BOP Challenge 2020 on 6D Object Localization

Tomáš Hodaň[1], Martin Sundermeyer[2], Bertram Drost[3], Yann Labbé[4],
Eric Brachmann[5], Frank Michel[6], Carsten Rother[5], Jiří Matas[1]

[1]Czech Technical University in Prague, [2]German Aerospace Center, [3]MVTec,
[4]INRIA Paris, [5]Heidelberg University, [6]Technical University Dresden

Abstract. This paper presents the evaluation methodology, datasets, and results of the BOP Challenge 2020, the third in a series of public competitions organized with the goal to capture the status quo in the field of 6D object pose estimation from an RGB-D image. In 2020, to reduce the domain gap between synthetic training and real test RGB images, the participants were provided 350K photorealistic trainining images generated by BlenderProc4BOP, a new open-source and light-weight physically-based renderer (PBR) and procedural data generator. Methods based on deep neural networks have finally caught up with methods based on point pair features, which were dominating previous editions of the challenge. Although the top-performing methods rely on RGB-D image channels, strong results were achieved when only RGB channels were used at both training and test time – out of 26 evaluated methods, the third method was trained on RGB channels of PBR and real images, while the fifth was trained on PBR images only. Strong data augmentation was identified as a key component of the top-performing CosyPose method, and the photorealism of PBR images was demonstrated effective despite the augmentation. The online evaluation system stays open and is available at the project website: `bop.felk.cvut.cz`.

1 Introduction

Estimating the 6D pose, *i.e.* the 3D translation and 3D rotation, of rigid objects from a single input image is a crucial task for numerous application fields such as robotic manipulation, augmented reality, or autonomous driving. The BOP[1] Challenge 2020 is the third in a series of public challenges that are part of the BOP project aiming to continuously report the state of the art in 6D object pose estimation. The first challenge was organized in 2017 [26] and the results were published in [25]. The second challenge from 2019 [22] and the third from 2020 share the same evaluation methodology and leaderboard and the results from both are included in this paper.

Participating methods are evaluated on the 6D object localization task [24], where the methods report their predictions on the basis of two sources of information. Firstly, at training time, a method is given 3D object models and training images showing the objects in known 6D poses. Secondly, at test time,

[1] BOP stands for Benchmark for 6D Object Pose Estimation [25].

the method is provided with a test image and a list of object instances visible in the image, and the goal of the method is to estimate 6D poses of the listed object instances. The training and test images consist of RGB-D (aligned color and depth) channels and the intrinsic camera parameters are known.

The challenge primarily focuses on the practical scenario where no real images are available at training time, only the 3D object models and images synthesized using the models. While capturing real images of objects under various conditions and annotating the images with 6D object poses requires a significant human effort [23], the 3D models are either available before the physical objects, which is often the case for manufactured objects, or can be reconstructed at an admissible cost. Approaches for reconstructing 3D models of opaque, matte and moderately specular objects are well established [39] and promising approaches for transparent and highly specular objects are emerging [42,52,14].

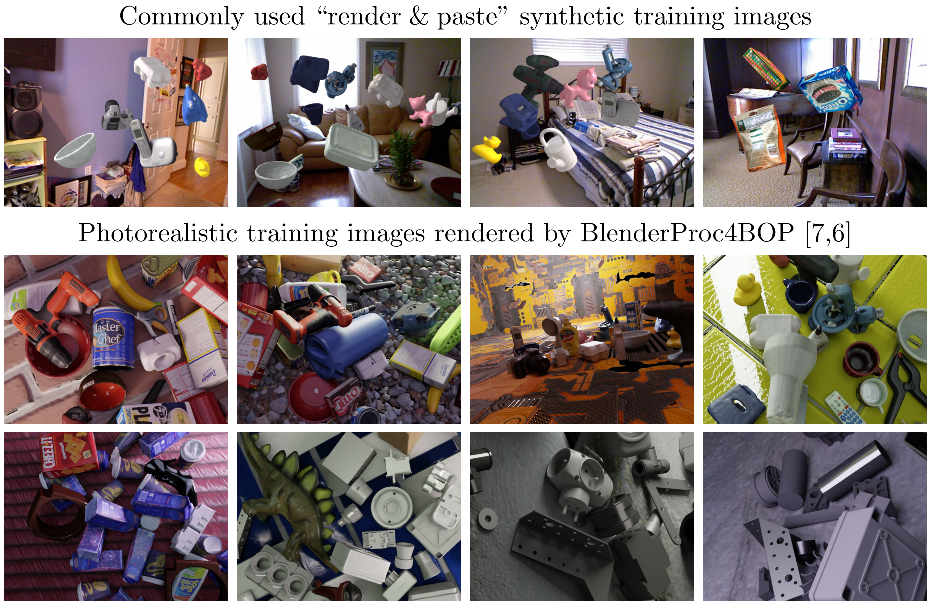
In the BOP Challenge 2019, methods using the depth image channel, which were mostly based on the point pair features (PPF) [10], clearly outperformed methods relying only on the RGB channels, all of which were based on deep neural networks (DNN). The PPF-based methods match pairs of oriented 3D points between the point cloud[2] of the test scene and the 3D object model, and aggregate the matches via a voting scheme. As each pair is described by only the distance and relative orientation of the two points, PPF-based methods can be effectively trained directly on the 3D object models, without the need to synthesize any training images. In contrast, DNN-based methods require large amounts of annotated training images, which have been typically obtained by OpenGL rendering of the 3D object models on top of random backgrounds [31,43,19,11]. However, as suggested in [29], the evident domain gap between these "render & paste" training images and real test images presumably limits the potential of the DNN-based methods.

To reduce the gap between the synthetic and real domains and thus to bring fresh air to the DNN-based methods, we have created BlenderProc4BOP [7,6], an open-source and light-weight physically-based renderer (PBR). Furthermore, to reduce the entry barrier of the challenge and to standardize the training set, the participants were provided with 350K pre-rendered PBR images (Fig. 1).

In 2020, the DNN-based methods have finally caught up with the PPF-based methods – five methods outperformed Vidal-Sensors18 [51], the PPF-based winner from 2017 and 2019. Three of the top five methods, including the top-performing one, are single-view variants of CosyPose, a DNN-based method by Labbé *et al.* [33]. A strong data augmentation, similar to [49], was identified as one of the key ingredients of this method. The second is a hybrid DNN+PPF method by König and Drost [32], and the fourth is Pix2Pose, a DNN-based method by Park *et al.* [40]. The first two methods used RGB-D image channels, while the third method achieved strong results with RGB channels only.

Methods achieved noticeably higher accuracy scores when trained on the PBR training images than when trained on "render & paste" images. Although adding real training images yielded even higher scores, competitive results were

[2] The point cloud is calculated from the depth channel and known camera parameters.


Commonly used “render & paste” synthetic training images

Photorealistic training images rendered by BlenderProc4BOP [7,6]

Figure 1. **Synthetic training images.** DNN-based methods for 6D object pose estimation have been commonly trained on “render & paste” images synthesized by OpenGL rendering of 3D object models randomly positioned on top of random backgrounds. Instead, participants of the BOP Challenge 2020 were provided 350K photorealistic training images synthesized by ray tracing and showing the 3D object models in physically plausible poses inside a cube with a random PBR material (see Sec. 3.2).

achieved with PBR images only – out of 26 evaluated methods, the fifth was trained only on the PBR images. Interestingly, the increased photorealism from the PBR images led to clear improvements of also the CosyPose method, despite the strong data augmentation which this method applies to the training images.

The rest of the paper is organized as follows. Section 2 defines the evaluation methodology, Section 3 introduces datasets and the implemented approach to synthesize photorealistic training images, Section 4 describes the experimental setup and analyzes the results, Section 5 presents the awards of the BOP Challenge 2020, and Section 6 concludes the paper. A discussion on the choices made when defining the evaluation methodology is provided in the supplement.

2 Evaluation Methodology

The evaluation methodology detailed in this section defines the challenge task, functions to measure the error of a 6D pose estimate, and calculation of the accuracy score used to compare the evaluated methods. The BOP Challenge 2020 follows the same evaluation methodology as the BOP Challenge 2019 – the scores have not been saturated and following the same methodology allowed using results from 2019 as baselines in 2020.

2.1 Task Definition

Methods are evaluated on the task of 6D localization of a **v**arying number of **i**nstances of a **v**arying number of **o**bjects in a single RGB-D image. This variant of the 6D object localization task is referred to as ViVo and defined as:

Training input: For each object with index $o \in O = \{1, \dots, k\}$, a method is given training data T_o including a 3D mesh model of the object, often with texture, and a set of synthetic or real RGB-D images showing instances of the object in known 6D poses. The method may use any of the image channels.

Test input: The method is provided with image I and list $L = [(o_1, n_1), \dots, (o_m, n_m)]$, where n_i is the number of instances of object o_i present in image I.

Test output: The method produces list $E = [E_1, \dots, E_m]$, where E_i is a list of n_i pose estimates for instances of object o_i. Each estimate is given by a 3×3 rotation matrix $\mathbf{R}$, a 3×1 translation vector $\mathbf{t}$, and a confidence score s. Matrix $\mathbf{P} = [\mathbf{R}|\mathbf{t}]$ defines a rigid transformation from the 3D coordinate system of the object model to the 3D coordinate system of the camera.

Note that in the first challenge in 2017 [26,25], methods were evaluated on a simpler variant of the 6D object localization task – the goal was to estimate the 6D pose of a **s**ingle **i**nstance of a **s**ingle **o**bject (this variant is referred to as SiSo). If multiple instances of the same object model were visible in the image, then the pose of an arbitrary instance may have been reported. In 2017, the simpler SiSo variant was chosen because it allowed to evaluate all relevant methods out of the box. Since then, the state of the art has advanced and we have moved to the more challenging ViVo variant.

2.2 Pose-error functions

The error of an estimated pose $\hat{\mathbf{P}}$ *w.r.t.* the ground-truth pose $\bar{\mathbf{P}}$ of an object model O is measured by three pose-error functions. The functions are defined below and discussed in more detail in the supplement.

VSD (Visible Surface Discrepancy):

$$e_{\mathrm{VSD}}(\hat{S}, \bar{S}, S_I, \hat{V}, \bar{V}, \tau) = \mathrm{avg}_{p \in \hat{V} \cup \bar{V}} \begin{cases} 0 & \text{if}\, p \in \hat{V} \cap \bar{V} \,\wedge\, |\hat{S}(p) - \bar{S}(p)| < \tau \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

Symbols $\hat{S}$ and $\bar{S}$ denote distance maps[3] obtained by rendering the object model O in the estimated pose $\hat{\mathbf{P}}$ and the ground-truth pose $\bar{\mathbf{P}}$ respectively. These distance maps are compared with the distance map S_I of the test image I to obtain the visibility masks $\hat{V}$ and $\bar{V}$, *i.e.* sets of pixels where model O is visible in image I. Parameter τ is a misalignment tolerance.

[3] A distance map stores at a pixel p the distance from the camera center to a 3D point $\mathbf{x}_p$ that projects to p. It can be readily computed from the depth map which stores at p the Z coordinate of $\mathbf{x}_p$ and which is a typical output of Kinect-like sensors.

Compared to [25,24], estimation of the visibility masks has been modified – an object is now considered visible at pixels with no depth measurements. This modification allows evaluating poses of glossy objects from the ITODD dataset [9] whose surface is not always captured in the depth image channel.

VSD treats poses that are indistinguishable in shape (color is not considered) as equivalent by measuring the misalignment of only the visible part of the object surface. See Sec. 2.2 of [25] and the supplement of this paper for details.

MSSD (Maximum Symmetry-Aware Surface Distance):

$$e_{\mathrm{MSSD}} = \min_{\mathbf{S} \in S_O} \max_{\mathbf{x} \in V_O} \|\hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{S}\mathbf{x}\|_2 \tag{2}$$

Set S_O contains global symmetry transformations of object model O, enumerated as described in Sec. 2.3, and V_O is a set of mesh vertices.

The maximum distance between mesh vertices is relevant for robotic manipulation, where the maximum surface deviation strongly indicates the chance of a successful grasp. Moreover, compared to the average distance used in pose-error functions ADD and ADI [24,18], the maximum distance is less dependent on the geometry of the object model and the sampling density of its surface.

MSPD (Maximum Symmetry-Aware Projection Distance):

$$e_{\mathrm{MSPD}} = \min_{\mathbf{S} \in S_O} \max_{\mathbf{x} \in V_O} \|\mathrm{proj}(\hat{\mathbf{P}}\mathbf{x}) - \mathrm{proj}(\bar{\mathbf{P}}\mathbf{S}\mathbf{x})\|_2 \tag{3}$$

Function proj(.) is the 2D projection (the result is in pixels) and the meaning of the other symbols is as in MSSD.

Compared to the pose-error function from [3], MSPD considers global object symmetries and replaces the average by the maximum distance to increase robustness against the geometry and sampling of the object model. Since MSPD does not evaluate the alignment along the optical (Z) axis and measures only the perceivable discrepancy, it is relevant for augmented reality applications and suitable for evaluating RGB-only methods, for which estimating the alignment along the optical axis is more challenging.

2.3 Identifying Global Object Symmetries

The set of global symmetry transformations of an object model O, which is used in calculation of MSSD and MSPD, is identified in two steps. Firstly, a set of candidate symmetry transformations is defined as $S'_O = \{\mathbf{S} : h(V_O, \mathbf{S}V_O) < \varepsilon\}$, where h is the Hausdorff distance calculated between vertices V_O of object model O in the canonical and transformed locations. The allowed deviation is bounded by $\varepsilon = \max(15\,mm, 0.1d)$, where d is the diameter of model O (the largest distance between any pair of vertices) and the truncation at $15\,mm$ avoids breaking the symmetries by too small details. Secondly, the final set of symmetry transformations S_O is defined as a subset of S'_O which consists of those symmetry transformations that cannot be resolved by the model texture (decided subjectively by the organizers of the challenge).

Set S_O covers both discrete and continuous global rotational symmetries. The continuous rotational symmetries are discretized such as the vertex which is the furthest from the axis of symmetry travels not more than 1% of the object diameter between two consecutive rotations.

2.4 Accuracy Score

An estimated pose is considered correct *w.r.t.* pose-error function e, if $e < \theta_e$, where $e \in \{e_{\mathrm{VSD}}, e_{\mathrm{MSSD}}, e_{\mathrm{MSPD}}\}$ and θ_e is the threshold of correctness.

The fraction of annotated object instances, for which a correct pose is estimated, is referred to as Recall. The Average Recall *w.r.t.* function e, denoted as AR_e, is then defined as the average of Recall rates calculated for multiple settings of threshold θ_e, and also for multiple settings of the misalignment tolerance τ in the case of e_{VSD}. In particular, $\mathrm{AR}_{\mathrm{VSD}}$ is the average of Recall rates calculated for the misalignment tolerance τ ranging from 5% to 50% of the object diameter with a step of 5%, and the threshold of correctness θ_{VSD} ranging from 0.05 to 0.5 with a step of 0.05. $\mathrm{AR}_{\mathrm{MSSD}}$ is the average of Recall rates calculated for θ_{MSSD} ranging from 5% to 50% of the object diameter with a step of 5%. Finally, $\mathrm{AR}_{\mathrm{MSPD}}$ is the average of Recall rates calculated for θ_{MSPD} ranging from $5r$ to $50r$ with a step of $5r$, where $r = w/640$ and w is the image width in pixels.

The accuracy of a method on a dataset D is measured by $\mathrm{AR}_D = (\mathrm{AR}_{\mathrm{VSD}} + \mathrm{AR}_{\mathrm{MSSD}} + \mathrm{AR}_{\mathrm{MSPD}})/3$. The overall accuracy on the core datasets is then measured by $\mathrm{AR}_{\mathrm{Core}}$ defined as the average of the per-dataset AR_D scores. In this way, each dataset is treated as a separate sub-challenge which avoids $\mathrm{AR}_{\mathrm{Core}}$ being dominated by larger datasets.

3 Datasets

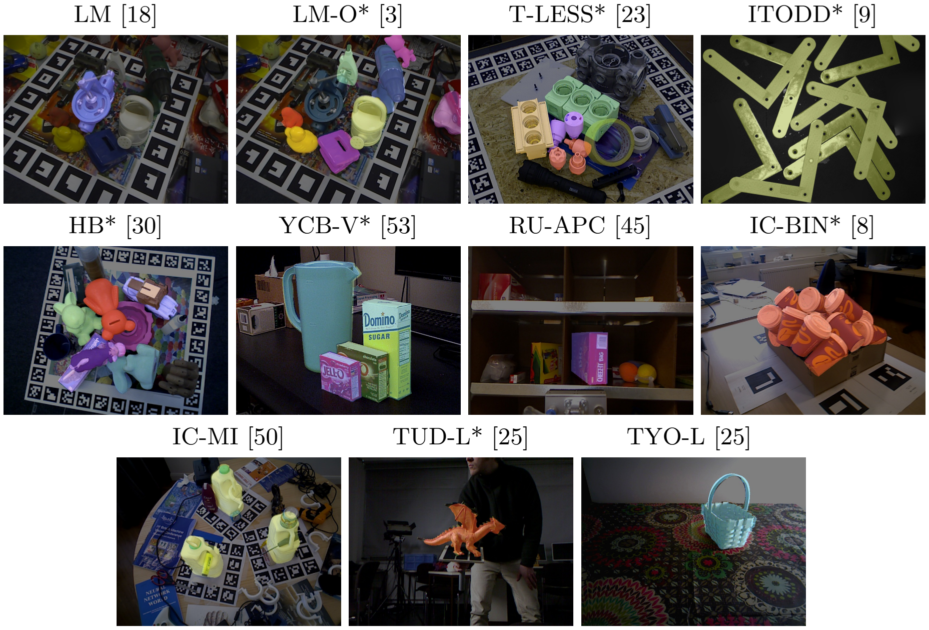
BOP currently includes 11 datasets in a unified format, detailed in Tab. 1, seven of which were selected as core datasets. A method had to be evaluated on all core datasets to be considered for the main challenge awards (Sec. 5).

3.1 Content of Datasets

Each dataset is provided in a unified format and includes 3D object models and training and test RGB-D images annotated with ground-truth 6D object poses. The HB and ITODD datasets include also validation images – in this case, the ground-truth poses are publicly available only for the validation images, not for the test images. The object models were created manually or using KinectFusion-like systems for 3D surface reconstruction [39]. The seven core datasets include photorealistic training images described in Sec. 3.2. Datasets T-LESS, TUD-L, and YCB-V include real training images, and most datasets include also training images obtained by OpenGL rendering of the 3D object models on a black background. The test images were captured in scenes with graded complexity, often with clutter and occlusion. The datasets can be downloaded from: `bop.felk.cvut.cz/datasets`.

			Train. im.		Val im.	Test im.		Test inst.	
Dataset	Core	Objects	Real	PBR	Real	All	Used	All	Used
LM [18]		15	–	50000	–	18273	3000	18273	3000
LM-O [3]	∗	8	–	50000	–	1214	200	9038	1445
T-LESS [23]	∗	30	37584	50000	–	10080	1000	67308	6423
ITODD [9]	∗	28	–	50000	54	721	721	3041	3041
HB [30]	∗	33	–	50000	4420	13000	300	67542	1630
YCB-V [53]	∗	21	113198	50000	–	20738	900	98547	4123
RU-APC [45]		14	–	–	–	5964	1380	5964	1380
IC-BIN [8]	∗	2	–	50000	–	177	150	2176	1786
IC-MI [50]		6	–	–	–	2067	300	5318	800
TUD-L [25]	∗	3	38288	50000	–	23914	600	23914	600
TYO-L [25]		21	–	–	–	1670	1670	1670	1670

Table 1. **Parameters of the BOP datasets.** Most datasets include also training images obtained by OpenGL rendering of the 3D object models on a black background (not shown in the table). Extra PBR training images can be rendered by BlenderProc4BOP [7,6]. If a dataset includes both validation and test images, the ground-truth annotations are public only for the validation images. All test images are real. Column "Test inst." shows the number of annotated object instances for which at least 10% of the projected surface area is visible in test images. Columns "Used" show the number of test images and object instances used in the BOP Challenge 2019 and 2020.


LM [18]

LM-O* [3]

T-LESS* [23]

ITODD* [9]

HB* [30]

YCB-V* [53]

RU-APC [45]

IC-BIN* [8]

IC-MI [50]

TUD-L* [25]

TYO-L [25]

Figure 2. **An overview of the BOP datasets.** The core datasets are marked with a star. Shown are RGB channels of sample test images which were darkened and overlaid with colored 3D object models in the ground-truth 6D poses.

3.2 Photorealistic Training Images

In the BOP Challenge 2020, the participants were provided with 50K photorealistic training images for each of the seven core datasets. The images were generated and automatically annotated by BlenderProc4BOP [7,6], an open-source and light-weight physically-based renderer of procedurally generated scenes.

Physically-based rendering (PBR) accurately simulates the flow of light energy in the scene by ray tracing. This naturally accounts for complex illumination effects such as scattering, refraction and reflection, including diffuse and specular interreflection between the scene elements [41]. The rendered images are very realistic and often difficult to differentiate from real photographs. Rendering techniques based on rasterization, *e.g.* OpenGL, can approximate the complex effects in an ad hoc way through custom shaders, but the approximations cause physically incorrect artifacts that are difficult to eliminate [38].

BlenderProc4BOP implements a PBR synthesis approach similar to [29]. However, to improve efficiency, objects are not rendered in 3D models of complete indoor scenes but inside an empty cube, with objects from other BOP datasets serving as distractors. To achieve a rich spectrum of generated images, a random PBR material from the CC0 Textures library [5] is assigned to the walls of the cube, and light with a random intensity and color is emitted from the room ceiling and from a randomly positioned point source. The number of rays traced per image pixel is set to 50 and the Intel Open Image Denoiser [1] is applied to reduce noise in the rendered image. This setup keeps the computational cost low – the full generation of one 640×480 RGB-D image takes **1–3 seconds** on a standard desktop computer with a modern GPU, and a set of 50K images can be therefore rendered on 5 GPU's overnight.

Instead of trying to perfectly model the object materials, properties such as specularity, roughness and metallicness are randomized. Such physically plausible domain randomization is important since objects in the challenge as well as in real-world scenarios are typically not modeled perfectly. Realistic object poses are achieved by dropping the 3D object models on the ground plane of the cube using the PyBullet physics engine integrated in Blender [4]. This allows to create dense but shallow piles of objects that introduce various levels of occlusion. Since test images from the LM dataset show the objects always standing upright, the objects from LM are not dropped but instead densely placed on the ground plane in upright poses using automated collision checks.

Each object arrangement is rendered from 25 random camera poses. Instead of fitting all objects within the camera frustum, each camera is pointed at a randomly selected object close to the center, which allows generating more diverse camera poses. Azimuth, elevation, and distances of the cameras are uniformly sampled from ranges determined by the ground-truth 6D object poses of the test images. In-plane rotations of the camera are generated randomly.

The generated data (object poses, camera intrinsics, RGB and depth) is saved in the BOP format, allowing to interface with utilities from the BOP toolkit [27]. Configuration files to reproduce or modify the generation process are provided[4].

[4] `github.com/DLR-RM/BlenderProc/blob/master/README_BlenderProc4BOP.md`

4 Evaluation

This section describes the experimental setup, presents the results of the BOP Challenge 2020, and analyzes the effectiveness of PBR training images.

4.1 Experimental Setup

Participants of the challenge were submitting the results of their methods to the online evaluation system at `bop.felk.cvut.cz` from June 5th, 2020, until the deadline on August 19th, 2020. The methods were evaluated on the ViVo variant of the 6D object localization task as described in Sec. 2. The evaluation script is publicly available in the BOP toolkit [27].

A method had to use a fixed set of hyper-parameters across all objects and datasets. For training, a method may have used the provided object models and training images, and rendered extra training images using the object models. However, not a single pixel of test images may have been used for training, nor the individual ground-truth poses or object masks provided for the test images. Ranges of the azimuth and elevation camera angles, and a range of the camera-object distances calculated from the ground-truth poses in test images, is the only information about the test set that may have been used for training.

Only subsets of test images were used to remove redundancies and speed up the evaluation, and only object instances for which at least 10% of the projected surface area is visible were to be localized in the test images.

4.2 Results

In total, 26 methods were evaluated on all seven core datasets. Results of 11 methods were submitted to the BOP Challenge 2019 and results of 15 methods to the BOP Challenge 2020 (column "Year" in Tab. 2).

In 2020, methods based on deep neural networks (DNN) have finally caught up with methods based on point pair features (PPF) [10] – five methods from 2020 outperformed Vidal-Sensors18 [51], the PPF-based winner of the first two challenges from 2017 and 2019 (columns "PPF" and "DNN" in Tab. 2). Almost all participating DNN-based methods applied neural networks only to the RGB image channels and many of these methods used the depth channel for ICP refinement at test time (columns "Train", "Test", and "Refine"). Only PointVoteNet2 [15] applied a neural network also to the depth channel. It is noteworthy that the overall third method does not use the depth channel at all.

Three of the top five methods, including the top-performing one, are single-view variants of the CosyPose method by Labbé *et al.* [33]. This method first predicts 2D bounding boxes of the objects using Mask R-CNN [16], and then applies to each box a DNN model for coarse pose estimation followed by a DNN model for iterative refinement. The top variant of CosyPose, with the AR_{Core} score of 69.8%, additionally applies a depth-based ICP refinement which improves the score by 6.1% (method #1 *vs.* #3 in Tab. 2). One of the key ingredients of CosyPose is a strong data augmentation similar to [48]. As reported

#	Method	Avg.	LM-O	T-LESS	TUD-L	IC-BIN	ITODD	HB	YCB-V	Time
1	CosyPose-ECCV20-Synt+Real-ICP [33]	69.8	71.4	70.1	93.9	64.7	31.3	71.2	86.1	13.74
2	König-Hybrid-DL-PointPairs [32]	63.9	63.1	65.5	92.0	43.0	48.3	65.1	70.1	0.63
3	CosyPose-ECCV20-Synt+Real [33]	63.7	63.3	72.8	82.3	58.3	21.6	65.6	82.1	0.45
4	Pix2Pose-BOP20_w/ICP-ICCV19 [40]	59.1	58.8	51.2	82.0	39.0	35.1	69.5	78.0	4.84
5	CosyPose-ECCV20-PBR [33]	57.0	63.3	64.0	68.5	58.3	21.6	65.6	57.4	0.47
6	Vidal-Sensors18 [51]	56.9	58.2	53.8	87.6	39.3	43.5	70.6	45.0	3.22
7	CDPNv2_BOP20-RGB-ICP [35]	56.8	63.0	46.4	91.3	45.0	18.6	71.2	61.9	1.46
8	Drost-CVPR10-Edges [10]	55.0	51.5	50.0	85.1	36.8	57.0	67.1	37.5	87.57
9	CDPNv2_BOP20-PBR-ICP [35]	53.4	63.0	43.5	79.1	45.0	18.6	71.2	53.2	1.49
10	CDPNv2_BOP20-RGB [35]	52.9	62.4	47.8	77.2	47.3	10.2	72.2	53.2	0.94
11	Drost-CVPR10-3D-Edges [10]	50.0	46.9	40.4	85.2	37.3	46.2	62.3	31.6	80.06
12	Drost-CVPR10-3D-Only [10]	48.7	52.7	44.4	77.5	38.8	31.6	61.5	34.4	7.70
13	CDPN_BOP19-RGB [35]	47.9	56.9	49.0	76.9	32.7	6.7	67.2	45.7	0.48
14	CDPNv2_BOP20-PBR [35]	47.2	62.4	40.7	58.8	47.3	10.2	72.2	39.0	0.98
15	leaping from 2D to 6D [37]	47.1	52.5	40.3	75.1	34.2	7.7	65.8	54.3	0.42
16	EPOS-BOP20-PBR [21]	45.7	54.7	46.7	55.8	36.3	18.6	58.0	49.9	1.87
17	Drost-CVPR10-3D-Only-Faster [10]	45.4	49.2	40.5	69.6	37.7	27.4	60.3	33.0	1.38
18	Félix&Neves-ICRA17-IET19 [46,44]	41.2	39.4	21.2	85.1	32.3	6.9	52.9	51.0	55.78
19	Sundermeyer-IJCV19+ICP [49]	39.8	23.7	48.7	61.4	28.1	15.8	50.6	50.5	0.86
20	Zhigang-CDPN-ICCV19 [35]	35.3	37.4	12.4	75.7	25.7	7.0	47.0	42.2	0.51
21	PointVoteNet2 [15]	35.1	65.3	0.4	67.3	26.4	0.1	55.6	30.8	-
22	Pix2Pose-BOP20-ICCV19 [40]	34.2	36.3	34.4	42.0	22.6	13.4	44.6	45.7	1.22
23	Sundermeyer-IJCV19 [49]	27.0	14.6	30.4	40.1	21.7	10.1	34.6	37.7	0.19
24	SingleMultiPathEncoder-CVPR20 [47]	24.1	21.7	31.0	33.4	17.5	6.7	29.3	28.9	0.19
25	Pix2Pose-BOP19-ICCV19 [40]	20.5	7.7	27.5	34.9	21.5	3.2	20.0	29.0	0.79
26	DPOD (synthetic) [54]	16.1	16.9	8.1	24.2	13.0	0.0	28.6	22.2	0.23

#	Method	Year	PPF	DNN	Train	...type	Test	Refine
1	CosyPose-ECCV20-Synt+Real-ICP [33]	2020	-	3/set	rgb	pbr+real	rgb-d	rgb+icp
2	König-Hybrid-DL-PointPairs [32]	2020	yes	1/set	rgb	syn+real	rgb-d	icp
3	CosyPose-ECCV20-Synt+Real [33]	2020	-	3/set	rgb	pbr+real	rgb	rgb
4	Pix2Pose-BOP20_w/ICP-ICCV19 [40]	2020	-	1/obj	rgb	pbr+real	rgb-d	icp
5	CosyPose-ECCV20-PBR [33]	2020	-	3/set	rgb	pbr	rgb	rgb
6	Vidal-Sensors18 [51]	2019	yes	-	-	-	d	icp
7	CDPNv2_BOP20-RGB-ICP [35]	2020	-	1/obj	rgb	pbr+real	rgb-d	icp
8	Drost-CVPR10-Edges [10]	2019	yes	-	-	-	rgb-d	icp
9	CDPNv2_BOP20-PBR-ICP [35]	2020	-	1/obj	rgb	pbr	rgb-d	icp
10	CDPNv2_BOP20-RGB [35]	2020	-	1/obj	rgb	pbr+real	rgb	-
11	Drost-CVPR10-3D-Edges [10]	2019	yes	-	-	-	d	icp
12	Drost-CVPR10-3D-Only [10]	2019	yes	-	-	-	d	icp
13	CDPN_BOP19-RGB [35]	2020	-	1/obj	rgb	pbr+real	rgb	-
14	CDPNv2_BOP20-PBR [35]	2020	-	1/obj	rgb	pbr	rgb	-
15	leaping from 2D to 6D [37]	2020	-	1/obj	rgb	pbr+real	rgb	-
16	EPOS-BOP20-PBR [21]	2020	-	1/set	rgb	pbr	rgb	-
17	Drost-CVPR10-3D-Only-Faster [10]	2019	yes	-	-	-	d	icp
18	Félix&Neves-ICRA17-IET19 [46,44]	2019	yes	1/set	rgb-d	syn+real	rgb-d	icp
19	Sundermeyer-IJCV19+ICP [49]	2019	-	1/obj	rgb	syn+real	rgb-d	icp
20	Zhigang-CDPN-ICCV19 [35]	2019	-	1/obj	rgb	syn+real	rgb	-
21	PointVoteNet2 [15]	2020	-	1/obj	rgb-d	pbr	rgb-d	icp
22	Pix2Pose-BOP20-ICCV19 [40]	2020	-	1/obj	rgb	pbr+real	rgb	-
23	Sundermeyer-IJCV19 [49]	2019	-	1/obj	rgb	syn+real	rgb	-
24	SingleMultiPathEncoder-CVPR20 [47]	2020	-	1/all	rgb	syn+real	rgb	-
25	Pix2Pose-BOP19-ICCV19 [40]	2019	-	1/obj	rgb	syn+real	rgb	-
26	DPOD (synthetic) [54]	2019	-	1/scene	rgb	syn	rgb	-

Table 2. **Results of the BOP Challenge 2019 and 2020.** The methods are ranked by the AR_{Core} score (the third column of the upper table) which is the average of the per-dataset AR_D scores (the following seven columns). The scores are defined in Sec. 2.4. The last column of the upper table shows the average image processing time [s] averaged over the datasets. The lower table shows properties discussed in Sec. 4.

in [33], using the augmentation for training the pose estimation models improved the accuracy on T-LESS from 37.0% to 63.8%. Access to a GPU cluster was also crucial as training of one network took ~10 hours on 32 GPU's.

The second is a hybrid method by König and Drost [32] with AR_{Core} of 63.9%. This method first predicts object instance masks by RetinaMask [12] or Mask R-CNN [16], whichever performs better on the validation set. Then, for each mask, the method selects the corresponding part of the 3D point cloud of the test scene, and estimates the object pose using the point pair features [10]. The method is noticeably faster than the top-performing CosyPose variant, mainly thanks to a highly optimized implementation of ICP from HALCON [2].

Another method which outperformed Vidal-Sensors18 is Pix2Pose by Park *et al.* [40] with AR_{Core} of 59.1%. This method predicts 2D-3D correspondences between densely sampled image pixels and the 3D object model, solves for the poses using the PnP-RANSAC algorithm, and refines the poses with a depth-based ICP algorithm. The ICP refinement is crucial for this method as it improves the AR_{Core} score by absolute 24.9% and teleports the method from the 22nd to the 4th place. The importance of a refinement stage has been demonstrated also by other methods – top nine methods applied ICP or an RGB-based refiner, similar to DeepIM [34] (column "Refine" in Tab. 2).

Training a special DNN model per object has been a common practise in the field, followed also by most participants of the challenge. However, the CosyPose and König-Hybrid-DL-PointPairs methods have shown that a single DNN model can be effectively shared among multiple objects (column "DNN" in Tab. 2). CosyPose trains three models per dataset – one for detection, one for coarse pose estimation, and one for iterative pose refinement, whereas König-Hybrid-DL-PointPairs trains only one model for instance segmentation.

4.3 The Effectiveness of Photorealistic Training Images

In 2020, most DNN-based methods were trained either only on the photorealistic (PBR) training images, or also on real training images which are available in datasets T-LESS, TUD-L, and YCB-V (column "Train type" in Tab. 2)[5]. Although adding real training images yields higher scores (compare scores of methods #3 and #5 or #10 and #14 on T-LESS, TUD-L, and YCB-V in Tab. 2), competitive results can be achieved with PBR images only, as demonstrated by the overall fifth PBR-only variant of the CosyPose method. This is an important result considering that PBR-only training does not require any human effort for capturing and annotating real training images.

[5] Method #2 used also synthetic training images obtained by cropping the objects from real validation images in the case of HB and ITODD and from OpenGL-rendered images in the case of other datasets, and pasting the cropped objects on images from the Microsoft COCO dataset [36]. Method #24 used PBR and real images for training Mask R-CNN [16] and OpenGL images for training a single Multi-path encoder. Two of the CosyPose variants (#1 and #3) also added the "render & paste" synthetic images provided in the original YCB-V dataset, but these images were later found to have no effect on the accuracy score.

Detection	Pose estim.	T-LESS	TUD-L	YCB-V
PBR+Real	PBR+Real	72.8	82.3	82.1
PBR	PBR	64.0	68.5	57.4
PBR	Render & paste v1	16.1	60.4	44.9
PBR	Render & paste v2	60.0	58.9	58.5
Render & paste v1	Render & paste v1	6.1	49.5	26.5
Render & paste v2	Render & paste v2	45.3	42.4	25.7

Table 3. **The effect of different training images**. The table shows the $\mathrm{AR}_{\mathrm{Core}}$ scores achieved by the CosyPose method [33] when different types of images were used for training its object detection (*i.e.* Mask R-CNN [16]) and pose estimation stage. The "render & paste v1" images were obtained by OpenGL rendering of the 3D object models on random real photographs. The "render & paste v2" images were obtained similarly, but the CAD models of T-LESS objects were assigned a random surface texture instead of a random gray value, the background of most images was assigned a synthetic texture, and 1M instead of 50K images were generated. Interestingly, the increased photorealism brought by the PBR images yields noticeable improvements despite the strong data augmentation applied by CosyPose to the training images.

The PBR training images yield a noticeable improvement over the "render & paste" synthetic images obtained by OpenGL rendering of the 3D object models on real photographs. For example, the CDPN method with the same hyper-parameter settings improved by absolute 20.2% on HB, by 19.5% on LM-O, and by 7% on IC-BIN when trained on 50K PBR images per dataset *vs.* 10K "render & paste" images per object (compare methods #13 and #20 in Tab. 2). As shown in Tab. 3, the CosyPose method improved by a significant 57.9% (from 6.1% to 64.0%) on T-LESS, by 19.0% on TUD-L, and by 30.9% on YCB-V when trained on 50K PBR images per dataset *vs.* 50K "render & paste v1" images per dataset. The "render & paste v1" images used for training CosyPose were obtained by imitating the PBR images, *i.e.* the 3D object models were rendered in the same poses as in the PBR images and pasted on real backgrounds.

As an additional experiment, we have trained the CosyPose method on another variant of the "render & paste" images, generated as in [33] and referred to as "render & paste v2". The main differences compared to the "render & paste v1" variant described in the previous paragraph are: (a) the CAD models of T-LESS objects were assigned a random surface texture instead of a random gray value, (b) the background was assigned a real photograph in 30% images and a synthetic texture in 70% images, and (c) 1M instead of 50K images were generated. As shown in Tab. 3, "render & paste v2" images yield a noticeable improvement of 39.2% over "render & paste v1" on T-LESS, but no improvement on TUD-L (−7.1%) and YCB-V (−0.8%). This may suggest that randomizing the surface texture of the texture-less CAD models of T-LESS objects improves the generalization of the network by forcing the network to focus more on shape than on lower-level patterns, as found in [13]. When generating the PBR images, which yield the highest accuracy on T-LESS, the CAD models were assigned a random gray value, as in "render & paste v1", but the effect of randomizing the surface texture may have been achieved by randomizing the PBR material (Sec. 3.2) – further investigation is needed to clearly answer these questions. The importance of both the objects and the background being synthetic, as suggested in [20], has not been confirmed in this experiment – "render & paste v1" images with only real backgrounds achieved higher scores than "render & paste v2" images on TUD-L and YCB-V. However, the first ten convolutional layers of Mask R-CNN ("conv1" and "conv2_x" of ResNet-50 [17]) used for object detection in CosyPose were pre-trained on Microsoft COCO [36] but not fine-tuned, whereas all layers were fine-tuned in [20]. The benefit of having 1M *vs.* 50K images is indecisive since 50K PBR images were sufficient to achieve high scores.

Both types of "render & paste" images are far inferior compared to the PBR images, which yield an average improvement of 35.9% over "render & paste v1" and 25.5% over "render & paste v2" images (Tab. 3). Interestingly, the increased photorealism brought by the PBR images is important despite the strong data augmentation that CosyPose applies to the training images. Since object poses in the PBR and "render & paste v1" images are identical, the ray-tracing rendering technique, PBR materials and objects realistically embedded in synthetic environments seem to be the decisive factors for successful "sim2real" transfer [6].

We have also observed that the PBR images are more important for training DNN models for object detection/segmentation (*e.g.* Mask R-CNN [16]) than for training DNN models for pose estimation from the detected regions (Tab. 3). In the case of CosyPose, if the detection model is trained on PBR images and the later two models for pose estimation are trained on the "render & paste v2" instead of the PBR images, the accuracy drops moderately (64.0% to 60.0% on T-LESS, 68.5% to 58.9% on TUD-L) or does not change much (57.4% *vs.* 58.5% on YCB-V). However, if also the detection model is trained on the "render & paste v1" or "render & paste v2" images, the accuracy drops severely (the low accuracy achieved with "render & paste v1" on T-LESS was discussed earlier).

5 Awards

The following BOP Challenge 2020 awards were presented at the 6th Workshop on Recovering 6D Object Pose [28], organized in conjunction with the ECCV 2020 conference. Results on the core datasets are in Tab. 2 and results on the other datasets can be found on the project website.

The Overall Best Method (the top-performing method on the core datasets): CosyPose-ECCV20-Synt+Real-ICP by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best RGB-Only Method (the top-performing RGB-only method on the core datasets): CosyPose-ECCV20-Synt+Real by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best Fast Method (the top-performing method on the core datasets with the average running time per image below 1s): König-Hybrid-DL-PointPairs by Rebecca König and Bertram Drost [32].

The Best BlenderProc4BOP-Trained Method (the top-performing method on the core datasets which was trained only with the provided BlenderProc4BOP images): CosyPose-ECCV20-PBR by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best Single-Model Method (the top-performing method on the core datasets which uses a single machine learning model, typically a neural network, per dataset): CosyPose-ECCV20-Synt+Real-ICP by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best Open-Source Method (the top-performing method on the core datasets whose source code is publicly available): CosyPose-ECCV20-Synt+Real-ICP by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best Method on Datasets LM-O, TUD-L, IC-BIN, and YCB-V: CosyPose-ECCV20-Synt+Real-ICP by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best Method on Datasets ITODD and TYO-L: Drost-CVPR10-Edges by Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic [10].

The Best Method on Dataset LM: DPODv2 (synthetic train data, RGB + D Kabsch) by Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic [54].

The Best Method on Dataset T-LESS: CosyPose-ECCV20-Synt+Real by Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic [33].

The Best Method on Dataset HB: CDPNv2_BOP20 (RGB-only) by Zhigang Li, Gu Wang, and Xiangyang Ji [35].

The Best Method on Dataset RU-APC: Pix2Pose-BOP19_w/ICP-ICCV19 by Kiru Park, Timothy Patten, and Markus Vincze [40].

The Best Method on Dataset IC-MI: Drost-CVPR10-3D-Only by Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic [10].

6 Conclusions

In 2020, methods based on neural networks have finally caught up with methods based on point pair features, which were dominating previous editions of the challenge. Although the top-performing methods rely on RGB-D image channels, strong results have been achieved with RGB channels only. The challenge results and additional experiments with the top-performing CosyPose method [33] have shown the importance of PBR training images and of strong data augmentation for successful "sim2real" transfer. The scores have not been saturated and we are already looking forward to the insights from the next challenge.

This research was supported by CTU student grant (SGS OHK3-019/20), Research Center for Informatics (CZ.02.1.01/0.0/0.0/16_019/0000765 funded by OP VVV), and HPC resources from GENCI-IDRIS (grant 011011181).

References

1. Intel Open Image Denoise (2020), https://www.openimagedenoise.org/ 8
2. MVTec HALCON (2020), https://www.mvtec.com/halcon/ 11
3. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6D object pose estimation using 3D object coordinates. ECCV (2014) 5, 7
4. Community, B.O.: Blender – a 3D modelling and rendering package (2018), http://www.blender.org 8
5. Demes, L.: CC0 Textures. https://cc0textures.com/ (2020) 8
6. Denninger, M., Sundermeyer, M., Winkelbauer, D., Olefir, D., Hodaň, T., Zidan, Y., Elbadrawy, M., Knauer, M., Katam, H., Lodhi, A.: BlenderProc: reducing the reality gap with photorealistic rendering. Robotics: Science and Systems (RSS) Workshops (2020) 2, 3, 7, 8, 13
7. Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H.: BlenderProc. arXiv preprint arXiv:1911.01911 (2019) 2, 3, 7, 8
8. Doumanoglou, A., Kouskouridas, R., Malassiotis, S., Kim, T.K.: Recovering 6D object pose and predicting next-best-view in the crowd. CVPR (2016) 7
9. Drost, B., Ulrich, M., Bergmann, P., Hartinger, P., Steger, C.: Introducing MVTec ITODD – A dataset for 3D object recognition in industry. ICCVW (2017) 5, 7
10. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3D object recognition. CVPR (2010) 2, 9, 10, 11, 14
11. Dwibedi, D., Misra, I., Hebert, M.: Cut, paste and learn: Surprisingly easy synthesis for instance detection. ICCV (2017) 2
12. Fu, C.Y., Shvets, M., Berg, A.C.: Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. arXiv preprint arXiv:1901.03353 (2019) 11
13. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018) 12
14. Godard, C., Hedman, P., Li, W., Brostow, G.J.: Multi-view reconstruction of highly specular surfaces in uncontrolled environments. 3DV (2015) 2
15. Hagelskjær, F., Buch, A.G.: Pointposenet: Accurate object detection and 6 dof pose estimation in point clouds. arXiv preprint arXiv:1912.09057 (2019) 9, 10
16. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017) 9, 11, 12, 13
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (2016) 13
18. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. ACCV (2012) 5, 7, 18
19. Hinterstoisser, S., Lepetit, V., Wohlhart, P., Konolige, K.: On pre-trained image features and synthetic images for deep learning. ECCVW (2018) 2
20. Hinterstoisser, S., Pauly, O., Heibel, H., Martina, M., Bokeloh, M.: An annotation saved is an annotation earned: Using fully synthetic training for object detection. ICCVW (2019) 13
21. Hodaň, T., Baráth, D., Matas, J.: EPOS: Estimating 6D pose of objects with symmetries. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 10

22. Hodaň, T., Brachmann, E., Drost, B., Michel, F., Sundermeyer, M., Matas, J., Rother, C.: BOP Challenge 2019. `https://bop.felk.cvut.cz/media/bop_challenge_2019_results.pdf` (2019) 1
23. Hodaň, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. IEEE Winter Conference on Applications of Computer Vision (WACV) (2017) 2, 7
24. Hodaň, T., Matas, J., Obdržálek, Š.: On evaluation of 6D object pose estimation. European Conference on Computer Vision Workshops (ECCVW) (2016) 1, 5, 17, 18
25. Hodaň, T., Michel, F., Brachmann, E., Kehl, W., Glent Buch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., Sahin, C., Manhardt, F., Tombari, F., Kim, T.K., Matas, J., Rother, C.: BOP: Benchmark for 6D object pose estimation. ECCV (2018) 1, 4, 5, 7
26. Hodaň, T., Michel, F., Sahin, C., Kim, T.K., Matas, J., Rother, C.: SIXD Challenge 2017. `http://cmp.felk.cvut.cz/sixd/challenge_2017/` (2017) 1, 4
27. Hodaň, T., Sundermeyer, M.: BOP Toolkit (2020), `https://github.com/thodan/bop_toolkit` 8, 9
28. Hodaň, T., Sundermeyer, M., Kouskouridas, R., Kim, T.K., Matas, J., Rother, C., Lepetit, V., Leonardis, A., Walas, K., Steger, C., Brachmann, E., Drost, B., Sock, J.: 6th international workshop on recovering 6D object pose. `http://cmp.felk.cvut.cz/sixd/workshop_2020/` (2020) 13
29. Hodaň, T., Vineet, V., Gal, R., Shalev, E., Hanzelka, J., Connell, T., Urbina, P., Sinha, S., Guenter, B.: Photorealistic image synthesis for object instance detection. IEEE International Conference on Image Processing (ICIP) (2019) 2, 8
30. Kaskman, R., Zakharov, S., Shugurov, I., Ilic, S.: HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects. ICCVW (2019) 7
31. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. ICCV (2017) 2
32. Koenig, R., Drost, B.: A hybrid approach for 6dof pose estimation. ECCVW (2020) 2, 10, 11, 13
33. Labbé, Y., Carpentier, J., Aubry, M., Sivic, J.: CosyPose: consistent multi-view multi-object 6D pose estimation. ECCV (2020) 2, 9, 10, 11, 12, 13, 14
34. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: DeepIM: deep iterative matching for 6d pose estimation. In: ECCV (2018) 11
35. Li, Z., Wang, G., Ji, X.: CDPN: Coordinates-based disentangled pose network for real-time RGB-based 6-DoF object pose estimation. ICCV (2019) 10, 14
36. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. ECCV (2014) 11, 13
37. Liu, J., Zou, Z., Ye, X., Tan, X., Ding, E., Xu, F., Yu, X.: Leaping from 2D detection to efficient 6DoF object pose estimation. ECCVW (2020) 10
38. Marschner, S., Shirley, P.: Fundamentals of computer graphics. CRC Press (2015) 8
39. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. ISMAR (2011) 2, 6
40. Park, K., Patten, T., Vincze, M.: Pix2Pose: Pixel-wise coordinate regression of objects for 6D pose estimation. ICCV (2019) 2, 10, 11, 14
41. Pharr, M., Jakob, W., Humphreys, G.: Physically based rendering: From theory to implementation. Morgan Kaufmann (2016) 8
42. Qian, Y., Gong, M., Hong Yang, Y.: 3d reconstruction of transparent objects with position-normal consistency. CVPR (2016) 2

43. Rad, M., Lepetit, V.: BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. ICCV (2017) 2
44. Raposo, C., Barreto, J.P.: Using 2 point+normal sets for fast registration of point clouds with small overlap. ICRA (2017) 10
45. Rennie, C., Shome, R., Bekris, K.E., De Souza, A.F.: A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place. RA-L (2016) 7
46. Rodrigues, P., Antunes, M., Raposo, C., Marques, P., Fonseca, F., Barreto, J.: Deep segmentation leverages geometric pose estimation in computer-aided total knee arthroplasty. Healthcare Technology Letters (2019) 10
47. Sundermeyer, M., Durner, M., Puang, E.Y., Marton, Z.C., Vaskevicius, N., Arras, K.O., Triebel, R.: Multi-path learning for object pose estimation across domains. CVPR (2020) 10
48. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3D orientation learning for 6D object detection from RGB images. In: ECCV (2018) 9
49. Sundermeyer, M., Marton, Z.C., Durner, M., Triebel, R.: Augmented autoencoders: Implicit 3D orientation learning for 6D object detection. IJCV (2019) 2, 10
50. Tejani, A., Tang, D., Kouskouridas, R., Kim, T.K.: Latent-class hough forests for 3D object detection and pose estimation. ECCV (2014) 7
51. Vidal, J., Lin, C.Y., Lladó, X., Martí, R.: A method for 6D pose estimation of free-form rigid objects using point pair features on range data. Sensors (2018) 2, 9, 10
52. Wu, B., Zhou, Y., Qian, Y., Cong, M., Huang, H.: Full 3D reconstruction of transparent objects. ACM TOG (2018) 2
53. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. RSS (2018) 7
54. Zakharov, S., Shugurov, I., Ilic, S.: DPOD: 6D pose object detector and refiner. ICCV (2019) 10, 14

A Discussion on the Evaluation Methodology

A.1 6D Object Localization *vs.* 6D Object Detection

Prior information about the object presence in the input image distinguishes two 6D object pose estimation tasks: 6D object localization, where the identifiers of present object instances are provided for each image, and 6D object detection, where no prior information is provided [24].

The aspect which is evaluated on the 6D object detection but not on the 6D object localization task is the capability of the method to calibrate the predicted confidence scores across all object classes. For example, a score of 0.5 for a cat should represent the same level of confidence as a score of 0.5 for a duck. This calibration is important for achieving good performance *w.r.t.* the precision/recall curve which is typically used for evaluating detection. The 6D object localization task still requires the method to sort the hypotheses, although only within the same object class – the method needs to output the top n pose estimates for an object class which are evaluated against n ground-truth poses of that class.

In BOP, methods have been so far evaluated on the 6D object localization task for two reasons. First, the accuracy scores on this simpler task are still far from being saturated. Second, the 6D object detection task requires computationally expensive evaluation as many more hypotheses need to be evaluated to calculate the precision/recall curve. Calculating the 6D pose errors is more expensive than, *e.g.*, calculating the intersection over union of 2D bounding boxes (used to evaluate 2D object detection).

A.2 The Choice of Pose-Error Functions

The object pose may be ambiguous, *i.e.*, there may be multiple poses that are consistent with the image. This is caused by the existence of multiple fits of the visible part of the object surface to the 3D object model. The visible part is determined by self-occlusion and occlusion by other objects and the multiple surface fits are induced by global or partial object symmetries. As a consequence, there may be (infinitely) many indistinguishable 6D poses which should be treated as equivalent, but explicitly enumerating all of these poses is often difficult.

The most widely used pose-error functions have been ADD/ADI [24,18], where the error is calculated as the average distance from vertices of the object model in the ground-truth pose to vertices of the model in the estimated pose. The distance is measured between corresponding vertices if all views of the object are distinguishable (ADD). Otherwise, for objects with indistinguishable views, the distance is measured between a vertex and its nearest neighbor in the 3D space, which may not necessarily be the corresponding vertex (ADI). ADI can yield unintuitively low errors even for poses that are distinguishable. Objects evaluated with ADI therefore tend to have low pose errors although the estimated poses might not be visually well aligned. Another limitation of ADD/ADI comes from a high dependence on the geometry of the object model and the sampling density of its surface – the average distance is dominated by higher-frequency surface parts such as the thread of a fuse. The maximum distance used in MSSD and MSPD is less dependent on the geometry and sampling of the object model.

MSSD is relevant for robotic grasping as it measures the error in the 3D space, and MSPD is relevant for augmented reality applications as it measures the error in the projective space. Both MSSD and MSPD can handle pose ambiguities due to global object symmetries. However, because both are calculated over the entire model surface, misalignments of invisible surface parts are penalized. This may not be desirable for applications such as robotic manipulation with suction cups where only the alignment of the visible part is relevant. VSD is calculated only over the visible object part and therefore treats all poses that are consistent with the image as equivalent. VSD evaluates the alignment of the object shape but not of its color. This is because most of the object models currently included in BOP have baked shadows and reflections in their surface textures, which makes it difficult to robustly evaluate the color alignment.

As each of VSD, MSSD, and MSPD evaluates different qualitites of the pose estimates and each is relevant for a different target application, we use all three of these pose-error functions for the evaluation in BOP.