universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

De-lighting color textures of 3D models

verfasst von | submitted by

Thomas Christian Birke B.Sc.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 935
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Medieninformatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Helmut Hlavacs

Acknowledgements

I would like to express my deepest gratitude to Univ.-Prof. Dipl.-Ing. Dr. Helmut Hlavacs, my supervisor, whose expertise, guidance, and unwavering support have been invaluable throughout the course of this thesis. Your insightful feedback and thoughtful suggestions have greatly enhanced the quality of my work. I am truly grateful for the time and effort you dedicated to mentoring me.

Abstract

The demand for high-quality 3D models has grown significantly across various domains, including video game development, film production, cultural heritage preservation, and documentation of real-world objects. However, the acquisition of 3D models through methods like laser scanning or photogrammetry often result in illumination effects being baked into the color texture of the 3D model, presenting challenges for dynamic lighting scenarios and adaptable model usage. This work addresses the task of de-lighting 3D models, focusing on the removal of shadows, highlights, and color tints from color textures. Two main approaches were developed and evaluated: an image-based method using a Conditional Generative Adversarial Network (CGAN) architecture, and a pixel-based method employing a simpler linear layered neural network that processes input textures on a per-pixel basis. Experiments were conducted to determine the optimal configuration and input data for each approach. Additionally, the performance of various color spaces was evaluated to identify the most suitable representation of the color data in the color texture of the digitized 3D model, as well as additionally generated supplementary input color textures. The Oklab color space was found to provide the best results, demonstrating the importance of appropriate color space selection in achieving high-quality de-lighting outcomes. For the image-based CGAN approach, utilizing all available supplementary textures, including lightmap, light color map, ambient occlusion map, and UV texture area map, yielded the highest quality output. The pixel-based approach performed best using just the light color map as additional input. Comparing the two approaches, the pixel-based method demonstrated several advantages, including requiring less training data, flexibly handling different input texture resolutions, and producing visually superior results with more preserved detail. While the image-based model is less flexible regarding the processed texture resolution, its processing of the complete texture in one step allow it to be able to analyze the global context of the texture to include, for example, the overall color tone of the texture. These findings highlight the trade-offs between global context understanding and local detail preservation in de-lighting tasks.

A light estimation optimization process was also developed to automatically refine a manually provided initial guess of the light environment, improving the recreation of light in Blender to generate supplementary input textures. Various optimization methods were evaluated in an attempt to find a suitable solution for the complex task of light environment estimation. This process is particularly challenging when the color texture of the digitized 3D model serves as the sole source of lighting information. The complexity arises from the need to infer multiple lighting parameters, such as light direction, intensity, and color, from a single image that contains the combined effects of these factors. Additionally, real-world lighting conditions often involve multiple light sources and complex interactions with the environment, further complicating the estimation process.

Abstract

While the proposed methods successfully remove illumination effects and shadows from color textures in a controlled test environment, some limitations remain when applying the techniques to real-world 3D scans. These include inaccuracies in captured 3D geometry and challenges in estimating complex real-world lighting environments. Future research approaches could focus on optimizing geometry capture as well as exploring advanced methods for estimating and simulating complex lighting scenarios. The goal would be to make the proposed approaches more robust and extend their applicability to any real-world scans.

Kurzfassung

Die Nachfrage nach hochwertigen 3D-Modellen ist in verschiedenen Bereichen, einschließlich der Videospielentwicklung, Filmproduktion, Erhaltung des kulturellen Erbes und Dokumentation von realen Objekten, erheblich gestiegen. Die Erfassung von 3D-Modellen durch Methoden wie Laserscanning oder Photogrammetrie führt jedoch oft dazu, dass Beleuchtungseffekte in die Farbtextur des 3D-Modells eingebettet werden, was Herausforderungen für dynamische Beleuchtungsszenarien und eine anpassbare Modellnutzung darstellt. Diese Arbeit befasst sich mit der Aufgabe der Entfernung von Lichteffekten in Farbtexturen von 3D-Modellen, wobei der Fokus auf der Entfernung von Schatten, Glanzlichtern und Verfärbung durch Licht aus Farbtexturen liegt. Zwei Hauptansätze wurden entwickelt und evaluiert: eine bildbasierte Methode unter Verwendung einer Conditional Generative Adversarial Network (CGAN) Architektur und eine pixelbasierte Methode, die ein einfacheres lineares neuronales Netzwerk einsetzt, das Texturen pixelweise verarbeitet. Experimente wurden durchgeführt, um die optimale Konfiguration und Eingabedaten für jeden Ansatz zu ermitteln. Zusätzlich wurde die Nutzung verschiedener Farbräume evaluiert, um die am besten geeignete Repräsentation der Farbdaten in der Farbtextur des digitalisierten 3D-Modells sowie zusätzlich generierte ergänzende Eingabefarbtexturen zu identifizieren. Es wurde festgestellt, dass der Oklab-Farbraum die besten Ergebnisse liefert, was die Bedeutung der Auswahl eines geeigneten Farbraums für hochwertige Lichtbereinigungsergebnisse unterstreicht. Für den bildbasierten CGAN-Ansatz erzielte die Nutzung aller verfügbaren ergänzenden Texturen, einschließlich lightmap, light color map, ambient occlusion map und UV texture area map, die besten Ergebnisse. Der pixelbasierte Ansatz zeigte die besten Ergebnisse unter ausschließlicher Verwendung der light color map als zusätzliche Eingabe. Im Vergleich der beiden Ansätze wies die pixelbasierte Methode mehrere Vorteile auf, darunter ein geringerer Bedarf an Trainingsdaten, eine flexible Handhabung verschiedener Eingabetexturauflösungen und visuell bessere Ergebnisse mit mehr erhaltenen Details. Während das bildbasierte Modell hinsichtlich der verarbeiteten Texturauflösung weniger flexibel ist, ermöglicht seine Verarbeitung der gesamten Textur in einem Schritt die Analyse des globalen Kontexts der Textur, um beispielsweise das allgemeine Farbbild der Textur einzubeziehen. Diese Erkenntnisse verdeutlichen die Kompromisse zwischen der ganzheitlichen Bildverarbeitung mit globalem Kontextverständnis und lokaler Detailerhaltung durch Verarbeitung auf Pixelebene bei der Aufgabe der Entfernung von Lichteffekten.

Zudem wurde ein Prozess zur Schätzung der Lichtverhältnisse entwickelt, um eine manuell bereitgestellte Anfangsschätzung der Lichtumgebung automatisch zu verfeinern und die Nachbildung von Licht in Blender zur Generierung ergänzender Eingabetexturen zu verbessern. Verschiedene Optimierungsmethoden wurden evaluiert, um eine geeignete Lösung für die komplexe Aufgabe der Lichtschätzung zu finden. Dieser Prozess ist

besonders herausfordernd, wenn die Farbtextur des digitalisierten 3D-Modells als einzige Quelle für Beleuchtungsinformationen dient. Die Komplexität ergibt sich aus der Notwendigkeit, mehrere Beleuchtungsparameter wie Lichtrichtung, Intensität und Farbe aus einem einzelnen Bild abzuleiten, das die kombinierten Effekte dieser Faktoren enthält. Zusätzlich beinhalten reale Beleuchtungsbedingungen oft mehrere Lichtquellen und komplexe Wechselwirkungen mit der Umgebung, was den Schätzungsprozess weiter erschwert.

Obwohl die vorgeschlagenen Methoden Beleuchtungseffekte und Schatten in einer kontrollierten Testumgebung erfolgreich aus Farbtexturen entfernen, bleiben einige Einschränkungen bei der Anwendung der Techniken auf reale 3D-Scans bestehen. Dazu gehören Ungenauigkeiten in der erfassten 3D-Geometrie und Herausforderungen bei der Schätzung komplexer realer Beleuchtungsumgebungen. Zukünftige Forschungsansätze könnten sich auf die Optimierung der Geometrieerfassung sowie die Erforschung fortschrittlicher Methoden zur Schätzung und Simulation komplexer Beleuchtungsszenarien konzentrieren. Ziel dabei wäre es, die vorgeschlagenen Ansätze robuster zu gestalten und ihre Anwendbarkeit auf beliebige reale Scans zu erweitern.

Contents

List of Figures

Listings

1. Introduction

With technologies like Virtual Reality and Augmented Reality gaining popularity and the introduction of modern graphics standards like WebGPU [50], the use of 3D visualization and its associated applications continues to increase across various domains. This trend spans from the preservation of cultural heritage, where digital replicas serve preservation and educational purposes [4], to the creation of 3D assets for entertainment in the production pipelines of games or movies. While physical museums offer invaluable experiences, the digitization of real-world objects provides enhanced accessibility and documentation. Smaller artifacts can be transported for research or exhibition purposes, while larger objects such as statues or buildings can be digitized and made available to a broader audience, independent of location. This approach facilitates a more inclusive and diverse representation of cultural heritage, enabling online exhibitions and virtual museums that allow global exploration and learning about history and culture. Additionally, the digitization of archaeological sites serves to document and preserve their current state, which proves beneficial for future research and restoration projects. In contrast, the entertainment industry utilizes 3D models for creating virtual worlds in games and films. The development of a 3D video game environment, for instance, requires a variety of 3D models, including characters and props, which must be integrated into the game world. The creation of these 3D models can be a time-consuming manual process, which may be simplified by utilizing 3D models obtained through the digitization of real-world objects. This is especially useful for assets used to enrich small details in the game world, such as tables or rocks, which can be tedious to model manually.

The acquisition of 3D models for these diverse applications involves multiple approaches, including methodologies like laser scanning and photogrammetry, which enable the generation of virtual replicas of real-world objects [5, 51]. Different digitization techniques offer various strengths and weaknesses. For example, laser scanning can provide highly accurate 3D geometry but may lack or provide limited color information. Alternatively, photogrammetry employs high-resolution color images from cameras, with some approaches even exploring the use of smartphone cameras [15, 35], to create 3D models with both geometry data and color textures, even though the resulting 3D model may have less accurate 3D geometry data compared to laser scanning methods. A more detailed overview of various digitization techniques is provided in Chapter 3.

Despite the capability of these techniques to achieve accurate replicas of real-world objects, they share a common challenge. During the process of generating a virtual 3D model through digitization techniques, the real-world object cannot be easily separated from its surrounding environment. Depending on the object to be scanned, this can mean that a bigger object needs to be scanned in the place it is located, for example a building in a city, a statue in a park or an archeological site. Consequently, the object cannot

be easily relocated to a controlled environment for the digitization process, but must be scanned in its original location, including surrounding objects not intended to be part of the 3D model but which nonetheless affect the visual appearance of the object being scanned. This can result in incomplete scans due to occlusion by other objects. Another factor influencing the digitization result is the light environment in the scene. Depending on the light sources present, the object may be illuminated in ways that affect the color information of the resulting color texture of the digitized 3D model. This can be caused by colored light adding a color tint to the object, shadows from occluding parts or objects, or highlights caused by intense light reflecting off the object's surfaces. Smaller objects can also be scanned in a controlled environment, like a studio, where other potentially occluding objects are removed and light conditions can be controlled with artificial light instead of sunlight, leading to a more consistent digitization result. Even though there

Figure 1.1.: Presentation of digitized 3D models with only their color texture shown, which includes highlights and shadows from the light environment during the digitization process. The beauty of palmyra [43] (left) and a monkeys figure 3D scan [25] (right).

are techniques to reduce the influence of the surrounding environment on the digitized 3D model, it is difficult to completely eliminate these effects during digitization. Figure 1.1 shows two examples of 3D models, where the surface color texture includes highlights and shadows from the light environment during the digitization process. The 3D models in the images have been rendered without any light shading, meaning that only the raw color from the 3D model's color texture is applied to the 3D geometry of the model. As evident, the images still display static shadows and highlights that are baked into the color texture. While including lighting in the color texture may be acceptable for visualizing objects within a static scene, this approach is inadequate for more complex scenes. For instance, scenes with multiple different 3D models digitized in varying light environments, and thus containing different illumination effects in their color textures, do not fit visually when put together into a single scene. Moreover, more complex scenes with moving models and dynamic lighting scenarios, such as those in video game environments or movie scenes,

require a more flexible approach to handle the lighting of 3D models. In the context of cultural heritage preservation and documentation, a 3D model with an illumination-free color texture can be desirable to provide clear details of the object's surface color and structure for study and analysis, as well as dynamic visualization, for example, in a digital museum exhibition with scenes in dynamic lighting conditions similar to the visualization in video games or movies. In such cases, where 3D models are used in scenes with dynamic lighting conditions, the color textures of the rendered 3D models are expected to be free of illumination effects such as shadows and highlights. The illumination effects are then dynamically applied to the 3D models during the rendering process, depending on the light sources in the scene. This allows for moving models or light sources in the scene to interact with each other, creating a more realistic and immersive experience for the viewer. Also since all 3D models in the scene are rendered with the same lighting conditions, they fit together visually and create a consistent scene.

Given the described difficulties of digitizing real-world objects to create 3D models with color textures that are free from illumination effects, there is a need for illumination-free color textures for the usage of 3D models in various scenarios. This thesis focuses on addressing the challenge of de-lighting color textures in 3D models, with a particular emphasis on removing shadows, highlights, and color tints. The main goal is to develop a processing method that can be applied to a given 3D model to remove the illumination effects from the color texture, resulting in a de-lighted 3D model. The de-lighted 3D model can then be used in the described scenarios, where the 3D model is rendered with dynamic lighting conditions.

The content of this thesis starts with a review of already existing work in the field of de-lighting and related topics, such as illumination detection and removal in color images as well as re-lighting attempts of single 2D portrait images of a human face. These works provide insights into the challenges of analyzing light and shadow information solely from color image data and the further processing to manipulate the color image data to be either illumination-free or to contain different illumination effects with a different simulated light environment. An overview of the Conditional Generative Adversarial Network (CGAN) architecture for image-to-image translation tasks is also provided, as it shows to be a good fit for the given task of processing color textures of 3D models. When analyzing the illumination effects in color textures, knowledge of the light environment configuration, such as the position of light sources and their color and intensity, can be beneficial. Some existing approaches to estimate the illumination configuration in a scene, as well as an existing solution for removing illumination effects in color textures, are presented. To provide insights into different approaches, existing work that does not process an already existing color texture, but aims at generating a new color texture with different input methods, is also reviewed.

The majority of this work focuses on the following aspects:

- What is the most effective method for providing meaningful input data to the de-lighting process to specify the illumination effects in the color texture that need to be removed?
- What neural network architectures and training methods yield the most effective de-lighting results?
- How can the lighting conditions that existed during the digitization of a real-world object be effectively estimated and reconstructed, in order to provide the required input data for the de-lighting process?

The aspect of good input data is addressed in multiple ways. First, the method of representing scene environment lighting information in the color texture is explored. This includes the exploration and evaluation of different additional supplementary textures to the already given color texture of the 3D model. These supplementary input textures are used to describe how the light interacts with the surface of the 3D model to give insights into which parts of the color texture are influenced by light. Therefore, textures such as lightmaps to describe the light intensity of the light hitting the object's surface, as well as light color maps to describe the light's color, are utilized. Also additional data such as ambient occlusion maps to describe which parts of the object are occluded from others and therefore receive less light are evaluated. Besides the type of input data, such as light intensity and color, the method of specifying the color data for further processing has been evaluated. The thesis compares the sRGB, HSV, HSL, CIELAB, and Oklab color spaces, assessing their effectiveness in representing color information for the de-lighting task. The outcome of the conducted experiments shows that characteristics like the separation of color hue and color lightness, as well as perceptual uniformity of color spaces, result in better de-lighting results. Consequently, the two color spaces CIELAB and Oklab performed the best, with the Oklab color space showing the best results, leading to its use in all further de-lighting processing and experiments.

The thesis presents two main approaches to tackle the de-lighting challenge: An image-based method utilizing the CGAN architecture and a pixel-based method employing a simpler neural network, consisting of linear layers, that processes input textures on a per-pixel basis. These approaches are explored and evaluated through extensive experimentation, focusing on the effectiveness of various color spaces, input configurations, and optimization techniques in achieving good de-lighting results. The image-based approach employs the CGAN architecture, which has shown promising results in image-to-image translation tasks [38]. This image-based de-lighter method treats the input color texture and its supplementary textures as a single entity, processing the entire image simultaneously. The CGAN architecture allows for additional control over the generated output by conditioning the generator on supplementary information, such as lightmaps and ambient occlusion textures. This approach has the advantage of capturing and learning global context and relationships within the input textures, which can be beneficial for a better understanding of the color patterns in the color texture. On the other hand, the

pixel-based de-lighter approach processes input image data on a per-pixel basis. This method iterates over each pixel in the input textures and feeds the data for every pixel into the de-lighting model individually. The pixel-based method offers several advantages, including flexibility in handling different input texture resolutions and efficiency in training data utilization. Since the model operates on individual pixels, a single low-resolution texture can provide a large number of training samples, reducing the computational effort required for training dataset generation. The evaluation of the proposed de-lighter methods reveals their strengths and limitations. The image-based approach demonstrates the ability to capture global context, potentially providing additional insights into color patterns. However, it faces challenges in handling different input texture resolutions and may result in some loss of fine details due to the downsampling and upsampling processes in its encoder-decoder architecture. The pixel-based approach, while more flexible and capable of preserving fine details, itself does not consider the context of surrounding pixels in its processing.

The thesis also addresses the step of estimating and reconstructing the light environment present during the object's digitization. This process is fundamental for generating the supplementary light data input textures required by the de-lighter models. The research presents an optimization method to enhance the accuracy of manual light environment estimation, focusing on estimating properties of light sources such as position, direction, intensity, and color. The optimization process is structured in multiple stages to first estimate the light source's properties such as position or light direction and intensity, followed by the estimation of the light color. Various optimization methods and their results are evaluated, leading to the conclusion that the Simulated Annealing optimization method provided best results, even though light environment estimation remains a challenging task, especially in more complex lighting scenarios.

In conclusion, this thesis presents a comprehensive exploration of de-lighting techniques for 3D model color textures, addressing a challenge in the field of 3D digitization and visualization. By developing and comparing image-based and pixel-based approaches, evaluating various color spaces, and addressing the challenge of light environment estimation, the research provides insights and tools for improving the quality and flexibility of digitized 3D models. While limitations remain, particularly in dealing with more complex real-world scans and complex lighting scenarios, the work provides an alternative de-lighting method for further processing and improving digitized 3D models.

2. Related Work

This chapter provides an overview of existing work in the field of de-lighting color textures of 3D models and related areas. It explores similar tasks, such as the re-lighting of human portrait images, which share the goal of accurately estimating and manipulating light conditions in given images. It also introduces the architecture and training methodology of CGANs, which have demonstrated promising results in image-to-image translation tasks. The chapter also presents proposals for estimating scene illumination, including the identification of existing light sources and their characteristics. Furthermore, the chapter discusses methods for illumination removal in images, including shadow detection and elimination, as well as the processing of highlights caused by bright light sources. Also, a brief workflow overview of an existing closed-source solution for illumination removal in color textures of 3D models is provided. Lastly, alternative methods for generating color textures for 3D models are explored, focusing on the use of text and image prompts to guide the texture generation process.

2.1. Re-Lighting Portrait Images

A common task found in literature is the processing of human face portraits and adapting them to fit into different scenes with varying light environments. Numerous methodologies have been proposed to solve this task, often leveraging neural networks of various architectures. The networks are tasked to process a single image of a human face as input and adjusted lighting composition to re-light the face to match or completely remove the lighting from the face image [76, 74, 34, 70]. While the specific use case of de-/re-lighting only portrait images of human faces appears to be sufficiently scoped for neural networks to learn the process of adjusting the lighting in single images, extending the input data with additional information about the 3D geometry of the human face or full body in the image can improve the details of the results, as shown in the approach from Chaonan Ji et al. [39]. This work splits the processing of the image into two main parts: de-lighting and re-lighting. In the de-lighting part, a modified HRNet [64] is used to remove the existing lighting effects from the given image of a full human body. The re-lighting part utilizes PIFuHD [57] to estimate the 3D geometry of the full human body in the input image. With the 3D geometry of the human body in the image given, the re-lighting step uses ray tracing via the Cycles rendering engine in Blender to re-apply light effects with the new light environment configuration onto the human body image. Since the 3D geometry estimation has limited accuracy additional further steps like smoothing the 3D geometry are applied. For training data generation, the researchers used a library of digital 3D human avatars [67] rendered in various indoor and outdoor lighting environments [30].

While this approach aims to improve de-lighting and re-lighting results, it faces challenges in removing specular highlights due to dataset limitations in material surface details [39]. Furthermore, inaccuracies in 3D geometry estimation can lead to artifacts in the re-lighting results, despite additional processing techniques like geometry smoothing.

2.2. CGAN Architecture

The concept of employing a network that takes the color texture, including illumination highlights and shadows, as input, and is tasked with removing the illumination to return a clear surface color texture aligns well with the Generative Aversarial Network (GAN) architecture as described in [38]. In the training phase of a GAN, the architecture is split into two models, which compete against each other while improving in the learning process. The generator model takes an input image and produces a synthetic image, while the discriminator model is tasked to distinguish real images from the fake synthetic images from the generator. This competition of the generator model trying to generate synthetic images, that are not recognized by the discriminator model and the discriminator trying to detect these fakes images is used as a mechanism for iterative learning through the training process. However, when only feeding the color data of the virtual 3D model into

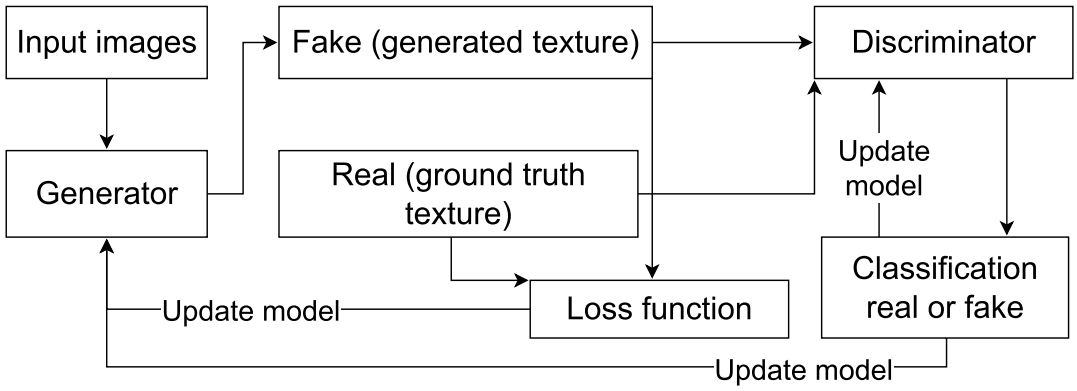

Figure 2.1.: Overview of training a CGAN network.

the generator model, the task of effectively removing lighting influence across diverse 3D models and their textures is challenging. Therefore, the detailed 3D geometry from the obtained 3D model offers the possibility to supplement additional data for the de-lighting process. By adapting the GAN to a CGAN architecture [38], the input of the single color texture for the generator model can be extended with supplementary details on the lighting configuration that affected the illumination of the object. This enables using the generation of additional supplementary input textures, such as lightmaps, with the given geometry of the 3D model, describing additional lighting information for the generator model. Lightmaps can be utilized to provide insight into various aspects of the light environment's impact on different regions of the geometry, including the intensity and color of light that interacts with the 3D object at specific locations. While this approach

appears promising to enable more detailed de-lighting results, the generation of additional supplementary input textures, such as a lightmap, requires the estimation of additional data, since the detailed specification of light sources that existed in the light environment during the process of laser scanning or taking pictures for use with photogrammetry methods is unknown.

2.3. Estimation of Scene Illumination

The determination of scene lighting is a crucial task, especially in tasks like illumination removal or the integration of virtual objects in the scene. The renderings of those inserted virtual elements must be adapted to the scene's lighting conditions to achieve a more realistic visual result. Various methodologies have been proposed to estimate the illumination conditions of a scene based on one or more images.

One approach from Lalonde et al. [42] involves estimating illumination conditions, particularly the position and visibility of the sun, using only a single outdoor image. Specific features, that are extracted from the input image, such as the sky and cast shadows on the ground or vertical surfaces, serve as cues for estimating the sun's position and visibility. While this method provides a rough approximation of the sun's light direction in a given image, its applicability to virtual 3D models and their textures may prove challenging, resulting in imprecise results. In contrast, Hold-Geoffroy et al. [33] use a Convolutional Neural Network (CNN) to map pixel data from spherical 360° panoramas, that include sun and sky, to illumination parameters like sun position or exposure. While this approach appears to deliver acceptable results for inserting virtual objects, with comparable rendered lighting for highlights and shadows, into existing 2D scenes, the approximate calculation of the sun's position for outdoor environments may lack the precision required for accurate shadow and highlight estimation on 3D models. In addition, this research primarily focuses on outdoor settings with natural sunlight and sky conditions, and does not address indoor environments featuring artificial illumination, which are frequently encountered in real-world object digitization scenarios.

An alternative proposal by Wang et al. [69] involves estimating the illumination environment of urban scenes using Neural Radiance Fields (NeRFs) [48]. This approach utilizes machine learning technologies to generate an environment map or High Dynamic Range (HDR) sky map from a set of posed RGB images. The generated HDR sky map can then serve as input for describing the lighting environment during the de-lighting process. The paper also suggests the use of a 360° camera to capture images of the surroundings to compute an environment map [69].

Other setups during the generation of the virtual 3D model could include light probes, that help the estimation of the light configuration in the scene, similar to the use in the work of Debevec [18] or Mandl et al. [44]. A light probe is a known real-world object, where surface properties like surface color are known, that is put into the captured scene. For instance, a simple white sphere could be useful to improve the estimation of the light color in the scene.

In cases where the scanner has control over and knowledge of the light source casting illumination effects on the scanned object, research has been conducted to eliminate undesired highlights or shadows resulting from these controlled light sources. For instance, when utilizing a camera equipped with a flashlight to illuminate an object for photogrammetric image capture, the flashlight's position can be determined relative to the camera's position. Consequently, light estimations and illumination effects from the flashlight can be calculated using the acquired 3D geometry data, enabling the removal of these illumination effects from the color data [19]. For example, given the position of the flashlight and the 3D geometry data of the object, Phong shading is employed to estimate the region where specular highlights are expected to be present on the 3D model. This information can then be used for further processing to remove these specular highlights in the color texture.

2.4. Illumination Removal Approaches

In various use cases, the removal of illumination effects from images is a frequently encountered task. This process may involve the removal of shadows or highlights in individual 2D color images or surface color textures of 3D models. In both scenarios, the acquisition of color information can be influenced by scene illumination, which may be undesirable and consequently require removal.

Various research efforts concentrate on the task of removing shadows in single 2D color images [49, 68, 32, 7]. The primary objectives typically include the identification of shadowed regions and the subsequent removal of color alterations caused by shadows within these areas. This is often achieved through blending or interpolation techniques that utilize information from adjacent non-shadowed regions.

The research conducted by Saritha Murali and V. K. Govindan [49] presents a method designed for the detection and removal of shadows from a single RGB image. It addresses challenges inherent in computer vision applications such as segmentation, object detection, and object counting. The method employs the LAB color space, wherein the L channel represents color lightness, the A channel indicates the red-green spectrum, and the B channel denotes the blue-yellow spectrum of a given color value. The shadow detection first converts the RGB color values of the image into LAB color values, followed by calculating the mean values of the L, A, and B channels for the image pixels. Subsequently, each pixel is classified as either shadow or non-shadow based on the conditions specified in Equation 2.1.

$$\text{Classification} = \begin{cases} \text{Shadow}, & \text{if } \begin{cases} \overline{A}_{\text{image}} + \overline{B}_{\text{image}} \leq 256 \\ \text{and } L_{\text{pixel}} \leq \overline{L}_{\text{image}} - \frac{\sigma_{L_{\text{image}}}}{3} \\ \text{or} \\ \overline{A}_{\text{image}} + \overline{B}_{\text{image}} > 256 \\ \text{and } L_{\text{pixel}} < \text{Threshold and } B_{\text{pixel}} < \text{Threshold} \end{cases} \\ \text{Non-shadow}, & \text{otherwise} \end{cases} \tag{2.1}$$

To reduce errors in shadow detection, the method incorporates strategies to prevent misclassification of pixels. For instance, pixels are only classified as shadows if a certain number of neighboring pixels in a given region are also identified as shadows. Shadow removal is then accomplished by multiplying the RGB color values of pixels in shadow regions by a constant. The constant is calculated as a ratio of the average RGB color values in shadow regions to the average RGB color values in non-shadow regions nearby. To ensure a smooth transition between shadowed and non-shadowed areas, the method applies additional correction to shadow edges using median filters. The proposed method offers low computational complexity and operates on a single input image, but it may incorrectly identify dark objects as shadows. Additionally, depending on the input image, shadows may not be entirely removed, potentially leaving a less pronounced shadow effect in the processed image.

In an alternate proposition presented in the work of Jifeng Wang, Xiang Li and Jian Yang [68], a STacked Conditional Generative Adversarial Network (ST-CGAN) is employed for shadow removal in an input image. The ST-CGAN architecture leverages two CGANs, where the first CGAN processes an image with shadows as input and is tasked with generating a shadow mask that marks regions that are affected by shadows in the image. Subsequently, the second CGAN takes the image with shadows and the generated shadow mask as input and removes the shadows in the detected regions. Judging the proposed methods by their provided experimental results, the ST-CGAN architecture suggests a better shadow removal performance compared to the previously described approach.

The work by Pintus et al. [54] provides a comprehensive survey on Multi-Light Image Collections (MLICs) for surface visualization and analysis. MLICs are sets of color images of an object captured from the same fixed viewpoint, but with varying lighting conditions. The utilization of MLICs encompasses the processing of illumination effects on the surface of the captured subject. This includes potential approaches for analyzing lighting influences and possible methods for removing these effects in color images [20, 53, 22, 75]. Regarding the reduction or removal of illumination effects such as shadows in a composite output image, many approaches utilize multiple images from a MLIC and interpolate the pixel color values between these images. Given that the image collection consists of photographs taken under varying lighting conditions, images with illumination from different angles can be utilized to interpolate between shadowed areas in one image and illuminated areas in another. This process results in the creation of an interpolated texture with less pronounced shadows.

Similar approaches that use multiple images of an object under varying lighting conditions to improve the quality of the resulting color texture use the name Multi-view Images [12, 13] instead of MLICs. Chen et al. [12] propose a method to remove highlights caused by light sources during the 3D scanning process by utilizing multiple images of the 3D model under various lighting conditions. Identification of highlights within the color data of a single image is accomplished through the analysis of RGB pixel color values for contiguous regions on the projected 3D geometry. The median of the color values within these regions is computed. A region is recognized as a highlight if its average color value diverges by more than 20 % from the calculated median. To remove a highlight identified

in a specific image, the method uses pixel color values from corresponding locations in other images with different lighting configurations where the given region does not contain a highlight. These values are then blended into the initially processed image to remove the highlight effect. The highlight detection method demonstrates good results on the provided sample 3D model, which features a relatively uniform color tone in its texture. However, it is anticipated that this approach may encounter challenges when applied to more complex color textures with greater variation in color tone.

Besides the difference in lightness of the color in the color texture of the digitized 3D model and the surface color of its real-world counterpart, like specular highlights on a glossy surface or shadows, the captured color can also be influenced by the color of the light source. To avoid color hue changes in the captured color caused by light sources in the scene, common digitization techniques use neutral white light to illuminate the object that has to be scanned. However, not all controlled light sources emit perfectly neutral white light (e.g., [19, 9]), or there may be uncontrolled and unwanted light sources in the scene that cause colored light. Especially in outdoor scenes, the color of the various external light sources cannot be controlled [46] and may cause unwanted color hue changes in the captured color of the scanned object. Some approaches expect the light sources in the scene to be scanned to have a relatively neutral white color tone and therefore do not see a requirement to add a color hue shift correction step in the processing pipeline. Other techniques include the measurement of the light source's color hue tint and subsequent calibration for color hue shift correction [19, 9]. Also, neural networks have been used to remove color influences from light sources [70]. Many approaches combine the color hue shift processing with the removal of shadows and highlights.

2.5. Existing Solution: Agisoft Texture De-Lighter

Agisoft [1] offers a suite of software solutions, including the stand-alone tool "Agisoft Texture De-Lighter". This specialized tool is designed to address the task of shadow and ambient occlusion removal from the color texture of a 3D model. Since this tool is not open source, and a detailed description of the applied algorithms could not be found, only a rough overview of the usage of the tool is given based on a usage tutorial [2]. It does not use machine learning techniques, but rather a classic algorithm, that "is optimized for 8-bit JPEG compressed textures" [2]. The fundamental workflow involves importing a 3D model, including geometry and a color texture with baked shadows. A visual editor allows users to mark areas of varying light intensity within an image. Users can indicate regions with reduced illumination, resulting in shadows, as well as areas of increased light exposure, producing highlights in the color texture. These user-defined markings serve as reference points for the subsequent de-lighting process. An example, as depicted in Figure 2.2, showcases the de-lighting workflow applied to a 3D model. Shadowed areas are marked in blue while lit areas are colored in yellow. The resulting de-lighted texture already shows a remarkable reduction of shadows in the color texture, although less noticeable illumination artifacts, such as a slight shadow beneath the left elbow of the figure, are still remaining. The processed result can be seen here [3].

Figure 2.2.: Example worklow of the de-lighting of a 3D model [2]

While the current process returns noticeable improvements, especially only with rather imprecise markings from the user, marking all shadowed and lit areas of bigger scans could result in increasing user effort with limited precision.

2.6. Color Texture Generation

Apart from work that focuses on processing already existing color textures of 3D models, there is also research on generating color textures for 3D models that have no color texture [72, 55, 11, 71, 10]. Many approaches commonly employ diffusion models to generate 2D color images, adapting their inputs and outputs to utilize the model's image generation capabilities for creating color textures for 3D models. To specify the theme and characteristics of the 3D model and influence the color texture generation process, the most widely used user inputs are text and image prompts.

The most common input type for users to specify the overall theme of the color texture is a text prompt, which is given as input into text-to-image generation models. These models produce 2D color images of a colored version of the textureless geometry data, which are then projected onto the 3D model's geometry to generate a color texture with UV mapping, as used in common 3D renderers. To provide a better understanding of texture generation methods, the proposal by Zeng et al. [72] has been selected as an example. It introduces Paint3D, a coarse-to-fine generative framework for creating color textures based on text input for 3D models with existing geometry data but no color texture. The texture is generated in multiple steps, divided into two stages: coarse texture generation and texture refinement. In the coarse texture generation stage, depth maps from various camera views of the 3D model are rendered and provided as an additional input condition to a depth-aware 2D diffusion model to generate color images. These images are then projected onto the 3D model's geometry and merged to create a single

coarse color texture. Since the coarse texture is generated based on depth maps from different camera views, there may be missing parts due to occlusions or lighting shadows caused by the 2D image diffusion model. The refinement stage aims to address these issues by applying image diffusion models to the coarse texture. As UV mapped color textures for 3D models typically contain various split fragments with seams, mainstream diffusion models may not produce satisfactory texture refinement without adaptation. To address this, the diffusion model is provided with additional information about the UV mapping of the 3D model. Specifically, the corresponding 3D point coordinate of each pixel in the color texture, in the form of a position map, is given as an additional input to the diffusion model. This input enables the diffusion model to inpaint missing parts of the coarse texture due to occlusions. Additionally, the functionality of a diffusion model is employed to enhance the color texture by adding more details. The diffusion models for the refinement stage process the color texture without information about a light environment and are trained to produce lighting-less textures.

An alternative user input approach involves using a color image as additional input to guide the image generation diffusion model in creating a color texture for the 3D model. While text input provides a simpler method, image input allows for a more precise definition of the desired color scheme for specific parts of the object. Since the image already contains a similar object to the 3D model, the process of generating a color texture with matching colors onto the 3D model can be more precise. In the context of digitized real-world objects where only 3D geometry data is available, a picture of the same real-world object could be used to guide the color texture generation process, resulting in a texture that resembles the surface color of the actual object. The aforementioned proposal by Zeng et al. [72] initially uses a text prompt as user input to describe the theme and characteristics of the 3D model for color texture generation. However, it also includes an adaptation that allows for image input to guide the texture generation process by replacing the text prompt encoder with an image prompt encoder.

Generating color textures for 3D models from scratch via text prompts can be beneficial when creating 3D models where the desired object surface color is not pre-defined and an approximation is acceptable. However, in the context of de-lighting color textures of digitized real-world objects, the surface color is already defined by the real-world object. Consequently, completely generating a new color texture may be less desirable, as the initial color texture of the scanned object should be considered and only improved through further processing of the existing color texture. For texture generation approaches using image input, an image of the scanned real-world object could be provided to obtain a color texture that resembles the surface color of the actual object. This approach could be useful in cases where digitization was performed using laser scanners that do not capture the object's surface color, resulting in a lack of color information. In contrast, for digitized objects that already include color information from the digitization process, generating a color texture from scratch may yield less accurate results compared to enhancing the existing color texture.

3. Fundamentals: Digitization Techniques

This chapter provides a non-exhaustive overview of the main categories of digitization techniques that are utilized to create digital 3D models from their real-world counterparts. These methods enable the capture and representation of real-world objects, resulting in 3D geometry data as well as a surface color texture, which can then be edited in further processing steps or used for visualization in programs such as 3D renderers. Since this work focuses on processing 3D models that have been obtained through digitization methods, some of the common digitization techniques are described in this chapter to provide a better background and understanding of the environment that provides the data that is being processed. This helps to explain why further processing of the obtained 3D models can be beneficial in addressing certain issues. One such issue is the lack of purity in color texture, which often includes unwanted light effects resulting from the digitization process. Understanding these challenges provides context for the subsequent processing steps and improvements discussed in this work.

3.1. Contact 3D Scanners

Contact 3D scanners, also known as tactile or touch-probe scanners, operate by physically touching the surface of an object to collect data points. These scanners use a probe that moves across the surface of the object, recording its position in three-dimensional space. The probe is mounted on for example an articulated arm creating a coordinate measuring machine (CMM). Contact scanners offer a high accuracy, which is why they are often used in manufacturing for applications such as quality control [63, 45]. The contact scanning process is not affected by lighting conditions, making it suitable for materials that may be challenging for optical methods, such as highly reflective or transparent surfaces. However, these scanners have limitations. The need for physical contact makes the process time-consuming, especially for complex geometries. The contact scanning device may encounter difficulties in accessing specific regions when attempting to capture complex geometries. This limitation arises when certain portions of the object are obscured by other parts, resulting in an inability to acquire a complete representation of the entire geometric structure. Physical contact between the scanner and the object introduces a potential risk of damage to the scanned item [45]. This concern is particularly important in applications such as the digitization of, for example, historical artifacts for cultural heritage preservation, where safeguarding fragile objects is of high importance. The necessity to avoid any deterioration or harm to these fragile items presents a challenge to

be considered in the scanning process. Furthermore, most of the contact scanners lack the capability to capture color data, thereby limiting their ability to generate color textures of scanned objects. Given these characteristics, contact 3D scanners find widespread application in manufacturing environments, particularly in scenarios that demand high-precision measurements. Such use cases include quality control processes and reverse engineering of mechanical components. The aforementioned limitations result in contact 3D scanners being less frequently used in the digitization of real-world objects for the purpose of creating virtual 3D models with color textures.

3.2. Non-Contact Active Scanners

In contrast to contact scanners, non-contact scanners do not require physical contact with the object to be digitized. As active scanners, they probe the object by actively emitting lasers or light patterns and detecting the reflection of these emissions. Three prominent types of non-contact active scanners are Time-of-Flight (ToF) scanners, laser triangulation scanners, and structured light scanners.

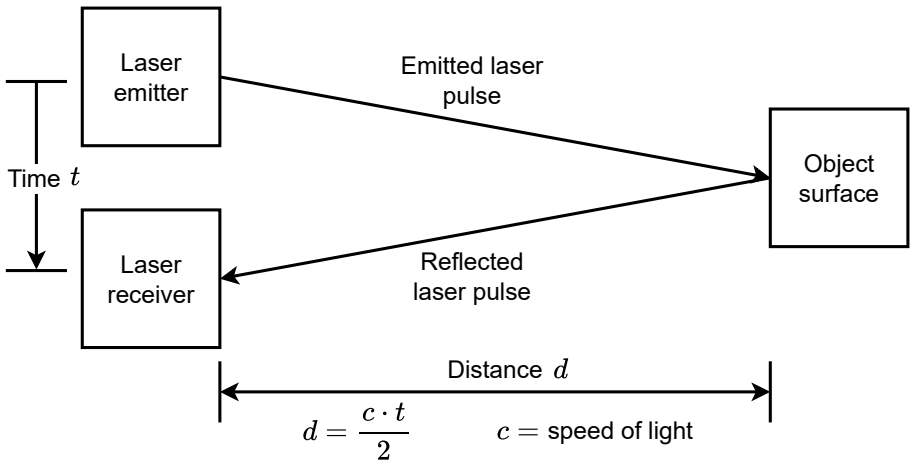

Figure 3.1.: Functionality of a ToF scanner.

As shown in Figure 3.1, a ToF scanner emits a laser pulse and measures the time it takes for the pulse to reflect off the object and return to the scanner. The distance is calculated using the known speed of light and the measured elapsed time. With a typical range of 5 to 300 meters [28], these scanners are suitable for scanning large objects or environments, such as buildings or landscapes. However, their accuracy is dependent on the precision of time measurement, and since the speed of light is very fast, their resolution is often lower compared to other scanning methods [73]. In addition to acquiring geometric data of scanned objects, modern ToF scanners often incorporate color capture capabilities through integrated color cameras [28].

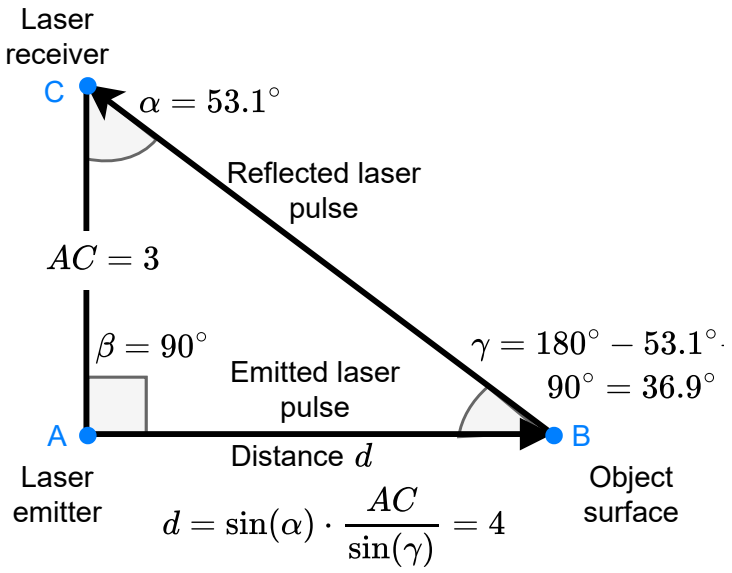

Figure 3.2.: Functionality of a triangulation laser scanner.

Triangulation laser scanners consist of a laser emitter and a laser receiver, which are positioned at a known fixed position and angle. The laser emitter projects a laser light onto the object's surface, and the laser receiver captures the reflection of the laser light. The system then calculates the distance from the laser emitter to the object's surface using trigonometric triangulation [28] as shown in an example in Figure 3.2. Triangulation laser scanners offer high accuracy and resolution for short-range scanning, typically within 5 meters [28]. This makes them particularly suitable for capturing small to medium-sized objects with complex geometries. However, the range of triangulation laser scanners is shorter compared to other types of laser scanners, such as ToF scanners. Many triangulation laser scanners can also capture the object's surface color [28].

Structured light scanners project a known pattern of light onto an object and analyze the deformation of the pattern to determine the 3D shape [27, 65, 73]. These scanners project a series of linear patterns, grids, or more complex light patterns onto an object, as shown in an example in Figure 3.3. Cameras capture the way these patterns deform when striking the object's surface. The system then calculates the 3D coordinates of points on the object's surface based on the deformation of the known pattern. Structured light scanners offer high speed, capturing the entire field of view, and high resolution, capable of capturing fine details and textures [65, 73]. They are versatile and suitable for a wide range of object sizes.

Overall, these non-contact scanning systems may encounter challenges when attempting to capture objects that are made of transparent materials, such as glass. The optical properties of these materials, including light transmission, refraction, and reflection, can

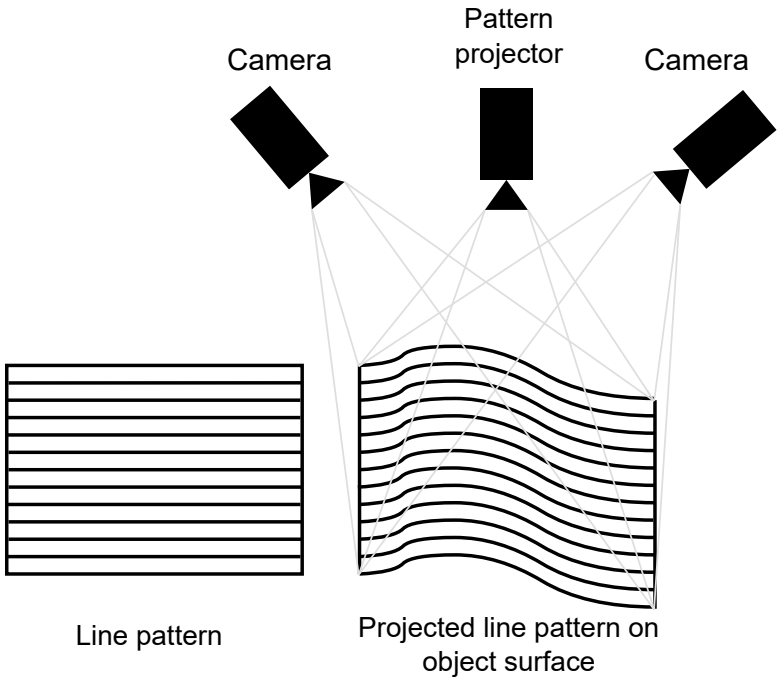

Figure 3.3.: Example setup of a structured line scanner using a line pattern.

interfere with the scanner's distance measurement mechanisms. Consequently, the scanner emissions may lead to inaccurate measurements of the distance to the object's surface, leading to distorted 3D models of the scanned object [41, 26]. As possible solutions, the object can be coated with a temporary spray or powder to alter the transparent surface to a non-transparent one [26]. This allows the laser to correctly bounce off the object's surface and therefore the distance to the object's surface can be measured accurately.

Scanners utilize a large quantity of measured surface points to generate a point cloud representation of the object. This point cloud contains the 3D coordinates of the scanned points on the object's surface. Further processing can be applied to create a 3D mesh representation from this data. The color information can be integrated directly into the point cloud or the 3D mesh, depending on the color capture method provided by the specific scanner. The result is a 3D model of the scanned object, which can be used for 3D visualization in common rendering software or further processing.

3.3. Non-Contact Passive Scanners

Non-contact passive scanners do not emit any kind of radiation themselves, but instead rely on detecting reflected ambient radiation, typically visible light. Two prominent types of non-contact passive scanner methods are stereo vision systems and photogrammetry.

Stereo vision systems typically use two cameras to capture 3D information, mimicking human binocular vision. These imaging systems utilize dual cameras to simultaneously capture images from two distinct perspectives, with a predetermined relative positioning between the cameras. By identifying corresponding points in the different images, the system can calculate depth through triangulation [29]. Stereo vision systems can provide 3D information in real-time, which is useful for applications like robotics and autonomous vehicles [21]. Given that these systems already capture color images for their 3D geometry data reconstruction, they naturally also provide color information that can be projected onto the resulting 3D mesh as color texture. Since stereo vision systems rely on determining corresponding points in the captured 2D images to create the 3D geometry data, they can struggle with featureless or repetitive textures, as well as with reflective surfaces [16].

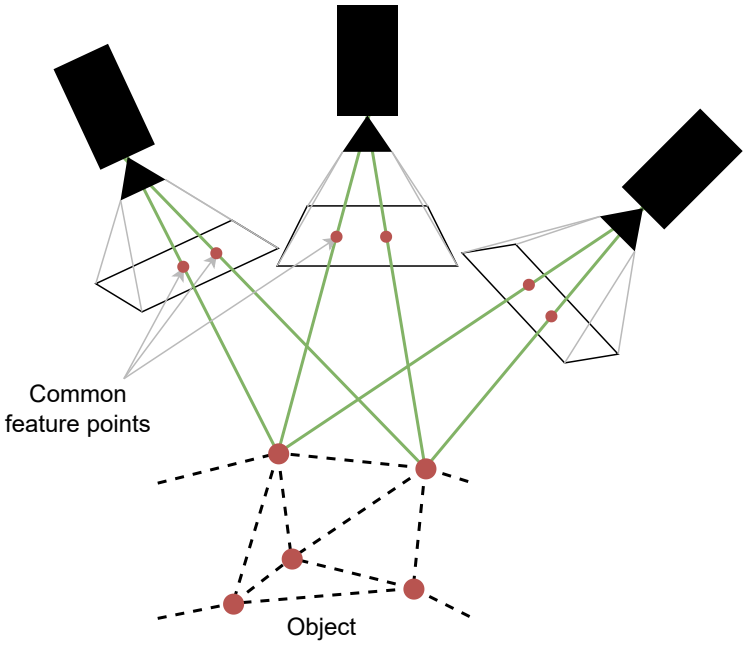

Figure 3.4.: Scene overview of a photogrammetry scan.

Photogrammetry is a technique that uses multiple photographs to create 3D models of objects or environments. Given a set of images of the object from different angles and positions, as shown in Figure 3.4, photogrammetry software can analyze these to recognize common features that are present among the various images to reconstruct the 3D geometry. The photogrammetry process typically involves several steps. First,

multiple overlapping photographs of the object are taken from different angles. Then, the software identifies distinct features in each image and matches them across multiple images. Given the positions and orientations of the camera during image capture, the software can employ triangulation to calculate the positions of matched feature points. The resulting set of 3D points forms a point cloud, which can be converted into a 3D mesh. Subsequently, the original images are projected onto this mesh to create a textured 3D model [47]. Photogrammetry offers several advantages. It is cost-effective, requiring only a camera and software, making it accessible for various applications [47]. It's suitable for objects and scenes of various sizes, from small artifacts to entire landscapes, and automatically captures high-quality color and texture information due to the fact that it already captures a high amount of high-quality color images to generate the 3D geometry data at first hand. Being non-invasive, no physical contact with the subject is required. However, photogrammetry also has limitations. It struggles with featureless, reflective, or transparent surfaces and requires consistent lighting conditions for best results [47, 17]. Processing large datasets can be time-consuming and requires significant computational resources. The accuracy of the 3D reconstruction depends on the quality and coverage of the input images.

3.4. Challenges in 3D Digitization

While 3D digitization techniques have advanced significantly, several challenges remain that affect the quality and applicability of the resulting 3D models. These challenges include dealing with moving objects, surface properties, scale and resolution issues.

One of the challenges in 3D scanning, regardless of the specific technique, is dealing with moving objects. Since the scanning process typically requires multiple measurements from different angles over time, any movement of the subject can lead to inaccuracies or distortions in the final 3D model. This affects different techniques in various ways. In laser scanning, movement during scanning can cause blurring or distortion in the point cloud data. For photogrammetry, motion between photographs can lead to misalignment of features and errors in 3D reconstruction. In structured light scanning, movement can cause misalignment between projected patterns and captured images, resulting in incorrect depth calculations. To mitigate these issues, several strategies can be considered. Using faster scanning techniques can minimize the impact of movement. For smaller objects, securing them in place during scanning is often effective. When dealing with larger scenes, such as scanning buildings, it's beneficial to minimize movement in the environment, for example, by scanning when there's less foot traffic.

The optical properties of surfaces can also significantly affect the quality of 3D scans, particularly for non-contact methods. Reflective surfaces can cause issues with laser reflection and create artifacts in the scan data. Transparent or translucent materials pose challenges as light can pass through or scatter within the material, leading to inaccurate measurements. Very dark or very light surfaces may absorb or over-reflect light, affecting the scanner's ability to detect surface details. Featureless surfaces can be problematic for photogrammetry and other techniques that rely on identifying distinct features. To

address these surface-related challenges, various mitigation techniques can be employed. Applying temporary coatings or sprays to alter surface properties can be effective for some surfaces.

In cases where digitization methodologies incorporate the capture of an object's additional surface information, the most common property is the capture of the raw color data acquired by a color camera. This data is subsequently processed to generate a color texture. As previously noted, digitization techniques may encounter difficulties when scanning real-world objects with complex surface properties, such as transparency. In addition to challenges in scanning precision caused by varying surface properties, the digitization of diverse surface characteristics, including diffuse and specular reflectance and surface roughness, is often incomplete. These surface properties could be beneficial when rendering the obtained 3D model in a 3D renderer to more accurately describe the object's surface interaction with simulated light. Potential approaches to recreate surface properties such as reflectance and roughness involve capturing color images of the object under varying lighting conditions. This can be achieved, for example, by using the same light source in different positions and angles while maintaining a consistent camera position. Analyzing the surface under different light environments allows for the estimation of surface properties [54, 66].

In some cases, combining multiple scanning techniques can help overcome the weaknesses of individual methods, improving the overall quality of the resulting 3D model [37, 58, 36, 9, 46]. Capturing both large-scale geometry and fine surface details can be challenging, often requiring a balance between coverage and resolution. For large-scale scanning, maintaining accuracy over large distances and dealing with occlusions and environmental factors are key challenges. Solutions include using long-range scanners, combining multiple scans, and the use of aerial photogrammetry for very large areas. When it comes to fine detail capture, the challenges lie in capturing microscopic surface details while balancing detail with data volume. High-performance computing is often necessary for processing large datasets, particularly in photogrammetry. GPU acceleration is increasingly used to speed up computationally intensive tasks. The research of Merchán et al. [46] specifically addresses the challenges associated with scanning real-world objects in outdoor environments, where variable lighting conditions can significantly impact results. For instance, when scanning large outdoor scenes, inconsistent cloud coverage may cast shadows on certain areas while leaving others in direct sunlight, resulting in non-uniform color textures from a single-time laser scan. To resolve these issues, they suggest utilizing additional color images of various sections of the scanned scene, captured at different time intervals. In the case of shadows caused by clouds, this methodology entails acquiring multiple color images of the scene at different intervals. This process ensures that all areas of the scenery are captured without cloud shadows at some point during the data collection. Subsequently, integrating the newly acquired color data with the previously generated 3D model, which was created using a laser scanner, can reduce inconsistencies in the color texture that were caused by the lighting conditions present during the initial digitization process.

One inherent challenge associated with all described 3D scanning techniques is the accurate capture of color data for the creation of a color texture for the 3D model. Typically, 3D scanners capture color data using color cameras that record the light reflected from the surface of the scanned object. This process can be affected by undesirable light effects from both the scanning setup and external sources. For instance, the use of lightboxes to enhance the lighting environment can result in highlights on the scanned object, depending on the surface material. In the case of complex objects, some parts of the model may cast shadows onto other parts. External light sources, such as sunlight in outdoor scanning environments or artificial light sources, can also introduce unwanted light effects. To address these issues, specialized scanning environments with reduced external light sources and controlled, well-placed light sources can be utilized. For outdoor environments, scanning during specific weather conditions, preferably on overcast days with uniform light distribution and reduced direct sunlight, can help minimize clear shadows. However, these methods are not always feasible, and the resulting 3D models may still contain unwanted light effects that require removal in subsequent processing steps, which will be addressed in the following chapters. The subsequent chapters will explore methods to further process the 3D model and its color texture obtained through scanning, with the aim of removing these unwanted light effects.

4. Methodology

Machine learning techniques have been selected to address the challenge of de-lighting an input color texture containing illumination effects such as shadows. Specifically two different approaches of processing the image data have been explored: an image-based method and a pixel-based method. The following sections will provide a detailed description of each approach, including the required data, architectures employed, and an analysis of their respective strengths and weaknesses.

4.1. Used Image Data

Regardless of whether an image- or pixel-based approach is being used, both approaches first require data, more specifically image data, to process. This data can be split into two categories. On the one side there is data, that is being provided by the existing 3D model that is meant to be processed. This 3D model provides a color texture that includes the color of its surface as well as illumination that is meant to be removed like light highlights, shadows and light color tint. In addition to the color texture, the 3D model also provides the 3D geometric data of the object, which enables further processing and data generation based on this information with the 3D computer graphics application Blender [24]. The following types of images were generated and used as possible input sources for the de-lighter models:

- Lightmap: A grayscale map that includes data about the light exposure of different parts of the geometry.
  - Areas exposed to high levels of light appear light gray, resulting in highlights in the color texture.
  - Areas exposed to lower levels of light appear dark gray, manifesting as shadows in the color texture.
- Light color: An RGB color texture that contains data about the light color that hit the geometry at different spots.
- UV texture area: A simple binary texture indicating which parts of the color texture map to the model geometry (white) and which do not (black). While this data has little visual impact, it helps indicate seams in the color texture and identifies which parts of the color texture can be ignored, as they do not affect the 3D model's appearance. Since the UV texture area is known from the 3D model, generating this input texture is trivial and requires no significant additional effort.

- Ambient Occlusion: A grayscale image representing the amount of ambient light reaching each point on the surface, accounting for occlusion caused by the object's geometry.

To provide a clearer illustration, a 3D model of a marble bust [14] was utilized as an example. The 3D model was placed into a simple example scene in Blender, which included a sunlight and a basic room-like structure surrounding the object, with a window in front allowing the sunlight to enter. The surrounding room geometry, window, and the marble bust itself were all illuminated by the sunlight, resulting in simulated lighting effects such as shadows and highlights on the 3D model within the scene. Figure 4.1 presents a camera render depicting the marble bust in this scene. The additional textures that can be generated using Blender are displayed in Figure 4.2.

4.2. Generation of Training Data

To train and evaluate the de-lighter models, a dataset of textures from various 3D models is required. The initial dataset comprises 3D models without any illumination baked into their diffuse color textures. These models can be manually crafted using 3D modeling software. A significant portion of the 3D models has been sourced from Poly Haven [31] and Sketchfab [59], which are repositories for 3D assets. The selection of 3D models for the dataset adheres to the following criteria:

- Diffuse Textures: The models should have diffuse color textures that are free from baked illumination or lighting effects. This ensures that the textures represent the true color of the materials and can be used as ground truth.

- Variety: The dataset should encompass a diverse range of materials and color schemes, including but not limited to wood, metal, stone, fabric, and plastic. This diversity is crucial for training the de-lighting models to generalize effectively across various material types and colors.

- Resolution: The textures should have a sufficiently high resolution to capture fine details and patterns. This ensures that the de-lighting models can accurately process and reconstruct fine-grained texture information.

- Licensing: The selected 3D models should have appropriate licensing terms that permit their use for research and development purposes.

Prior to processing 3D models for texture generation in a dataset, several preprocessing steps may be required to ensure consistency and compatibility with the dataset generation pipeline. These steps are crucial for creating a homogeneous set of textures that can be efficiently processed in the training pipelines of de-lighting models. When dealing with complex 3D models consisting of multiple meshes, simplification can be achieved by merging multiple meshes into a single mesh where applicable. Additionally, unnecessary meshes that are not meant to be the focus of the training dataset can be removed.

Figure 4.1.: Camera render of the marble bust 3D model [14].

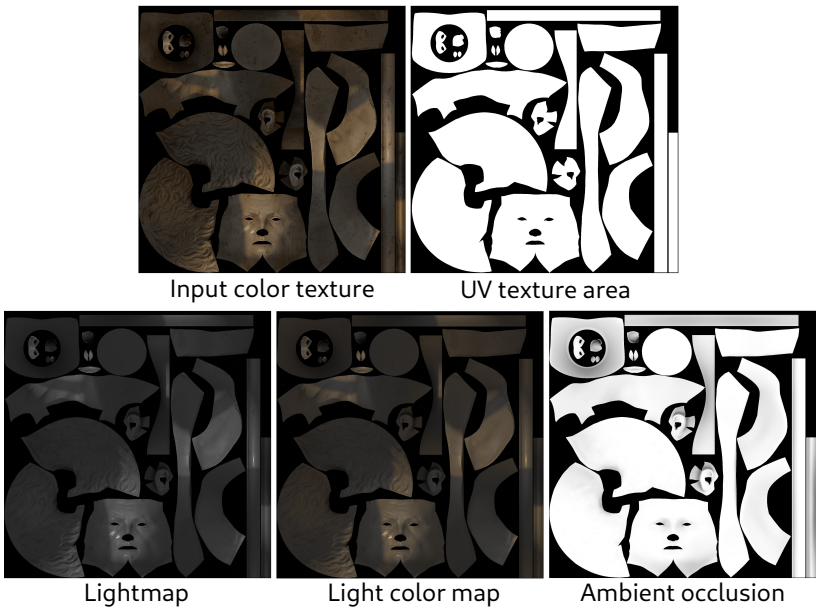

Figure 4.2.: Input textures for the de-lighter models.

This simplification process may also involve merging multiple shader materials of a 3D model into a single material, which includes combining diffuse, roughness, normal map, and other textures of the model. Furthermore, updating the UV mapping of the merged materials may be necessary to ensure proper texture alignment. In cases where a 3D model or scene contains multiple complex meshes representing separate visual objects, it may be beneficial to split these objects into separate Blender scenes. Each scene would then represent a stand-alone 3D model for generating input textures for the de-lighter model.

After completing the essential preprocessing steps for each individual 3D model, the models are imported into separate Blender scenes. These scenes are configured to facilitate the generation of the required textures, which includes the addition of at least one light source. The light source is used to bake illumination into the model's color texture and to generate the supplementary textures. To maintain a focused and manageable work environment, the training dataset creation process employs a simplified scene setup. This setup consists of a sunlight and a room-like geometry encapsulating the object. The room features a window through which the sunlight enters and illuminates the 3D model being processed. This setup is similar to the example showcasing a marble bust 3D model from the previous section, which is depicted in Figure 4.1. The illumination of a scene, i.e., sunlight, is baked into the color texture of the 3D model, resulting in a representation of the input color texture that is analogous to what would be obtained from a digitization process of a real-world object. Correspondingly, the light environment is baked into additional generated textures like the lightmap and light color textures. These textures are generated using Blender's physically-based path tracer engine, Cycles.

The Blender Python API [23] has been utilized to automate the generation of these textures with various light setups in the scene, such as sunlight from different angles and varying light color. Throughout the different experiments that have been made during the development of this work, diverse approaches for varying light setups have been employed, ranging from a range of sunlight direction and light color hues to randomly selected light color hue, intensity, and other properties. By leveraging the Blender Python API, scripts have been developed to automate the process of adjusting light parameters and subsequently invoking the rendering pipeline to generate the corresponding textures. The automated texture generation pipeline described herein enables the configuration of a consistent, uniform resolution for all generated textures. For instance, the pipeline can be configured to generate textures with a resolution of 256×256 pixels, 512×512 pixels, or any other resolution to be specified. This ensures that the resulting textures maintain a standardized size and quality throughout the entire generation process and can be used for further processing in the training pipelines of the de-lighter models without the necessity to individually resize them to potential texture resolution requirements imposed by the different de-lighting approaches.

4.3. Color Space for Image Data

In the field of image processing, color spaces are fundamental as they define the way colors are represented and manipulated digitally. Machine learning models that process color images require an understanding of these color spaces to effectively interpret and modify image data.

Before going into more details of each color space, it is essential to describe an important characteristic of a color space that is desired for the task of de-lighting. The characteristic of perceptual uniformity is beneficial for a color space to be used in the processing of de-lighting color textures. Perceptual uniformity refers to the attribute of a color space wherein numerical differences between color values correspond proportionally to the perceived differences in color by the human eye. This characteristic is particularly significant in de-lighting tasks, as the outcomes are intended for human perception, whether in the context of rendering 3D models with the processed color textures or direct inspection of the de-lighting results in the texture itself. Given that human color perception is non-linear, a color space that adapts to the nuances of human visual perception is advantageous. The benefits of perceptual uniformity become evident when contrasted with non-uniform color spaces. For instance, in a perceptually non-uniform color space, a fully saturated blue may appear inherently darker than a fully saturated green. Consequently, applying a fixed-step linear change to both colors would result in disparate perceived changes in color and lightness. This non-linearity would necessitate additional complexity in the de-lighting model, requiring it to learn and compensate for the color space's non-uniform characteristics. By utilizing a perceptually uniform color space, the de-lighting process can operate more efficiently and accurately, as the model can rely on a consistent relationship between numerical color changes and human perception.

As a general note, irrespective of the color space employed, when processing image data with the machine learning model, the color data values are normalized to the range of $[-1.0, 1.0]$ relative to the corresponding possible value range of each color space. The purpose of this normalization is twofold: firstly, to prevent color properties with a larger possible value range from dominating smaller scale color properties; and secondly, to match the range of commonly used activation functions, such as the hyperbolic tangent (tanh), which also produce outputs within the range of $[-1.0, 1.0]$.

The following sections discuss various commonly used color spaces, namely sRGB, HSV, HSL, CIELAB, and Oklab, with respect to their definitions, properties, value ranges, strengths, weaknesses, and potential use-cases, particularly focusing on their behavior in calculations and their suitability for machine learning applications.

4.3.1. sRGB

The sRGB color space, which stands for Standard Red Green Blue, is a standard RGB color space created cooperatively by Hewlett-Packard and Microsoft in 1996 for use on monitors, printers, and the Internet [6]. It was designed to approximate the gamut of the average cathode-ray tube (CRT) monitor of that time, making it a common choice

for software, hardware, and web content and is the most widely used color space for digital images. It is an additive color model where colors are represented as combinations of red, green, and blue color channels. Each color channel has a value range from 0 to 255, where 0 represents the minimum intensity (no contribution of that color) and 255 represents the maximum intensity of the color, with $(0, 0, 0)$ representing black and $(255, 255, 255)$ representing white. This 8-bit per channel format allows for 16,777,216 possible color combinations. sRGB is a device-dependent color space, meaning that the colors represented are subject to the characteristics of the device displaying them.

Despite its widespread adoption, the sRGB color space exhibits a perceptually non-uniform characteristic, which has implications for color management and image processing. It uses the D65 whitepoint, which corresponds to average daylight illumination, and has a gamma curve (usual gamma value is 2.2) to encode luminance (brightness) values [6]. This gamma curve was initially specified for the use of CRT monitors, where the double amount of voltage did not return the double amount of brightness output. Therefore a gamma correction factor of 2.2 has been commonly used to compensate for this non-linearity. The perceptually non-uniform characteristic of sRGB means that linear calculations, e.g., blending colors, directly in the sRGB space can lead to inaccurate results. For accurate color manipulation, it is often recommended to convert sRGB to a linear color space, perform the necessary calculations, and then convert back to sRGB for display or output. This process ensures more precise control over color adjustments and transformations.

4.3.2. HSV

The HSV (Hue, Saturation, Value) color space is a cylindrical-coordinate representation of colors, which was introduced in the late 1970s as an alternative representation to the RGB color space [60, 40]. It is designed to be orienting to the human user, being more intuitive and perceptually relevant than the RGB model [8]. In the HSV color space, hue (H) represents the color's position on the color wheel, with values ranging from 0° to 360°. Red is at 0°, green at 120°, and blue at 240°. Saturation (S) describes the purity or intensity of the color, with 0 % being gray and 100 % being fully saturated. Value (V) represents the brightness, with 0 % being black and 100 % being the brightest version of the color.

One strength of the HSV color space is its ability to separate color hue information from brightness, making it easier to perform color-based operations like thresholding and segmentation or in the case of this work the adaptation of the lightness or brightness of a given color. However, the HSV color space is not perceptually uniform, which can lead to issues when performing color comparisons or applying certain image processing techniques.

4.3.3. HSL

The HSL (Hue, Saturation, Lightness) color space is another cylindrical-coordinate representation of colors, similar to HSV. It also separates color information (hue and saturation) from lightness, but defines lightness differently than the value component

in HSV [40]. In the HSL color space, the hue and saturation components maintain comparable semantic meanings and ranges to those in the HSV color space. However, the HSL model employs a distinct approach to specifying lightness. In this system, a lightness value of 0 % corresponds to black, 50 % represents the pure, fully saturated color, and 100 % denotes white. In this model, the fully saturated colors are located in a plane midway between black and white, rather than at the top of the hexcone as in HSV. This mapping scheme aims to provide a more intuitive and user-friendly representation of colors [40].

Similar to the previously described HSV color space, the HSL color space also has the advantage of modeling a color's lightness in a separate value. This gives a promising color property for color-based operations like the adjustment of the lightness of a color when processing the color data in the de-lighting pipeline. Nevertheless the HSL color space is also not perceptually uniform, which can potentially reduce the quality of the results.

The aforementioned property renders both color spaces more intuitive and easier to control from a user's perspective compared to the (s)RGB color space in many cases. However, these color spaces are potentially less suitable for certain computations proposed in this work.

4.3.4. CIELAB

The CIELAB color space, also known as LAB, is a color space defined by the International Commission on Illumination (CIE) in 1976 [62]. It is designed to be perceptually uniform, meaning that the numerical differences between colors in the CIELAB color space correspond to the perceived differences by the human eye. This property makes CIELAB particularly useful for color management, color manipulation, and color difference calculations in various applications, including image processing and computer vision. The CIELAB color space is a three-dimensional color space that represents colors using three values: L*, a*, and b*. The L* value represents the lightness of the color, ranging from 0 (black) to 100 (white). The a* value represents the green-red component, with negative values indicating green and positive values indicating red, with a typical value range of $[-128, 127]$. The b* value represents the blue-yellow component, with negative values indicating blue and positive values indicating yellow. Similar to the a* value, the b* value has a typical range of $[-128, 127]$.

One of the most significant properties of the CIELAB color space is its attempt to be perceptually uniform. This property allows for more accurate color difference calculations and color manipulations that align with human perception. CIELAB is particularly useful for tasks such as color correction, color grading, and color matching, where preserving the perceived color relationships is crucial. It is worth noting that although the CIELAB color space strives for perceptual uniformity, it does not achieve perfect uniformity across the entire color space. In particular, the prediction of blue hues can suffer from a lack of precision [52]. Another advantage of CIELAB is its device independence. CIELAB colors can be consistently reproduced across different devices and media, making it an ideal choice for color communication and color management workflows.

4.3.5. Oklab

The Oklab color space, proposed by Björn Ottoson in 2020 [52], is also a perceptually uniform color space designed to improve upon the limitations of existing color spaces, especially the perceptual properties of the CIELAB color space. Similar to the CIELAB color space, Oklab is an opponent color space using three values: "L", "a" and "b". Lightness (L) represents the perceived lightness of a color, ranging from 0.0 (black) to 1.0 (white). The coordinate "a" represents the balance between green and red colors. Negative values indicate a shift towards green, while positive values indicate a shift towards red. The coordinate "b" represents the balance between blue and yellow colors. Negative values indicate a shift towards blue, while positive values indicate a shift towards yellow. Although the theoretical ranges of the "a" and "b" coordinates are unlimited, practical considerations typically constrain their values to a range that encompasses the full gamut of human color perception. When using the sRGB color space as a reference, which represents a smaller subset of the visible color gamut, the necessary range to fully represent its colors would be approximately $[-0.233, 0.276]$ for the "a" coordinate and $[-0.311, 0.198]$ for the "b" coordinate.

The described characteristics of the Oklab color space show that it has the same advantages as the CIELAB color space in terms of color data processing. Figure 4.3

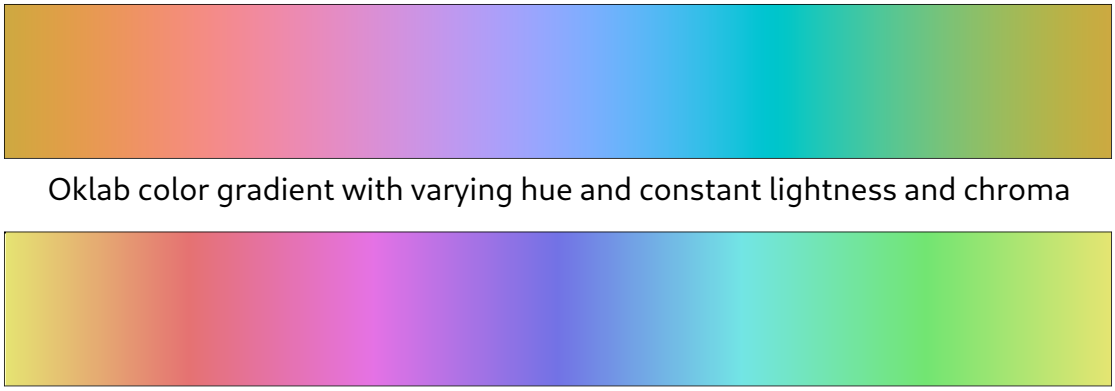

Figure 4.3.: Comparing Oklab to HSV as shown in [52].

compares a color gradient with varying hue using the Oklab and HSV color spaces. As can be seen, the HSV gradient is uneven compared to the Oklab gradient [52]. Some areas in the HSV color gradient are perceived to be darker or lighter than others, which is not the case in the Oklab color gradient. This demonstrates the improved perceptual uniformity of the Oklab color space compared to HSV. Figure 4.4 shows a color blend between white and a blue color using the three color spaces CIELAB, Oklab and HSV. This comparison visualizes the lack of precision of the CIELAB and HSV color spaces. In specifics the example with a blue color is picked, because it highlights the limitations

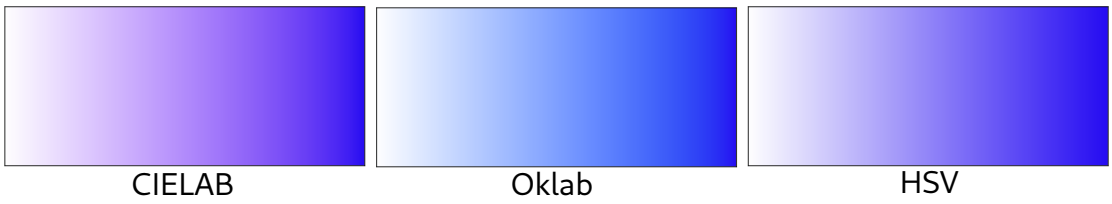

Figure 4.4.: Color blend between white and blue as shown in [52].

of precision of the CIELAB color space especially with blue color hues [52]. While the color blend using the Oklab color space shows an even transition from white to blue, the CIELAB and HSV color spaces introduce a noticeable shift in hue into a slight purple during the transition.

4.4. Image-Based Architecture

This section explores an image-based machine learning approach for removing illumination baked into the color texture of a 3D model. This treats the input color texture and its supplementary textures as a single entity, processing the entire image simultaneously. The proposed method employs a CGAN training architecture, which has been shown to be effective in learning a mapping from an input image to an output image [38]. CGANs are a variant of generative adversarial networks (GAN) that allow for additional control over the generated output by conditioning the generator on supplementary information, such as additional images. This characteristic makes CGANs particularly suitable for a task like the de-lighting color textures in 3D models. In the context of de-lighting color textures, the initial input to the generator, in this work named de-lighter model, is the color texture of the 3D model. To enhance the performance and control over the output, additional data is generated and utilized, including a lightmap, a light color map, an ambient occlusion texture and a UV texture area map.

In the current implementation, the lightmap and light color textures have been maintained as separate entities. This design decision was based on the observation that utilizing a single-channel grayscale value provides a more straightforward and manageable approach for representing the spectrum of light intensity, ranging from low to high illumination levels, compared to employing a three-channel RGB color representation. The usage of a color space that maps a color's brightness to a specific property, such as the lightness property in Oklab, can minimize the need for a separate grayscale lightmap texture. This approach could lead to the possibility of merging the information from the illumination environment in the grayscale lightmap and the colored light color map into a single, colored, light environment map. However, since the proposed approach also aims to compare the usage of different color spaces, including color spaces that use RGB color channels and do not map a value to a color's brightness, the grayscale lightmap is kept as a separate input. Further details on the performance of the various de-lighter model input configurations are provided in the later Chapter 5.

The generator and discriminator neural networks employed in this work follow the architecture described by Isola et al. [38]. This means that the generator is a modified U-Net consisting of an encoder and decoder. The encoder is a series of Convolution - Batch normalization - ReLu layers, with the exception of the first layer, which does not apply batch normalization. The detailed architecture of the layers of the generator is shown in Figure 4.5. Conversely, the decoder consists of Transposed convolution - Batch normalization - Dropout - ReLu layers, with dropout applied only to the first three layers. Skip connections, as in the original U-Net architecture [56], are established between corresponding layers of the encoder and decoder.

The discriminator utilizes a convolutional PatchGAN architecture to classify patches of the input image data. Its layer structure resembles that of the generator's encoder, consisting of Convolution - Batch Normalization - ReLU layers. A detailed overview of the discriminator's layers can be seen in Figure 4.6.

The generator loss is computed as a combination of two components. The first component is the feedback from the discriminator, meaning the binary cross-entropy between the discriminator output and an array of ones with the same shape. A discriminator output of one for a given image means that the discriminator classified the given image as a real, genuine image. The second component is the L1 loss, calculated as the mean absolute error between the generator output and the ground truth image from the dataset.

The discriminator loss is composed of two distinct components. The first component is the binary cross-entropy between the discriminator's output when provided with the "real" images containing the ground truth color texture and an array of ones with an identical shape. The second component is the binary cross-entropy between the discriminator's output when given the "fake" images, which include the synthesized color texture of the generator, and an array of zeros. By comparing the discriminator's output with the ground truth labels (ones for real images and zeros for fake images), the loss function effectively measures the discriminator's ability to distinguish between genuine and generated images. Minimizing this loss encourages the discriminator to accurately classify the images, thus providing valuable feedback to the generator during the training process.

During the training the generator is fed with the input textures obtained from the training dataset and tasked to output an illumination-free "fake" color texture. The generated output is then given to the discriminator, which then tries to recognize it as "fake" or "real" which is the feedback that is then given back to the generator as one component of its loss function. The training iteration loop and competition between generator and discriminator is employed as described in Chapter 2.2 and shown in Figure 2.1.

The development and implementation of the described approach relied on several key technologies. The primary machine learning library used was TensorFlow, which provided the necessary tools for creating, training, and running the models. TensorFlow's extensive collection of pre-built functions and classes greatly facilitated the construction and manipulation of the neural network architectures, as well as the management of the training process. The Python programming language served as the primary language for orchestrating the overall process structure and interacting with the TensorFlow library.

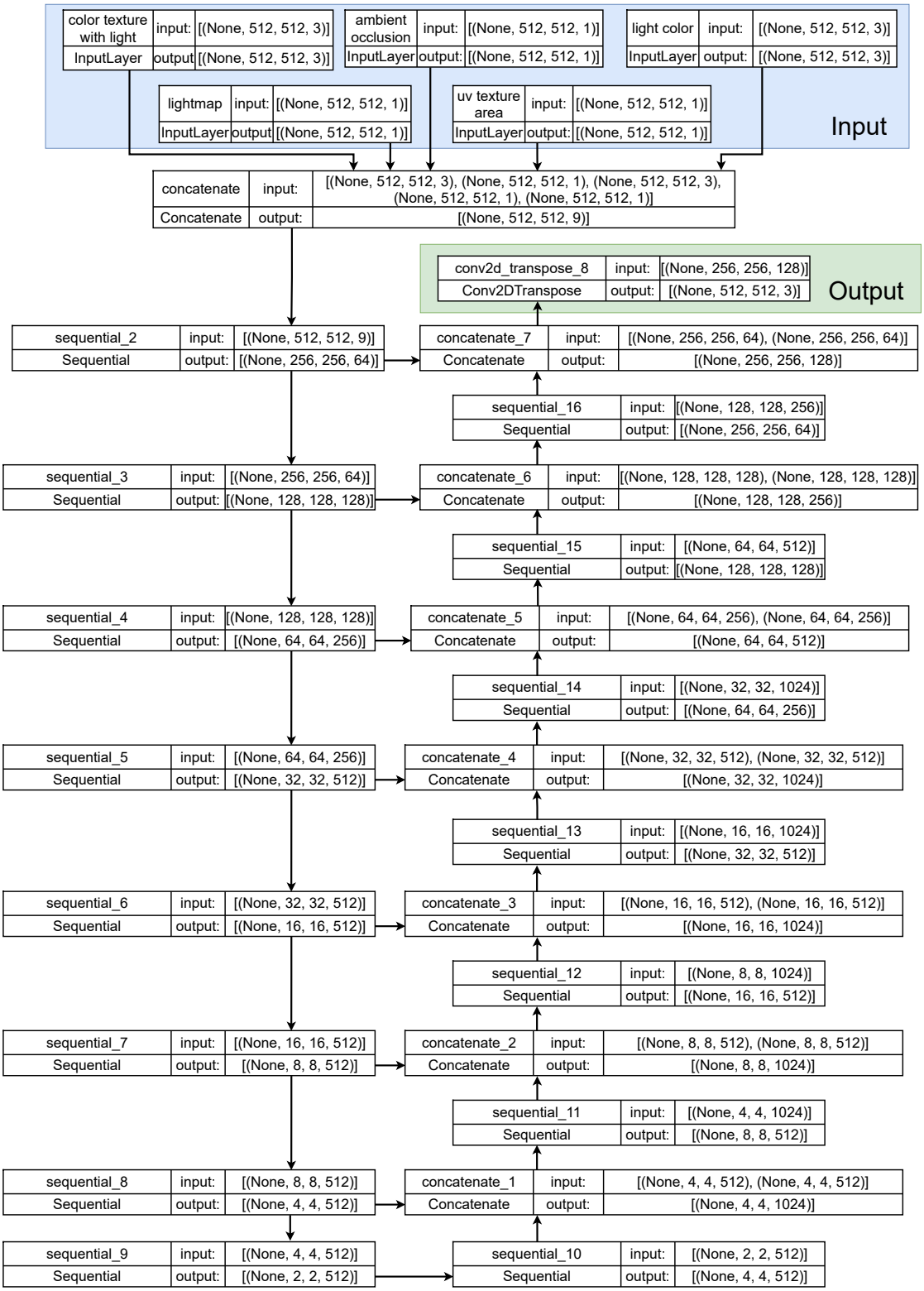

Figure 4.5.: Detailed architecture of the image-based de-lighter model (CGAN generator).

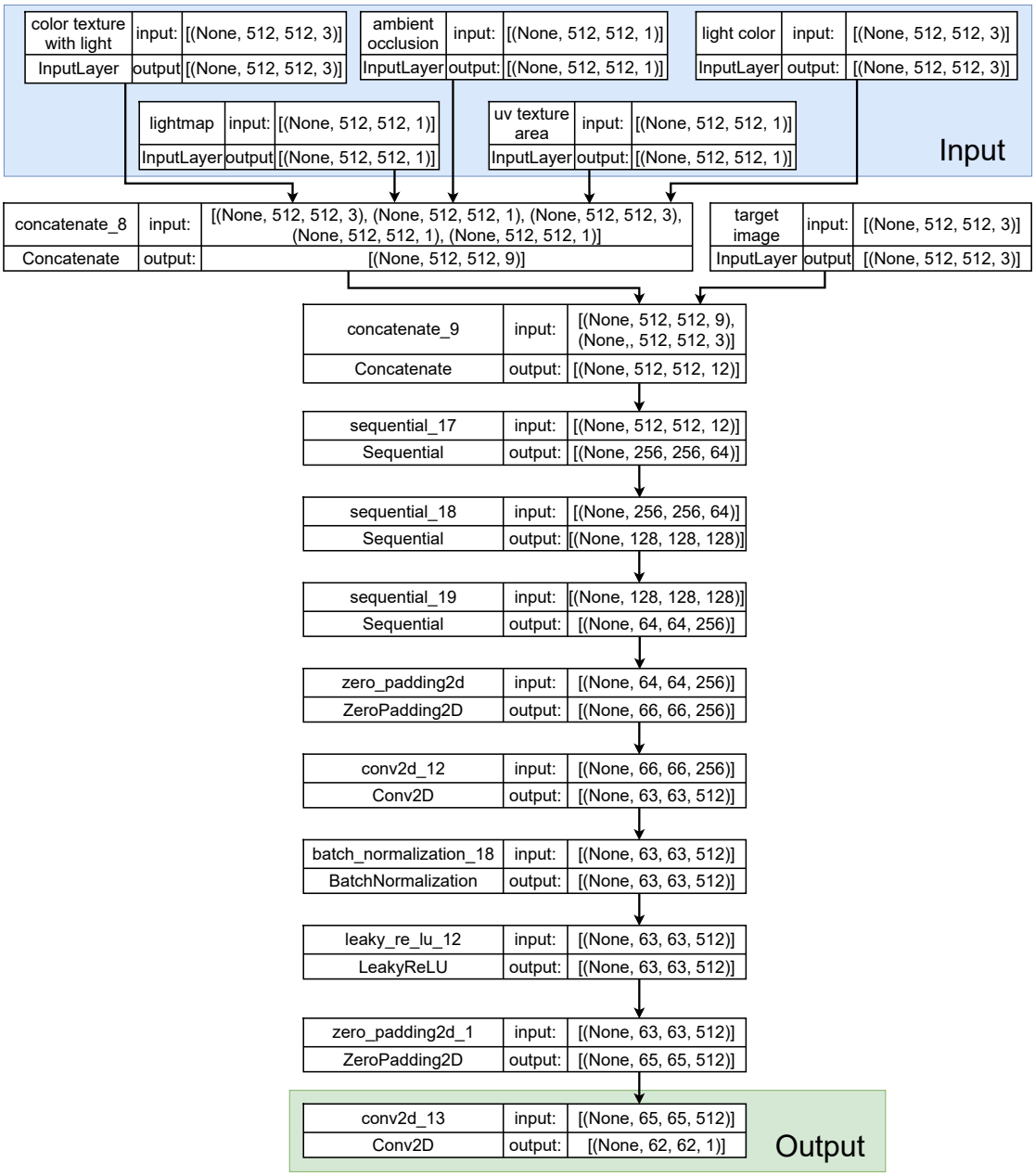

Figure 4.6.: Detailed architecture of the discriminator.

The majority of the Python logic was organized and executed within Jupyter Notebooks, which provided an interactive and user-friendly environment for developing, testing, and documenting the code. In addition to Python, the Rust programming language was utilized to handle various preprocessing tasks and data manipulations. The functionality written in Rust was primarily focused on loading image data from disk, preprocessing the data into the appropriate formats required by the machine learning model, and performing color space conversions and normalizations. This functionality is being used in Python through the use of a Python library that wraps that Rust code, allowing for a direct integration in the overall workflow.

Overall about 5 to 20 3D models were utilized for the dataset generation throughout the development of this work. For this image-based approach, 10×10 image pairs were generated per 3D model, consisting of 10 different sunlight directions and 10 varying sunlight color hue combinations, resulting in a total of 100 image pairs per model. Consequently, the dataset contains a total of up to 2000 image pairs. Given the selected configuration, a training dataset for the image-based de-lighting model necessitates the computation of five textures, including the input color texture with baked illumination, UV texture area map, lightmap, light color map and ambient occlusion. Since the 3D models used for the generation of the training dataset have color textures that are free from illumination effects, their original color textures are used as ground truth textures as is. This requirement of requiring to compute five additional textures for a single training step of this model significantly increases the computational effort for dataset generation. To reduce this effort and maintain a practical iteration time, lower texture resolutions were considered during the dataset generation process. Throughout the development phase, texture resolutions of 256×256, 512×512, and 1024×1024 pixels were utilized. As a compromise between computational efficiency and output quality, a resolution of 512×512 pixels was chosen for the final iteration and results.

4.5. Pixel-Based Architecture

In contrast to the previously described image-based approach, this section presents an alternative de-lighting model architecture that processes input image data on a per-pixel basis. Rather than treating the input color texture and supplementary textures as a single entity, this model iterates over each pixel in the input textures and feeds the data for every pixel into the de-lighting model individually.

The proposed approach leverages the PyTorch machine learning library, specifically its C++ API known as LibTorch, for implementing and training the machine learning models. The Rust programming language is used for all custom logic, including data loading, preprocessing, and manipulation. The integration between Rust and LibTorch is achieved through the tch Rust crate, which provides Rust bindings to the LibTorch library. As opposed to the previously proposed image-based approach, which utilized the TensorFlow machine learning library and a combination of Python and Rust, this approach streamlines the development process by using LibTorch and Rust exclusively. Python is no longer used to orchestrate calls to the machine learning library, all custom

logic is implemented in Rust instead. This change simplifies the codebase and reduces the overhead of integrating multiple programming languages.

It is worth noting that the development of the pixel-based approach started later than the development of the image-based approach. The decision to use PyTorch here instead of TensorFlow again was made solely to explore the different available machine learning libraries. It is not expected that the choice between PyTorch and TensorFlow will bring a significant advantage in terms of the quality of the de-lighting results of the respective solutions.

For a given 3D model, the proposed pixel-based method utilizes a selected set of texture inputs: the color texture with embedded lighting, a UV texture area map, and a light color map. This choice is grounded in the exploration of various color spaces, which ultimately led to the adoption of the Oklab color space for further development. The Oklab color space is particularly advantageous because it incorporates a lightness component, which conveys the brightness of colors. Consequently, the grayscale lightmap texture was deemed redundant and thus omitted, as its information is encapsulated within the light color map. Similarly, the ambient occlusion map was set aside, since investigations into different combinations of image data suggested that including the ambient occlusion texture did not significantly enhance the results. A more comprehensive discussion on this topic will be presented in the subsequent Chapter 5.

When iterating over each individual pixel of the image data, the first step is to evaluate the corresponding pixel of the UV area map. If the UV area map pixel is black, it indicates that the pixels of the textures do not map to any part of the geometry of the 3D model. In this case, the de-lighter logic can skip this pixel and proceed to the next one. However, if the UV area map pixel is white, the corresponding pixels of the other textures, namely the input color and the light color map, are retrieved. Since the colored images are typically stored on disk as standard image files, the color data of those two images is in the sRGB color space. Therefore, the sRGB color data needs to be converted into the chosen color space, which in this case is Oklab. Additionally, the Oklab color values are then normalized to a range of $[-1.0, 1.0]$ to match the input format required by the de-lighter model. As described in the earlier section about the Oklab color space (see Section 4.3.5), the “a” and “b” values of an Oklab color theoretically have no bounds. However, since the color data from the image files is originally in the sRGB color space, the possible color range of sRGB has been adapted to Oklab with the value bounds as stated in Section 4.3.5. The normalized color data is then fed into the de-lighter model logic, which concatenates the six values of the pixels from the two input textures, i.e., the color texture and the light color map, into a single tensor and passes it to the de-lighter neural network. The network returns three output values, which represent the normalized color values of the processed pixel, ideally free from illumination effects. These values are then de-normalized back to the corresponding Oklab color values and converted to the sRGB color space. The resulting output color data is stored in an image buffer that was initialized with all black pixels and is overwritten where necessary according to the UV area map. Once the iteration is complete, the image data in the output image buffer is written back to a file, which can be used as the color texture for the 3D model.

To complement the description of the process, a code snippet containing Rust code that demonstrates the processing steps can be found in Listing A.1. This code snippet should be considered as a simplified version of the actual implementation, providing a better overview of the general data processing flow. Aspects such as batching pixel data for batched calls to the de-lighter model, reconstructing an image from the batched output data, color space conversion, and normalization specifics have been encapsulated away in function calls for clarity. The provided example code in Listing A.1 demonstrates the usage of two generic types, `ModelType` and `ColorSpace`, in the function `run_pixel_de_lighting`. The `ModelType` type is required to implement the `PixelDelighterModel` trait, which encapsulates the logic for the de-lighter model. This trait is essential during both the training of the neural network and the actual de-lighting process. The trait mandates the implementation of methods such as `loss` and `get_optimizer`, which specify the loss function and optimizer used for training. Additionally, the `forward` method, used to apply the neural network to a given input tensor, is also required. This method is utilized in the provided code snippet (Listing A.1) through the call to the model's `delight` method. The `PixelDelighterModel` trait defines additional functionality requirements and provides general logic that is consistent across all implementing models, thereby reducing code duplication. The `ColorSpace` type is required to implement the `ColorSpacePixel` trait, which specifies the necessary logic for handling a specific color space. This includes converting sRGB color data into normalized values of the given color space and de-normalizing the output color values of the de-lighter model back into the corresponding color space values before further converting them to sRGB. The method calls `convert_rgb_to_normalized_color_space` and `convert_normalized_color_space_to_rgb` in the provided code snippet utilize this functionality.

The use of generic types that require the implementation of specific traits allows for the creation of multiple different concrete types for color spaces and de-lighter models. This approach enables the interchangeable selection of the desired implementation to be used in the training process of the de-lighter model and its subsequent usage for de-lighting functionality. As a result, different `ColorSpace` trait implementations for color spaces such as sRGB, HSV, HSL, CIELAB and Oklab have been added. Additionally, various interchangeable `PixelDelighterModel` trait implementations for different neural network approaches, including diverse architectures with various layers and activation functions, have been explored during development. As a result, the layer architecture depicted in Figure 4.7 was found to yield the best results.

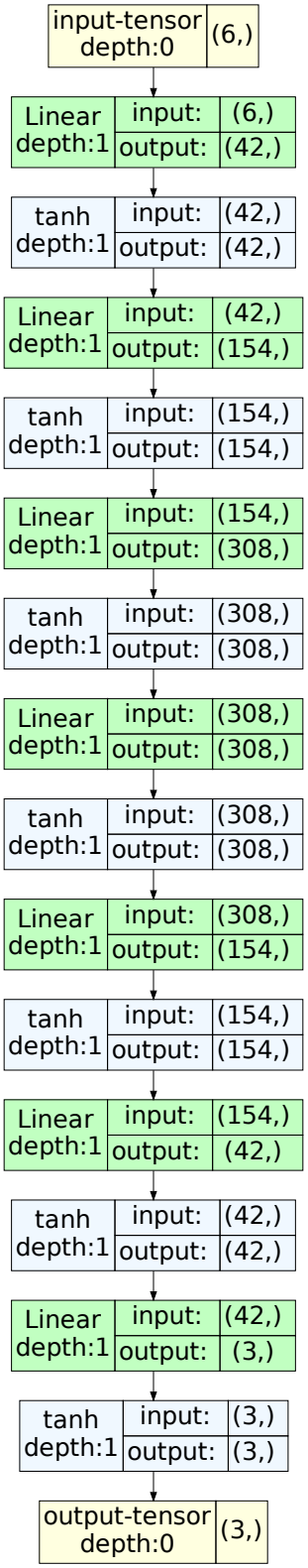

Figure 4.7.: Detailed architecture of the pixel-based de-lighter model.

The architecture of the pixel-based de-lighter model is as follows:

- Input layer: Accepts the input data with the appropriate dimensions.
- Linear layers: A series of linear layers with varying dimensions are employed to learn the necessary transformations. The dimensions of these layers are determined based on the complexity of the task and the size of the input data.
- Activation function: Each linear layer is followed by a tanh activation function, which introduces non-linearity into the model and enables it to learn complex patterns and relationships in the data.
- Output layer: The combination of the final linear layer and the tanh activation function generates the model's output, which constitutes the illumination-free representation of the corresponding pixel from the input image.

The simplicity of this architecture, combined with the appropriate choice of layer dimensions and activation functions, has shown to be a good fit in achieving accurate results for the pixel-based de-lighter model when evaluated against the dataset's ground truth images.

The training of this neural network requires only a single pixel of the input textures for each training step. With input color textures and their supplementary generated textures at a resolution of 256×256 pixels, this results in a maximum of 65,536 possible training steps from a single texture. However, this maximum assumes a complete white texture as the UV area map, which would include every pixel of the texture in the processing. In practice, the content of textures for many 3D models does not map to the geometry of the model to an extent of 100 %, meaning the number of pixels used for training per texture varies depending on the specific 3D model and its UV mapping. During the exploration of different neural network architectures and other tweaks, various texture resolutions were used for the training dataset, including 256×256, 512×512, and 1024×1024 pixels. Ultimately, a texture resolution of 256×256 pixels was preferred for several reasons. Firstly, a single texture at this resolution already encompasses a substantial number of pixels and therefore training steps, which are sufficient to meet the required training steps for the neural network to effectively learn from a single given input texture. Secondly, the lower resolution of 256×256 pixels offers an advantage in terms of dataset diversity. By opting for this resolution, it becomes possible to include a greater number of distinct textures sourced from a wider array of 3D models and varying light environments. This increased diversity in the training dataset is beneficial for effectively training the neural network to handle a broad spectrum of scenarios and conditions. By exposing the network to a rich variety of textures and lighting conditions, it is expected to develop a more robust and adaptable understanding of the underlying patterns and features, enabling it to generalize better to novel situations.

For the training dataset generation, a total of 30 3D models were used. For this pixel-based approach 20 image pairs were generated per 3D model. These image pairs consisted of 10 distinct sunlight directions, with each direction being rendered twice: once using a relatively neutral white sunlight color and once using a randomly selected light color. This process resulted in the creation of a training dataset including 600 image pairs, each with a resolution of 256×256 pixels.

4.6. Light Environment Estimation

When given a 3D model, one of the first steps in the de-lighting process involves setting up the light environment in the 3D scene to generate the additional light data input textures required by the de-lighter model. To achieve this, the 3D model is loaded into an empty Blender scene, and light sources are added to represent the original light environment that existed during the digitization of the real-world object. In cases where no additional data about the scene illumination has been recorded, such as a light probe to capture the environment and ambient light or the specification of artificial light sources like LED light boxes, the only source of information about the light environment is the color texture of the 3D model itself, which has the illumination baked in.

Reconstructing the light setup in the scene involves estimating the number, position or direction, strength, and color of the light sources. When a real-world object has been processed in a simple light environment, such as a clear sunny day with the sun as the predominant light source casting distinct shadows with clear-cut edges, the estimation for reconstructing the light environment in the Blender scene can be attempted based on the clear shadows baked into the color texture of the 3D model. By adding a sunlight source to the scene and adjusting its properties, such as light direction, strength, and color, until the light rendered in Blender matches the shadows in the color texture, the light environment in the Blender scene can be approximated. However, estimating the light environment based on the color texture information is not always straightforward and can be challenging in more complex lighting scenarios. A light environment with multiple individual light sources, each with different light properties, can make the initial estimation process more difficult and may require additional information about the light setup.

To improve the accuracy of the manual light environment estimation process, an optimization process has been developed that takes an initial manual estimation of the light environment as input and then attempts to improve this approximation to more closely match the actual light scene present during the digitization of the real-world object. This optimization process is tasked with estimating a light source's properties, such as position or direction, strength, and color. It is an iterative process that alternately adjusts the individual light source properties and compares the resulting light environment with the light environment baked into the 3D model's color texture. For this iterative optimization task, a key metric to describe the difference between the estimated light scene and the actual light environment baked into the color texture must be defined. This metric is used as a cost function that is minimized by the optimization process.

The optimization process is split into multiple stages. First, the position or light direction, depending on the light source type, and strength of the light source are estimated. To provide a cost function as a metric to minimize for this iterative process, the sRGB color data from the 3D model's color texture is converted into a color space that separates the color's brightness property from its hue values, such as HSL, HSV, CIELAB, or Oklab. The lightness color property is then used as an approximation of the strength of light hitting each pixel of the model's color texture. At each iteration step of the optimization process, a lightmap is created that reflects the intensity of light hitting each pixel of the model's color texture with grayscale color data. The grayscale pixel data of the lightmap is then compared to the lightness value of the corresponding pixel in the model's color texture, forming a difference between the generated lightmap representing the estimated light environment in this iteration and the original light environment baked into the 3D model's color texture. Given this metric as a cost function to be minimized, an optimization method is applied to iteratively adjust the light source position or light direction and strength, attempting to achieve the best estimation of these properties with a minimal cost function value.

With an approximation of the light's position (or direction) and strength, a similar process is performed to estimate the light color. Instead of relying on the lightness value of the 3D model's color texture, the color hue properties, such as the "a" and "b" values in the Oklab color space for example, are considered. In this case, the light is baked into the input color texture that already includes the original illumination. As a metric for defining the cost function for the optimization process for the light source's color, the hue difference, i.e., the difference of the "a" and "b" values, between the original and newly baked color textures is computed. Assuming that baking the same light source with the same light color again will primarily affect the color's brightness while preserving the hue, the hue difference can be used as a cost function to be minimized. With the cost function for the light source's color defined, the iterative optimization process can be run similarly to the previous step for estimating the light source's position or light direction and its strength, leading to a full approximation of the given light environment properties, including light source position or direction, strength, and color.

As a result, the approximated light scene in Blender can then be used to generate the required light data textures, i.e., lightmap and light color map, for the de-lighter model to process the color texture of the 3D model.

5. Evaluation

This chapter presents the evaluation of the proposed de-lighting methods for color textures of 3D models. First, the different color spaces mentioned in the previous chapter are compared regarding their performance in the de-lighting process. Then, the two different model architectures, the image-based and the pixel-based approach, are evaluated.

As a general performance measurement metric, the peak signal-to-noise ratio (PSNR) is used to quantify the quality of the de-lighted textures during the training process. By calculating the PSNR, the difference between the original illumination-free color texture, i.e., ground truth, and the de-lighted texture output from the de-lighter model is computed, providing insights into the effectiveness of the de-lighter models. As a basis for the calculation of the PSNR, six different 3D models have been selected as a test dataset. Each of the six models has been rendered in an individual pre-defined light environment, including a sun light source with a specific angle and color, resulting in six individual de-lighting benchmark cases.

To evaluate the progress of the training of a de-lighter model, the performance of the model in its current state at a given training step is measured as follows. During the training of a de-lighter model, the model is tasked with de-lighting the color textures of the given six 3D models in the test dataset at a given rate of training steps, e.g., compute the PSNR for each of the six test 3D models after every 100 training steps. The individual PSNR values for each of the test 3D models are then averaged to receive an average performance of the de-lighter model at a given training step. The averaged PSNR values at their given training steps are then plotted and provided in figures for better comparison.

The training processes of both de-lighter model approaches also contain a part of randomization to reduce overfitting of the model to the provided training dataset. This includes random selection of training textures for both approaches, as well as random flipping and reorganization of the pixels in the images for the image-based model and randomized pixel ordering for the dataset of the pixel-based model. To enable running multiple training runs with different configurations, i.e., the usage of different color spaces for the image data or different model architectures, and still obtain comparable results, a benchmark training process has been created. This process includes the generation of a static training dataset that is used for all training runs in a specific benchmark comparison. This means the randomized parts, such as randomly flipped images for the image-based approach or random pixel ordering for the pixel-based approach, are generated once and stored in a benchmark data file that specifies all properties for the input data of a specific training step. This benchmark data file is then used for all training runs in a specific benchmark comparison, enabling a comparable training data environment for the different training runs. In addition to utilizing static benchmark

training and test datasets, the machine learning frameworks employed, namely TensorFlow and PyTorch, have been configured to minimize randomness and enhance reproducibility of the training process. This configuration facilitates more accurate comparisons between different benchmark runs. The measures implemented include specifying a fixed random seed to ensure consistent randomization across all training runs, such as setting identical initial weights for the de-lighter model to guarantee that each model begins from the same starting point. Furthermore, the use of deterministic operations has been configured where feasible to further reduce variability.

The light environment estimation process is also evaluated, as it is crucial for generating reliable input data for the de-lighter models. Finally, the limitations of the proposed methods are discussed, and potential areas for future work are identified.

5.1. Color Space Comparison

To assess the impact of different color spaces on the performance of the de-lighter models, both the image-based and pixel-based approaches were evaluated using various color spaces. By comparing the evolution of the PSNR values for each color space, insights can be gained into the potential benefits and limitations of each color space in the context of de-lighting.

As previously described, the PSNR metric is used to evaluate the performance of the de-lighter model at different training steps. Since the different color spaces vary in their specification of describing a given color using different properties, the same visual color change can result in different pixel value changes in the different color spaces, which then results in different numeric differences when comparing color values in different color spaces. To mitigate this discrepancy, the de-lighter models are trained and used using the individually specified color space of choice, but when comparing the de-lighter output image with the ground truth image to calculate the PSNR metric, the image data is always handled in the sRGB color space for this color space comparison.

As a first comparison use case, the pixel-based model was trained using the sRGB, HSV, HSL, CIELAB, and Oklab color spaces. The pixel-based model architecture used in this evaluation benchmark uses the pixel values of the color texture and light color map texture as input and outputs the de-lighted texture. This architecture resembles the same input and output values as described in detail in the previous chapter in the corresponding Section 4.5. Since this architecture does not contain the grayscale lightmap that represents the light intensity in a single grayscale value, but only the colored pixel values of the light color map as light data, this architecture is a fitting benchmark to test whether the de-lighter model is able to recognize light intensity imbalances in the input color texture and also correct them where necessary when provided with input color data in the different color spaces. This scenario is especially interesting regarding the properties of some color spaces like HSV, HSL as well as CIELAB and Oklab, which are designed to represent the color's lightness property in a designated channel. The sRGB color space, on the other hand, does not separate the lightness property. Figure 5.1 presents the PSNR values obtained during the training of the pixel-based models using the sRGB,

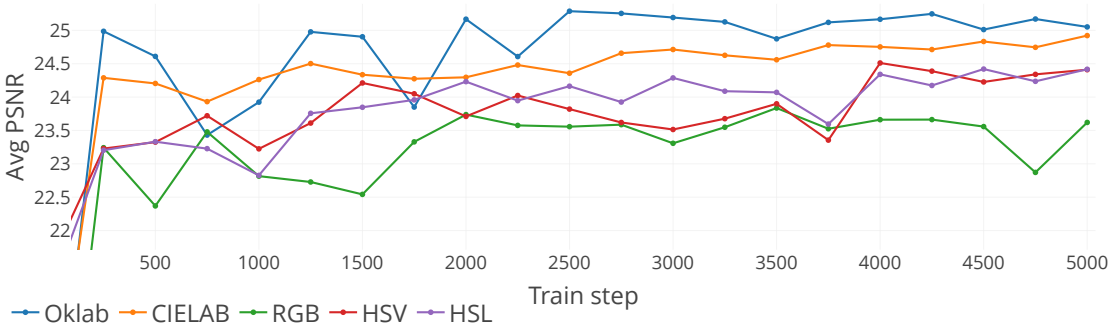

Figure 5.1.: Comparison of averaged PSNR values during training of the pixel-based model using different color spaces.

HSV, HSL, CIELAB, and Oklab color spaces. The experimental results demonstrate that the usage of the Oklab color space yields superior PSNR values during the majority of training iterations, followed by the CIELAB color space as the second-best. Conversely, the sRGB, HSV, and HSL color spaces demonstrate lower PSNR values, indicating a worse de-lighting performance. Among the evaluated color spaces, sRGB demonstrates the poorest performance, suggesting its limited capacity to accurately represent essential color properties for de-lighting applications, such as color lightness. This limitation impairs the de-lighter model's ability to effectively remove illumination effects. The HSV and HSL color spaces show improvements compared to sRGB, yet their performance remains inferior to that of CIELAB and Oklab. Despite incorporating dedicated channels for color lightness / brightness, which enhances their performance relative to sRGB, these color spaces lack crucial characteristics such as perceptual uniformity. Consequently, their effectiveness in de-lighting tasks falls short of the results achieved with CIELAB and Oklab color spaces.

As a second comparison, the image-based model was trained similarly using the sRGB, HSV, HSL, CIELAB, and Oklab color spaces. In contrast to the previous benchmark with the pixel-based model, the image-based model architecture used in this evaluation benchmark uses all available input textures, i.e., the illuminated color texture, lightmap, light color map, and ambient occlusion texture, to output the de-lighted texture, as mentioned in the previous Section 4.4. The PSNR values observed during the individual training steps of the image-based model, utilizing various color spaces, are illustrated in Figure 5.2. Consistent with the results obtained from the pixel-based model, the image-based model trained with the Oklab color space consistently demonstrates superior performance compared to other color spaces, with the CIELAB color space following as the second-best performer. The sRGB, HSV, and HSL color spaces consistently demonstrate lower PSNR values. In this scenario, where the grayscale lightmap specifically represents light intensity information in the input data independently of color space values, the HSV and HSL color spaces do not maintain the same level of advantage over sRGB as observed in the pixel-based model's benchmark. Nevertheless, the Oklab and CIELAB

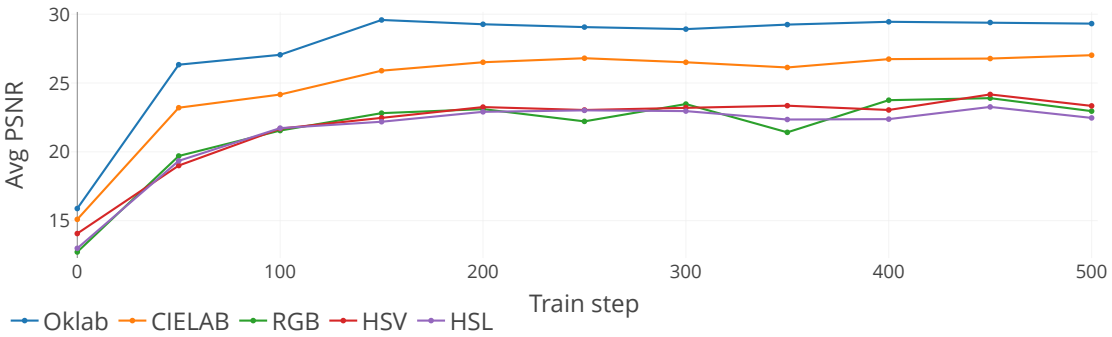

Figure 5.2.: Comparison of averaged PSNR values during training of the image-based model using different color spaces.

color spaces outperform the sRGB, HSV, and HSL color spaces. Despite the presence of dedicated channels for color lightness / brightness in HSV and HSL, the Oklab and CIELAB color spaces demonstrate superior performance. This can be attributed to their perceptual uniformity and enhanced numerical linear characteristics, which contribute to more effective representation and processing of color information in the context of this image-based model.

These findings highlight the importance of selecting an appropriate color space for the de-lighting task. The perceptual uniformity and separation of lightness information in color spaces like Oklab and CIELAB appear to be beneficial for accurately removing illumination effects while preserving the underlying color information. The superior performance of Oklab over CIELAB can be attributed to its improved perceptual uniformity, particularly in the blue hue range. Based on these results, the Oklab color space is used when handling image color data in all further evaluation of the de-lighting process, running with the image-based as well as pixel-based architectures.

5.2. Image-Based Architecture

During the development of the image-based de-lighter model various tweaks and adjustments were implemented and evaluated to improve the results of the de-lighting process. A critical aspect of this evaluation was the utilization of various input textures for the de-lighter model. Besides the 3D model's color texture with illumination baked in, various combinations of the available supplementary input textures, including the lightmap, light color map, ambient occlusion texture and UV texture area map have been explored. Their impact on the de-lighting process was assessed using a methodology akin to the approach used in the color space comparison outlined in Section 5.1, which involved comparing PSNR values of de-lighting results from a test dataset during model training. Figure 5.3 presents the PSNR values obtained during the training of the image-based de-lighter model. The results demonstrate an improvement in image quality as the training

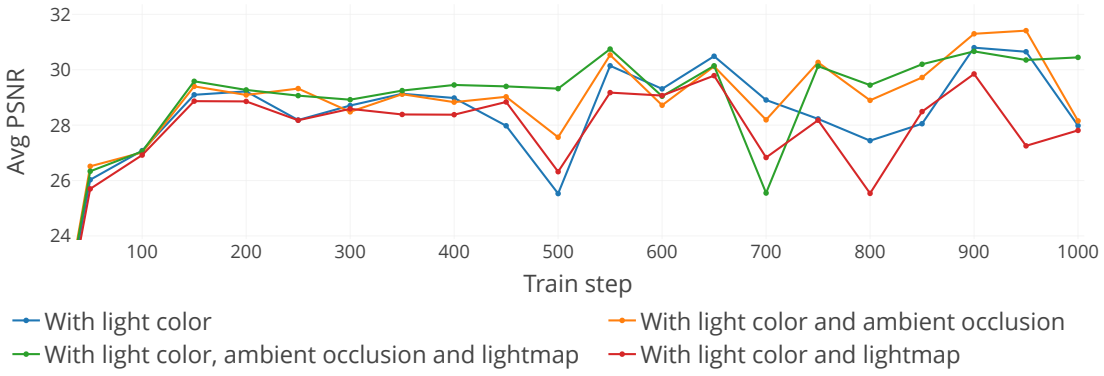

Figure 5.3.: PSNR values during training of the image-based de-lighter model comparing the different input combinations using the lightmap, light color map, and ambient occlusion textures.

progresses, with the PSNR values converging towards a satisfactory level. These findings validate the effectiveness of the image-based approach in removing illumination effects and preserving the underlying color information. While the results of different input texture combinations do not reveal a clear superior option in terms of PSNR values, the model trained with all available input textures (lightmap, light color map, ambient occlusion, and UV texture area map) demonstrates slightly more consistent performance compared to models with alternative input texture combinations. The two variants incorporating all input textures (green) and all input textures except the grayscale lightmap (yellow) demonstrate comparable performance. This similarity suggests that the de-lighter model effectively learns to utilize color lightness information from the Oklab color space in the light color map to correct illumination effects in the color texture. Consequently, the grayscale lightmap becomes less critical when the appropriate color space is employed.

During the development, the inclusion of ambient occlusion information as an additional input texture was a later enhancement to further improve the image-based approach's performance. The benchmarks indicate that incorporating ambient occlusion leads to a modest improvement in de-lighted texture quality. This additional information aids the model in better comprehending spatial relationships and local shading variations, resulting in more precise removal of illumination effects. However, the improvement is relatively minor, suggesting that the model can effectively learn and compensate for most illumination effects even without explicit ambient occlusion data.

Based on these benchmark results and manual observations made during the development process, the image-based model is configured to utilize all available input textures. This includes all supplementary textures (lightmap, light color map, ambient occlusion, and UV texture area map) in addition to the 3D model's color texture with illumination baked in. This chosen input architecture aligns with the description provided in the previous chapter, Section 4.4.

The image-based approach, utilizing a CGAN architecture, has demonstrated several strengths and weaknesses in the context of de-lighting color textures. One of the primary strengths of the image-based approach is its ability to capture and learn global context and spatial relationships within the input textures. By processing the entire image simultaneously, the model can effectively identify and remove illumination effects while preserving the overall structure and coherence of the texture. This is particularly advantageous for textures with complex patterns or fine details, as the model can leverage the contextual information to produce visually consistent results. However, the image-based approach also presents some challenges and limitations. The computational requirements for training are relatively high, as each training step involves processing a full-resolution image. This can lead to longer training times and increased resource utilization compared to pixel-based approaches. Another limitation of the image-based approach is its missing flexibility to input texture resolution. Since the model is trained on a fixed image size, it struggles to handle textures with different resolutions. One potential workaround is to train the model on high-resolution textures, e.g., 8K, and pad smaller textures with black pixels to match the expected input size. However, training on such high resolutions is computationally intensive and time-consuming, making it impractical for the scope of this work.

An alternative attempt to enable flexible input texture resolution involved splitting the input textures into multiple evenly cut smaller images that can then be processed individually by the de-lighter model. For instance, input textures with a resolution of 4096×4096, such as input color, lightmap, light color map, and ambient occlusion, could be individually split into 64 smaller 512×512 textures. These smaller textures can then be individually processed by the de-lighter model which is designed to work with 512×512 input images. The output of the de-lighter model would consist of 64 smaller 512×512 textures that can be reassembled into the original 4096×4096 texture. This approach was tested during the exploratory development phase of the image-based de-lighter model. Figure 5.4 illustrates an example output of the de-lighter model when splitting the input textures into smaller textures and reassembling the output. Although this experimental trial utilized a less developed image-based de-lighter model at the time, the results still provide a good indication of the drawbacks of this approach. The image-based model, comprising its encoder and decoder layers, processes the entire input image simultaneously. This approach can lead to inconsistencies in the output, such as an overall darker appearance when a single smaller input texture contains a darker color tone, while another smaller texture patch may display a brighter overall color tone. This results in visible seams in the reassembled output texture where the individual smaller textures have been stitched together. Further post-processing steps, such as blurring the areas around the seams, could be applied to the reassembled output texture to reduce the visibility of the seams. However, this would not address the underlying issue of varying color tones in the individual smaller patches. Due to the estimated unsatisfactory results of further developing this approach to enable input texture flexibility, this attempt for the image-based model was not investigated further in this work.

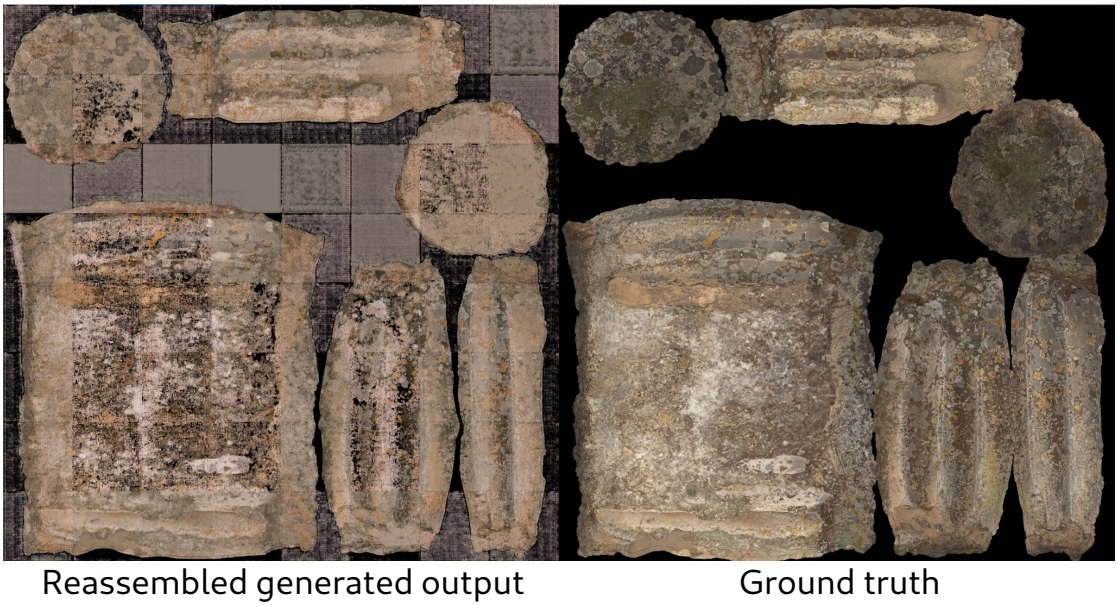

Figure 5.4.: Example output of splitting a bigger texture into smaller textures and reassembling the output of the de-lighter model.

5.3. Pixel-Based Architecture

The pixel-based de-lighter model, as described in the previous chapter in Section 4.5, utilizes only the initial color texture with illumination and the light color map as sources of the single input pixels. While the UV texture area map is also required as input for the de-lighting process to determine which pixels of the input textures to process, it is not provided as specific input data for the de-lighter model itself. Similar to the image-based approach, various possible additional input data besides the color texture have been explored, including the lightmap, the light color map, and the ambient occlusion texture. To evaluate the effectiveness of various combinations of input data for the pixel-based de-lighter model, multiple training runs have been conducted with different input data combinations, computing the averaged PSNR metric of the six 3D model benchmark test cases. The results, presented in Figure 5.5, show a noticeable difference between two groups of input combinations. The first group includes a de-lighter model trained with all possible additional input data, i.e., the light color map, ambient occlusion, and the lightmap data (green), together with another model trained with the light color map and ambient occlusion data (yellow). The second group includes a de-lighter model trained with only the light color map data (blue) and another model trained with the light color map and the lightmap data (red).

The results indicate two key findings. First, the group with more input data performs worse than the group using less input textures, suggesting that simply increasing the input data does not necessarily improve the output result. This leads to the assumption that

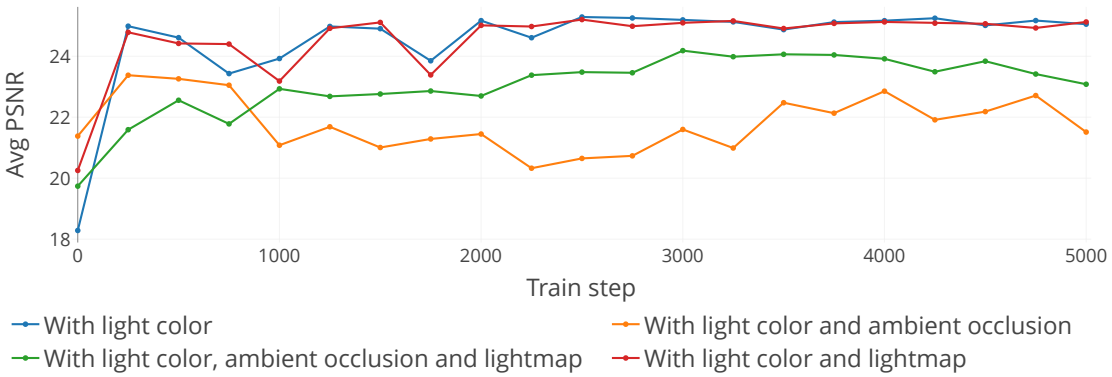

Figure 5.5.: PSNR values during training of the pixel-based de-lighter model comparing the different input combinations using the lightmap, light color map, and ambient occlusion textures.

some input features are more important than others and that the model might not learn as much from the additional input data, or worse, the additional input data may introduce noise to the more important input features, thus reducing the de-lighting performance. Second, the de-lighter model with only the light color map as input performed as well as the model trained with the light color map and the lightmap data combined. This indicates that the additional grayscale lightmap data does not provide more beneficial information to the model for the de-lighting process. This supports the results from the color space comparison in Section 5.1, showing that the Oklab color space used for the light color map color data already includes the light intensity information in its lightness property. Therefore, the light intensity information is no longer needed as an additional lightmap input texture. Based on these observations, the light color map is the only additional input texture used for the pixel-based de-lighter model.

The pixel-based architecture for de-lighting color textures offers several advantages and unique characteristics compared to the image-based approach. One of the key strengths of the pixel-based architecture is its flexibility and efficiency in handling training data. Since the model operates on individual pixels, a single low-resolution texture, e.g., 256×256, can provide a large number of training samples. For instance, a 256×256 texture can yield up to 65,536 training steps, assuming a fully white UV area map, meaning that all pixels of the texture are relevant for the 3D model. This abundance of training data from a single image results in the requirement of only a small training dataset, which also keeps the computational effort for generating this training dataset low. When using low-resolution textures for the training dataset, the generation and processing of a more diverse training dataset, including a greater variety of 3D models with different color schemes in their color texture as well as varying illumination conditions, is more feasible within the computational constraints of this work. Another advantage of the pixel-based approach is its adaptability to different input texture resolutions. As the model processes each pixel independently, it can seamlessly handle textures of various

sizes without the need for resizing or padding. This flexibility enables the model to be applied to a wide range of texture resolutions, from low-resolution textures, e.g., 256×256, to high-resolution textures, e.g., 8192×8192, without any modifications to the model architecture.

5.4. Model Architecture Comparison

In this section, the two proposed de-lighter model architectures, i.e., image-based and pixel-based, for processing color textures of 3D models are compared.

The pixel-based model has several advantages over the image-based approach. Firstly, it requires significantly fewer images for training data, resulting in lower computational effort for creating the training dataset with Blender. This is particularly beneficial when resources are limited or when the dataset needs to be created quickly. Secondly, the pixel-based model can handle different input texture resolutions without the need for resizing or padding. This flexibility allows for the processing of textures with varying dimensions, making it more adaptable to different scenarios. Thirdly, the pixel-based model tends to provide visually better-looking output compared to the image-based approach. The resulting textures appear more detailed and less washed out. However, it is important to note that the image-based approach in this work is currently constrained by computational limitations, restricting its application to textures of lower resolution. While increasing the texture resolution for the image-based architecture mitigates this limitation to some extent, the loss of certain textural details persists due to the downsampling and upsampling processes inherent in the encoder and decoder layers. Figure 5.6 illustrates the

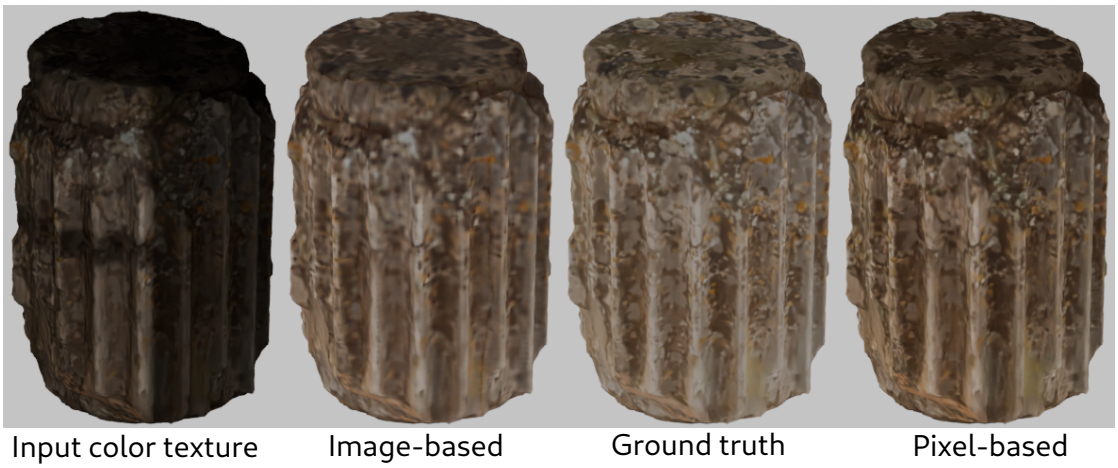

Figure 5.6.: Comparison of de-lighting results using the image-based and pixel-based de-lighter models with textures of 512×512 resolution.

de-lighting results of the image-based and pixel-based de-lighter models when processing a 3D model's color texture with illumination baked in using Blender with a texture resolution

of 512 × 512. To provide a clearer view of the results, the individual color textures have been applied to the material of the 3D model, displaying only the plain diffuse color of the textures without any additional shading or lighting effects. The de-lighted result from the image-based model appears to better preserve the overall color tone of the original texture compared to the pixel-based model. However, the pixel-based model retains more details of the original texture's structure, while the image-based model result displays a more washed-out appearance. Figure 5.7 presents a similar comparison using the same

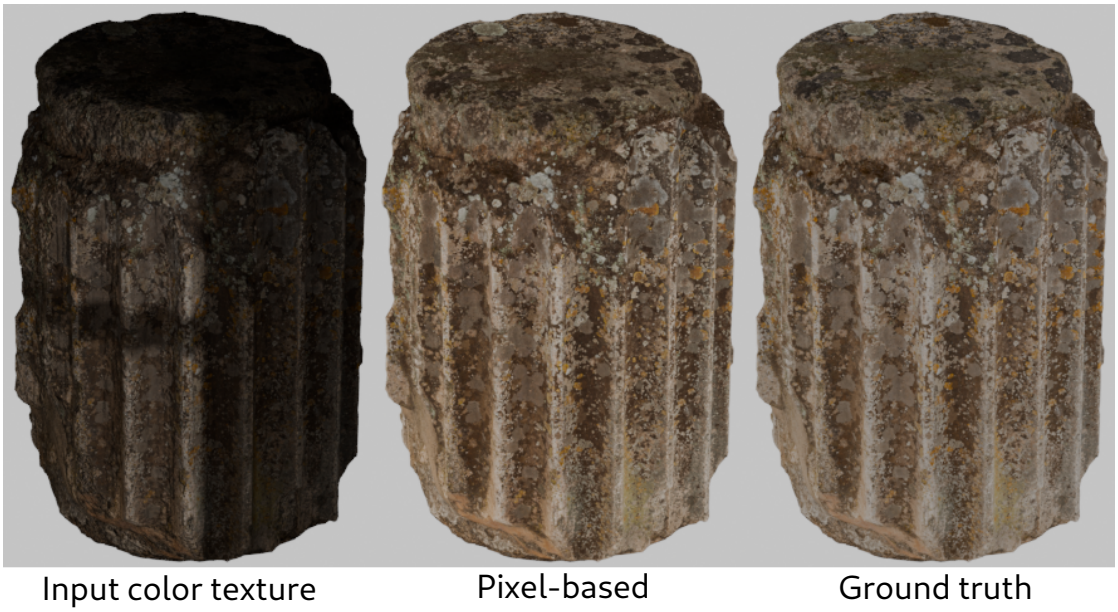

Figure 5.7.: Showing of de-lighting result using the pixel-based de-lighter model with textures of 2048 × 2048 resolution.

3D model and texture but with a higher resolution of 2048 × 2048. This comparison solely showcases the pixel-based de-lighter model's result, as the image-based model is unable to process textures of this resolution due to previously described limitations in the scope of this work. This example with a higher texture resolution demonstrates the pixel-based model's capability to handle higher texture resolutions and successfully maintain the details of the original texture structure, while removing the illumination effects. It is crucial to acknowledge that while this result appears highly promising, it represents a best-case example for the proposed de-lighter models' use case. The given color texture with illumination baked in has been rendered in Blender with the light environment known in detail, ensuring that the supplementary light data input textures, such as the light color map, perfectly match the light conditions of the rendered color texture. Nevertheless, it serves as an example to showcase the potential of the proposed de-lighter models when a precise definition of the light environment is available.

While the training of the image-based architecture requires more data and is more computationally expensive, the de-lighting process itself is faster than the pixel-based approach when processing high-resolution textures. This is because the encoder-decoder architecture of the image-based approach downsamples and upsamples the input textures in their respective encoding and decoding layers, resulting in a faster de-light processing of the given data. In contrast, the pixel-based de-lighter model has to process every single pixel individually, which results in a longer processing time for high-resolution textures. It is worth noting that due to previously stated texture resolution restrictions, the processing times of the image-based approach for high-resolution textures, such as 8K, have not been evaluated in this work and are based on observations made during exploration. As a comparison of the processing times of the de-lighter models, the time taken for the de-lighting process of the 512×512 texture that is shown in Figure 5.6 has been measured. While the pixel-based model required on average 24.5 seconds to process the given 512×512 texture, the image-based model only took 7.3 seconds. It is worth noting that the individual pixels are processed sequentially by the pixel-based de-lighter model. The individual processing of the pixels lends itself to parallelized processing. This could increase the processing speed of the pixel-based model to more closely match the processing times of the image-based model. Since the focus of this work is on achieving satisfactory results in terms of good images and not on low processing times, the parallelization of pixel processing with the pixel-based model using multiple threads was not prioritized and stands as potential performance improvement. Another advantage of the image-based approach is its ability to include the surrounding pixels or whole patches of the input texture in the de-lighting process. This allows for the possibility of gaining additional insight into overall color patterns in a color texture. The pixel-based approach, on the other hand, only processes single pixels of the input textures and does not consider the context of the surrounding pixels.

In summary, both the pixel-based and image-based approaches have their strengths and weaknesses. The pixel-based model requires less training data, can handle different texture resolutions, and provides visually more detailed output. However, the image-based approach can be faster for processing high-resolution textures and can consider the context of surrounding pixels, potentially providing additional insights into color patterns. In the environment of this work, the pixel-based approach has been preferred due to its advantages in handling varying and higher input texture resolutions, and therefore providing visually better output results.

5.5. Light Environment Estimation

In the process of de-lighting a 3D model, one of the initial and crucial steps involves the estimation and reconstruction of the light environment present during the object's digitization. This step is fundamental for generating the supplementary light data input textures for the de-lighter model. The procedure typically begins with loading the 3D model into an empty Blender scene, followed by the addition of light sources that aim to replicate the original illumination conditions. In ideal scenarios, additional data regarding

the scene illumination, such as light probe captures of the environment and ambient light, or specifications of artificial light sources (e.g., LED light box positions and colors), would be available. However, in many cases, the color texture of the 3D model itself, which inherently contains baked-in illumination information, serves as the sole source of lighting data. The task of reconstructing the light setup in the scene involves estimating several key parameters of the light sources, including their number, position or direction, intensity, and color. For objects digitized under simple lighting conditions, such as a clear sunny day with the sun as the predominant light source casting distinct shadows with sharp edges, the estimation process can be initiated based on the shadows baked into the model's color texture. By introducing a sunlight source in the Blender scene and iteratively adjusting its properties to match the observed shadows, an approximation of the light environment can be achieved. However, the complexity of this estimation process increases significantly in more complex lighting scenarios. Environments with multiple light sources, each possessing unique properties, present a considerable challenge and may necessitate additional information about the light setup for accurate reconstruction.

To enhance the accuracy of the manual light environment estimation process, an estimation optimization method has been developed. This approach takes an initial manual estimation as input and tries to iteratively refine it to more closely approximate the actual light conditions present during the object's digitization. The optimization process focuses on estimating the properties of light sources, including their position or direction, intensity, and color. The optimization procedure operates iteratively, alternating adjustments to individual light source properties and comparing the resulting light environment with that baked into the 3D model's color texture. A crucial aspect of this process is the definition of a key metric that quantifies the difference between the estimated light scene and the actual light environment. This metric serves as a cost function to be minimized during the optimization process.

The optimization is structured in multiple stages. Initially, the position (or direction) and intensity of the light source are estimated. To establish a cost function for this process, the sRGB color data from the 3D model's color texture is converted into a color space that separates brightness from hue values. Based on the comparative analysis of color spaces presented earlier, the Oklab color space has been selected as the most suitable option for this application. In each iteration of the optimization process, a lightmap is generated, representing the intensity of light on each pixel of the model's color texture using grayscale values. This lightmap is then compared to the lightness value of corresponding pixels in the model's color texture, yielding a difference metric between the estimated light environment and the original baked-in lighting. This difference serves as the cost function to be minimized. To optimize the light source properties, three different optimization methods have been explored: Nelder-Mead, Particle Swarm, and Simulated Annealing. Each of these methods were evaluated for its effectiveness in estimating light source parameters. Figure 5.8 provides a comparison of the performance of various optimization methods for lightmap estimation. The graph depicts the temporal evolution of each method's performance, with the x-axis representing elapsed time in minutes. It is worth noting that the horizontal axis measures elapsed time rather than optimization

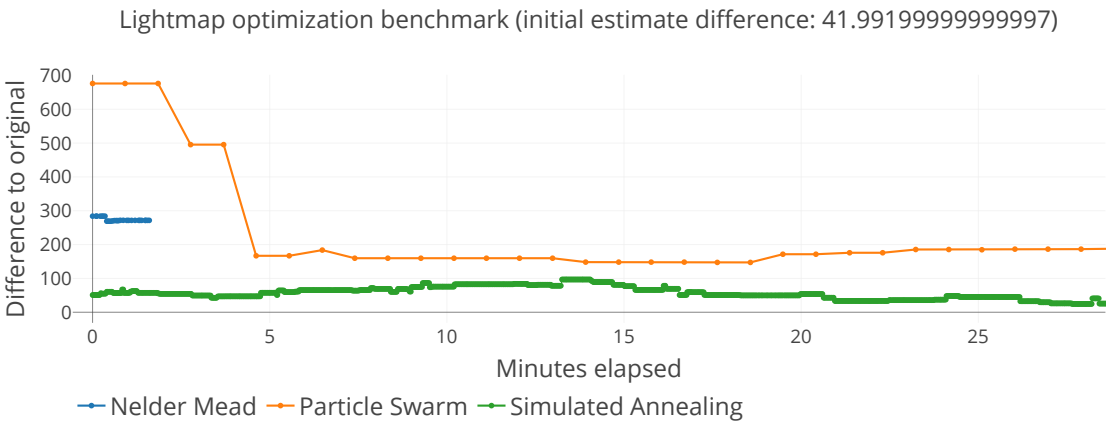

Figure 5.8.: Comparing light map estimation results using different optimization methods.

iteration count. This choice is motivated by the fact that individual iteration steps can exhibit varying computational effort and duration across different optimization methods. Consequently, iteration count is not a reliable metric for comparing the "effort required" by each optimization method. By using elapsed time, the computational efficiency of each method can be assessed, providing a more meaningful comparison of their performance. The vertical axis represents the performance of the resulting estimated light environment configuration, calculated as the absolute difference between the properties of the original light environment configuration and the estimated one. For a sunlight source, the configuration difference refers to the difference between the light's direction and strength values. This metric was chosen for the evaluation to provide a more intuitive comparison of the resulting light configurations. However, it is essential to note that the optimization methods still rely on the cost function based on the difference between the estimated lightmap and the lightness values of the color texture, as previously described.

The presented data shows that the Simulated Annealing method returns the most rapid convergence and achieves the lowest cost value, resulting in a configuration difference of 35.93. The Particle Swarm method converges to its best result in an adequate duration of time, but with higher and therefore worse difference values, yielding a configuration difference of 188.83. In contrast, the Nelder-Mead method starts with better results than the Particle Swarm method but plateaus at a higher difference value at a relatively early iteration stage, resulting in the worst results of the three optimization methods, with a configuration difference of 271.57. Given that the initial manual light environment configuration estimation resulted in a configuration difference of approximately 42, the Simulated Annealing method is the only optimization method that achieves a further improvement over the initial estimation in this evaluation test run. Following the estimation of the light source's position (or direction) and intensity, a similar process is employed to estimate the light color. In this phase, instead of relying solely on lightness values, the color hue properties, specifically the "a" and "b" values of the Oklab color

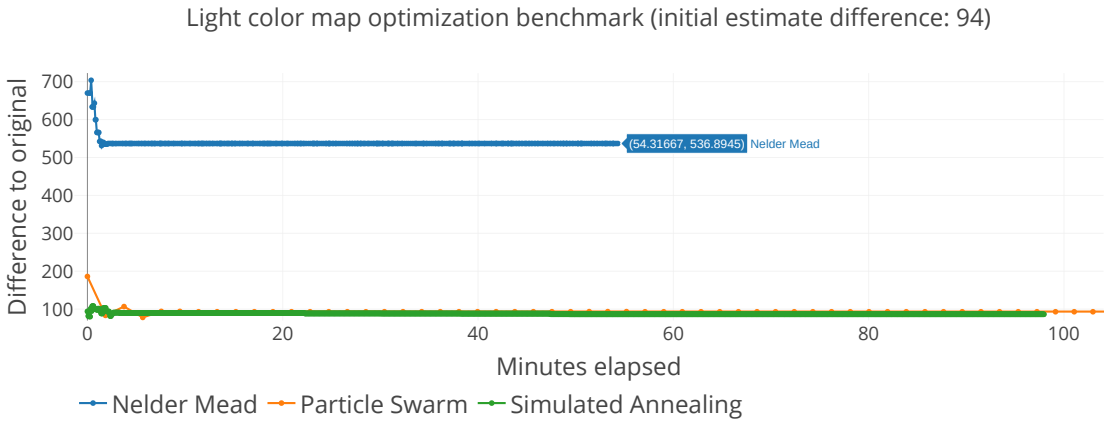

Figure 5.9.: Comparing light color map estimation results using different optimization methods.

space, are considered. The light is baked into the input color texture that already includes the original illumination. The cost function for this stage of optimization is based on the hue difference between the original and newly baked color textures. Figures 5.9 and 5.10 present the results of the light color map estimation process. Figure 5.9 provides an overview of all three optimization methods, while Figure 5.10 offers a zoomed-in comparison of the Simulated Annealing and Particle Swarm methods. The data in these figures corroborate the findings from the lightmap optimization. Simulated Annealing consistently outperforms the other methods, achieving the lowest difference value of 86.39 and exhibiting the most stable convergence, while Particle Swarm optimization shows competitive performance with the lowest difference value of 93.07. The Nelder-Mead method, while showing initial rapid improvement, fails to achieve the low difference values of the other two methods, plateauing at a difference value of 536.89. With the initial manual light color estimation resulting in a difference value of 94, the Simulated Annealing method still provides some improvement, while the Particle Swarm method results in a rather negligible improvement over the initial estimation. In contrast, the Nelder-Mead method fails to achieve a satisfactory result. The characteristic of Simulated Annealing to escape local optima through its probabilistic acceptance of worse solutions, which allows it to explore the solution space more thoroughly, is expected to contribute to its good performance in the task of light environment estimation.

Overall, the presented estimation optimization process can be used to improve already manually estimated light environments in Blender. Depending on the accuracy of the initial manual estimation, the optimization process can further refine the light source's properties to some extent. Upon completion of the optimization process, the approximated light scene in Blender can be utilized to generate the required light data textures, namely the lightmap and light color map, for subsequent processing by the de-lighter model. It is important to note that the current implementation of this estimation process has

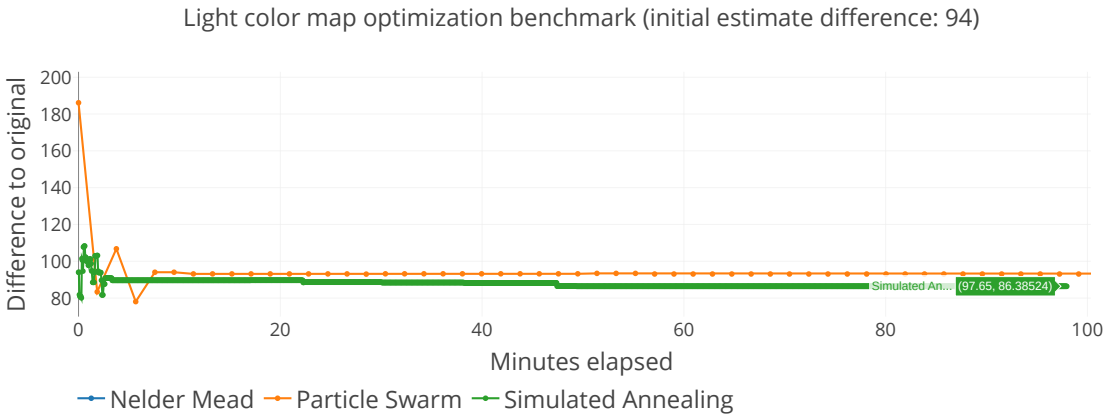

Figure 5.10.: Comparing light map estimation results zoomed to compare the results using simulated annealing and particle swarm methods.

been constrained to simpler light environments, such as those dominated by a single primary light source (e.g., the sun on a clear day) with distinct shadow patterns. This limitation was imposed to maintain a manageable scope for the present work. Although the developed methodology demonstrates possible improvements for simple lighting scenarios, its accuracy, and therefore usability, for more complex light environments is expected to be limited due to the complexity of extracting light environment configuration properties from the color texture as the sole source of information.

5.6. Limitations

The proposed de-lighting methods for color textures of 3D models have several limitations that need to be considered. The main development of the varying prototypes have been done in a controlled environment using Blender for the creation of training as well as test and evaluation datasets. Therefore, existing 3D models have been used to bake effects of light in their previously illumination-free color textures to obtain images resembling the case as if a digital 3D model would be obtained from a process such as 3D scanning of a real-world object. This approach was beneficial in terms of having a controlled environment for the creation of a larger and standardized dataset to be processed by the de-lighter models. However, this approach also results in the given 3D geometry data that can be used for further processing, like the generation of the additional light data textures, i.e., grayscale lightmap, light color map, or ambient occlusion, being the exact same 3D geometry data that existed during the time of the baking of the light effects into the color texture.

In a scenario where a real-world object is scanned and the obtained 3D model is used for the de-lighting process, the dataset that has been obtained from the previously described controlled environment would require that the geometry data of the digitized

3D model and its real-world equivalent have to be exactly the same to be able to generate supplementary light data textures that match the illumination effects that have been baked into the color texture during the production of the digitized 3D model. This requirement is difficult to achieve in practice, as the correct digitization of real-world objects and their accurate representation of 3D geometry is a challenging task and is not easily done with perfect details. There are multiple reasons for this. Difficult lighting conditions could, for example, result in dark surfaces of the real-world object, which may lead to reduced details in the generated digitized representation. Small and complex structures of a real-world object are also difficult to capture with technologies such as photogrammetry. Given the example of a tree with many small branches and leaves, it can be a very tedious or even unfeasible task to capture every single detail of the tree by capturing images of every single branch and leaf from every required angle. Additionally, structures that are not completely static and potentially move during the capturing process can also impose a challenge to correctly digitize the real-world object. These potential challenges in the digitization process of real-world objects result in 3D models with limited accuracy in their geometry data, leading to virtual 3D models that do not resemble the real-world object with 100 % fidelity. In the example of a tree, this can include missing branches or leaves, or additional geometry that did not exist in the real world as artifacts of the digitization process. This discrepancy between the real-world object and the digitized 3D model therefore also results in a discrepancy between the light data textures that can be additionally generated with the available geometry of the digitized 3D model and the illumination effects from sunlight or other light sources that existed in the real world during the time of the scan of the real-world object. This disparity in available data in the color texture containing illumination effects of a real-world object and the generated light data textures leads to mismatching input data for the de-lighter model and therefore to a reduced performance of the de-lighter model in removing the illumination effects from the color texture. This could result in shadows not being removed by the de-lighter model since the generated light data textures do not include the shadow information that was present in the real-world scene, or conversely, shadows being removed from the color texture that were not present in the real-world scene. An example of this limitation can be observed in the digitization of the Krzyztopor Castle [61], a historic site with many details. An overview of the complete 3D model can be seen in Figure 5.11. Due to the complexity and size of the castle's structure and its surrounding area, and the limitations of the digitization process, in this case photogrammetry, the resulting 3D model lacked fine details present in the real-world object. It is important to acknowledge that the provided examples of limitations in detail and geometric inconsistencies are not intended as a critique of the specific model's quality. Instead, they serve to highlight the challenges and constraints associated with the digitization process as a whole. The complexity of the task, coupled with the limitations of current technologies, inevitably leads to some degree of detail loss and geometric approximations in the final digital representation. Figure 5.12 shows a tree within the digitized 3D model of Krzyztopor Castle. Due to the extensive size and coverage area of the complete model, the fine details of the individual tree branches are not fully captured in this representation.

Figure 5.11.: A 3D model of the Krzyztopor Castle obtained via photogrammetry [61].

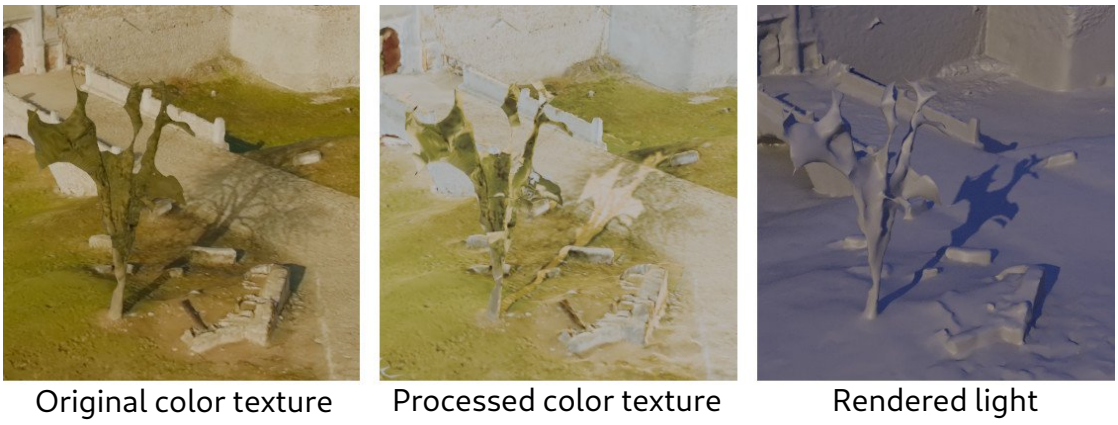

Figure 5.12.: A detailed view of a tree object in the Krzyztopor Castle model and its processed result.

However, the precise shadow cast by the tree is baked into the color texture. The absence of comprehensive data in the 3D geometry results in a less detailed shadow of the tree within the Blender light scene. Consequently, the de-lighter model, lacking the necessary information in the light data textures, is unable to accurately remove the tree's shadow from the color texture. Figure 5.13 illustrates another example, displaying a grass-covered

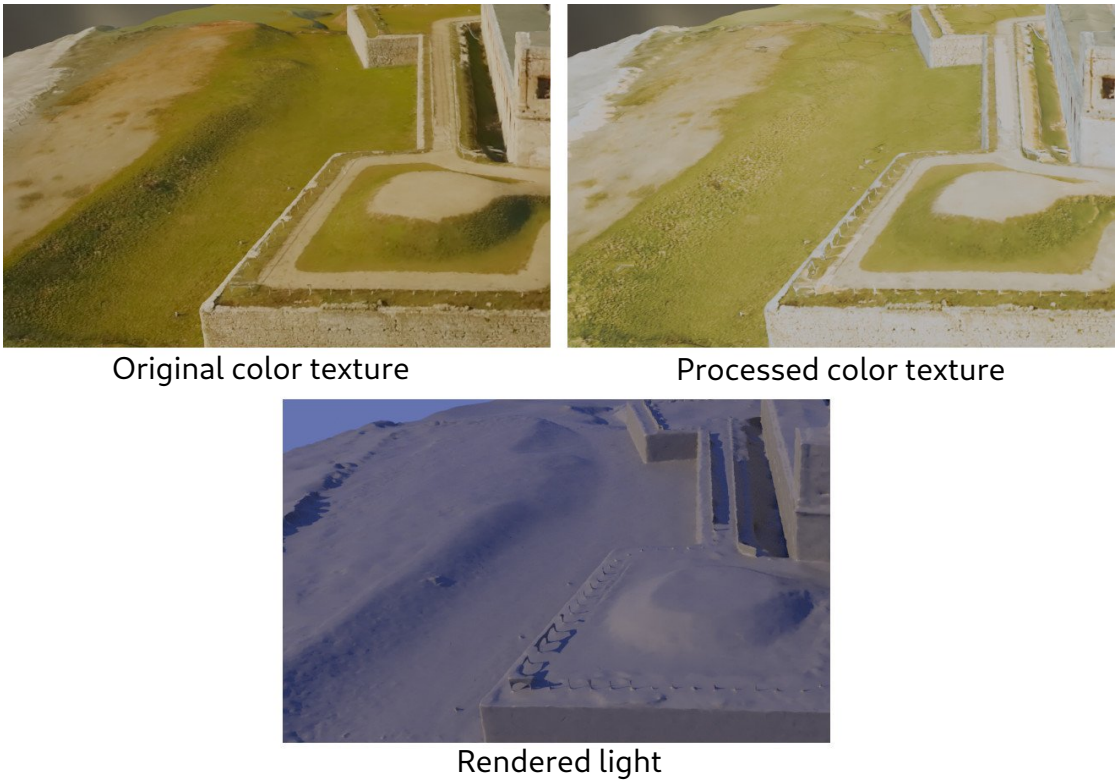

Figure 5.13.: A detailed view of a grass region in the Krzyztopor Castle model and its processed result.

region in the digitized 3D model of Krzyztopor Castle. The grass-covered slope on the left, which is casting a shadow, is nicely captured in the geometry data and thus can be processed with good results. However, the de-lighting process does not yield entirely error-free results for the fine details of the grass, such as small bumps, or the thin railings on the stone wall in the bottom center of the image, as these elements are not fully captured in the 3D model. Considering the overall context and scale of the model, the absence of precise details in smaller objects is less noticeable and does not significantly detract from the model's intended purpose. Rather than emphasizing the fine details of individual small objects, the model aims to provide a holistic representation of the site.

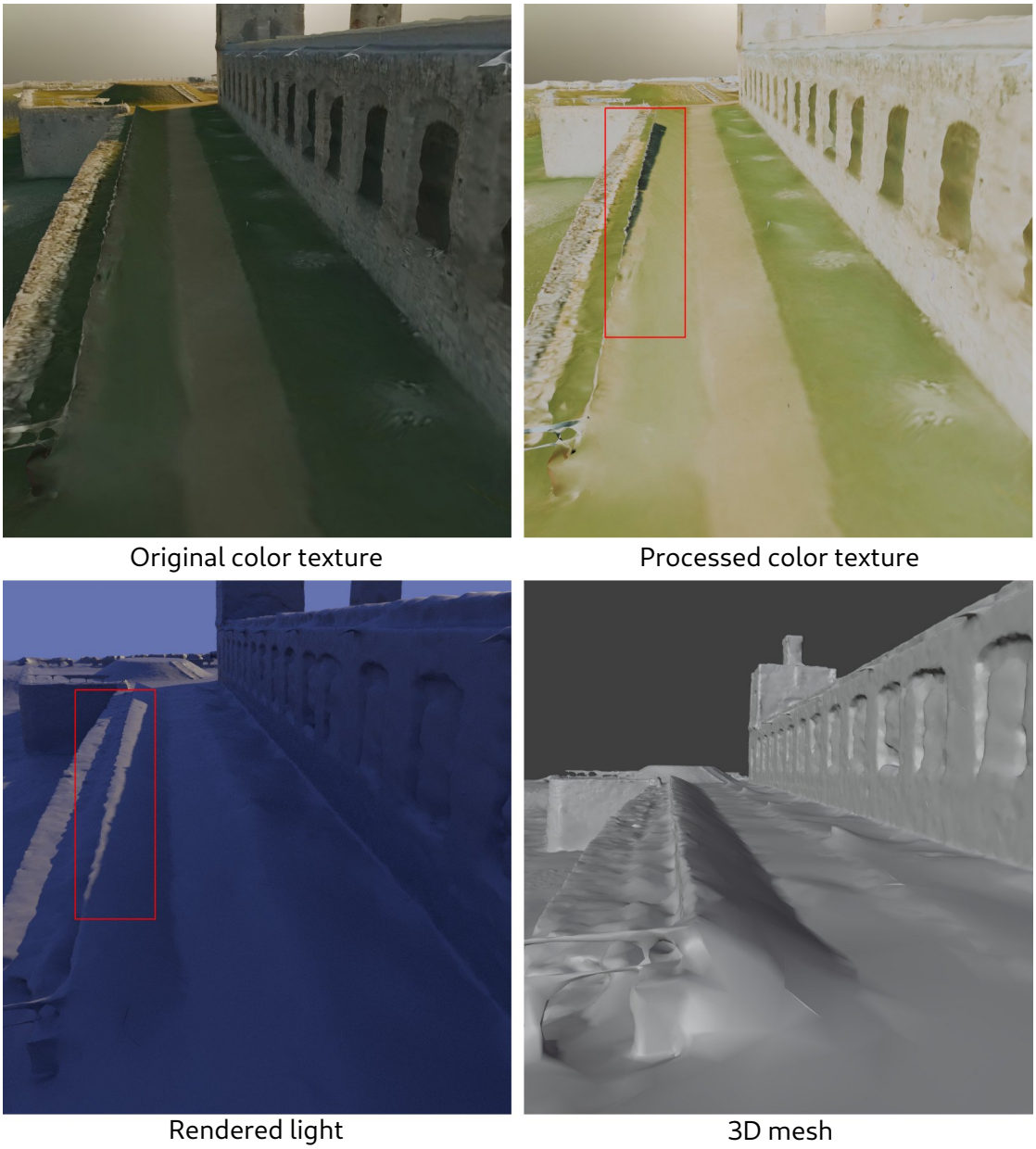

Figure 5.14.: A detailed view of a railing geometry that includes errors in the digitized geometry leading to de-lighting errors in the Krzyztopor Castle model.

Incorrectly added geometry data that introduces additional volume, potentially casting unwanted shadows, is another source of de-lighting errors, complementing the issue of missing geometry that fails to cast required shadows for light data texture rendering, as described in the previous examples. This problem is illustrated in Figure 5.14, specifically on the right side of the railing. In the area highlighted in red, the captured model's geometry data includes an additional volume that does not correspond to the real-world object, resulting in an inaccurately rendered light environment in Blender. A portion of this added volume extends beyond the shadow region cast by the roof of the building on the right, exposing it to sunlight in the scene. This exposure is subsequently baked into the light data textures as a region with high light intensity, which the de-lighter model must adjust by darkening. As a result, the de-lighter model incorrectly adjusts the brightness of this part of the color texture, producing an undesirable dark region in the de-lighter model's output.

These described cases of geometry data inaccuracy lead to discrepancies between the generated light data textures and the actual illumination effects captured in the color texture of the digitized model, reducing the overall quality of the de-lighter model's output. This shows that the quality of the processed output texture also relies on the quality of the geometry data of the 3D model that is processed.

Another limitation of the proposed de-lighting methods is the reliance on the generation of the light data textures via the physically-based path tracer engine Cycles of Blender. While this technology is very capable of simulating realistic-looking light effects, it is not 100 % physically accurate and therefore cannot perfectly resemble the light effects of the real world given the limited data provided. In combination with possibly very complex real-world scenes, the imitation of the given scene illumination in Blender can be very complex and difficult to achieve, which is required to generate the light data textures that are used as input data for the de-lighter models. As a consequence, this work has been focusing on the processing of 3D models with a relatively simple light setup, e.g., a sun as the primary light source, to ensure relatively straightforward illumination effects in the textures, such as strong shadows with a clear edge as in the case of a sunny day with a clear sky. A scene with cloudy daylight or multiple light sources with diffuse light leads to more complexity in the estimation and imitation, and therefore requires more effort to generate the light data textures accurately.

In addition to the challenges posed by complex lighting environments, the current de-lighting methods that have been evaluated in this work also have been reduced to rather simple 3D model surface properties such as the diffuse color or roughness property as these are also more commonly found with digitized virtual 3D models. Other surface material properties such as translucency or highly reflective surfaces have not been evaluated in depth as the material data is also not commonly provided in the 3D models that have been obtained via 3D laser scanners or photogrammetry techniques for example. These material properties can significantly influence how light interacts with the object and how it is captured in the color texture. Accurately modeling and separating these effects from the underlying diffuse color of the object represents a challenge that has not been addressed in this work.

5.7. Further Research

The proposed methodology relies on generating additional light data textures, such as lightmaps or light color maps, as input for the de-lighter models. Consequently, the quality of the processed output is highly dependent on the accuracy of these light data textures. Estimating the light source or multiple light sources in the scene during the digitization of the real-world object is challenging, given only the 3D geometry data and the color texture with illumination effects. As a result, the accuracy of the light configuration estimation is limited. Currently, the methodology is constrained to relatively simple lighting scenarios, such as a clear sunny day with the sun as the predominant light source or a single point light source. To address more complex lighting setups and enhance the robustness of the light environment estimation technique, further research and development will be essential. Future enhancements may include implementing machine learning algorithms to more accurately estimate complex light behavior and interactions in environments with multiple light sources. Additionally, integrating supplementary data capture methods during the digitization process, such as the utilization of light probes, could enhance the accuracy of light environment estimation.

The digitized object's surface properties can influence how light interacts with the object and how it is captured in the color texture. Many digitization techniques primarily include captured 3D geometry data and a color texture, but often lack information about surface material properties such as specularity or transparency. This limitation leads to less accurate rendering in Blender for the generation of supplementary light data textures. Furthermore, inaccuracies in the 3D geometry data obtained from scanning real-world objects can result in imprecise light data textures. To improve the quality of the de-lighting process, it is beneficial to enable better imitation of the real-world light environment and its interaction with the object. This can be achieved by providing more details about the surface material or more accurate 3D geometry data, which in turn helps generate more precise light data textures.

In case of the specific limitation of a maximum texture resolution to be processed by the image-based de-lighter model, further development could focus on utilizing more hardware resources to enable processing of higher texture resolutions. This includes more computational resources for generating high-resolution training images in a reasonable time, as well as increased memory capacity, particularly VRAM for GPU-based execution, to accommodate larger high-resolution textures and model data such as weights.

6. Conclusion

This thesis has presented methods for de-lighting color textures of 3D models to remove illumination effects like shadows, highlights and color tints that are baked into the textures during the digitization process of real-world objects. Two main approaches have been developed and evaluated: an image-based method using a CGAN architecture, and a pixel-based method using a simpler neural network architecture that processes the input textures on a per-pixel basis.

Extensive experiments were conducted to determine the optimal configuration and input data for each approach. The Oklab color space was found to provide the best results for representing the color data for the given task. For the image-based CGAN approach, using all available supplementary textures including a lightmap, light color map, ambient occlusion map and UV texture area map yielded the highest quality output. The pixel-based approach performed best using just the light color map as additional input.

Comparing the two approaches, the pixel-based method demonstrated several advantages. It requires significantly less training data, can flexibly handle different input texture resolutions, and produces visually superior results with more preserved detail. The main drawbacks are slower processing of high resolution textures compared to the image-based approach as well as its inability to recognize and capture global structure and color tone present in the complete color texture.

A light estimation optimization process was also developed to automatically refine a manually provided initial guess of the lighting environment. This improves recreation of the lighting in Blender to generate the supplementary input textures. The Simulated Annealing optimization method was found most effective for this task.

The proposed methods were able to successfully remove illumination effects and shadows from color textures in a controlled test environment. However, some limitations remain when applying the techniques to real-world 3D scans. Inaccuracies and missing data in the captured 3D geometry can lead to discrepancies between the actual illumination in the color texture and the lighting information in the generated supplementary input textures. This reduces the quality of the de-lighting result. Complex real-world lighting environments are also challenging to estimate and simulate in Blender.

In summary, this thesis has demonstrated the feasibility and potential of neural network based techniques for automatically de-lighting color textures of 3D models. The developed methods can already handle simpler controlled cases well. To be robustly applicable to arbitrary real-world scans, future work should focus on improving geometry capture to provide complete, detailed 3D surfaces, and investigate enhanced techniques for estimating and simulating complex lighting environments. With such improvements, the proposed de-lighting approaches can be a valuable tool to enhance 3D digitization pipelines.

Bibliography

[1] Agisoft. *Agisoft Metashape: Agisoft Metashape*. URL: https://www.agisoft.com/. (accessed: 27.11.2023).

[2] Agisoft. *Agisoft Texture De-Lighter - general workflow*. URL: https://agisoft.freshdesk.com/support/solutions/articles/31000158376-agisoft-texture-de-lighter-general-workflow?id=71. (last modified: 17.02.2022, accessed: 27.11.2023).

[3] Agisoft. *Yogini with a Jar, early 10th century*. URL: https://sketchfab.com/3d-models/yogini-with-a-jar-de-lighter-30705a28827344138bcddfc530262ef9. (accessed: 27.11.2023).

[4] Irene Aicardi et al. 'Recent trends in cultural heritage 3D survey: The photogrammetric computer vision approach'. In: *Journal of Cultural Heritage* 32 (2018), pp. 257–266. ISSN: 1296-2074. DOI: https://doi.org/10.1016/j.culher.2017.11.006. URL: https://www.sciencedirect.com/science/article/pii/S129620741630423X.

[5] Bashar Alsadik. 'Practicing the geometric designation of sensor networks using the Crowdsource 3D models of cultural heritage objects'. In: *Journal of Cultural Heritage* 31 (2018), pp. 202–207. ISSN: 1296-2074. DOI: https://doi.org/10.1016/j.culher.2017.11.001. URL: https://www.sciencedirect.com/science/article/pii/S1296207417303850.

[6] Matthew Anderson et al. 'Proposal for a Standard Default Color Space for the Internet - sRGB'. In: *International Conference on Communications in Computing*. 1996. URL: https://api.semanticscholar.org/CorpusID:6586058.

[7] Eli Arbel and Hagit Hel-Or. 'Texture-Preserving Shadow Removal in Color Images Containing Curved Surfaces'. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383081.

[8] I. Gede Puja Astawa et al. 'The Impact of Color Space and Intensity Normalization to Face Detection Performance'. In: *TELKOMNIKA Telecommunication Computing Electronics and Control* 15 (2017), pp. 1895–1900. URL: https://api.semanticscholar.org/CorpusID:54604464.

[9] Fabio Bettio et al. 'Improving the digitization of shape and color of 3D artworks in a cluttered environment'. In: *2013 Digital Heritage International Congress (DigitalHeritage)*. Vol. 1. 2013, pp. 23–30. DOI: 10.1109/DigitalHeritage.2013.6743709.

[10] Alexey Bokhovkin, Shubham Tulsiani and Angela Dai. *Mesh2Tex: Generating Mesh Textures from Image Queries*. 2023. arXiv: 2304.05868 [cs.CV]. URL: https://arxiv.org/abs/2304.05868.

[11] Dave Zhenyu Chen et al. *Text2Tex: Text-driven Texture Synthesis via Diffusion Models*. 2023. arXiv: 2303.11396 [cs.CV]. URL: https://arxiv.org/abs/2303.11396.

[12] Zhaolin Chen et al. '3D Texture Mapping in Multi-view Reconstruction'. In: *Advances in Visual Computing*. Ed. by George Bebis et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 359–371. ISBN: 978-3-642-33179-4.

[13] Shu-Kam Chow and Kwok-Leung Chan. 'Removal of Specular Reflection Component Using Multi-view Images and 3D Object Model'. In: *Advances in Image and Video Technology*. Ed. by Toshikazu Wada, Fay Huang and Stephen Lin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 999–1009. ISBN: 978-3-540-92957-4.

[14] Rico Cilliers. *Marble Bust 01*. URL: https://polyhaven.com/a/marble_bust_01. (accessed: 05.06.2024).

[15] Paolo Dabove, Nives Grasso and Marco Piras. 'Smartphone-Based Photogrammetry for the 3D Modeling of a Geomorphological Structure'. In: *Applied Sciences* 9.18 (2019). ISSN: 2076-3417. DOI: 10.3390/app9183884. URL: https://www.mdpi.com/2076-3417/9/18/3884.

[16] Arpita Dawda, Minh Nguyen and Reinhard Klette. 'Accurate 3D Measurement of Highly Specular Surface using Laser and Stereo Reconstruction'. In: *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. 2019, pp. 1–6. DOI: 10.1109/IVCNZ48456.2019.8960991.

[17] Lucio Tommaso De Paolis et al. 'Photogrammetric 3D Reconstruction of Small Objects for a Real-Time Fruition'. In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis and Patrick Bourdot. Cham: Springer International Publishing, 2020, pp. 375–394. ISBN: 978-3-030-58465-8.

[18] Paul Debevec. 'Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography'. In: *ACM SIGGRAPH 2008 Classes*. SIGGRAPH '08. Los Angeles, California: Association for Computing Machinery, 2008. ISBN: 9781450378451. DOI: 10.1145/1401132.1401175. URL: https://doi.org/10.1145/1401132.1401175.

[19] M. Dellepiane et al. 'Improved color acquisition and mapping on 3D models via flash-based photography'. In: *J. Comput. Cult. Herit.* 2.4 (Mar. 2010). ISSN: 1556-4673. DOI: 10.1145/1709091.1709092. URL: https://doi.org/10.1145/1709091.1709092.

[20] Mark S. Drew et al. 'Robust estimation of surface properties and interpolation of shadow/specularity components'. In: *Image and Vision Computing* 30.4 (2012), pp. 317–331. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2012.02.012. URL: https://www.sciencedirect.com/science/article/pii/S0262885612000273.

[21] Rui Fan et al. 'Computer Stereo Vision for Autonomous Driving'. In: *CoRR* abs/2012.03194 (2020). arXiv: 2012.03194. URL: https://arxiv.org/abs/2012.03194.

[22] Raanan Fattal, Maneesh Agrawala and Szymon Rusinkiewicz. 'Multiscale shape and detail enhancement from multi-light image collections'. In: *ACM Trans. Graph.* 26.3 (July 2007), 51–es. ISSN: 0730-0301. DOI: 10.1145/1276377.1276441. URL: https://doi.org/10.1145/1276377.1276441.

[23] Blender Foundation. *Blender 4.1 Python API Documentation — Blender Python API*. URL: https://docs.blender.org/api/current/index.html. (accessed: 02.04.2024).

[24] Blender Foundation. *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. URL: https://www.blender.org/. (accessed: 05.06.2024).

[25] Fovea. *Monkeys figure 3d scan*. URL: https://sketchfab.com/3d-models/monkeys-figure-3d-scan-80defc87f54e41869f76f8dd6bed99b2?utm_medium=embed&utm_campaign=share-popup&utm_content=80defc87f54e41869f76f8dd6bed99b2. (accessed: 27.11.2023).

[26] Diego García-Molina et al. 'Digitalization and 3D Documentation Techniques Applied to Two Pieces of Visigothic Sculptural Heritage in Merida Through Structured Light Scanning'. In: *Journal on Computing and Cultural Heritage* 14 (Aug. 2021). DOI: 10.1145/3427381.

[27] Jason Geng. 'Structured-light 3D surface imaging: a tutorial'. In: *Adv. Opt. Photon.* 3.2 (June 2011), pp. 128–160. DOI: 10.1364/AOP.3.000128. URL: https://opg.optica.org/aop/abstract.cfm?URI=aop-3-2-128.

[28] Svetlana Golubeva. *What is laser 3D scanning?* Jan. 2022. URL: https://www.artec3d.com/learning-center/laser-3d-scanning. (accessed: 25.08.2024).

[29] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press, 2004, pp. 310–324.

[30] Poly Haven. *HDRIs • Poly Haven*. URL: https://polyhaven.com/hdris. (accessed: 05.06.2024).

[31] Poly Haven. *Models • Poly Haven*. URL: https://polyhaven.com/models. (accessed: 05.06.2024).

[32] Kai He et al. 'Single-Image Shadow Removal Using 3D Intensity Surface Modeling'. In: *IEEE Transactions on Image Processing* 26.12 (2017), pp. 6046–6060. DOI: 10.1109/TIP.2017.2751142.

[33] Yannick Hold-Geoffroy et al. 'Deep Outdoor Illumination Estimation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[34] Andrew Hou et al. *Towards High Fidelity Face Relighting with Realistic Shadows*. 2021. arXiv: 2104.00825 [cs.CV].

[35] Jakob Iglhaut et al. 'Structure from Motion Photogrammetry in Forestry: a Review'. In: *Current Forestry Reports* 5 (2019). DOI: 10.1007/s40725-019-00094-3. URL: https://doi.org/10.1007/s40725-019-00094-3.

[36] L. Inzerillo, F. Di Paola and Y. Alogna. 'High Quality Texture Mapping Process Aimed at the Optimization of 3D Structured Light Models'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W9 (2019), pp. 389–396. DOI: 10.5194/isprs-archives-XLII-2-W9-389-2019. URL: https://isprs-archives.copernicus.org/articles/XLII-2-W9/389/2019/.

[37] C. Ioannidis et al. 'Laser and Multi-Image Reverse Engineering Systems for Accurate 3D Modelling of Complex Cultural Artefacts'. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W11 (2019), pp. 623–629. DOI: 10.5194/isprs-archives-XLII-2-W11-623-2019. URL: https://isprs-archives.copernicus.org/articles/XLII-2-W11/623/2019/.

[38] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[39] Chaonan Ji et al. 'Geometry-Aware Single-Image Full-Body Human Relighting'. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 388–405. ISBN: 978-3-031-19787-1.

[40] George H. Joblove and Donald Greenberg. 'Color spaces for computer graphics'. In: *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '78. New York, NY, USA: Association for Computing Machinery, 1978, pp. 20–25. ISBN: 9781450379083. DOI: 10.1145/800248.807362. URL: https://doi.org/10.1145/800248.807362.

[41] Antreas Kantaros, Theodore Ganetsos and Florian Ion Tiberiu Petrescu. 'Three-Dimensional Printing and 3D Scanning: Emerging Technologies Exhibiting High Potential in the Field of Cultural Heritage'. In: *Applied Sciences* 13.8 (2023). ISSN: 2076-3417. DOI: 10.3390/app13084777. URL: https://www.mdpi.com/2076-3417/13/8/4777.

[42] Jean-François Lalonde, Alexei A. Efros and Srinivasa G. Narasimhan. 'Estimating the Natural Illumination Conditions from a Single Outdoor Image'. In: *International Journal of Computer Vision* 98 (2012), pp. 123–145. DOI: doi:10.1007/s11263-011-0501-8. URL: https://doi.org/10.1007/s11263-011-0501-8.

[43] Gustaw Mackay. *The Beauty of Palmyra (200-250 AD)*. URL: https://sketchfab.com/3d-models/the-beauty-of-palmyra-200-250-ad-1c1d6311981542289de7a6cc49bcab57?utm_medium=embed&utm_campaign=share-popup&utm_content=1c1d6311981542289de7a6cc49bcab57. (accessed: 27.11.2023).

[44] David Mandl et al. 'Learning Lightprobes for Mixed Reality Illumination'. In: *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2017, pp. 82–89. DOI: 10.1109/ISMAR.2017.25.

[45] Matthew McMillion. *What is a CMM machine?* Apr. 2021. URL: https://www.artec3d.com/learning-center/what-is-cmm-machine. (accessed: 25.08.2024).

[46] Pilar Merchán et al. 'Geometric and Colour Data Fusion for Outdoor 3D Models'. In: *Sensors* 12.6 (2012), pp. 6893–6919. ISSN: 1424-8220. DOI: 10.3390/s120606893. URL: https://www.mdpi.com/1424-8220/12/6/6893.

[47] Natan Micheletti, Jim Chandler and Stuart N. Lane. 'Structure from motion (SFM) photogrammetry'. In: (Jan. 2015). URL: https://repository.lboro.ac.uk/articles/journal_contribution/Structure_from_motion_SFM_photogrammetry/9457355.

[48] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV].

[49] Saritha Murali and V. K. Govindan. 'Shadow Detection and Removal from a Single Image Using LAB Color Space'. In: *Cybernetics and Information Technologies* 13.1 (2013), pp. 95–103. DOI: doi:10.2478/cait-2013-0009. URL: https://doi.org/10.2478/cait-2013-0009.

[50] Kai Ninomiya et al. *WebGPU*. URL: https://www.w3.org/TR/webgpu/. (accessed: 05.06.2024).

[51] Amparo Núñez Andrés et al. 'Generation of virtual models of cultural heritage'. In: *Journal of Cultural Heritage* 13.1 (2012), pp. 103–106. ISSN: 1296-2074. DOI: https://doi.org/10.1016/j.culher.2011.06.004. URL: https://www.sciencedirect.com/science/article/pii/S1296207411000628.

[52] Björn Ottosson. *A perceptual color space for image processing*. Dec. 2020. URL: https://bottosson.github.io/posts/oklab/. (accessed: 05.06.2024).

[53] Fornaro Peter et al. 'Enhanced RTI for gloss reproduction'. In: *Electronic Imaging* 29.8 (2017), pp. 66–66. DOI: 10.2352/ISSN.2470-1173.2017.8.MAAP-284. URL: https://library.imaging.org/ei/articles/29/8/art00011.

[54] R. Pintus et al. 'State-of-the-art in Multi-Light Image Collections for Surface Visualization and Analysis'. In: *Computer Graphics Forum* 38.3 (2019), pp. 909–934. DOI: https://doi.org/10.1111/cgf.13732. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13732. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13732.

[55] Elad Richardson et al. *TEXTure: Text-Guided Texturing of 3D Shapes*. 2023. arXiv: 2302.01721 [cs.CV]. URL: https://arxiv.org/abs/2302.01721.

[56] Olaf Ronneberger, Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[57] Shunsuke Saito et al. 'PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization'. In: *CoRR* abs/2004.00452 (2020). arXiv: 2004.00452. URL: https://arxiv.org/abs/2004.00452.

[58] Martin Schaich. 'Combined 3D Scanning and Photogrammetry Surveys with 3D Database Support for Archaeology & Cultural Heritage. A Practice Report on ArcTron's Information System aSPECT3D'. In: *Photogrammetric Week 2013*. Ed. by Dieter Fritsch. Wichmann/VDE Verlag, Berlin & Offenbach, 2013, pp. 233–246.

[59] Sketchfab. *Popular 3D models - Sketchfab*. URL: https://sketchfab.com/3d-models/popular. (accessed: 05.06.2024).

[60] Alvy Ray Smith. 'Color gamut transform pairs'. In: *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '78. New York, NY, USA: Association for Computing Machinery, 1978, pp. 12–19. ISBN: 9781450379083. DOI: 10.1145/800248.807361. URL: https://doi.org/10.1145/800248.807361.

[61] Andrea Spognetta. *Krzyztopor Castle -RAWscan*. June 2018. URL: https://sketchfab.com/3d-models/krzyztopor-castle-rawscan-2ff9f0b15e5242b9acea7033cd07a972. (accessed: 03.05.2024).

[62] CIE Standard et al. 'Colorimetry-part 4: CIE 1976 L* a* b* colour space'. In: *International Standard* (2007), pp. 2019–06.

[63] Zeljko Stojkic, Eva Čuljak and Luka Šaravanja. '3D Measurement - Comparison of CMM and 3D Scanner'. In: Jan. 2020, pp. 0780–0787. ISBN: 9783902734297. DOI: 10.2507/31st.daaam.proceedings.108.

[64] Ke Sun et al. 'Deep High-Resolution Representation Learning for Human Pose Estimation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[65] Polyga Support. *3D Scanning 101: Basics of Structured Light 3D Scanning*. Dec. 2023. URL: https://www.polyga.com/blog/3d-scanning-101-structured-light-3d-scanning/. (accessed: 25.08.2024).

[66] Niranjan Thanikachalam et al. 'Handheld Reflectance Acquisition of Paintings'. In: *IEEE Transactions on Computational Imaging* 3.4 (2017), pp. 580–591. DOI: 10.1109/TCI.2017.2749182.

[67] Twindom. *Full Body 3D Scanners for 3D Printed Figurines, 3D Portraits and 3D Selfies by Twindom*. URL: https://web.twindom.com/. (accessed: 05.06.2024).

[68] Jifeng Wang, Xiang Li and Jian Yang. 'Stacked Conditional Generative Adversarial Networks for Jointly Learning Shadow Detection and Shadow Removal'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[69] Zian Wang et al. 'Neural Fields Meet Explicit Geometric Representations for Inverse Rendering of Urban Scenes'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 8370–8380.

[70] Joshua Weir et al. 'Deep Portrait Delighting'. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 423–439. ISBN: 978-3-031-19787-1.

[71] Xin Yu et al. *Texture Generation on 3D Meshes with Point-UV Diffusion*. 2023. arXiv: 2308.10490 [cs.CV]. URL: https://arxiv.org/abs/2308.10490.

[72] Xianfang Zeng et al. *Paint3D: Paint Anything 3D with Lighting-Less Texture Diffusion Models*. 2023. arXiv: 2312.13913 [cs.CV]. URL: https://arxiv.org/abs/2312.13913.

[73] Song Zhang. 'High-speed 3D shape measurement with structured light methods: A review'. In: *Optics and Lasers in Engineering* 106 (2018), pp. 119–131.

[74] Xuaner Cecilia Zhang et al. *Portrait Shadow Manipulation*. 2020. arXiv: 2005.08925 [cs.CV].

[75] Jinghong Zheng et al. 'Collaborative image processing algorithm for detail refinement and enhancement via multi-light images'. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 1382–1385. DOI: 10.1109/ICASSP.2010.5495456.

[76] Hao Zhou et al. 'Deep Single-Image Portrait Relighting'. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 7193–7201. DOI: 10.1109/ICCV.2019.00729.

A. Appendix

```rust
use crate::{
    color_processing::color_space_pixel::ColorSpacePixel,
    data_loader::data_image_pair::DataImagePair,
    model::{PixelDelighterModel, SavedModelFilePath},
};

pub fn run_pixel_de_lighting<
    ModelType: PixelDelighterModel,
    ColorSpace: ColorSpacePixel,
>(
    data_image_pair: DataImagePair,
) {
    // Load the de-lighter model.
    let mut model = ModelType::new(
        "./my_model_directory/de_light_model.safetensors",
    );

    // Load input image data.
    // The image data is stored as RGB data on disk.
    let color_input_image =
        image::open(data_image_pair.get_color_image_path())
            .unwrap()
            .to_rgb8();

    let light_color_input_image =
        image::open(data_image_pair.get_light_color_image_path())
            .unwrap()
            .to_rgb8();

    let uv_are_map_image =
        image::open(data_image_pair.get_uv_area_image_path())
            .unwrap()
            .to_luma8();

    let height = color_input_image.height();
    let width = color_input_image.width();

    // Create an all black output image that will
    // be filled with the processed pixels.
    let mut processed_output_image =
        image::ImageBuffer::from_fn(width, height, |_x, _y| {
            image::Rgb([0, 0, 0])
        });


```

```
    // Iterate over every single pixel in the images.
    for y in 0..height {
        for x in 0..width {
            let uv_area_map_pixel = uv_are_map_image.get_pixel(x, y);
            // This pixel is white in the UV area map and therefore
            // the pixel of the color texture is mapped to
            // geometry of the 3D model.
            // -> Run de-lighter model on it.
            // Ignore all black pixels in the UV area map.
            if uv_area_map_pixel[0] > 0 {
                let color_pixel = color_input_image.get_pixel(x, y);
                let light_color_pixel =
                    light_color_input_image.get_pixel(x, y);

                // Convert the RGB pixel data to the given color
                // space and normalize the values to the range
                // of [-1.0, 1.0].
                let color_pixel =
                    ColorSpace::convert_rgb_to_normalized_color_space(
                        color_pixel,
                    );
                let light_color_pixel =
                    ColorSpace::convert_rgb_to_normalized_color_space(
                        light_color_pixel,
                    );

                // The de-lighter model outputs the processed pixel in
                // the same normalized range of [-1.0, 1.0] as the
                // input pixels.
                let processed_normalized_pixel =
                    model.delight(&color_pixel, &light_color_pixel);

                // Convert the processed pixel back to RGB color space
                // by de-normalizing the color data again and converting
                // the used `ColorSpace`, e.g., Oklab, back to RGB
                // to write the pixel to the output image.
                let processed_de_normalized_pixel =
                    ColorSpace::convert_normalized_color_space_to_rgb(
                        &processed_normalized_pixel,
                    );
                processed_output_image.put_pixel(x, y,
                    processed_de_normalized_pixel);
            }
        }
    }

    // Save the processed output image to disk.
    processed_output_image
        .save("./my_output_directory/output.png")
        .unwrap();
}
```

Listing A.1: General image pixel data iteration process of the pixel-based de-lighter model.