Generalizibility of 6D Pose Recognition Algoirthms on Unseen Data

Bo-Hsun Chen bchen293@wisc.edu
Keaton Leppanen kleppanen@wisc.edu
Saym Imtiaz simtiaz@wisc.edu
Carter Sifferman siferman@wisc.edu

Abstract

For this project we investigate the generalizability of state of the art 6D pose recognition approaches. In recent years, new approaches in this area have significantly improved the state of the art in terms of efficiency and accuracy, introducing new techniques to deal with occlusions and limited training data. However, the datasets with which these techniques are trained and tested on tend to be homogeneous and unrealistic, which can encourage the development of techniques that over-fit to the test set, rather than perform well in real-world environments. In this work, we consider three state of the art 6D pose recognition approaches: DPOD, EPro-PnP, and EfficientPose. We evaluate how well they generalize to unseen data in which the object of interest's color, background, and/or lighting have been perturbed. We find that existing pose recognition algorithms which perform well on typical testing datasets do not necessarily generalize well. Furthermore, we find that approaches which rely on generating keypoints are more generalizable than those that rely on direct inference. Our results speak to the need for a re-evaluation of 6D pose recognition performance evaluation and a more diverse testing dataset for benchmarking.

1. Introduction

Object detection is an important and fundamental computer vision task that has elicited much attention and research. However, for some real-world applications, such as robotic manipulation of objects, augmented reality, and autonomous driving, standard object detection is not sufficient. Since these tasks require a more sophisticated understanding of an object's position and orientation, they necessitate the use of 6D pose recognition [1] .

6-dimensional (6D) pose recognition is the task of predicting the 6D pose (3D translation and 3D rotation relative to a camera) of an object, given an image of that object. This is inherently a more difficult problem than standard, 2D object detection and brings with it a variety of unique challenges.

For instance, current approaches rely on the algorithm having some representation of the 3D object they are predicting the pose of, meaning that an individual model must be trained for each specific object in order to find its pose. Exacerbating these intensive training requirements is the fact that data creation and labeling for 6D pose recognition is substantially more difficult than for other tasks as the images must contain information about the camera which took the picture as well as a 6D bounding box denoting the ground truth [4]. As a consequence of this, most of the benchmark datasets used for training and testing are limited in size, homogeneity, and are generally unrealistic.

These challenges naturally give rise to questions about the models' generalizability - the hope being that a more generalizable model would adapt better to real-world applications and help reduce the amount of models and training data needed.

Interested in exploring this area, we set out to evaluate and compare different state of the art 6-D pose recognition approaches in terms of generalizability. Specifically, we generate a perturbed test dataset and then use it as the basis to evaluate 3 state of the art approaches: DPOD, EPro-PnP, and EfficientPose. Our results indicate that approaches which use direct inference to directly predict an object's 6D pose, such as EfficientPose, are far less generalizable than those that use as intermediate step of identifying keypoints and then solving the Perspective-n-Points (PnP) problem to predict the pose, such as DPOD and EPro-PnP.

2. Existing Approaches

There exists a wide variety of methods for solving the 6D pose recognition task, each with various differences in their approaches, such as whether or not they rely on depth information or simply RGB images for prediction. However, at a high level, most of the methods competing for state of the art results fall into one of two categories: PnP-based approaches and direct prediction based approaches [10]. Some approaches include accommodations for dealing with occluded objects, and special datasets exist for testing occlusions. For the sake of simplicity, we focus on non-occluded objects in this work.

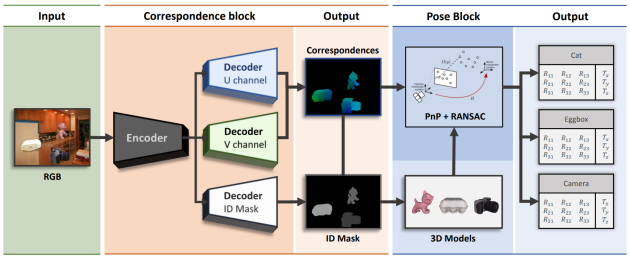


Figure 1. The model structure of DPOD [16].

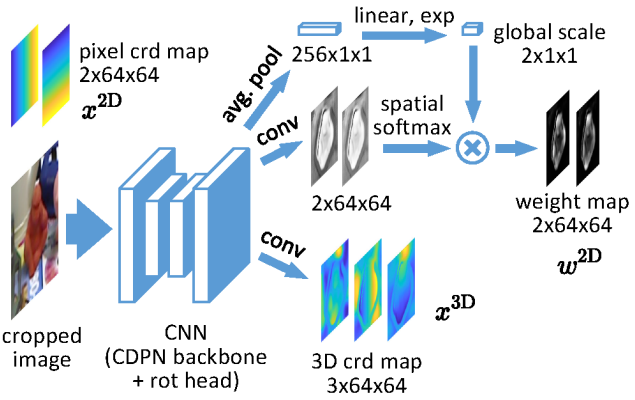


Figure 2. The model structure of EPro-PnP [2].

2.1. PnP-Based

PnP-based methods are two stage methods which involve first establishing a 2D-3D correspondence between a 3D model of the object of interest and a 2D picture of said object. These correspondence points are often referred to as 'keypoints' and are generally found through the use of a trained convolutional neural network (CNN). Once these keypoints are found, they can be used to solve the PnP problem, which is a classic computer vision problem which uses keypoints to solve for the pose of the object in the 2D image [16]. There exists many algorithmic approaches for solving the PnP problem, but most of the state of the art 6D pose recognition methods rely on a RANSAC-based algorithm [5].

PnP-based approaches have shown that this method can be used to achieve impressive precision and passable efficiency, however this paradigm of 6D pose recognition does have its limitations. For instance, the fact that it is a 2-stage method instead of an end-to-end method prevents it from being applied directly to some downstream tasks. In addition, the necessity to solve the PnP problem for each prediction represents a substantial computational overhead which scales for each object present in a scene as it must be solved individually for each object [1].

Two of the models we investigated in this project fall into this category: DPOD and EPro-PnP.

2.1.1 DPOD

The DPOD method is from Zakharov et al. [16], the algorithm structure is shown in Figure 1, and the implementation is modified from [15] for working on Google Colab.

During data pre-processing, each UV texture map corresponding to the surface texture of the object 3D model was firstly built for each kind of object. So, given UV texture coordinates, it obtains the related 3D vertex point position on the object by this map. The neural network (NN) structure in Figure 1 can be separated into two stages. In the first stage, it trains a neural network as a correspondence mapping, which maps points on an input 2D image of the object to the corresponding UV coordinates of its UV texture map as output. The model backbone in this stage is UNet [11], and all classes of objects share the same correspondence mapping model with predicting the ID mask for each recognized object in the 2D input image.

The second stage fine-tunes the prediction results. It compares the rendered object image generated from the predicted correspondence from the first stage with the original real input object image. It trains another neural network model to take these two images as inputs and then outputs the final refined predicted correspondences. The backbone model in this stage is ResNet18 [6],which utilizes the residual concept to refine the prediction for original real input images.

Lastly, it uses the PnP and RANSAC algorithms to find the pose transformation.

2.1.2 EPro-PnP

EPro-PnP is a method from Chen et al. [2], an end-to-end probabilistic PnP approach for 6D pose estimation. The main idea is that deterministic pose is non-differentiable, but that a probability density of pose is differentiable. Therefore, the approach outputs PnP as a probabilistic distribution that learns 2D-3D correspondences.

EPro-PnP, given an object proposal, predicts a distribution of poses made up of a set of corresponding keypoints with 3D object coordinates, 2D image coordinates, and 2D weights. A Kullback-Leibler (KL) divergence loss is applied during training to encourage a higher probability for correct predictions and penalize incorrect predictions. The model also uses an iterative PnP solver based on the Levenburg-Marquadt (LM) algorithm to solve for the true pose.

EPro-PnP uses the the network from CDPN [8] as a baseline. However, one notable difference is that EPro-PnP removes the translation head used in CDPN. The model then uses the original confidence map from CDPN by ex-

panding it to two-channel XY weights and applies spatial Softmax and global weight scaling. EPro-PnP also utilizes the masked coordinate regression loss from CDPN, but removes the confidence loss in order to make use of geometric knowledge from the 3D models. The 6-DOF pose estimation network as modified from CDPN can be seen in Figure 2.

2.2. Direct Prediction

Direct prediction methods are end-to-end methods which involve training a convolutional neural network to take a 2D image of an object as input and directly output the 6D pose of said object. Much of the variation in methods in this paradigm come in the form of different architectures [13]. Although early direct prediction approaches failed to achieve state of the art results with PnP-based approaches, newer model architectures and approaches have achieved not only competitive results, but have actually surpassed many PnP-based approaches in terms of accuracy and efficiency. The fact that they are trained end-to-end gives them the added benefits of being less computationally intensive, they can scale relatively easily to the prediction of multiple instances of objects, making them more attractive for real-world applications. In addition, they have the potential for further learning via self supervised learning, and they can be more easily incorporated into other down-steam tasks compared to PnP-based approaches [1].

In this project, we investigated one such direct prediction method: EfficientPose.

2.2.1 EfficientPose

With an accuracy in terms of the ADD(-S) metric of 97.35%, EfficientPose is the current state of the art when in comes to 6D pose recognition. In addition, it is extremely efficient, running end-to-end at over 27 FPS, and can predict multiple objects in a single shot, qualities which make it a good option for many real-world applications [1]. These benefits come in large part from the one stage nature of direct prediction approaches like EfficientPose as it cuts out the added time of solving the PnP problem.

At a high level, EfficientPose can be thought of the adaptation of the widely known EfficientNet and EfficientDet from standard 2D object detection to the task of 6D pose recognition. EfficientDet starts with an EfficientNet convolutional network as its backbone and builds upon it by introducing a bidirectional feature pyramid network and subnets which output both a class and box prediction [12]. In order to aptly adapt this architecture to the task of 6D pose recognition, EfficientPose incorporates two additional output subnets which predict the detected object's rotation and translation, as seen in Figure 3. Compared to PnP-based approaches, EfficientPose is not only more efficient, but also more accurate.

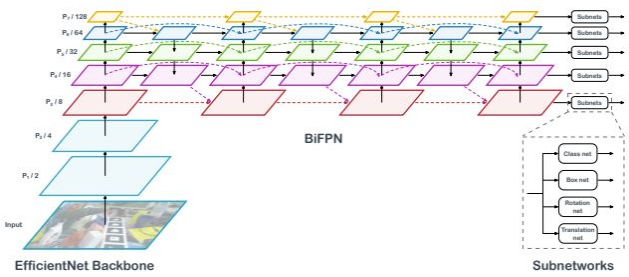


Figure 3. The model structure of EfficientPose [1].

2.3. Datasets

There are a number of datasets available for 6D pose recognition, which include images of 6D objects, their models, and labeled ground truth and camera intrinsics. While the Object Recognition Challenge Website[1] lists 11 total datasets, the majority of recent papers [1, 3, 9, 14] make use of only a few, with all of them using the Linemod dataset. Linemod is also used on the popular paperswithcode.com leaderboard[2] as *the* dataset for 6D pose recognition, analagous to ImageNet [4] for image recognition.

Linemod is effectively two datasets: 1253 real images of 15 rigid objects on a desktop, and 50,000 synthetic images of 3D scanned models of the same 15 objects scattered on a ground plane (this is sometimes referred to as "PBR"). The real images are typically reserved for testing, while the synthetic images are sometimes used for training. Some approaches split the real images into a training and test set, and use them both for training. The test set for Linemod is very homogeneous. All images are taken on the same desk, under the same lighting, and while the objects are moved around, they always stand upright. Examples of images from the Linemod synthetic and real dataset are shown in Fig. 4.

3. Technical Approach

3.1. Novel Datasets

We generate three new datasets, which are variations on the widely-used Linemod dataset [7]. While Linemod contains fifteen object models, for simplicity and ease of comparison, we focus on one model: the yellow rubber duck, shown in Fig. 5. All of our datasets contain only the duck model, and all of our experiments (Section Sec. 4) are performed on the duck alone. We provide an open-source implementation of the programs used to generate

[1] https://bop.felk.cvut.cz/datasets/

[2] https://paperswithcode.com/sota/6d-pose-estimation-on-linemod

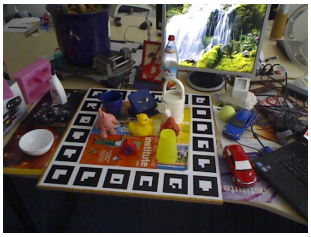
(a) Real image from Linemod

(b) Real image from Linemod

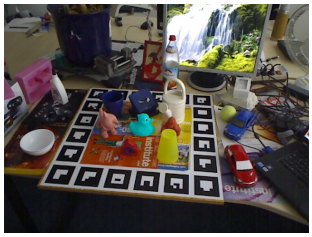
(c) Synthetic image from Linemod

(d) Synthetic image from Linemod

Figure 4. Example images from the Linemod base dataset



(a) Unmodified Linemod



(b) Real Blue Ducks

(c) Synthetic Yellow Ducks

(d) Synthetic Blue Ducks

Figure 5. Example images from various datasets

these datasets[3]. Downloads are available upon request.

3.1.1 Real Blue Ducks

We introduce a dataset based on the Linemod test set, which we refer to as the "real blue ducks" dataset. This dataset is exactly the same as the Linemod test dataset, containing 1253 real images of objects on a table, as shown in Fig. 5a, except the pixels corresponding to the duck (object 9) have been hue shifted 180°. In order to do this hue shift on only the duck, we rely on the pixel-wise masks provided by EfficientPose's processed dataset [1]. The result is a dataset which tests only how well these models generalize to a differently colored object, without modifying any other context in the image which the model might be using to "cheat".

3.1.2 Synthetic Yellow Ducks

In order to evaluate how much models are over-fitting to their training data and/or the real image testing data, we render our own never-before-seen synthetic dataset. This dataset eliminates the context clues which the models could be using to "cheat": the ground plane, which the objects are always upright on in Linemod, the other objects, which are always positioned upright relative to each other, and the consistent lighting. To accomplish this, we write our own dataset generator, which uses the same rendering toolkit, BlenderProc[4], as synthetic Linemod images. However, our generator includes the following changes from the synthetic Linemod dataset:

1. Only one object (object 9, duck) is included in each rendered image
2. No ground plane is present
3. Random images from VOC2011[5] are used as background
4. 1-3 random point lights are included in each scene

An example image of this dataset is shown in Fig. 5c.

3.1.3 Synthetic Blue Ducks

We also generate a dataset in the same manner as "Synthetic Yellow Ducks", but with ducks which are blue. Rather than using a pixel-wise mask to make the ducks blue, we change the object model itself, by writing a custom Python script to modify the RGB values in the `obj_000009.ply` file provided by Linemod. This dataset can be used, for example, to evaluate how well a model trained on yellow synthetic ducks can infer the pose of synthetic blue ducks. This is a similar use case to "Real Blue Ducks" on real images. However, the color modification of the synthetic ducks is more realistic as it takes place on the model itself, rather than the final image.

[3] https://github.com/cpsiff/LineMOD-Synthetic-Extension
[4] https://github.com/DLR-RM/BlenderProc
[5] http://host.robots.ox.ac.uk/pascal/VOC/voc2011/index.html

Dataset	Approach (ADD-S Portion Correct) EfficientPose	DPOD	EPro-PnP
Base Linemod	**0.9099**	0.8453*	0.8479
Real Blue	0.3850	**0.8445**	0.0900
Synthetic Yellow	0.0100	**0.9000**	-
Synthetic Blue	0.0000	**0.9500**	-

Table 1. Results of naively applying author-provided pre-trained models to various datasets. While EfficientPose performs better on base LineMOD, it does not generalize well. EPro-PnP: no quantitative results found, see qualitative results in Sec. 4. (* 15% of data were used to train the DPOD model.)

4. Experiments

We evaluate three existing approaches: EfficientPose, DPOD, and EPro-PnP. For EfficientPose and EPro-PnP, first, we recreate the results shown in each paper, using the pre-trained models provided by the authors, and performing inference on the real-world Linemod test dataset (sometimes called `lm_test_all`). We then use those same pre-trained models to perform inference on our three modified datasets. And for DPOD, the NN models were trained from scratch since no trained models accessible. The DPOD models were trained on Linemod test dataset, where each class contains about 1200 images. only 15% of all images in this dataset were used for training. Besides, 99% of training images were changed the background by using MS COCO dataset to reduce background influence. The first-staged model was trained for 20 epochs, while the second-staged model was trained for 10 epochs. All training and testing of DPOD models were run on Google Colab.

For each method, we only compare the ADD-S metric, first proposed in [14]. We choose this metric because it is supported by each of the approaches that we implement, and it is the most commonly used across recent papers. The ADD-S metric follows a different formula depending on whether the object of interest is symmetrical, or not. Because our duck model is not symmetrical, ADD-S is always calculated as follows:

$$ADD = \frac{1}{m} \sum_{x \in \mathcal{M}} ||(\mathbf{R}\mathbf{x} + \mathbf{T}) - (\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{T}})||_2 \qquad (1)$$

Where $\mathcal{M}$ is the set of all points on the 3D object, $m = |\mathcal{M}|$, $\mathbf{R}$ and $\mathbf{T}$ represent the ground truth rigid transform from camera to object space, and $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{T}}$ are the predicted transform. We count a prediction as correct if its ADD is less than 10% of the object's diameter. The values shown in Tab. 1 and Tab. 2 are the portion of the test set which the model predicted correctly by this metric.

Due to technical issues, we are unable to extract quantitative results from EPro-PnP on our synthetic dataset. However, we can visualize the results qualitatively in Figures 6, 7, 8, and 9.

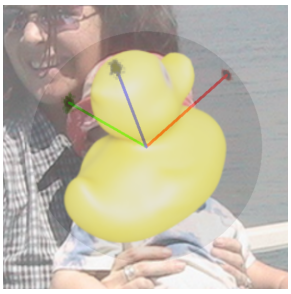


Figure 6. An example image of a synthetic yellow duck overlaid with the estimated pose distribution from EPro-PnP.

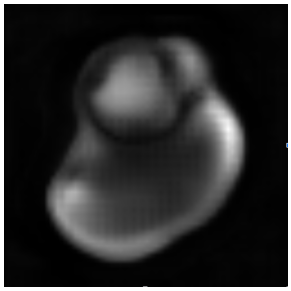


Figure 7. Generated pose visualization from EPro-PnP based off the synthetic duck from Figure 6.

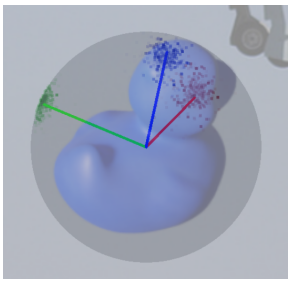


Figure 8. An example image of a synthetic blue duck overlaid with the estimated pose distribution from EPro-PnP.

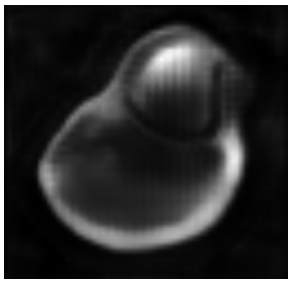


Figure 9. Generated pose visualization from EPro-PnP based off the synthetic duck from Figure 8.

Figures 6 and 8 show a randomly selected synthetic image that was fed into the EPRO-PnP model. Overlaid over these synthetic images of a yellow and blue duck respectively is the corresponding pose distribution estimated by the model. Figures 7 and 9 then show the corresponding pose visualization generated by the model. One noteworthy qualitative result is that the pose visualization from the model closely resembles the pose of the original synthetic image. Likewise, the overlaid pose distribution also closely matches the true pose of the synthetic image. Based on this, it seems highly likely that EPro-PnP is quite effective on generalizing to unseen synthetic data. However, it is also interesting to note that the pose distribution in Figure 8, corresponding to the blue synthetic ducks, has a greater standard deviation than in Figure 6 corresponding to the yellow synthetic ducks. This suggests that EPro-PnP is much less confident in its prediction of the blue synthetic ducks, likely because it did not see blue ducks in its original training data. This result matches Tab. 1, where EPro-PnP suffers a huge performance loss when changing the duck color to blue.

Dataset	Fine-Tuned On (size)	Approach (ADD-S Portion Correct) EfficientPose	DPOD
Real Blue Ducks	Synthetic Blue Ducks (2500)	0.00	**0.47**
Synthetic Yellow Ducks	Synthetic Yellow Ducks (2500)	**0.76**	-
Synthetic Blue Ducks	Synthetic Blue Ducks (2500)	**0.88**	0.83

Table 2. Results of fine-tuning via transfer learning, and evaulating on our custom datasets. EfficientPose benefits greatly from fine-tuning when the fine-tuning dataset closely matches the test set. However, EfficientPose does not generalize between synthetic and real data. DPOD sees no benefit from fine-tuning. Missing value: we do not fine-tune DPOD on yellow ducks due to lack of resources.

From these experiments, it is clear that DPOD generalizes better to unseen data than EfficientPose does, despite EfficientPose achieving higher results on the original Linemod test set. While EfficientPose is able to perform prediction on the real blue ducks to some extent, it fails completely when predicting synthetic images. This indicates that EfficientPose is, in a sense, over-fit to the real images. It also indicates that EfficientPose might be "cheating" by using queues which are consistent throughout the base Linemod dataset, such as light direction and the ground plane, to help predict the object pose. For example, EfficientPose might be finding the orientation of the ground plane, and using that to help predict the duck's orientation, as it is always upright on the ground plane in the original Linemod. When the ground plane is removed, as in the synthetic dataset, EfficientPose fails completely. DPOD does not suffer from this problem, because it is a smaller and more specialized network based on correspondence mapping of object, and does not have the capacity to "cheat" in this same way.

Also benefiting DPOD is the fact that it has seen some synthetic data during training, while EfficientPose has not. We suspect that one of the reasons that EfficientPose was trained entirely on real images is because it does not generalize well between synthetic and real images. This is further supported by our experiments in the following section.

4.1. Fine-tuning

The comparison between EfficientPose and DPOD thus far is slightly unfair, as DPOD as seen some form of synthetic data before (although different than the test set) while EfficientPose has seen only real-world data. Perhaps, if given a chance to learn from synthetic data, EfficientPose would outperform DPOD. To address this, we fine-tune both algorithms on a larger dataset containing data from the same distribution as the test set. We do this fine-tuning twice for each model: once with Synthetic Yellow Ducks, and once with Synthetic Blue Ducks. We test on datasets with matching distributions, as shown in Tab. 2.

For fine-tuning EfficientPose, as we considered the synthetic data we generated to be similar in nature to the original Linemod dataset, we started by freezing the majority of the layers and fine-tuning only the top layers of the model, the classification subnetworks. However, we saw suboptimal results and decided to instead freeze the EfficientNet backbone and retrain the bidirectional feature pyramid network on top of the subnetworks. Since we had limited compute resources, we stopped fine-tuning we trained using this approach until we saw a plateauing of accuracy. This achieved superior results to the first attempt at fine-tuning and are the ones reported.

To fine-tune the DPOD models, transfer learning was used to maintain its original ability of recognizing other classes of objects well. Only the synthetic blue duck dataset was used for transfer learning since it has already performed well on the synthetic yellow duck dataset. Additionally, only the model of the first stage, the correspondence mapping, was further trained since we think the refinement model did not need to change for unseen objects. The first-stage NN model in DPOD was further trained for more 10 epochs with training all the parameters. Only 7.5% of 2500 blue duck images were used for training here considering amount balance.

These experiments show that EfficientPose does benefit greatly from transfer learning. When fine-tuned on data from the same distribution, EfficientPose is actually able to recognize synthetic images. DPOD sees no benefit from fine-tuning, and performance even decreases in some cases. This could be because DPOD is already well-suited to synthetic data before fine-tuning, and performing the fine-tuning simply jostles it around an already good minima. Our theory that EfficientPose cannot generalize between real and synthetic data is also supported by our fine-tuning experiments. After fine-tuning on synthetic data, EfficientPose loses its ability to recognize poses in real data, as shown by its score of 0.00 on Real Blue Ducks.

We believe that part of the reason why EfficientPose does not generalize as well as the PnP-based approaches is because its nature as a convolutionally based, direct prediction model makes it far more attentive to context of the objects, such as color and background, in both the objection and 6D pose prediction phases, meaning that it cannot generalize as well to images with unfamiliar contexts.

5. Conclusion

This work demonstrates that 6D pose recognition algorithms which work well on narrow, homogeneous test sets do not necessarily generalize to data from different distributions, for example with different lighting, color, or background. We find that PnP-based approaches, such as DPOD and EPro-PnP, tend to generalize better than the direct prediction approach of EfficientPose. While EfficientPose may perform the best on leader boards, we suspect that a more generalizable approach, such as DPOD, will perform better in real-world applications.

To avoid over-fitting to test sets, algorithms, especially those which perform direct prediction, should be trained on a wide variety of data. This helps ensure that the algorithm does not learn to "cheat" by making some assumptions about the data which may not hold up in real-world usage. In order to reward algorithms trained to be highly generalizable, it is important that we move away from the homogeneous Linemod test set, which is now 10 years old. A new standard testing dataset, which contains highly diverse scenes and objects should be used in its place.

Our work has a number of limitations. Most importantly, we only evaluate two methods thoroughly. It is possible that evaluating other methods in the same way could break our expectations. It is possible that EfficientPose alone has the issue with overfitting, and other direct prediction models (presently only PoseCNN) may perform better. Similarly, DPOD may be particularly good at generalizing. Future work should investigate more methods. Additionally, we only evaluate performance on a single object model, which is a) not symmetrical, b) homogeneous in texture, and c) relatively small. It is possible that results would be different for different models. Future work should perform the same tests on other models, and extend the tests to cover aberrations in domains other than color, *e.g.* texture, shape, and size.

References

[1] Yannick Bukschat and Marcus Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach, 2020. 1, 2, 3, 4

[2] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation, 2022. 2

[3] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation, 2022. 3

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1, 3

[5] Yan Di, Fabian Manhardt, Gu Wang, Xiangyang Ji, Nassir Navab, and Federico Tombari. So-pose: Exploiting self-occlusion for direct 6d pose estimation. *CoRR*, abs/2108.08367, 2021. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[7] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012. 3

[8] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7677–7686, 2019. 2

[9] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7677–7686, 2019. 3

[10] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. *CoRR*, abs/1908.07433, 2019. 1

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. 2

[12] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. 3

[13] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *CoRR*, abs/1711.00199, 2017. 3

[14] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2017. 3, 5

[15] yashs97. https://github.com/yashs97/dpod, 2020. 2

[16] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019. 2