Deep Light Direction Reconstruction from single RGB images

Markus Miller
Dept. of Computer Science and Mathematics
University of Applied Sciences Munich
Lothstr. 64
D-80335, Munich, Bavaria
markus.miller@hm.edu

Alfred Nischwitz
Dept. of Computer Science and Mathematics
University of Applied Sciences Munich
Lothstr. 64
D-80335, Munich, Bavaria
nischwitz@cs.hm.edu

Rüdiger Westermann
Chair of Computer Graphics and Visualization
Technical University Munich
Boltzmannstr. 3/II
D-85748, Garching, Bavaria
westermann@tum.de

ABSTRACT

In augmented reality applications, consistent illumination between virtual and real objects is important for creating an immersive user experience. Consistent illumination can be achieved by appropriate parameterisation of the virtual illumination model, that is consistent with real-world lighting conditions. In this study, we developed a method to reconstruct the general light direction from red-green-blue (RGB) images of real-world scenes using a modified VGG-16 neural network. We reconstructed the general light direction as azimuth and elevation angles. To avoid inaccurate results caused by coordinate uncertainty occurring at steep elevation angles, we further introduced stereographically projected coordinates. Unlike recent deep-learning-based approaches for reconstructing the light source direction, our approach does not require depth information and thus does not rely on special red-green-blue-depth (RGB-D) images as input.

Keywords

Light, source, direction, estimation, reconstruction, RGB, deep learning.

1 INTRODUCTION

In the past decade, augmented reality (AR)-capable hardware and virtual reality (VR) devices have become increasingly available. Successful AR applications should create an immersive user experience, as immersion in AR is important to prevent a barrier between the virtual world and real world that can impede user's acceptance of AR. To minimise this barrier, it is important to avoid mismatches between virtual and real objects, such as illumination deviations. To achieve consistent illumination between virtual and real objects, virtual illumination in an AR application must adapt to the real illumination conditions.

Depending on the illumination model used to render virtual objects, virtual illumination may consist of an emissive and reflective light term, known as the rendering equation (Kajiya, 1986). The reflective term is an integral over the entire positive hemisphere above a given point, containing the bidirectional reflectance distribution function (BRDF) and incident light. To begin with a simple scenario, we restrict our approach to situations in which only one infinite point light source illuminates the scene. Thus, we neglect the emissive light term of the rendering equation, more complex lighting situations (e.g. extended, textured and multiple light sources) and indirect illumination from other surfaces. However, with an infinite point light source, we can obtain illumination conditions that contain the most important elements for three-dimensional spatial perception. In our study it is thus sufficient to focus on the light direction to be reconstructed to achieve virtual illumination while providing consistency for an immersive AR experience. We further disregard the intensity of the light, as it does not affect the illumination unless it exceeds a certain range, which constitutes a special case to be addressed in future work.

Illuminated scenes, as perceived by humans, are the result of the interaction of several parameters present in a scene. By observing the shading and light reflection of an object's surface, the human brain can estimate the characteristics and shape of the surface. The human

brain can further estimate the origin of the main light source illuminating a scene, which is usually the sun. Deep neural networks (DNNs) can also learn to estimate a light's origin from visual input (Section 2) and can even reformat this information to be used as a parameter in virtual illumination models. Immersive and responsive AR applications may require updated illumination parameters for each frame; as a result, approaches with less complex network architectures are preferable. In addition, classification approaches require a large number of classes to reconstruct a light's origin with satisfactory resolution; therefore, creating suitable training datasets is a cumbersome task. In contrast, regression approaches yield continuous output values and therefore only require as many output neurons as the number of parameters to be reconstructed.

Considering the aforementioned requirements, we propose a deep-learning-based regression approach to reconstruct the light direction in a scene given as azimuth and elevation angles (ϕ, θ) from RGB input images using a modified VGG-16 convolutional neural network (CNN). In addition, we introduce stereographic coordinates (s_x, s_y) instead of angular coordinates (ϕ, θ) to reduce inaccuracies due to the coordinate uncertainty occurring at steep elevation angles. Our approach shows that it is plausible to reconstruct the light direction in a scene from a single RGB image of the illuminated scene using a CNN. Our approach does not require additional depth information and thus can be used on devices without special depth sensors.

The main contributions of our proposed approach can be summarised as follows:

- Reconstruction of the dominant light source direction is possible using only RGB input images, and does not require special RGB-D information.
- A stereographic coordinate representation significantly improves the estimation results.

Our approach exceeds existing learning-based approaches for inferring light direction from images (Kán and Kaufmann, 2019) in that it enables reconstruction from RGB images. This is achieved by increasing the amount of training data and using pre-trained model weights, which has been successfully applied to a similar problem (Marques et al., 2018), and inferring the light direction in a stereographic coordinate representation.

2 STATE OF THE ART

Classical image processing approaches reconstruct the location of a visible light source within an image (Laskowski, 2007) or determine the light direction with additional user input (Lopez-Moreno et al., 2009). However, this is impractical for immersive AR applications.

More recent approaches utilise DNNs to reconstruct light source parameters, such as the direction, location or intensity of the light. Using the association between an object and its shadow, it is possible to determine the screen space light direction (Wang et al., 2020), which can be converted into world space coordinates with additional camera parameters. In addition, generative adversarial networks are useful tools to directly create artificial content. They also can add missing shadows to marked virtual shadowless objects in real scene images (Liu et al., 2020) and output a complete AR scene image. AR content can also be created with image-based lighting (IBL) renderers, which utilise environment textures containing the illumination details of a scene. Real scene environment textures can be derived from RGB input images using regression (Gardner et al., 2017) and continuously loaded into video memory to match illumination changes in the real scene. LeGendre et al. (2019) extended this concept by deriving more detailed textures for IBL renderers from RGB images using an encoder-decoder network. Garon et al. (2019) introduced a DNN that uses an RGB image and an image location to estimate a fifth order spherical harmonic representation of the lighting, which can be used to illuminate virtual objects placed at this location. A more straightforward approach to create AR content, is to simply place a virtual light source in the virtual scene that is combined with the real scene. To determine where to place the virtual light source to match the real illumination conditions, either the location or direction of the light source is required. The light direction can be reconstructed by classifying real-scene RGB images into directional classes using a neural network (Pemasiri et al., 2015).

Marques et al. (2018) used a similar concept in their approach to overlay a VR scene with an image of a user's hand pose. Because the real illumination conditions were embedded in the hand pose image, it was more practical to simply adjust the virtual illumination to match the real conditions. To achieve this, the authors trained a residual CNN (ResNet) starting with initial model weights pre-trained on the ImageNet and COCO datasets to classify the point light source in the VR scene that was most suitable for producing similar illumination. With 100 possible point light sources available in the virtual scene, their network achieved a top 1 accuracy of approximately 82% in classifying the correct point light source.

Kán and Kaufmann (2019) proposed a regression model to reconstruct continuous azimuth and elevation angle values indicating the dominant light direction in real scenes using RGB-D input images for their ResNet. They also conducted experiments using the RGB com-

ponents of their RGB-D dataset; however, their network failed when it relied solely on these three colour channels. They integrated their network into an AR application and achieved a mean angular error of approximately 28 °and an inference time of approximately 380 ms while running on a central processing unit.

Both approaches used ResNet to avoid exploding or vanishing gradients and both trained their networks with synthetic training data. In contrast, Marques et al. (2018) generated images of hand poses that were illuminated from different angles with the Unreal Engine, Kán and Kaufmann (2019) rendered five different objects illuminated from different angles using a Monte Carlo path tracing renderer. Kán and Kaufmann (2019) further attempted to include real training images; however, their network performed best when trained purely with synthetic data. Apart from the additional image channel and generation process, the main difference between the training data used in these approaches was the dataset size. Kán and Kaufmann (2019) used a small synthetic training set of 23,111 images. In addition, they had 5,650 real images available that were omitted for training. Marques et al. (2018) had a synthetic dataset of 83,799 images, from which they used 54,471 images as training data for their pre-trained ResNet.

The rendering equation (Section 1) suggests all required information to be present in an RGB image to reconstruct the illumination origin in a scene, however, given the unsuccessful experiment of Kán and Kaufmann (2019) using only RGB information, we begin with a simple scenario to investigate if RGB images provide enough information to reconstruct the dominant light direction. Considering the different dataset sizes and training strategies of Kán and Kaufmann (2019) and Marques et al. (2018), a DNN with model weights pre-trained on a large dataset as initial values for training and a large number of training examples appears to be a reasonable basis for a regression approach that aims to reconstruct the dominant light direction from RGB images.

3 LIGHT DIRECTION INFERENCE

In our proposed approach, we aim to reconstruct the azimuth and elevation angles (ϕ, θ) of the dominant light direction from RGB input images. This is similar to the approach of Kán and Kaufmann (2019); however, our approach focuses on light direction reconstruction using only RGB information and explicitly omits the depth information. We assume that the network is capable of reconstructing the light direction without additional depth information. For this, we use an ImageNet (Russakovsky et al., 2015) pre-trained VGG-16-like CNN (Simonyan and Zisserman, 2014) as a regression model, and train it with a dataset consisting of synthetic and real images (Fig. 1). Considering the real-time requirements for immersive and responsive AR applications, we decided to use a VGG-16-like network, as this architecture achieved the best performance relative to its complexity in the ImageNet competition (Russakovsky et al., 2015). By greatly increasing the dataset size, we were able to improve the reconstruction performance of the network for (ϕ, θ) using RGB input images.

We start with a network to predict continuous values (ϕ, θ) for azimuth and elevation in the range $(0°, 0°)$ to $(360°, 90°)$. The influence of the azimuth angle ϕ on the prediction error of the network decreases as the elevation angle θ gradually reaches $\theta = 90°$, and ϕ loses any influence on the prediction error, as $\theta = 90°$ denotes the pole of the hemisphere. Thus, ϕ cannot be properly estimated by the network, leading to an increased angular error at steep values of θ, as illustrated in Fig. 7b. To compensate for this, we convert (ϕ, θ) into stereographic coordinates (s_x, s_y) ranging from $(-1, -1)$ to $(1, 1)$, and train a second network predicting the light direction in stereographic coordinates, as they do not suffer from coordinate uncertainty.

3.1 Stereographic Coordinates

A stereographic representation (s_x, s_y) of the angular coordinates (ϕ, θ) is introduced to avoid coordinate uncertainty at $\theta = 90°$. The coordinate uncertainty refers to an undefined value of ϕ at $\theta = 90°$, as any value of ϕ denotes the pole of the sphere, leading to ϕ not able to be properly estimated by the network, as ϕ no longer has any meaningful influence on the prediction error.

A stereographic projection (Fig. 2) projects spherical coordinates onto a circular projection plane E_p. The southern pole S of the spherical coordinate system is the projection centre, and as such, undefined in the stereographic domain. However, this is not a problem, as any light source located in the southern hemisphere would shade the entire scene, not providing any useful information for estimating its direction. From S, a given point A is projected onto E_p resulting in A'. The distance m from the origin O' of E_p to A' computes (s_x, s_y) by scaling $\cos(\phi)$ or $\sin(\phi)$, respectively. The spherical coordinate space can be expressed by its origin O and radius r. By assuming that the domain region is half of a unit sphere, one can assume $r = 0.5$, as it simplifies the computation of (s_x, s_y) in

$$m(\theta) = 2 \cdot r \cdot \tan\left(\frac{90° - \theta}{2}\right) \tag{1}$$

$$= \tan\left(\frac{90° - \theta}{2}\right) \tag{2}$$

$$s_x(\phi, \theta) = m(\theta) \cdot \cos(\phi) \tag{3}$$

$$s_y(\phi, \theta) = m(\theta) \cdot \sin(\phi) \tag{4}$$

To compare the results of Net_{s_x, s_y} using stereographic coordinates to the results of $\text{Net}_{\phi, \theta}$, the predicted stere-

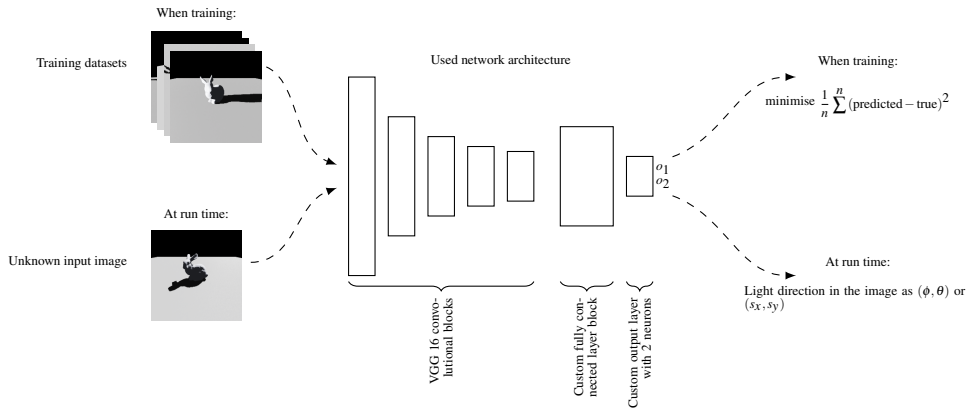


Figure 1: Diagram of network architecture and training and test procedures. The architecture utilises the standard VGG-16 convolutional blocks and a custom fully connected block and output layer with two neurons. Here the light direction is inferred from red-green-blue (RGB) input images as (ϕ,θ) or (s_x,s_y), respectively.

ographic coordinate values (s_x,s_y) can be transformed back into angular values (ϕ,θ) using

$$m(s_x,s_y) = l(s_x,s_y) = \sqrt{s_x^2+s_y^2} \quad \forall\, m \neq 0 \tag{5}$$

$$\theta(s_x,s_y) = 90 - 2\cdot\arctan\left(l(s_x,s_y)\right) \tag{6}$$

$$\phi(s_x,s_y) = \arcsin\left(\frac{s_y}{l(s_x,s_y)}\right) \tag{7}$$

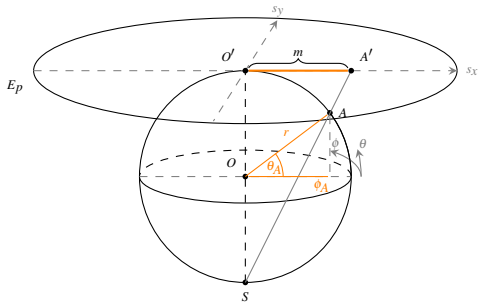


Figure 2: Depiction of stereographic projection using example $A = (\phi_A,\theta_A)$ with $\phi_A = 0°, \theta_A = 37°$, resulting in stereographic coordinates $A' = (s_{xA},s_{y_A}) = (0.4986,0)$.

4 NETWORK ARCHITECTURE AND TRAINING

We start with a pre-trained VGG-16 CNN in Keras and modify the fully connected (FC) layers of the network architecture to meet our requirements. For each of the two light direction representations, one dedicated network is trained and optimised: the first network $\text{Net}_{\phi,\theta}$ is predicting the light direction in angular values of (ϕ,θ) and the second network Net_{s_x,s_y} is predicting stereographic coordinates (s_x,s_y).

Prior to training, well-performing hyperparameter combinations are identified. To achieve this, the FC layers and the soft-max output layer are removed from the standard VGG-16 architecture, and only the five convolutional blocks are retained. The train flag is disabled on these five convolutional blocks during training; thus, all pre-trained weights are kept. After the convolutional blocks, a dynamically generated FC subnetwork is added, which consists of a FC input layer, a variable number of hidden layers and a FC output layer containing two neurons. The dynamically generated subnetwork is created just before training using parameters from a range of hyperparameters. Optimisation is performed using Talos[1]. Talos runs a grid search across the entire specified set of hyperparameter values, returning a list of hyperparameter combinations and trained model weights for each FC subnetwork. Each resulting subnetwork is trained for a single epoch with a uniform learning rate that is automatically adjusted to the used optimiser on a reduced synthetic training dataset of 20,000 images. The reduced dataset is split into a training and validation set with a ratio of 80:20. After the optimisation process, the two hyperparameter combinations and subnetworks with the best estimation performance are selected for further refinement.

To refine the two most accurate networks and determine the most suitable network architectures, the fifth convolutional block is unlocked to be trained, simultaneously reducing the learning rate to $1/1,000$ of the optimised rate. Thus, the fifth convolutional block is able to adapt to the new task while maintaining and improving what was previously learned by the dynamic FC layers. To make the network more robust to input images of varying brightness and representing objects of varying sizes

[1] Autonomio Talos [Computer software]. (2019). Retrieved from http://github.com/autonomio/talos.

Hyperparameter	$\text{Net}_{\phi,\theta}$	Net_{s_x,s_y}
optimiser	Adam	Adam
batch size	32	32
uniform learning rate	2	1
hidden layers	0	0
FC input neurons	4,096	4,096
activation	leaky ReLU	ReLU
dropout	0.25	0.25

Table 1: Hyperparameters of the resulting networks.

and locations, the training data are augmented before use in the training process by shifting the images vertically and horizontally, zooming in and out and varying the brightness. The refinement training is performed for up to 400 epochs, reducing the learning rate by a 10th if the validation mean absolute error (MAE) did not decrease over 13 epochs. If the validation MAE did not decrease over a total of 20 epochs, the training is stopped. All networks use a mean squared error loss function. Each network is trained with three different datasets: one purely synthetic, one purely real and one mixed dataset. The synthetic training dataset contains 100,000 images from the synthetic image dataset, while the real training dataset contains 800 images from the real image dataset[2]. The mixed training dataset is a combination of the real and synthetic image datasets, containing 800 real and 99,200 synthetic images.

Finally, all resulting fully trained network architectures and hyperparameter sets are tested on synthetic and real test sets and the best in each category is selected as the final architecture and hyperparameter combination (Table 1) for $\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y}. The test sets consist of images the networks had not seen before and contain 10,000 synthetic images and 61 real images, respectively. Training and testing are performed on NVidia RTX 2080 Ti and NVidia Titan XP graphics processing units.

4.1 Datasets

Kán and Kaufmann (2019) used 23,111 synthetic images in their training but were unable to achieve a satisfactory reconstruction using only the RGB components of their training data. Marques et al. (2018) used a total of 83,799 synthetic RGB images and achieved satisfactory classification results. Hence, we use a large dataset in our approach, assuming that approximately 150,000 real and synthetic images (one third to be designated as test data and two thirds as training and evaluation data) are sufficient. When creating the datasets, the data are directly labelled using the angular values (ϕ,θ) of the configured camera and light settings. For the stereographic network, the angular labels $\Lambda_{\phi,\theta}$ are converted into stereographic labels Λ_{s_x,s_y} (Section 3.1).

4.1.1 Synthetic Dataset

To create the synthetic image dataset, a simple scene is created using the Unreal Engine that is composed of a single object placed on a base surface illuminated by a directional light source. Due to the wide variety of possible light directions, identifying shadow bias values that do not produce artefacts is cumbersome. Therefore, the RTX shadow capabilities of the NVidia RTX 2080 Ti are used for rendering the shadows. Five models of varying complexity are used as the centre object: a box, a cone, the Stanford bunny, the Stanford Buddha and a sphere (Fig. 3). Suitable physically based rendering (PBR) surface materials (Karis and Epic Games, 2013) are assigned to the base surface and the models to capture the material structure of their real-world counterparts.

A total number of 1,225 different light directions are obtained by illuminating the scene with a directional light source L from angles evenly distributed around the centre object. The direction values of L (ϕ_L,θ_L) range from $(0^\circ,5^\circ)$ to $(360^\circ,90^\circ)$ with a step size Δ_L of $(5^\circ,5^\circ)$. Light directions from below are not considered. Duplicate images at light elevation angles of $\theta_L = 90^\circ$ are omitted. At elevation angles of $\theta_L = 90^\circ$, a single image using an azimuth value of $\phi_L = 0^\circ$ is generated. The camera C is placed in a similar fashion as L at a fixed distance with spherical coordinates (ϕ_C,θ_C), ranging from $(0^\circ,1^\circ)$[3] to $(360^\circ,90^\circ)$ with a step size Δ_C of $(45^\circ,30^\circ)$, resulting in 32 different camera positions. Using the same camera settings for point-symmetric objects, such as the sphere, would result in duplicate images. Therefore, redundant camera settings are omitted, reducing 32 different camera positions to four positions placed around the sphere. Though having an axis-symmetric model structure, the cone object appears different from every angle due to its wrinkled paper surface material and is therefore rendered with the entire set of camera positions.

Combining camera positions and light directions, 39,200 images of the box, cone, Stanford bunny and the Stanford Buddha, and 4,900 images of the sphere are obtained. Therefore, the entire synthetic dataset contains 161,700 images that are directly captured in the required resolution of 224×224 pixels. In addition to capturing the images, we further export the angular labels of the synthetic dataset $\Lambda^s = (\phi_L,\theta_L)$.

4.1.2 Real Dataset

The images in the real dataset are photographed with a Canon EOS 5D Mark II under controlled light condi-

[2] Due to the great effort required to generate real data, we only have 861 images available thus far, 800 for training and 61 for testing. However, we plan to increase our dataset in the future.

[3] 1° instead of 0° on the first step for visibility reasons, the next elevation step is 30°

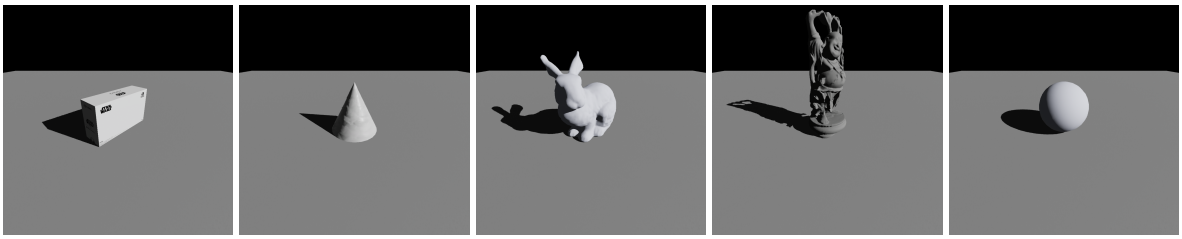


Figure 3: Examples of three-dimensional models used.

tions. The laboratory is completely darkened so that no light source other than the used halogen spotlight would interfere. A box, a three-dimensional print of the Stanford bunny and a ping-pong ball (Fig. 4) as real-world representations of their virtual counterparts are placed on a table with a surface material similar to the base surface material in our synthetic image dataset.

Camera C is placed on a height-adjustable tripod focussing on the centre object O, which is photographed from three different heights, representing elevation angles in the range $[30°, 65°]$. To adjust the view angle of C, O is rotated in the centre of the table, and photographs are taken from seven different angles. To vary the light, the height-adjustable tripod with an attached spotlight L is moved to five different locations[4] around the table, and photographs of the scene illuminated from four different heights are taken.

This way, by the end of the available time frame, a labelled real image dataset is obtained, containing a total of 861 images (i.e. 403 images of the box, 402 images of the Stanford bunny and 56 images of the sphere). To label the images, the vertical and horizontal distances of the scene are measured before taking the photographs, and the angular labels $\Lambda_{\text{real}}(\phi,\theta)$ are computed using the measured distances. For the computation, the measured distances between the following measuring points are required: O and the base of the camera tripod C_b, O and C, C and C_b, a freely chosen reference point R and C_b, O and the base of the light tripod L_b, O and L, L and L_b, and R and L_b (Fig. 5).

With the measured distances, the spherical coordinates of $C = (\Phi_C, \Theta_C)$ and $L = (\Phi_L, \Theta_L)$ can be computed using the law of cosines as

$$f_{\sphericalangle}(a,b,c) = \arccos\left(\frac{a^2+b^2-c^2}{2\cdot a\cdot b}\right) \tag{8}$$

The camera and light azimuth angles Φ_C and Φ_L are computed using $\Phi_C = f_{\sphericalangle}(\overline{OC_b}, \overline{OR}, \overline{RC_b})$ and $\Phi_L = f_{\sphericalangle}(\overline{OL_b}, \overline{OR}, \overline{RL_b})$, respectively. The elevation angle of the camera Θ_C and of the light Θ_L are computed similarly with $\Theta_C = f_{\sphericalangle}(\overline{OC_b}, \overline{OC}, \overline{CC_b})$ and $\Theta_L = f_{\sphericalangle}(\overline{OL_b}, \overline{OL}, \overline{LL_b})$. Both $C = (\Phi_C, \Theta_C)$ and $L = (\Phi_L, \Theta_L)$ are then used to compute the angular labels of the real images $\Lambda^r = (\phi_{lbl}, \theta_{lbl})$ with

$$\phi_{lbl} = \begin{cases} |\Phi_L - \Phi_C| & \text{if } \Phi_L \geq \Phi_C \\ |360 - |\Phi_L - \Phi_C|| & \text{else} \end{cases} \tag{9}$$

$$\theta_{lbl} = \Theta_C + \Theta_L \tag{10}$$

The photographs are taken as RGB images in a resolution of $2,784 \times 1,856$ pixels, cropped to $1,856 \times 1,856$ pixels with the object centred to keep the aspect ratio and then resized to 224×224 pixels using the default Keras ImageDateGenerator function before being used for training.

5 RESULTS

$\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y} were tested by estimating the light direction on the synthetic and real test datasets. To compare the results of the networks (Table 2), the mean angular estimation error was computed as

$$\overline{E}_{\measuredangle} = \frac{1}{n}\sum^{n} \left|\arccos\left({v_p}^T v_t\right)\right| \tag{11}$$

over all test images between the predicted direction v_p and the ground truth direction v_t. Both v_p and v_t were given in spherical coordinates and therefore needed to be converted to Cartesian coordinates.

Train – Test	$\text{Net}_{\phi,\theta}$	Net_{s_x,s_y}
synth – synth	7.8°	3.7°
synth – real	99.2°	25.5°
real – real	12.4°	8.8°
mixed – real	16.8°	7.1°

Table 2: Average angular error $\overline{E}_{\measuredangle}$ on synthetic and real test data.

Furthermore, each network was trained with three different training datasets: a purely synthetic training dataset (Section 4.1.1), an entirely real training dataset (Section 4.1.2), and a mixed training dataset consisting of images from both synthetic and real image datasets.

To determine whether it is theoretically possible to estimate the light direction from images given only RGB information, the synthetically trained $\text{Net}_{\phi,\theta}$ was tested on synthetic test data images, as this test dataset does

[4] Due to space limitations it was not possible to illuminate the scene from $[0°, 360°]$; hence, the scene was illuminated from directions in the range $[\approx 40°, \approx 270°]$.

Figure 4: Examples of real image dataset displaying the real-world models – a box, a Stanford bunny and a sphere.

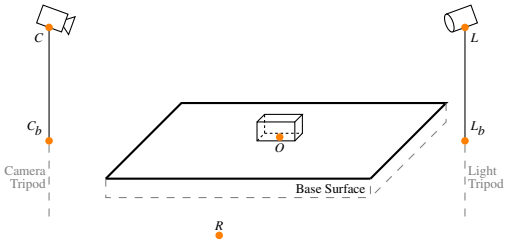


Figure 5: Illustration of measured points in a real scene.

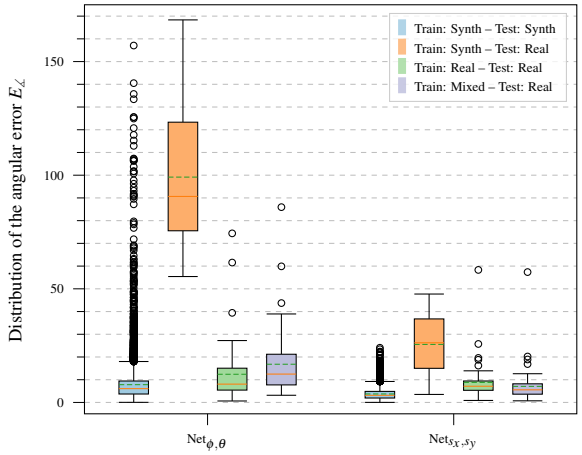


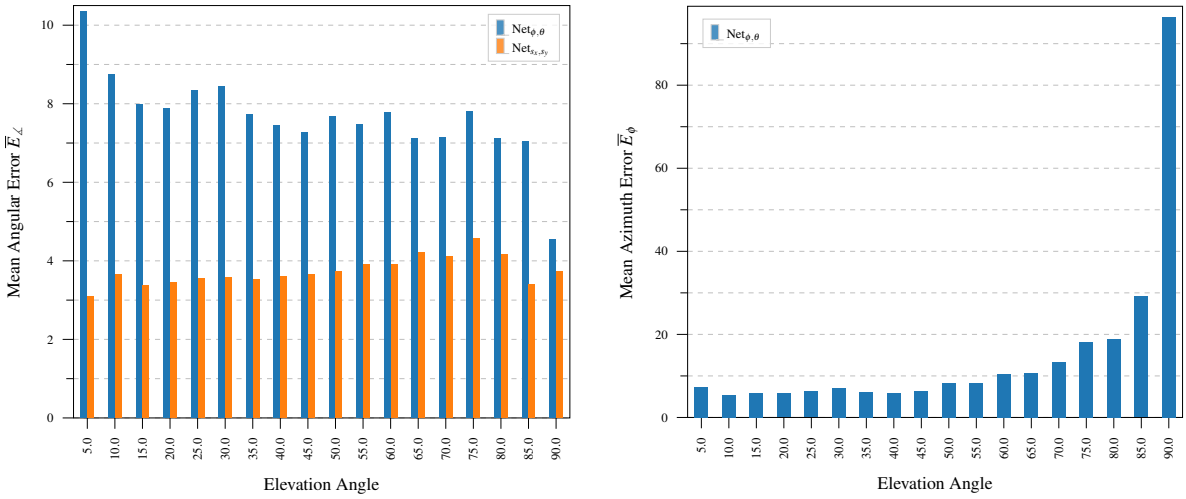
Figure 6: Box-and-whisker diagram of the angular estimation error distribution of $\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y}. The median of the distribution is displayed as a green dashed line, the mean as red line, and outliers as circular marks.

not suffer from noise interference, leading to distorted results. In this test, $\text{Net}_{\phi,\theta}$ achieved a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 7.8°$. This result indicates that reconstructing the light direction from only RGB images is reasonable; however, it requires further refinement. The synthetically trained Net_{s_x,s_y} tested on the synthetic test dataset had the best performance, achieving an error of $\overline{E}_{\measuredangle s_x,s_y} = 3.7°$. Considering the angular error distribution of the synthetically trained networks on synthetic test data (Fig. 6, blue graphs), introducing the improvements of Net_{s_x,s_y} not only reduced the estimation error, but also decreased the range of the outlier deviation.

To evaluate the difficulties in predicting the light direction in spherical coordinates (ϕ,θ), the estimation errors of the networks depending on the elevation angle θ (Fig. 7) were investigated. While $\overline{E}_\measuredangle$ of Net_{s_x,s_y} remained fairly constant over all angles of θ, the errors of $\text{Net}_{\phi,\theta}$ did not reveal any clear trend (Fig. 7a). Therefore, the mean error of the estimated azimuth angle $\overline{E}_\phi$ of $\text{Net}_{\phi,\theta}$ (Fig. 7b) was investigated, revealing an increasing error on steep elevations θ, as initially expected. Considering the scale, the estimation of the light direction in stereographic coordinates (s_x,s_y), performed by Net_{s_x,s_y} only revealed a minor deviation (Fig. 7a, green bars). This observed behaviour supports the initial concept of introducing a stereographic representation, as the results indicate that a network using (s_x,s_y) can avoid this inherent error. Because only a small real test dataset could be provided that contained no examples of steeply illuminated objects, this particular case was investigated using only the synthetic test dataset to obtain statistically useful information.

The synthetically trained networks were tested on real image data to investigate whether they would be effective for real data without any necessary adjustment. However, the domain gap between the synthetic and real datasets appears to be large, as $\text{Net}_{\phi,\theta}$ failed completely with a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 99.2°$. With an error of $\overline{E}_{\measuredangle s_x,s_y} = 25.5°$ (Fig. 6, orange graphs), Net_{s_x,s_y} achieved reasonable estimation performance. To determine whether the small real image training dataset would be sufficient to obtain reasonable estimation performance, the networks, which were trained with real image data, were tested on the real image dataset. In this test run, $\text{Net}_{\phi,\theta}$ improved its performance with a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 12.4°$; Net_{s_x,s_y} outperformed the other network with $\overline{E}_{\measuredangle s_x,s_y} = 8.8°$ (Fig. 6, green graphs). Trained with mixed image data, the networks were tested on real test data to investigate how the small fraction of real image data would benefit from being augmented with the synthetic training set. With a fraction of 0.8% (i.e. a total of 800 real images), the networks achieved a mean angular error of $\overline{E}_{\measuredangle\phi,\theta} = 16.8°$ and $\overline{E}_{\measuredangle s_x,s_y} = 7.1°$ (Fig. 6, purple graphs). Unlike $\text{Net}_{\phi,\theta}$, Net_{s_x,s_y} improved its prediction results by being trained on a mixed dataset as opposed to being trained on a purely real, but small dataset. Augmenting the real dataset with synthetic images appears to worsen the performance in this case, since the synthetically trained $\text{Net}_{\phi,\theta}$ tested on real images performed poorly whereas Net_{s_x,s_y} performed reasonable.

As images of different models were used in the training process, we investigated whether the networks learned



(a) mean angular error $\overline{E}_{\measuredangle}$.

(b) mean azimuth error $\overline{E}_{\phi}$.

Figure 7: Estimation errors of Net$_{\phi,\theta}$ and Net$_{s_x,s_y}$ depending on elevation θ.

a model preference when estimating the light direction. Hence, the mean angular error $\overline{E}_{\measuredangle}$ for each model (Fig. 8) was computed and examined for any significant deviation. Despite having different error amplitudes from each other, the synthetically trained and tested networks only display minor deviations on the model-dependent $\overline{E}_{\measuredangle}$ (Fig. 8a). Trained with mixed image data and tested on the real dataset (Fig. 8b), Net$_{\phi,\theta}$ and Net$_{s_x,s_y}$ display a similar behaviour of the model-dependent $\overline{E}_{\measuredangle}$. Thus, the networks do not appear to favour a certain centre model when estimating the light direction.

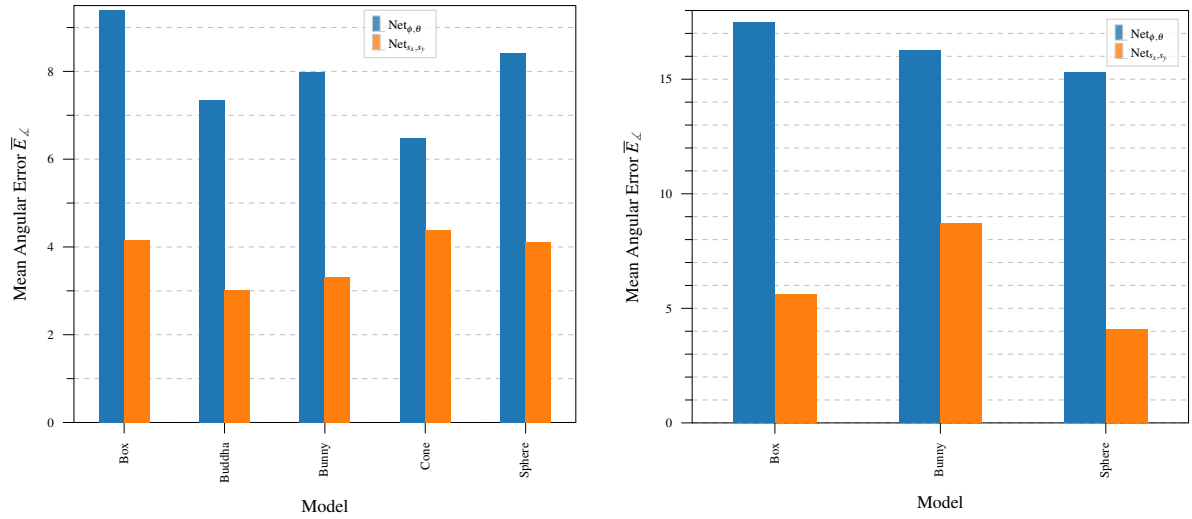
6 DISCUSSION

Considering the performance of the synthetically trained networks on synthetic test data, the results demonstrate that it is possible to reconstruct the dominant light direction (ϕ,θ) in a scene from RGB input images using a VGG-16-like CNN, without requiring additional depth information or relying on special RGB-D images. The light direction reconstruction performance could be further improved by introducing the estimation of stereographic coordinates (s_x,s_y) with Net$_{s_x,s_y}$.

On real test data, the lower prediction accuracy of Net$_{\phi,\theta}$ and Net$_{s_x,s_y}$ are likely to be caused by a domain gap between the synthetic training dataset and the real test dataset, as the prediction performance of the synthetically trained networks on real test data significantly improved by adding even a small fraction of real image data to the training dataset. Considering this domain gap, the prediction performance of Net$_{s_x,s_y}$ on real test data is satisfying, even when synthetically trained.

The small number of real images for testing the networks is a problem, because most of the real images were required for training to improve the prediction performance on real test data, leaving only 61 images available for testing our CNNs, which is critical for obtain statistically useful information. However, the real dataset is sufficient to demonstrate the general feasibility of our proposed approach. Generating the real dataset was a cumbersome task, because to label the images, it was necessary to measure all required distances for each photograph and then prepare the scene for the next scene image, which was time-consuming. Labelling the photos afterwards was not possible, as the labels would have been mere estimates lacking the accuracy necessary for training.

Because we were using a halogen spotlight, which is a spotlight source with small extent in a finite distance in terms of the rendering equation, to illuminate the real scene when taking photographs, there was a structural illumination difference between the directional light used in the synthetic dataset and the extended spotlight in a finite distance in the real dataset. As a directional light source represents a light source, infinitely far away, it is difficult to recreate such a light source in real scenes, as the light source in the scene can not be placed infinitely far away. With increasing distance from an object, however, light from a real source gradually becomes more parallel, transforming into an infinite light source. Considering the distance of the spotlight, which varied between 2 and 3 m, the size of the real models and the input resolution of the CNNs, we considered the difference between the synthetic directional light and the real light source to be negligible. Furthermore, the directional light source in the synthetic dataset did not have a spatial extent, resulting in an umbra without a penumbra, which we attempted to address in our real dataset by using a very small halo-



(a) trained and tested on synthetic image data.

(b) trained with mixed and tested with real image data.

Figure 8: Mean angular errors of $\overline{E}_\angle$ of $\text{Net}_{\phi,\theta}$ and Net_{s_x,s_y} depending on the depicted model.

gen spotlight that was powerful enough to illuminate the scene.

Summarising our contribution to advance the state of the art decribed in Section 2, our findings demonstrate that RGB images are sufficient to estimate the direction of the dominant light source, and that RGB-D images are not required, although they may improve results, which we will investigate in future work. Furthermore, we demonstrated that a stereographic representation of the light direction avoids the coordinate uncertainty of the azimuth angle ϕ at steep elevation angles θ, leading to significantly improved estimation results.

7 FUTURE WORK

We plan to considerably extend our small real dataset to achieve statistically stronger results on real test data. In addition, this extended real dataset can be used to test the proposed approach and additional approaches for different real scene setups. To reduce the effort required to extend the dataset, the generation and labelling process can be automated by a robotic device.

As gathering real image datasets even with an automated device may still require significant effort, we will also pursue the concept of using a semi-real image dataset. We will create such image data by augmenting distinct real scene photographs with synthetic billboards displaying green-screen extractions from images of real objects.

Our long-term goal is illumination reconstruction with DNNs trained on synthetic image datasets that are aided by as little real image data as possible to close the gap between virtual and real scene illumination. Not requiring many real image samples is crucial, as gathering labelled real image data is cumbersome and sometimes not possible. Hence, in the future, we will focus on gaining a better understanding of which details in synthetic training images are important to improve the results achieved by mixed training data. We will first investigate the influence of specific illumination components on the estimation performance, including the surface shading, appearance and presence of a shadow, and indirect lighting. Finally, we will investigate, which DNN architecture and input data in addition to the RGB information can further improve the estimation results.

ACKNOWLEDGEMENTS

We would like to thank the Competence Center Image Processing of the University of Applied Sciences in Munich[5] with all their funding companies[6], as well as the Chair of Computer Graphics and Visualization[7] of the Technical University Munich for allowing us to conduct this research.

We would further like to thank IBM for their support and help. They kindly granted us access to their OpenPOWER[8] deep learning system, which we used for training and optimising our network and other experiments.

Finally, we would like to thank Tobias Kroiss for his assistance in creating the proof of concept, and Bigyan Karki for his effort in creating the real image dataset during his internship here in Munich.

[5] https://www.hm.edu/allgemein/forschung_entwicklung/forschungsfelder/competence_center/bildverarbeitung/index.de.html

[6] https://www.hm.edu/allgemein/forschung_entwicklung/forschungsfelder/competence_center/bildverarbeitung/partner.de.html

[7] https://www.in.tum.de/cg/

[8] https://openpower.ucc.in.tum.de

REFERENCES

Gardner, M.-A., Sunkavalli, K., Yumer, E., Shen, X., Gambaretto, E., Gagné, C., and Lalonde, J.-F. (2017). Learning to predict indoor illumination from a single image. *ACM Trans. Graph.*, 36(6):176:1–176:14.

Garon, M., Sunkavalli, K., Hadap, S., Carr, N., and Lalonde, J.-F. (2019). Fast spatially-varying indoor lighting estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kajiya, J. T. (1986). The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150.

Kán, P. and Kaufmann, H. (2019). Deeplight: light source estimation for augmented reality using deep learning. *The Visual Computer*, 35(6):873–883.

Karis, B. and Epic Games (2013). Real shading in unreal engine 4. In *SIGGRAPH 2013 Course Notes*.

Laskowski, M. (2007). Detection of light sources in digital photographs. *Central European Seminar on Computer Graphics (CESCG)*.

LeGendre, C., Ma, W., Fyffe, G., Flynn, J., Charbonnel, L., Busch, J., and Debevec, P. E. (2019). Deeplight: Learning illumination for unconstrained mobile mixed reality. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, abs/1904.01175.

Liu, D., Long, C., Zhang, H., Yu, H., Dong, X., and Xiao, C. (2020). ARShadowGAN: Shadow generative adversarial network for augmented reality in single light scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lopez-Moreno, J., Hadap, S., E., R., and Gutierrez, D. (2009). Light source detection in photographs. In *Congreso Espanol de Informatica Grafica (CEIG)*, pages 161–168. Other identifier: 2001321.

Marques, B. A. D., Drumond, R. R., Vasconcelos, C. N., and Clua, E. (2018). Deep light source estimation for mixed reality. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP*, pages 303–311. INSTICC, SciTePress.

Pemasiri, A., Wijebandara, C., Wijayarathna, S., Perera, A., and Gamage, C. (2015). An online lighting model estimation using neural networks for augmented reality in handheld devices. In *2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 4–8.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, volume abs/1409.1556.

Wang, T., Hu, X., Wang, Q., Heng, P.-A., and Fu, C.-W. (2020). Instance shadow detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.