# Learning Implicit Representations of 3D Object Orientations from RGB

Martin Sundermeyer[1], En Yen Puang[1], Zoltan-Csaba Marton[1], Maximilian Durner[1], Rudolph Triebel[1,2]

***Abstract*— This work presents a fast and robust algorithm for object orientation estimation that is solely trained on synthetic views rendered from a 3D model. We introduce a dense encoder-decoder architecture that learns implicit representations of 3D object orientations. Since our training is self-supervised, we avoid the necessity of real, pose-annotated training data. Furthermore, it prevents issues related to ambiguous object views. To encode latent representations that are robust against occlusions, clutter and the differences between synthetic and real data, a new domain randomization strategy is proposed. We motivate our approach by experiments on abstract 2D shapes and evaluate it on the challenging T-LESS dataset. In addition to the results in this paper, we provide a live presentation of the system during the workshop, on a Nvidia Jetson TX2 board.**

## I. Introduction

Object pose estimation is essential for autonomous manipulation tasks. In recent literature, there exist a lot of approaches (e.g [1], [2], [3]) with different strengths and weaknesses. The significance of object pose estimation is further underlined by the latest Amazon Picking Challenge and SIXD Pose Estimation Challenge.

Compared to the current state of the art, our approach is significantly different in that it represents object orientations *implicitly*, i.e. we do not train a mapping from input images to explicit pose labels. Instead, we learn a latent representation of object views to retrieve a set of hypotheses for the object orientation. This has several advantages, as we show in the experiments. First, being independent from a concrete representation within $SO(3)$ allows us to handle ambiguous poses caused by symmetric views, because we avoid one-to-many mappings from images to orientations. Second, our implicit representation is learned such that it is robust against occlusion, background clutter and translational variations. And last but not least, our approach does not require any real annotated training data. Figure 1 depicts how our method fits into an object grasping system.

We offer a live demonstration of our system to be shown at the workshop, implemented on an embedded GPU with a simple camera connected to it.

## II. Related Work

Recent contributions in the field of Deep Learning surpassed human performance level for 2D object detection from RGB images [4], [5] on ImageNet [6]. To further improve efficiency, DenseNet[7] proposed feature reuse by having a densely connected Convolutional Neural Network

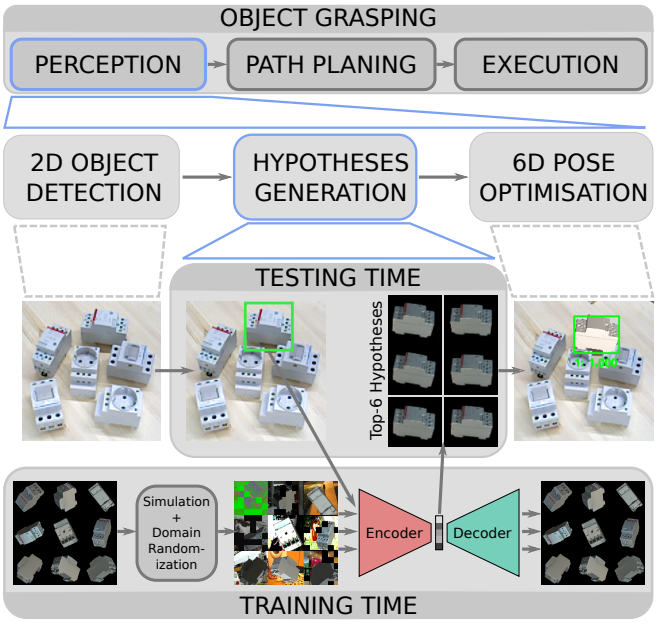


**Fig. 1:** Our method fits into an object grasping pipeline by generating hypotheses for 3D object orientations from a scene crop. The Autoencoder architecture is trained to revert geometric and color input augmentations on random synthetic object views. This allows us to extract implicit representations of object orientations from real camera recordings.

(CNN) and have shown to achieved the state of the art with fewer parameters.

However, such approaches rely on large amounts of manually annotated data, which becomes more difficult and cumbersome to produce for object orientations. Nonetheless, Deep Learning was applied for such problems, at first as a way to describe and re-detect keypoints during traditional matching methods.

[1] proposed to estimate orientation by having orientation classification heads in its Single Shot Multibox Detector (SSD) [4] style network. Besides the original object classification and localization heads in SSD, it has two additional heads for predicting viewing angle and in-plane rotation of object in feature maps. This method keeps detection and orientation estimation in single shot (one forward-pass) and therefore able to run in very high speed (excluding the pose refinement stage). However, symmetry and view ambiguity of each objects need to be identified and removed manually before training to assure convergence.

[8] proposed to estimate orientation using regression directly on axis-angle and quaternion. Its loss function defines a geodesic distance between predicted and target rotation matrix or quaternion, and then simplify it with Rodrigues' rotation formula. Because there are constraints in the representation and ambiguities in observed object poses, con-

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany. Email: martin.sundermeyer@dlr.de, [2]Department of Computer Science, Technical University of Munich (TUM), Germany. Email: rudolph.triebel@in.tum.de

vergence issues appear [9], and thus regressing orientations in SO(3) is challenging to be made practically usable in the generic case.

Extracting latent code in unsupervised manner seem to be a good alternative for a few reasons: It handles pose ambiguities directly and most importantly it has fast and robust feature extraction. The main disadvantage is that it cannot be performed is single shot manner and requires processes to estimate final pose. Feature extraction is often done by using an Autoencoder (AE) which takes in a patch of image and produces latent code encoded with 3D rotation or full 6D pose information.

[10], [11], [12] have similar approach in which they train the AE with input patch that only cover part of the object and therefore the latent code contain information of rotation and translation with respect to object center. A codebook is then used to store the latent-label code pair and provides nearest neighbors when queried. Pose is then estimated by a voting mechanism involved those nearest neighbors. This method is robust against occlusion because an input patch never cover the entire object and it ensembles patches with votes. However, it builds a huge codebook which require efficient searching technique e.g. kd-tree for large amount of queries.

Instead of using codebook, [13] proposed to use a Hough Forest to estimate patch's pose. A sparse AE is used to encode image patches so that its latent code is at most of time zero. Similar to the above, each of the leaf nodes contains all class, translation and rotation information about the patch. Each of the trees in the forest casts a vote to tell what is in the bigger picture. It trades intensive codebook searching with training a Hough forest and gained the ability to measure prediction uncertainty by looking at leaf node distributions.

We propose to train a similar AE-codebook combination but with complete object patches. The AE is fed with patches that cover the entire object and through various methods we predispose its latent code to encode the 3D rotation information. Moreover, we rely solely on synthetic training data and thus avoid the labeling problem of approaches like [14], [15].

To solve synthetic feature over-fitting, transfer learning is a straight forward and time-saving approach. [16] showed that transferred and frozen feature extractor helps to mitigate over-fitting in object detection. To achieve better result, heavy data augmentations were applied to synthetic data. However, results differ across test images captured by different cameras suggests that there exist other type of over-fittings yet to be resolved.

Although most of the approaches discussed above use heavy data augmentations as a way to avoid synthetic feature over-fitting, [17] proposed to learn the feature discrepancies directly. A mapping network maps or matches features extracted from an real image patch with features extracted from the same image patch but replaced real object pixels with synthetically rendered object at the same pixel location. While this is a more direct way to tackle feature discrepancies, it requires the same amount of real data to the amount of synthetic data used in training.

Adversarial examples was first introduced as a reliability concern and security threats of models trained with machine learning techniques. [18] and many other have proved that neural networks and other techniques are highly vulnerable to adversarial attacks. Adversarial attacks refers to techniques that generate slightly perturbed input which then causes huge and destructive changes on the output. [19] proposed Fast Gradient Sign Method (FGSM) as a way to generate adversarial examples. [20] proposed an algorithm to train robust model against adversarial examples with variant of attacks such as iterative and least-likely FGSM. [21] proposed to add random noise to input data before performing FGSM as to better estimate the local gradient of a data point.

We will present several experiments exploring the feasibility of the above concepts.

## III. Method

In this chapter we derive a new training strategy for an encoder-decoder CNN-architecture to generate 3D object orientation hypotheses.

### A. Autoencoders

An AE is a variant of a CNN that is made of two parts: encoder and decoder. A typical encoder transforms, i.e. encodes, high dimensional input like an image into a low dimensional representation, i.e. a latent code. The decoder reconstructs the original input from this latent code.

In the original Autoencoder, the latent code does not have constraints. However, Variational Autoencoder (VAE)s impose prior distribution on latent code by minimizing Kullback-Leibler (KL) divergence between latent code and a prior distribution of choice. To be able to back-propagate the KL divergence loss, the exact functional form of the prior distribution need to be accessible. For the representation of SO(3), however, we found such constraints to be counterproductive, and thus leave the latent code take any shape.

There are several variants of AEs, aimed mainly to acquire various invariance properties and avoid over-fitting. Regularized AE proposed to impose weight decay which favours small weights to avoid over-fitting.

Vincent et al. [22] proposed a modified training procedure. By applying artificial random noise to the input images $x \in \mathcal{R}^{\mathcal{D}}$ while keeping the reconstruction target $y \in \mathcal{R}^{\mathcal{D}}$ clean, the trained model learns to reconstruct denoised test images. Furthermore, the latent representations becomes invariant to noise because it is irrelevant for the reconstruction of denoised images.

In the following, we will experimentally show that using this training strategy we can actually enforce invariance not only against noise but against a variety of input augmentations. Eventually, this technique lets us close the domain gap between rendered and real image data.

### B. Translation Invariant Autoencoder

In order to specifically encode 3D object orientations, we have to control what the code represents and what

input variations should be ignored. Therefore, we apply random augmentations to the input images against which the encoding shall become invariant. On the other hand, the reconstruction target remains to reconstruct a non-augmented ouput view. Since all augmentations are irrelevant for the reconstruction, the code does not contain any information about them. We learn latent representations of binary images depicting a 2D square at different viewpoints to justify our claims. While training with different scales and translations, we only want to encode the in-plane rotations in a two dimensional latent space independent of scale or translation. We train a CNN-based AE architecture similar to the model in Fig. 2. AEs trained on reconstructing squares at fixed scale and translation or random scale and translation do not clearly encode rotation alone, but are also sensitive to other latent factors. Instead, if the input images are randomly translated while the output stays centered, the encoding of the Augmented Autoencoder (AAE) becomes invariant to translation such that all squares with coinciding orientation are mapped to the same code. The latent representation is also traversing much smoother through the different orientations. The latent dimensions follow a sine function with frequency $f = \frac{4}{2\pi}$ respectively, because the square has two axes symmetries. Every $\frac{\pi}{2}$ rotation, the square appears the same. The AAE represents orientations based on the appearance of an object and not based on pose labels. Learning individual representations of 3D object orientations is beneficial in case of symmetric object views.

### C. 3D Orientations from Synthetic Data

Using geometric augmentations, we can explicitly learn representations of object in-plane rotations. This can also be applied to encode the whole SO(3) space of object views rendered from a 3D model (CAD or 3D reconstruction). It assures robustness against inaccurate object detections. Still, image crops from real RGB sensors could not be interpreted since a 3D model is always imperfect, the encoder can't differentiate the object from occlusions and background clutter and not all lighting conditions can be realistically simulated. The AAE imposes a new Domain Randomization (DR) stategy to generate encodings invariant to irrelevant differences between real and simulated images. The goal is that the trained encoder treats the differences to real camera images as just another variation. We randomly apply color augmentations to the input training image, but leave the reconstruction target with black background and fixed light. The input object views are produced by rendering with random light positions and randomized diffuse and specular reflection [24]. We also use contrast, brightness, Gaussian blur and color distortions. To simulate occlusion we crop out squared parts of the object. Instead of black background we insert random background images from the Pascal VOC dataset [25].

### D. Architecture and Training details

Our baseline architecture is a convolutional Autoencoder with filter size 5x5. In our experiments we also use a densely connected AE that is depicted in Figure 2. We use a bootstrapped pixel-wise L2 loss which is only computed on the pixels with the largest errors (per image bootstrap factor b=16). Thereby, finer details are reconstructed and the training does not converge to local minima. Using OpenGL, we render 20000 object views uniformly at random 3D orientations and constant distance along the camera axis (700mm). The resulting images are quadratically cropped and resized to $64 \times 64 \times 3$. All geometric and color input augmentations besides the rendering with random lighting conditions are applied online during training at uniform random strength, parameters are depicted in Table I. We use the Adam optimizer with a learning rate of $2 \times 10^{-4}$, Xavier initialization, a batch size = 64 and 30000 iterations of training which takes $\sim 4$ hours on a single GPU.

**TABLE I:** Augmentation Parameters; Scale and translation is in relation to image shape and occlusion is in proportion of the object mask

| | add | contrast | multiply | invert | Gaussian blur |
|---|---|---|---|---|---|
| 50% chance (30% per channel) | $\mathcal{U}(-25, 25)$ | $\mathcal{U}(0.4, 2.3)$ | $\mathcal{U}(0.6, 1.4)$ | per channel | $\mathcal{U}(0.0, 1.2\sigma)$ |

| ambient | diffuse | specular | scale | translation | occlusion |
|---|---|---|---|---|---|
| 0.4 | $\mathcal{U}(0.7, 0.9)$ | $\mathcal{U}(0.2, 0.4)$ | $\mathcal{U}(0.8, 1.2)$ | $\mathcal{U}(-0.15, 0.15)$ | $\mathcal{U}(0, 0.25)$ |

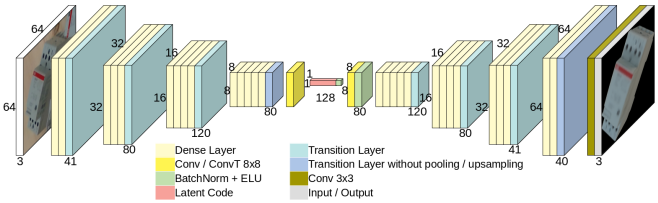


**Fig. 2:** Dense Autoencoder CNN architecture

### E. Codebook Generation and Lookup

Given the trained AAE, we are able to reconstruct a 3D object of a real scene crop for a variety of camera sensors (Fig. 4). The clarity and orientation of the decoder reconstruction is an indicator of the encoding quality. In order to obtain the 3D orientation estimation of an object a so-called codebook has to be created. Therefore we first of all generate clean – meaning centered and without augmentations on a black background – renderings from the object from equidistant viewpoints given a full view-sphere (based on a refined icosahedron [26]). In the next step, to cover the whole SO(3), each of the generated views are rotated in-plane at fixed intervals. Finally, all obtained views are forwarded to the AAE. The obtained latent code $z \in \mathcal{R}^{128}$ in combination with the corresponding rotation $R_{cam2obj} \in \mathcal{R}^{3x3}$ form the codebook, as can be seen in Fig. 5 (blue encodings).

At test time, the detected object crop(s) (see Sec. III-F) from a RGB scene are resized (to match the input size) and forwarded into the AAE. The obtained latent code (see Fig. 5 red encodings), depicted as test code $z_{test} \in \mathcal{R}^{128}$, is then used to compute the cosine similarity against all codes $z_i \in \mathcal{R}^{128}$ from the codebook: $cos_i = \frac{\boldsymbol{z}_i \, \boldsymbol{z}_{test}}{\|\boldsymbol{z}_i\| \|\boldsymbol{z}_{test}\|}$.

The highest similarities are determined in a k-Nearest-Neighbor (kNN) search and the corresponding rotation matri-

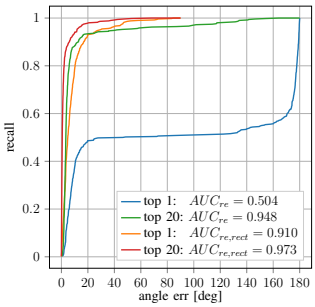


**Fig. 3:** Object 9 from T-LESS (has a symmetry axis) tested on all 504 Primesense RGB views of scene 11; recall at different cutoffs; with and without symmetry error correction; best of top 1 and top 20 predictions

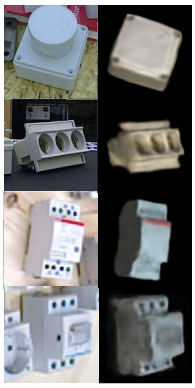

**Fig. 4:** AAE decoder reconstruction of T-LESS (top) and THR (bottom) scene crops

ces $\{R_{kNN}\}$ from the codebook are returned as hypotheses of the 3D object orientation. The use of the cosine similarity is due to the computation efficiency on a single GPU for large codebooks (in our experiments 2562 viewpoints $\times$ 36 in-plane rotations result in 92232 total entries). Furthermore, experiments showed that there is no effect of scaling a latent test code towards the object orientation of the decoder reconstruction. We assume, that this a result of the circular nature of rotations.

### F. Training an Object Detector

To apply the AAE on whole scenes we train SSD with VGG16 base [4] using object recordings on black background from different viewpoints which are provided in the training set of T-LESS. Multiple objects are copied in a scene at random scale and translation. Bounding box annotations are automatically adapted. As for the AAE, the black background is replaced with Pascal VOC images. During training with 60000 scenes, we apply weak color distortions and random flips.

### G. Inference Time

SSD with VGG16 backbone and 31 classes plus the AAE with a codebook size of 92232 $\times$ 128 yield the average inference times depicted in Table III. We conclude that the RGB-based pipeline is real-time capable at $\sim$ 42Hz on a Nvidia GTX 1080 and modern CPU cores. Multiple encoders and corresponding codebooks easily fit into the GPU memory. This enables augmented reality and robotic applications and leaves room for tracking algorithms.

## IV. Evaluation

We evaluate the AAE on the T-LESS [2] dataset, our THR dataset [27] and provide qualitative results on an IKEA cup.

### A. Metrics

*1) Axis-angle Rotation Error:* A single-valued absolute angle error $e_R \in [0^o, 180^o]$ can be computed between the estimated rotation matrix $\boldsymbol{R}_{est\_cam2obj}$ and the ground truth rotation matrix $\boldsymbol{R}_{gt\_cam2obj}$

$$e_R = \arccos\left(Tr(\boldsymbol{R}^T_{est\_cam2obj}\,\boldsymbol{R}_{gt\_cam2obj} - \boldsymbol{I})\,/\,2\right) \quad (1)$$

**TABLE II:** Ablation study on color augmentations for different test sensors. Object 5, all scenes, T-LESS [2]. Standard deviation of three runs in brackets.

| Train RGB | Test RGB | dyn. light | add | contrast | multiply | invert | $AUC_{vsd}$ |
|---|---|---|---|---|---|---|---|
| 3D Reconstruction | Primesense | ✓ | | | | | 0.472 (± 0.013) |
| | | ✓ | ✓ | | | | 0.611 (± 0.030) |
| | | ✓ | ✓ | ✓ | | | 0.825 (± 0.015) |
| | | ✓ | ✓ | ✓ | ✓ | | 0.876 (± 0.019) |
| | | ✓ | ✓ | ✓ | ✓ | ✓ | **0.877** (± 0.005) |
| Primesense | Primesense | | ✓ | ✓ | ✓ | | 0.890 (± 0.003) |
| 3D Reconstruction | Kinect | ✓ | | | | | 0.461 (± 0.022) |
| | | ✓ | ✓ | | | | 0.580 (± 0.014) |
| | | ✓ | ✓ | ✓ | | | 0.701 (± 0.046) |
| | | ✓ | ✓ | ✓ | ✓ | | 0.855 (± 0.016) |
| | | ✓ | ✓ | ✓ | ✓ | ✓ | **0.897** (± 0.008) |
| Kinect | Kinect | | ✓ | ✓ | ✓ | | 0.917 (± 0.007) |

This metric depicts an intuitive estimation of the object orientation error. It is convenient in scenarios where the exact orientation is uniquely defined. However, for the evaluation of ambiguous objects, it is necessary to predefine the symmetries to correct the acquired rotation errors. We define the area under the $e_R$ / recall curve (trapezoidal integration) over multiple predictions as

$$AUC_{re} = \frac{1}{\pi}\int_0^{\pi} recall(e_R)\,de_R \quad (2)$$

In case of objects with a symmetry axis we rectify the metric

$$AUC_{re,rect} = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} recall(min(|e_R|, |e_R - \pi|))\,de_R \quad (3)$$

*2) Visible Surface Discrepancy:* The Visible Surface Discrepancy ($e_{vsd}$) [28] is an ambiguity-invariant pose error function that measures the pixel-wise depth deviation between the visible 3D surface of a rendered object model at ground truth pose and at the estimated pose. To individually evaluate the AAE we only predict the 3D orientation, here. Thereby, the issue of evaluating pose estimations resulting from symmetric object views is circumvented. The area under the $e_{vsd}$ / recall curve (trapezoidal integration) over multiple scene predictions reads

$$AUC_{vsd} = \int_0^1 recall(e_{vsd})\,de_{vsd} \quad (4)$$

### B. Test Conditions

Most RGB-based pose estimation approaches not only rely on 3D model information but also pose-annotated data. It is common practice to ignore in-plane rotations or only consider object poses that appear in the dataset [29], [15] which limits applicability. Learning-based approaches are commonly allowed to overfit when training and testing on different viepoints of the same scene. Symmetric object views are often individually treated [29], [14] or ignored [15]. We do not rely on any of these assumptions and thus provide a more applicable solution.

### C. Results

To assess the AAE alone, we predict the 3D orientation of object 5 from the T-LESS dataset on Primesense and Kinect RGB scene crops. Table V shows the influence of reverting different input augmentations. It can be seen that the effect

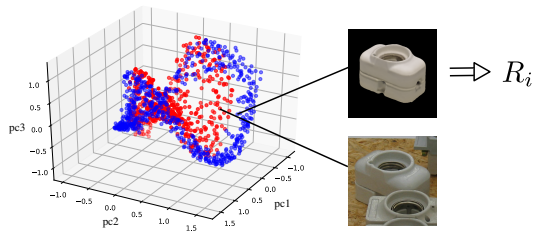


**Fig. 5:** Axes: 3 principle components $pc_i$ of the latent space; red: test encodings of scene 2, object 5, Canon RGB, T-LESS; blue: encodings of synthetic model views (upper hemisphere, upright)

**TABLE III:** Inference time of pipeline blocks

| | 8 CPU | GPU |
|---|---|---|
| SSD | - | **∼17ms** |
| Encoder | - | **∼5ms** |
| Cosine Similarity | 2.5ms | **1.3ms** |
| Nearest Neighbor | **0.3ms** | 3.2ms |
| | | **∼24ms** |

**TABLE IV:** $AUC_{re,rect}$ in the THR Dataset's scenes 2 and 4.

| Object | Scene 2 | Scene 4 |
|---|---|---|
| 1 | 0.792 | 0.785 |
| 2 | 0.818 | 0.823 |
| 3 | 0.833 | 0.820 |
| 4 | 0.741 | 0.634 |
| 5 | 0.743 | 0.565 |
| 6 | 0.636 | 0.645 |
| 7 | 0.812 | 0.795 |
| 8 | 0.771 | 0.771 |
| 9 | 0.672 | 0.581 |
| Average | 0.769 | 0.713 |

of the color augmentations is cumulative. For texture-less objects, even the inversion of color channels seems to be beneficial since it prevents over-fitting to synthetic color information. Furthermore, training with real object recordings provided in T-LESS with random Pascal VOC background and augmentations yields only slightly better performance than the training with synthetic data. Figure 3 shows typical $e_R$ / recall curves for an axis symmetric object. While the nearest neighbor is likely to yield a good approximation of the 3D orientation, the top 20 nearest neighbors quite certainly include a correct estimate. This property of fast search space reduction is valuable for costly pose refinement methods. The latent space size is important for inference time. The performance increases with more dimensions and starts to saturate at $dim = 64$. Since inference is very efficient we chose $dim = 128$ in our experiments. Our Domain Randomization strategy even allows the generalization from RGB views of an untextured CAD model. In that case, even more radical augmentations help but also slow down the convergence. Figure 6 shows the AAE predictions after being trained on views of an IKEA cup model.

**Fig. 6:** Incomplete IKEA mug 3D orientation estimation from webcam stream (left), nearest training neighbors (right)

### D. Extensions

We also adopted the Autoencoder using a DenseNet [7] architecture with growth $k = 40$ and 2/3/4/5 dense layers in 4 consecutive dense blocks for both encoder and decoder (mirrored). Dilated convolution [30] is used in every dense layer to increase the perception range as dense blocks go deeper. The amount of dilation grows from 1 to 3 and repeat in consecutive dense layers until the end of dense block. During training, cosine learning rate annealing with warm restarts [31] is used to achieve better parameters utilization. As the result, size of DenseNet model is decreased to 3M parameters in the entire AE without negatively affecting the performances.

**TABLE V:** Effects of densely connected blocks, embedding loss (E) and adversarial augmentation (A) on top 1 $AUC_{re,rect}$ for TLESS scene 11. Baseline with ordinary 5x5 convolutional layers.

| Object | Baseline | Dense | Dense$^E$ | Dense$^A$ | Dense$^{EA}$ |
|---|---|---|---|---|---|
| 5 | 0.896 | 0.928 | 0.916 | 0.934 | 0.935 |
| 8 | 0.923 | 0.915 | 0.910 | 0.930 | 0.913 |
| 9 | 0.909 | 0.890 | 0.883 | 0.904 | 0.905 |
| 10 | 0.848 | 0.860 | 0.851 | 0.882 | 0.887 |
| Average | 0.894 | 0.898 | 0.890 | 0.912 | 0.910 |

Furthermore, we tried an embedding loss that penalizes the deviation of latent codes between training data with and without augmentations using cosine similarity, so is to reduce variance in latent code induced by Domain Randomization (DR) and therefore improve efficiency of decoder. On top of this we augment the synthetic input training data with adversarial examples generated using a randomized combination of different methods described in [20].

## V. Conclusion

We introduced an augmented training method for a dense Autoencoder architecture that enables 3D object orientation estimation on RGB camera data. We showed that it is possible to solely train on rendered views of a 3D model. By reverting geometric and color input augmentations in the reconstruction, we learn representations that encode 3D object orientations independent of the domain, i.e. synthetic or real RGB images. Furthermore, pose ambiguities stemming from symmetric object views do not affect our approach since the self-supervised training is independent of pose labels.

## References

[1] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1521–1529.

[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," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[3] J. Vidal, C.-Y. Lin, and R. Martí, "6d pose estimation using an improved method based on point pair features," *arXiv preprint arXiv:1802.08516*, 2018.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[7] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, p. 3.

[8] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," *arXiv preprint arXiv:1708.05628*, 2017.

[9] A. Saxena, J. Driemeyer, and A. Y. Ng, "Learning 3-d object orientation from images," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 794–800.

[10] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 205–220.

[11] V. A. Knyaz, O. Vygolov, V. V. Kniaz, Y. Vizilter, V. Gorbatsevich, T. Luhmann, N. Conen, W. Forstner, K. Khoshelham, S. Mahendran, *et al.*, "Deep learning of convolutional auto-encoder for image matching and 3d object reconstruction in the infrared range," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2155–2164.

[12] H. Zhang and Q. Cao, "Combined holistic and local patches for recovering 6d object pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2219–2227.

[13] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3583–3592.

[14] V. Balntas, A. Doumanoglou, C. Sahin, J. Sock, R. Kouskouridas, and T.-K. Kim, "Pose guided rgbd feature learning for 3d object pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3856–3864.

[15] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3109–3118.

[16] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," *arXiv preprint arXiv:1710.10710*, 2017.

[17] M. Rad, M. Oberweger, and V. Lepetit, "Feature mapping for learning fast and accurate 3d pose inference from synthetic images," *arXiv preprint arXiv:1712.03904*, 2017.

[18] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[21] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[22] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[23] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 833–840.

[24] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[26] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab, "Simultaneous recognition and homography extraction of local patches with a simple linear classifier," in *Proceedings of the British Machine Conference, pages*, 2008, pp. 10–1.

[27] M. Durner, S. Kriegel, S. Riedel, M. Brucker, Z.-C. Márton, F. Bálint-Benczédi, and R. Triebel, "Experience-based optimization of robotic perception," in *Advanced Robotics (ICAR), 2017 18th International Conference on*. IEEE, 2017, pp. 32–39.

[28] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 606–619.

[29] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," *arXiv preprint arXiv:1703.10896*, 2017.

[30] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[31] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2016.