

# DSP-SLAM: Object Oriented SLAM with Deep Shape Priors

Jingwen Wang    Martin Rünz    Lourdes Agapito  
 Department of Computer Science, University College London  
 {jingwen.wang.17, martin.runz.15, l.agapito}@ucl.ac.uk

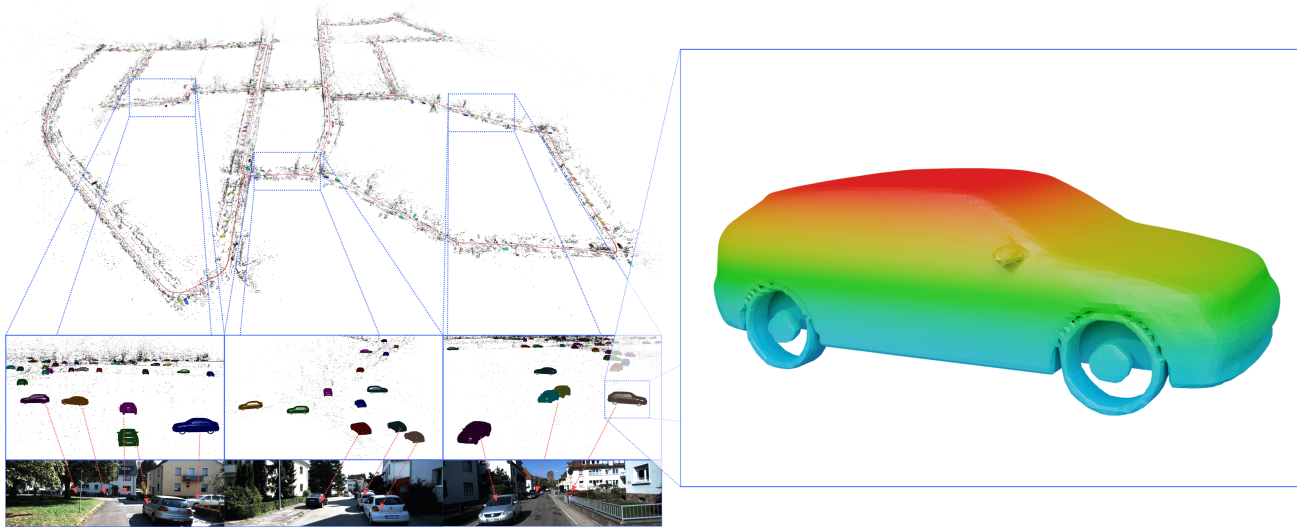


Figure 1: DSP-SLAM builds a rich object-aware map, providing complete detailed shapes of detected objects, while representing the background coarsely as sparse feature points. Reconstructed map and camera trajectory on KITTI 00.

## Abstract

We propose DSP-SLAM, an object-oriented SLAM system that builds a rich and accurate joint map of dense 3D models for foreground objects, and sparse landmark points to represent the background. DSP-SLAM takes as input the 3D point cloud reconstructed by a feature-based SLAM system and equips it with the ability to enhance its sparse map with dense reconstructions of detected objects. Objects are detected via semantic instance segmentation, and their shape and pose are estimated using category-specific deep shape embeddings as priors, via a novel second order optimization. Our object-aware bundle adjustment builds a pose-graph to jointly optimize camera poses, object locations and feature points. DSP-SLAM can operate at 10 frames per second on 3 different input modalities: monocular, stereo, or stereo+LiDAR. We demonstrate DSP-SLAM operating at almost frame rate on monocular-RGB sequences from the Freiburg and Redwood-OS datasets, and on stereo+LiDAR sequences on the KITTI odometry dataset showing that it achieves high-quality full object reconstructions, even from partial observations,

while maintaining a consistent global map. Our evaluation shows improvements in object pose and shape reconstruction with respect to recent deep prior-based reconstruction methods and reductions in camera tracking drift on the KITTI dataset. More details and demonstrations are available at our project page: <https://jingwenwang95.github.io/dsp-slam/>

## 1. Introduction

Simultaneous Localization and Mapping (SLAM) is the process of estimating the trajectory of a moving camera while reconstructing its surrounding environment. From a purely geometric perspective, SLAM is often regarded as a well-understood or even solved problem. Many state-of-the-art dense SLAM algorithms can achieve accurate trajectory estimation and create high-quality geometric reconstructions that can be used in obstacle avoidance or path planning for mobile robots. However, when it comes to more complex tasks that require scene understanding, geometry-only scene representations fall short of providing key semantic information. Taking advantage of recent deep

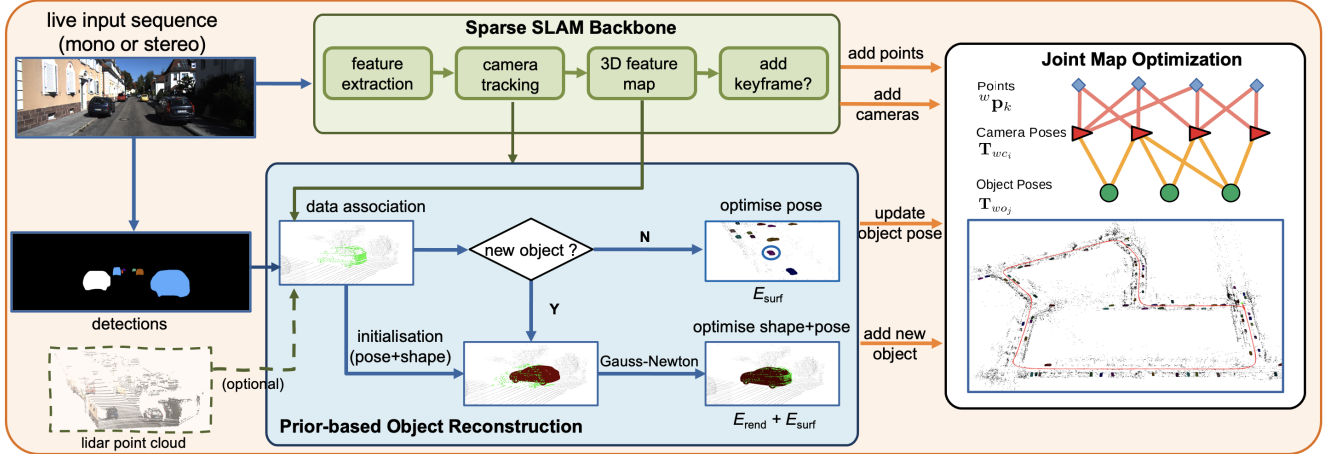


Figure 2: System overview: DSP-SLAM takes a live stream of monocular or stereo images, infers object masks, and outputs a joint map of feature points and dense objects. The sparse SLAM backbone provides per-frame camera poses and a 3D point cloud. At each keyframe, a shape code is estimated for each new detected object instance, using a combination of 3D surface consistency and rendered depth losses. DSP-SLAM can operate in 3 different modes: monocular, stereo, and stereo+LiDAR (when an optional LiDAR point cloud is available).

learning breakthroughs in semantic segmentation and object detection algorithms [12, 30, 29] semantic SLAM systems augment geometric low-level map primitives by fusing semantic labels into the 3D reconstruction [21, 40, 4]. However, the resulting scene maps merely consist of a set of labelled 3D points where reasoning about the scene at the level of objects to infer meaningful information such as the number of objects of each category, their size, shape or relative pose remains a challenging task. Better use of the semantic information is required in the form of an object-centric map representation that allows detailed shape estimation and meaningful instantiation of scene objects.

Our proposed approach forms part of a more recent family of *object-aware* SLAM methods that reconstruct object-centric maps grouping all the low-level geometric primitives (voxels, points ...) that make up the same object into a single instance. Front-end camera tracking and back-end optimization are both performed at the level of object instances. While the first object-level methods, such as SLAM++ [33], mapped previously known object instances, more recent systems have taken advantage of instance level semantic segmentation masks [12] to achieve object level reconstruction for unknown objects [20] even in the presence of dynamic objects [31, 44].

However, these early object level SLAM systems exhibit major drawbacks: They either require a pre-known database of object instances [33]; or reconstruct objects from scratch without exploiting shape priors [31, 44, 20], which results in partial or incomplete object reconstructions. We improve this by exploiting the regularity of shapes within an object category in the form of learned shape priors, defined as a latent code  $z$  and a generative model  $G(z)$  that decodes it



Figure 3: Qualitative shape and pose results on a stereo+LiDAR KITTI sequence. A very sparse set of LiDAR points was used to reconstruct each car. LiDAR points on the road are only shown for illustration.

into its full geometry. This brings us several advantages; object shapes decoded from latent codes are guaranteed to be detailed and complete, regardless of partial observations or changes in view-points, they provide a compact representation and they can be optimized using the gradients obtained through back-propagation.

Using ORB-SLAM2 [22] as a sparse camera tracking and mapping backbone, DSP-SLAM takes the reconstructed 3D point-cloud as input and fits a latent code to each detected object instance, using a combination of 3D surface consistency and rendered depth losses. Foreground objects, background features and camera poses are further refined via bundle adjustment using a joint factor graph. We show DSP-SLAM operating in 3 different modes: monocular, stereo, and stereo+LiDAR. The monocular and stereo systems use the respective ORB-SLAM2 modalities as the SLAM backbone and the reconstructed 3D point-clouds to reconstruct the detected objects. The stereo+LiDAR system uses stereo ORB-SLAM2 as the SLAM backbone but in addition it incorporates a sparse set of LiDAR measurements (as few as 50 per object) for object reconstruction and pose-only optimization.

**Contributions:** While DSP-SLAM is not the first approach

to leverage shape priors for 3D reconstruction [32, 36] from image sequences, it innovates in various ways. Firstly, unlike [32, 36], our map does not only represent objects, but also reconstructs the background as sparse feature points, optimizing them together in a joint factor graph, marrying the best properties of feature-based [22] (highly accurate camera tracking) and object-aware SLAM (high level semantic map). Secondly, although Node-SLAM [36] also incorporates shape priors within a real-time SLAM system [36], it uses dense depth images for shape optimization, while DSP-SLAM can operate with RGB-only monocular streams and requires as few as 50 3D points per object to obtain accurate shape estimates. Finally, although both FroDO [32] and DSP-SLAM can operate in a monocular RGB setting, FroDO is a slow batch approach that requires all frames to be acquired in advance and associated with their camera poses, while DSP-SLAM is an online, sequential method that can operate at 10 frames per second.

In terms of object shape and pose estimation, we improve quantitative and qualitatively over auto-labelling [47], a state-of-the-art prior-based object reconstruction method. Experiments on the KITTI odometry [9] dataset show that, with stereo+LiDAR input our joint bundle adjustment offers improvements in trajectory estimation over the feature-only stereo system ORB-SLAM2 [22], used as our backbone. Moreover, DSP-SLAM offers comparable tracking performance to state-of-the-art stereo [41], LiDAR-only [4] and dynamic [1] SLAM systems, while providing rich dense object reconstructions. DSP-SLAM also achieves promising qualitative reconstruction results with monocular input on Freiburg Cars [34] and Redwood-OS [5] dataset.

## 2. Related work

**Object-aware SLAM:** SLAM++ [33] pioneered object-aware RGB-D SLAM, representing the scene at the level of objects using a joint pose-graph for camera and object poses. A database of pre-scanned objects was created in advance and object instances were detected and mapped using a pre-trained 3D detector, ICP losses and pose-graph optimization. In later work, Tateno *et al.* [38] aligned object instances from a pre-trained database to volumetric maps while Stuckler *et al.* [35] performed online exploration, learning object models on the fly and tracking them in real time. An important drawback of instance-based approaches is their inability to scale to a large number of objects and their need for object models to be known in advance. More recent object-aware RGB-D SLAM systems have dropped the requirement for known models and instead take advantage of state-of-the-art 2D instance-level semantic segmentation masks [12] to obtain object-level scene graphs [21] and per-object reconstructions via depth fusion, even in the case of dynamic scenes [31, 44].

Extensions of object-aware SLAM to the case of monoc-

ular video input deal with the additional challenge of relying only on RGB-only information [8, 13, 24, 46, 26] which results in the use of simplistic shape representations. In QuadricSLAM [24] objects are represented as ellipsoids and fit to monocular observations while in CubeSLAM [46] cuboid proposals generated from single-view detections are optimized in a joint bundle adjustment optimization.

While the above SLAM systems represent an important step forward towards equipping robots with the capability of building semantically meaningful object-oriented maps, they fall short of exploiting semantic priors for object reconstruction. In this paper we take this direction of using a category-specific learnt shape prior and embed this within an object-aware SLAM system.

**3D Reconstruction with Shape Priors:** The use of learnt compact shape embeddings as priors for 3D reconstruction has a long tradition in computer vision. From 3D morphable models for the reconstruction of faces or bodies [2, 19], to PCA models to represent category specific object shape priors [42]. Other examples of the use of embedding spaces for single or multi-view shape reconstruction include GPLVMs [6, 27, 28] or neural representations [14, 48] such as a variational autoencoder [36], AtlasNet [11, 17] or DeepSDF [25, 32, 43, 47]. DeepSDF [25] provides a powerful implicit learnt shape model that encapsulates the variations in shape across an object category, in the form of an auto-decoder network that regresses the signed distance function (SDF) values of a given object and has been used as a shape prior for single-view [43] and multi-view [32] reconstruction. Similarly to [47] DSP-SLAM adopts DeepSDF as the shape prior and takes sparse LiDAR and images as input, however [47] takes single frames and is not a SLAM method. DOPS [23] is a single-pass 3D object detection architecture for LiDAR that estimates both 3D bounding boxes and shape.

Our approach is most closely related to those that build consistent multi-object maps over an entire sequence such as FroDO [32] and Node-SLAM [36]. Unlike FroDO [32] ours is a sequential SLAM system and not a batch method. Unlike Node-SLAM [36], in our system low-level point features and high-level objects are jointly optimized to bring the best of both worlds: accurate tracking and rich semantic shape information. DeepSLAM++ [15] leverages shape priors in a SLAM pipeline by selecting 3D shapes predicted by Pix3D [37], but forward shape generation is often unstable and lead to poor results on real data.

## 3. System Overview

DSP-SLAM is a sequential localisation and mapping method that reconstructs the complete detailed shape of detected objects while representing the background coarsely as a sparse set of feature points. Each object is represented as a compact and optimizable code vector  $\mathbf{z}$ . We



Figure 4: Shape reconstruction: qualitative results.

employ DeepSDF [25] as the shape embedding, that takes as input a shape code  $\mathbf{z} \in \mathbb{R}^{64}$  and a 3D query location  $\mathbf{x} \in \mathbb{R}^3$ , and outputs the signed distance function (SDF) value  $s = G(\mathbf{x}, \mathbf{z})$  at the given point. An overview of DSP-SLAM is shown in Fig. 2. DSP-SLAM runs at almost real time (10 frames per second) and can operate on different modalities: monocular, stereo or stereo with LiDAR; depending on the available input data.

**Sparse SLAM backbone:** ORB-SLAM2 [22] is used as the tracking and mapping backbone, a feature-based SLAM framework that can operate on monocular or stereo sequences. While the tracking thread estimates camera pose at frame-rate from correspondences, the mapping thread builds a sparse map by reconstructing 3D landmarks.

**Detections:** We perform object detection at each key-frame, to jointly infer 2D bounding boxes and segmentation masks. In addition, an initial estimate for the object pose estimation is obtained via 3D bounding box detection [18, 45].

**Data association:** New detections will either be associated to existing map objects, or instantiated as a *new* object via object-level data association. Each detected object instance  $I$  consists of a 2D bounding box  $\mathcal{B}$ , a 2D mask  $\mathcal{M}$ , the depth observation of sparse 3D point cloud  $\mathcal{D}$ , and the initial object pose  $\mathbf{T}_{co,0}$ .

**Prior-based object reconstruction:** Newly instantiated objects will be reconstructed following the object reconstruction pipeline described in Sec. 4. DSP-SLAM takes the set of sparse 3D point observations  $\mathcal{D}$ , which can come from reconstructed SLAM points (in monocular and stereo modes) or LiDAR input (in stereo+LiDAR mode) and optimises the shape code and object pose to minimise surface consistency and depth rendering losses. Objects already present in the map will only have their 6-dof pose updated via pose-only optimization.

**Joint map optimisation:** A joint factor graph of point features (from SLAM), objects and camera poses is optimised via bundle adjustment to maintain a consistent map and incorporate loop closure. New objects are added as nodes to the joint factor graph and their relative pose estimates  $\mathbf{T}_{co}$  as camera-object edges. Object-level data association and joint bundle adjustment will be discussed in Sec. 5.

## 4. Object Reconstruction with Shape Priors

We aim to estimate the full dense shape  $\mathbf{z}$  and 7-DoF pose  $\mathbf{T}_{co}$ , represented as a homogeneous transformation

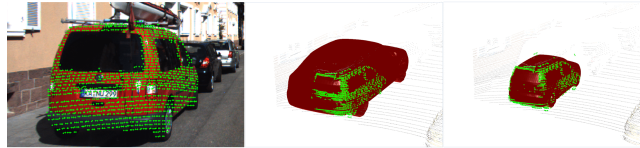


Figure 5: Illustration of the effectiveness of the rendering term in the presence of partial observations. **Left:** Detected object and partial surface point observations (green). **Middle:** Optimisation result with  $E_{surf}$  only. The loss is minimised but the shape grows larger than its actual size. **Right:** Optimisation result with the rendering term. Enforcing the silhouette constraint results in the correct scale.

matrix  $\mathbf{T}_{co} = [s\mathbf{R}_{co}, \mathbf{t}_{co}; \mathbf{0}, 1] \in \mathbf{Sim}(3)$ , for an object with detections  $I = \{\mathcal{B}, \mathcal{M}, \mathcal{D}, \mathbf{T}_{co,0}\}$ . We formulate this as a joint optimization problem, which iteratively refines the shape code and object pose from an initial estimate. We propose two energy terms  $E_{surf}$  and  $E_{rend}$  and formulate a Gauss-Newton solver with analytical Jacobians.

### 4.1. Surface Consistency Term

This term measures the alignment between observed 3D points and the reconstructed object surface.

$$E_{surf} = \frac{1}{|\Omega_s|} \sum_{\mathbf{u} \in \Omega_s} G^2(\mathbf{T}_{oc}\pi^{-1}(\mathbf{u}, \mathcal{D}), \mathbf{z}) \quad (1)$$

where  $\Omega_s$  denotes the pixel coordinates of the set of sparse 3D points  $\mathcal{D}$ , which can come from reconstructed SLAM points (in monocular and stereo modes) or LiDAR input (in stereo+LiDAR mode). Ideally, the back-projected point at pixel  $\mathbf{u}$  should perfectly align with the object surface resulting in zero SDF value, giving a zero error residual. In practice, we observed that the surface consistency term alone is not sufficient for correct shape and pose estimation in the case of partial observations. Fig. 5 illustrates a case where only points on the back and the right side of a car are observed (shown in green). Using the surface consistency alone term leads to incorrect shape estimation – much larger than its actual size. To address this issue, we propose a rendering loss, that provides point-to-point depth supervision and enforces silhouette consistency to penalize shapes that grow outside of the segmentation mask.

### 4.2. Differentiable SDF Renderer

Following [39, 36], we build our SDF renderer via differentiable ray-tracing. For each pixel  $\mathbf{u}$ , we back-project a ray  ${}^c\mathbf{x} = \mathbf{o} + d\mathbf{K}^{-1}\hat{\mathbf{u}}$  parameterized by the depth value  $d$  under camera coordinate space, with  $\mathbf{o}$  being the camera optical centre and  $\mathbf{K}$  being camera intrinsic matrix. We sample  $M$  discrete depth values  $\{d_i\}$  along each ray within the range  $[d_{min}, d_{max}]$ , with  $d_i = d_{min} + (i - 1)\Delta d$ , and  $\Delta d = (d_{max} - d_{min}) / (M - 1)$ . The bounds of the depth

range are determined by the current estimation of object translation and scale, and are re-computed at each iteration. **Occupancy Probabilities** The SDF value  $s_i$  at each sampled point can be obtained by transforming sampled points to the object coordinate frame and passing through the DeepSDF decoder. The SDF value encodes the probability that a given point is occupied by the object or belongs to free space. We apply a piecewise linear function to the predicted SDF values to indicate the occupancy probability  $o_i$ , defined in Eq. 2, where  $\sigma$  represents the cut-off threshold which controls the smoothness of the transition. We fix  $\sigma = 0.01$  throughout our experiments.

$$s_i = G(\mathbf{T}_{oc}^c \mathbf{x}, \mathbf{z}) \quad \text{and} \quad o_i = \begin{cases} 1 & s_i < -\sigma \\ -\frac{s_i}{2\sigma} & |s_i| \leq \sigma \\ 0 & s_i > \sigma \end{cases} \quad (2)$$

**Event Probabilities** When tracing points along the ray, the ray either terminates or escapes without hitting other points. These  $M + 1$  event probabilities can be defined as:

$$\phi_i = o_i \prod_{j=1}^{i-1} (1 - o_j), i = 1, \dots, M \\ \phi_{M+1} = \prod_{j=1}^M (1 - o_j) \quad (3)$$

**Rendered Depth and Rendering Term** With the probabilities defined above, the rendered depth value at each pixel  $\mathbf{u}$  can be computed as the expected depth value of the terminating point as in Eq. 4. To make it consistent, we set  $d_{M+1}$ , the depth value associated with escape probability, to a constant value  $1.1d_{max}$ , as in [36].

$$\hat{d}_{\mathbf{u}} = \sum_{i=1}^{M+1} \phi_i d_i \quad (4)$$

Since the rendering is fully differentiable, it can be integrated in our optimization. Unlike [36, 39], we perform ray-tracing in continuous space and do not require to discretize the object model. The final rendering term is as follows:

$$E_{rend} = \frac{1}{|\Omega_r|} \sum_{\mathbf{u} \in \Omega_r} (d_{\mathbf{u}} - \hat{d}_{\mathbf{u}})^2 \quad (5)$$

where  $\Omega_r = \Omega_s \cup \Omega_b$  is the union of surface pixels and pixels not on object surface but inside the 2D bounding box  $\mathcal{B}$ . Surface pixels  $\Omega_s$  are the same set of pixels used in Eq. 1, obtained by projecting the 3D reconstructed SLAM points onto the image masks as discussed in Sec. 3. The pixels in  $\Omega_b$  are assigned the same depth value as  $d_{M+1} = 1.1d_{max}$  and provide important silhouette supervision for our optimization since they penalize renderings that lie outside the object boundary, forcing empty space. As the pixels in  $\Omega_b$  do not require a depth measurement, we perform uniform sampling inside the 2D bounding box and filter out those inside the segmentation mask.

### 4.3. Optimization details

Our final energy is the weighted sum of the surface and rendering terms and a shape code regularization term:

$$E = \lambda_s E_{surf} + \lambda_r E_{rend} + \lambda_c \|\mathbf{z}\|^2 \quad (6)$$

The hyperparameter values used for optimization  $\lambda_s = 100$ ,  $\lambda_r = 2.5$  and  $\lambda_c = 0.25$  were tuned such that the Hessian matrices of the energy terms are of the same order of magnitude. Since all terms are quadratic, we adopt a Gauss-Newton optimisation approach with analytical Jacobians (Please refer to supplemental material for detail), initialized from a zero shape code  $\mathbf{z} = \mathbf{0}$ . The initialisation for the object pose  $\mathbf{T}_{co,0}$  is given by a LiDAR 3D detector [45] when LiDAR is available. In the monocular/stereo case, it is given by an image-based 3D detector [18] or by performing PCA on the sparse object point cloud.

## 5. Object SLAM

As an object-based SLAM system, DSP-SLAM builds a joint factor graph of camera poses, 3D feature points and object locations and poses. As Fig. 3 shows, the factor graph introduces object nodes and camera-object edges.

### 5.1. Object Data Association

Data association between new detections and reconstructed objects is an important step in object-level SLAM. We aim to associate each detection  $I$  to its *nearest* object  $o$  in the map, adopting different strategies depending on the different input modalities. When LiDAR input is available we compare the distance between 3D bounding box and reconstructed object. When only stereo or monocular images are used as input, we count the number of matched feature points between the detection and object. If multiple detections are associated with the same object, we keep the *nearest* one and reject others. Detections not associated with any existing objects are initialised as new objects and their shape and pose optimised following Sec. 4. For stereo and monocular input modes, reconstruction only happens when enough surface points are observed. For detections associated with existing objects, only the pose is optimised by running pose-only optimization and a new camera-object edge added to the joint factor-graph.

### 5.2. Joint Bundle Adjustment

Our joint map consists of a set of camera poses  $C = \{\mathbf{T}_{wc_i}\}_{i=1}^M$ , object poses  $O = \{\mathbf{T}_{wo_j}\}_{j=1}^N$  and map points  $P = \{w \mathbf{p}_k\}_{k=1}^K$ . Our joint BA can be formulated as a non-linear least squares optimization problem:

$$C^*, O^*, P^* = \arg \min_{\{C, O, P\}} \sum_{i,j} \|\mathbf{e}_{co}(\mathbf{T}_{wc_i}, \mathbf{T}_{wo_j})\|_{\Sigma_{i,j}} \\ + \sum_{i,k} \|\mathbf{e}_{cp}(\mathbf{T}_{wc_i}, w \mathbf{p}_k)\|_{\Sigma_{i,k}} \quad (7)$$

Diff.	Auto-labelling[47]				Ours			
	BEV@0.5	3D@0.5	NS@0.5	NS@1.0	BEV@0.5	3D@0.5	NS@0.5	NS@1.0
E	80.70	<b>63.96</b>	86.52	94.31	<b>83.31</b>	62.58	<b>88.01</b>	<b>96.86</b>
M	63.36	44.79	64.44	85.24	<b>75.28</b>	<b>47.76</b>	<b>76.15</b>	<b>89.97</b>

Table 1: Quantitative comparison of object cuboid prediction quality with Auto-labelling on KITTI3D on Easy and Moderate samples. Results of Auto-labelling are taken from their paper. Best results are shown as bold numbers.

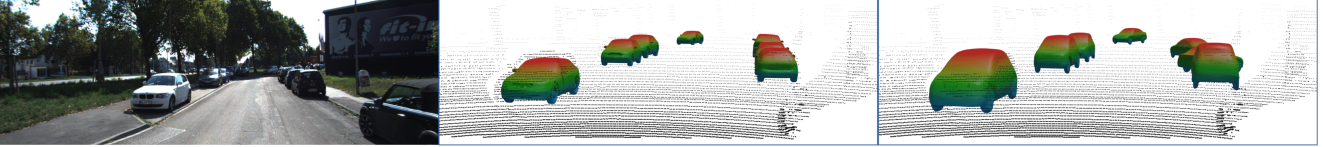


Figure 6: A qualitative comparison of shape reconstruction and pose estimation against Auto-labelling [47]. **Left:** input RGB image. **Middle:** result with DSP-SLAM **Right:** result with auto-labelling [47]

where  $e_{co}$  and  $e_{cp}$  represent the residuals for camera-object and camera-point measurements and  $\Sigma$  is the co-variance matrix of measurement residuals. Objects act as additional landmarks, which results in improvements in tracking performance as shown in our evaluation on KITTI. The optimization is solved with Levenberg-Marquardt in g2o [16].

**Camera-Object Measurements:** Object-camera poses  $\mathbf{T}_{co}$  are evaluated by minimising the surface alignment term in Eq. 1 while keeping the shape code and scale fixed. New pose observations serve as edges between camera pose  $\mathbf{T}_{wc}$  and object pose  $\mathbf{T}_{wo}$ , and the residual is defined as:  $e_{co} = \log(\mathbf{T}_{co}^{-1} \cdot \mathbf{T}_{wc}^{-1} \cdot \mathbf{T}_{wo})$  where  $\log$  is the logarithm mapping from  $\mathbf{SE}(3)$  to  $\mathfrak{se}(3)$ . Poses in the factor graph are 6-DoF, as object scale is only optimised when first detected.

**Camera-Point Measurements:** We use the standard formulation of re-projection error used in ORB-SLAM2 [22]:  $e_{cp} = \pi(\mathbf{T}_{wc}^{-1} \mathbf{w} \mathbf{p}) - \tilde{\mathbf{u}}$ , where  $\tilde{\mathbf{u}}$  is the measured pixel coordinate of map point  $\mathbf{p}$ . We follow a similar strategy as ORB-SLAM2 to tune  $\Sigma_{ij}$  such that the two energy terms contribute roughly the same to the overall optimization.

## 6. Experimental Results

We perform a quantitative evaluation of our novel prior-based object reconstruction optimisation, using LiDAR input on the KITTI3D Dataset [10], comparing with auto-labelling [47], the most related approach. In addition, we evaluate the camera trajectory errors of our full DSP-SLAM system on both stereo+LiDAR and stereo-only input on the KITTI Odometry [9] benchmark, comparing with state-of-the-art approaches. We also provide qualitative results of our full SLAM system on pure monocular input on Freiburg Cars [34] and Redwood-OS [5] Chairs dataset.

### 6.1. 3D Object Reconstruction

We conduct a quantitative comparison of our object pose estimation on the KITTI3D benchmark, against auto-labelling [47], a recent approach to prior-based object shape

and pose reconstruction based on image and LiDAR inputs, and using the same shape prior embedding (DeepSDF [25]) and similar level of supervision (object masks and sparse depth from the LiDAR measurements).

**Experimental Setting:** For a fair comparison, we evaluate our approach using a single image and LiDAR input and take the 2D segmentation masks and initial pose estimates from the auto-labelling code release [47] as initialization for our own optimization approach. We evaluate the results of pose estimation on the trainval split of KITTI3D which consists of 7481 frames, using the same metrics proposed in [47]: BEV AP @ 0.50, 3D AP @ 0.50, and the distance threshold metric (NS) from the nusenes dataset [3].

**Results:** We report quantitative results in Tab. 1. Our method achieves better performance under almost all metrics, especially on harder samples. We also visualize the comparison of reconstructed shapes and pose in Fig. 6. Auto-labelling [47] does not capture shape accurately for several vehicles: The first two cars on the left side are sedans, but auto-labelling [47] reconstructs them as "beetle"-shaped. In addition, some of the cars on the right side are reconstructed with incorrect poses which do not align with the image. In contrast, DSP-SLAM obtains accurate shape and pose.

**Timing Analysis:** To achieve close to real-time performance, we employ a Gauss-Newton solver with faster convergence than first-order methods during our optimization, leading to significant speed-ups. Tab. 3 shows a run-time comparison between a first-order optimizer and our Gauss-Newton solver with analytical gradients. Our method is approximately one order of magnitude faster to complete a single iteration, and requires fewer iterations to converge.

**Ablation Study:** We conducted an ablation study for DSP-SLAM with stereo+LiDAR input to analyse the effect of the number of LiDAR points used for shape optimization on the reconstruction error. Fig. 7 shows that there is no significant difference when reducing the number of LiDAR

Approach	00*	01	02*	03	04	Sequence		06*	07*	08*	09*	10	Average
	rpe/rre	rpe/rre	rpe/rre	rpe/rre	rpe/rre	05*	05*	rpe/rre	rpe/rre	rpe/rre	rpe/rre	rpe/rre	
SuMa++ [4]	<b>0.64/0.22</b>	1.60/0.46	1.00/0.37	0.67/0.46	<b>0.37/0.26</b>	0.40/0.20	0.46/0.21	<b>0.34/0.19</b>	1.10/0.35	<b>0.47/0.23</b>	<b>0.66/0.28</b>	<b>0.70/0.29</b>	
Ours St+LiDAR (250pts)	0.75/ <b>0.22</b>	<b>1.49/0.20</b>	<b>0.79/0.23</b>	<b>0.60/0.18</b>	0.47/0.11	<b>0.32/0.15</b>	0.39/0.21	0.52/0.28	<b>0.94/0.27</b>	0.79/0.28	0.69/ <b>0.26</b>	<b>0.70/0.22</b>	
Ours St+LiDAR (50pts)	0.80/0.24	<b>1.50/0.15</b>	0.84/0.26	0.61/0.18	<b>0.44/0.10</b>	0.32/0.16	<b>0.35/0.15</b>	0.57/0.24	1.03/0.30	0.78/0.27	0.67/0.30	<b>0.72/0.22</b>	
ORB-SLAM2 [22]	0.70/0.25	<b>1.38/0.20</b>	0.76/0.23	0.71/0.17	0.45/0.18	<b>0.40/0.16</b>	0.51/0.15	<b>0.50/0.28</b>	1.07/0.31	<b>0.82/0.25</b>	0.58/0.28	<b>0.72/0.22</b>	
St DSO [41]	0.84/0.26	<b>1.43/0.09</b>	0.78/ <b>0.21</b>	0.92/ <b>0.16</b>	0.65/0.15	0.68/0.19	0.67/0.20	0.83/0.36	<b>0.98/0.25</b>	<b>0.98/0.18</b>	<b>0.49/0.18</b>	0.84/ <b>0.20</b>	
St LSD-SLAM [7]	<b>0.63/0.26</b>	2.36/0.36	0.79/0.23	1.01/0.28	<b>0.38/0.31</b>	0.64/0.18	0.71/0.18	0.56/0.29	1.11/0.31	1.14/0.25	0.72/0.33	0.91/0.27	
DynaSLAM [1]	0.74/0.26	1.57/0.22	0.80/0.24	0.69/0.18	0.45/ <b>0.09</b>	<b>0.40/0.16</b>	0.50/0.17	0.52/0.29	1.05/0.32	0.93/0.29	0.67/0.32	0.76/0.23	
Ours St only	0.71/ <b>0.24</b>	1.45/0.30	<b>0.75/0.23</b>	0.73/0.19	0.47/0.11	0.57/0.23	0.57/0.22	0.51/0.29	1.02/0.32	0.87/0.26	0.65/0.31	0.75/0.25	
Ours St only (5Hz)	0.71/0.26	1.43/0.23	0.78/0.24	<b>0.67/0.18</b>	0.46/ <b>0.09</b>	<b>0.40/0.16</b>	<b>0.47/0.14</b>	0.52/0.29	0.99/0.31	0.90/0.28	0.63/0.31	<b>0.72/0.22</b>	

Table 2: Comparison of camera tracking accuracy - average  $t_{rel}$  [%] and  $r_{rel}$  [°/100m] against state-of-the-art stereo and LiDAR SLAM systems. Sequences marked with \* contain loops. Note that Stereo-DSO is a purely visual odometry system, so their result is without loop closing. We keep it in the table for completeness.

Method	Energy Terms	msec. / iter	# of iter
1st order	$E_{surf} + E_{rend}$	183	50
1st order	$E_{surf}$	88	50
Ours GN	$E_{surf} + E_{rend}$	20	10
Ours GN	$E_{surf}$	4	10

Table 3: Speed comparison between first-order optimization and our Gauss-Newton method with analytical Jacobians

points from 250 to 50. The reconstruction quality starts to degrade when the number of points is further reduced to 10.

## 6.2. KITTI Odometry Benchmark

We evaluate the camera trajectory error for our full DSP-SLAM system on the KITTI odometry benchmark with both stereo+LiDAR and stereo-only input. We evaluate on the 11 training sequences and compare with state-of-the-art SLAM systems of different input modalities using relative translation error  $t_{rel}$  (%) and relative rotation error  $r_{rel}$  (degree per 100m). Quantitative results are shown in Table 2.

**Stereo+LiDAR input:** The upper part of Tab. 2 shows trajectory errors of our system with stereo+LiDAR input. Results suggest our method achieves comparable results with SuMa++, a state-of-the-art LiDAR-based semantic SLAM system [4]. Note however, that our method only takes very few LiDAR points (several hundred per frame) while SuMa++ uses a full LiDAR point-cloud. It is interesting to see the comparison between our stereo+LiDAR system and stereo ORB-SLAM2, which is used as our backbone system. With our LiDAR-enhanced object reconstruction and joint BA, tracking accuracy improves on most sequences, especially 03, 05, 06, 08 where adequate number of static objects are observed throughout the sequence. However, our system performs slightly worse on some sequences which contain only moving objects (01, 04) or long trajectory segments where no static objects are observed (02, 10). The table also shows the effect on the camera trajectory error when using 250 vs 50 points for object reconstruction. The results suggest that the impact of reducing the number of points on camera tracking accuracy is minimal.

**Stereo-only input:** The lower part of Tab. 2 contains the

results of our stereo-only system. It can be seen that our stereo-only system performs slightly worse than stereo ORB-SLAM2, which means dense shape reconstruction and joint BA does not help improve tracking accuracy with stereo-only input. We argue that the reason is two-fold. Firstly, 3D measurements based on stereo images are noisier than LiDAR-based measurements, giving rise to lower accuracy in object pose estimates. Secondly, in the stereo-only case, the surface points are obtained from the SLAM system, where the same features are repeatedly measured and not from multiple (LiDAR) measurements. We also noticed that, to guarantee timings, we were performing bundle-adjustment less frequently than ORB-SLAM2. We re-ran DSP-SLAM, at a slightly reduced frame-rate (5Hz), performing BA after every key-frame (as ORB-SLAM2) and the average performance increased, matching ORB-SLAM2 at 0.72/0.22. A comparison with state-of-the-art stereo SLAM systems is also included in Tab. 2.

## 6.3. Freiburg Cars & Redwood-OS Dataset

Finally, we evaluate our SLAM system with monocular input on the Freiburg Cars dataset [34] and Redwood-OS Chairs dataset. Both datasets consist of object-centric sequences with the camera moving around the object. Demonstrations can be seen on Fig. 8 and 9 and in the supplementary video.

**Experimental Setting:** 3D Bounding boxes are estimated using PCA on the reconstructed surface points. Note that this approach cannot differentiate between the front and back side of the car. To address this issue, we initialize with two flipped hypothesis and keep the orientation that yields a smaller loss.

**Results:** Fig. 8 provides qualitative reconstruction results on 4 Freiburg Cars sequences. Our system is capable of reconstructing dense, accurate and high-quality shapes for cars solely from monocular input at 10-20 fps. Fig. 9 illustrates results on chairs from the Redwood-OS [5] dataset. Reconstruction accuracy is slightly worse than on cars as chairs have more complex shape variations. Results are promising nonetheless – our method still produces dense

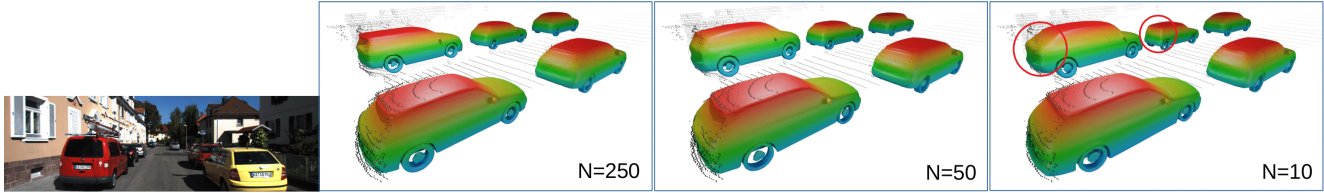


Figure 7: Object reconstruction results when using different number of LiDAR points per object ( $N=250, 50, 10$ ). There is no noticeable difference when the number of points is reduced from 250 to 50. The reconstruction quality starts to degrade when further reducing to 10. The degraded parts are marked with a red circle.

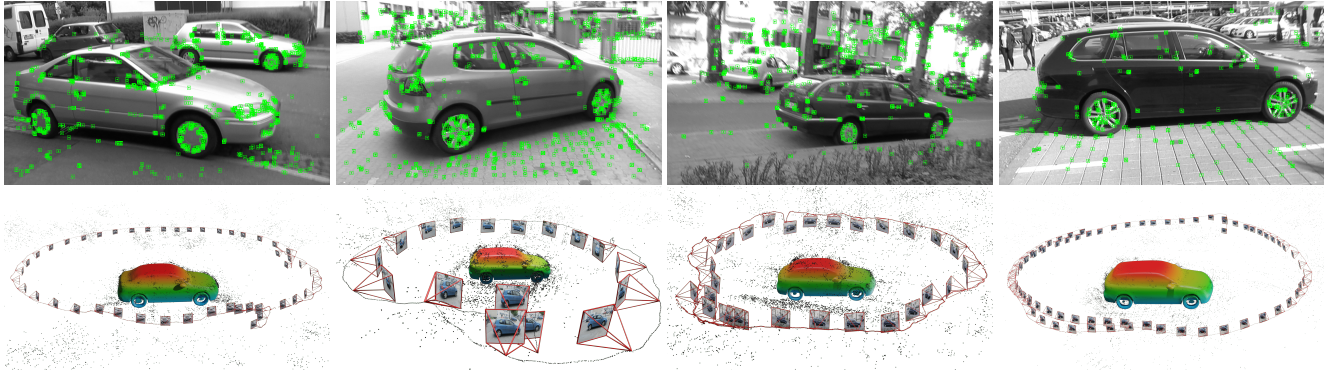


Figure 8: Qualitative results on Freiburg Cars dataset

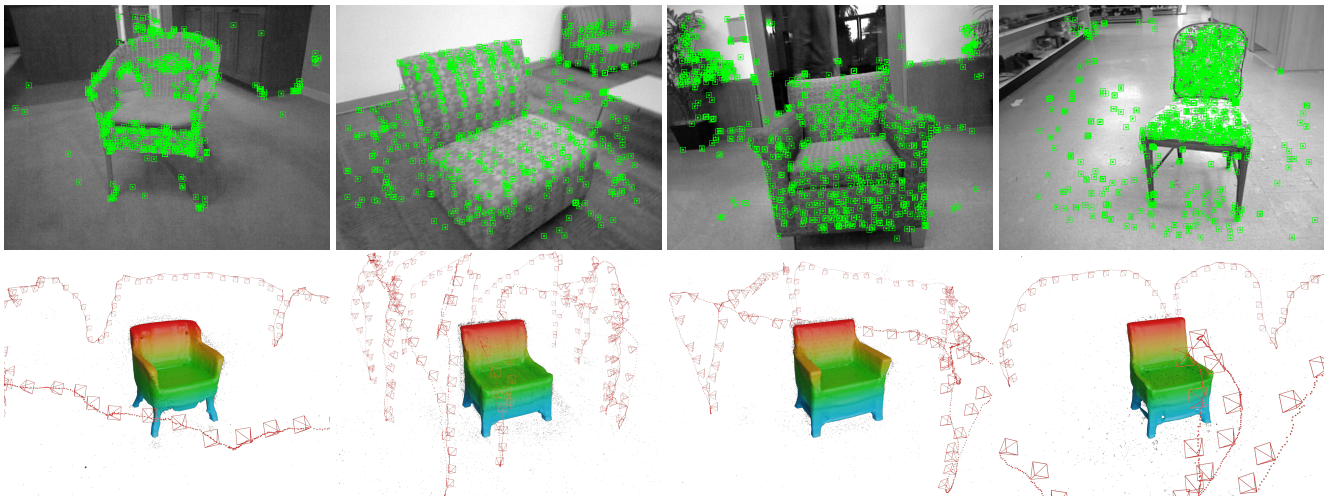


Figure 9: Qualitative results on Redwood-OS Chairs dataset

meshes that capture the overall object shape from monocular RGB-only sequences, in quasi-real time.

## 7. Conclusions

We have presented DSP-SLAM, a new object-aware real-time SLAM system that exploits deep shape priors for object reconstruction, produces a joint map of sparse point features for the background and dense shapes for detected objects. We show almost real-time performance on challenging real-world datasets such as KITTI (stereo and

stereo+LiDAR), and even on monocular setting Freiburg cars and Redwood-OS. Our quantitative comparisons with competing approaches on camera trajectory estimation and shape/pose reconstruction show comparable or superior performance to state of the art methods.

## Acknowledgements

Research presented here has been supported by the UCL Centre for Doctoral Training in Foundational AI under UKRI grant number EP/S021566/1. We thank Wonbong Jang and Adam Sherwood for fruitful discussions.



## References

- [1] B. Bescos, J. M. Fácil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018. [3](#), [7](#)
- [2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999. [3](#)
- [3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nusscenes: A multi-modal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. [6](#)
- [4] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019. [2](#), [3](#), [7](#)
- [5] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016. [3](#), [6](#), [7](#)
- [6] Amaury Dame, Victor A Prisacariu, Carl Y Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1288–1295, 2013. [3](#)
- [7] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015. [7](#)
- [8] Dorian Gálvez-López, Marta Salas, Juan D Tardós, and JMM Montiel. Real-time monocular object slam. *Robotics and Autonomous Systems*, 75:435–449, 2016. [3](#)
- [9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. [3](#), [6](#)
- [10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. [6](#)
- [11] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. [3](#)
- [12] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [2](#), [3](#)
- [13] Mehdi Hosseinzadeh, Kejie Li, Yasir Latif, and Ian Reid. Real-time monocular object-model aware sparse slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7123–7129. IEEE, 2019. [3](#)
- [14] Lan Hu, Yuchen Cao, Peng Wu, and Laurent Kneip. Dense object reconstruction from rgbd images with embedded deep shape representations. *arXiv preprint arXiv:1810.04891*, 2018. [3](#)
- [15] Lan Hu, Wanting Xu, Kun Huang, and Laurent Kneip. Deep-slam++: Object-level rgbd slam based on class-specific deep shape priors. *arXiv preprint arXiv:1907.09691*, 2019. [3](#)
- [16] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, May 2011. [6](#)
- [17] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 969–978, 2019. [3](#)
- [18] Zechen Liu, Zizhang Wu, and Roland Tóth. SMOKE: Single-stage monocular 3d object detection via keypoint estimation. *arXiv preprint arXiv:2002.10111*, 2020. [4](#), [5](#)
- [19] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. [3](#)
- [20] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level slam. In *2018 International Conference on 3D Vision (3DV)*, pages 32–41, Sep. 2018. [2](#)
- [21] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2017. [2](#), [3](#)
- [22] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. [2](#), [3](#), [4](#), [6](#), [7](#)
- [23] Mahyar Najibi, Guangda Lai, Abhijit Kundu, Z. Lu, V. Rathod, Tom Funkhouser, C. Pantofaru, D. Ross, L. Davis, and Alireza Fathi. Dops: Learning to detect 3d objects and predict their 3d shapes. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11910–11919, 2020. [3](#)
- [24] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2018. [3](#)
- [25] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. [3](#), [4](#), [6](#)
- [26] Parv Parkhiya, Rishabh Khawad, J Krishna Murthy, Brojeshwar Bhowmick, and K Madhava Krishna. Constructing category-specific models for monocular object-slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. [3](#)

- [27] Victor Adrian Prisacariu and Ian Reid. Shared shape spaces. In *2011 International Conference on Computer Vision*, pages 2587–2594. IEEE, 2011. 3
- [28] Victor Adrian Prisacariu, Aleksandr V Segal, and Ian Reid. Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. In *Asian conference on computer vision*, pages 593–606. Springer, 2012. 3
- [29] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 2
- [30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(6):1137–1149, 2017. 2
- [31] M. Runz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, Oct 2018. 2, 3
- [32] Martin Runz, Kejie Li, Meng Tang, Lingni Ma, Chen Kong, Tanner Schmidt, Ian Reid, Lourdes Agapito, Julian Straub, Steven Lovegrove, and Richard Newcombe. Frodo: From detections to 3d objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3
- [33] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H.J. Kelly, and Andrew J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 2, 3
- [34] N. Sedaghat and T. Brox. Unsupervised generation of a viewpoint annotated car dataset from videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 3, 6, 7
- [35] Jörg Stückler and Sven Behnke. Model learning and real-time tracking using multi-resolution surfel maps. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. 3
- [36] Edgar Sucar, Kentaro Wada, and Andrew Davison. Neural object descriptors for-multi view shape reconstruction. In *arXiv preprint arXiv:2004.04485*, 2020. 3, 4, 5
- [37] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B. Tenenbaum, and William T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3
- [38] Keisuke Tateno, Federico Tombari, and Nassir Navab. When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2295–2302, 2016. 3
- [39] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 4, 5
- [40] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Köhler, D. W. Murray, S. Izadi, P. Pérez, and P. H. S. Torr. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 75–82, 2015. 2
- [41] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017. 3, 7
- [42] Rui Wang, Nan Yang, Joerg Stueckler, and Daniel Cremers. Directshape: Photometric alignment of shape priors for visual vehicle pose and shape estimation. *arXiv preprint arXiv:1904.10097*, 2019. 3
- [43] WeiyueXu Wang, Qiangeng Xu, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *NeurIPS*, 2019. 3
- [44] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5231–5237, May 2019. 2, 3
- [45] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. 4, 5
- [46] Shichao Yang and Sebastian Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35(4):925–938, 2019. 3
- [47] S. Zakharov, Wadim Kehl, A. Bhargava, and Adrien Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12221–12230, 2020. 3, 6
- [48] Rui Zhu, Chaoyang Wang, Chen-Hsuan Lin, Ziyang Wang, and Simon Lucey. Object-centric photometric bundle adjustment with deep shape prior. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 894–902. IEEE, 2018. 3

# Supplemental Material

## DSP-SLAM: Object Oriented SLAM with Deep Shape Priors

### 1. Shape Priors for Object SLAM

In this section, we explain in more detail the benefit of using learnt shape priors in object SLAM. We evaluate the related object-oriented SLAM methods under three important properties: 1. *Can the system reconstruct previously unseen objects?* 2. *Is the reconstruction complete?* 3. *Does the reconstruction preserve detailed shape?*

The first line of works [6, 9] relies on a pre-scanned CAD model database to perform online detection and registration. The reconstructed object shapes are complete and detailed, but the system cannot handle new objects. The second category of works [5, 3] leverages 2D segmentation masks, treat each segmented object as individual identities and perform reconstruction on-the-fly. This reconstruction-by-segmentation strategy can reconstruct arbitrary unseen object shapes in great detail, but the reconstructed shape is incomplete due to partial observation and missing depth. The last line of research represents objects as simple geometric shapes, such as ellipsoids [4] or cuboids [13]. This kind of geometric representation can be applied to arbitrary unseen shapes and preserve some important properties like scale and orientation, but we’ve lost the level of details in the shapes.

With learnt shape priors, all the three properties can be achieved at the same time. Its generative property means it can generalize to unseen shapes. Object shape decoded from latent code is guaranteed to be detailed and complete, even from a single partial view. The compact representation also benefits the optimization process. Table 1 provides an overview of the related works under the different properties. Only NodeSLAM [8] and DSP-SLAM can achieve all the three important properties. Unlike NodeSLAM, DSP-SLAM can also deal with large scale environments and monocular RGB inputs.

### 2. Full Derivation of Jacobians

As mentioned in the main paper, one of our contribution is the fast Gauss-Newton optimizer, which is crucial to achieve real-time performance. This section provides full derivation of Jacobians of each individual residual term.

#### 2.1. Jacobian of Surface Term Residual

As shown in Eq. 1 in the main paper, the surface term residual at pixel  $\mathbf{u} \in \Omega_s$  is simply the SDF value of the back-projected point under object coordinate:

Method	Unseen Objects	Full Shape	Detailed Shape	RGB Input	Large Scene
2.5D is not enough[9]		✓	✓		
SLAM++ [6]		✓	✓		
Mask-Fusion [5]	✓		✓		
Fusion++ [3]	✓		✓		
Quadric-SLAM [4]	✓	✓		✓	✓
Cube-SLAM [13]	✓	✓		✓	✓
Node-SLAM [8]	✓	✓	✓		
<b>DSP-SLAM</b>	✓	✓	✓	✓	✓

Table 1: Comparison of the properties of DSP-SLAM with respect to other object-oriented SLAM systems.

$$e_s(\mathbf{u}) = G(\mathbf{T}_{oc}\pi^{-1}(\mathbf{u}, \mathcal{D}), \mathbf{z}) = G({}^o\mathbf{x}, \mathbf{z}) \quad (1)$$

Its Jacobian with respect to object pose and shape  $[\xi_{oc}; \mathbf{z}]$  is defined as:

$$\mathbf{J}_s = \frac{\partial e_s(\mathbf{u})}{\partial [\xi_{oc}; \mathbf{z}]} \quad (2)$$

where  $\xi_{oc} \in \mathbb{R}^7$  is the Cartesian representation (twist coordinate) of the corresponding element in Lie Algebra  $\mathfrak{sim}(3)$ . The Jacobian with respect to the shape code  $\frac{\partial e_s(\mathbf{u})}{\partial \mathbf{z}}$  could be obtained directly via back-propagation. To derive the Jacobian with respect to object pose  $\xi_{oc}$ , we first factorize it using chain rule:

$$\frac{\partial e_s(\mathbf{u})}{\partial \xi_{oc}} = \frac{\partial G({}^o\mathbf{x}, \mathbf{z})}{\partial {}^o\mathbf{x}} \frac{\partial {}^o\mathbf{x}}{\partial \xi_{oc}} \quad (3)$$

Then the first term  $\frac{\partial G({}^o\mathbf{x}, \mathbf{z})}{\partial {}^o\mathbf{x}}$  can also be obtained via back-propagation. The second Jacobian term could be computed by applying a left perturbation to the pose  $\mathbf{T}_{oc}$ :

$$\frac{\partial {}^o\mathbf{x}}{\partial \xi_{oc}} = \lim_{\delta \xi=0} \frac{\exp(\delta \xi^\wedge) \mathbf{T}_{oc} {}^c\mathbf{x} - \mathbf{T}_{oc} {}^c\mathbf{x}}{\delta \xi} \quad (4)$$

$$= \lim_{\delta \xi=0} \frac{(\mathbf{I} + \delta \xi^\wedge) \mathbf{T}_{oc} {}^c\mathbf{x} - \mathbf{T}_{oc} {}^c\mathbf{x}}{\delta \xi} \quad (5)$$

$$= \lim_{\delta \xi=0} \frac{\delta \xi^\wedge \mathbf{T}_{oc} {}^c\mathbf{x}}{\delta \xi} = \lim_{\delta \xi=0} \frac{\delta \xi^\wedge {}^o\mathbf{x}}{\delta \xi} \quad (6)$$

where  $\exp(\cdot)$  is the exponential map from Lie Algebra  $\mathfrak{sim}(3)$  to the corresponding Lie Group  $\mathbf{Sim}(3)$ , and  $\wedge$  is the hat-operator that maps a twist coordinate in  $\mathbb{R}^7$  to  $\mathfrak{sim}(3)$ :

$$\xi^\wedge = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\phi} \\ s \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}_\times + s\mathbf{I} & \boldsymbol{\nu} \\ \mathbf{0} & 0 \end{bmatrix} \quad (7)$$

$\boldsymbol{\nu} \in \mathbb{R}^3$ ,  $\boldsymbol{\phi} \in \mathbb{R}^3$  and  $s \in \mathbb{R}$  represent the translation, rotation and scale part of the twist coordinate respectively.  $(\cdot)_\times$  maps from  $\mathbb{R}^3$  to  $\mathfrak{so}(3)$  (skew-symmetric matrices). With the equation above, the Jacobian term in Eq. 6 can be computed as:

$$\lim_{\delta \xi=0} \frac{\delta \xi^\wedge {}^o\mathbf{x}}{\delta \xi} = \lim_{\delta \xi=0} \frac{\delta \boldsymbol{\phi}_\times {}^o\mathbf{x} + \delta s {}^o\mathbf{x} + \delta \boldsymbol{\nu}}{\delta \xi} \quad (8)$$

$$= \lim_{\delta \xi=0} \frac{\delta \boldsymbol{\nu} - {}^o\mathbf{x}_\times \delta \boldsymbol{\phi} + \delta s {}^o\mathbf{x}}{\delta \xi} \quad (9)$$

$$= \begin{bmatrix} \mathbf{I} & -{}^o\mathbf{x}_\times & {}^o\mathbf{x} \end{bmatrix} \quad (10)$$

The full Jacobian of the surface consistency term residual with respect to the object pose can be obtained by combining Eq. 3, 6, 10. Note that here we derive the Jacobian with slight abuse of notations and neglected the notation of homogeneous representation. For a more detailed explanation of Lie Theory, please refer to [1] or [7].

## 2.2. Jacobian of Rendering Term Residual

As stated in the main paper, the rendering term residual of pixel  $\mathbf{u}$  is

$$e_r(\mathbf{u}) = d_{\mathbf{u}} - \hat{d}_{\mathbf{u}} \quad (11)$$

To compute the Jacobian of the rendering terms  $\mathbf{J}_r$ , we can expand it using chain rule:

$$\mathbf{J}_r = \frac{\partial e_r}{\partial [\xi_{oc}; \mathbf{z}]} \quad (12)$$

$$= \sum_{k=1}^M \frac{\partial e_r}{\partial o_k} \frac{\partial o_k}{\partial s_k} \frac{\partial G({}^o\mathbf{x}_k; \mathbf{z})}{\partial [\xi_{oc}; \mathbf{z}]} \quad (13)$$

where  $\{{}^o\mathbf{x}_k\}_{k=1}^M$  is the depth-ordered set of sampled points along the ray of back-projecting pixel  $\mathbf{u}$ . The third

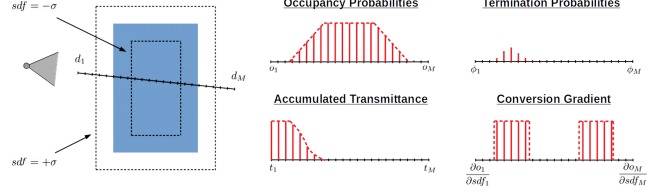


Figure 1: Demonstration of the depth rendering process. Only very few ray points contribute to the overall Jacobian term.

term  $\frac{\partial G({}^o\mathbf{x}_k; \mathbf{z})}{\partial [\xi_{oc}; \mathbf{z}]}$  has exactly the same form as the surface term Jacobian in Eq. 2, thus it can be computed following Sec. 2.1. The second term  $\frac{\partial o_k}{\partial s_k}$  is either a constant value or 0, as the sdf-occupancy conversion is linear inside the cut-off threshold and constant elsewhere.

$$\frac{\partial o_k}{\partial s_k} = \begin{cases} -\frac{1}{2\sigma} & |s_k| < \sigma \\ 0 & \text{elsewhere} \end{cases} \quad (14)$$

To compute the first term, we expand the residual term using Eq. 3 and Eq. 4 in the main paper following [10]:

$$\begin{aligned} e_r &= d_{\mathbf{u}} - \left( \sum_{i=1}^M d_i o_i \prod_{j=1}^{i-1} (1 - o_j) + d_{M+1} \prod_{j=1}^M (1 - o_j) \right) \\ &= \sum_{i=1}^M \psi(i) o_i \prod_{j=1}^{i-1} (1 - o_j) + \psi(M+1) \prod_{j=1}^M (1 - o_j) \\ &= \sum_{i=1}^{M+1} \psi(i) \prod_{j=1}^{i-1} (1 - o_j) - \sum_{i=1}^M \psi(i) \prod_{j=1}^i (1 - o_j) \\ &= \psi(1) + \sum_{i=1}^M (\psi(i+1) - \psi(i)) \prod_{j=1}^i (1 - o_j) \end{aligned}$$

where  $\psi(i) = d_{\mathbf{u}} - d_i$  is the depth residual for each of the sampled points along the ray. Differentiating with respect to the sample point, we reach:

$$\frac{\partial e_r}{\partial o_k} = \sum_{i=1}^M (\psi(i+1) - \psi(i)) \frac{\partial}{\partial o_k} \prod_{j=1}^i (1 - o_j) \quad (15)$$

$$= \sum_{i=k}^M (\psi(i) - \psi(i+1)) \prod_{j=1, j \neq k}^i (1 - o_j) \quad (16)$$

$$= \Delta d \sum_{i=k}^M \prod_{j=1, j \neq k}^i (1 - o_j) \quad (17)$$

As we are sampling uniformly along the ray, the coefficient  $\psi(i) - \psi(i+1) = d_{i+1} - d_i$  reduces to  $\Delta d$ . To understand this expression we can multiply  $(1 - o_k)$  on both sides, of Eq. 17, which gives us:



Figure 2: Results with GT (←) vs. MaskRCNN (→) masks

$$\frac{\partial e_r}{\partial o_k} (1 - o_k) = \Delta d \sum_{i=k}^M t_i \quad (18)$$

where  $t_i = \prod_{j=1}^i (1 - o_j)$  represents the accumulated transmittance at point  $o_k$  along the ray. Before hitting the surface,  $o_k$  is zero, so the term  $(1 - o_k)$  has no effect, and the Jacobian term becomes the sum of transmittance of all the points starting from point  $k$ . And after the ray entering the solid body, this Jacobian term will be zero. This means only points before entering the solid body contribute to the rendered depth value.

Now from Eq. 14 and Eq. 17, we already know that many Jacobian terms that make up the final Jacobian in Eq. 13 should be zero. Therefore, we could further speed up the optimization by not evaluating the third term  $\frac{\partial G(o_k, \mathbf{z})}{\partial [\xi_{oc}; \mathbf{z}]}$  at those points. Figure 1 is a demonstration of the rendering process and the sparse nature of the resulting Jacobians.

### 2.3. Jacobian of Prior Terms

Based on the shape regularisation energy defined in Eq. 6 in the main paper. The residual of this energy term is simply the shape code vector itself:

$$e_c = \mathbf{z} \quad (19)$$

Based on Eq. 19, we have

$$\frac{\partial e_c}{\partial \xi_{oc}} = \mathbf{0}, \quad \frac{\partial e_c}{\partial \mathbf{z}} = \mathbf{I} \quad (20)$$

Optionally, we can also apply structural priors such as constraining the object orientation to be aligned with the ground plane, as in [11]. We can define the rotation prior residual as

$$e_{rot} = 1 - \mathbf{R}_{co}(0 : 2, 1) \cdot \mathbf{n}_g \quad (21)$$

$$= 1 - [0 \ 1 \ 0] \mathbf{R}_{oc} \mathbf{n}_g \quad (22)$$

where  $\mathbf{R} \in \mathbf{SO}(3)$  denotes the rotation part of the transformation matrix, and  $(0 : 2, 1)$  represents the operation of getting the second column of the matrix.  $\mathbf{n}_g$  is the normal direction to the ground plane under camera coordinate frame. The Jacobian with respect to pose can be easily obtained following Eq. 10:

Component	Time (ms)
Mask R-CNN Detector	70 / frame
3D LiDAR Detector	60 / frame
Pose-shape Optimization	20×10 / new object
Pose-only Optimization	4×5 / vis. object

Table 2: Run-time analysis with system components

$$\frac{\partial e_{rot}}{\partial \xi_{oc}} = - [0 \ 1 \ 0] \frac{\partial \mathbf{R}_{oc} \mathbf{n}_g}{\partial \xi_{oc}} \quad (23)$$

$$= [0 \ 1 \ 0] [\mathbf{0} \ (\mathbf{R}_{oc} \mathbf{n}_g)_{\times} \ \mathbf{0}] \quad (24)$$

$$= [0 \ 0 \ 0 \ (\mathbf{R}_{oc} \mathbf{n}_g)_{\times} (1, 0 : 2) \ 0] \quad (25)$$

where  $(0 : 2, 1)$  represents the operation of getting the second row of the matrix. As this residual has no effect on object shape, we have:

$$\frac{\partial e_{rot}}{\partial \mathbf{z}} = \mathbf{0} \quad (26)$$

## 3. Experiment Details and Run-time Analysis

We evaluate the run-time performance of our full SLAM system (stereo+LiDAR) on a Linux system with Intel Core i9 9900K CPU at 3.60GHz, and an nVidia GeForce RTX2080 GPU with 8GB of memory. The 2D detection boxes and masks are obtained using MaskRCNN [2],<sup>1</sup> throughout all the experiments. Initial object poses are inferred using the LiDAR-based 3D b-box detector SECOND [12].<sup>2</sup> Our object reconstruction pipeline is implemented in Python with PyTorch. The back-end object-SLAM framework is implemented in C++ as an extension of the original ORB-SLAM2 implementation. The Python part is embedded into C++ using pybind11.

Table 2 lists the breakdown of the run-time performance of different major components. Note that all those components are performed only at key-frames. For KITTI sequences, we adopted two design choices to achieve real-time performance. Firstly, we noticed shape estimation did not improve much over time in KITTI, thus we perform full optimization only for new objects and only update poses for existing objects (as stated in the main paper). Secondly, we abort new object optimization whenever necessary to ensure new key-frame is inserted in time. These make our system able to operate at 10Hz on KITTI. In sequences such as Freiburg and Redwood-OS, it is beneficial to update the shape more frequently, with fewer GN iterations to guarantee real-time performance. Object meshes can be extracted optionally using marching cube with an extra GPU, for visualization only.

<sup>1</sup><https://github.com/facebookresearch/detectron2>

<sup>2</sup><https://github.com/traveller59/second.pytorch>

## 4. Ablation with GT masks

We have shown promising reconstruction results on cars on both KITTI and Freiburg dataset. Chairs have thin structures and a complex topology and are more challenging to reconstruct than cars. We noticed the thin legs of the last chair in Fig. 10 in the main paper were not properly reconstructed. This is because our reconstruction makes use of Mask-RCNN segmentation masks, which are noisy and affected by shadows. This can result in background points being associated to the chair, leading to incorrect results. We conducted ablation study using ground-truth masks as input. Fig. 2 illustrates an upper bound quality that could be achieved with clean GT masks.

## 5. More Qualitative Results

We show more qualitative reconstruction results in Fig. 3. For each scene the RGB image is shown on the top and the reconstruction results are shown underneath. We also show an example of reconstructed map in Fig. 4.

## References

- [1] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017. 2
- [2] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 3
- [3] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level slam. In *2018 International Conference on 3D Vision (3DV)*, pages 32–41, Sep. 2018. 1
- [4] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2018. 1
- [5] M. Runz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, Oct 2018. 1
- [6] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H.J. Kelly, and Andrew J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 1
- [7] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv*, abs/1812.01537, 2018. 2
- [8] Edgar Sucar, Kentaro Wada, and Andrew Davison. Neural object descriptors for-multi view shape reconstruction. In *arXiv preprint arXiv:2004.04485*, 2020. 1
- [9] Keisuke Tateno, Federico Tombari, and Nassir Navab. When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2295–2302, 2016. 1
- [10] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 2
- [11] R. Wang, N. Yang, J. Stueckler, and D. Cremers. Direct-shape: Photometric alignment of shape priors for visual vehicle pose and shape estimation. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 3
- [12] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018. 3
- [13] Shichao Yang and Sebastian Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35(4):925–938, 2019. 1



Figure 3: More qualitative results on object reconstruction from a single view-point

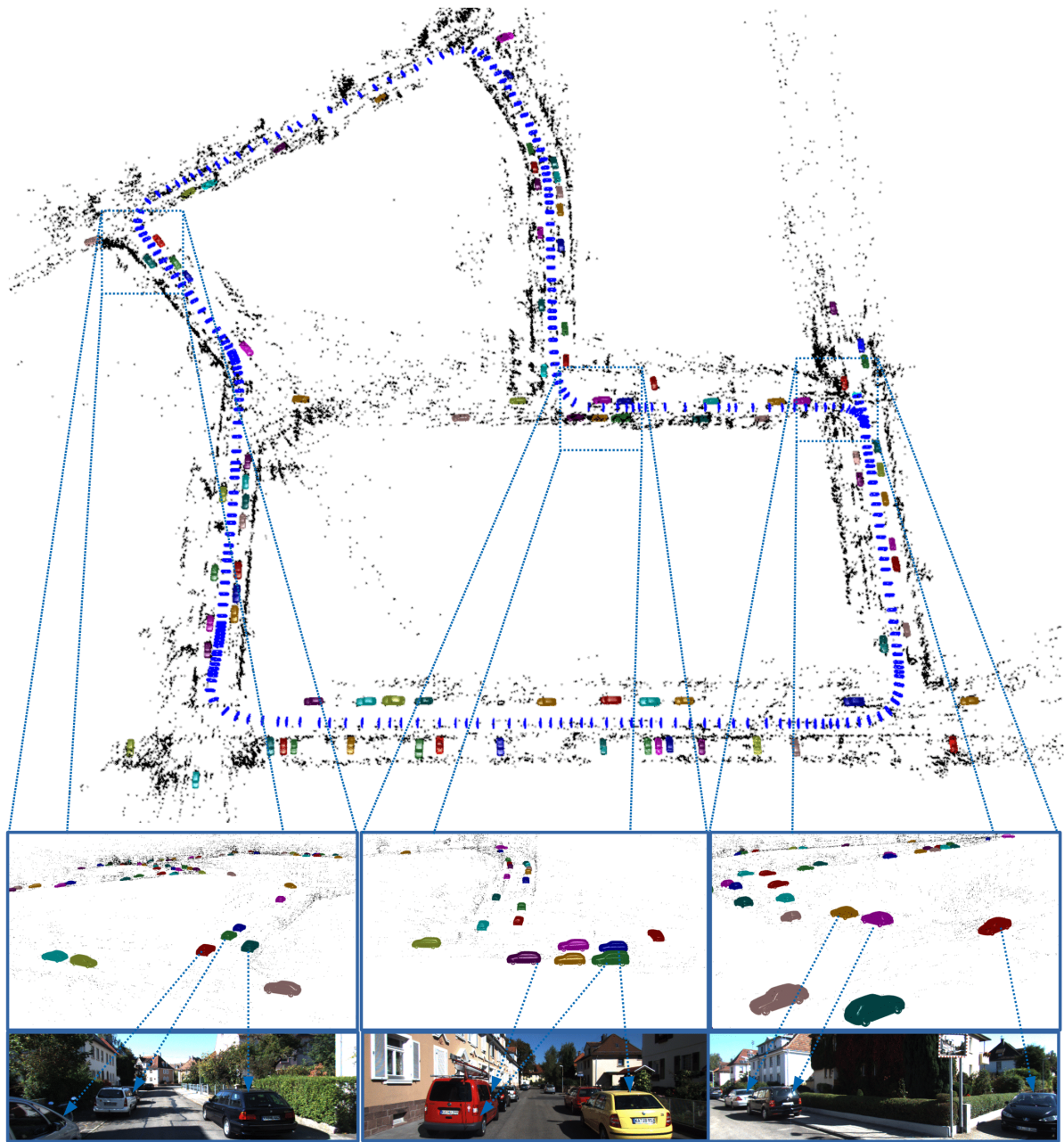


Figure 4: Reconstructed map of KITTI-07. Note the dense reconstruction of the objects and the shape variations.