Learning 6DoF Object Poses from Synthetic Single Channel Images

Jason Rambach* Chengbiao Deng† Alain Pagani‡ Didier Stricker§

German Research Center for Artificial Intelligence, DFKI
Department Augmented Vision
Kaiserslautern, Germany

ABSTRACT

Estimation of 6DoF object poses from single images is a problem of great interest in augmented reality and robotics research since it enables interaction with the object or initialization of pose tracking. Approaches utilizing deep neural networks have shown good performance, however the majority of them rely on training on real images of the objects which can be challenging in terms of ground truth pose acquisition, scalability and full coverage of possible poses. In this paper, we disregard all depth and color information and train a CNN to directly regress 6DoF object poses using only synthetic single channel edge enhanced images. We evaluate our approach against the state-of-the-art using synthetic training images and show a significant improvement on the commonly used *LINEMOD* benchmark dataset.

Index Terms: Computing methodologies—Artificial intelligence—Computer vision—Computer vision tasks; Computing methodologies—Machine learning—Machine learning approaches—Neural networks;

1 INTRODUCTION

6 Degree of Freedom (6DoF) pose estimation of arbitrary objects from single camera images is a very important problem for robotics and augmented reality applications that has received a lot of attention from the scientific community recently. A precise object pose is needed for interaction with an object and for initialization of frame-to-frame object pose tracking.

Traditional approaches match texture features of objects and estimate the pose by solving a Perspective-n-Point (PnP) Problem with the recovered to 2D-3D correspondences, often in a RANSAC framework for outlier rejection [19]. Such approaches can only be applied to sufficiently textured objects and are often sensitive to illumination changes. Augmented Reality systems based on CAD models of non-textured objects using line tracking methods often require manual initialization, requesting the user to align the model with the object pose which can be quite cumbersome [5].

The work of Hinterstoisser et al. [9] was focused on the detection and pose estimation of texture-less objects in RGB-D images by employing learning on templates created using the object model. The released *LINEMOD* dataset is one of the most commonly used benchmark datasets for object pose estimation. Brachmann et al. [7] use uncertainty modelling of the estimated pose from RGB images in a multistage approach with a classification-regression forest, RANSAC pose estimation from 2D-3D correspondences and subsequent refinement of the pose. However the approach is computationally demanding and requires training on real images of the objects thus not tackling the challenge of generalizing synthetic image training to real images.

Figure 1: We use only synthetic images to train a CNN to regress the 6DoF pose of objects in real images

Deep neural network learning approaches have shown excellent results on many computer vision problems over the last few years especially in detection, classification and segmentation tasks [20]. However, their full potential on pose estimation problems may not have been realized yet. A first attempt to use a Convolutional Neural Network (CNN) for direct regression of 6DoF poses was PoseNet [12]. The network based on the GoogLeNet [23] architecture was used for relocalization from real images showing moderate accuracy but was not evaluated for object pose estimation. Several approaches followed attempting to estimate 6DoF poses of objects using a CNN from end-to-end or as part of the estimation pipeline.

SSD-6D [10] is a CNN-based approach to the problem that uses the SSD [15] network architecture to place bounding boxes in images around the object and then use their size to estimate depth. In BB8 [17] a first CNN finds the bounding boxes and a second CNN estimates the 3D positions of the edges of the bounding box that are then used to estimate a 6DoF pose by solving a PnP Problem. Both approaches require an additional refinement step of the pose which has an impact on the time required for estimation. Finally, Tekin et al. [24] introduce a similar approach to BB8, using however only one CNN to directly estimate the 3D positions of the object bounding box corners. It should be noted that both BB8 [17] and Tekin et al. [24] use real training images for their networks which significantly simplifies the problem as the challenge of transferring training on synthetic images to the real world is avoided. Using synthetic images is however a much more viable option for training considering real world applications as it does not require acquisition of images of the real object in real conditions using a tracking system to obtain ground truth and enables the generation of large training datasets covering all possible object poses. To the best of our

*e-mail:Jason.Rambach@dfki.de
†e-mail:Chengbiao.Deng@dfki.de
‡e-mail:Alain.Pagani@dfki.de
§e-mail:Didier.Stricker@dfki.de

knowledge, SSD-6D [10] is the only deep learning pose estimation approach trained only on synthetic data. The general problem of domain adaptation in neural networks has recently been addressed in [6, 21] but not in the context of 6DoF object pose estimation.

In this paper, we present a CNN based approach for direct 6DoF pose estimation of arbitrary objects using single one-channel images. Using the pencil filter image preprocessing step initially introduced in [18] for edge feature matching, we manage to deploy a CNN to perform 6DoF object pose estimation while training only on synthetic rendered images of the object 3D models without relying on color information. We use a modified PoseNet architecture for the first time in the object pose problem to directly regress the 6DoF poses and introduce a novel loss function fit to the problem at hand to facilitate the training process. In detail we propose the following contributions in our work:

- The use of pencil images for training a CNN. Using this illumination invariant representation allows for a successful deployment of models trained only on synthetic images to estimate poses of real life images.
- A novel loss function (ADD loss) for training a modified version of PoseNet, inspired by an error metric commonly used to evaluate the performance of object pose estimation.
- An overall approach that outperforms the state-of-the-art in similar conditions (i.e. no depth images, training exclusively on synthetic images) while also being computationally efficient.
- A new dataset of textured object images taken under challenging illumination conditions is made available to the community for experimentation.

The rest of this paper is organized as follows: In Section 2 we give a thorough description of the tackled problem and introduce the notation used throughout the paper. In Section 3 we give an overview of our proposed pose estimation system and describe the introduction of the ADD loss function for training as well as the use of the pencil filter for bridging the gap between synthetic and real images. Subsequently, in Section 4 we evaluate the use of the ADD loss function and the pencil filter and provide comparisons to the state-of-the-art on the commonly used *LINEMOD* benchmark dataset. We also present a comparison to feature matching-based estimation for textured objects on our own dataset.

2 Problem Formulation

We address the problem of estimating the 6DoF camera pose of arbitrary objects from single RGB or grayscale images. The estimated pose consists of a rotation matrix $\boldsymbol{R}_{co} \in SO(3)$ representing the rotation from the object coordinate system O to the camera coordinate system C and a translation vector $\boldsymbol{o}_c \in \mathbb{R}^3$ representing the position of the object coordinate system origin in the camera coordinate system. Throughout this work we often use the equivalent unit quaternion representation $\boldsymbol{q}_{co}$ for the rotation $\boldsymbol{R}_{co}$ with $\boldsymbol{q}_{co} = [q_w, q_x, q_y, q_z]$ and $||\boldsymbol{q}_{co}|| = 1$. We define the set of vertices of the objects' 3D models as $\mathscr{V}$.

3 Proposed Approach

We propose the use of a CNN to directly regress 6DoF poses of objects from single monocular images. The problem is of importance for the initialization and reinitialization of augmented reality object tracking or in SLAM systems. We do not use depth information since depth cameras are not widely available in commonly used devices such as smartphones. Our focus lies mainly on the problem of training neural networks on synthetic rendered images and deploying on real life images. Synthetic training offers the advantages that no ground truth setup for real objects needs to be constructed and that rendered images of any object pose can be easily created. Additionally, it is easily possible to create large datasets of rendered images. However, transferring training on synthetic images to real images can be very challenging especially in alternating lighting conditions and given low-quality 3D models of the objects. Commonly this problem is addressed by approaches simulating different lighting conditions during rendering. We expand on this and additionally propose to perform both the training and the evaluation using a normalized lighting invariant representation of the input images, namely the pencil filter (see Section 3.3).

Our proposed pipeline consists of the following steps: We use the 3D models of the objects to create a large dataset of synthetic RGB images of the objects from random poses on a sphere around the object. These images are transformed to 1-channel pencil images and used to train a CNN to regress the 6DoF poses. The CNN architecture is a modified version of the PoseNet architecture (Section 3.1). We consider two different loss functions for the training (Section 3.2). The trained models can then be used for forward prediction of poses on real input images after applying the pencil filter transformation to them.

3.1 Neural Network Architecture

As mentioned previously we use the network architecture of PoseNet [12] which is based on the GoogLeNet [23] architecture. GoogleNet is a 22 layers deep network, originally designed for the task of classification and detection. This architecture was modified in its output layers by replacing softmax classifiers with fully connected feature vectors and 6DoF pose regressor layers to form PoseNet. We change the original input image size from 224×224 to 448×448 in order to learn finer features and improve the pose regression and also to better cope with small objects at a larger distance from the camera. We also modify the input image from 3-channel color images to single channel pencil images. Discarding color information improves the transfer from synthetic image training to real image deployment and reduces the input size. The output of the network is a 6DoF pose estimate consisting of a quaternion $\hat{\boldsymbol{q}}_{co}$ and a translation vector $\hat{\boldsymbol{o}}_c$.

3.2 Loss Function Selection

Using an appropriate loss function is very important to train the network efficiently. Originally, for PoseNet [12] the following simple loss function $\mathscr{L}_{PoseNet}$ of Euclidean distance between the estimated pose and the ground truth pose was used:

$$\mathscr{L}_{PoseNet} = \|\hat{\boldsymbol{o}}_c - \boldsymbol{o}_c\|_2 + \gamma \left\| \boldsymbol{q}_{co} - \frac{\hat{\boldsymbol{q}}_{co}}{\|\hat{\boldsymbol{q}}_{co}\|} \right\|_2 \tag{1}$$

where $\boldsymbol{o}_c, \boldsymbol{q}_{co}$, denote the ground truth position and orientation and $\hat{\boldsymbol{o}}_c, \hat{\boldsymbol{q}}_{co}$ the estimated position and orientation by the network. There are two issues concerning the use of this loss function. First, the computation of a Euclidean loss on quaternions is suboptimal since the spherical constraint is disregarded. However this becomes less of an inaccuracy for small angular distances between quaternions as suggested in [12]. The second issue is the use of the scaling parameter γ used to weigh the error between the position and orientation. This parameter turns out to be very important for a successful training of the network but is highly dependant on the scale of the position in the dataset and the overall range of poses. In a follow-up work to PoseNet [11], the use of the reprojection error in pixels using the estimated poses was suggested as a loss function for the network.

For the evaluation of object pose estimation systems a commonly used metric is the average model vertices distance introduced in [9] and also often referred to as ADD error. We introduce the use of the ADD also as a loss function for the training of our CNN for object

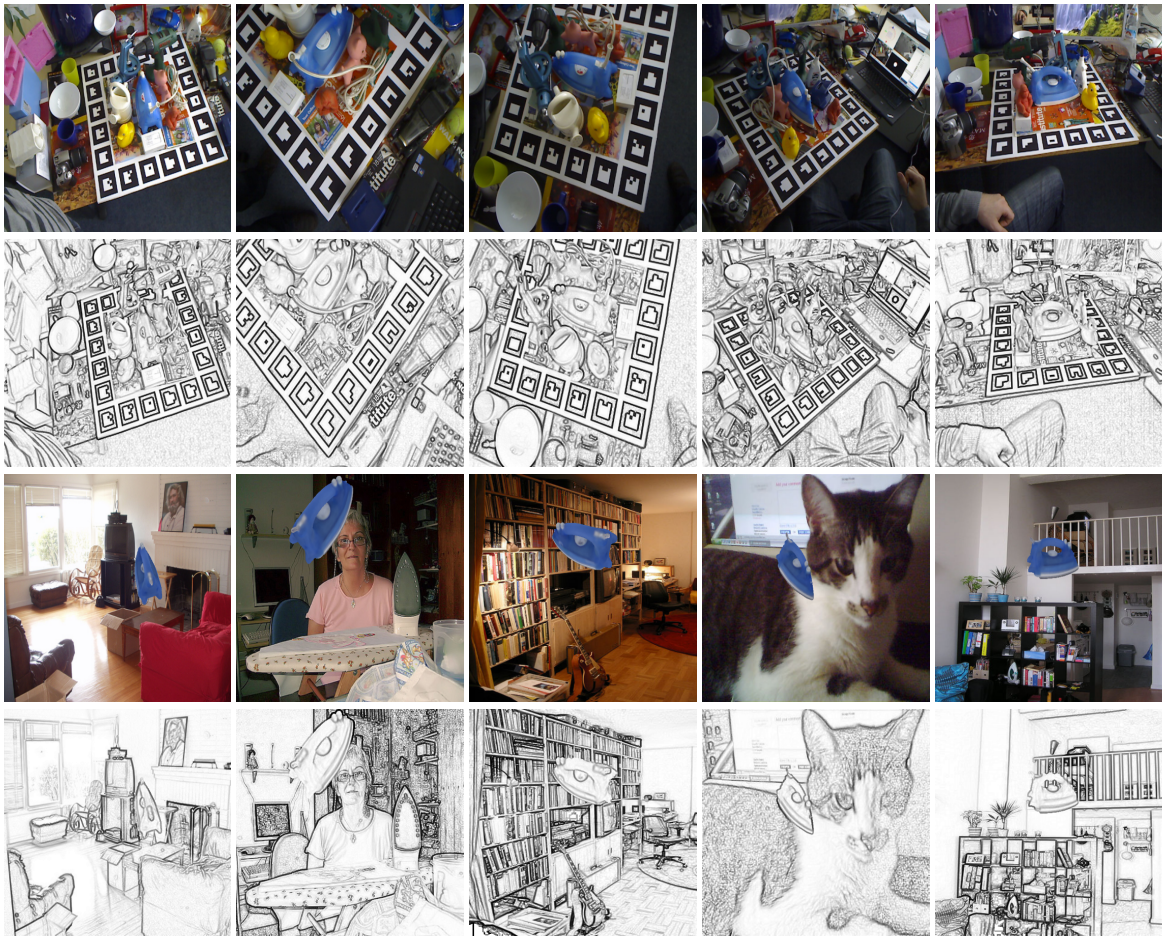

Figure 2: Examples of synthetic training images (i.e. object models rendered over real random backgrounds) in the third row and their pencil filter version in the fourth row. Real images from the *LINEMOD* dataset are displayed in the first row with their equivalent pencil images on the second row.

tracking. Similarly to Equation 1 we define this loss function as:

$$\mathscr{L}_{ADD} = \frac{1}{|\mathscr{V}|} \sum_{\boldsymbol{v} \in \mathscr{V}} \|(\boldsymbol{R}_{co}\boldsymbol{v} + \boldsymbol{o}_c) - (\hat{\boldsymbol{R}}_{co}\boldsymbol{v} + \hat{\boldsymbol{o}}_c)\| \tag{2}$$

We evaluate and discuss the use of both loss functions in the evaluation Section 4.

3.3 Pencil Filter

The use of the pencil filter was recently introduced in [18, 19] in a different context. Specifically, it was used as a preprocessing step for images before performing KLT feature matching [16] for object pose tracking. The pencil filter is an image operator that produces a contrast image by measuring pixel difference with the maximum value in a local neighborhood. In order to retain details in dark parts of the image, the difference is computed in the log domain, then converted back to the standard domain. By noting that a difference in the log domain is equivalent to a division in the standard domain, we can produce the pencil image by dividing the value of a pixel by the maximum value in its neighborhood. In practice, we use a dilation filter with an elliptical structuring element to compute the local maximum, and divide pixel-wise the original image by the dilation image. The generation of a single-channel pencil filter image **P** from an input RGB image **I** is given in Algorithm 1. In the algorithm we assume 8-bits per pixel of the images with values from 0 to 255.

Algorithm 1 Pencil Filter

```
procedure PENCIL FILTER(I)
    G = Convert to grayscale(I)
    P = Dilate(G, ELLIPSE)
    for y=0 to y=G.rows - 1 do
        for x=0 to x=G.columns - 1 do
            if (P(y,x) == 0) then
                P(y,x) = 255
            else
                P(y,x) = int(G(y,x) * 255)/P(y,x)
    return Pencil Image P
```

The pencil filter was successfully applied for matching features between rendered images of object models and images of the corresponding real objects. In this work, we utilise the pencil filter in a machine learning context for the first time in order to tackle the problem of training a deep neural network on synthetic images only and then applying it to real images. Therefore, we train our models on pencil images of the rendered object 3D models and evaluate on pencil images of the real world objects.

Using the pencil representation for learning allows us to close the gap between synthetic and real images. Thus, we do not provide any color information in the network which can be volatile when apply-

ing across different datasets with different illumination conditions or between synthetic and real images. Instead, we provide only edge information in the pencil filter domain. A set of example images are given in Figure 2 showing the similarity of synthetic and real images after application of the pencil filter.

3.4 Network Training

The training sets consist of a number of $20,000 - 30,000$ random poses of the object. These poses are created by first creating a random camera pose on the unit sphere with the object model placed at the origin. Subsequently the camera distance to the object is randomly set and an additional random pertrubation is added to the camera position so that the object is not centered in the image. For experimentation with specific datasets we constrain the training poses to the range of the data in the dataset. We use random backgrounds of indoor scenes taken from the PASCAL VOC dataset [8] and the IKEA Dataset [14]. We augment our rendered training data by adding various effects on the images. In detail, we split the training data in 6 groups and apply Gaussian noise, random contrast and brightness adjustment, motion blur (with a 5×5 kernel that averages in horizontal or vertical direction), speckle noise and a random mixture of all previous effects to each of 5 groups respectively as can be seen in Figure 3. On the 6th group, no image disturbances are added. Subsequently, the pencil filter is applied on the synthetic training dataset and the resulting images are then used to train the network.

Figure 3: Examples of image effects added on synthetic training images before application of the pencil filter. From left to right and top to bottom the effects are: no effect, gaussian noise, contrast and illumination changes, motion blur, speckle noise and mixture of all effects.

4 Evaluation

In this section we present the results of our evaluation of the proposed solution for 6DoF object pose estimation. We first evaluate the two proposed training loss functions in Section 4.2.1. Subsequently in Section 4.2.2, we provide an evaluation of our system on the *LINEMOD* benchmark dataset where we show an improvement towards the state-of-the-art in object pose estimation with synthetic training. We also provide results that directly verify the benefit of using the pencil filter instead of RGB images when training on synthetic data. Finally, we perform a comparison to an edge feature based approach on our newly created challenging textured object sequences dataset in Section 4.2.3.

4.1 Implementation Details

Our neural network implementation was based on the PoseNet implementation of [1] for TensorFlow [2]. We train our models using the ADAM optimizer [13] with a learning rate of 0.0001 and parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The models are trained for $75 - 100$ epochs which requires approximately 14 hours on an Nvidia GeForce GTX 1080Ti GPU. Our network implementation is available at [4].

4.2 Evaluation Results

4.2.1 Loss function comparison

In Figure 4 we evaluate the training procedure of the network depending on the choice of the loss function from Section 3.2. We present the angular loss as well the position loss. It can be seen that the convergence of the network is faster when using the ADD error $\mathscr{L}_{ADD}$ as loss function compared to the weighted euclidean distance loss $\mathscr{L}_{PoseNet}$. The $\mathscr{L}_{ADD}$ loss function however does not allow the angular error to converge since it focuses on minimizing the position error which appears to have a more direct influence on the loss. On the other hand, the $\mathscr{L}_{PoseNet}$ loss with a carefully selected weighting factor γ (we set $\gamma = 100$ in our implementation) allows both the angular error and the position error to converge smoothly. The final verdict is that the $\mathscr{L}_{ADD}$ loss can be used to accelerate the network training at an early stage, but the $\mathscr{L}_{PoseNet}$ still has to be used for the training to converge.

4.2.2 *LINEMOD* dataset evaluation

In Table 1 we provide an evaluation of our approach in the well known object pose estimation benchmark dataset *LINEMOD* [9]. The evaluation metric used is the commonly used ADD error. The error measures the average distance of the projected model vertices $\mathscr{V}$ using the estimated object pose $\hat{\boldsymbol{R}}_{co}, \hat{\boldsymbol{o}}_c$ and the real object pose $\boldsymbol{R}_{co}, \boldsymbol{o}_c$. This error is then compared to the diameter of the object and if it is under a certain percentage of the diameter ($10\%, 30\%$ thresholds in Table 1) then it is considered a correct pose estimate for that object. We provide the percentage of correct pose estimations per object using this metric. For comparison, we add the performance of the state-of-the-art SSD-6D approach [10] as provided in [24]. We choose to compare to SSD-6D for fairness since this is the only other approach using only synthetic data for training as opposed to many other using real images from the *LINEMOD* dataset for training. Furthermore, SSD-6D uses an additional pose refinement step after the initial pose estimation but we compare our results on the unrefined poses in order to have a direct comparison of the estimate that is output directly from the neural networks. The refinement step can be applied to any approach once there is an initial pose estimate and can also be seen as a first object frame-to-frame tracking step, thus we will not consider it here.

The results on Table 1 show that our approach using the pencil filter as input (OURS Pencil) clearly outperforms the one of SSD-6D [10] on the *LINEMOD* dataset on all three ADD error thresholds. Most importantly, there is a large difference in the 10% threshold meaning that our network is capable of directly providing very accurate poses at a much higher rate. When the threshold is increased the gap in performance between the two approaches is decreasing, with our approach still being superior. This can be due to some pose outliers in our system which can be credited to the absence of explicit training of the network to detect the object in the image. It is also worth noting that for our approach two of the worst performing objects were the cam and the eggbox for which the provided mesh is of lower quality compared to the rest of the objects. This indicates that our pose estimator is more dependant on the quality of the object meshes and that the results could probably be improved further by having higher quality 3D scans of the tested objects. It should be mentioned that although our method clearly outperforms SSD-6D [10] which also uses training on synthetic data, it still performs worse than approaches that train on real images such as BB8 [17]

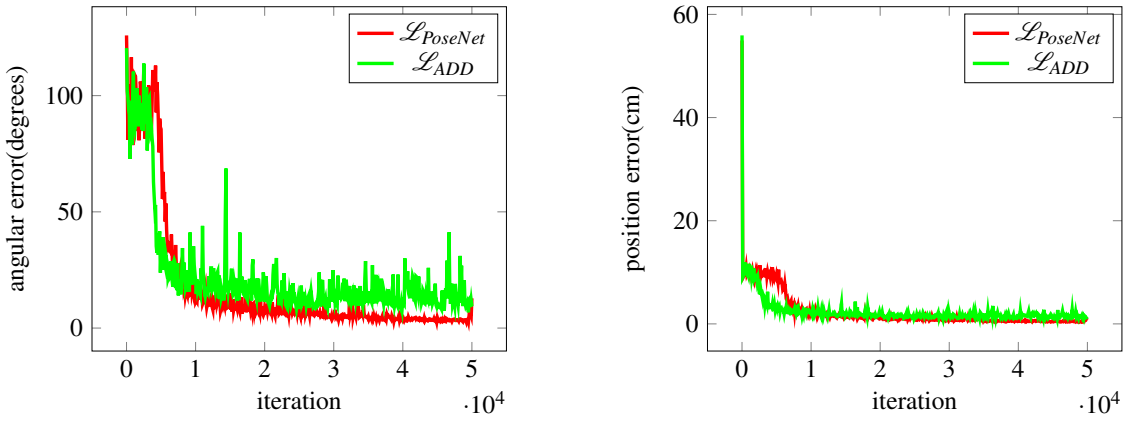

Figure 4: Comparison of network training with the two proposed loss functions $\mathscr{L}_{PoseNet}$, $\mathscr{L}_{ADD}$

which reports 43.6% accuracy on the 10% threshold on unrefined poses.

Threshold	10%			30%		
Object	[10]	OURS Pencil	OURS RGB	[10]	OURS Pencil	OURS RGB
Ape	0	**4.37**	0	5.62	**27.93**	0.08
Benchvise	0.18	**21.74**	5.76	2.07	**61.36**	36.73
Cam	0.41	**1.25**	0.16	**34.52**	7.58	1.5
Can	1.35	**2.09**	0	**61.43**	27.78	0.33
Cat	0.51	**2.54**	0	**36.87**	24.53	1.87
Driller	2.58	12.46	**12.97**	**56.01**	49.87	54.33
Duck	0	**4.78**	1.83	5.56	**29.05**	17.63
Eggbox	**8.9**	1.43	0	**24.61**	13.57	0
Glue	0	**7.38**	0.49	14.18	**41.92**	7.05
Holepuncher	0.30	**3.88**	2.26	18.23	23.38	**24.19**
Iron	8.86	38.22	**39.18**	59.26	87.66	**89.48**
Lamp	8.20	**27.35**	0.32	57.64	**71.66**	5.95
Phone	0.18	**5.39**	0	**35.55**	30.67	3.94
Average	2.42	**10.22**	4.84	31.65	**38.22**	18.69

Table 1: Comparison of our approach to SSD-6D [10] using the ADD metric on unrefined poses. For our approach, we report results using one-channel pencil images as well as RGB images for training and evaluation

We also provide results to compare our approach using the pencil filter for training and forward estimation in our network against using RGB images(OURS RGB in Table 1) in the same manner. The results clearly show that the pencil filter outperforms the use of RGB images for training on synthetic images and transferring to real ones. On most of the tested objects the pencil filter has a significantly higher score than the RGB images while also being computationally more efficient due to the reduced input size (single channel image vs. 3 channels). We observed that for some objects with very distinctive colours (e.g. the Iron) there can be a slight benefit in the use of RGB images. However, for other objects like the Duck the RGB trained network was often fooled by a second object of similar colour in the image which made the output very unstable and inferior to the pencil. For white objects (Lamp, Can) and for objects with strong edge and contour information (Benchvise) the pencil filter performed again better.

4.2.3 Textured object evaluation

The *LINEMOD* dataset consists mainly of completely textureless objects. However,for textured objects other methods for initial pose estimation can be used such as matching of edge features (e.g. ORB [22]). Therefore, we also compare our learning based method against the ORB pose initializer of [18, 19] on Table 2. The ORB initializer uses a set of N rendered images of different object views as references for feature matching. We perform the comparison on 4 textured objects on our own recorded sequences with ground truth obtained from the augmented things 3D object tracker [18, 19]. This allows us to have poses from all possible angles of the object and even poses where the object is handheld in contrast to the marker based ground truth of existing datasets that confine the object pose. Additionally, the ground truth pose quality is visibly very high which can be verified from the provided data. Furthermore, the dataset is made challenging by lighting variations and reflections on the objects, discrepancies between the objects and their models (e.g. missing components, differently connected cables) and interactions with partial occlusions of the objects. The used objects as well as their models are presented in Figure 5. The sequences and object models are available for download at [3].

We used again the ADD error metric for the comparison on the same thresholds as in Table 1. For fairness, we chose a number of $N = 64$ poses for the ORB Initializer since in that case the runtime for pose estimation per image is approximately the same as our network. In [19] a refinement step with KLT feature matching is applied after the initial pose estimation from ORB in order to initiate the tracking. We will compare the unrefined poses here, however the refinement step is applicable to both the ORB Initializer and our proposed CNN-based approach. The results on Table 2 confirm that our approach is highly suitable for pose estimation of textured objects under challenging conditions as it clearly outperforms the more traditional multiple reference image ORB edge feature based approach. There is a significant gain in the rate of highly accurate poses (under 10% threshold) but also for a higher threshold (30%). This indicates that the ORB initializer has a much higher rate of complete failures in pose estimation compared to our CNN-based system.

Threshold	10%		30%	
Object	[19]	OURS Pencil	[19]	OURS Pencil
Dragon	12.96	**15.95**	25.78	**71.75**
Totem	**0.58**	0.47	4.24	**15.80**
Controller	5.83	**6.61**	13.61	**49.67**
Sewing Machine	2.90	**29.21**	24.56	**80.08**
Average	5.56	**13.06**	17.04	**54.32**

Table 2: Comparison of our approach to the ORB Initializer of [19] on textured objects using the ADD metric

Finally, regarding the complexity of the proposed approach, our network runs at $\approx$ 8fps on our previously described device, which is comparable to the 10fps reported for SSD-6D [10].

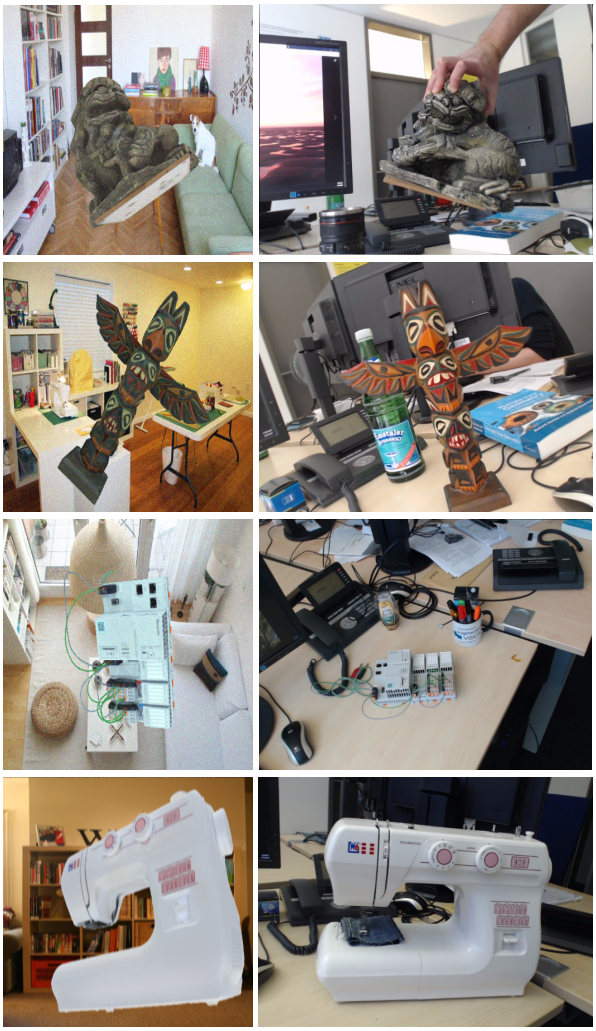

Figure 5: The objects used in our new textured object dataset for pose estimation. Images of the rendered object models on the left column and real images of the dataset on the right column

5 Conclusion

We presented an end-to-end learning based approach for 6DoF object pose estimation using only synthetic images for training. In contrast to previous approaches, we focus more on the correct data conditioning for training to successfully transfer our trained models to real images. Using the pencil filter image preprocessing we shift the learning from the more unstable color information to edge and contour features of the objects. We applied the PoseNet CNN architecture to the object pose estimation problem and outperformed the state-of-the-art in the *LINEMOD* benchmark dataset. In additional experiments we verified the suitability of the pencil filter for the synthetic to real problem against the use of RGB images. Finally, a new challenging dataset for object pose estimation of textured objects was introduced in order to show that our learning approach also outperforms the more traditional texture feature matching approaches. Possible future work includes coupling the pose estimation with an object detection step to support multiple objects as well as using the recovered object poses in a SLAM context.

Acknowledgments

This work has been partially funded by the Federal Ministry of Education and Research of the Federal Republic of Germany as part of the research projects PROWILAN and BeGreifen (Grant numbers 16KIS0243K and 16SV7525K).

References

[1] https://github.com/kentsommer/tensorflow-posenet.
[2] https://www.tensorflow.org/.
[3] https://av.dfki.de/members/rambach/.
[4] https://gitlab.com/matthewd1993/posenet.
[5] https://www.vuforia.com/.
[6] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pp. 343–351, 2016.
[7] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. 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 (CVPR)*, pp. 3364–3372, 2016.
[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
[9] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision (ACCV)*, pp. 548–562. Springer, 2012.
[10] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1521–1529, 2017.
[11] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 3, p. 8, 2017.
[12] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision (CVPR)*, pp. 2938–2946, 2015.
[13] D. Kingma and J. L. Ba. Adam: A Method for Stochastic Optimization. 2015.
[14] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA Objects: Fine Pose Estimation. *ICCV*, 2013.
[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.
[16] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.
[17] 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. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[18] J. Rambach, A. Pagani, M. Schneider, O. Artemenko, and D. Stricker. 6DoF Object Tracking based on 3D Scans for Augmented Reality Remote Live Support. *Computers*, 7(1):6, 2018.
[19] J. Rambach, A. Pagani, and D. Stricker. [POSTER] Augmented Things: Enhancing AR Applications leveraging the Internet of Things and Universal 3D Object Tracking. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 103–108. IEEE, 2017.
[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.
[21] A. Rozantsev, M. Salzmann, and P. Fua. Beyond sharing weights for deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pp. 2564–2571. IEEE, 2011.
[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions.
[24] B. Tekin, S. Sinha, and P. Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. *arXiv preprint arXiv:1711.08848*, 2017.