TU WIEN Informatics

Deep Learning-based Light Source Estimation from Face Images

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Philipp Hochhauser, BSc
Matrikelnummer 01527619

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Mag. Dr. Peter Kán

Wien, 1. August 2024

Philipp Hochhauser	Peter Kán

TU WIEN Informatics

Deep Learning-based Light Source Estimation from Face Images

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Philipp Hochhauser, BSc
Registration Number 01527619

to the Faculty of Informatics

at the TU Wien

Advisor: Mag. Dr. Peter Kán

Vienna, August 1, 2024

Philipp Hochhauser	Peter Kán

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Philipp Hochhauser, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. August 2024

Philipp Hochhauser

Acknowledgements

I would like to sincerely thank my supervisor, Mag. Dr. Peter Kán for supporting me in conducting this thesis. You always provided valuable tips and insights. Thank you for letting me freely explore the topic and applicable methodologies. All our discussions really helped me choose a suitable way to finish my thesis, and your positive attitude towards the topic greatly motivated me. I also want to thank all my friends and family, who have always supported and motivated me throughout my studies.

Kurzfassung

Diese Arbeit zeigt einen neuen Ansatz, um realistische Umgebungsbeleuchtung aus einem Gesichtsbild zu berechnen. Genaue Beleuchtungsinformationen sind essenziell für eine Vielzahl an Virtual und Mixed Reality Anwendungen. Um diese Information zu berechnen, benötigen Deep Learning Methoden eine große Anzahl an Daten, die nicht einfach zu erhalten sind. Wir behandeln dieses Problem, indem wir einen synthetischen Datensatz aus Gesichtsbildern und dazugehörigen Umgebungsbildern mithilfe von digitalen menschlichen Charakteren erstellen. Diese digitalen Menschen aus dem *MetaHuman* Framework werden von 360° Panoramabildern beleuchtet und mit *Unreal Engine* gerendert. Indem wir pro Bild mehrere Parameter automatisch verändern, erhalten wir einen diversen Datensatz mit über 150000 Bildern. Mit diesem Datensatz trainieren wir ein *CNN*, um die Helligkeit der Umgebung als Graustufenbild zu berechnen. Das Netzwerk kann sowohl für Innen- als auch Außenszenen die dominanten Lichtrichtungen erkennen, jedoch wirkt das Gesamtbild oft nicht wie ein Graustufenbild im ursprünglichen Format der Umgebungsbilder. In Beispielen mit echten Gesichtsbildern schafft das Netzwerk, die Position der Sonne zu erkennen. Um realistische Fotos statt Graustufenbildern zu erzeugen, wird ein existierendes *Diffusion Network* weitertrainiert. Dazu wird zusätzlich eine Textbeschreibung aus den Gesichtsbildern erstellt, um Kontext für das *Diffusion Network* zu liefern. Um das Netzwerk dazu zu bewegen, dem ursprünglichen Layout der Umgebung zu folgen, wird das berechnete Helligkeitsbild als weiterer Kontext angefügt. Das Gesamtsystem schafft es, realistisch aussehende 360° Panoramas zu erzeugen, die dem Layout der originalen Szene folgen. Zusammengefasst ist unser vorgestelltes System daher eine sequenzielle Abfolge mehrerer Neural Networks, ausgehend von einem einzelnen Gesichtsbild. Zuerst wird die Umgebungshelligkeit mithilfe eines *CNN* berechnet. Zusätzlich wird eine textuelle Beschreibung der Umgebung von demselben Gesichtsbild mithilfe eines existierenden Text-zu-Bild-Generators erstellt. Aus der Textbeschreibung wird von einem angepassten *Diffusion Network* ein Umgebungspanorama erzeugt. Dabei wird die berechnete Umgebungshelligkeit als zusätzliche Information verwendet. Diese Abfolge ergibt ein modulares System, um aus einem einzelnen Gesichtsbild die Umgebung als realistisch aussehendes Panoramabild zu generieren.

Abstract

This thesis proposes a novel method to estimate realistic-looking environment images from an input face image. Having correct light information is crucial for a variety of virtual and mixed reality applications, but training deep neural networks to calculate this information requires large datasets, which are not easily obtainable for pairs of face images and corresponding environment maps. We address this problem by creating a synthetic dataset using digital human characters from the *MetaHuman* framework. These human characters are illuminated by environment maps obtained from different sources and rendered using *Unreal Engine*. Through parameter augmentation, we achieve a diverse dataset of over 150000 face images with high-quality light information. Using this dataset, we trained a CNN to estimate the brightness of a scene given a single face image. The network is able to identify the most dominant light directions for most indoor and outdoor scenes, but sometimes fails in generating output that topologically matches the layout of equirectangular environment images. For unseen real-life examples of outdoor scenes, it was able to correctly identify the position of the sun. To enable generating realistic-looking images from text input, we finetuned a pretrained diffusion network on environment images. The text prompts are generated from face images using existing image-to-text models. By adding the estimated brightness images from our CNN, we can guide the model to follow the layout of the original scenes. Our final proposed pipeline is therefore a sequential combination of multiple different neural networks, starting from a single face image. First, the brightness of the surrounding scene is estimated from the face image with a CNN. Using the same face image, a text prompt that describes the surrounding scene is generated using a pretrained image-to-text model. Then, the text prompt is fed to a finetuned diffusion network which is additionally conditioned by the estimated brightness image. This yields a modular system for estimating the surrounding environment from a single image of a human face.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

Virtual and augmented reality applications have seen a significant increase over the last few years. There is a variety of applications, each with an individual target audience, ranging from end-user-focused smartphone games and image-editing apps, to medical applications, and even industrial applications. What all these use cases have in common is that rendering images as accurately as possible is crucial for creating an immersive experience. To enable high-fidelity rendering, the illumination present in a scene needs to be known or estimated. Otherwise, it is impossible to shade objects correctly. Having this information allows for tasks such as inserting virtual objects into a natural scene with accurate lighting and altering the look of an existing image by manipulating the estimated illumination and relighting the image.

Manually designing the light setup of a scene can be a labor-intensive and time-consuming task. Furthermore, it is not usable in real-time applications, where the lightning has to be calculated in fractions of a second. Debevec [Deb08] proposed capturing the environmental light of a scene by photographing a reflective steel ball, acting like a mirror, that contains all the available light information. This removes the need to manually recreate the environment, but still does not allow real-time applications and requires on-site presence when capturing the environments. Consequently, researchers began to create mathematical models to describe light information. They tried to map specific features of the real world, such as the presence of the sun in the upper hemisphere or the angle of the most dominant light direction, onto mathematical equations. Due to the use of these hand-crafted features, such systems would often only work in controlled environments or under certain assumptions.

Recreating the complex lighting of real-life scenes is a challenging task. Similarly to other computer vision-related tasks, deep learning-based approaches have emerged as prominent

solutions in the last few years, as with large enough datasets, machine learning systems can learn to estimate complex information. The acquisition of a large enough dataset is a problem due to the difficulty in capturing the ground truth environmental light of a scene. Most systems therefore use some form of synthetic data, which often lacks realism and can make neural models unable to adapt to real-world examples when the synthetic data is not diverse enough. Due to this lack of data, many systems place heavy constraints on the scenes they can reconstruct. Some can only classify indoor scenes, while others can only classify outdoor scenes or work only when specific features, such as shiny objects, are present. Although some systems can capture the overall intensity and most prominent light sources quite well already, the output often does not result in a photorealistic image. Artefacts of not having a photorealistic scene estimation can become apparent when rendering objects with reflective materials, where the actual environment is visible in the reflection.

1.2 Approach

This work proposes a method to achieve photorealistic light estimation from human face images by training multiple neural networks on a large synthetic dataset. Face images were chosen as the input due to their high occurrence in numerous scenarios, such as virtual try-on in augmented reality applications. Furthermore, face images always contain a similar distribution of features, revealing possible light directions.

The dataset consists of images showing individual human faces, and the corresponding scene illuminations as high dynamic range (HDR) images. Using more than 3000 distinct environments from different sources in combination with the *MetaHuman* framework [Gama], we were able to generate synthetic images of realistic-looking digital human characters, lit by the obtained environments. We created a simple setup where a *MetaHuman* character is placed in front of a camera and illuminated by an environment map. Each avatar was rendered with multiple environment maps, while randomly manipulating different parameters such as the rotation of the environment and facial expression of the rendered avatars. In this way, we created a dataset that contains 147056 images in the training set and 11341 images in the test set.

We trained a convolutional neural network (CNN), based on existing state-of-the-art architectures, to estimate the scene illumination given a face image. The CNN compresses the face image to a small latent vector and subsequently upscales that vector to generate an image that contains the brightness of the environmental lighting. As the output of this CNN is quite noisy and contains no information on the color of the illumination, we finetuned a diffusion network on tone-mapped HDR environments to create a network that can generate photorealistic panorama images from a single prompt. Combining these two networks with conditional input in the form of a *ControlNet* [ZRA23], which adds the estimated brightness calculated using our CNN as conditional input, yielded a system to estimate photorealistic panorama images from a single face image.

1.3 Research Questions and Contributions

The main contributions of this thesis try to answer the following research questions:

- *Will the rendering of MetaHuman characters in Unreal Engine allow us to create a realistic-looking dataset for Light Source Estimation from Face Images?* We describe a method for generating a realistic-looking dataset using *Unreal Engine*. The system outputs a rendered image of a human avatar, illuminated by a given environment map. By randomly manipulating multiple parameters, such as the azimuth angle of the environment map and the pose of the avatar, the system can create diverse data.
- *Using this synthetic dataset, can we train a network to estimate the brightness from real face images?* Using our synthetic dataset, we successfully trained a convolutional encoder-decoder network capable of capturing the most prominent light direction from a real scene in an outdoor setting.
- *Can we condition a diffusion-based network on the estimated brightness of a scene, to create panorama images that plausibly describe the light information of a given scene?* We present a novel deep learning-based method to create realistic-looking panorama images from a single face image. As the generated images are conditioned on the estimated brightness of a given scene, we can create photorealistic panorama images where the overall appearance and dominant light direction closely match the ground truth, provided the brightness was estimated well enough by the initial network.

1.4 Structure of the Thesis

We first present the theoretical background and the current state-of-the-art in light source estimation from face images in Chapter 2. In Chapter 3 we explain how we were able to create a realistic-looking dataset and present some necessary preprocessing steps to be able to use the dataset for training neural networks. Chapter 4 describes the main contribution of this thesis, the light source estimation system. We first show the CNN-based approach to estimate the brightness of the scene, and then describe our novel approach to train a diffusion-based network conditioned on the estimated brightness to create a realistic-looking panorama image. In addition, we also evaluate both of these tasks and explain our design choices in the same chapter. Finally, we summarize our work in Chapter 5 and provide ideas for future work based on our contributions.

CHAPTER 2

Theoretical Background and Related Work

This chapter describes a brief theoretical background and the current state of the art in light estimation. As this thesis and current state-of-the-art systems heavily depend on deep learning methods, Section 2.1 gives a summary of important concepts in deep learning. We describe how light information can be represented in Section 2.2 and then explore different methods of generating datasets to train light estimation systems on face images in Section 2.3. This should give an overview of the theory needed to understand current and previous state-of-the-art light estimation systems. Finally, Section 2.4 presents the evolution of the current state of the art from classical, non-deep learning-based approaches, to CNN-based methods, and to methods based on generative models.

2.1 Deep Learning

Deep learning has become the most dominant method for virtually any visual computing-related task over the last few years. Knowing and understanding the fundamentals is crucial for understanding current state-of-the-art systems in light estimation. In this section, we summarize some key deep learning concepts and network architectures as presented in two books *Deep Learning* [GBC16], by Goodfellow et al. and *Dive into Deep Learning* [ZLLS23] by Zhang et al. Overall, the task of defining and training a (deep) neural network can be described in the following steps:

- *Defining the architecture.* The most basic layout of the network has to consist of an input layer that can accept values in the form of the input data and an output layer, that outputs values in the desired output format. Hidden layers are added between the input layer and the output layer to allow the network to learn more

complex functions. A layer typically consists of one or more learnable parameters and must be differentiable. Many different layer types can be chosen depending on the purpose of the network or the topology of the data. Activation functions are commonly used to introduce non-linearity.

- *Choosing a loss function and optimization method.* A loss function is used to measure how well a model performs. Common loss functions such as the squared error, calculating the squared difference between the input and output, or hand-crafted functions can be used. The loss function needs to be differentiable such that an optimization algorithm can update the network parameters based on this loss function. Optimization algorithms are usually some modern variation of the gradient descent algorithm.

- *Training the network.* During training, an input is fed into the network to compute an output. The output is compared to the ground truth using the loss function. The optimization algorithm then updates the weights according to this loss. Gradients are calculated using the backpropagation algorithm.

- *Evaluation and hyperparameter tuning.* The performance of the training should be continuously measured during training, by calculating different metrics. Different combinations of hyperparameters, such as the number of epochs, the loss function, or different parameters of the optimization algorithm, can be optimized by running multiple training runs.

2.1.1 CNN

Since their introduction in 1995 by LeCun and Bengio [LB95], networks using convolutional layers, also known as *ConvNets* or CNNs, have become the standard in processing image data with neural networks. Through their use of convolutions, they can extract features based on the structure of the grid-like input data. Figure 2.1 shows a comparison of *LeNet* [LB95] by LeCun and Bengio from 1995 to the more advanced *AlexNet* [KSH12]. Both networks use similar layers. Convolutional layers, visualized in a light blue color, were used to sparsely connect neurons from one layer to the next, while utilizing the grid-like topology of the input data. The kernel size ranges from 3x3 to 11x11. The pooling layers, visualized in a dark blue color, were added to reduce the dimensionality by outputting a single value for each input to the kernel. Fully connected layers, visualized with no background color, are used to transform the data from the grid-like structure into a single output vector. The total number of layers was still quite low for AlexNet compared to modern neural networks. A key contribution to allow for deeper CNNs was proposed by He et al. [HZRS16] by introducing residual layers. Residual layers allow blocks to more easily learn the identity function by adding a skip connection from the input of the block to the output. CNNs are still widely used as a feature extraction backbone for various tasks such as object tracking, object detection, and semantic segmentation.

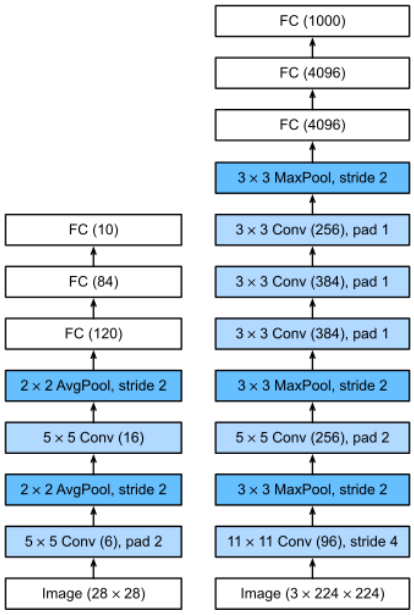


Figure 2.1: Comparison of two well known CNN architectures. The left graph describes *LeNet* [LB95] and the right graph describes the more advanced AlexNet [KSH12]. Image obtained from the *Dive into Deep Learning* book [ZLLS23].

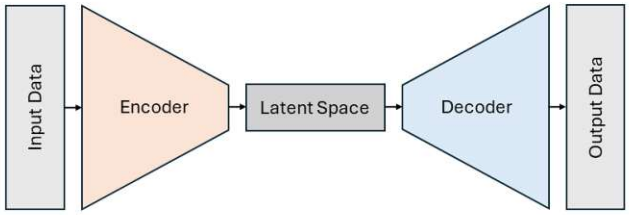


Figure 2.2: Visualization of a typical autoencoder network. Input data is transformed into a low-dimensional latent space representation by an encoder network. The decoder network tries to replicate the input data by decompressing the compressed latent representation.

2.1.2 Autoencoder

Autoencoders consist of two modules, an encoder, and a decoder, consequently they are often referred to as encoder-decoder networks. Figure 2.2 visualizes this concept. The encoder module compresses the input into a lower-dimensional representation, whereas the decoder module tries to decode this compressed data into a higher dimensionality, ideally creating a copy of the initial input. There are many different forms of autoencoders, and they are often part of more complex network architectures. An important variation

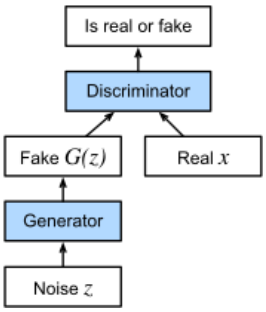


Figure 2.3: Visualization of a typical generative adversarial network (GAN). The generator creates fake data from some noise that the discriminator has to distinguish from real data. Image obtained from the *Dive into Deep Learning* book [ZLLS23].

of the classic autoencoder are variational autoencoders (VAEs) [KW22], proposed by Kingma and Welling in 2013. Their key contribution is that, instead of simply outputting a latent variable from the encoder, the output of the encoder describes parameters of a Gaussian distribution. By enforcing this constraint, the latent space will be structured and continuous. This enables generative capabilities for autoencoders, as points that are close in the latent space are also similar after decoding. This characteristic is not given for standard autoencoders, where encoding an input x into latent space z and then decoding a point $z+\epsilon$ very close to z in latent space could produce a meaningless result.

2.1.3 Generative Adversarial Networks

The introduction of GANs by Goodfellow et al. [GPM+14] marks a breakthrough in the design of generative models. In addition to training a generative model, they propose training a second neural network, called a discriminator, to classify whether data was generated by the generative model or originates from the true data distribution. Figure 2.3 shows a high-level visualization of this adversarial network structure. Training a GAN is a back-and-forth between the discriminator getting better at distinguishing fake data from real data and the generator learning to trick the discriminator into thinking that generated data is actually real.

2.1.4 Diffusion Models

With the paper *Denoising Diffusion Probabilistic Models* [HJA20] Ho et al. presented a new way to synthesize images in the form of diffusion networks. Other network architectures typically go from input noise to output data in a single step by passing the input through the network. The idea of diffusion networks is to instead use a finite number of steps to gradually generate data from noisy input. Their proposed solution consists of a forward diffusion process generating random noise given an image, and a reverse process generating an image given random noise. Figure 2.4 visualizes the reverse diffusion process by going from random noise $\mathbf{x}_T$ to an image $\mathbf{x}_0$ in T finite timesteps. The following two paragraphs explain these two processes in more detail.

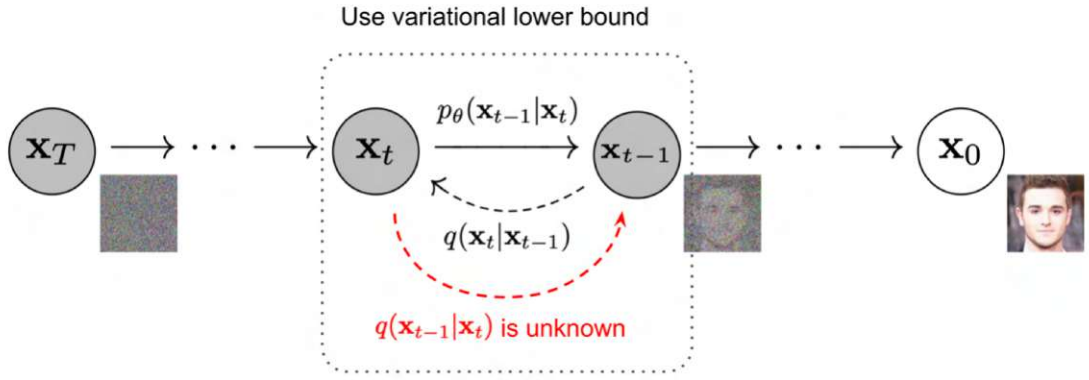


Figure 2.4: Visualization of the reverse diffusion process. As the actual distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is not known, a neural network p with learnable parameters θ is trained. Image is a modified version [Wen21] of an original image from the paper *Denoising Diffusion Probabilistic Models* [HJA20].

Forward Process When adding random Gaussian noise in each step t from 0 to T, the result will eventually end up with pure random noise. The image $\mathbf{x}_t$ at timestep t can be described by a Gaussian distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ of $\mathbf{x}_t$ given $\mathbf{x}_{t-1}$. The strength of the Gaussian is dependent on the current timestep t and some noise scheduling function.

Reverse Process To reverse the forward process, one would need to sample the reverse distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ which is not known. The authors show that for small timesteps, the reverse process can be estimated to also be a Gaussian distribution and train a neural network p to estimate the parameters of this Gaussian distribution. This means that the neural network is called once for each timestep. The inputs for this diffusion network are the noisy image $\mathbf{x}_T$ and the current timestep t.

To condition the image generation, most diffusion networks use an additional input vector generated by encoding a text prompt with the *CLIP* model [RKH+21]. Several contributions have improved on the proposed solution by Ho et al. [HJA20]. One of the most important contributions is by Rombach et al. [RBL+22]. They claim that input images are very high dimensional and that a lot of the actual information is irrelevant for a generative system. Instead of focusing on pixel-level information, they want to focus on the semantic meaning of the input instead. They propose using an autoencoder to compress the image into a latent space and performing the diffusion operations in this lower-dimensional space. This vastly increases training speed. Additionally, they propose a network architecture with conditional input in the form of a masked input image. Using this additional input, they are able to inpaint the conditional image in the masked regions. The result of their contributions are the well-known *Stable Diffusion (SD)* models.

Training Diffusion Networks

Training a diffusion model takes a lot of computational power. To efficiently finetune models on a specific domain, researchers have invented various methods. *Low-Rank Adaptation (LoRA)* [HSW+21] invented by Hu et al. was originally proposed to improve the training speed for large language models (LLMs). Their key insight is that instead of retraining all parameters of a network, rank decomposition matrices can be inserted and trained instead. They claim to reduce the number of trainable parameters by up to 10000 times. The *Dreambooth* [RLJ+23] training method proposed by Ruiz et al. describes a novel way to enable few-shot image generation for a specific subject. This method does not alter the architecture of the network but finetunes the network on a unique token. The authors present a method to find tokens that work well with their training technique. Common English words do not allow the networks to train well, because the network has a lot of prior knowledge on them. According to the authors, choosing a random token also does not work well. When the chosen token is present in the input text prompt, the network trained with this method can successfully generate high-fidelity images of the specified subject in novel contexts. Summarizing, the two mentioned training methods together allow for efficient finetuning of a diffusion network on a specific style or subject, with a low number of input images. However, there is no way to easily control the models without retraining the whole network on a different architecture. The authors of *ControlNet* [ZRA23] propose to train a copy of a *SD* network on images showing a specific condition for the generated image. Examples for conditional input images include depth maps, edge maps, normal maps, or skeletal poses. The output of this *ControlNet* is appended to the original model using convolution layers with a kernel size of 1x1. These layers are called "zero convolution layers", as both the weight and bias are initialized as zero. The authors demonstrate that multiple conditions can be used by concatenating different *ControlNets*. Once a *ControlNet* is trained, it can be used on different diffusion models, as long as the architectures match. This enables reuse of a few *ControlNets* for many different systems, as the conditional input is usually independent of the style or subject that should be generated from the diffusion model.

2.2 Light Representation

In order to use neural networks for light estimation, some form of representing the light information is needed. This influences the choice of the final output layer in the network architecture. Lighting estimated from a scene is typically either represented in parametric form, described by mathematical formulas yielding different light representations based on parameters, or in an image format. The lighting representation should be as accurate as possible and describe the light intensity as well as its color and the background texture.

As Einabadi et al. [EGH21] describe, Parametric light models include Spherical Harmonics [KK14], dominant light direction [KK19], or the Lalonde-Matthews outdoor illumination model [LM14]. Image-based light models include spherical or rectangular environment maps [CLG+18, FCZ+24], which are usually directly generated from some deep neural

network. Most rectangular 360 degree panorama images are represented as equirectangular images, where spherical coordinates are projected onto a rectangular image [SG17]. Some parametric models are discretized into a matrix, resulting in an image-based representation. In standard low dynamic range (LDR) image formats, the dynamic range is limited to 256 values. The underexposed and overexposed areas are clipped. Image-based representations therefore typically use HDR image formats to cover the wide dynamic range or real-world illumination.

2.3 Face Image Datasets

Together with the network architecture and a way of representing the estimated light, a good dataset may be the most important part to train a deep learning-based light estimation system. In this section, we describe a few ideas that researchers have used to create real and synthetic datasets.

2.3.1 Real Datasets

Obtaining a pair of face image and corresponding environmental light description can either be done by simultaneously taking a face image and recording a lightprobe, or by using a controlled setup where the environmental light is already known at the time of taking the face image. Both methods are not trivial and require specialized hardware. Calian et al. [CLG+18] captured the environmental light, by using a robotic tripod and taking multiple images for each orientation under different exposure settings. These image sequences were then merged into a single HDR image. They immediately afterward removed the tripod and photographed a person at the same location, creating pairs of face images and corresponding illumination maps.

2.3.2 Synthetic Datasets

Calian et al. [CLG+18] created a synthetic dataset by randomly sampling a statistical face model [PKA+09] and then rendering that model with environment maps. Yi et al. [YZTL18] used the same face model to render synthetic images. To improve the performance of their system on real data, they proposed an unsupervised finetuning method based on a large dataset of celebrity images [GZH+16]. Sun et al. [SBT+19] and LeGendre et al. [LMP+20], used a setup of multiple LED lights tessellating a sphere with a person sitting inside to create synthetic data using a method they call "one-light-at-a-time" (OLAT). Existing environment maps can be projected onto the images generated in the OLAT setup, to create persons rendered with the given environment map. To increase the number of environment maps that can be used to relight the persons, LeGendre et al. [LMP+20] placed three small spheres with different reflective properties in front of a camera while filming different environments. They calculated an LDR light representation by using the appearances of the three spheres and upscaled it to estimate an HDR representation by solving a system of linear equations. Fei et al. [FCZ+24] used publicly available 3D face scan data with a physically-based renderer

to generate synthetic data. They applied data augmentation such as random resizing, cropping, white balance adjustments, and added random Gaussian noise. They used additional datasets featuring a larger range of skin tones to reduce the bias of data consisting mostly of lighter skin tones.

2.4 Light Source Estimation

2.4.1 Light Source Estimation on General Scenes

With the rise of deep learning-based approaches to tasks in visual computing, many researchers started using network architectures like (residual) CNNs or encoder-decoder networks for light estimation. Zhang and Lalonde [ZL17] proposed a system to extrapolate HDR information from a single LDR panorama. They used an autoencoder network with additional skip connections between each encoder layer and the corresponding decoder layer with the same resolution. In addition, they added a second head to the network after the latent vector, using only fully connected layers to calculate a single value representing the angle of the sun elevation. Gardner et al. [GSY+17] proposed a system to estimate HDR illumination from a single LDR image with a small field of view. They used a convolutional autoencoder with residual layers and two distinct decoder networks. One decoder branch estimates the RGB texture of the light, while the other decoder branch estimates the light intensity in logarithmic scale. Both outputs are represented as equirectangular panorama images. In the paper *Deep Parametric Indoor Lighting Estimation*, Gardner et al. [GHS+19] presented a different method to generate light information from a single image. Compared to the work in 2017 [GSY+17], this method uses a parametric description of the light sources instead of outputting two image representations, greatly reducing the size of the decoder part. By using a pretrained feature extraction network trained on a very large dataset, training speed and stability were improved, according to the authors.

More modern light estimation systems typically use GANs instead. Weber et al. presented a system that allows for editable indoor light estimation. Editable light information is generated by a CNN and editable scene layout is generated by a GAN. Together with the original input image, this information is fed into a third network to output the final HDR environment. Dastjerdi et al. proposed a system to allow for editable light estimation from a single input image by using a GAN that is guided by encoded light. The encoded light is estimated with a network similar to the light estimation network of Gardner et al. [GHS+19] and transformed into a parametric representation of spherical Gaussians. This parametric representation can be edited by the user. Wang et al. [WYLL22] presented a similar idea.

The system of Tang et al. [TZC+23] generates panorama images using a diffusion model by simultaneously generating eight 90 ° images with a 45 ° overlap, resulting in a full 360 ° panorama image. To achieve this, they proposed a novel network block called *correspondence-aware attention* to merge the features of the multiple views together. This representation is not consistent with equirectangular panorama images and can

cause artefacts when used as a substitute. Wang et al. [WCL+23] proposed a system that first projects multiple low field-of-view images with unknown orientation onto an equirectangular image and then uses an outpainting diffusion model to generate a consistent panorama image. During training, they enforce the rotational consistency of panorama images by altering the latent vectors. At inference time, they additionally pad the left and right part of the latent vectors to prevent artefacts on the border of the generated images. Building upon this idea, Feng et al. [FLCX23] proposed a simpler system which blends the latent vectors at inference time with variable weights. They claimed not to need any modification of the training process, but instead directly enable finetuning a diffusion network with their proposed adaption.

2.4.2 Light Source Estimation on Human Face Images

Similar to light estimation on general scenes, in recent years research has shifted from calculating the light using hand-crafted features to CNNs to GANs or diffusion-based models. Knorr and Kurz [KK14] proposed a solution that generates a spherical harmonics light representation. They calculated specific feature points on a face and obtain radiance transfer functions by least-squares optimization. Calian et al. [CLG+18] estimated outdoor lighting by calculating a high-resolution mesh of the face and minimizing a loss based on priors of face albedo and possible outdoor lighting. Sztrajman et al. [SNWS20] proposed a system for outdoor light estimation that first trains an encoder-decoder network on environment maps. The encoder module is then replaced by a CNN-based encoder to estimate the latent bottleneck vector from a greyscale face image. They split their input HDR data into two parts, based on a threshold of the pixel brightness. The high-intensity part is trained to fit a Gaussian distribution, modeling the position, color, and intensity of the sun. The method of Fei et al. [FCZ+24] used a stacked network architecture that decomposes the input face image into diffuse albedo, diffuse shading, surface normal, and specular reflection maps using two modified U-Net architectures. They calculated a spherical HDR environment by first estimating the brightness and position of light sources and then the texture of the light sources using two GANs.

CHAPTER 3

Dataset Generation

As mentioned in Section 1.2, we used a synthetic dataset consisting of pairs of face images along with corresponding ground truth lighting information to train a light source estimation system. In our case, the ground truth consisted of 360 ° HDR equirectangular environment images. The panoramas were obtained from multiple sources on the internet [Wro, iHD, Hav] and authors of other light source estimation systems [GSY+17, BGH+23, HGAL19, CLG+18]. Environment maps from a single source were exclusively assigned to either the training or test dataset partition.

Section 3.1 describes how we render realistic-looking face images using *Unreal Engine* in combination with the *MetaHuman* framework. The integration of the framework, which allows the rendering and animation of digital human characters, was the main reason we chose *Unreal Engine* to render our scenes. When using a different framework for human characters, choosing a different engine such as *Blender* [DT] could be more beneficial, as *Python* scripting in *Unreal Engine* is still an experimental feature and requires some tinkering to get right.

Figure 3.1 presents a few examples of the resulting dataset. As HDR images cannot directly be displayed on standard displays, all HDR files displayed in this thesis were tonemapped using the operation described by Drago et al. [DMAC03]. The left column shows the images rendered using our *Unreal Engine* project, while the right column shows the corresponding environment maps. Table 3.1 shows an overview of the total number of images generated and the number of different environments. We did not create any *MetaHuman* avatar ourselves, but used the 66 provided example avatars. Rendering a single image took around 10 seconds, resulting in a total runtime of over 17 days to create the dataset.

Some additional steps, which we describe in Section 3.2 are needed to use the dataset for training the brightness estimation network. The training and architecture of the networks are presented in Section 4.1. Finetuning a diffusion-based model requires additional input

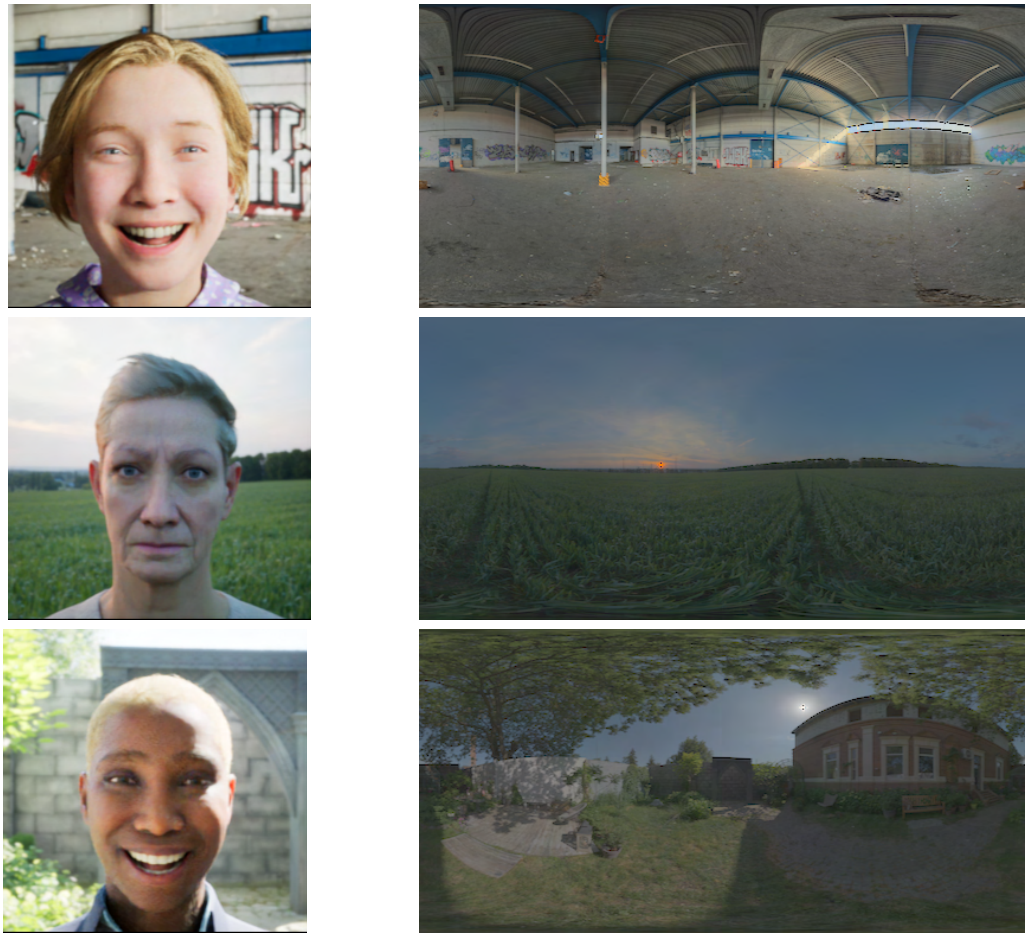


Figure 3.1: Comparison of different rendered face images with the corresponding rotated environment maps. The original environment maps were obtained from different websites [iHD, Hav].

in the form of captions describing the images. Section 3.3 describes how these captions were generated using a LLM. Training of the diffusion network is presented in Section 4.2.

3.1 Rendering Face Images in Unreal Engine

Using *Unreal Engine* 5.3.2 as our rendering engine, we created a template scene only consisting of a single Blueprint of a *MetaHuman* avatar, a *Cine Camera Actor*, and a *Sky Light*. Rendering was performed using the *Path Tracer* module in the *Movie Render Queue Plugin*. To make the environment map visible to the path tracer, the variable *r.PathTracing.VisibleLights* was set to *2.0*. Further information on using the path tracer in *Unreal Engine* can be found in the official documentation [Gamb].

	Training Set	Test Set
Number of *MetaHuman* Avatars	52	14
Number of Environment Maps	2828	810
Total Number of Images	147056	11340
Images Below Brightness Threshold	3802	-

Table 3.1: Overview of the number of images rendered for our dataset using *Unreal Engine*.

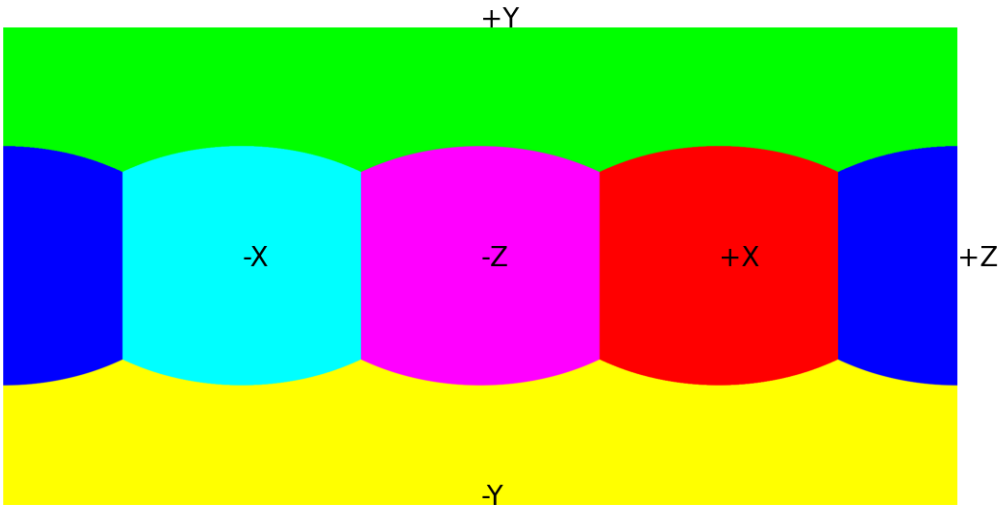


Figure 3.2: Visualization of the coordinate axis in an equirectangular environment map. Image obtained from the description of the Skylibs library [Hol24].

3.1.1 Scene Setup

After setting up the *Unreal Engine* Project and the template scene, we used a *Python* script to generate a separate scene for each *MetaHuman* character. The script replaced the avatar *Blueprint* in the template scene with the new avatar and adjusted the camera height to five cm below the head position. This was necessary due to the different height of the individual avatars. When loading the *MetaHuman* Blueprint, we forced the engine to use the highest level of detail. Other than the scene itself, the script also created a *Level Sequence* for each avatar. This *Level Sequence* is used to define the individual frames that will be rendered.

3.1.2 Automatic Render Utility

The *Python* script for automatically rendering the images first collects a list of all available environment maps and sorts them alphabetically. It creates a single *Movie Pipeline*

Parameter	Values
Facial Expression	1 out of 11
Focal Length	32 mm – 35 mm
Head Rotation	-5 ° – 5 ° in x, y, z
Body Position	-1 cm – 1 cm in x, y, z
Environment Azimuth Rotation	0 ° – 360 °

Table 3.2: Overview of the randomized parameters for each render pass.

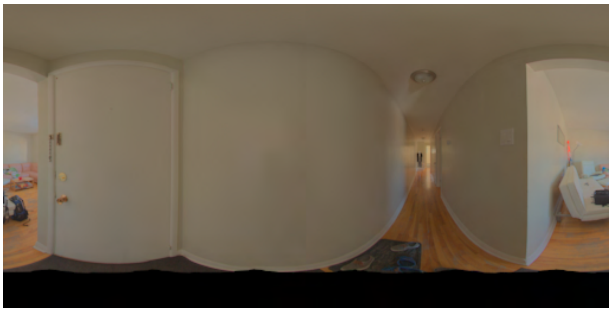
Job for each pair of *MetaHuman* avatar and environment map. After each render pass, multiple variables were randomized to make the dataset more diverse. Table 3.2 shows the different parameters and the allowed values. The randomized parameters were saved in a .csv file together with additional metadata. Images were rendered at 256x256 pixels with 256 samples per pixel.

Although we deleted all unused references and explicitly called the garbage collection, *Unreal Engine* would accumulate memory for each render pass. As it did not crash when the memory usage got too high, the rendering PC would eventually freeze. Therefore, we checked the available RAM and terminated the engine from within the *Python* script to avoid running out of memory. A second *Python* script was continuously monitoring if *Unreal Engine* was still running and restarting the engine otherwise. After restarting the engine, the script continued rendering from the last rendered image.

3.2 Preprocessing for Network Training

To use our data to train a neural network, some additional preprocessing steps were performed. At this point, our dataset consisted of over 150000 rendered face images, around 3000 environment maps, and a *.csv* file containing metadata about the rendered images. The actual ground truth for each face image is a rotated version of the original input environment map. To avoid performing the rotations at training time, we performed them as a preprocessing step and saved each rotated environment map as a separate .hdr file, resulting in a dataset with the same number of input and ground truth images. Figure 3.2 visualizes the coordinate axis of an equirectangular environment map. The light information in the environment map is distorted compared to the real world, due to projecting the information from a spherical representation onto a single rectangular image. We used the Skylibs library [Hol24] to perform the rotations.

The environment maps provided by the authors of previous research [GSY+17, BGH+23, HGAL19] needed an additional inpainting preprocessing step, as they would otherwise contain empty pixels at the bottom of the image. We used the method provided in their *Python* script [BGH+] to perform the inpainting. Figure 3.3 compares an environment before preprocessing to after preprocessing.



(a) Original environment map

(b) Rotated and inpainted environment map

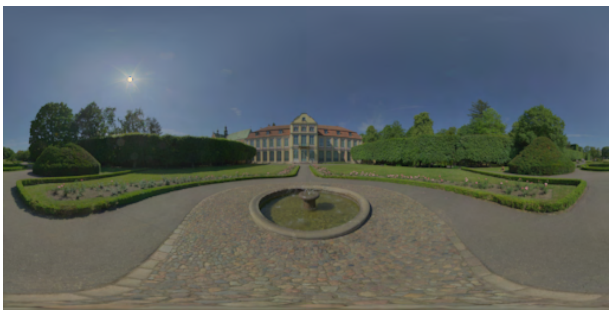
Figure 3.3: Comparison of an HDR environment map before and after the rotation operation. The rotated file was inpainted to remove the empty pixels at the bottom. Original environment map was obtained from a dataset consisting of indoor HDR images [BGH+23].

To test whether face-to-environment models would perform better if only the faces were visible, we created a separate dataset where the background was removed using a semantic segmentation model. Some face images were rendered very dark. We calculated the average pixel values of each face image after rendering and deleted every image whose average pixel value was below 10.0 to increase the quality of the dataset. This concludes the image preprocessing that was done before using the dataset for any kind of neural network training. Further preprocessing steps that are performed at training time, such as converting to the right datatype and additional data augmentation, are discussed in the next chapter.

3.3 Dataset for Diffusion Network Training

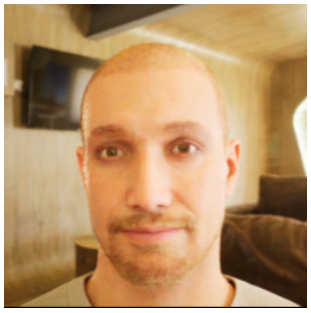
As mentioned in Section 2.1.4, standard diffusion networks typically take a prompt as input instead of an image. Because we want to use the ability of diffusion networks to create photorealistic images, we used the image-to-text generation functionality of *BLIP-2* [LLSH23], utilizing the *Flan T5-xxl* [CHL+22] LLM, to create descriptive captions for each tonemapped environment. This allows us to finetune a diffusion model on our domain

(a) "a 360 degree view of a living room with hardwood floors and a couch in the middle"

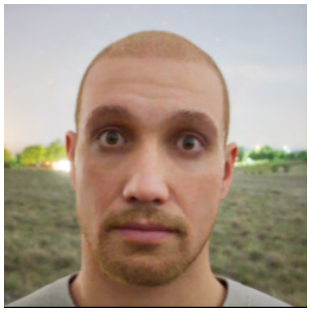


(b) "a 360 degree view of a large mansion with a fountain in front of it"

Figure 3.4: Environment maps with captions generated by BLIP-2. Images were obtained from a website [Wro].



(a) "a room with a couch and tv"



(b) "a grassy field with a starry sky"

Figure 3.5: Face images from our dataset with guided captions generated using a *BLIP-2* [LLSH23] model.

of realistic-looking 360 ° panorama images. Figure 3.4 shows two prompts generated from the tonemapped environments.

The same functionality was used to create captions from our rendered face images. Because we are interested in the environmental lighting and not the description of the person in the image, we had to give additional guidance to the *BLIP-2* model. This is done by including a question-answer style prompt when inferring the model. For our face images, we used the following prompt: *"Question: What do you see? Answer: A person standing somewhere. Question: Where does the person stand? Be as precise as you can and describe the color and light of the environment. Answer: In ".*

Figure 3.5 shows some example captions generated from face images. The captions describe the scene well, considering the small amount of the environment that is visible behind the person. Some captions are not accurate. An example of this is the "starry sky" caption in Subfigure (b), which is not present in the image.

CHAPTER 4

Light Source Estimation

The previous chapter introduced a solution for generating a synthetic face image dataset from HDR environment maps using *Unreal Engine* and the *MetaHuman* framework. In this chapter, we show how this dataset was used to develop our novel light source estimation system. The system is divided into two parts, a brightness estimation module, and a diffusion-based image generation module conditioned on the estimated brightness. All networks were implemented in *Python* using the *PyTorch* library [PGM+19].

To create our brightness estimation module, we follow the approach of a state-of-the-art system [SNWS20], training a simple CNN to estimate the brightness of a scene illumination from an input face image. Section 4.1 details the creation of the network, starting with an environment-to-environment autoencoder that learns a latent space by encoding and decoding panorama environment images. We then replaced the environment image encoder with a more advanced residual CNN, which compresses an input face

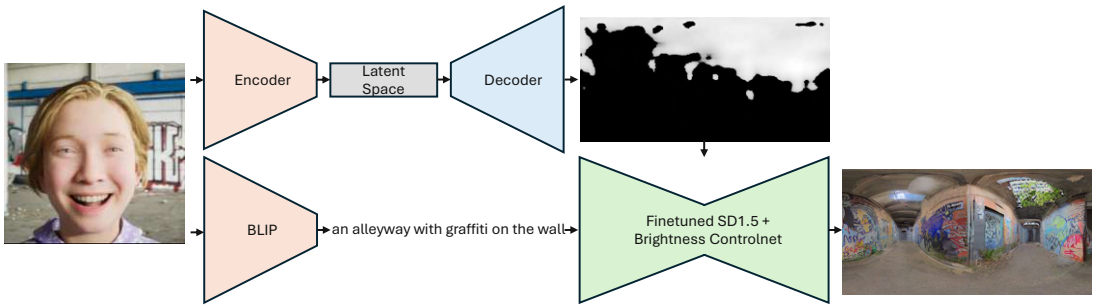


Figure 4.1: Overview of the proposed light source estimation system. A brightness image and a text prompt describing the scene is generated from a face image. Text description is input to a finetuned *SD* model. The estimated brightness is added as conditional input via a *ControlNet* model.

image into this latent space, resulting in the final CNN-based brightness estimation network.

Section 4.2 details the diffusion-based image generation system. We first describe the process of finetuning an existing *SD* [RBL+22] network in Section 4.2.1. Adding conditional input in the form of a *ControlNet* [ZRA23] yields our final light estimation system. The creation and evaluation of the conditioned diffusion model is presented in Section 4.2.2.

Figure 4.1 shows an example of a complete run through our proposed system. A brightness image is generated from the input face image using our face-to-environment network. A short text description that describes the scene is generated using *BLIP-2*. The text is fed into the finetuned diffusion network. The brightness image is added as a conditional input to a *ControlNet* pretrained on brightness images.

4.1 CNN-based Brightness Estimation

Our brightness estimation network is mostly based on previous work by Sztrajman et al. [SNWS20] and Weber et al. [WPL18]. Similarly to their approach, we first train an autoencoder network to encode an environment image into a small latent vector and then decode it back to the original image, ideally replicating the input. This is an easier task than directly calculating the environment map from a face image because the information does not need to be extrapolated from facial features. The goal of this approach is to generate a small latent space containing all the necessary information to recreate the original panorama image, allowing us to subsequently train a second model to encode a face image into the created latent space.

Parameter	Values
Activation Function	*LeakyReLU*
Batch Size	128
Bottleneck Size	64 | 512
Number of Epochs	1000
Loss Function	ℓ_{AE} | ℓ_{AElog}
Optimizer	*Prodigy*
Learning Rate	1
Weight Decay	0 | 0.1
Cosine Annealing	True | False

Table 4.1: Overview of the hyperparameters used for training the autoencoder.

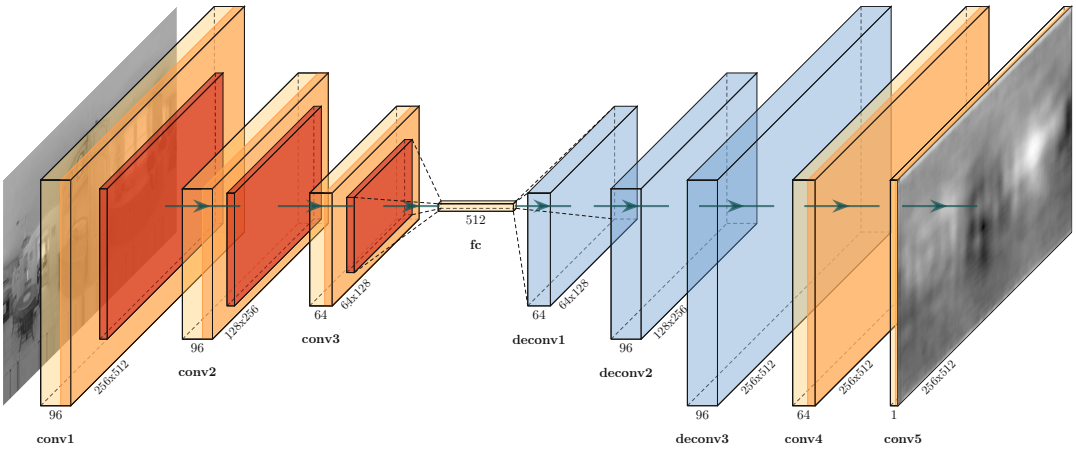


Figure 4.2: Overview of the environment-to-environment autoencoder network. The input is transformed into a latent vector of size 1x512 using convolutional blocks with maximum pooling operations. The deconvolutional layers, visualized in blue, upscale the latent vector, ideally replicating the original input. The input environment image is taken from the *Laval Photometric Indoor HDR Dataset* [BGH+23]

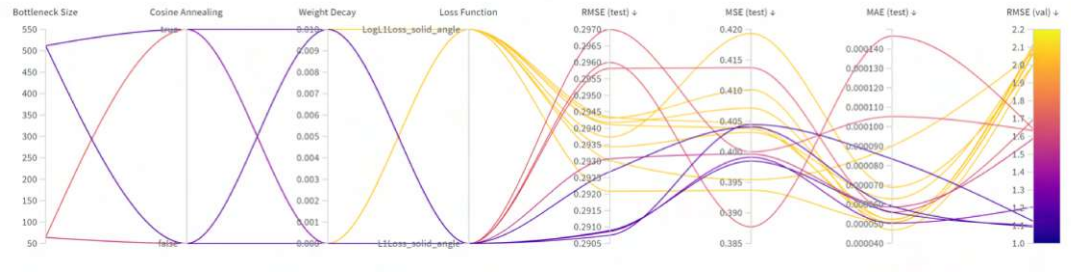


Figure 4.3: Results of the autoencoder hyperparameter optimization. The color gradient displays the performance on the validation accuracy. The choice of loss function had the biggest impact on validation performance.

4.1.1 Environment-to-environment Autoencoder

Architecture

Figure 4.2 illustrates the final architecture of our autoencoder network. The illustrations were made using the *PlotNeuralNet* library [Iqb18]. The same tool was used to create the other figures of our proposed network architectures in this thesis. The encoder module consists of three convolutional blocks, each containing a convolutional layer, a *LeakyReLU* activation function, and a batch normalization layer. After each block, a maximum pooling operation reduces the height and width by half. In the figure, the convolutional layers and the activation function are represented as light and dark orange

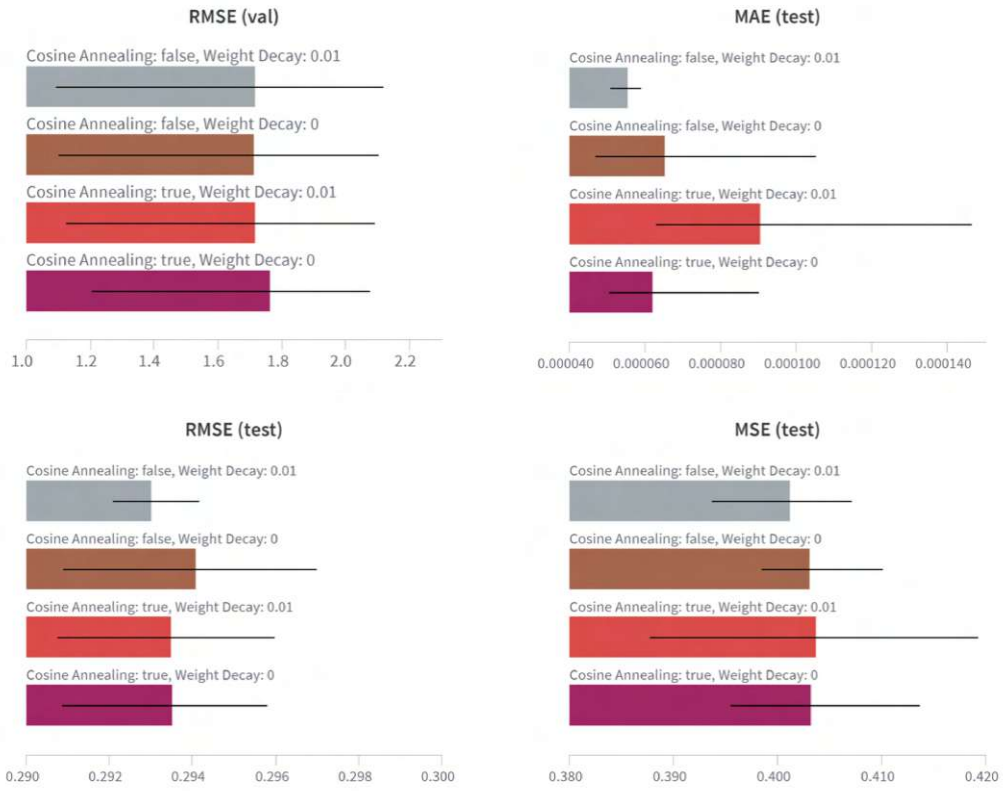


Figure 4.4: Effects of using weight decay and cosine annealing when training the autoencoder.

boxes, respectively, while the maximum pooling operation is visualized by a dark red box. The output of these convolutional blocks is compressed into a vector of size 1x512 using a linear layer without an activation function.

The decoder module consists of three convolutional upscaling blocks followed by two additional convolutional blocks. Before the first upscaling block, the bottleneck vector of size 1x512 is transformed into the proper input format for the following blocks using a linear layer. Each upscaling block contains a transposed convolutional layer to increase the resolution, a *LeakyReLU* activation function, and a batch normalization layer. A *ReLU* activation function is used after the final layer to ensure positive output values from our network.

Training

As we are interested in estimating brightness, the environment images are converted to greyscale in a preprocessing step. To train our autoencoder models, we used a grid search-based hyperparameter optimization method, which runs a training with every

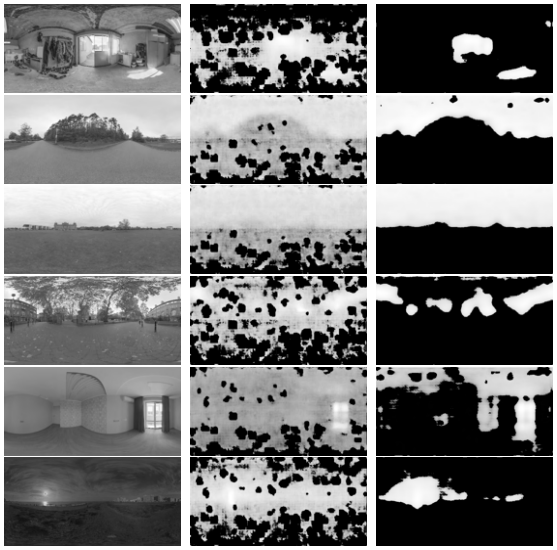
Parameter	Model A	Model B
Bottleneck Size	512	512
Loss Function	ℓ_{AE}	ℓ_{AElog}
Weight Decay	0.01	0
Cosine Annealing	True	False

Table 4.2: Overview of the hyperparameters for the autoencoder models that were later used to train the face-to-environment models.

	MAE↓	MSE↓	RMSE↓	MAE (log)↓	MSE (log)↓	RMSE (log)↓
Model A (test)	8.434×10^{-5}	**0.467**	**0.311**	3.206×10^{-5}	1.660×10^{-5}	0.004
Model B (test)	$\mathbf{4.851 \times 10^{-5}}$	0.473	0.314	$\mathbf{1.424 \times 10^{-5}}$	$\mathbf{5.249 \times 10^{-6}}$	**0.002**
Model A (val)	**0.002**	**3.719**	**1.129**	2.893×10^{-5}	2.047×10^{-5}	0.004
Model B (val)	0.003	12.932	2.006	$\mathbf{2.419 \times 10^{-5}}$	$\mathbf{1.860 \times 10^{-5}}$	**0.004**

Table 4.3: Comparison of different metrics for the test set and the validation set. The values indicate that our models perform better on the test set compared to the validation set. This is likely due to the uneven data distribution, leading to higher expected errors for the validation set, as the average brightness is higher by a factor of 100. The best results for each metric are highlighted in bold.

possible combination, as the number of hyperparameters we were optimizing was small. Table 4.1 presents the hyperparameters used to train our autoencoder network. Following Weber et al. [WPL18], we use a weighted L1 loss. Our hyperparameter optimization included experimenting with two different loss functions. A logarithmic compressed loss function ℓ_{AElog} as described in Equation 4.1 and a loss function ℓ_{AE} without a logarithmic operation as described in Equation 4.2. In the equations, e_i describes an environment image, while $f(e_i)$ represents the output of our network given an environment image. We add $+1$ to the values before calculating the logarithm to avoid numerical issues. Each loss function is weighted by the solid angle using pixel-wise multiplication, represented as $w\odot$. An image containing the solid angle subtended by each pixel was obtained from the *Skylibs* library [Hol24]. To avoid tuning the learning rate manually, we used the *Prodigy* optimizer [MD24], which automatically tunes the learning rate at each step. As detailed in Section 3.2, we created a rotated environment image for each character. In each training epoch, we randomly choose one rotated version per environment map, which reduces the size of each training epoch from 117624 to 2262 images and the size of each validation epoch from 29432 to 566 images. This random selection probably introduced some variance in the different performance metrics. The average training time for the models with 1000 epochs each was around 7.75 hours. Training was performed with two *NVIDIA A40* graphics cards.



(a) GT (b) Model A (c) Model B

Figure 4.5: Results on six different panoramas from our test dataset using our trained environment-to-environment autoencoder models. The ground truth environment images were obtained from multiple online sources [Hav, iHD].

$$\ell_{AElog} = \sum_{i=1}^{N} \| w \odot ((log_{10}(e_i + 1) - log_{10}(f(e_i) + 1) \| \tag{4.1}$$

$$\ell_{AE} = \sum_{i=1}^{N} \| w \odot (e_i - f(e_i)) \| \tag{4.2}$$

Evaluation

The different hyperparameters resulted in 16 different models. For each training run, the best-performing model, based on the validation accuracy, was chosen for quantitative evaluation. We computed the root mean squared error (RMSE), the mean squared error (MSE), and the mean average error (MAE), each weighted by the solid angle, on the validation and the test set. Figure 4.3 shows the results of these metrics based on the hyperparameter selection, with arrows indicating the direction of performance increase. As visible by the color gradient, every model trained on the log-scaled loss function (Equation 4.1) performed worse on the validation set compared to models trained using the other loss function (Equation 4.2). On the test set, many models trained on the log-scaled loss function (Equation 4.1) outperformed other models. Models trained with the larger bottleneck size of 512 consistently performed better than models with a smaller bottleneck size of 64. Figure 4.4 illustrates the effects of varying values for cosine annealing and weight decay, showing that models trained with a weight decay of 0.01 and without

cosine annealing generally performed the best on average for all the metrics on the test set. Certain combinations of these values yielded better individual results. Two models were chosen as baselines to train the face-to-environment models. Table 4.2 lists the hyperparameters of the two selected models. Further evaluation was performed for these two models, with fixed test and validation epochs, without rotation. Table 4.3 presents the average metrics on the test and validation set for both models. Better performance is indicated by bold letters. Model A performed better on the MSE and the MAE, while Model B performs better for all log-scaled metrics on the test set. Comparing the metrics for the validation and the test set indicate that our models perform better on the test set compared to the validation set. This is a result of the unequal data distribution, with an average brightness of 56.56 for images in the validation set, compared to the average brightness of 0.66 for images in the test set, leading to higher expected errors in the validation set. The visual quality of output for panoramas from the validation set is superior to the quality of output for panoramas from the test set, likely due to overfitting, as the panorama images for the validation and the train set are obtained from the same sources, whereas test set images originate from a different set of sources. Figure 4.5 shows multiple results for the test set. While the most dominant light direction is usually detected, the resulting image does not resemble a panorama image. An example of the performance of the trained models on the validation set is presented in Figure 4.6. Figure 4.7 presents a further example from the test set. These figures also show results when using estimated panoramas as environment light sources in a rendering engine. The images were generated using *Blender 3.2.0* [DT] with modified *Python* scripts from the *StyleLight* repository [WYLL22]. Each rendered image contains an object with glossy, diffuse, or mirror material and a diffuse plane acting as a shadow catcher. The results on the validation set are barely distinguishable from the ground truth, while the test set results show noticeable difference in objects with reflective materials, though the overall light direction was estimated well, as is visible by the shadows. The inference time for the environment-to-environment models is approximately 0.056 seconds for a single image, and around 0.039 seconds per image when using a batch size of 16. The measurements were taken on a Windows PC with an *NVIDIA RTX 3060Ti* graphics card.

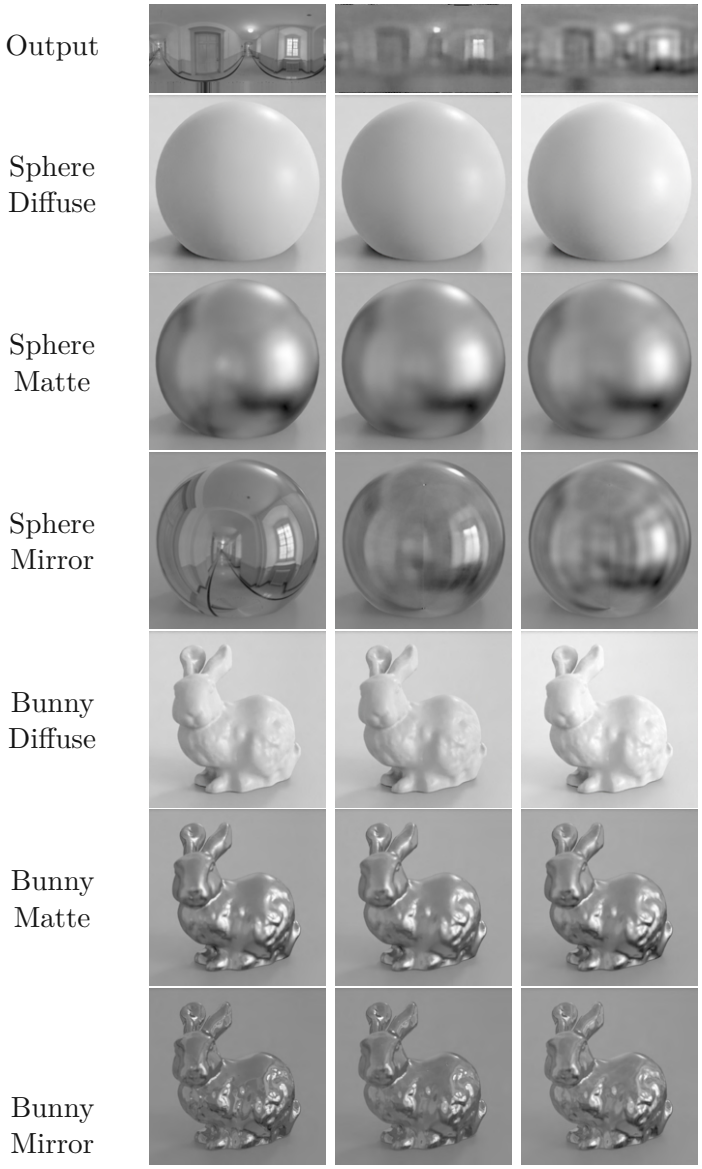


Figure 4.6: Multiple objects rendered using the output of our trained environment-to-environment autoencoder models as environment light sources. The input to the brightness estimation network was taken from the validation set. The ground truth environment image is part of the *Laval Photometric Indoor HDR Dataset* [BGH+23].

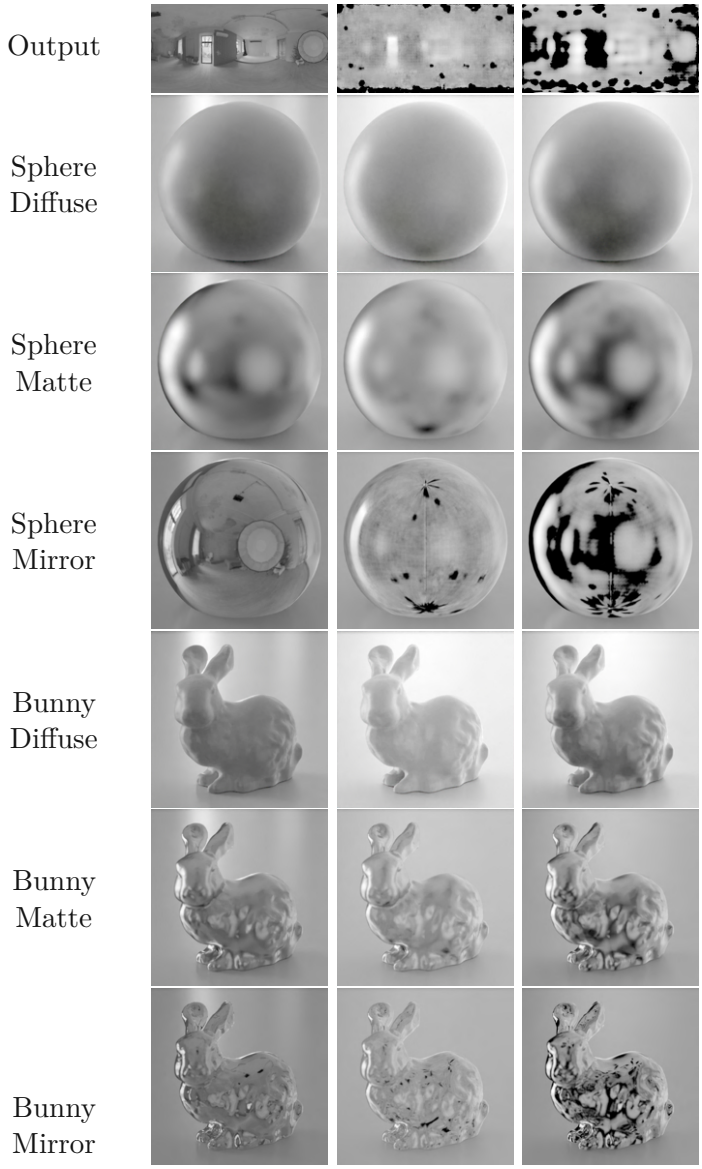


Figure 4.7: Multiple objects rendered using the output of our trained environment-to-environment autoencoder models as environment light sources. The input to the brightness estimation network was taken from the test set. The ground truth environment image was obtained from an online source [Hav].

4.1.2 Face-to-environment Network

In this subsection, we present the results of replacing the environment encoder presented in the previous subsection with a new encoder. Instead of compressing environment images to a latent vector, the new encoder compresses face images, yielding a network that can estimate the brightness of the surrounding illumination from face images.

Architecture

The architecture of the face-to-environment network consists of an encoder module and a decoder module, similar to our autoencoder network described in Section 4.1.1. Figure 4.8 illustrates the final architecture for this network. The encoder module consists of a single convolutional block followed by four residual blocks. The convolutional block contains a convolutional layer, a *LeakyReLU* activation function, and a batch normalization layer. Figure 4.9 shows the slightly more sophisticated residual blocks, which contain a maximum pooling layer, two convolutional layers, and two activation functions. Batch normalization is used after each of the two convolutional layers. A skip connection is placed before the final activation function. To allow concatenation of arrays with different numbers of channels, a 1x1 convolution layer is added to the skip connection. The decoder follows the same layout as the one used in our environment-to-environment autoencoder.

Training

We trained our face-to-environment models similarly to the environment-to-environment models using a hyperparameter optimization, generating 20 distinct models. Table 4.4

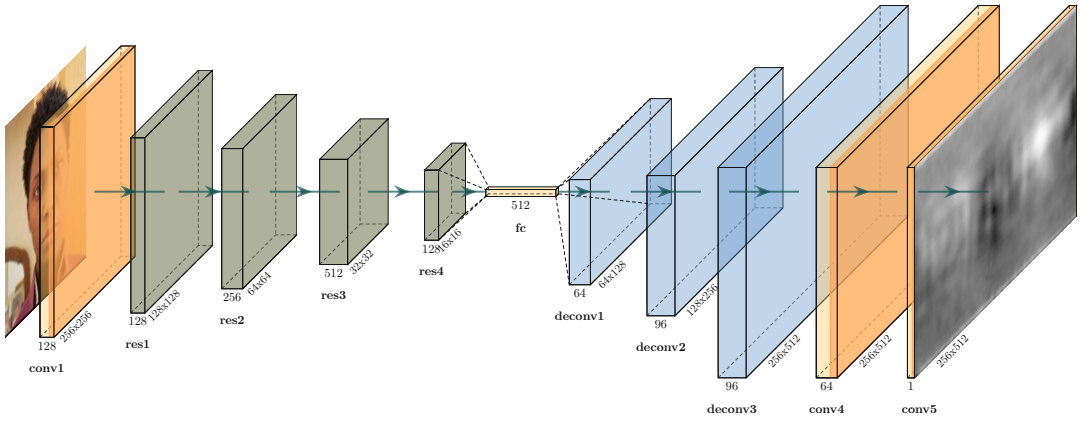


Figure 4.8: Overview of the full CNN-based brightness estimation network. An input face image is transformed into a latent vector of size 1x512 using residual blocks. The encoder decompresses this latent vector into a 256x512x1 image, representing the estimated brightness of the scene.

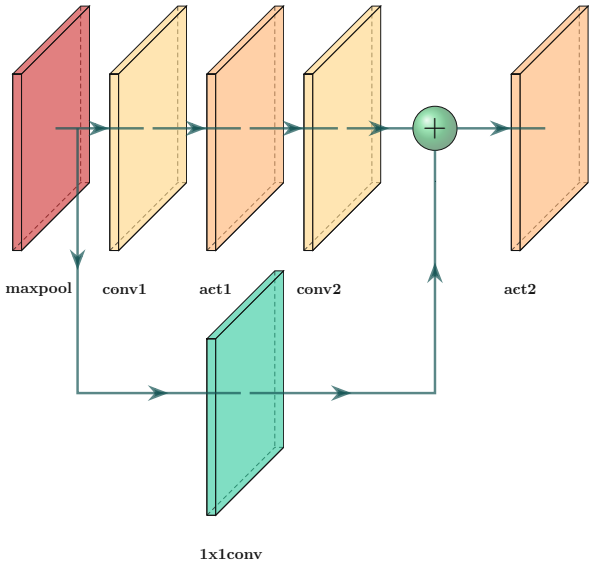


Figure 4.9: Visualization of the residual block used in our face encoder network. Input is passed through a maximum pooling layer to reduce the resolution, followed by two convolutional layers and an activation function between them. Batch normalization is used after each convolutional layer. A shortcut using a 1x1 convolution is added before the final activation function.

lists the hyperparameters used to train our face-to-environment models. We implemented three different training methods:

1. **Both** The decoder was initialized with weights from a pretrained autoencoder, and then the entire network was trained.

2. **Encoder** The decoder was initialized with pretrained weights, but the weights were not updated during training.

3. **Latent** The network was truncated after the encoder module, so the output was a latent vector instead of an image.

For the *Latent* training method, we used an L2 loss function similar to Weber et al. [WPL18], shown in Equation 4.3. Here, $f_{enc}(e_i)$ represents the latent vector generated by encoding environment e_i with an already trained encoder module from an autoencoder, while $f(I_i)$ describes the estimated latent vector from the face-to-environment network encoder module. For the other two methods, we used solid-angle weighted L1 losses, as described in Equation 4.4 and Equation 4.5, where $f(I_i)$ describes the output brightness image generated from an input image. To reduce the size of each training and validation epoch, we randomly selected one rotated version per environment map as the ground

Parameter	Values
Training Method	Both | Encoder | Latent
Remove Background	True | False
Activation Function	*LeakyReLU*
Batch Size	64
Bottleneck Size	512
Number of Epochs	1000
Loss Function	ℓ_{F2Elat}[1] | ℓ_{F2E}[2] | ℓ_{F2Elog}[2]
Decoder Weights	Model A | Model B
Optimizer	*Prodigy*
Learning Rate	1
Weight Decay	0.1
Cosine Annealing	True
Initial Feature Maps	128

Table 4.4: Overview of the hyperparameters used for training the face-to-environment network.

[1] Only for the *Latent* training method.
[2] Only for *Both* and *Encoder* training methods.

truth, in combination with the corresponding face image. Consequently, input images from a specific *MetaHuman* avatar were used in both the training and the validation set, but the environment maps were exclusively assigned to either the training or the validation split. The training time for the models with 1000 epochs each ranged from 7.3 to 10.8 hours. Training was performed with two *NVIDIA A40* graphics cards.

$$\ell_{F2Elat} = \sum_{i=1}^{N} \|(f_{enc}(e_i) - f(I_i))\|_2 \tag{4.3}$$

$$\ell_{F2E} = \sum_{i=1}^{N} \|w \odot (e_i - f(I_i))\| \tag{4.4}$$

$$\ell_{F2Elog} = \sum_{i=1}^{N} \|w \odot ((log_{10}(e_i + 1) - log_{10}(f(I_i) + 1)\| \tag{4.5}$$

Evaluation

As with the autoencoder training, we selected the best-performing model from each training run and calculated the RMSE, MSE, and MAE, each weighted by the solid angle,

Parameter	Model C	Model D	Model E	Model F
Training Method	Latent	Latent	Encoder	Both
Decoder Weights	Model A	Model B	Model A	Model B
Loss Function	ℓ_{F2Elat}	ℓ_{F2Elat}	ℓ_{F2E}	ℓ_{F2Elog}
Remove Background	False	False	False	False

Table 4.5: Overview of the hyperparameters for the best performing face-to-environment models. Hyperparameters not described were chosen as in Table 4.4.

on both the validation and the test set. To allow for comparison with models trained with the *Latent* training method, we computed all metrics on the decoded latent vectors. Figure 4.10 presents the impact of using images with and without background removal. Models trained without removing the background consistently performed better. Table 4.5 lists the hyperparameters that resulted in well-performing models, that were then chosen for further evaluation. Table 4.6 gives an overview of the performance on the test set for these models. Although the data suggest that Models E and F outperformed Models C and D, comparing the metrics with the visual results of Figure 4.11 indicates that the results of Models C and D could actually be better. This discrepancy is likely due to the low average brightness of the test data, where simply outputting very low pixel values results in good performance on the calculated metrics. Figures 4.12, and 4.13 present results of using the output values of our face-to-environment models environment light sources in *Blender*. Models C and D effectively captured the most dominant light directions in both examples. Model E was able to capture the most dominant light direction for the input shown in Figure 4.12, while Model F failed to do so in both examples. Table 4.7 shows additional metrics calculated on the rendered images. These values suggest that Models E and F performed better with HDR output, but due to the tonemapping, Models C and D yielded better results. Figure 4.14 shows the performance of the models on a real face input from an outdoor scene. The best visual result was again obtained from Model C, accurately estimating the most dominant light direction. The other models were unable to generate a meaningful output. The inference time of the environment-to-environment models is approximately 0.068 seconds for a single image, and around 0.042 seconds per image when using a batch size of 16. The measurements were taken on a Windows PC with an *NVIDIA RTX 3060Ti* graphics card.

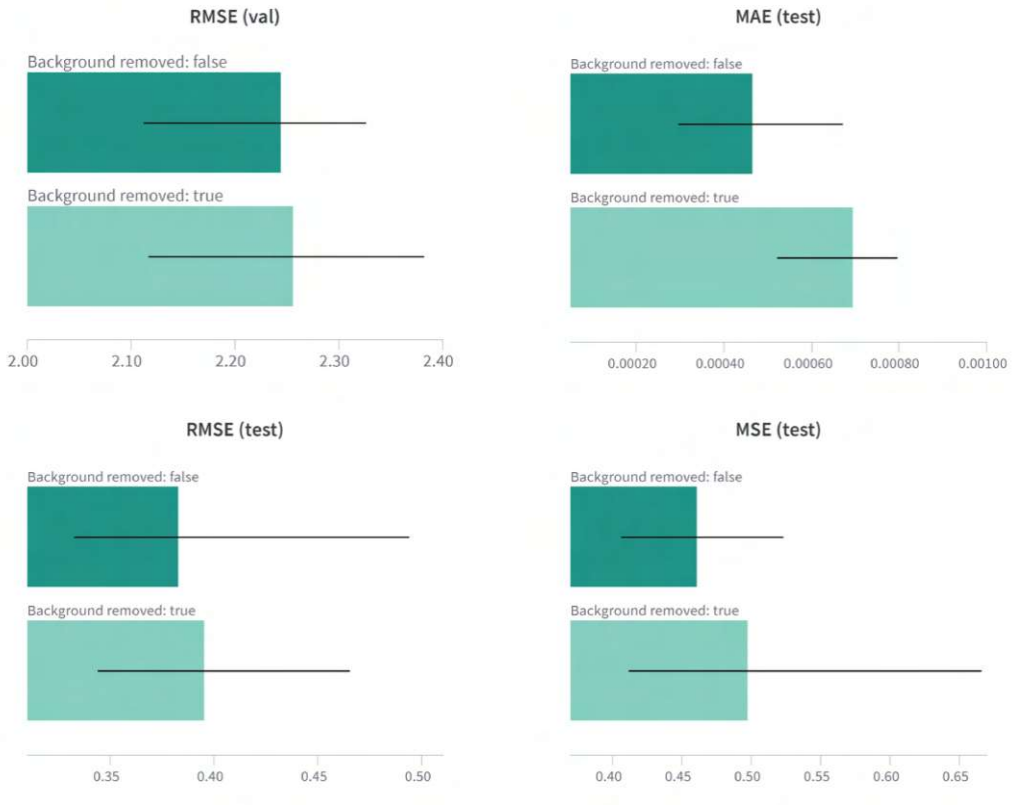
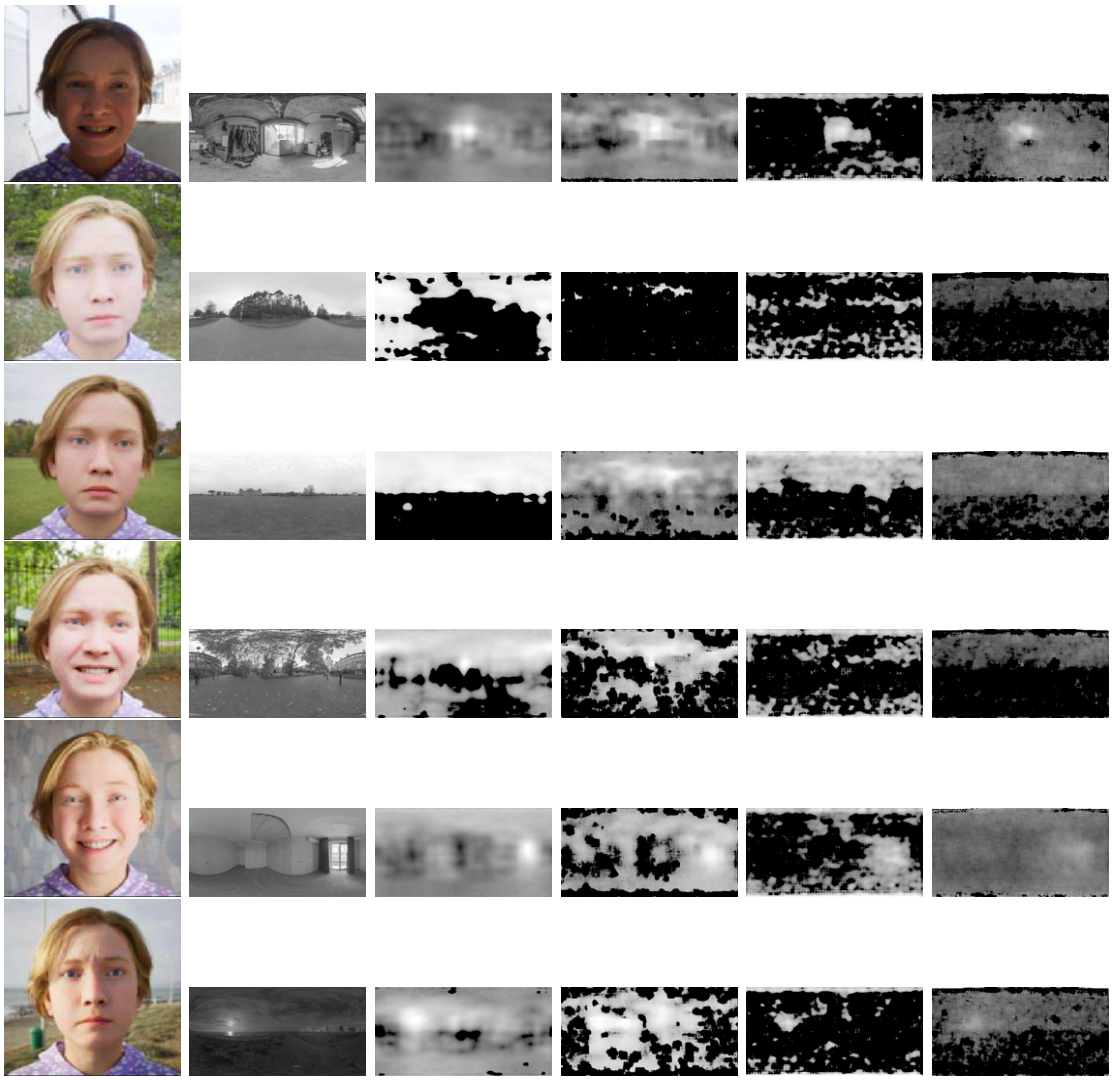


Figure 4.10: Effects of using images with and without removed background for training the face-to-environment network. The models trained without removing the background consistently performed better.

	MAE↓	MSE↓	RMSE↓	MAE (log)↓	MSE (log)↓	RMSE (log)↓
Model C	0.000 25	0.475	0.331	7.190×10^{-5}	9.233×10^{-5}	0.008
Model D	0.000 22	0.486	0.350	5.494×10^{-5}	6.810×10^{-5}	0.007
Model E	0.000 12	0.597	0.516	$\mathbf{3.207 \times 10^{-5}}$	$\mathbf{2.492 \times 10^{-5}}$	**0.004**
Model F	**0.000 11**	**0.475**	**0.329**	3.372×10^{-5}	3.109×10^{-5}	0.005

Table 4.6: Performance of different models on the test set. The data suggests that Models E and F perform superior to Models C and D. Comparing the metrics to the visual results in Figures 4.11, 4.12, and 4.13 could suggest that the results of Models C and D are actually better, likely due to the low average brightness of the test data, where outputting very low pixel values results in a good performance on the calculated metrics. The best results for each metric are highlighted in bold.



(a) Input Face (b) GT (c) Model C (d) Model D (e) Model E (f) Model F

Figure 4.11: Results on six different panoramas from our test dataset using our trained face-to-environment models. The ground truth environment images were obtained from multiple online sources [Hav, iHD].

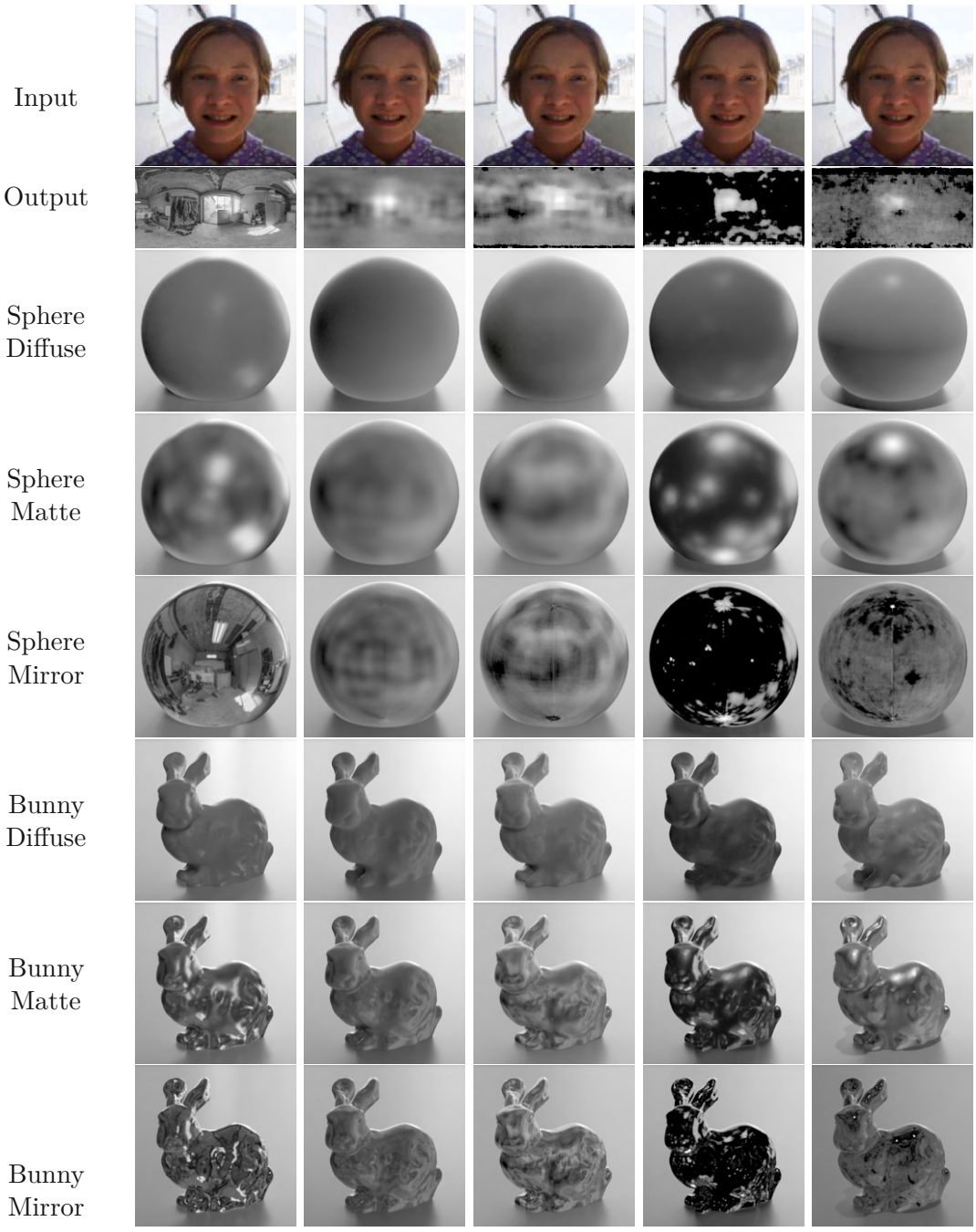


Figure 4.12: Multiple objects rendered using the output of our trained face-to-environment models as environment light sources. The input to the brightness estimation network is a rendered face from a test set panorama. The ground truth environment image was obtained from an online source [iHD].

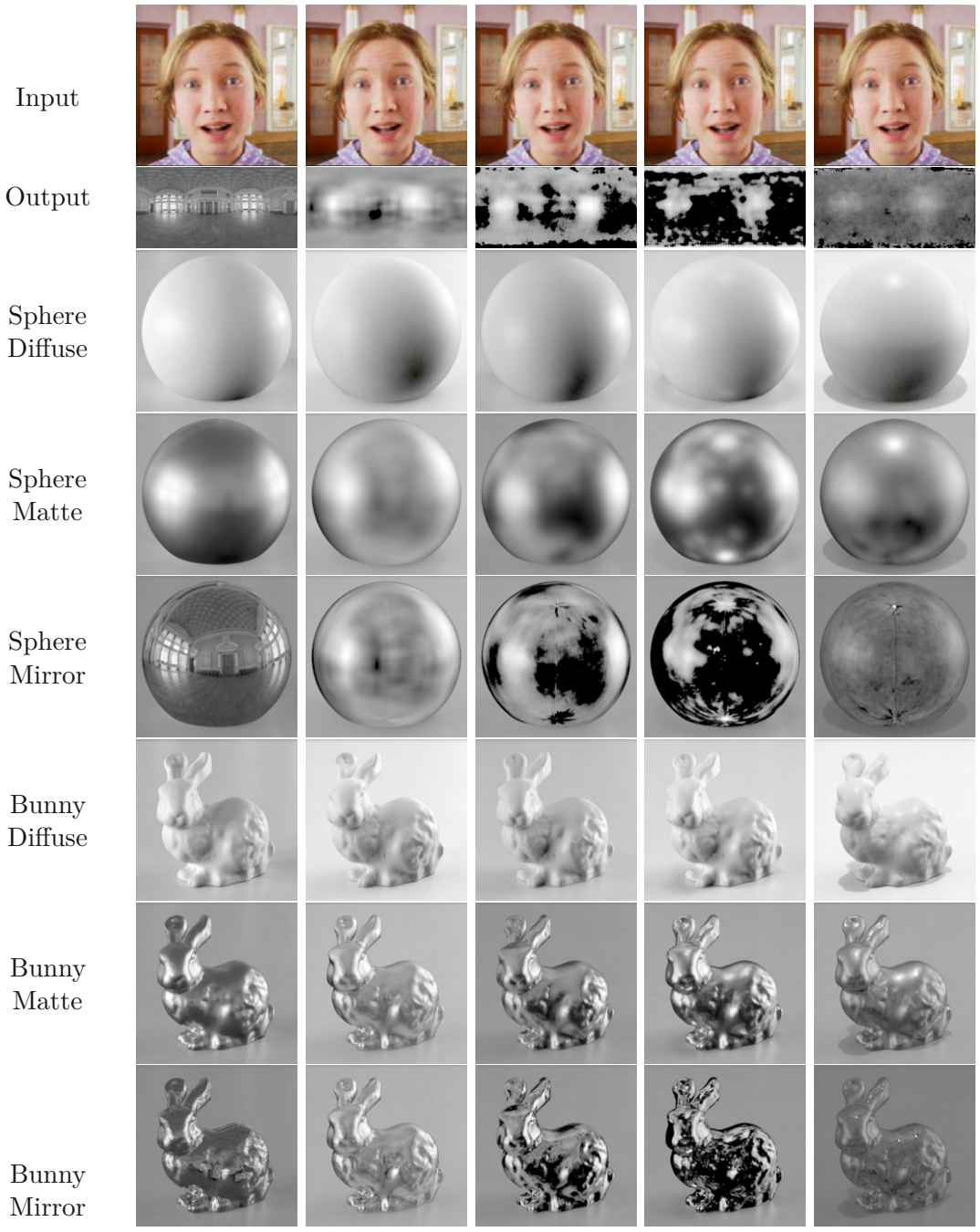


Figure 4.13: Multiple objects rendered using the output of our trained face-to-environment models as environment light sources. The input to the brightness estimation network is a rendered face from an indoor panorama from our test set. The ground truth environment image was obtained from an online source [Hav].

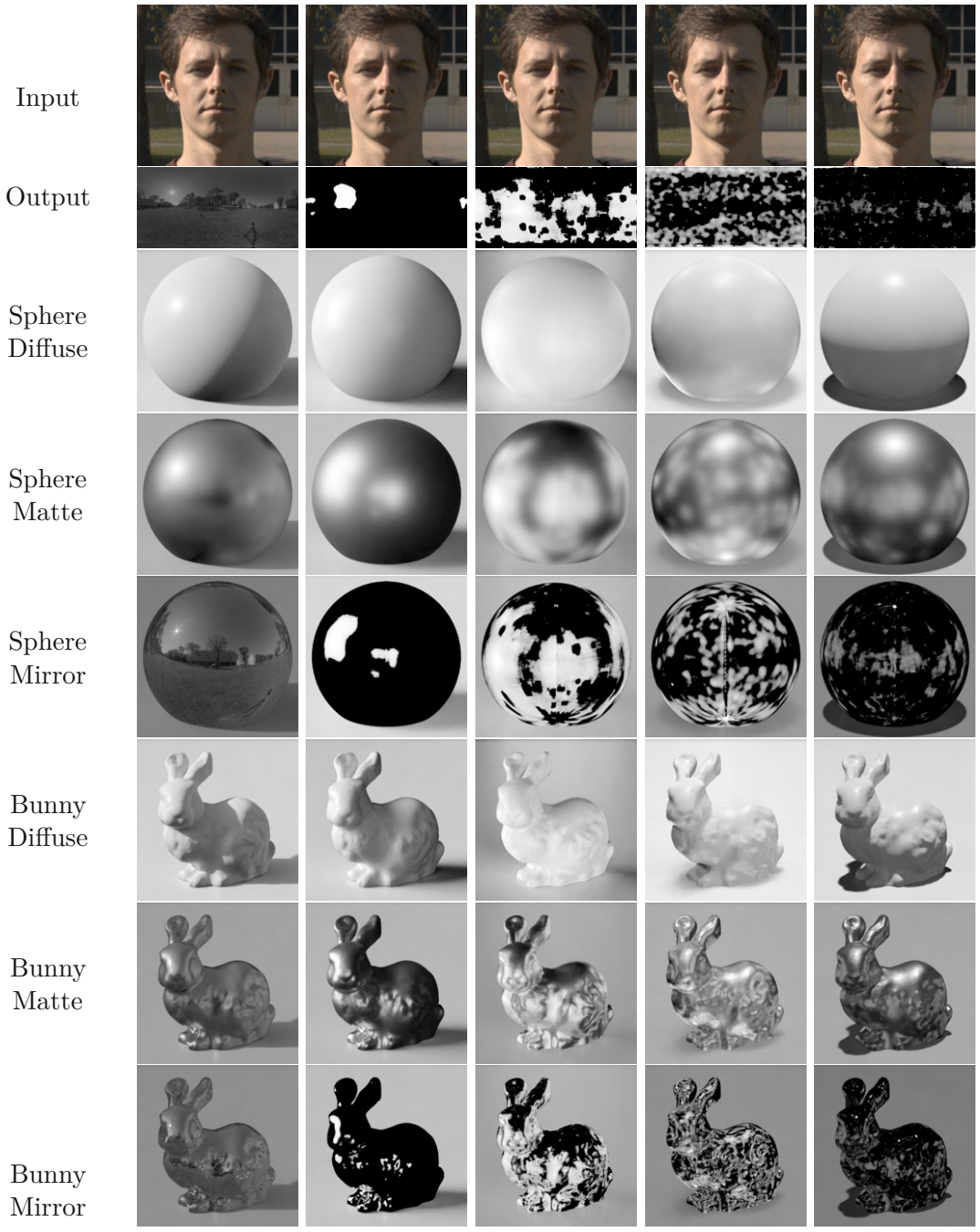


Figure 4.14: Multiple objects rendered using the output of our trained face-to-environment models as environment light sources. The input to the brightness estimation network is a real face image from an outdoor scene. The input face image and ground truth environment are part of the *Laval Face + Lighting HDR Dataset* [CLG+18]

	RMSE↓	PSNR↑	SSIM↑	RMSE(tm)↓	PSNR(tm)↑	SSIM(tm)↑
Model C	0.003 19	54.445 71	0.924 08	0.007 37	43.291 05	**0.969 90**
Model D	0.001 44	59.543 49	0.977 01	**0.006 69**	**44.134 66**	0.960 39
Model E	**0.000 36**	**70.214 99**	**0.999 08**	0.008 96	41.863 22	0.893 12
Model F	0.000 46	68.085 78	0.998 76	0.007 47	42.970 66	0.969 78

Table 4.7: Average metrics calculated on images rendered using our estimated brightness panoramas as environment light sources. For each model, the metrics were averaged over the 12 rendered images shown in Figures 4.12 and 4.13. The metrics were calculated on both the rendered HDR images and the tonemapped (tm) images that are shown in the figures. The best results for each metric are highlighted in bold.

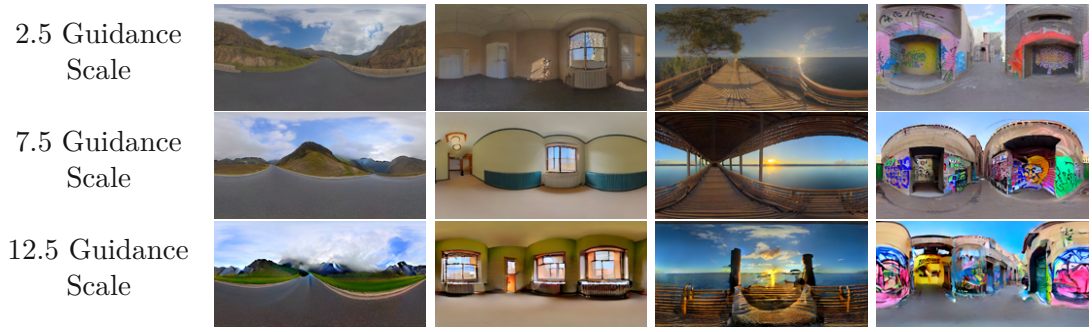
4.2 Brightness-Conditioned Stable Diffusion for Panorama Image Generation

This section details the diffusion-based image generation system. We first describe the process of finetuning an existing *SD* [RBL+22] network in Section 4.2.1. Section 4.2.2 describes adding conditional input in the form of a *ControlNet* [ZRA23], yielding our final light estimation system.

4.2.1 Finetuning Generative Diffusion Models for Panorama Images

Our diffusion model is based on the *SD 1.5* [RBL+22] checkpoint. We finetuned this model using a modified version of the *advanced stable diffusion training script* [PT] from the *Diffusers* library [vPPL+22]. This script employs *LoRA* to efficiently finetune the model with a small number of necessary weight updates, in combination with the *Dreambooth* [RLJ+23] training method to train the model on a specific subject or style. Advanced usage of the script is detailed in a blog post by Tsaban and Passos [PT]. We did not use pivotal tuning, but employed full text encoder training. Our training set consists of 682 pairs of environment images and captions, as described in Section 3.3. Table 4.8 lists the relevant training hyperparameters. We use the word *TOK* as the special token for the *Dreambooth* [RLJ+23] training method. Finetuning the model took 1.62 hours using an NVIDIA A40 graphics card.

After training the model, we experimented with different model inference parameters. Figure 4.15 shows the comparison of different guidance scales. According to the *Diffusers* [vPPL+22] documentation, increasing the guidance scale should direct the model to follow the text prompt more closely, while decreasing overall image quality. Our results show that the models already follow the text prompt well for a low guidance scale of 2.5, while increasing the guidance scale to 12.5 significantly reduced the quality of the resulting images. We continued to only using a guidance scale of 2.5 for generating images. Figure 4.16 displays the effects of varying the number of inference steps. Increasing



(a) a TOK 360 degree view of a road with mountains in the background and clouds in the sky

(b) a TOK 360 degree view of a room with a window and radiator in the corner of the room

(c) a TOK 360 view of a wooden bridge with the sun in the sky and water in the background

(d) a TOK alleyway with graffiti on the wall

Figure 4.15: Comparison of four text prompts with different prompt guidance scales. The finetuned panorama diffusion model was used to generate the outputs.

(a) a TOK 360 degree view of a road with mountains in the background and clouds in the sky

(b) a TOK 360 degree view of a room with a window and radiator in the corner of the room

(c) a TOK 360 view of a wooden bridge with the sun in the sky and water in the background

(d) a TOK alleyway with graffiti on the wall

Figure 4.16: Comparison of four text prompts with different number of inference steps. The finetuned panorama diffusion model was used to generate the outputs.

the number of inference steps did not increase the quality of the generated images, but slightly increased the inference time. Figure 4.17 demonstrates the effects of rotating the generated panorama images by 180 °. The border artefact is visible in the middle of every rotated image. Furthermore, Subfigure 4.17b shows signs of overfitting, with dark spots resembling a camera tripod, a feature present in several images in the training data, particularly in images of indoor scenes.

(a) a TOK room with a window and a view of a wooded area

(b) a TOK 360 degree view of a classroom with desks and a chalkboard on the wall

(c) a TOK 360 degree view of a room with a lot of tools and equipment in it

(d) a TOK alleyway with graffiti on the wall

(e) a TOK 360 view of a wooden bridge with the sun in the sky and water in the background

Figure 4.17: Effects of rotating the generated panorama images from the finetuned diffusion network by 180 °. The border artefacts are visible in every image. The captions in Columns (a) and (c) were generated using an image-to-text model on the ground truth environment images, while the captions in Columns (b) and (d) were generated using the image-to-text model on the face image.

Parameter	Values
Token Abstraction	TOK
Batch Size	4
Number of Epochs	50
Number of Images	682
Train Text Encoder	True
Learning Rate	1
Text Encoder Learning Rate	1
Learning Rate Scheduler	Constant
Optimizer	*Prodigy*
Adam Beta 1	0.9
Adam Beta 2	0.99
Adam Weight Decay	0.01
LoRA Matrix Rank	32
SNR Gamma	5.0
Precision	bf16

Table 4.8: Hyperparameters for finetuning a *SD* model for panorama image generation.

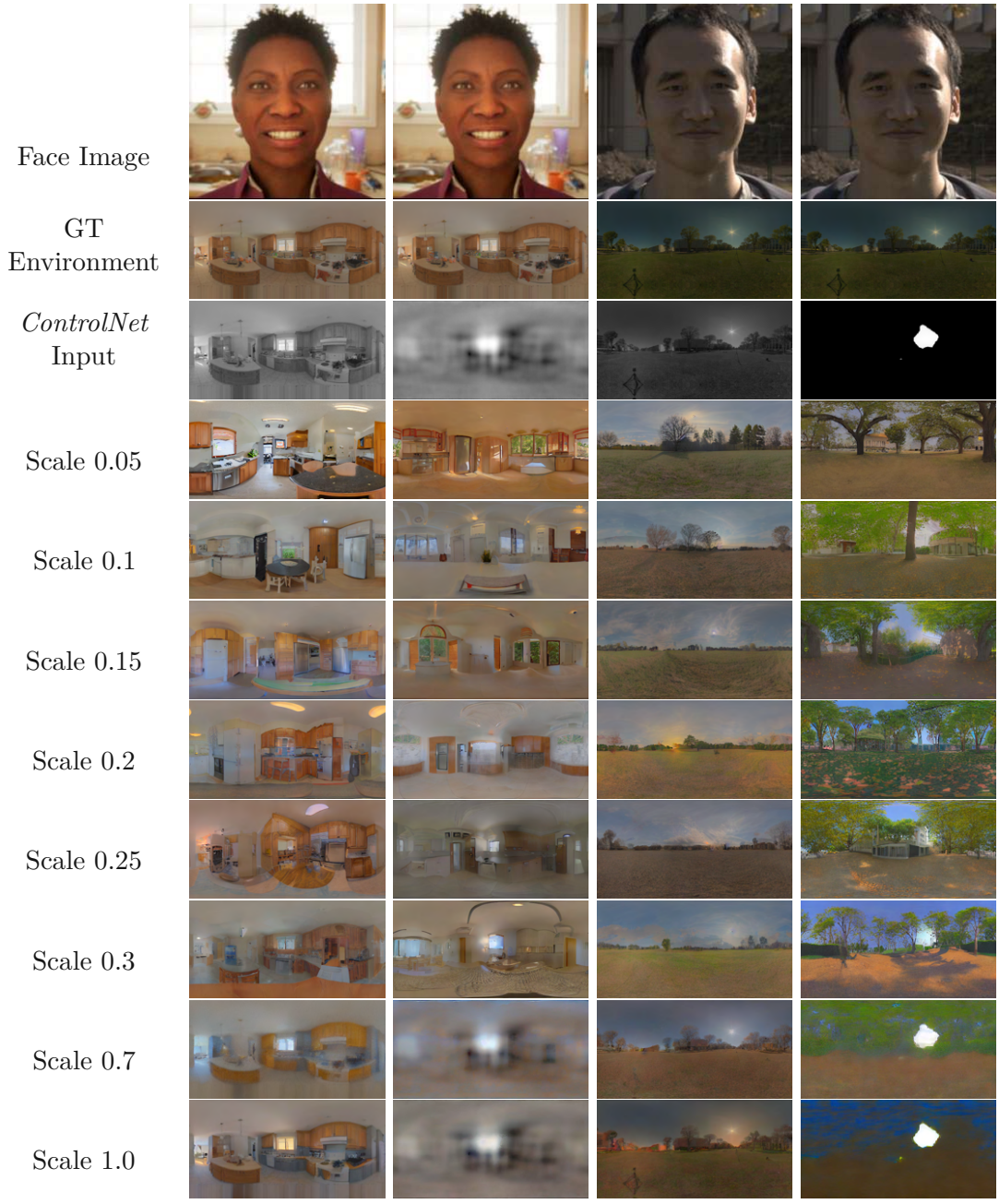
4.2.2 Adding Conditional Control to Diffusion-based Panorama Generation

The finetuned diffusion network presented in the previous subsection demonstrates the capability to produce realistic looking results. However, the layout of the generated scenes did not match the layout of the original scenes and visible artefacts appear when rotating the images. We hypothesized that adding additional input in the form of estimated brightness images could reduce these issues. To address this, we experimented with using a pretrained *ControlNet* [ZRA23], trained to colorize existing greyscale images, as an additional input to our diffusion model. Loading the *ControlNet* for use with our finetuned diffusion model is straightforward with the *Diffusers* library, requiring only a single additional line of code. Source Code 4.1 shows this integration and illustrates the usage of our complete system, generating a conditioned panorama image from a single face image.

To evaluate the model performance, we experimented with different model inference parameters. To see the influence of the quality of the brightness images, we experimented with using ground truth greyscale images and estimated brightness images from the test and validation set, with different estimation quality. When using a *ControlNet* to condition a diffusion network with an input brightness image, a scale parameter influences the strength of the condition. Figure 4.18 illustrates how changing this parameter affects the resulting images. As expected, the degree to which the model output follows the conditional input is proportional to the scale. Columns (b) and (d) show the output when generating *BLIP-2* [LLSH23] labels and brightness estimation from the face images, as opposed to using the ground truth greyscale environment panoramas in (a) and (c), respectively. Column (b) uses a face image from the validation set, while Column (d) uses a real face image. Scales greater than 0.3 tend to generate very visible artefacts, especially if the quality of the estimation is not good. Artefacts from the inpainting preprocessing step of the panorama in (a) and the visible camera tripod in the ground truth scene of (c) are also visible in the generated images. The results in Column (b) suggest that even when the estimated brightness appears to be a realistic greyscale environment image, increasing the conditioning scale can lead to unrealistic-looking results. This is most likely due to the blurred appearance of the estimated brightness images, as the *ControlNet* was trained to recolorize greyscale images without any blurring. Overall, while the model generally places the main light sources correctly in most images, it fails to do so in some examples.

	15 Inference Steps	25 Inference Steps	40 Inference Steps
Finetuned *SD*	1.23 seconds	1.87 seconds	2.91 seconds
Finetuned *SD* + ControlNet	1.61 seconds	2.46 seconds	3.65 seconds

Table 4.9: Average speed for generating a single image using different number of inference steps. Adding the *ControlNet* increases the time by around 30 %.



(a) a TOK kitchen with wooden cabinets, a stove, and a refrigerator in the center of the room. there is also a table and chairs on the side of the room

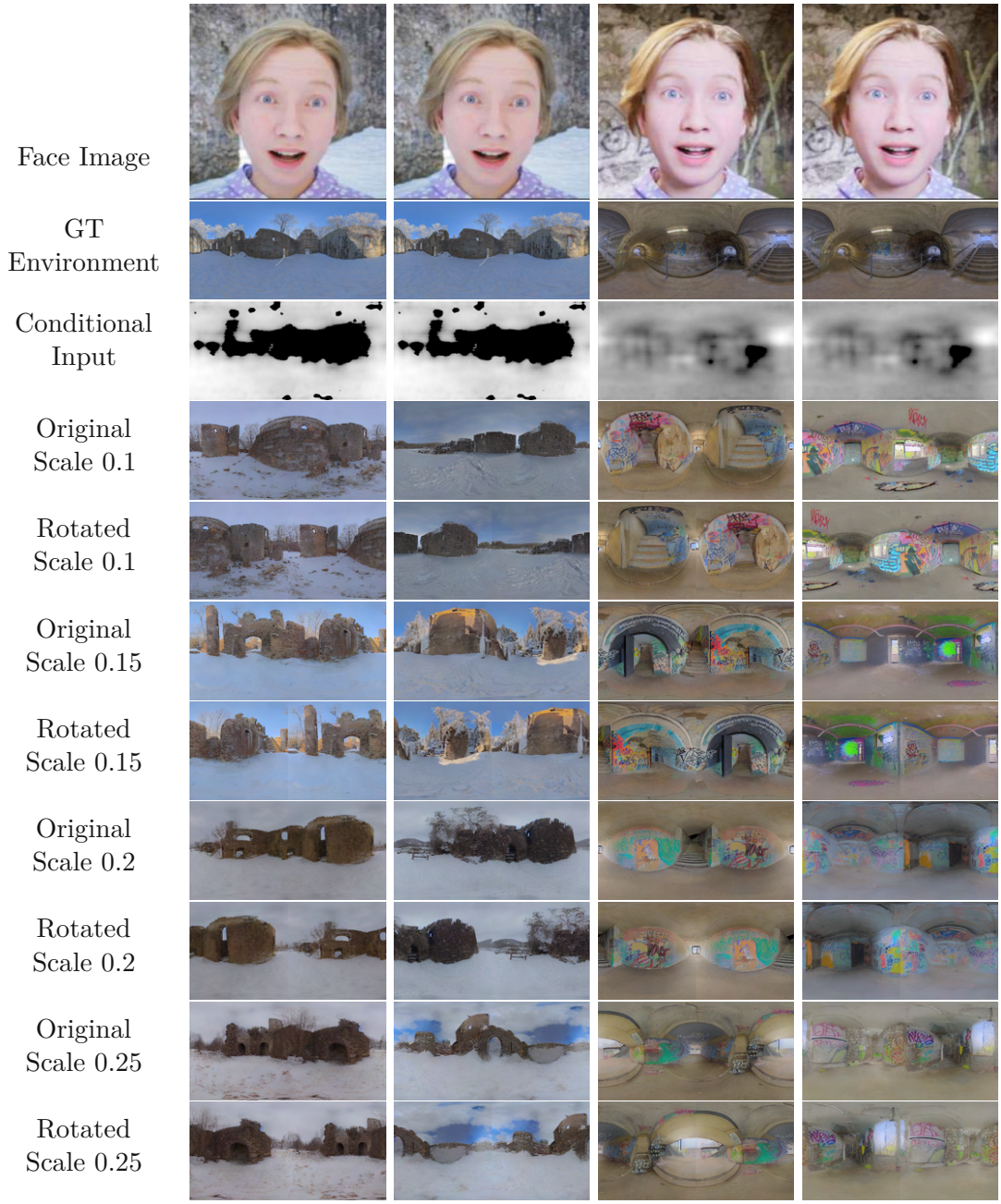
(b) a TOK kitchen with light coming in from the windows

(c) a TOK 360 view of a field with trees and buildings in the distance, with the sun in the middle of the sky behind it. The sun is shining brightly in the middle of the sky

(d) a TOK building with trees in the background and a tree in front of the building

Figure 4.18: Influence of changing the *ControlNet* scale. Columns (b) and (d), show the output when generating *BLIP-2* [LLSH23] labels and brightness estimation from face images instead of from ground truth greyscale environment panoramas in (a) and (c) respectively. The ground truth environments and the input face image shown in Columns (c) and (d) are part of the *Laval Face + Lighting HDR Dataset* [CLG+18].

The results in Figure 4.19 further illustrate the influence of varying *ControlNet* scales. The conditional input is calculated using face images from our test set. We present rotated versions of the generated images to visualize artefacts at the borders. Additionally, we experimented with using prompts generated from the ground truth panoramas to prompts generated from the corresponding face images. Columns (b) and (d) show results when using face image prompts. Although these prompts are much shorter and less descriptive, the results do not show a significant difference in quality. Table 4.9 compares the inference speed of our finetuned *SD* model with and without the added *ControlNet*, for different number of inference steps. As expected, the inference time scales linearly with the number of inference steps. Adding the *ControlNet* increases the inference time by approximately 30 %. We used the precomputed text prompts from our dataset, as described in Section 3.3 for this comparison. Creating text prompts on our inference machine with an *NVIDIA RTX 3060Ti* is possible using a quantized version of the *BLIP-2* [LLSH23] model utilizing the *OPT-2.7b* [ZRG+22] LLM. Generating a single text prompt using the described model takes around 4.5 seconds, making it the most time-consuming part of our system.



(a) TOK ruins of a medieval fortress in a snowy landscape with a lake in the foreground and a snowy forest in the background

(b) a TOK snowy area near an old castle

(c) a TOK view of a stairway in a tunnel with graffiti on the wall and a bench on the other side of the stairway

(d) a TOK dark area with graffiti on the wall

Figure 4.19: Effects of rotating the generated panorama images from the finetuned diffusion network by 180 °. The artefacts are clearly visible in every image. Columns (b) and (d), show the output when generating *BLIP-2* [LLSH23] labels from face images instead of from ground truth greyscale environment panoramas in (a) and (c) respectively. The ground truth environment images were obtained from an online source [iHD].

```python
from diffusers import StableDiffusionControlNetPipeline as SDCNP
from diffusers import ControlNetModel
import torch
def generate_panorama(face_image_path,
                      controlnet_path = "latentcat/control_v1p_sd15_brightness",
                      lora_path = "path/to/trained/lora,
                      model_id = "runwayml/stable-diffusion-v1-5"):
    # Load the pretrained Stable Diffusion 1.5 model
    pipe = SDCNP.from_pretrained(model_id,
                                 controlnet=controlnet,
                                 torch_dtype=torch.float16,
                                 variant="fp16",
                                 use_safetensors=True,
                                 safety_checker = None,
                                 requires_safety_checker = False).to("cuda:0")
    # Load our custom LORA + Dreambooth model
    pipe.load_lora_weights(lora_path)
    # Load the ControlNet model
    controlnet = ControlNetModel.from_pretrained(controlnet_path,
                                                 torch_dtype=torch.float16,
                                                 use_safetensors=True,
                                                 safety_checker = None,
                                                 requires_safety_checker = False)
    # Generate caption for the face image. Special token TOK has to be present
    prompt = get_blip_prompt_from_face(face_image_path)
    # Estimate the brightness of the face image using face-to-environment model
    brightness_estimation = estimate_brightness_from_face(face_image_path)
    # Generate the final image
    final_environment_image = pipe(prompt,
                  brightness_estimation,
                  controlnet_conditioning_scale = 0.7,
                  height=256,
                  width=512,
                  num_inference_steps=25,
                  guidance_scale = 2.5).images
```

Source Code 4.1: *Python* Code to generate a panorama from a single face image using the *Diffusers* library.

CHAPTER 5

Conclusion

In this chapter, we summarize our work. Section 5.1 discusses the current limitations of our system and proposes ideas for future work to address these challenges. Section 5.2 provides a compact overview of our main contributions and insights.

5.1 Limitations and Future Work

Although our proposed dataset worked well for training light estimation systems, further improvements could be made. To enhance the realism of the dataset, individual aspects of each *MetaHuman*, such as hair color, hair style, or facial features, could be randomized. Randomly morphing different facial expressions instead of using a fixed set could further improve the individuality of each face image. While we did not directly observe any negative impact of choosing a low number of possible states for our character customization, more randomization should help the model to generalize better for real-world scenarios.

Our brightness estimation network performs well in some scenes but fails to do so in others, particularly when comparing results on the validation set to those on the test set. This is likely due to assigning environment images from one source to either the training/validation set or the test set, resulting in the average brightness being lower by a factor of 100 for the test set. Furthermore, the distribution of indoor to outdoor scenes should be balanced. LDR to HDR upsampling networks could be used if needed. Additionally, real-world examples of indoor scenes should be included as a baseline for comparison.

One major drawback of our diffusion-based environment generation system is the lack of providing texture information to the diffusion model, which can result in images where the overall layout matches the original scene, but the colors are completely different. Incorporating techniques such as background removal and warping onto textures for use with inpainting models could improve the quality of the generated scenes, due to including

parts of the original scene. A similar approach is shown by Wang et al. [WCL+23] for panorama generation from unregistered images. Another limitation is the reliance on text prompts. The system could be enhanced by exploring alternative encoders like SeeCoder [XGW+24], which replaces text encoders with image encoders. This could lead to more accurate scene generation, similar to using parts of the input image with inpainting models. Furthermore, creating the text prompts is the most time-consuming task in our pipeline, so removing the text prompts could drastically speed up the proposed system. Our proposed method of using a *ControlNet* [ZRA23] to ensure that the output of the diffusion network matches the layout of equirectangular environment images only partially works. In some cases the scene layout is fine, but the colors are different on the edges of the image, resulting in very visible artifacts when rotating the panorama. Combining our method with the latent blending method proposed by Feng et al. [FLCX23] could improve this problem. Because we used an existing *ControlNet* trained to recolorize greyscale images, our diffusion model sometimes fails to generate meaningful output when the strength of the *ControlNet* is too high, due to the blurry appearance of the estimated brightness images. Training a model to estimate a parameter that describes the accuracy of the estimated brightness images could be used to automatically adapt the strength parameter. Otherwise, training a new *ControlNet* on estimated brightnesses should also help to improve the results. The biggest limitation is that our system does not produce HDR images, which are crucial for describing realistic light information. Multiple solutions for this problem could be explored:

- Adding an additional trainable layer combining the diffusion output with the estimated HDR brightness
- Using existing LDR to HDR upsampling systems
- Retraining the diffusion model to directly output HDR data
- Using exposure bracketing with versions of similar outputs by modifying the noise scheduling of the diffusion model

5.2 Summary

In this thesis, we present a novel technique to generate datasets for light source estimation from face images. We used realistic digital human characters illuminated by HDR environment maps. High-quality face images were rendered using the *Path Tracer* module from *Unreal Engine*. Through parameter augmentation, including random environment rotation and varying body position and facial gestures, we diversified the dataset with a wide range of appearances. We showed that CNN-based brightness estimation networks can successfully recover dominant light directions when trained on our proposed dataset, even with unseen real-life data.

For realistic-looking panorama image generation, we finetuned a state-of-the-art diffusion model, enabling text-to-image panorama generation from a single text prompt. The

finetuned model delivers impressive results compared to the original diffusion model, by recreating the layout of equirectangular environment images. By integrating a *ControlNet* [ZRA23] pretrained on brightness images, we allow for conditioning the model using estimated scene brightness from the CNN-based brightness estimation network, enhancing the realism of the generated panoramas by guiding the model to follow the layout of the original scene. The proposed system can be described in three steps:

1. Estimating the scene brightness using our CNN-based face-to-environment model.
2. Generating a text prompt describing the background of the face image with an existing image-to-text model.
3. Creating a 360 degree panorama image using our finetuned diffusion-network, conditioned with the generated text prompt and estimated scene brightness.

The limitations mentioned in the previous section provide a great starting point for further research. Furthermore, we can say that the system shows promising results when looking at the generated images in Section 4.2. Integrating the proposed solution into existing applications in augmented reality could therefore already be beneficial, by enabling use cases such as realistic shading of artificial objects.

List of Figures

List of Tables

Acronyms

CNN convolutional neural network. ix, xi, 2, 3, 5–7, 12, 13, 23, 24, 32, 50, 51, 53, 54

GAN generative adversarial network. 8, 12, 13, 53

HDR high dynamic range. 2, 11–13, 15, 19, 23, 35, 41, 49, 50, 53, 57

LDR low dynamic range. 11, 12, 49, 50

LLM large language model. 10, 16, 19, 46

LoRA Low-Rank Adaptation. 10, 41

MAE mean average error. 28, 29, 34

MSE mean squared error. 28, 29, 34

RMSE root mean squared error. 28, 34

SD Stable Diffusion. 9, 10, 23, 24, 41, 43, 44, 46, 53, 57

VAE variational autoencoder. 8

Bibliography

[BGH+] Christophe Bolduc, Justine Giroux, Marc Hébert, Claude Demers, and Jean-François Lalonde. beyondthepixel/learning tasks/prepare_dataset.py at main · lvsn/beyondthepixel. https://github.com/lvsn/beyondthepixel/blob/main/Learning%20Tasks/prepare_dataset.py. (Accessed on 04/23/2024).

[BGH+23] Christophe Bolduc, Justine Giroux, Marc Hébert, Claude Demers, and Jean-François Lalonde. Beyond the pixel: a photometrically calibrated HDR dataset for luminance and color prediction. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, October 2023.

[CHL+22] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. arXiv:2210.11416.

[CLG+18] Dan A. Calian, Jean-François Lalonde, Paulo Gotardo, Tomas Simon, Iain Matthews, and Kenny Mitchell. From Faces to Outdoor Light Probes. *Computer Graphics Forum*, 37(2):51–61, 2018.

[Deb08] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 2008 Classes*, pages 1–10, Los Angeles California, August 2008. ACM.

[DMAC03] Frédéric Drago, Karol Myszkowski, Thomas Annen, and Norishige Chiba. Adaptive Logarithmic Mapping For Displaying High Contrast Scenes. *Computer Graphics Forum*, 22(3):419–426, November 2003.

[DT] Blender Development Team. Blender (version 3.2.0). https://www.blender.org/. (Accessed on 04/24/2024).

[EGH21] Farshad Einabadi, Jean-Yves Guillemaut, and Adrian Hilton. Deep Neural Models for Illumination Estimation and Relighting: A Survey. *Computer Graphics Forum*, 40(6):315–331, September 2021.

[FCZ+24] Fan Fei, Yean Cheng, Yongjie Zhu, Qian Zheng, Si Li, Gang Pan, and Boxin Shi. SPLiT: Single Portrait Lighting Estimation via a Tetrad of Face Intrinsics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):1079–1092, February 2024.

[FLCX23] Mengyang Feng, Jinlin Liu, Miaomiao Cui, and Xuansong Xie. Diffusion360: Seamless 360 Degree Panoramic Image Generation based on Diffusion Models, November 2023. arXiv:2311.13141.

[Gama] Epic Games. MetaHuman | Realistic Person Creator. `https://www.unrealengine.com/en-US/metahuman`. (Accessed on 03/07/2024).

[Gamb] Epic Games. Path tracer | epic developer community. `https://dev.epicgames.com/documentation/en-us/unreal-engine/path-tracer-in-unreal-engine`. (Accessed on 04/23/2024).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GHS+19] Marc-Andre Gardner, Yannick Hold-Geoffroy, Kalyan Sunkavalli, Christian Gagne, and Jean-Francois Lalonde. Deep Parametric Indoor Lighting Estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7174–7182, Seoul, Korea (South), October 2019. IEEE.

[GPM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., December 2014.

[GSY+17] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to Predict Indoor Illumination from a Single Image, November 2017. arXiv:1704.00090.

[GZH+16] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 87–102, Cham, September 2016. Springer International Publishing.

[Hav] Poly Haven. The public 3d asset library. `https://polyhaven.com/`. (Accessed on 04/18/2024).

[HGAL19] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., December 2020.

[Hol24] Yannick Hold. Soravux/skylibs, April 2024.

[HSW+21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. arXiv:2106.09685.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, June 2016.

[iHD] iHDRI.COM. lighting for creatives. `https://www.ihdri.com/`. (Accessed on 04/18/2024).

[Iqb18] Haris Iqbal. HarisIqbal88/PlotNeuralNet v1.0.0. Zenodo, December 2018.

[KK14] Sebastian B. Knorr and Daniel Kurz. Real-time illumination estimation from faces for coherent rendering. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 113–122, September 2014.

[KK19] Peter Kán and Hannes Kaufmann. DeepLight: Light source estimation for augmented reality using deep learning. *The Visual Computer*, 35(6-8):873–883, June 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[KW22] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. arXiv:1312.6114.

[LB95] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[LLSH23] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large

language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 19730–19742. PMLR, July 2023.

[LM14] Jean-Francois Lalonde and Iain Matthews. Lighting Estimation in Outdoor Image Collections. In *2014 2nd International Conference on 3D Vision*, pages 131–138, Tokyo, December 2014. IEEE.

[LMP+20] Chloe LeGendre, Wan-Chun Ma, Rohit Pandey, Sean Fanello, Christoph Rhemann, Jason Dourgarian, Jay Busch, and Paul Debevec. Learning Illumination from Diverse Portraits, August 2020. arXiv:2008.02396.

[MD24] Konstantin Mishchenko and Aaron Defazio. Prodigy: An Expeditiously Adaptive Parameter-Free Learner, March 2024. arXiv:2306.06101.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., December 2019.

[PKA+09] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3D Face Model for Pose and Illumination Invariant Face Recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301. IEEE, September 2009.

[PT] Apolinário Passos and Linoy Tsaban. Dreambooth lora with stable diffusion. `https://github.com/huggingface/diffusers/blob/main/examples/advanced_diffusion_training/train_dreambooth_lora_sd15_advanced.py`. (Accessed on 06/13/2024).

[RBL+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695. IEEE, June 2022.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 8748–8763. PMLR, July 2021.

[RLJ+23] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510. IEEE, June 2023.

[SBT+19] Tiancheng Sun, Jonathan T. Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single Image Portrait Relighting. *ACM Transactions on Graphics*, 38(4):1–12, August 2019.

[SG17] Vinay K Sriram and Wesley Griffin. A sampling agnostic software framework for converting between texture map representations of virtual environments. *Journal of Research of the National Institute of Standards and Technology*, 122:1, May 2017.

[SNWS20] Alejandro Sztrajman, Alexandros Neophytou, Tim Weyrich, and Eric Sommerlade. High-Dynamic-Range Lighting Estimation From Face Portraits. In *2020 International Conference on 3D Vision (3DV)*, pages 355–363, Fukuoka, Japan, November 2020. IEEE.

[TZC+23] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. MVDiffusion: Enabling Holistic Multi-view Image Generation with Correspondence-Aware Diffusion, December 2023. arXiv:2307.01097.

[vPPL+22] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022.

[WCL+23] Jionghao Wang, Ziyu Chen, Jun Ling, Rong Xie, and Li Song. 360-Degree Panorama Generation from Few Unregistered NFoV Images, August 2023. arXiv:2308.14686.

[Wen21] Lilian Weng. What are diffusion models? `https://lilianweng.github.io/posts/2021-07-11-diffusion-models/`, Jul 2021. (Accessed on 06/13/2024).

[WPL18] Henrique Weber, Donald Prévost, and Jean-Francois Lalonde. Learning to estimate indoor lighting from 3d objects. In *2018 International Conference on 3D Vision (3DV)*, pages 199–207. IEEE, September 2018.

[Wro] Grzegorz Wronkowski. Library of 20k hdri maps, textures, hdri timelapses for cg artists. `https://hdrmaps.com/`. (Accessed on 04/18/2024).

[WYLL22] Guangcong Wang, Yinuo Yang, Chen Change Loy, and Ziwei Liu. StyleLight: HDR Panorama Generation for Lighting Estimation and Editing, July 2022. arXiv:2207.14811.

[XGW+24] Xingqian Xu, Jiayi Guo, Zhangyang Wang, Gao Huang, Irfan Essa, and Humphrey Shi. Prompt-free diffusion: Taking "text" out of text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8682–8692, June 2024.

[YZTL18] Renjiao Yi, Chenyang Zhu, Ping Tan, and Stephen Lin. Faces as Lighting Probes via Unsupervised Deep Highlight Extraction. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018*, volume 11213, pages 321–338. Springer, 2018.

[ZL17] Jinsong Zhang and Jean-Francois Lalonde. Learning High Dynamic Range from Outdoor Panoramas. In *2017 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4529–4538, Venice, October 2017. IEEE.

[ZLLS23] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, February 2023. `https://D2L.ai`.

[ZRA23] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847, October 2023.

[ZRG+22] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models, June 2022. arXiv:2205.01068.