A Overview

In the following, we provide additional technical details and supporting results. The sections are organized as follows:

- Sec. B reviews how training with geometric correspondences is conducted.
- Sec. C provides further details for training models.
- Sec. D offers an overview and example images of our datasets.
- Sec. F.1 discusses further results from the augmentation ablation study,
- Sec. G details grasp preference heatmap generation and the objects used in the grasping experiment.

B Training with Geometric Correspondences

While originally introduced by [2], we utilize the adapted method by [4] for training without masks in multi-object settings.

The training relies on sampling a set of corresponding pixels in image A and B, where both images observe the same static scene and objects, but from different view points. By employing a contrastive loss, the descriptors for each pixel pair are trained to have the same embedding, while separating all other pixels in latent space. The view-invariance of the descriptors is the consequence of utilizing images with different view points.

Geometric correspondence training exploits the geometric prior provided by a registered RGBD sequence. As the relative pose between any two images in the sequence is known, and given the depth and camera information, one can establish the per-pixel correspondence between each image pair, allowing for straight-forward sampling correspondences.

In practice, depth data can be noisy or incomplete. For example, structured light cameras struggle with transparent or black surfaces, and with higher measurement uncertainty around edges of objects. Thus, [2] perform a 3D-reconstruction of the scene, to render synthetic depth images which are complete and denoised, albeit not perfect ground-truth. In the original approach, which focuses on training with singulated objects, an automatic or manual mask generation of the object is performed. As we deal with multi-object scenes, we follow [4] and instead record scenes with multiple objects present and do not compute any masks. Hence, we sample correspondences anywhere in the image and do not differentiate between object or background.

Instead, to sample correspondences we first prune the correspondence map from image A to B by occlusion and field of view masking, then we sample a set of N pixel correspondences. As we apply augmentations, the process can shift or remove pixels from either image. We need to account for this in the sampling process. Given a set of sampled correspondences, we employ the same loss as described for our synthetic view training.

C Training Details

The training settings, see Table 2, are shared for both the geometric and synthetic training, with exceptions specified below. They follow the findings made by [4] for geometric training. They are used for all experiments shown, unless specified otherwise.

We perform validation after each epoch and retain the checkpoint with the best score. The score is evaluated with respect to the area under the curve (AUC) of the PCK@K (percentage of correct keypoints). PCK@K is determined by taking a set of predictions and calculating the pixel error e with respect to the corresponding ground-truth pixels. The percentage is given by the number of predictions with a pixel error $e < K$. We evaluate the AUC for the range $K \in [1 \cdots 100]$.

Both approaches, geometric and synthetic, are trained with the same augmentation parameters as listed in Table 3. The major difference is that for synthetic training, we sample each augmentation with probability $p = 1.0$, whereas for geometric training each augmentation is sampled with $p = 0.5$. We found that the latter performed better.

Table 2: Default training settings for both geometric and synthetic view training.

Parameter	Setting
Latent dimension (default)	64
Temperature (NTXent)	0.07
Optimizer	Adam [25]
Learning rate	0.0003
Number of correspondences per image pair	2048
Batch size	2
Batches per epoch	500
Validation every n epochs	1
Total epochs	250

Table 3: Settings for augmentations with respect to the Torchvision library implementation.

Parameter		Setting
Color Jitter		
	Brightness	0.2
	Contrast	0.2
	Saturation	0.2
	Hue	0.2
Affine		
	Rotation Angle	$[0 \ldots 359]$
	Scale	$[0.5 \ldots 1.0]$
Perspective		
	Distortion Scale	0.4
Resize&Crop		
	Scale	$[0.7 \ldots 1.0]$

D Training Datasets

This section gives more details on the data used during training of the described methods. Overall, two different training dataset were used. One taken with a robot-mounted camera used for comparing training with geometric and synthetic correspondence and for all results presented in section 4. And another one with a fix-mounted camera used for the grasping experiment described in section 5.

D.1 Dataset with robot-mounted camera

Fig. 7 shows example images from this dataset. We prepared seven objects in a static scene and recorded a stream of images with 30 frames per second, while the robot-mounted Realsense D435 camera moved in different perspectives around the scene. In total we took nine recordings like this with different configurations of the same objects. Six of those recordings were used for training, one for validation and one for testing. Each recording contains about 4400 images.

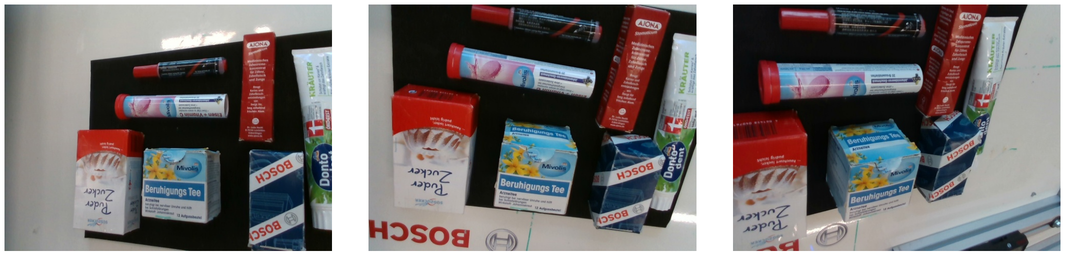


Figure 7: Three example images of the mixed training dataset that was taken with a robot-mounted camera.

D.2 Dataset for invariance tests

For the invariance tests reported in Sec. 4.2 we took data with the same setup of the robot mounted camera and the same objects as reported above in Sec. D.1, but with dedicated camera movements. These camera transformations are visualized in Fig. 8. The left shows the camera rotation (z-axis), where the location of the camera is fixed with the camera plane parallel to the table. The camera is then rotated around the z-axis. In the middle we show the camera moving in/out along the z-axis, closer to and further away from the scene. The x and y-positions as well as the camera orientation are kept stable. The last movement is the camera perspective movement. Here, the camera is moved in x direction on a half-sphere around the scene changing the orientation to keep focus on the center of the scene. During this movement the distance of the camera to the center of the scene is fixed.

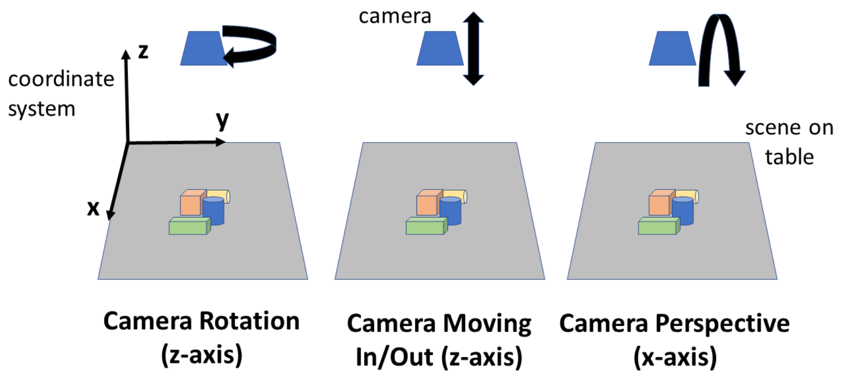


Figure 8: Camera transformations used for the invariance tests. Note thatt the origin of the coordinate system is located in the center of the scene on the table.

D.3 Dataset with fixed-mounted camera

Example images of this dataset are shown in Fig. 9. It was recorded with a fixed-mounted Zivid One+ camera. We recorded 529 images of randomly shuffled heaps of the objects presented in section G. Additional data was held back for validation and testing.

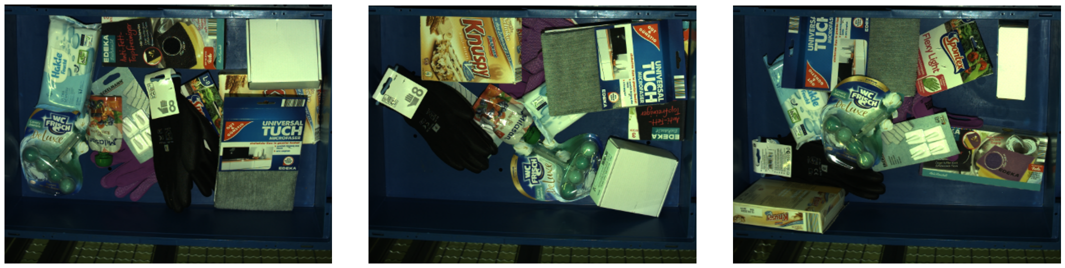


Figure 9: Three example images of the training dataset for the grasping experiment.

E Additional Results

In the main section we provide results mostly in terms of the median pixel error and the 75% quantile of pixel errors. In the following section, we provide the main results with additional metrics. Furthermore, we present results on the generalization capabilities on a small test set featuring unknown objects.

E.1 Extended Main Results

The results of Sec. 4.1 are summarized in the Table 4 with additional metrics.

Table 4: Pixel errors (mean, median, quantiles), and percentage of correct keypoints (PCK@k) metrics for geometric and synthetic correspondence training, evaluated on our kfold-cross validation dataset as used in Sec. 4.1.

			Quantile			PCK@				
Type	Mean	Median	75%	90%	95%	3	5	10	25	50
Geometric	13.50	3.16	6.4	18.03	55.15	0.46	0.68	0.84	0.92	0.95
Synthetic	21.49	5.10	11.0	50.25	103.25	0.29	0.49	0.73	0.85	0.90

E.2 Results on Unknown Objects

We further investigate the performance of our model on objects not seen during train time. Both the geometric and synthetic correspondence models were trained on the same kfold dataset splits of the main section. We test on five new objects, not previously seen during training and validation. The objects and their arrangement in the two test scenes are shown in Figure 10. We kept the training setup as in Sec. 4.1, although we note, that better generalization might be achieved with different configuration of hyper-parameters, the choice and amount of augmentation applied. However, the overall trend is evident in the results compiled in Table 5.

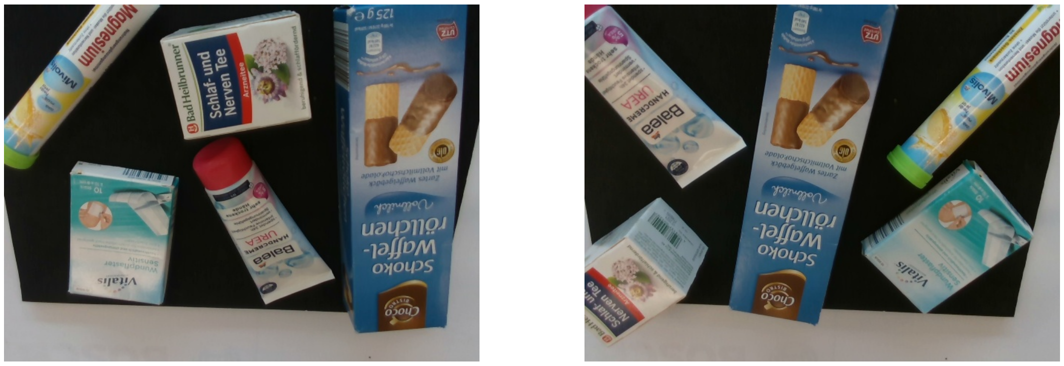


Figure 10: Novel object test set consisting of two new scenes, with 5 novel objects.

Both approaches, SV and GC, exhibit a loss in performance, especially the geometric correspondence training. While the median changes only slightly, we find a large increase with respect to the 90% and 95% quantile for both methods. Up to 25% of the sampled keypoints are now mispredicted with an error nearly three times as high as before.

Using features from a purely pre-trained backbone, without further training, fails completely. Training on a generic dataset, such as COCO, yields surprisingly good results, but still fails to work accurately. For more details on the pre-trained and SV-COCO setup, see the ablation study in Section F.3.

We note that, as we train only on a set of unordered RGB images, fine-tuning the model for additional new objects is as easy as adding a few new image taken of the novel objects. Hence, despite limited generalization to completely new objects, the simple and efficient training of our proposal may effectively compensates for it.

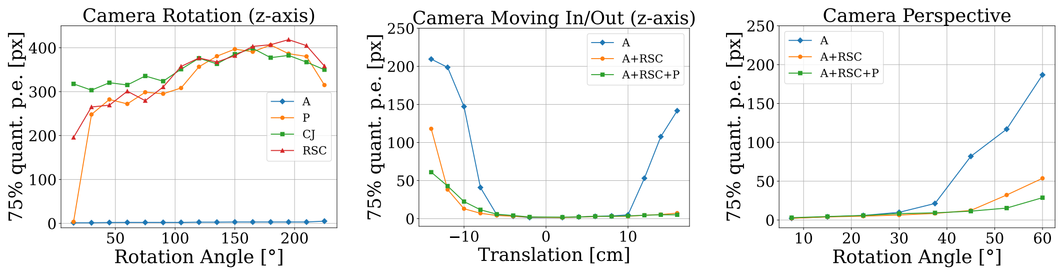
Table 5: Pixel errors (mean, median, quantiles), and percentage of correct keypoints (PCK@k) metrics for geometric and synthetic correspondence training, evaluated on two new scenes with 5 novel objects.

			Quantile			PCK@				
Type	Mean	Median	75%	90%	95%	3	5	10	25	50
GC	43.40	6.40	21.19	162.25	280.64	0.23	0.41	0.63	0.77	0.82
SV	42.95	6.00	17.09	148.76	284.02	0.25	0.44	0.66	0.78	0.82
Pretrain Only	49.96	22.47	42.01	112.70	260.41	0.03	0.08	0.21	0.55	0.80
SV-COCO	42.35	4.12	9.43	165.06	302.76	0.36	0.58	0.76	0.82	0.85

F Ablation

F.1 Augmentations

Complementing the findings in section 4.2 we study the influence of different augmentations for the synthetic correspondence training on the ability of the network to generalize to different camera transformations. Fig. 11 shows the 75% quantile of the pixel error distribution for the synthetic correspondence training with different augmentations. We find that *affine transformations* are most critical, as only a model trained with it shows invariance to rotations, see Fig. 11a. This matches the expectations as standard CNN are by default not invariant to rotations. Nevertheless, we find that both *resize&crop* and *perspective distortion* both further improve the performance of just affine transformations. In particular, for camera movements that induce perspective distortions and scale changes, see Fig. 11c and Fig. 11b, the error decreases considerably. Lastly, we find that *color jitter* further reduces the overall mean pixel error from 19.4 to 17.1 pixel. The improvement appears modest, but we note that our test dataset was recorded at the same time as train and validation, and the lighting conditions of the scene were not explicitly altered. For a complete table of all the combinations of augmentations, see the Appendix.



(a) (b) (c)

Figure 11: 75% quantile pixel error for different combinations of augmentation. **A**ffine, **P**erspective, **C**olor **J**itter, **R**e**s**ize & **C**rop, with respect to different tasks: (a) camera rotates around z-axis, (b) camera moving closer and further away from the objects, (c) camera moving on a sphere in x-direction around the objects, facing the objects. Note the scale of the y-axes.

To investigate the impact of each augmentation, we trained the synthetic approach with each combination and tested it on the same dataset as in Section 4.1. The full results are listed in Table 6.

We see that affine transformations have a strong impact on the overall performance. All other augmentations, even the combination of color jitter, perspective and resize+crops, performs considerably worse. This result confirms that for our approach affine transformations are indeed essential to obtaining invariance to rotations with a CNN-based backbone network. Generally, all combinations with affine augmentation further improve the models accuracy and robustness. An exception is the combination of color jitter and affine, which seems to find a worse solution when combined.

We find that when using perspective distortion, the median typically seems to slightly decrease, while the mean, as well as the 75% and 90% quantiles improve considerably. Hence, perspective

Table 6: Pixel error (mean, median, 75% and 90% quantile) for different combinations of augmentations for synthetic correspondence training. Abbreviations are as follows: **A**ffine, **P**erspective, **C**olor **J**itter, **Res**ize & **C**rop.

Combination	Mean	Median	75% Quantile	90% Quantile
CJ + A + P + RSC	17.18	5.00	10.05	37.48
A + P + RSC	19.42	5.10	10.44	49.38
CJ + A + RSC	19.62	4.47	9.49	55.63
A + RSC	23.06	4.47	10.44	67.08
A + P	36.64	8.60	21.02	104.05
CJ + A + P	37.27	9.00	21.02	105.54
A	52.46	6.71	49.65	181.03
CJ + A	56.58	9.22	62.43	176.26
CJ + P + RSC	97.10	20.81	168.44	296.19
P + RSC	110.80	72.45	185.33	288.45
CJ + P	144.76	118.43	226.37	342.49
P	152.97	122.25	234.08	365.40
CJ + RSC	176.97	164.47	255.64	345.12
RSC	180.55	168.58	267.59	359.61
CJ	227.43	211.21	316.31	412.00

distortions seem to play an important role in improving model robustness, but requires further investigation as to why the accuracy is negatively affected.

Color jitter seems to be the augmentation with the smallest impact. However, we note that while we tested on different scenes, the lighting conditions are generally the same. Hence, on datasets, and more importantly during model deployment, the impact of color jitter with respect to model robustness and reliability could be larger.

Lastly, we see that the combination of all augmentations ensure the learned descriptor is not overly focused on single type of invariance, but different kinds yielding the overall best result.

F.2 Probability of Augmenting and Number of Augmented Frames

In this experiment we vary two hyperparameters of our training: i) the chance that any given augmentation might be applied (independently drawn), ii) number of views that will be augmented.

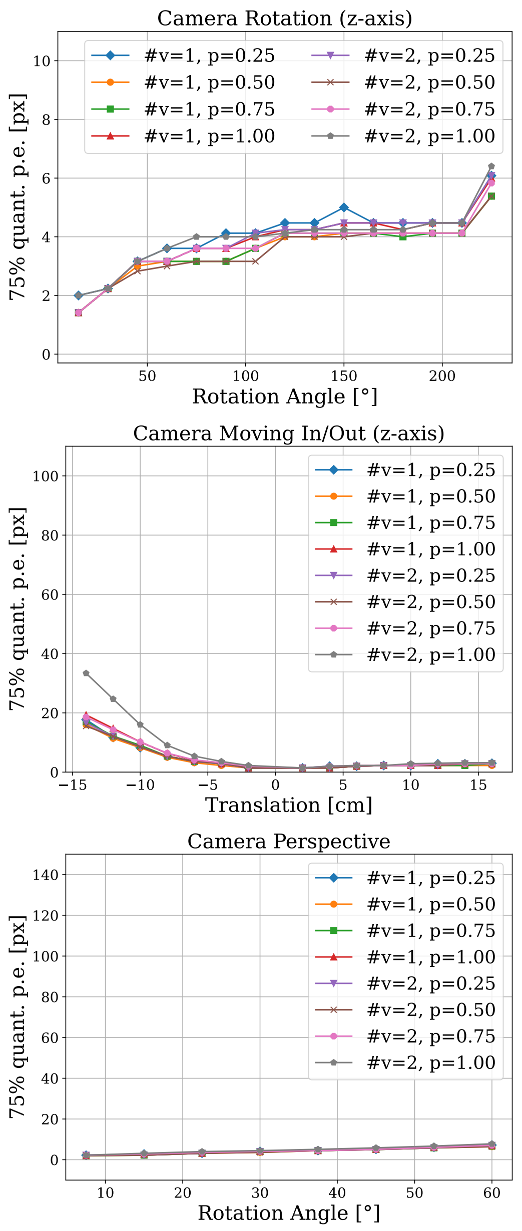
We evaluated each configuration on the invariance test dataset, with the results shown in Figure 12.

We find that augmenting just one or both images, has limited impact for both geometric and synthetic correspondence training. For geometric training we find that our setting, which is using 50% probability per augmentation, yields similar results compared to augmenting just one view. This was already observed in [4]. We reconfirm, as reported by [4], that augmenting more heavily, e.g., both frames with each augmentation at 100%, has adverse effects on the performance of geometric correspondences trained networks.

In contrast, the synthetic correspondence training is most strongly impacted by the probability, less by the number of augmented images. This is not surprising, as unlike the geometric training, augmentations are essential for the synthetic training, cf. Section F.1. Without any augmentations, both views are identical and the network will only learn a trivial solution. Consequently, it is important to increase the chance, or guarantee in some ways, that at least one frame is augmented. The difference between augmenting one or both views with high probability yields nearly the same results. We note, that a more refined selection of differing probabilities per augmentation type would most likely yield even better results, rather than just one global parameter choice.

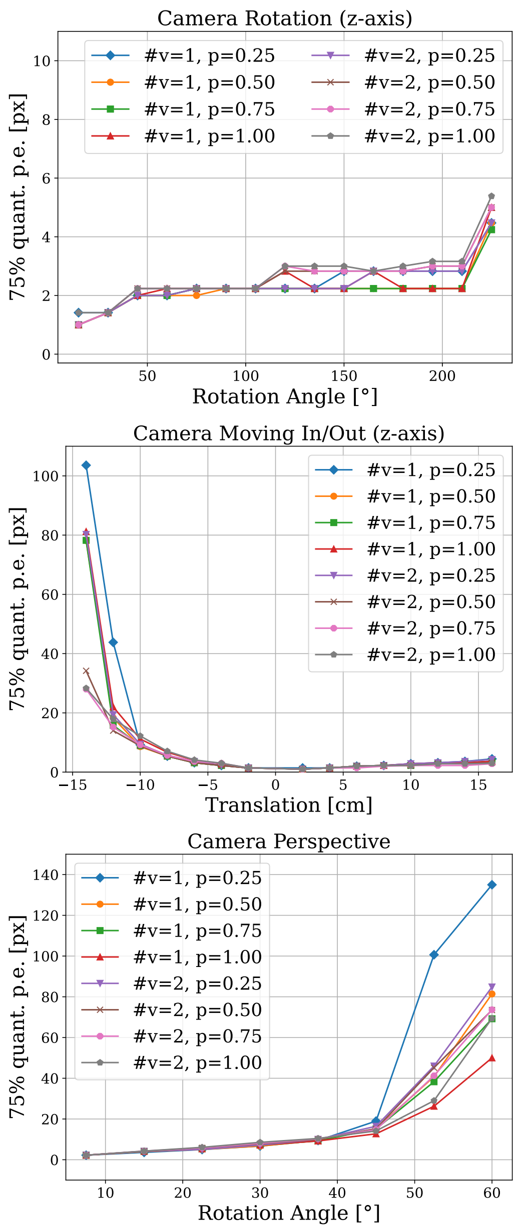
F.3 Comparison to Baseline Methods and State-Of-The-Art Approaches

With this additional set of experiments we validate our assumption that domain specific data and explicit augmentations for perspective changes are crucial for a good performance on the target domain. For this, we compare our method to four different baseline methods:



(a) Trained using Geometric Correspondence

6



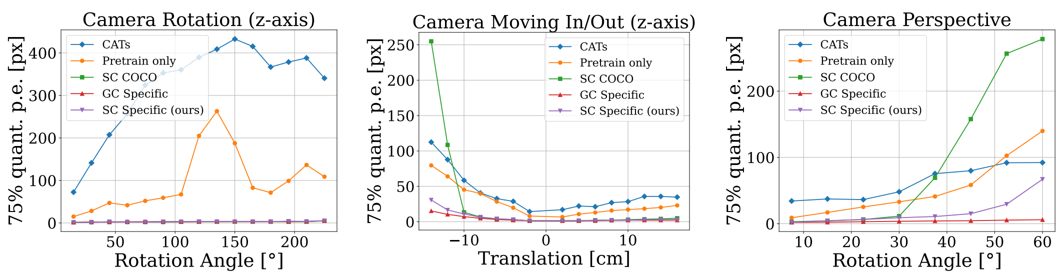
(b) Trained using Synthetic Correspondence

Figure 12: 75% quantile pixel error for different combinations augmentation probabilities p, and number of augmented frames (#v), evaluated with respect to different tasks: (a) camera rotates around z-axis, (b) camera moving closer and further away from the objects, (c) camera moving on a sphere in x-direction around the objects, facing the objects. Note the scale of the y-axes.

1. *GC Specific*: Using data from different viewpoints ([1, 2, 4]), as already reported in the main paper.

2. *Pretrain only*: Using the features from a pretrained ResNet backbone (on ImageNet), without further fine-tuning. This serves as a naïve baseline.

3. *SC COCO*: Using the method presented in this paper, but fine-tuned on COCO data instead of domain-specific data.



(a) (b) (c)

Figure 13: 75% quantile pixel error for different network and data configurations. Networks named *SC* are trained using our proposed synthetic correspondence setup, *GC* using geometric correspondences, and lastly, as sanity check, raw features extracted from our ImageNet-pretrained ResNet backbone are named *Pretrain only*. *SC COCO* was trained using the COCO dataset, while networks named *Specific* were trained using our dataset. We compare with respect to three view invariance tasks: (a) camera rotates around z-axis, (b) camera moving closer and further away from the objects, (c) camera moving on a sphere in x-direction around the objects, facing the objects. Note the scale of the y-axes.

4. *CATs*: A state-of-the-art keypoint matching algorithm among the top ranking methods on various keypoint matching datasets [26]. We compare to the pretrained method on PF-Pascal as provided by the authors.

We evaluated all methods on our view-invariance test dataset, as described in 4.2. Figure 13 shows the results.

Not surprisingly, the raw pretrain-only features, exhibit little to no rotational invariance, and generally lack view-invariance on other tests.

SC COCO does perform better than the pretrained-only method, especially for smaller transformation angles. However, it seems to not generalize well to larger transformation angles in our invariance tests. This indicates that in-distribution training data is important for the accuracy we need for the robotics use-case.

The keypoint matching method *CATs* achieves very impressive results for semantic keypoint matching, where e.g., the tip of a dog's nose will be matched to a completely different dog's nose in a second image. Surprisingly, the method does not outperform the *Pretrain only* baseline on our test dataset. We account this result to the following: (a) PF-Pascal has a limited number of classes and the pretrained network overfits to those (b) The goal of *CATs* (and similar approaches) is semantic keypoint matching. In this goal it achieves very impressive results. However, the goal of these methods are not the very accurate matching of geometric points on target objects, which we need for robotic grasping (see the reported mean pixel error in the original *CATs* paper).

The above experiments support our claim that that our proposed training schema with the choice of augmentations and loss together with in-distribution training data containing scenes of the target objects, plays an important role to achieve sufficiently high accuracy for robotic grasping applications.

G Grasping Experiment

G.1 Objects

In Fig. 14 we show the objects we used in the grasping experiment. Each object is either challenging to grasp with a suction gripper while relying only on depth images and geometrical features, or successful grasps may damage the objects. Therefore, we wish to rely on human annotated grasp preferences to avoid damage and improve success chances. Fig. 14a and Fig. 14b show gloves which are only graspable on the paper label, which is also the preferred grasp location. Additionally, a cutout at the top and plastic strips in the middle of the paper label make these objects challenging to grasp. The depth camera may not recognize the small bumps in the depth image for the hangers in Fig. 14c, therefore we would like to enforce grasping on the paper label. The non-rigid object in

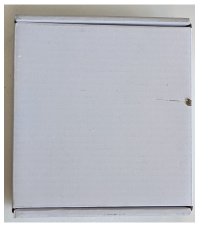
Fig. 14d is not particularly hard to grasp, but we wish to improve success chance by grasping in the middle of the object. Similarly, the white box in Fig. 14e is not challenging to grasp, but grasping in the middle improves the chance of success. The dense descriptor representation allows to accurately locate the center for such textureless objects. The sponge in Fig. 14f has a cutout in the middle of the packaging where suction grasps will fail, therefore we prefer to grasp towards the top, or bottom. As the towel in Fig. 14g is clearly visible in the depth image, suction grasps will often fail on the towel, but not on the label. The plastic cover in Fig. 14h does not show up on depth images, therefore we prefer to focus the grasp towards the top of the packaging. Finally, the wet wipe in Fig. 14i is not difficult to grasp, but we wish to avoid grasping by the package opening, which might be damaged when using a suction gripper.

(a)

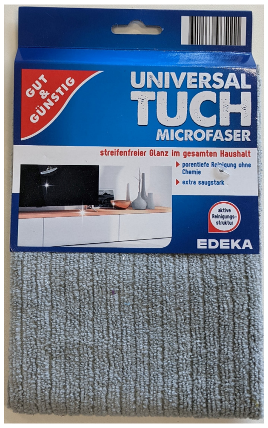
(b)

(c)

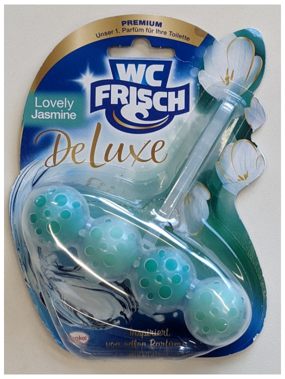
(d)


(e)

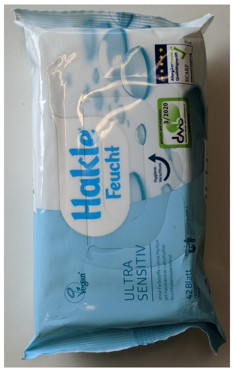
(f)



(g)



(h)



(i)

Figure 14: We used nine different objects in the grasping experiment. Some of these objects are prone to fail when planning the grasp based on 3D geometry. In other cases a suction grasp may damage parts of the objects. Therefore, using grasp preferences both improves grasp success and avoids damaging object packaging.

G.2 Generating Grasp Preference Heatmaps

In this section we detail the computation steps of the grasp preference heatmap. The experiment consists of an offline grasp preference annotation phase, and an online autonomous operation phase performing the grasps in the bin.

Annotation phase. First an RGB image I showing the objects in the bin is presented. Then the human clicks at pixel locations $\{k_i^j\}$ corresponding to preferred grasp locations. The descriptor values $d_j = f(I;\theta)(k_i^j)$ at these pixel locations are then stored into a keypoint database $\mathcal{D} = \{d_j\}$. See Fig. 6a for an example image of the objects in the bin in a random configuration.

Autonomous operation phase. During autonomous operation, the latest RGB image I taken of the bin is evaluated with the trained network resulting in the descriptor image $I_d = f(I; \theta)$. Then, keypoint heatmaps are generated from the database with $h_j(u, v) = \exp(-\text{dist}\,(I_d, d_j)\,/\eta),\ \forall d_j \in \mathcal{D}$, with η as a temperature parameter that controls the width of the heatmap. Finally, the individual keypoint heatmaps are fused into a single heatmap function $h(u, v) = \sum_j h_j(u, v)/H$, with H as a normalization constant (see Fig. 6c for an illustration).