Synthesizing the Physical World with the Virtual World: Computer Vision for Augmented Reality Applications

Matt Vitelli
Department of Computer Science
Stanford University
Stanford, CA
mvitelli@stanford.edu

Saumitro Dasgupta
Department of Computer Science
Stanford University
Stanford, CA
saumitro@stanford.edu

Ayesha Mudassir Khwaja
Department of Electrical Engineering
Stanford University
Stanford, CA
aysh@stanford.edu

***Abstract*—The purpose of this project was to create an augmented reality system for commodity smartphones. Users gather a variety of images for a given scene and a room layout is computed from the stream of images. Once the layout has been computed, the lighting within the scene is estimated for photo-realistic rendering. The room layout provides support surface constraints for the virtual world, allowing users to interact with the virtual world and place virtual objects within the scene.**

I. Introduction

Augmented Reality (AR) technology has received increased attention in recent years. Due to the proliferation of commodity smartphones and improvements in mobile hardware, we now have mobile devices capable of performing computationally demanding tasks, such as image processing and graphics rendering in real-time. While the hardware has improved, the general methods for compositing virtual objects have remained fairly similar to the approaches used in the late 90's, relying on known 3D shape priors to detect the camera pose with respect to an established reference point. The goal of this project was to use the methods of Furlan et al [1] to estimate the scene layout and camera position for use in augmented reality applications. Once the room layout and camera poses have been established, users are able to interact with the virtual world and overlay virtual objects on top of the physical world observed by their camera.

II. Prior Work

The prior work related to this project has primarily focused on camera pose acquisition or room layout inference, with augmented reality posed as a secondary goal. The task of identifying the pose of a camera with respect to some world reference frame has been extensively explored in the computer vision community. Most methods can be broken down into one of two categories: approaches that make use of semantic information regarding the scene, and those that do not. Among the non-semantic techniques, common approaches attempt to localize a known marker (e.g. a checkerboard, QR codes) and minimize the reprojection error between the observed 2D points and their 3D correspondences [7]. More sophisticated approaches utilize structure-from-motion (SfM) techniques and estimate camera poses by decomposing the essential matrix, which is obtained by point correspondences across consecutive images. Recent methods such as [4] perform a combination of localization and dense reconstruction in parallel, using the intermediate reconstruction as a model to aid in localization. While these non-semantic techniques are general, in practice they often fail to generate dense reconstructions of the scene and are sensitive to noise and outliers during the localization phase. To overcome this, and also simplify the problem, semantic techniques are used that make assumptions about the indoor layout and attempt to find a layout model that fits the scene. A common assumption is that most indoor environments obey the Manhattan world assumptions. In [6], the indoor layout is modeled as a box parameterized by 4 variables corresponding to the angles from the horizontal vanishing points.

III. System Architecture

To accomplish the task of indoor room augmentation, we developed a system that runs on commodity smartphones. The overall system architecture is displayed as a flow chart in figure 1. The system is composed of two key components - offline layout estimation and the real-time augmented reality application.

A. Image Acquisition

The user initiates the process by using our mobile app to capture a video of an indoor scene. These video frames are analyzed by a heuristic that automatically selects frames suitable for use with structure-from-motion estimation. This heuristic works by first detecting ORB [5] keypoints in each frame and matching them against the last selected frame. If a sufficient number of matches are detected, we verify that the disparity is sufficiently large between the two frames. Finally, we compute the fundamental matrix as well as the homography for these two frames and compare the number of inliers for each transformation, as determined by RANSAC. The decision to preserve/reject the frame is then arrived at using:

$$\text{decision} = \begin{cases} \text{preserve} & \text{if } |\text{Inliers}(F)| > C_n \cdot |\text{Inliers}(H)| \\ \text{reject} & \text{otherwise} \end{cases}$$

B. *Offline Layout Estimation*

The images acquired in the previous stage are fed into the structure-from-motion pipeline. The resulting point-cloud from the structure-from-motion pipeline is often noisy and sparse, due to the limited number of images gathered and the complexity of the indoor scene. To combat this and provide a simpler parameterization for the final room layout, we make use of the methods of [1] and use RANSAC to estimate the planes passing through strong support surfaces such as the ceiling, the floor, and the walls in the image. These serve as the candidate layout components for the room and using the particle filter optimization procedure proposed in [1], we are able to hypothesize a box shape for the room structure.

C. *Lighting Estimation*

Once a plausible room layout has been generated, we estimate the lighting parameters for the scene. The process for this is largely based on the approach of [3], however our method benefits from using estimates across multiple views and does not require any user-interaction to provide initial estimates for the lighting or scene geometry. The process for estimating the lighting from a sequence of images is described as follows (figure 3: first we perform the Color-Retinex method [2] to decompose each image used for the room layout into the product of two images an albedo image and a grayscale shading image describing the lighting in the scene. The Color-Retinex method works by estimating the partial derivatives of a log-intensity function describing the change of lighting within a scene. Since lighting is largely constant across flat surfaces, many of the partial derivatives will be zero, with non-zero entries occurring at surface discontinuities. We can form a system of linear equations of the form $Ax = b$, where x represents the log-intensity function, A represents a matrix that computes a finite-difference approximation of the partial derivatives of x, and b represents the estimated partial derivatives of the log-intensity function. By solving this system of linear equations, we are able to obtain an estimate of x, the log-intensity function representing the shading image.We obtain the albedo map by dividing the original image B by e^x and normalizing the resulting albedo map so that all values in it lie between 0 and 1.

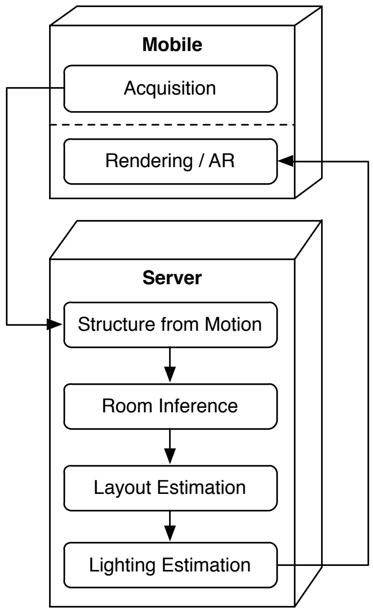

Fig. 1. Overview of System Architecture

In addition to computing the albedo, we must also somehow account for the indirect illumination within the scene. Fortunately, because we have access to both coarse scene geometry and the original images of the scene, this turns out to be fairly straightforward. First, we discretize the room geometry into a number of tiny patches. Each patch represents an approximation of the indirect irradiance in the scene and we compute this indirect irradiance hitting a patch by using a gathering variant of the radiosity equation. For each patch, we generate a number of rays distributed around the hemisphere of the patch's surface normal. These rays are then intersected with the scene geometry and projected into the camera images used for Structure-from-Motion. We weight each pixel sampled within each image by the cosine of the angle between the sample's ray and the patch's surface normal. As the number of patches shrinks to infinity and the number of samples grows to infinity, this approximation is equivalent to the integral in the radiosity equation.

After computing the indirect irradiance map, we are able to cancel out the effect of indirect illumination by the following equation:

$$D = B - pT$$

where D represents the direct lighting image, B represents the original image, p represents the albedo estimates from the Color-Retinex method, and T represents the indirect irradiance image. We compute direct lighting images for each of the original images used for Structure-from-Motion. With the direct lighting, albedo images, and geometry estimates, we are able to set up a non-linear least-squares optimization in which we seek to minimize the summed squared error between our synthetic lighting model and our direct lighting estimates. This optimization is quite general and nearly any collection of relatively simple lighting models can be used, as long as they contain only a small number of parameters. For the purposes of this project, we assume that nearly all indoor scenes contain either a single point-light or spotlight that accounts for most of the light within a scene. While this assumption does not hold true in general, in practice these simplified models perform quite well and users are usually not very sensitive to small changes in illumination. After the optimization is complete, we have a collection of lighting parameters that can then be used for real-time rendering.

D. *Real-Time Application*

The server returns the best candidate room layout, lighting parameters from the lighting optimization procedure described earlier, estimates for the camera intrinsics, and poses for each of the images used for reconstruction. The box is encoded by the eight vertices corresponding to each corner of the box. Once the room layout has been estimated, the user can launch the augmented reality application on their phone and begin interacting with virtual objects. The augmented reality application is written in Unity 3D, a popular game engine used by independent game developers and large commercial companies. The real-time application takes the box structure, intrinsic parameters, and pose estimates for a given view as an input and outputs a rendered scene of the augmented reality world from a physically accurate perspective (i.e. the virtual world is rendered from the same perspective as the real camera). To accomplish this, we must transform the intrinsic parameters expressed in pixels to homogeneous device coordinates in OpenGL. To perform this, we convert the intrinsic matrix as follows:

$$K_{SfM} = \begin{bmatrix} \alpha & \gamma & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$K_{GL} = \begin{bmatrix} \frac{-2\alpha}{w} & \frac{2\gamma}{w} & 1-\frac{2c_x}{w} & 0 \\ 0 & \frac{-2\beta}{h} & 1-\frac{2c_y}{h} & \frac{2f\cdot n}{f-n} \\ 0 & 0 & -\frac{f+n}{f-n} & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

where K_{SfM} denotes the intrinsic matrix estimated from the Structure-from-Motion pipeline, K_{GL} denotes the intrinsic matrix used by OpenGL, n and f are the distances to the near and far planes, while w and h are the width and height respectively.

The third and fourth rows are required for hardware rasterization and the third row takes the virtual camera's far and near clip planes into account. The non-zero terms are necessary for depth values to be properly mapped to the Z-buffer. Depending on the representation of screen coordinates in the graphics display, the second row of the matrix may be negated. This corresponds to how the graphics display defines the display origin - either by defining it from the upper-left corner or the bottom-left corner, respectively. In addition to transforming the intrinsic matrix, we must also transform the extrinsic parameters of the camera to align with Unity's coordinate frame. Since the room layout inference encodes the box and pose coordinates in a right-handed coordinate system but Unity encodes its coordinate frame in a left-handed coordinate system, a transformation is needed to convert the right-handed coordinate frame into a left-handed coordinate frame. To accomplish this, we transform the camera's rotation matrix by π radians in the X direction and -π radians in the Z direction. In addition, we also negate the camera's X position to align with the transformed right vector. It should be noted that the choice of these transforms is arbitrary and the reasoning behind choosing this transform is entirely dependent on how the augmented reality application chooses to define the axes of the world coordinate frame. In the case of Unity's representation of the world coordinate frame, this transformation happens to be appropriate.

E. Using Support Surfaces for Augmented Interactions

To model the augmented reality interactions between the virtual world and the physical world, we use the supporting surfaces from the box layout to constrain the placement of objects to lie along one of the planes corresponding to the box. We project a ray from the current camera pose to the environment and find the nearest intersection between the box and the camera. This nearest intersection lies along one of the box's walls. Once we have found the nearest intersection, we transform the user's virtual object's pose to be centered at the intersection point with the up vector aligned to the support surface's normal direction. This allows users to interact with the scene and construct virtual worlds on top of the physical world (see figure 2.

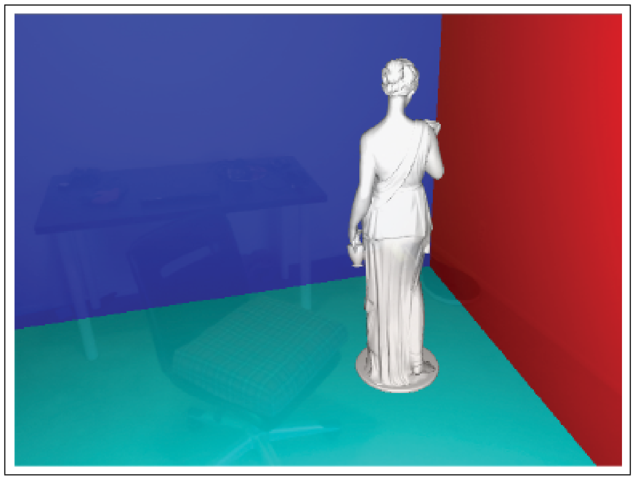
Fig. 2. The estimated layout overlayed on top a static frame. The test model can be interactively positioned anywhere within the room by the user.

F. Physical Material Augmentation

An interesting application of our pipeline is that it offers users the ability to not only insert virtual objects into the scene, but also augment the physical materials of the original scene. A wall or floor can be digitally re-textured using the combination of geometric and image-based cues. To accomplish this, we project the albedo maps from each image onto the room's geometry. At each point on the room's geometry, we have a color value representing the average albedo from all images. Since most albedos tend to be fairly constant in their brightness and color responses, they are easy to segment reliably. By segmenting the albedo colors for each face, we are able to build a collection of alpha-masks that users can use to blend between the original scene's materials and a set of augmented materials. Using this approach, we can transform simple scenes such as a conference room or lounge into fantastic scenes such as medieval dungeons or futuristic cityscapes. An example of this can be seen in figure 4.

IV. Results

Figure 2 shows a screenshot of our AR app running on an Apple iPad. The estimated layout overlayed on the scene from which it was generated. The virtual model in the foreground can be interactively positioned anywhere within the room by the user

The images shown in figure 4 demonstrate the renderings produced by the augmented reality application from the estimated room layout. While the rendered results demonstrate a 1-1 correspondence with the estimated layout, our room layout inference algorithm does not always produce physically plausible layouts from the set of input images. Generally, the inference algorithm tends to produce realistic layouts for simple, clutter-free scenes where at least three support surfaces are directly visible. Because our inference algorithm makes use of the Manhattan world assumptions, the algorithm struggles with rooms with complex geometry and multiple support surfaces that are not orthogonal to one another.

V. Future Work

The work presented in this paper scratches the surface of what is possible in virtual augmentation. There are a variety of directions we could take to improve the current system. Some of the most prominent ideas for improvements are as follows:

A. Real-Time Pose Estimation

While our system produces reliable estimates for the support surfaces of a room from a number of images, currently only the images used in the structure-from-motion pipeline can be augmented. One possible improvement would be to perform camera pose estimation in real-time and provide users with the ability to view virtual scenes from a variety of perspectives. Using the images with known poses from the structure-from-motion pipeline, we could establish point correspondences between the images with known poses and live camera pose, allowing us to infer the pose of the live camera.

B. Real-Time Layout Refinement

In addition to estimating the camera pose in real-time, we could also use the live pose estimates to refine our structure-from-motion reconstructions. From these, we could use the improved structure-from-motion point-cloud to establish better candidate components and infer a more accurate box layout for the scene.

C. Improved Support Surface Modeling

One possible extension to this project is to provide a more accurate representation of the room and model additional support surfaces within the scene. Rather than simplifying the room representation as a box, we could also model the geometry contained within the room, such as tables, couches, and chairs. By modeling these additional support surfaces, we can provide more realistic interactions between the virtual world and the physical world. In addition, knowing the placement of common household objects and their support surfaces could provide additional information for the room layout estimation pipeline, possibly providing more accurate estimates of the room structure (e.g. couches are usually against walls and tables must be supported by the ground plane). There are a variety of ways this could be accomplished. One method would be to detect common household objects in 2D and fit the pose of a known 3D model to the evidence observed in the image. Another method would be to perform a dense reconstruction of the room and use the reconstruction to provide constraints for support surfaces.

VI. Conclusion

We developed a system for indoor augmented reality interactions using room layout inference to establish support constraints from the physical world. Unlike previous methods, our method does not require estimation of the ground plane and runs on a commodity smartphone. Because we only model the primary support surfaces of the room, we are able to provide realistic interactions from only a small handful of images and overcome the need for a full dense reconstruction of the physical scene.

VII. Acknowledgments

We gratefully acknowledge Yingze Bao and Professor Silvio Savarese for their valuable help and feedback on this project and providing us with the resources to make this project possible.

References

[1] Furlan, A., Miller, S., Sorrenti, D. G., Fei-Fei, L., and Savarese, S. Free your camera: 3D indoor scene understanding from arbitrary camera motion. In *British Machine Vision Conference (BMVC)* (2013), p. 9.

[2] Grosse, R., Johnson, M. K., Adelson, E. H., and Freeman, W. T. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 2335–2342.

[3] Karsch, K., Hedau, V., Forsyth, D., and Hoiem, D. Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics (TOG) 30*, 6 (2011), 157.

[4] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 2320–2327.

[5] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 2564–2571.

[6] Schwing, A. G., and Urtasun, R. Efficient exact inference for 3d indoor scene understanding. In *Computer Vision–ECCV 2012*. Springer, 2012, pp. 299–313.

[7] Zhang, Z. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 11 (2000), 1330–1334.

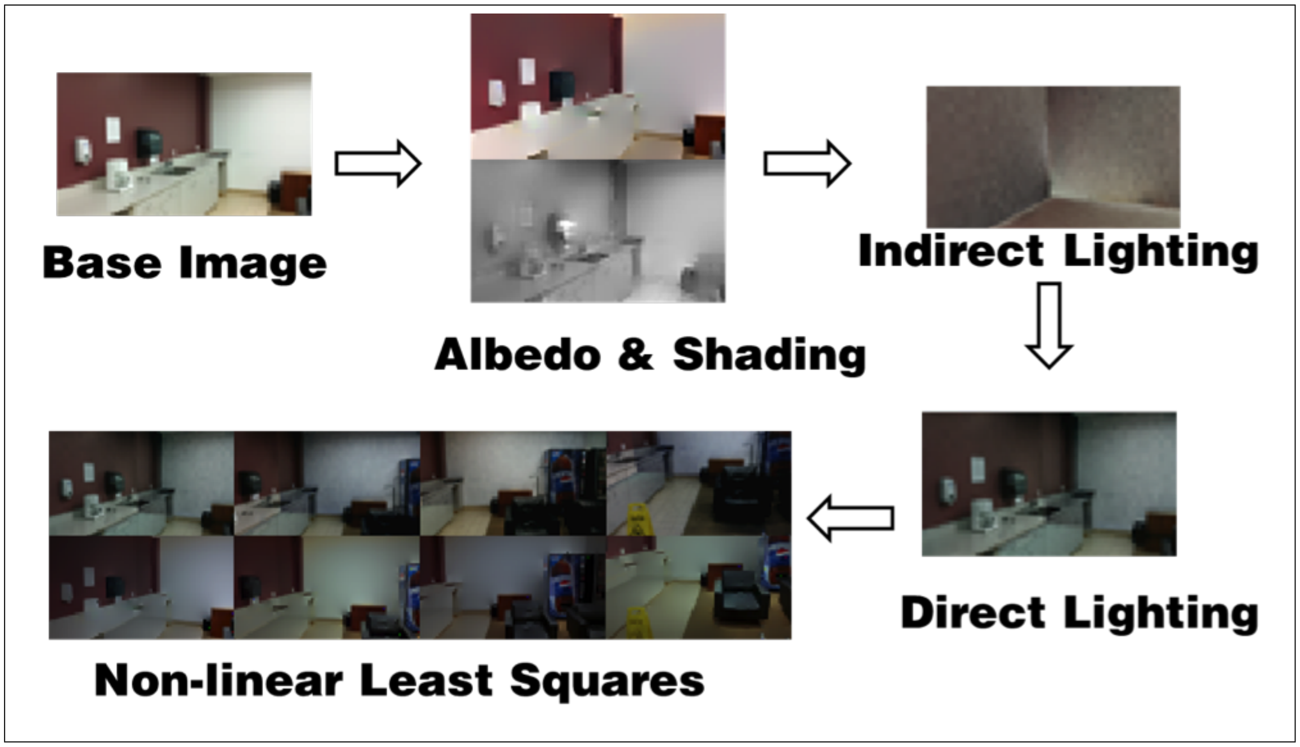

Fig. 3. An overview of the lighting estimation pipeline. Images are decomposed into albedo and shading components and an approximation of the scenes irradiance is computed. These estimates are used to approximate the direct lighting in the scene, which is then used to optimize parameters for the scene lighting model.

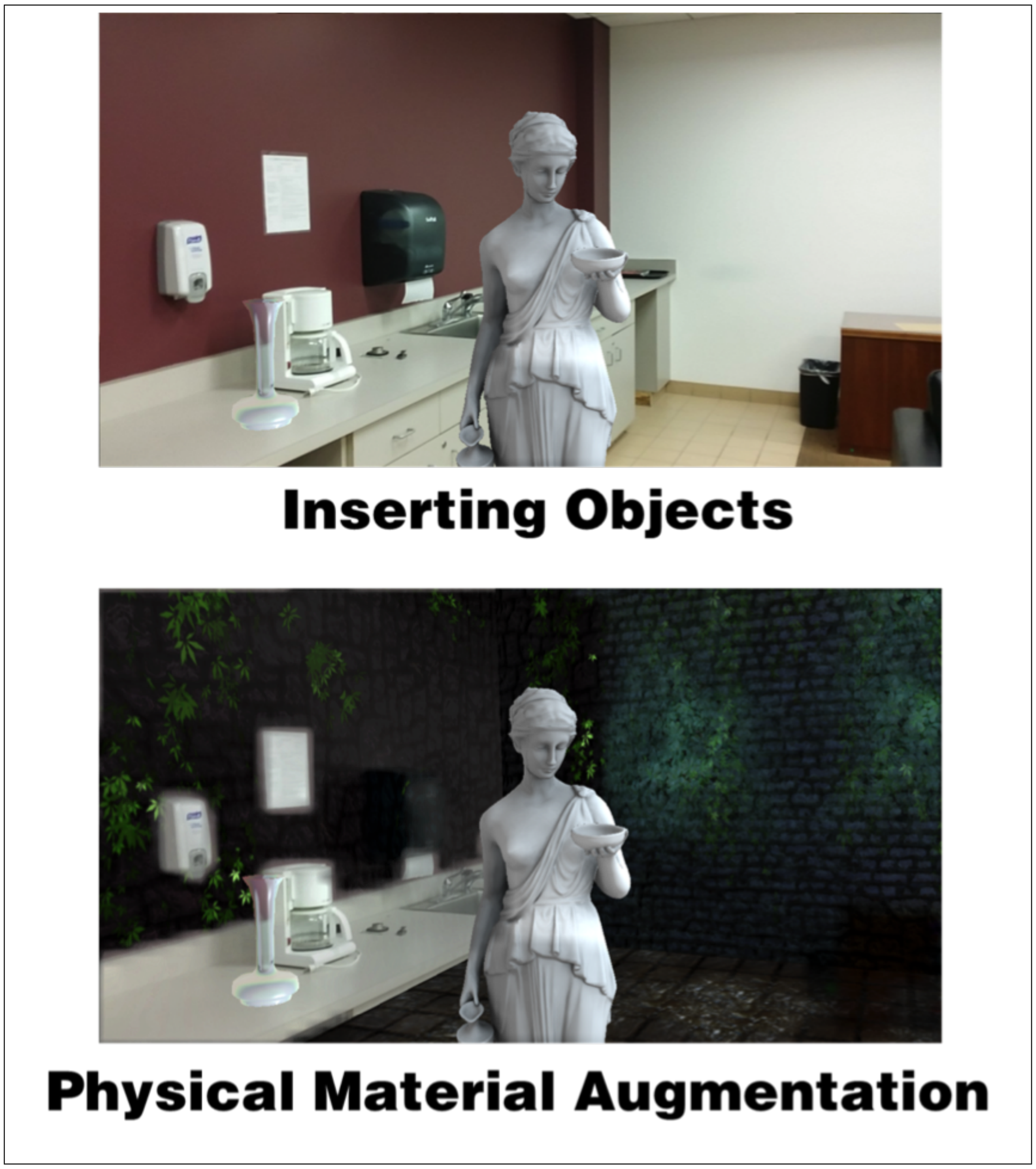

Fig. 4. The final composited image of the virtual and physical scene.