UNIVERSITÀ
di VERONA

Department of Computer Science

Lights on cultural heritage: analysis and visualization of surfaces for cultural heritage based on multi-light imaging and artificial intelligence

Leonardo Righetto
Supervisor: Andrea Giachetti

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science of the University of Verona.

April 24, 2026

La borsa di dottorato è stata cofinanziata con le risorse del PNRR:

• per il DM 351 nell'ambito della Missione 4 ("Istruzione e ricerca") – Componente 1 ("Potenziamento dell'offerta dei servizi di istruzione: dagli asili nido all'Università"), Investimento 3.4. ("Didattica e competenze universitarie avanzate") e Investimento 4.1 ("Estensione del numero di dottorati di ricerca e dottorati innovativi per la pubblica amministrazione e il patrimonio culturale") - progetto M4C1 –Inv. 3.4 e progetto M4C1 – Inv. 4.1

• per il DM 352, nell'ambito della Missione 4 ("Istruzione e Ricerca") – Componente 2 ("Dalla Ricerca all'Impresa"), Investimento 3.3 ("Introduzione di dottorati innovativi che rispondono ai fabbisogni di innovazione delle imprese e promuovono l'assunzione dei ricercatori da parte delle imprese") – progetto M4C2 Investimento 3.3

Abstract

Reflectance Transformation Imaging (RTI) is a well-established imaging technique, which uses multi-light data to create relighting models, and is used to improve the visualization of Cultural Heritage (CH) objects. This thesis presents the work carried out during my PhD project, focused on improving and evaluating RTI techniques as a tool to assist experts to analyze cultural finds. In chapter 1 we introduce the problem, briefly describing which were the challenges and how they've been tackled. In chapter 2 we provide a background description of RTI, other techniques, and what has been done until the beginning of the project. In chapter 3 we discuss a neural based method to create RTI models, namely NeuralRTI, the modifications we've done to the network structure and experiments we performed to assess the improvements achieved. In chapter 4 we demonstrate how we integrated the relighting capability of NeuralRTI into an online viewer for visualizing RTI images, showing challenges we encountered and strategies we adopted to reach real-time rendering. In chapter 5 we discuss the usefulness of RTI when applied to real world use cases, presenting collaborations with researchers and conservators who evaluated the use of RTI to analyze different types of objects. In chapter 6 we make an overview of further problems regarding multi-light data, for example the detection and removal of self-casted shadows in the stack of acquired images. Finally, in chapter 7 we conclude describing the outcomes of this projects and which are the future directions, and we also briefly describe ideas on how to extend the use of RTI beyond its usual application, discussing data annotation, the creation of relightable images of translucent materials, and the combination of relighting and multispectral imaging.

Sommario

Reflectance Transformation Imaging (RTI) è una tecnica di imaging ben consolidata, che utilizza dati provenienti da illuminazioni multiple, chiamati dati multi-light, per creare modelli di re-illuminazione, ed è impiegata per migliorare la visualizzazione di oggetti appartenenti al Patrimonio Culturale (CH). Questa tesi presenta il lavoro svolto durante il mio progetto di dottorato, focalizzato sul miglioramento e sulla valutazione di tecniche RTI come strumento di supporto per esperti nell'analisi dei reperti culturali. Nel capitolo 1 introduciamo il problema, descrivendo brevemente le sfide affrontate e le soluzioni adottate. Nel capitolo 2 forniamo una descrizione generale del RTI, di altre tecniche correlate, e di cosa è stato fatto fino a quando il progetto è iniziato. Nel capitolo 3 discutiamo un metodo basato su reti neurali per la creazione di modelli RTI, chiamato NeuralRTI, le modifiche apportate alla struttura della rete e gli esperimenti condotti per valutare i miglioramenti ottenuti. Nel capitolo 4 mostriamo come abbiamo integrato la capacità di re-illuminazione di NeuralRTI in un visualizzatore online per immagini RTI, spiegando le sfide incontrate e le strategie adottate per raggiungere un rendering in tempo reale. Nel capitolo 5 discutiamo l'utilità del RTI quando applicato a casi d'uso reali, presentando collaborazioni con ricercatori e conservatori che hanno valutato l'impiego del RTI per analizzare diversi tipi di oggetti. Nel capitolo 6 discutiamo ulteriori problematiche legate ai dati multi-light, come ad esempio la rilevazione e la rimozione delle ombre, che l'oggetto proietta su se stesso, nel set di immagini acquisite. Infine, nel capitolo 7 concludiamo descrivendo i risultati di questo progetto e le possibili direzioni future, e descriviamo brevemente delle idee per estendere l'uso del RTI oltre la sua applicazione tradizionale, discutendo di annotazione di dati, la creazione di immagini re-illuminabili di materiali traslucidi, e la combinazione del RTI con l'imaging multispettrale.

Publications

Righetto, L., Gobbetti, E., Ponchio, F., Traviglia, A., De Bernardin, M., & Giachetti, A. (2023, August). Ancient coins' surface inspection with web-based neural RTI visualization. In Optics for Arts, Architecture, and Archaeology (O3A) IX (Vol. 12620, pp. 91-98). SPIE.

Righetto, L., Bettio, F., Ponchio, F., Giachetti, A., & Gobbetti, E. (2023). Effective interactive visualization of neural relightable images in a web-based multi-layered framework. In GCH 2023-Eurographics Workshop on Graphics and Cultural Heritage (pp. 57-66).

Righetto, L., Khademizadeh, M., Giachetti, A., Ponchio, F., Gigilashvili, D., Bettio, F., & Gobbetti, E. (2024). Efficient and user-friendly visualization of neural relightable images for cultural heritage applications. ACM Journal on Computing and Cultural Heritage, 17(4), 1-24.

Righetto, L., Marchioro, G., & Giachetti, A. (2025, August). Enhancing manuscript analysis with reflectance transformation imaging: a user study on interactive visualization options. In Optics for Arts, Architecture, and Archaeology (O3A) X (Vol. 13569, pp. 101-111). SPIE.

Righetto, L., Ullah, S., & Giachetti, A. (2025). BASS-MLIC: a Novel Synthetic Dataset for Single-View Inverse Rendering Tasks on Cultural Heritage Artifact. In Smart Tools and Applications in Graphics - Eurographics Italian Chapter Conference. Eurographics. (Poster)

Dulecha, T. G., **Righetto, L.**, Pintus, R., Gobbetti, E., & Giachetti, A. (2024). Disk-NeuralRTI: Optimized NeuralRTI Relighting through Knowledge Distillation. In Smart Tools and Applications in Graphics - Eurographics Italian Chapter Conference (pp. 1-10). Eurographics.

Dulecha, T. G., **Righetto, L.**, Pintus, R., Gobbetti, E., & Giachetti, A. (2025). Fast and accurate neural reflectance transformation imaging through knowledge distillation. Computers & Graphics, 104475.

Ponchio, F., Bettio, F., Marton, F., Pintus, R., **Righetto, L**., Giachetti, A., Gobbetti, E. "OpenLIME: An open and flexible web framework for creating and exploring complex multi-layered relightable image models". Digital Heritage 2025 - 4th International Congress & Expo, 2025.

Fsian, A., Khawaja, M. A., Kresovic, M., **Righetto, L**., & Hardeberg, J. Y. (2025, July). Apple bruise detection using hyperspectral imaging. In Seventeenth International Conference on Digital Image Processing (ICDIP 2025) (Vol. 13709, pp. 784-795). SPIE.

Acknowledgments

I'd like to thank my supervisor Andrea and my friends at the lab in Verona, as their help and motivation allowed me to reach the end of this PhD project. I also thank my friend Arsalan, who shared with me the same research topic and so we could discuss together while playing table tennis. Finally, I thank my cat, who sat next to me while writing this thesis  .

Contents

List of Figures

List of Tables

List of Abbreviations

CH	Cultural Heritage
DL	Deep Learning
GLSL	OpenGL Shading Language
GT	Ground Truth
HSH	HemiSpherical Harmonics
ML	Machine Learning
MLIC	Multi-Light Image Collection
NN	Neural Network
OpenLIME	Open Layered IMage Explorer
PCA	Principal Component Analysis
PS	Photometric Stereo
PTM	Polynomial Texture Map
RBF	Radial Basis Functions
RTI	Reflectance Transformation Imaging
STRESS	Spatio-Temporal Retinex-Inspire Envelope with Stochastic Sampling
WebGL	Web Graphics Library

Chapter 1: Introduction

This thesis presents the work done during my PhD project in Computer Science at Università degli studi di Verona, in collaboration with the Center for Cultural Heritage Technology of the Italian Institute of Technology for 9 months, in Venice, Italy, and the Norwegian University of Science and Technology for 6 months, in Gjøvik, Norway. My PhD scholarship was supported by the EU Next-GenerationEU PNRR, M4C1 Inv. 3.4 "dottorati innovativi per la pubblica amministrazione e il patrimonio culturale", Decreto Ministeriale n. 351 del 9 aprile 2022.

The project proposal aimed at using advanced imaging technologies, such as multi-light, multi-spectral, and polarized light, to digitize and analyze cultural heritage surfaces, which often present challenges due to material variability and degradation. The research initially explore BRDF estimation and AI-based automatic annotation, using custom acquisition systems developed at UNIVR. Potential applications included coins, inscriptions, ceramics, and glass artifacts. The project's purposes aligned with national initiatives for cultural heritage and all tools and data would have been released publicly under Open Science principles.

As we will see while going through this thesis, the central theme that we followed during the realization of the project has been developing imaging tools designed to be used by both researchers and other type of experts in the field of cultural heritage. The project's partners, i.e., CCHT-IIT and NTNU, helped by providing interesting and practical use cases to test relighting algorithms on complex surface materials, and by allowing to evaluate such algorithms with the help of users which were experts in the field, like aforementioned museum conservators and university researchers.

Reflectance Transformation Imaging (RTI) is an imaging technique, which uses multi-light data to create relighting models. Multi-light data are images captured placing the camera in a fixed position, and moving the source of light for every image. These images give cues on how the light reflects on the surface of the captured object, so they can be interpolated to create a relighting model, i.e., an algorithm that can digitally change the illumination of the image, creating what is called relightable image. The light information contained in the multi-light data is first encoded into a new set of data, which we will call planes or features, and then decoded into the newly generated image with novel illumination. Applications of RTI can be many, but in the recent years its use has been prioritized in the context of Cultural Heritage (CH), in order to improve the visualization of objects and assist experts to perform their studies. This thesis presents the work carried out during my PhD project, focused on improving and evaluating RTI techniques.

Chapter 2 provides a background explanation of RTI. We analyze how to acquire a set of images to create a RTI model and which setups are mostly used, which algorithms can be used to interpolate the images, and which software are available to visualize the result.

In chapter 3 we analyze the content of the papers [1] and [2], discussing the use of encodings based on neural structures, as they have the potential to replace common polynomial fitting when creating RTI models. Structures based on Neural Networks (NNs) proved to lead to more compact encoding with better relighting quality. However, NNs are complex non-linear models which require many mathematical operations, thus replacing classical algorithms with neural ones leads to the problem of achieving real-time rendering, a critical issue in the context of RTI, as the main objective is interactivity with the relightable image. The work consists in optimizing NeuralRTI [3], an encode-decode NN used as a relighting model.

The first step was to perform several experiments testing different encoder and decoder architectures, evaluating the quality of the results on specific benchmarks to find the optimal tradeoff between relighting quality and efficiency. Then, the system has been tested for the analysis of a group of ancient Roman bronze coins, that present different levels of preservation and the content is hard to identify. The level of corrosion and degradation, which in some cases hinders the recognition of the images, numerals, or text represented on them, makes the system testing particularly challenging and complex.

In a second instance, the model, already improved and simplified, has been further refined using a technique called knowledge distillation. The idea is to train first the standard model (teacher), and then train a smaller model (student), using the first model as a reference. This way, the performance reached by the smaller model are generally higher with respect to if it was trained from scratch. The refined model, called Disk-NeuralRTI [2], with a reduced number of parameters provides better rendering speed, and was tested on high-resolution real-world datasets showing comparable relighting quality to its teacher model.

First, my main contribution has been simplifying the network, testing different parametric configurations to find a good trade-off between relighting quality and rendering speed. In the case of Disk-NeuralRTI, I took care of testing the rendering speed of different network's configurations at visualization time, to assess the improvement introduced with the new method.

In chapter 4 we discuss how NeuralRTI has been integrated into OpenLIME [4], an open-source library to create web-based RTI viewers, presented in the papers [5] and [6].

The integration of neural based models into interactive visualization tools has generally been limited due to the high number of operations required at inference time, which makes it harder to reach real-time rendering especially when inspecting high-resolution images. Moreover, integrating NNs in a different tool usually requires the use of Deep Learning (DL) libraries ad-hoc for the destination tool, which is not a common thing for RTI viewers. Therefore, we discuss how the simple structure of

NeuralRTI allowed to integrate its decoder, responsible for the relighting operations, without resorting to DL libraries, and which strategies have been adopted to achieve real-time rendering nonetheless.

Thanks to the work discussed in chapter 3, we started to work with a model that has already been optimized with the purpose of being usable in real-time. Furthermore, my main contribution for this work has been implementing a custom WebGL shader specific to the task, which made DL libraries unnecessary for the integration in the code. Finally, we implemented a management system supporting adaptive resolution rendering through on-the-fly resampling in latent feature space.

Chapter 5 concerns the usefulness of using RTI for analyzing objects in practical use cases, where we asked users to perform evaluations of RTI tools, which have been presented in the papers [6] and [7].

So far, we discussed about advancements that have been made from the technical point of view, like new interpolation techniques which lead to better relighting quality and data compression. The effectiveness of the relighting and enhancement methods, however, is difficult to assess in practice, as it requires experts' feedback on domain-specific data. We show the importance of interviewing experts of different branches of the CH field, as their feedback can lead to a more focused way to improve RTI visualization, and can assess whether new developed techniques are actually useful for objects investigation.

Therefore, we present two practical use cases where every new advancement has been tested, like the improved NeuralRTI, its online integration, and more. Different evaluations have been performed, both from the objective point of view using numerical metrics, and the subjective point of view asking human users to compare different relighting methods. This chapter also introduces the use of image enhancement algorithms applied to relightable images, and the efficiency of creating an application that allows to relight and enhance images within the same interface.

In the first case, we analyze a comparison between classical RTI and neural based RTI, where an expert in cultural heritage analyzes advantages and disadvantages of the two for visualizing Viking textiles. Moreover, the expert also points out in which ways using OpenLIME improves the visualization experience, with respect to previous visualization systems. For the second work we conducted a more exhaustive user evaluation, asking ten users to assess the usefulness of a group of RTI techniques for analyzing the ancient manuscripts. For these works, my main contributions have been the realization of custom versions of the web viewer, implementing new visualization functionalities, and the creation of a questionnaire for collecting users feedback.

In chapter 6, we introduce a novel synthetic dataset: BASS-MLIC (BAS-reliefs Synthetic Multi-Light Image Collections). The motivation of creating such dataset is to evaluate different multi-light image processing tasks on realistic objects and materials, as well as to train neural models able to perform inverse rendering on them.

The dataset includes multi-light image collections created with Blender, featuring orthographic

views of typical surfaces analyzed in the Cultural Heritage domain, like bas-reliefs or inscribed tablets. We applied a set of homogeneous materials to the surfaces, as well as realistic combinations of different materials assigned to manually segmented parts of the objects. We created the materials with the Blender BSDF model, designing them to reproduce realistic materials such as stone, metal, and ceramic.

For each MLIC, we included ground truth annotations for all the relevant parameters, such as normal maps, depth maps, shadow maps, material segmentation masks, and BRDF parameters. Thanks to this complete annotation, the dataset can be used not only to validate relighting quality and Photometric Stereo methods estimating normal maps, as done with the existing benchmarks, but also to validate shadow segmentation methods, to perform shadow-aware Photometric Stereo and BRDF fitting quality, and to evaluate the quality of depth estimation methods.

In order to motivate the creation of a dataset with this kind of images, we also include a series of preliminary experiments showing the potential use of BASS-MLIC for practical tasks. Despite the results that we present are not particularly good, the main reason was not to propose new methods, but instead shows which research questions can emerge when working with 2.5D, multi-light images. For this work I took care of choosing the 3D models and material's parameters, rendering the images in Blender and defining which methods to use for the practical tasks.

Finally, chapter 7 summarize the work done during the PhD project, giving an idea of what can be future directions and how this work can be further extended. In addition, it includes a collection of works which address less conventional uses of RTI acquisitions. First, an overview about data annotation in OpenLIME, where I assisted a CH expert in annotating an old metallic plate with hard to read engravings. Then, the use of RTI for analyzing translucent materials, where I collaborated in a group project acquiring the images and creating the web visualization. Finally, a brief discussion of mutlispectral imaging and possible RTI applications, having worked together with other PhD students to projetc that has been presented in the paper [8]. The ideas presented in this chapter can be seen as a starting point for RTI projects in the future.

Chapter 2: Background

Reflectance Transformation Imaging (RTI) [9, 10] is nowadays a widely used technique for digital and interactive inspection of objects, especially in the field of Cultural Heritage (CH). The overall process consists of different steps: image acquisition, image processing, image interpolation to create a relighting model, and visualization and interaction with such model. The acquisition consists in capturing a set of images in different illumination conditions, i.e., placing the camera in a fixed position with respect to the object, and moving the light source in a different position for every picture. A set of images captured this way is called Multi-Light Image Collection (MLIC) [11]. RTI turns a MLIC into a relighting model, that is able to generate brand new images from novel light directions in a real-time and interactive setting. A schematic of RTI is shown in Figure 2.1.

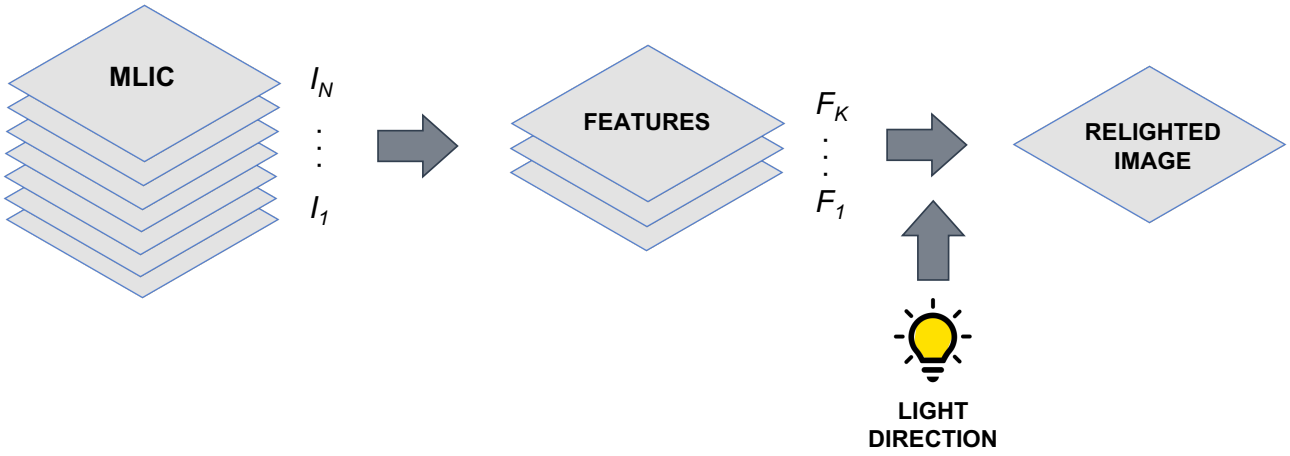


Figure 2.1: RTI scheme. A MLIC consists of a large number of images, which is reduced in dimensionality, extracting useful features used to render the relighted image.

A MLIC is a stack of data rich of information about how the light changes in the scene. Photometric Stereo (PS) is a technique that uses the MLIC to extract the normal map of the scene, i.e., an image where each pixel represents the normal vector with respect to the surface in that point. The normal map can be used in different ways to enhance the visualization of a RTI image.

Another way to create a relighting model is to use physically based models like the Bidirectional Reflectance Distribution Function (BRDF) [12] or another function from the same family. A RTI model simply interpolates pixel-wise the values of the images, trying to reproduce their appearance. A BRDF model, instead, uses a set of maps, e.g., albedo, normal map, and others, as input to a mathematical formulation that describes the behavior of the material. Such models, however, are hardly feasible to be computed, as they required a deep knowledge of the properties of each material in the scene. When visualization is the key feature, RTI provides and easier, more reliable solution.

2.1 RTI pipeline

2.1.1 Acquisition

The acquisition setup can vary. Originally, the light source was manual-handled by a human user. This setup is easy to implement, and provides a certain degree of freedom in regards of where to place the light source, but requires a strong calibration method to assess the light source position in every shot. One popular device used to acquire a RTI stack of images is the RTI Dome [13], shown in Figure 2.2, which is a hemispherical structure equipped with LEDs in fixed positions, where a camera can be attached to the center of the hemisphere. This setup allows to acquire a fixed number of images from known light directions, providing a portable tool which is a good tradeoff between stability during the acquisition and transportation of the device, but on the other side it allows to acquire objects of limited size. Another alternative is to use a robotic arm, placing the light source on the wrist of the arm. This method provides a high degree of freedom, high precision, and makes the acquisition easy to be repeated in the same conditions. Different devices imply different cost, the acquisition time, and calibration requirement. A more complex setup can reduce the amount of processing that has to be applied on the acquired images.

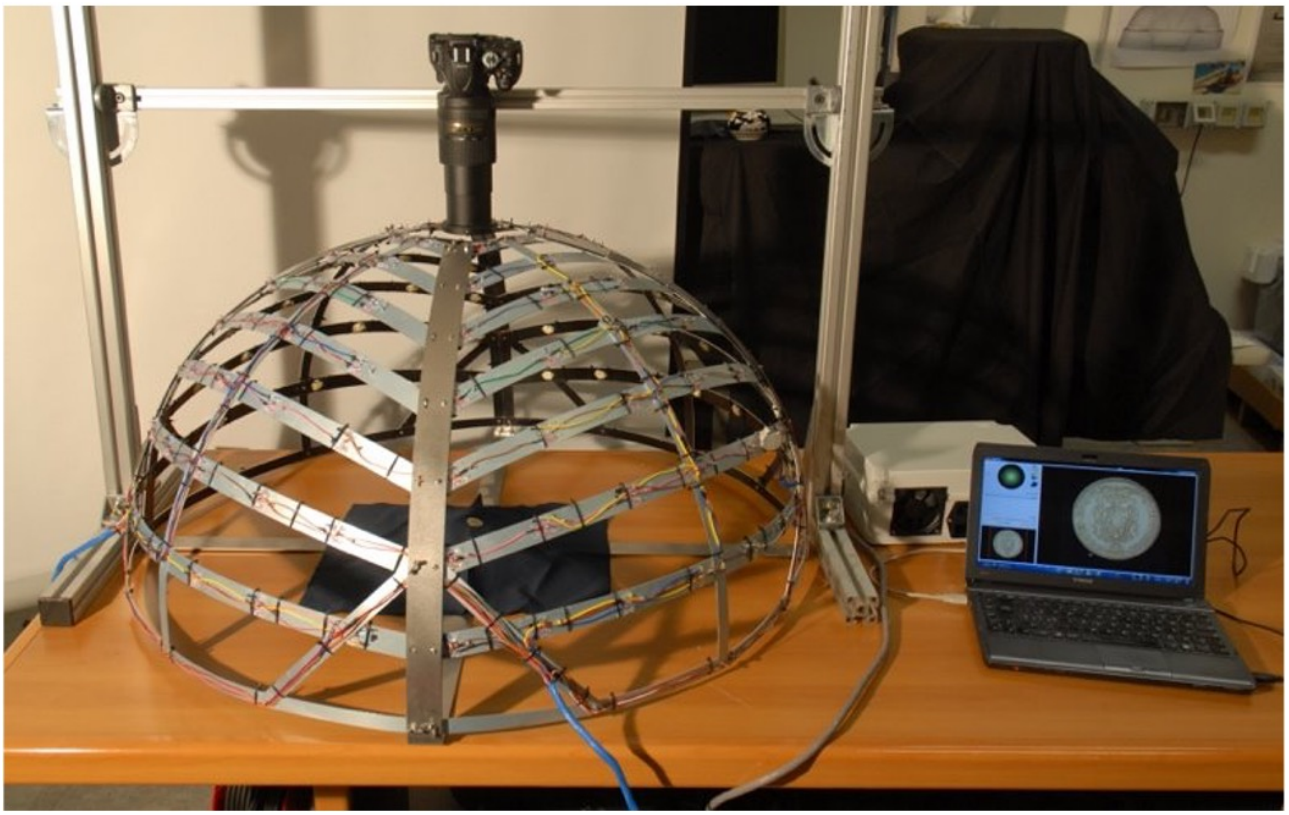


Figure 2.2: RTI dome which consists of a metallic structure, where LED lights are mounted on. Image is taken from `https://vcg.isti.cnr.it/~palma/rti/acquisition.php`.

2.1.2 Pre-processing

The images obtained after acquisition are considered raw data. Collected images must be aligned before being used as input for any algorithm, because it's required that each pixel represent the same content across different images. Moreover, a reference frame has to be defined to know the directions of light, and this is often done on the raw data, exploiting different techniques (e.g., calibration objects in the scene). In the case of RTI, the pre-processing step mainly involve balancing light distribution. A spot in the scene is brighter close to the light source, therefore the pixel intensity of every image has to be rescaled accordingly. One way to do it is by placing a white reference in the scene and using a technique called flat fielding. Alternatively, by knowing the physical dimensions of both the objects and the acquisition setup, it's possible to use the laws of light propagation, i.e., inverse-squared distance and cosine, to adjust the pixel intensity.

2.1.3 Post-processing

Once the raw images have been refined, the relighting model is created by interpolating pixel by pixel the images of the MLIC. The relighting model takes as input the light direction, and gives as output an image, containing the same scene that was acquired, but estimating its intensity as if the light source position was the light direction given as input. The interpolation algorithm can vary, and methods have been proposed both in the classical approach, where there is a mathematical formulation in closed form and models are linear, and in the neural approach, where the relighting model is a Neural Network (NN), making the model non-linear and data-dependent.

2.1.4 Visualization

Especially with RTI, where relighting models give as output what is called "relightable image", and the aim is to interactively change the light direction, one fundamental task is to create an application where users can visualize the result. At this time, this kind of applications has been used in the CH field to analyze archeological finds or objects of cultural interest, hoping that the novel light directions, viewed in a homogeneous way, might highlight some peculiar details that weren't noticeable otherwise. Among the numerous possibilities for developing a visualization tool on modern hardware, web-based applications are a best choice: they allow fast computation of relighting models, exploiting parallelization and power computation of Graphic Processing Units (GPUs), and they are easy to share with anyone as they run on commonly available web browsers.

2.2 Relighting algorithms

In order to generate the relighting model, images from the MLIC are interpolated pixel-wise. The interpolation process produces a new set of images, that represent the encoded light information extracted from the original images. We call these new images encoded features. Afterwards, it's possible to use these features, in combination with a light direction, as input to an algorithm which generates the relighted image. The light direction is generally given as a vector $\vec{L} = (l_x, l_y, l_z)$, which describe the position of the light source in the space. As a common rule, in a RTI setting, the light source is positioned on the surface of an imaginary hemisphere around the object, and the camera is positioned at the top of the hemisphere. Therefore, only two coordinates are necessary, as the third one is dependent on the others two by the equation:

$$l_x^2 + l_y^2 + l_z^2 = 1 \tag{2.1}$$

Alternatively, the light position can be described by the angles with respect to the center of the reference frame, namely elevation angle $\theta = \arccos(l_z)$ and azimuth angle $\phi = \arctan\left(\frac{l_y}{l_x}\right)$. In order to generate the encoded features, a set of algorithms have been developed through the years, both based on classical and neural architectures. Such algorithms operate at pixel level, therefore equations shown in following subsections show the mathematical operations that are applied separately to each pixel in the stack of images. As mentioned above, l_x and l_y or θ and ϕ will represent the light position, c_i with $i = 0, ..., n$ represent each coefficients of the encoded features, and C will represent the final color rendered. Usually, these algorithms are applied independently to each color channel, thus C represents either the red, green or blue color channel.

2.2.1 Polynomial Texture Map

The first applications of RTI have to be dated back to when the Polynomial Texture Map (PTM) [9] algorithm was published. In this case, the light reflectance function is estimated using a second order polynomial, described by the equation:

$$C = c_0 + c_1 l_x + c_2 l_y + c_3 l_x^2 + c_4 l_y^2 + c_5 l_x l_y \tag{2.2}$$

In order to encode the features, the least square algorithm is applied using the MLIC and all the associated light directions, stacked together in a single matrix, as input data. This model offers an easy solution, but with many limitations when the object's materials present a complex behavior, like strong specular reflections.

2.2.2 HemiSpherical Harmonics

This method utilizes the concept of Hemispherical Harmonics (HSH) [14, 15] to give a spatial description of how the light is reflected. With respect to PTM, reflections are better reproduced, but it still presents limitations for very complex materials.

$$C = c_0 H_0(\theta, \phi) + ... + c_n H_n(\theta, \phi) \tag{2.3}$$

The set of functions H_i, with $i = 0, ..., n$, is figuratively described in Figure 2.3.

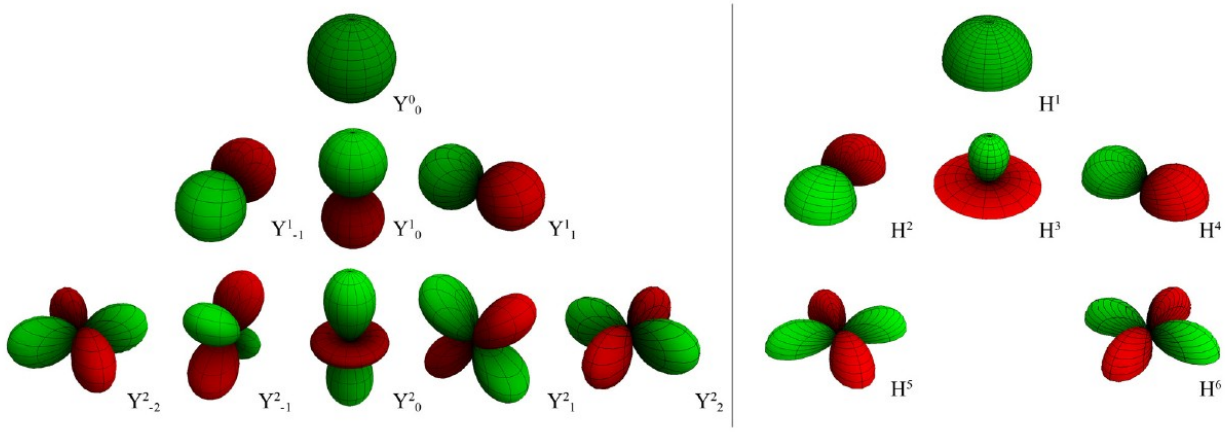


Figure 2.3: Figurative description of Spherical Harmonics (left) and HemiSpherical Harmonics (right). Image is taken from `https://vcg.isti.cnr.it/vcgtools/relight/presentation`.

2.2.3 Radial Basis Functions

This method is based on Radial Basis Functions (RBF) [16, 17], which encode the images using a set of N gaussian radial functions, with standard deviation σ, centered at the sampled light direction l_i:

$$C = \sum_{i=1}^{N} \alpha_i e^{\frac{||\vec{L} - \vec{L}_i||}{\sigma^2}} \tag{2.4}$$

Principal Component Analysis (PCA) can be applied for data compression, approximating the N RBF solutions to M PCA basis where μ is the average of data, B_k are the principal components, i.e., the directions in feature space along which the data varies the most, and β_k are the projection coefficients which tell you the coordinates of the data in the PCA space:

$$\alpha_{1...N} \cong \mu + \sum_{k=1}^{M} \beta_k B_k \tag{2.5}$$

2.2.4 NeuralRTI

The same mechanism adopted for classical algorithms can be applied to neural models. NeuralRTI [3] is a simple neural network composed of a sequence of fully-connected layers, also known as Multi-Layer

Perceptron (MLP). The architecture is called autoencoder, which means it presents an encoder-decoder structure. The encoder is trained to encode the MLIC into the features, while the decoder takes the features and the light direction as an input, concatenated together, to generate the relighted image. The network is trained pixel-by-pixel, trying to reproduce a different pixel, in a specific lighting condition, at every iteration. The potential of neural networks is that they are non-linear models, as after every layer a non-linear operation, called activation function, is applied to the data. For this reason, NeuralRTI has shown great abilities at reproducing complex materials' behaviors like specular reflections and shadows. A visual description of the functioning of one of NeuralRTI's layers is shown in Figure 2.4.

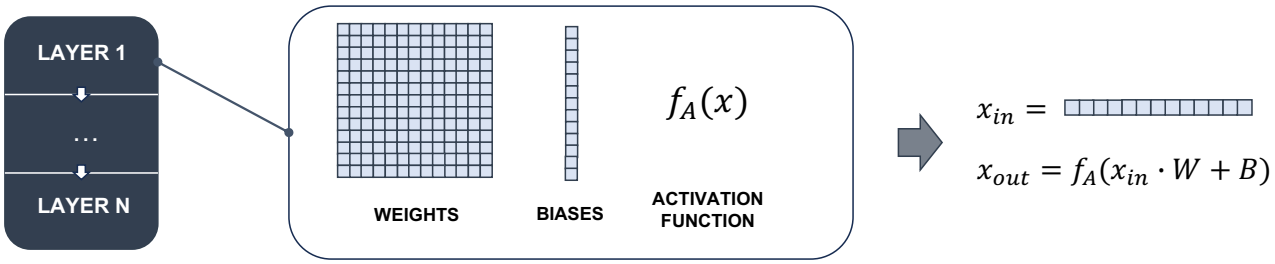


Figure 2.4: MLP Layer Scheme. This represents the architecture of each layer in NueralRTI, which consists of a matrix of weights, a vector of biases, and an activation function which adds non-linearity to the model.

2.3 RTI visualization software

One of the main purposes of relighting models is to provide to experts of cultural heritage a visualization tool to improve their analysis and investigations. In 2010, the Visual Computing Lab of ISTI-CNR `https://vcg.isti.cnr.it/` developed an application to visualize RTI images: RTIViewer. Afterwards, its web-based counterpart has been proposed, namely WebRTIViewer. The web version can be opened in a standard web page, without installing additional plugins. Furthermore, it has been used in many museum web pages to give visitors the possibility to interact with the relightable object. A third application, RTITool, was provided to generate the RTI models. More information is available on the web [LINK]. Nowadays, these applications are slowly being replaced by new ones, which better suit modern architectures and hardware. First, a standalone application named RelightLab `https://vcg.isti.cnr.it/vcgtools/relight/`, which can create RTI models, visualize relightable images, and more. Second, an open-source library named OpenLIME `https://github.com/cnr-isti-vclab/openlime`, completely written in native JavaScript, that allows to create web-based RTI viewers, provided of numerous features to enhance the visualization experience.

Chapter 3: Advancements of Neural Models

NeuralRTI [3] was proposed as a neural based alternative to encode multi-light data, and it showed huge improvements in the quality of the relighted images with respect to classic methods, as it was better at reproducing complex materials' behaviors like strong specular reflections and shadows casting.

In the original work, the network's architecture is an asymmetric autoencoder, which means it's composed of two sections: the first is called encoder, and it's used to encode the MLIC into latent space features; the second is called decoder, and it takes features and light direction as an input to generate the relighted image. Both the encoder and the decoder are composed of fully-connected layers of fixed size $3N$ where N is the number of input images. The encoder has three layers, while the decoder has four layers, and both of them use elu activation function, with $\alpha = 1$, shown in Equation 3.1. The network's scheme is shown in Figure 3.1.

$$\text{elu}(x) = \begin{cases} x & \text{if x}\geq 0 \\ \alpha(e^x - 1) & \text{if x}<0 \end{cases} \tag{3.1}$$

As for many methods relying on neural networks, the main limit is the complexity of the runtime decoding depending on thousands of parameters due to the fully connected architecture.

A modified version of NeuralRTI was proposed by replacing the encoder with PCA [18], in order to process a huge number of input images obtained from a video sequence. However, in the decoder section one layer was added and the per-pixel encoding is also heavier. There are also methods, based on neural networks, that have been proposed to encode relightable images, even if not for real-time inspection in the style of RTI visualization [19, 20].

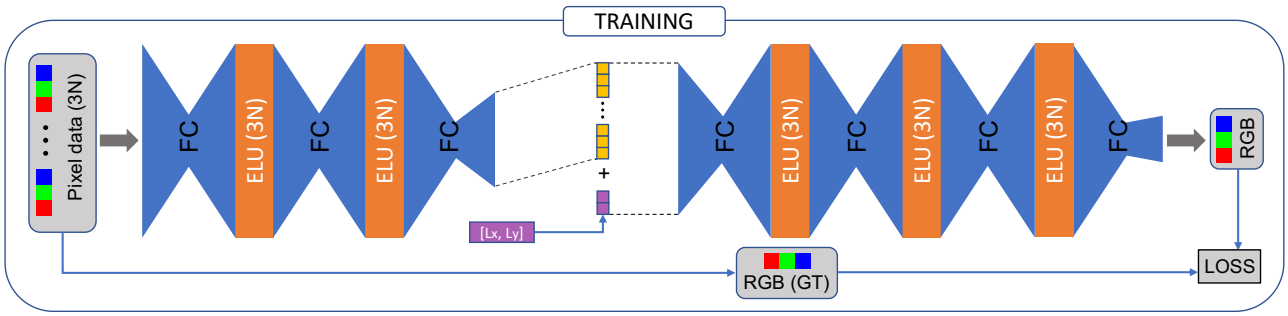


Figure 3.1: NeuralRTI original architecture. The decoder has more layer than the encoder, which slows down the inference.

It is important to notice that NeuralRTI uses a neural network architecture, but was developed to work as a classical RTI algorithm. For this reason, it does not follow the usual deep learning procedure

to create a generalized model which works across different datasets. Given a specific dataset (a MLIC), NeuralRTI is trained over all stacks of its pixels across all light directions, and it learns to relight that specific dataset. The model's strength relies in the complexity and non-linearity of the neural architecture. The main reason we decided to refine this model was specifically to make the decoder lighter in order to integrate it in a web application, while also preserving or improving the relighting quality.

3.1 Network simplification

The network is trained end-to-end on the set of RGB values (with corresponding light direction) with the goal of minimizing the Mean Squared Error (MSE) loss between predicted and measured pixel values for each specific light direction in the input data.

For the moment, we need to know that after the training, the decoder's parameters, as well as the encoded feature, have to be extracted and saved. Later on, in chapter 4, we will see how this data are stored, and why it is important that we make the decoder as light as possible, when we will explain how the visualization software is realized. Particular attention has to be put into data memory occupation. The features, outputted by the encoder, are a 3-dimensional matrix having the same resolution as the input images and a number of channels equal to the number of encoding parameters, set by the user. In the original NeuralRTI's work, it has been shown that the method is able to give a high relighting quality with just 9 parameters per pixel, a lower number with respect to classical models.

Despite a very compact per-pixel encoding, the main problem still remains, as the computational cost of the relighting is high. It requires loading $11 * 3N + (3N)^2 + 3 * 3N$ trained weights plus $3N + 3N + 3$ trained biases, and do the corresponding multiplications, for every pixel, every time the relighting operation is requested. To improve the decoder, making it lighter and decreasing the computational cost at inference time, we modified its original structure decreasing both the number of layers and the number of parameters for each layer.

The first ideas came from simple observations. For example, from the relighting point of view, there are no issues in adding layers to the encoder, as it will not be used at inference time. Instead, it is fundamental to reduce the complexity of the decoder. Therefore, we removed one layer from the decoder, and we added one layer to the encoder. The modified structure is shown in Figure 3.2. To test the performance of the modified network, we used the same benchmark used in the original work: SynthRTI `https://github.com/Univr-RTI/SynthRTI`, a synthetic dataset composed of many single-material and multi-material objects.

Table 3.1 shows that adding one more layer in the encoder and removing one layer in the decoder, keeping the layer size unchanged, the quality of the relighting measured on the benchmarks used in [3] is actually increased. Removing a second layer, instead, create a large error in the relighting of surfaces

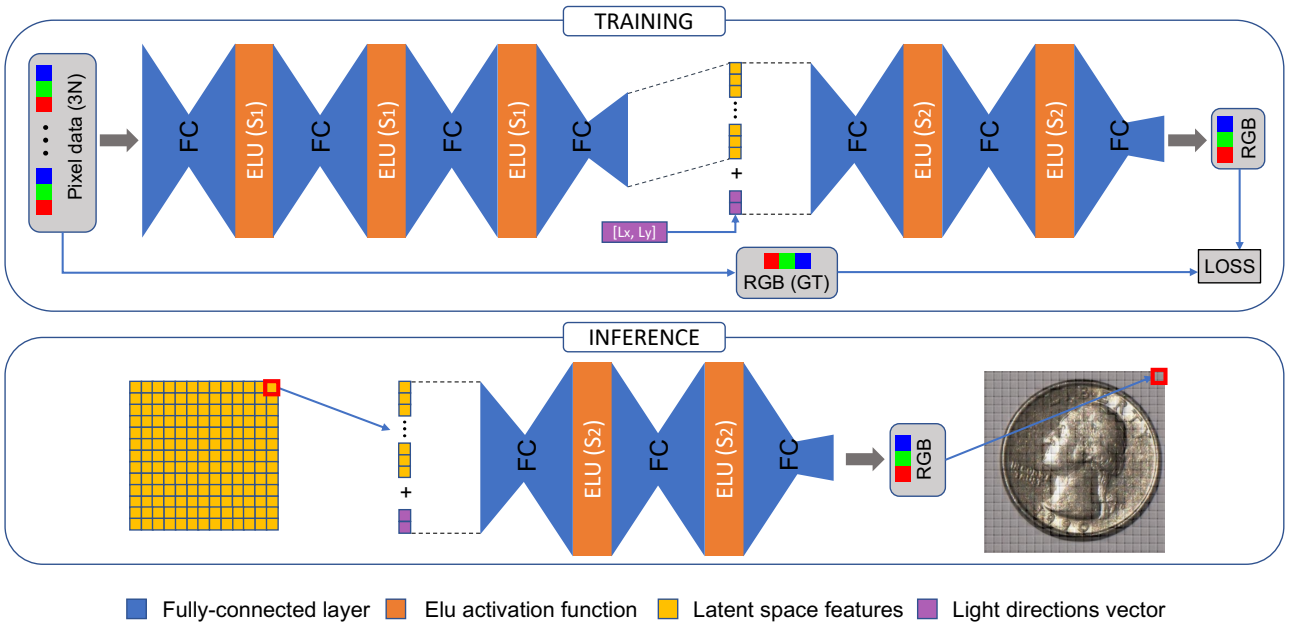


Figure 3.2: NeuralRTI modified architecture. A layer has been moved from the decoder to the encoder, and the number of parameters per layer has been changed in order to speed up the inference.

with multiple materials. The configuration with four encoder layers and three decoder layers seems therefore the optimal one.

Dataset	Neural 3L-4L	Neural 4L-3L	Neural 4L-2L
SynthRTI Single	31.00	**34.60**	31.92
SynthRTI Multi	27.39	**28.80**	25.76

Table 3.1: PSNR derived from testing different versions of NeuralRTI on the SyhthRTI dataset (xL-yL indicates x layers in the encoder, y layers in the decoder)

Moreover, it is not necessary that the fully-connected layers have a number of elements equal to the input size. The input size is 3N, as there are N input images from the MLIC, times 3 color channels per image (red, green and blue). The aim is to minimize the total number of the decoder's parameters. In our version of the model, we set the dimension of the layers to a fixed value: S_1 for the encoder, S_2 for the decoder. Therefore, to calculate the total number of decoder's parameters, we have to sum the weights (W) and the biases (B) of each layer, following Equation 3.2.

$$\begin{aligned} W &= 11 * S_2 + S_2^2 + S_2 * 3 \\ B &= S_2 + S_2 + 3 \end{aligned} \tag{3.2}$$

In a traditional RTI setup, the number of acquired images is approximately 50, therefore a common rule is $N \cong 50$. In the case of SynthRTI, each training set contains $N = 49$ images. The original network set the dimension of each layer to $3N$, thus to 147 when trained on SynthRTI. Therefore, the idea was to test the effect of changing the value of S_2, for $S_2 \leq 147$. The tests were performed on

SynthRTI in order to be comparable with the results from the original work. Specifically, we used $S_2 \in [150, 50, 25, 12]$. The metric used to evealuete the relighted images is Peak-Signal to Noise Ratio (PSNR), obtained by comparing each relighted image with its corresponding ground truth (image from the MLIC). Figure 3.3 shows the PSNR values for different number of parameters. It's interesting to notice how for multi-material surfaces (the green dots in the Figure), more relevant for practical applications, $S_2 = 50$ works well with respect to $S_2 = 150$, which corresponds to a large increment of parameters, but does not result in increased quality.

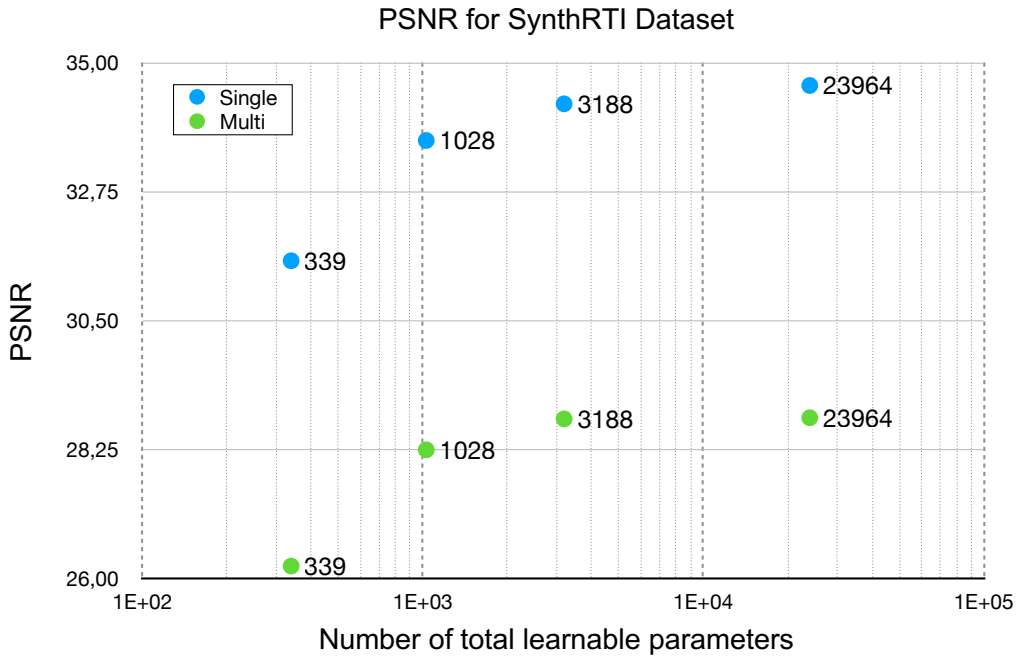


Figure 3.3: PSNR values for different number of network's parameters, tested on SynthRTI single and multi material settings.

3.2 Knowledge Distillation

Knowledge distillation [21] is a training technique which requires a pre-trained, larger model called teacher, and a smaller version of the same model to be trained, called student. The aim is to distill what the teacher model has learned into a compact representation of useful information, and transfer it to the student model. In the case of NeuralRTI, the teacher model is a version of the network trained to relight a specific dataset, while the student model is a version of the network with less parameters, which will learn to relight the same dataset. Reducing the number of parameters, especially in the decoder, allows to reach even higher rendering speed, and the way the student model is trained using the knowledge of the teacher one allows to preserve the relighting quality.

As in the original formulation, the training for the teacher model aims at minimizing the mean squared error between the predicted pixel and the ground truth pixel for each light direction of the

MLIC. After the training, the encoder is used to generates latent features images and is then discarded. The decoder, instead, is ready for generating relighted images. The student model is derived by simplifying the original network, specifically modifying only the decoder component,decreasing the number of parameters for each layer.

In our experiment, the teacher's decoder contains $N = 50$ parameters per layer. Referring to Equation 4.2, with decoder layers of this size, and a latent feature vector of size $K = 9$, the total number of parameters is 3303 (3200 wights and 103 biases). The student model has been tested with layers size set to $N = 20$, which leads to 723 total parameters (680 weights, 43 biases), and $N = 10$, which leads to 263 total parameters (240 weights, 23 biases). The speed up theoretically achieved theoretical is then significant, respectively almost 5 times and 10 times for these configurations. In both cases, the adjustments allowed smooth interactive relighting, and preserved relighting accuracy comparable to the original NeuralRTI. The student model is trained using a specific loss function, described in Equation 3.3.

$$L = \frac{1}{n}\sum_{k=1}^{n} \alpha||P_s - P_{gt}||_k^2 + (1-\alpha)||P_s - P_t||_k^2 \tag{3.3}$$

This function combines two different losses: the student loss, which measures the difference of the student's predictions (P_s) with respect to the ground truth pixel (P_{gt}), and the distillation loss, which measures the difference between the student's predictions and the teacher's predictions (P_t). The combination of the two losses is weighted by a parameter α. By minimizing the distillation loss during training, we help the student model to better reproduce predictions of the teacher model. This approach improves the learning phase of a small model, which would not reach the same performance if trained from scratch on the original data. In addition, the teacher's predictions can be smoother, less noisy, and contain richer information with respect to the target values of the original images.

Knowledge distillation is an intelligent way to train simple models, as the teacher can describe to the student how the underlying data is distributed, leading to better data fitting and generalization.

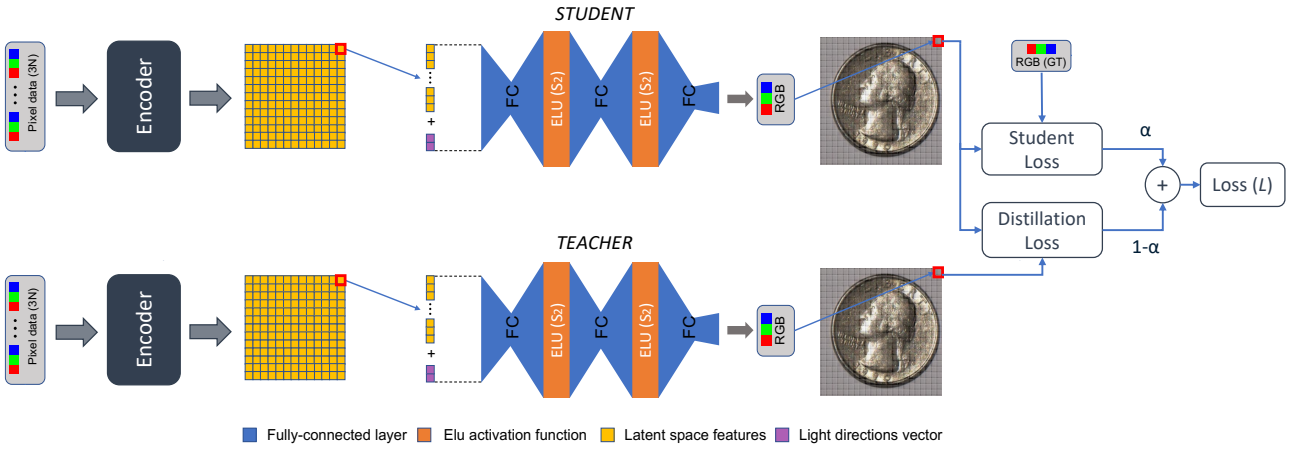


Figure 3.4: Knowledge distillation training scheme. The teacher model is pre-trained, and the student refines its training using the knowledge of the teacher.

3.2.1 Interactive performance

In Chapter 4 we will explain how NeuralRTI has been integrated into a web-based applications, a non-straightforward task due to the complexity of integrating neural models in already existing frameworks. The web application is a viewer for RTI images, built using an open-source library called OpenLIME, mentioned in Section 2.3 and later explained in Section 4.1.1. OpenLIME uses graphic shader to render the image, applying a specific set of functions to each pixel in parallel, allowing fast rendering on modern graphic cards. NeuralRTI is thought the same way of other RTI algorithms, and it also works pixel-by-pixel, thus making it easier to integrate it in this kind of applications.

Despite the use of graphic cards, the computation of the neural relighting, in a real-time interactive application, could be unfeasible on commercially available computers without dedicated GPU. Later in Section 4.3 we will see how a set of strategies has to be adopted to solve this issue, especially to visualize high-resolution images. Summarizing, we can say that the image is divided into tiles, and each tile is relighted independently. This way, only the visible tiles, present on screen at when relighting is required, must be rendered. Furthermore, the resolution of the visualized image is dynamically handled, so that when a change in the visualization occurs, e.g., relighting or zooming, the resolution is lowered to preserve a stable frame rate. When no changes occur, the image is shown again at full-resolution.

However, to compare the performance of the NeuralRTI obtained with knowledge distillation (Disk-NeuralRTI) against the original version, we need to deactivate all these optimizations, and calculate the rendering speed in terms of frames-per-second (fps), for the two cases. The fps measurement is an average value obtained after a series of continue relighting operations, performed in a time period of a few seconds. The evaluation has been done on a commercially available MacBook Pro from 2019. The specifications are: processor 1,4 GHz Intel Core i5 quad-core, graphic card Intel IrisPlus Graphics 645 1536 MB, RAM 8 GB 2133 MHz LPDDR3, operative system macOS Sonoma version 14.3.1 (23D60). Moreover, the evaluation depends also on the web-browser. We used Google Chrome (version 128.0.6613.138). Figure 3.5 shows the comparison between NeuralRTI with $N = 50$ parameters per decoder's layer, and Disk-NeuralRTI with $N = 20$ parameters per decoder's layer, tested on images with different number of pixels (1 million pixels is a 1000×1000 image, 10 millions pixels is a 5000×2000 image).

To summarize, from the network's architecture point of view, Disk-NeuralRTI is version of NeuralRTI with less parameter. Trained from scratch it would perform worse, but knowledge distillation allows to extract the information from a larger, pre-trained network in order to improve the quality of the training. In Chapter 4 we will see how the network was implemented for visualization, and it will be clear how reducing the number of network's parameters directly affect the relighting speed.

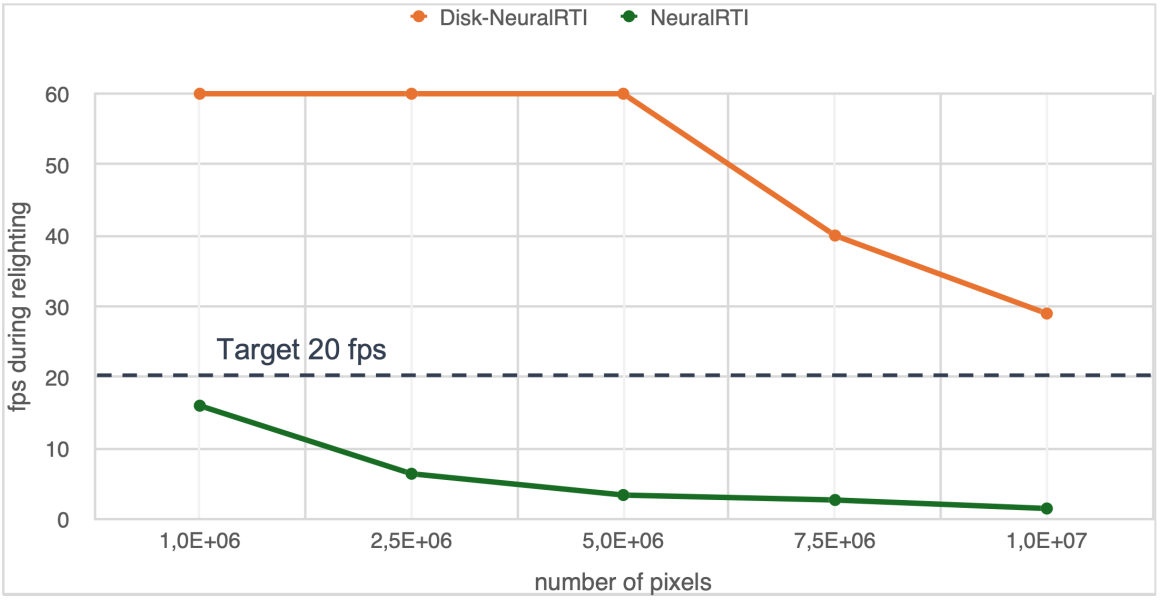


Figure 3.5: NeuralRTI (N = 50) vs Disk-NeuralRTI (N = 20). Frame rate measurement for varying number of pixels relighted at the same time.

3.3 Evaluation on ancient Roman Coins

In chapter 5 we will discuss practical use cases where RTI in general, both neural and classical, has been evaluated for the analysis of real-world objects. However, the first real-world use of NeuralRTI that we have done is a simple evaluation focused on the ability of the neural relighting model to reproduce real complex objects, against the performance of other relighting algorithms. The objects under study are ancient Roman coins, provided by the National Museum of Aquileia in Italy. The coins were studied by the Center for Cultural Heritage Technology of the Italian Institute of Technology (CCHT-IIT). MLICs of the coins have been acquired with a multi-spectral RTI dome, capturing the reflectance in the visible, infrared, and ultraviolet domains [13]. Three or four coins have been acquired for each scan, placed at a distance of approximately 5cm over a dark, matte background. This way, the light direction's variation inside the region of a single coin is small enough to make the assumption of directional light realistic. Using this assumption, it's possible to assign to each pixel of a single coin the same average light direction. Figure 3.6, which is taken from [1], shows a comparison of relighting obtained from NeuralRTI and HSH, with the ground truth image from the MLIC as a reference.

In chapter 5 we will see how the integration of NeuralRTI into OpenLIME, its user-friendliness and high level of customization, allowed us to carry out more developed evaluations, asking experts of the cultural heritage field to assess the validity of RTI algorithms for the analysis of the objects they study. Validating imaging algorithms on real-world objects is challenging due to the fact that we don't have precise knowledge about, for example, the object's material composition or geometry. These studies can be more easily motivated when there is a specific task to perform, like the identification of pigments in a painting, or of iconographic patterns in a textile. In our studies, we focused on

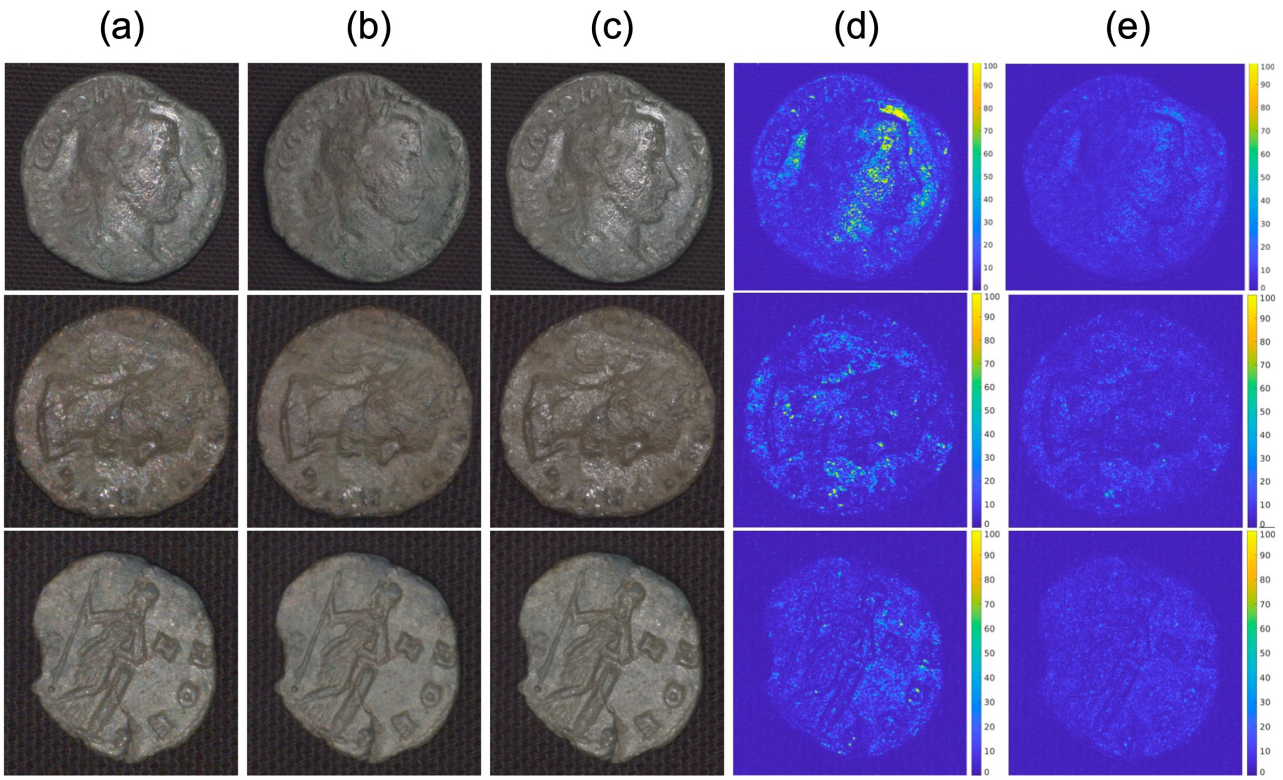


Figure 3.6: Top row: Sestertius (Roma, 243-4 CE), representing Emperor Godianus III. Middle row: Antoninianus coin of the Emperor Claudius II (Siscia, 268-70 CE). Bottom row: Antoninianus of Emperor Gallienus (Roma, 260-8 CE). Ground truth (a), HSH (b), NeuralRTI (c), HSH error (d) and NeuralRTI error (e). The error maps representing the RGB distance of pixel colors reveal a large improvement in the relighting quality, especially in highly reflective areas.

motivating the use of relighting algorithms and investigating their usefulness, always interacting with domain experts and asking their opinion.

3.4 Outcomes and discussion

The work on NeuralRTI showed how relighting algorithms still have room for improvements. The way we used NeuralRTI has been to exploit its capability of representing complex functions, making easier to encode a broad range of reflectance behaviors. This means that we always specialized the network to relight the specific dataset on which it was trained. We did not work to create a model with a broader scope, one that could incorporate different reflectance functions. In other words, NeuralRTI does not follow a classical deep learning path where the model is trained on a large dataset so that it becomes able to generalize on new, unseen datasets. For this reason, a future direction in the context of neural relighting models can be the development of such a model. However, the brute force approach of having a network that learns from a large number of images (MLICs) could be impractical on real-world data. Instead, a neural network could be used to create a new function able differentiate many reflectance behaviors, such as those of lambertian, specular or glossy materials.

In Table 3.2 the available models are summarized. The main difference between classical methods and neural-based ones is that the first are computed in close form, which means the relighting takes only one calculation. Neural models main drawback is the time they require to be generated and to calculate the relighted image. This becomes less of a problem on really powerful devices, but in our work we focused on optimizing performances on less powerful machines. Nonetheless, the improvements discussed in this chapter, and strategies that will be discussed in Chapter 4, highlights how it's possible to achieve good relighting quality and real-time speed in all our cases.

Model	Number of encoded features	Creation time	Relighting time
PTM	9/18	$\propto$ seconds	1 iteration
HSH	27/48	$\propto$ seconds	1 iteration
RBF-PCA	27	$\propto$ seconds	1 iteration
NeuralRTI	9	$\propto$ minutes	$\propto$ seconds
Disk-NeuralRTI	9	$\propto$ minutes	$\propto$ seconds

Table 3.2: Technical comparison of available relighting models. Classical models compute features in closed form, thus the creation is quick and only depends on image resolution, and the relighting only takes one iteration. The creation of neural models depends on image resolution, number of images and the data itself, while real-time relighting can be achieved on any machine by adopting rendering strategies to lower the resolution when neccesary.

For the classical algorithms, the implementation can be found in the RelightLab application (`https://vcg.isti.cnr.it/vcgtools/relight/`), mentioned in Section 2.3. RelightLab provides a graphical interface to make the creation of relighting models easier, but presents some limitations like accepting only JPEG images as input. For the neural models, a Python implementation is available here: `https://github.com/tgdulecha/NeuralRTI` and `https://github.com/tgdulecha/DiskNeuralRTI-code`. In this case, the network source code is available, making it less user-firenldy but allowing customization.

Chapter 4: WEB-Based Interactive Visualization of NeuralRTI

As discussed in Chapter 2, the main purpose of relighting models is to provide enhanced visualization of objects, especially in the context of cultural heritage. Interactive visualization using flexible relighting methods is widely used to support or replace physical artifact inspection for both experts and the public [11, 22]. In this chapter, we will see how we realized a WebGL-based integration of NeuralRTI for interactive relighting of stratigraphic data, supporting multiresolution and adaptive rendering, by interpolating the images at the network's latent space level. Figure 4.1 anticipates how NeuralRTI can be used inside OpenLIME, on different type of screens.

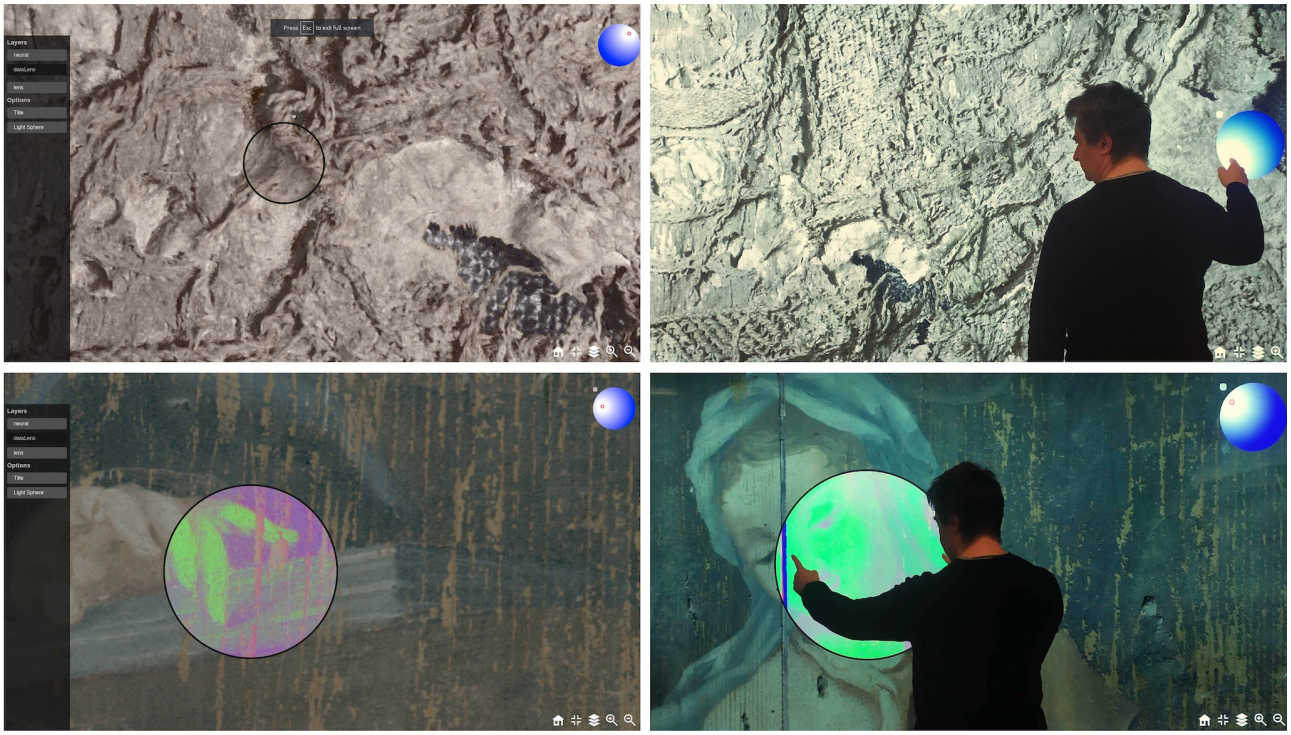


Figure 4.1: Left: interactive exploration on a 98-inch multi-touch display connected to a desktop PC (4K, NVIDIA RTX 2080 Ti graphics). Right: interactive exploration on a desktop monitor connected to a laptop with integrated graphics (Full HD, Intel Coffee Lake GT2 graphics). Top: textile from the Oseberg Find from a Viking Age burial mound at Oseberg in South Norway. Bottom: organ door with an announcing angel, 17th century, belonging to the Diocesan Museum of Vicenza and reconstructed from a handheld-light MLIC capture in collaboration with the Accademia delle Belle Arti of Verona.

4.1 Overview on relighting viewers

Relighting viewers offer an easy and effective way to inspect cultural artifacts by combining simple 2D navigation with virtual lighting control [23, 24]. Their success comes from practical setup advantages, reduced complexity compared to 3D viewers, and their similarity to traditional inspection methods [11, 25]. Relighting viewers can rely on either physically-based models, like BRDF, or reflectance fields, like RTI, but the more advanced methods are often limited by the need for precise object reconstruction. Moreover, even when a complex realistic model is available, it necessitates long run-time to take into account all illuminations effects.

For these reasons, most viewers use RTI methods that use multi-light image sets to encode reflectance data without separating shape and material, enabling flexible and efficient image generation. While not as detailed as fully physically-based models, RTI viewers offer realistic relighting and flexible lighting controls, including support for complex setups like multiple or localized light sources.

WEB-based relighting viewers has often used low-frequency models like PTM or HSH for simplicity and efficiency, but these methods can miss fine lighting details due to their limited accuracy [11, 26]. Relighting methods based of neural models, like NeuralRTI, offer better image quality but are hard to use in interactive viewers due to high computational demands and integration complexity, especially in web environments.

4.1.1 OpenLIME

A legacy of software has been available through the years to allow users to create RTI models. However, these software have always been limited to use basic relighting models, and a small set of enhancement options. Nowadays, the focus is shifting towards web-based applications, as they allow strong customization and they are easy to distributed to the public. A web-based application can be seen as a web page that runs a specific software, thus it can be made available online, and opened by any user who has a browser.

OpenLIME [4] `https://github.com/cnr-isti-vclab/openlime` is an open-source library, completely written in pure JavaScript, developed to create online viewers of RTI images. It was published by the Visual Computing Lab `https://vcg.isti.cnr.it/` and the Visual and Data-intensive Computing Group of the CRS4 group `https://www.crs4.it/en/`. OpenLIME allows dynamic resizing of high-resolution images using a tiling technique. Moreover, it is structured as a multi-layer visualization system. For example, if a MLIC of an object is acquired both in the visible and ultraviolet spectrum, then content of the two relightable images can be seen together, using opacity and blending modes, or interactive lenses. OpenLIME is distributed as an open-source software, under the GNU General Public License version 3. OpenLIME has already integrated tools for classical RTI

models (Polynomial Texture Map, HemiSpherical Harmonics, Radial Basis Function). During the project, our challenge was to integrate NeuralRTI into the same system. This has been a particularly challenging as it meant to adapt something that usually requires many parameters and calculations (neural network) inside a context that works at its best with light data and few required computations (real-time image rendering).

4.2 Neural relighting in the WEB

When coding neural networks, a set of libraries are used to define the structure, parameters initialization, and training of the network at a high level. For example, NeuralRTI is implemented in Python, using the TensorFlow library. On the web, a JavaScript implementation of TensorFlow is available, namely TensorFlow.js [27], so using this library looked like the right way to transport the network in the web [28]. However, we faced problems that were hard to solve. For example, the relighted image eventually was rendered, but with the wrong colors, probably due to texture's format that the library uses for computing calculation, incorrect for displaying images. In general, this might be because machine learning libraries are designed to speed up mathematical operations, not for graphics. On the other hand, to access the GPU on the web, the WebGL open-source library is used, in order to parallelize pixel-wise operations. WebGL is a library thought for graphics, not for machine learning models. Then, using TensorFlow in the web might also mean that a lot of intermediate steps are performed to adapt WebGL to the network structure. This may change with the recent advent of WebGPU, a new library which is primarily thought to work with machine or deep learning models in the web, making a better use of modern GPUs architecture.

However, we decided to resort to a more straightforward solution. In our case, the main advantage is that the network is rather simple, and can be decomposed in its basic mathematical operations and manually coded inside what is called a WebGL shader. The shader is a piece of program written in GLSL, a dedicated language to write operations that are computed on the graphic card. The main program, written in JavaScript, calls the shader each time the image has to be rendered again, applying the defined operations simultaneously to each pixel (parallelization). NeuralRTI is a MLP, thus each layer receive a vector as input, it multiplies this vector by a matrix of weights (dot product), it adds a vector of biases to the result, and it applies a non-linear activation function to the overall output. The general formula is the following:

$$x_{out} = f_A(x_{in} \cdot W + B) \tag{4.1}$$

where x_{in} is the input vector, x_{out} is the output vector, W is the weights matrix, B is the biases vector, and $f_A(x)$ is the activation function, which adds non-linearity to the model. From the code point of view, the network's operations are implemented as a sequence of for loops, dot products, sums, and activation functions (only the decoder, as the encoder is discarded after training). It must be noted that,

implementing the neural decoding this way, we cannot transmit an unlimited number of parameters to the shader, thus the decoder's size is limited by the WebGL specifications. However, the simplifications that we performed to the original model, discussed in Chapter 3, allowed us to store weights and biases of the decoder in a separate file and give them altogether to the shader.

Taking Figure 3.2 as a reference, NeuralRTI is composed of a encoder-decoder structure. The encoder is used to extract a set of features from the input images, outputting a much lower amount of data, therefore applying a compression operation. In a general scenario, the number of input images is $N = 50$, thus for every pixel the encoder receives $3N = 150$ values, and outputs $K = 9$ values, reducing the required memory storage of more than one order of magnitude. On the other side, the decoder takes the features, derived from the encoding, and concatenates them with a given light direction, to produce pixel-by-pixel the relighted image corresponding to the given light direction. The network is trained end-to-end on all the pixels (or a random subset) of the MLIC, and it learns how to reproduce the color of each pixel, by minimizing a loss measuring the difference between the predicted and the ground truth pixel's color.

After training, the network is set in what is called evaluation mode, i.e., the network's parameters stop learning, and the encoder is used one more time to encode the MLIC and extract the final version of the features. Afterwards, the encoder can be discarded, while extracted features and decoder have to be stored somehow. For example, the features can be saved as images, using image formats like PNG or JPEG, while the decoder's parameters can be extracted and saved in a binary file.

As already discussed, the main issue of using NeuralRTI for interactive relighting is the the elevated cost in terms of operations the decoder has to perform, which may not be suitable for real-time rendering. Equation 3.2 indicates how to calculate the total number of parameters (weights and biases) contained in the decoder. The more parameters, the heavier the relighting operation. The equation can be generalized as follows:

$$\begin{aligned} W &= (K+2) \times S + (S \times S) * (L-1) + S \times 3 \\ B &= L * S + 3 \end{aligned} \tag{4.2}$$

where K is the size of the decoder's input vector, S is the size of the layers, and L is the number of layers.

In the original version, NeuralRTI needed to perform more than 45,000 operations, per pixel. Therefore, even parallelizing the computation on every pixel, which can be easily done on modern graphic cards, the amount of computation is still demanding. However, the simplifications and tests discussed in Chapter 3 allowed us to lighten the decoding, while preserving the quality of relighted images. In a first approach, by setting the decoder with a number of parameters per layer equal to $S = 50$, and keeping the number of features equal to $K = 9$, the number of operations required for the relighting operation is slightly more than 3000.

The decoder's parameters are then extracted and stored in a binary file, specifically a file in JSON format for our web-based application. Later on, we will better explain how the software works at code level. The faetures, instead, are stored as images with the same resolution as the input images. This comes particularly in handy for web-based applications, which are optimized to upload and work with image formats. However, the raw features have to be first rescaled, in order to be saved as images. Moreover, we need to introduce a quantization step, as the features are naturally floating point numbers with 32 bits, but they have to be saved as 8 bit images. Setting the number of features equal to $K = 9$, as it was in the original version of NeuralRTI, has been proved to generate high quality relighted images, better than classical methods, using a very compact number of features, where $3^{\mathbf{rd}}$-order HSH, for example, would use $K = 48$. Besides, a lower number of features leads to a lighter memory usage when running the application, important especially if high-resolution images are used. In this case, $K = 9$ means that the features will be split in three different images (every image has three channels), rather than twelve images in the $3^{\mathbf{rd}}$-order HSH case.

4.2.1 Shader implementation

To render relighted images in the web, we implemented a custom shader, using WebGL 2 and the GLSL programming language. A shader is a piece of code, written in GLSL, which is called by the main program, written in JavaScript, at rendering time. This allowed us to avoid using external libraries or dependencies, and made easier to integrate the neural relighting operation into the applications built with OpenLIME. The shader has to perform the following operations: use pixel coordinates to sample the features, which are stored as images; concatenate each pixel with the given light direction; propagate the input value so obtained through the network, so to output the final color. Each layer has to compute its own output using dot product and sum, and applying the activation function (ELU in Equation 3.1). The final layer has no activation function.

The features outputted by the encoder are generated as a matrix of size $H \times W \times K$, where $H \times W$ is the resolution of the input images and K is the number of features saved for each pixel. As already mentioned, we store in a series of RGB images, which facilitates the uploading in a web application. Therefore, the matrix is split into sub-matrices with dimensions $H \times W \times 3$, and values are quantized from 32 to 8 bits. For each channel of the K features, a maximum and a minimum value are saved in the JSON file, in order to rescale the features between their original values and the interval $[0, 255]$, used for 8 bit image storage. In Figure 4.2 an example of how the features images may look like is presented. The errors introduced by this approach can be of two different types: quantization and compression. The first derives from converting data from 32 bits to 8 bits. The second occurs only if lossy image formats like JPEG are used. Both these situations can be addressed with further experiments by introducing the quantization or the compression at the training stage, so that the network learns how to relight using already quantized or compressed features. Later on, we will see that the features

images are further split into tiles, in order to build a multi-resolution approach, which enables adaptive rendering to relight only the portion of the image that is visualized on the screen, rather than the full image.

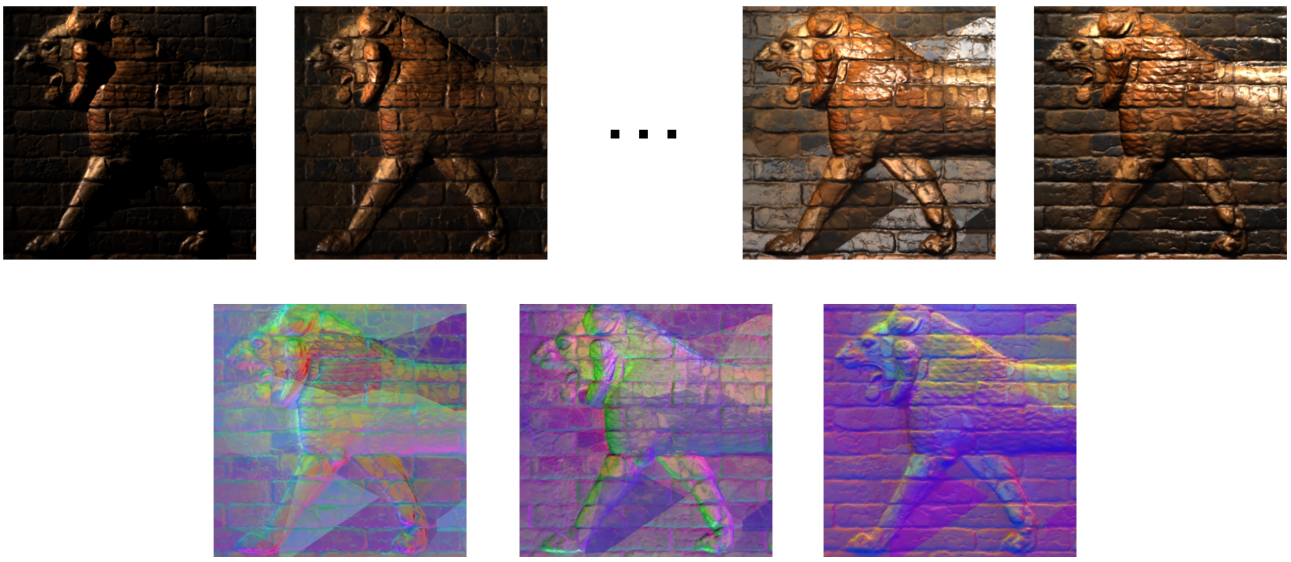


Figure 4.2: Top row: examples from the MLIC. Bottom row: the encoded features are quantized and stored in three parallel RGB images. The example MLIC is taken from SynthPS (`https://github.com/Univr-RTI/SynthPS`).

The first rendering step, that the shader has to perform, is to sample the images containing the features, and for each pixel coordinate the values are stored in a vector of size $K+2$, where the last two components are the light direction given to the shader as a uniform variable. In WebGL, uniform variables are values given to the shader from the main program, which are the same for all the pixels. Therefore, the light direction is the same for all the pixels. Alternatively, it is theoretically possible to load an additional map containing different light directions for each pixel.

For the network configurations we tested, the storage cost required to to transmit all the network's parameters was typically less than 3.5K floating point values, calculated using Equation 4.2. Therefore, it has been possible to store the parameters in uniform variable arrays of type `vec4`. The type `vec4` is a specific type of GLSL, which defines variables as 4-elements vectors, and has been implemented to optimize calculations between vectors of this specific dimension. This type is particularly useful when working with images, as the color information is stored as a 4-element vector containing red, green, blue for colors, and alpha for transparency. WebGL allows to transmit a limited amount of data to the shader, so we exploited this built-in structure to load the network's parameters in a compact way. Moreover, this way we can use vectorization to compute more operations at the same time. Given the same network architecture, i.e., the one based on NeuralRTI, we can handle any network configuration, with different number of layers or parameters, as each array can be padded with zeros when its size is not a multiple of four. Also the input vector, which has size $K+2$, is saved in an array of `vec4` variables and padded with zeros if necessary.

4.3 Data optimization

To handle large image datasets efficiently, it's possible to resort to a method that involves pre-processing the data and storing it in a pyramid-like structure. This means the data is repeatedly filtered and scaled down by half, creating multiple levels of detail. Each level is then divided into small tiles, which makes possible to explore very large datasets using limited computing power. By matching the tile size to the screen resolution, we can render images with just one sample per pixel, and the number of tiles needed depends only on the screen size. This technique was first used for viewing large images but is now commonly used for relighting models. Creating these levels of detail involves filtering high-resolution data to make lower-resolution versions and blending between them to allow smooth transitions. Without this approach, the resolution could be changed only with powers of two factors.

In the case of neural relighting, we can build this pyramid either by zooming the images first and then encoding each tile with the neural model, or by encoding only the high-resolution version and applying the tiling technique to the features images outputted by the encoder. The second approach is easier and more efficient, since we use the same code for building the pyramid and only change the pixel decoder. In the case of classical relighting models like PTM or HSH, which are linear models, filtering the relighted image (output of the model) or the features images (input of the model) leads to the same result. For neural models, instead, which include nonlinear operations, filtering the features images could lead to a different result. However, we will show how, in the case of NeuralRTI, the error introduced by the filtering operation, applied when the features images are resized and split into tiles, can be considered negligible.

When visualizing data on the web, compression is very important to reduce the amount of data that needs to be uploaded online. In our case, most of the data being transmitted comes from the features images, which store per-pixel information. The neural network, i.e., its parameters, is shared across the whole dataset and is very light, just a few kilobytes. To compress the features for using them in web browsers, we rely on standard image formats like PNG, JPEG, or WebP, since they are supported by all major browsers. As visible in Figure 4.2, the features images look similar to regular RGB images, so JPEG works well for compressing them efficiently. However, JPEG is normally optimized for human vision, thus it treats color and brightness differently and assumes that the RGB channels are related. The neural data doesn't follow these patterns, so we adjust the JPEG settings by turning off the conversion to YUV color space and chroma sub-sampling, and using custom quantization tables that don't introduce bias. This helps preserve the quality of the neural data during compression.

In our case, the tiles were generated using the *vips* library (`https://github.com/libvips/libvips`), the *Deep Zoom* method (from Microsoft) and a tile size of 256 pixels for each side. For a single image named "image.jpg" the following shell command can be used:

```
vips dzsave image.jpg --layout dz --tile-size 256
--overlap 0 --depth onetile --suffix .jpg[Q=95];
```

4.3.1 Adaptive rendering approach

Running neural network-based relighting interactively can be demanding, especially on web platforms where devices vary widely in performance. To keep things fast and responsive, we optimized our server to meet time limits. The most expensive part of the process is calculating the final color from the combination of features and light direction. Therefore, we improved speed by limiting how many pixels are relighted at once. These optimizations use extra memory (like a framebuffer) to pre-render tiles before showing them on screen.

The first optimization takes advantage of the fact that shading under direct light doesn't change when the camera moves. This means we can reuse already-shaded tiles when the user pans the view, and only compute new tiles that come into view.

The second optimization focuses on performance during time-critical tasks. We measure how long it takes to relight a few tiles, then calculate how many pixels can be processed per second. Based on a target frame rate (like 30 fps), we set a pixel budget for each frame. We then adjust the resolution of intermediate textures so that it fits within this budget. For each pixel, we use the closest available level of detail that meets the resolution requirement, or the highest available one if none match. We use bilinear filtering to sample the data.

As discussed earlier, for older models like PTM and HSH, image filters work well because their math is linear. NeuralRTI isn't linear, but we found that averaging spatially-close features still gives good results. We ran extensive tests and found that relighting at a lower resolution and then up-scaling gives results very similar to relighting at full resolution and then down-scaling. The difference is small and doesn't affect real-world use, especially when compared to errors from capturing ground truth images or simulating light movement. Visually, both methods look the same, with no noticeable issues at material edges.

To test how much error is introduced when using adaptive rendering in the at the features level, we compared images relighted from full-resolution features images with those relighted from the down-scaled version, resizing the images to 50% of their original size along both height and width. We also compared these with ground truth images that were similarly down-scaled. The tests were performed on the SynthRTI dataset, which includes surfaces with different shapes and materials that change clearly in specific areas. The dataset contains three surfaces with varying geometric complexity, each textured with regions made of different materials and combinations of material types.

The visual errors caused by filtering and interpolation should be most noticeable in areas where material properties change suddenly. In some examples from the SynthRTI (`https://github.com/Univr-RTI/SynthRTI`) dataset, the surface is divided into patches with different BRDF settings.

One of the materials combination presents sharp changes in color and roughness along horizontal lines, and a vertical transition from dielectric to metallic materials in the center. One example can be seen in Figure 4.3, which shows one image from the MLIC (ground truth), the two relighted versions resampling either the input or the output of the decoder, and corresponding ꟻLIP images and histograms to show the amount of error introduced.

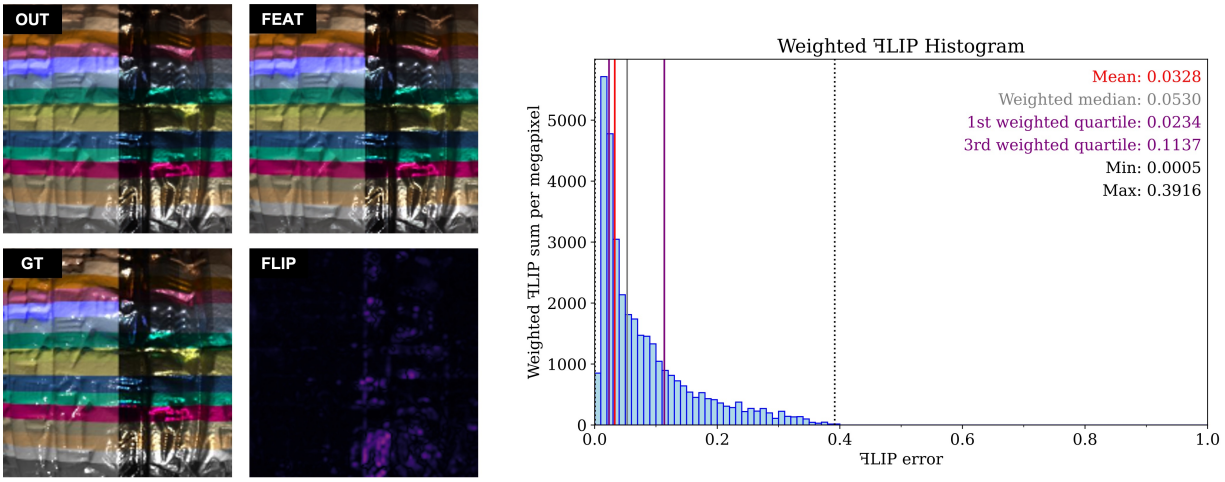


Figure 4.3: Image obtained by resampling the relighted image at full resolution (OUT). Image obtained by relighting after resampling the features (FEAT). Resampled ground truth image from the MLIC (GT). All three images are from the same light direction. ꟻLIP error map (FLIP) and corresponding histogram between (OUT) and (FEAT). Images have been resampled of a factor 0.5 both in height and width. Original resolution: 320×320 pixels.

In addition to the visual cue, Table 4.1 shows the average ꟻLIP differences for the three objects in the dataset, comparing two relighting methods applied to downsampled images: one using coefficient interpolation and the other using RGB interpolation. These differences, measured across various lighting directions and material combinations, are generally small in both average and median values. To see why these differences are negligible, we can compare them with the values in Table 4.2 and Table 4.3. These tables show how each relighting method compares to the ground truth images, which were also downsampled using linear interpolation. The results clearly show that the error introduced by coefficient interpolation is minimal compared to the overall error from the relighting models themselves. Additionally, the values in both tables are nearly identical, confirming that coefficient interpolation does not cause any noticeable loss in quality.

Object	Mean	Median	1st quartile	3rd quartile	Min	Max
1	0.012	0.015	0.011	0.023	0.000	0.097
2	0.030	0.044	0.024	0.083	0.001	0.360
3	0.028	0.042	0.021	0.086	0.001	0.345
Average	0.024	0.034	0.019	0.064	0.000	0.267

Table 4.1: ꟻLIP difference between resampling at the features level vs resampling at the output level. Average values calculated on the SynthRTI dataset.

Object	Mean	Median	1st quartile	3rd quartile	Min	Max
1	0.091	0.120	0.081	0.170	0.005	0.325
2	0.158	0.209	0.127	0.341	0.004	0.901
3	0.125	0.161	0.100	0.264	0.002	0.789
Average	0.125	0.164	0.103	0.258	0.004	0.671

Table 4.2: ΔLIP difference between resampling at the features level vs resampling the ground truth image. Average values calculated on the SynthRTI dataset.

Object	Mean	Median	1st quartile	3rd quartile	Min	Max
1	0.089	0.120	0.081	0.169	0.005	0.321
2	0.158	0.210	0.127	0.341	0.004	0.900
3	0.124	0.160	0.099	0.263	0.003	0.786
Average	0.124	0.163	0.102	0.258	0.004	0.669

Table 4.3: ΔLIP difference between resampling at the output level vs resampling the ground truth image. Average values calculated on the SynthRTI dataset.

4.4 Outcomes and discussion

The main achievement of working with OpenLIME has been the integration of NeuralRTI. The use of a web viewer facilitates significantly sharing it to a wide number of different users. In our implementation, the main limitations is given by the use of the WebGL library: in the browser, WebGL takes care of graphics and rendering the images. When the graphic shader is created, WebGL allows a maximum number of parameters to be loaded externally. With our network's implementation we were close to this limit. Larger networks could not be loaded using the same approach. This can change using different solutions. For example, the WebGPU technology should be able to solve this problem because it's been created to properly work with modern GPUs and deep learning libraries. However, our implementation was meant to work also on a broad range of less powerful machines.

In addition to technical improvements, using NueralRTI in the web raised many issues that were not completely solved. For example, the use of image formats to save the features, the features quantization to 8 bits, are all problems that can be addressed in future works by taking them into account when designing and training the network. For now, we limited to assess the error introduced by features resampling, that happens when using the tiling approach for adaptive rendering, which qualitatively appeared to be negligible. This shows that NeuralRTI, despite being a non-linear model, does not introduce a huge amount of non-linearity. We encode the images pixel-by-pixel and the feature images still resemble the original object. This could be different with other models, where the feature space (latent space) could have a much more complex representation.

Chapter 5: Real-World Use Case Evaluations

This chapter addresses the use and evaluation of different RTI techniques for practical use cases. During the project, it has been possible to talk directly to cultural heritage experts and conservators, seeing their ongoing research and having the possibility to test the efficacy of RTI to improve their research. The two experiments present objects with different properties: fragmented textiles, extremely ruined, coming from the Viking age but with unverified timeline, and ancient palimpsests, i.e., old manuscripts where old text has been replaced by new text, and where chemical reagents, among other reasons, degraded the readability of the text.

OpenLime is a great tool for visualization tasks and creating case studies. Its original interface lets users zoom in and rotate RTI images, a straightforward feature that cultural heritage professionals find useful. However, for this tasks, especially the one about Viking textiles, we added support for displaying multiple relightable layers either blended together or shown side-by-side on the same canvas. This way, users can now compare how the same object looks when processed with different RTI encodings or image enhancement techniques. Another use case is viewing several objects at once, each of which can be moved and rotated independently. This is especially helpful for experts who need to align fragments interactively and relight them in a consistent way.

A global rotation can also be applied to all objects at once. We've added the option to use a mask for background removal, although the mask needs to be pre-generated. Figure 5.1 shows an example of viewing multiple objects or different areas of the same textile fragment with individual rotations. A button in the interface controls the global rotation, while each layer has its own controller that activates when the mouse hovers over it. To move an image, users can click and drag a layer; to rotate it, they simply scroll the mouse wheel. Light direction is managed globally through the light control (light rays), and relighting is adjusted to remain consistent within the visualization canvas's reference system.

5.1 Use of NeuralRTI and OpenLIME for analyzing Viking textiles

The first real-world application presented is both a comparison between NerualRTI and the PTM algorithm, and an evaluation of the usability of OpenLIME with respect to its predecessor RTIViewer. The objects under study are fragments from the Oseberg findings [29–31]. These tapestries are highly fragmented, and research is needed to identify matching fragments and virtually reconstruct them. The

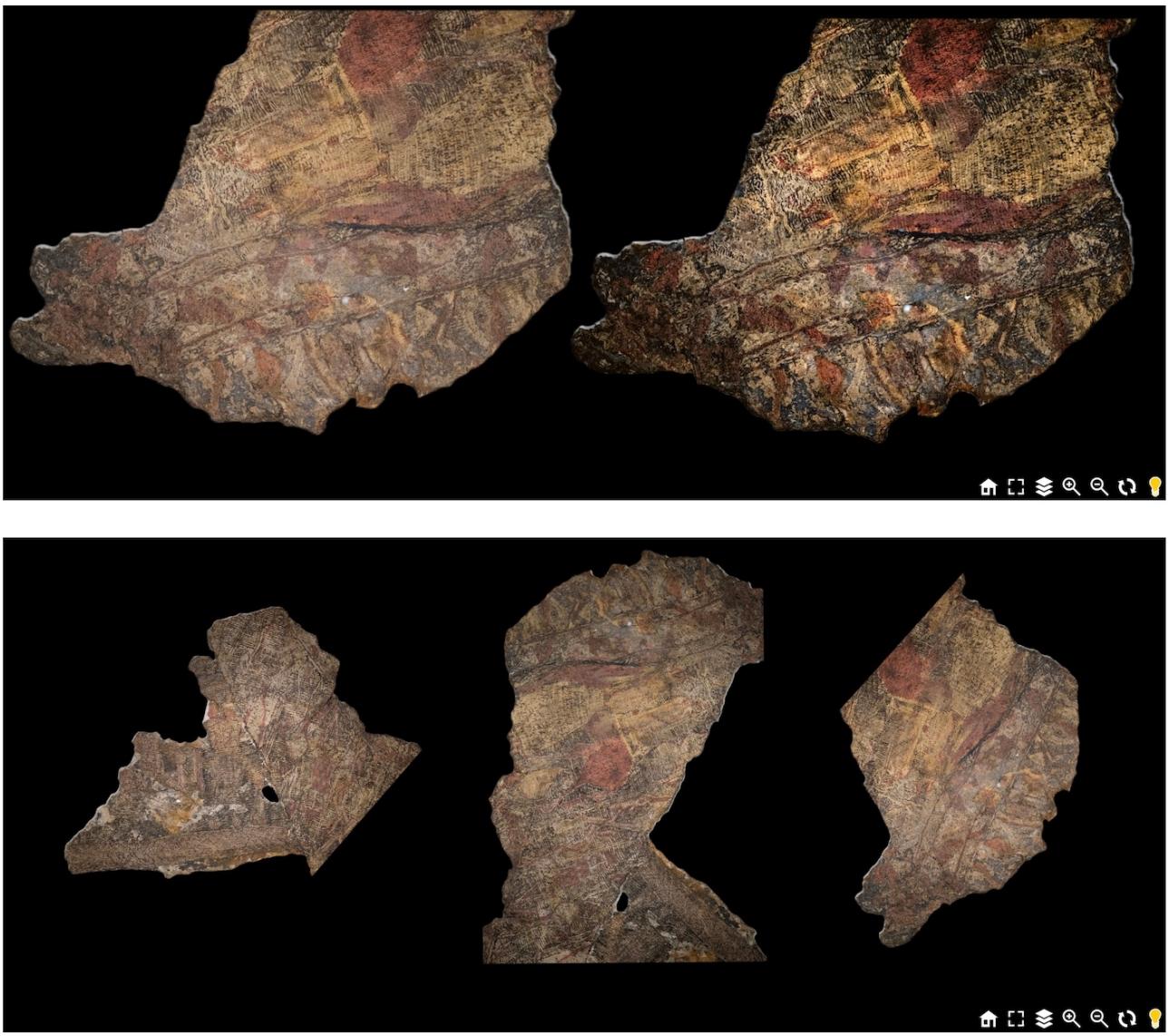


Figure 5.1: Example of multiple objects view. Top: comparison between two different visualizations of the same fragment. Bottom: independently rotated fragments.

real fragments are extremely fragile, so interaction with them has to be limited to preserve the state of conservation. For this reason, digital visualization of the fragments is preferred. Recent work has described the limitations of color images for this objective and proposed RTI as one of the alternative solutions [32]. The main tasks that an conservator wants to identify have been classified as: detect figures motifs, useful to identify semantic and stylistic similarities among the fragments; count threads, as the number of threads per centimeter can be one of the similarity criteria; segment the object by separating the object's external outline from the background and identifying holes and missing parts, in order to make the use of an identification algorithm more robust and noise-free.

This work as been carried out in collaboration with the Norwegian University of Science and Technology (NTNU), where an expert in conservation, actively working on the fragments reconstruction, performed the visualization tool evaluation. The evaluation was conducted on a color-calibrated 15.8-inch LCD display with a resolution of 3840×2400 pixel and following parameters: sRGB, D65

White Point (x=0.313; y=0.329), Gamma=2.2, Peak luminance = 80 cd/m^2.

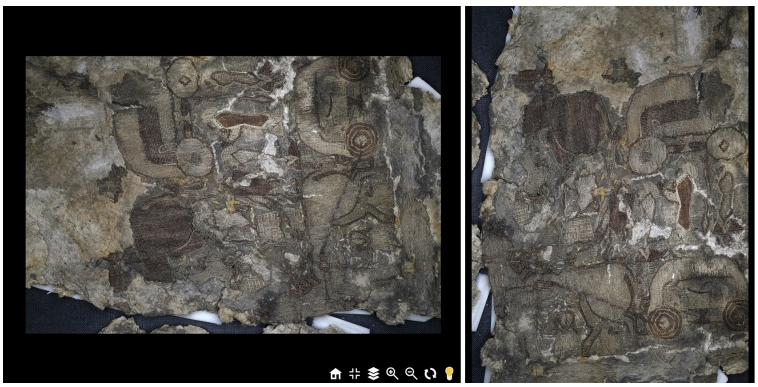
5.1.1 Comparing RTIViewer and OpenLIME

As mentioned in Section 2.3, RTIViewer is the traditional software that cultural heritage experts have used to visualize RTI images, but nowadays it presents limitations. During this evaluation we collected feedback about the user experience while using OpenLIME from a user-friendliness point of view. OpenLIME stands out for its interactivity and practical features. One key advantage is the ability to rotate fragments, which is crucial for correctly interpreting motifs and figures, and preventing users from having to tilt their head, which would be quite inconvenient (see Figure 5.2a). Another useful feature is the ability to load multiple layers, allowing users to switch between different relighting versions of the same fragment. This helps in choosing the most suitable relighting method for a specific task or fragment. Additionally, by displaying layers side by side on the same canvas, either different versions of the same fragment or entirely different fragments, it is easier to compare fragments and find matches (see Figures 5.2b and 5.2c). Users can select which fragments to display directly from the graphical user interface (GUI). Finally, OpenLIME includes a feature to control and visualize the direction of illumination, alongside the standard light sphere already implemented in previous viewers. OpenLIME provides dynamic light rays overlaid on the fragment, helping users choose optimal lighting angles without being distracted by the light sphere, which is usually shown on the side of the screen (see Figure 5.2d). Other helpful features include a Home button to reset the view and zoom in/out buttons in the GUI.

5.1.2 Comparing PTM and NeuralRTI

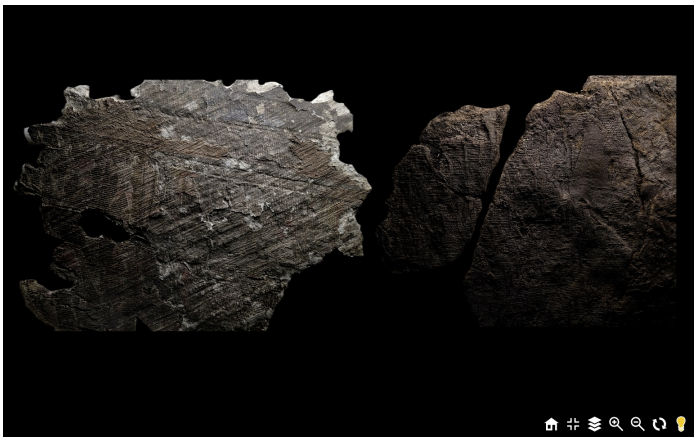
On a more technical aspect, the relighting capabilities of NeuralRTI has been compared to those of the standard of classical algorithms, namely PTM. When viewed under overhead lighting, which is typically used for photographing such artifacts, neural-relighted images appear less colorful and more faded compared to those processed with PTM (see Figures 5.3b and 5.3a). However, it's worth noting that neural relighting makes it easier to segment holes and gaps within the object, which is important for classification and clustering tasks (Figure 5.3d). Color plays a key role in identifying figures and classifying textures, but in heritage artifacts that have suffered significant degradation, color can be unreliable. When color is used, it's usually based on standard color images rather than RTI data [30, 32].

RTI was recommended in the literature for this type of case study mainly because it captures 2.5D surface topography, which is more reliable than color in this kind of degraded artifacts. The goal is to identify thread areas that rise above the background and to separate the object from the canvas. For this, fragments should be observed under grazing angle illumination, where raised parts begin to cast

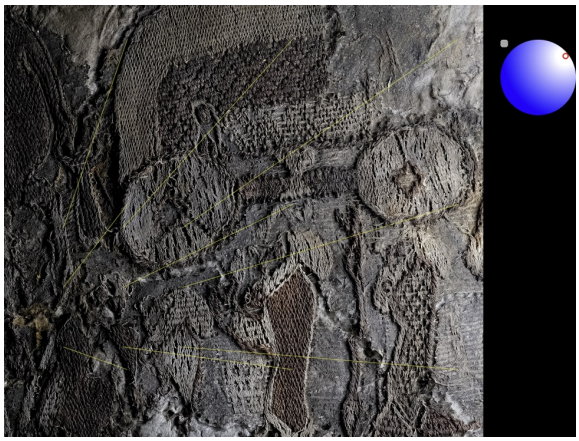


(a) Rotation

(b) Comparing versions



(c) Multiple fragments



(d) Illumination rays

Figure 5.2: Series of utilities included with OpenLIME. The most important aspect is the possibility of customize the viewer's source code, which allowed to easily introduce the possibility to visualize multiple fragments at the same time.

shadows (see Figure 5.4). Both PTM (Figure 5.4a) and Neural (Figure 5.4b) relighting enhance the visibility of the figure, but Neural relighting offers higher contrast and sharper shadows on elevated areas. In Figure 5.4c, cracks and concave regions appear lighter in PTM, resulting in lower contrast with the surrounding surface.

This has important implications for thread counting, which experts use to compare weaving techniques and identify matching fragments, a process still done manually. Figure 5.5 shows that threads are less visible under overhead lighting, while grazing angle illumination significantly improves contrast. This higher contrast makes manual thread counting easier and slightly more accurate with Neural relighting (although figures still appear less reddish). Recent research has explored automated thread counting by mimicking human behavior through pulse-counting, where contrast is also a key factor [33]. Therefore, Neural relighting can support both manual analysis and computational approaches.

An important point to explore is how contrast differences influence segmentation. In Figure 5.6, we see a fragment illuminated from a grazing angle. Neural relighting, in this context, produces sharper and more extensive shadows along the object's boundaries. At the same time, it brightens the surrounding background. This contrast makes surface variations, whether in 2.5D or 3D, much easier

to detect. As a result, both the outer contours of the object and its internal features, such as holes and gaps, become more distinguishable and easier to segment.

(a) PTM overhead *(b) Neural overhead* *(c) PTM - Neural* *(d) Neural - PTM*

Figure 5.3: Comparison for overhead illumination (a,b). PTM better retains chromatic appearance. Subtracting channel-wise Neural from PTM shows a stronger reddish channel in PTM (c), while Neural is usually lighter in the holes where the background canvas is to be segmented (d).

(a) PTM grazing angle *(b) Neural grazing angle* *(c) Pixel-wise difference*

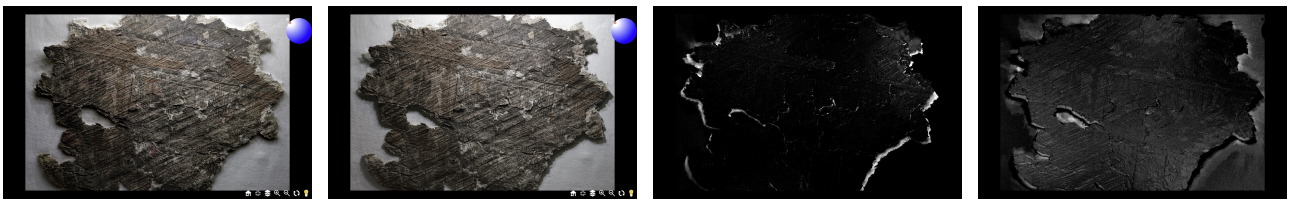
Figure 5.4: Comparison for grazing illumination (a,b). Neural seems to better reproduce shadows casted by elevated figures and threads. PTM is found to be lighter in the cracks and concavities, visible by subtracting Neural from PTM in grayscale, where negative numbers were set to 0 (c).

(a) PTM overhead *(b) Neural overhead* *(c) PTM grazing* *(d) Neural grazing*

Figure 5.5: Overhead illumination makes 3D structure less noticeable, and more difficult to count threads. PTM preserves reddish chromatic components (a), while Neural looks more faded (b). For grazing illumination, the thread structures is clearly visible, and Neural offers slightly higher contrast (d) than PTM (c).

5.1.3 Evaluation outcome

This qualitative case study was conducted by an expert in virtual reconstruction. Compared to RTIViewer, the OpenLIME visualization tool offers several practical advantages, including interactive features like rotation and the ability to handle multiple layers. These capabilities proved useful during



(a) PTM grazing *(b) Neural grazing* *(c) PTM - Neural* *(d) Nueral - PTM*

Figure 5.6: Grazing illuminations (a,b) and grayscale difference maps, where negatives values were set to 0 (c,d).

the comparison of PTM and Neural relighting methods, specifically for reconstructing archaeological tapestry artifacts. PTM tends to preserve color more effectively, while Neural relighting produces images that appear more faded or achromatic. However, Neural relighting excels in contrast and shadow definition, especially under grazing angle illumination. These qualities enhance the visibility of surface topography and support tasks such as thread counting, segmentation, and texture classification. For cultural heritage applications, this makes Neural relighting a promising direction to explore. Looking ahead, the next step would be improving contrast further using image enhancement techniques like Retinex or STRESS [34]. In the following case, discussed in Section 5.2, we will see a general evaluation on the use of RTI techniques for analyzing old manuscripts, where also the use of STRESS has been evaluated.

5.2 Evaluation of RTI techniques for enhancing the visualization of ancient Manuscripts

The second real-world application concerns a broad analysis on the usefulness of RTI, specifically to analyze ancient palimpsests. In recent years, the digitization of cultural heritage objects has gained increasing importance, driven by the growing demand for accessible cultural content through web-based platforms. When developing an online application for visualizing and interacting with cultural heritage images, especially one intended for public use, it is essential to follow established guidelines. These ensure accurate representation of the artifacts, provide relevant contextual information, and make the experience accessible to all users, including those with physical limitations. That said, the primary focus of this evaluation is not on public dissemination but rather on evaluating the effectiveness of specific algorithms within a defined research context, i.e., the application of RTI techniques to the analysis of palimpsests.

RTI has become a well-established technique, and numerous studies have evaluated the performance of different algorithms using quantitative metrics such as PSNR, SSIM or ꟻLIP [35, 36]. However, relatively few works have focused on assessing these methods within specific application contexts. Therefore, we carried out a study to address this gap, by exploring how various relighting techniques

perform in the analysis of ancient palimpsests. While using the application, a pool of ten participants, experts of the field, completed a questionnaire in which they described the manuscript features they were examining, such as text or ruling, and provided both quantitative ratings and qualitative feedback on the different algorithms.

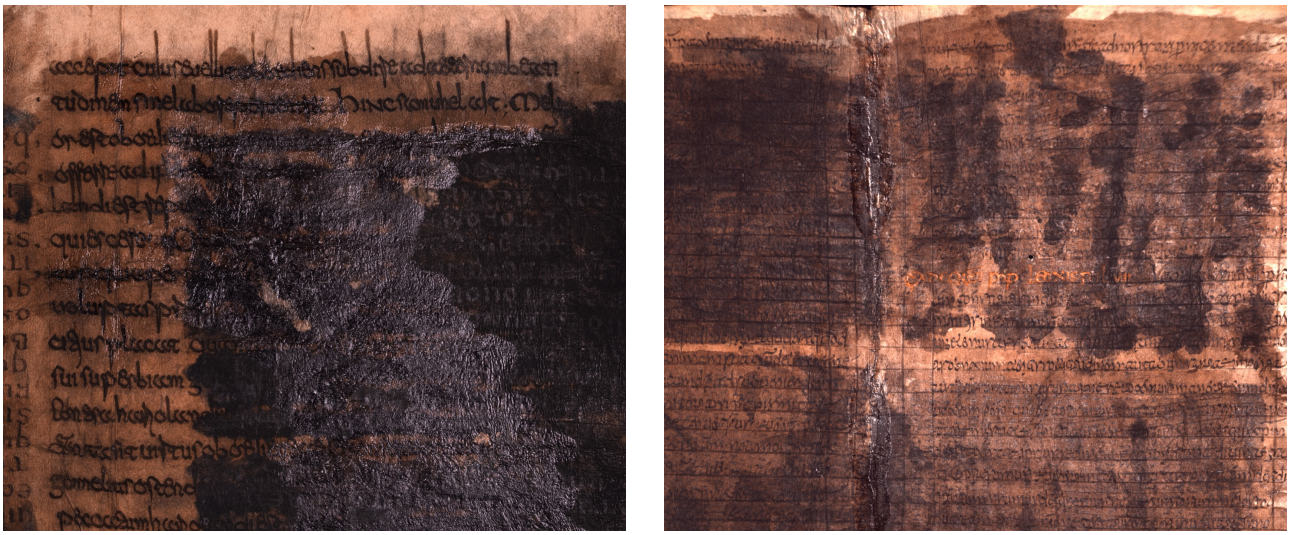
The evaluation of RTI algorithms and applications has been explored during the years. Various algorithms have been applied to enhance relightable images in real time, with their performance evaluated in previous studies [37]. RTI has found applications across a wide range of object types, including textiles, books, coins, painted surfaces, inscriptions, and fossils [1, 38–43]. Although less common, some works have explored spectral RTI, which adds an extra dimension to the visualization process. This approach enables the detection of features that may not be visible in the standard visible spectrum [44, 45]. Surveys have also addressed the comparison of RTI algorithms from both objective and subjective perspectives [46]. Still, there remains a noticeable gap when it comes to evaluating which relighting methods are most effective for specific analytical tasks. To bridge this gap, our study focuses on letting domain experts, specifically those working with palimpsests, to assess which techniques best support the analysis of particular features within the objects they study.

5.2.1 Data acquisition and description

The setup used to acquire the images involved a Phase One iXG 100MP Wide Spectrum camera equipped with a Schneider Kreuznach RS 72 mm lens, mounted on a reprographic column for stability and precision. The object was illuminated from 49 distinct angles using a DLED7-D LED light by Dedolights, capturing one image per lighting direction. The setup included four black spheres placed at the corners of the field of view to estimate light direction accurately. Additionally, a GoldenThread/DICE Object-Level Target from Image Science Associates was used to evaluate image quality. The full image stack was processed following the methodology described in detail by Giachetti et al. [47].

The manuscripts examined in this study are two palimpsests preserved at the Biblioteca Capitolare di Verona. A palimpsest is a manuscript written at least twice, typically containing an older text, called the *scriptio inferior*, that is often faint or invisible to the naked eye. Recovering this underlying writing has long been a focus of scholarly interest, and over the centuries, various techniques have been employed for this purpose. One of the manuscripts, palimpsest XL (38), contains fragments of Virgil in its *scriptio inferior*. In the 19th century, scholars attempted to recover these texts using chemical reagents, which unfortunately led to significant darkening of the manuscript and obscured much of the content. A similar situation is found in codex LXII, a 6th century palimpsest that includes Greek marginalia. Both cases present unique challenges for analysis and highlight the need for advanced imaging and relighting techniques.

The complex conservation history of the two selected manuscripts offers a range of features that are particularly valuable for domain experts, many of which become more accessible through RTI visualization, as shown in Figure 5.7. Beyond the barely visible text, RTI reveals additional elements such as the layout rulings made by the original scribes. It also highlights traces of conservation interventions, including the application of protective films and the presence of parchment patches added during recent restoration efforts.



(a) Manuscript LXII, folio 29 recto *(b) Manuscript XL (38) folios 323 recto*

Figure 5.7: The two manuscripts shows signs of the conservation treatment and the scriptio inferior hidden by the chemical reagents used by the scholars of the 19^{th} century.

5.2.2 Relighting options

In this evaluation, we provided RTI visualizations using three relighting algorithms that features different characteristics. However, in this case, NeuralRTI was not included as the evaluation was more generic, focused on understanding which enhancements are useful to improve the investigation of the manuscripts, and did not aim at comparing NeuralRTI with other classical algorithms. The three choices are the following:

- **PTM**: the image is rendered using a second order polynomial function, from the coefficients estimated with a per-pixel fitting performed on the input images. This is the original method proposed by Malzebender et al. [9] and is still very popular, even if preserves only low-frequency effects, and struggles at reproducing specular highlights and shadows. For each channel, the equations that describes how the color is rendered is:

$$c = p_0 + p_1 * l_x + p_2 * l_y + p_3 * l_x^2 + p_4 * l_y^2 + p_5 * l_x * l_y \tag{5.1}$$

  where c is a color channel, l_x and l_y are the coordinates of the light direction, and p_i with $i = 0, \ldots, 5$ are the polynomial coefficients. For the algorithm computation we used the Relight application (`https://vcg.isti.cnr.it/vcgtools/relight`).

- **RBF**: images are interpolated using radial basis functions [16], and Principal Component Analysis (PCA) compression is also applied. This method provides a better reproduction of high-frequency effects, but the result is dependent on parameters that are manually tuned, and transitions from one light to another might appear cranky. Also in this case, we used the Relight application.
- **BRDF**: the relighting is obtained using a Physically-Based Rendering (PBR) shader using normal, albedo, roughness and metallic maps. The method can provide an interesting flexibility for the rendering, enabling a separate enhancement of different parameters, but cannot handle the reproduction of realistic shadows and inter-reflections. The albedo and normal maps used for this rendering were computed using lambertian photometric stereo [48]. In addition, the roughness and metallic maps, and the rendering algorithm, were taken from the work of Ikehata [49].

Figure 5.8 shows that the methods produce very different relighted images for the same light direction.

(a) PTM *(b) RBF* *(c) BRDF*

Figure 5.8: Comparison between the three relighting algorithms for the same lighting condition. Main differences are in the reproduction of specularities, shadows, irregularities, and color. PTM preserves only low-frequency effects, while RBF and BRDF can reproduce the specular behavior of the object, with different level of accuracy. However, they can present different colors or artifacts based on the chosen light directions.

5.2.3 Enhancement options

In addition to the three relighting algorithms, we implemented a set of enhancement methods aimed at improving the visualization of specific details. Each enhancement can be applied to each relighting option, on the fly and in real-time, preserving the interactivity of the application. To do so, every algorithm either consists of few operations, or is provided of pre-computed maps to speed up computations.

STRESS enhancement

The STRESS algorithm [34] is based on the idea that both a bright detail in a bright region and a dark detail in a dark region might result difficult to be detected. To address this, STRESS computes reference values for brightness and darkness and rescales each pixel accordingly. For every pixel, the algorithm calculates a maximum and minimum envelope by comparing the central pixel with its neighbors. These neighbors are sampled randomly, with a higher sampling probability near the center. The central pixel value is then rescaled between these two envelopes, using the following equation:

$$p' = \frac{p - E_{min}}{E_{max} - E_{min}} \tag{5.2}$$

where p is the original central pixel, p' is the stressed version of the central pixel, E_{min} is the minimum envelope and E_{max} is the maximum envelope.

The implementation used in this study is available at: `https://github.com/ifarup/colourlab/tree/master/colourlab`. STRESS operates with three parameters: M, the number of random samples per iteration; N, the number of iterations; and R, the radius of the circular neighborhood from which samples are drawn. In our setup, we used the default values: M = 10, N = 100, and R = 0, meaning the radius corresponds to the diagonal length of the image.

The BRDF model uses an albedo map as input, which we replaced with its stressed version. For PTM and RBF, the input consists of multiple maps referred to as planes, numbered from plane_0 to plane_n. In PTM, plane_0 contains the constant polynomial coefficients for the RGB channels (denoted as p_0 in Equation 5.1) and serves as a base color map, thus this plane was substituted with its stressed version. RBF, on the other hand, does not include a base color map. Therefore, we computed a lambertian rendering using the albedo and normal map derived from PS, and then applied STRESS to the albedo to generate a stressed lambertian rendering. In real time, we subtracted the original lambertian rendering from the RBF output and added the stressed version, effectively merging RBF and STRESS features.

This approach allowed us to explore whether the color modifications introduced by STRESS could enhance the visibility of features compared to the default RGB rendering. A visual comparison between the original and stressed renderings is provided in Figures 5.9a and 5.9b.

Specular enhancement

Another technique we used is called specular enhancement, which renders the image monochromatically and applies an algorithm described in the Relight application. The rendering is obtained by computing the scalar product between the normal map and the light direction vector, pixel by pixel, and raising the result to a power. The rendering is calculated using the following equation:

$$I = (N \cdot l)^{k_s} \tag{5.3}$$

where I is the rendered image, N the normal map, l the light direction vector, and k_s the exponent value. In the online viewer, the user can change the value of k_s using an interactive slider, so to modify the amount of specular enhancement. Incrementing the value of k_s the image gains a stronger virtual specularity effect. The rendering result can be seen in Figure 5.9c.

(a) Default image *(b) STRESS* *(c) Specular*

Figure 5.9: Comparison between default rendering (a), STRESS enhancement rendering (b), and specular enhancement rendering (c). The STRESS enhancement allows to better differentiate colors, while the specular enhancement allows to better identify patterns in the geometry of the page.

Further enhancement options

In addition to the two previous enhancements, which apply a strong visual change to the image, we integrated a set of simpler enhancements that can be applied independently, and in real-time, to any relighting algorithm. In the following equations, I is the original image, and I_e is the enhanced image. The values of I are floating point and scaled in the range $[0,1]$:

- **Surface**: it applies an operation called normals unsharp masking [37] in order to increase the surface level of details in the image, using a pre-computed normal map. When applying normals unsharp masking, the unsharped version of the normal map is calculated as:

$$N_u = N + k * (N - N_s) \tag{5.4}$$

where N is the normal map, N_s is a smooth version of the normal map, obtained by applying a low-pass filter to N, k is a multiplication factor, used to increase or decrease the unsharp effect, and

N_u is the unsharped normal map. Afterwards, the enhanced image is obtained as:

$$I_e = I * (\max(N_u \cdot l, 0) + k_a) \tag{5.5}$$

where l is the light direction vector, and k_a is a term to simulate ambient light. In the online viewer, the user can change the values of k and k_a using interactive sliders, so to modify the amount of unsharp effect. Changing the value of k results in changing the level of detail in the image, as normals unsharp masking aims at giving a sharper visualization of the surface. This operation, however, can decrease the overall brightness of the image, and changing the value of k_a can solve side effect. A comparison between a non-unsharped image and its unsharped version is shown in Figure 5.10.

- **Intensity**: multiply by a factor the intensity of the image color. The enhanced image is obtained as:

$$I_e = I * k_i \tag{5.6}$$

where k_i is the multiplication factor. The user can change the value of k_i using an interactive slider.

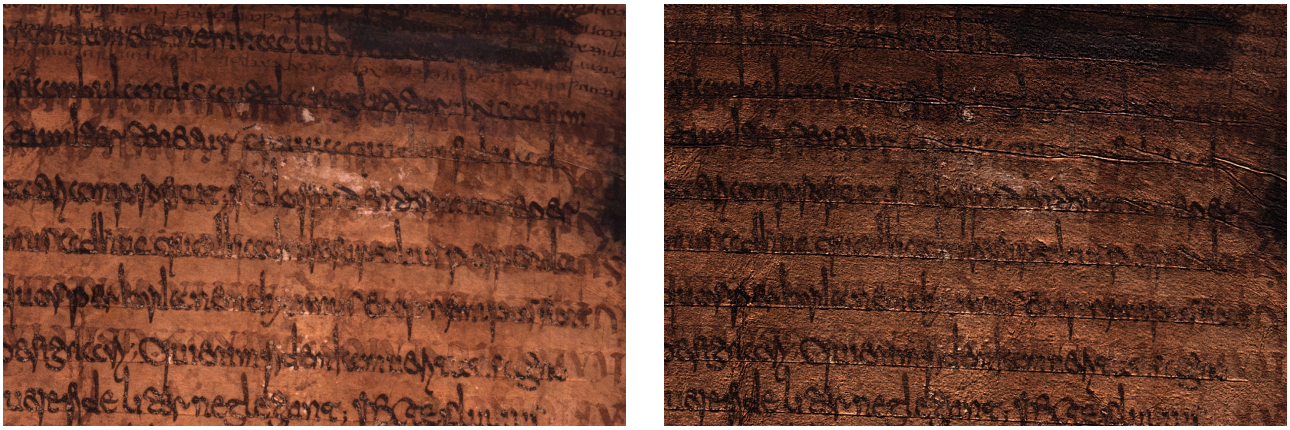
- **Contrast**: rescale the image color between two thresholds. The enhanced image is obtained as:

$$I_e = \frac{I}{k_{max} - k_{min}} + k_{min} \tag{5.7}$$

where k_{max} is the upper threshold, k_{min} is the lower threshold. Both k_{max} and k_{min} are contained in the range $[0, 1]$. The user can change the values of k_{max} and k_{min} using interactive sliders.

- **Gamma**: apply gamma correction with a value of $\gamma = 2.2$ (i.e., exponent = 1/2.2). The enhanced image is obtained as:

$$I_e = I^{\frac{1}{\gamma}} \tag{5.8}$$



(a) Default image *(b) Unsharped image*

Figure 5.10: Effect of normals unsharp masking. The unsharped image (b) has more evident geometry details than the default rendering (a), possibly allowing to better visualize the structure of the page, and other irregularities.

5.2.4 Viewer description

The viewer, realized with OpenLIME, in this case supports the visualization of one relightable image at a time. Users can control the light direction either by interacting directly with the image or through a dedicated control interface. An overlay menu provides access to different relighting algorithms and enhancement options. Additionally, the image can be freely manipulated dragged, resized, and rotated to suit the user's needs.

As illustrated in Figure 5.11, the interface is divided into four regions. Region 1 hosts the relighting options. Below the main algorithm selector (e.g., PTM in the figure), three smaller buttons allow users to switch between different enhancement modes: "color" displays the default RGB relightable image, "stress" activates the STRESS-enhanced version, and "specular" enables a specular enhancement mode. Notably, in the online viewer, specular enhancement is treated as a distinct relighting mode rather than a post-processing effect. Region 2 includes controls for real-time post-processing algorithms, along with corresponding sliders, as explained in a previous section. Region 3 features a circular control area where a small red dot can be moved to adjust the light direction. Finally, Region 4 contains general purpose tools such as zoom, rotation, screenshot capture, and full-screen mode. When the light bulb icon is highlighted in yellow, users can also change the light direction by dragging the mouse directly over the image.

The application supports seamless switching between relighting algorithms, enabling immediate visual comparison. Since the viewer is web-based, users can also open multiple browser tabs to examine different versions of the same image side by side.

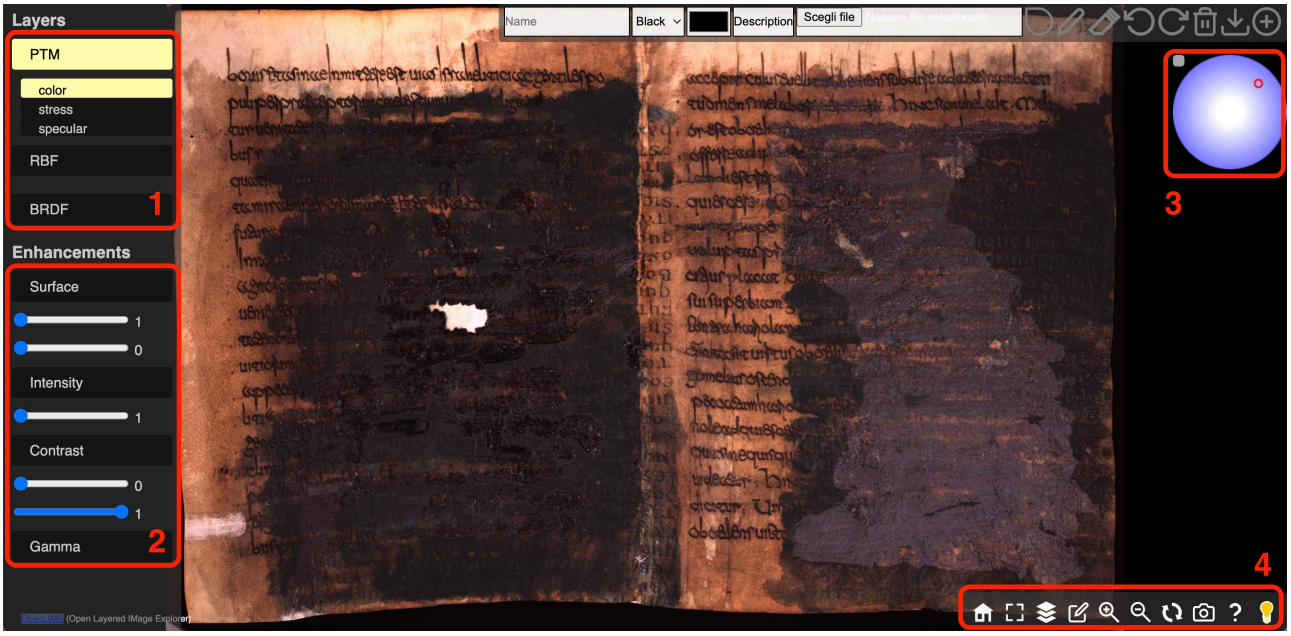


Figure 5.11: Web viewer interface. Region 1 is the relighting algorithms, region 2 is the image processing algorithms, region 3 is the light controller, region 4 is the menu for rotation, zoom, screenshot, etc.

5.2.5 Questionnaire and feedback

The primary objective of this experiment was to evaluate how different rendering options contribute to the visualization of specific features within the studied objects, and whether they assist in identifying patterns. Participants were guided through a set of instructions and asked to rate the usefulness of each method and visualization option. Each user selected a particular region of the manuscript to analyze and, after completing the evaluation, provided a brief explanation of why that area was significant and challenging to analyze.

The questionnaire was divided into four main sections. The first focused on relighting algorithms, where users rated the usefulness of PTM, RBF, and BRDF on a scale from 1 (not useful) to 5 (very useful), based on how well each method helped the visual inspection. The second section addressed the STRESS algorithm for color enhancement. Participants were asked to indicate whether STRESS improved, had no effect, or worsened the visualization for each relighting method. Although the specular enhancement mode was not part of the formal evaluation, since the study focused on color image enhancements, users were free to explore it and provide optional feedback in written form. In the third section, participants evaluated the image processing tools, using the same scale from 1 to 5. This included a specific rating for the Surface enhancement, described in Section 5.2.3, as well as a general assessment of the image processing functionalities as a whole. The final section included broader questions, such as the participants' main interests when studying a manuscript, which features RTI helped to reveal, the perceived importance of combining relighting with image processing, and the relevance of accurate color rendering. Each section concluded with an optional field for additional comments.

The evaluation focused on the combined use of multiple relighting algorithms and image enhancement techniques for the specific task of palimpsest analysis. Ten participants took part in the study. Each completed the questionnaire, offering both quantitative scores and qualitative insights regarding the effectiveness of the algorithms, the key features they look for during palimpsest examination, and the role RTI plays in supporting their work.

Results are summarized in Table 5.1, which contains the average score for the three algorithms, and the users' opinion whether STRESS improves the visualization in the three cases or not.

Algorithm	Score (out of 5)	STRESS improves it (out of 10)
PTM	3.7	7
RBF	4.6	9
BRDF	3.5	8

Table 5.1: The second column (Score) indicates the average score given by the users to each relighting algorithm. The third column (STRESS) indicates how many users considered the use of STRESS enhancement an improvement w.r.t. the non-stressed rendering.

Figure 5.12 instead shows the score distribution for the three algorithms. Among the evaluated methods, RBF emerged as the preferred option, as users noted that it was particularly effective in emphasizing the irregularities of the manuscript's surface, which are often critical for detailed analysis. However, as we're going to see soon, this result is not absolute.

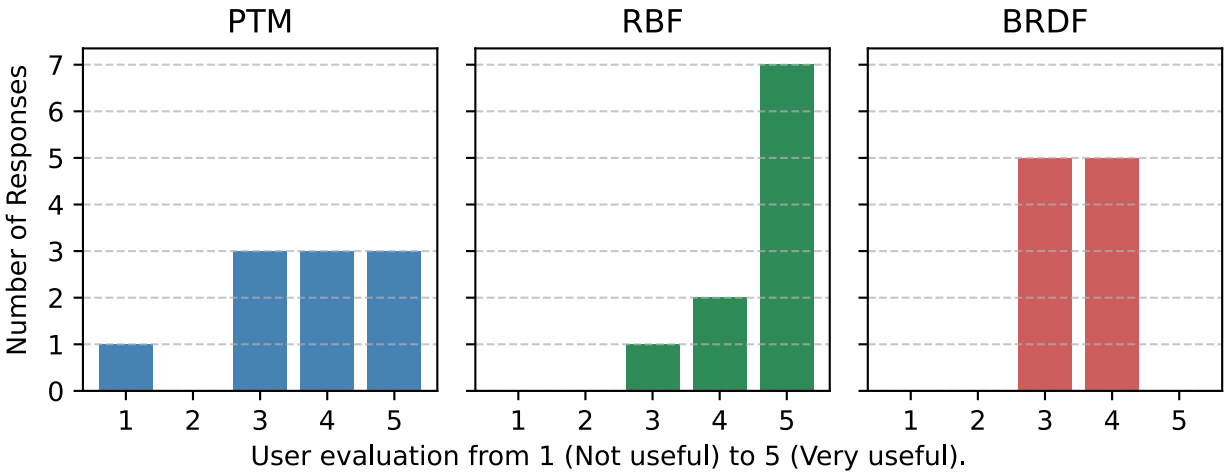
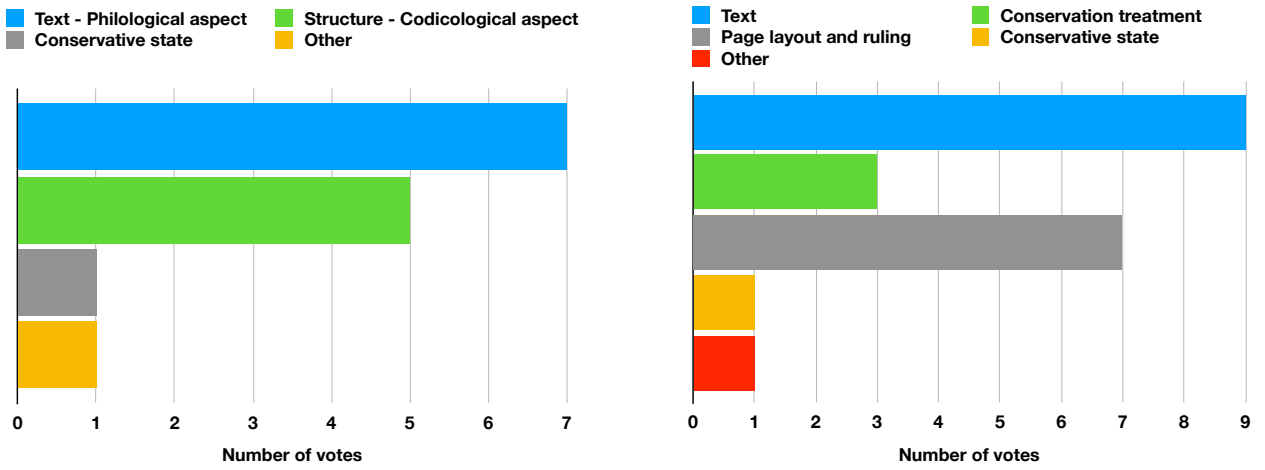


Figure 5.12: Distribution of scores for the three relighting algorithms.

These results can be better understood by examining the visual characteristics of the relighted images shown in Figure 5.8. RBF stands out for its ability to produce sharp, well-defined visual segmentation, particularly in areas with varying surface specularity. In contrast, BRDF appears less effective in distinguishing differences in gloss. While this limitation might be mitigated through improved fitting procedures, it could also stem from the reliance on non-local information during estimation. Participants also shared several meaningful comments. Some noted that "PTM makes the text more readable", while others emphasized that "The efficacy changes based on the manuscript conservation state". One user remarked that "The combination of the three is a perfect tool to study palimpsest" and another pointed out that "The evaluation is to be related to the single image and it's not absolute".

Regarding the image enhancement tools, participants were asked to evaluate the "Surface" enhancement separately, given its slightly more complex nature, and then provide an overall rating for the full set of image processing features. The "Surface" method received an average score of 2.8 out of 5, while the overall image processing tools were rated more favorably, with an average score of 4.3 out of 5. Participants shared a range of comments. One noted that "The ability to see much more detail related to the support and layered materials is valuable for the codicological study of the manuscript, though it may be distracting when the focus is solely on reading the text", while another remarked "This experiment shows that the tools must be used in relation to the manuscript's conservation state and the type of writing". Overall, the presence of a group of image processing algorithms has been appreciated.

As part of the evaluation, users were asked to indicate their main interests when studying a manuscript, as well as which features RTI helped them visualize more effectively. This section allowed for multiple selections. A summary of these opinions is shown in Figure 5.13.



(a) Which are your interests when analyzing a palimpsest?

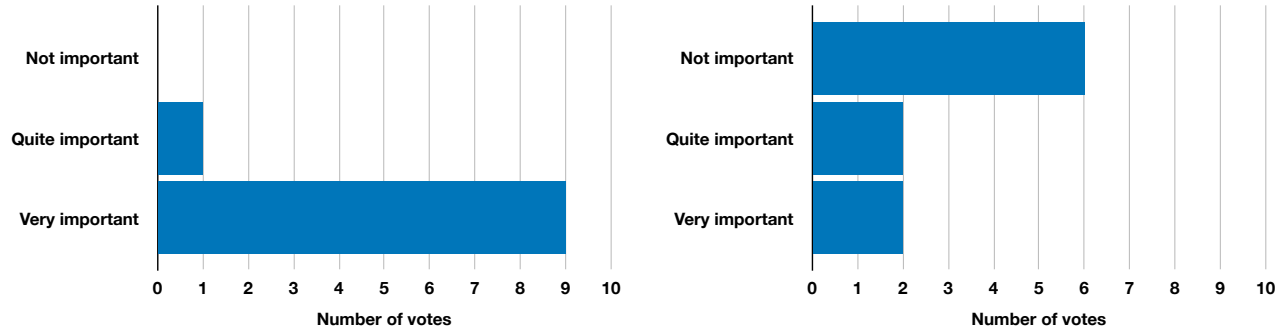
(b) Which features did RTI helped you to better identify?

Figure 5.13: Users opinion on main interests about palimpsests study and for which features RTI helped them. For this voting, users had multiple choices

In the final part of the evaluation, participants were asked whether they found it useful to combine multiple relighting algorithms with enhancement methods within a single application for the study of these objects. They were also asked whether accurate color rendering is important in their opinion, considering that color perception can be influenced by the chosen visualization method, compression artifacts, and other factors. The majority of users considered the integration of various tools within one application to be highly valuable. They appreciated the flexibility it offers, enabling them to adapt the visualization to different research needs. Opinions on color fidelity, however, were more diverse. While some emphasized its importance, others noted that its relevance depends largely on the specific features being analyzed within the manuscript. A summary of these responses is shown in Figure 5.14.

5.2.6 Evaluation results

This study was designed to explore which RTI features are most effective for visualizing specific types of objects. To do so, we developed an application that enables users to view relightable images using various relighting techniques, while also offering a set of image processing tools to fine-tune the rendering in real time. We selected ancient palimpsests as our use case, given that RTI provides a non-invasive way for experts to examine these fragile manuscripts. Prolonged manual inspection can pose a risk to their preservation, whereas digital analysis can reveal subtle details that are often missed by the naked eye or in static photographs.



(a) How much this application is useful for the case study? *(b) How much is the correct color rendering important?*

Figure 5.14: Users overall opinion on the usefulness of this application, and about the necessity of correct color rendering.

The evaluation results showed no single relighting method to be universally superior. Instead, participants appreciated the flexibility of being able to switch between different relighting options and apply real-time image processing adjustments. This adaptability was seen as a key strength of the application.

The approach presented here can be extended to other domains, offering a framework for assessing how RTI can support visual analysis of various materials, such as paintings, textiles, or other cultural artifacts. Since RTI primarily enhances the perception of surface geometry under changing lighting conditions, it is particularly well-suited for opaque materials. However, further research could investigate its effectiveness on more complex surfaces, including reflective, glossy, or translucent materials. Importantly, such studies should not rely solely on numerical metrics, but also involve domain experts, like color scientists, to evaluate how well different algorithms perform in practice. This kind of interdisciplinary feedback can guide both end users in selecting appropriate digital tools and developers in refining existing methods or introducing new features.

5.3 Outcomes and discussion

Considering that RTI, in general and especially in this thesis, is widely used for cultural heritage studies, confronting with experts can help focusing research in the right path for developing new technologies that are also useful to the users. In our evaluations, we mainly tried to assess the interest in using RTI inside an application that allows you to interact with different models and processing algorithms. The number of possible cases where RTI could be used and improve the visualization can be an opportunity for further researches. However, more experiments can also be done to address the usefulness of different RTI models in automatic tasks, such as thread counting in the case of textiles.

Chapter 6: BASS-MLIC: a new synthetic dataset for multi-light applications

In this chapter we discuss which direction is a good choice when creating synthetic datasets for multi-light applications, showing the dataset we have developed. As we have seen, MLICs are a widely used especially for nearly-flat objects like bas-reliefs, inscribed tablets or rocks, paintings or ancient books. From MLICs processing we typically derive methods for image relighting (RTI) or normal map estimation (PS) [9, 48]. However, the acquired data, thanks to the recent advancements of Computer Vision techniques, can be exploited also to recover further parameters. Specific inverse rendering methods have been proposed to derive SV-BRDF maps from MLICs [50], material segmentation masks and material parameters can be recovered as well [51], depth maps or shadow maps could be extracted exploiting fine-tuned foundation models, even from single images [52]. A limited number of annotated benchmarks is, however, available to validate the estimated parameters, especially if we focus on bas-relieved objects typically studied in Cultural Heritage. Dataset with MLICs with rich sets of annotation are not only fundamental to validate the algorithms, but also to train or fine tune the methods proposed for the parameters' estimation. The currently available multi-light image datasets are mostly designed for the evaluation of normal estimation (PS), or relighting quality.

PS algorithms are mostly evaluated using the DiLiGenT (`https://photometricstereo.github.io/`) benchmark [53], made of real images of different 3D objects with large depth variations and captured in the center of the camera view and provided with object masks. Images are captured with directional lights and are associated with ground truth, estimated normal maps. Novel versions of DiLiGenT benchmarks have been presented to evaluate the performances of PS methods on different types of objects and materials. DiLiGent10^2 [54] was created with 100 objects with 10 shapes and 10 materials combinations. The controlled shapes and materials allow to evaluate the performance of PS methods on each dimension independently. DiLiGenT-Pi [55] contains 30 near-planar and rich-detail objects with 4 group materials combinations. DiLiGenRT [56] includes 54 objects with controlled surface roughness and structure translucency. A synthetic dataset to evaluate PS methods, SynthPS (`https://github.com/Univr-RTI/SynthPS`) has also been proposed in [57]. The dataset include renderings (realized with the Blender Cycles engine) of bas-relieved surfaces with assigned single materials or combinations of different ones. The masks used to assign different materials to different regions are not, however, simulating realistic objects. Interestingly, the dataset includes also ground-truth shadow masks estimated in the rendering, but these masks have not used practically in

research works.

For the evaluation of RTI relighting, two datasets have been proposed in [3]. The first, SynthRTI (https://github.com/Univr-RTI/SynthRTI), includes renderings of bas-relieved surfaces with assigned single materials or combinations of different ones. The second, RealRTI, includes cropped regions of typical RTI acquisitions of cultural heritage artifacts. In this case, the evaluation is based on splitting the data in a training set, to estimate the relighting model and a test set, to estimate the quality of the relighting with image comparison metrics.

While these benchmarks allow the evaluation of the accuracy of normals estimation and relighting on different types of objects and materials, it is evident that they cannot be used to assess other relevant tasks that can be based on MLICs. For this reason, we decided to create a novel synthetic benchmark, BASS-MLIC (BAS-reliefs Synthetic Multi-Light Image Collections), allowing the evaluation of a more complete set of processing techniques. The main characteristics of the new dataset are:

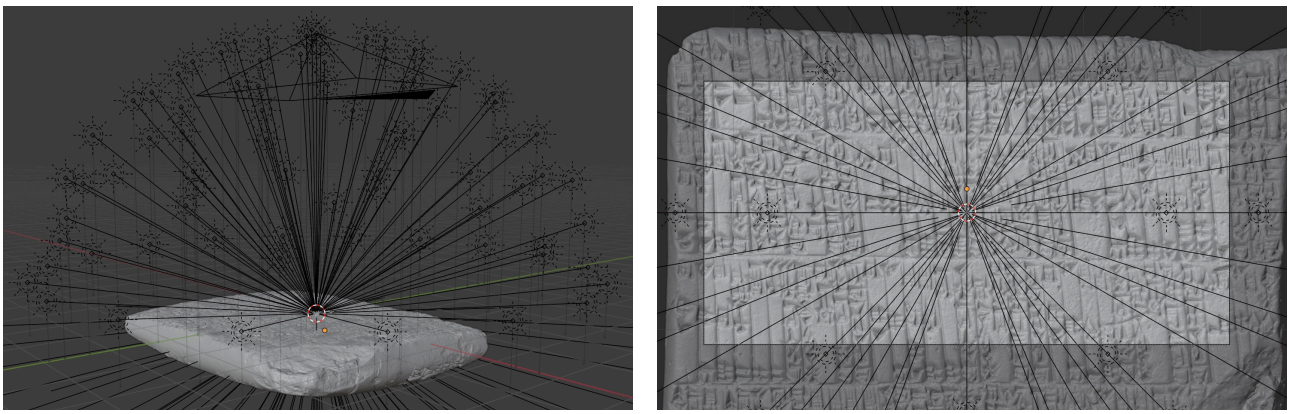
- It provides ground truth metric depth, allowing the evaluation of depth estimation frameworks
- It provides ground truth shadow masks, allowing the evaluation of heuristics or neural methods to automatically segment shadowed pixels
- It provides single-material surfaces as well as multi-material ones with partitioning of regions made of different materials simulating real-world surfaces.
- It is easy to extend with novel objects and materials as it is created with a simple Blender script.

6.1 Motivation and design

To create BASS-MLIC we used Blender (www.blender.org). We designed a virtual acquisition setup with an object area where bas-relieved surfaces are placed, a fixed camera (orthographic in this first version to match the assumptions made in many estimation frameworks, but easily modifiable for future versions) with axis perpendicular to the surface. We added to the scene a dome of virtual lights (that can actually be loaded from a file to be easily modifiable). To create the BASS-MLIC dataset, we used a classical dome distribution with 49 lights to train relightable images and 28 additional lights for testing. Figure 6.1 shows an example of the virtual dome configuration.

6.1.1 Objects selection and segmentation

Following the idea behind SynthRTI and SynthPS, we decided to produce a dataset containing objects from the cultural heritage domain, with different levels of detail in the in-plane pattern and various amounts of depth variability. The current version of the dataset has been created with five 3D models, downloaded from Sketchfab (https://sketchfab.com/). We refer to these model as:



(a) Blender's virtual dome layout. *(b) Camera viewpoint.*

Figure 6.1: Overview of Blender's layout, showing the 3D model, light sources, and camera, and detail of the camera viewpoint.

- **Wall** (`https://skfb.ly/o7L7W`), featuring limited depth variations
- **Cuneiform** (`https://skfb.ly/o9LQN`), also with slight depth variation, but with an in-plane pattern with high spatial frequency
- **Christ** (`https://skfb.ly/oQwQI`), representing human figures with relatively large depth variations
- **Doctor**(`https://skfb.ly/6UIPO`), also representing human figures with large depth variations
- **Maya** (`https://skfb.ly/6wzV9`), containing stylized figures and symbols and smaller depth variations

Figure 6.2 shows how the 3D models look like if seen from Blender.

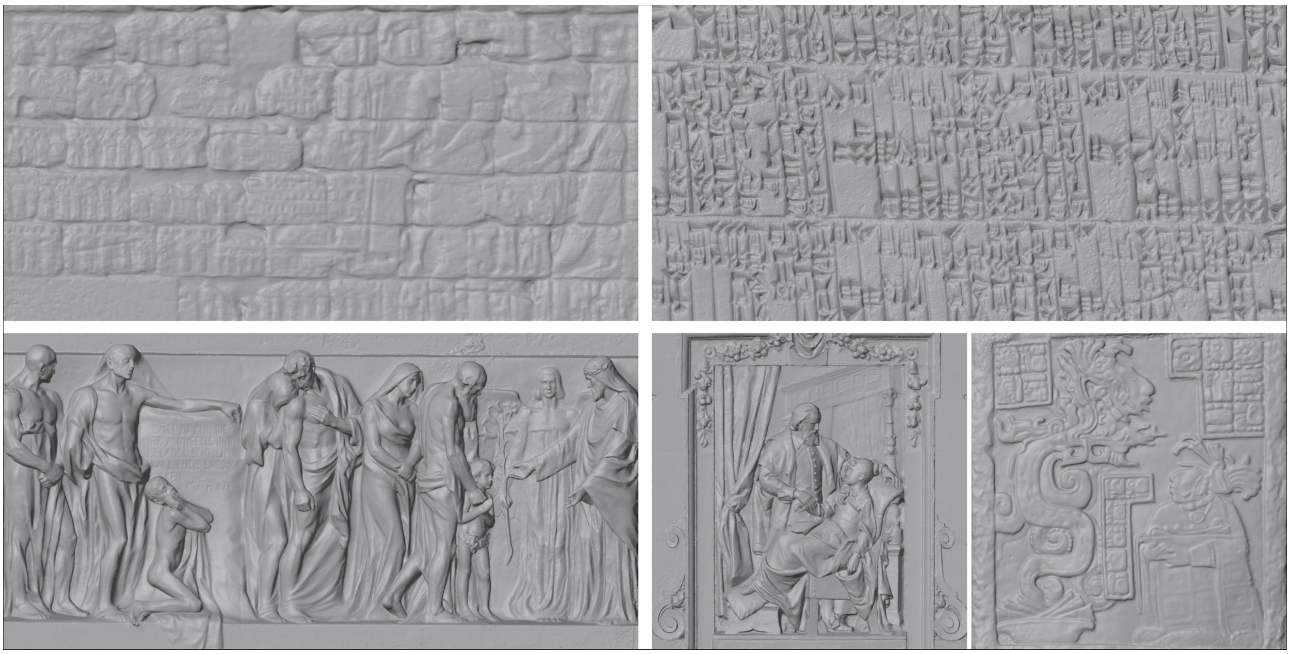
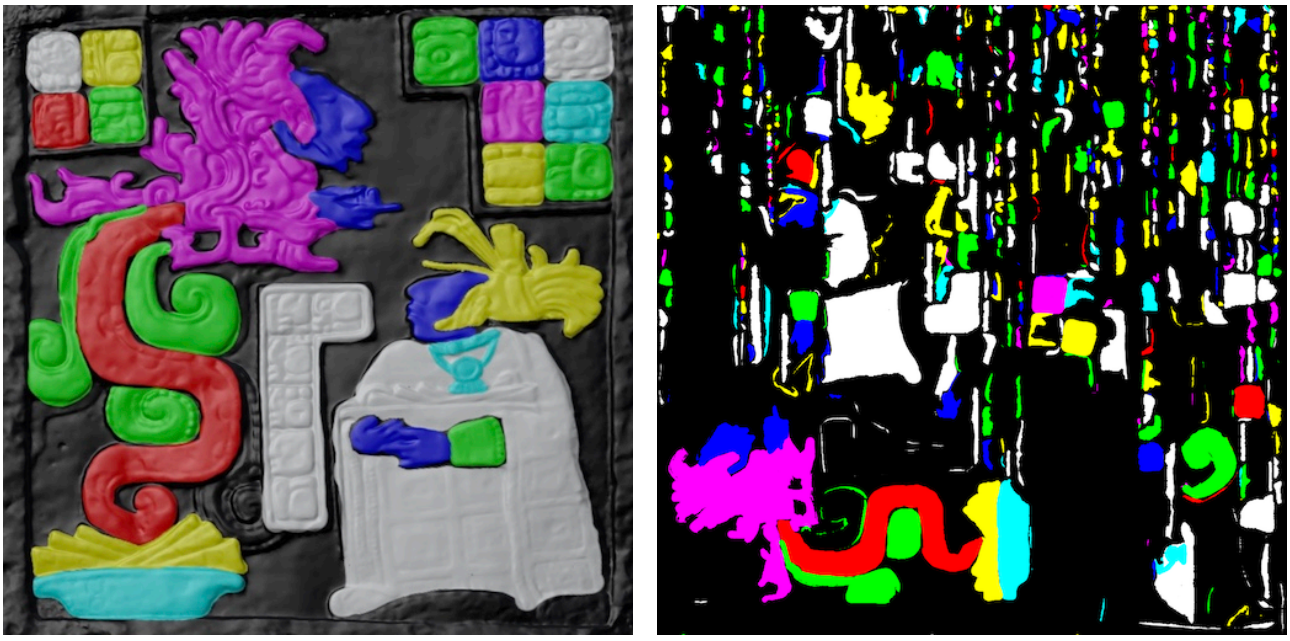


Figure 6.2: Screenshots of the 3D models used in the dataset. From top to bottom and left to right: Wall, Cuneiform, Christ, Doctor, and Maya.

The surface properties were assigned according to the Blender BSDF model, and we used two settings: **single-material**, with uniform properties across the whole image, and **multi-material**, where different properties are assigned to different regions. To create reasonable regions where to apply different parameters in the multi-material setting, we used the texture paint workspace of Blender, directly painting semantic segmentation masks on the 3D surface with a set of 8 reference colors. Masks are then stored as textures, as shown in 6.3. In multi-material rendering, these textures are then loaded and used within Blender's node-based rendering workflow to apply different materials to the regions that are differently colored.

While previous works adopting a similar strategy to generate multi-material surfaces relied on non-realistic segmentation masks to assign the parameters [3, 57], we performed a manual semantic segmentation of the elements of the surfaces to enable a more realistic result.

In the case of human figures, we segmented the bodies, hair, clothes, and other details. For the cuneiform tablet and part of the wall, we resorted instead to applying a spray of randomly distributed segmented areas to increase the complexity of the overall material. In a realistic setting, this procedure can simulate the noise of degraded materials, for example, where many layers of materials are applied and both the upper and lower ones are visible due to the degradation.



(a) Object's surface after texture painting. *(b) Texture obtained as a result of texture painting.*

Figure 6.3: Mask and texture after texture painting for the object Maya.

6.1.2 Materials choice

We assigned the materials using the native rendering options of Blender, trying to replicate realistic scenarios. The level of complexity varies for each material, e.g., the presence or absence of specular or

metallic materials.

For the single-material version, we simulated red stone (using only roughness), metal (using metallic and low roughness), ceramic (using coating), and wood (using sheen). For the multi-material version, materials have been grouped in four categories: materials with various properties, metals (like gold or steel), colored marbles, and architectural materials (like concrete or plaster). In Tables 6.1, 6.2, 6.3, 6.4, and 6.5, all the materials and properties are shown. Figure 6.4 shows an example for all objects and all materials, using the same illumination condition in every image.

For the sake of simplicity, in the following sections we will refer to the four materials in the single-material version as S1, S2, S3, and S4, while to the four groups of materials in the multi-material version as M1, M2, M3, and M4.

	Base Color	Roughness	Specular	Metallic	Coating	Sheen
Red stone	(0.72, 0.35, 0.25)	0.7	0.2	0.0	0.0	0.0
Metal	(0.4, 0.4, 0.4)	0.3	0.5	1	0.0	0.0
Ceramic	(1.0, 0.9, 0.8)	0.2	0.6	0.0	0.3	0.0
Wood	(0.4, 0.25, 0.1)	0.6	0.3	0.0	0.0	0.2

Table 6.1: Blender BSDF parameters for the four materials in the single-material version.

	Base Color	Roughness	Specular	Metallic	Coating	Sheen
Steel	(0.5, 0.5, 0.5)	0.3	0.5	1	0.0	0.0
Leather	(0.25, 0.15, 0.05)	0.5	0.3	0	0.0	0.0
Plastic	(0.1, 0.1, 0.1)	0.1	0.5	0	0.0	0.0
Wood	(0.3, 0.15, 0.05)	0.6	0.4	0	0.0	0.5
Ceramic	(1.0, 0.9, 0.8)	0.2	0.6	0	0.3	0.0
Gold	(1.0, 0.766, 0.336)	0.1	0.5	1	0.0	0.0
Canvas	(0.8, 0.75, 0.7)	0.8	0.2	0	0.0	0.6
Concrete	(0.5, 0.5, 0.5)	0.9	0.1	0	0.0	0.0

Table 6.2: Blender BSDF parameters for the 1^{st} group of the multi-material version.

	Base Color	Roughness	Specular	Metallic	Coating	Sheen
Steel	(0.6, 0.6, 0.6)	0.3	0.5	1.0	0.0	0.0
Gold	(1.0, 0.766, 0.336)	0.1	0.5	1.0	0.0	0.0
Copper	(0.72, 0.45, 0.2)	0.5	0.4	1.0	0.0	0.0
Bronze	(0.55, 0.4, 0.2)	0.6	0.4	1.0	0.0	0.0
Aluminum	(0.7, 0.7, 0.75)	0.6	0.4	1.0	0.0	0.0
Iron	(0.3, 0.3, 0.3)	0.7	0.3	1.0	0.0	0.0
Zinc	(0.75, 0.7, 0.7)	0.6	0.4	1.0	0.0	0.0
Titanium	(0.5, 0.5, 0.55)	0.4	0.5	1.0	0.0	0.0

Table 6.3: Blender BSDF parameters for the 2^{nd} group of the multi-material version.

	Base Color	Roughness	Specular	Metallic	Coating	Sheen
White marble	(0.95, 0.95, 0.95)	0.3	0.5	0.0	0.2	0.0
Black marble	(0.1, 0.1, 0.1)	0.4	0.6	0.0	0.3	0.0
Red marble	(0.6, 0.2, 0.2)	0.5	0.5	0.0	0.2	0.0
Green marble	(0.2, 0.4, 0.3)	0.4	0.5	0.0	0.2	0.0
Yellow marble	(0.9, 0.8, 0.5)	0.5	0.4	0.0	0.2	0.0
Gray marble	(0.5, 0.5, 0.5)	0.4	0.5	0.0	0.2	0.0
Pink marble	(0.9, 0.7, 0.7)	0.5	0.4	0.0	0.2	0.0
Blue marble	(0.3, 0.3, 0.6)	0.4	0.5	0.0	0.2	0.0

Table 6.4: Blender BSDF parameters for the 3^{rd} group of the multi-material version.

	Base Color	Roughness	Specular	Metallic	Coating	Sheen
Concrete	(0.5, 0.5, 0.5)	0.9	0.1	0.0	0.0	0.0
Brick	(0.6, 0.2, 0.15)	0.75	0.2	0.0	0.0	0.0
Plaster	(0.9, 0.85, 0.8)	0.8	0.2	0.0	0.0	0.1
Stone	(0.4, 0.4, 0.4)	0.85	0.2	0.0	0.0	0.0
Chalk	(0.95, 0.95, 0.95)	0.85	0.1	0.0	0.0	0.0
Slate	(0.2, 0.2, 0.25)	0.9	0.2	0.0	0.0	0.0
Ceramic	(1.0, 0.9, 0.8)	0.2	0.6	0.0	0.3	0.0
Paint	(0.8, 0.8, 0.8)	0.7	0.2	0.0	0.0	0.0

Table 6.5: Blender BSDF parameters for the 4^{th} group of the multi-material version.

6.1.3 Rendering settings

All the RGB images, shadow maps, normal maps, and depth maps have been rendered using Blender's Cycles engine. The resolution is 512x512 (Doctor and Maya) and 1024x512 (Christ, Cuneiform and Wall). All the images are saved using the PNG format without loss. The RGB images are linear (gamma correction is not applied) and saved using 16 bits to preserve color differences even in dark regions. The shadow maps are binary, using 0 (black) for shadowed pixels and 1 (white) for non-shadowed pixels. The normal maps are generated using custom Blender nodes mapping the color codes corresponding to the camera-view normal vectors onto the surface points, and the rendered texture is saved linearly onto a 16-bit PNG file. The depth map is generated using custom Blender nodes, mapping the pixel depth divided by the far clip plane distance set to 6 meters onto the 16-bit grayscale linear depth range. This setting corresponds to a z-resolution of the PNG-encoded maps of 0.091mm. Figure 6.5 shows an example of ground truth shadow, normal, and depth maps.

6.2 Shadow estimation in MLIC data

In the following sections, we talk about possible applications for the dataset. In this section in particular, we investigate shadow detection and removal from multi-light images, aiming at improving

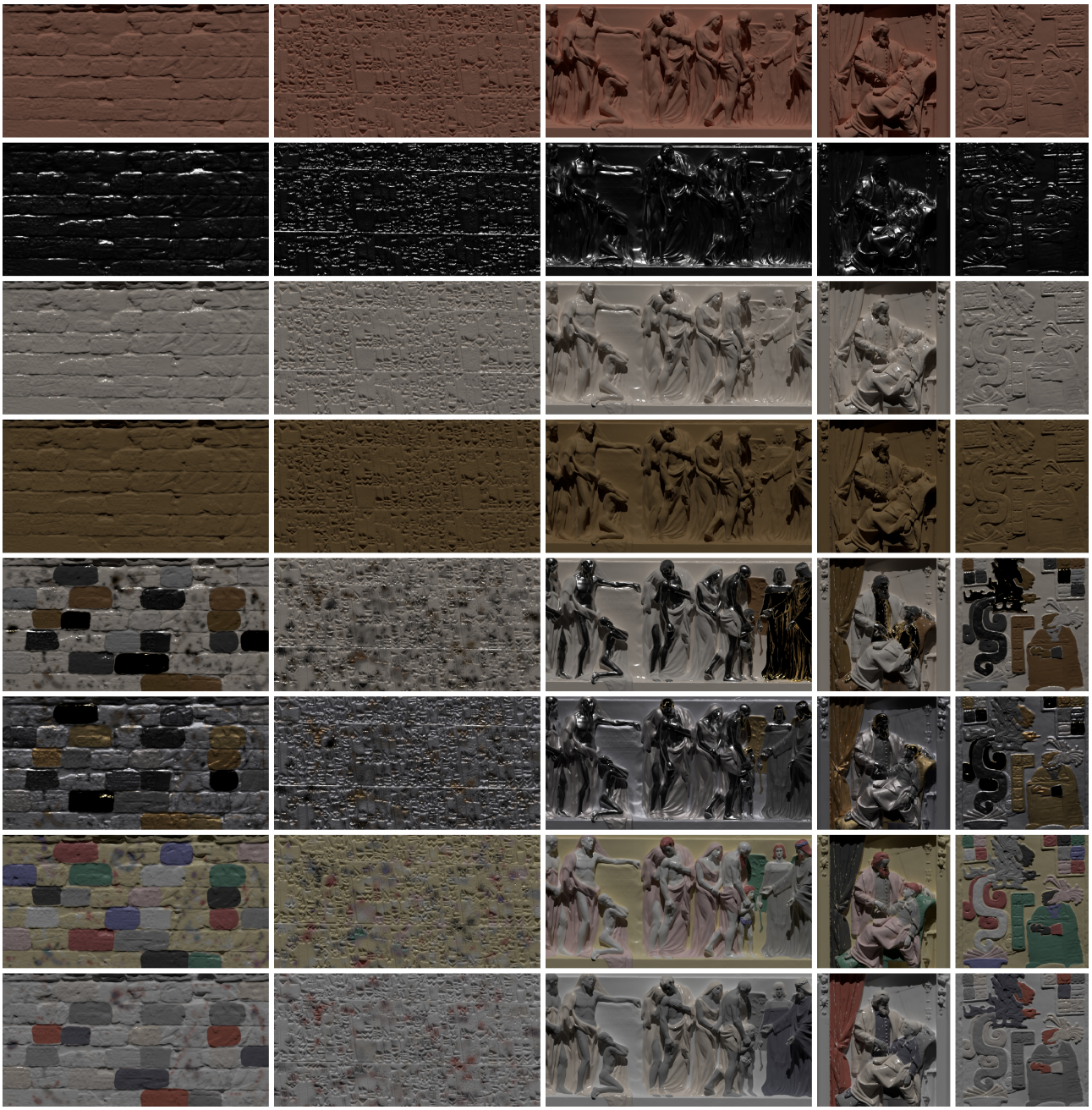


Figure 6.4: Examples of the five objects from the dataset, for each material, illuminated from the same light direction.

photometric stereo and relighting algorithms.

The segmentation of accurate shadow maps of single images of an MLIC is not straightforward, due to inter-reflections, making them not completely dark. Shadows are a fundamental cue for reconstructing shape from multi-light images, and have been explicitly modeled in some PS methods [58]. Estimating shadowed pixels before trying to estimate normals or fitting BRDF models could improve the accuracy of the results. Two strategies can be employed for this goal: the use of foundation models learning shadow masks from example data, or the use of heuristics exploiting the multi-light information, as often done in robust photometric stereo approaches.

In our preliminary experiments, we tested two methods following these ideas. The first is based

(a) Shadow map *(b) Normal map* *(c) Depth map*

Figure 6.5: Example of ground truth maps available in the dataset. The shadow map (a) is from a single light directions. Each light direction has its shadow map.

on a heuristic method, which exploits the per-pixel average and variance of the color along the full stack of images. In the second case, we fine-tuned a popular foundation model, Segment Anything Model (SAM) [59], to segment the shadows on images of bas-relieved surfaces captured under nearly directional light.

6.2.1 Average and variance thresholding

The first approach exploits the idea that shadowed regions are darker in color than their non-shadowed counter part. Therefore, for each pixel, we can calculate the average value and variance across all illuminations to discriminate pixels that are too small with respect to the average. Given a MLIC with N images, where I_i is the i-th image and S_i is its corresponding shadow map, μ_{MLIC} and σ_{MLIC} are respectively the per-pixel average and standard deviation for all images across illuminations. Images can be gamma corrected first to increase the differentiation between dark and bright values. We used gamma correction with a γ value equal to 2.2. We can estimate S_i with the following boolean equation:

$$S_i = I_i < \mu_{MLIC} - \sigma_{MLIC} \tag{6.1}$$

Pixels where S_i is true are classified as shadows. The problem with this approach is that if a pixel is mostly shadowed, we consider several shadowed pixels non-shadowed (false negatives). On the other side, if a pixel is mostly non-shadowed, we will consider several non-shadowed pixels as shadowed (false positives). This happens especially with flat surfaces, where the intensity distribution across illuminations is more homogeneous (lower variance). We can make the approach more robust by using the surface's normal map to adjust the threshold based on the pixel orientation: for flat pixels, the threshold will be lower (less shadows), for oblique pixels, the threshold will be higher (more shadows). Equation 6.1 can be modified as:

$$S_i = I_i < \mu_{MLIC} - k_s * \sigma_{MLIC} \tag{6.2}$$

To compute k_s, we calculate the internal product (dot product) between the normal vector of a pixel ($\vec{n_p}$) and the normal vector along the z-axis ($\vec{n_z} = [0,0,1]$). This way, k_s is 0 for if $\vec{n_p}$ is horizontal, and 1 if $\vec{n_p}$ is vertical. Therefore, we rescale k_s in the range [-1,1] in order to further increase the threshold when $\vec{n_p}$ is more horizontal (oblique surface). The equation is:

$$k_s = 2 * (\vec{n_p} \cdot \vec{n_z}) - 1 \tag{6.3}$$

In following sections we will refer to this method as "μ-σ" .

6.2.2 SAM fine-tuning

SAM is a general-purpose segmentation model for images and videos, published by Meta [59] (`https://github.com/facebookresearch/sam2`). The model is thought to receive either a video sequence or a single image as input and potentially segment any object in the scene, given additional guiding input, such as the coordinates of where to find the object.

Several works have adapted SAM for shadow detection, but typically on general-purpose datasets like urban scenes. To make the method effective on our images, we fine-tuned the pre-trained version of SAM, available on the official repository, on a set of images similar to those included in BASS-MLIC, but rendered with different objects and materials.

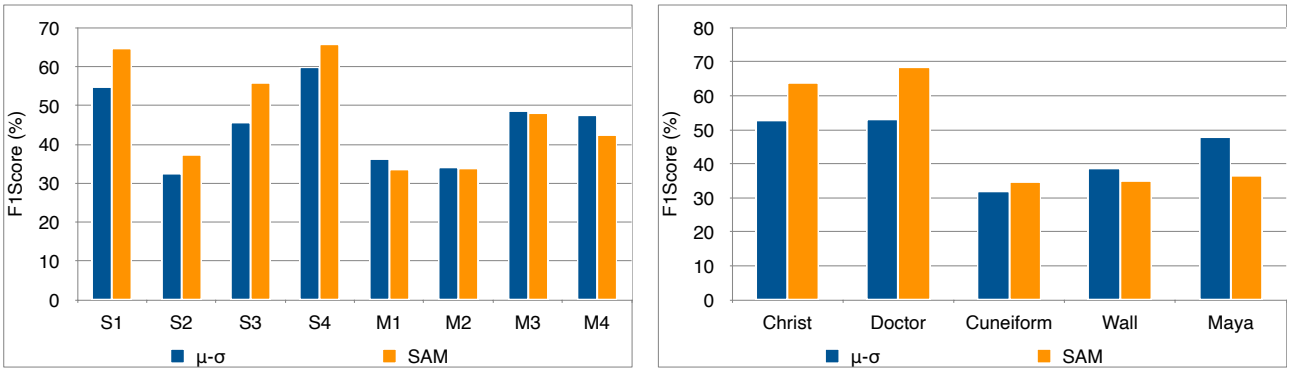
The model we used is SAM version 2.1 large, the one with the most parameters. SAM comes with a specific library for fine-tuning and inference, which allows to train either only the image decoder, which is responsible for generating multiple segmentation masks, or the image encoder too. In our case, we trained both encoder and decoder to generate a single segmentation mask, using each image as an input and its corresponding ground truth shadow map as a target.

6.2.3 Shadow estimation results

Shadow segmentation methods involve the generation of a binary mask, where shadowed pixels are classified with one value (0), and non-shadowed pixels with the other value (1). The performance of such algorithms can be evaluated computing the amount of true positives (TP), which are shadowed pixels classified as shadows, true negatives (TN), which are non-shadowed pixels classified as non-shadows, false positives (FP), which are non-shadowed pixels classified as shadows, and false negatives (FN), which are shadowed pixels classified as non-shadows. We can thus validate shadow detectors using metrics like F1 score, used to balance the presence of FP and FN in the final score:

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Recall} = \frac{TP}{TP+FN}$$
$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

Figure 6.6 shows the F1 score of the two methods discussed in previous sections. We show both the average for each material (Figure 6.6a) and the average for each object (Figure 6.6b).



(a) Average across objects. *(b) Average across materials.*

Figure 6.6: F1 score for shadow segmentation. The average is computed for each material across all objects (a) or for each objects across all materials (b).

The results are not good and both methods have strong limitations. SAM appears clearly more effective on single-material surfaces, but is not similarly good on multi-material ones. The failures of the methods are evident looking at Figure 6.7, where we show the results obtained on the object Maya with materials S4 and M4. It is, however, worth noting that, despite these limitations, the recovered shadow masks are still useful to improve the normal estimation as we will see in Section 6.3.

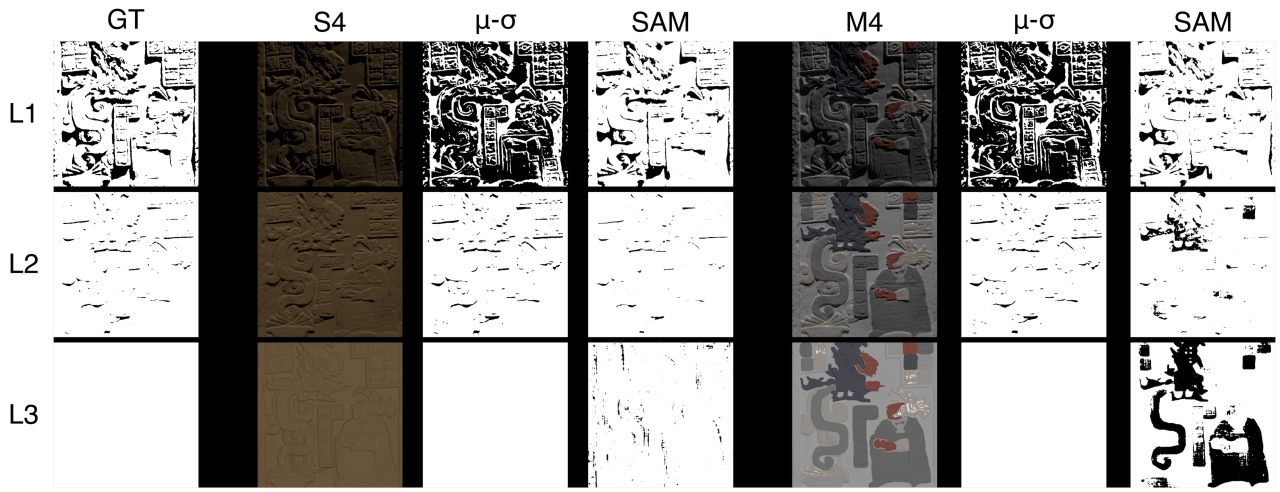


Figure 6.7: Examples of shadow detection. The method based on the simple multi-light heuristics fails in case of dark images. Fine-tuned SAM works poorly on multi-material objects when light comes from above, not being able to discriminate between a shadowed area and a dark colored one.

6.3 Shadow-aware Photometric Stereo

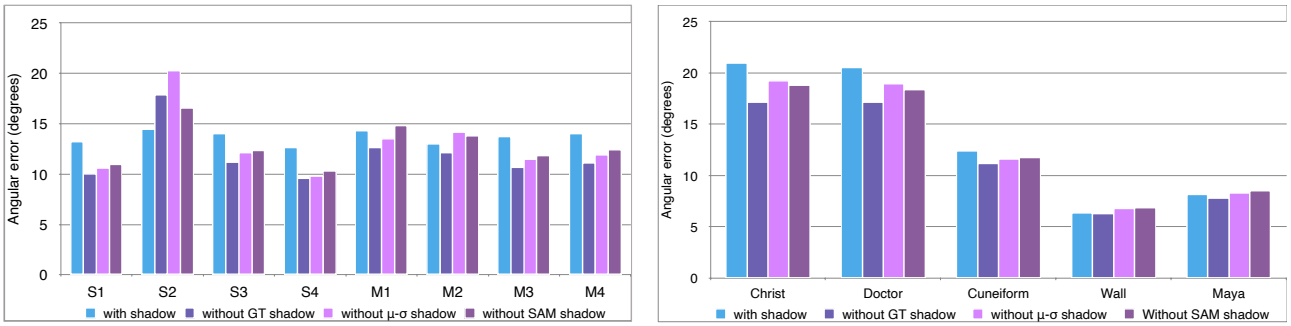
As mentioned in Section 6.2, PS algorithms can benefit from removing shadowed pixels from their input data. An example can be seen computing Lambertian photometric stereo, which consists in estimating the normal map using the least square algorithms, trying to solve the following equations:

$$N \cdot L - I = 0 \tag{6.5}$$

where N represents the per-pixel normal map that we have to estimate, L is the matrix obtained by stacking together all the light directions, and I a matrix obtained by stacking all the images, usually converted to gray scale.

Thanks to the fact that normals are computed independently for each pixel, we can specifically filter out shadowed pixels. Therefore, using the lambertian photometric stereo approach, we present a comparison of normal map estimation including and removing shadowed pixels, using both ground truth shadows and estimated shadows. For a more refined estimation, we remove also highlights pixels, i.e., pixels that are clipped to the maximum value (65535 in the case of 16 bits images).

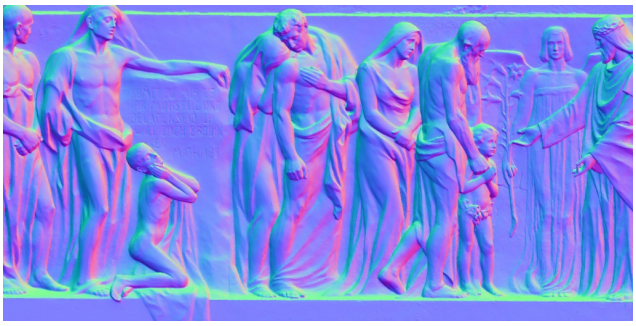
Figure 6.8 shows the average angular error, which gives a per-pixel error map of how much two normal vectors are different, between estimated normal maps and ground truth normal maps. It is possible to see that the removal of estimated shadows provides a non-negligible improvement in the accuracy of the estimated normals, with the exception of S2 and M2 that are fully metallic surfaces. It is likely that, at this stage, SAM estimates shadows in an inconsistent manner, possibly adding noise, with respect to the μ-σ method. Therefore, despite it performs better for single-materials, as shown in Figure 6.6a, it does not lead to better normal map estimation. In Figure 6.8 we also report the results obtained removing ground truth shadows that ideally should set a bound to the improvements provided by the removal of estimated shadows. Despite the limitations in the shadow masks segmentation, the improvement obtained is close to this bound for the non-metallic surfaces. For the metallic surfaces, the ground truth shadow removal actually makes the estimation worse.



(a) Average across objects. *(b) Average across materials.*

Figure 6.8: Angular error (measured in degrees) between estimated normal map and ground truth normal map. The average is computed for each material across all objects (a) or for each objects across all materials (b).

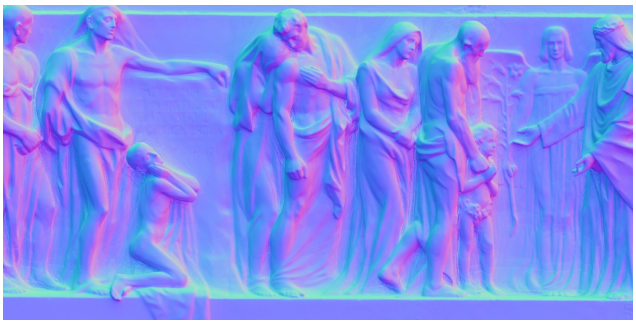
Figure 6.9 shows a comparison between ground truth normal map and normal map estimated with and without shadows, for object Christ (S1), where the improvement is visible as removing shadows leads to a more detailed normal map. The average angular error in Figure 6.9c is 20.2, in Figure 6.9e is 13.57, and in Figure 6.9g is 15.32.



(a) Ground truth

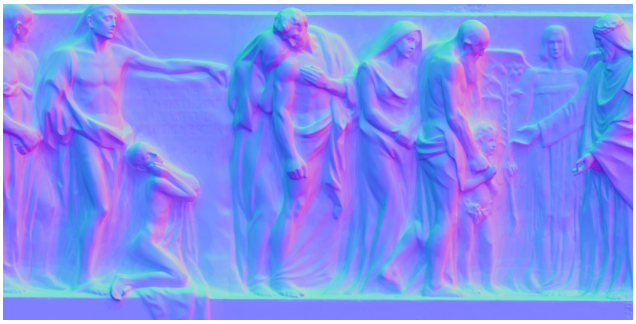
(b) Lambertian PS

(c) Angular error lambertian PS



(d) Lambertian PS without GT shadows

(e) Angular error lambertian PS without GT shadows



(f) Lambertian PS without estimated shadows

(g) Angular error lambertian PS without estimated shadows

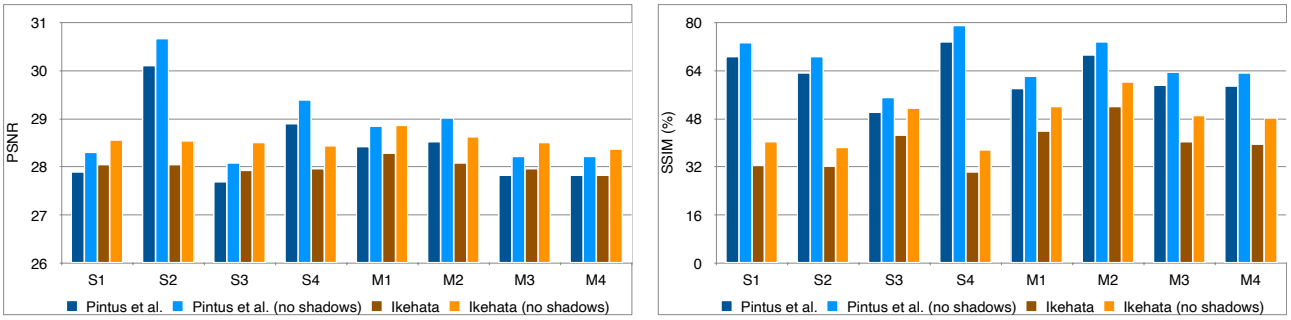
Figure 6.9: Normal map estimation. Comparison between ground truth data and lambertian PS with and without shadows. For (f) the μ-σ method has been used. The average angular error is: 20.2 (c), 13.57 (e), and 15.32 (g).

6.4 Shadow-aware BRDF evaluation

Ground truth shadow masks are useful not only to evaluate the performances of shadow segmentation methods, but should also be used to perform a fair comparison of reflectance models' fitting. In fact, if we perform image relighting using the reflectance model estimated for the test directional light, the result is meaningful only for the points actually receiving the direct illumination. The other points are not useful to verify the quality of the material reconstruction.

To show this fact with a toy example, we present the evaluation of two BRDF estimation methods performed on BASS-MLIC, showing the difference when a metric is calculated on the whole image or only on the non-shadowed region (obtained from the ground truth shadow maps included in the dataset). The first method is the parametric BRDF fitting proposed by Pintus et al. [50], while the second is the neural Photometric Stereo method called SDM-UniPS, proposed by Ikehata [60] (`https://github.com/satoshi-ikehata/SDM-UniPS-CVPR2023`), which also provides a direct estimation of reflectance maps.

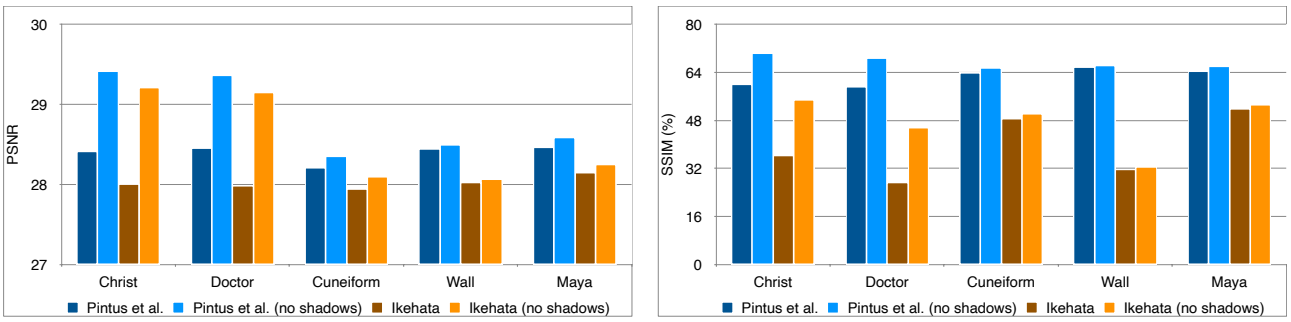
To show the gain in score, PSNR and SSIM are calculated between original and relighted images, both leaving and removing shadowed regions. Evaluating this kind of algorithms only on non-shadowed areas leads to increased metrics and avoid potential biases in the evaluation. Figure 6.10 shows the metrics for each material, averaged across all objects, while Figure 6.11 shows the metrics for each object, averaged across all materials.



(a) PSNR *(b) SSIM*

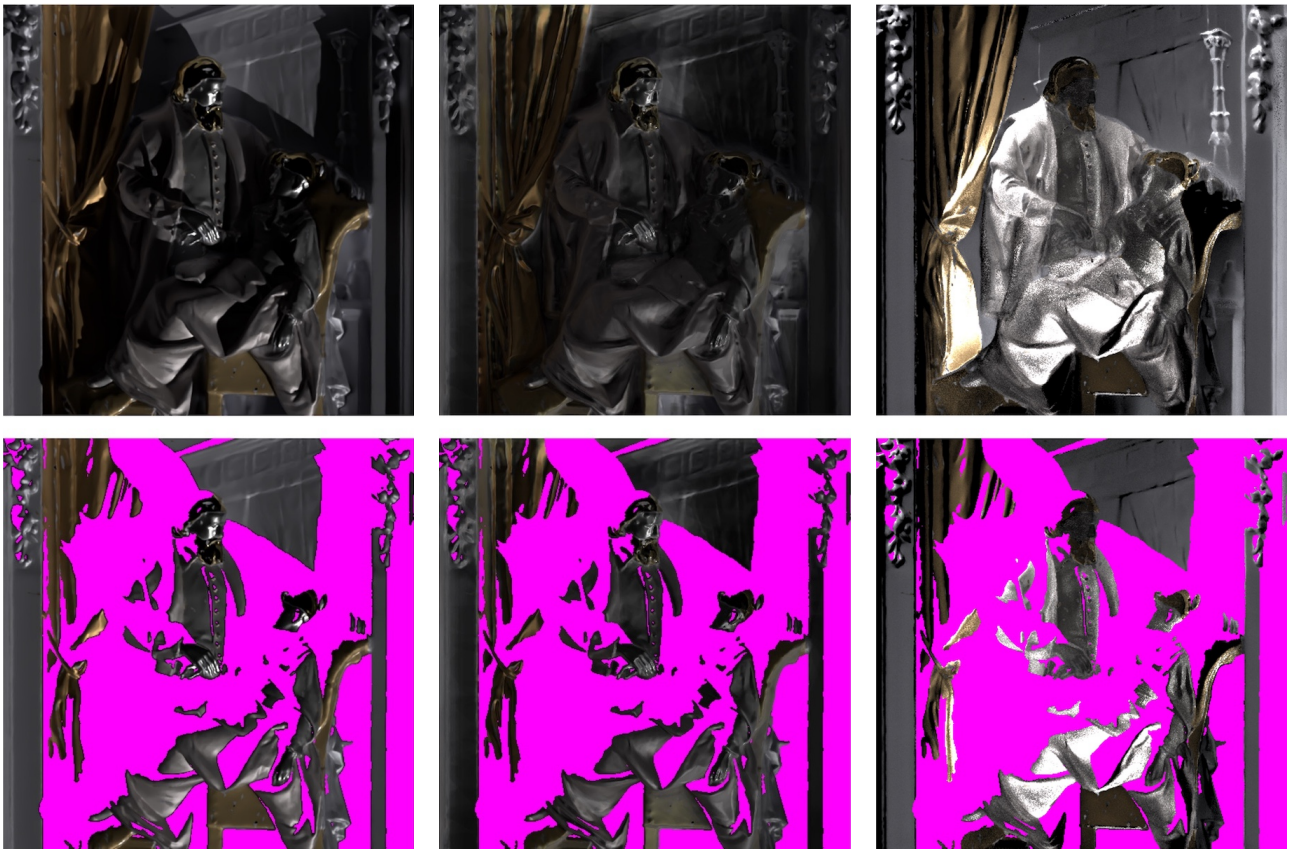
Figure 6.10: Computing metrics only on non-shadowed areas results in higher scores. For each material, the average is computed across all objects.

Figures 6.12 shows a comparison between one of the ground truth images from the MLIC, for object Doctor (M2), and its relighted versions using the two methods, showing the whole image together with a masked version where the shadowed area has been highlighted in magenta. The results obtained with the method from Ikehata would be quite bad in shadowed regions, but the use of the direct light illumination in that regions is not correct.



(a) PSNR *(b) SSIM*

Figure 6.11: Computing metrics only on non-shadowed areas results in higher scores. For each object, the average is computed across all materials.



(a) Ground truth *(b) Pintus et al.* *(c) Ikehata*

Figure 6.12: On the bottom row, images have been masked using the ground truth shadow map to show only non-shadowed pixels.

6.5 Depth from normals estimation

Ideally, the synthetic images created with the BASS-MLIC framework can be used to train or finetune models to estimate depth directly. However, BASS-MLIC can be used to evaluate the accuracy of the depth estimated with the standard Photometric Stereo approach, where normal integration techniques can be used to recover a metric depth value for pixel locations in a calibrated/known camera system.

Robust normal integration methods have been proposed to overcome the intrinsic limitations of the approach, especially in the case of discontinuities. Bilateral Normal Integration [61] handles well discontinuities by using a bilaterally weighted functional. To demonstrate the potential use of BASS-MLIC for depth estimation, we estimated the errors in depth estimations performed with bilateral integration of the normal maps computed with Lambertian PS.

Table 6.6 shows the Root Mean Squared Error (RMSE), calculated in millimeters, in a reference system where the size of one pixel is 4.4mm. It is possible to see how, for different materials, the depth estimation error varies, and is larger for objects with a higher overall depth variability, like Christ and Doctor. These objects have an extensive depth range (more than 1 meter), so the errors measured seem reasonable.

	Christ	Doctor	Cuneiform	Wall	Maya
S1	75.92	55.76	6.68	7.64	3.72
S2	66.18	43.61	24.58	11.7	15.05
S3	82.05	59.76	6.78	8.05	3.86
S4	73.24	54.15	6.87	6.97	3.78
M1	76.43	56.58	7.39	6.18	6.56
M2	65.26	49.67	14.87	6.26	10.18
M3	79.14	57.28	6.84	7.55	3.79
M4	78.6	59.51	6.51	8.27	3.75
Average	74.60	54.54	10.07	7.83	6.34

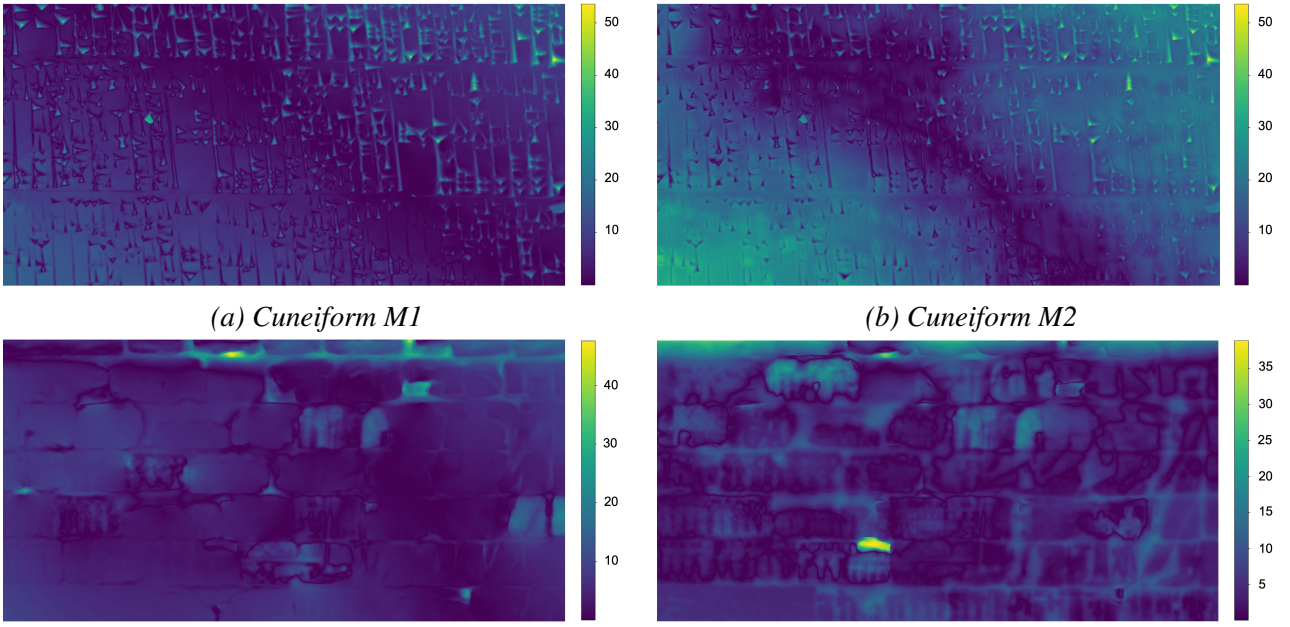
Table 6.6: RMSE for depth estimation, averaged across illumination directions, for each object and material.

Figure 6.13 shows an example of error maps for the objects Cuneiform and Wall, using materials M1 and M2. The error maps are obtained by computing the absolute difference between the ground truth depth map and the estimated one. The numeric scales are in millimeters.

These maps provide a visual cue of how the error distributes when the normal map is integrated. For example, for the object Wall, despite a similar score in Table 6.6, the error distribution between M1 and M2 is very different.

6.6 Outcomes and discussion

The examples provided are only preliminary results aimed at showing the potential of the BASS-MLIC dataset for developing different techniques tailored explicitly for reconstructing shape and materials from this kind of image collection, exploiting the most recent achievements in Computer Vision. However, they show some interesting facts that provide helpful hints for future work. Current methods for deep shadow estimation are not too effective for this kind of image. A larger synthetic dataset



(a) Cuneiform M1 *(b) Cuneiform M2*

(c) Wall M1 *(d) Wall M2*

Figure 6.13: Depth estimation error maps for objects Cuneiform and Wall, in the material configurations M1 and M2. Color mapping represents the per-pixel absolute difference between ground truth and estimated depth, in millimeters.

with varied materials and bas-relief objects could help the community train specific detection models. Similarly, a large dataset could help train neural networks to estimate normals or depths directly.

Neural Photometric Stereo methods [62] are now popular, but still perform poorly on complex materials and non-uniform surfaces. Monocular Metric Depth estimation methods [63] are currently revolutionizing the domain of Computer Vision, as they can provide reliable continuous maps even in regions with poor texture, but this happens only for the specific kind of images for which they are trained. Working on single frames of MLIC captures could be hard for these models, as there is insufficient context, but we could exploit multi-light information to adapt the approach to the case.

Extending BASS-MLIC by increasing the number of objects and materials, and providing two separate sections for training and testing of machine learning algorithms, would lead to more refined works for the tasks of shadow segmentation, normal estimation, and depth estimation. The availability of ground truth normals, depths, and illumination for a large set of images can also be exploited to train inverse rendering methods able to simultaneously estimate depth and normals by optimizing iterative rendering loss.

Chapter 7: Conclusion and Perspectives

7.1 Beyond Traditional RTI

This section is a collection of works, that include the use of RTI, other imaging techniques, and the combination of the two. Furthermore, the ideas presented here can be the starting point for new projects, and we will see how there is the potential for RTI visualization of being further improved and used in research activities.

7.1.1 Data annotation in OpenLIME

Nowadays, data annotation can be a critical task, necessary, for example, in Deep Learning algorithms to train models on ground truth data. From the point of view of a cultural heritage experts, however, annotating digital data of the objects they study can be fundamentally important for reasons of preservation and keeping track of their discoveries.

OpenLIME, as other tools, includes the possibility to annotate the image, using simple geometric shapes, text and drawings that are positioned on top of the image as an additional layer, and are saved in SVG format [64].

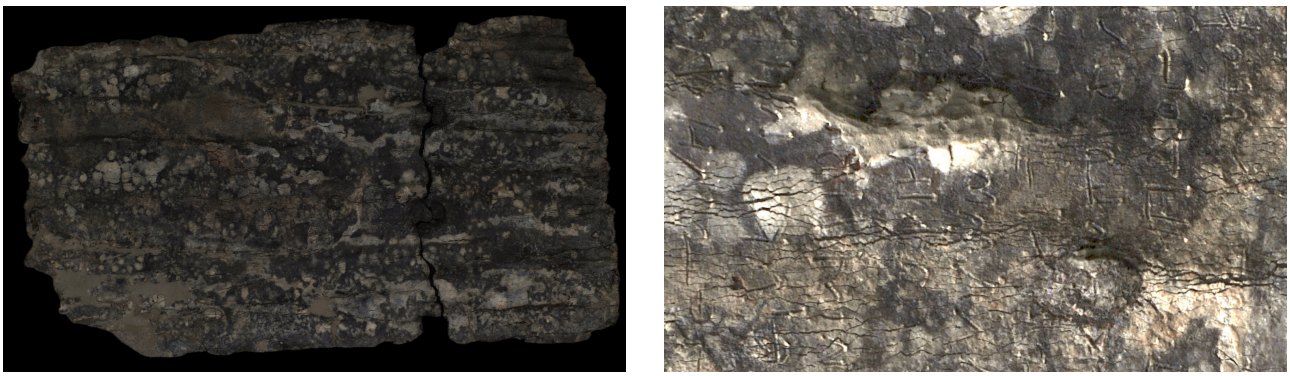
During the project, we had many ideas about how to carry out use case evaluations, and we also considered asking users to use the OpenLIME viewer to annotate data and give feedback about it. Despite this kind of works has not been continued, it is worth mentioning that we collaborated with an expert who was studying an old metallic plate, extremely ruined, with possible Greek incisions on it. The plate is shown in Figure 7.1.

We provided RTI visualization of the plate and the expert had the possibility to annotate the text directly on the relightable image, gradually, while trying to interpret new text. Figure 7.2 shows the final result of one side of this object, with the annotations.

From this first attempt of use case, that was focused on the use of RTI and the OpenLIME viewer for a quite specific task, i.e., the annotation of text, we then developed the work that has been discussed in Section 5.2, also thanks to the interest of conservators in analyzing philological artifacts.

7.1.2 RTI for translucent materials

This section proposes an idea about using RTI in an innovative way.The aim is to use RTI to analyze how light interacts with translucent materials, whereas it is usually used for opaque materials, analyzing



(a) Metallic plate *(b) Close up to better visualize engravings*

Figure 7.1: Metallic plate used during the annotations experiment. The close up picture (b) has been enhanced for visualization purposes.

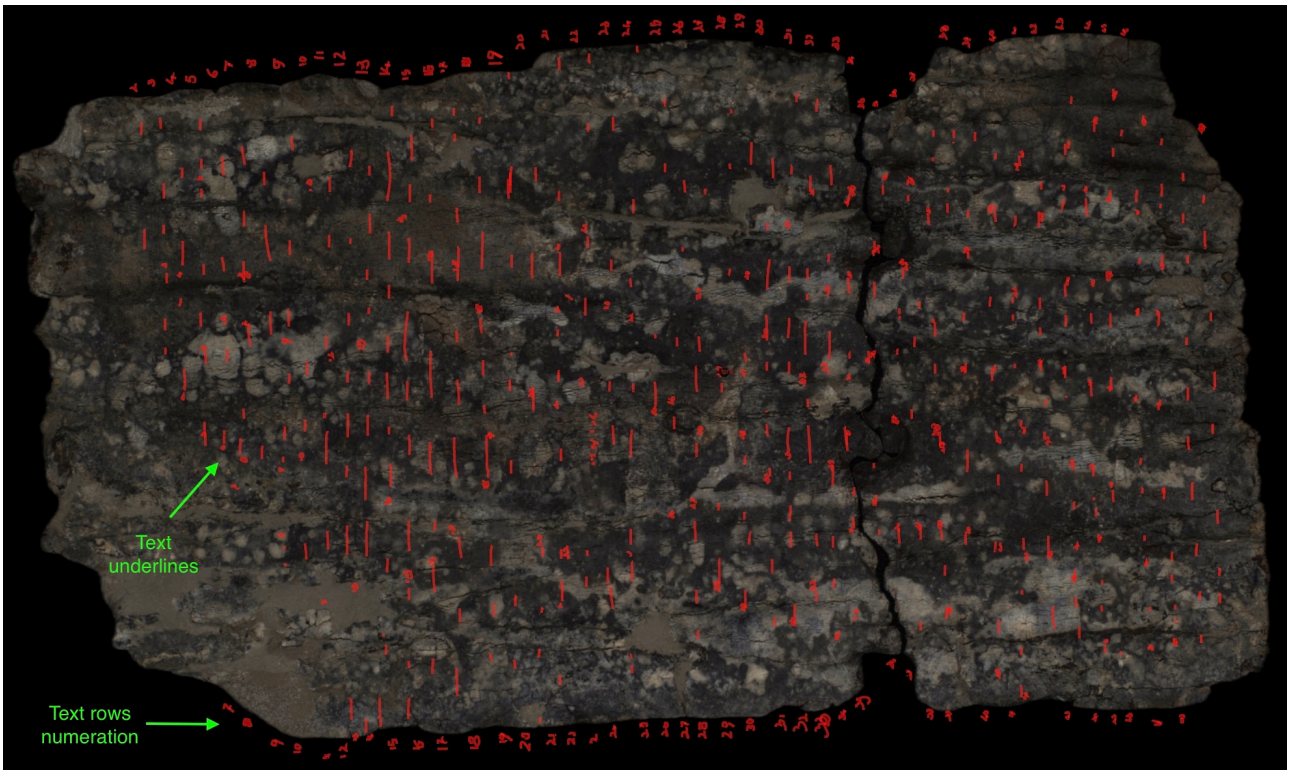


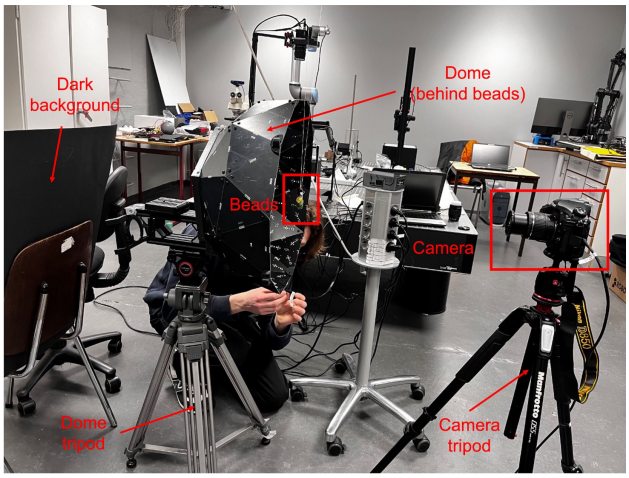
Figure 7.2: Text annotations on the metallic plate. The annotations were done by Attilio Matsrocinque, professor at University of Verona.

only how their surfaces reflect light. For this work, I collaborated with researchers from the Norwegian University of Science and Technology (NTNU) and the The Museum of Cultural History of Oslo (Kulturhistorisk museum).

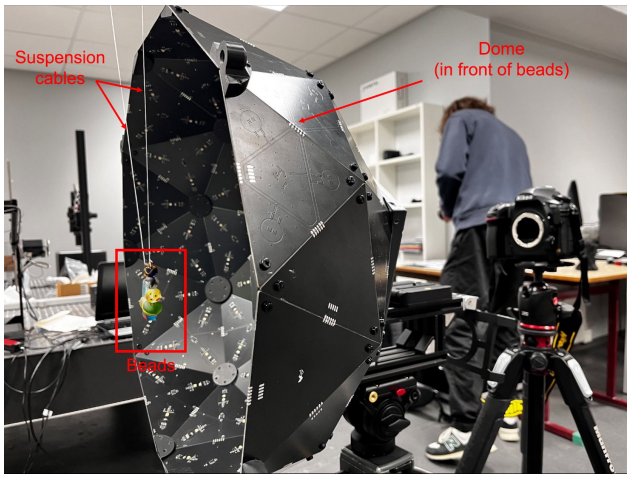
The aim was to analyze Viking beads made of translucent materials, and create a RTI model that could exploit the complex nature of these materials to provide a rich visualization experience. Therefore, we acquired multi-light images, illuminating the objects both from the front, i.e., from the same side as the camera, and from behind, i.e., from the opposite side. The devices specification used for the acquisition are:

- Dome: Mercurio RTI Dome (105 light positions)
- Camera: Nikon DSLR D800
- Lens: Nikon 18-55mm f/3.5-5.6G

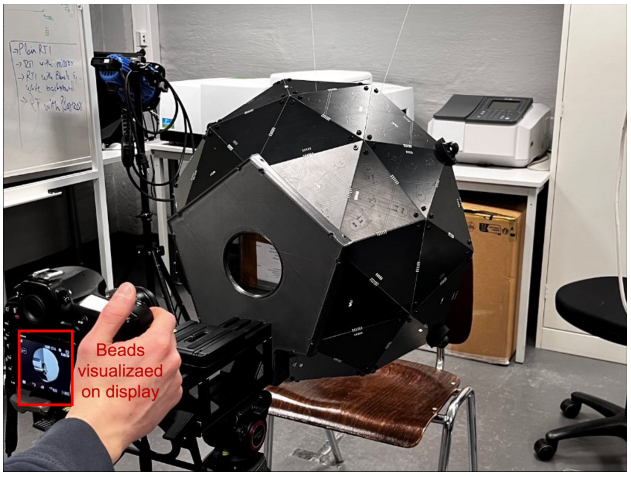
Figure 7.3 shows an overview of the acquisition setup. In order to position the dome behind the beads, these were suspended in mid-air using a cable. Both the camera and the dome were mounted on tripods, allowing for vertical adjustment. While the camera and beads remained stationary throughout the acquisition process, the dome was manually moved from side to side to simulate a spherical distribution of light sources. The pictures in Figure 7.3 were taken at NTNU's laboratory in Gjøvik, Norway, where the acquisition was performed.



(a) Dome positioned behind the beads



(b) Dome positioned in front of the beads



(c) Display visualization of the beads

Figure 7.3: RTI setup for acquiring translucent materials. The beads have been suspended mid-air using a cable, in order to allow the positioning of the dome behind them. Both the camera and the dome have a tripod for vertical positioning. The camera and the beads remain fixed, while the dome is moved from one side to another to simulate a sphere of light positions.

After the acquisition, my contribution has been generating RTI visualizations for both the setting with the lights in front of the beads and the one with the lights behind, and implementing them in OpenLIME in order to compare them and also adding the possibility to virtually combine the two

relightable images. Figure 7.4 contains two screenshots from the OpenLIME visualization of the beads, both illuminating the beads from the front and from the back. It is clear how the light's behavior and the material's appearance change in the two cases. After a quick comparison, the RTI model that we chose to generate the relightable images was NeuralRTI, as it appeared to better reproduce the colors and the complex behavior of these objects. However, a more in depth study would be necessary to assess which method could better fit this case study, possibly developing new solutions.

(a) Beads illuminated from the front *(b) Beads illuminated from the back*

Figure 7.4: Screenshots from the OpenLIME visualization of the beads. The model used to create the relightable images is NeuralRTI.

A possible future project could be to evaluate how the RTI methods developed until now can replicate the interaction of light with these types of materials, also creating a model that can relight the image 360 degrees, that is, illuminating the object from both the front and the back. The evaluation can be conducted both objectively, using metrics, or by having a group of users, including experts and museum visitors, compare the relightable images with the real objects.

7.1.3 Multispectral RTI

Collaborating with researchers from NTNU, we presented a methodology for detecting bruises in apples using hyperspectral imaging. By analyzing spectral data, we developed machine learning

models based on Principal Component Analysis (PCA) and band-ratio analysis to identify bruised areas before they become visible to the human eye.

We analyzed two specific types of apple: Alpe Golden green variety, and Royal Gala red variety. In the first case, we showed how bruises can be detected as early as 30 minutes after impact, and PCA components were particularly effective in highlighting affected regions. In the second case, bruise detection proved to be more challenging.

The spectral response showed minimal variation in the visible range (400–600 nm) between healthy and bruised tissue, with the most noticeable differences appearing in the near-infrared region (700–900 nm). Band-ratio analysis identified optimal wavelengths for bruise detection: around 650 nm and 870 nm for red apples, and approximately 550 nm and 850 nm for green apples. Figure 7.5 illustrates the experimental setup and it's taken from [8].

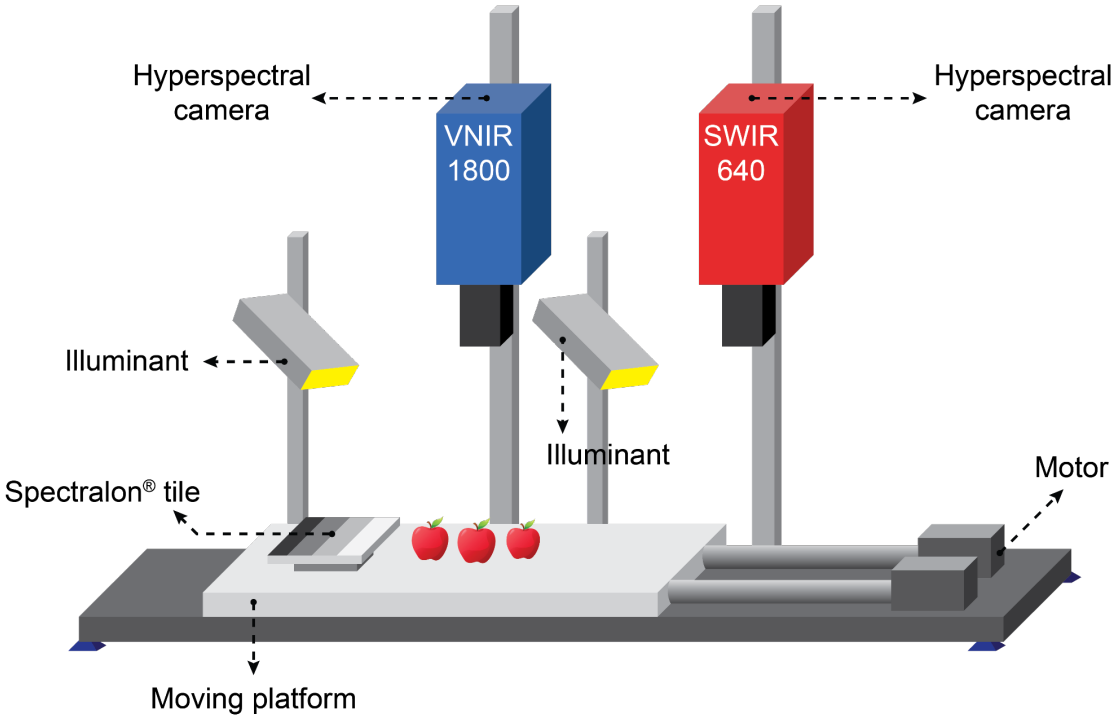


Figure 7.5: Schematic illustration of the experimental apparatus featuring dual hyperspectral imaging systems (VNIR-1800 and SWIR-640) mounted above a motorized platform along with calibrated illuminants.

This introduction leads us to the idea of combining multispectral imaging and RTI [65]. Although less common, some works have explored spectral RTI, which adds an extra dimension to the visualization process. This approach enables the detection of features that may not be visible in the standard visible spectrum [44, 45]. Furthermore, even works about computing PS algorithms from spectral images have been investigated [66].

Many works already addressed the problem of how many lights and how the lights should be distributed when acquiring a MLIC to create RTI models or compute PS [67–69]. However, the use of

different multispectral lights would require new research in order to determine which wavelengths and combination of light sources are the best configuration for specific analyzed objects.

Moreover, from the visualization point of view, it is interesting to reason about how viewers could be updated to allow a human-friendly visualization of this type of data. Works of this kind already exist [70], and the main problematic is that image formats developed to have just three channels (red, green and blue). Strategies could be to allow simultaneous view of different spectral ranges or create false color images where specific bands are chosen for visualization.

OpenLIME and its layered nature is a strong starting point to evolve this idea, as it was developed to combine together different versions (layers) of the same image. For example, The implementation of NeuralRTI into OpenLIME, that we discussed in Chapter 4, gives us the possibility to perform an interesting visualization when MLICs of the same object are acquired in different ways, like in this case where images in the visible, infrared, and ultraviolet range were acquired. This is because OpenLIME allows to create lenses on top of the relightable image [24], which in principle contain a different visualization for the same object. Taking the example from Section 3.3, Figure 7.6, which is taken from [1], shows the visible range relightable image as a whole, and the ultraviolet information in a lens.

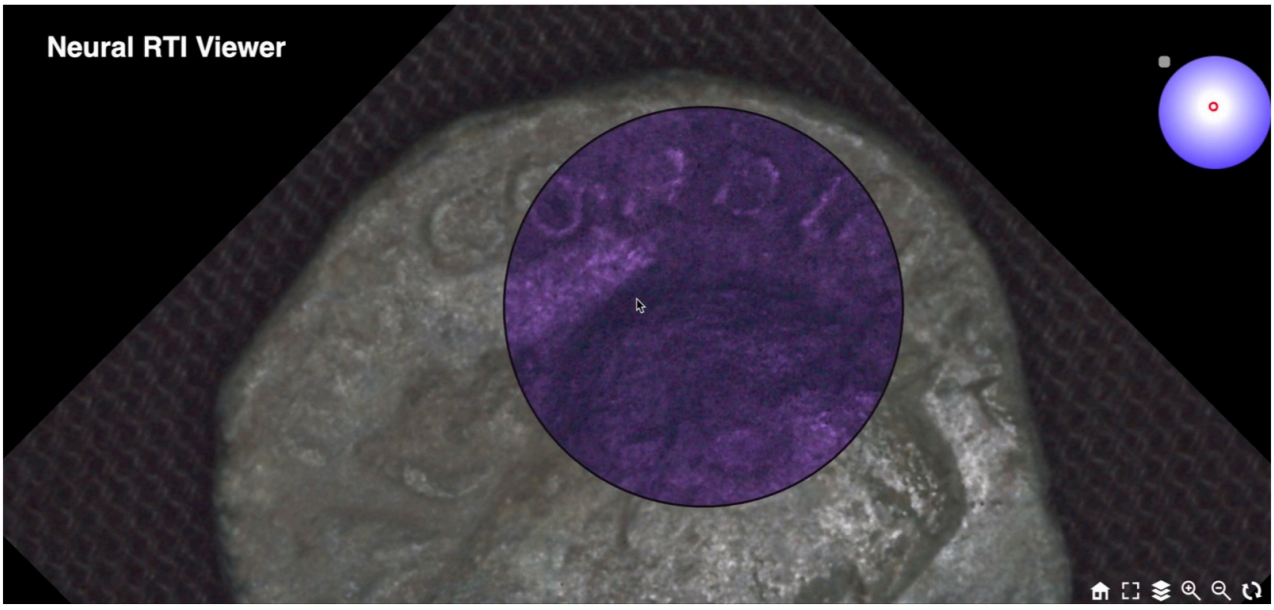


Figure 7.6: Example of OpenLIME's lens tool usage. The main layer shows the coin captured in the visible range, while the lens shows a portion of the coin captured in the ultraviolet range.

7.2 Final Considerations

During my PhD project I developed and evaluated imaging tools with focus on RTI and multi-light data, designed to support cultural heritage research. The main contributions have been advancements in real-time relighting, practical validation with domain experts, and the creation of new datasets to

support future research in RTI, multi-light but also inverse rendering techniques. These contributions, with their limitations and potential future directions are further explained below.

7.2.1 Contributions

A substantial contribution of this project has been the optimization of NeuralRTI to achieve real-time performance suitable for interactive applications. I explored different network configurations to find an effective balance between relighting quality and computational complexity. Thanks to collaborations with CNR-ISTI (Pisa) and CRS4 (Cagliari), NeuralRTI was integrated into OpenLIME, a web-based visualization platform. Since existing web-deployment tools were insufficient, I implemented NeuralRTI's decoding operations directly in a dedicated shader, allowing seamless real-time visualization inside OpenLIME's layered architecture.

A second contribution consists of the evaluations of RTI methods and their usability. Organizing user studies with cultural heritage specialists proved challenging due to the diversity and specificity of required expertise, yet two focused evaluations were successfully conducted. These studies assessed both the improvements introduced by NeuralRTI and the broader usefulness of RTI techniques through applications to damaged textiles and manuscripts. The results validated the practical relevance of the tools and highlighted how RTI can support detailed inspection tasks.

Another achievement is the design of a new synthetic, multi-light dataset which can be used for RTI, photometric stereo, and inverse rendering. Inspired by existing datasets such as SynthRTI and SynthPS, the BASS-MLIC dataset includes 3D models related to cultural heritage, rendered using multiple materials and lighting conditions. It offers not only the multi-light image collections (MLICs) but also ground-truth maps such as shadows, normals, and depth. These additions make the dataset suitable for investigating surface geometry, appearance modeling, and inverse rendering tasks.

7.2.2 Limitations

Limitations emerged during the development and evaluation of the proposed tools. First, while NeuralRTI demonstrated strong relighting performance, its design is inherently specialized: the network is always trained on a single dataset and is optimized for that specific reflectance behavior. This differs from the standard deep learning approaches, which would aim at generalize across diverse materials and lighting conditions. As a result, NeuralRTI cannot currently handle heterogeneous reflectance functions within a single model. Building a broader model capable of differentiating between multiple material types remains an open challenge.

Limitations also arise from the deployment of NeuralRTI inside OpenLIME. The use of WebGL places strict constraints on the number of external parameters that can be passed to shaders. Our implementation operates close to this limit, preventing the use of larger or more expressive neural

networks. Although emerging technologies like WebGPU could overcome these restrictions by supporting more advanced GPU features and modern machine-learning workloads, our goal of ensuring compatibility with less powerful hardware limited the adoption of such solutions within this project.

Further challenges emerged from the practical integration of NeuralRTI in a web-based environment. Decisions regarding feature image formats, 8-bit quantization, and the resampling introduced by tiled adaptive rendering all influence the fidelity of the reconstructed relighting. While preliminary assessments indicated that the resampling error was negligible for our model, partly because NeuralRTI maintains a relatively interpretable latent representation, these issues remain only partially addressed. Other models with more complex latent spaces might be more sensitive to quantization and could require dedicated training strategies to mitigate such artifacts.

Finally, conducting extensive user studies in cultural heritage contexts remained difficult due to the need for specialized expertise, resulting in fewer evaluations than initially planned. Additionally, the integration of advanced relighting models into web environments still requires custom, non-standard solutions, which may limit long-term maintainability.

7.2.3 Future directions

The work conducted opens several promising directions for future research. NeuralRTI, or a new architecture, can be extended with support for more complex materials, including translucency and subsurface scattering, to explore underrepresented relighting scenarios. User studies can be expanded with additional cultural heritage domains and tasks, enabling broader validation and refinement of visualization tools. The BASS-MLIC dataset has the potential to be used for assessment of existing techniques, or the realization of new methods for inverse rendering or surface property estimation. Multispectral RTI can be explored, combining spectral and geometric information to create richer visualization frameworks and new analysis workflows.

Taken together, the project successfully achieved its objectives. NeuralRTI was improved, deployed, and made publicly accessible through OpenLIME, supporting both research and practical conservation activities. The conducted evaluations demonstrated the usefulness of the proposed tools, while the new dataset and exploratory experiments set the stage for future developments in multi-light imaging. The combination of technical innovation, interdisciplinary collaborations, and applied validation ensures that this work contributes meaningfully to the research advancement of imaging for cultural heritage.

References

[1] L. Righetto, E. Gobbetti, F. Ponchio, A. Traviglia, M. De Bernardin, A. Giachetti, Ancient coins' surface inspection with web-based neural rti visualization, in: Optics for Arts, Architecture, and Archaeology (O3A) IX, Vol. 12620, SPIE, 2023, pp. 91–98.

[2] T. G. Dulecha, L. Righetto, R. Pintus, E. Gobbetti, A. Giachetti, Disk-neuralrti: Optimized neuralrti relighting through knowledge distillation, in: Smart Tools and Applications in Graphics-Eurographics Italian Chapter Conference, Eurographics, 2024, pp. 1–10.

[3] T. G. Dulecha, F. A. Fanni, F. Ponchio, F. Pellacini, A. Giachetti, Neural reflectance transformation imaging, The Visual Computer 36 (2020) 2161–2174.

[4] F. Ponchio, F. Bettio, F. Marton, R. Pintus, L. Righetto, A. Giachetti, E. Gobbetti, OpenLIME: An open and flexible web framework for creating and exploring complex multi-layered relightable image models, in: Digital Heritage 2025 - 4th International Congress & Expo, 2025.
URL https://openreview.net/forum?id=oh2a5YHsce

[5] L. Righetto, F. Bettio, F. Ponchio, A. Giachetti, E. Gobbetti, et al., Effective interactive visualization of neural relightable images in a web-based multi-layered framework, in: GCH 2023-Eurographics Workshop on Graphics and Cultural Heritage, 2023, pp. 57–66.

[6] L. Righetto, M. Khademizadeh, A. Giachetti, F. Ponchio, D. Gigilashvili, F. Bettio, E. Gobbetti, Efficient and user-friendly visualization of neural relightable images for cultural heritage applications, ACM Journal on Computing and Cultural Heritage 17 (4) (2024) 1–24.

[7] L. Righetto, G. Marchioro, A. Giachetti, Enhancing manuscript analysis with reflectance transformation imaging: a user study on interactive visualization options, in: Optics for Arts, Architecture, and Archaeology (O3A) X, Vol. 13569, SPIE, 2025, pp. 101–111.

[8] A. Fsian, M. A. Khawaja, M. Kresovic, L. Righetto, J. Y. Hardeberg, Apple bruise detection using hyperspectral imaging, in: Seventeenth International Conference on Digital Image Processing (ICDIP 2025), Vol. 13709, SPIE, 2025, pp. 784–795.

[9] T. Malzbender, D. Gelb, H. Wolters, Polynomial texture maps, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pp. 519–528.

[10] G. Earl, P. Basford, A. Bischoff, A. Bowman, C. Crowther, J. Dahl, M. Hodgson, L. Isaksen, E. Kotoula, K. Martinez, et al., Reflectance transformation imaging systems for ancient documentary artefacts, Electronic visualisation and the arts (EVA 2011) (2011) 147–154.

[11] R. Pintus, T. G. Dulecha, I. Ciortan, E. Gobbetti, A. Giachetti, State-of-the-art in multi-light image collections for surface visualization and analysis, in: Computer Graphics Forum, Vol. 38, Wiley Online Library, 2019, pp. 909–934.

[12] D. Guarnera, G. C. Guarnera, A. Ghosh, C. Denk, M. Glencross, Brdf representation and acquisition, in: Computer graphics forum, Vol. 35, Wiley Online Library, 2016, pp. 625–650.

[13] I. M. Ciortan, T. Dulecha, A. Giachetti, R. Pintus, A. Jaspe-Villanueva, E. Gobbetti, Artworks in the spotlight: characterization with a multispectral led dome, in: IOP Conference Series: Materials Science and Engineering, Vol. 364, IOP Publishing, 2018, p. 012025.

[14] M. Mudge, T. Malzbender, A. Chalmers, R. Scopigno, J. Davis, O. Wang, P. Gunawardane, M. Ashley, M. Doerr, A. Proenca, et al., Image-based empirical information acquisition, scientific reliability, and long-term digital preservation for the natural sciences and cultural heritage., Eurographics (Tutorials) 2 (4) (2008).

[15] S. Y. Elhabian, H. Rara, A. A. Farag, Towards accurate and efficient representation of image irradiance of convex-lambertian objects under unknown near lighting, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 1732–1737.

[16] M. D. Buhmann, Radial basis functions, Acta numerica 9 (2000) 1–38.

[17] A. Giachetti, I. Ciortan, C. Daffara, R. Pintus, E. Gobbetti, et al., Multispectral rti analysis of heterogeneous artworks (2017).

[18] M. Pistellato, F. Bergamasco, On-the-go reflectance transformation imaging with ordinary smartphones, in: Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I, Springer, 2023, pp. 251–267.

[19] J. Li, H. Li, Neural reflectance for shape recovery with shadow handling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 16221–16230.

[20] W. Yang, G. Chen, C. Chen, Z. Chen, K.-Y. K. Wong, S˜3-nerf: Neural reflectance field from shading and shadow under a single viewpoint, arXiv preprint arXiv:2210.08936 (2022).

[21] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 (2015).

[22] R. Scopigno, M. Callieri, P. Cignoni, M. Corsini, M. Dellepiane, F. Ponchio, G. Ranzuglia, et al., 3d models for cultural heritage: Beyond plain visualization., Computer 44 (7) (2011) 48–55.

[23] A. Jaspe-Villanueva, M. Ahsan, R. Pintus, A. Giachetti, F. Marton, E. Gobbetti, Web-based exploration of annotated multi-layered relightable image models, Journal on Computing and Cultural Heritage (JOCCH) 14 (2) (2021) 1–29.

[24] F. Bettio, M. Ahsan, F. Marton, E. Gobbetti, A novel approach for exploring annotated data with interactive lenses, in: Computer Graphics Forum, Vol. 40, Wiley Online Library, 2021, pp. 387–398.

[25] J. Jankowski, M. Hachet, Advances in interaction with 3d environments, in: Computer Graphics Forum, Vol. 34, Wiley Online Library, 2015, pp. 152–190.

[26] F. Ponchio, M. Corsini, R. Scopigno, A compact representation of relightable images for the web, in: Proceedings of the 23rd International ACM Conference on 3D Web Technology, 2018, pp. 1–10.

[27] D. Smilkov, N. Thorat, Y. Assogba, C. Nicholson, N. Kreeger, P. Yu, S. Cai, E. Nielsen, D. Soegel, S. Bileschi, et al., Tensorflow. js: Machine learning for the web and beyond, Proceedings of Machine Learning and Systems 1 (2019) 309–321.

[28] Y. Ma, D. Xiang, S. Zheng, D. Tian, X. Liu, Moving deep learning into web browser: How far can we go?, in: The World Wide Web Conference, 2019, pp. 1234–1244.

[29] M. Vedeler, The Oseberg Tapestries, Scandinavian Academic Press Oslo, 2019.

[30] C. F. Gulbrandsen, Reconstructing the Original: Machine Learning Puzzle Assembly for Matching Archaeological Textile Fragments., Master Thesis. Norwegian University of Science and Technology, 2023.

[31] D. Gigilashvili, H. T. Nguyen, C. F. Gulbrandsen, M. Havgar, M. Vedeler, J. Y. Hardeberg, Texture-based clustering of archaeological textile images, in: The Proceedings of Archiving 2023 Conference, Vol. 20, 2023, pp. 139–142.

[32] D. Gigilashvili, H. Lukesova, C. F. Gulbrandsen, A. Harijan, J. Y. Hardeberg, Computational techniques for virtual reconstruction of fragmented archaeological textiles, Heritage Science 11 (259) (2023) 1–16.

[33] R. Z. Fadillah, D. Gigilashvili, M. Havgar, M. Vedeler, J. Y. Hardeberg, Automatic thread counting for archaeological textiles, in: Proc. InART, 2024.

[34] Ø. Kolås, I. Farup, A. Rizzi, Spatio-temporal retinex-inspired envelope with stochastic sampling: a framework for spatial color algorithms, Journal of Imaging Science and Technology 55 (4) (2011). doi:10.2352/J.ImagingSci.Technol.2011.55.4.040503.

[35] A. Hore, D. Ziou, Image quality metrics: Psnr vs. ssim, in: 2010 20th international conference on pattern recognition, IEEE, 2010, pp. 2366–2369.

[36] P. Andersson, J. Nilsson, T. Akenine-Möller, M. Oskarsson, K. Åström, M. D. Fairchild, Flip: A difference evaluator for alternating images., Proc. ACM Comput. Graph. Interact. Tech. 3 (2) (2020) 15–1.

[37] G. Palma, M. Corsini, P. Cignoni, R. Scopigno, M. Mudge, Dynamic shading enhancement for reflectance transformation imaging, Journal on Computing and Cultural Heritage (JOCCH) 3 (2) (2010) 1–20.

[38] E. Frank, Lights, camera, archaeology: Documenting archaeological textile impressions with reflectance transformation imaging (rti), Textile Specialty Group Postprints 25 (2015) 11–42.

[39] J. Sanchez-Genao, Digital approaches to textiles, The Textile Museum Journal 51 (1) (2024) 134–143.

[40] M. Grzelec, T. Łojewski, Applications of reflectance transformation imaging (rti) for books and objects on paper, Journal of Paper Conservation 24 (1) (2023) 16–30.

[41] R. Clarricoates, E. Kotoula, The potential of reflectance transformation imaging in architectural paint research and the study of historic interiors: a case study from stowe house, england, Journal of the Institute of Conservation 42 (2) (2019) 135–150.

[42] D.-Ø. E. Solem, E. Nau, Two new ways of documenting miniature incisions using a combination of image-based modelling and reflectance transformation imaging, Remote Sensing 12 (10) (2020) 1626.

[43] O. Béthoux, A. Llamosi, S. Toussaint, Reinvestigation of¡ i¿ protelytron permianum¡/i¿(insecta; early permian; usa) as an example for applying reflectance transformation imaging to insect imprint fossils, Fossil Record 20 (1) (2016) 1–7.

[44] T. R. Hanneken, New technology for imaging unreadable manuscripts and other artifacts: Integrated spectral reflectance transformation imaging (spectral rti), Clivaz, Claire, Dilley, Paul, Hamidović, David, Ancient Worlds in Digital Culture, Digital Biblical Studies (DBS 1), Leiden/Boston, Brill (2016) 180–195.

[45] H. Mytum, J. R. Peterson, The application of reflectance transformation imaging (rti) in historical archaeology, Historical Archaeology 52 (2018) 489–503.

[46] R. Pintus, T. G. Dulecha, A. Jaspe, A. Giachetti, I. Ciortan, E. Gobbetti, et al., Objective and subjective evaluation of virtual relighting from reflectance transformation imaging data, in: EUROGRAPHICS Workshop on Graphics and Cultural Heritage, 2018, pp. 87–96.

[47] A. Giachetti, I. M. Ciortan, C. Daffara, G. Marchioro, R. Pintus, E. Gobbetti, A novel framework for highlight reflectance transformation imaging, Computer Vision and Image Understanding 168 (2018) 118–131.

[48] R. J. Woodham, Photometric method for determining surface orientation from multiple images, Optical engineering 19 (1) (1980) 139–144.

[49] S. Ikehata, Scalable, detailed and mask-free universal photometric stereo, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 13198–13207.

[50] R. Pintus, M. Ahsan, A. Zorcolo, F. Bettio, F. Marton, E. Gobbetti, Exploiting local shape and material similarity for effective sv-brdf reconstruction from sparse multi-light image collections, ACM Journal on Computing and Cultural Heritage 16 (2) (2023) 1–31.

[51] O. Wang, P. Gunawardane, S. Scher, J. Davis, Material classification using brdf slices, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 2805–2811.

[52] U. Rajapaksha, F. Sohel, H. Laga, D. Diepeveen, M. Bennamoun, Deep learning-based depth estimation methods from monocular image and videos: A comprehensive survey, ACM computing surveys 56 (12) (2024) 1–51.

[53] B. Shi, Z. Wu, Z. Mo, D. Duan, S.-K. Yeung, P. Tan, A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 3707–3716.

[54] J. Ren, F. Wang, J. Zhang, Q. Zheng, M. Ren, B. Shi, Diligent102: A photometric stereo benchmark dataset with controlled shape and material variation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12581–12590.

[55] F. Wang, J. Ren, H. Guo, M. Ren, B. Shi, DiLiGenT-Pi: Photometric stereo for planar surfaces with rich details-benchmark dataset and beyond, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 9477–9487.

[56] H. Guo, J. Ren, F. Wang, B. Shi, M. Ren, Y. Matsushita, DiLiGenRT: A photometric stereo dataset with quantified roughness and translucency, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 11810–11820.

[57] T. G. Dulecha, R. Pintus, E. Gobbetti, A. Giachetti, et al., Synthps: a benchmark for evaluation of photometric stereo algorithms for cultural heritage applications, in: Eurographics Workshop on Graphics and Cultural Heritage, 2020, pp. 13–22.

[58] Z. Li, Q. Zheng, B. Shi, G. Pan, X. Jiang, Dani-net: Uncalibrated photometric stereo by differentiable shadow handling, anisotropic reflectance modeling, and neural inverse rendering, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 8381–8391.

[59] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, C. Feichtenhofer, Sam 2: Segment anything in images and videos, arXiv preprint arXiv:2408.00714 (2024).
URL https://arxiv.org/abs/2408.00714

[60] S. Ikehata, Scalable, detailed and mask-free universal photometric stereo, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

[61] X. Cao, H. Santo, B. Shi, F. Okura, Y. Matsushita, Bilateral normal integration, in: European Conference on Computer Vision, Springer, 2022, pp. 552–567.

[62] Y. Ju, K.-M. Lam, W. Xie, H. Zhou, J. Dong, B. Shi, Deep learning methods for calibrated photometric stereo and beyond, IEEE Transactions on Pattern Analysis and Machine Intelligence 46 (11) (2024) 7154–7172.

[63] J. Zhang, Survey on monocular metric depth estimation, arXiv preprint arXiv:2501.11841 (2025).

[64] M. Ahsan, G. Altea, F. Bettio, M. Callieri, A. Camarda, P. Cignoni, E. Gobbetti, P. Ledda, A. Lutzu, F. Marton, et al., Ebb & flow: Uncovering costantino nivola's olivetti sandcast through 3d fabrication and virtual exploration (2022).

[65] B. Vandermeulen, H. Hameeuw, L. Watteeuw, L. Van Gool, M. Proesmans, Bridging multi-light & multi-spectral images to study, preserve and disseminate archival documents, Archiving2018: Final Program and Proceedings 2018 (2018) 64–69.

[66] G. Fyffe, Single-shot photometric stereo by spectral multiplexing, in: ACM SIGGRAPH ASIA 2010 Sketches, 2010, pp. 1–2.

[67] A. Spence, M. Chantler, Optimal illumination for three-image photometric stereo acquisition of texture, in: Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis, Citeseer, 2003, pp. 89–94.

[68] M. A. Khawaja, S. George, F. Marzani, J. Y. Hardeberg, A. Mansouri, Can surface topography give us best light positions for reflectance transformation imaging?, in: Archiving Conference, Vol. 20, Society for Imaging Science and Technology, 2023, pp. 12–17.

[69] M. A. Khawaja, S. George, F. Marzani, J. Y. Hardeberg, A. Mansouri, An interactive method for adaptive acquisition in reflectance transformation imaging for cultural heritage, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 1698–1706.

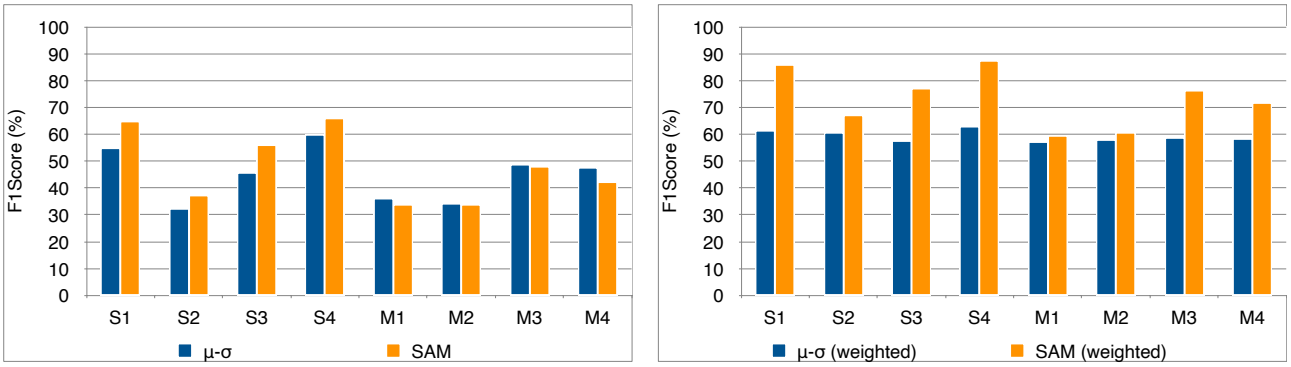
[70] P. Colantoni, J.-B. Thomas, M. Hébert, J.-C. Caissard, A. Trémeau, Web-based interaction and visualization of spectral reflectance images: Application to vegetation inspection, SN Computer Science 3 (1) (2022) 12.

Appendix A: Additional results on BASS-MLIC

Here we present additional results about the tasks that can be done using BASS-MLIC.

A.1 Weighted scores for shadow segmentation

In the case of 2.5D, multi-light data like those contained in BASS-MLIC, it is likely that images taken at a grazing light contains most of the shadows. Therefore, it can be interesting to look at the results for shadow segmentation weighting the score of each image by the amount of ground truth shadow that it contains. Looking at both Figure A.1 and A.2, what appears is that the models are less bad at estimating shadows if we consider mainly images where there is a high amount of them. Therefore, the main failure of the models is probably that they cannot discriminate shadows when the light comes from above and only a few pixels are shadowed. We see this aspect, for example, in Figure 6.7.

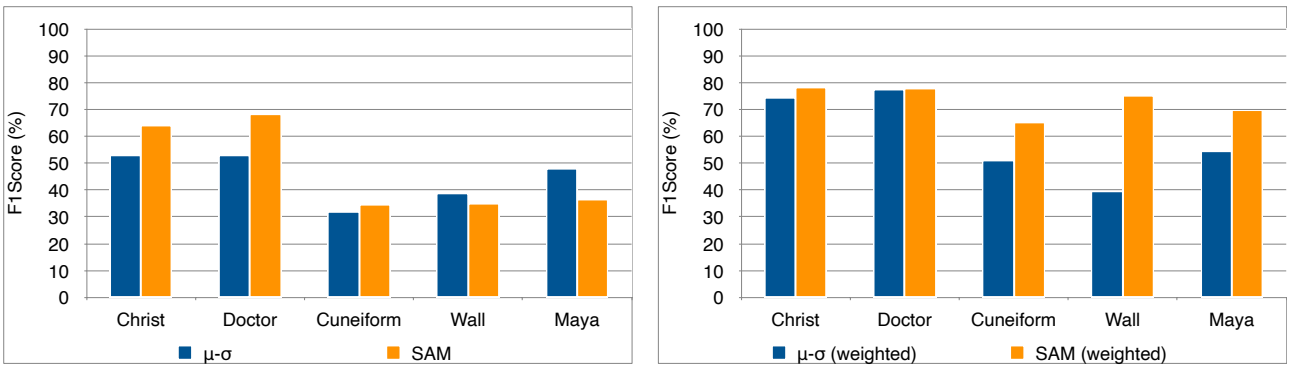


(a) F1 score *(b) Weighted F1 score*

Figure A.1: F1 score for shadow segmentation, averaging for each material across all objects. Arithmetic average (a) vs. weighted average giving more importance to images with more GT shadows, i.e., images from a grazing light (b).

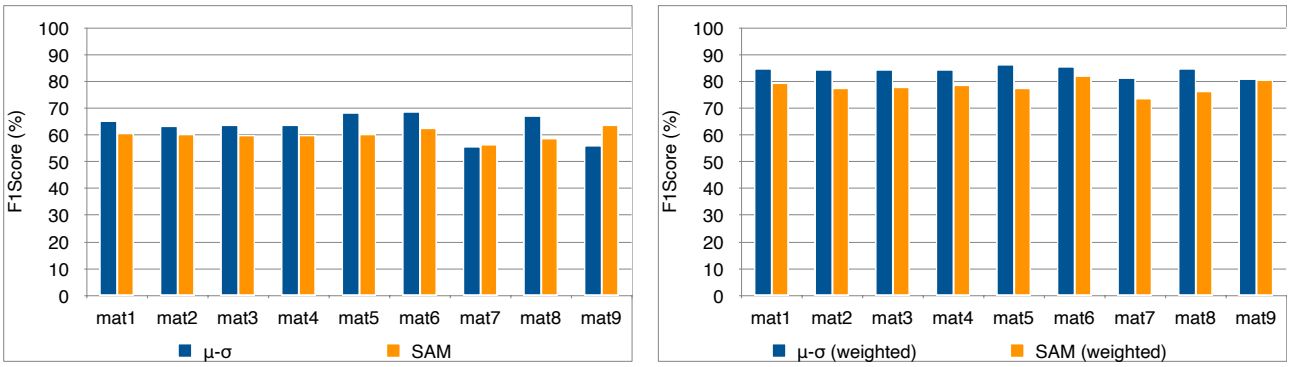
A.1.1 Results on SynthPS

The same results can be computed on other datasets which contain the ground truth maps for shadows. For example, we can check how the models perform on SynthPS (`https://github.com/Univr-RTI/SynthPS`), showing the results in Figure A.3.



(a) F1 score *(b) Weighted F1 score*

Figure A.2: F1 score for shadow segmentation, averaging for each object across all materials. Arithmetic average (a) vs. weighted average giving more importance to images with more GT shadows, i.e., images from a grazing light (b).

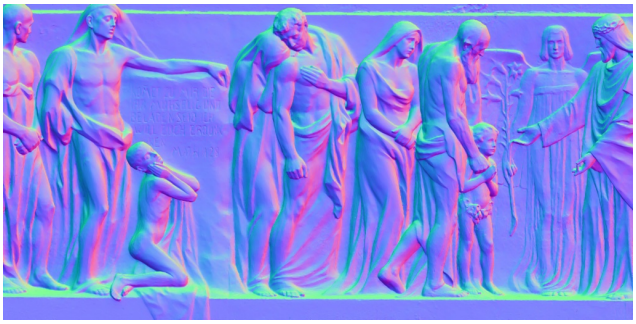


(a) F1 score *(b) Weighted F1 score*

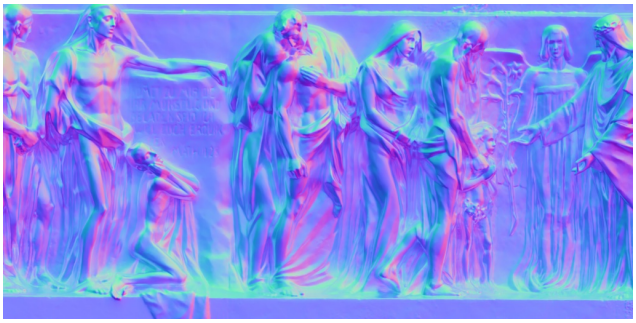
Figure A.3: F1 score for shadow segmentation, averaging for each material across all objects on the SynthPS dataset (single material). Arithmetic average (a) vs. weighted average giving more importance to images with more GT shadows, i.e., images from a grazing light (b).

A.2 Another example of normals estimation

As discussed in Section 6.3, the removal of shadows from normal map estimation methods can be further investigated, as in some cases might lead to worse results. The methods we used expects a lambertian material, and with metallic ones, which have strong specularity effects, only removing shadows does not compensate for the high nonuniform behavior of the material itself. In Figure A.4 we can see how the normal map gets worsen for the object Christ (S2), which is fully metallic, getting an average angular error of 19.73 when including the shadows, and one of 25.0 when removing them.



(a) Ground truth



(b) Lambertian PS

(c) Lambertian PS without GT shadows

(d) Angular error lambertian PS

(e) Angular error lambertian PS without GT shadows

Figure A.4: Normal map estimation. Comparison between ground truth data and lambertian PS. The angular error in (d) is calculated between (a) and (b), while the one in (e) is calculated between (a) and (c). The average angular error in (d) is 19.73, while in (e) is 25.0.