Optimizing Structured Light for Physical HDRI Reconstruction

Brayden Sue[*] SFU
John Mitton[†] SFU
Rafael Guevara[‡] SFU
Kathy Lee[§] SFU

Figure 1: Physical Light Box, HDRI Environment, Physical Reconstructed HDRI Light

ACM Reference Format:
Brayden Sue, John Mitton, Rafael Guevara, and Kathy Lee. 2025. Optimizing Structured Light for Physical HDRI Reconstruction. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Lighting plays a critical role in accurately representing the appearance of objects and environments, both in physical settings and digital environments. In computer graphics, HDRI (High Dynamic Range Imaging) maps are often used to replicate real-world lighting through spherical environment mapping. However, although the maps are effective and trivial to implement in virtual scenes, it remains a challenge to reproduce these lighting conditions in the physical world in a controlled and consistent way.

This project, Optimizing Structured Light for Physical HDRI Reconstruction, introduces a physical setup designed to recreate HDRI lighting environments using a programmable light box. The light box is built from 2ft x 2ft cardboard-like panels, each with Twinkly controlled LED light strings arranged in a snaking pattern to form a light grid. The programmable LEDs are controlled using a Python API, enabling setup-specific, HDRI-optimized RGB values to be mapped into the real-world lighting environment. To ensure alignment between the virtual and real-world lighting setups, both the camera and LED positions in the virtual setup are calibrated and mapped to the corresponding positions in the physical setup.

Although this system was implemented independently, it is inspired by Ghosh et al. and their paper Practical Multispectral Light Reproduction [1], which explores a comprehensive method in recreating real world photos in a controlled studio environment. Their approach allows highly accurate reproduction of how subjects appear under natural lighting by using a multispectral LED lighting setup and HDRI imagery to closely match the spectral qualities of illumination. Both our approach and Chosh et al's approach have the same overall goal for enabling reproducible lighting conditions that bridge the gap between physical and virtual environments.

2 [PIPELINE, METHOD, FORMULATION ETC.]

2.1 Hardware setup

The physical lighting environment was constructed using low cost, accessible materials and designed to support high spatial resolution and controllable illumination. The main structure consists of four 2x2 ft boards, made of a cardboard-like material that is lightweight yet sturdy enough to support the embedded lighting components. Our light box was constructed with four illuminating faces: top, back, left, and right. The top and back panels had a 13x13 grid of lights while the left and right panels had a 13x10 grid of lights totalling up to a total of 598 lights. Two sets of Twinkly controllable LED string lights to fill all the holes. The boards were all painted black and held together by metal L-brackets. For the bottom face of the setup, black paper was used to minimize unwanted reflection. The camera used for calibration and data capture was a Canon EOS 1200D with the following settings: 400 ISO, 1-second shutter speed, and f/8.5 aperture. A 20x20 cm printed black and white checkerboard pattern was used as our known geometric reference for calibration.

We handled communication with the Twinkly LEDs using the open-source xled library, which connected to the devices over Wi-Fi Direct and allowed us to create custom lighting patterns from our computers. We controlled the camera via the Canon SDK over a

[*]Denotes equal contribution.
[†]Denotes equal contribution.
[‡]Denotes equal contribution.
[§]Denotes equal contribution.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

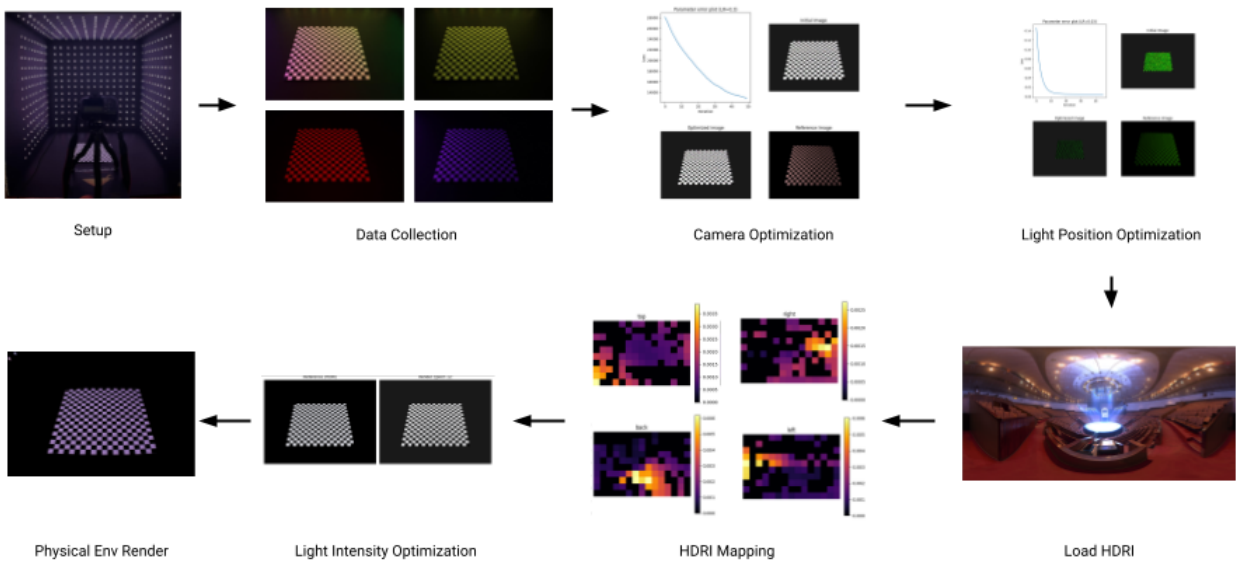

Figure 2: Full pipeline

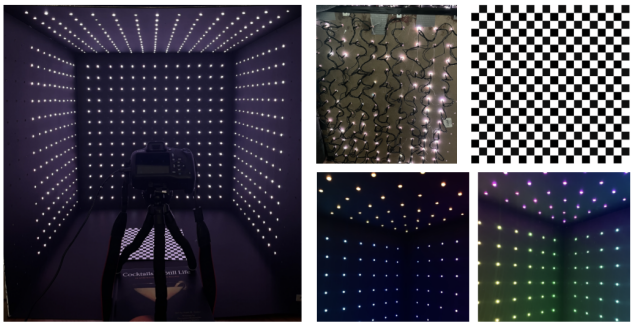

Figure 3: Physical camera and light box setup inside and outside with printed checkerboard pattern used.

USB connection. To improve maintainability and readability, we encapsulated each library in its own controller class—LEDController and CameraController. These classes worked together to synchronize LED changes with image capture, automating the collection of calibration images. For each calibration step, we illuminated one face with a predefined colour pattern, captured an image, and saved the file to a designated calibration folder.

2.2 Software setup

To match the virtual scene to the real-world image, we first obtain a mask of a checkerboard in the real environment by taking the difference between a picture with the object, and a picture without it. This mask will be used for both camera and light optimization to isolate the checkerboard and ensure optimization is performed only on the object in the scene. This picture is then downsampled from 5184×3456 to 600×800 to match the resolution of the virtual renders. While comparison at the camera's resolution is possible, it is computationally expensive without offering noticeable benefits over lower resolutions

2.3 Camera Optimization

Before optimizing the lighting setup, the virtual camera must be accurately aligned with the real-world scene. In this optimization, there are a few assumptions we can make to simplify the computation and allow for faster convergence. For vector translation, we take initial measurements for the camera's position. Any discrepancies between the virtual and real world can be minimized by computing the loss of small displacements around the x, y, and z direction. We iterate through all candidates and pick the translation vector that minimizes the mean squared error between the virtual and real image. Once this is done, we can use Mitsuba to directly optimize the camera's FOV (field of view) by using the Adam optimizer with the same mean squared error formulation for the loss.

2.4 Light Optimization

After finding the camera parameters with the least loss, we can now optimize the cube of lights. To do this, we define the cube using seven parameters: x, y, z translation, roll, pitch, yaw, and uniform scale. These are represented as five latent variables in the optimizer that will be used to build a transformation matrix that can be applied to every emitter to calculate the resulting loss. To help with optimization we choose a set of lighting configurations that will be applied to both the real and virtual world. For each configuration, we use the Adam optimizer to minimize the mean squared error loss between the rendered output and the real-world images by transforming the 598 emitters at each step. We then choose the optimized cube parameters with the lowest loss. Due to differences in the real and virtual world, the losses are quite high,

but they will converge into an optimal solution that can then be passed into the HDRI mapping modules.

2.5 HDRI Mapping and Optimization

In order for the light box to properly replicate HDRI lighting, we must approximate the environment map illumination and represent it using a finite number of fixed real-world emitters. This involves converting the HDRI into directional data using spherical coordinates, calculating pixel luminance, and clustering the data using the *KMeans* algorithm to determine dominant lighting directions. Each cluster is then assigned to the nearest physical light sources with respect to the scene's calibration using a *cKDTree* to query the nearest neighbors. The resulting intensities are tone-mapped, averaged per emitter, and scaled to generate the initial values for optimization.

The HDRI-mapped emitter intensities are then loaded as initial values for the final optimization step. The goal of this step is to minimize the *Mean Squared Error (MSE)* loss between 3D environments, in which the reference scene is lit using the *HDRI-to-point-light* mapping, while the target scene uses global HDRI environment lighting. Both scenes share the same object and camera configuration, calibrated with the optimized position and field-of-view parameters, as shown in Figure 4. Then, an *Adam* optimizer is used to learn a *Mitsuba Color3f* object and modify the intensities of RGB values for each emitter in the virtual light box. The emitter intensities are optimized quickly over an average of 10 epochs with early stopping. The low final loss values displayed in Figure 4 indicate an effective learning rate and the proper application of calibration parameters. Finally, the indexed and optimized RGB values for each emitter are written to a JSON file for direct mapping to the physical lighting setup.

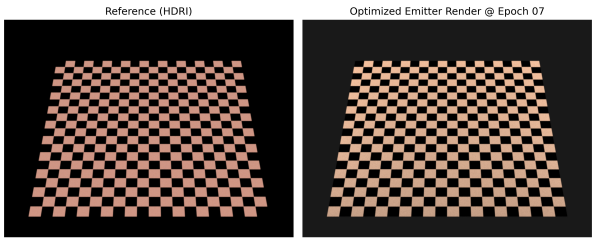

Figure 4: Optimized Intensity

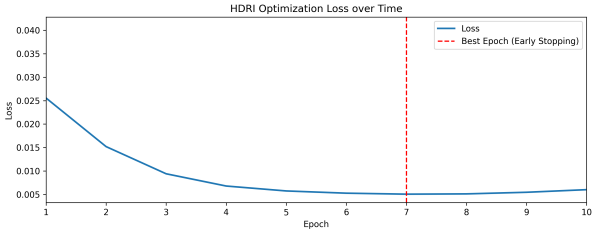

Figure 5: Intensity Optimization Loss

2.6 Final Pipeline

The pipeline starts by loading the specified HDRI. If it is the first time setting up or if any of the physical components have been repositioned, a calibration step is performed. This involves capturing calibration images, optimizing the virtual camera partners, and estimating the transformation (translation, rotation, and scale) between the virtual and real world light box. Once calibration is complete, the HDRI is clustered into dominant lighting directions, mapped to the nearest emitters, and optimized for light intensity to further minimize the loss. The lighting configuration is then sent to the physical setup.

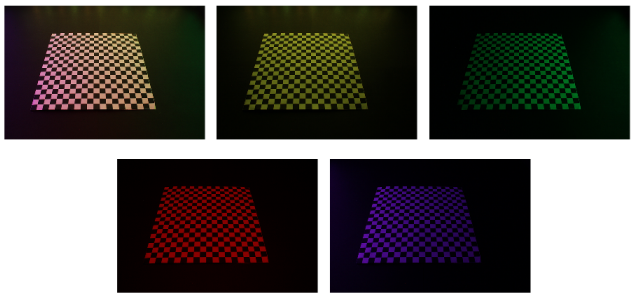

Figure 6: Final Results with HDRI Map

3 RESULTS

The results of our pipeline, Figure 7-14, show the masked outputs of each HDRI environment mapped to our physical environment with objects of known and unknown geometry. For our calibration object, a checkerboard, the lighting and masks are accurate and leave little to be desired. With objects of unknown geometry, gnomes, the lighting is accurate however, the mask struggles in areas with shadows as a higher threshold will mask parts of the gnomes.

4 CONCLUSION

This project presents a low cost and programmable approach to physically reproducing HDRI lighting using a custom LED light box. By combining camera calibration, light mapping, and intensity optimization, the system can closely approximate lighting environments in the real world.

Some future directions include improved camera position estimation, modeling reflectance, and a spherical physical dome rather than a cube. Mitsuba currently does not offer direct optimization over all camera parameters so the camera's translation needs good initial measurements to converge on a solution. Exploring ways to minimize the need for measurements is a good future direction. Modeling the reflections in Mitsuba will lead to lower errors between the real and virtual scene allowing for faster convergence and more accurate results. To allow a more accurate and precise replication of the HDRI, a spherical dome with an increased number of emitters would provide a more uniform light distribution.

REFERENCES

[1] Chloe LeGendre, Xueming Yu, Dai Liu, Jay Busch, Andrew Jones, Sumanta Pattanaik, and Paul Debevec. 2016. Practical multispectral lighting reproduction. ACM Trans. Graph. 35, 4, Article 32 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925934

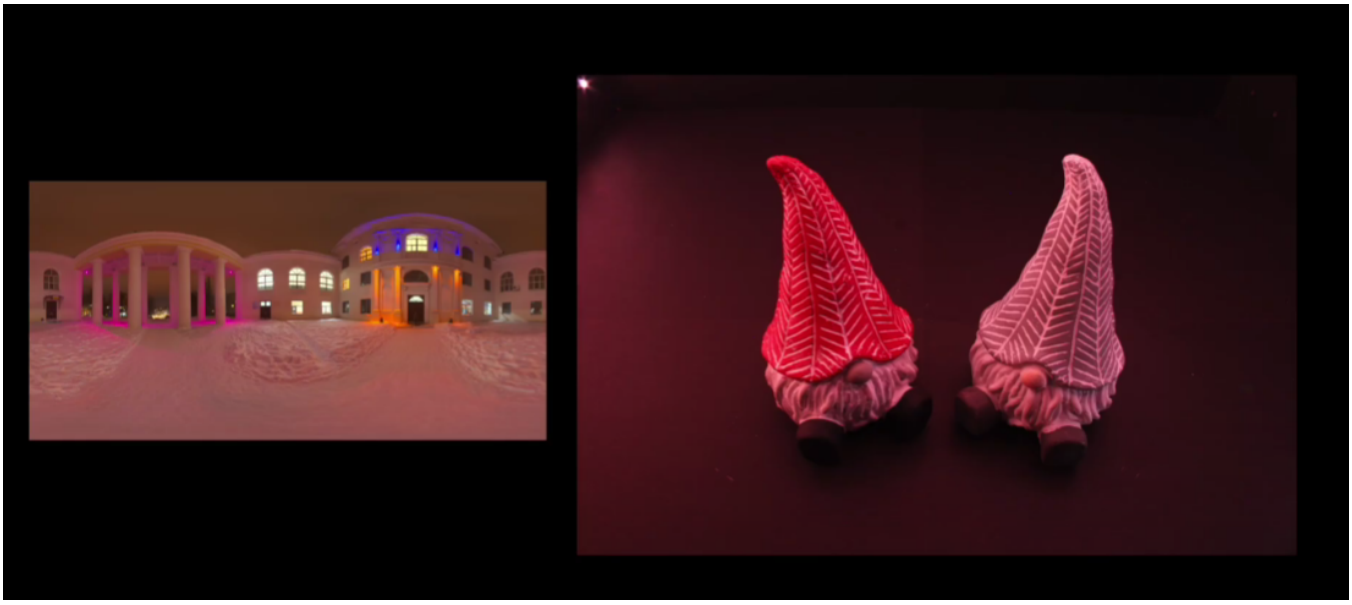

Figure 7

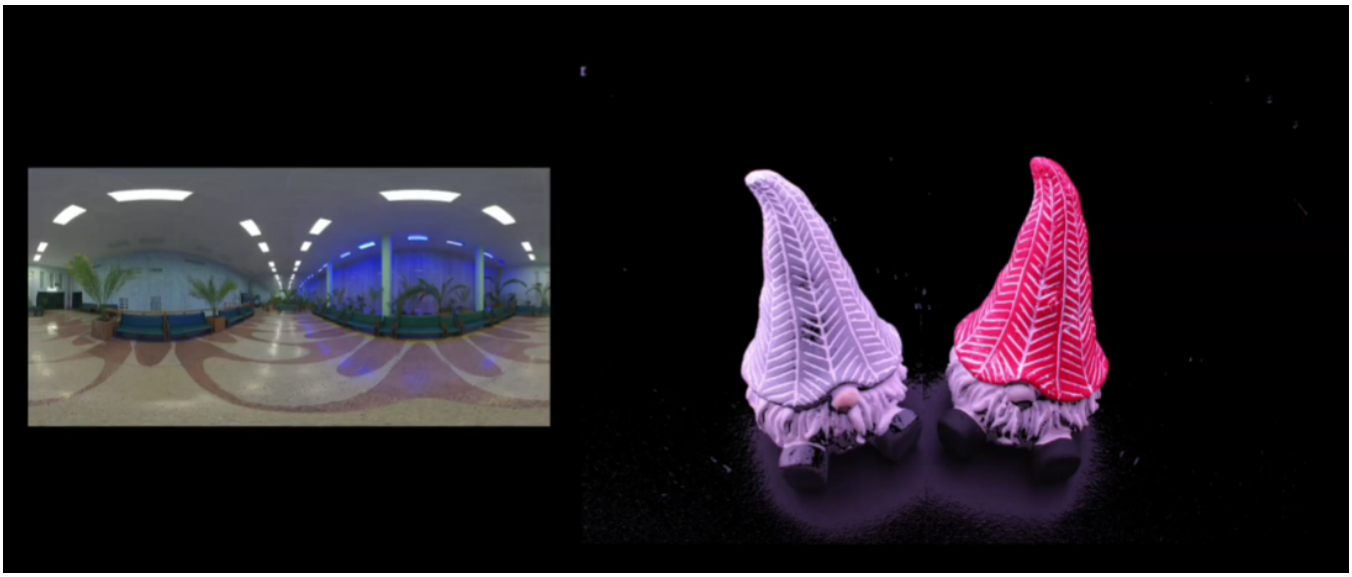

Figure 8

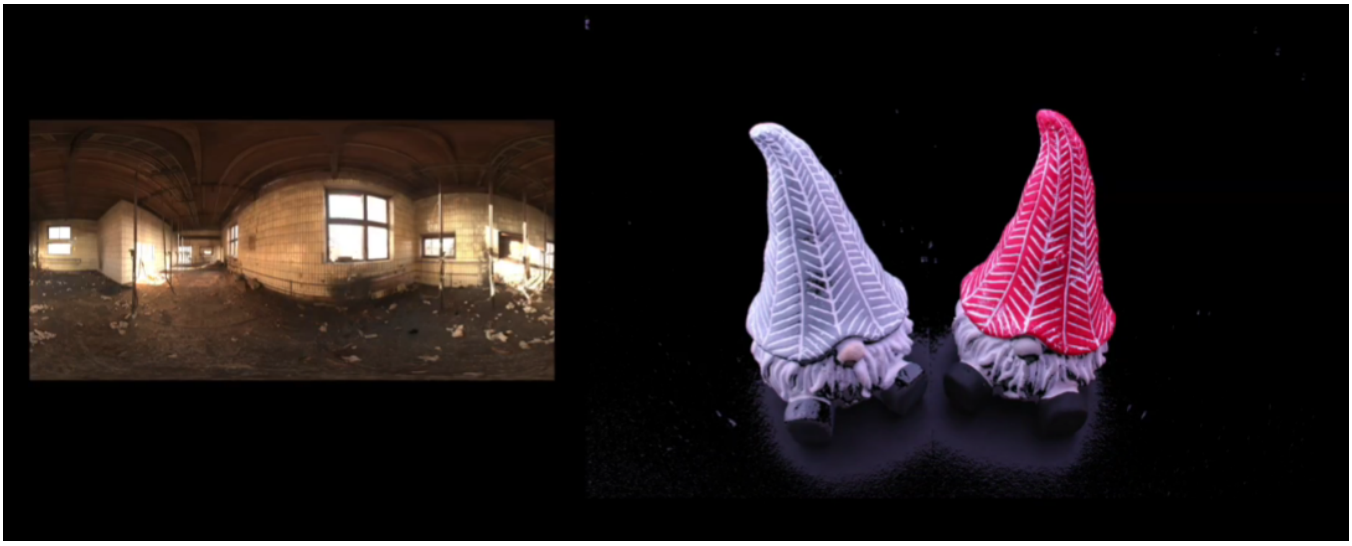

Figure 9

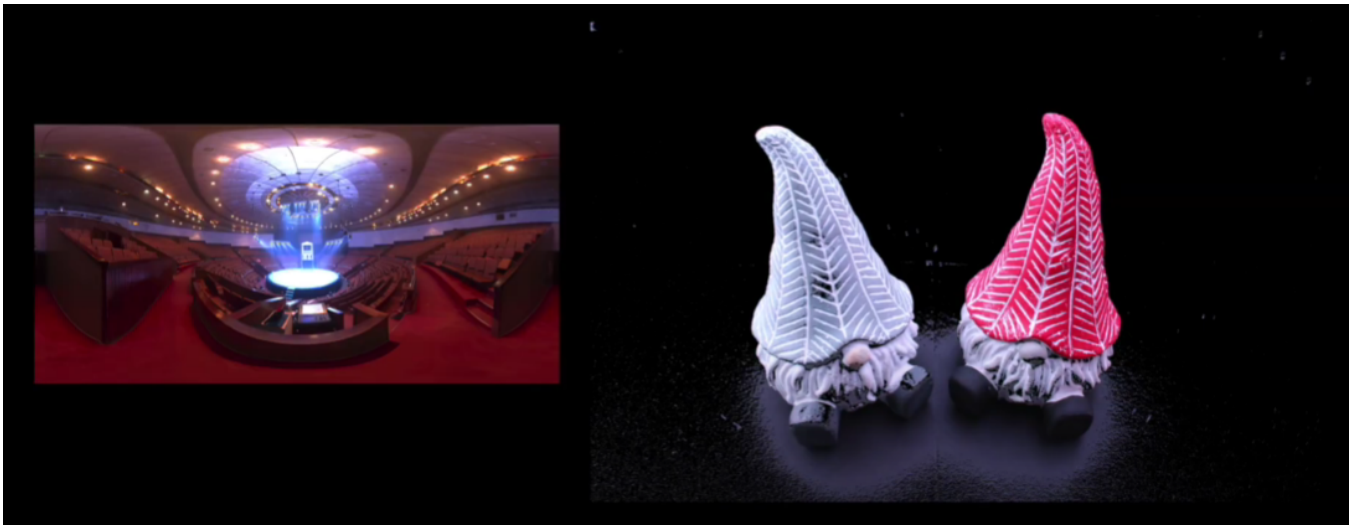

Figure 10

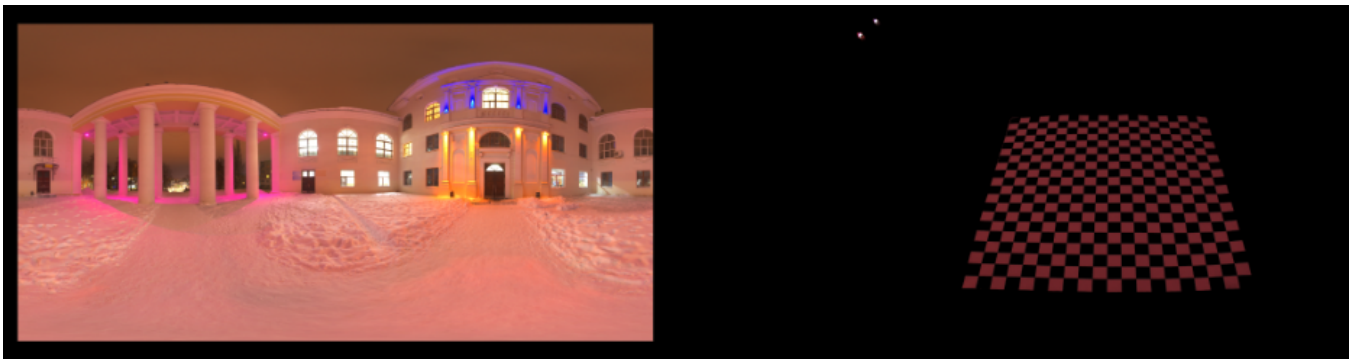

Figure 11

Figure 12

Figure 13

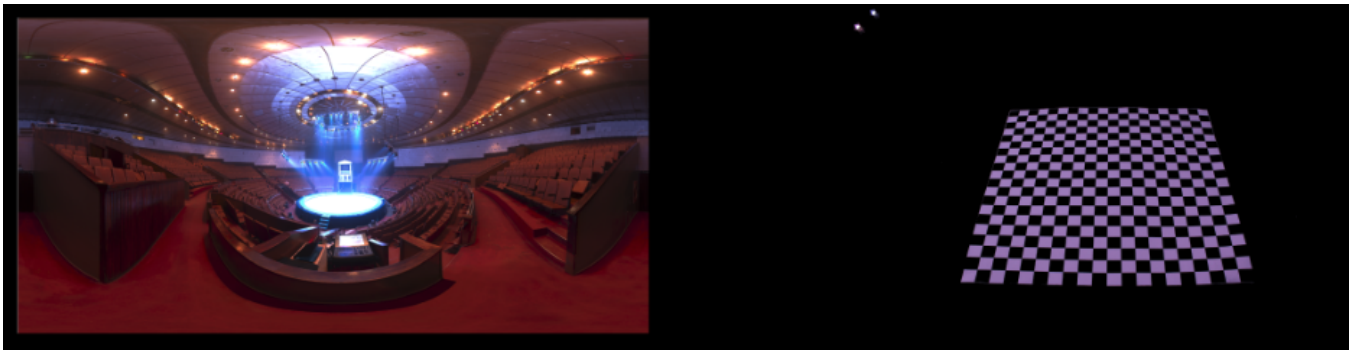

Figure 14