PolarMesh: A Star-Convex 3D Shape Approximation for Object Pose Estimation

Fu Li, Ivan Shugurov, Benjamin Busam, Minglong Li, Shaowu Yang, and Slobodan Ilic

***Abstract*—In this letter, we introduce PolarMesh as a star-convex approximation of a 3D object based on spherical projection and can be applied to monocular object pose and shape estimation. The proposed PolarMesh can be stored in a discrete 2D map that allows a trivial conversion between it and the object surface, as the adjacent map elements form faces of the 3D model. The fixed-grid 2D nature of this object surface approximation ensures that it can be estimated from an image patch of the object with an auto-encoder neural network. Moreover, the PolarMesh not only encodes the object shape but also its orientation. Consequently, it is used for accurate estimation of object orientation and can be extended to the full 6D pose estimation using a projection-based method for translation. An exhaustive quantitative evaluation on LineMod, LineMod-Occlusion and HomebrewedDB shows the competitive performance of the proposed method compared to other monocular RGB pose estimation pipelines. Furthermore, we tested the generalization capabilities of the representation on the recent NOCS dataset for category-level pose estimation and achieved very compelling results in terms of predicted rotations.**

***Index Terms*—Shape representation, object pose estimation.**

I. Introduction

MOVING an object in space manipulates its position and orientation. The knowledge of these 6 degrees of freedom (DoF) is crucial in our automated world where vehicles learn to drive autonomously, augmentation and manipulations change our perception of image content [1], and collaborative robots support medical treatments [2]. Accurate estimation of poses is thereby the basis for these setups to function properly.

Manuscript received September 9, 2021; accepted January 14, 2022. Date of publication February 1, 2022; date of current version February 24, 2022. This letter was recommended for publication by Associate Editor A. I. Comport and Editor C. C. Lerma upon evaluation of the reviewer's comments. The work of Fu Li was supported by China Scholarship Council (CSC). This work was supported by the National Natural Science Foundation of China under Grants 61803375 and 91948303. *(Corresponding author: Shaowu Yang.)*

Fu Li is with the Institute for Quantum Information and State Key Laboratory of High Performance Computing, College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China, and also with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany (e-mail: lifu11@nudt.edu.cn).

Ivan Shugurov and Slobodan Ilic are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany, and also with the Siemens AG, 81739 Munich, Germany (e-mail: ivan.shugurov@tum.de; slobodan.ilic@siemens.com).

Benjamin Busam is with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany (e-mail: b.busam@tum.de).

Minglong Li and Shaowu Yang are with the Institute for Quantum Information and State Key Laboratory of High Performance Computing, College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: liminglong10@nudt.edu.cn; shaowu.yang@nudt.edu.cn).

Digital Object Identifier 10.1109/LRA.2022.3147880

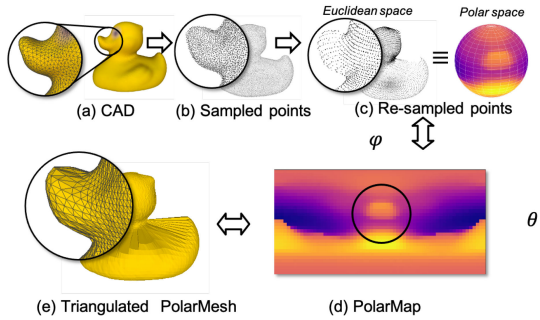

Fig. 1. PolarMesh generation. The polygon mesh (a) is uniformly sampled with a set of points (b). These points are re-sampled based on polar coordinates into vertices in polar space (c). Then, a 2D PolarMap representation has been created as a 2D grid matrix (d). From there the faces can be build and an approximate 3D mesh model generated (e).

This fundamental 3D vision problem has been addressed by scholars for decades, and is particularly difficult if the estimation is done only from a single RGB image [3]–[8]. Recent research tackles the ill-posed nature of this problem in a data-driven way where the pose is usually computed with respect to a 3D CAD model [9]–[18]. Only recently, Wang *et al.* [19] proposed a more generic method (NOCS) to estimate both metric pose and shape in the absence of a 3D model. While NOCS initiated *category-level* pose estimation for objects of the same class with RGB-D data, the two consecutive frameworks [20], [21] only require RGB input for the same task.

In all these methods, the parametrization of the underlying models can vary. For example, DPOD [16] encodes correspondences via UV-texture map representation, AAE [22] uses auto-encoder to encode object shape from images. These methods do not explicitly encode object shape, but rather target to use some 2D representation that is easy to learn with CNNs. Other methods, like DeepSDF [23] explicitly encode object shape represented via the signed distance function. This representation has been coupled with correspondence estimation in AutoLabel [24] letter used for 9D pose and scale estimation problem in RGB-D images applied for class of cars. These shape representations, however, do not contain pose information, and when used for pose estimation require PnP+RANSAC or differentiable rendering steps for the accurate pose estimation.

In this letter, we propose PolarMesh, an object shape approximation based on the spherical projection, which embeds object shape and orientation. We explore the spherical nature of polar coordinates and formulate a 2D shape representation leveraging

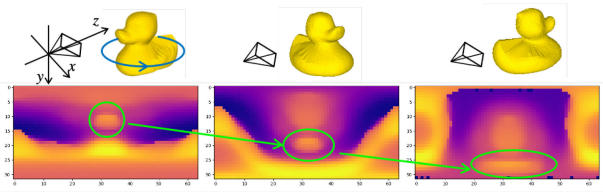

Fig. 2. PolarMesh Rotation. As the PolarMesh M is rotated, the 2D PolarMap P makes a transformation as the green arrow shows.

a star-convex model approximation that allows to parametrize a 3D model on a 2D grid. More precisely, each axis of the 2D map represents rotation along with one of the two angles needed to parametrize a sphere. The value at a particular map location represents the distance of the radially convex object hull from the centroid. We learn a direct mapping from RGB to this PolarMesh which naturally extends to novel objects. The process of PolarMesh generation is depicted in Fig. 1. The orientation-aware characteristics of the PolarMesh allow us to formulate a pose estimation framework on top of it. To this end, our contribution is twofold:

- *PolarMesh:* We propose a star-convex 3D model approximation based on spherical projection for object 6D pose estimation.
- *PM-Net:* A pipeline that supports the learning of a mapping from RGB to PolarMesh proves the advantage of the representation for the task of 6D pose estimation by achieving compelling performance on LineMod, LineMod-Occlusion, HomeBrewedDB and NOCS datasets.

II. Related Works

The first type of Object pose estimation methods that gained popularity in the recent years are dense correspondence-based methods [11], [16], [19], [25]–[28]. While being different in implementation, their common denominator is the key idea to train a neural network to predict 2D-3D correspondences between each object pixel in the image and the 3D location of the corresponding point on the object's surface. Those correspondences are used either with PnP+RANSAC [5], [29] or the Umeyama algorithm [30] to compute the 6D object pose. DPOD [16] proposed to use discrete UV maps to uniquely parametrize the object surface. With this parameterization, the UNet-like network [31] predicts two discrete UV coordinates for each visible pixel occupied by the object. Pix2Pose [26] and CDPN [11], on the other hand, use 3D normalized vertex coordinates to parameterize the correspondences, and two-stage detectors. CDPN also uses the correspondences only to estimate rotation. Translation is predicted directly. NOCS [19] uses MaskRCNN [32] and a normalized object coordinate space to predict correspondences. However, the main focus of the letter is scale and 6D pose estimation of objects not seen during training. EPOS [25] extends the idea of dense correspondences to a discrete object fragment. Prediction of fragments for each pixel and establishing many-to-many 2D-3D correspondences allow for effective handling of symmetric objects. GDR [18] extends EPOS by the geometry-guided dirtect pose regression from the correspondences and fragments outputs.

On the other hand, CosyPose [17] and Self6D [33] use a similar pose parameterization which allows direct pose prediction. They predict the pose by estimation of the 2D centre location of the object and its distance. Combined with the intrinsic parameters of the camera, it gives the estimate of the translational component of the 6D pose. Allocentric rotation parameterization is used for simpler rotation prediction. Self6D is first trained on synthetic data and then fine-tuned on real data without pose annotations in a self-supervised manner. CosyPose also proposes a refinement pipeline utilizing un-calibrated multiple frames. However, in this letter we compare only to the monocular version of CosyPose without the multi-view refinement procedure and without the ICP refinement as we want to explore monocular pose estimation without depth. AAE [14], on the opposite, relies on manifold learning to match a descriptor of a given object patch to a precomputed database of descriptors for the object under various rotations. Translation is estimated based on the object size in the image. MHP [34] predicts multiple pose hypothesis to estimate the pose of symmetric or occluded objects. Following NOSC, NAbs [20] obtains the category-level object pose by inversely refining the pose and latent code input of their well trained image generation network to minimize the discrepancy between the generated images and target images.

Recently, spherical projections have been used for 3D shape retrieval [35], classification [36] and shape completion [37] tasks. In [37], the predicted depth is first converted to the unit sphere. The shape is then completed from this partial observation. Differently from them, our proposed Polarmap stores distances from the object surface to the object center for the pre-defined polar angles. To the best of our knowledge, we are the first applying the spherical representation for object pose estimation.

III. PolarMesh

A. PolarMesh Parameterization

One of the most popular ways to represent 3D models is polygonal mesh. A polygonal mesh $M = (V, F)$ is mathematically defined by a set of vertices $V \subset \mathbb{R}^3$ and a set of triangular faces $f = (v_1, v_2, v_3) \in F$ with $v_i \in V, i \in \{1, 2, 3\}$. The non-parametric nature of polygon meshes allows for arbitrary vertex density changes to model fine geometry precisely. However, the potentially irregular structure is not particularly suitable for current deep learning techniques. To circumvent this problem, we propose a simplistic structure approximation of the surface based on spherical projection [37]. We named it **PolarMesh**. The proposed representation is based on polar coordinate discretization which allows for regular sampling to simplify the use of deep learning approaches. The PolarMesh is stored as a regularly-structured 2D matrix of pre-defined resolution. Additionally, it allows for a simple conversion from the original polygon model to the representation and back to the approximate polygonal model.

Fig. 1 illustrates the steps to convert a polygonal mesh into PolarMesh. First, a polygonal mesh is transformed into a point

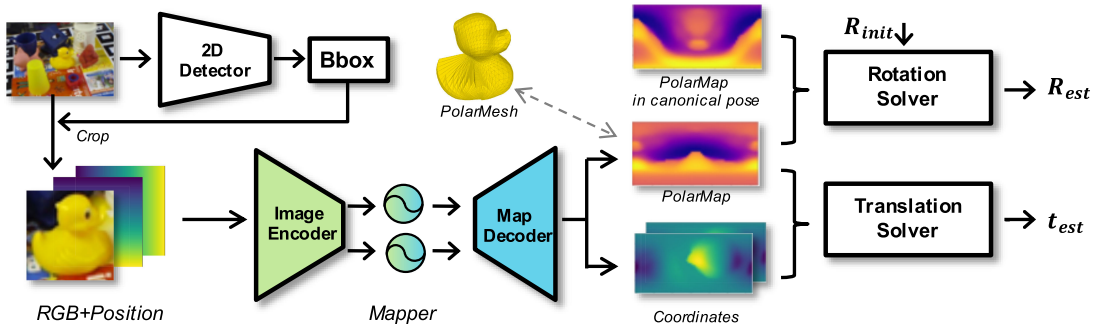

Fig. 3. Pipeline. Firstly, 2D detector detects the object of interest, Then, a batch of resized (128×128) images with two extra pixel positional channels is passed through a feature extractor that encodes a feature vector which is then passed through a fully connected mapper before being decoded to the PolarMap and coordinates. In the end, rotation is initialized by directly regressing a rotation matrix hypothesis(R_{init}), which is then further optimized by aligning the rotated canonical PolarMap to the predicted PolarMap. Translation is estimated by the projection-based PnP+RANSAC method.

cloud by uniform sampling [38] of points on its surface. Then, the resulting vertices $v = (x, y, z) \in \mathbb{R}^3$ are first transformed into polar coordinates with the origin in the object center where $v(x, y, z) = v(\theta, \phi, r)$ with the polar angle $\theta \in [0, \pi)$, the azimuthal angle $\phi \in [0, 2\pi)$ and the radial distance $r \in \mathbb{R}_0^+$. From these coordinates, we filter vertices based on the angular proximity in the spherical coordinate system. This is indicated as re-sampling in Fig. 1(c). A regular sampling with θ into N bins $I \ni i$ and ϕ into 2 N bins $J \ni j$ allows for a 2D grid-like representation of the mesh with vertices V where

$$P : [0, \pi) \times [0, 2\pi) \times \mathbb{V} \to I \times J \times \mathbb{R}_0^+ \tag{1}$$

$$(\theta, \phi, V) \mapsto (i(\theta), j(\phi), p(i, j, V)). \tag{2}$$

The third coordinate is defined by the maximal surface point distance in the spherical segment with

$$p(i, j, V) = \max\{\|v(\theta, \phi, r)\| \mid v \in V, (\theta, \phi) \mapsto (i, j)\} \tag{3}$$

This allows us to represent the unstructured mesh with a 2D matrix of entries $(i, j, p(i, j, V))$, which we call PolarMap, as shown in Fig. 1(d). By triangulating the vertices of this representation we can define **PolarMesh** $M^p = (V^p, F^p)$, shown in Fig. 1(e), with

$$V^p = \{v_{i,j} = (i, j, p(i, j, V)) \mid i \in I, j \in J\} \tag{4}$$

$$F^p \ni (v_{i,j}, v_{i+1,j}, v_{i+1,j+1}), (v_{i,j}, v_{i,j+1}, v_{i+1,j+1}) \tag{5}$$

While the approximation with a star-convex hull neglects some minor shape details, it allows us to represent the unstructured shape as a structured 2D grid. Moreover, this representation not only encodes an approximation of the original 3D model but also contains its pose information. The PolarMesh is essentially a surface approximation, which inevitably leads to the loss of the surface detalization. The PolarMap representation can be viewed as an object-centred spherical depth map of the object surface. The uniform sampling in polar coordinates allows for an image-like representation with the downside of denser samples near the poles. This is out-weighted by a number of advantages:

- *Implicit rotation parametrization:* The PolarMesh naturally encodes the exact object orientation in model coordinates as shown in Fig. 2 which allows for intuitive use in rotation estimation.
- *CNN friendly:* The image-like structure allows for direct processing with off-the-shelf 2D convolutional neural networks, whereas polygon meshes and point clouds require special architectures.
- *Flexibility:* The PolarMesh resolution is controlled by the discretization level N which can be adjusted depending on the objects and tasks at hand.
- PolarMeshes are good approximation of the object 3D shapes. They are **water-tight models** and by definition without self-intersections or invisible internal structures.

Our key idea is to train a CNN, which takes an RGB patch containing the object and construte a PolarMesh in terms of estimating 2D PolarMap depending on the observed object rotation.

B. Orientation and Translation

On any triangular mesh $M_0^p \in \mathcal{M}_0^p$ in some canonical pose, rotation operation $R \in SO(3)$ can be applied as $\mathcal{M}_0^p \times SO(3) \to \mathcal{M}^p$ resulting in rotated mesh $M^p \in \mathcal{M}^p$. We depict in Fig. 2 how the rotation of the mesh is reflected in its 2D PolarMap for some discrete rotations. It is evident that a rotation of the 3D model results in a transformation in the PolarMap. This brings us to our key intuition that given an image patch I of the object seen from some viewpoint, we can recover its corresponding PolarMesh in that pose using an autoencoder-like neural network. We use a large number of image patches depicting the object from a different viewpoints for training. It can be either real or synthetic data. Particularly, the standard rotation is converted to the allocentric rotation as [21] to ensure that the visual appearance of the object is dependent only on rotation. In this way, the rotation is implicitly encoded into the predicted PolarMesh.

Our proposed PolarMesh M^p encodes object orientation. The vertices of the PolarMesh represent the 3D vertices of the object model. Translation information, which is missing in our PolarMesh representation, can be recovered from 2D-3D

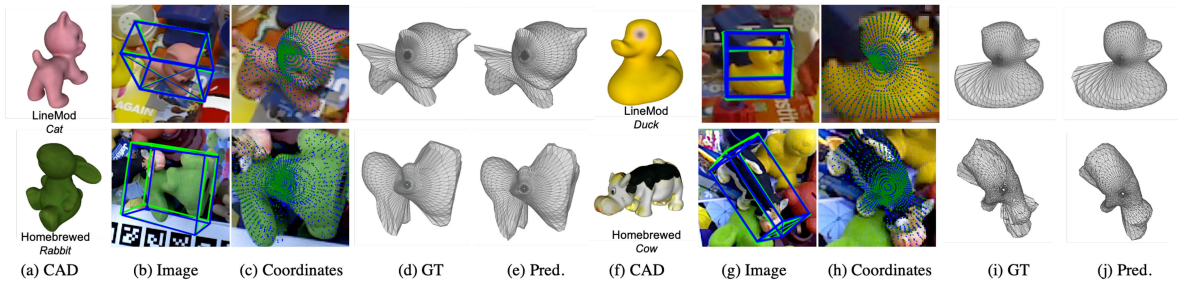

Fig. 4. Pose and shape results for LineMod and Homebrewed dataset. (a), (f) are the Object CAD models, and (b), (g) are the images with estimated (blue) and ground truth (green) poses. (c), (h) are the images with our predicted Coordinates (blue) and the groud truth (green). (d), (i) show our ground truth PolarMeshes generated from the CAD model in image view. (e), (j) are our predicted PolarMeshes.

correspondences established between the image pixels and PolarMesh M^p. Therefore, we also estimate projected location of the PolarMesh vertices $p_{ij} \in M^p$ onto the 2D image patch as $C^p \in \mathbb{R}^{2\times N\times 2^{\bar{}}N}$, $c_{ij} \in C^p$ and call them PM_{Coord}. To be more specific, C^p contains two channels of the same size as Polarmap, which represents x and y coordinates. The x and y are the predicted coordinates of the PolarMesh vertices projected onto the image plane. The 2D-3D correspondences are constructed using the estimated C^p and M^p. Translation is then estimated with a projection-based RANSAC method.

Therefore, we aim at training a neural network which takes cropped object images as input and directly regresses proposed PolarMesh in terms of PolarMap representation P, and correspondence in terms of the coordinates C^p conditioned on the observed object rotation $R \in SO(3)$. Thus, our problem can be formalized as:

$$\arg\min_{\theta} \mathcal{L}(f_\theta(\mathrm{I}), (P, C^p) \mid R) \tag{6}$$

where θ defines the best network parameters to approximate the ground truth PolarMap P and C^p with the prediction f_θ given the object image patch I in pose with orientation R. A naive approach would directly estimate 3D object shape from a given patch. This estimated 3D shape will be in some canonical pose regardless of the input patch. Thus, the pose information is lost. In our case, since the predicted PolarMap representation encodes object rotation in respect to the object's canonical pose, we can estimate this rotation in a subsequent post-processing step.

IV. PM-Net: A PolarMap and Coordinates Predictor

The overall three-stage pipeline is shown in the Fig. 3. The first two stages depend on separately trained convolution neural networks. The third stage contains two parts: a purely optimization-based rotation and translation estimation. The initial rotation regression is performed with the direct regression neural network in this stage. The first stage represents some off-the-shelf 2D detector trained on ground truth crops of the objects of interest. In practice, we used YOLOv3 [39] or MaskRCNN [32] depending on the dataset. In the second stage, an auto-encoder Encoder-Decoder like network (PM-Net) is trained on images I to predict PolarMap P_{pred} and its projected coordinates C^p_{pred} in a fully supervised fashion. Below, we define our loss functions used in the second stage of our pipeline.

A. Loss Functions of PM-Net

The loss function is defined similarly to other depth regression works [40], [41]. Our loss function is composed of PolarMap and coordinates loss: $\mathcal{L} = w_1\mathcal{L}_{PM} + w_2\mathcal{L}_{Coord}$. The two losses are identically formulated, so for the space reasons we define in details only PolarMap loss. PolarMap loss is defined between the predicted PolarMap P_{pred} and the ground truth PolarMap P_{gt} representation obtained from the input training image like:

$$\mathcal{L} = \lambda_1\mathcal{L}_d + \lambda_2\mathcal{L}_{grad} + \lambda_3\mathcal{L}_{normal} + \lambda_4\mathcal{L}_{adp}, \tag{7}$$

with the loss terms $\mathcal{L}_i \in [\mathcal{L}_d, \mathcal{L}_{grad}, \mathcal{L}_{normal}, \mathcal{L}_{adp}]$ and the weights $\lambda_i \in [\lambda_1, \lambda_2, \lambda_3, \lambda_3]$. The first term $\mathcal{L}_d$ is the element-wise $\mathcal{L}_1$ loss that measures the average distance between these two PolarMap representations written as:

$$\mathcal{L}_d = \mathrm{Avg}\left(W_d F\left(\|P_{pred} - P_{gt}\|_1\right)\right) \tag{8}$$

where $F(x) = ln(x + \alpha)$, $\mathrm{Avg}(x)$ determines the mean. $W_d \in \mathbb{R}^{N\times 2N}$ are the weights to correct the distortion in our PolarMap P.

The sphere distortion problem are studied in [35], [42]. They design an size-adaptive kernel while doing convolution over the feature map, or they use the normal convolution layer but input the corrected images for the classification task. In our problem, we perform correction using weight matrix $W_d(\theta) = sin(\theta)$, where the weights are proportional to this area change.

$\mathcal{L}_{grad}$ and $\mathcal{L}_{normal}$ are used to penalize the PolarMap errors in first and second order. Similar to the loss of [41], L_{grad} and L_{normal} are defined as:

$$\mathcal{L}_{grad} = \mathrm{Avg}(F(||\nabla_\theta P_{pred} - \nabla_\theta P_{gt}||_1) \tag{9}$$

$$+ F(||\nabla_\phi P_{pred} - \nabla_\phi P_{gt}||_1)) \tag{10}$$

$$\mathcal{L}_{normal} = \mathrm{Avg}(1 - \cos(n(P_{pred}), n(P_{gt}))) \tag{11}$$

Where ∇_θ and ∇_ϕ are the gradients for the PolarMap representation along θ and ϕ axis, $n(p) = [-\nabla_\theta(p), -\nabla_\phi(p), 1]$ defines the surface normal vector and $\cos(n1, n2)$ measures the Cosine similarity of the two vectors.

$\mathcal{L}_{adp}$ is the general, adaptive robust loss function of [40], used for regularization. Again including the weighted distortion correction, we can define

$$\mathcal{L}_{adp} = \mathrm{Avg}\left(\mathcal{L}_{barron}\left(W_d\|P_{pred} - P_{gt}\|_1, \alpha, c\right)\right) \tag{12}$$

TABLE I
CHAMFER DISTANCE ON LINEMOD OBJECTS

	Ape	Bvise	Cam	Can	Cat	Drill	Duck
PM:32	0.011	0.078	0.030	0.055	0.024	0.073	0.013
PM:64	0.006	0.073	0.019	0.031	0.013	0.039	0.006
voxel:32	0.013	0.075	0.032	0.056	0.026	0.067	0.017
voxel:64	0.005	0.028	0.015	0.021	0.010	0.026	0.007
	Eggb.	Glue	Holep	Iron	Lamp	Phone	Mean
PM:32	0.031	0.034	0.058	0.070	0.214	0.121	0.062
PM:64	0.027	0.009	0.038	0.035	0.119	0.080	0.038
voxel:32	0.037	0.031	0.022	0.085	0.115	0.047	0.048
voxel:64	0.015	0.012	0.011	0.034	0.076	0.019	0.021

where α and c are the parameters from $\mathcal{L}_{barron}$ [40] which will be adapted during training time.

The coordinates loss function $\mathcal{L}_{Coord}$ is identical to the above one with the difference that instead of P_{pred} and the ground truth PolarMap P_{gt}, the predicted C^p_{pred} and ground truth C^p_{gt} correspondence maps have been used. Naturally the set of weights λ_i are different.

B. Object Pose Estimation

A predicted PolarMap can be used to solve a variety of tasks. We focus on 6D pose estimation, as depicted in Fig. 3. Pose estimation is possible thanks to the fact that the predicted PolarMap implicitly encodes the object rotation. We decouple the estimation of rotation (3DoF) and translation (3DoF) estimation.

Even though the PolarMap implicitly encodes rotation, estimation of the actual rotation is not straightforward. This reason is that the PolarMap represents shape in a particular pose rather than correspondences.

Therefore, we propose to predict a rotation $R \in SO(3)$ by solving the following optimization problem

$$R^* = \underset{R \in SO(3)}{\arg\min}\, \mathcal{L}_1(P_{pred}(R), \mathcal{P}(M_0^p)) \tag{13}$$

with $P_{pred}(R)$ being the predicted PolarMap rotated by R and $\mathcal{P}(M_0^p)$ is the PolarMap in the canonical pose. The $\mathcal{L}_1$ loss, weighted with W_d similar to the training loss, is used to decrease the effect of imprecision in P_{pred}. As the optimization has no closed-form solution, we propose to optimize it using gradient descent over the rotation R.

To obtain a better initial rotation(R_{init}), we train a simple rotation regression network, which takes the same image patches as input and predicts the object rotation. Our direct rotation regression network can be formulated as: $R_{6\text{-}d} = f_\theta(I)$. We use continuous rotation representation [43] and allocentric rotation parametrization [21]. Additionally, we utilize the $\mathcal{L}_1$ and $\mathcal{L}_{adaptive}$ [40] loss to guide the network training. In spite of its simplicity, our initial rotation prediction achieves a reasonable accuracy, which is discussed in the ablation study. Then, this rotation is optimized to minimize the loss. The transpose of R^* defines the approximated rotational component of the 6D pose.

For the translation estimation we will directly use estimated correspondences C^p_{pred}, which for every point on the PolarMap give us corresponding coordinates in the input image patch. This establishes 2D-3D correspondences allowing 6D pose estimation with PnP+RANSAC.

TABLE II
RESULTS ON LINEMOD DATASET

Object	Tekin [15]	DPOD	PVNet	CDPN	GDR	Ours
Ape	21.6	53.3	43.6	64.4	-	**83.5**
Bvise	81.8	95.2	**99.9**	97.8	-	**99.9**
Cam	36.6	90.0	86.9	**91.7**	-	84.9
Can	68.8	94.1	95.5	95.9	-	**97.0**
Cat	41.8	60.4	79.3	83.8	-	**90.0**
Drill	63.5	97.4	96.4	96.2	-	**98.0**
Duck	27.2	66.0	52.6	66.8	-	**70.8**
Eggbox*	69.6	99.6	99.2	99.7	-	**99.9**
Glue*	80.0	93.8	95.7	**99.6**	-	99.5
Holep	42.6	64.9	81.9	85.8	-	**88.0**
Iron.	75.0	**99.8**	98.9	97.9	-	99.5
Lamp	71.1	88.1	**99.3**	97.9	-	98.0
Phone	47.7	71.4	**92.4**	90.8	-	92.0
Mean	56.0	82.6	86.3	89.9	**93.7**	92.4

V. EXPERIMENTS

A. Implementation Details

We implement all neural networks using Pytorch [44]. Unless specified otherwise, we use a PolarMesh of resolution $N = 32$. For the 2D detector in stage one, we use the standard YOLOv3 [39] detector, and for the PolarMap predictor in stage two, a pre-trained Resnet50 [45] is used as the encoder backbone to extract the features from input image patches. Practically, we mute the Maxpooling layer before the Resnet first block to capture the object scale and position better. The mapper uses convolution layers with 1×1 kernels instead of the linear layers for network simplification. For the experiments on the NOCS dataset [19], we used the detections produced by MaskR-CNN [32] provided with the dataset. The hyper-parameters in our loss function are $\lambda_{1-4} = (1.5, 1.0, 5.0, 0.5)$. a, c are (1e-2, 2e-2) and (1e-4, 2e-4) for PolarMap and Coordinates respectively.

B. Datasets

In this section, we demonstrate that PolarMesh is a good 3D object representation that allows for high-accuracy pose estimation. Firstly, we present qualitative results in Fig. 4 and quantitative results in Table II, Table III and Table IV on LineMod **LM** [46], LineMod-Occlusion (**LMO**) [47] and Homebrewed DB(**HB**) [48] datasets, where the proposed method achieves compelling results. Then, we demonstrate the generalization abilities of the proposed representation on the NOCS category-level pose estimation dataset [19].

We utilized the PBR synthetic train data provided by the BOP challenge [49], [50] for the experiments on **HB** and **LMO**. For each object, about 20 K synthetic images are used to train our model. The proposed method was evaluated on the BOP subset of the test sequences in order to facilitate the comparison with the related methods. Following the setup of the BOP challenge, we report the following metrics for **LMO** and **HB** datasets: Visible

TABLE III
COMPARISON WITH STATE OF THE ART ON LMO [47]

Methods	Train data	AR	VSD	MSSD	MSPD
DPOD [16]	synt.	16.9	10.1	12.6	27.8
Sundermeyer [22]	PBR	21.7	15.0	15.3	34.6
Pix2Pose [27]	PBR	36.3	23.3	30.7	55.0
DPODv2 [16]	PBR	41.1	26.9	35.1	61.3
EPOS [25]	PBR	54.7	38.9	50.1	75.0
CDPN [11]	synt.	56.9	39.3	53.7	77.9
CDPNv2 [11]	PBR	62.4	44.5	61.2	81.5
CosyPose [17](1 view)	PBR	63.3	48.0	60.6	81.2
GDR [18]	PBR	**67.2**	**50.2**	**65.2**	**86.4**
Ours	PBR	63.0	45.1	58.4	85.6

TABLE IV
COMPARISON WITH STATE OF THE ART ON HB

Methods	Train	AR	VSD	MSSD	MSPD	time(s)
Sundermeyer [22]	synt.	34.6	27.3	30.6	46.1	0.190
Pix2Pose [27]	PBR	44.6	35.2	39.4	59.4	0.982
DPODv2 [16]	PBR	45.4	37.1	41.3	57.9	0.288
CDPN [11]	synt.	47.0	39.1	45.1	56.9	0.311
EPOS [25]	PBR	58.0	48.4	52.7	72.9	0.657
CosyPose [17](1 view)	PBR	65.6	**61.3**	**63.4**	72.1	0.419
CDPNV2 [11]	PBR	**67.2**	55.7	64.2	81.6	0.311
Ours	PBR	65.7	53.6	61.4	**82.1**	0.461

Surface Discrepancy (VSD) [51], [52], Maximum Symmetry-Aware Surface Distance (MSSD) [53], Maximum Symmetry-Aware Projection Distance (MSPD) and average recall, $AR = (AR_{VSD} + AR_{MSSD} + AR_{MSPD})/3$. However, on the **LM** dataset, we train on the real images following the same split on [54]. We report the standard ADD(-S) metrics [46] with the 10% diameter threshold.

The NOCS dataset [19] aims to test how well the pose estimation methods generalize to objects not seen during training. The dataset provides synthetic and real training sets for 6 broader object classes such as bottles, cameras, etc. As we neither use depth information nor the ground truth CAD models with known real-world dimensions, we refrain from translation comparison. Instead, we evaluate the estimated rotation, as it does not depend on the scale of the object.

C. Shape Loss Analysis

We compare the shape quality of PolarMesh against cubified mesh [55] in terms of the Chamfer distance, following [55]. More precisely, to obtain the cubified mesh in resolution N, the CAD model is first generated in a $N \times N \times N$ occupancy grids with respect to the object center. Then the meshes are regenerated in these grid cells. The voxelization algorithm is the one used in [56]. We uniformly sample 5 K points both from cubified mesh and reconstructed PolarMesh as two point sets P and Q. Then, the Chamfer distance is evaluated for the two produced point clouds.

In Table I, *PM:N* defines our proposed PolarMesh of resolution N. *Voxel:N* refers to the cubified mesh [55] of resolution N. The shape represented by PolarMesh in resolution is generally worse than the Cubic Mesh in the same resolution. The star-convex nature of the PolarMap loses some minor non-convex details. However, for some objects, such as Ape, Camera, Cat, Duck and Eggbox, which are simple to be represented by a star-convex approximation, which is better in the shape quality in terms of Chamfer distance. Overall, PolarMesh has an advantage of being easy to process with 2D CNNs while still approximating the shape only slightly worse than the standard voxelization techniques.

TABLE V
ROTATION ERROR ANALYSIS

	dataset	*Init.*	PolarMap	AR	VSD	MSSD	MSPD
C1		*GT*	✓	66.8	54.0	62.4	83.9
C2	HB	*Pred.*	✓	65.7	53.6	61.4	82.1
C3		*Pred.*		57.0	45.7	52.9	72.9

D. 6D Pose Estimation on LM

Table II reports the results on the LineMod dataset. The proposed approach reaches 92.4% of pose accuracy w.r.t. the ADD score on real data, which is slightly worse (-1.3%) than SoTA method GDR [18], but clearly outperforming all other methods trained on real data (DPOD [16], PVNet [12], and CDPN [11]). Notably, from the object details, the methods works well for the objects Ape, Camera, Drill, Eggbox, Can, and Iron. These objects are mostly star convex and thus well-suited for the proposed PolarMesh representation.

E. 6D Pose Estimation on LMO

We evaluate our approach on the LineMod-Occlusion dataset. LMO provides a more challenging test set with more occlusions than the LineMod dataset. We train our methods on synthetic PBR images from the BOP challenge [49]. The results on the dataset in Table III show that we achieve comparable performance on Average Recall metric, especially on AR_{MSPD} metric. MSPD measures the symmetry-aware reprojection error, which emphasises rotational accuracy. We believe that this indicates that the pose error comes from the imprecision in translation estimation rather than from imprecise rotations.

F. 6D Pose Estimation on HB

We also evaluate our approach on the HB dataset. Table IV indicates that the proposed method performs better than the others and almost on-par with CDPN [11] in terms of the MSPD metric, while it is outperformed by CDPN and CosyPose in terms of VSD and MSSD. Again, we do not use any pose refinement to improve the predicted poses. We can conclude that our proposed PolarMesh method ensured precise rotation estimation.

G. Rotation Estimation for Novel Object

Pose estimation from RGB for novel objects that were not seen during training is a fundamentally difficult problem, which was also highlighted in the related letters [20]. The projective nature of imaging makes it impossible to distinguish between

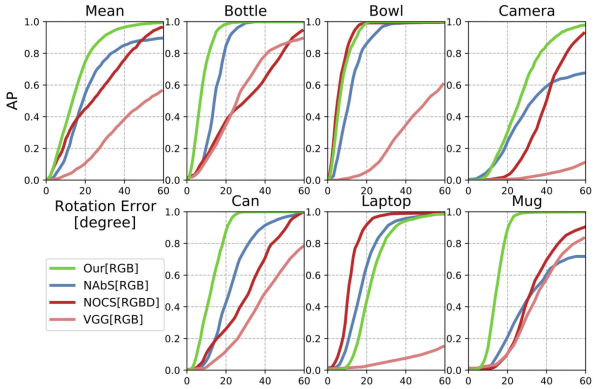

Fig. 5. Category-level average precision at different rotation error thresholds on the test set of the NOCS REAL275 dataset.

translation and scale changes. Therefore, we focus on evaluation of the quality of the predicted rotation and shape. Fig. 5 provides a comparison of the rotation precision for various angular error thresholds on the real test set of the NOCS dataset. The plot confirms that the proposed method clearly outperforms NOCS [19] and NAbS [20] on the majority of the objects. It even performs significantly better than the original NOCS approach, even though we are neither using correspondences nor depth. PM-Net performs surprisingly good even on object categories which are not star-convex, such as Mug and Bowl.

H. Ablation Study

Here, we analyze the errors produced by our PM-Net and those caused by rotation initialization on HB dataset. In the Table V, we compare direct rotation regression(C3), our method(C2) and our method with GT initial rotation(C1). Our method (C2) achieves the Average Recall improvement (8.7%) compared to the direct rotation regression (C3), indicating the benefits of our proposed PolarMesh representation. The method using ground truth rotation as initial value (C1) shows the upper bound of our method. Our proposed method has only a slight drop compared with the upper bound, which is caused by the imprecision of the directly regressed initial rotation.

We tried to analyze the root of imprecision in pose prediction on LineMod and HB datasets by visual examination of reconstructions and pose errors for the images with the largest ADD errors. Several representative examples of such images are shown in Fig. 6. We conclude that the error mainly comes from the rotation estimation algorithms. In the case of the yellow rabbit, we observe the imprecise Polarmesh prediction caused by occlusions. Noticeably, even though the PolarMesh prediction is not perfect, the estimated pose is still acceptable due to the similar overall shape.

I. Runtime Analysis

During inference, on a machine with Intel(R) Xeon(R) 2 CPU and a Tesla V100 GPU with 16 GB memory, our approach takes about 0.461 s for one object, including 0.026 s for Yolov3 2D detector, 0.039 s for PM-Net, 0.388 s for rotation optimizing, and 0.008 s for translation estimation. We train YOLO to detect

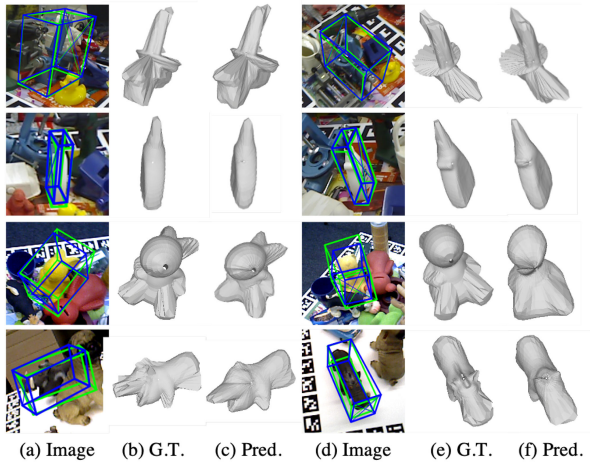

Fig. 6. Bad cases in LineMod and HB datasets. Colomns (a) and (d) are the pose visualization (blue are our predicted bounding boxes and green are ground truth bounding boxes. Columns (b) and (e) are the generated ground truth Polarmeshes. And, columns (c) and (f) are our predicted Polarmeshes.).

all object of a particular scene and one PM-net per object. Therefore, YOLO is run once per image, while PM-Net with the subsequent optimization is run separately for each detected object in the image. This results in linear scaling of run time depending on the number of objects.

VI. Conclusion

In this letter, we propose PolarMesh, a 3D shape approximation for object pose estimation. PolarMesh, which encodes shape and orientation, is directly regressed by an auto-encoder neural network, PM-Net, from a single RGB image. A thorough evaluation on LineMod, Occlusion, Homebrewed and NOCS datasets shows our compelling performance on the task of object pose estimation while being capable of generalizing even to unseen objects.

We firmly believe that the properties of the PolarMesh can inspire future works not only for rotation estimation but also for related tasks that benefit from shape and orientation encoding accessible for 2D CNNs.

Acknowledgment

The authors would thank the anonymous reviewers for their valuable suggestions.

References

[1] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, vol. 56, 2009.
[2] M. Esposito, B. Busam, C. Hennersperger, J. Rackerseder, N. Navab, and B. Frisch, "Multimodal US-gamma imaging using collaborative robotics for cancer staging biopsies," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 11, no. 9, pp. 1561–1571, 2016.
[3] M. L. Liu and K. H. Wong, "Pose estimation using four corresponding points," *Pattern Recognit. Lett.*, vol. 20, no. 1, pp. 69–74, 1999.
[4] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. 2nd IEEE ACM Int. Workshop Augmented Reality*, 1999, pp. 85–94.
[5] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate o (n) solution to the PNP problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, 2009.

[6] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[7] N. Müller, Y.-S. Wong, N. J. Mitra, A. Dai, and M. Nießner, "Seeing behind objects for 3D multi-object tracking in RGB-D sequences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6067–6076.

[8] E. Ataer-Cansizoglu and Y. Taguchi, "Object detection and tracking in RGB-D SLAM via hierarchical feature grouping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4164–4171.

[9] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *Robot.: Sci. Syst. (RSS)*, 2018.

[10] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1530–1538.

[11] Z. Li, G. Wang, and X. Ji, "CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7677–7686.

[12] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DoF pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4556–4565.

[13] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3848–3856.

[14] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from RGB images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 699–715.

[15] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 292–301.

[16] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: 6D pose object detector and refiner," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1941–1950.

[17] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6D pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 574–591.

[18] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "GDR-Net: Geometry-guided direct regression network for monocular 6D object pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 16 606–16 616.

[19] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2637–2646.

[20] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges, "Category level object pose estimation via neural analysis-by-synthesis," in *Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 139–156.

[21] F. Manhardt *et al.*, "CPS : Improving class-level 6D pose and shape estimation from monocular images with self-supervised learning," 2020, *arXiv:2003.05848v3*.

[22] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented autoencoders: Implicit 3D orientation learning for 6D object detection," *Int. J. Comput. Vis.*, vol. 128, no. 3, pp. 714–729, 2020.

[23] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[24] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon, "Autolabeling 3D objects with differentiable rendering of SDF shape priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12 221–12 230.

[25] T. Hodan, D. Barath, and J. Matas, "EPOS: Estimating 6D pose of objects with symmetries," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11 700–11 709.

[26] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6D pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7667–7676.

[27] I. Shugurov, I. Pavlov, S. Zakharov, and S. Ilic, "Multi-view object pose refinement with differentiable renderer," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2579–2586, Apr. 2021.

[28] I. Shugurov, S. Zakharov, and S. Ilic, "DPODv2: Dense correspondence-based 6 DoF pose estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021, doi: 10.1109/TPAMI.2021.3118833.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[30] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, 2015, pp. 234–241.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[33] G. Wang, F. Manhardt, J. Shao, X. Ji, N. Navab, and F. Tombari, "Self6d: Self-supervised monocular 6D object pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 108–125.

[34] F. Manhardt *et al.*, "Explaining the ambiguity of object detection and 6D pose from visual data," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6840–6849.

[35] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning so (3) equivariant representations with spherical CNNs," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 52–68.

[36] Z. Cao, Q. Huang, and R. Karthik, "3D object classification via spherical projections," in *Proc. Int. Conf. 3D Vis.*, 2017, pp. 566–574.

[37] X. Zhang, Z. Zhang, C. Zhang, J. B. Tenenbaum, W. T. Freeman, and J. Wu, "Learning to reconstruct shapes from unseen classes," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[38] N. Ravi *et al.*, "Accelerating 3D deep learning with pytorch3d," in *SIGGRAPH Asia 2020 Courses*, 2020, pp. 1–1.

[39] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, *arXiv:1804.02767*.

[40] J. T. Barron, "A general and adaptive robust loss function," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4326–4334.

[41] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," in *Proc. Winter Conf. Appl. Comput. Vis.*, 2019, pp. 1043–1051.

[42] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs," in *Int. Conf. Learn. Representations*, 2018.

[43] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5738–5746.

[44] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Neural Inf. Process. Syst.*, vol. 32, pp. 8026–8037, 2019.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[46] S. Hinterstoisser *et al.*, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Proc. Asian Conf. Comput. Vis.*, 2012, pp. 548–562.

[47] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 536–551.

[48] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "Homebreweddb: RGB-D dataset for 6D pose estimation of 3D objects," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, 2019, pp. 2767–2776.

[49] T. Hodan and A. Melenovsky, "Bop: Benchmark for 6D object pose estimation," 2019. [Online]. Available: https://bop.felk.cvut.cz/home/

[50] M. Denninger *et al.*, "Blenderproc: Reducing the reality gap with photo-realistic rendering," in *Proc. Robot., Sci. Syst. Workshops*, 2020.

[51] T. Hodan *et al.*, "Bop: Benchmark for 6D object pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 19–34.

[52] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6D object pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 606–619.

[53] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, "Introducing MVTec ITODD-a dataset for 3D object recognition in industry," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2017, pp. 2200–2208.

[54] E. Brachmann *et al.*, "Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3364–3372.

[55] G. Gkioxari, J. Malik, and J. Johnson, "Mesh R-CNN," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9784–9794.

[56] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 523–540.