Real-Time Object Pose Estimation with Pose Interpreter Networks

Jimmy Wu[1], Bolei Zhou[1], Rebecca Russell[2], Vincent Kee[2], Syler Wagner[3], Mitchell Hebert[2], Antonio Torralba[1], and David M.S. Johnson[3]

***Abstract*— In this work, we introduce pose interpreter networks for 6-DoF object pose estimation. In contrast to other CNN-based approaches to pose estimation that require expensively annotated object pose data, our pose interpreter network is trained entirely on synthetic pose data. We use object masks as an intermediate representation to bridge real and synthetic. We show that when combined with a segmentation model trained on RGB images, our synthetically trained pose interpreter network is able to generalize to real data. Our end-to-end system for object pose estimation runs in real-time (20 Hz) on live RGB data, without using depth information or ICP refinement.**

I. Introduction

Object pose estimation is an important task relevant to many applications in robotics, such as robotic object manipulation and warehouse automation. In the past, 6-DoF object pose estimation has been tackled using template matching between 3D models and images [1], which uses local features such as SIFT [2] to recover the pose of highly textured objects. Recently, there has been growing interest in object manipulation as a result of the Amazon Picking Challenge [3], leading to the introduction of a number of different approaches for 6-DoF object pose estimation, specifically in the competition setting [4], [5], [6], [7], [8]. Many of these approaches, along with other recent works such as PoseCNN [9], SSD-6D [10], and BB8 [11], use deep convolutional neural networks (CNNs) to provide real-time, accurate pose estimation of known objects in cluttered scenes.

CNN-based pose estimation techniques enable significant improvements in the accuracy of object detection and pose estimation. However, these approaches usually require a large amount of training data containing objects of interest annotated with precise 6-DoF poses. Object poses are expensive to annotate and were often hand annotated in the past [4], [12]. More recently, automatic annotation methods have been proposed using motion capture [13] or 3D scene reconstruction [14], [15], but these methods still require significant human labor and are not able to generate significant variability in pose since objects must remain stationary during data capture.

[1]JW, BZ, and AT are with the MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA

[2]RR, VK, MH are with the Charles Stark Draper Laboratory, Cambridge, MA, USA

[3]SW and DMSJ are with Dexai Robotics, Boston, MA, USA

Corresponding authors: Jimmy Wu jimmywu@alum.mit.edu and David M.S. Johnson dave@dexai.com

Datasets, code, and pretrained models are available at https://github.com/jimmyyhwu/pose-interpreter-networks

Fig. 1: Our end-to-end network takes in an RGB image and outputs 6-DoF object poses for all recognized objects in the image.

To address this issue, we propose a novel pose estimation approach that leverages synthetic pose data. Our approach decouples the object pose estimation task into two cascaded components: a segmentation network and a pose interpreter network. Given an RGB image, the segmentation network first generates object segmentation masks, which are then fed into the pose interpreter network for pose estimation.

Pose interpreter networks perform 6-DoF object pose estimation on object segmentation masks and are trained entirely using synthetic pose data, thus obviating the need for expensive annotation of object poses. Using a rendering engine, we cheaply acquire large quantities of synthetic object segmentation masks and their 6-DoF pose ground truth, covering the full space of object poses. After training, our pose interpreter networks are able to estimate object pose accurately given only the segmentation mask of the object. The overall object pose estimation pipeline including segmentation runs in real-time with no postprocessing steps such as ICP refinement or smoothing.

The main contributions of this work are: (1) an end-to-end approach for real-time 6-DoF object pose estimation from RGB images, (2) a pose interpreter network for 6-DoF pose estimation in both real and synthetic images, which we train entirely on synthetic data, and (3) a novel loss function for regressing 6-DoF object pose. In the following sections, we discuss related work in Section II, describe our technical approach in Section III, present experimental results in Section IV, and conclude in Section V.

II. Related Work

Prior work in object pose estimation from RGB images include template matching approaches [16], [17], [18], [19] and parts-based models [20], [21], which work well for highly textured objects. Feature-based methods match features in images with corresponding parts of 3D models [2], [22], [1],

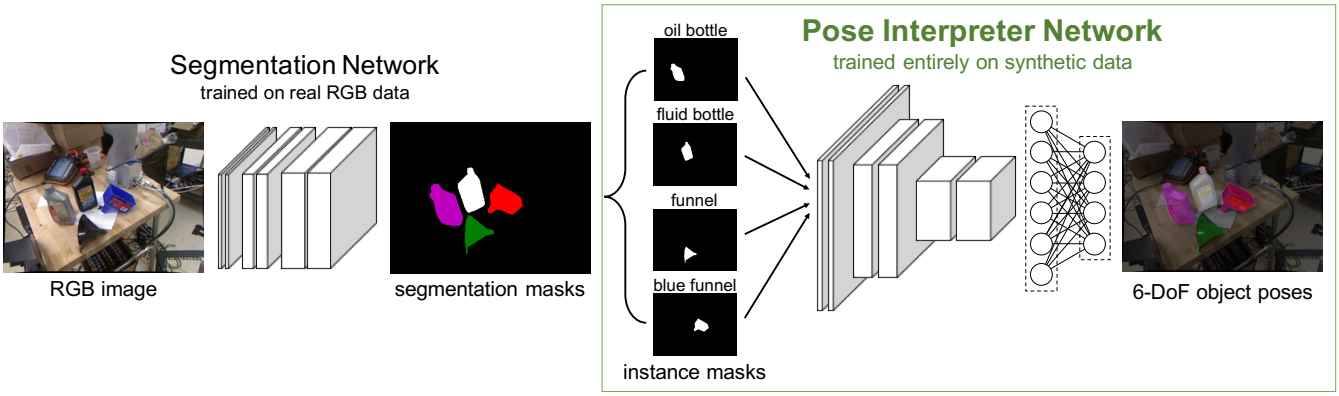

Fig. 2: **Full end-to-end architecture for object pose estimation on RGB images.** We use a segmentation network to extract instance masks labeled with object class, which serve as input to the pose interpreter network. The pose interpreter network operates on single object instances of known object classes and is trained entirely on synthetic object masks. The system makes one forward pass through the segmentation network for each image. Then, for each object instance, it makes one forward pass through the pose interpreter network to predict the object's pose. During evaluation, the two component networks are combined into a single end-to-end neural network.

[23]. For RGB-D data, pose estimation has traditionally used variants of the iterative closest point (ICP) algorithm [24], [25]. More recent works have used feature matching on 3D data [26], [27], [28], [29], [30] or probabilistic methods [31], [32].

With the recent successes of object recognition [33], [34], object detection [35], [36], and segmentation [37], [38] in 2D images, many works have extended or incorporated these methods in 6-DoF pose estimation [39], [40], [10], [11], including end-to-end systems for robotic manipulation [13], [4], [41]. In contrast to these approaches, which often require expensively obtaining lots of annotated training data, we focus on the use of cheaply acquired synthetic data to train our pose interpreter network, and show that it generalizes well to real RGB images.

Perhaps most closely related to our work is PoseCNN [9], a well-known CNN for 6-DoF object pose estimation. We emphasize that our pose interpreter network is trained entirely on synthetic data, whereas PoseCNN uses a large annotated pose dataset augmented with synthetic images. Additionally, our system runs in real-time and uses neural network forward passes to directly output pose estimates, without any further postprocessing.

Our work, which use CNNs for regression, is also related to [42] and [43], known for successfully demonstrating camera pose regression with CNNs. Additionally, the use of rendering software to cheaply acquire large quantities of synthetic training images for training deep networks has been proposed by several previous works [44], [45], [46]. In particular, [45] also uses an intermediate representation to bridge synthetic data and real data for 3D object structure recovery.

III. Technical Approach

Our approach to object pose estimation consists of a two step process. We first use a segmentation network to generate object instance masks. The masks are then passed individually through the pose interpreter network, which outputs a 6-DoF pose estimate for each object. While we train the segmentation model on real RGB images, our pose interpreter network is trained entirely on synthetic data. We first describe our segmentation network in Section III-A. Then, we describe the pose interpreter network in Sections III-B through III-D.

A. Segmentation Network

We use a dilated residual network (DRN) [38] trained for semantic segmentation as the first component of our end-to-end system. The network takes in real RGB images and outputs segmentation labels, which are converted into binary instance masks with associated object classes and fed into the subsequent pose estimation network.

In contrast to regular residual networks [47], which use subsampling to increase receptive field size at the cost of spatial acuity, DRNs use dilated convolutions, which preserve spatial resolution while maintaining high receptive fields. As a result, these networks are particularly well suited for dense prediction tasks such as semantic segmentation. Compared to other architectures for semantic segmentation such as SegNet [48] or DeepLab [49], we observed that DRNs trained on our RGB image dataset generated higher fidelity segmentations with fewer false positives.

While our segmentation training data is not synthetically generated, we note that compared to CNNs for pose estimation, CNNs for segmentation can use cheaper data acquisition techniques and much more aggressive data augmentation. We also note that our use of a semantic segmentation model for instance segmentation assumes that there is at most one instance of every object class. However, our system can be adapted to handle multiple instances by simply swapping out the semantic segmentation component with an instance segmentation model such as Mask R-CNN [50].

B. Object Pose Representation

We represent the pose of an object by its position $\mathbf{p} = (x, y, z)$ and orientation $\mathbf{q} = (q_0, q_x, q_y, q_z)$, which are translations and rotations relative to the camera coordinate frame. Any given rotation can have multiple equivalent forms, and we found it crucial to enforce that only a single unique form is valid. For example, the axis-angle rotation (ω, θ) is equivalent to $(-\omega, -\theta)$, a rotation of $-\theta$ about the axis $-\omega$. These two forms resolve to a unique form in unit quaternion. The rotation (ω, θ) is also equivalent to $(-\omega, 2\pi - \theta)$, which resolves to $-q$ in unit quaternion. We resolved this equivalence of q and $-q$, known as double cover [51], by requiring that the real component of the quaternion q_0 be nonnegative, equivalent to constraining the rotation angle θ to be in the range $(-\pi, \pi)$.

C. Pose Interpreter Network

The pose interpreter network operates on single object instances of known object classes, and is trained entirely on synthetic data. The network follows a simple CNN architecture consisting of a ResNet-18 [47] feature extractor followed by a multilayer perceptron, as illustrated in Fig. 2. We removed the global average pooling layer from the feature extractor to preserve spatial information in the feature maps.

The multilayer perception is composed of one fully connected layer with 256 nodes, followed by two parallel branches corresponding to position and orientation, respectively. Each branch consists of another single fully connected layer, with a separate set of outputs for each object class. We train our pose interpreter network with five object classes, so the position branch has 15 outputs, while the orientation branch has 20.

The quaternion orientation outputs are normalized to unit magnitude. We found this normalization to be crucial, as it is difficult to directly regress unit quaternion values. By normalizing the outputs, we are instead having our network predict the relative weights of the four quaternion components.

D. Point Cloud L1 Loss for Pose Prediction

We propose a new loss function for object pose prediction, the Point Cloud L1 Loss. We compare the proposed loss with several baseline loss functions and show our experimental results in Section IV-D.

The simplest baseline is L1 loss on the target and output poses, with a weighting constant α to balance the position and orientation terms:

$$L_1 = |\hat{\mathbf{p}} - \mathbf{p}| + \alpha \, |\hat{\mathbf{q}} - \mathbf{q}| \tag{1}$$

We also consider a modified version of L_1 in which we replaced the orientation loss with one proposed by Xiang et al. in PoseCNN [9], which approximates the minimum distance between the target and predicted orientations:

$$L_2 = |\hat{\mathbf{p}} - \mathbf{p}| + \alpha \, (1 - \langle \hat{\mathbf{q}}, \mathbf{q} \rangle) \tag{2}$$

Next, we propose a new loss function, denoted L_3 below, that operates entirely in the 3D space rather than the quaternion space. Using the 3D models of objects in our synthetic dataset, we generate point clouds representing each object. Given a target pose and output pose for an object, we transform the object's point cloud using both the target and output poses and compare the two transformed point clouds. We compute an L1 loss between pairs of corresponding points as follows:

$$L_3 = \sum_{i=1}^{m} \left| H(\hat{\mathbf{p}}, \hat{\mathbf{q}}) \mathbf{x}^{(i)} - H(\mathbf{p}, \mathbf{q}) \mathbf{x}^{(i)} \right| \tag{3}$$

where each $x^{(i)}$ is one of m points in the object point cloud, and the function H transforms an object pose $(\mathbf{p}, \mathbf{q})$ into the equivalent transformation matrix. This loss function directly compares points clouds in 3D space and does not require tuning an additional hyperparameter α, as in L_1 and L_2, to balance the position and orientation loss components.

As discussed in Section III-B, we require that the first component q_0 of unit quaternion orientations be nonnegative. In practice, we found that adding an additional term to penalize predictions with negative q_0 helped with convergence. Thus, we favor the use of loss L_4 below, a variant of our proposed loss L_3.

$$L_4 = \max(-q_0, 0) + \sum_{i=1}^{m} \left| H(\hat{\mathbf{p}}, \hat{\mathbf{q}}) \mathbf{x}^{(i)} - H(\mathbf{p}, \mathbf{q}) \mathbf{x}^{(i)} \right| \tag{4}$$

IV. Experiments

We present our experimental results in the following sections. Section IV-A describes in detail the two datasets we used. Section IV-B describes the performance of our segmentation network, which serves as the first component of our end-to-end system. Sections IV-C, IV-D, and IV-E describe the results of our experiments with the pose interpreter network on synthetic data. In Sections IV-F and IV-G, we combine the segmentation network and pose interpreter network into our full end-to-end system and evaluate its performance on real RGB data. Finally, in Sections IV-H and IV-I, we discuss limitations of our approach as well as experiments to address some of those limitations.

A. Datasets

As shown in Fig. 2, we use separate datasets for training the segmentation network and the pose interpreter network. The segmentation network is trained on real RGB images, while the pose interpreter network is trained entirely on synthetic data. We describe each of the two datasets in detail below.

Oil Change Dataset. We use an extension of the dataset used in the SegICP system [13], which we will refer to as the Oil Change dataset. The dataset consists of indoor scenes with 10 categories of densely annotated objects relevant to an automotive oil change, such as oil bottles, funnels, and engines. Images were captured with one of three sensor types (Microsoft Kinect1, Microsoft Kinect2, or Asus Xtion Pro

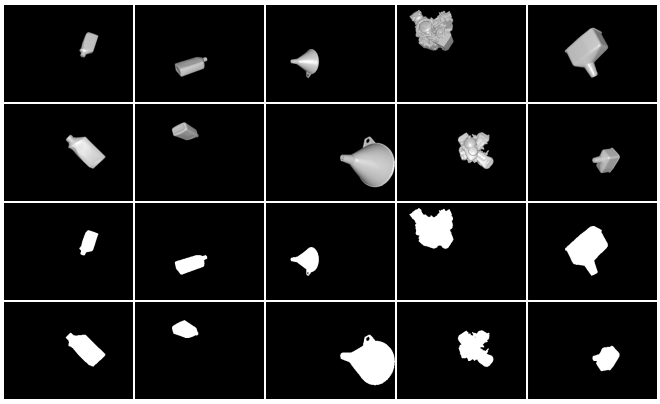

Fig. 3: We use a synthetic image dataset of five object classes to train our pose interpreter network. The dataset contains both synthetic object images (top two rows) and synthetic mask images (bottom two rows).

Live) and were automatically annotated with object poses and pixelwise instance masks using either the motion capture setup described in [13] or the LabelFusion [15] pipeline.

In total, there are 7,879 images used for training and another held out 1,950 test images used for evaluation. For our segmentation model, we used the full training and testing splits. For evaluating our end-to-end system, we used a subset of the testing images corresponding to one of the Kinect1 cameras. This is because unlike in most 2D image recognition tasks, the task of recovering 3D pose from a 2D image depends on the intrinsics of the camera with which the image was taken, so we cannot use images from several different cameras. As a result, we focus on our most portable Kinect1 camera for our end-to-end system. The subset we used for evaluating the end-to-end system consisted of 229 test images with 683 object instances from 5 categories. These same 5 object categories were used for our synthetic image dataset, described in further detail below.

Synthetic Image Dataset. We train and evaluate our pose interpreter network entirely on a synthesized dataset of object images and mask images, examples of which are shown in Fig. 3. We load 3D model files for five object categories from the Oil Change dataset and use the Blender rendering software to render the objects in random poses. As previously discussed, we focus on one Kinect1 camera for evaluating our end-to-end system. In order to ensure compatibility with the evaluation images, we calibrated our camera and used the same camera intrinsics in the rendering pipeline.

While it is possible to render images on the fly during neural network training, we found that the rendering time far exceeded the network training step time. We thus rendered and saved a total of 3.2 million training images and 3,200 testing images, evenly spanning the five object classes. We note that much less training data is actually needed, and in Section IV-E, we investigate the effect of training data quantity on the performance of the pose interpreter network.

TABLE I: Performance of the pose interpreter network trained and evaluated on synthetic data.

model type	pos. error (cm)	ori. error (deg)
object	1.12	8.93
mask	1.43	14.83

B. Semantic Segmentation with DRN

Our segmentation model is a DRN-D-22 trained on the Oil Change dataset with batch size 16, learning rate 0.001, momentum 0.99, and weight decay 0.0001. The dataset was annotated with 11 classes, corresponding to the 10 object classes and an additional background class. We trained for 900 epochs and used aggressive data augmentation to improve generalization, including random scaling, random rotations, random cropping, and gamma jittering. On the Oil Change test images, our final model achieves a pixelwise accuracy of 99.82% and a mean IoU of 0.9650 across the 10 object classes.

C. Pose Interpreter Network

As described in Section IV-A and Fig. 3, we used a synthetic dataset containing both object images and mask images. We use the synthetic object images for further experiments with the pose interpreter network. The synthetic mask images are used to train the pose interpreter network for use in our full end-to-end system.

In Table I, we show the performance of our pose interpreter network after training and evaluating on the synthetic dataset for both object images and mask images. We use a batch size of 32, weight decay of 0.0001, and train for 21 epochs with an initial learning rate of 0.01, which we decay by a factor of 10 after 7 epochs and 14 epochs. A single network is trained to handle all five object classes with separate output heads per class. By design, the network operates on instance masks of known object class, so for each training example we are able to select the appropriate outputs to compute the loss on.

As one might expect, the model trained on synthetic mask images does not perform as well as the model trained on synthetic object images, as there is less information and more ambiguity when given only a binary mask of an object. However, the model trained on synthetic mask images can be directly used on real RGB data when combined with a segmentation model. We describe the results of this combination applied to real RGB data in Section IV-F.

D. Pose Prediction Loss Functions

We considered four loss functions and evaluated them on a subset of the synthetic dataset consisting of synthetic blue funnel object images. We used an initial learning rate of 0.01 for all four training runs, which was decayed after observing the validation performance plateau (10 epochs for L_1, 30 epochs for L_2, and 15 epochs for L_3 and L_4). The weighting term α was set to 1 for L_1 and L_2. We show in Table II the best performance for each network after 30 epochs (45 for L_2 due to slower convergence).

TABLE II: Performance of pose interpreter network trained with various loss functions on synthetic blue funnel object images. The loss functions are described in Section III-D.

loss function	pos. error (cm)	ori. error (deg)
L_1	0.75	5.79
L_2	1.38	13.76
L_3	0.58	6.25
L_4	0.50	6.01

TABLE III: Performance of pose interpreter network trained on synthetic object images from the blue funnel object class. We vary the number of training images used to see the effect on performance.

images	pos. error (cm)	ori. error (deg)
12,800	3.16	104.6
25,600	1.86	48.15
51,200	1.50	11.85
102,400	1.06	10.99
640,000	1.33	11.02

We observed that the network trained using L_1 attains comparable performance to the point cloud loss functions L_3 and L_4. However, the weighting term α for L_1 must be tuned to balance the position and orientation errors. Lowering the position error by adjusting the weighting term would raise the orientation error, and vice versa. By contrast, our proposed point cloud loss function naturally balances the position and orientation errors by computing the loss in the 3D point space, so there is no weighting term to tune.

E. Synthetic Training Data Quantity

We used 3.2 million training images to train our pose interpreter network. Here we investigate whether comparable performance can be attained with fewer training images, and whether the training scales well with more object classes. We run two sets of experiments using subsets of our synthetic dataset, the first on synthetic object images of a single object class (blue funnel) as shown in Table III, and the second on synthetic object images of all five object classes as shown in Table IV. We used a learning rate of 0.01 with no decay for all experiments, and show the best performance attained after an equal number of training iterations (400k for single object class, 700k for five object classes).

The results in Table III indicate that when training the pose interpreter network on a single object class, using more than 100k training images does not yield further gains. When training the pose interpreter network on multiple object classes instead, the results in Table IV indicate that 25k images is the cutoff point. We interpret this as evidence that the network is learning some shared knowledge between the different object classes, as only 25k images per class are needed rather than 100k. Although more training images and training iterations are required for multiple object classes, the required quantities are far from proportional to the number of classes.

TABLE IV: Performance of pose interpreter network trained on synthetic object images from all five object classes. We vary the number of training images used per object class.

images / class	pos. error (cm)	ori. error (deg)
6,400	4.91	82.45
12,800	3.36	27.06
25,600	3.21	22.86
51,200	2.94	25.16
640,000	2.96	24.26

TABLE V: End-to-end performance of our object pose estimation system on real RGB images. We also show the performance for SegICP evaluated on the same test images.

		ours	SegICP
pos. error (cm)	mean	3.76	2.09
	median	3.23	1.32
ori. error (deg)	mean	19.64	68.93
	median	6.17	55.63
success (%)		71.01	42.08

F. Object Pose Estimation on Real RGB Images

We evaluate our end-to-end object pose estimation system on real RGB images from the Oil Change dataset test split. The end-to-end system is composed of a DRN segmentation model followed by a object mask pose interpreter network. As described in Section IV-A, we evaluate only on the images taken with a specific Kinect1 sensor, consisting of 229 images with 683 total object instances.

We show the performance of our end-to-end system evaluated on our Oil Change dataset in Table V. Following the convention used in [4] and [13], a successful pose estimate is defined as < 5 cm position error and $< 15^\circ$ orientation error. Histograms of the position and orientation errors in Fig. 4 show the distribution of errors relative to the success cutoffs, marked in red.

We also show in Table V a comparison with SegICP, which reported a success rate of 77% in [13]. However, that success rate was evaluated on an older version of our dataset. For a fair comparison, we evaluated the performance of SegICP on the updated test set, which proved to be more challenging than the set used in the SegICP paper. The current test set includes many object instances that lie in close proximity to other objects. This means that mistakes in the segmentation can result in the inclusion of erroneous points from neighboring objects, leading to poor ICP performance in SegICP. Furthermore, although SegICP attains better position errors than our system, we would like to emphasize that SegICP uses ICP to iteratively refine predictions, whereas our system outputs a one-shot prediction per object with no postprocessing or refinement.

G. Object Pose Estimation on Live RGB Data

In order to verify that our approach does not suffer from overfitting or dataset bias, we tested the efficacy of our approach on live RGB data. Our experimental setup used an Ubuntu 16 workstation equipped with a dedicated

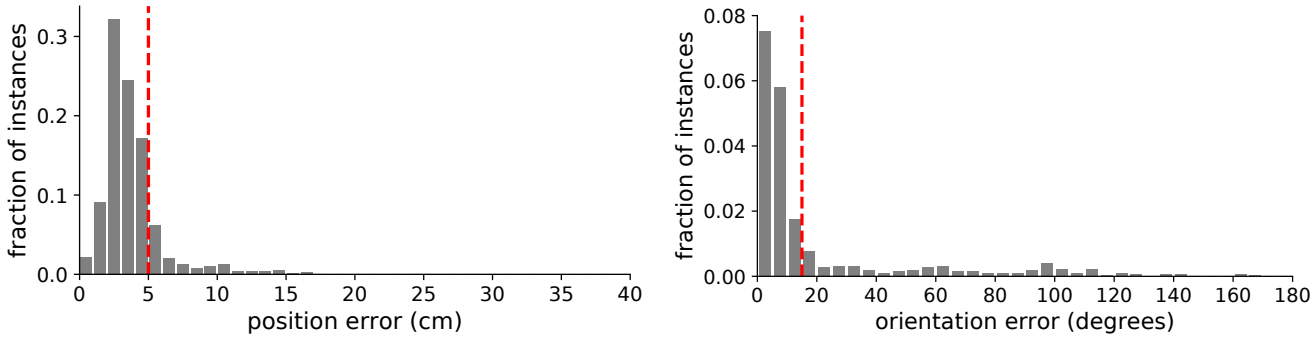

Fig. 4: Histograms showing position errors and orientation errors of our end-to-end pose estimation system on real RGB data. The success criterion of < 5 cm position error and $< 15^\circ$ orientation error is indicated by the dotted red lines. Our system attains a success rate of 71.01%.

Nvidia Titan Xp graphics card. A ROS node processes live images from our Kinect1 and runs them through trained PyTorch models to generate 6-DoF pose predictions for each recognized object. We benchmark the processing time per frame by averaging over 50 consecutive frames. The DRN forward pass takes 28.9 ms on 640×480 RGB images. The pose interpreter network inputs 320×240 binary instance masks and requires a 3.6 ms forward pass per detected object instance. Visualization adds an additional overhead of about 10 ms. Overall, the end-to-end system takes about 32–47 ms per frame depending on the number of objects in the image. In the accompanying video supplement, we demonstrate our system performing pose estimation in real-time on live data.

H. Limitations

The main limitations of our approach are sensitivities to both occlusion and segmentation failures. Our pose interpreter network is trained entirely on synthetic data, and thus only generalizes well when input object masks resemble the rendered masks seen during training. Hence, the performance of our end-to-end system is closely tied to the quality of the segmentation model.

We quantitatively evaluate the effect of occlusions on the performance of the object mask pose interpreter network by introducing circular occlusions of various sizes centered at points on mask boundaries, as shown in Fig. 5. We show in Fig. 6 how the position and orientation errors of the network relate to the amount of occlusion introduced. The occlusion amount measures the fraction of the original mask area that has been artificially occluded. The results (trained without occlusion) indicate high sensitivity to occlusion, particularly for orientation prediction at low occlusion amounts. In Section IV-I, we investigate whether training with artificially occluded mask images improves the robustness of the pose interpreter network.

Another limitation of our approach is the lack of additional information such as texture, color, or depth in our binary mask representation, which presents difficulties in situations where such information is needed to resolve ambiguities in the mask representation. However, extending to other domains such as depth would require either more realistic rendering or domain adaptation techniques. Here, we would like to emphasize that our object mask representation enables us to directly apply our pose interpreter network to real RGB data without any domain adaptation.

Fig. 5: Some examples of the artificially occluded mask images we generated. These images were used to quantify the pose interpreter network's sensitivity to occlusions. We also ran experiments, as described in Section IV-I, using occluded images as training data.

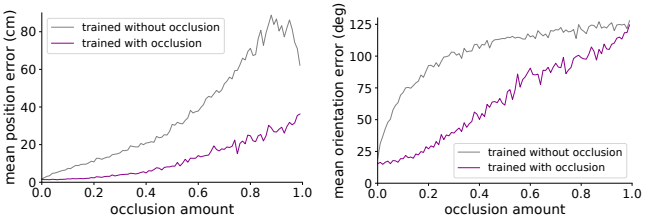

Fig. 6: We artificially introduce circular occlusions of various sizes, as shown in Fig. 5, and quantify how the performance of the object mask pose interpreter network varies with the amount of occlusion. We then train a model with occluded mask images, as detailed in Section IV-I, and compare the resulting performance with the baseline model trained without occlusion.

I. Training with Occlusion

As discussed in Section IV-H, our model is not robust to occlusions, particularly for orientation prediction. We investigate here whether we can improve the robustness of object mask pose interpreter networks by training on occluded mask images. Using the same occlusion scheme as in Section IV-H, we artificially occlude mask images during training and evaluate the resulting performance.

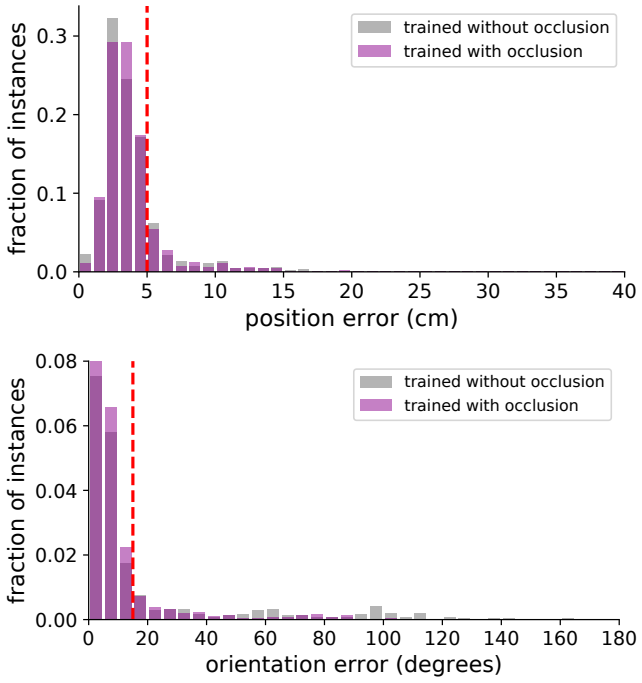

Fig. 7: Histograms showing position errors and orientation errors of the end-to-end system when the pose interpreter network is trained with and without occlusion. The end-to-end system is evaluated on real RGB data. For the model trained with occlusion, note that the orientation errors are much more concentrated in the low error region.

We show in Figs. 6 and 7 comparisons between the baseline mask model and a model trained with circular occlusions of maximum radius 24. Fig. 6 indicates that the pose interpreter network is more robust when trained with occlusion, especially for orientation prediction at low occlusion amounts. In Fig. 7, we show that when applied to the end-to-end system and evaluated on real RGB images, training with occlusion does not show meaningful improvement in position prediction, but does markedly improve orientation prediction, allowing the system to successfully address many examples that previously gave high orientation errors.

We note that introducing occlusions that cover too much of the original object mask will destroy the information present in the mask, so we experimented with different settings of the maximum occlusion radius. As shown in Table VI, we found 24 pixels to be optimal, and confirmed that training with occlusions that were too large resulted in worsened performance. We additionally found that larger occlusions generally resulted in longer training times since the task was more difficult. While 21 epochs was sufficient for the baseline model, we trained models with larger occlusions for over 100 epochs.

V. Conclusion

In this work, we present pose interpreter networks for real-time 6-DoF object pose estimation. Pose interpreter networks are trained entirely using cheaply rendered synthetic data,

TABLE VI: Performance of our end-to-end model when the pose interpreter network is trained with occlusion. We vary the maximum radius of the circular occlusions.

max occ. radius	pos. error (cm)		ori. error (deg)		success (%)
	mean	median	mean	median	
baseline	3.76	3.23	19.64	6.17	71.01
12 pixels	3.94	3.46	15.79	5.89	75.40
16 pixels	3.90	3.30	15.79	5.67	73.06
24 pixels	3.76	3.31	11.55	5.90	77.89
32 pixels	4.00	3.67	13.69	6.92	72.18
48 pixels	2.84	2.32	16.88	7.05	74.38
64 pixels	4.22	3.88	19.69	7.78	59.74

allowing us to avoid expensive annotation of large pose datasets. We use pose interpreter networks as part of an end-to-end system for pose estimation in real RGB images. The system consists of two steps: (1) a segmentation network to generate object instance masks, and (2) a pose interpreter network which takes in instance masks and outputs pose estimates. We use the object mask as a context-independent intermediate representation that allows the pose interpreter network, trained only on synthetic data, to also work on real data. Our end-to-end system runs in real-time on live RGB data, and does not use any filtering or postprocessing to refine its pose estimates.

Acknowledgement

We thank Lucas Manuelli, Russ Tedrake, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Scott Kuindersma for their insight and feedback.

References

[1] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006.

[2] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[3] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, 2016.

[4] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," 2017.

[5] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger *et al.*, "Team delfts robot winner of the amazon picking challenge 2016," in *Robot World Cup*. Springer, 2016, pp. 613–624.

[6] R. Jonschkowski, C. Eppner, S. Höfer, R. Martín-Martín, and O. Brock, "Probabilistic multi-class segmentation for the amazon picking challenge," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 1–7.

[7] M. Schwarz, A. Milan, C. Lenz, A. Munoz, A. S. Periyasamy, M. Schreiber, S. Schüller, and S. Behnke, "Nimbro picking: Versatile part handling for warehouse automation," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3032–3039.

[8] M. Schwarz, A. Milan, A. S. Periyasamy, and S. Behnke, "Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter," *The International Journal of Robotics Research*, p. 0278364917713117, 2016.

[9] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[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 *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[11] 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 *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016.

[13] J. M. Wong, V. Kee, T. Le, S. Wagner, G. L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. S. Johnson, J. Wu, B. Zhou, and A. Torralba, "Segicp: Integrated deep semantic segmentation and pose estimation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 5784–5789.

[14] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1817–1824.

[15] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, "A pipeline for generating ground truth labels for real rgbd data of cluttered scenes," *arXiv preprint arXiv:1707.04796*, 2017.

[16] 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*. Springer, 2012, pp. 548–562.

[17] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[18] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[19] D. G. Lowe, "Local feature view clustering for 3d object recognition," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[20] S. Savarese and L. Fei-Fei, "3d generic object categorization, localization and pose estimation," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[21] J. J. Lim, A. Khosla, and A. Torralba, "Fpm: Fine pose parts-based model with 3d cad models," in *European Conference on Computer Vision*. Springer, 2014, pp. 478–493.

[22] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[23] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-dof object pose from semantic keypoints," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2011–2018.

[24] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.

[25] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 2001, pp. 145–152.

[26] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[27] 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.

[28] 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.

[29] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T.-K. Kim, "Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation," *arXiv preprint arXiv:1607.02257*, 2016.

[30] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *CVPR*, 2017.

[31] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[32] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, and c. Rother, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[36] 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.

[37] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[38] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[39] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Aligning 3d models to rgb-d images of cluttered scenes," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 4731–4740.

[40] A. Bansal, B. Russell, and A. Gupta, "Marr Revisited: 2D-3D model alignment via surface normal prediction," in *CVPR*, 2016.

[41] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," 2018.

[42] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2938–2946.

[43] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[44] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[45] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, "Single image 3d interpreter network," in *European Conference on Computer Vision*. Springer, 2016, pp. 365–382.

[46] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, "MarrNet: 3D Shape Reconstruction via 2.5D Sketches," in *Advances In Neural Information Processing Systems*, 2017.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[48] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[49] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[50] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[51] S. L. Altmann, *Rotations, quaternions, and double groups*. Courier Corporation, 2005.