Synthesis of Environment Maps for Mixed Reality

David R. Walton * University College London
Diego Thomas † Kyushu University
Anthony Steed ‡ University College London
Akihiro Sugimoto§ National Institute of Informatics

ABSTRACT

When rendering virtual objects in a mixed reality application, it is helpful to have access to an environment map that captures the appearance of the scene from the perspective of the virtual object. It is straightforward to render virtual objects into such maps, but capturing and correctly rendering the real components of the scene into the map is much more challenging. This information is often recovered from physical light probes, such as reflective spheres or fisheye cameras, placed at the location of the virtual object in the scene. For many application areas, however, real light probes would be intrusive or impractical.

Ideally, all of the information necessary to produce detailed environment maps could be captured using a single device. We introduce a method using an RGBD camera and a small fisheye camera, contained in a single unit, to create environment maps at any location in an indoor scene. The method combines the output from both cameras to correct for their limited field of view and the displacement from the virtual object, producing complete environment maps suitable for rendering the virtual content in real time. Our method improves on previous probeless approaches by its ability to recover high-frequency environment maps. We demonstrate how this can be used to render virtual objects which shadow, reflect and refract their environment convincingly.

Index Terms: H.5.1 [Multimedia Information Systems]: Artificial, augmented and virtual realities—Evaluation/methodology I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality—

1 INTRODUCTION

Ensuring that rendered virtual objects appear consistent with the real world is an important goal in the field of mixed reality (MR). Part of this involves rendering the virtual objects in such a way that they appear to be illuminated by the world around them, by reproducing effects such as shadowing, reflection and refraction.

Rendering these effects requires information about the lighting environment, which is often captured using light probes. These light probes may be objects with known geometry and material properties, such as chrome or glass spheres. Alternatively, a camera with a suitably wide field of view may be used. When such a probe is placed at the location of the virtual objects, the information can be used to light the virtual objects accurately.

However, there are applications where it may be impractical to place physical light probes in the real scene. In a mobile see-through AR app, for example, it would be preferable to use a single, self-contained device, such as a phone, tablet or the Microsoft Hololens. A variety of methods exist to capture lighting information without the requirement to place separate physical light probes in the real scene. Such methods, however, typically capture lighting in the form of point light source locations [40, 41, 42, 18, 11, 3] or low-frequency spherical harmonic maps [38, 16, 15, 17]. These are useful for illuminating virtual content with diffuse shading, but, unlike physical light probes, cannot be used to render effects such as mirror reflection and refraction of light. Other approaches [35] capture these higher frequencies, but require the whole environment to be reconstructed as a preprocess, and cannot respond correctly to subsequent changes in the real environment.

The method introduced in this paper attempts to recover full environment maps at the location of each virtual object, which can be used to render these high-frequency effects. The method uses an RGBD camera and a small fisheye camera, contained in the same unit. An indoor, single room environment is assumed, and it is assumed that rough geometry for this room is available (a 2D floorplan, the height of the ceiling, and the initial RGBD camera pose). Both cameras are used to construct and update a detailed model of the room, which can then be used to render the environment maps at any place in the scene in real time. The 3D model of the scene consists of a dense, detailed model of the surfaces around the virtual objects, and a coarse model of the walls, ceiling and floor of the room. Both are updated in real time, enabling the virtual objects to respond to changes in the environment. This division of the real scene into distant and nearby components is similar to that proposed by Debevec [8].

Figure 1: Image augmented with virtual reflective teapot, rendered using the proposed system. Note the reflections of the nearby real objects. No physical light probe was placed near the teapot.

Creating and updating this model is challenging. The main contributions of this paper are as follows:

1. An efficient method for updating the texture of the coarse model using the fisheye camera, whilst correctly handling occlusions.
2. A method for responding to dramatic lighting changes in the room, updating the whole model including regions not visible to the camera pair.
3. A system which uses these data to produce accurate environment maps at desired locations in the real scene.

We demonstrate these contributions by implementing a real-time AR system capable of rendering virtual objects with a variety of material properties in a number of challenging environments. We

*e-mail: david.walton.13@ucl.ac.uk
†e-mail:thomas@ait.kyushu-u.ac.jp
‡email:a.steed@ucl.ac.uk
§e-mail:sugimoto@nii.ac.jp

also evaluate the system, comparing our results to those achievable using the fisheye camera directly. Overall, the results suggest that our approach can produce more accurate results and reduce visual artefacts.

2 Related Work

Existing work on capturing real-world lighting for augmented reality can be roughly grouped into two categories. One uses physical light probes of various forms to capture the necessary information. The other attempts to recover the lighting by other means. In this section, we review some of the existing work in these categories.

2.1 Light Probes for Augmented Reality

Environment maps have long been used to simulate reflection from virtual objects [4]. They have seen widespread use, particularly as an efficient alternative to ray-tracing for simulating effects such as refraction and mirror reflection [13].

Physical light probes are often used to obtain such environment maps. These can be classified into passive probes, such as chrome spheres, and active probes, such as fisheye cameras. Chrome spheres [19, 8, 12, 1, 27] can provide detailed reflection information, which is useful for rendering specular and translucent virtual objects. Other types of passive probes also exist, however, including diffuse spheres [2], which provide useful information for rendering Lambertian virtual objects. Calian et al. [7] presented a novel probe design intended for directly capturing spherical harmonics, for adding virtual content shaded using Precomputed Radiance Transfer (PRT) [44]. Yao et al. [50] demonstrated a method for using arbitrary objects of known geometry as passive light probes.

Cameras with a wide field of view can be used as active probes [24, 25]. They can also provide detailed, high-frequency lighting information, and, unlike passive probes, do not need to remain in the camera's field of view. Grosch et al. [14] made use of this, placing a fisheye camera at the entrance to a room to measure incoming light, and then simulating light transport within the room to add virtual content. Rohmer et al. [39] demonstrated a technique combining information from multiple fisheye cameras to collect detailed lighting over a volume. Other active light probes have also been developed, such as that of Matsuoka et al. [34], who used a hemispherical array of photodiodes to find the dominant light direction.

In the approach presented in this paper, all the data is captured by the single hand-held unit. This provides an active light probe in the form of the fisheye camera. However, the camera pair is typically displaced from the virtual object, and can move arbitrarily as the user moves the unit. One of the main challenges tackled by this work is to make use of this incomplete information to produce a useful virtual light probe at the virtual object's location.

2.2 AR Lighting Without Probes

A wide variety of approaches exist that attempt to recover lighting information for mixed reality applications without the use of physical light probes. Such methods typically attempt to recover some information about the light in the real environment indirectly from its effect on the image to be augmented. These approaches can be categorised by the type of lighting model they attempt to fit to the visible scene.

One class of approach fits a point-source model. In this case, the application's goal is to determine the locations and intensities of one or more point sources in the real scene. For example, a number of techniques [40, 41, 42, 18, 3] attempt to identify light source directions by finding shadow boundaries in the real image, and inferring the locations of the point sources casting them. Zheng et al. [53] uses similar techniques to allow real objects to be manipulated in a static image, and to allow their shadows to change consistently.

Frahm et al. [11] present an alternative approach, using a hardware setup somewhat similar to the one proposed in this paper. They combine a television camera with an upward-facing fisheye camera. In a precapture stage, the pair is moved through the scene, and the fisheye camera is used to estimate the position of light sources in the studio. These are then used for lighting virtual content via shadow mapping.

Another possible model type represents the incident light at the point of interest as a linear combination of spherical basis functions, with the Spherical Harmonic (SH) basis being commonly used. These representations are typically appropriate for representing low-frequency lighting. Okabe et al. [38] introduced an inverse illumination technique which also analysed shadows in order to compute a SH representation of the low-frequency illumination in the scene.

Karsch et al. [28, 29] use a combination of an environment map and visible light sources to estimate illumination from a single image. Since the problem of fitting the environment map is severely under-constrained, samples are selected from a database of environment maps captured from similar real-world environments.

Gruber et al. [16, 15, 17] have presented a number of techniques designed for rendering virtual content with consistent illumination, using only the output of a single RGBD camera. These techniques involve capturing a geometric model of the scene using Kinect Fusion [21, 37]. The environment illumination and albedo of the scene are then estimated using an inverse rendering approach. The lighting is captured in the form of spherical harmonics, which can then be used to illuminate virtual content added to the scene. The approach can produce impressive results, but is limited to recovering low frequency lighting.

Meilland et al. [35] presented an approach for probeless AR. Their approach uses the output of an RGBD camera to construct a dense 3D model of the environment. The auto-exposure feature of the camera is enabled to allow detail to be captured in bright and dark areas of the environment. The resulting HDR model is rendered into cubemaps, and processed to find real light locations to use for rendering shadow maps. The approach can produce impressive results in static environments, but cannot respond correctly to dynamic lighting changes outside the field of view of the RGBD camera. Additionally, it requires the whole real environment to be scanned using a narrow field of view RGBD camera, which can be time-consuming.

Kan [23] introduced a method for capturing a HDR environment map by projecting and aligning multiple images taken using a mobile device, to form a single spherical image. They later showed how this precaptured map could be used to illuminate virtual objects [26].

Whelan et al. [49] developed a method for reconstructing a scene and detecting the locations of point light sources, using an RGBD camera. This involves observing the locations of specular highlights over time, and applying geometric reasoning to find the 3D location of the light sources which caused them.

In summary, the existing literature provides useful ways to estimate low-frequency lighting or point light source locations without the use of physical light probes. This is suitable for rendering diffuse virtual objects, but insufficient to render highly specular reflections. Methods also exist to recover high-frequency information without probes, but these typically require a precapture step and do not update in real time to reflect changes in the lighting environment. This work attempts to build on this by allowing the capture of high-frequency, localised information in real time. This information can be used to render a wider range of virtual materials.

3 System Overview

This section contains an overview of the structure of the system, and details of the hardware setup used.

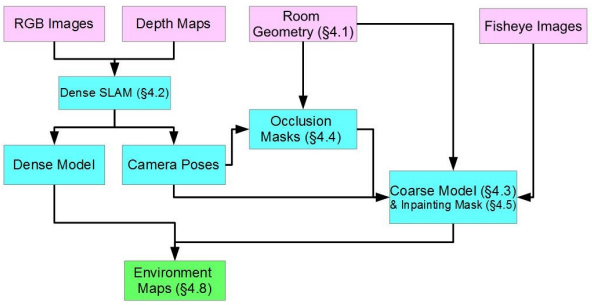

Figure 2: System overview flowchart. Inputs are shown in pink, intermediate data in blue, and the output is shown in green.

Figure 3: Hardware setup used for the proposed method.

3.1 Data Flow

Figure 2 contains a flowchart, giving an overview of the presented system. The inputs to the system (shown in pink) consist of RGB and depth frames from the RGBD camera, fisheye images from the upward-facing camera and rough geometry for the room (generated from the floorplan as described in §4.1). The method produces as output an environment map, rendered from the virtual object's location. This environment map can then be used to render one or more virtual objects, as described in §4.8.

The coarse model geometry is generated as a preprocess, as the application starts. The other stages are performed in real time, as new frames arrive from the RGBD and fisheye cameras. A new environment map is rendered at each frame, prior to rendering the virtual object.

3.2 Hardware Setup

The hardware setup used in this system consisted of an Xtion Pro Live RGBD camera attached to a small upward-facing camera by a rigid bracket (Point Grey Chameleon3 CM3-U3-13Y3C-CS). The upward-facing camera is equipped with a 182 degree fisheye lens (Lensagon CF5M1414), as shown in figure 3. The device is intended to be held by the user, with the RGBD camera facing forward, and the fisheye camera facing upward. The colour output of the RGBD camera is displayed to the user, augmented with the virtual content.

Both of the cameras were calibrated. This involved calibrating the cameras individually, in addition to finding the 6DoF transform between the two cameras.

The RGBD camera was calibrated using the camera calibration app from MATLAB's Computer Vision Toolbox [33], which was applied to the RGB images. The camera used for this implementation was capable of automatically warping the depth image to correspond to the RGB image, and the authors made use of this feature.

The fisheye camera was somewhat more complex to calibrate. There are a range of camera models available for omnidirectional cameras, and each model has advantages and disadvantages, in terms of accuracy and computational complexity. Here, the camera was first calibrated using the calibration method of Scaramuzza et al. [43]. This provides a high-accuracy camera model, but it is also unfortunately somewhat expensive to evaluate. For this reason, at runtime, the images were processed so that they conformed to the simpler model of Ying and Hu [51]. This undistortion stage was performed efficiently using a precalculated look-up table (LUT).

Once both cameras were calibrated, the 6DoF transform connecting their camera poses was recovered. This was achieved by finding point matches in the overlapping region in the images captured by the two cameras, and solving for the transform which best aligned the matches, using the Levenberg-Marquardt algorithm [31, 32].

4 System Detail

This section covers each of the stages involved in the system in greater detail.

4.1 Initialising the Coarse Model

The output of the fisheye camera cannot be used directly as an environment map. This is due to the displacement between the camera and the virtual objects, as well as its limited field of view - the camera used by the authors captured a roughly hemispherical region. One of the main goals of the software is to interpret the output of both cameras, to synthesise an environment map at the location of the virtual object.

In order to render the virtual environment maps, a geometric model of the scene is created. This consists of two components. The first is a dense 3D model generated using the RGBD data, capturing the region directly around the virtual objects. The second is a less-detailed, coarse model of the whole room. This is generated from the rough geometry of the room, which is assumed to be known a priori; the floorplan of the room (a 2D polygon) and the height of the ceiling. From this information, the vertices, indices and texture coordinates of the coarse model are generated automatically.

It is assumed that the system is used indoors, within a room. This was felt to be a reasonable assumption, as this corresponds to the typical use case for the system, and it is not generally possible to use current RGBD cameras outdoors. For outdoor scenes, other environment models could be developed.

For the examples in this paper, in order to simplify the implementation, the floorplan and ceiling height were measured by hand. In an end-user application, however, this information could be obtained using the output of the camera pair, for example by applying the method of Cabral and Furukawa [6] to a suitable image captured by the fisheye camera. Alternatively, a SLAM technique such as LSD-SLAM could be employed. Whilst 3D reconstruction using the depth camera would also be possible, it would be more time consuming due to the narrow field of view of the depth sensor, and in practice was found to be challenging due to loss of tracking on flat featureless walls.

4.2 Dense SLAM

The dense SLAM stage takes RGB and depth frames as input, and uses these to construct a detailed colour model of the region around the virtual content. Additionally, the SLAM process provides an

estimated transform for the RGBD camera. Using the known transform between the cameras (see §3.2), the pose of the fisheye camera can also be estimated.

The dense SLAM algorithm used in this implementation was InfiniTAM [22]. InfiniTAM was selected primarily for its efficiency, but other RGBD reconstruction methods could also be used, such as ElasticFusion [48] or the parametric surface approach of Thomas and Sugimoto [47].

4.3 Updating the Coarse Model Texture

As each new frame is captured by the fisheye camera, it is used to update the texture of the coarse model of the whole room. This is achieved efficiently by making use of graphics hardware.

First, the current location of the fisheye camera is determined using the position of the RGBD camera, and the known transform between the two cameras. The RGBD and fisheye cameras are not synchronised, however, and may capture frames at different times. We correct for this, estimating the RGBD pose at the instant the fisheye frame was captured, and using this pose estimate to find the correct fisheye pose. This is achieved by linearly interpolating, using the two most recent RGBD poses, and the times at which the images were captured. The rotational components of the transforms are interpolated by converting to quaternion form and applying spherical linear interpolation.

Second, a fragment shader is applied to each texel of the coarse room texture. This identifies the position in world space that the texel corresponds to, projects it into the fisheye camera image, and, providing the image location is valid (i.e. the point is not behind the camera), samples the fisheye image and renders the result to the texel.

This procedure is efficient, but does not account for the possibility of occlusions, in the event that the coarse room model is not convex (the middle left image in figure 4 shows an example of such a non-convex floorplan). The process used to handle occlusion is detailed in the following section.

4.4 Handling Occlusion in the Coarse Model

Determining which parts of the coarse model texture are currently visible to the fisheye camera is potentially a challenging problem to solve in real time. A naïve approach might involve casting a ray from the camera location to each texel, but due to the large number of texels to be processed this would be prohibitively slow. However, we can exploit the structure of the model to simplify this task. Since the model is a right prism, it suffices to determine which parts of the 2D floorplan are occluded.

Finding which parts of a polygon are visible from a given viewpoint is a well-studied problem in the literature, and very efficient approaches are available. For this implementation, the implementation of Bungiu et al. [5] from the CGAL library [46] was used. The visible region is referred to as a visibility polygon.

Once the polygon has been determined, it is then used to generate a mesh in the texture space of the coarse model, which is rendered using the graphics hardware to generate a binary occlusion mask efficiently. This mask indicates which texels are potentially visible to the fisheye camera, and is used during the coarse model update step (§4.3).

Figure 4 shows an example. Here, the real environment consists of two rooms, partly separated by a partition, as can be seen in the panoramic image and floorplan. The example is taken from the first frame of the sequence. A 2D visibility polygon is computed from the floorplan, based on the fisheye camera's location (shown as a blue frustum). This is then converted into the binary occlusion mask for the 3D coarse room model. Finally, the coarse model texture is updated with the reprojected fisheye camera image. Only visible components are updated - the rest of the texture is filled in using the inpainting approach described in §4.5.

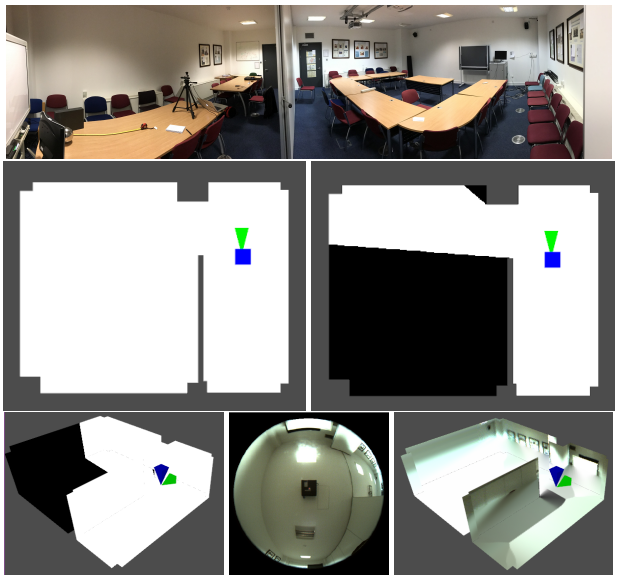

Figure 4: An example of occlusion handling when updating the coarse model. Top: Panoramic image of two rooms, partly separated by a partition (centre of image). Middle left: floorplan of rooms and partition. Middle right: 2D visibility polygon. Bottom Left: 3D coarse model, textured with visibility mask. Bottom middle: Frame from fisheye camera (undistorted) Bottom right: Coarse model, updated with this fisheye camera image. In all examples, the poses of the fisheye and RGBD cameras are indicated by blue and green frustra, respectively.

In many cases, the coarse model is in fact convex (i.e. the floorplan is a convex polygon). In these cases, this occlusion testing step is not necessary, as the coarse model cannot occlude itself, and it can be skipped for added efficiency.

4.5 Completing Missing Regions of the Texture

At a given time, some parts of the coarse model texture may not yet have been observed by the fisheye camera. This may be due to the camera only having seen the upper hemisphere of the room, or also due to occlusions, if the coarse model is not convex. In order to complete the missing regions of the texture, an inpainting approach [45] is applied. One advantage of this approach is its speed, which enables it to be reapplied incrementally at runtime as more of the room is observed, keeping the room texture consistent. A binary inpainting mask is used to indicate which texels have been updated with real data, ensuring that the inpainting is only applied to areas which have never been observed.

Inpainting is generally applied to the walls, ceiling and floor separately, as they are typically of different colours. An exception to this is made in situations where the floor has not yet been observed by the fisheye camera. In this case, the inpainting process propagates data to the floor from the surrounding walls to provide a plausible initial texture.

Figure 5 demonstrates the effect of this inpainting process on the first frame of a sequence. Here, in the example without inpainting, much of the coarse model has not yet been observed. The texture was initialised to a neutral grey colour, which is visible in the reflection from the virtual teapot. In the example with inpainting, the walls and floor have been inpainted, resulting in a more plausible reflection from the virtual teapot.

Figure 5: Scene containing a virtual teapot, with and without inpainting applied. Top: augmented image (with inpainting). Middle: view of coarse model and closeup of teapot, without inpainting. Bottom: view of coarse model and closeup of teapot (with inpainting).

Figure 6: Example of adjusting to a change in lighting conditions. Above: Image with ceiling light turned off. Middle: Image after ceiling light has been turned on, without lighting change estimation, and closeup of virtual sphere. Below: Image after ceiling light has been turned on, with lighting change estimation, and closeup of virtual sphere.

4.6 Reacting to Large-Scale Lighting Changes

Changes in the lighting environment, such as lights turning on or off, or curtains being opened, have a global effect on the appearance of the room. The fisheye camera is able to observe a large portion of the room, and can update areas in its field of view in real time. Regions not currently visible to the camera are not updated by the system described above, however. If the room becomes dramatically brighter or darker, this can lead to noticeable artefacts. These typically take the form of sudden lighting changes across the coarse model texture, as shown in figure 6.

An efficient method was developed to address these problems. At each frame, before integrating the fisheye frame into the coarse model texture, a global illumination change Δ is estimated. This RGB colour value is an estimate of the average intensity change over the coarse model texture, relative to the previous frame. This is computed as part of the coarse model update step.

For each coarse model texel to be updated with new data from the fisheye camera, the intensity difference between the current and new pixel values is calculated. The mean of these differences Δ is then found. When calculating the mean, parts of the coarse model which have not yet been updated with real data are excluded. These areas will contain inpainted texels, which do not necessarily reflect the true appearance of the room. To identify these pixels, the binary inpainting mask is used (see §4.5). Saturated and black pixels are also avoided, as their true brightness is unknown. More precisely:

$$\Delta = \frac{\sum_{t \in T} v(t) \cdot (F(f(t)) - T(t))}{\sum_{t \in T} v(t)}$$

Here, T is the coarse model texture, and t a texel location. F is the fisheye image, and f is a function taking texel locations in the coarse model to the corresponding pixel locations in the fisheye image. v is a validity delta function, defined as follows:

$$v(t) = \begin{cases} 1, & \text{if } t \text{ is visible to fisheye camera} \\ & \text{and } T(t) \text{ is not saturated/black} \\ & \text{and } F(f(t)) \text{ is not saturated/black} \\ & \text{and } t \text{ has been directly observed} \\ 0, & \text{otherwise} \end{cases}$$

This change Δ is added to all texels in the coarse model which were not updated with new data from the fisheye camera. This has the effect of propagating changes in lighting to these texels, ensuring the coarse model appears to be of a consistent brightness. Δ is also used to adjust the appearance of the dense model when rendering the cubemap.

An example is shown in figure 6. Here, a ceiling light was turned on, increasing the brightness of the room. In the middle example, without lighting change estimation, the lower half of the reflected scene (i.e. the sewing machine, table and box) has not changed, and now appears too dark. In the lower example, the change in illumination was accounted for and the brightness and colour of the lower half of the sphere appear more consistent.

This approach uses a simplified ambient lighting model. In reality, light transport through the scene is much more complex. This approximation is efficient, however, and helped to reduce visible brightness inconsistencies in the environment maps.

4.7 Rendering the Environment Maps

The virtual environment maps are then rendered, using the coarse and dense models. One environment map is rendered from the per-

spective of each virtual object. Each environment map is rendered in the form of a cube map, which can then be used directly by the graphics hardware.

First, the textured coarse model is efficiently rendered to the cubemap, using the graphics hardware (each face of the cube is rendered in turn). Secondly, the dense model captured using InfiniTAM is rendered. The dense model is rendered on top of the coarse model (i.e. without depth testing).

This approach was chosen because the dense model is typically closer to the virtual object than the coarse model. Additionally, in cases where the dense SLAM captures geometry already present in the coarse model (i.e. a wall, ceiling or floor) this ensures the version from the dense model is visible. This provides improved results, as the dense model typically has more geometric and texture detail. The dense model is rendered using the GPU-based raycasting procedures in InfiniTAM.

Although the examples shown in this paper involve a single virtual object, if others were present, they could also be rendered into the cubemap at this stage, allowing the environment map to capture the complete mixed reality scene.

4.8 Rendering the Virtual Objects

Environment maps provide much richer information about the lighting environment around a virtual object than simpler real-time lighting models such as point and directional lights. Although originally developed to produce mirror reflection [13], they can be used to simulate a wide variety of virtual materials. Some examples were implemented below, to demonstrate the capabilities of the proposed system.

Environment maps can be used to produce convincing refraction effects, an example of which is shown in figure 7. The top example shows simulation of refraction through a bottle of water. The lower two examples show an example of rendering a metallic teapot, simulating material colour and surface roughness. Surface roughness can be simulated by either combining suitable environment map samples, or by applying a filter to the environment map [36] (or both [30]). The examples shown here simply sample from coarser mipmap levels, which has a similar effect.

Environment maps are not limited to rendering specular virtual objects. They can also be used to render diffuse objects - for example by approximating the environment map using directional lights, via importance sampling. They can also be projected into an SH representation, and used as input to PRT rendering [44], as shown in figure 7, middle. In all the examples shown in this paper, diffuse shadowed PRT is used, with 5 bands of SH (i.e. 25 coefficients).

The implementation of PRT used here also renders a contact shadow onto the table, using differential rendering [8]. Rough real geometry is required for this when precomputing. Here a plane is used, implicitly assuming that the virtual object is placed on top of a locally planar real surface at runtime. This is a common scenario, as realistic virtual objects are typically placed on tables, floors etc.. This allows the virtual object to cast a convincing contact shadow. Since the cubemap is updated and reprojected anew each frame, the virtual object also responds in real time to lighting changes, such as the ceiling light turning on in figure 7 above.

5 Results

This section contains results of using the system in a typical setting. Qualitative comparisons with a real reflective object and a simpler baseline method are shown.

5.1 Full Pipeline Results

Figure 8 shows an example of the data captured when using the application in a typical setting. The top and middle images show views of the textured coarse model, and the estimated camera locations. The lower image shows a raycast of the dense model, from

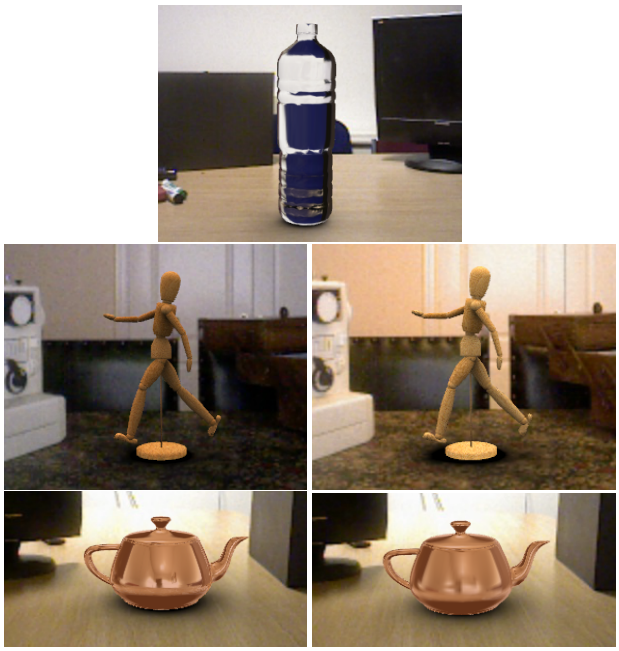

Figure 7: Examples of some of the other virtual objects that can be simulated using environment maps. Top: Water bottle, simulating refraction of the environment. Middle: Wooden mannequin, simulating diffuse shadowing, reacts to a ceiling light being turned on. Bottom: Metallic teapots, simulating different levels of surface roughness.

the viewpoint of the RGBD camera. The appearance of objects near the virtual object (here, the sewing machine, table and box) are captured in the dense model. The appearance of more distant parts of the environment (e.g. walls, ceiling) are captured in the coarse model.

Figure 9 shows an example environment map rendered using this information. The cubemap is visualised as a net.

Figure 10 shows the final augmented image, containing a reflective virtual sphere.

It can be seen that often, the lower walls and floor of a room will not be visible to the fisheye camera during typical use. However, the surface under the inserted virtual objects, such as a table or floor tends to be captured in the dense reconstruction, meaning that a complete environment map can be produced.

5.2 Comparison to Physical Light Probe

In order to test the effectiveness of the approach, a qualitative comparison was performed between virtual and real reflective spheres. The shadow cast by the virtual sphere was rendered using differential PRT, as described in section 4.8. In the examples shown, the capture process was performed, and a virtual sphere rendered. Afterwards, a real sphere of the same dimensions was placed in the same location, and another image taken. The two images were taken using the same camera pose, by attaching the camera to a sturdy tripod. The results are shown in figure 11.

The two spheres appear quite similar - both the nearby objects and the more distant components of the scene, such as the ceiling lights, are reflected in the correct locations on the virtual sphere. Like the real sphere, the virtual sphere also casts a shadow on the table, and this shadow is reflected on the sphere. There are some differences, particularly in the shadowed region under the sphere. In the virtual example, the shadow is softer. This is partly a consequence of the use of spherical harmonics (which can only represent low-frequency illumination) and partly a consequence of the low

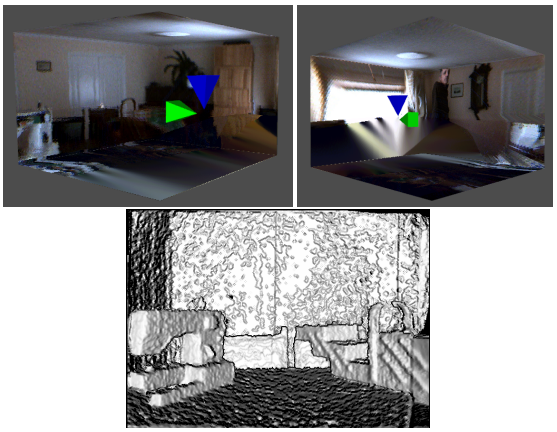

Figure 8: Coarse and dense models captured by the application, in the room of a home. Top: Two views of the textured coarse model. Locations of the RGBD and fisheye cameras are shown as blue and green frustra, resepectively. Bottom: dense reconstruction (ray-casted).

Figure 9: Environment cubemap produced by proposed approach. Rendered using the coarse and dense models shown in figure 8.

dynamic range of the camera pair. The reflections on the lower edge of the virtual sphere are also slightly incorrect, causing the thin bright band on the lower edge of the virtual sphere. This is a consequence of using an environment cubemap, which was rendered from the centre of the sphere.

Additionally, the dense model has not yet fully captured the nearby real scene, so some parts of the table are missing in the reflection. Finally, the reflections on the real sphere are slightly blurred, due to the limited dynamic range and focal depth of the camera, as well as the imperfect surface of the real sphere.

Figure 12 shows a challenging situation for this system. In the middle image, taken as the application started, the purple side of the box, to the right of the sphere is not reflected. This is because the RGBD camera did not obtain depth values for this side of the box, and it was not added to the dense reconstruction. The lower image shows an example after the camera has been moved to the left: the side of the box facing the sphere is now visible, and is reflected properly. As the dense model becomes more complete, the environment maps generated become gradually more accurate, as can be seen in the bottom image.

Figure 10: Image augmented with virtual sphere, rendered using the environment map from figure 9.

5.3 Baseline Approach

The approach was also compared to a simpler baseline approach, without the coarse model approach proposed here. This approach renders the fisheye image into the virtual object's environment map each frame, after applying a rotation based on the current fisheye camera pose. The environment map is not cleared after each frame, allowing it to build up as more of the room is observed. This approach implicitly assumes that the environment observed by the fisheye camera is distant from the object, and does not account for the translation between the virtual object and the camera pair. Example frames from this comparison are shown in figure 13.

These frames were taken from the end of a short sequence during which the camera was moved around the virtual content. Even so, one can see that the environment map generated by the baseline approach is still very incomplete, due to the lack of inpainting and dense modelling. The lower half of the cubemap has yet to be observed, the geometry is incorrect (for example, the window is too large) and the reflections of the sewing machine, chest and table are not present. An example is also shown with the dense reconstruction added. This is an improvement, but there are still incomplete regions, and the reflections of distant objects are still geometrically inaccurate.

5.4 Timing Analysis

In order to assess the performance characteristics of the approach, the application was executed, and the tasks involved in rendering a single frame were individually timed. The input sequence was taken from the rooms shown in figure 4. The virtual object rendered was the teapot from figure 1.

The timings shown were obtained on the CPU, however, many components of the system execute on the GPU. In order to obtain meaningful timings, the GPU was explicitly synchronised after each of these stages. These timings were taken on a desktop PC, using an Intel i7-4970 CPU and an Nvidia GTX 1080 GPU.

Table 1 contains the results. Note that, for simplicity, the less time-consuming stages were combined into the "Other" category. The main component of this is swapping the framebuffers (including vsync). This frame completed in 47.42ms, corresponding to a framerate of 21.1fps.

The brightness estimation stage is the most time-consuming stage. The current implementation of this involves rendering a difference image, and summing it serially on the CPU. The performance could be improved significantly by parallelising or moving to the GPU. It could be further optimised by sparsely sampling,

Figure 11: Comparison between a virtual reflective sphere and a real chrome sphere. Top: virtual sphere. Bottom: real sphere.

Task	Time (ms)	Percentage
SLAM Update	3.16	6.7
Undistort fisheye image	8.55	18.0
Upload images to GPU	0.84	1.8
Render occlusion mask	0.61	1.3
Estimate brightness	21.37	45.1
Update coarse model	0.29	0.6
Render cubemap (coarse model)	0.91	1.9
Render cubemap (dense model)	7.46	15.7
Project cubemap to SH	0.38	0.8
Render scene	0.228	0.5
Other	3.61	7.6
Total	47.42	100.0

Table 1: Timing breakdown of a typical frame rendered by the application.

rather than using all pixels in the coarse model texture. The undistortion and SH projection stages are also currently implemented serially on the CPU, and could benefit from a parallel GPU implementation.

Rendering the dense model component of the cubemap is also quite time-consuming. Whilst rendering the coarse model just involves rendering a textured mesh, adding the dense reconstruction requires a costly colour raycast of the InfiniTAM volume. This performance of this stage also depended heavily on the cubemap resolution; in this example, with 256x256 cubemap faces, it was relatively fast. At higher resolutions, it consumed over 50% of the total time.

6 Conclusion

In this paper, a system was presented to synthesise environment maps for virtual objects in indoor mixed reality applications. The approach used a single, self-contained device containing two cameras. These were used to render reflective virtual objects, and the

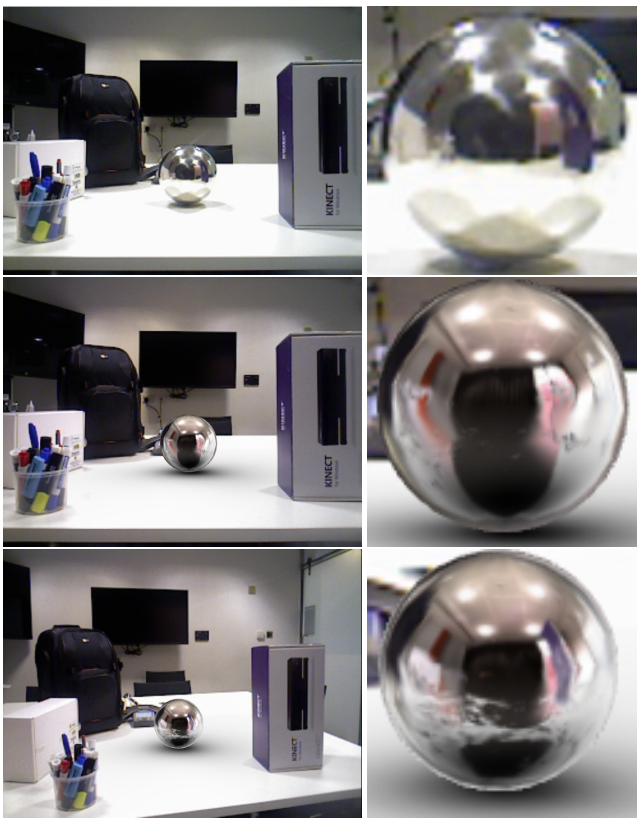

Figure 12: Example scenario where surfaces are not initially visible to the RGBD camera. Top: Image of scene with real reflective sphere, for comparison. Middle: AR image with virtual sphere, on first frame of sequence. Bottom: AR image after moving camera, exposing more of the real scene to the RGBD camera.

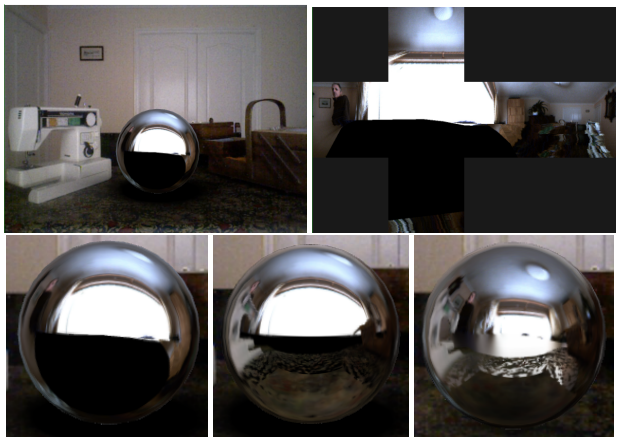

Figure 13: Comparison of the proposed approach with a simpler, baseline approach. The sequence used is the same as that used in figures 8, 9 and 10. Top left: AR Image, using baseline approach. Top right: Environment map, using baseline approach. Bottom left: Closeup of virtual sphere, using baseline approach. Bottom middle: Closeup of virtual sphere, using baseline approach with dense reconstruction. Bottom right: Closeup of virtual sphere, using our approach.

results were compared qualitatively to real mirror surfaces. The system was shown to be able to produce detailed environment maps, which could be used to render virtual objects with high-frequency lighting effects such as reflection and refraction.

Since the coarse model updating and environment map rendering take place in real time, the presented approach can handle a number of dynamic changes. Changes in the surrounding environment such as a video playing on a nearby television screen, a change in the weather outside or a person walking past can be handled correctly. The virtual objects are also capable of changing their position arbitrarily within the real scene. This opens up new possibilities for more dynamic and engaging mixed reality content.

The approach presented here focused on using camera tracking and 3D reconstrution to solve the geometric issues involved in generating environment maps using our hardware setup. Future work might also aim for photometric accuracy, possibly using high dynamic range techniques to capture the full dynamic range of the environment. This would help to reduce artefacts such as the excessively soft shadow in figure 11, and allow for more accurate rendering of diffuse virtual objects. The brightness change compensation stage could also be improved by using a more complete light transport model, to produce more accurate results.

In this paper, the reconstructions generated were used to generate environment maps. These are efficient to render, but are not strictly geometrically accurate, unless the surrounding real scene is sufficiently distant from the virtual object. It would be possible to use a more advanced approach, such as reflection mapping with parallax [52] or multi-perspective rendering [20], to provide more realistic results. Alternatively, the coarse and dense models could be used as input to other rendering techniques such as ray-tracing.

Environment maps also perform poorly when rendering flat, planar surfaces. When such a virtual object is to be rendered, it would be preferable to render the reflection using a camera placed at the reflected (virtual) viewpoint [9]. This could be added to enable the system to render a wider variety of objects

There are also ways in which the system presented here could be enhanced, to improve ease of use. For example, by generating the coarse model automatically, or by using the fisheye camera or a small inertial measurement unit to improve camera tracking.

The dense model only contains surfaces observed by the RGBD camera. If the user does not capture enough of the region around the virtual objects, incomplete parts of the dense model may be visible in the environment maps (see fig. 12). A completion method such as [10] could be added to address this problem.

We believe that the incorporation of omnidirectional cameras into future MR devices such as handhelds or head-mounted displays is quite practical at relatively low expense. We thus believe that the extra information they provide can be key to making more visually convincing virtual content for mixed reality.

Acknowledgements

This research was supported by the NII International Internship Program, as well as a studentship sponsored by Imagination Technologies and the UK's EPSRC. This work was also partly supported by Grant-in-Aid for Scientific Research (Grant No. 16H02851) of the Ministry of Education, Culture, Sports, Science and Technology of Japan. The authors would like to thank the reviewers for their helpful feedback, including a suggested improvement to the brightness adjustment approach.

References

[1] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. Photorealistic rendering for augmented reality using environment illumination. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 208–216. IEEE, 2003.

[2] M. Aittala. Inverse lighting and photorealistic rendering for augmented reality. *The Visual Computer*, 26(6-8):669–678, 2010.

[3] I. Arief, S. McCallum, and J. Y. Hardeberg. Realtime estimation of illumination direction for augmented reality on mobile devices. In *Color and Imaging Conference*, volume 2012, pages 111–116. Society for Imaging Science and Technology, 2012.

[4] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[5] F. Bungiu, M. Hemmer, J. Hershberger, K. Huang, and A. Kröller. Efficient computation of visibility polygons. *arXiv preprint arXiv:1403.3905*, 2014.

[6] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635. IEEE, 2014.

[7] D. A. Calian, K. Mitchell, D. Nowrouzezahrai, and J. Kautz. The shading probe: fast appearance acquisition for mobile ar. In *SIGGRAPH Asia 2013 Technical Briefs*, page 20. ACM, 2013.

[8] 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*, pages 189–198. ACM, 1998.

[9] P. J. Diefenbach. *Pipeline rendering: interaction and realism through hardware-based multi-pass rendering*. PhD thesis, University of Pennsylvania, 1996.

[10] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow. Structured Completion of Unobserved Voxels from a Single Depth Image. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J.-M. Frahm, K. Koeser, D. Grest, and R. Koch. Markerless augmented reality with light source estimation for direct illumination. In *Conference on Visual Media Production CVMP, London*, pages 211–220, 2005.

[12] S. Gibson and A. Murta. Interactive rendering with real-world illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 365–376. Springer-Verlag, 2000.

[13] N. Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

[14] T. Grosch, T. Eble, and S. Mueller. Consistent interactive augmentation of live camera images with correct near-field illumination. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 125–132. ACM, 2007.

[15] L. Gruber, T. Langlotz, P. Sen, T. Höherer, and D. Schmalstieg. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *2014 IEEE Virtual Reality (VR)*, pages 15–20. IEEE, 2014.

[16] L. Gruber, T. Richter-Trummer, and D. Schmalstieg. Real-time photometric registration from arbitrary geometry. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 119–128. IEEE, 2012.

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

[18] M. Haller, S. Drab, and W. Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 56–65. ACM, 2003.

[19] G. Hirota, D. T. Chen, W. F. Garrett, M. A. Livingston, et al. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 429–438. ACM, 1996.

[20] X. Hou, L.-Y. Wei, H.-Y. Shum, and B. Guo. Real-time multi-perspective rendering on graphics hardware. *Rendering Techniques*, 6:93–102, 2006.

[21] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[22] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very High Frame Rate Volumetric Integration of Depth

Images on Mobile Device. *IEEE Transactions on Visualization and Computer Graphics (Proceedings International Symposium on Mixed and Augmented Reality 2015*, 22(11), 2015.

[23] P. Kán. Interactive hdr environment map capturing on mobile devices. In *Eurographics (Short Papers)*, pages 29–32, 2015.

[24] P. Kán and H. Kaufmann. Physically-based depth of field in augmented reality. In *Eurographics (Short Papers)*, pages 89–92, 2012.

[25] P. Kan and H. Kaufmann. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 133–141. IEEE, 2013.

[26] P. Kán, J. Unterguggenberger, and H. Kaufmann. High-quality consistent illumination in mobile augmented reality by radiance convolution on the gpu. In *Advances in Visual Computing*, pages 574–585. Springer, 2015.

[27] M. Kanbara and N. Yokoya. Real-time estimation of light source environment for photorealistic augmented reality. In *ICPR (2)*, pages 911–914. Citeseer, 2004.

[28] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30, page 157. ACM, 2011.

[29] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. *ACM Transactions on Graphics (TOG)*, 33(3):32, 2014.

[30] J. Křivánek and M. Colbert. Real-time shading with filtered importance sampling. In *Computer Graphics Forum*, volume 27, pages 1147–1154. Wiley Online Library, 2008.

[31] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[32] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[33] MATLAB. *MATLAB R2017a: Computer Vision Toolbox*. The MathWorks Inc., Natick, Massachusetts, 2017.

[34] H. Matsuoka, A. Onozawa, and E. Hosoya. Environment mapping for objects in the real world: a trial using artoolkit. In *Augmented Reality Toolkit, The First IEEE International Workshop*, pages 2–pp. IEEE, 2002.

[35] M. Meilland, C. Barat, and A. Comport. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 143–152. IEEE, 2013.

[36] G. S. Miller and C. R. Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments: Siggraph course notes. ACM, 1984.

[37] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[38] T. Okabe, I. Sato, and Y. Sato. Spherical harmonics vs. haar wavelets: Basis for recovering illumination from cast shadows. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–50. IEEE, 2004.

[39] K. Rohmer, W. Buschel, R. Dachselt, and T. Grosch. Interactive near-field illumination for photorealistic augmented reality with varying materials on mobile devices. *Visualization and Computer Graphics, IEEE Transactions on*, 21(12):1349–1362, 2015.

[40] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *Visualization and Computer Graphics, IEEE Transactions on*, 5(1):1–12, 1999.

[41] I. Sato, Y. Sato, and K. Ikeuchi. Stability issues in recovering illumination distribution from brightness in shadows. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–400. IEEE, 2001.

[42] I. Sato, Y. Sato, and K. Ikeuchi. Illumination from shadows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(3):290–300, 2003.

[43] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006.

[44] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 527–536. ACM, 2002.

[45] A. Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[46] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.8.1 edition, 2016.

[47] D. Thomas and A. Sugimoto. Parametric surface representation with bump image for dense 3d modeling using an rbg-d camera. *International Journal of Computer Vision*, pages 1–20, 2016.

[48] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: science and systems*, volume 11, 2015.

[49] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, page 0278364916669237, 2016.

[50] Y. Yao, H. Kawamura, and A. Kojima. Shading derivation from an unspecified object for augmented reality. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 57–60. IEEE, 2012.

[51] X. Ying and Z. Hu. Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model. In *European Conference on Computer Vision*, pages 442–455. Springer, 2004.

[52] J. Yu, J. Yang, and L. McMillan. Real-time reflection mapping with parallax. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 133–138. ACM, 2005.

[53] Y. Zheng, X. Chen, M.-M. Cheng, K. Zhou, S.-M. Hu, and N. J. Mitra. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph.*, 31(4):99–1, 2012.