# Rendering Trees in Augmented Reality

Marius Debussche* David Kamm* Adrien Lanne* Boxiang Rong*
ETH Zürich
{mdebussche, kammd, alanne, borong}@student.ethz.ch

## Abstract

*Augmented reality devices provide urban designers and architects with a tool for visualizing their drafts. In the field of biophilic design, this technology for placing virtual trees into real-world environments improves the designing process. While focusing on the realistic rendering of trees, the limiting factor is computational resources. With the use of billboards and 3D mesh representation of the tree model combined with textures used for albedo, normal, specular and animation, we were able to reduce memory storage.*

*Further, we focused on the interactive aspect of the application. The user should be able to intuitively plant and interact with the trees.*

## 1. Introduction

Rendering and animating trees in a virtual scene is a well-known task for game design. Using augmented reality, we can use these virtual objects in the fields of architecture and biophilic design. Enabling the user to test drafts of their design in a real-world scene would improve the decision-making process by providing the user with a visualization and better intuition for composition. Therefore, our interest lies in the realism of the rendering and the interactivity of placing the objects.

The difficulty of this task lies within the trade-off between the realism of the trees and the computational resources of the augmented reality device used. The more realistic the representation should be, the more memory and computational power are needed. We therefore decided on the following approach:

- Billboard and 3D mesh representation of the trees
- Albedo, normal and specular textures for improved visualization
- Tree branch animation for wind simulation

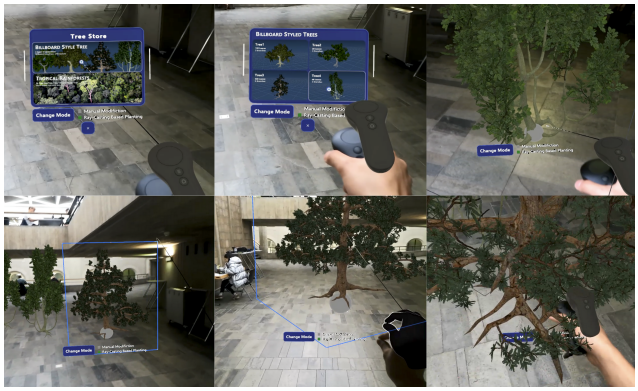



Figure 1. **Screenshots of running program.** The first row shows the start menu, tree selection, and ray-casing-based tree planting. The second row shows the bounding-box modification, hand manipulation and close-up view.

- Pre-captured environment map for lighting conditions in the scene
- UI design for interactively planting the trees
- Plane reconstruction for ground attachment

With the combination of billboards and 3D meshes, we could reduce the memory used by a model using textures to store albedos, normals, and specular colors. Animating the tree branches improves the realism of the object and its inclusion in the outdoor scene. Furthermore, the scene's illumination can be approximated using a sampled environment map under the assumption that the light conditions outdoors would be the same for the whole scene.

To improve the interaction between the user and the application, we design a UI and provide a planting method for placing the trees on the ground. Our implementation is based on the Unity Engine and specifically designed for the augmented reality device MagicLeap using MRTK [6].

*These authors contributed equally to this work

## 2. Related Work

### 2.1. Object representation

For rendering realistic images, Physically Based Rendering (PBR) has become the standard nowadays, as it allows to approximate the visuals of highly detailed surfaces using coarser meshes, and textures which are fast to sample thanks to the GPU's hardware linear interpolation.

For vegetation and specifically the trees, it is impossible to model the whole foliage as it is dense and complex. To create an illusion of volume, billboard are commonly used, which maintains the triangle count low, as done in [2].

Finally, Unity's framework provides various tools that facilitate the development of real-time 3D applications, such as Shader Graph, a graphical interface that can be used to design fragment and vertex shaders.

### 2.2. Animation

Multiple methods are commonly used to create animations. For characters, skeleton animation is typically used [4], and could be adapted to trees. On the other hand, for simpler techniques that use procedural animation, it is more common to see noise-based approaches, such as what is done in God Of War [3].

### 2.3. Lighting

The final challenge is the lighting of the trees. It's necessary to have some form of approximation of global illumination to reach somewhat realistic images. For the time being, it's impossible to compute the exact global illumination in real-time using ray tracing, but it's possible to do approximations using light probes [5]. This approach only works if we have information about the whole scene though, which is not the case in Augmented Reality. Instead, we have to use the device's camera to approximate the environment, which is done in [7], but this is a highly complex technique that does not run in real-time yet.

A simpler idea is to use environment maps. They capture the light coming from all directions at a given point, and can be used to light the scene. This technique is widely used in video games to improve the lighting because of its simplicity, low cost, and acceptable results.

## 3. Rendering of the trees

To achieve the purpose of the application of giving a good enough sense of the presence of trees when virtually planting them, a sufficiently high realism of their look has to be attained when rendering them. Dozens of techniques exist that improve how realistic a virtual object looks while still being rendered in real-time. Among those are albedo texturing, normal maps, specular maps, shadows, environment maps, animation, ambient occlusion, translucency, displacement maps, etc... For rendering our trees, we used a subset of those techniques using the tools provided by the Unity Engine.

### 3.1. Models

Trees are not simple uniform flat geometric shapes. They are composed of thousands of leaves, an intricate branch system, non-homogeneous bark, etc... As such, all those characteristics must be specified in a way that allows for coherent and harmonious results when rendered, which is the work of artists. The content of those models is part of the elements that determine the rendering techniques that we can use for rendering trees. For example, we can only use displacement maps for rendering the trunks of the trees if one is provided for every tree trunk used in our application. Also, the quality of this content highly impacts the quality of the final look of the rendering, and sets a limit to how realistic the trees can look, regardless of the efforts put on our part.

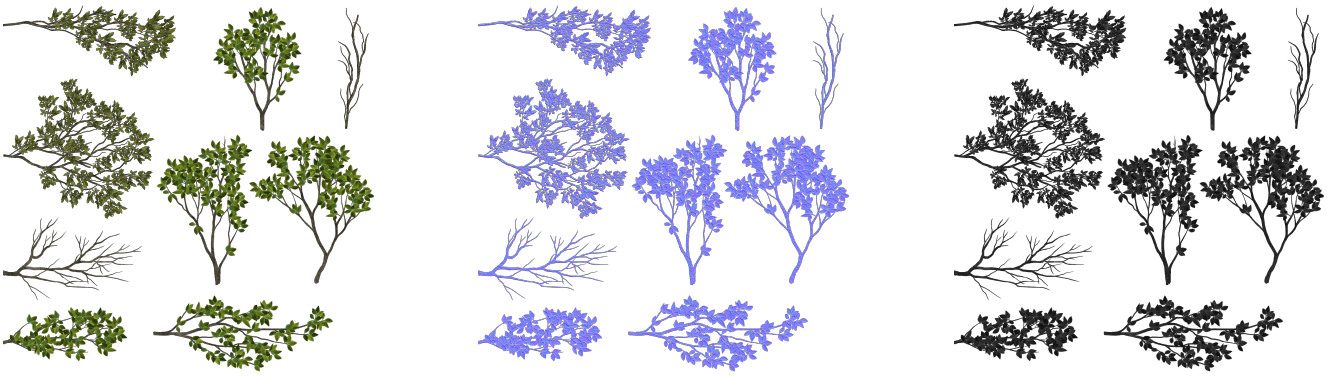
Not having talented artists in our team, we had to rely on existing models found on the internet. Finding sufficiently high quality models to use proved to be quite challenging as the datasets we searched in were composed of free models. Eventually, we found decent ones that provided a complex enough base mesh, texture, normal maps and specular maps.

### 3.2. Render pipeline

The render pipeline is a combination of techniques used to transform the data contained in the models, and additional inputs such as camera position and orientation, into a final image. Different parts of a model can have different pipelines and which one is used for a given part is specified through a material description that encapsulates the pipeline, the textures, and eventually other settings. The Unity Engine already provides a default rendering pipeline for that suits most of the needs in rendering photorealistic trees. Customizing this rendering pipeline allows us to adapt it for specific needs or add more advanced effects and is made possible using the Unity Shader Graph tool that gives control over some parts of the pipeline: the vertex and fragment shaders. Typically, the fragment shader is where the different textures are used to compute the final look of the model.

For rendering trees, mainly two types of materials are needed: one for the trunk, and one for the leaves. Both use albedo texture 2a describing the color of the material, normal maps 2b specifying the direction of the surface normal at each point of the material, which is needed for light computation, and specular maps 2c giving the amount of light reflected. Those are set up by linking the textures to the corresponding slot in the corresponding material.

Since the trunk material describes a classic opaque object with a closed boundary, it does not need further special



(a) Albedo texture. (b) Normal map. (c) Specular map.

Figure 2. Textures used for the rendering branches.

handling, but that is not the case for the leaf material as it is rendered using billboards.

As those are transparent quads using an opaque texture, we have to make the material able to handle transparency, which is made by enabling alpha clipping in the settings of the corresponding pipeline. Generally, to allow for transparency, we use the transparent material setting, which uses the alpha channel of the texture and alpha blending to mix the resulting color of the object with the resulting color of the background behind the object. However, alpha blending is not commutative respectively to the order in which of multiple transparent objects are rendered, and creates in our case a lot of artifacts in the final image. Alpha clipping does not allow a continuous level of transparency but can handle as many transparent objects one behind another as desired.

Billboard quads need also to be rendered on both sides, which by default is not the case. This is enabled easily with the corresponding pipeline setting, however, further care is needed for an accurate rendering: the normal map that describes the direction the material is facing at any given point becomes inaccurate for the backface: we need to account for this by inverting the component of the normal map that is along the normal to the quad.

### 3.3. Environment Map

One difficulty of rendering trees lies within the task of illumination. The virtual object should be affected by real-world lighting conditions and fit into the scene's global illumination. Since we are using an augmented reality device, we do not have the global illumination data of the whole scene and thus have no further information about light source positions and light directions. We have to approximate the illumination for a virtual object in a real-world scene.

In an outdoor scene, we observed that the most significant light sources would be consistent over the whole scene, e.g. the sky and the ground reflection. This assumption would help us to reduce the memory needed to store the illumination. If we sample the global illumination at a single position, we could approximate the light conditions for the whole scene.

To store this global illumination sample, we capture an environment map and store it as a cubemap. A cubemap consists of six images, one for each side. Each image captures the view in a specific direction and provides us with the corresponding illumination information. This again reduces the storage memory needed to only six images.

During the rendering process, to sample the global illumination, a ray from the viewer to the surface is reflected and a sample is received at the intersection point with the cubemap.

### 3.4. Animation

Animation is crucial for the realism of a tree, leaves and branches swaying make a tree feel a lot more integrated in its environment. There are multiple ways to animate a mesh, and we first thought of skeleton-driven deformation [4]. However, this approach is unnecessarily complex for our application, as it would require rigging every tree and designing animations for each one. In comparison, the technique used in God Of War [3] is cheaper, and easier to use for every tree, and the results look convincing. This technique is based on Perlin noise 4a (a type of gradient noise that is often used in real-time computer graphics to increase realism) that is scaled by a weight value 4c that encodes the distance from the attach point of the branch to the tree. It is then applied to the billboard vertices where the texture 4b is rendered. The following computation is done in the vertex shader:

$$\mathbf{v}_{\text{world}} = \hat{\mathbf{v}}_{\text{world}} + \mathbf{d} * s * w * (\text{noise}(d * \hat{\mathbf{v}}_{\text{world}}, t) - 0.5)$$

where $\mathbf{v}_{\text{world}}$ (resp. $\hat{\mathbf{v}}_{\text{world}}$) is the vertex coordinate in world space after (resp. before) deformation. The wind pa-

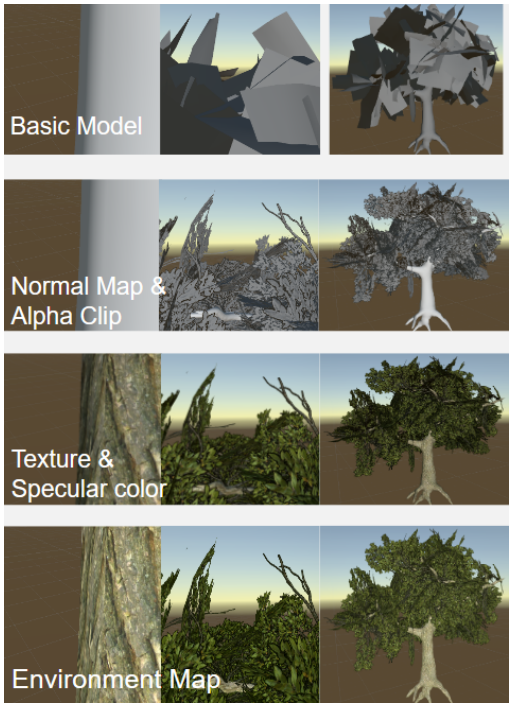



Figure 3. Rendering steps.

rameters are the wind strength $s$, wind direction $\mathbf{d}$, and wind density $d$. In practice, this is done in Unity using Shader Graph.

## 4. User Interface

User interfaces are crucial for giving good user experiences and efficiency. In the field of architectural design and landscape planning, urban planners, the main target user group of this project, usually decorate large volumes of trees in big open areas. With this consideration, we designed a working pipeline that is user-friendly enough and realizes efficient tree planting. Our pipeline has four steps: plane reconstruction, tree planting, bounding-box-based modification, and ground attachment. We provide detailed explanations of each step in the following sections. Additionally, we will distinguish between two modes of tree planting.

### 4.1. Plane recontruction

MagicLeap integrates sensor data over time from multiple viewpoints. The device incorporates built-in features known as "world reconstruction" [1] to continually track its 3D position and reconstruct the environment in real-time. We extract the planar regions, using them as collision surfaces to plant trees. For optimal reconstruction results, it's recommended to scan objects within a distance ranging from 40cm to 5m, and a static environment is preferable. We observed that reconstructions may contain gaps or inaccurate geometry when dealing with outdoor settings or environments characterized by constant changes. Since world reconstruction relies highly on accurate head tracking, pose drift can occur more often in larger scale, darker, or low-texture environments, which results in large holes and badly reconstructed meshes.

### 4.2. Tree planting

Once starts the program, selection windows appear. Users can press the trigger to achieve selection. We provide 2 types of trees for selection, mesh-based low-poly trees and billboard-based trees, and we have 4 models for each type of tree. Once users have made their selection, they can initiate tree planting by employing ray-casting from the controller. Specifically, the controller emits a straight line, which indicates the controller direction, and the point of intersection on the reconstructed planes and object. Then, users can press "Trigger" to instantiate a new tree and press "Bumper" to remove a tree.

### 4.3. Bounding box modification

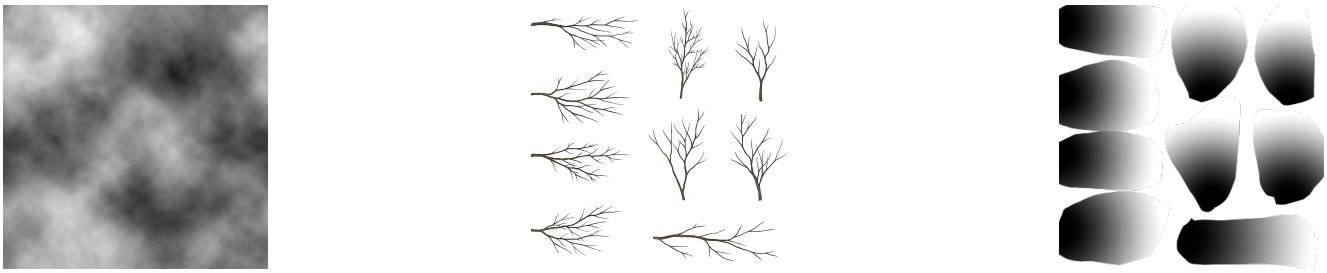
We leveraged the bounding box feature from MRTK [6] to modify the size and orientation of trees. Click once on the tree model to activate the bounding box, we can drag the cubic-shaped handles to achieve scaling and rotating in a user-friendly manner. During this procedure, hands can serve as a replacement for the controller, enhancing the interactive experience for users. However, box handles usually become too small when the model grows too big or is located in the distance, which makes it hard for users to select each handle. In this case, users can use the touchpad to move objects inward and outward, and rotate easily. See instructions in Fig. 6.

### 4.4. Ground attachment

Typically, the intention is to have trees planted directly on the ground. However, after modifications, manually aligning the model with the ground surface can be impractical. To address this, we provide a feature that automatically attaches trees to the ground, shown in Fig. 5. By holding the "Trigger", users can drag and place models. In this process, we keep tracking the distance between the lowest point on the model and the ground surface, where a green anchor ball is placed on the ground to help locate the position. When this distance is lower than a threshold, the anchor ball will turn white, which means it's ready to get attached. Then, release the trigger, the model will automatically be attached to the ground.

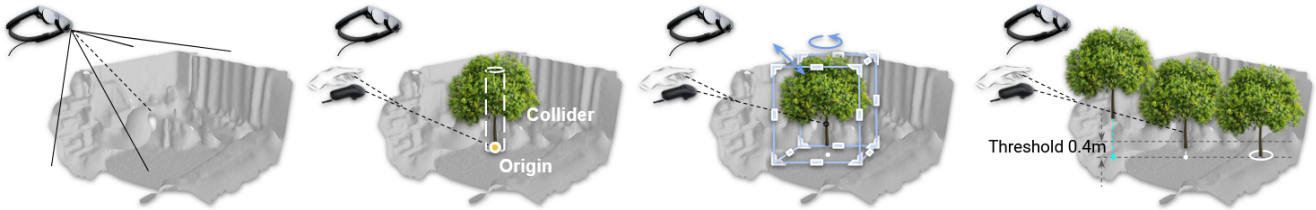
### 4.5. Two Modes

In "Manual Modification" mode, a tree object will be instantiated directly after selection. Users have the flexibility to adjust its scale, orientation, and translation using a



(a) 2D Perlin noise image. (b) Branch texture. (c) Deformation weights.

Figure 4. Textures used in the noise based animation.




(a) (b) (c) (d)

Figure 5. **Tree Planting Workflow.** (a) Once start the program, the surrounding environment is reconstructed from MagicLeap captured 3d information. (b) Using "Ray-caseting Mode", users can plant trees at the intersection point on the floor. (c) After activating bounding-box manipulation, users can adjust model size and orientation. (d) Models get attached to the ground automatically when the distance is lower than a threshold.

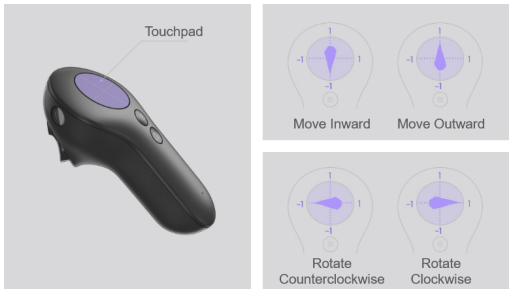



Figure 6. **Touchpad Instructions.** Finger touches in the front and back area, will change the translation of selected models, moving it outward and inward. Finger touches in the left and right area, will change the rotation of selected models, in counterclockwise and clockwise directions.

bounding box. One has to open the menu again and select to instantiate another tree. In "Ray-casting Based Planting" mode, users not only retain the capabilities of the first mode, but they can also instantiate and delete trees using the controller ray, which largely facilitates the planting process.

## 5. User study

The ultimate goal of this product is to enable users with efficient tools to decorate large volumes of realistic trees in big open areas. To assess the effectiveness of our product, we conduct a user study encompassing both quantitative and qualitative measurements. We focus on evaluating planting efficiency, tree realism, and product usability.

### 5.1. Settings

Before user study, each user has to watch an instruction video and play around with the device to get familiar with the operations. Then, they are asked to do a planting experiment, in which the time cost is recorded as the quantitative assessment (See section 5.2). Finally, they will finish a questionnaire with 7 questions, as the qualitative assessment (See section 5.3):

1. Rate our "User Guidance" (0 to 10).
2. Which parts do you think need more instructions? (A. planting tree by ray casting; B. mode changing; C. bounding box manipulation; D. attach tree to ground)
3. Rate the realism of the billboard tree (0 to 10).
4. Rate the realism of the low-poly tree (0 to 10).
5. In which parts, do you think it can be more realistic? (A. appearance of leaves; B. animation of leaves; C. appearance of truck, branches; D. light condition; E. shadow; F. sound)
6. What do you think this App can be used in your work/life? (describe the possible usage)

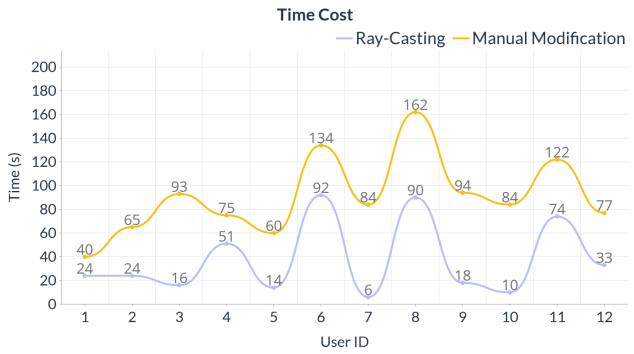



Figure 7. **Ablation Study.** Each candidate is asked to plant the 7 trees at specific locations using both "Manual Modification" and "Ray-casting Mode".

7. Will you consider using this in your work/life? (Yes/No)

### 5.2. Ablation Study

As mentioned in section 4.5, we provide two modes for tree planting. In "Manual Modification", users have to re-open the menu every time to instantiate each model. Based on that, the "Ray-casting mode" is designed with a ray-cast feature to facilitate fast planting. To validate the effect of this design, we experimented with 12 users. In the experiment, users have to finish two tasks and the time cost for each task is measured.

1. Using "Ray-casting Mode", plant 7 trees on the ground, and there should be at least 2 different types.
2. Using "manual Modification", plant the same 7 trees at the same location.

The quantitative results are depicted in Fig. 7. We have observed substantial variations in the time cost for each user, likely stemming from differences in proficiency in product usage. Despite this, all cases prove that "Ray-casting Mode" largely accelerates the tree planting process.

### 5.3. Results

Looking into the qualitative results, almost everyone (11 in 12 people) rated our user guidance as clean and easy to follow, which guarantees the validity of the ablation study. When comparing the two types of trees, all candidates think "billboard-based trees" are more realistic than low-poly mesh trees, which confirms to our expectations. In practice, for real-time operation, it is feasible to plant over 14 billboard-based trees, whereas the device has a maximum capacity of holding only 4 complex mesh-based trees. Therefore, we can tell that billboard-based trees achieve better realism and cheaper computations.

For the product's usability, answers from 12 candidates can be categorized into three groups: Interior Design and Gardening Planning, Games and movie creation, and room decoration for mental health and relaxation. Majorities of people (66%) think highly of the future potential of this product, and express their willingness to use it in their daily lives.

## 6. Conclusion

While rendering in real-time trees that achieve a correct amount of realism, our application allows for intuitive placement and manipulation of those trees by the user. Limitations however are present in terms of interactivity as the maximum performance of the device is quickly reached and only a small set of trees can be rendered at once, and in terms of realism as the trees do not look realistic when seen from up close and are integrated in their environment only through the environment map (they do not shed shadows in the real world, nor contribute to the global lighting).

To go further, more complex rendering techniques such as leaf translucency, displacement maps, light source characteristics estimation from the environment map, or different levels of detail for performance could be impleted. UI wise, allowing trees to be placed in specific patterns, making the capture of the environment map automatic, or connecting the app to a bank of high quality tree models could make the application more complete.

## References


[1] Magicleap developer portal, April 21, 2020. https://ml1-developer.magicleap.com/en-us/learn/guides/developer-portal. 4

[2] Alberto Candussi, Nicola Candussi, and Tobias Höllerer. Rendering realistic trees and forests in real time. *Proc. Eurographics 2005*, 01 2005. 2

[3] GDC. Interactive wind and vegetation in 'god of war', Jan. 2000. 2, 3

[4] J. P. Lewis, Matt Cordner, and Nickson Fong. *Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023. 2, 3

[5] Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. Real-time global illumination using precomputed light field probes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, New York, NY, USA, 2017. Association for Computing Machinery. 2

[6] Microsoft. *Mixed Reality Toolkit 3*, 2023. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/. 1, 4

[7] Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, 2022. 2