

# FAST 6DOF POSE ESTIMATION WITH SYNTHETIC TEXTURELESS CAD MODEL FOR MOBILE APPLICATIONS

Bowen Chen      Juhan Bae      Dibyendu Mukherjee

Epson Research

{Bowen.chen, Juhan.Bae, Dibyendu.Mukherjee}@ea.epson.com

## ABSTRACT

Performance of 6DoF pose estimation techniques from RGB/RGB-D images has improved significantly with sophisticated deep learning frameworks. These frameworks require large-scale training data based on real/synthetic RGB/RGB-D information. Difficulty of obtaining adequate training data has limited the scope of these frameworks for ubiquitous application areas. Also, fast pose estimation at inference time often requires high-end GPU(s) that restricts the scope for its application in mobile hardware. To address the requirement of training data adequacy, we propose a novel domain adaptation strategy to train from textureless CAD models with synthetic depth information only, and facilitate inferring poses from RGB images only. To allow faster inference on mobile hardware, we propose two lightweight architectures with a trade-off between ease of training and performance required by different applications. Experiments show comparable performance to the state-of-the-art in the challenging T-LESS dataset, with an inference time of  $\sim 200$  ms using CPU on Google Pixel 2.

**Index Terms**— Object detection, Pose estimation, Synthetic training, Domain Adaptation, Mobile Platform

## 1. INTRODUCTION

Estimating 6DoF (6-Degrees of Freedom) poses for 3D objects from RGB and RGB-D images has been a prominent subject in literature. Such pose estimation has been a key element in robot manipulations, bin-picking, augmented reality applications, and various other challenging scenarios. Hence, there have been a lot of work specializing on 6DoF pose estimation in the recent literature [1–4]. With the advent of deep learning, the accuracy benchmark has been significantly raised, and real-time pose estimation covering a wide view-range has been successfully obtained.

Existing literature on 6DoF pose estimation can be clearly divided into a few distinct groups based on the training and inference modality used, and the inference platform used.

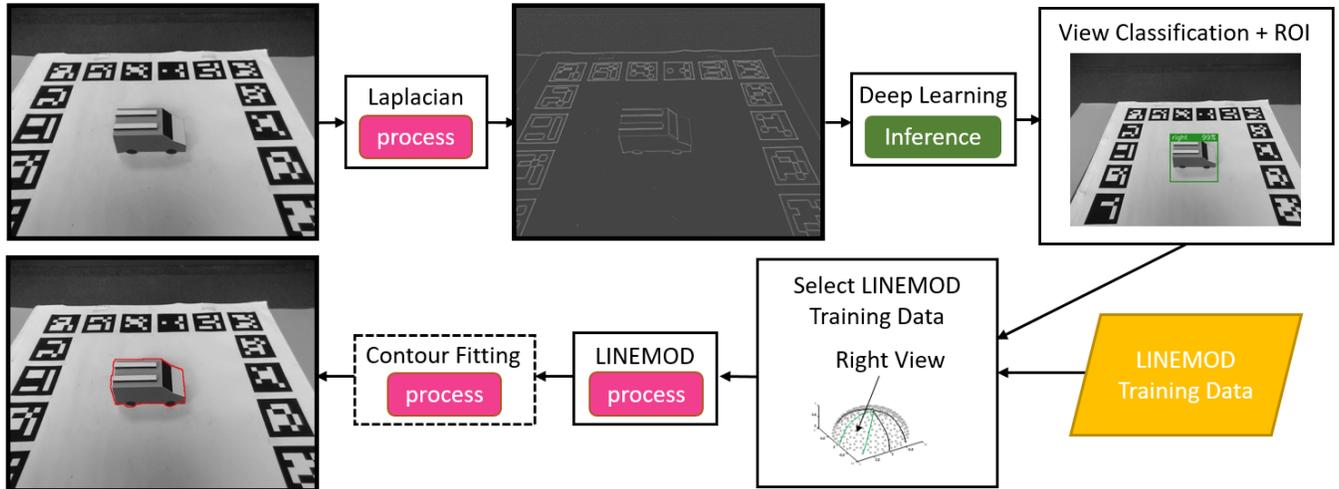
- *Training vs inference modality*: So far, researchers have successfully exploited the following modality combinations for training vs inference: (1) RGB-D vs RGB-D [5, 6], (2) D vs D [7], (3) RGB vs RGB [5, 8]. Inherent problems with RGB-D training ground-truth generation lie in proper registration between RGB and depth, extraction of accurate 6DoF information, and availability of high quality depth data, especially for mobile applications. On the other hand, RGB-based methods demonstrate reduced accuracy [5] or heavy execution time [8]. Generating ground truth RGB data over a wide view-range is also difficult. Although training based on synthetic textures [3, 7, 9] has shown promise, synthetic textures

may also be hard to find, and do not resemble real texture under different lighting conditions, noise, and capturing devices. Domain adaptation between synthetic and real texture [10], RGB and RGB-D data [10, 11] has been a topic of high interest so far. While there have been attempts [12, 13] to train a network with a synthetic CAD model, these methods still rely on real images or depth information. To the best of our knowledge, performance of domain adaptation from *synthetic depth* information in training time to *real RGB* information in inference time is yet to be explored.

- *Inference platform*: A majority of deep-learning based algorithms have shown real-time performance, but on high-end GPUs (e.g. Titan X). Template or feature-based methods, such as LINEMOD, provide faster pose estimation on CPU, but their performance reduce with increasing view-range. Moreover, conventional template based methods suffer from low accuracy in wide view-range due to large number of confusing templates to match and demonstrate poor support for large texture variations from training to inference time.

Based on the above insights, we introduce an efficient framework for fast 6DoF pose estimation on mobile devices without a need for a GPU, and trained strictly using textureless CAD models. We show how proper domain adaptation can generalize to objects with different textures and intensity variations. We build upon previous works on MobileNetV2 [14], Single Shot Multibox Detector (SSD) [15]. However, the work can be generalized to other network architectures supporting the object detection pipelines as well, provided that the architecture can support mobile hardware. We propose two different architectures, based on the accuracy requirements, acceptable interpretability and robustness, and false alarm rates allowed by an application. In the first architecture (VIEWMOD), MobileNetV2 + SSD preprocesses the input image by localizing and classifying object’s view (e.g. front, back). Then, LINEMOD estimates the 6DoF pose of the object. In contrast, the second architecture (BBOX9), proposes a direct regression of a 3D bounding box surrounding the object, followed by a PnP routine [16] to estimate the 6DoF pose.

Our contributions are as follows: (1) designing an effective domain adaptation method to train from textureless CAD models and infer only from RGB domain, (2) enabling inference on a mobile platform, or any low-end computer architecture, (3) building two architectures with relative benefits in accuracy and interpretability. We discuss the architectures, the rationale behind their propositions, and relative benefits later in this work. The rest of the sections are as follows. In section 2, we describe the training procedure and the architectures. In section 3, we conduct experiments for benchmarking. Finally, we conclude on section 4.



**Fig. 1.** Inference process for the VIEWMOD pipeline. Contour fitting is optional in the pose estimation step. The final estimated object’s 6 DoF pose is shown as red re-projected contour on the last image. Note: the BBOX9 pipeline directly provides the bounding box from the “Deep Learning” inference block, and does not contain the LINEMOD block.

## 2. METHODOLOGY

In this section, we describe the synthetic data generation supporting the unique domain adaptation, followed by a summary of the two architectures and their respective rationale.

### 2.1. Synthetic Training Data Generation from 3D models

With uniformly chosen azimuth covering the  $360^\circ$  range, elevation covering a range of  $10^\circ - 70^\circ$  (can be easily extended), and a distance range depending on the application and object to detect, the 3D model is projected to a 2D image on a background randomly selected from PASCAL VOC dataset [17]. The set of 2D images is augmented with motion blur, Gaussian blur, additive Gaussian noise, and random lighting. As texture information between synthetic CAD data and real objects may be different, we propose an effective domain adaptation using two strategies. (1) We wrap the CAD model before projecting to 2D images with random texture selected from DT dataset [18] to simulate possible texture variations, lighting reflections, and shadows. (2) We apply a Laplacian filter on the final 2D image to reduce the gap between synthetic domain and real domain. It enhances the object’s shape information such as contours and inner edges, while reducing the effect of appearance and color information. Without such filtering, the network has no way of knowing which data from the rendered image is would be unavailable during inference (i.e. texture and color information), and can easily overfit to the appearance of the rendered CAD model image. Neural networks are very powerful in learning sophisticated regularities [19], and hence, this domain adaptation technique helps in reducing its power to learn such regularities that represent the domain gap between the synthetic textureless and real images. In addition, multiple random textures make the network more insensitive to spurious texture edges. We show through our experiments in section 3 the effectiveness of this domain adaptation.

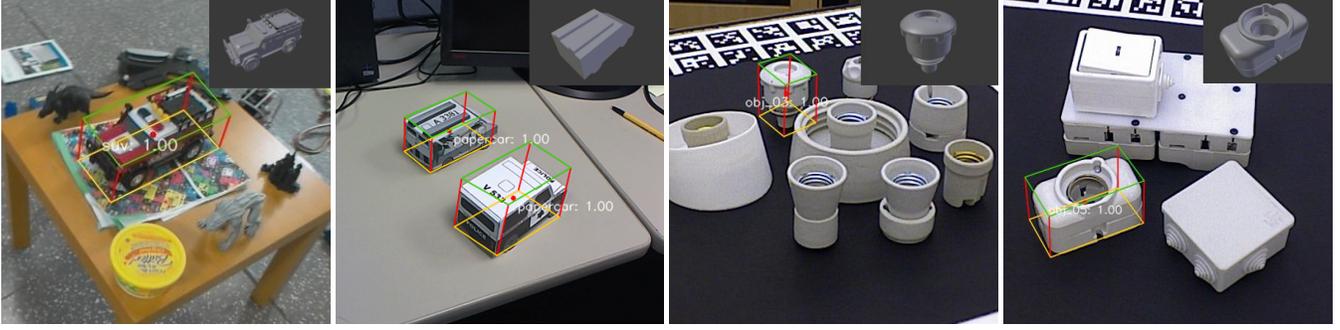
In the next subsections, we explain training processes for the two proposed architectures. Both architectures are built on the same base network, but differ in terms of the goal (view classification vs 3D bounding box regression) and the loss functions used. Each archi-

ture is trained with the synthetic 2D images, as described above. At the inference time, the test image is always Laplacian filtered before processing it through the network.

### 2.2. VIEWMOD

In this architecture, we trained a 2D object detection CNN based on TensorFlow Object Detection API [20], though the scope can be extended to other light-weight models. Conceptually, it is composed of two parts: a MobileNetV2 [14] feature extractor (pre-trained on MS-COCO) to produce feature maps, and a modified SSD multi-box detector to predict view classifications along with 2D bounding box locations. For each anchor box in the SSD head, the model predicts an offset value of the bounding box, and one of the 9 class labels (8 distinct view-classes as described in the next paragraph, and one background class). Note that we modify the original SSD structure having two square anchor boxes with different sizes for each point on the feature map instead of 6 anchor boxes with different aspect ratios and sizes, since the LINEMOD algorithm in the next part of our framework only requires the center point of the bounding box to search for the object in a region-of-interest (ROI) around the center.

The view classification discussed above can be arbitrarily narrow or wide supporting different applications and object types. In this scope, we used only 8 views (front, front-left, left, back-left, back, back-right, right, front-right) each covering about  $45^\circ$  azimuth range. However, we provided a  $15^\circ$  azimuth gap while generating the training data for view-classification to reduce confusion on view-range boundaries. The view-label and ROI from detection are transferred to a LINEMOD based pipeline (pre-trained for the complete view-range with the corresponding rendered CAD model) to select the LINEMOD shape templates corresponding to the view-label returned, and matched in the ROI for the object to detect. Due to this constrained matching, the confusion due to contour based template matching is significantly reduced for objects with simple geometry or in heavy clutter, while the accuracy is improved. The complete pipeline is shown in Fig. 1. As demonstrated, we can optionally postprocess the pose returned by LINEMOD using a contour-fitting technique such as in [21] to yield a more accurate pose.



**Fig. 2.** Pose estimation results of the BBOX9. Green and yellow lines are connections among top and bottom control points respectively, while red dot denotes the centroid of the object. Object’s category label and the confidence score are shown as white text. The CAD model used for training is shown on the top-right corner for reference. From left to right, we show detections under texture variations and texture clutter, object deformations from CAD model, occlusion and clutter, and symmetric objects, respectively.

### 2.3. BBOX9

In this architecture, we extend on [22], and modify the regression header of the SSD to predict the 2D image coordinates of the object’s 3D bounding box directly. We use 9 3D control points to represent the object’s 3D bounding box. During inference, we obtain 2D coordinate predictions of these control points to compute the object’s 6 DoF pose using a PnP algorithm. This pose can be further improved using a similar pose refinement strategy as discussed in Section 2.2.

We choose the 8 corners of the 3D bounding box from the 3D model as the first 8 control points and the centroid of the 3D model as the 9<sup>th</sup> control point. The number of control points may be greater than 9 for different use cases or objects, or can be estimated from the object’s uniqueness manually or automatically by analyzing the CAD model.

For each anchor box, we use the object’s 2D bounding box information to determine whether it contains the object or not. Similar to the original SSD [15], if the intersection over union (IoU) score of anchor box and object’s 2D bounding box is higher than a threshold (0.5 in our case), we assign the ground-truth to that anchor box. In case of all anchor boxes having IoU less than the threshold for some ground-truth objects, we assign each ground-truth object to the anchor box that has the highest IoU.

Suppose  $K$  is the number of objects’ categories we want to classify. Then, for each anchor box, our model predicts  $K$  probabilities of the object’s class and 18 offset values to the anchor box’s center.

Let  $x_{ij}^p = \{1, 0\}$  be an indicator for matching the  $i$ -th anchor box to the  $j$ <sup>th</sup> ground-truth box of category  $p$ . In the matching strategy above, we can have  $\sum_i x_{ij}^p \geq 1$ . We treat matched anchor boxes as positive examples and all others as negative ones. The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$\mathcal{L}(x, c, l, g) = \frac{1}{N} (\mathcal{L}_{\text{conf}}(x, c) + \alpha \mathcal{L}_{\text{loc}}(x, l, g)) \quad (1)$$

where  $N$  is the number of matched anchor boxes. If  $N = 0$ , we set the loss to 0. The localization loss is a Smooth L1 loss [23] between the predicted box ( $l$ ) and the ground-truth box ( $g$ ). We regress to offsets for the center ( $c_x, c_y$ ) of the default bounding box ( $d$ ).

In the following equation,  $g_j^m$  represents the  $m$ <sup>th</sup> value of the  $j$ <sup>th</sup> ground-truth box;  $l_j^m$  represents the  $m$ <sup>th</sup> value of the  $j$ <sup>th</sup> predicted box and  $d_i^w$  is the width of the  $i$ <sup>th</sup> default bounding box.

$$\mathcal{L}_{\text{loc}}(x, l, g) = \sum_{i \in \text{Pos}} \sum_{m=1}^{18} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m) \quad (2)$$

$$\hat{g}_j^m = \begin{cases} (g_j^m - d_i^{c_x})/d_i^w, & \text{if } m \text{ is odd} \\ (g_j^m - d_i^{c_y})/d_i^w, & \text{otherwise} \end{cases}$$

The confidence loss, similar to [15], is the softmax loss over multiple classes confidences.

The complete pipeline for BBOX9 is a reduced version of VIEWMOD, as already pointed out in Fig. 1. Since we can directly infer the pose from this pipeline, we can also refine this pose using an optional contour-fitting strategy. The unrefined output poses from BBOX9 for a number of objects are shown in Fig. 2. Due to the domain adaptation, the pipeline supports objects with different texture variations even if they are trained using a single CAD model. This is shown in the first and second images from the left, where the CAD models (top-right corner) do not contain any texture information. Also, BBOX9 considers the 9 3D points independently, allowing certain deformations in their placements. This facilitates scope for detections using approximate CAD models as shown in the second image from the left with two cars. The car on the right has some deformations from the CAD model, but can still be supported by this pipeline. Fig. 2 also shows that the pipeline supports high texture clutter, object clutter, and occlusion.

### 2.4. Discussion on Two Architectures

In sections 2.2 and 2.3, we described a view classification based architecture and an end-to-end architecture respectively. A view-classification based deep learning architecture is a contrast to the popular trend of the end-to-end architectures. Apart from the reason to improve accuracy through the use of accurate shape templates from LINEMOD, there are two important reasons behind such an architecture. (1) End-to-end architectures often act like blackboxes. They are difficult to interpret and hence, their failures cannot be explained easily. In contrast, view-classification and its failure is easier to understand, especially when an object has symmetry or simple geometry. Following this, a shape template matching is also easy to explain. Hence, our first architecture increases interpretability. (2) Two independently trained pipelines (view classifier + LINEMOD) increase the precision. If a false candidate is passed through view-classification, it is less likely to pass through the LINEMOD pipeline

**Table 1.** Comparison between BB8 [8] and our methods on T-LESS [24] in terms of recall. Note that BB8 uses real RGB images while VIEWMOD and BBOX9 only use synthetic textureless CAD models.

Scene ID: [Obj. IDs]	BB8 (real training) [8]	VIEWMOD (textureless training)		BBOX9 (textureless training)	
	>10% visibility	>10% visibility	>70% visibility	>10% visibility	>70% visibility
1: [2, 30]	50.8, 55.4	64.1, 66.0	<b>71.3, 75.8</b>	44.0, 35.8	48.9, 40.7
2: [5, 6]	56.5, 55.6	81.0, 55.0	<b>90.7, 62.0</b>	75.4, 60.1	84.4, <b>67.8</b>
4: [5, 26, 28]	68.7, <b>53.3</b> , 40.6	68.0, 46.0, 56.7	<b>80.7</b> , 46.0, <b>64.2</b>	65.6, 37.7, 35.1	78.9, 37.7, 39.8
5: [1, 10, 27]	<b>39.6</b> , 69.9, 50.1	20.8, 69.7, 50.8	21.6, <b>77.6</b> , <b>56.7</b>	18.7, 56.7, 24.0	19.3, 63.3, 28.9
7: [1, 3, 13]	42.0, 61.7, <b>64.5</b>	42.5, 64.1, 18.5	<b>47.4</b> , 70.4, 21.3	41.5, 64.7, 12.4	46.2, <b>71.0</b> , 14.3
7: [14, 15]	<b>40.7</b> , <b>39.7</b>	34.1, 17.1	37.7, 20.6	28.2, 17.1	31.1, 20.6
7: [16, 17, 18]	<b>45.7</b> , 50.2, 83.7	33.1, 64.3, 76.7	38.5, <b>75.4</b> , <b>86.4</b>	21.0, 33.9, 71.8	24.4, 39.1, 81.9
<b>Average</b>	55.3	51.6	<b>58.0</b>	41.3	46.6

since the features used by the two pipelines are different. On the other hand, an end-to-end pipeline is preferred in a situation where a compact framework is appreciated. It lowers the burden of training separate pipelines. Additionally, approximate CAD models can be supported in this pipeline as shown in Fig. 2. This allows to train a network with approximate CAD models generated using multiple-view 3D reconstruction, or objects having flexible parts that are visually different from the CAD models. Of course, the amount of deformation supported depends on the shape of the object used.

### 3. EXPERIMENTS

#### 3.1. Evaluation Metric

We adopt the standard evaluation protocol and metric for recall: 6DoF Pose [7]. Based on this metric, if the average of 3D distances between the estimated pose and ground-truth is less than 10% of the object’s diameter, it is a correct detection. Symmetric objects are evaluated in a different manner as described in [7].

To empirically evaluate the proposed framework, we choose the T-LESS dataset [24] as it has been shown as one of the most challenging datasets in the literature. The survey paper [25] discusses the performance of the best state-of-the-art methods on T-LESS. From [25], it is evident that the best methods (e.g. [26]) for T-LESS are based on point-pair features that require depth information during inference and have very slow execution time (the fastest methods have execution time over 2s per image). We compare our methods to BB8 [8] as it is one of the best RGB-based inference method on the T-LESS datasets. However, the results for BB8 are evaluated with a visibility constraint of 10%. In real-world scenarios, specifically for applications where background information change drastically, such low visibility constraint can largely increase the false-alarm rate, and hence, reduce the precision of the algorithm. Hence, while we show the results evaluated with 10% visibility constraint, we also evaluate using 70% visibility. A high visibility constraint may be more suitable in real-world applications.

#### 3.2. Evaluation on T-LESS Dataset

While T-LESS is an RGB-D dataset for 6D pose estimation of texture-less objects, we only used CAD models for training and RGB images for testing. T-LESS dataset has very challenging test sequences with a high amount of clutter, occlusion and texture-less objects exhibiting symmetries and mutual similarities in shape and/or size. For comparison, we consider Scenes #1, #2, #4, #5, and #7 in our experiments similar to BB8.

Even though, our results are not directly comparable to BB8 because we do not use real images in training, we are still able to

achieve the state-of-the-art result as shown in Table 1. With a minimum 10% visibility (of the object surface in the ground truth pose), our results are close to BB8 but not as good. The main reason is: since the inference is highly dependent on edges and contours on the object, such low amount of visibility may occlude important edges crucial for a valid pose estimation and hence, affect the recall rate. However, this is a limitation largely dependent on an object’s shape complexity. Since our training effectively brings out shape features, any uniqueness of an object is exploited to provide the best pose estimate. Hence, the evaluation results are close to that of BB8. With 70% visibility, performance of both VIEWMOD and BBOX9 significantly improves. In addition, our methods have over 84% precision on average with 70% visibility.

#### 3.3. Mobile Inference

To evaluate the applicability of VIEWMOD and BBOX9 on mobile devices, we use TensorFlow Object Detection API [20] on Google Pixel 2. All models are implemented in device CPU. VIEWMOD takes 200ms per frame (150ms for view-classification network, 50ms for LINEMOD). Inference time for BBOX9 is similar to that of VIEWMOD since the regression takes a bit more time in comparison to view-classification. In comparison, BB8 [8] runs over 291ms per image on a GeForce TITAN X. Note that the execution time will vary with a change in the neural network architecture, and can be further improved by weight pruning algorithms.

### 4. CONCLUSION

We introduced an efficient and user-friendly 6DoF pose estimation framework for mobile applications. The framework introduces an effective domain adaptation strategy to use synthetic textureless CAD models for training and facilitate inference from real RGB data. The unique strategy adapts to a variety of CAD models irrespective of the texture information they may have during inference. We also propose two different architectures: one integrating neural networks with LINEMOD to achieve fast and accurate inference with improved interpretability to detect failures and debug the pipeline (VIEWMOD), while the other as an end-to-end architecture facilitating single training process and support for approximate CAD models (BBOX9). Even with the constraint imposed on the training data and the use of lightweight networks to support mobile hardware, performance on T-LESS datasets show comparable results to state-of-the-art. The two architectures proposed in the work open up a large scope for ubiquitous 3D object pose estimation framework. One future direction is to improve BBOX9 for fine pose level accuracy, in low-end hardware.

## 5. REFERENCES

- [1] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.
- [2] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian Reid, "Deep6dpose: Recovering 6d object pose from a single rgb image," *arXiv preprint arXiv:1802.10367*, 2018.
- [3] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," .
- [4] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox, "Deepim: Deep iterative matching for 6d pose estimation," *arXiv preprint arXiv:1804.00175*, 2018.
- [5] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al., "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3364–3372.
- [6] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *European Conference on Computer Vision*. Springer, 2014, pp. 462–477.
- [7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.
- [8] Mahdi Rad and Vincent Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *International Conference on Computer Vision*, 2017, vol. 1, p. 5.
- [9] Jason Rambach, Chengbiao Deng, Alain Pagani, and Didier Stricker, "Learning 6dof object poses from synthetic single channel image," in *Proceedings of the 17th IEEE International Symposium on Mixed and Augmented Reality*.
- [10] Mahdi Rad, Markus Oberweger, and Vincent Lepetit, "Feature mapping for learning fast and accurate 3d pose inference from synthetic images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4663–4672.
- [11] Mahdi Rad, Markus Oberweger, and Vincent Lepetit, "Domain transfer for 3d pose estimation from color images without manual annotations," *arXiv preprint arXiv:1810.03707*, 2018.
- [12] Manuel Brucker, Maximilian Durner, Zoltan-Csaba Marton, Ferenc Balint-Benczedi, Martin Sundermeyer, and Rudolph Triebel, "6dof pose estimation for industrial manipulation based on synthetic data," in *International Symposium on Experimental Robotics (ISER)*. IFRR, 2018, accepted for publication.
- [13] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.
- [14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [16] Long Quan and Zhongdan Lan, "Linear n-point camera pose determination," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 8, pp. 774–780, 1999.
- [17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [18] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [19] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.
- [20] Object Detection API, "Speed/accuracy trade-offs for modern convolutional object detectors," 2017.
- [21] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba, "Parsing ikea objects: Fine pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2992–2999.
- [22] Bugra Tekin, Sudipta N Sinha, and Pascal Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.
- [23] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [24] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.
- [25] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al., "Bop: benchmark for 6d object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.
- [26] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. Ieee, 2010, pp. 998–1005.