3D Scene Part Decomposition

Jake Austin

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-104
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-104.html

May 15, 2024

Acknowledgement

An enormous thank you to professors Kanazawa and Efros, who both radically altered the trajectory of my life for the better with their kindness, openness, encouragement, and mentorship. It’s been the joy and honor of a lifetime to be able to work with them and learn from them.

3D Scene Part Decomposition

by Jake Austin

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Angjoo Kanazawa
Research Advisor

Date:

Signature: [illegible]

* * * * * * *

Professor Alexei Efros
Second Reader

Date: 5/9/2024

Signature: DocuSigned by: Alexei Efros
9D33F2A3CF02498...

3D Scene Part Decomposition

Jake Austin

May 13, 2024

Abstract

3D scene reconstruction for static scenes is a challenging and difficult problem. There has been a rise in highly unconstrained neural 3D representations like NeRF which use neural networks as black box functions to model the density and color of the scene, with no constraints on geometry or feasibility. While the expressive nature of NeRFs can lead to high visual fidelity, it can also lead to overfitting and poor geometry. In this work, we detail a number of avenues for attempting to constrain the expressive nature of modern 3D reconstruction methods, through decomposing the scene into simpler basic components. We experiment with novel representations and seek to exploit the inductive bias that the world around us is simply a composition of basic shapes and parts, and examine the pros and cons of such a strong inductive bias. We first attempt to regularize geometry by constraining not the individual shapes, but rather constrain that the shapes be reusable, experimenting with neural field representations. We then explicitly regularize geometry as being a composition of simple parts and simple geometric shapes.

Contents

Chapter 1

Introduction

3D scene reconstruction and creation is a difficult and challenging problem even for humans tasked with modeling the 3D world. While highly unconstrained neural 3D representations have taken off for their potential for high visual fidelity, they ignore inductive biases that humans use when modeling. When a human artist models a scene or an object, the process usually involves starting from basic shapes and composing them to form more complex shapes and geometry. A water bottle can be represented by two cylinders, one for the lid and one for the bottle itself, a table is just a rectangular platform and four cylindrical legs. At a high level, a scene is also just a composition of objects. A room might be a table composed with a water bottle and whatever other objects are sitting on the table. The real world also features repeated objects being composed, say if there are multiple water bottles of similar shape and size sitting on the table.

The following inductive biases are used by humans in modeling a scene by hand that we seek to bring into our representations when we reconstruct the world

1. Scenes are composed of (repeated) objects and parts
2. Objects are composed of basic shapes

All of these inductive biases should be represented in the way that our algorithms reconstruct 3D scenes. Our representations in the automated 3D reconstruction process should explicitly feature a composition of more primitive components. This is the primary aim of this work that we seek to explore. We discuss the effectiveness of this inductive bias in the following experiments

1. NeRF based reconstruction that allows for unconstrained geometry and explicit reuse of our learnt shapes
2. Mesh based reconstruction that only allows for basic shapes to be composed in scene reconstruction

In all of these experiments, we follow the same test time optimization setup as NeRF [52], where the input is a set of posed images and we optimize our representation through differentiable rendering. In all cases, a photometric loss will be used, and sometimes additional loss terms will be applied.

In the end, we will try to identify a set of minimum conditions for convergence of primitive based methods.

Chapter 2

Related Works

2.1 Multi-view Stereo

Our work can be seen as an end-to-end primitive-based approach to multi-view stereo (MVS), whose goal is to output a 3D reconstruction from multiple images taken from known camera viewpoints. We refer the reader to [26, 17] for an exhaustive review of classical methods. Recent MVS works can be broadly split into two groups.

Modular multi-step approaches typically rely on several processing steps to extract the final geometry from the images. Most methods [89, 19, 80, 81, 86, 24, 72], including the widely used COLMAP [70], first estimate depth maps for each image (through keypoint matching [70] or neural network predictions [80, 81, 86, 24, 72]), then apply a depth fusion step to generate a textured point cloud. Finally, a mesh can be obtained with a meshing algorithm [35, 40]. Other multi-step approaches directly rely on point clouds [18, 40] or voxel grids [71, 39, 30, 57]. Note that, although works like [30, 57] leverage end-to-end trainable networks to regress the geometry, we consider them as multi-step methods as they still rely on a training phase requiring 3D supervision before being applied to unknown sets of multi-view images. Extracting geometry through multiple steps involves careful tuning of each stage, thus increasing the pipeline complexity.

End-to-end approaches directly optimize a 3D scene representation using photometric consistency across different views along with other constraints in an optimization framework. Recent methods use neural networks to implicitly represent the 3D scene, in the form of occupancy fields [59], signed distance functions [83] or radiance fields, as introduced in NeRF [52]. Several works incorporate surface constraints in neural volumetric rendering to further improve the scene geometry [60, 82, 78, 10, 16], with a quality approaching that of traditional MVS methods. Other methods [20, 85, 22, 56] instead propose to leverage recent advances in mesh-based differentiable rendering [51, 34, 47, 9, 65, 41] to explicitly optimize textured meshes. Compared to implicit 3D representations, meshes are highly interpretable and are straightforward to use in computer graphic pipelines, thus enabling effortless scene editing and simulation [56]. However, all the above approaches represent the scene as a single mesh, making it ill-suited for manipulation and editing. We instead propose to discover the primitives that make up the scene, resulting in an interpretable and actionable representation. A concurrent work PartNeRF [75] introduces parts in NeRFs. However, only synthetic scenes with a single object are studied and the discovered parts mostly correspond to regions in the 3D space rather than interpretable geometric primitives.

2.2 Static Scenes

Scene decomposition into 3D primitives. The goal of understanding a scene by decomposing it into a set of geometric primitives can be traced back to the very fist computer vision thesis by Larry Roberts on *Blocks World* [66] in 1963. In it, Roberts shows a complete scene understanding system for a simple closed world of textureless polyhedral shapes by using a generic library of polyhedral block components. In the 1970s, Binford proposes the use of Generalized Cylinders as general primitives [3], later refined by Biederman into the recognition-by-components theory [2]. But applying these ideas to real-world image data has proved rather difficult.

A large family of methods does not consider images at all, instead focusing on finding primitives in 3D data. Building upon the classical idea of RANSAC [14], works like [5, 7, 69, 68, 45, 58, 64] accurately extract various primitive shapes (*e.g.*, planes, spheres and cylinders for [69, 68, 45]) from a point cloud. In particular, MonteBoxFinder [64] is a recent RANSAC-based system that robustly extracts cuboids from noisy point clouds by selecting the best proposals through Monte Carlo Tree Search. To avoid the need for RANSAC hyperparameter tuning while retaining robustness, Liu *et al.* [48] introduce a probabilistic framework dubbed EMS that recovers superquadrics [1]. Other methods instead leverage neural learning advances to robustly predict primitive decomposition from a collection of shapes (*e.g.*, ShapeNet [6]), in the form of cuboids [76], superquadrics [63, 61, 79], shapes from a small dictionary [44, 42] or learnable prototypical shapes [12, 62, 49]. However, they are typically limited to shapes of known categories and require perfect 3D data. AutoSDF uses an auto-regressive transformer to predict shape primitives to complete a shape, where each primitive is an SDF, and all primitives are learnt and exist in a codebook, allowing shapes to be reused. More generally, the decomposition results of all 3D-based methods highly depend on the quality of the 3D input, which is always noisy and incomplete for real scenes. For a complete survey of 3D decomposition methods, we refer the reader to [32].

More recently, there has been a renewed effort to fit 3D primitives to various image representations, such as depth maps, segmentation predictions or low-level image features. Attend infer repeat [13] is an early example of this where in the 2D case, we are attempting to reconstruct multi-MNIST digits by reconstructing them one at a time, predicting the next primitive (a digit) that is then subtracted from the image, repeating the process again, applying this to meshes in 3D scenes as well. CMR [33] or U-CMR [23] have a pre-computed mean shape for different classes, a mean shape that is transferred / used as a starting point for mesh deformation that will give us our specific instance, but the idea of using a mean shape across different instances is very much related, especially since deformations are leant from only images. Depth-based approaches [31, 15, 46, 21, 37] naturally associate a 3D point cloud to each image which is then used for primitive fitting. However, the resulting point cloud is highly incomplete, ambiguous and sometimes inaccurately predicted, thus limiting the decomposition quality. Building upon the single-image scene layout estimation [27, 28], works like [25, 43] compute cuboids that best match the predicted surface orientations. Façade [11], the classic image-based rendering work, leverages user annotations across multiple images with known camera viewpoints to render a scene with textured 3D primitives. In this work, we *do not* rely on 3D, depth, segmentation, low-level features, or user annotations to compute the 3D decomposition. Instead, taking inspiration from Façade [11] and recent multi-view modeling advances [77, 59, 52], our approach only requires calibrated views of the scene and directly optimizes textured primitives through photometric consistency in an end-to-end fashion. That is, we solve the 3D decomposition and multi-view stereo problems simultaneously.

Most closely related to our experiments on unconstrained shape repetition are Mixture of Volumetric Primitives [50], Nerflets [88] and AutoSDF [53], all of which are entirely data driven in that the shape primitives have unconstrained geometry within the bounding box. Mixture of Volumetric Primitives and Nerflets use composed implicit fields each transformed by a rotation and translation. However, these primitives aren't reusable, and these method relies on strong initialization of the primitives, whereas in this work we will only use random initialization of primitive poses.

Single Scene Codebook Learning Codebook learning has also seen great success in the case of Variable Bitrate Neural Fields [73]. This paper puts a strong prior on repetition and reuse of color and local shape throughout the scene, however the grid interpolation provides no strong inductive bias towards large scale shape regularization and reuse. A strong direction for future work will be in unifying our approach with this paper to provide strong inductive bias about local texture and color in addition to the strong inductive bias about large scale structure and shape which we pursue in this work. vMAP [38] even comes close as to explicitly represent a scene in with radiance fields per-object in the scene, although detection of objects is not done in a learnt way but instead through SLAM.

Chapter 3

Composable NeRFs

3.1 Problem Statement

In this set of experiments, we want to test the first inductive bias to see if allowing arbitrary geometry, but using a representation which explicitly contains repeated parts has an effect on reconstruction quality. To test this, we choose to represent the world with composable, posed NeRFs, each of which is a linear combination of a set of basis NeRFs. In this way, because our set of basis shapes are NeRFs, we can represent arbitrary geometry and shapes, but because each placed NeRF in the scene is a linear combination of our basis shapes, we have a way to represent repeated components.

3.2 Method

We will start with some definitions to be used over this chapter.

- Primitive: A primitive is some small radiance field that only outputs for queries that are inside the 0-1 aabb box
- Basis / Codebook: The set of primitives
- Object: A linear combination of primitives that has been placed in the scene according to a (randomly initialized) parameterized pose transformation
- BaseRF: Our set of objects which when composed make an entire scene

The parameters associated with our codebook are our individual volumetric primitives themselves. This can be a single hashgrid with an MLP, a set of triplanes, or any other NeRF representation that has an aabb box.

The parameters associated with an object are 1) the coefficients on the linear combination for scaling primitives in our basis (one linear combination for shape $(S_1, S_2, ...)$ and one linear combination for texture $(T_1, T_2, ...)$) 2) 6DOF pose (translation $\Delta x, \Delta y, \Delta z$, rotation θ, ϕ, ψ) + per-axis scale parameterization

(X, Y, Z) that determine the pose / size of the object 3) an existence coefficient between $[0, 1]$ that scales the density, allowing an object to turn itself on or off. We also ablate using distributions to represent object poses instead of explicitly representing pose, using reparameterized samples to get gradients onto the parameters of the distribution.

We will always have as many or more objects in a scene than primitives.

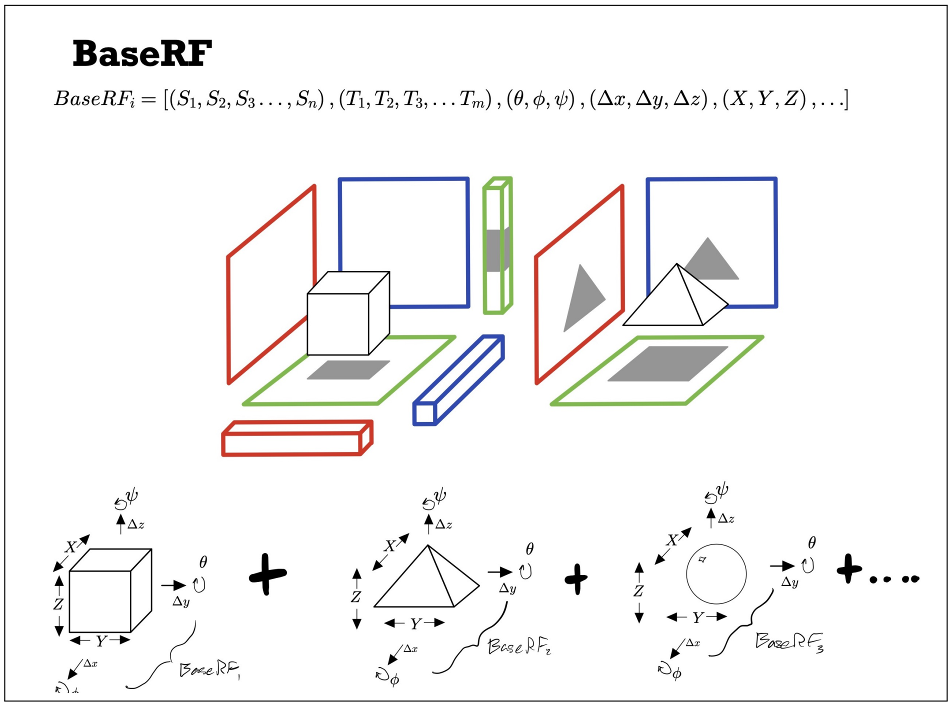

BaseRF
$BaseRF_i = [(S_1, S_2, S_3 \ldots, S_n), (T_1, T_2, T_3, \ldots T_m), (\theta, \phi, \psi), (\Delta x, \Delta y, \Delta z), (X, Y, Z), \ldots]$
ψ
Δz
X
Z
Y
θ
Δy
Δx
ϕ
BaseRF$_1$
BaseRF$_2$
BaseRF$_3$
+
+
+....

Figure 3.1: Each shape primitive is backed by a NeRF of some kind, be it a triplane or a set of 3 TensoRF [8] components, or a hash grid [55]. These objects can be placed around the scene multiple times, each with a unique pose and axis aligned stretch, and the implicit functions are composed where they overlap.

3.2.1 Method Notation

Our primary objective function is the same as NeRF [52], we are simply regressing onto the rendered RGB of the *BaseRF*, where we use the same volume rendering over samples σ and $\overline{c}$ along a ray. As we will demonstrate, all our operations are simple and therefore fully differentiable, allowing us to use gradient descent to backprop onto all our parameters. The uniqueness of this method comes from how we calculate each σ and $\overline{c}$ at each point along the ray.

Say we want to query our model $BaseRF(\overline{x})$ for the density σ and color $\overline{c}$ at a location $\overline{x_{query}}$. We first transform $\overline{x_{query}}$ into each of our object's coordinate frames

$$\overline{x_{query}}^i = OC(\overline{x_{query}}, \phi_i, \theta_i, \psi_i, \Delta x_i, \Delta y_i, \Delta z_i, X_i, Y_i, Z_i) \tag{3.1}$$

where OC is the transform operation that takes in the pose parameters associated with the ith object, taking us into object coordinates from world coordinates; rotating, translating, and re-scaling the world coordinate into the object's frame. Now we have the location of where to query each one of our objects $BaseRF_i$.

Our objects $BaseRF_i$ are a linear combination of each of the NeRFs in our basis / codebook, allowing us to smoothly change an object between shapes in our codebook. To get the density and color σ_i and $\overline{c_i}$ of object $BaseRF_i$, we need to linearly combine the n density fields and m color fields in our basis / codebook at location $\overline{x_{query}}^i$ which is in object coordinates. We do this directly, using our shape and texture weights for $BaseRF_i$. This gives the object $BaseRF_i$ a color and density of

$$\sigma_i = \sum_{j=0}^{n} S_j^i \sigma_j^i \quad \overline{c_i} = \sum_{k=0}^{m} T_k^i \overline{c}_k^i \tag{3.2}$$

where σ_j^i is the value of the jth density field in the codebook queried at $\overline{x_{query}}^i$, $\overline{c}_k^i$ is the value of the kth color field in the codebook queried at $\overline{x_{query}}^i$, S_j^i are our learnt linear combination weights for the basis density fields making up object i, and T_k^i are our learnt linear combination weights for the basis color fields making up object i. In practice, we set $j = k$, giving you the same number of basis density and color fields, allowing us to treat pairs of density and color fields as a NeRF, giving

$$\sigma_j^i, \overline{c}_j^i = NeRF_j(\overline{x_{query}}^i) \tag{3.3}$$

where $NeRF_j$ is the jth basis NeRF in our codebook. You can allow the linear combination coefficients to be unbounded and negative, but we find that in testing this causes the model to diverge to being empty, so we softmax over the coefficients before using them in the linear combination.

Once we have the density and color of object $BaseRF_i$, σ_i and $\overline{c_i}$, we need to combine our i densities and colors of our objects, since these objects overlap in 3D space. The density σ of all our objects $BaseRF_i$ combined is simply

$$\sigma = \sum \alpha_i \sigma_i \tag{3.4}$$

where α_i is the existance coefficient of object i between [0,1]. We naively combine the colors based on the relative densities as

$$\overline{c} = \text{softmax}(\alpha_i \sigma_i)\overline{c_i} \tag{3.5}$$

When we ablate using distributions over poses instead of explicitly parameterizing the pose of each object, we simply exchange the $\phi_i, \theta_i, \psi_i, \Delta x_i, \Delta y_i, \Delta z_i$

in equation 3.1 with reparameterized samples from distributions whose parameters we are now optimizing. Specifically,

$$\phi_i, \theta_i, \psi_i = \text{SphericalNormal}(\overline{\gamma_{R_i}}).\text{rsample}() \tag{3.6}$$

$$\Delta x_i, \Delta y_i, \Delta z_i = \text{Normal}(\overline{\gamma_{\Delta_i}}).\text{rsample}() \tag{3.7}$$

where $\overline{\gamma_{R_i}}$ and $\overline{\gamma_{\Delta_i}}$ are the new parameters for the distributions that we are now updating with gradient descent instead of the explicit values we were updating before, backpropagating through the reparameterized samples $\phi_i, \theta_i, \psi_i, \Delta x_i, \Delta y_i, \Delta z_i$ we use in 3.1 onto these new parameters. We use Pyro's [4] implementations for reparameterized samples through a spherical normal distribution.

3.2.2 Method Analysis and Information.

Let us assume that a NeRF query of position p runs in $\mathcal{O}(1)$. For our primitive approach, we will assume we have a codebook of size c and n objects in the scene, in order to query point p, we will need to compose $\mathcal{O}(nc)$ primitive densities and colors. For each of the n objects we need to query, the point p needs to be transformed according to the object-specific 6DOF pose + scales before we can query the codebook, resulting in a lot of pose transforms. Every point in the resulting set of n points needs to be used to query each of the c primitives.

The $\mathcal{O}(nc)$ runtime of a single query is the major bottleneck in training. At test time, this can likely be baked into a single compressed NeRF or mesh, but it is the major factor for why this method is slow, and requires a smaller batch size than other methods. To get some performance back, we mask out and don't query primitives for points outside the aabb box which works fairly well, with the negative consequence of having this method's memory consumption change over time.

To combine primitives, we take a naive approach of simply adding the densities and colors. A more comprehensive follow up work will need to do better for the sake of having more interpret-able primitives, using a more complex NeRF composition technique akin to Nerflets [88].

3.3 Experiments

The high level idea is that we will be trying to examine the viability of primitives as a representation by using radiance fields. We will do this through the following experiments:

1. Experimenting with the NeRF representation of primitives: low rank vs unconstrained, hash grid with MLP vs dense grid, high resolution vs low resolution
2. Experimenting with reuse of said primitives: how does reuse effect the end quality vs not reusing, how does the ratio of objects total vs size of primitive set effect quality

3. Gradient Flow: We will experiment with learning parameterized pose distributions to explore the space and get better gradients, and with things like blurring for better pose gradients. We use the reparameterization trick to get gradients through samples of these distributions when we render
4. Rendering shortcuts: Can we approximate the rendering by only considering the closest objects in the render
5. Multi-Scene Training: training jointly one shape basis across many scenes

To do this, we will use a combination of blender data and simpler custom data with scenes that have duplicate objects scattered throughout. For testing 1-5 we will be trying to reconstruct common blender scenes like the lego bulldozer so that a comparison can be made between our method and other SOTA NeRF methods, giving us a nice view of how viable our method is in this landscape of high performing methods.

We will implement these experiments in nerfstudio [74], an open source NeRF framework.

Minimum Conditions for Convergence In a naive implementation without any regularizers, it is very common for primitives to shrink down to zero scale early on in training, or move out of sampling range as a way to minimize RGB loss on synthetic data, where our white background matches the default end of ray color. The following regularizers are required to prevent the reconstruction from degenerating early on in training. Failure to use these regularizers will result in a blank scene.

1. Out of bounds loss
2. Primitive border interpolation
3. Per axis + Per volume minimum scale loss

We will penalize the primitives for existing outside an aabb box of [-1, 1] for blender scenes that largely exist within this aabb box, just to avoid having primitives that escape the scene to minimize the rgb reconstruction loss since most of the pixels in blender synthetic scenes are the uniform white background. We penalize it as the L1 distance to the unit cube, with the loss being zero if it is inside the cube already. We found that without this, objects (especially when using distributions for pose) would often leave the scene and go outside of sampling range.

Primitive border interpolation: To encourage the primitives to grow, we will have the primitive's density linearly interpolated to zero in the outer 10% of the primitive on any side. We find that this is a make or break factor for keeping the primitives from shrinking down to scale of zero, giving stronger gradients to the scale parameters for the NeRF that encourage larger primitives.

Scale losses: To help avoid numerical instabilities with primitives that grow small as they also lower their existence parameter, we will penalize the model for having object scales lower than some minimum threshold.

As a final note, if you desire subtractive modeling (where objects can subtract density from each other) and would like existence to be between $[-1, 1]$, you will need an additional regularizer penalizing the system for having subtractive volumes. We found that early on in training, objects will minimize rgb loss by subtracting all density and color from the entire scene.

3.4 Results

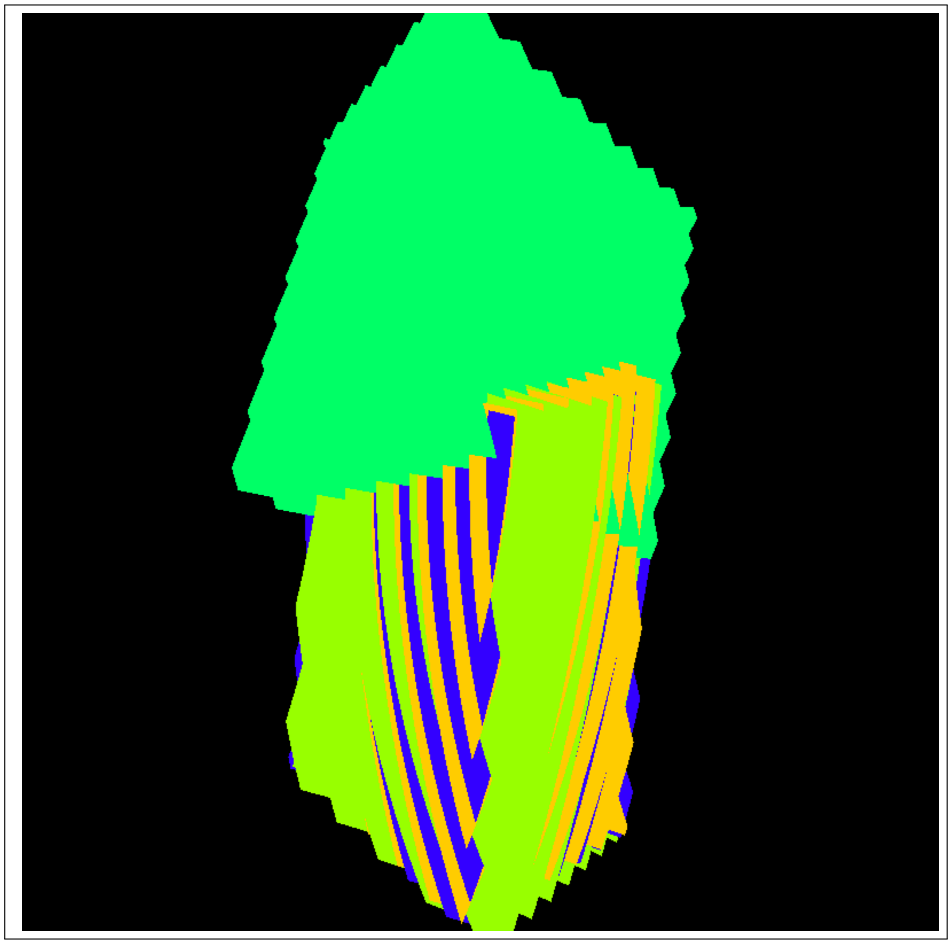


Figure 3.2: This figure shows the aabb boxes of all objects in our scene trained with no pose distribution with existence value greater than .9 . You can also see the issue of duplicate objects presented, as multiple objects clearly have aligned and are using the same linear combination of primitives.

In all, we were able to get this method working, but the general results are mixed. Training for a single scene on 30k iterations was 12 hours, with a codebook size of 4 and 30 objects in the scene. In general, with our best method (using instant-ngp hash grid backed primitives), we find that while reconstruction quality is faithful, the objects themselves are not individually very semantically meaningful or semantically singular (semantically singular meaning each object would correspond to one semantic idea). On our custom blender scene with basic shapes, we find that the most likely convergence is when one large object is learnt in the middle of the scene, a degenerate result considering that we are hoping for a natural scene decomposition to emerge. Fig. 3.4 shows the scene decomposition for the ficus plant scene, the primary benchmark we used due to its repeated fractal structure. Fig. 3.2 shows the aabb boxes of all objects with existance value of .9 or greater. This graphic is using a distribution for pose. At best, on this scene we can get 24.4 PSNR at best with no distribution for pose on this scene, which we show later, which is below most modern NeRF methods. Most objects in the scene turn themselves off or learn to duplicate another object (same linear combination of primitives and same location).

3.4.1 Rank Constraints

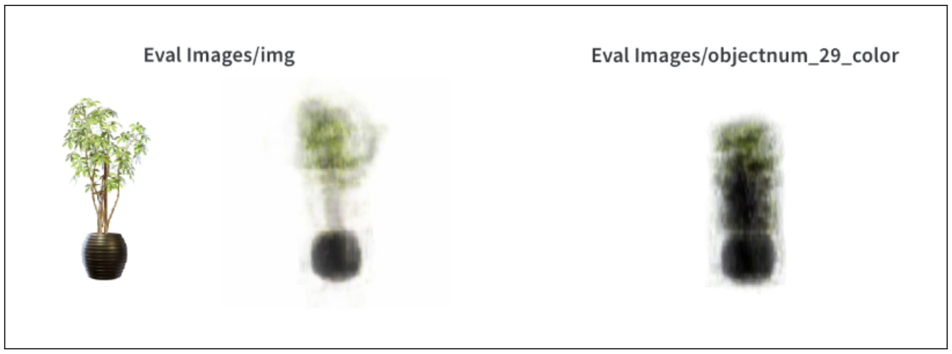


Figure 3.3: Left is ground truth, middle is our reconstruction using a pose distribution and tensorf backed components, and on the right is one prominent object in our scene. You can observe that the reconstruction is grainier (a common artifact in using pose distributions), and the component on the right is attempting to learn the entire plant in one shot.

We experimented with 2 primitive designs: one with triplanes (bringing inherent rank constraints) and a small MLP, and one with instant-NGP [55] hash grids and a small MLP. In general, we find that instant-NGP hashgrids have the fastest training time and give the best reconstruction quality. Fig. 3.3 shows the quality drop off when using triplanes, with the artifact of oversized and blocky primitives being a common artifact of using triplanes in this method.

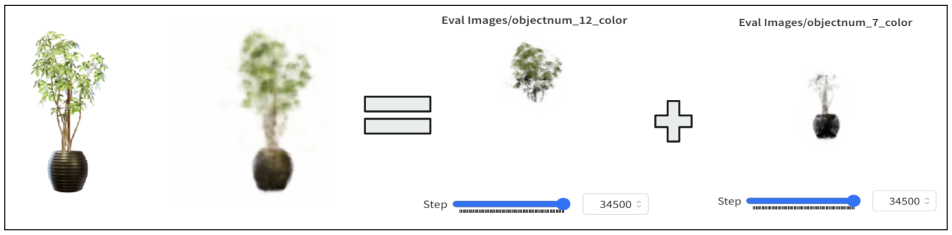


Figure 3.4: An example decomposition of the blender ficus scene. Two dominant objects are learnt by the codebook, whether we use the default of 4 basis NeRFs, or go down to 2 basis NeRFs as done in section 3.4.2.

3.4.2 Reuse of Primitives

We experimented with two setups: 1) constraining our codebook to only having 4 primitives, and 2) using one independent primitive per object, which reduces down to [50] but with random initialization. We find that using repeated components does not make a huge difference in what we converge to, due to our primitives growing to be fairly large in both cases. What we mean by this is rather than one object being some small repeatable shape, the primitives just grow to be the size of the entire region with repeatable geometry, rendering the ability to reuse primitives that is baked into our method unused. We believe this is a result of not having semantic constraints on the primitives themselves, allowing the scene to be reconstructed by as few as 2 objects without penalty. This choice may become relevant when you begin constraining the maximum scale of objects, though we chose not to do this in this work which is more concerned with establishing convergence with minimal constraints. Another choice would be to explicitly initialize the primitives sizes to be very small and their locations to be exactly where we know the geometry to be in 3D space, just as [50] does, but this is a heavy assumption we don't want to make as we are again concerned about using a minimal set of constraints and assumptions on the scene.

An example of the parsimony that emerges in the ficus scene regaurdless of the codebook size and number of objects can be seen in figure 3.4. In lots of different random initializations, no matter how many codes in the codebook or objects in the scene, this is the most common decomposition that emerges for this scene.

3.4.3 Gradient Flow

We experimented with explicitly parameterizing pose and parameterizing pose with distributions (spherical normal distribution for rotation, normal distribution for translation) using the reparameterization trick and reparameterized samples. We find that using a distribution is not necessary, nor do you need to blur the input images or rendered patches for this to work. Ideal gradient flow likely comes from 3x3 blurring (similar to what is used for differentiable mesh

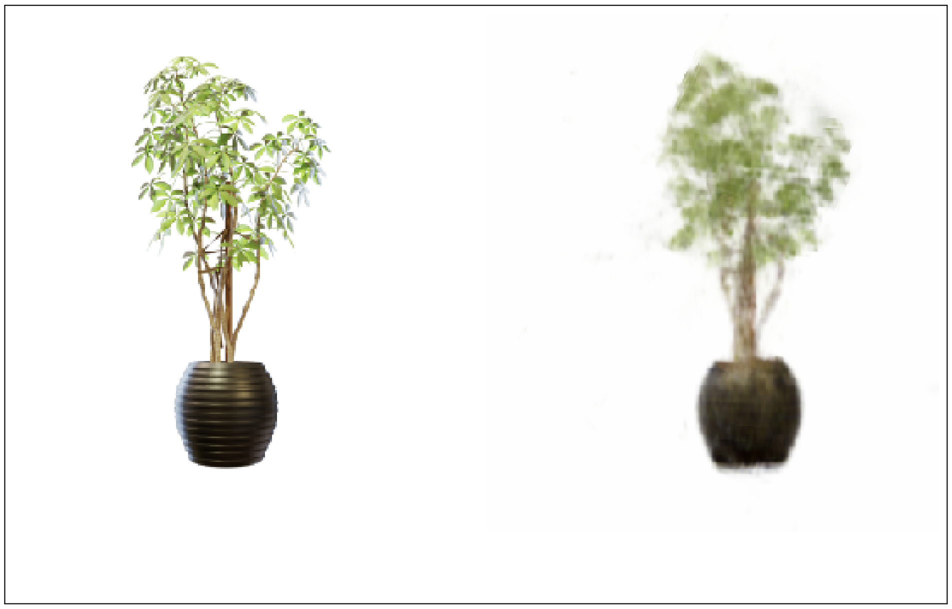


Figure 3.5: Left is ground truth, right is our reconstruction without a distribution backing the pose. You can see that without a distribution backing our poses, we get much crisper reconstructions, with branches being better formed and clearer compared to 3.3 The individual objects look to be about the same size and shape as when using the distributions, with one main object for leaves, and one main object for the pot and stem.

rasterization) of rendered patches, but in general this is not necessary. Not using a distribution for pose results in clearer scene texture. In general, using a distribution for pose is more robust to initialization, whereas directly estimating pose is a bit more brittle and has a higher likelihood of not converging to a good solution. The optimal solution is to use a distribution for the first half of training so the primitives can "explore" the highly non-convex optimization landscape that is optimal primitive pose prediction, and then using the mean pose explicitly instead of sampling during the second half of training after the distributions have settled down, so the texture no longer has to account for small changes in primitive location. Fig. 3.5 demonstrates this upper bound quality texture and accumulation obtained when not using pose distributions. Fig. 3.2 as shown earlier demonstrates that we do indeed get good object poses despite not using a distribution or blurring for good gradients.

3.4.4 Rendering Shortcuts

We experimented with only rendering the closest object to the camera, changing our $\mathcal{O}(nc)$ runtime down to $\mathcal{O}(c)$, making our method scale better with the

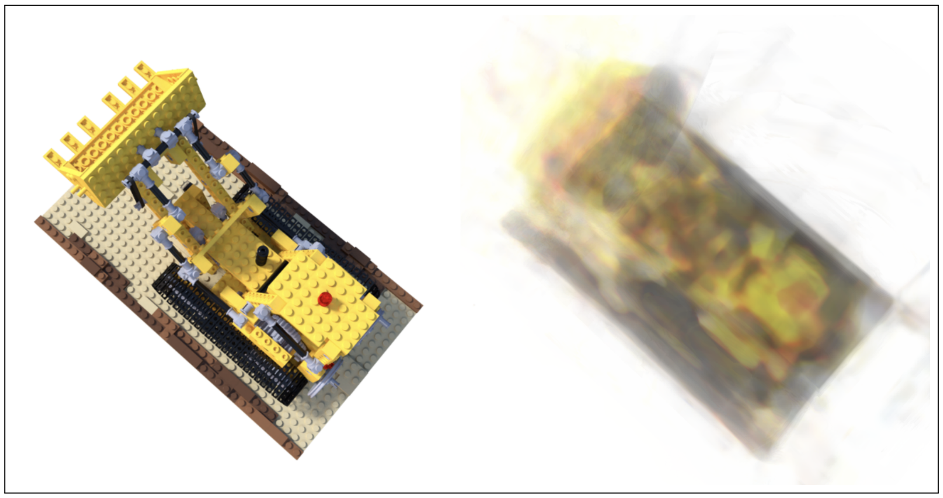


Figure 3.6: Our initial reconstruction results experimenting with rendering shortcuts, rendering only the component with the closest center.

number of objects in the scene. We find that this does not converge well, giving very blurry results, even if we are able to have orders of magnitude more primitives in our scene. Fig. 3.6 has an example of this blurry result.

3.4.5 Multi-Scene Training

Training jointly across multiple scenes was also not helpful. We trained one shared codebook across 2 custom synthetic blender scenes, both with the same set of basic shapes that have random poses. The hope was that this would regularize the codebook and prevent objects from learning to grow large, yielding a scene where one large object in the middle learns the entire scene. This was not what happened, and we still observed one object in the middle of the scene. Fig. 3.7 shows the results of this experiment on custom data. For multi-scene training to be effective, it would require many more scenes than you have primitives in your codebook to avoid the overfitting to specific scenes as we see here. In general on in the wild data, it may also not be the case that you have shared shapes as we had here, which makes this even harder.

3.5 High Level Takeaways

The minimum convergence conditions we identified for our method to work were

1. Out of bounds loss
2. Primitive border interpolation
3. Per axis + Per volume minimum scale loss

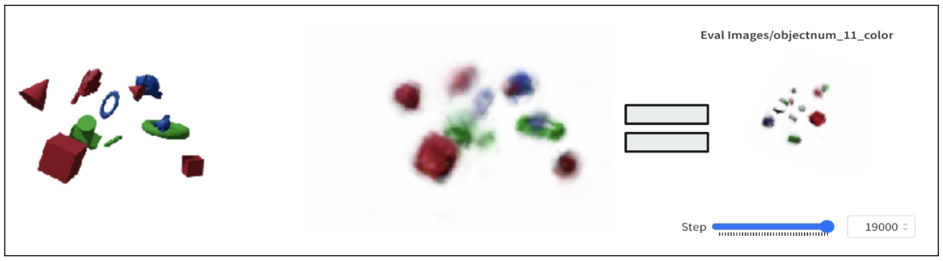


Figure 3.7: Custom data scene. From left to right: the ground truth, our reconstruction, and then one single object in our scene. There is no decomposition happening as the entire scene is just one object. The two scenes we jointly optimize over have the same set of shapes but in a different configuration.

and the most common artifacts that we found were

1. Primitive overlap
2. General lack of parsimony due to expressivity of primitives
3. Primitive overlap

Future work using highly unconstrained geometric primitives backed by implicit fields like NeRFs that don't use explicit 3D information as input to the system while training will require another prior like Segment Anything [36] in order to create a loss that will give gradients that explicitly encourage individual codes in the codebook to correspond to semantically similar concepts, unlike what we saw with the blender ficus plant, where one object would contain the pot of the plant and the stem which are semantically and geometrically very different.

The hypothesis was that the ability to represent repeated geometry would lead to learning the repeated shapes as elements in our codebook, but because of the expressivity of the codebook NeRFs, we found that it was still far too easy for the model to cheat and represent unrelated shapes or concepts as part of the same NeRF, and our hypothesis was disproved. In order to get parsimony where individual objects correspond to individual shapes and concepts, in the next chapter, we will use simpler geometric representations that don't have the expressivity of NeRFs as a means to extract more parsimonious scene decompositions. We will also use better regularization to prevent primitive overlap to the degree that we saw with this set of experiments.

Chapter 4

Meshes

4.1 Problem Statement

In this set of experiments, we want to simply test the second inductive bias, that when reconstructing a scene we should only allow geometry composed of basic shapes. To this end, we explicitly model the scene as a set of transparent superquadric meshes, whose parameters, texture and number are optimized to maximize photoconsistency through differentiable rendering. Note that compared to recent advances in neural volumetric representations [59, 52, 84], we *do not* use any neural network and directly optimize meshes, which are straightforward to use in computer graphic pipelines.

Introduced by Barr in 1981 [1] and revived recently by [63], superquadrics define a family of parametric surfaces that exhibits a strong expressiveness with a small number of continuous parameters, thus making a good candidate for primitive fitting by gradient descent. To be exact, a superquadratic has exactly 2 parameters controlling shape, which continuously deform the surface from cylinders to cubes to spheres, and other basic shapes like that. More concretely, we derive a superquadric mesh from a unit icosphere. For each vertex of the icosphere, its spherical coordinates $\eta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\omega \in [-\pi, \pi]$ are mapped to the superquadric surface through the parametric equation [1]:

$$\Phi(\eta, \omega) = \begin{bmatrix} s_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ s_2 \sin^{\epsilon_1} \eta \\ s_3 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \end{bmatrix}, \tag{4.1}$$

In this equation, s_1, s_2, s_3 are axis aligned scale parameters updated through gradient descent during training, and ϵ_1, ϵ_2 are the 2 learnt degrees of freedom controlling the superquadratic shape, updated also through gradient descent.

Superquadratics and differentiable mesh rendering allow us to explicitly represent a scene through basic shapes, and get gradients from an RGB loss onto the superquadratic parameters, pose, and texture, allowing us to test our hypothesis effectively.

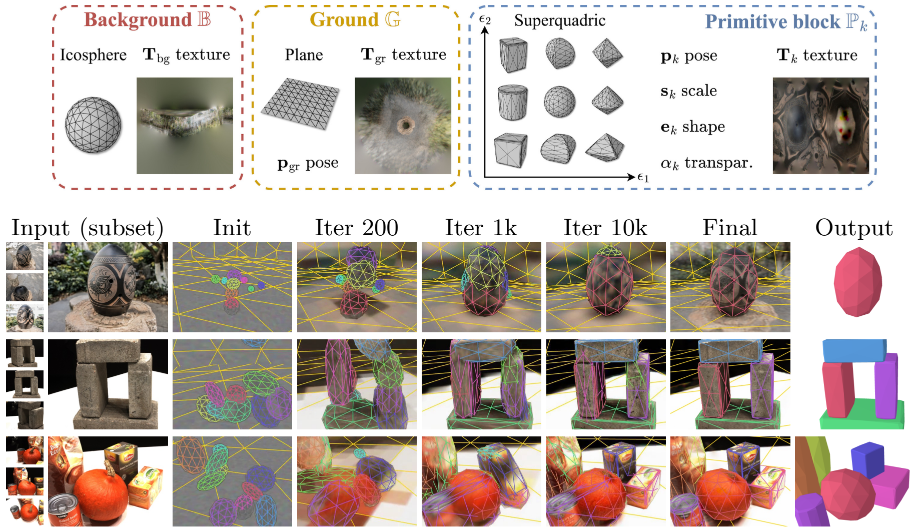


Figure 4.1: **Overview**. **(top)** We model the world as an explicit set of learnable textured meshes that are assembled together in the 3D space. **(bottom)** Starting from a random initialization, we optimize such a representation through differentiable rendering by photometric consistency across the different views.

Notations. In this chapter, we use bold lowercase for vectors (*e.g.*, $\mathbf{a}$), bold uppercase for images (*e.g.*, $\mathbf{A}$), double-struck uppercase for meshes (*e.g.*, $\mathbb{A}$) and write $a_{1:N}$ the ordered set $\{a_1, \ldots, a_n\}$.

4.2 Method

We propose to represent the world scene as an explicit set of textured meshes positioned in the 3D space. 4.1 summarizes our modeling and the parameters updated (top) during the optimization (bottom). Specifically, we model each scene as a union of primitive meshes: (i) an icosphere $\mathbb{B}$ modeling a background dome and centered on the scene, (ii) a plane $\mathbb{G}$ modeling the ground, and (iii) K primitive blocks $\mathbb{P}_{1:K}$ in the form of superquadric meshes, where K is fixed and refers to a maximum number of blocks. Unless mentioned otherwise, we arbitrarily use $K = 10$.

The goal of the background dome is to model things far from the cameras that can be well approximated with a planar surface at infinity. In practice, we consider an icosphere with a fixed location and a fixed scale that is much greater than the scene scale. On the contrary, the goal of the planar ground and the blocks is to model the scene close to the cameras. We thus introduce rigid transformations modeling locations that will be updated during the optimization. Specifically, we use the 6D rotation parametrization of [90] and associate to each block k a pose $\mathbf{p}_k = \{\mathbf{r}_k, \mathbf{t}_k\} \in \mathbb{R}^9$, transforming the vertices of the mesh. Similarly, we associate a rigid transformation $\mathbf{p}_{\text{gr}} = \{\mathbf{r}_{\text{gr}}, \mathbf{t}_{\text{gr}}\}$ to the ground plane.

Block existence through transparency. Modeling a variable number of primitives is a difficult task as it involves optimizing over a discrete random variable. Recent works tackle the problem using reinforcement learning [76], probabilistic approximations [63] or greedy algorithms [54], which often yield complex optimization strategies. In this work, we instead propose to handle variable number of primitive blocks by modeling meshes that are *transparent*. Specifically, we associate to each block k a learnable transparency value α_k, parametrized with a sigmoid, that can be pushed towards zero to change the effective number of blocks. Such transparencies are not only used in our rendering process to softly model the blocks existence and occlusions (4.2), but also in regularization terms during our optimization, *e.g.*, to encourage parsimony in the number of blocks used (4.2.1).

Texturing model. We use texture mapping to model scene appearance. Concretely, we optimize $K + 2$ texture images $\{\mathbf{T}_{\text{bg}}, \mathbf{T}_{\text{gr}}, \mathbf{T}_{1:K}\}$ which are UV-mapped onto each mesh triangle using pre-defined UV mappings. Textures for the background and the ground are trivially obtained using respectively spherical coordinates of the icosphere and a simple plane projection. For a given block k, each vertex of the superquadric mesh is associated to a vertex of the icosphere. Therefore, we can map the texture image $\mathbf{T}_k$ onto the superquadric by first mapping it to the icosphere using a fixed UV map computed with spherical coordinates, then mapping the icosphere triangles to the superquadric ones (see supplementary material for details).

Differentiable Rendering We leverage PyTorch3D [65] and their differentiable rasterization and rendering in order to differentiably render images of our scene. This works through soft rasterization and alpha compositing all the semi-transparent faces, giving gradients even to partially occluded objects.

4.2.1 Optimizing a Differentiable Blocks World

We optimize our scene parameters by minimizing a rendering loss across batches of images using gradient descent. Specifically, for each image $\mathbf{I}$, we build the scene mesh and use the associated camera pose to render an image $\hat{\mathbf{I}}$ using the differentiable mesh rendering engine [65]. We optimize an objective function defined as:

$$\mathcal{L} = \mathcal{L}_{\text{render}} + \lambda_{\text{parsi}}\mathcal{L}_{\text{parsi}} + \lambda_{\text{TV}}\mathcal{L}_{\text{TV}} + \lambda_{\text{over}}\mathcal{L}_{\text{over}} \ , \tag{4.2}$$

where $\mathcal{L}_{\text{render}}$ is a rendering loss between $\mathbf{I}$ and $\hat{\mathbf{I}}$, $\lambda_{\text{parsi}}, \lambda_{\text{TV}}, \lambda_{\text{over}}$ are scalar hyperparameters and $\mathcal{L}_{\text{parsi}}, \mathcal{L}_{\text{TV}}, \mathcal{L}_{\text{over}}$ are regularization terms respectively encouraging parsimony in the use of primitives, favoring smoothness in the texture maps and penalizing the overlap between primitives. Our rendering loss is composed of a pixel-wise MSE loss $\mathcal{L}_{\text{MSE}}$ and a perceptual LPIPS loss [87]

$\mathcal{L}_{\text{perc}}$ such that $\mathcal{L}_{\text{render}} = \mathcal{L}_{\text{MSE}} + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}}$. In all experiments, we use $\lambda_{\text{parsi}} = 0.01, \lambda_{\text{perc}} = \lambda_{\text{TV}} = 0.1$ and $\lambda_{\text{over}} = 1$.

Encouraging parsimony and texture smoothness. We found that regularization terms were critical to obtain meaningful results. In particular, the raw model typically uses the maximum number of blocks available to reconstruct the scene, thus over-decomposing the scene. To adapt the number of blocks per scene and encourage parsimony, we use the transparency values as a proxy for the number of blocks used and penalize the loss by $\mathcal{L}_{\text{parsi}} = \sum_k \frac{\sqrt{\alpha_k}}{K}$. We also use a total variation (TV) penalization [67] on the texture maps to encourage uniform textures. Given a texture map $\mathbf{T}$ of size $U \times V$ and denoting by $\mathbf{T}[u, v] \in \mathbb{R}^3$ the RGB values of the pixel at location (u, v), we define:

$$\mathcal{L}_{\text{tv}}(\mathbf{T}) = \frac{1}{UV} \sum_{u,v} \Big(\big\|\mathbf{T}[u+1, v] - \mathbf{T}[u, v]\big\|_2^2 + \big\|\mathbf{T}[u, v+1] - \mathbf{T}[u, v]\big\|_2^2 \Big)\ , \quad (4.3)$$

and write $\mathcal{L}_{\text{TV}} = \mathcal{L}_{\text{tv}}(\mathbf{T}_{\text{bg}}) + \mathcal{L}_{\text{tv}}(\mathbf{T}_{\text{gr}}) + \sum_k \mathcal{L}_{\text{tv}}(\mathbf{T}_k)$ the final penalization.

Penalizing overlapping blocks. We introduce a regularization term encouraging primitives to not overlap. Because penalizing volumetric intersections of superquadrics is difficult and computationally expensive, we instead propose to use a Monte Carlo alternative, by sampling 3D points in the scene and penalizing points belonging to more than λ blocks, in a fashion similar to [62]. Following [62], λ is set to 1.95 so that blocks could slightly overlap around their surface thus avoiding unrealistic floating blocks. More specifically, considering a block k and a 3D point $\mathbf{x}$, we define a soft 3D occupancy function $\mathcal{O}_k^{\text{3D}}$ as:

$$\mathcal{O}_k^{\text{3D}}(\mathbf{x}) = \alpha_k \,\text{sigmoid}\Big(\frac{1 - \Psi_k(\mathbf{x})}{\tau}\Big)\ , \quad (4.4)$$

where τ is a temperature hyperparameter and Ψ_k is the superquadric inside-outside function [1] associated to the block k, such that $\Psi_k(\mathbf{x}) \leq 1$ if $\mathbf{x}$ lies inside the superquadric and $\Psi_k(\mathbf{x}) > 1$ otherwise. Given a set of M 3D points Ω, our final regularization term can be written as:

$$\mathcal{L}_{\text{over}} = \frac{1}{M} \sum_{\mathbf{x} \in \Omega} \max\Big(\sum_{k=1}^{K} \mathcal{O}_k^{\text{3D}}(\mathbf{x}),\ \lambda \Big)\ . \quad (4.5)$$

Note that in practice, for better efficiency and accuracy, we only sample points in the region where blocks are located, which can be identified using the block poses $\mathbf{p}_{1:K}$.

Optimization details. We found that two elements were key to avoid bad local minima during optimization. First, while transparent meshes enable differentiability w.r.t. the number of primitives, we observed a failure mode where two semi opaque meshes model the same 3D region. To prevent this behavior, we propose to inject gaussian noise before the sigmoid in the transparencies $\alpha_{1:K}$ to create stochasticity when values are not close to the sigmoid saturation, and thus encourage values that are close binary. Second, another failure mode we observed is one where the planar ground is modeling the entire scene. We

Table 4.1: **Quantitative results on DTU [29].** We use the official DTU evaluation to report Chamfer Distance (CD) between 3D reconstruction and ground-truth, **best** results are highlighted. We also highlight the average number of primitives found (#P) in green (smaller than 10) or red (larger than 10). Our performances correspond to a single random run (random) and a run automatically selected among 5 runs using the minimal rendering loss (auto). We augment the best concurrent methods with a filtering step removing the ground from the 3D input.

		Chamfer Distance (CD) per scene										Mean	Mean
Method	Input	S24	S31	S40	S45	S55	S59	S63	S75	S83	S105	CD	#P
EMS [48]	NeuS-mesh	8.42	8.53	7.84	6.98	7.2	8.57	7.77	8.69	4.74	9.11	7.78	9.6
EMS [48]	3D GT	6.77	5.93	3.36	6.91	6.52	3.50	4.72	7.08	7.25	6.10	5.82	7.4
MBF [64]	NeuS-mesh	3.97	4.28	3.56	4.76	3.33	3.92	3.63	5.58	5.3	6.07	4.44	53.5
MBF [64]	3D GT	3.73	4.79	4.31	3.95	3.26	4.00	3.66	3.92	3.97	**4.25**	3.98	16.4
Ours (random)	Image	5.41	**3.13**	1.57	4.93	3.08	3.66	**3.40**	2.78	3.94	4.85	3.67	**4.6**
Ours (auto)	Image	**3.25**	**3.13**	**1.16**	**3.02**	**2.98**	**2.32**	**3.40**	**2.78**	**3.43**	5.21	**3.07**	5.0
EMS [48] + filter	3D GT	6.32	4.11	2.98	4.94	4.26	3.03	3.60	5.44	3.24	4.43	4.23	8.3
MBF [64] + filter	3D GT	**3.35**	**2.95**	**2.61**	**2.19**	**2.53**	**2.47**	**1.97**	**2.60**	**2.60**	**3.27**	**2.65**	29.9

avoid this by leveraging a two-stage curriculum learning scheme, where texture maps are downscaled by 8 during the first stage.

4.3 Experiments

Benchmark details. DTU [29] is an MVS dataset containing 80 forward-facing scenes captured in a controlled indoor setting, where the 3D ground-truth points are obtained through a structured light scanner. We evaluate on 10 scenes (S24, S31, S40, S45, S55, S59, S63, S75, S83, S105) that have different geometries and a 3D decomposition that is relatively intuitive. We use standard processing practices [83, 82, 10], resize the images to 400×300 and run our model with $K = 10$ on all available views for each scene (49 or 64 depending on the scenes). We use the official evaluation presented in [29], which computes the Chamfer distance between the ground-truth points and points sampled from the 3D reconstruction, filtered out if not in the neighborhood of the ground-truth points. We evaluate two state-of-the-art methods for 3D decomposition, EMS [48] and MonteboxFinder (MBF) [64], by applying them to the 3D ground-truth point clouds. We also evaluate them in a setup comparable to ours, where the state-of-the-art MVS method NeuS [78] is first applied to the multi-view images to extract a mesh, which is then used as input to the 3D decomposition methods. We refer to this input as "NeuS-mesh".

4.4 Results

4.4.1 Benchmark Results.

We compare our Chamfer distance performances to these state-of-the-art 3D decomposition methods in 4.1. For each method, we report the input used and highlight the average number of discovered primitives #P in green (smaller than 10) or red (larger than 10). Intuitively, overly large numbers of primitives lead to less intuitive and manipulative scene representations. Our performances correspond to a single random run (random) and a run automatically selected among 5 runs using the minimal rendering loss (auto). We augment the best concurrent methods with a filtering step using RANSAC to remove the planar ground from the 3D input. Overall, we obtain results that are much more satisfactory than prior works. On the one hand, EMS outputs a reasonable number of primitives but has a high Chamfer distance reflecting bad 3D reconstructions. On the other hand, MBF yields a lower Chamfer distance (even better than ours with the filtering step) but it outputs a significantly higher number of primitives, thus reflecting over-decompositions.

Our approach is qualitatively compared in 4.2 to the best EMS and MBF models, which correspond to the ones applied on the 3D ground truth and augmented with the filtering step. Because the point clouds are noisy and incomplete (see 360° renderings in our supplementary material), EMS and MBF struggle to find reasonable 3D decompositions: EMS misses some important parts, while MBF over-decomposes the 3D into piecewise planar surfaces. On the contrary, our model is able to output meaningful 3D decompositions with varying numbers of primitives and very different shapes. Besides, ours is the only approach that recovers the scene appearance (last column). Also note that it produces a complete 3D scene, despite being only optimized on forward-facing views.

4.4.2 Influence of K and λ_{parsi}.

In 4.2, we evaluate the impact of two key hyperparameters of our approach, namely the maximum number of primitives K and the weight of the parsimony regularization λ_{parsi}. Results are averaged over the 10 DTU scenes for 5 random seeds. First, we can observe that increasing K slightly improves the reconstruction and rendering performances at the cost of a higher effective number of primitives. Second, these results show that λ_{parsi} directly influences the effective number of primitives found. When $\lambda_{\text{parsi}} = 0.1$, this strong regularization limits the reconstruction to roughly one primitive, which dramatically decreases the performances. When λ_{parsi} is smaller, the effective number of primitives increases without significant improvements in the reconstruction quality.

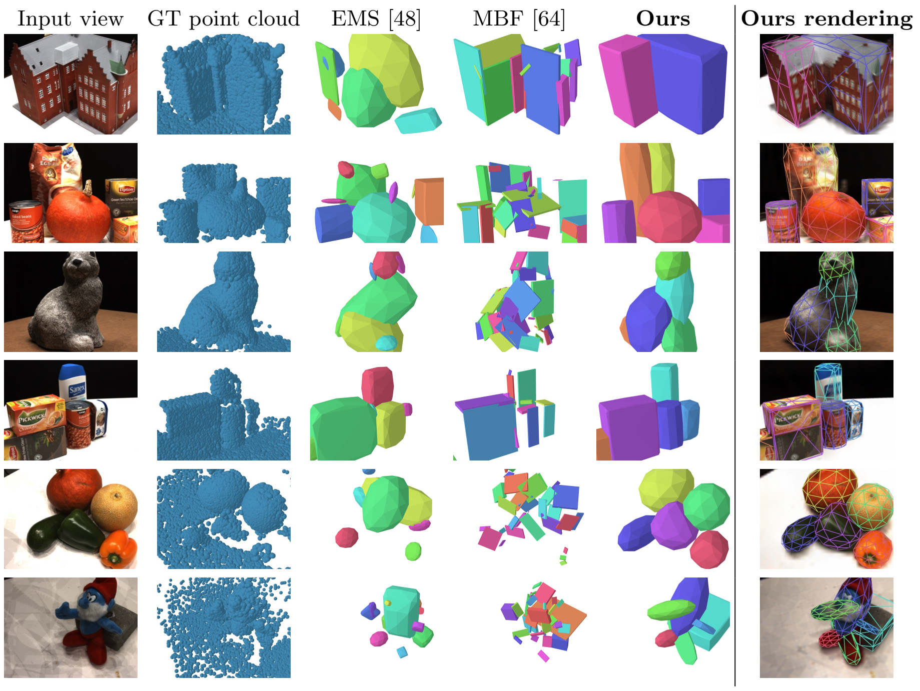
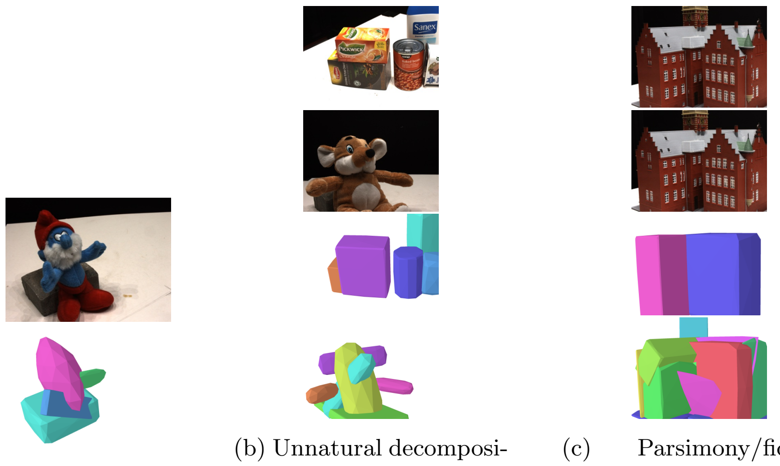


Figure 4.2: **Qualitative comparisons on DTU [29].** We compare our model to state-of-the-art methods (augmented with a preprocessing step to remove the 3D ground) which, unlike ours, find primitives in the ground-truth point cloud that is noisy and incomplete. Additionally, our approach is the only one able to capture the scene appearance (last column).

Table 4.2: **Effect of hyperparameters on DTU [29].** We evaluate the influence of two key hyperparameters of our model: the maximum number of primitives K **(left)** and the parsimony regularization λ_{parsi} **(right)**.

Method	#P↓	CD↓	PSNR↑	SSIM↑	LPIPS↓
$K = 10$ (default)	4.60	3.63	20.5	73.5	23.9
$K = 25$	7.00	3.58	21.0	74.6	22.5
$K = 50$	9.26	3.52	20.9	74.7	22.8

Method	#P↓	CD↓
$\lambda_{\text{parsi}} = 0.001$	7.44	3.61
$\lambda_{\text{parsi}} = 0.01$ (default)	4.60	3.63
$\lambda_{\text{parsi}} = 0.1$	1.30	6.88



(a) Missing parts (b) Unnatural decomposition (c) Parsimony/fidelity trade-off

Figure 4.3: **Failure cases.** We show typical failure cases of our approach. All models are optimized with $K = 10$ except the rightmost model which is optimized with $K = 50$. See text for details.

4.4.3 Limitations and failure cases.

In 4.3, we show typical failure cases of our approach. First, for a random run, we may observe bad solutions where parts of the geometry are not reconstructed (4.3a). This is mainly caused by the absence of primitives in this region at initialization and our automatic selection among multiple runs alleviates the issue, yet this solution is computationally costly. Note that we also tried to apply a Gaussian kernel to blur the image and propagate gradients farther, but it had little effect. Second, our reconstructions can yield unnatural decompositions as illustrated in 4.3b, where tea boxes are wrongly split or a single primitive is modeling the bear nose and the rock behind. Finally, in 4.3c, we show that increasing K from 10 (left) to 50 (right) allows us to trade-off parsimony for reconstruction fidelity. However, while this provides a form of control over the decomposition granularity, the ideal decomposition in this particular case does not seem to be found: the former seems to slightly under-decompose the scene while the latter seems to over-decompose it.

The other notable shortcoming of this method is in that we require multiple runs and select the best one by looking at the run with the lowest rendering loss. In practice, if you just take one run, you will often get a sub-optimal decomposition. This demonstrates the complexity of the optimization landscape and just how sensitive to initialization this method is, bringing back the potential need for off the shelf priors like Segment Anything [36] as a means to get better gradients and more consistent optimization that is less sensitive to initialization.

4.5 High Level Takeaways

The convergence conditions for higher quality convergence with this method are

1. TV Texture loss
2. Loss on total number of primitives active (use transparency for proxy)
3. Penalizing overlapping blocks
4. Adding noise to transparency

By more explicitly constraining the primitives to be simple shapes, we get better parsimony. By putting a loss on the total number of primitives, we can actually control just how expressive this method is without explicitly constraining the number of primitives at initialization.

All that being said, we do still find that the complexity of the optimization landscape and not all primitives yielded from this method in any one decomposition actually correspond with semantically meaningful concepts, furthering the need for future work to use off the shelf priors for stronger semantically meaningful grouping.

We do find that just by strongly constraining the expressivity of the geometry in contrast to other attempts with NeRFs, we are able to achieve better and more meaningful decompositions, but it still isn't enough to guarantee a meaningful decomposition.

Chapter 5

Takeaways for Primitive Based Reconstruction

In all, from using different representations for our primitives, we come to the following conclusions about primitive based approaches for the analysis-by-synthesis pipeline of using only posed images for optimizing a single scene reconstruction (as opposed to using 3D information like depth or pointclouds in the reconstruction and decomposition process).

1. Primitive overlap must be constrained
2. Simple primitive geometry (whether through representations that only allow simple geometry or regularizing the geometry) leads to better parsimony, but it is not enough to yield meaningful decompositions
3. The optimization landscape is complex and sensitive to initialization even with strong regularization
4. In general representations that enable parsimonious and repeatable parts to be explicitly represented (just by itself) is not enough to guarantee that your method yields decompositions that are broken down into meaningful parts or repeated parts

5.1 Constraining Overlap

We see in both methods qualitatively and quantitatively that it is almost a certainty that your primitives will attempt to overlap, and overlap of primitives needs to be regularized for decompositions. When you have a set number of primitives and multiple of them overlap and represent the same object in a purely redundant way, your model's maximum expressively is greatly diminished. This in itself is no guarantee of your method working, but is a best practice that needs to be followed for better quality.

5.2 Geometry Regularization

By simply switching from a NeRF based primitive to a superquadratic (ignoring the repeated parts idea from the first method), we get significantly nicer decompositions. With NeRFs, we would get arbitrary decompositions that didn't align with semantics OR basic geometric shapes, like how one primitive could represent the blender ficus stem and the plant's pot at the same time. While switching to superquadratics and their simple geometry is not a guarantee of semantic meaningfulness of the primitives, it does at least mean that our primitives will align with basic shapes in the scene which is a step in the right direction.

5.3 Optimization Landscape Complexity

We experimented with distributions over poses in the first method, and lots of regularizers in the second method, but in both methods, we observed a high sensitivity to initialization, where different random seeds for initialization would give very different final decompositions. In the first method, these decompositions would be purely random and not aligned with borders of different shapes, and in the second method, these decompositions wouldn't always align to the basic shapes. Occasionally, as seen in our failure modes you can get one large primitive that will take the place of what should ideally be many smaller basic shapes. For example, a set of boxes nearby each other will sometimes end up being represented by one large blob of a superquadratic which is a suboptimal result considering how easy and optimal it seems for the method to assign a rectangular prism superquadratic to each box. No matter how you slice it, the optimization landscape is very complex and photometric constraints alone aren't enough to guarantee you a parsimonious and accurate decomposition, no matter how much or how little you constrain the geometry, as we went from NeRFs which are highly expressive all the way to superquadratic meshes which are simplistic and minimally expressive.

5.4 Future Work

In general, no matter how much or how little you constrain the geometry, it simply is not enough to yield simple decompositions that make sense for a human. Because a human observer looking at the resulting decomposition sees that the decomposition is not along semantic lines or is missing important parts, or is assigning too many primitives to the wrong part of the scene that is least important, moving forward it may be necessary to use off the shelf priors that can do grouping of the input images (whether this be in a supervised manner like with [36] or an unsupervised 2D grouping algorithm), potentially providing more meaningful signal than photometric constraints alone. At the end of the day, geometric groupings of a 3D scene are not the same as or as meaningful as semantic groupings, which needs to be the direction of this line of work in order to get better, more pleasing, more meaningful for downstream applications, and more robust decompositions.

Bibliography

[1] Barr, A.H.: Superquadrics and Angle-Preserving Transformations. IEEE Computer Graphics and Applications (1981) 6, 19, 22

[2] Biederman, I.: Recognition-by-components: a theory of human image understanding. Psychological review (1987) 6

[3] Binford, T.: Visual Perception by Computer. In: IEEE Conference on Systems and Control (1971) 6

[4] Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., Goodman, N.D.: Pyro: Deep Universal Probabilistic Programming. Journal of Machine Learning Research (2018) 11

[5] Bolles, R.C., Fischler, M.A.: A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data. In: IJCAI (1981) 6

[6] Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. arXiv:1512.03012 [cs.CV] (2015) 6

[7] Chaperon, T., Goulette, F.: Extracting cylinders in full 3D data using a random sampling method and the gaussian image. In: VMV (2001) 6

[8] Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields (2022). https://doi.org/10.48550/ARXIV.2203.09517, https://arxiv.org/abs/2203.09517 9

[9] Chen, W., Gao, J., Ling, H., Smith, E.J., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In: NeurIPS (2019) 5

[10] Darmon, F., Bascle, B., Devaux, J.C., Monasse, P., Aubry, M.: Improving neural implicit surfaces geometry with patch warping. In: CVPR (2022) 5, 23

[11] Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In: SIGGRAPH (1996) 7

[12] Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: Learning elementary structures for 3d shape generation and matching. In: NeurIPS (2019) 6

[13] Eslami, S.M.A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., Hinton, G.E.: Attend, infer, repeat: Fast scene understanding with generative models (2016). https://doi.org/10.48550/ARXIV.1603.08575, https://arxiv.org/abs/1603.08575 6

[14] Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM (1981) 6

[15] Fouhey, D.F., Gupta, A., Hebert, M.: Data-Driven 3D Primitives for Single Image Understanding. In: ICCV (2013) 7

[16] Fu, Q., Xu, Q., Ong, Y.S., Tao, W.: Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction. In: NeurIPS (2022) 5

[17] Furukawa, Y., Hernández, C.: Multi-view stereo: A tutorial. Foundations and Trends® in Computer Graphics and Vision (2015) 5

[18] Furukawa, Y., Ponce, J.: Accurate, Dense, and Robust Multi-View Stereopsis. In: CVPR (2007) 5

[19] Galliani, S., Lasinger, K., Schindler, K.: Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In: ICCV (2015) 5

[20] Gao, J., Chen, W., Xiang, T., Tsang, C.F., Jacobson, A., McGuire, M., Fidler, S.: Learning Deformable Tetrahedral Meshes for 3D Reconstruction. In: NeurIPS (2020) 5

[21] Geiger, A., Wang, C.: Joint 3D Object and Layout Inference from a Single RGB-D Image. In: GCPR (2015) 7

[22] Goel, S., Gkioxari, G., Malik, J.: Differentiable Stereopsis: Meshes from multiple views using differentiable rendering. In: CVPR (2022) 5

[23] Goel, S., Kanazawa, A., Malik, J.: Shape and viewpoint without keypoints (2020). https://doi.org/10.48550/ARXIV.2007.10982, https://arxiv.org/abs/2007.10982 7

[24] Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., Tan, P.: Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching. In: CVPR (2020) 5

[25] Gupta, A., Efros, A.A., Hebert, M.: Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics. In: ECCV (2010) 7

[26] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2003) 5

[27] Hoiem, D., Efros, A., Hebert, M.: Geometric context from a single image. In: ICCV (2005) 7

[28] Hoiem, D., Efros, A.A., Hebert, M.: Recovering Surface Layout from an Image. IJCV (2007) 7

[29] Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanaes, H.: Large Scale Multi-view Stereopsis Evaluation. In: CVPR (2014) 23, 25

[30] Ji, M., Gall, J., Zheng, H., Liu, Y., Fang, L.: SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis. In: ICCV (2017) 5

[31] Jiang, H., Xiao, J.: A Linear Approach to Matching Cuboids in RGBD Images. In: CVPR (2013) 7

[32] Kaiser, A., Ybanez Zepeda, J.A., Boubekeur, T.: A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data. Computer Graphics Forum (2019) 6

[33] Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections (2018). https://doi.org/10.48550/ARXIV.1803.07549, https://arxiv.org/abs/1803.07549 7

[34] Kato, H., Ushiku, Y., Harada, T.: Neural 3D Mesh Renderer. In: CVPR (2018) 5

[35] Kazhdan, M., Hoppe, H.: Screened Poisson Surface Reconstruction. ACM Transactions on Graphics (2013) 5

[36] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything (2023) 18, 26, 29

[37] Kluger, F., Ackermann, H., Brachmann, E., Yang, M.Y., Rosenhahn, B.: Cuboids Revisited: Learning Robust 3D Shape Fitting to Single RGB Images. In: CVPR (2021) 7

[38] Kong, X., Liu, S., Taher, M., Davison, A.J.: vmap: Vectorised object mapping for neural field slam (2023). https://doi.org/10.48550/ARXIV.2302.01838, https://arxiv.org/abs/2302.01838 7

[39] Kutulakos, K., Seitz, S.: A Theory of Shape by Space Carving. In: ICCV (1999) 5

[40] Labatut, P., Pons, J.P., Keriven, R.: Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. In: ICCV (2007) 5

[41] Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular Primitives for High-Performance Differentiable Rendering. ACM Transactions on Graphics (2020) 5

[42] Le, E.T., Sung, M., Ceylan, D., Mech, R., Boubekeur, T., Mitra, N.J.: CPFN: Cascaded Primitive Fitting Networks for High-Resolution Point Clouds. In: ICCV (2021) 6

[43] Lee, D.C., Gupta, A., Hebert, M., Kanade, T.: Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces. In: NIPS (2010) 7

[44] Li, L., Sung, M., Dubrovina, A., Yi, L., Guibas, L.J.: Supervised Fitting of Geometric Primitives to 3D Point Clouds. In: CVPR (2019) 6

[45] Li, Y., Wu, X., Chrysanthou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: GlobFit: Consistently fitting primitives by discovering global relations. ACM Transactions on Graphics (2011) 6

[46] Lin, D., Fidler, S., Urtasun, R.: Holistic Scene Understanding for 3D Object Detection with RGBD Cameras. In: ICCV (2013) 7

[47] Liu, S., Li, T., Chen, W., Li, H.: Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In: ICCV (2019) 5

[48] Liu, W., Wu, Y., Ruan, S., Chirikjian, G.S.: Robust and Accurate Superquadric Recovery: a Probabilistic Approach. In: CVPR (2022) 6, 23, 25

[49] Loiseau, R., Vincent, E., Aubry, M., Landrieu, L.: Learnable Earth Parser: Discovering 3D Prototypes in Aerial Scans. arXiv:2304.09704 [cs.CV] (2023) 6

[50] Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., Saragih, J.: Mixture of volumetric primitives for efficient neural rendering (2021). https://doi.org/10.48550/ARXIV.2103.01954, https://arxiv.org/abs/2103.01954 7, 15

[51] Loper, M.M., Black, M.J.: OpenDR: An Approximate Differentiable Renderer. In: ECCV (2014) 5

[52] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020) 4, 5, 7, 9, 19

[53] Mittal, P., Cheng, Y.C., Singh, M., Tulsiani, S.: Autosdf: Shape priors for 3d completion, reconstruction and generation (2022). https://doi.org/10.48550/ARXIV.2203.09516, https://arxiv.org/abs/2203.09516 7

[54] Monnier, T., Vincent, E., Ponce, J., Aubry, M.: Unsupervised Layered Image Decomposition into Object Prototypes. In: ICCV (2021) 21

[55] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102:1–102:15 (Jul 2022). https://doi.org/10.1145/3528223.3530127, https://doi.org/10.1145/3528223.3530127 9, 14

[56] Munkberg, J., Chen, W., Hasselgren, J., Evans, A., Shen, T., Muller, T., Gao, J., Fidler, S.: Extracting Triangular 3D Models, Materials, and Lighting From Images. In: CVPR (2022) 5, 6

[57] Murez, Z., van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., Rabinovich, A.: Atlas: End-to-End 3D Scene Reconstruction from Posed Images. In: ECCV (2020) 5

[58] Nan, L., Wonka, P.: PolyFit: Polygonal Surface Reconstruction from Point Clouds. In: ICCV (2017) 6

[59] Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In: CVPR (2020) 5, 7, 19

[60] Oechsle, M., Peng, S., Geiger, A.: UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In: ICCV (2021) 5

[61] Paschalidou, D., Gool, L.V., Geiger, A.: Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image. In: CVPR (2020) 6

[62] Paschalidou, D., Katharopoulos, A., Geiger, A., Fidler, S.: Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In: CVPR (2021) 6, 22

[63] Paschalidou, D., Ulusoy, A.O., Geiger, A.: Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids. In: CVPR (2019) 6, 19, 21

[64] Ramamonjisoa, M., Stekovic, S., Lepetit, V.: MonteBoxFinder: Detecting and Filtering Primitives to Fit a Noisy Point Cloud. In: ECCV (2022) 6, 23, 25

[65] Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3D Deep Learning with PyTorch3D. arXiv:2007.08501 [cs.CV] (2020) 5, 21

[66] Roberts, L.G.: Machine perception of three-dimensional solids. Ph.D. thesis, Massachusetts Institute of Technology (1963) 6

[67] Rudin, L.I., Oshe, S.: Total variation based image restoration with free local constraints. In: ICIP (1994) 22

[68] Schnabel, R., Degener, P., Klein, R.: Completion and reconstruction with primitive shapes. Computer Graphics Forum (2009) 6

[69] Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection. Computer Graphics Forum (2007) 6

[70] Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise View Selection for Unstructured Multi-View Stereo. In: ECCV (2016) 5

[71] Seitz, S., Dyer, C.: Photorealistic Scene Reconstruction by Voxel Coloring. In: CVPR (1997) 5

[72] Sinha, A., Murez, Z., Bartolozzi, J., Badrinarayanan, V., Rabinovich, A.: DELTAS: Depth Estimation by Learning Triangulation and Densification of Sparse Points. In: ECCV (2020) 5

[73] Takikawa, T., Evans, A., Tremblay, J., Müller, T., McGuire, M., Jacobson, A., Fidler, S.: Variable bitrate neural fields (2022). https://doi.org/10.48550/ARXIV.2206.07707, https://arxiv.org/abs/2206.07707 7

[74] Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development (2023). https://doi.org/10.48550/ARXIV.2302.04264, https://arxiv.org/abs/2302.04264 12

[75] Tertikas, K., Paschalidou, D., Pan, B., Park, J.J., Uy, M.A., Emiris, I., Avrithis, Y., Guibas, L.: PartNeRF: Generating Part-Aware Editable 3D Shapes without 3D Supervision. In: CVPR (2023) 6

[76] Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning Shape Abstractions by Assembling Volumetric Primitives. In: CVPR (2017) 6, 21

[77] Tulsiani, S., Zhou, T., Efros, A.A., Malik, J.: Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. In: CVPR (2017) 7

[78] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In: NeurIPS (2021) 5, 23

[79] Wu, Y., Liu, W., Ruan, S., Chirikjian, G.S.: Primitive-based Shape Abstraction via Nonparametric Bayesian Inference. In: ECCV (2022) 6

[80] Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: MVSNet: Depth Inference for Unstructured Multi-view Stereo. In: ECCV (2018) 5

[81] Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., Quan, L.: Recurrent MVSNet for High-Resolution Multi-View Stereo Depth Inference. In: CVPR (2019) 5

[82] Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume Rendering of Neural Implicit Surfaces. In: NeurIPS (2021) 5, 23

[83] Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Basri, R., Lipman, Y.: Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. In: NeurIPS (2020) 5, 23

[84] Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural Radiance Fields from One or Few Images. In: CVPR (2021) 19

[85] Zhang, J.Y., Yang, G., Tulsiani, S., Ramanan, D.: NeRS: Neural Reflectance Surfaces for Sparse-view 3D Reconstruction in the Wild. In: NeurIPS (2021) 5

[86] Zhang, J.: Visibility-aware Multi-view Stereo Network. In: BMVC (2020) 5

[87] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: CVPR (2018) 21

[88] Zhang, X., Kundu, A., Funkhouser, T., Guibas, L., Su, H., Genova, K.: Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision (2023) 7, 11

[89] Zheng, E., Dunn, E., Jojic, V., Frahm, J.M.: PatchMatch Based Joint View Selection and Depthmap Estimation. In: CVPR (2014) 5

[90] Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the Continuity of Rotation Representations in Neural Networks. In: CVPR (2019) 20