Chair for Computer Aided Medical Procedures & Augmented Reality
Department of Informatics
Technical University of Munich

Light source estimation for photorealistic object rendering

Alexander Epple[1] ✉ and **Pengyuan Wang**[2]

[1]Department of Informatics, Technical University of Munich

[2]Chair for Computer Aided Medical Procedures & Augmented Reality, Technical University of Munich

✉ alexander.epple@tum.de

October 20, 2021

Abstract —
Synthetic data generation is becoming ever more important and prevalent. Having access to large and well annotated datasets is especially relevant for Pose Estimation tasks. Thus, we proposed a novel synthetic dataset generator in a previous work that combines real images with simulated and rendered objects. While the idea showed promise, it was lacking in the department of lighting. To solve this problem, we introduce a light estimation extension to the existing framework, which provides both detected light sources and exposure estimations to the renderer. In addition, we evaluate whether light source estimation contributes valuable information to training data and show how it can improve training.

1 Introduction

Every neural network is only as good as the dataset it is trained on. This is especially true for Convolutional Neural Networks (CNNs), which are used primarily in computer vision, such as in 6D Pose Estimation (6D PE) applications. In a previous work, we addressed the issue of availability of extensive, high quality datasets for 6D PE, by synthetically generating images [6]. This was done by simulating both rigidbody and light transport physics to generate synthetic renders of only those objects, which the 6D PE network is supposed to detect. This simulation is based on real world data and is then combined with real images, to generate data that is as realistic as possible. While some of the generated images did fullfill this criteria, many of them did not due to lack of realistic lighting.

Thus, we propose an extension to the original pipeline, which automatically detects and extracts lighting information from the underlying real world scenes, to enhance the quality of the synthetic renders. With this extracted lighting data, it is possible to make the scenes appear closer to the original conditions when captured, therefore improving overall rendering quality. No additional input data is required, making it possible to use the same scenes and then directly compare CNNs trained with and without proper lighting.

Problem statement. To render realistic images, accurate lighting is crucial. Hence, we have to extract this information from the available data, which consists of reconstructed scenes and low dynamic range (LDR) images. To model the conditions in the scenes from our dataset, which are mostly indoors, both directional light and point light sources have to be detected reliably. To match the light intensities, high dynamic range (HDR) as well as illuminance data is required. Therefore, the light estimation pipeline needs to first acquire HDR information and then detect directional and point lights, which have realistic intensities, colors and positions.

Contribution. We implemented further improvements on a synthetic dataset generator, namely light and exposure estimations. HDR relighting is combined with illuminance calculations to enable accurate light source detection, which can work on most reconstructed scenes automatically. Additionally, we compared how a CNN performs when trained on a dataset with and without light source estimation.

2 Fundamentals

In order to understand the importance of light estimation for synthetic image generation, we first need to understand how pose estimation works and how synthetic images are generated. The following section explains the general concepts and gives a rough overview of these topics.

2.1 Pose Estimation

There have been many attempts to determine the pose, which is the corresponding position and rotation of an object, in three dimensional (3D) space from images alone [14, 21]. Many others have additionally used depth maps to provide even more data to the neural network [26, 17]. This allows for an additional level

of information, namely, the ability to restore positions from a fixed point of view.

There are three main methods for pose estimation: Holistic, keypoint-based and dense methods [19]. While holistic methods such as PoseNet [14] and PoseCNN [27] try to predict object poses directly in one step using techniques such as template matching or CNNs, keypoint-based methods such as BB8 [21] and YOLO9000 [22] approach the problem via correspondence matching. First, keypoints are extracted from an image and then matched with a 3D object. This matching implicitly defines the pose of the object in the scene. There also are dense methods such as PVNet [19] and [13] that rely on pixel or patch based voting schemes.

While not all methods require depth information for training, all of them do need large amounts of annotated RGB images to function. Most training tasks require more than 100k annotated images to work reliably [5]. Annotations usually consist of masks, labels and pose information, which often have to be done manually [5].

2.2 Rendering

In the field of computer graphics, rendering is considered to be the process of generating 2D images from virtual cameras, light sources and 3D objects [2]. While the 3D objects and cameras are available to our generation pipeline, light sources are not. However, lights do play a critical role in physically based rendering (PBR), as they provide direct and indirect illumination [4]. While indirect illumination is radiance that has been reflected or refracted multiple times before it reaches an object in question, direct illumination stems directly from a light source.

Local illumination typically dominates the overall appearance of objects in indoor scenes, while indirect illumination provides a more subtle and nuanced refinement. This is due to local light sources being both spatially-varying throughout the scene, instead of distant, and in close proximity to the objects in the scene. Hence, indoor scenes require local light sources to be represented and as close to the original as possible in order to look realistic. Annotating lights in scenes is therefore difficult and often requires extensive manual input [7].

PBR often relies on path-tracing for rendering. This method is a Monte-Carlo based ray-tracing algorithm, that accumulates and averages many light paths for each pixel of the final image [2]. Most commercially available renderers, such as Blenders Cycles and Disney Pixars RenderMan, rely on path-tracing for rendering, as it strikes a good balance between rendering speed and quality. In our approach, we also use a physically-based renderer, Appleseed, to generate our datasets.

3 Related Work

While synthetic dataset generators are on the rise, there are still not many that aim to be photo realistic. Most notably, BlenderProc [5] by Denninger *et al.* has been used to generate large datasets to be used in the BOP Challenge 2020 [3]. This rendering pipeline, similar to our own, is built on top of Blender and uses PBR and path-tracing. Their method is fully synthetic and renders objects in scenes from the SUNCG dataset. In their paper, they also point out the importance and advantages of photo realistic images [5].

In the work of Hodan *et al.*, they use Autodesks Arnold to render fully synthetic training data. This renderer also features PBR and path-tracing and the six scenes that they used were a mix of scans and hand-crafted models [11]. While Tremblay *et al.* simulate rigidbody physics for realistic object placements similarly to our approach, they don't use a PBR based rendering pipeline [25].

When it comes to indoor light estimation, Zhang *et al.* propose an approach to recover HDR information from recreated real world scenes. They then use this data to recover various light sources and for the inverse rendering of the scene. While their method does produce good results, it requires both manual input and many assumptions to be true in order to work [28]. Gardner *et al.* on the other hand, use deep learning to infer HDR illumination from a single LDR image. They train their network on LDR panoramas in order to predict light directions and then fine tune it on HDR panoramas to improve light intensity estimations [8]. In a later work, they train a deep neural network to predict parametric light sources from a single LDR image instead. To train their model, they propose an algorithm to extract ground truth 3D light sources from HDR panoramas [7].

4 Approach

The light estimation pipeline and its integration into the generator will be discussed in detail. First, we relight the scene mesh in HDR, ensuring the scene

is lit evenly, linearly and as close to the conditions when captured. Next, illuminance is calculated for a rendered scene panorama to set the initial intensities. From here, one directional light source and multiple point light sources are detected and optimized.

4.1 HDR Relighting

In order to estimate light sources, we first need HDR information. Since the 3RScan dataset [12] we use does not provide HDR meshes or textures, this step has to be done by the estimator. We closely follow the HDR mesh recovery as described in a paper by Zhang *et al.* [28]. Their method is based on the assumption, that each pixel of each scene mesh vertex have to represent the same radiance b_i, which turns the corresponding frame exposure t_j and pixel radiance pairs X_{ij} into the solvable optimization problem 1.

$$\min_{t_j, b_i} \sum_{i,j} (t_j b_i - X_{ij})^2 \tag{1}$$

After inverse gamma correcting all available images, they are then projected onto the scene mesh. This is done by reprojecting every mesh vertex for every capture, and storing the corresponding pixel value if the vertex is visible. Then, it is possible to perform the aforementioned minimization using the list of found radiance values per vertex. The reprojection is done in a compute shader, while the minimization uses Ceres Solver [1] as described in the paper.

The minimization differs slightly from the implementation in the paper, whereas they use the radiance values directly and have three seperate values for exposure, we opted for relative luminance and only one single exposure value instead. The resulting per-frame exposure values are also stored in a JSON file to be used during rendering later on.

Next, the exposure corrected radiance $\frac{X_{ij}}{t_j}$ is distributed onto the mesh vertices using the geometric weighting function 2, which favors head-on vertex values and those closer to the surface. Additionally, a confidence term is employed to ensure over- and underexposed pixels do not skew the result, as they are less likely to be accurate.

$$g_{ij} = \frac{(-\hat{v_{ij}} \cdot n_i)(\hat{v_{ij}} \cdot o_j)}{\left\| v_{ij} \right\|^2} \tag{2}$$

For the final vertex radiance values, we sum the weighted samples and the weights, dividing the two to get a weighted, averaged sample. The radiance reprojection is again performed in a compute shader, which speeds up the HDR relighting process drastically, completing within a minute for most cases.

4.2 Light Source Detection

For the light source detection, we also follow an established method as described in a paper by Gardner *et al.* [7]. In their approach, they extract light sources from HDR panoramas to train a deep learning light estimator. With the previously relit scene mesh now being available, we render a HDR panorama using PlotOptiX [20]. The camera position is chosen from the original capture points, as we assume those poses are guaranteed to not be within any mesh geometry. The pose closest to the center of the mesh is selected and an equirectangular panorama is rendered, using flat shading with the vertex radiance values and no lighting. This results in a panoramic image close to those available in the Laval Indoor HDR dataset [8].

4.2.1 Illuminance Calculation

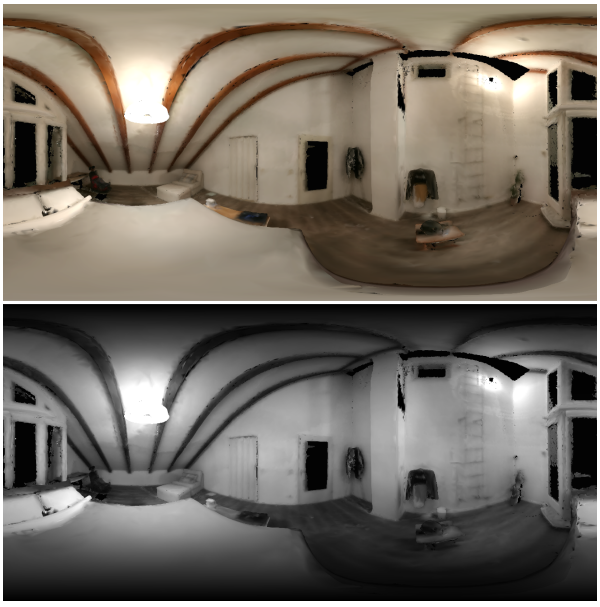

Figure 1 Comparison of the rendered panorama (top) and the calculated illuminance map (bottom).

According to an article by Kodak [15], luminance is the measure of the brightness of a surface, whereas illuminance is a measure for the incident light on an area. Since the goal is to estimate light sources, illuminance is required to get a good result for initial light intensity. To calculate illuminance, we adapt the method described in a paper by Li *et al.* [16] to work with an equirectangular instead of a fisheye projection. To accomplish this, the differential solid angle

$d\Omega$ is calculated as a latitude-longitude rectangle [23]. This is possible, because the globe is also projected equirectangular and is thus identical to the panoramic projection. For each pixel, the edges at the four cardinal directions are calculated, converted to longitude and latitude angles ϕ & θ and finally used to acquire $d\Omega$. Then, the per pixel illuminance E_i can be calculated from the relative pixel luminance L_i as seen in equation 3. It should also be noted, that we do not need to cosine correct and hence do not have to integrate.

$$E_i = L_i d\Omega = L_i(\sin\phi_N - \sin\phi_S)(\theta_E - \theta_W) \quad (3)$$

With an illuminance map being available, light source detection can now be performed. Figure 1 gives an example for the resulting illuminance map. Some key differences are that the light source is more prominent and the top and bottom edges are less pronounced, as they represent less solid angle in the panorama.

4.2.2 Directional Light

Figure 2 Directional radiance map (top) and resulting light in Blender, with highlighted light direction (bottom).

Light source detection is done in two steps, starting with directional light. We assume, that indoor scenes usually have none or only one directional light source, namely the sun. Another assumption is, that sunlight enters the indoor scene through a window, which is most of the times missing in the scene mesh. This is the case, because the reconstruction algorithm has no depth data available in those areas, since the glass is invisible to the depth sensor. When parsing the scene data, each depth map is checked for empty areas, which are subsequently marked.

During the sun detection step, all images are consumed by yet another compute shader, which first calculates the vector from the camera to the pixel in world space. If the pixel has no depth associated with it, this vector is interpreted as a direction where sunlight is potentially shining into the room. This world space direction is used to index a pixel in panoramic image. The original pixels' radiance is exposure corrected and stored in this output panorama at the previously calculated index. Illuminance is calculated as well, as shown in the previous section and also stored in a separate output panorama. If multiple pixels have the same index in the panorama, the radiance and illuminance values are added and a per-pixel counter is increased. Finally, the output radiance and illuminance panoramas are averaged by utilising these per-pixel counters.

After acquiring those directional radiance and illuminance maps, the sun direction can be estimated. We assume, that the overall brightest spot in the illuminance map is the sun or main light source in the scene. Using the radiance and illuminance maps, both color and intensity can be estimated. This step is identical to the point light estimation and will thus be described in more detail in the following section. The detected direction and light properties are stored in a JSON file for later use in the rendering pipeline.

Image 2 shows an example of a directional radiance map and the resulting directional light set up in Blender.

4.2.3 Point Lights

Point light detection is performed directly on the illuminance, radiance and depth maps rendered previously. We closely follow the algorithm described by Gardner *et al.* [7] to generate their panoramic indoor lighthing dataset. First, the brightest spot in the illuminance map, which has been blurred to remove outliers, is extracted. Next, a flood fill algorithm is used to find the connected area, where the the intensity is above 40% of this peak. The total intensity is calculated by summing up the pixels and converted to exposure value (EV), which is commonly used to describe light intensity. This step is necessary, as Appleseed's lights use this parameter instead of photometric units. In case the total light intensity is below 80% of the overall peak, the light detection ends. Otherwise the detected

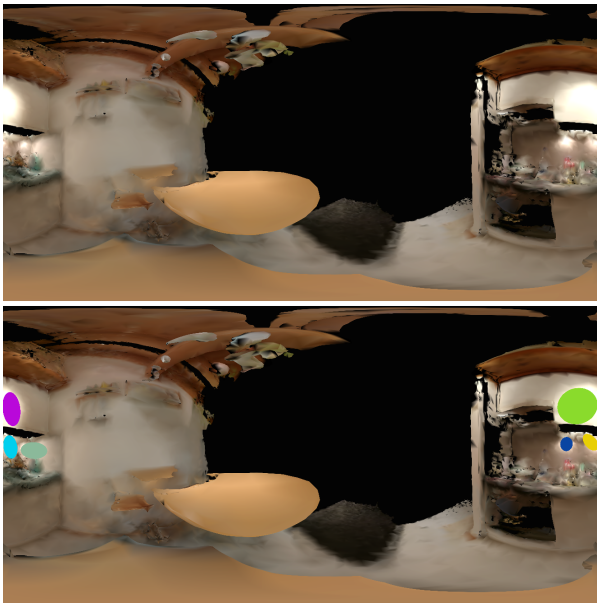

Figure 3 The rendered HDR panorama (top) and the detected point light sources (bottom).

Figure 4 Comparison of the rendered panorama (top) and the detected, rendered SG light only (bottom).

light is processed and masked out. The algorithm will then proceed with the new brightest spot later on.

To process the detected light, an ellipse is fitted around the detected pixels and used to approximate the solid angle of the corresponding light source. The extrema points of the ellipse are calculated using its position and axes, and the angle between the closest and furthest points is determined and averaged. Interpreting the resulting angle θ as the opening angle of a cone, we can calculate the spherical cap of the cone and thus its solid angle $\Omega = 4\pi \sin \frac{\theta}{2}$ [24].

The light color is determined by averaging the corresponding pixel radiance values and normalizing the result using the previously calculated EV. The position can be calculated by averaging the depth values of the pixels, finding the center of mass of all and thus, the average direction and multiplying the two. This, in addition to the camera position, which was used to render the panoramas, gives the lights position in world coordinates. An example for detected light sources can be seen in figure 3, the colored ellipses are the ones fitted around the pixels associated with the light source.

In order to improve the calculated light intensity, each detected light is optimized individually. This is done by solving the minimization problem 4, with r_i being the calculated radiance, R_i being the observed radiance of pixel i from the rendered panorama, and p being the peak of the detected light. The weighting term ensures that pixels more likely to be part of the light source are more important than those near the threshold.

$$\min_{r_i} \sum_i ((r_i - R_i) \cdot \frac{r_i}{p})^2 \tag{4}$$

$$r_i = G_i(v, \mu, \lambda, \alpha) = \alpha e^{\lambda(\mu \cdot v - 1)} \tag{5}$$

The radiance r_i is calculated using spherical gaussians (SG) [18] as seen in equation 5. The amplitude α is the detected light color multiplied by its EV, the sharpness λ being the solid angle of the fitted ellipse. Finally, the axis μ is the direction from the camera to the light source. Spherical gaussian lights are a quick and good way of simulate point light sources, thus making them the obvious choice. See image 4 for an example of a SG light as it is used during optimization.

The minimization is done using the Ceres Solver [1] as well, with everything but the amplitude α being kept constant. While many initial intensities are already quite good, this optimization does help the light sources match the corresponding areas of the relit scene even more closely. If the converted EV after the optimization is above zero, the light and its parameters are output to the JSON file as well. This marks the end of the light source detection.

Figure 5 draws a comparison between a real and rendered image. While the estimated lighting is, of course, not perfect, it is quite comparable everywhere except for the corner of the room.

Figure 5 Comparison of a real image (left) and a recreation using the detected light sources (right).

4.3 Rendering

Figure 6 Two successive frames with differing exposure. The image on the left is overexposed, which is reflected in the rendered drills.

In the extended rendering pipeline, everything occurs in the same order and way as described in the previous work [6]. The only differences are, that the light estimator is run before processing the scene, as well as using the detected instead of default lights, if available. The per-frame exposure is also used during post processing of the rendered objects, this ensures that they are not over- or underexposed in comparison to the real image. Image 6 shows two successive frames with differing exposures and its effect on rendering.

5 Evaluation

To evaluate the light estimation extension of the dataset generator, we perform a qualitative and quantitative evaluation on a dataset consisting of 10k images. We compare our improved synthetic images to state of the art synthetic generators introduced in 3 as our qualitative analysis. For the quantitative evaluation, Mask R-CNN [9] is trained on both the old and new 10k datasets and then evaluated on the same test sets from LineMOD [10].

5.1 Quantitative Evaluation

To evaluate the extended generator quantitatively, we compare Mask R-CNN [9] models trained on corresponding datasets of the same size and at a total of 6k steps each. This ensures, that the only difference between the performance of the two models is the additional use of light source estimation during training data generation. Each model is tested on LineMod [10] test sets and the F1 and IoU scores are determined.

With the datasets being rather small, the results are split into total (TO) and detected-only (DO), since the networks often do not detect anything due to insufficient training. While more training data would be necessary to compare our method properly to other datasets, we can still show how our additions impacted performance.

-	New		Old	
Object	F1	IoU	F1	IoU
Benchvise	0.53	0.63	0.49	0.62
Phone	0.12	0.19	0.10	0.15
Drill	0.16	0.20	0.06	0.07
Mean	0.27	0.34	0.22	0.28

Table 1 Total F1 and IoU scores for models trained on the new, old and combined datasets.

-	New		Old	
Object	F1	IoU	F1	IoU
Benchvise	0.64	0.69	0.63	0.67
Phone	0.55	0.60	0.62	0.64
Drill	0.64	0.62	0.63	0.56
Mean	0.61	0.64	0.63	0.62

Table 2 F1 and IoU scores of models trained on the new and old datasets when considering only those images where something was detected.

While the differences between the old and new datasets are small, they are also apparent. Overall

Figure 7 Examples from our dataset, BlenderProc [5] and from Hodan *et al.* [11] (left to right).

Figure 8 Examples from our old dataset, without light estimation.

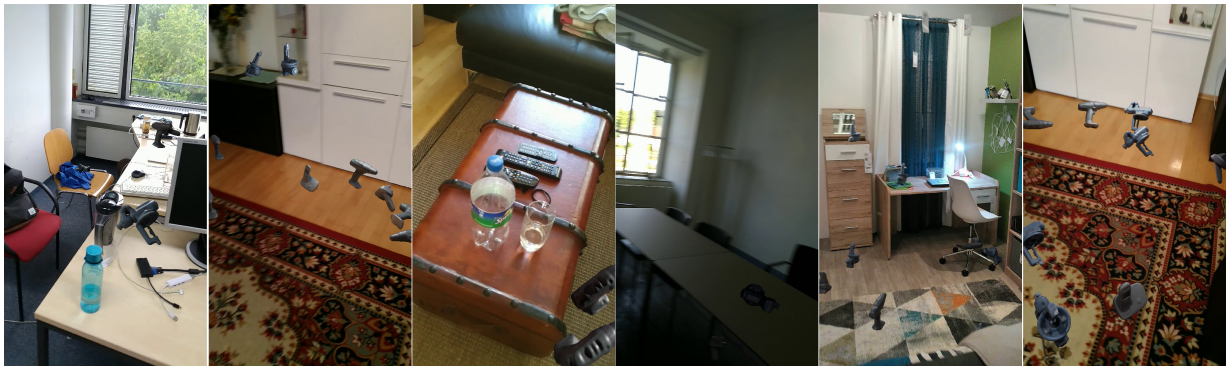

Figure 9 Examples from our new dataset, with light estimation.

performance is improved by 23% for F1 and 21% for IoU scores, as listed in table 1. Performance for test images with detections differs marginally, with F1 and IoU being within 3% respectively, as can be seen in table 2. It seems as though the dataset with light estimation enables the model to converge faster, as overall performance is increased. When it comes to detections alone, the differences are negligible and may also be due to chance.

5.2 Qualitative Evaluation

For the qualitative analysis, we present four images from our own dataset, BlenderProc [5] and Hodan *et al.* in figure 7. While both fully artifical image sets feature shadows and slightly more consistent lighting, our examples are not far behind. Lighting and self shadowing looks mostly realistic and true to the scene, only the lack of contact shadows breaks the illusion. Our approach also has the advantage of having more diverse indoor scenarios in comparison to Hodan *et al.*, as the used scene dataset consists of only six scenes.

Additionally, we also compare our old and new datasets in figure 8 and 9 directly. The images previously had fixed light sources and no added exposure, which sometimes looks acceptable, but in other cases breaks the illusion. The images with both new features are more diverse in shading, both from scene to scene, as well as from frame to frame. The addition of light source and exposure estimation definitely makes a big

difference in terms of making the objects look more realistic and believable.

5.3 Discussion

Our results show, that including light source detection in a mixed real and synthetic image generation pipeline is advantageous. Comparing datasets from our generator with and without light estimation shows that the inclusion of light source estimation enables a CNN to converge faster than without. Qualitatively speaking, the generated images look much more realistic, as appropriate shading, self shadowing and exposure make the images appear more harmonious with the scene than before. Our results are now quite close to those of fully artificial rendering frameworks, often only falling behind due to the lack of shadows.

6 Conclusion & Outlook

In this paper, we described an approach to estimate light sources in reconstructed, LDR indoor scenes. While the implementation is specific to the 3RScan [12] dataset, it can be adapted to work with other datasets as well.

While overall, the addition of realistic lighting seems to have a negligible impact on detection performance for object detection, it does make training converge faster. Additionally, the shading in images is now more diverse and appropriate, making the results look more realistic and believable.

Figure 10 An early look at reflective and transparent materials.

In the future, adding more material properties to the rendered objects might be an interesting way to add even more visual complexity to the training images. Experiments with metallic and transparent surfaces already show a lot of promise but would need more refinement, as can be seen in figure 10.

Finally, some problems described in the original work still persist and also need to be addressed. Scene meshes still need improvement, as the objects often "clip" into existing geometry, due to holes in the reconstructed mesh. Image selection is still a problem. Many real images are blurred and thus should not be included, while others that are not blurry are ignored. It would also be interesting to see if new scenes scanned with both the previously mentioned issues in mind would allow for even better results and especially if it would further improve our light detection.

Switching to a different renderer would also be a large improvement, as render time is far from where it could be. This could make it possible to generate large enough datasets to truly put our rendering pipeline to the test.

References

[1] S. Agarwal, K. Mierle, et al. *Ceres Solver*. (accessed 14.10.2021). URL: http://ceres-solver.org.

[2] T. Akenine-Mller, E. Haines, and N. Hoffman. *Real-Time Rendering, Fourth Edition*. 4th. USA: A. K. Peters, Ltd., 2018. ISBN: 0134997832.

[3] *BOP Challenge 2020 Website*. (accessed 13.10.2021). URL: https://bop.felk.cvut.cz/challenges/bop-challenge-2020/.

[4] P. H. Christensen and W. Jarosz. "The path to path-traced movies". In: *Foundations and Trends® in Computer Graphics and Vision* 10.2 (2016), pp. 103–175.

[5] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. "Blenderproc". In: *arXiv preprint arXiv:1911.01911* (2019).

[6] A. Epple. "Photorealistic Rendering of Training Data for Object Detection and Pose Estimation with a Physics Engine". unpublished. 2020.

[7] M.-A. Gardner, Y. Hold-Geoffroy, K. Sunkavalli, C. Gagné, and J.-F. Lalonde. "Deep parametric indoor lighting estimation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7175–7183.

[8] M.-A. Gardner, K. Sunkavalli, E. Yumer, X. Shen, E. Gambaretto, C. Gagné, and J.-F. Lalonde. "Learning to predict indoor illumination from a single image". In: *arXiv preprint arXiv:1704.00090* (2017).

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[10] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes". In: *Asian conference on computer vision*. Springer. 2012, pp. 548–562.

[11] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha, and B. Guenter. "Photorealistic image synthesis for object instance detection". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 66–70.

[12] A. A. Johanna Wald, F. T. Nassir Navab, and M. Niessner. "RIO: 3D Object Instance Re-Localization in Changing Indoor Environments". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2019.

[13] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation". In: *European conference on computer vision*. Springer. 2016, pp. 205–220.

[14] A. Kendall, M. Grimes, and R. Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.

[15] E. L. Kodak. "Illuminance with Reflection-Type Exposure Meters and an 18% Neutral Test Card". In: *Kodak publication AM-105KIC* (1999).

[16] H. Li, H. Cai, and G. Wang. "Improving High Dynamic Range Image Based Light Measurement". In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2020, pp. 274–279.

[17] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. "Deepim: Deep iterative matching for 6d pose estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 683–698.

[18] *Lighting using Spherical gaussians*. (accessed 16.10.2021). URL: https://therealmjp.github.io/posts/sg-series-part-2-spherical-gaussians-101/.

[19] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. "Pvnet: Pixel-wise voting network for 6dof pose estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4561–4570.

[20] *PlotOptiX Website*. (accessed 14.10.2021). URL: https://plotoptix.rnd.team/home.html.

[21] M. Rad and V. Lepetit. "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3828–3836.

[22] J. Redmon and A. Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[23] *Solid angle of a latitude-longitude rectangle*. (accessed 15.10.2021). URL: https://en.wikipedia.org/wiki/Solid_angle?section=6#Latitude-longitude_rectangle.

[24] *Solid angle of cone*. (accessed 16.10.2021). URL: https://en.wikipedia.org/wiki/Solid_angle#Cone,_spherical_cap,_hemisphere.

[25] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. "Deep object pose estimation for semantic robotic grasping of household objects". In: *arXiv preprint arXiv:1809.10790* (2018).

[26] C. Wang, D. Xu, Y. Zhu, R. Martin-Martin, C. Lu, L. Fei-Fei, and S. Savarese. "Densefusion: 6d object pose estimation by iterative dense fusion". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3343–3352.

[27] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes". In: *arXiv preprint arXiv:1711.00199* (2017).

[28] E. Zhang, M. F. Cohen, and B. Curless. "Emptying, refurnishing, and relighting indoor spaces". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–14.