SDFEst: Categorical Pose and Shape Estimation of Objects from RGB-D using Signed Distance Fields

Leonard Bruns and Patric Jensfelt

***Abstract*— Rich geometric understanding of the world is an important component of many robotic applications such as planning and manipulation. In this paper, we present a modular pipeline for pose and shape estimation of objects from RGB-D images given their category. The core of our method is a generative shape model, which we integrate with a novel initialization network and a differentiable renderer to enable 6D pose and shape estimation from a single or multiple views. We investigate the use of discretized signed distance fields as an efficient shape representation for fast analysis-by-synthesis optimization. Our modular framework enables multi-view optimization and extensibility. We demonstrate the benefits of our approach over state-of-the-art methods in several experiments on both synthetic and real data. We open-source our approach at `https://github.com/roym899/sdfest`.**

I. Introduction

We investigate the problem of joint pose and shape estimation of objects from RGB-D data. Pose estimation of known objects [1] and shape modeling of aligned objects [2], [3] have made significant progress in recent years, but the joint task has received less attention so far [4], [5]. Assuming knowledge of the full 3D model and pose of an object is common in various classic robotic algorithms, such as motion planning and grasp computation, but is not easy to achieve from partial sensor information. Furthermore, pose and shape estimation at a category level could be used in a mapping context to create an object-based world representation [6]. Such object-based representations could, for example, enable interaction with objects in virtual or augmented reality when only partial sensor information is available.

Methods inferring image-based abstractions, such as classified bounding boxes and instance masks, have made remarkable progress in recent years due to the availability of large annotated datasets [7]. However, using such abstractions in a robotics context remains challenging. To bridge this gap from image-based abstractions to an actionable representation, we build on this progress by using a classified instance mask as the starting point to extract a cropped point set of the object of interest with the goal of subsequently estimating the full 3D shape and pose.

Our work is inspired by the remarkable ability of humans to estimate the full shape of most objects from only a single view. This enables humans to grasp many objects without knowledge of the full shape or to plan more complicated tasks such as stacking. We hypothesize that this ability stems from integrating the prior knowledge accumulated from having seen many instances of an object category and the partial observation of the novel instance.

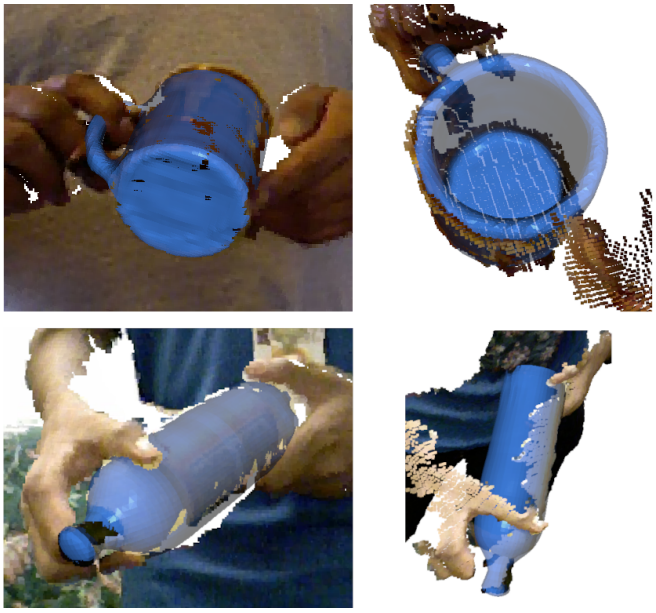


Fig. 1. Single-view pose and shape estimation results for two objects obtained with SDFEst (left: front view, right: alternative view).

In this work, we model this intuition as a per-category generative object model. This generative model is trained to compress the shape variations of an object category in a low-dimensional representation. To then estimate pose and shape from partial sensor information, we propose to train a point-based network on synthetic data generated by the generative object model. We further refine this initial estimate using a differentiable renderer to better match the observations from a single or multiple views.

We present a modular architecture for single- and multi-view pose and shape estimation of objects from a known category. Our approach, SDFEst, only requires a collection of categorized and aligned meshes at training time and estimates the 6D pose and shape of an object at inference time (see Fig. 1). Compared to purely discriminative approaches, SDFEst is modular in nature and allows optimizing the object's pose and shape from a single or multiple views by integrating a generative shape model with a differentiable renderer. Contrary to most existing approaches that use point sets, we use signed distance fields to represent shapes.

This work was supported in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

The authors are with the Division of Robotics, Perception and Learning (RPL), KTH Royal Institute of Technology, SE-10044 Stockholm, Sweden (e-mail: leonardb@kth.se; patric@kth.se).

To summarize, our contributions are:

- a novel extensible modular architecture for categorical pose and shape estimation from a single or multiple RGB-D images,
- a novel parametrization for multimodal orientation distributions, and
- an open-source implementation for pose and shape estimation using discretized signed distance fields (SDFs).

We compare our method to other related categorical pose and shape estimation methods and find that our method achieves state-of-the-art performance when poses are constrained and outperforms existing methods on unconstrained poses.

II. Related Work

We will summarize work of three related areas: RGB-based shape estimation, categorical pose and shape estimation, and optimizable shape and pose estimation.

A. RGB-based Shape Estimation

Several methods have been proposed to estimate the shape of an object from a single [8]–[13] or multiple RGB images [14], [15]. See Han et al. [16] for a recent survey on image-based 3D reconstruction.

In many cases, pose is not modeled explicitly and instead only the shape is predicted [8], [10], [14]. Although such an approach in principle can learn entangled representations of pose and shape, Zhu et al. [17] showed that explicitly modeling pose and predict shapes in a canonical reference frame reduces the learning complexity significantly and further allows finetuning of the pose estimation on real-world silhouette annotations.

Engelmann et al. [18] address the single RGB view multi-object case with a single-shot architecture. They frame the shape estimation problem as classification of the best matching shape. This allows them to decouple the shape estimation from the shape representation, but limits deformations to scalings of shapes in the database. Principled fusion of multiple such single-view predictions remains challenging.

In this work, we explicitly decouple pose, scale, and shape. We predict the shape in a canonical reference frame, which can be used to simplify downstream tasks such as grasp computation (e.g., grasping a mug from the top at the rim).

B. Categorical Pose and Shape Estimation

While pose estimation of known objects has matured significantly [1], pose estimation on a per-category level has only recently received more attention.

To estimate pose on a per-category level, [19] proposed the normalized object coordinate space (NOCS). In this space, objects of one category are aligned in a unit cube. To estimate the object pose, the projected NOCS coordinates (also called NOCS map) are predicted from the RGB image. This NOCS map and the observed depth map can be considered as correspondences, which together with the Umeyama algorithm [20] and RANSAC can be used to robustly estimate 6D pose and scale of the object. As part of their contribution, the authors also published the synthetic CAMERA dataset and the real-world REAL275 dataset. The latter being the most common dataset to evaluate categorical object pose estimation.

Building on these datasets proposed by [19], several methods were introduced to also address the categorical pose estimation problem. [21] proposed canonical shape space (CASS), which directly regresses a rotation matrix and translation vector from the observed point set and as a by-product also reconstructs the full point set. Shape prior deformation (SPD) [5] follows a similar idea but instead predicts deformations of a canonical point set for each category. To estimate the pose, they also use the NOCS. [22] and [23] extend SPD with a recurrent architecture and a transformer architecture for better shape adaptation, respectively. All of these methods train on a mix of mixed-reality images from the CAMERA dataset and real images from the REAL dataset. In contrast, the recently proposed ASM-Net [24] also estimates pose and shape, but showed that competitive results can be obtained by training on synthetic renderings of meshes only. Similar to ASM-Net, we also only require a collection of meshes for training.

Most methods (including ours) employ two-stage pipelines in which an object detection or instance segmentation module first detects bounding boxes or masks, which are later used to estimate the object's pose and shape. In contrast, Irshad et al. [25] proposed to use a single-shot architecture to detect objects and estimate their shape and pose jointly. While such an end-to-end approach might be easier to scale, data collection and data generation becomes more challenging compared to two-stage approaches, which can benefit from large-scale segmentation datasets.

In prior work [26], we showed that existing methods do not generalize well to unconstrained orientations due to the constrained orientations present in the CAMERA and REAL datasets. In this work, we propose a method that can achieve competitive results without constraining the orientation and outperforms existing approaches when constraining orientations to those included in the training set.

C. Optimizable Pose and Shape Estimation

Fewer works have investigated how to iteratively optimize the pose and shape given one or multiple observations. The approaches mentioned in the previous section are typically discriminative models, making it difficult to integrate additional information in a principled way. In contrast, the methods discussed in this section integrate a generative model into the estimation, which allows one to incorporate additional information by optimizing a latent representation, such that it matches one or multiple observations better.

FroDO [28] is a framework for pose and shape estimation of objects from bounding box detections in multiple views. The approach uses keypoints tracked over multiple RGB frames to formulate a loss that refines the shape descriptor. Most notably, FroDO employs DeepSDF [2], a continuous SDF representation, to represent the shape. MOLTR [29] is another RGB-based approach that also uses DeepSDF as the

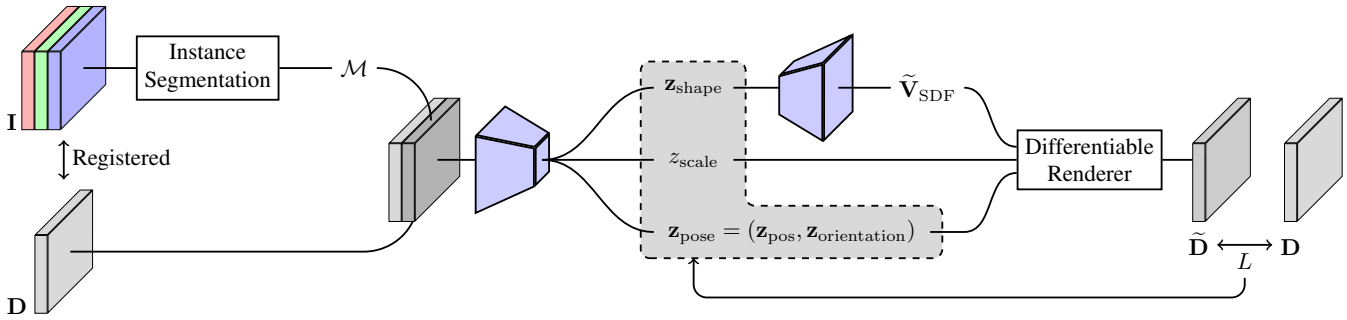


Fig. 2. Pipeline of SDFEst. We first apply instance segmentation and feed the masked point set into a PointNet-like network [27] which predicts the pose $\mathbf{z}_{\text{pose}}$, scale z_{scale}, and latent shape descriptor $\mathbf{z}_{\text{shape}}$. The decoder decodes $\mathbf{z}_{\text{shape}}$ into a discretized signed distance field (SDF) $\widetilde{\mathbf{V}}_{\text{SDF}}$ containing the full object shape in canonical pose. Given the pose, scale, and SDF, we render the depth map and iteratively optimize the latent representation by minimizing a loss L between the rendered depth map $\widetilde{\mathbf{D}}$ and measured depth map $\mathbf{D}$.

shape representation. MOLTR focuses on multi-object tracking, and instead of optimizing the shape descriptor through a loss, they fuse multiple single-view shape estimates through averaging. Compared to these works, we use discretized SDFs to represent shape and perform dense optimization on RGB-D data using a differentiable renderer.

Chen et al. [4] proposed an analysis-by-synthesis framework for pose and shape estimation of unknown objects. Based on a single RGB image, their approach samples and optimizes a large number of randomly sampled candidate poses to find the best matching candidate. Their approach does not allow direct extraction of the geometry; instead, novel views can be generated by their generative model. On the contrary, our method predicts an SDF, scale, and 6D pose, which can be used directly for geometric operations, such as collision checking or grasp planning.

In concurrent work, Deng et al. [30] introduced iCaps which, similar to our method, iteratively optimizes pose and shape from an initial estimate. Like FroDO and MOLTR, iCaps uses DeepSDF [2] to represent shapes. Instead, we investigate the use of discretized SDFs, which promises faster shape reconstruction and allow faster differentiable rendering, as shown in Section V-D. For faster optimization, iCaps alternates between pose refinement and discriminative shape estimation given a pose. In contrast, our approach jointly optimizes pose and shape. Furthermore, due to iCaps' discriminative shape estimation, incorporating additional observations requires modifying the approach, while our gradient-based joint optimization directly supports incorporation of additional observations.

Our pipeline is inspired by NodeSLAM [6], which employs a variational autoencoder (VAE) to model objects in a SLAM (simultaneous localization and mapping) framework. This VAE generates a discretized occupancy grid from a latent shape descriptor, which can be iteratively optimized using a probabilistic differentiable renderer. Orientation estimation in [6] is limited to a single axis by assuming objects to be upright on a table plane. Our work follows a similar idea to represent and optimize the object shape, but focuses on single-object, unconstrained, and possibly ambiguous object pose estimation.

III. Problem Definition

We study the problem of estimating 6D pose and shape at a per-category level. More formally, given an image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, depth map $\mathbf{D} \in \mathbb{R}^{H \times W}$, object mask $\mathcal{M} \subseteq \{(i,j) \in \mathbb{N}^2 \mid i \leq H, j \leq W\}$, and category $c \in \mathbb{N}$ we try to find estimates ${}^{\text{w}}\widetilde{\mathbf{T}}_{\text{o}}$ and $\widetilde{\mathcal{O}}$ of the true pose ${}^{\text{w}}\mathbf{T}_{\text{o}}$ and shape $\mathcal{O}$, respectively. We denote by o the object frame, by w the world frame, and by c the camera frame. We assume camera pose(s) ${}^{\text{w}}\mathbf{T}_{\text{c}}$ to be known. Given the availability of depth maps, we consider metrically scaled shapes.

IV. Method

In this section, we will first describe the overall pipeline (Section IV-A) and subsequently describe and motivate the design of the individual components (Section IV-B to IV-D). After introducing the components, we detail the inference step in Section IV-E. Finally, we provide further details of the training process in Section IV-F.

A. Pipeline Overview

Fig. 2 shows an overview of our proposed pipeline. The three main components are (from left to right) an initialization network (Section IV-C), a generative shape model (Section IV-B), and a differentiable renderer (Section IV-D). Together, these components enable initialization and iterative optimization of pose and shape in an analysis-by-synthesis framework (Section IV-E).

Our method, SDFEst, takes a cropped point set $\mathcal{P} \subset \mathbb{R}^3$ of the object as input[1], which we generate from the mask $\mathcal{M}$, depth map $\mathbf{D}$, and the camera projection matrix $\mathbf{P}$. The first part of our method is a novel initialization network, which estimates an initial pose and shape. The shape is predicted as a latent shape descriptor in a low-dimensional shape space learned by a VAE and a separate scalar scaling factor. This initialization gives a coarse estimate which is subsequently refined in an analysis-by-synthesis fashion. To do so, we use the decoder of our VAE to decode the latent shape descriptor into a discretized SDF, render it with a

[1]To simplify usage of our pipeline, we also integrate our method with Mask R-CNN [31] from Detectron2 [32] to enable inference starting from an RGB image $\mathbf{I}$ and depth map $\mathbf{D}$.

differentiable renderer, compare it with one or more observed depth maps, and compute an optimizable loss to iteratively refine the latent variables.

The embedded generative model is trained with complete shapes, not partial views. Therefore, it decodes the latent shape descriptor, which is inferred from a partial view, to the full reconstructed shape. That is, the pipeline finds the pose, scale, and normalized shape that best matches the observation.

B. Shape Modeling

To model per-category shape, we employ a VAE to find a low-dimensional, smooth representation of an object category's shape. The idea is that the VAE constrains the reconstructions to valid shapes from a category and hence automatically completes partial observations when fitting pose and shape to the observations. To represent the shape, we use discretized SDFs which can be easily converted to a mesh using the marching cubes algorithm [33].

We follow common practice for training VAEs [34] and jointly train an encoder

$$\begin{aligned} f_{\text{enc}} : \mathbb{R}^{R\times R\times R} &\to \mathbb{R}^N \times \mathbb{R}^N_{>0} \\ \mathbf{V}_{\text{SDF}} &\mapsto (\boldsymbol{\mu}, \boldsymbol{\lambda}) \end{aligned} \tag{1}$$

and a decoder

$$\begin{aligned} f_{\text{dec}} : \quad \mathbb{R}^N &\to \mathbb{R}^{R\times R\times R} \\ \mathbf{z}_{\text{shape}} &\mapsto \widetilde{\mathbf{V}}_{\text{SDF}}. \end{aligned} \tag{2}$$

Given a discretized SDF $\mathbf{V}_{\text{SDF}} \in \mathbb{R}^{R\times R\times R}$ of fixed resolution R, the encoder predicts the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\lambda}$ of a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\lambda}))$ with diagonal covariance in an N-dimensional latent space. The decoder reconstructs the SDF given a sample $\mathbf{z}_{\text{shape}} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\lambda}))$ from the latent space. During inference in the full pipeline, only the VAE's decoder is used.

As is standard for VAEs, we use Kullback-Leibler divergence to regularize the predictions. When computing the reconstruction error, we aim to give higher priority to correctly capturing the surface instead of the distance in empty space. To do so, we modify a simple reconstruction loss in two ways.

First, following [13], we increase the weight for distances below a threshold δ to give higher priority to correctly capturing the surface instead of the distance in empty space. Second, we give additional supervision to the reconstructed surfaces by rendering a depth map of a random view of $\mathbf{V}_{\text{SDF}}$ and creating a point set $\mathcal{P}$ from it. Applying trilinear interpolation at these points in the reconstructed SDF $\widetilde{\mathbf{V}}_{\text{SDF}}$ should yield 0. Therefore, we add the sum of these interpolations as an additional loss term.

The total loss function for training the VAE is given by

$$\begin{aligned} L_{\text{VAE}} =& \lambda_{<\delta} \|\mathbf{V}^{<\delta}_{\text{SDF}} - \widetilde{\mathbf{V}}^{<\delta}_{\text{SDF}}\|_2^2 \\ &+ \lambda_{\geq\delta} \|\mathbf{V}^{\geq\delta}_{\text{SDF}} - \widetilde{\mathbf{V}}^{\geq\delta}_{\text{SDF}}\|_2^2 \\ &+ \lambda_{\text{SDF}} \sum_{\mathbf{p}\in\mathcal{P}} |\text{trilinear}(\widetilde{\mathbf{V}}_{\text{SDF}}, \mathbf{p})|^2 \\ &+ \lambda_{\text{KLD}} D_{\text{KL}} \left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\lambda}) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})\right). \end{aligned} \tag{3}$$

We tuned the relative importance of these loss terms by visual inspection, such that unconditioned samples $\mathbf{z}_{\text{shape}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ gave good reconstructions.

C. Initialization Network

For each object category, we train an initialization network

$$\begin{aligned} f_{\text{init}} : \mathbb{R}^{M\times 3} &\to \mathbb{R}^3 \times \mathbb{R} \times [0,1]^G \times \mathbb{R}^N \\ \mathcal{P} &\mapsto (\mathbf{z}_{\text{pos}}, z_{\text{scale}}, \mathbf{o}, \mathbf{z}_{\text{shape}}). \end{aligned} \tag{4}$$

Given a point set $\mathcal{P}$ of variable size M (determined by the number of depth pixels within the object mask), the network predicts the position $\mathbf{z}_{\text{pos}}$, the scale z_{scale}, a distribution over orientations $\mathbf{o}$, and the latent shape vector $\mathbf{z}_{\text{shape}}$.

To handle ambiguous sensor data, we use a probabilistic orientation representation by discretizing $\text{SO}(3)$ and predicting the probability of the orientation being inside each grid cell. We use the base grid as specified by [35] since it uniformly discretizes $\text{SO}(3)$ and allows $\mathcal{O}(1)$ conversion from a continuous orientation to the corresponding index. While this introduces a discretization error, it allows the network to represent uncertainty and arbitrary distributions on $\text{SO}(3)$, which is important to handle objects with any (potentially view-dependent) symmetries using the same framework. In Section V-C we further show that, if multiple views are available, this probabilistic output can be used to identify the most certain initial orientation. Specifically, let $g : \text{SO}(3) \to \{1, ..., G\}$ denote the mapping from an orientation (we use unit quaternions for $\mathbf{z}_{\text{orientation}}$) to the index of the $\text{SO}(3)$ grid with G cells. Furthermore, let h denote the inverse operation. Note that in general $h(g(\mathbf{q})) \neq \mathbf{q}$, since a discretization error is introduced and unit quaternions are a double cover of $\text{SO}(3)$. The discretization error that this representation introduces in the initialization of the orientation will subsequently be removed during the optimization which operates on quaternions.

We use a PointNet-like architecture [27] and train it on synthetic point sets generated by sampling shapes from the VAE and randomizing their pose and scale. To avoid introducing any bias towards upright objects we use uniform distributions for position and orientation. However, we include a prior on possible sizes of an object category by specifying a distribution for the scale during training. We normalize the masked point set to have zero mean before passing it to the network. That is, the network's position output is only the offset from the masked point set's mean, which we will not denote explicitly in the following. Given such samples, we train the initialization network in a supervised manner with the following loss function:

$$\begin{aligned} L_{\text{init}} =& \lambda_{\text{pos}} \|\mathbf{z}_{\text{pos}} - \hat{\mathbf{z}}_{\text{pos}}\|_2^2 \\ &+ \lambda_{\text{orientation}} (-\log(o_{g(\hat{\mathbf{z}}_{\text{orientation}})})) \\ &+ \lambda_{\text{scale}} (z_{\text{scale}} - \hat{z}_{\text{scale}})^2 \\ &+ \lambda_{\text{shape}} \|\mathbf{z}_{\text{shape}} - \hat{\mathbf{z}}_{\text{shape}}\|_2^2 \end{aligned} \tag{5}$$

where $\hat{\cdot}$ indicates the sampled quantities and o_i denotes the ith element of $\mathbf{o}$. Since real masks will typically not perfectly align with the depth image, and RGB-D sensors exhibit noise

close to object edges, we augment the depth images prior to converting to a point set. We found this step to be crucial for usability on real-world data, where robust outlier rejection is difficult, depending on the object's shape and camera angle.

D. Differentiable Renderer

To enable analysis-by-synthesis optimization with the SDF we render the depth map using a differentiable renderer. We follow the idea of SDFDiff [36] and apply sphere tracing to quickly find the zero crossing in the SDF and use trilinear interpolation at the last step to compute derivatives with respect to pose, scale, and SDF.

E. Inference

The input to our network is the masked point set $\mathcal{P}$ in the camera frame, which is passed to the initialization network to obtain an initial position $\mathbf{z}_{\text{pos}}$, discretized orientation distribution $\mathbf{o}$, scale z_{scale}, and latent shape descriptor $\mathbf{z}_{\text{shape}}$. The discretized orientation distribution $\mathbf{o}$ is converted to a continuous unit quaternion $\mathbf{z}_{\text{orientation}} = h\left(\arg\max_i o_i\right)$. Since we never regress the quaternion directly, we do not suffer from the discontinuity issue discussed in [37].

Starting from this initial latent estimate, we decode the SDF, render it at the current pose, and compute a loss based on the observations. We combine two complementary losses, which are visualized in Fig. 3.

The SDF loss is computed by transforming the observed point set $\mathcal{P}$ into the object frame and interpolating the discretized SDF at the observed points. Since the observed points should lie on the object's surface, the interpolation should be close to 0. Therefore, we use the following loss:

$$L_{\text{SDF}} = \frac{1}{|\mathcal{P}|} \sum_{{}^{\text{c}}\mathbf{p} \in \mathcal{P}} |\text{trilinear}(\widetilde{\mathbf{V}}_{\text{SDF}}, z_{\text{scale}}^{-1}\, {}^{\text{o}}\widetilde{\mathbf{T}}_{\text{w}}\, {}^{\text{w}}\mathbf{T}_{\text{c}}\, {}^{\text{c}}\mathbf{p})|, \tag{6}$$

where ${}^{\text{o}}\widetilde{\mathbf{T}}_{\text{w}}$ is the 6D transformation computed from $\mathbf{z}_{\text{pose}}$.

The depth loss is computed based on the observed depth map $\mathbf{D}$ and estimated depth map $\widetilde{\mathbf{D}}$ as

$$L_{\text{depth}} = \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{(i,j) \in \widetilde{\mathcal{D}}} |\widetilde{D}_{i,j} - D_{i,j}|, \tag{7}$$

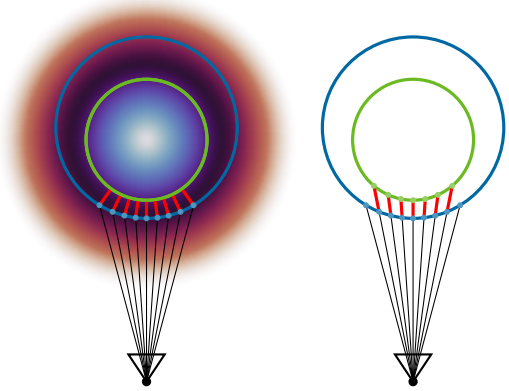
where $\widetilde{\mathcal{D}} = \{(i,j) \subseteq \mathcal{M} \mid D_{i,j} \neq 0 \wedge \widetilde{D}_{i,j} \neq 0\}$, that is, the set of pixels where both the masked depth map and the rendered depth map are valid.

The total loss is then computed as

$$L = \lambda_{\text{SDF}} L_{\text{SDF}} + \lambda_{\text{depth}} L_{\text{depth}}. \tag{8}$$

Since the whole pipeline is differentiable, we can use any first-order optimization algorithm to jointly optimize pose, scale, and shape of the object.

Our framework readily allows incorporating additional information from multiple views by summing up the per-view losses. Specifically, when K views and their poses ${}^{\text{w}}\mathbf{T}_{\text{c}k}, k = 1, ..., K$ are available, we evaluate (6)-(8) to retrieve per-view losses L_k and compute the total loss as $L = \sum_{k=1}^{K} L_k$.



(a) SDF (b) Depth

Fig. 3. Losses used to optimize the pose, scale, and shape of the object. The blue circle is the real observed object, green visualizes the current estimate, and red visualizes the losses. Note that each loss term captures different cases of misalignments, while neither by itself handles all.

F. Training Details

We train a separate VAE and initialization network for each object category. Our implementation is based on PyTorch [38] and Open3D [39]. Both networks are trained using Adam optimizer [40] with a learning rate of 1×10^{-3}.

1) VAE Training: We use CAD models from the ShapeNet dataset [41], convert them to SDFs with a resolution of $R = 64$, and manually remove erroneous CAD models (i.e., conversion to SDF failed, CAD model is misclassified / contains multiple objects / etc.) prior to training. The exact subsets are published as part of our open-source software.

To stabilize the VAE training, we set $\lambda_{\text{KLD}} = 0$ for the first 1000 iterations of training. This helped to avoid local minima where the encoder predicts $\boldsymbol{\mu} \approx \mathbf{0}$, and the decoder predicts only a constant (mean) value.

2) Initialization Network Training: To train the initialization network, we need to generate single-view point sets as close as possible to the expected preprocessed real-world point sets captured by an RGB-D camera. To do so, we sample $\mathbf{z}_{\text{shape}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, that is, from the prior distribution of the VAE. To generate random object positions, we first uniformly sample a pixel in the image plane (we generate 640x480 depth maps) to compute the ray on which the SDF center will be. We then sample a typical object distance along that ray to define the 3D position of the center. Next, we sample the scale of the SDF from per-category specified distributions. Finally, we uniformly sample a quaternion representing the orientation of the object. To robustify our network to typical outliers observed for noisy masks, we apply a random affine transformation to the mask and generate a uniform outlier value for the parts where the noisy mask does not overlap with the rendered depth image anymore. With a probability of 0.5, we further apply a Gaussian filter to the previously augmented depth image. We found this to be a simple way of simulating flying pixels at object boundaries commonly observed with RGB-D cameras. We train the initialization network by generating the data on the fly.

TABLE I

REAL275 RESULTS

	CASS	SPD	ASM	iCaps	Ours	Ours†
10°,2 cm	0.331	0.535	0.331	0.205	0.506	**0.589**
5°,1 cm	0.073	0.205	0.069	0.030	0.224	**0.242**
10°,2 cm,0.6	0.031	0.471	0.215	0.106	0.442	**0.491**
5°,1 cm,0.8	0.000	0.170	0.050	0.013	0.185	**0.191**

† Initialization constrained to orientations in REAL train split

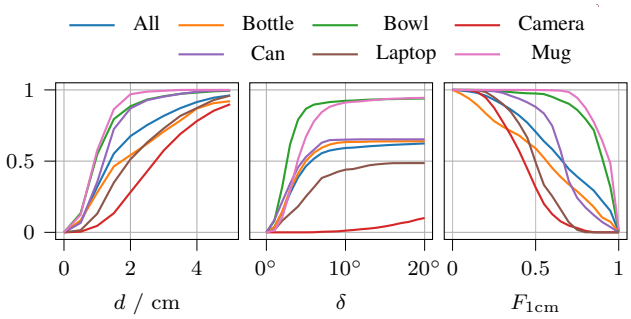


Fig. 4. Per-category estimation precision on REAL275 for varying position, orientation and $F_{1\text{cm}}$-score thresholds.

V. Experiments

In this section, we will discuss four experiments. First, we follow our proposed evaluation protocol of pose and shape estimation from real single-view data [26]. Second, we follow the evaluation protocol used by NodeSLAM [6] to compare pose and shape estimation with multiple views. Third, we perform an ablation study to assess the effect of the different components. Finally, we provide a run time analysis of our method.

Unless otherwise stated, we use the same networks ($N = 8, G = 576$) trained on synthetic data only and optimize for 50 iterations, which is typically sufficient for convergence.

A. Single-view Real Data

For all real-data experiments, we follow the protocol and evaluation metrics defined in [26]. To summarize, we report precision (i.e., # correct estimates / # total estimates) based on varying thresholds on position, orientation, and $F_{1\text{cm}}$-score. As baselines, we use CASS [21], SPD [5], ASM-Net [24], and iCaps [30]. For a fair comparison of pose and shape estimation, we use the same ground-truth masks and categories for all methods as described in [26]. In Table I and Table II we show results on REAL275 [19] and REDWOOD75 [26], [42], respectively. The green highlighted methods have only access to synthetic data.

On REAL275 (Table I), our method performs better than CASS, ASM-Net, and iCaps for all thresholds and on par with SPD. We further show results where our method is constrained to orientations present in the REAL training data (denoted by Ours†). With this modification, our method outperforms all other methods on REAL275, at the cost of poorer generalization for unconstrained orientations.

To better understand these results, we show the results of our unconstrained method for different categories and varying thresholds in Fig. 4. We observe that for cans and laptops the orientation precision of our method saturates at 40-60%. We find that in both cases, pose is ambiguous from geometry alone, and since our method was trained with a uniform orientation distribution, we cannot always infer the correct orientation. If we constrain[2] the initial orientations to those in the REAL train split, these ambiguities disappear, which explains the improved performance in Table I.

On REDWOOD75 (Table II), our unconstrained method outperforms the baselines by a large margin. To analyze this, we show a qualitative comparison for typical REDWOOD75 samples in Fig. 5. We find that other methods, which were trained on constrained orientations, fail at estimating objects with unconstrained orientations. Our method, on the other hand, only uses synthetic data and, therefore, we were able to generate unconstrained orientations during training.

TABLE II

REDWOOD75 RESULTS

	CASS	SPD	ASM	iCaps	Ours
10°,2 cm	0.013	0.200	0.307	0.270	**0.653**
5°,1 cm	0.000	0.013	0.080	0.080	**0.466**
10°,2 cm,0.6	0.000	0.173	0.173	0.20	**0.586**
5°,1 cm,0.8	0.000	0.013	0.053	0.040	**0.413**

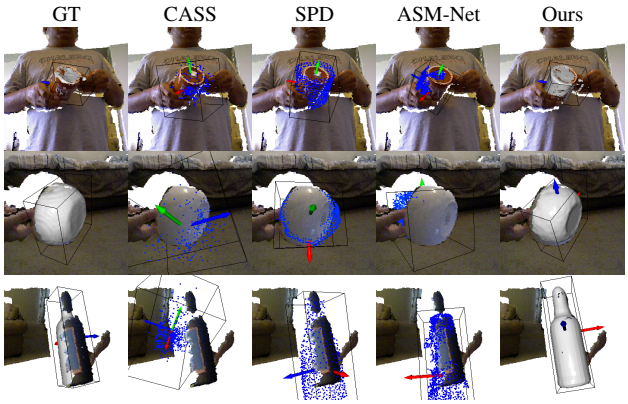


Fig. 5. Comparison of the estimated pose and shape on REDWOOD75 data. CASS and SPD, which are trained on datasets that only contain upright objects, fail at estimating unconstrained orientations. The exact training distribution of ASM-Net is unknown. Our method is trained with uniformly distributed orientations and therefore performs significantly better on unconstrained orientations. See Table II for quantitative results.

B. Multi-view Synthetic Data

To quantitatively compare the shape completion and pose estimation capabilities of our approach with NodeSLAM's [6], without their code available, we attempt to replicate their evaluation procedure: meshes from the ShapeNet [41] mug category are scaled by 0.1 (making them approximately 6 cm large), 1, 2, and 3 camera orientations are uniformly sampled and the camera positions are chosen such that the origin of the mesh lies 30 cm in front of each camera on its principal axis. We use Open3D [39] to render the

[2]We use our orientation discretization to change the prior distribution without retraining any network.

input images. Note that some details, such as the image resolution and camera parameters, were unknown to us, and the experimental setup might not be completely the same. For this experiment, we only optimize for 30 iterations similar to NodeSLAM. Additionally, we perform 6D pose estimation, whereas [6] only describes orientation estimation around the object's up-axis. We report the same metrics as [6]: reconstruction precision P, chamfer distance CD, thresholded reconstruction precision $\mathrm{P}_{1\mathrm{cm}}$, and thresholded reconstruction recall $\mathrm{R}_{1\mathrm{cm}}$.

The comparison between our results and those reported in [6] is shown in Table III. Our method achieves better results on most metrics. It can be seen that both methods successfully improve the estimates with the additional information provided by additional views. Note that such a multi-view optimization is not readily supported by the single-view methods we compared with in Section V-A.

TABLE III
MULTI-VIEW COMPARISON WITH NODESLAM [6]

	1 view		2 views		3 views	
	Ours	[6]	Ours	[6]	Ours	[6]
P/mm	4.625	**4.459**	**3.290**	3.752	**2.572**	3.484
CD/mm	**3.782**	4.439	**2.818**	3.854	**2.330**	3.648
$\mathrm{P}_{1\mathrm{cm}}/\%$	**86.53**	–	**92.80**	–	**96.40**	–
$\mathrm{R}_{1\mathrm{cm}}/\%$	**94.55**	93.49	**96.62**	95.72	**97.53**	96.07

C. Ablation Study

To verify the effectiveness of different components in our pipeline, we performed an ablation study using the 3-view experimental setup described in Section V-B. Table IV summarizes the results. First, instead of initializing with a random first view, we assume that all views are available and initialize based on the view that gives the highest probability in the orientation distribution (*best view*). It can be seen that this simple change further improves the results. Qualitatively, we found that unambiguous views score reliably higher than ambiguous ones, which is consistent with the improved results. *Depth loss only* and *SDF loss only* show that none of the losses are sufficient by themselves, which supports the discussion in Section IV-E. We further investigate the importance of initialization and iterative optimization. *Init only* shows that skipping the optimization completely gives significantly worse results, which is expected due to the quantization error that is introduced due to the $\mathrm{SO}(3)$ discretization. Skipping only the shape optimization (*no shape opt.*) also gives worse results, but to a lesser degree. Initializing the shape to the mean shape $\mathbf{z}_{\mathrm{shape}} = \mathbf{0}$, but optimizing it, similar to [6], also gives significantly worse results. Finally, we compare our discretized $\mathrm{SO}(3)$ grid to regressing a single *quaternion* and using a *finer grid*. Both perform worse. For quaternions, the most likely reason for the drop in performance is the inability of handling multimodal distributions. The finer grid resolution converged slower during training than the coarser one we used for our experiments. We believe that training for a longer time might close the performance gap.

TABLE IV
ABLATION STUDY

	P/mm	CD/mm	$\mathrm{P}_{1\mathrm{cm}}/\%$	$\mathrm{R}_{1\mathrm{cm}}/\%$
First view ($G = 576$)	2.776	2.439	95.56	97.55
Best view	2.586	2.290	96.43	97.96
Depth loss only	2.866	2.918	94.90	93.49
SDF loss only	3.464	2.827	94.58	97.66
Init only	8.095	6.432	67.26	87.75
No shape opt.	2.917	2.582	95.28	97.22
Mean shape init	3.066	2.612	93.75	97.14
Quaternion	3.866	3.193	90.67	96.15
Finer grid ($G = 4608$)	3.458	2.888	92.64	96.83

D. Run Time Analysis

In general, analysis-by-synthesis approaches impose run time limitations on the network and components used in the optimization loop. Here, we break down the run time of our pipeline into its different components, namely, the initialization network, the SDF decoder, and the differentiable renderer. Table V shows the resulting run times on a GeForce GTX 1070 Mobile when optimizing for 50 iterations, which is typically sufficient for convergence (rendering is performed at a resolution of 640×480). Our custom CUDA implementation achieves rendering times below $6\,\mathrm{ms}$ (for a single forward and backward pass), which is significantly faster than the decoding step. This indicates that the bottleneck in our analysis-by-synthesis pipeline is the decoding of the latent representation into the SDF, not the differentiable rendering. Including the decoding step, rendering a single image takes around $30\,\mathrm{ms}$.

To compare with the rendering of continuous neural field representations, DIST [43], a differentiable renderer for continuous SDFs, requires $0.99\,\mathrm{s}$ to render a single 512×512 image (on a GeForce GTX 1080 Ti) despite various tricks to improve the run time. This highlights the drawback of neural field representations, which require thousands of evaluations during rendering. The discretized shape representation of this work only requires a single forward pass to render from the latent variables.

TABLE V
RUN TIME BREAKDOWN OF THE PIPELINE

	Time (ms)	% of Total
Segmentation ($\times1$)	268.23	15.50
Initialization ($\times1$)	4.67	0.27
Decoding-Forward ($\times50$)	46.22	2.67
Rendering-Forward ($\times50$)	8.53	0.49
Losses ($\times50$)	136.06	7.86
Decoding-Backward ($\times50$)	910.98	52.63
Other-Backward ($\times50$)	247.41	14.29
Total	1731.02	

VI. LIMITATIONS

General limitations inherent to analysis-by-synthesis methods apply to our method. Although our method achieves better results than existing discriminative methods, this improvement comes at the cost of slower run time. Furthermore, the

complexity of all components included in the optimization loop must be limited to avoid slow optimization. Therefore, we chose to use one VAE per category to limit the network size of the decoder.

A possible way towards real-time application would be to use a more complex network for the initialization. This might allow for direct inference of a higher-quality estimate and would reduce the number of necessary optimization iterations. Furthermore, iterative optimization could be performed online with changing sensor data in a tracking fashion [30].

We observed that shape estimation is currently mostly limited to interpolations of shapes seen in training. Generalization to novel shapes is limited. Training on more shapes, cross-category, and other generative models might be possible ways towards more general shape estimation.

Furthermore, our method does not take into account the color information, and pose and shape are estimated based on depth data only.

VII. Conclusion and Outlook

We presented an architecture to estimate pose and shape of an object. The architecture can be used for single- and multi-view estimation. Our approach is trained on unconstrained orientations and is capable of handling ambiguous views during training due to our $\mathrm{SO}(3)$ parametrization. If the poses in an environment are known to be constrained, such constraints can easily be incorporated into our framework. We open-source our approach to facilitate further research in this area.

Our modular architecture naturally lends itself to various extensions and modifications. The differentiable renderer is currently only used during inference and to generate a depth map. Modifying it to render other modalities and using it for end-to-end training with unannotated data could be an interesting research direction. Other future directions include multi-category models, single-stage training, and a fully probabilistic pose (and shape) estimation framework.

Acknowledgement

The authors thank Raghav Bongole for his contributions to the software repository.

References

[1] T. Hodaň, *et al.*, "BOP challenge 2020 on 6D object localization," in *ECCV*, 2020.
[2] J. J. Park, *et al.*, "DeepSDF: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019.
[3] L. Mescheder, *et al.*, "Occupancy networks: Learning 3D reconstruction in function space," in *CVPR*, 2019.
[4] X. Chen, *et al.*, "Category level object pose estimation via neural analysis-by-synthesis," in *ECCV*, 2020.
[5] M. Tian, M. H. Ang, and G. H. Lee, "Shape prior deformation for categorical 6D object pose and size estimation," in *ECCV*, 2020.
[6] E. Sucar, K. Wada, and A. Davison, "NodeSLAM: Neural object descriptors for multi-view shape reconstruction," in *3DV*, 2020.
[7] T.-Y. Lin, *et al.*, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.
[8] J. Wu, *et al.*, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *NeurIPS*, vol. 32, 2016.
[9] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *CVPR*, 2017.
[10] T. Groueix, *et al.*, "A papier-mâché approach to learning 3D surface generation," in *CVPR*, 2018.
[11] J. Wu, *et al.*, "MarrNet: 3D shape reconstruction via 2.5D sketches," in *NeurIPS*, vol. 30, 2017.
[12] N. Wang, *et al.*, "Pixel2Mesh: Generating 3D mesh models from single RGB images," in *ECCV*, 2018.
[13] Q. Xu, *et al.*, "DISN: deep implicit surface network for high-quality single-view 3D reconstruction," in *NeurIPS*, vol. 32, 2019.
[14] C. B. Choy, *et al.*, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *ECCV*, 2016.
[15] H. Xie, *et al.*, "Pix2Vox++: multi-scale context-aware 3d object reconstruction from single and multiple images," *IJCV*, vol. 128, no. 12, 2020.
[16] X.-F. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *TPAMI*, vol. 43, no. 5, 2019.
[17] R. Zhu, *et al.*, "Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image," in *ICCV*, 2017.
[18] F. Engelmann, *et al.*, "From points to multi-object 3D reconstruction," in *CVPR*, 2021.
[19] H. Wang, *et al.*, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *CVPR*, 2019.
[20] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *TPAMI*, vol. 13, no. 04, 1991.
[21] D. Chen, *et al.*, "Learning canonical shape space for category-level 6D object pose and size estimation," in *CVPR*, 2020.
[22] J. Wang, K. Chen, and Q. Dou, "Category-level 6D object pose estimation via cascaded relation and recurrent reconstruction networks," in *IROS*, 2021.
[23] K. Chen and Q. Dou, "SGPA: Structure-guided prior adaptation for category-level 6D object pose estimation," in *ICCV*, 2021.
[24] S. Akizuki and M. Hashimoto, "ASM-Net: Category-level pose and shape estimation using parametric deformation," in *BMVC*, 2021.
[25] M. Z. Irshad, *et al.*, "CenterSnap: Single-shot multi-object 3D shape reconstruction and categorical 6D pose and size estimation," in *ICRA*, 2022.
[26] L. Bruns and P. Jensfelt, "On the evaluation of RGB-D-based categorical pose and shape estimation," in *IAS*, 2022.
[27] C. R. Qi, *et al.*, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *CVPR*, 2017.
[28] K. Li, *et al.*, "FroDO: From detections to 3d objects," in *CVPR*, 2020.
[29] K. Li, H. Rezatofighi, and I. Reid, "MOLTR: Multiple object localization, tracking and reconstruction from monocular RGB videos," *RAL*, vol. 6, no. 2, 2021.
[30] X. Deng, *et al.*, "iCaps: Iterative category-level object pose and shape estimation," *RAL*, vol. 7, no. 2, 2022.
[31] K. He, *et al.*, "Mask R-CNN," in *ICCV*, 2017.
[32] Y. Wu, *et al.*, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.
[33] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *SIGGRAPH*, 1987.
[34] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *ICLR*, 2014.
[35] A. Yershova, *et al.*, "Generating uniform incremental grids on SO(3) using the Hopf fibration," *IJRR*, vol. 29, no. 7, 2010.
[36] Y. Jiang, *et al.*, "SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization," in *CVPR*, 2020.
[37] Y. Zhou, *et al.*, "On the continuity of rotation representations in neural networks," in *CVPR*, 2019.
[38] A. Paszke, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *NeurIPS*, vol. 32, 2019.
[39] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv preprint arXiv:1801.09847*, 2018.
[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
[41] A. X. Chang, *et al.*, "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015.
[42] S. Choi, *et al.*, "A large dataset of object scans," *arXiv preprint arXiv:1602.02481*, 2016.
[43] S. Liu, *et al.*, "DIST: Rendering deep implicit signed distance function with differentiable sphere tracing," in *CVPR*, 2020.