Learning Lightprobes for Mixed Reality Illumination

David Mandl*[1] Kwang Moo Yi[2] Peter Mohr[1] Peter Roth[1] Pascal Fua[2] Vincent Lepetit[3] Dieter Schmalstieg[1] Denis Kalkofen[1]
[1]Graz University of Technology [2]EPFL [3]University of Bordeaux

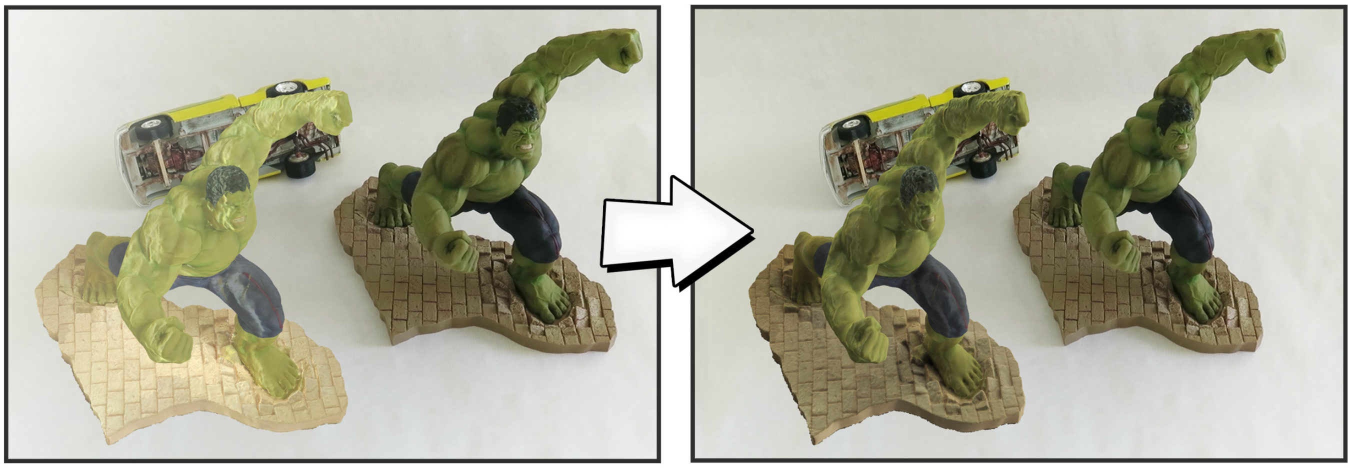


Figure 1: Coherent Illumination. The real scene consists of an action figure of The Hulk and a toy car. To demonstrate the result of our system, we place a 3D scan next to the real action figure and display it using Mixed Reality: (left) The 3D reconstruction is rendered without real world light estimation. (right) Our system estimates the current lighting from a single input image. The estimated lighting is used to illuminate the 3D reconstruction. Note that we only register the real world lighting and do not consider any camera effects such as exposure or blur.

Abstract

This paper presents the first photometric registration pipeline for Mixed Reality based on high quality illumination estimation by convolutional neural network (CNN) methods. For easy adaptation and deployment of the system, we train the CNN using purely synthetic images and apply them to real image data. To keep the pipeline accurate and efficient, we propose to fuse the light estimation results from multiple CNN instances, and we show an approach for caching estimates over time. For optimal performance, we furthermore explore multiple strategies for the CNN training. Experimental results show that the proposed method yields highly accurate estimates for photo-realistic augmentations.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Artificial, augmented,Virtual Realities; I.4.8 [Image Processing and Computer Vision]: Photometric registration—3D Reconstruction

1 Introduction

Mixed Reality (MR) extends the real environment with virtual objects, enabling many powerful applications. Many MR applications, such as home shopping [10], require the rendering of virtual objects into the scene, so that they are indistinguishable from real objects.

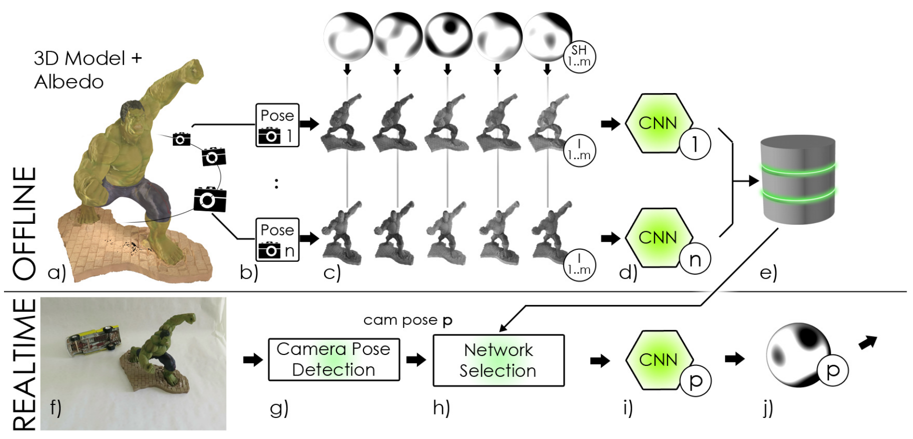
Rendering convincing MR scenes requires photometric registration to ensure that the appearance of virtual objects is rendered consistently with the lighting of the real scene. A key step in doing so is to estimate the incident lighting of a scene. Common approaches rely on active lightprobes, e.g., fisheye cameras strategically placed in the scene [14], or passive lightprobes, like chrome spheres, or other reference objects with known shape and reflectivity [4]. An obvious major downside of having a lightprobe is that it interferes with the user's experience. As an alternative, Gruber et al. [8] demonstrated a probe-less method that estimates the incident light purely from reflections in the unmodified scene, scanned in real-time with a commodity RGB-D sensor. This approach forgoes artificial lightprobes, but suffers from high computational demands and cannot be used on mobile devices [7].

We propose *learned lightprobes* as an alternative to the probeless method of Gruber et al. [8]. Like the probeless method, learned lightprobes work in an unmodified scene, providing unobtrusive user experience, while still retaining the computational benefits of a normal lightprobe. The incident lighting is estimated with a pre-trained convolutional neural network (CNN), which is orders of magnitude cheaper to evaluate at runtime than inverse rendering.

Our method estimates the incident lighting by analyzing the appearance of a known object in the scene, the learned lightprobe. A great advantage of the learned lightprobe is that practically *anything* can be a lightprobe. Application developers are free in their choice of what constitutes a lightprobe. The lightprobe can be an arbitrary object of the scene, or it can be a prop used in the MR application. For example, an MR video game can make use of a live action figure, such as the Hulk in Figure 1, and animate the figure in a closeup view, which is not possible with traditional lightprobes. The only practical requirement is that the lightprobe should have a good distribution of surface normals. If a single lightprobe is found to be insufficient, multiple lightprobes can easily be combined.

During preparation, we first reconstruct the 3D geometry and the material of a real object to be used as lightprobe, using the inverse rendering framework of Richter-Trummer et al. [21]. Second, we use the reconstructed model to train a set of CNN instances cor-

*e-mail: mandl@icg.tugraz.at



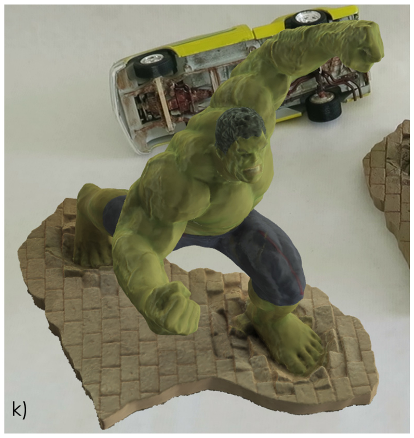


Figure 2: Our system is split into a preparation and an online phase. (a) We reconstruct the geometry and albedo map of a 3D object, (b) which we render from a set of uniformly distributed camera poses. (c) For each pose, we illuminate the model with different spherical harmonics variations (d) and the resulting images are used to train a set of CNN instances. (e) The CNN results for all camera poses are stored in a database. (f) During AR rendering we take a live camera frame, (g) which we use to detect the camera pose. (h) Subsequently, we use the camera pose to select the CNN which was trained from the most similar pose in the database. (i) Furthermore, we input the camera frame into the selected CNN, (j) which outputs estimated Spherical Harmonics. (k) We use the estimated lighting to illuminate augmented 3D objects within the camera frame.

responding to a discrete set of quantized viewing directions. For each of these viewing directions, we synthesize images taken under a large variety of different illumination conditions and use them as training data.

At runtime, we estimate the 3D pose of the live camera to select the corresponding CNN. The camera image is sent through the selected CNN, which outputs the estimate of the current incident illumination. Finally, a differential rendering step inserts virtual objects into the real scene. Optionally, we exploit temporal consistency to improve the results for a scene with constant or only slowly changing lighting: A single view of the lightprobe may not cover a sufficiently large range of surface normals to reliably estimate the entire hemisphere of incident light directions. We overcome this restriction by retaining and merging a history of illumination results from previous frames, as the camera moves to explore the scene.

Our work is the first to estimate incident lighting in real-time from arbitrary scenes, providing the following contributions:

- We present the first end-to-end system for learning and using arbitrary 3D objects to act as lightprobes for coherent illumination of augmentations in AR. Our system is suitable for real-time applications on consumer devices, such as smartphones and tablets.
- We present a novel approach for light estimation based on renderings of a reconstruction under various illuminations.
- We discuss how caching of illumination estimates can be used to reconstruct the current lighting from multiple observations.
- We evaluate three strategies to learn a set of CNN instances across an object's bounding sphere. We compare results from a CNN trained from a single point of view with interpolated results from multiple neighboring CNN instances and CNN instances trained with data from multiple neighboring views.

2 Related Work

A large body of work on coherent rendering for MR exists [22]. Here, we focus on methods that provide an estimate of the current lighting in the user's environment only.

Photometric registration has its origin in computer graphics research. The pioneering work of Debevec et al. [4] introduced the idea of light estimation from a set of 2D photographs of a mirror-coated ball placed inside the scene. Light reflections on passive lightprobes can be observed in the photographs and subsequently used to compute an environment map. Active lightprobes capture the lighting directly using a set of video cameras. They are placed inside the scene in order to capture the entire environment map directly [14]. To observe the entire environment lighting with a minimal number of cameras, active lightprobes commonly make use of fisheye lenses.

While lightprobes deliver high quality environment maps, they are invasive and require preparing the environment. To avoid preparation, some research on light detection from objects naturally occurring in the scene has been done. For example, Pilet et al. [19] propose a method for photometric calibration by reconstructing a lightfield from a planar surface. Similarly, Jachnik et al. [11] capture a lightflight from observations of a planar surface, but they include specular reflections. Specular reflections have also been used to infer lighting on arbitrary objects with known shape [16]. However, this is not always possible, as there is no guarantee that there is always a specular objects in the scene.

Alternatively, one can attempt to estimate illumination from diffuse reflections. Gruber et al. [8] demonstrated that a spherical harmonics (SH) framework is able to recover real-world lighting from an online reconstruction using an RGB-D camera. They solve for incident directional lighting in SH form from selected sample points on the reconstructed surfaces. However, the sample points must have a suitable distribution of surface normals. Furthermore, solving an inverse light transport problem is computationally expensive. Even with image-space optimizations [9], the system requires a desktop GPU to estimate the lighting at approximately 20 frames per second, while the sparse sampling imposes restrictions on the visual quality.

The approach of Knorr and Kurz [15] estimates incident light from human faces using machine learning from a variety of faces under different illumination conditions. The method applies a face tracker and matches distinctive observation points of the detected face to the training database to estimate the real-world light. However, the accuracy of the face tracking is a limiting factor on the quality of the recovered illumination. Moreover, the method is limited to applications where only frontal face views are available.

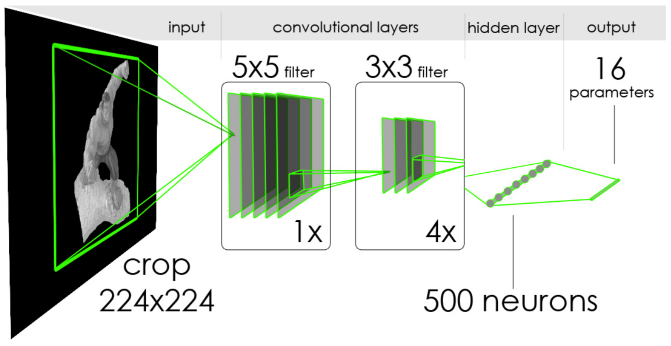


Figure 3: The CNN input are images with 224x224 pixels. The input images pass through five convolution layers, one hidden layer with 500 neurons and arrive at an output layer, which combines the result to 16 SH coefficients used to render the input image.

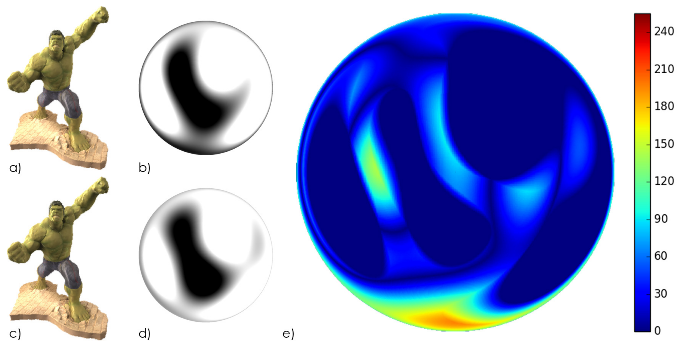


Figure 4: (a) Input image, (b) illumination corresponding to the input image, (c) synthetic image, (d) illumination estimated from input image, which was used to render synthetic image, (e) color-coded per-pixel difference between the illuminations.

3 Method

Our system (Figure 2) consists of an offline preparation phase, in which the appearance of a lightprobe object in synthetic images (Section 3.1) is used to train a CNN to recognize the current incident illumination (Section 3.2), followed by an online phase estimating the lighting in real time for arbitrary scenes containing the lightprobe (Section 3.3).

3.1 Synthesis of training images

A key aspect for CNN training is rich training data. Especially, given the fact that we want our system to be adaptable for all possible camera poses, this becomes even more important. Using image synthesis to supply sizeable amounts of training data without additional cost in human labor is an obvious approach, but often suffers from the limitations of image synthesis to faithfully reproduce real-world phenomena, such as noise. Our method does not suffer from such limitations. We first reconstruct real-world exemplars and then synthesize many variations of the exemplar's appearance. Since we use bandwidth-limited lighting variations based on spherical harmonics [6] (SH), our process is largely free of the problems that plague previous training-by-synthesis approaches.

We use an SH representation with four bands. This SH representation consists of 16 coefficients, $\mathbf{l}_j = [C_0, \cdots, C_{15}]^{\mathrm{T}}$ in the range $[-1, 1]$, which we found very suitable as a low-dimensional learning target for a CNN, at the same time being powerful enough to provide a detailed light setup. We empirically found a space of 2^{15} illumination variations sufficient, which is created by setting the ambient light $C_0 = 1$ and quantizing the higher-order SH coefficients C_1 to C_{15} with two different values in the range of -1 to 1. Note, selecting only two different values for each coefficient keeps the size of the database small enough for training the CNNs in an acceptable amount of time. Since only a subset of possible light conditions is used for training, the detection of very different light settings might suffer from missing training data. Though we found the size of our database to be sufficient we can easily increase it by simply making the sampling interval for the SH values finer. Therefore, in a future work we will compare results from learning different numbers of SH bands and variations of patterns for selecting values for the SH coefficients.

To build a model for generating the synthetic images, we use the approach of Richter-Trummer et al. [21], which jointly derives the geometry of a 3D model and corresponding material properties. Unlike other reconstruction methods designed for digital content studios, this approach requires only a consumer RGB-D sensor and can deliver the material estimates in a matter of minutes on a conventional desktop computer.

To provide the training data, we repeatedly render the reconstructed object, while systematically varying the incident illumination. For each camera pose $\mathbf{P}_i$, we render a set of training images $\mathbf{T}_i$, each with size 224×224. The image size was chosen carefully to match the overall receptive field of the standard six-layer CNN, shown in Figure 3. To cover the entire object in the field of view, we select the camera poses $\mathbf{P}_i$ on a sphere with a diameter of 1.5 times the object diameter, looking towards the object's center. For each image set, we systematically vary the environmental lighting $\mathbf{l}_j$. The resulting database for a camera pose P_i consists of 2^{15} images $\mathbf{T}_{i,j}$.

3.2 Training the CNN

We train one CNN per $\mathbf{P}_i$ directly on the SH coefficients, using $\mathbf{T}_{i,j}$ as training data and $\mathbf{l}_j$ lighting as reference for the corresponding loss function. Note that this approach does not require an intermediate image storage, since only one single synthetic image is considered for training at a time.

Our CNN is implemented on top of the Theano framework [23], using the layout shown in Figure 3. Each image passes through five convolution-and-pooling layers with Rectifide Linear Unit (ReLU) activation. The first layer is using a 5×5 filter kernel, while the remaining four layers use a 3×3 kernel size. After each convolution, we downsample the image to half of its size with max-pooling, i.e. max-pooling with a 2×2 kernel with a stride of 2, which is then followed by ReLU. At the end of the pipeline, we use a standard fully connected layer with 500 neurons and *tanh* activation. We use ReLUs in the convolution layers to enhance the training speed [5], and use *tanh* at the end to keep the output of the network within the range of SH coefficients.

The loss function is a standard mean squared error function, thus solving a linear regression problem. For validation during training, similar to the standard MNIST dataset, [17] we use one sixth of the training data as the validation set to have both rich training and validation data. For validation, we observe the Euclidean distance of the estimate to the ground-truth. We also follow the standard practices when training the CNNs. We use Stochastic Gradient Descent (SGD) and iterate the CNN with a maximum of 200 epochs, using a learning rate $l_r = 0.1$ and an improvement threshold of $t < 0.995$.

3.3 CNN selection and rendering

At runtime, we estimate the environmental lighting for every frame, allowing real-time response to illumination changes. First, the pose of the lightprobe in the image is detected. We use SLAM [3] to retrieve the 3D pose of the MR display. More specifically, our implementation uses PTAMM [2], an open source extension of PTAM [13], which is able to save the point map for re-initialization

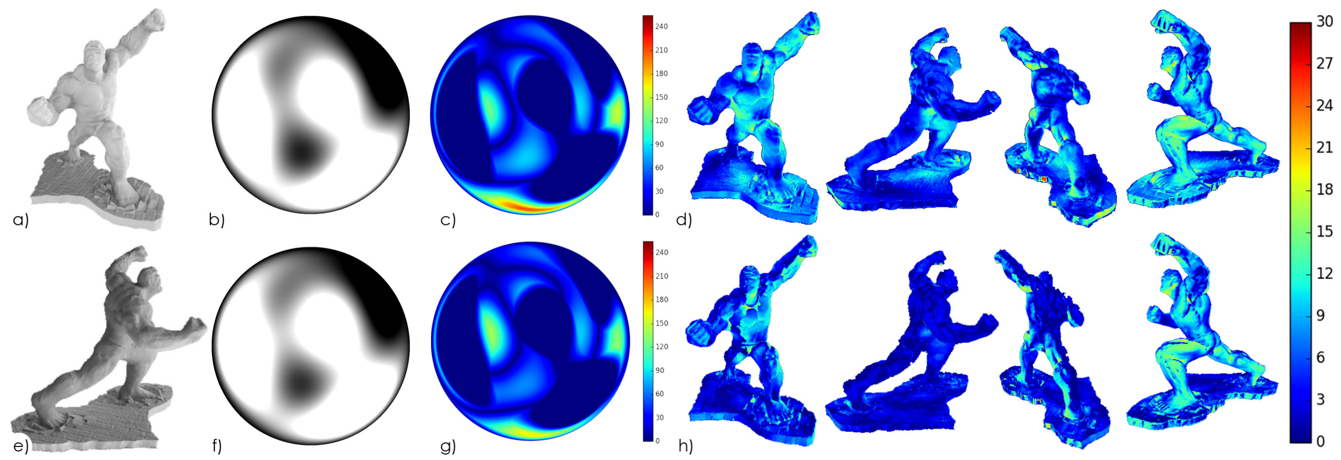


Figure 5: Caching Illumination Estimates. (a) Input image and (b) the estimated lighting. (c) The color-coded difference between the lighting used to generate the image in a) and the estimated lighting. (d) The difference between used and estimated lighting projected to the lightprobe. (e) Another image using the same illumination is used to (f) update the light estimate, resulting in (g) a smaller error, (h) which is illustrated by projection on the lightprobe.

at any later time. The re-initialization allows us to register our lightprobe reconstruction to the map once in order to retrieve the camera pose relative to the reconstruction.

Second, the $\mathbf{P}_i$ closest to the current camera pose is selected. We compare the view vector of the tracked camera with the view vectors of the reference poses, and select one or multiple CNN instances depending on the fusing strategy (Section 5). We mask out pixels that do not correspond to the lightprobe, as indicated by a phantom rendering of the lightprobe [1]. We crop the image to a square area around the target object and scale it to the resolution of 224×224 pixels to match the CNN input size. This image is fed to the CNN, which returns an estimate of the current illumination encoded in SH form. The resulting SH coefficients are used to render virtual objects coherently to real objects in the users environment.

4 Illumination Caching

Figure 4 illustrates the estimated lighting and the remaining errors. In this figure, we compare the input image to a synthetic image rendered from the reconstructed geometry and the lighting estimated by the CNN. Since the difference between the input image and the rendering under the estimated illumination is very subtle, we color-code the pixel differences between the input illumination map and the estimated illumination map with a rainbow color map for error visualization: blue is referring to small error values and yellow to red encode high error values. Note that the high error is concentrated near the border of the illumination sphere. This error refers to directions illuminating the object in the synthetic image from behind, leading to no visible effects in the synthetic image.

Since the CNN is trained with observations of light reflections, it can only estimate those parts of the environmental lighting which caused these reflections. Therefore, certain parts of the environmental lighting do not appear in a single synthetic image, for example the borders of the illumination sphere. Consequently, the CNN is not able to learn these illumination effects.

This implies that the error depends on the geometry of the lightprobe (specifically, on a sufficient distribution of surface normals), and will always include an unmatched region of lighting directions along the negative optical axis. Clearly this is a problem, as for certain visual effects, like shadows, the entire lighting needs to be known.

We overcome this limitation by caching previous light estimations in a conventional environment map and update this map with newer estimations, as they become available. This however is not as trivial as it sounds, because an SH representation uses basis functions with global support (in directional space), and we cannot cache and update individual SH coefficients directly. Instead, we use the reconstructed mesh of the lightprobe to cache illumination information per vertex.

After estimating the lighting from the current camera pose, we illuminate an untextured version of the reconstructed mesh and save the resulting intensity values per vertex. Finally, we iterate over the mesh vertices and project the lighting samples into a SH representation of the environment map. We use a single pass inverse rendering similar to [21] to calculate the values of the SH coefficients from the illuminated vertices. New values projected from the mesh will replace old values in the cache, while vertices not seen in the current camera image remain unchanged.

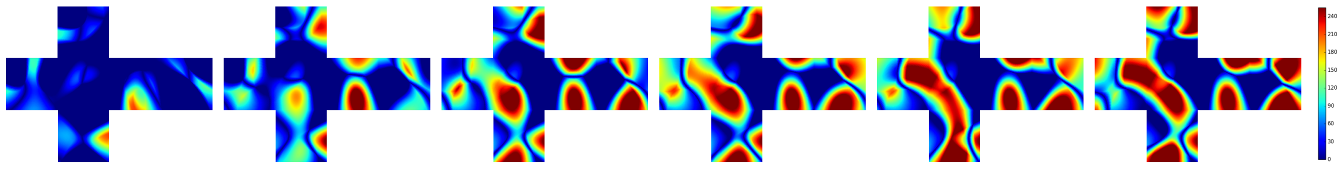
Figure 5 shows the environment lighting estimated from two independent observations. Figure 5(a), (b), (c), and (d) show the estimated illumination and the error from a single point. Figure 5(e), (f), (g) and (h) illustrate the light estimation and the error after the light estimation from the point of view in (e) has been included. Note the lower error in (g) and (h) compared to (c) and (d).

5 Pose Sampling and Interpolation Strategies

Our approach generates a CNN for every single camera pose on the scaled bounding sphere of the object. This requires not only to train many CNN instances, but it also requires to re-initialize the CNN for every new frame. For improved efficiency, we increase the distance between poses used for training the convolutional networks adaptively.

We conducted a series of tests applying a CNN trained to predict lighting for a certain pose, on input images obtained from increasingly different camera poses. The input images are rendered with randomly chosen spherical harmonics coefficients. Figure 6(a) shows the resulting color-coded environment maps. The environment maps are compared per pixel, and the resulting difference is color-coded. We use a blue-red map where blue pixel refers to zero to low error and red pixels encode high error values.

From left to right, we increased the distance from the learned pose in steps of 1°. The picture on the left shows the error for the learned pose. The results indicate a tolerable residual error only for increments of $2-3^\circ$, placing too much demand on storage and runtime.


(a) Deviation by 1° from left to right

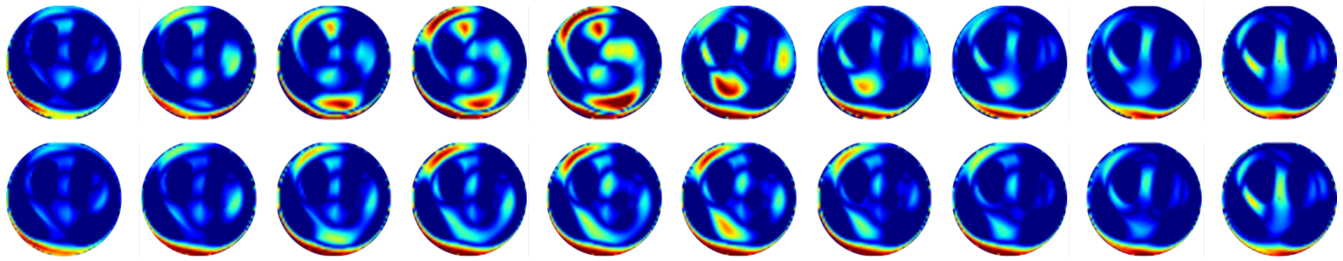

(b) Upper row: nearest CNN is used for light estimation. Lower row: interpolation of results is used for light estimation

Figure 6: (a) Difference between the light estimation received from our system for camera poses with an increased distance to the pose used in CNN training. From left to right, we increase the distance by 1° on the bounding sphere. (b) Comparison of estimates from nearest neighbor selection (upper row) to an interpolation of the results from two neighboring poses (lower row). The images on the left and on the right show the results for trained poses. The poses have been picked with a distance of 10°. The results have been generated from input images which have been captured at increments of 1°. Note the improved interpolation results for locations between trained poses.

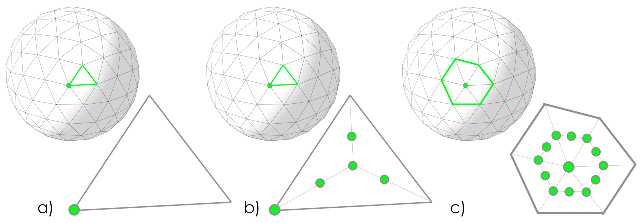


Figure 7: (a) We train a CNN for each sample on a triangulated sphere. (b) To improve results, we augment the training data with images rendered from sample points inside the triangle of the corner point. (c) Data augmentation via disk sampling. To reduce the number of CNN instances, we augment the training data with renderings from camera points which belong to all surrounding triangles.

As a remedy, we exploited the fact that an SH representation allows for linear interpolation. We interpolate the results of neighboring CNN instances enclosing the current pose. As indicated by Figure 6(b), interpolating the results from neighboring CNN instances can improve the estimation and allows to increase the increments between poses used for training CNN instances.

While interpolation can improve the results, a rather high error is still present for camera poses which do not agree with any of the learned poses (middle sphere in Figure 6(b), lower row). It seems likely that the high error at those viewpoints is because of the fact that the system has never seen any rendering from nearby poses. Therefore, to alleviate this problem, when training a CNN for a pose, we augment the training data with images generated from similar poses. In particular, we triangulate the bounding sphere, and we train a CNN at each vertex with images that correspond to the corner and additional images rendered from four more sample positions inside the triangle (see Figure 7(b) for an illustration).

Figure 8(a) shows the difference between the estimates and the ground-truth illumination for nine sampling points inside a triangle which do not coincide with either a vertex or a CNN sample position. The upper row illustrates the error using only the CNN learned for the nearest vertex, and the lower row shows interpolated results from the three vertices of a triangle. Note that the error is very small (except for the red areas at the border corresponding to backside lighting). Furthermore, notice that the error is homogeneous across all samples for both, the nearest neighbor CNN and the interpolated results, in contrast to the errors without data augmentation, shown in Figure 6(b).

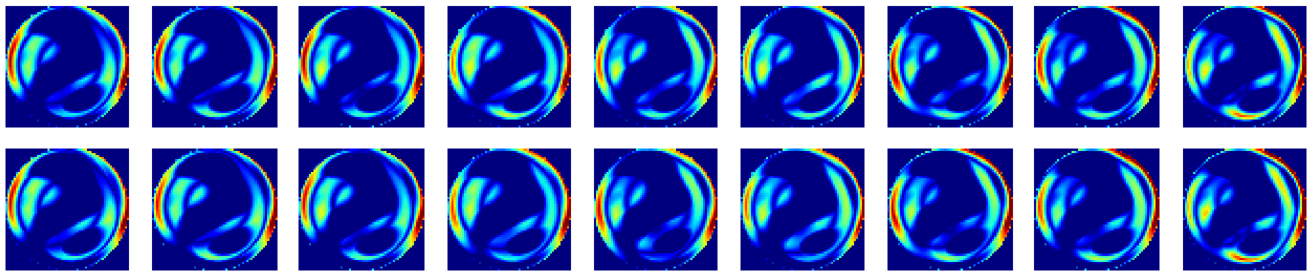
Thus, we conclude that training a CNN with additional images rendered at surrounding poses improves the quality of the estimates, without any noticeable degradation of the estimation for the given pose. However, a naive strategy of simply selecting additional viewpoints from the CNN sampling grid (vertexes in Figure 7) requires training a CNN at each vertex, i.e., six CNN instances at each sample point (one CNN per triangle). To reduce the overall number of CNN instances, we use an alternative strategy for selecting the poses used for data augmentation. Instead of selecting additional sample points, we train a CNN at each vertex with samples from the adjacent triangles. To keep the overall number of images used to train a single CNN manageable, we take only a single sample position inside a triangle in addition to one sample at each edge connected to the vertex of the CNN. Thus, we sample a disk around the corner point (see Figure 7(c) for an illustration).

Figure 8(b) shows the difference between the estimates and the ground-truth using the disk sampling approach. Similar to the triangle sampling, the difference between interpolation and nearest neighbor selection is very small. Comparing the results in Figure 8(a) to Figure 8(b) furthermore reveals that both strategies generate similar results. However, the disk sampling approach requires *six times fewer* CNNs.

6 Visual Quality and Performance

Our approach is very flexible and allows to use any real world object as a lightprobe. To illustrate this, Figure 10 shows a number of further results rendered with light estimates derived from our system on different objects. Note that regardless of the choice of the lightprobe, we get high quality renderings.

Figure 9 visualizes the quality of our results in a real world environment. It shows the difference of the estimated lighting to a direct


(a) Data Augmentation via Triangle Sampling - Distance between Corner Points = 10 Degree

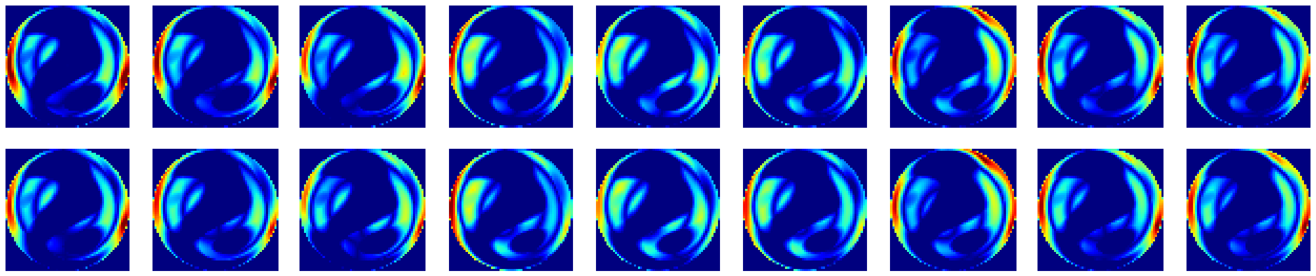

(b) Data Augmentation via Disk Sampling - Distance between Corner Points = 10 Degree

Figure 8: Triangle vs. disk sampling for data augmentation. Upper rows show nearest neighbor selection and lower rows show results from interpolating between corner points. (a) Results obtained from using data augmentation via triangle sampling. (b) Results obtained from using data augmentation via disk sampling. Notice that the difference is not significant, whereas the Disk Sampling requires six times less number of CNNs.

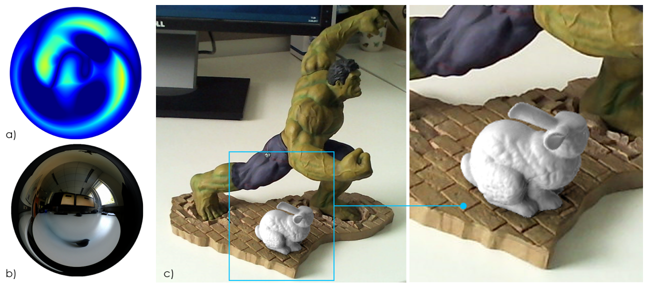


Figure 9: Augmented Reality rendering using our light estimation and the real world object as light probe. (a) Visualization of the difference between our estimated lighting and (b) the current real world lighting, which has been directly captured using a 360 camera. (c) Augmentation of a virtual bunny using the estimated lighting.

capture of the environment map using a 360 panoramic camera (in this example a Samsung 360). The difference is color-coded using a rainbow color map visualization. Notice that the difference between the captured lighting and the one estimated by our system is slightly higher compared to the difference when using synthetic images as input to the system (see Figure 8 and Figure 6). We believe this is caused by the virtual nature of our training samples and the fact that we generate only light combination from two different values in four SH bands. Therefore, we will further investigate more realistic rendering techniques to create training samples as well as other light sampling strategies in order to provide a high range of different lightings and more photo-realistic training data.

As Table 1 shows, the run-time performance of our system is very efficient, achieving realtime. *N.N* refers to nearest neighbor selection and *Interpol.* refers to interpolating the results between the 3 corners of the triangle. Training is performed on a desktop PC using an i7-4770S with 3.1Ghz, 32 GB RAM and a Geforce 980 GTX GPU with 4 GB VRAM. The measurements refer to training a single CNN. A singled trained CNN requires approximately 4MB of memory. We tested the runtime performance of our system on the same desktop PC and a tablet. For the tablet configuration we choose a Razor tablet running Windows 8.1. The tablet is using a i5 1.7Ghz CPU with 4GB of RAM and a mobile GPU, a Geforce GT 640MLE with 2GB VRAM.

Table 1: Performance Measurements

	Training	Estimation PC		Estimation Tablet	
Sampling Strategy		N.N.	Interpol.	N.N.	Interpol.
Single Pose	1h	1 ms	1 ms	42 ms	126 ms
Within Triangle	5h	5 ms	5 ms	42 ms	126 ms
Within Disk	13h	5 ms	5 ms	42 ms	126 ms

7 Conclusion and Future Work

This paper presents the first real-time photometric registration pipeline with low computational demands. A set of CNN instances is trained based on purely synthetic images of an arbitrarily shaped lightprobe and applied in real-time to live video frames. Our experimental results show that the proposed method gives highly accurate estimates and enables photo-realistic rendering. Robustness and efficiency of the basic approach are significantly enhanced by exploiting temporal coherence through illumination caching and by exploiting spatial coherence with sampling and interpolation of camera poses used for training. Furthermore, our experimental results show that with our data augmentation strategy, the scope of permissible poses supported by a given CNN is expanded, ultimately reducing residual errors at the online stage.

Our work provides a flexible tool for photo-realistic rendering in MR applications. In order to further improve its ease of use, we will combine it with an object recognition system to automatically select lightprobes at run-time from a database of prepared objects. We expect that, as this database grows, optimization of the CNN will become more important. Therefore, additional future work will focus on further reducing the number CNN instances required to provide high quality estimates. To this aim, we will investigate jointly trained CNN instances and combinations of multiple lightprobes, exploiting the fact that real-world scenes usually contain multiple objects.

Since more realistic training data will eventually generate better results in real world environments, we will also investigate other global illumination techniques to generate photo-realistic training samples. In addition, we will investigate different strategies to efficiently sample the full range of possible light situations into a small to medium sized database, which can be used to train a CNN in an acceptable amount of time.

Furthermore, we will extend our system to include more visual effects, such as shadow rendering based on point-light detection [20] and camera effects as proposed by Murray and Klein [18] [12]. These features will supplement our work, building a toolbox for photo-realistic MR applications.

ACKNOWLEDGEMENTS

This work was funded in part by the EU FP7-ICT project MAGELLAN under contract 611526 and the Christian Doppler Laboratory for Semantic 3D Computer Vision.

REFERENCES

[1] D. E. Breen, R. T. Whitaker, E. Rose, and M. Tuceryan. Interactive Occlusion and Automatic Object Placement for Augmented Reality. *Computer Graphics Forum*, 1996.

[2] R. O. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proceedings of the 12th IEEE International Symposium on Wearable Computers, Pittsburgh PA, Sept 28 - Oct 1, 2008*, pages 15–22, 2008.

[3] A. J. Davison, W. W. Mayol, and D. W. Murray. Real-time localisation and mapping with wearable active vision. In *Proceedings of the 2Nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 18–, Washington, DC, USA, 2003. IEEE Computer Society.

[4] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 189–198, New York, NY, USA, 1998. ACM.

[5] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, and M. Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[6] R. Green. Spherical harmonic lighting: The gritty details. In *Archives of the Game Developers Conference (Vol. 56, p. 4)*, 2003.

[7] L. Gruber, T. Langlotz, P. Sen, T. Hollerer, and D. Schmalstieg. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *Proceedings of IEEE Virtual Reality 2014 (VR'14)*, Minnesota, MN, USA, March 2014.

[8] L. Gruber, T. Richter-Trummer, and D. Schmalstieg. Real-time photometric registration from arbitrary geometry. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'12)*, pages 119–128. IEEE Computer Society, 2012.

[9] L. Gruber, J. Ventura, and D. Schmalstieg. Image-space illumination for augmented reality in dynamic environments. In *Proceedings of IEEE Virtual Reality (VR'15)*, pages 127 – 134, 2015.

[10] S. Hauswiesner, M. Straka, and G. Reitmayr. Coherent image-based rendering of real-world objects. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 183–190, 2011.

[11] J. Jachnik, R. A. Newcombe, and A. J. Davison. Real-time surface light-field capture for augmentation of planar specular surfaces. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'12)*, pages 91–97, 2012.

[12] G. Klein and D. Murray. Compositing for small cameras. In *Proceedings of Seventh IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'08)*, Cambridge, September 2008.

[13] G. Klein and D. W. Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pages 225–234, 2007.

[14] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential instant radiosity for mixed reality. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'11)*, pages 99–107, 2010.

[15] S. B. Knorr and D. Kurz. Real-time illumination estimation from faces for coherent rendering. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'14)*, pages 113–122, Los Alamitos, CA, USA, 2014. IEEE Computer Society.

[16] P. Lagger and P. Fua. Using specularities to recover multiple light sources in the presence of texture. In *Proceedings of ICPR*, pages 587–590, 2006.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[18] D. W. Murray and G. Klein. Simulating low-cost cameras for augmented reality compositing. *IEEE Transactions on Visualization & Computer Graphics*, 16:369–380, 2009.

[19] J. Pilet, A. Geiger, P. Lagger, V. Lepetit, and P. Fua. An all-in-one solution to geometric and photometric calibration. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'06), pages 69–78, Washington, DC, USA, 2006. IEEE Computer Society.

[20] T. Rhee, L. Petikam, B. Allen, and A. Chalmers. Mr360: Mixed reality rendering for 360 panoramic videos. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1379–1388, April 2017.

[21] T. Richter-Trummer, D. Kalkofen, J. Park, D. Schmalstieg, undefined, undefined, undefined, and undefined. Instant mixed reality lighting from casual scanning. *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'17)*, 00:27–36, 2016.

[22] D. Schmalstieg and T. Hollerer. *Augmented Reality: Principles and Practice*. Addison Wesley Professional, 2015.

[23] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

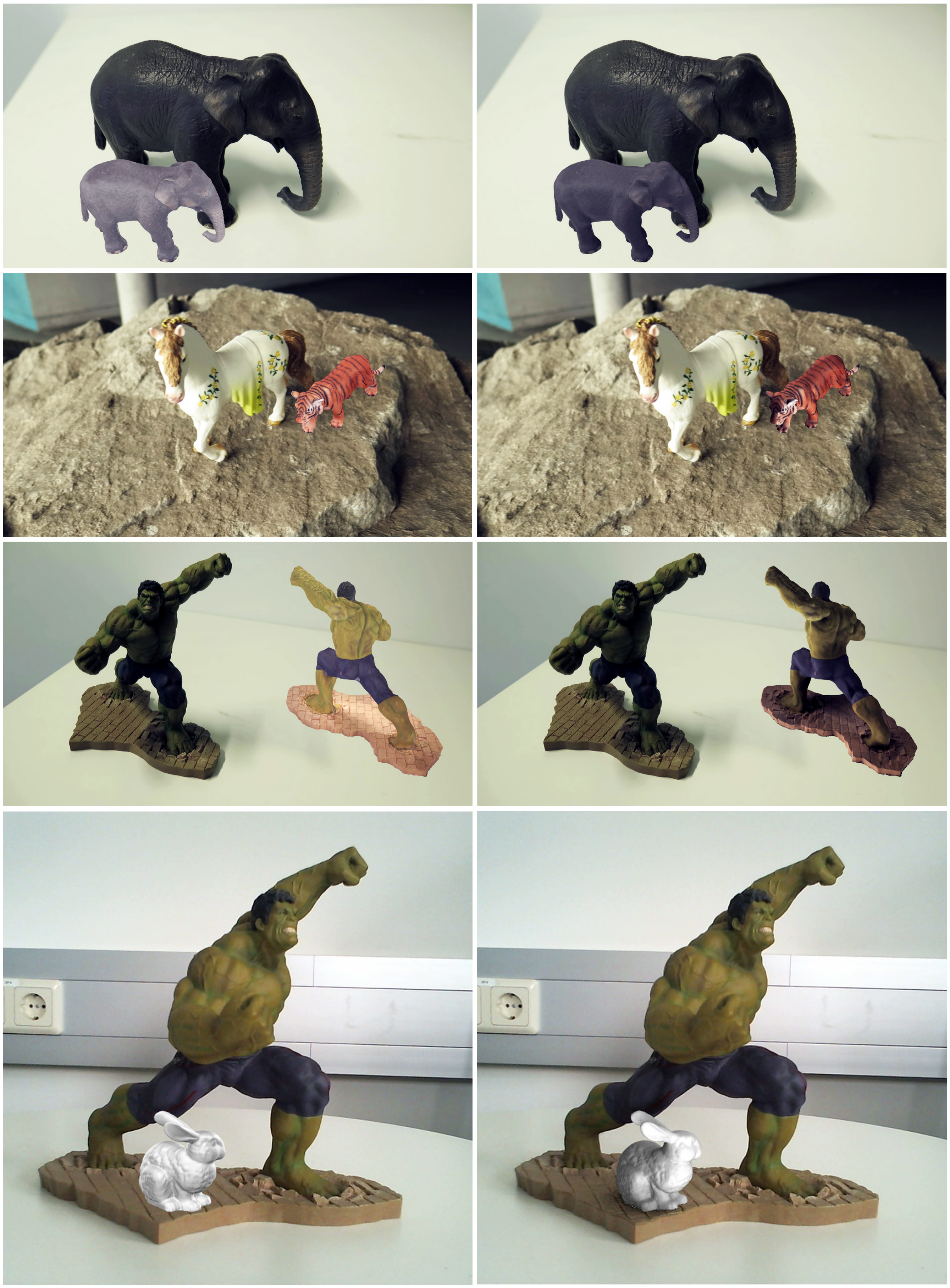


Figure 10: Results. (left) Rendering without light estimation (right) Rendering with light estimation obtained from our system. Virtual objects are the smaller elephant, the tiger, the second The Hulk, and the bunny. Lightprobes are the bigger elephant, the horse and the action figure of The Hulk.