A. Appendix

In the following we provide implementation details and show qualitative results. The code will be open sourced upon publication.

A.1. Architecture

We closely follow [34] to design our Multi-Path (MP) encoder and decoders.

In our T-LESS experiments the encoder is symmetric to each of the decoders. The encoder consists of four convolutional layers with kernel size = 5, stride=2, ReLU activation and $\{128, 256, 512, 512\}$ filters respectively. It follows, a shared fully connected layer with $W_0 \in \mathcal{R}^{32768x128}$. The resulting codes are propagated to the corresponding decoders via $j \in 1, .., n$ fully connected layers $W_j \in \mathcal{R}^{128x32768}$. After reshaping, each decoder has four consecutive convolutional layers with kernel size = 5, stride=1, ReLU activation and $\{512, 512, 256, 128\}$ filters. Nearest neighbor upsampling is performed after each convolutional layer. The final reconstruction output is preceded by a sigmoid activation.

We initially expected that an encoder with up to hundred corresponding decoders would require a much larger architecture to be able to adequately encode the views of all training objects. Therefore, we first adopted the [4], with a Resnet101 backbone and a spatial pyramid pooling (SPP) layer. Although, the total training loss can be reduced with such a network, the downstream performance on encoding object poses of both, trained and untrained objects, does not increase compared to a more shallow encoder/decoder with 4-5 plain convolutional layers as used by [34]. As usual, multiple explanations are plausible. Shallow features could contain all necessary information for class-agnostic pose estimation. The lost symmetry in the encoder-decoder architecture could make an invertible compression more difficult to learn. A deeper encoder could simply overfit more to synthetic and generalize less to real data. As architecture search and explanation is not our main goal here, we simply stick with the more efficient shallow CNN.

A.2. Training Details

It is possible to train our architecture with up to 80 decoders on a single GPU. In this case, the encoder receives a joint batch size of 80, while each decoder has a batch size = 1. For single-GPU training on T-LESS with 18 decoders, we use a batch size = 4. We train for 300.000 iterations which takes about 48 hours on a single GPU. Multiple GPUs almost linearly speed up training due to the inherent parallelism in the architecture. Therefore, the batch and the decoders are divided into equal parts on all available GPUs. For the ModelNet experiments where we trained on 80 object instances, we used 4 GPUs with a total encoder batch size of 240 and decoder batch size of 4. Higher decoder batch sizes also stabilize the training.

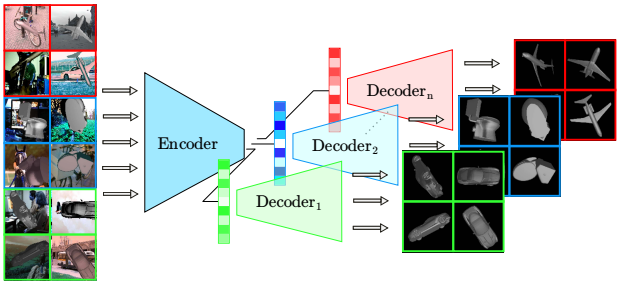

Figure 5: Our multi-path training process. **Left:** Joint augmented input batch with uniformly sampled SO(3) views of n object instances; **Right:** Individual reconstruction targets $\boldsymbol{x}_j$ for each decoder.

We use Xavier initialization and the Adam [24] optimizer with a learning rate of $10^{-4} \times b_{dec}$, with b_{dec} = decoder batch size. In our experiments we did not have to assign individual learning rates to the encoder and decoder but this could potentially accelerate the training.

A.3. Synthetic Data Generation

For each instance we render 8000 object views sampled randomly from SO(3) at a constant distance of 700mm. The resulting images are quadratically cropped and resized to $128 \times 128 \times 3$.

A.4. Augmentation Parameters

All geometric and color input augmentations besides the random lighting are applied online during training at uniform random strength (see Table 5). As background images we use Pascal VOC2012 [11]. One could argue that random background images during training are redundant where we use MaskRCNN. Training without backgrounds works, but in our experiments it does help with the generalization from synthetic to real data and is also beneficial when the MaskRCNN does not output perfect masks.

Table 5: Augmentation Parameters; Scale and translation is in relation to image shape

color	50% chance (30% per channel)	light (random position) & geometric	
add	$\mathcal{U}(-0.1, 0.1)$	ambient	0.4
contrast	$\mathcal{U}(0.4, 2.3)$	diffuse	$\mathcal{U}(0.7, 0.9)$
multiply	$\mathcal{U}(0.6, 1.4)$	specular	$\mathcal{U}(0.2, 0.4)$
invert		scale	$\mathcal{U}(0.8, 1.2)$
gaussian blur	$\sigma \sim \mathcal{U}(0.0, 1.2)$	translation	$\mathcal{U}(-0.15, 0.15)$

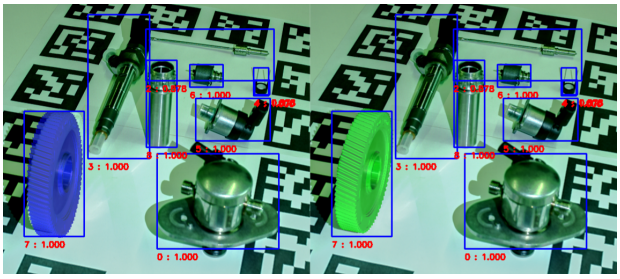

Figure 6: Left: MP-Encoder trained on T-LESS objects 1-18, and tested on a metallic, industrial object; Right: ICP-refined result

A.5. ICP Refinement

Optionally, the estimate is refined on depth data using a point-to-plane Iterative Closest Point (ICP) approach with adaptive thresholding of correspondences based on [5, 46]. The refinement is first applied in direction of the vector pointing from camera to the object where most of the RGB-based pose estimation errors stem from and then on the full 6D pose.

A.6. Results on metallic parts

We briefly tested the MP-Encoder trained on T-LESS objects on metallic objects from industry (Fig. 6).

A.7. Full T-LESS results

Table 3 shows the full T-LESS results on each object tested on all scene views of the Primesense test set. Figure 7 show qualitative results with and without refinement on different T-LESS test scenes. Results on objects sometimes differ because the AAE submission uses RetinaNet (0.73mAP@0.5) and the MP-Encoder uses MaskRCNN (0.68mAP@0.5) because it does not learn to distinguish object and background (see Table 2).

A.8. Runtime

The MaskRCNN with ResNet50 takes $\sim 150ms$ for all instances in a scene on a modern GPU. Lighter architectures can be chosen for simpler problems. All resulting image crops can be batched together such that the inference of the MP-Encoder is parallelized. For a single instance the inference takes $\sim 5ms$, nearest neighbor search in the codebook takes $1-2ms$ and the projective distance estimation is negligible. While our RGB-based pipeline runs at high speed and makes our approach applicable for mobile applications, the depth-based ICP refinement takes in average $0.6s$ per target and is thus suitable for robotic manipulation tasks.

Table 6: Evaluation of the full 6D Object Detection pipeline with MaskRCNN + Multi-Path Encoder + optional ICP. We report the mean VSD recall following the SIXD Challenge [18] on the T-LESS Primesense test set.

	Template matching	PPF based			Learning-based				
object	Hodan-15 Depth	Vidal-18 Depth +ICP	Drost-10 Depth	Drost-10-edge Depth	Kehl-16 RGB-D + ICP	**OURS** RGB + ICP	Brachmann-16 RGB-D	Sundermeyer-18 RGB	**OURS**
1	66	43	34	53	7	73.61	8	9.48	5.56
2	67	46	46	44	10	66.40	10	13.24	10.22
3	72	68	63	61	18	87.24	21	12.78	14.74
4	72	65	63	67	24	82.91	4	6.66	6.23
5	61	69	68	71	23	86.16	46	36.19	37.53
6	60	71	64	73	10	92.79	19	20.64	30.36
7	52	76	54	75	0	80.83	52	17.41	14.62
8	61	76	48	89	2	81.32	22	21.72	10.73
9	86	92	59	92	11	85.15	12	39.98	19.43
10	72	69	54	72	17	82.31	7	13.37	32.75
11	56	68	51	64	5	72.60	3	7.78	20.34
12	55	84	69	81	1	68.80	3	9.54	29.53
13	54	55	43	53	0	53.37	0	4.56	12.41
14	21	47	45	46	9	50.54	0	5.36	21.30
15	59	54	53	55	12	45.25	0	27.11	20.82
16	81	85	80	85	56	82.32	5	22.04	33.20
17	81	82	79	88	52	72.27	3	66.33	39.88
18	79	79	68	78	22	80.60	54	14.91	14.16
19	59	57	53	55	35	48.17	38	23.03	9.24
20	27	43	35	47	5	25.74	1	5.35	1.72
21	57	62	60	55	26	47.53	39	19.82	11.48
22	50	69	61	56	27	50.27	19	20.25	8.30
23	74	85	81	84	71	46.99	61	19.15	2.39
24	59	66	57	59	36	78.20	1	4.54	8.66
25	47	43	28	47	28	50.00	16	19.07	22.52
26	72	58	51	69	51	64.61	27	12.92	30.12
27	45	62	32	61	34	73.09	17	22.37	23.61
28	73	69	60	80	54	86.14	13	24.00	27.42
29	74	69	81	84	86	77.66	6	27.66	40.68
30	85	85	71	89	69	92.96	5	30.53	56.08
Average	**63.18**	66.51	56.81	**67.5**	24.6	**69.53**	17.84	19.26	**20.53**
Time (s)	13.5	4.7	2.3	21.5	1.8	0.8	4.4	0.1	0.2

Figure 7: Qualitative 6D Object Detection results on T-LESS Primesense scenes. Only 18 of 30 objects are used to train the Multi-Path encoder. Left: Predictions of the RGB-based pipeline; Middle-left: Error in depth [mm]; Middle-right: ICP-refined results; Right: Refined error in depth [mm]. Note that only one instance per class is predicted following the BOP [21] rules.