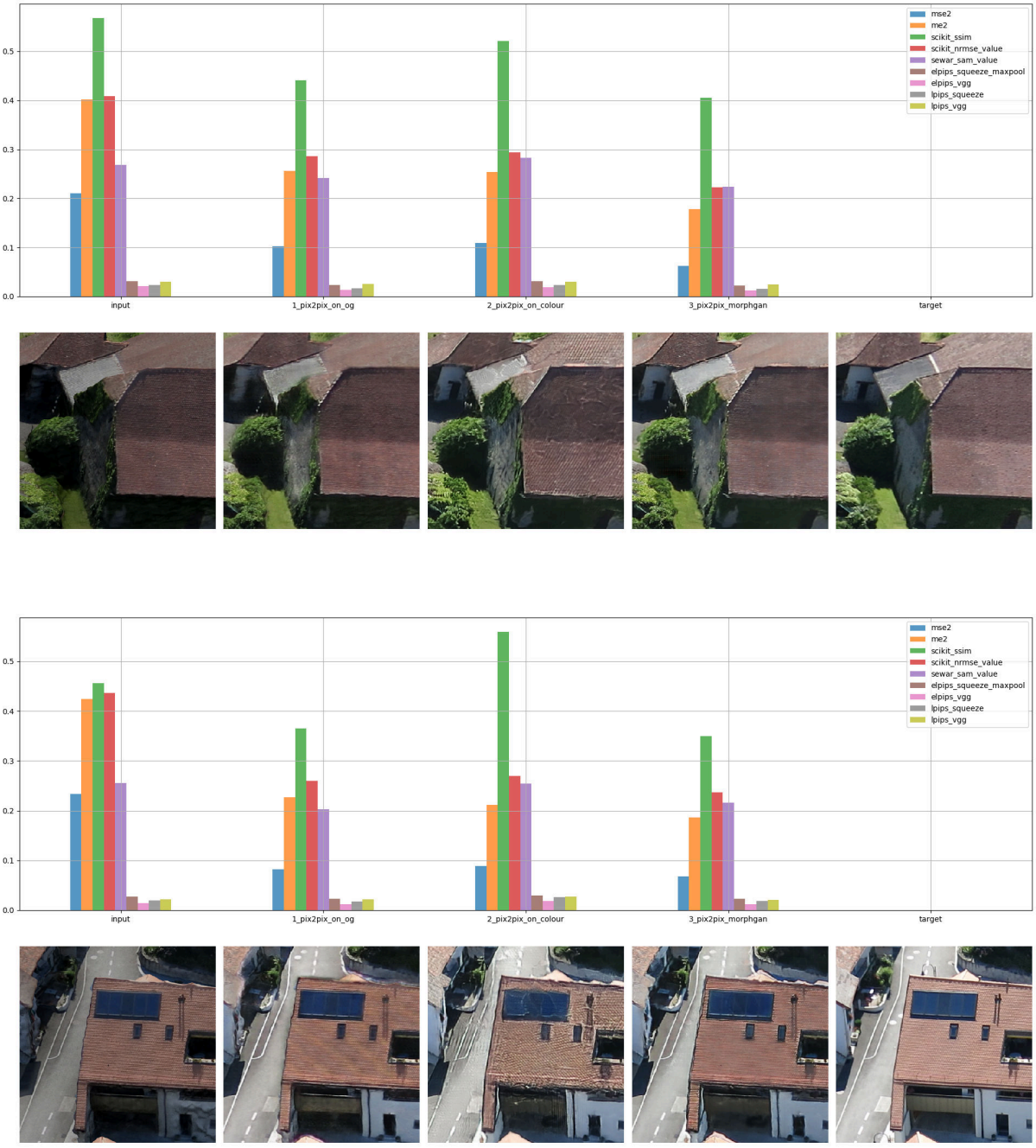
The only noticeable failure cases can be seen whenever the geometrical optimisation step fails. In some specific scenarios, usually at the edges of the geometrical model, the direct geometrical optimisation causes very evident geometrical visual artifacts. Border areas of the geometrical model are in fact less present within the dataset images, compared to central objects. This causes for the optimisation model to optimise their appearance only over a limited set of view-points. Whenever then we render a different point of view, the resulting geometries appear distorted.
The results of the metric analysis can be observed in the following figures. Please be aware that for visualisation purposes some of the losses have been reversed. For all the proposed metrics, the lower the score, the better the performances.


mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.0
0.1
0.2
0.3
0.4
0.5
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target

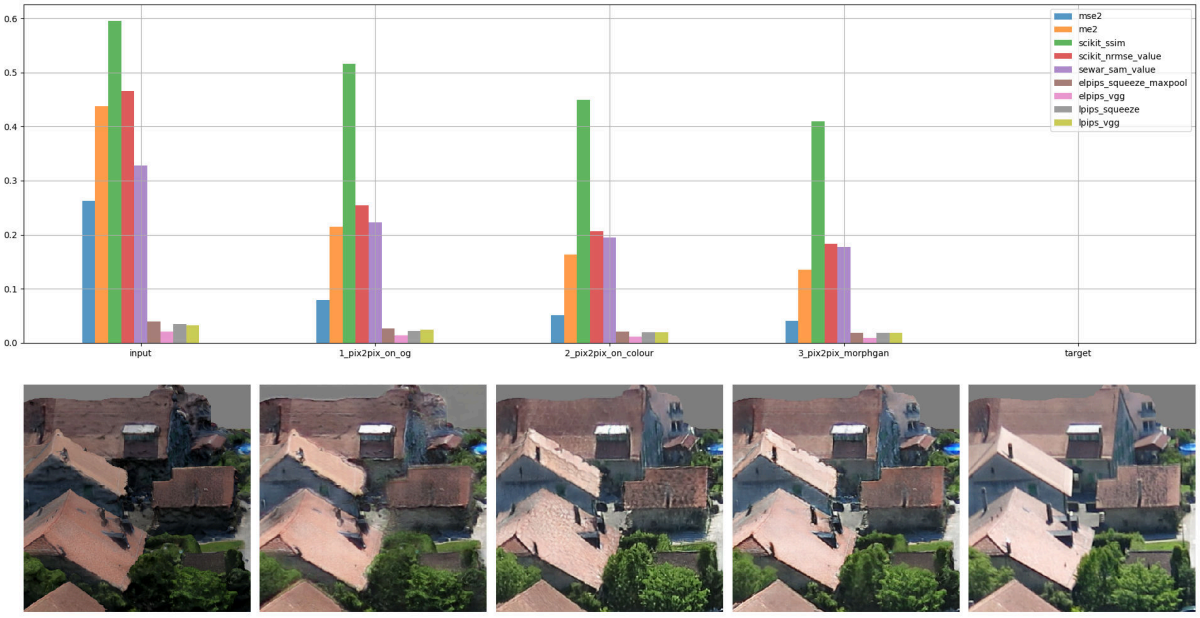

mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.0
0.1
0.2
0.3
0.4
0.5
0.6
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target

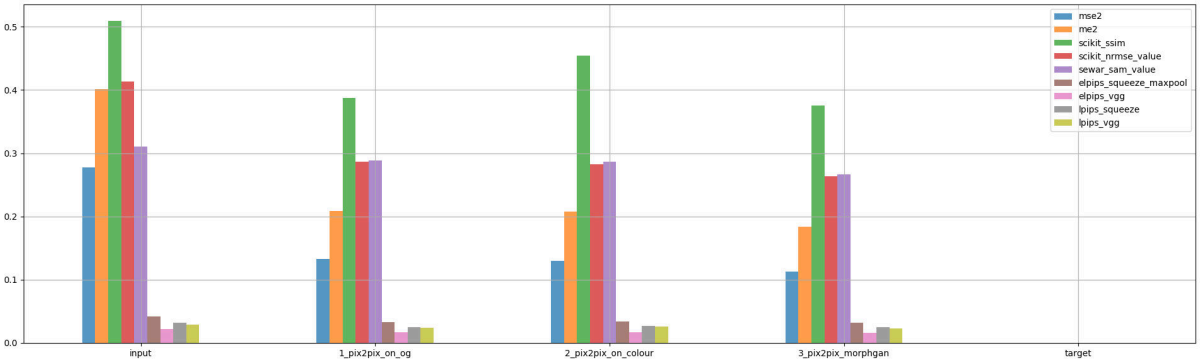

mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.0
0.1
0.2
0.3
0.4
0.5
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target


mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.0
0.1
0.2
0.3
0.4
0.5
0.6
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target


mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.0
0.1
0.2
0.3
0.4
0.5
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target

mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.0
0.1
0.2
0.3
0.4
0.5
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target


mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.6
0.5
0.4
0.3
0.2
0.1
0.0
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target


mse2
me2
scikit_ssim
scikit_nrmse_value
sewar_sam_value
elpips_squeeze_maxpool
elpips_vgg
lpips_squeeze
lpips_vgg
0.5
0.4
0.3
0.2
0.1
0.0
input
1_pix2pix_on_og
2_pix2pix_on_colour
3_pix2pix_morphgan
target

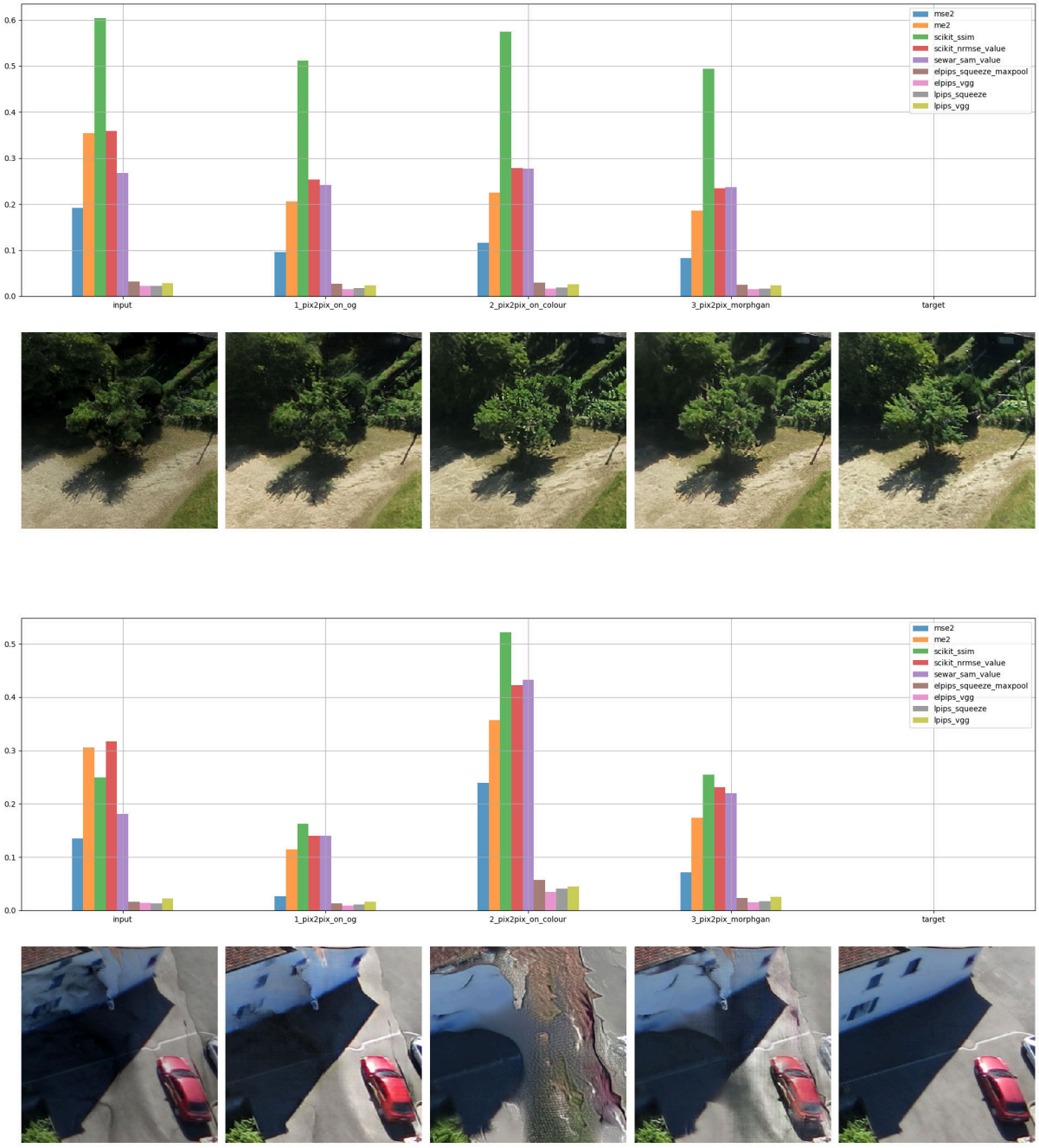


Figure 5.9: *Metric Analysis over image patches exhibiting specific visual artifacts.*

Finally an in depth analysis of know visual artifacts over the presented image patches of complete Morph-GAN results can be seen in the following Table 5.1. Patches that still exhibit visual artifacts correspond to marginal areas of the model as previously explained.

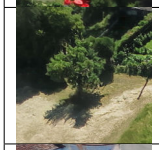
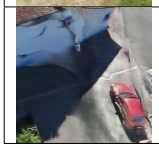
Image Patch	Description	Eroded or Floating Geometries	Texture Inaccuracies	Lighting or Color Issues	Wobbly Lines	Bumpy Walls
	A house with chimneys					
	A church					
	Various rooftops					
	A roof with windows	X				
	A roof with vegetation					
	A roof with solar panels					
	Some far away roofs					
	A small house	X			X	
	A small tree					
	A very close building and a car	X	X	X		

Table 5.1: *n depth visual artifacts analysis of Morph-GAN results over Sullens image patches. Each X marks the presence of a specific visual artifact.*

5.3 Conclusion

Given the very positive final results shown in the previous section, we can conclude our analysis of the proposed Morph-GAN model with the following assessments.
The model performs incredibly well with respect to all geometric visual artifacts, exhibiting the same advantages of the per vertex direct optimisation method. Unfortunately the model suffers also from some of the same issues: the model perform at its worst over marginal areas of the image. Over these areas the model performs worst than Pix2pix and even worst than the original input image.

The model also performs exceptional well with respect to all texture and color artifacts. The model exhibits the same generative power of the Pix2pix model, allowing it to and fix any lighting or color issues producing very detailed images.
Moreover the Morph-GAN doesn't suffer from Pix2pix greatest downfall: generation of blurry areas. As previously discussed, Pix2pix tends to generate blurry areas in places of uncertainty. This phenomenon usually happened around geometrical errors such as holes in roof edges or missing geometries. Once the geometric inaccuracies are fixed via direct optimisation, those areas of uncertainty disappear and Morph-GAN can simply fill them in with texture information.

Ultimately we have to note some peculiar small artifacts introduced by the Morph-GAN. While extracting information from both the original rendered image and the morphed image, the model does sometimes generate double edges or borders on some surfaces. These artifacts are slightly noticeable on the church's roof and on some buildings edges. These artifacts are very faint and do not affect the realism of the generated images.

6

Conclusion and Future work

Within the realisation of this work, we took various paths towards novel view synthesis and image based rendering.

We introduced a novel GAN-based rendering pipeline, analysing what benefits can these models bring to our particular task. We learned how GAN models provide a stable and powerful tool for high-frequency detail enhancement of texture and tuning for color inaccuracies. Similarly we noticed how these models suffer from geometrical inaccuracies.

We then experimented with the introduction of novel state-of-the-art tools such as neural renderers. We proposed an optimisation pipeline over the model initial geometry that aims at minimising geometry related visual artifacts in the rendered images. We show how this approach produces astounding results, greatly improving the quality of the 3D geometry and the resulting rendered images.

We therefore explored the innovative field of geometrical deep learning. With a series of experiments, we show how geometrical convolutions have great potential for shape analysis and understanding. We showed how such convolution represent a useful tool for shape reconstruction when used jointly with a geometrical loss computed directly over 3D objects. When the loss is computed in the image space, the geometrical optimisation fails. We therefore conclude that within our application context, perfect geometrical reconstruction is a non trivial task and could eventually be addressed given ground truth geometrical information.

We finally presented a new model for novel view synthesis and image based rendering called **Morph-GAN** which perfectly embraces the advantages of both the geometrical optimisation and GAN approaches. The proposed model features both the geometrical accuracy of the per vertex direct optimisation approach and the texture optimisation and fidelity offered by the Pix2pix model.
This new model shows incredibly realistic and pleasing visual results in comparison to all previous approaches. Visual results are supported by a metric analysis that shows clear improvements

in comparison with the initial rendering images and both the Pix2pix and direct optimisation results.
Our experiments show that the proposed model is capable of significantly improving image quality and the overall realism of generated images.

For future work, we see room for improvement over various aspects of our newly proposed Morph-GAN model.
An important improvement would be the introduction of a simultaneous training procedure where both the geometrical optimisation and GAN training are performed jointly in an end-to-end fashion.
Moreover, despite the already very good results offered by the Pix2pix part of our Morph-GAN model, we believe that the introduction of more complex and deeper models such as Pix2PixHD [41] could allow for additional high-level fidelity and an overall increase in detail generation. Additional losses over the L1 loss could also improve the quality of the generated images.

Multiple improvements can be made with respect to the geometrical optimisation step as well. During the realisation of this work, due to the nature of the available data, we had to abandon the goal of shape aware geometry optimisation. We believe that ulterior steps towards this direction can be made. The primary issues faced during DGL was the lack of a loss function computed directly geometrical objects.

We believe that a possible solution to this issue could be the introduction of a geometric discriminator as shown in a newly proposed Geometric-GAN framework shown in Fig. 6.1.

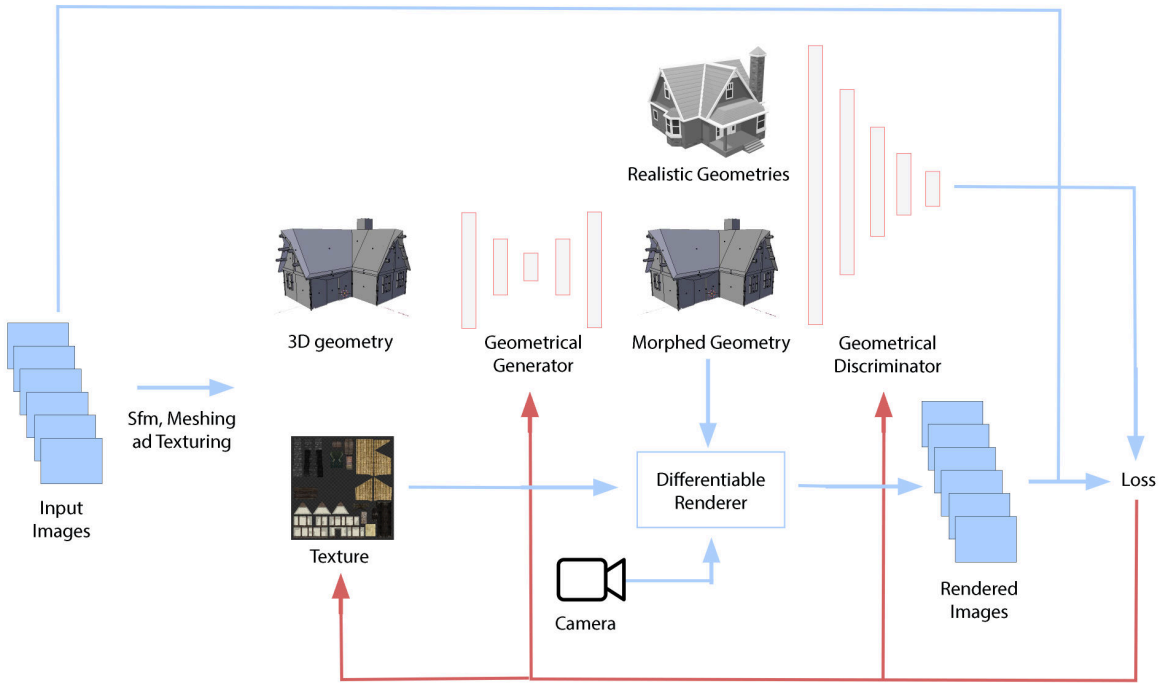


Figure 6.1: *Proposed Geometric-GAN architecture for future work. House image taken from [1].*

Ideally the geometric discriminator would classify areas of the generated geometry and behave exactly as GAN discriminators behave in image-to-image translation applications. In order to do so, during training, the discriminator should receive areas of real geometries as positive examples in order to enforce geometrical righteousness. The positive examples should be a few hand-made geometrical models of real buildings.
We believe that the introduction of a geometric discriminator could provide the model with a geometry based adversarial loss, allowing the model to be optimised over both an adversarial loss computed over geometries and an L1 loss computed over rendered images.
The introduction of the discriminator in the presented pipeline would result in the realisation of a complete **Geometrical GAN** model which could potentially show results as good in geometry generation as regular GAN models show in image generation.
We believe that a Geometrical GAN Model could represent an exciting and innovative tool in the fields of Geometrical Deep Learning, image based rendering a generally computer graphics.

Bibliography

[1] Home mysite, private eye security. https://www.privateeyesecurity.ca/?lightbox=dataItem-jwidsdtm.

[2] How to make uv maps? unity forum. https://forum.unity.com/threads/how-to-make-uv-maps.224634/.

[3] Picture of Duomo's square in Milan, Italy. https://www.nyx-hotels.de/wp-content/uploads/2017/08/Fotolia_53276412_XL.jpg.

[4] James Arvo. Transfer equations in global illumination. In *Global Illumination, SIGGRAPH '93 Course Notes*, 1993.

[5] Dominik Aufderheide, Werner Krybus, and Gerard Edwards. Inertial-aided sequential 3d metric surface reconstruction from monocular image streams. 07 2013.

[6] T. Deane. Next-gen drone footage puts trinity on the 3d map. https://www.tcd.ie/newsevents/articles/.

[7] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29:1944–57, 12 2007.

[8] DNEG. VFX in the Marvel's Avengers Engame movie. https://www.dneg.com/show/avengers-endgame/.

[9] Google Earth. Screenshot at Duomo's square in Milan, Italy. https://earth.google.com/web/@45.46478106,9.19179548,130.96400168a,248.1998491d,35y,74.4404977h,86.46259891t,-0r, Feb 2020. Coordinates: 45°27'53"N, 9°11'28"E.

[10] John Flynn, Michael Broxton, Paul E. Debevec, Matthew DuVall, Graham Fyffe, Ryan S. Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned

gradient descent. *CoRR*, abs/1906.07316, 2019.

[11] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. *CoRR*, abs/1506.06825, 2015.

[12] Hongyang Gao and Shuiwang Ji. Graph u-net, 2019.

[13] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.

[15] J. Weissenberg H. Riemenschneider. Ethz cvl ruemonge dataset. `https://www.research-collection.ethz.ch/handle/20.500.11850/184859`.

[16] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *CoRR*, abs/1809.05910, 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. 01 2018.

[19] Paul Henderson and Vittorio Ferrari. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. *International Journal of Computer Vision*, 2019.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016.

[21] J. Su V. Krylov M. Bourke D. Moloney R. Dahyot J. Byrne, J. Connelly. Trinity college dublin drone survey dataset. `http://www.tara.tcd.ie/handle/2262/81836`.

[22] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. E-LPIPS: robust perceptual image similarity via random transformation ensembles. *CoRR*, abs/1906.03973, 2019.

[23] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*, February 2017. arXiv: 1609.02907.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[25] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[26] Hsueh-Ti Liu, Michael Tao, and Alec Jacobson. Paparazzi: Surface editing by way of multi-view image processing. volume 37, pages 1–11, 12 2018.

[27] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *CoRR*, abs/1703.07511, 2017.

[28] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien P. C. Valentin, Sameh Khamis, Philip L. Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B. Goldman, Cem Keskin, Steven M. Seitz, Shahram Izadi, and Sean Ryan Fanello. Lookingood: Enhancing performance capture with real-time neural re-rendering. *CoRR*, abs/1811.05029, 2018.

[29] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. *CoRR*, abs/1904.04290, 2019.

[30] Warren Moore. Textures and samplers in metal. `https://metalbyexample.com/textures-and-samplers/`, 09 2014.

[31] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics*, 38, 07 2019.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[33] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Björn Ommer. A style-aware content loss for real-time HD style transfer. *CoRR*, abs/1807.10201, 2018.

[34] senseFly Parrot Group. Sullens dataset details: found under small village. `https://www.sensefly.com/education/datasets/?dataset=1420`.

[35] senseFly Parrot Group. Sullens dataset drone information: ebee x drone. `https://www.sensefly.com/drone/ebee-x-fixed-wing-drone/`.

[36] A. Zisserman T. Werner. Trinity iii. `https://www.robots.ox.ac.uk/~vgg/data/mview/`.

[37] TensorFlow. Tensorflow graphics overview. `https://www.tensorflow.org/graphics/overview`.

[38] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *CoRR*, abs/1904.12356, 2019.

[39] Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. IGNOR: image-guided neural object rendering. *CoRR*, abs/1811.10720, 2018.

[40] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Dynamic filters in graph convolutional networks. *CoRR*, abs/1706.05206, 2017.

[41] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[42] Zhizhong Wang, Lei Zhao, Wei Xing, and Dongming Lu. Glstylenet: Higher quality style transfer combining global and local pyramid features. *CoRR*, abs/1811.07260, 2018.

[43] Martin Wicke, Sandro Olibet, and Markus Gross. Conversion of point-sampled models to textured meshes. pages 119–124, 01 2005.

[44] Felix Wu, Tianyi Zhang, Amauri H. Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. *CoRR*, abs/1902.07153, 2019.

[45] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.