Single-Shot Scene Reconstruction

Anonymous Author(s)
Affiliation
Address
email

Abstract: We introduce a novel scene reconstruction method to infer a fully editable and re-renderable model of a 3D road scene from a single image. We represent movable objects separately from the immovable background, and recover a full 3D model of each distinct object as well as their spatial relations in the scene. Based on transformer-based detectors and neural implicit 3D representations, we build a Scene Decomposition Network (SDN) that reconstructs the scene, and the reconstruction can further be used in analysis-by-synthesis via differentiable rendering. Trained only on simulated road scenes, our method generalizes well to real data in the same class without any adaptation thanks to its strong inductive priors. Experiments on two synthetic-real dataset pairs (PD-DDAD and VKITTI-KITTI) show that our method can robustly recover scene geometry and appearance, as well as reconstruct and re-render the scene from novel viewpoints.

1 Introduction

Decomposing images into disjoint symbolic representations is the ultimate goal of robotics and computer vision, as it allows semantic reasoning over all scene parts. Such a decomposition benefits different possible applications (e.g. robotics, augmented reality, autonomous driving) where a decomposed scene can be reassembled in different ways, enabling interaction or reenactment. Unfortunately, single image scene reconstruction is inherently ill-posed due to a variety of reasons. Most prominently, ambiguity in projective geometry can only be resolved with prior knowledge about the observed scene. Current state-of-the-art methods either do not provide a full decomposition of the scene and regress per-pixel depth [1, 2], or estimate the poses of already known objects and ignore the rest of the geometry [3, 4].

In this work, we design a pipeline to obtain an interpretable, disentangled scene representation for 3D-aware manipulation of driving scenes. Our system is capable of reconstructing full geometries and appearances of detected object instances from a single RGB image. Such a decomposition is made possible by three components: (1) a scene decomposition network (SDN) that detects object instances, recovers their partial geometries and poses, and predicts the full geometry of the background and RGB color behind the occluders; (2) a differentiable database of colored object priors that encodes their full geometry and appearance in the form of signed distance fields (SDF) and luminance fields (LF); and (3) a 2D/3D optimization pipeline that uses surfel-based differentiable rendering to fit initial observations to the object database, leveraging their shape and appearance.

Our contributions are summarized as follows:

- A holistic single-shot scene reconstruction system that splits the scene into background and object instances and recovers their full geometry and appearance;
- An optimization pipeline based on NOCS-SDF and RGB-LF fitting as well as surfel-based differentiable rendering to refine geometries, appearances, and poses of detected instances;
- A scene manipulation and generation pipeline allowing to place and render objects of various geometry and appearance, and change view points;
- Applicable to real data without domain adaptation.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

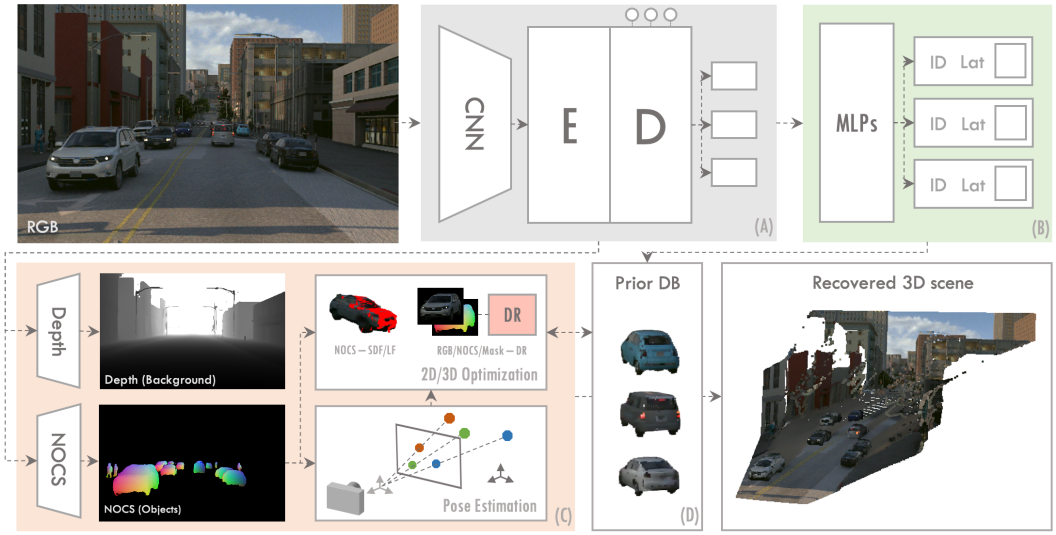

Figure 1: **Scene Decomposition Network (SDN)**: Given an RGB image, our SDN recovers partial canonicalized object shapes in the form of NOCS maps and predicts background depth. NOCS maps are then used to directly estimate object poses using a PnP solver. We further fit learned differentiable object priors to align with predicted partial shapes in terms of geometry and appearance, and refine initially predicted poses.

2 Related Work

Single-Shot Scene Reconstruction Holistic single image scene reconstruction is a highly ill-posed problem, tackled from different angles by a variety of works. The most common approach is monocular depth estimation that predicts per-pixel depth values [5, 1, 2]. While achieving remarkable results, these methods reconstruct the scene as a whole and do not incorporate knowledge about the scene's objects and layout. Alternatively, 3D detection pipelines detect separate objects and recover their masks and 3D bounding boxes [3, 6, 4], or incorporate relationships between objects by using a graph or physical simulation [7, 8, 9]. While having understanding about the object appearance in 2D and their 3D pose, these methods do not recover the shapes of the objects nor the rest of the scene. Lastly, there are pipelines aiming at reconstructing objects from detections by aligning 3D CAD models to fit the predicted masks and input images [10, 11, 12], albeit without generalization to unknown instances outside the model database. In our work we recover the geometry and appearance for both background and detected objects by using a differentiable database of object priors allowing us to smoothly interpolate between plausible object shapes.

6DoF Object Detection via Correspondence Regression Object detection and pose estimation from RGB images is a well-established but ongoing research problem due to, among others, the inherent ambiguity of perspective projection. The myriad of existing solutions can loosely be grouped in three categories: direct pose regression (using a neural net to output the pose directly by learning from data) [13, 14, 15], template-matching (comparing patches with a predefined set of templates either in pixel space or feature space) [16, 17, 18, 19], and correspondence-based methods (directly utilizing correspondences between image coordinates and 3D models of interest) [20, 21, 3, 4] - each having their own benefits and drawbacks. The current state-of-the-art methods in object pose estimation almost exclusively belong to the latter group with such representatives as PVNet [6], CDPN [22], EPOS [21], Pix2Pose [4], and DPOD [3]. They all regress per-pixel correspondences and use a PnP solver to estimate a 6DoF pose given a set of 2D-3D correspondences and camera parameters, minimizing reprojection errors. While some works regress manually defined UV correspondences, others regress *Normalized Object Coordinate Space* (NOCS) maps [23] that represent visible xyz object coordinates in RGB space. The benefits of NOCS are their straightforward creation through rendering, and their simultaneous definition of both the object's pose and shape. In our work we show how to effectively leverage this representation for geometry retrieval as well.

Neural Implicit Representations Recently, a new prominent direction for 3D shape and appearance representation has emerged with DeepSDF [24], Occupancy networks [25], and IM-Net [26],

using neural nets for scalar function approximation. All three methods take 3D coordinates as input and output either a binary occupancy estimate or a continuous SDF value, encoding the object's surface. Subsequently, follow-up work such as Deep Local Shapes (DLS) [27], Neural Geometric Level of Detail (NGLOD) [28] or NASA [29] improved the direction further. While DLS stores SDFs as voxel grid cells to allow larger scenes, NGLOD uses an efficient Octgrid representation and a feature composition scheme based on trilinear interpolation to further improve the reconstruction quality, whereas NASA extends the approach to articulated objects.

Similar representations have been extended to represent object appearance. Scene Representation Networks (SRN) [30] add a ray marching routine and also regress RGB colors at surface intersections to learn from multi-view images, and Differentiable Volumetric Rendering [31] which couples an implicit shape representation with differentiable rendering. Alternatively, instead of regressing SDF or RGB values for predefined 3D coordinates, NeRF [32] proposes to regress density and color values along rays (5D coordinates) and compute the true color via numerical integration. This approach, drawing inspiration from Neural Volumes [33] and their volumetric representation, boosted interest in implicit volumetric rendering and resulted in a multitude of works tackling problems from training and rendering time performance [34, 35, 36, 37], to covering dynamic scenes [38, 39, 40, 41], scene relighting [42, 43, 44], and composition [45, 46, 47].

While providing realistic renderings, most NeRF-based methods overfit to a single scene (with the exception of PixelNeRF [48]) and require a set of very dense view point annotations. Our pipeline, on the other hand, is capable of reconstructing scenes never seen during training, and is also not limited to viewpoints seen during training.

3 Methodology

Given a single RGB image of a typical driving scene, our pipeline holistically reconstructs the scene by not only explaining every visible pixel, but also predicting the full geometry and appearance of detected objects and the occluded areas that are not visible on the image. This is made possible by a combination of three components: a scene decomposition network (SDN), a differentiable database of object priors, and a 2D/3D optimization pipeline.

3.1 Scene Decomposition Network

The first component of our pipeline (Fig. 1) is the scene decomposition network, or SDN. Its main role is to estimate the layout of the scene, detect and identify visible objects, and recover the geometry of both the background and the detected objects. For clarity, it is divided into three separate blocks: a detection transformer-based decomposition block, an object reasoning block, and a 3D reasoning block.

Detection Transformer Block (Fig. 1a) The core of our SDN is a transformer-based decomposition. An encoder-decoder object detector based on transformers, a popular architecture for sequence prediction and has been first applied to computer vision in DETR [49]. DETR first generates distinctive image features from an input RGB image, and feeds them to a transformer module to build attention maps between features. This self-attention mechanism explicitly models all pairwise interactions between elements in a sequence, making this architecture particularly suitable for specific constraints of set prediction (e.g. duplicate suppression). Using the notion of object queries, we are able to retrieve output features for each of the detected objects, which are then used as input to our object reasoning block and 3D reasoning block.

Object Reasoning Block (Fig. 1b) Our object reasoning block takes output features of the detection transformer and uses a collection of MLPs to predict important object properties: class IDs, 2D bounding boxes, and SDF and LF feature vectors providing an initial state for object reconstruction. These outputs are essential to effectively reason about and refine the 3D component of these objects.

3D Reasoning Block (Fig. 1c) Our 3D reasoning block aims to recover 3D scene information by splitting it into two parts: a background containing road surfaces, buildings, and other objects not detected by the detection transformer, and a foreground entirely consisting of detected objects. We use the output of the transformer decoder for each object to compute multi-head attention scores of

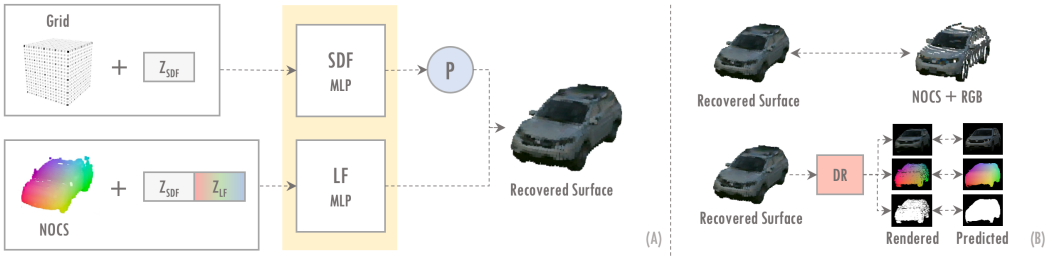

Figure 2: **Differentiable database of object priors (A)** and **shape, appearance, and pose optimization (B)**. Our object priors are learned using coordinate-based MLPs, taking positionally encoded features as input and recovering geometry in form of SDF and appearance on the object's surface. We use a 0-isosurface projection (marked P), to retrieve object's surface from an SDF field. Our initial recovered object is then refined to fit both predicted observations in terms of masks and NOCS map, as well as GT RGB values from the input image.

this embedding over the output of the encoder, generating M attention heatmaps per object. These masks are then used to regress the geometry for both foreground and background.

The foreground is predicted as a set of normalized shape maps which are essentially NOCS renderings. Since they encode 3D coordinates via RGB, visualizing each 3D color component in 3D space allows us to recover a partial 3D shape of the object in its normalized coordinate space (Fig. 3). This representation is crucial for our pipeline to retrieve the object's full geometry and pose.

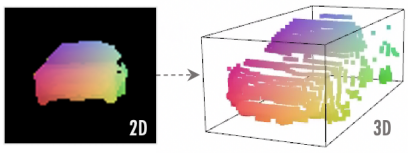

Figure 3: Partial shape recovery from a predicted 2D NOCS map.

The background, on the other hand, is represented as a depth map since we solely care about its geometry. For view synthesis applications, we also predict the depth and appearance behind the detected objects by utilizing a GAN-like encoder-decoder generator architecture. This architecture takes masked RGB and depth images and inpaints RGB and depth values for occluded regions, employing a fully-convolutional PatchGAN discriminator to judge the genuineness of the inpainted maps.

Given estimated object masks and NOCS maps, we recover three things enabling pose estimation: (1) 2D coordinates of the object in the image, (2) 3D points of the partial object shape in the canonical frame of reference, (3) and their 2D-3D correspondences. This unique representation frees us from the need of storing a collection of 3D primitives and identifying them to find a detected model, because both 3D and 2D information is contained in a form of a 3-channel map. We multiply the recovered normalized shape by the per class scale factor to recover absolute scale. The 6DoF pose is then estimated using PnP [50] that predicts the object pose from given correspondences and camera intrinsics. Since we get a large set of correspondences for each model, we also combine PnP with RANSAC to robustify against outliers. For the results presented in the evaluation section, we run 1000 RANSAC iterations per pose with a $1px$ reprojection error threshold.

3.2 Shape, Pose, and Appearance Optimization

The second crucial component of our pipeline is a 2D/3D optimization that is enabled by the differentiable database of object priors (Fig. 5a). Given SDF and LF features corresponding to detected objects regressed by the object reasoning block (Fig. 1b), our differentiable database generates full shapes and appearances of the objects.

Our optimizer takes partial canonical object shape and appearance observations (recovered from predicted NOCS maps), as well as initial *PriorDB* features. Due to the robustness of NOCS regression, and despite the occasional noisy outliers, we set it as our main optimization prior. The initially predicted SDF is used to recover the full surface of the object in the form of a point cloud with surface normals using a 0-isosurface projection. Then we estimate nearest neighbors between the two point clouds and minimize the distance between the points by optimizing the feature vector $\mathbf{z}_{sdf}$. The RGB component is optimized similarly by minimizing the color difference between the

correspondences, but we also allow the LF module weights to vary for a finer level of detail. In the case of RGB optimization we also use the L_{symm} loss between the left and right sides of the predicted shape.

While NOCS-based SDF and LF optimization allows us to reconstruct plausible objects from images, it does not account for the pose component. It might very well be the case that estimated poses are off and harming the full scene reconstruction, which can be resolved by a 2D alignment. Additionally, 2D alignment can be used as a complimentary source for shape optimization when predicted NOCS maps are noisy, resulting in a lower reprojection error and better reconstructions. To make it possible, we define three losses on the rendering output $L_{NOCS_{2D}}$, $L_{LF_{2D}}$, and L_{mask}. $L_{NOCS_{2D}}$ compares the output of the renderer with predicted NOCS, $L_{LF_{2D}}$ compares the output of the renderer with input RGB images, and L_{mask} compares the rendered and the predicted masks.

2D alignment starts after 3D alignment is completed, as it could sometimes lead to pose misalignment. When the shape and appearance fits to a predicted partial point cloud observation, we begin optimizing the pose of the object aiming to fit the input RGB image, predicted mask, and NOCS. If the pose optimization is successful and losses go below a certain threshold, we further continue our optimization to refine the shape of the object resulting in a maximum alignment. If the pose could not be refined, we leave the 3D optimized result untouched.

3.2.1 Differentiable Database of Object Priors (Fig. 1d)

Our differential database of object priors or *PriorDB* represents objects as signed distance fields, in which each value corresponds to the distance to the closest surface (with positive and negative values pointing to exterior and interior areas respectively). A single MLP can be used to represent multiple geometries using a single feature vector $\mathbf{z}_{sdf}$ and query 3D locations $\mathbf{x} = \{x_1, ..., x_N\}$ as $f_{sdf}(\mathbf{x}; \mathbf{z}_{sdf}) = \mathbf{s}$. Object appearances, on the other hand, are represented as luminance fields (LF), defining the prior perceived luminance of the object as seen in the training set. Similarly to SDN, the LF module is implemented as an MLP, but it takes a feature vector by concatenating $\mathbf{z}_{sdf}$ and $\mathbf{z}_{lf}$ as well as query locations $\mathbf{x}$ as input, and outputs resulting luminance as $f_{lf}(\mathbf{x}; \mathbf{z}_{sdf}; \mathbf{z}_{lf}) = \mathbf{l}$.

Both SDF and LF modules are trained using synthetic data provided by Parallel Domain [51], containing high-quality realistic road scene renderings together with GT annotations. To train the SDF module, we use a collection of canonical GT point clouds and associated surface normals representing rendered objects. In total, we encode 27 cars in our *PriorDB*. We first define a multi-level octree spanning the volume $[-1, 1]^3$, with points located at the corners of the imaginary voxels of various sizes depending on the level. For each of the point clouds we define unsigned distances by computing nearest neighbors to the octree points. Next, we define the loss L_{sdf} as the distance to the surface at grid points starting from the coarsest level, gradually increasing the resolution up to the highest level (in our implementation we use 3 levels of granularity as shown in Fig. 4), while only considering points located at pre-defined per-level distances to the surface. To ensure that we learn a correctly signed field, we additionally minimize the distance between the GT surface normals and analytically-derived surface normals from the SDF module. Moreover, to ensure that we always regress plausible shapes regardless the input feature vector, we do an additional training round by sampling random feature vectors and enforcing the output to be equal to the shape of the nearest plausible neighbor.

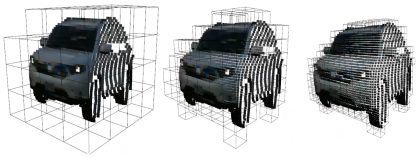

Figure 4: Different octree levels of detail used for training SDF primitives.

Once the SDF module is trained, we store SDF features $\mathbf{z}_{sdf}$ associated with objects and use them to train the LF module. Differently, it is trained on partial canonical shapes recovered from provided RGB renderings (see Fig. 5a), associated depth maps and poses. We compute the luminance component from RGB renderings and enforce the LF of each feature to be the same. The network weights are then optimized to fit the average luminance per object using an L1 loss. Additionally, we define a symmetry loss L_{symm} enforcing colors on the left side of the car to be similar to those on the right one. For that we compute nearest neighbors between the left side of the car and an inverted right side and then minimize the L1 distance between these correspondences.

Figure 5: **Example reconstructions on VKITTI2.** Our Scene Decomposition Network recovers partial reconstructions (left), which are then completed with shape priors from *PriorDB* using our 2D-3D optimization pipeline (right).

3.2.2 Surfel-based Differentiable Renderer

Differentiable rendering allows to optimize objects with respect to pose and shape, and is an essential component of our optimization scheme. Since we use an SDF representation, we cannot directly use differentiable renderers for triangulated meshes. Instead, we implemented a renderer that uses surfels as the primary representation (inspired by [52]), estimating surface points and normals using an 0-isosurface projection, followed by a surfel-based rasterizer. We opted for this due to its simplicity and efficiency.

Surface Tangent Discs Surface elements or surfels are a common concept in computer graphics [53] as an alternative to connected triangular primitives. In order to render watertight surfaces, the individual surface normals must sufficiently approximate the local region of the object's geometry.

To construct surface discs we first estimate the 3D coordinates of the resulting tangent plane given the normal $n_i = \frac{\partial f(p_i;\mathbf{z})}{\partial p_i}$ of a projected point p_i. The distance d of the plane to each 2D pixel (u, v) is computed by solving a system of linear equations for the plane and camera projection:

$$d = \frac{n_i \cdot p_i}{n_i \cdot \mathbf{K}^{-1}(u, v, 1)^T} \tag{1}$$

where $\mathbf{K}^{-1}$ is the inverse camera matrix, followed by backprojection to get the final 3D plane coordinate:

$$P = \mathbf{K}^{-1} \cdot (u \cdot d, v \cdot d, d)^T. \tag{2}$$

Then we estimate the distance between the plane vertex and surface point and clamp if it is larger than a disc diameter to get final discs:

$$M = max(diam - ||p_i - P||_2, 0) \tag{3}$$

Rendering Function Similarly to [52, 54], we combine colors from different surfel primitives based on their depth values to compose a final rendering. The distance of the primitive to the camera defines its final intensity contribution to the image. To ensure that all primitive contributions sum up to 1 at each pixel we use a modified softmax function. The final rendering function is defined as:

$$\mathcal{I} = \sum_i S(p_i) * w_i, \tag{4}$$

where $\mathcal{I}$ is the output image, S the estimated NOCS map, and w_i the weighting masks defined as:

$$w_i = \frac{exp(-\tilde{\boldsymbol{D}}_i \boldsymbol{\sigma}) \boldsymbol{M}_i}{\sum_j exp(-\tilde{\boldsymbol{D}}_j \boldsymbol{\sigma}) \boldsymbol{M}_j}, \tag{5}$$

where $\tilde{\boldsymbol{D}}$ is the normalized depth and $\boldsymbol{\sigma}$ is a transparency weight with $\boldsymbol{\sigma} \rightarrow \infty$ defining a completely opaque rendering as only the closest primitive is rendered.

4 Experiments

In this section, we first demonstrate the capabilities of our pipeline for the tasks of scene reconstruction, object recovery and manipulation, and novel scene generation. Then we evaluate the individual

Figure 6: **Scene reconstruction using our pipeline on KITTI.** Our pipeline decomposes the scene into objects (c) and background (d) and recovers full geometries and appearances (e). This allows for scene manipulation, view synthesis and novel scene arrangements (b). Despite being trained on synthetic data, our pipeline generalizes well to real images without any adaptation.

components of the SDN on common benchmarks for depth estimation and 3D object detection. The evaluation is performed on four datasets: PD, VKITTI2, DDAD, and KITTI.

4.1 Datasets

VKITTI2 [55] and KITTI [56]. Virtual KITTI2 (VKITTI2) is a synthetic proxy to the KITTI dataset – the standard benchmark for autonomous computer vision. It contains five virtual worlds rendered under ten different conditions of weather and camera configuration, in total comprising 21,260 images. In the experiments we use the first 80% consecutive frames for training and the rest for testing.

Parallel Domain (PD) [51] and DDAD [2]. The Parallel Domain (PD) dataset is a procedurally-generated dataset of driving scenes, and contains fully annotated photo-realistic renderings of urban driving scenes, including multiple cameras and LiDAR sensors. It contains 200 200-frame sequences resulting in total of 40000 images. Similarly to VKITTI2, we use the first 90% for training and the rest for testing. The Dense Depth for Automated Driving (DDAD) dataset is a real counterpart of the PD dataset, and is mainly used as a depth evaluation benchmark. We consider only the front camera, resulting in 50 validation sequences with 3950 images with panoptic [57] labels (the middle frame in each sequence).

Method	PSNR↑	SSIM ↑	LPIPS↓
PixelNeRF	16.63	0.45	0.57
Ours	**17.52**	**0.58**	**0.34**

Table 1: **View synthesis results.** We evaluate our method for the task of stereo-pair recovery on the VKITTI dataset.

4.2 Scene Decomposition, Manipulation, and Object Recovery

The first experiment aims to demonstrate the capability of our pipeline to reconstruct and manipulate the scene. In Fig. 6, we show the results of scene decomposition on the KITTI dataset. The reference image (Fig. 6a) is an input to our Scene Decomposition Network, which recovers full 3D objects (Fig. 6c) and background (Fig. 6d) of the scene. Recovered objects can be freely manipulated as shown in Fig. 6b. The full geometrical scene reconstruction is shown in Fig. 6e. Note that our pipeline has been trained entirely on synthetic data, and generalizes to real images without any adaptation. We refer the readers to our supplementary videos for the visualization of the full capability of scene manipulation.

4.3 View Synthesis

In this experiment we quantitatively evaluate our pipeline for the task of view synthesis by recovering a stereo-pair of the input image. We compare our results with a state-of-the-art generalizable NeRF-based view synthesis method PixelNeRF [48]. Both pipelines were trained on the same training split of VKITTI and are then evaluated on a pre-defined val splits of VKITTI. From the results in Table 1, we observe that PixelNeRF is not capable of generalizing to images that have not been seen during training. Our method on the other hand can reliably re-evaluate the scene from a novel view point, thanks to its integrated prior and geometrical nature.

Train	Method	Test	*Lower is better*				*Higher is better*		
			AbsRel	SqRel	RMSE	RMSE_{log}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
VKITTI	Monodepth2	KITTI	0.239	3.217	8.425	0.354	0.469	0.655	0.767
	PackNet-SfM		0.211	2.746	7.701	**0.319**	0.546	0.704	0.780
	Ours		**0.175**	**2.375**	**7.384**	0.360	**0.642**	**0.756**	**0.803**
	Monodepth2	DDAD	0.437	15.621	26.578	0.731	0.025	0.118	0.241
	PackNet-SfM		0.340	11.290	22.332	0.461	0.138	0.309	0.521
	Ours		**0.238**	**10.402**	**18.228**	**0.346**	**0.535**	**0.652**	**0.704**
PD	Monodepth2	KITTI	0.368	7.976	14.432	0.653	0.320	0.480	0.580
	PackNet-SfM		0.332	3.779	9.205	0.395	0.318	0.672	0.836
	Ours		**0.176**	**2.407**	**7.687**	**0.372**	**0.696**	**0.795**	**0.840**
	Monodepth2	DDAD	0.187	3.951	6.465	0.184	0.429	0.506	0.586
	PackNet-SfM		0.165	**2.500**	**6.017**	**0.165**	0.366	**0.591**	**0.620**
	Ours2		**0.139**	3.808	6.612	0.231	**0.497**	**0.559**	0.590

Table 2: **Depth estimation results** for direct transfer from synthetic (VKITTI and PD) to real (KITTI and DDAD) datasets. KITTI results are reported with the *Garg* crop at distances up to 80m, and DDAD results are reported without crop at distances up to 80m.

4.4 Depth Evaluation

Here we evaluate the quality of our geometrical output in terms of common metrics for depth evaluation. We argue that our NOCS maps, given the estimated poses, can be transformed to depth maps and significantly outperform standard depth estimation pipelines due to the prior geometry knowledge. To conduct this experiment, we choose current state-of-the-art depth estimation pipelines, Monodepth2 [1], and PackNet-SfM [2] and train them on 2 different synthetic datasets - VKITTI and PD - similarly to our method. Then we evaluate the depth quality for foreground objects on real datasets - KITTI and DDAD - using a number of standard depth estimation metrics. This experiment evaluates three aspects of our model: 1) quality of the reconstructed object geometry, 2) quality of NOCS estimated poses, and 3) transferrability across domains. Table 2 shows that our model consistently outperforms the baselines of monocular depth estimation, demonstrating the superior geometric accuracy and generalization qualities achieved thanks to the encoded object prior.

4.5 3D Detection

Lastly, we aim to demonstrate how well our method generalizes to other datasets in terms of 3D detection quality. To show that we take a state-of-the-art method for KITTI pose estimation, SMOKE [58], and train it on VKITTI *train* subset following its training protocol. To compare with our algorithm that has access to ground-truth size of 3D boxes, we substitute the size of predicted boxes with the ones of nearest-neighbor ground-truth, when evaluating SMOKE. Table 3 shows that we perform competitively with SMOKE, and demonstrates that our underlying NOCS representation generalize to diverse shapes and poses of objects in the real-world.

Method	BEV AP			3D AP		
	Easy	Med	Hard	Easy	Med	Hard
SMOKE	14.30	9.51	5.94	**11.47**	7.40	5.31
Ours	**16.50**	**11.88**	**10.15**	10.94	**7.54**	**6.72**

Table 3: **KITTI-3D detection benchmark.** We compare our method with a state-of-the-art single-shot monocular 3D detection algorithm. Both methods are trained on VKITTI2 *train* subset, and evaluated on KITTI-3D *val* subset. Reported are $AP|_{R_{40}}$ metrics of *Car* with 50% as IoU threshold.

5 Conclusion

We present a novel holistic scene reconstruction pipeline that disentangles the scene by representing objects separately from the scene. Our Scene Decomposition Network detects and recovers partial geometries and poses of objects, and estimates depth of the background. The partial geometries are then completed using our differentiable database of object priors and a 2D-3D optimization procedure. As a result, our pipeline explains every pixel on the image and more - the full object geometries and appearances. Despite being trained only on synthetic data our experiments show good generalization to real data on KITTI and DDAD datasets.

References

[1] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth prediction. In *ICCV*, 2019.

[2] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] S. Zakharov, I. Shugurov, and S. Ilic. Dpod: Dense 6d pose object detector in rgb images. In *ICCV*, 2019.

[4] K. Park, T. Patten, and M. Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *ICCV*, 2019.

[5] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.

[6] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019.

[7] Y. Chen, S. Huang, T. Yuan, S. Qi, Y. Zhu, and S.-C. Zhu. Holistic++ scene understanding: Single-view 3d holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8648–8657, 2019.

[8] Y. Du, Z. Liu, H. Basevi, A. Leonardis, B. Freeman, J. Tenenbaum, and J. Wu. Learning to exploit stability for 3d scene parsing. In *NeurIPS*, pages 1733–1743, 2018.

[9] S. Huang, S. Qi, Y. Xiao, Y. Zhu, Y. N. Wu, and S.-C. Zhu. Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. *arXiv preprint arXiv:1810.13049*, 2018.

[10] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *ECCV*, 2018.

[11] F. Manhardt, W. Kehl, N. Navab, and F. Tombari. Deep model-based 6d pose refinement in rgb. In *ECCV*, 2018.

[12] I. Shugurov, I. Pavlov, S. Zakharov, and S. Ilic. Multi-view object pose refinement with differentiable renderer. *IEEE Robotics and Automation Letters*, 6(2):2579–2586, 2021.

[13] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015.

[14] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017.

[15] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *ICCV*, 2017.

[16] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011.

[17] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, 2012.

[18] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit. Hashmod: A hashing method for scalable 3d object detection. In *BMVC*, 2015.

[19] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *CVPR*, 2015.

[20] O. H. Jafari, S. K. Mustikovela, K. Pertsch, E. Brachmann, and C. Rother. ipose: instance-aware 6d pose estimation of partly occluded objects. In *ACCV*, 2018.

[21] T. Hodan, D. Barath, and J. Matas. Epos: estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11703–11712, 2020.

[22] Z. Li, G. Wang, and X. Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *CVPR*, 2019.

[23] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[24] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[25] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[26] W. Chen, J. Gao, H. Ling, E. J. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NeurIPS*, 2019.

[27] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020.

[28] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *arXiv preprint arXiv:2101.10994*, 2021.

[29] B. Deng, J. P. Lewis, T. Jeruzalski, G. Pons-Moll, G. Hinton, M. Norouzi, and A. Tagliasacchi. Nasa: neural articulated shape approximation. *arXiv preprint arXiv:1912.03207*, 2019.

[30] V. Sitzmann, M. Zollhoefer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32:1121–1132, 2019.

[31] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020.

[32] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[33] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[34] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[35] D. Rebain, W. Jiang, S. Yazdani, K. Li, K. M. Yi, and A. Tagliasacchi. Derf: Decomposed radiance fields. *arXiv preprint arXiv:2011.12490*, 2020.

[36] D. B. Lindell, J. N. Martel, and G. Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[37] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng. Learned initializations for optimizing coordinate-based neural representations. *arXiv preprint arXiv:2012.02189*, 2020.

[38] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R.-M. Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[39] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020.

[40] W. Xian, J.-B. Huang, J. Kopf, and C. Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv preprint arXiv:2011.12950*, 2020.

[41] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[42] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

[43] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. P. Lensch. Nerd: Neural reflectance decomposition from image collections, 2020.

[44] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.

[45] M. Niemeyer and A. Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. *arXiv preprint arXiv:2011.12100*, 2020.

[46] W. Yuan, Z. Lv, T. Schmidt, and S. Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. *arXiv preprint arXiv:2101.01602*, 2021.

[47] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide. Neural scene graphs for dynamic scenes. *arXiv preprint arXiv:2011.10379*, 2020.

[48] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020.

[49] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end detection with transformers. In *ICCV*, 2020.

[50] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[51] Parallel domain. https://paralleldomain.com/, May 2021.

[52] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *CVPR*, 2020.

[53] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *SIGGRAPH*, 2000.

[54] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019.

[55] Y. Cabon, N. Murray, and M. Humenberger. Virtual kitti 2, 2020.

[56] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[57] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[58] Z. Liu, Z. Wu, and R. Tóth. SMOKE: Single-stage monocular 3d object detection via keypoint estimation. *arXiv preprint arXiv:2002.10111*, 2020.