A.1 Additional Details

Reconstructing any 3D representation just from the object's binary shadow masks is a very hard and unsolved problem. Shadows possess important geometric information about the object, however, due to the ill-posedness, many different kinds of objects that have similar appearance in 2D can cast the same shadow. Moreover, these objects can map to the same shadow mask and have very different underlying 3D shapes making this problem more difficult than recovering 3D geometry from images. Our algorithm makes limited assumptions about the underlying scene working with hard shadows. It also doesn't assume the number of objects in the scene and tries to fit the best 3D model that explains the shadows masks.

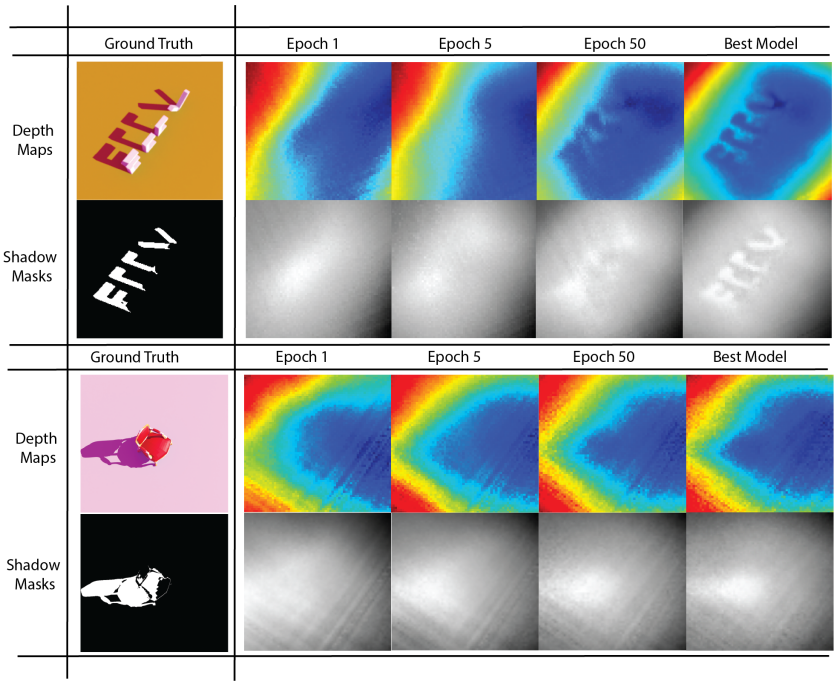

Fig. A.1. Predicted depth maps and shadow masks for a scene with multiple objects and a chair during training on validation poses. Reconstructing any 3D representation from the object's binary shadow masks is a very hard and we show that we can extend our algorithms to a scene with **arbitrary** number of objects. This is due to our use of implicit volumetric representation and that our our algorithm makes limited assumptions about the scene or its objects.

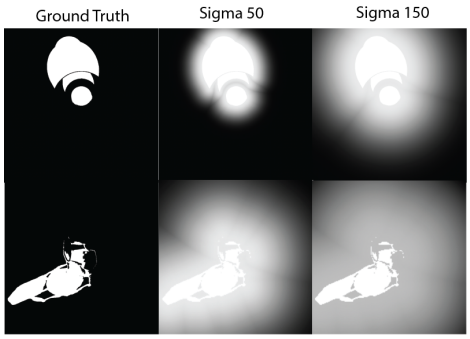

Fig. A.2. Increasing the distance transform weight "sigma" hyperparameter. We show the shadow maps that the model actually learns from. We apply the distance transform and slowly decrease it in our training. The transform helps with taking smoother gradients w.r.t. the model's parameters.

A.2 Results

We show predicted depth and shadow masks on a scene with multiple objects and the chair in Figure A.1, and on a bunny object in Figure A.4. In the first row of Figure A.1, we run our method on a scene with multiple objects that spell **ECCV**. If we had used explicit meshes, we would have to specify the number of objects before training, however, by parametrizing the scene with opacities, we can scale up to an arbitrary number of objects placed in any position in the scene. For the scene with multiple objects, we generate shadow masks and feed them to the proposed algorithm as input, without specifying additional information. Since our algorithm does not assume any prior knowledge about the scene or its objects, it can optimize over and find the best volumetric densities that explain the shadow mask. This result further shows how traditional algorithms like shape-from-shadows/X can benefit from differentiable volumetric rendering as with it they can scale to an arbitrary number of objects and more complex scenes.

For the chair object in Figure A.1, observe how the chair is on the right of the casted shadows and the model accurately predicts and segments out that area, and is able to infer the sharp corner.

As shown in Figure A.4, the model is able to learn the coarse shape and localize the bunny well through its shadow map. However, it fails to carve our the ears and the and finer details. Our shape-from-shadow algorithm has the poorest performance on the bunny (Figure A.4), most likely due to its complexity, curved surfaces, and its protruding ears that can imply many different possible combinations of the underlying 3D shape. Note that the model is trained on coarser versions of the fine shadow masks (shown in first column).A.3.

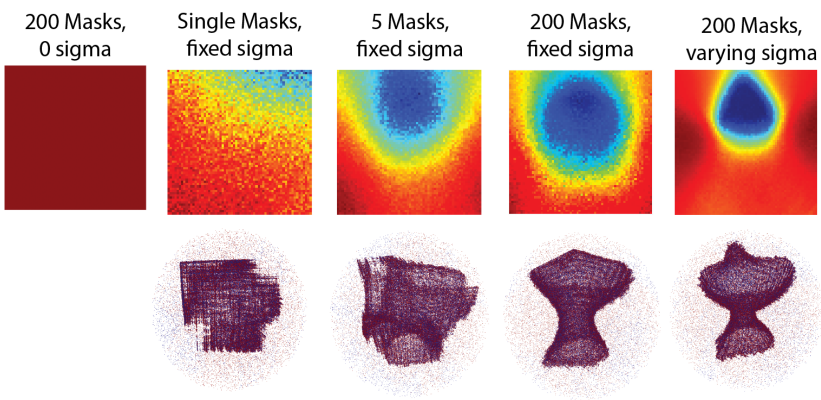

Fig. A.3. Ablation on sigma and number of shadow masks. We study the importance of the distance transform in our method. Without the distance transform to guide and smoothen the shadow masks, there is no convergence in the depth maps or the mesh. We notice an iterative coarse to fine mesh estimation done by the framework as we increase the number of masks and vary sigma. We observe that the number of masks carves the space down to create finer meshes. Varying sigma with many shadow masks yields the best results, although it can overfit to the shadow mask and cause mesh artifacts.

A.3 Implementation

Model Details. During training, we set β $1\mathbf{e}^{-2}$ or $1\mathbf{e}^{-3}$ and ϵ is usually set to 0. We set the μ_{min}, μ_{max} to be 0 and 1 respectively and compute gradients directly on the normalized values instead of using the sigmoid. For the position encoding we also use parameters from the original NeRF implementation. We use an 8-layer MLP to parametrize our scene. To enable better depth reconstruction we also use a coarse and a fine MLP, the coarse is sampled 64 times and the fine 128. Our MLP, however, does not have an extra head for the color and does not take the viewing direction as input.

Training Scheme. To train the model, we use the PyTorch implementation of the Adam optimizer [**?**], use a learning rate of 5×10^{-4}, and use a step function to decay the learning rate at 20 epochs. We evaluate our model on the validation data, and test it against real meshes from blender. We start training with a high distance setting for our distance transform $\sigma = \{100, 150\}$ and train for 150 epochs. Then we lower the $\sigma = \{50, 45\}$ to continue training. We note that using a σ value lower than this causes our model to diverge and start to learn spurious depth maps. Our entire implementation trains on one Tesla V-100 and takes a half a day. The models were typically trained to run for 200/300 epochs where each epoch runs on all rays generated from all camera pixels. Only the cuboid was trained on 2 Tesla V100s on an image of size 128×128 with 64 fine and coarse samples. This training took 3 days to complete.

Real-World Dataset Pipeline. We place a mannequin object of the hand in the scene and use the flashlight from a smartphone and take a video of the scene using another smartphone. We start the data collection near the flashlight so that the first 30 poses are roughly similar to the pose of the light. We process the remaining 74 frames and perform an intensity threshold to extract the shadow masks. This simple technique also causes the darker regions to be classified as shadows, as visible in Fig. **??**. We then run COLMAP [**?**] [**?**] on the video to extract the poses, and use the first frame's pose as the light pose discarding the rest. We note that our method is robust to the coarsely estimated light pose as there are shadows visible from that viewpoint.

Efficient Differentiable Shadow Rendering. Our approach requires two NeRF forward passes per epoch to train: one from the light's perspective and one from the camera's. To make training faster, we implemented a more efficient method to compute depths. We exploited the fact that our method requires the shadow map array to be indexed by the projected camera pixels, therefore we only need to compute a full shadow map on $H \times W$ rays and can batch the camera rays. This approach worked well and decreased our computational cost by roughly half. Moreover, our method does not make any assumption on the size of the camera and light depth arrays, and therefore we could also use a smaller shadow map. We also implemented a more efficient method of projecting the camera pixels into the light frame of reference, only computing the light depths on the projected locations instead of computing the full shadow map. This implementation, however, does not lead to any convergence, indicating that the loss computed on out-of-shadow pixels is also important and gives valuable information that helps in carving away that space to refine the mesh.

We also experimented with changing the number of fine samples used to sample opacities for a given light ray, computing gradients on the light opacities every K steps, in addition to sampling light depths at varying intervals instead of every step. Albeit many such methods did speed up training, we decided to use a basic setup which samples the light and camera rays with 64 coarse and 128 fine samples every iteration, with gradients being computed at each step.

Explicit Mesh from Implicit Representations. Extracting an explicit mesh from an implicit volumetric representation typically involves running marching cubes [**?**]. We used the PyMCubes library implementation of the marching cubes algorithm. To do this, we create a bounding volume around the object and query the 3D volume for opacities at points inside the cuboid. In practice, we query along each dimension 128 to 256 of the volume of size (H, W, D) therefore creating voxels of size $H/128, W/128, D/128$. We query for an opacity at every voxel once to create a volume that can then be fed into the marching cubes algorithm. We use a set contour value of 0 to search for isosurfaces in the volume. This gives us a non-smooth, explicit mesh with vertices and triangular faces which is then used for visualization and evaluation. Note that the volume bounds are different for each object and we find them using trial and error. In (x, y, z) direction, they are $\{\pm 5\}$ for cuboid, $\{\pm 35\}$ for chair, $\{\pm 45\}$ for vase and $\{\pm 35\}$ bunny. We refer our readers to our code for more infor-

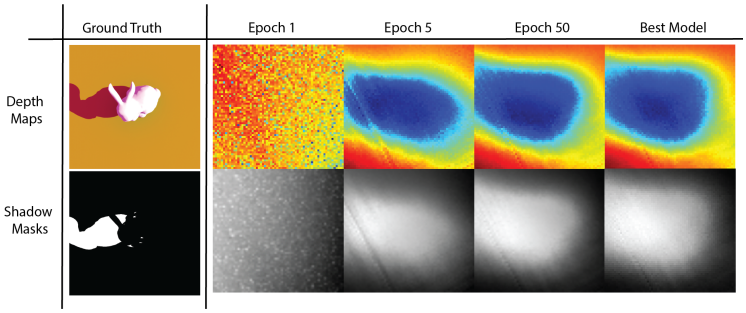

Fig. A.4. Predicted depth maps and shadows masks during training on the bunny's validation poses. Even though the model is able to learn the greater shape and localize the bunny well through its shadow map, its poor performance is probably due to its complexity, curves surfaces and its protruding ears that can imply many different possible combinations of the underlying 3D shape.

mation. Additionally, we do want to note that extracting explicit meshes from implicit representations is an extremely difficult task which requires a lot of trial and error. Even when results look great for novel view synthesis with the RGB NeRF framework, marching cubes does not produce perfectly smooth meshes. Often these meshes have jagged edges and protruding elements. Moreover, this becomes a tremendous problem when there is more than one object, as we start to extract extremely noisy meshes. However, alignment of explicit meshes is one of the most accurate metrics for 3D reconstruction evaluation, which is why we use it as an evaluation metric.

A.4 Ablation Studies

We show in Figure 6 in the main text the final reconstructions from different experiments varying different parameters. We observe that starting from a smoothed-out shadow mask is critical for the algorithm to construct a coarse mesh and the algorithm almost never converges without smoothing applied to the initial binary shadow masks. This makes sense as the gradients are now non-zero around the edges of the binary mask, and can help guide the network in constructing a mesh that is consistent with all the shadow images. Smoothing techniques have also been used in [**?**] and [**?**] where the goal was to recover 3D mesh from a single silhouette image. Since the light is fixed in the scene and unchanged, the amount of information available about the object's geometry is also fixed. The varying camera positions provide different viewpoints to the object and help the algorithm differentiably carve away that space. We note that even with five shadow masks our algorithm is able to reconstruct a coarse object, however it is not able to carve away the right and the left parts of the vase, which we believe is due to the lack of viewpoints from those poses. We note that 200

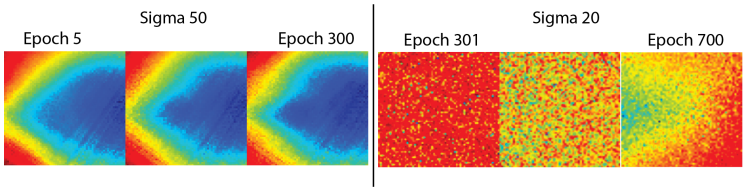

Fig. A.5. Drastically changing distance transform weight during training for the Chair Object. This figure shows the challenge in using shadows with a gradient based technique. Even though we reduce the weight during training, drastically reducing it can cause the model to unlearn the scene representation as the information it relied on converge to the object is now missing.

views of the object means that some positions are sampled more than once, and is more than enough to help the algorithm carve out the unnecessary elements in the mesh. We also show that varying the distance transform weight from 150 to 50, as shown in Figure 6 for the vase, can create crisper depth maps and faster convergence. However, this also leads to overfitting to the shadow mask since exactly fitting a mesh to the shadow does not necessarily translate to inferring the actual underlying shape of the object due to fundamental ill-posedness of the problem.

In Figure A.5 we also show drastically reducing the distance transform weight from 50 to 20 can cause the model to unlearn the scene representation. This shows how little gradient information shadows contain and that the model needs to be consistently nudged in order for it to be guided to reconstruct the underlying mesh.