Pose Estimation from RGB Images of Highly Symmetric Objects using a Novel Multi-Pose Loss and Differential Rendering

Bengtson, Stefan Hein; Åström, Hampus; Moeslund, Thomas B.; Topp, Elin A.; Krüger, Volker

Published in:
2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

DOI (link to publication from Publisher):
10.1109/IROS51168.2021.9636839

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

Pose Estimation from RGB Images of Highly Symmetric Objects using a Novel Multi-Pose Loss and Differential Rendering

Stefan Hein Bengtson*[1] and Hampus Åström*[2], Thomas B. Moeslund[1], Elin A. Topp[2], Volker Krueger[2]

***Abstract*— We propose a novel multi-pose loss function to train a neural network for 6D pose estimation, using synthetic data and evaluating it on real images. Our loss is inspired by the VSD (Visible Surface Discrepancy) metric and relies on a differentiable renderer and CAD models. This novel multi-pose approach produces multiple weighted pose estimates to avoid getting stuck in local minima. Our method resolves pose ambiguities without using predefined symmetries. It is trained only on synthetic data. We test on real-world RGB images from the T-LESS dataset, containing highly symmetric objects common in industrial settings. We show that our solution can be used to replace the codebook in a state-of-the-art approach. So far, the codebook approach has had the shortest inference time in the field. Our approach reduces inference time further while a) avoiding discretization, b) requiring a much smaller memory footprint and c) improving pose recall.**[3]

I. Introduction

As robotics moves towards flexible and autonomous solutions, computer vision is gradually playing a bigger role in robotic solutions, especially for 6D pose estimation. This topic has actively been researched in the robotics community [1] for many years, as the pose of an object is very useful when figuring out how to interact with it. Pose estimation is useful in other areas as well, e.g. in augmented reality.

However, it is still a challenging problem, and pose estimation has hence been the focus of many public datasets and challenges issued by the community. One challenging aspect of pose estimation is the symmetry of objects, as it complicates both the process of labeling the data and constructing methods that can adequately deal with these ambiguities in the object pose. The T-LESS dataset [2] is an industry benchmark for this problem, featuring 30 industry-like objects with multiple symmetries, examples shown in Fig. 1. Estimating poses from the T-LESS dataset is also more challenging due to the lack of distinguishable features in the textures of the objects, which could otherwise help solve pose ambiguities caused by symmetries.

In this paper we propose an adaptation of the 6D pose estimation approach in [3], [4], that relies on an autoencoder for feature extraction in a codebook-based approach. By replacing their codebook with a neural network and utilizing differential rendering [5], we provide a solution that has a significantly smaller memory footprint, is faster at inference and has improved pose recall when tested on the T-LESS dataset. The solution we propose does not require discretizing poses and it is therefore more easily extendable. Like [3], [4] our method is trained on synthetic RGB images rendered from CAD models or reconstructions and requires no labelled data or predefined symmetries.

Fig. 1: Examples from the T-LESS dataset [2]. From left to right: object 2, 30, 5 and 10. The two left-most objects exhibit continuous semi-symmetries where the two others include discrete semi-symmetries.

Our solution retains many of the properties making [4] interesting for use in robotics. Low inference time allows real-time execution. Only requiring RGB images imposes less restrictions on the hardware. Training on synthetic images makes the process of operating on new objects automatic, with no need for manual labor.

The main contributions of this paper are:

- We propose a depth-based loss function which inherently handles object symmetries without using predefined global symmetries. Our loss does not require a depth sensor as we leverage a differentiable renderer to produce depth maps from CAD models.
- We demonstrate that a pose regression network can be trained to do pose estimation of objects in RGB images using this new loss. We show that this network can replace the codebook used in [4], thereby avoiding discretization.
- We introduce a scheme where the network outputs multiple pose estimates and a weighting between them, and we show that this increases pose recall.
- We show that our pose regression network consumes orders of magnitude less memory, results in faster inference and improves pose estimation recall.

II. Related Work

Traditional methods for pose estimation have commonly relied on matching features, edges and templates [6], [7]. Other approaches use iterative search to find the pose of an object, such as the widely used ICP (Iterative Closest Point) algorithm. Due to their iterative nature, ICP and similar methods are slow, unless optimized for speed [8].

*Authors with equal contribution
[1]Visual Analysis and Perception (VAP) Laboratory, Aalborg University
[2]Robotics and Semantic Systems, Department of Computer Science, Lund University
[3]Code can be found here:
https://github.com/shbe-aau/multi-pose-estimation

Lately, many of these methods have started to be replaced or complemented by machine learning methods [1], [9], [10], [11], [12]. Supervised machine learning relies on ground truth labels to estimate performance and generate the loss that drives the learning. This makes the way in which that loss is affected by visual traits, like symmetries, an important part of any method for pose estimation.

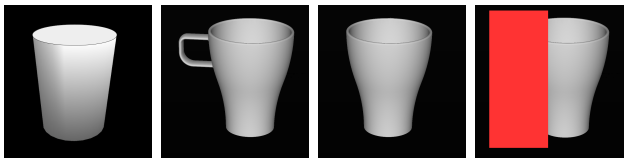
An object that is symmetrical or semi-symmetrical has many poses that are similar. Such poses should often be treated equally. That means that each input image might match many poses and should not be punished for predicting one of the symmetries rather than the real ground truth pose. For example, a cylinder rotated around its major axis should be treated identically independent of angle, as shown in Fig. 2a. This problem is especially important for learning algorithms, as they need a consistent way to evaluate if a proposed pose is good or bad.

There are however some types of apparent symmetry that are more complex than others. When parts of an object are occluded, either by external objects or by other parts of the object itself, i.e. self-occlusion, an object can appear identically for many different poses even if it is not actually symmetrical, such as the example in Fig. 2. Any method that wishes to use learning to estimate poses for these objects need to address how to resolve these ambiguities as well.

Machine learning solutions for 6D pose estimation come in many varieties. Some methods focus on comparing the pose they produce directly with the target pose and predefined global symmetries for each object [13], while other methods utilize 2D or 3D comparisons as loss metrics to train neural networks [14], [3], [4]. In the latter cases, objects with symmetries and semi-symmetries automatically avoid being penalized for miss-classifying along those symmetries. 2D image comparison methods that only consider visible parts of the object can handle apparent symmetries that arise from self-occlusions [3], [4].

Machine learning needs a large amount of data on which to train. For pose estimation, accurate labeling of that data is difficult and costly. To alleviate this problem synthetic data can be produced, for instance by rendering CAD models of target objects together with real images and domain randomization [3], [15].

The work in this paper is based on [3], [4] and relies on a similar synthetic data regime for training. Their work is based on training one or several autoencoders for the objects one wishes to estimate the pose of. The latent space vector produced by the encoder is in their work compared to a codebook of reference latent space vectors, the closest of which becomes the initial pose estimate. When higher accuracy is required, the estimate can be improved by producing extra temporary codebook entries of poses similar to the initial estimate.



(a) (b) (c) (d)

Fig. 2: (a) Rotationally symmetric objects should be treated equally independent of angle around its major axis. Examples of how symmetries can occur for a mug with a handle. (b) Handle visible, no pose ambiguity. (c) Self-occluded due to a slight rotation and (d) occluded by another object, both of these have ambiguities in pose.

While a codebook is in general a good solution, it has a large memory footprint, and it requires a discretization of the predictions. By replacing the codebook in [4] with a pose regression neural network and utilizing differential rendering [5] we show that these problems can be alleviated while simultaneously improving performance.

III. Method

The novel method for 6D pose estimation proposed in this paper is based on the approach initially proposed by [3], [4]. This method retains the central element of the autoencoder, but a neural network replaces the example-based codebook approach for pose regression, as shown in Fig. 3. The encoder is a feature extractor, producing a 128-dimensional latent vector from an input image of an object. The latent vector is provided as input for the pose regression network. We use the encoder provided by [4]. During training, synthetic images are rendered based on CAD models or reconstructions, with backgrounds and augmentation in accordance with domain randomization [15]. This allows the system to be trained on new objects without manual data collection.

The benefits of replacing the codebook with a neural network are to provide a continuous pose space instead of having to discretize it into a codebook while reducing the memory consumption. Furthermore, it should be more easily extendable than the codebook. The size of a codebook grows exponentially with the number of degrees of freedom, while a pose estimation network only needs to add three more output parameters to expand a rotational representation to include e.g. translations in 3D. Our approach of using a neural network instead of a codebook can be considered a more general and scalable solution as it does not suffer from these limitations.

The pose regression network is structured loosely on the network proposed by [16]. Their network is designed to estimate the pose of an object from a feature vector produced from a pre-trained CNN. Their setup is thus similar to how our pose regression network estimates poses from the output of a pre-trained encoder. Our network consists of seven fully connected layers. To maintain training performance with a deeper network, skip-connections between the first three fully connected layers are added.

Note that our network, shown in Fig. 3, produces not only one but multiple pose estimates along with a confidence for each, which is described in more detail in Sec. III-B. The network outputs the pose in terms of the representation proposed by [17], as it performs better than regression directly on e.g. rotation matrices or quaternions. This pose

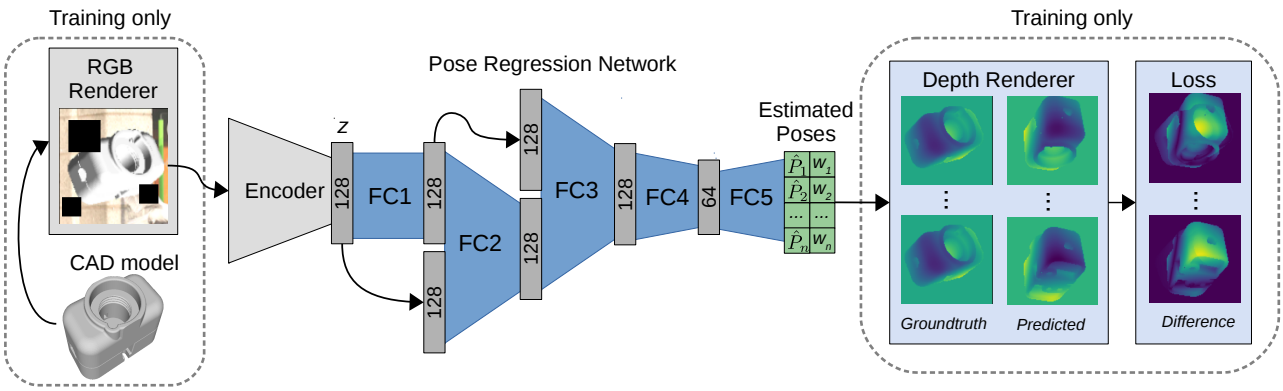


Fig. 3: Overview of the proposed pose estimation pipeline. During training, synthetic data are continuously generated by rendering RGB images of the relevant object from a CAD model. These RGB images are augmented and fed into the encoder part [4], resulting in a latent vector z. This latent vector z is fed into a pose regression network, consisting of five fully-connected layers, which outputs n poses $\hat{P}_1, \hat{P}_2, ..., \hat{P}_n$ and associated confidences $w_1, w_2, ...w_n$. These confidences are normalized with the softmax function. Depth maps are generated from the poses using a differentiable depth renderer in order to produce the final loss. During inference, both the RGB renderer and the differentiable depth renderer are omitted. We use a pre-trained encoder provided by [4] as the first part of the pipeline.

representation consists of two vectors in $\mathbb{R}^3$, i.e. 6 parameters in total, which are converted into an orthonormal basis. A 3×3 rotation matrix can then be formed using this basis as column vectors. Using this method also ensures that the resulting rotation matrix is orthogonal.

During training, each predicted pose is used to render depth images using CAD models and a differentiable renderer [5]. Those images are in turn used in the network's loss function. The motivation for this depth-based loss function and how it works are described in the following section.

A. Single-Pose Depth Loss

The depth-based loss function used in our pose regression network is heavily inspired by the VSD (Visible Surface Discrepancy) error metric proposed by [18], [19]. The VSD metric is defined as:

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

where $\hat{S}$ and $\bar{S}$ are depth maps (called distance maps in [18], [19]) based on the estimated pose $\hat{P}$ and the ground-truth pose $\bar{P}$ respectively. Both poses have an associated visibility mask, $\hat{V}$ and $\bar{V}$, that contains the set of pixels actually visible in the given test image I. These visibility masks are found using the real depth map S_I for each test image I. It includes all objects in the scene and can thus be used to determine how those objects occlude each other. The union of these two visibility masks makes up the set of pixels p that are considered by the VSD metric and ensures that only visible pixels are considered. Lastly, τ defines a threshold for the tolerance when comparing the distance maps $\hat{S}$ and $\bar{S}$.

One of the main benefits of the VSD metric, and why it is often used in pose estimation benchmarks, is how it inherently can cope with symmetric objects, as it is solely based on the appearance of the object. I.e., the estimated pose and the ground truth pose may be off by 180^o but the VSD metric would still report a low error for symmetric objects where such an error will not be visible. The VSD metric is able to deal with global symmetries, such as discrete and continuous symmetries, but it is also able to cope with symmetries caused by self-occlusion of the object. Furthermore, the inclusion of the visibility masks, $\hat{V}$ and $\bar{V}$, makes it robust to symmetries caused by occlusion from other objects in the scene.

The loss function proposed in this paper relies on comparing depth maps. In an effort to achieve the same benefits as the VSD metric in Eq. 1 each pose is evaluated by the loss function:

$$L_{\text{single}}(\hat{S}, \bar{S}) = \underset{p \in \hat{V} \cup \bar{V}}{\text{avg}} \left(\frac{\min(\delta, |\hat{S}(p) - \bar{S}(p)|)}{\delta} \right) \tag{2}$$

The value δ serves as a threshold such that there is an upper limit of how much each individual pixel in the distance maps can contribute to the final loss. Without this threshold the loss would be dominated by pixels where the object is present in one of the distance maps but not in the other. This is similar to comparing silhouettes and is hence not ideal as any distance/depth discrepancies on the object itself are dominated by the silhouette.

In order to train the pose regression network with back-propagation, this equation needs to be differentiable with respect to $\hat{P}$ and thereby $\hat{S}$. Regular renderers do not produce differentiable output, but some differentiable renderers have recently become available [5].

However, one major difference from the VSD metric is how our loss is continuous instead of being limited to the binary set $\{0, 1\}$ as is the case of Eq. 1. This is necessary as the binary set is essentially a step function and thus not differentiable.

B. Multi-Pose Depth Loss

The loss function described in Eq. 2 introduces many local minima, as can be seen in Fig. 4. These new minima are

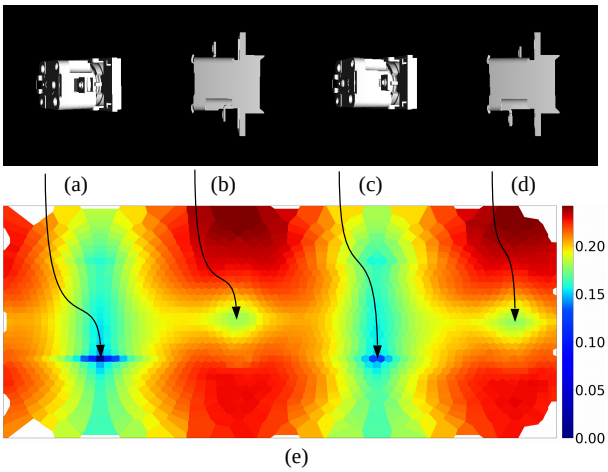


Fig. 4: Visualization of L_{single} for different pose estimates in relation to specific ground truth pose for object 10. Rotations in the image plane are omitted to get a 2D visualisation. The global minimum (ground truth) pose is (a), and its 180 deg semi-symmetry is (c). The two most isolated non-symmetry local minima are given by (b) and (d). The loss landscape is visualized in (e), ignoring in-plane rotations.

problematic, as the training of the network easily gets stuck in them, owing to the output being constrained to the SO(3) space. Common training methods only consider the local environment and can therefore not easily overcome these issues. By extending the network to output multiple pose estimates, along with a confidence associated to each, this limitation can be circumvented. The final prediction for a given input is the pose with the highest confidence. As the output confidence distribution changes, the estimated pose can change drastically, if the set of predicted poses are spread over the output space. The benefits of multiple pose estimates are explored in the ablation study in Sec. IV-B.

To bring these concepts together we propose a new loss function L, that we call "multi-pose loss". It expands on Eq. 2, and is defined as follows:

$$L(\hat{\boldsymbol{S}}, \bar{S}, \hat{\boldsymbol{P}}) = L_{\text{pose}}(\hat{\boldsymbol{P}}) + \sum_{i=1}^{n} L_{\text{single}}(\hat{S}_i, \bar{S}) \cdot (\gamma + w_i) \quad (3)$$

where w_i is the confidence associated with the i'th estimated pose $\hat{P}_i$ for the set of poses $\hat{\boldsymbol{P}}$, $\hat{\boldsymbol{S}}$ the set of depth maps associated with those poses and n is the number of poses output by the network. The confidences, $w_1, w_2, ...w_n$, predicted by the network are normalized using the softmax function. The γ parameter ensures that pose estimates with a near zero confidence also contribute to the loss. This is necessary to make sure that the network improves all pose estimates.

The L_{pose} term in Eq. 3 forces the network to spread its pose estimates by penalizing poses that are too similar. This loss is defined as follows:

$$L_{\text{pose}}(\hat{\boldsymbol{P}}) = \frac{\sum_{R_A \in \hat{\boldsymbol{P}}} \left(\sum_{R_B \in \hat{\boldsymbol{P}}} \Delta(R_A, R_B) \right)}{n^2} \quad (4)$$

where R_A and R_B are rotation matrices converted from the 6D pose representation, and $\hat{\boldsymbol{P}}$ is the set of predicted poses from the network. The function $\Delta(R_A, R_B)$ is a measure of similarity between the two rotation matrices as shown in Eq. 5.

$$\Delta(R_A, R_B) = 1 - \frac{\min(\phi, \theta)}{\phi} \quad (5)$$

with

$$\theta = \arccos\left(\frac{Tr(R_B R_A^T) - 1}{2}\right) \quad (6)$$

where $R_B R_A^T$ is the rotation matrix needed to transform R_A to R_B, and $Tr(...)$ is the trace of this matrix. This similarity measure is essentially a conversion of the rotation matrix $R_B R_A^T$ into its corresponding axis-angle representation, while ignoring the axis of rotation. The threshold ϕ serves as a boundary, with rotation matrix pairs that differ by ϕ or more not contributing any loss, while loss is maximized for rotation matrices that are identical. It ensures that the pose regression network has some leeway when predicting the multiple poses instead of just spreading them uniformly, while punishing poses that are close to one another.

C. Training

Our method is trained solely on synthetic data. Objects from the T-LESS dataset [2] are rendered in different poses, randomly sampled in SO(3) based on a uniform sampling of quaternions as done by [3]. CAD models of the objects from the dataset are rendered using OpenGL and the resulting images are augmented in a similar way to [3].

A shared encoder is used for all objects as proposed by [4] and we use the publicly available pre-trained encoder they supply[1]. It is trained on 3D reconstructions of object 1-18 from the T-LESS dataset.

A separate pose regression network is trained for each individual object using a depth max of $\delta = 30$mm and a pose similarity threshold of $\phi = 0.7$ radians (i.e. $\approx 40^o$). We render $n = 10$ poses, per input image and the minimum loss weight for each pose is $\gamma = 0.01$. Each pose regression network is trained for 200 epochs of 10.000 samples each, using a learning rate cycle [20] between 0.005 and 0.0005. All these training parameters are selected through trial-and-error.

It should be noted that all weights in the pre-trained encoder are frozen when training our pose regression networks. This is done to ensure that any differences in performance are directly linked to replacing the codebook-based approach by [4] with our pose regression network, rather than additional training of the encoder.

IV. Evaluation

In the following we evaluate our approach against the one proposed in [4]. We test against the publicly available codebooks and pre-trained encoder from [4], the same encoder used in our solution. The methods are compared on their ability to predict correct poses and performance in terms of memory consumption and inference time.

[1] github.com/DLR-RM/AugmentedAutoencoder/tree/multipath

The pose prediction performance of our method is evaluated on real-world images from the T-LESS dataset [2] using the scripts provided as part of the BOP benchmark [21]. We report the recall of each object averaged across different thresholds for the VSD metric e_{VSD} (defined in Eq. 1) and different tolerance thresholds τ.

In this paper we focus on the correctness of the estimated rotation of the object pose. Errors related to the translation estimate are hence ignored by using the ground truth translation at all times during evaluation for all approaches. Furthermore, the ground truth bounding boxes are used to make the results independent of any errors introduced by an object detector.

A. Pose Estimation Performance

In terms of pose recall, the results in Table I show that our method outperforms [4] on average. Our approach has a higher performance when evaluating on object 1-18 in comparison to object 19-30. This is expected as the pre-trained encoder is only trained on object 1-18. The approach by [4] suffers similarly and to a greater extent than our method, as shown in Table I.

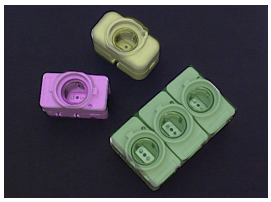
In Fig. 5a an example of our predictions are superimposed on a test image. Here, objects 5 (yellow), 6 (magenta) and 7 (green) all match the target well, even though object 7 has a bad overall recall when compared to the two other objects, as seen in Table I. This discrepancy could be explained by instances as the one found in Fig. 5b where the pose prediction for object 7 failed, likely due to the partial occlusion by the two objects in front of it.

Occlusion is of course a challenging scenario in general for pose estimation, but our approach appears to be able to handle it well for some objects. An example of such is the cylinder-shaped objects in Fig. 6b, where the predicted poses appear correct even for heavily occluded objects.

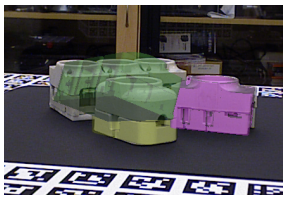
In general, our approach performs better on cylinder-shaped objects with continuous symmetries. This is exemplified in Fig. 6a where the pose predictions for the box-shaped objects do not appear to fit as well with the test image as the cylinder-shaped objects.

The observation that our approach does better on cylinder-shaped objects is further supported by Table II, showing the average recall when dividing the T-LESS test dataset into objects with continuous symmetries and objects without. For objects with continuous symmetries, i.e. cylinder-shaped objects, our method outperforms [4] by a clear margin. For non-cylindrical objects there is little difference in performance between the methods.

The performance discrepancy between objects with contin-

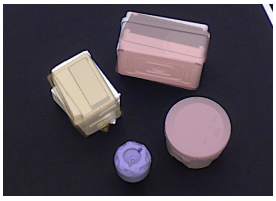


(a) Scene 2, image 170.



(b) Scene 2, image 490.

Fig. 5: Colorized renditions of pose predictions superimposed onto images from the T-LESS test dataset. In (a) the poses for objects 5 (yellow), 6 (magenta) and 7 (green) all fit well, but in (b) the pose of object 7 is severely wrong. This is probably due to occlusion.



(a) Scene 1, image 190.

(b) Scene 20, image 10.

Fig. 6: Colorized renditions of pose predictions superimposed onto images from the T-LESS test dataset. The pose predictions for the cylindrical objects in (a) are better than those for the rectangular objects. In (b) cylinder-shaped objects are well predicted, even though it is a complicated scene with a lot of occlusion.

TABLE I: Average VSD recall for the T-LESS primesense test dataset, for our solution and the codebook-based solution [4]. The table is split in two parts; object 1-18 for which the encoder was trained and object 19-30 not seen before by the encoder. Finally, the average VSD recall across all objects is listed in the lower right corner. Results from our method are shown by the mean and standard deviation from three experiments.

Obj.	Codebook	Ours	Obj.	Codebook	Ours
01	37.82	**51.84** ± 2.8	19	51.19	**54.15** ± 1.7
02	51.88	**63.74** ± 1.8	20	**40.71**	35.96 ± 1.6
03	62.87	**71.53** ± 3.3	21	43.25	**43.31** ± 1.4
04	56.00	**62.66** ± 3.5	22	**38.15**	32.03 ± 0.5
05	77.18	**80.82** ± 0.3	23	39.18	**56.68** ± 1.1
06	**68.04**	66.71 ± 4.6	24	58.97	**61.93** ± 3.3
07	65.18	**65.68** ± 4.9	25	**69.86**	63.08 ± 1.6
08	**63.11**	61.21 ± 0.8	26	57.94	**58.87** ± 2.3
09	**68.96**	55.66 ± 0.5	27	68.09	**77.62** ± 1.2
10	**58.55**	54.14 ± 2.0	28	68.06	**73.33** ± 1.3
11	**52.15**	51.48 ± 2.4	29	76.43	**80.67** ± 0.7
12	**62.19**	56.58 ± 1.6	30	77.81	**83.41** ± 2.1
13	63.56	**64.21** ± 5.0	mean	57.47	**60.09** ± 0.4
14	57.29	**63.01** ± 1.2			
15	64.91	**66.37** ± 3.8			
16	**75.82**	73.16 ± 2.7			
17	76.62	**77.72** ± 0.9			
18	**71.26**	62.71 ± 2.0	All	Codebook	Ours
mean	62.97	**63.85** ± 1.2	mean	60.77	**62.34** ± 0.9

TABLE II: Average VSD recall for the T-LESS primesense test dataset divided into objects with continuous symmetries and objects with discrete symmetries. Both our and the codebook-based approach performs better on objects with continuous symmetries. However, the difference in performance between continuous and discrete symmetries is more pronounced for our method. Our results are shown with mean and standard deviation as in the previous table.

	Codebook [4]	Ours
Continuous symmetries	62.14	**67.23** ± 2.7
Discrete symmetries	**59.97**	59.51 ± 0.3

uous symmetries and those with discrete symmetries could be because cylinder-shaped objects are easier than other objects to estimate the pose for. This is sensible, as objects with a continuous symmetry around an axis essentially ignore any rotation around that axis. The number of degrees of freedom in the pose estimation problem are thus less for objects with continuous symmetries. This pattern of higher performance for cylindrical objects is also present in our baseline experiments using the method in [4], but to a lesser extent. However, for our proposed approach, the difference in performance between objects with and without continuous symmetries is much more pronounced than for [4]. A possible explanation could be that our depth-based loss landscape exhibits less of the problematic local minima for objects with continuous symmetries than for those without.

B. Multi-Pose Ablation Study

Through an ablation study we show that using the multi-pose depth loss (in this case with 10 poses) increases the performance of the pose prediction recall considerably compared to the single-pose depth loss, as shown in Table III. As we discussed earlier in Sec. III-B, the single-pose depth loss may be more prone to get stuck in local minima during training.

C. Memory Consumption

A comparison of the memory consumption between our approach and the one by [4] is shown in Table IV. The memory consumption of both the encoder and the codebook are taken directly from [4] while the memory consumption of our pose regression network is found by calculating the theoretical size of the network and confirming it in a PyTorch implementation. Our approach consumes significantly less memory than [4]. The codebook is replaced entirely by the pose regression network, where the latter consumes $\approx$ 70 times less memory.

Loading the necessary encoder, codebooks, and pose regression network for all 30 T-LESS objects would hence require $\approx$ 1365 MB for [4] and only $\approx$ 33 MB for our method. The relative difference gets larger as the number of objects increases. It should be noted that the reported memory consumption does not include the overhead of loading the different machine learning frameworks, such as TensorFlow and PyTorch, into memory.

TABLE III: Average VSD recall for the T-LESS primesense test dataset with the single-pose loss function and with the multi-pose loss function (10 poses). Multiple poses increases performance considerably, especially for objects with discrete symmetries. Results are shown with mean and standard deviation as in previous tables.

	1 pose	10 poses	improvement
Continuous symmetries	57.37 ± 1.6	**67.23** ± 2.7	9.86
Discrete symmetries	50.62 ± 0.9	**59.51** ± 0.3	8.89
All objects	53.10 ± 0.6	**62.34** ± 0.9	9.24

TABLE IV: Memory consumption during inference of 30 objects. Our solution consumes $\approx$ 40 times less memory than the codebook-based solution.

	Encoder	**Codebook**	**Pose Regression Network**	**Total**
Codebook [4]	15 MB	30×45 MB	-	1365 MB
Ours		-	30×0.6 MB	33 MB

D. Inference Time

The inference time of our approach, implemented in PyTorch, is evaluated against the public codebase by [4], implemented in TensorFlow. All timings were measured on a laptop equipped with the following hardware: an i7-7700HQ CPU (2.80GHz) and a NVIDIA GTX 1060 6GB GPU. Note that any measurements related to the projective distance calculation originally mentioned by [4] have been excluded as it is only needed for translation estimation.

Our approach achieves real-time performance with an inference time of $\approx$ 6.2 ms, to estimate the pose of an object. This is an improvement over the current state-of-the-art codebook-based approach by [4] which takes $\approx$ 7.0ms per object. Replacing the codebook-based approach, and thereby both the cosine similarity and nearest neighbor computations, with our network, decreases inference time slightly. The computation time for the encoder should be nearly identical for both approaches as the exact same architecture is used with the only exception being the deep learning framework. Note that the slight increase in computation time when comparing to the measurements reported in [4] is due to the differences in hardware used in the two evaluations.

V. Future Work

The method in this paper predicts the rotation of each object, given a bounding box placing the object in the image. We base our output on a rotation matrix, but thanks to the differential rendering scheme this can easily be extended to output both a rotation and translation estimate instead. This could then be used to do small translation corrections within the bounding box or even determine the full translation in the input image if larger images are provided to the encoder. We expect this would improve the final 6D pose prediction without costly fine-tuning procedures.

Another natural extension for our method is to replace the individual pose regression networks for each object with a single shared pose regression network. This would decrease memory consumption further and it is possible that a pose regression network trained on all objects simultaneously would generalize better. Another benefit of using a shared pose regression network is that it does not require classification of the detected objects as our pose regression network could be trained to also perform the classification.

Our current results are based on freezing the weights of the encoder during training of our pose regression networks. It may be possible to increase the performance of our approach

by fine-tuning the encoder or some part of it while training the pose regression network. Another unexplored option is to increase the size of the latent space produced by the encoder as it could increase performance of our pose regression network. Keeping the latent space small makes sense for the codebook-based approach by [4] as the memory consumption of the codebook scales linearly with it. Our approach is much less affected by the size of the latent space.

In many applications where our method would be useful, for instance real-time robotics, inference is done on video rather than independent images. In that context the system can be improved by integrating the temporal aspect, though a particle filter or similar methods [22]. For such a solution, multiple candidates from the multi-pose approach can be utilized.

Finally, we would also like to explore how each pose in our multi-pose solution behaves as a function of the pose in the input image. One question is whether each pose estimate is localized to a certain pose region, while the confidence jumps between them, or if the pose estimates vary more as the input changes. Analysis similar to the principal component analysis done by [4] could reveal this, as well as strengths and weaknesses of our approach.

VI. Conclusion

In this paper, we proposed a novel multi-pose loss function to train a neural network to estimate the rotation of an object from an RGB image. This loss is constructed such that it accounts for any symmetries, an important issue in pose estimation, without relying on predefined symmetries. Our loss is inspired by the VSD (Visible Surface Discrepancy) metric and relies on evaluating the estimated pose by depth comparison. This solution only requires RGB images as input, as the depth maps for the loss are produced by differential renderings of CAD models.

Our network is trained purely on synthetic data. We expand upon an existing state-of-the-art method which utilizes an encoder and codebook to estimate poses [4]. We show that our pose regression network can replace the codebook entirely by directly estimating poses from the output of the encoder. By making our network output multiple poses together with confidences that selects one of them, we show that the recall, as measured by the VSD metric, can be increased. When training our network on top of a pre-trained encoder, shared for all objects, we get a solution that requires a fraction of the memory and has higher pose recall than the state-of-the-art codebook-based approach. It is slightly faster and is not limited by a discretization. Our solution retains or improves many of the interesting properties for robotic applications such as real-time inference, low memory usage, training on synthetic data and only requiring RGB images.

Relying on a neural network instead of a codebook should also make our approach more easily extendable. For instance, integrating a translation estimate into the pose regression network, or training a single pose regression network for multiple objects, instead of having separate networks for each object. These extensions are left for future work.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP).

References

[1] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "BOP challenge 2020 on 6d object localization," in *ECCV Workshops*, 2020.

[2] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *WACV*, 2017.

[3] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, September 2018.

[4] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel, "Multi-path learning for object pose estimation across domains," in *CVPR*, June 2020.

[5] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Pytorch3d," https://github.com/facebookresearch/pytorch3d, 2020.

[6] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *ECCV*, 2016, pp. 834–848.

[7] J. Vidal, C. Lin, and R. Martí, "6d pose estimation using an improved method based on point pair features," in *ICCAR*, 2018, pp. 405–409.

[8] B. Grossmann and V. Krüger, "Fast view-based pose estimation of industrial objects in point clouds using a particle filter with an icp-based motion model," in *INDIN*, 2017, pp. 331–338.

[9] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, June 2019.

[10] C. Song, J. Song, and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *CVPR*, June 2020.

[11] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *CVPR*, June 2020.

[12] K. Park, A. Mousavian, Y. Xiang, and D. Fox, "Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation," in *CVPR*, June 2020.

[13] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *ECCV*, 2020.

[14] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *CoRL*, 2018.

[15] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IROS*, 2017, pp. 23–30.

[16] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *ICCV Workshops*, 2017, pp. 2174–2182.

[17] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei, and L. Hao, "On the continuity of rotation representations in neural networks," in *CVPR*, June 2019.

[18] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. G. Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "Bop: Benchmark for 6d object pose estimation," in *ECCV*, Cham, 2018, pp. 19–35.

[19] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *ECCV Workshops*, G. Hua and H. Jégou, Eds., 2016, pp. 606–619.

[20] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.

[21] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "BOP: Benchmark for 6D object pose estimation," *ECCV*, 2018.

[22] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking," in *RSS*, 2019.