PoseMatcher: One-shot 6D Object Pose Estimation by Deep Feature Matching

Pedro Castro
Imperial College London
p.castro18@imperial.ac.uk

Tae-Kyun Kim
Imperial College London, KAIST
tk.kim@imperial.ac.uk

Abstract

*Estimating the pose of an unseen object is the goal of the challenging one-shot pose estimation task. Previous methods have heavily relied on feature matching with great success. However, these methods are often inefficient and limited by their reliance on pre-trained models that have not be designed specifically for pose estimation. In this paper we propose **PoseMatcher**, an accurate one-shot object pose estimator that overcomes these limitations. We create a new training pipeline for object to image matching based on a three-view system: a query with a positive and negative templates. This simple yet effective approach emulates test time scenarios by cheaply constructing an approximation of the full object point cloud during training. To enable PoseMatcher to attend to distinct input modalities, an image and a pointcloud, we introduce IO-Layer, a new attention layer that efficiently accommodates self and cross attention between the inputs. Moreover, we propose a pruning strategy where we iteratively remove redundant regions of the target object to further reduce the complexity and noise of the network while maintaining accuracy. Finally we redesign commonly used pose refinement strategies, zoom and 2D offset refinements, and adapt them to the one-shot paradigm. We outperform all prior real-time one-shot pose estimation methods on the Linemod and YCB-V datasets as well achieve results rivaling recent instance-level methods. The source code and models are available at github.com/PedroCastro/PoseMatcher.*

1. Introduction

Accurately retrieving the relative position and orientation of an object is the first step for any task that requires interaction with objects in the real world. Estimating the pose of an object is an indispensable step for robotic manipulation as well as in VR/AR applications. It is imperative for the pose estimation to be accurate and robust to external obstacles such as occlusion, illumination and symmetries. Existing methods excel at retrieving the pose of known objects to a very high accuracy standard [42, 7, 2]. Some methods can even produce pose estimation at high throughput [42, 4], be specially robust to symmetries [10] and can even surpass synthetic to real domain gaps [41, 36]. However, most share a very large limitation: the target objects must be known. This constraint is severely limiting as it necessitates model retraining for each object addition. Retraining new objects on existing models might lead to catastrophic forgetting if not handled properly [19]. Category level approaches [46, 43] try to generalize up to a category, where each target object can be obtained with a simple deformation of a canonical model and all share semantic keypoints (ex. handle on a mug or the cap of a bottle). Nonetheless, much of the same one-shot domain problems still remain: objects category must have been seen before and the target object must not lie outside of the scope of the category for the estimation to be possible. In order to overcome these problems we must tackle object pose estimation in the one-shot paradigm.

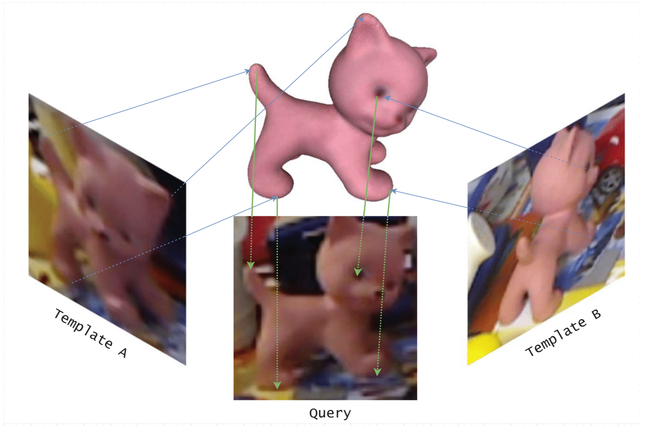


Figure 1: **Illustrative diagram of PoseMatcher training diagram.** For each training instance query we sample two templates. From these templates, we reconstruct a partial point cloud simulating the object point cloud at test time.

By one-shot pose estimation, we refer to estimating the pose of an novel object based on information seen only at test-time, without the object,or its category, being present in the training dataset. Older methods have some very hard constraints such as needing masking and depth maps at test-time [13], a full colored 3D model of the target object [22], need to be fine-tuned on images of the same dataset for difficulties of overcoming domain gap [25] or rely on an extremely expensive test-time optimization through renderers [47, 21, 28]. More recently, feature matching methods have been shown to achieve impressive results. Particularly, OnePose [38] and OnePose++ [12] make use of existing state of the art pre-trained descriptor extractors on top of which a pose estimation pipeline is built. However, by relying on a fixed pre-trained model to extract descriptive keypoints from templates it fails to capture the optimal keypoint object descriptions. Ideally, the template extraction model should be the same as the query model and should be jointly optimized.

We rethink the approach to the problem and introduce a three-view pipeline that allows us to jointly train the template and query extractors. Just by redesigning the training pipeline, we can improve OnePose++[12] without additional changes to its methodology. We also introduce a new efficient image to object attention layer which we call IO-Layer, which reduces both parameters when compared to the modules used by OnePose++ [12] and also separates the two input modalities, an image and a pointcloud, allowing the model to optimize weights specifically for either image or pointcloud keypoints. We also found that working with a full object at all stages of the feature matching is redundant and reduces pose estimation accuracy which lead us to introduce a novel iterative matching based pointcloud pruning. On top of these changes, we employ a 3D-refinement technique based on zoom refinement used on instance-level pose estimation [22, 4, 20].

In summary our contributions are as follows:

- We redesign the training pipeline for one-shot object-to-image matching. Our three-view training approach allows us to train from scratch PoseMatcher, a pose estimation method leveraging detection free keypoint matching.
- We introduce a more efficient image to object attention layer we call IO-Layer, specifically built to accommodate the two input modalities, the image and the object pointcloud.
- We propose a layered pruning of the target object point cloud. We improve runtime by reducing the amount of *attented* keypoints while reducing noisy matching, leading to an improvement in accuracy.
- We create a fine level 3D based refinement where we directly estimate relative 2D and depth positions, replacing the 2D keypoint refinement used by prior one-shot approaches.

2. Literature

Fully supervised pose estimation. Instance-level pose estimation is designed to support a single object. Due to this task's narrowed scope, the accuracy of instance-level methods are becoming increasingly more impressive. Early 6D pose estimation works focused on recovering the 2D position of specific keypoints, mainly the 3D bounding box of the target [33, 27, 39]. PVNet [30] found that choosing keypoints that lie within the object's silhouette would yield better results. This idea has become a mainstay among current keypoint based methods [15, 4]. An alternative to chosing a limited amount of features was to estimate the coordinate of the surface of the object at each pixel [43, 48, 29]. GDR[42] and SO-Pose[7] approximated PnP through a small neural network and optimizing directly through pose errors. Direct pose estimation with a combination of a dense intermediate representation makes up most of the state of the art methods as per the BOP challenge [14]. Other ideas have focused on designing textures as learnable features that can be more easily estimated [10, 16] and as a result, better correspondences are found. In order to decrease the need for real data, Sock *et al*. [36] proposed a differentiable pipeline where learning is done through comparison with a rendering of the estimation which is done using keypoints, where Self6D [41] directly outputs pose.

One-shot pose estimation. Until recent versions of the BOP challenge [14], Point Pair Features [13] held the top spot on its leaderboard. It relies on selecting geometrical relevant keypoints from an existing 3D object model and matching these to a depth map. This method relies on capturing information with a depth sensor at inference time which is not commonly available. Extending the prior instance-level approaches to category-level allows for object deformation and texture change within a narrow category [49, 9, 43]. The main idea from NOCS [43], although used for instance-level by a wide range of works [42, 7, 48], was initially to predict pose of slightly deformable objects within a category, as it was inspired by its human reconstruction contemporary DenseBody [1] . Recent advancements have been made that allow for training for in the wild [9]. However, these are sill limited by the similarity to objects of the same category. Our aim is to overcome this limitation.

Methods such as Pitteri *et al*. [31] and CorNet[32] rely on training with a subset of similar objects of the same dataset and/or scene, which includes biases towards illuminations, noise, background and shape. Recent attempts have shown researchers are keen on tackling one-shot pose estimation. OSOP [35] introduces a global template matching method where a given query is matched to the closest viewpoint from a database of pre-processed synthetic viewpoints, which requires the object model, an assumption we do not make. Gen6D [25] works in a similar fashion by

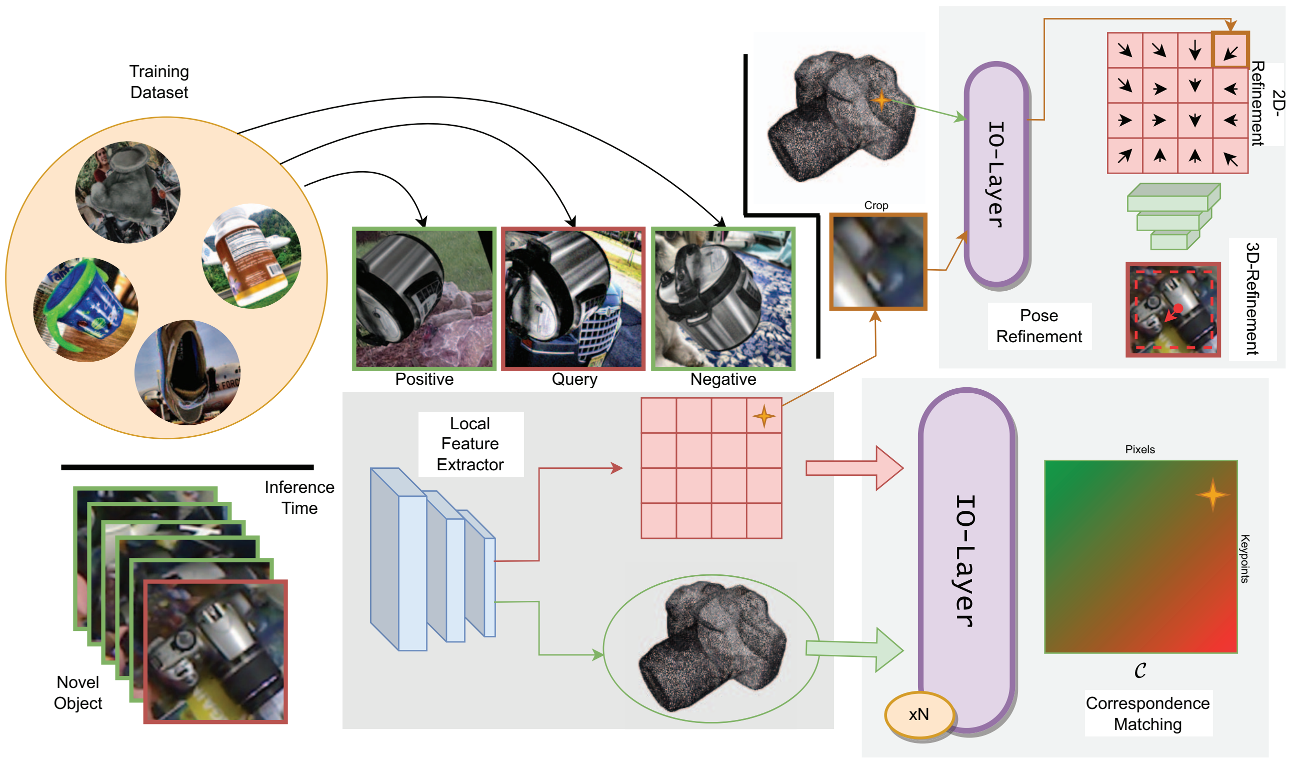


Figure 2: **Diagram of PoseMatcher.** At training time we sample from the Google Scanned Objects[8] dataset and from two opposing views templates we construct a partial point cloud. At test time, a novel unseen object is used instead. We extract local features from both the query and the templates. Using our novel IO-Layer, we compute dense correspondences between the image and the object. We then take the matches (yellow star) and further refine them using 2D and 3D techniques.

matching a query to a small amount of real annotated images and then refining the pose. However, it is susceptible to poor pose initialization and poor 2D bounding box detections. The closest work to our own is OnePose++, an improved version of OnePose[38], where the feature extraction is replaced by the detector free feature matcher LoFTR [37]. However, OnePose++[12] adapts a pre-trained LoFTR to one-shot pose estimation without taking consideration the different modalities of image and point cloud, self-occlusion redundancy and the possibility of using 6D pose based refinement.

3. PoseMatcher

The goal of the PoseMatcher is to establish matches between the two sets of keypoints. We establish correspondence between the 2D keypoints features extracted from the query image $\mathcal{I}^Q$ and the object point cloud $\{\mathcal{P}_O\}$. From those matches we can apply PnP and recover full 6D pose ζ. A diagram of PoseMatcher can be seen in Fig. 2.

3.1. Template Based Training

We adopt a new training methodology that allows us to design a one-shot object-image detector free feature matching model from scratch, specifically aimed at pose estimation task. We therefore remove the need for pre-trained descriptor extractors used by OnePose and OnePose++ [38, 12].

However, a single viewpoint will result in a partial reconstruction of the object. At inference time, both the visible and occluded regions of the object will be represented in the template point-cloud. Therefore, a single template is not enough to emulate the conditions at test-time.

To address these limitations, we sample an additional template image we refer to as the negative template $\mathcal{I}^-$. This template shares low co-visibility area with the anchor image. The keypoints extracted from $\mathcal{I}^-$ serve to generate nearly complete reconstruction of the target object, with the visible sections being sampled from $\mathcal{I}^+$ while the self-occluded ones from $\mathcal{I}^-$. This 2-template paradigm simulates the full object point-cloud available at the inference time.

In order to optimize through the matching task we apply the differentiable dual-softmax operator as proposed by LoFTR [37] and subsequently used by both OnePose and OnePose++ [38, 12].

In order to output matches, we start by using a local feature extractor, such as a Resnet [11] to extract coarse and fine level feature maps, $\hat{\mathcal{F}}_{\mathcal{I}^Q} \in \mathbb{R}^{HxWx\hat{C}}$ and $\tilde{\mathcal{F}}_{\mathcal{I}^Q} \in$

$\mathbb{R}^{HxWx\tilde{C}}$ from the query image. We repeat the same process for both $\mathcal{I}^+$ and $\mathcal{I}^-$ and sample M points from both templates. We make sure that the template positions lie within the objects mask such that:

$$\begin{aligned} p^+ &= \{p_k^+ \,|\, k \in \mathcal{M}^+\}, \\ p^- &= \{p_k^- \,|\, k \in \mathcal{M}^-\}, \end{aligned} \tag{1}$$

where $\mathcal{M}$ refers to the segmentation mask. We back-project, using the a depth map only available at training time, and generate a template point ${\mathcal{P}_\mathcal{T}}^{3D} \in \mathbb{R}^{2Mx3}$ with its corresponding extracted features $\hat{\mathcal{F}_\mathcal{T}} \in \mathbb{R}^{2Mx\hat{C}}$ and $\tilde{\mathcal{F}_\mathcal{T}} \in \mathbb{R}^{2Mx\tilde{C}}$, where $\mathcal{T}$ refers to the combined positive and negative templates.

At this point, we perform dense matching between the coarse level query features $\hat{\mathcal{F}}_{\mathcal{I}Q}$ and the extracted point cloud $\hat{\mathcal{F}_\mathcal{T}}$. We now aim to globally match each pixel of the query image that contains the object to its respective point cloud keypoint. We make use of positional embeddings to encode positional information into each feature point. For 2D keypoints we use the sinusoidal fixed version used by DETR [3] while for 3D positional encoding we use a simple 3-layer MLP as used by SurfEmb [10] and OnePose++ [12].

We flatten both feature maps and apply self and cross attention layers following other feature matching papers [34, 37, 38, 12, 44] to generate more easily separable features on each set. Our objective is to construct a matrix $\mathbb{P}^{HWx2M}$ that reflects the level of confidence of the correspondence between $\mathcal{P}^{2D}$ and $\mathcal{P}^{3D}$. A score matrix $\mathcal{S}$ is computed by measuring the cosine similarity between the two sets of transformed features in a contrastive manner. We apply dual-softmax [40] over $\mathcal{S}$ to calculate the correspondence confidence matrix $\mathbb{P}$:

$$\mathbb{P} = softmax(\mathcal{S}(i,\cdot))_c \cdot softmax(\mathcal{S}(\cdot,c))_i, \tag{2}$$

where $\mathcal{S} = \frac{1}{\tau}\langle\, \hat{\mathcal{F}}'_{\mathcal{I}Q} \,,\, \hat{\mathcal{F}}'_\mathcal{T} \,\rangle$, τ being a temperature hyperparameter, with i and c being the indices of the flattened image pixels and point cloud, respectively. The image to object correspondences $\mathcal{C}$ are established by choosing only those that exceed a certain level of confidence, represented by the threshold value θ, and meet the mutual nearest neighbor (MNN) constraint to remove false matches:

$$\mathcal{C} = \{(i,c) | \forall(i,c) \in MNN(\mathcal{P}_i^{2D}, \mathcal{P}_c^{3D}), \mathbb{P}_{i,c} \geq \theta\}. \tag{3}$$

The coarse loss $\mathcal{L}_c$ used to optimize $\mathcal{C}$ uses focal loss [24] as suggested by LoFTR [37].

3.2. IO-Layer: An object-image attention layer

Image to image feature matching has seen its effectiveness increase with the introduction of self and cross attention mechanisms [44, 34, 37, 26]. These layers are particularly efficient for this task seeing as both inputs are from the same modality and share the same spatial structure. For this reason, self attention layers can share weights and cross attention only requires swapping the query and template image, as you can see in Fig 3. OnePose++ [12] has suggested using the same approach for image to object matching however in this scenario we are working with two different modalities, a 2D image and a 3D point cloud.

We postulate that mixing modalities has an adverse effect on learning distinctive features. The positional encoding for each input is structurally different which means that attention layers will have difficulty separating the features of each modality. OnePose++ [12] shares a sign of this problem when they observe that adding 3D positional encoding results in a small ($< 1\%$) improvement. A simple ad-hoc solution would be to have modality specific weights, however that would double the amount of parameters and subsequently the necessary amount of training data and compute. In standard self attention layers an input sequence is linearly projected into a query, key and value: Q_i,K_i,V_i,stemming from the same input i. For cross attention, one would linearly project Q_i, K_j, V_j where i and j are different inputs. Usually, in attention mechanisms, the encoded message is parsed to the decoder and is not returned. However, since it is important for the image features to be distinct from the object's and vice-versa, the message must be passed bi-directionally and therefore requires careful redesign.

We propose a simple layer, specially designed for **I**mage to **O**bject matching we call **IO-Layer**. (Q_i,K_i,V_i) and (Q_o,K_o,V_o), the inputs to the attention mechanism for inputs and object respectively, are computed only once and used for both self and cross attention. Therefore cross attention is computed as $softmax(Q_i K_o^T/\sqrt{d})K_o$ and $softmax(Q_o K_i^T/\sqrt{d})K_i$ for image to object and object to image attention respectively, where we only have to project each sequence once. We modify self and cross attention accordingly to support Linear Attention [17]. Much like the layers used in LoFTR and OnePose++, these can be stacked together. We exemplify the IO-Layer on Fig.3.

3.3. Object Pruning

OnePose++ proposes sampling and using a point cloud template with over 15k keypoints per object. Performing attention over such a higher number of keypoints is expensive even if using more efficient mechanisms such as Linear Attention [17]. The set of keypoints that are not in visible in the query image should be quickly identifiable in early matching. Removing these keypoints from the point cloud allows for better separability of the features of the remaining visible keypoint, which leads to a better match. As a peripheral advantage, by removing a set of keypoints from the template point cloud, we reduce the complexity of the following attention layers. During pruning, we do not impose a hard threshold on the confidence of the matches but rather

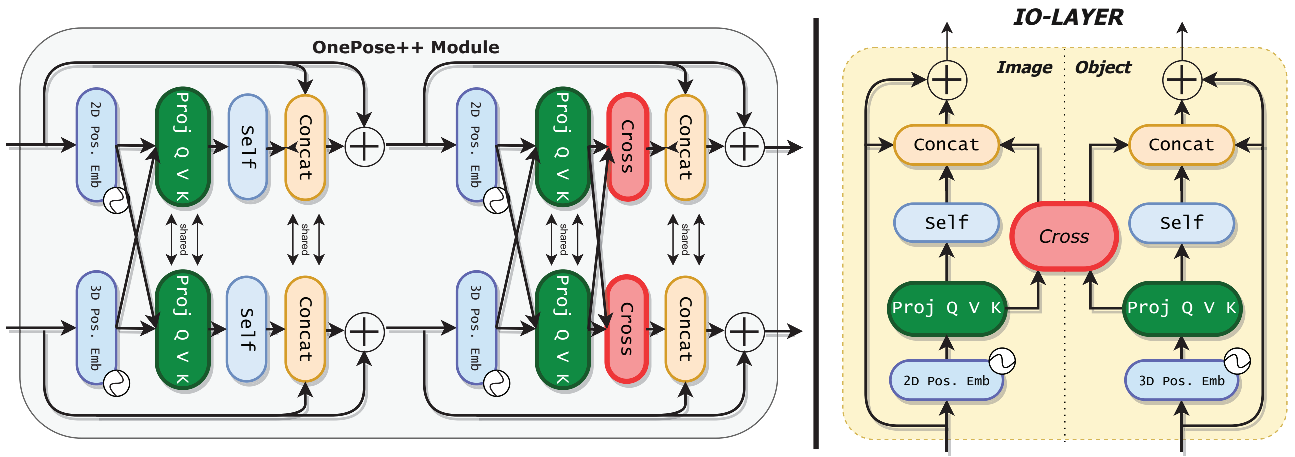


Figure 3: **IO-Layer architecture.** We improve on the attention modules used by LoFTR and OnePose++ [37, 38]. We have modality specific projections for image features embedded with 2D positional encoding and for object template features with its 3D positional encoding. We found that separating the two modalities improves the final pose accuracy.

but rather we select the keypoints with higher confidences of existing in the image. We extend the matching operation in Eq. 3 where we sum the keypoint confidences over every pixel and prune the lowest ones. We insert a pruning step after each IO-Layer. The amount of pruning is subjected to ablation studies in Section 4.5.1.

3.4. Pose Refinement

We add a fine level 2D-refinement post-processing step as described by LoFTR [37] and OnePose++[12]. For each match, we crop a window at the match location and perform simple matching w.r.t. the matched keypoints. We supervise it by minimizing the Euclidean distance between the center of the grid and the projected keypoints. More details about 2D refinement are available in LoFTR [37].

However, its impact is limited as we show in the experimental section. We believe this is due to the use of PnP following the keypoint position refinement. Prior 6D object pose literature has shown that estimating translation through PnP leads to poor results [23, 42]. An alternative approach, first introduced by CDPN [23], is to directly estimate the translation as the 2D offset of the object centroid as well as bounding box relative depth estimation. Directly applying this step to PoseMatcher cannot be done seeing as the objects are not known.

In order to perform translation estimation, we adopt translation refinement, which will refer to as 3D-refinement, a strategy used by iterative methods [4, 20, 22]. Instead of estimating the translations directly, we estimate the translation errors given an initial pose. Intuitively, we estimate the necessary 2D translation and *zoom* necessary to align the initial pose to the target pose.

In order to generate the initial pose ζ^0, we perform PnP over the coarse matching keypoints. In the 2D-refinement step, the initial position of the matched keypoints are the image grid default positions. However, if we are performing a refinement over an initial pose, this alignment might be broken due to erroneous matching, pose estimation or even sub-pixel positioning. Therefore we project the matched keypoints using the initial pose and use those 2D locations as the center for the fine-sampling grid crops F^i_{crop}. We perform self and cross attention over the grids and the corresponding 3D fine feature T^i_f using a single IO-Layer. We compute the 2D expectation and supervise its output using the same methodology as described for the 2D-refinement.

We propose a lightweight CNN similar to the one used for learned Patch-PnP in GDR-Net [42]. We build our pose representation as a pixel-wise map where at each coarse pixel we collect each matched keypoint refinement prediction (i.e. refinement direction). This CNN outputs the *3D-refinement* in the form of ΔT and ΔZ, the 2D location and zoom offsets respectively. Moreover, in order to supervise the refinement step, LoFTR [37] proposes computing the total variance of the matching heatmap in order to penalize more confident but erroneous estimations. We propose reusing this variance to introduce a confidence measure to each keypoint refinement output and append to each input its corresponding refinement.

We model the *zoom* as a classification problem and discrete the possible zooms in K classes [2]. We output the estimated *zoom* δ_z as the expectation over all classes and 2d translation δ_{2d} necessary to align ζ^0 with ζ^{GT} as following:

$$\begin{cases} \zeta_z^{GT} = \zeta_z^{GT} \cdot \delta_z, \\ p(\zeta^{GT}) = p(\zeta^{GT}) \ \cdot \ \delta_{2d}, \end{cases} \tag{4}$$

where $p(P) = \mathcal{K}\zeta[0,0,0]^T$ is the 2D projection location of the object's centroid adjusted for the crop bounding box and K are the camera intrinsic parameters.

We supervise the *zoom* and 2D location offset directly with a sparse loss:

$$\begin{cases} \mathcal{L}_z = ||\delta_z - \varepsilon_z||_1 \\ \mathcal{L}_{2d} = ||\delta_{2d} - \varepsilon_{2d}||_1 \end{cases} \tag{5}$$

Type	Fully Supervised			Self-Supervised		One-Shot			
Method	PVNet [30]	GDR [42]	SO-Pose [7]	Self6D [41]	Sock *et al.* [36]	Gen6D [25]	OnePose [38]	OnePose++ [12]	**PoseMatcher**
Ape	43.6	85.9	-	38.9	37.6	-	11.8	31.2	**59.2**
Benchvise	99.9	99.8	-	75.2	78.6	62.1	92.6	97.3	**98.1**
Camera	86.9	96.5	-	36.9	65.6	45.6	88.1	88.0	**93.4**
Can	95.5	99.3	-	65.6	65.6	-	77.2	89.8	**96.0**
Cat	79.3	93.0	-	57.9	52.5	40.9	47.9	70.4	**88.0**
Driller	96.4	100.	-	67.0	48.8	48.8	74.5	92.5	**98.4**
Duck	52.6	65.3	-	19.6	35.1	16.2	34.2	42.3	**54.1**
Eggbox*	99.2	99.9	-	99.0	89.2	-	71.3	99.7	**97.8**
Glue*	95.7	98.1	-	94.1	64.5	-	37.5	48.0	**91.5**
Holepuncher	81.9	73.4	-	16.2	41.5	-	54.9	69.7	**73.4**
Iron	98.9	86.9	-	77.9	80.9	-	89.2	97.4	**97.9**
Lamp	99.3	99.6	-	98.2	70.7	-	87.6	97.8	**98.1**
Phone	92.4	86.3	-	50.1	60.5	-	60.6	76.0	**92.1**
Average	86.3	91.0	96.0	58.9	60.6	-	63.6	76.9	**87.5**

Table 1: **Comparison study on Linemod.** We present the results for ADD(-S) metric and compare them to state of the art. While Linemod is close to saturated for fully supervised methods, one-shot pose estimation is still challenging. PoseMatcher achieves the best results for all objects for the one-shot category, surpassing self-supervised methods and close to fully supervised methods. Best results for one-shot are bolded. * denotes symmetric objects.

where ε is noise introduced at training time.

The full loss with our improvements becomes:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_f + \alpha\,(\mathcal{L}_z + \mathcal{L}_{2d}). \tag{6}$$

4. Experiments.

4.1. Data preparation

In order to train PoseMatcher from scratch we must use a sufficiently large dataset to encompass a large number of shapes and textures. We use the Google Scanned Objects[8] dataset with 1023 household objects. We use the renderings provided by [25] for fair comparison where each object is rendered at 250 different viewpoints. We use an additional set of 500 keypoints from ShapeNet [5] with renderings provided by [9]. For each instance we find a close viewpoint, $\mathcal{T}^+$, sampled within $[5°, 25°]$ orientation to ensure high co-visibility while being sufficiently disparate. $\mathcal{T}^-$ is sampled randomly from a set of the 5 farthest viewpoints from the query, which ensures enough data variety. We apply the standard color and noise augmentations [10, 42, 4] as well as bounding box *zoom-ins* as proposed by CDPN [23]. We also perform object level augmentations by changing the object's canonical reference frame. Although only 1500 objects are used, by providing strong augmentations we are able to avoid overfitting.

4.2. Implementation Details.

For fair comparison with OnePose++, we use 3 IO-Layers. We set the contrastive temperature term $\tau = 0.1$. For 3D-refinement we set $K = 100$ and $\alpha = 100$. During training we sample $M = 2048$ from each templates for a total of 4096. For refinement, we use the top 512 matches with confidences up to $\theta = 0.1$. If not enough matches during training, we pad the pointcloud with groundtruth matches. At test time, we sample 16k keypoints from the available training images. We prune 50% of the point cloud after each IO-Layer, to 8k and 4k after the first two layers, respectively.

For all experiments, we use AdamW [18] with cosine learning rate decay and we linearly warmup the learning rate for 5 epochs. We train PoseMatcher for 50 epochs with a batch size of 8 and an intial learning rate of 0.0001. At each epoch, we sample 10 different viewpoint queries for each of the 1500 objects.

4.3. RGB vs Grayscale images

Interest point descriptor models are mostly used over grayscale images as it has been shown to improve generalization. OnePose and OnePose++ [38, 12] are applied to grayscale images because they make use of pre-trained descriptor models trained solely on grayscale, SuperPoint [6] and LoFTR respectively [37]. PoseMatcher does not rely on prior methods thus is not limited to grayscale. To provide a comprehensive analysis and ensure thoroughness we train our method on both RGB and grayscale input. Intuitively, RGB inputs should provide more information and allow PoseMatcher to better separate textured regions of an object as well as from the background. Surprisingly, we found that using RGB data improves our method. Grayscale only reaches an ADD-(S) accuracy of 84.1% on Linemod while RGB inputs reach an accuracy 87.5%. All our experiments and further ablations use RGB inputs.

4.4. Evaluation Results

Our results on Linemod and the YCB-V datasets use the standard 2D bounding boxes provided by the BOP Challenge [14]. To note these are trained using synthetic versions of the target datasets which is done in order for a fair

Three-View	IO-Layer	Pruning	2D-Ref	3D-Ref	Grayscale	Linemod	YCB-V
			✓			76.9	-
✓						81.1	22.1
✓			✓			81.4	22.5
✓	✓					83.4	24.1
✓	✓		✓			83.4	24.2
✓	✓	✓	✓			83.9	25.8
✓	✓	✓		✓		**87.5**	**31.3**
✓	✓	✓		✓	✓	84.1	27.4

Table 2: **Ablation studies for each component.** We present the results on Linemod and YCB-V measured by ADD-(S). The first line are the results gathered from OnePose++ [38] The model used for the second and third rows refer is identical to the OnePose++ [12] model trained with our three-view based pipeline.

comparison to other methods that rely on the same detections. To measure the performance of PoseMatcher, we employ the same metrics used by prior methods. The ADD(-S) metric considers a pose correct if the average distance to the groundtruth falls below a threshold, usually 10% of the objects diameter, with a slightly modified version for symmetric objects [45].

We outperform every existing one-shot method by a significant margin, specially on more complicated objects such as the Ape or the Cat. We have a large advantage over DeepIM [22] which needs a strong pose initialization from PoseCNN [45] whereas PoseMatcher does not require any type of pose initialization or the 3D mesh model. It is interesting to observe that we also outperform PVNet[30], an instance-level pose estimator.

4.5. Ablation Studies

Our new training pipeline elevates the results from OnePose++ [12]. By learning meaningful descriptors specifically trained for pose estimation instead of pretrained ones, we are able increase the quality of the feature matching thereby improving the pose accuracy by 5%. This confirms that carefully designing a training pipeline can yield better results.

The IO-Layer, our novel attention based layer, yields an improvement of 2% on both datasets. While it does not increase runtime performance when compared to OnePose++ [38], we can see an improvement stemming from the use of specialized modality weights.

Replacing 2D based refinement with pose specific 3D-Refinement improved our results significantly. We found improvements in both Linemod and the YCB-V datasets. The improvement cannot be solely observed by measuring the ADD-(S) since the average threshold for Linemod is mm. We provide further information for Linemod on Fig. 5 where the error curves for each object in Linemod can be seen. The 3D-refinement has a much larger impact on lower thresholds accuracies than higher ones. An addition like 3D-refinement is invaluable for applications that require high precision.

We found 2D-refinement to result in no significant improvements. This step does not correct significant matching errors and PnP seems to overcome small imperfections stemming from resolution errors. PnP is also subject to biasing for translation as observed by prior works [42, 23]. Our results on 3D-refinement show a large increase in the translation accuracy when we use 3D-refinement over 2D-refinement, which is also reflected on ADD-(S).

4.5.1 Object pruning ablations

.

As discussed before, pruning an object provides two advantages. Firstly, it eliminates self-occluded regions of the object which removes possible noise during feature matching. Secondly, by reducing the size of the pointcloud, not only is the complexity of the attention modules reduced but also due to less keypoints correspondences PnP becomes faster to compute. In practice, without pruning and measured on our hardware, PoseMatcher takes 160ms (2D/3D refinements are not greatly affected by pointcloud size). For 10%, 25%, 50%, 75% and 90% prunning, PoseMatcher takes 151, 146, 135, 118, 100 ms. OnePose++ runtime is equivalent to our method without pruning. However, pruning a large amount of keypoints leads to poorer results as it excludes essential keypoints. We perform inference time ablation studies to help us determine the optimal pruning percentage and plot the results of percentages ranging from 10% to 90% after each IO-Layer in Fig 4 and measure the ADD-(S) accuracy on Linemod. If a low amount of keypoints are removed, the performance is not highly affected. However, we see a small drop at 75% with a large drop when crop 90% of the existing pointcloud. The latter might be due to the number of initial keypoints as the number of keypoints drops to 150 keypoints. Although increasing the number of pruned keypoints leads to higher throughput, the peak accuracy results is at 50%.

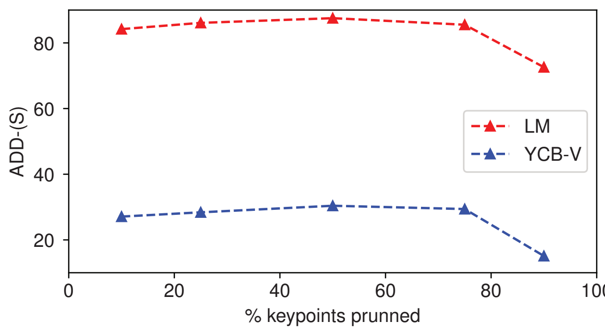


Figure 4: **Pruning ablations.** With this ablation, we show that PoseMatcher is not reliant on a large number of keypoints and only degrades significantly when more than 75% of the keypoints are prunned.

%	Ape	BW	Camera	Can	Cat	Driller	Duck	EB*	Glue*	HP	Iron	Lamp	Phone	Avg
10	2.1	15.1	8.8	0.0	1.9	6.4	0	25	2.5	0	12.4	2.1	12.4	6.8
25	5.9	24.1	15.2	12.4	13.1	12.4	1.2	45.1	8.3	0	24.4	12.6	29.1	15.7
50	12.4	59.4	35.6	19.7	19.3	34.7	9.4	78.9	41.9	5.7	34.1	21.8	57.3	33.1
75	39.9	78.9	81.2	76.6	57.3	65.1	27.1	92.1	72.1	46.8	87.1	79.	88.5	68.6
100	59.2	98.1	93.4	96	88.0	98.4	54.1	97.8	91.5	73.4	97.9	98.1	92.1	87.5

Table 3: **Number of templates.** We present the results on Linemod for % of training images used. Each object has around 180 images therefore 10% is only 18 templates to use.

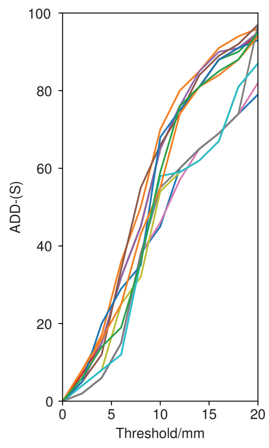
4.5.2 Number of Templates

For all our experiments on Linemod, we have been using all the available training images in order to sample template keypoints. Due to our training pipeline, PoseMatcher is robust to incomplete and noisy templates. We perform an ablation study over the number of templates needed. Since each object in Linemod contains different number of training images, we present the percentage of training images used. To sample which images to use, we sample from the available viewpoints using furthest point sampling [30] w.r.t. orientations in order to cover the widest range of viewpoints. PoseMatcher almost achieves the same level of accuracy as OnePose++ [12] using only 75% of the available training images.

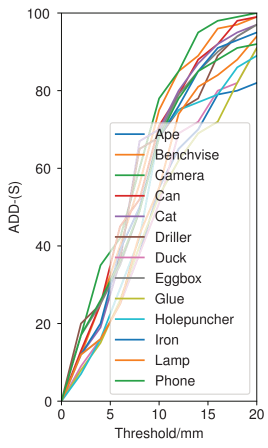
5. Conclusions

We proposed PoseMatcher, a novel model free one-shot pose estimator based on deep feature matching. Given a sequence of template images, we can reconstruct a feature point cloud and extract matches from a query image. In order to avoid using pre-trained descriptor models, we introduce a new training pipeline that allows us to train from scratch. With this simple addition, we show that we can improve OnePose++ [12] without any additional changes. We build on top of OnePose++ by designing a new attention layer IO-Layer, designed specifically for image to object matching. Additionally, we propose improvements to the pipeline including object pruning and 3D based refinement.

Limitations. Unfortunately we found that PoseMatcher has limited domain adaptation capability. If there is a large domain gap between the template domain (ex. synthetic renderings) and queries PoseMatcher is unable to correctly match keypoints, even coarsely. We specifically find this on YCB-V where each scene contains different levels of sensor noise and illumination. We also found symmetric objects particularly difficult, even though object pruning did show improvements.



(a) 2D-refinement



(b) 3D-refinement

Figure 5: **Difference between refinement operations.** We can see that for all objects in Linemod, the 3D-refinement has a big impact on lower thresholds.

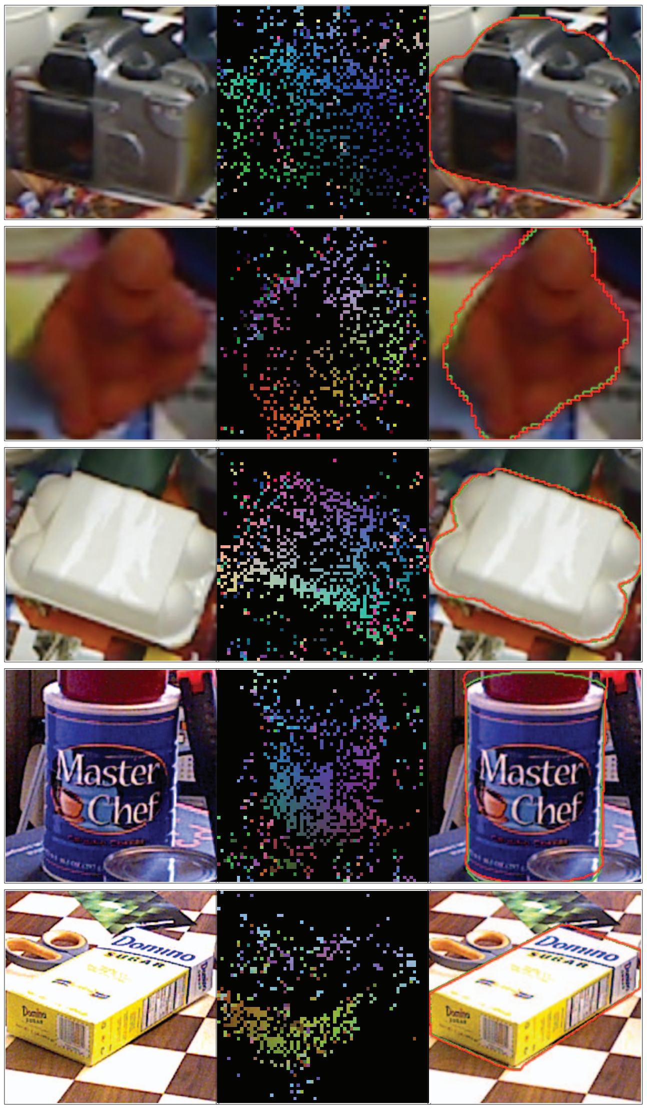


Figure 6: **Visualization of matching keypoints.** Here we show PoseMatcher output $\mathcal{C}$, where we assigned a normalized coordinate of the highest confidence keypoint to the corresponding pixel. The last two rows are examples from the YCB-V dataset.

References

[1] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. DensePose: Dense human pose estimation in the wild. In *CVPR*, 2018. 2

[2] Dingding Cai, Janne Heikkilä, and Esa Rahtu. Sc6d: Symmetry-agnostic and correspondence-free 6d object pose estimation. *arXiv preprint arXiv:2208.02129*, 2022. 1, 5

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 4

[4] Pedro Castro and Tae-Kyun Kim. Crt-6d: Fast 6d object pose estimation with cascaded refinement transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5746–5755, 2023. 1, 2, 5, 6

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 6

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018. 6

[7] Yan Di, Fabian Manhardt, Gu Wang, Xiangyang Ji, Nassir Navab, and Federico Tombari. So-pose: Exploiting self-occlusion for direct 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 6

[8] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. 3, 6

[9] Yang Fu and Xiaolong Wang. Category-level 6d object pose estimation in the wild: A semi-supervised learning approach and a new dataset. *arXiv preprint arXiv:2206.15436*, 2022. 2, 6

[10] Rasmus Laurvig Haugaard and Anders Glent Buch. Surfemb: Dense and continuous correspondence distributions for object pose estimation with learnt surface embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6749–6758, 2022. 1, 2, 4, 6

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[12] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and Xiaowei Zhou. Onepose++: Keypoint-free one-shot object pose estimation without CAD models. In *Advances in Neural Information Processing Systems*, 2022. 2, 3, 4, 5, 6, 7, 8

[13] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. *CoRR*, abs/1711.04061, 2017. 1, 2

[14] Tomáš Hodaň, Frank Michel, Eric Brachmann, Wadim Kehl, Anders Glent Buch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiří Matas, and Carsten Rother. Bop: Benchmark for 6d object pose estimation. *ECCV*, 2018. 2, 6

[15] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[16] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M. Kitani. Repose: Fast 6d object pose refinement via deep texture rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[17] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020. 4

[18] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1

[20] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, 2020. 2, 5

[21] Fu Li, Hao Yu, Ivan Shugurov, Benjamin Busam, Shaowu Yang, and Slobodan Ilic. Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation. *arXiv preprint arXiv:2203.04802*, 2022. 2

[22] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 5, 7

[23] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *ICCV*, 2019. 5, 6, 7

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2017. 4

[25] Yuan Liu, Yilin Wen, Sida Peng, Cheng Lin, Xiaoxiao Long, Taku Komura, and Wenping Wang. Gen6d: Generalizable model-free 6-dof object pose estimation from rgb images. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 298–315. Springer, 2022. 2, 6

[26] Xiaoyong Lu, Yaping Yan, Bin Kang, and Songlin Du. Paraformer: Parallel attention transformer for efficient feature matching. *arXiv preprint arXiv:2303.00941*, 2023. 4

[27] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3D object pose estimation. In *ECCV*, 2018. 2

[28] Keunhong Park, Arsalan Mousavian, Yu Xiang, and Dieter Fox. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10710–10719, 2020. 2

[29] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *ICCV*, 2019. 2

[30] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 2, 6, 7, 8

[31] Giorgia Pitteri, Aurélie Bugeau, Slobodan Ilic, and Vincent Lepetit. 3d object detection and pose estimation of unseen objects in color images with local surface embeddings. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[32] Giorgia Pitteri, Slobodan Ilic, and Vincent Lepetit. Cornet: generic 3d corners for 6d pose estimation of new objects without retraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[33] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *ICCV*, 2017. 2

[34] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 4

[35] Ivan Shugurov, Fu Li, Benjamin Busam, and Slobodan Ilic. Osop: A multi-stage one shot object pose estimation framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6835–6844, 2022. 2

[36] Juil Sock, Guillermo Garcia-Hernando, Anil Armagan, and Tae-Kyun Kim. Introducing pose consistency and warp-alignment for self-supervised 6d object pose estimation in color images. In *2020 International Conference on 3D Vision (3DV)*, pages 291–300, 2020. 1, 2, 6

[37] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021. 3, 4, 5, 6

[38] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. OnePose: One-shot object pose estimation without CAD models. *CVPR*, 2022. 2, 3, 4, 5, 6, 7

[39] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-Time seamless single shot 6D object pose prediction. In *CVPR*, 2018. 2

[40] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *Advances in Neural Information Processing Systems*, 33:14254–14265, 2020. 4

[41] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6D: Self-supervised monocular 6d object pose estimation. In *European Conference on Computer Vision (ECCV)*, pages 108–125, 2020. 1, 2, 6

[42] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 5, 6, 7

[43] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[44] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. Matchformer: Interleaving attention in transformers for feature matching. In *Asian Conference on Computer Vision*, 2022. 4

[45] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018. 7

[46] Yang Xiao, Xuchong Qiu, Pierre-Alain Langlois, Mathieu Aubry, and Renaud Marlet. Pose from shape: Deep pose estimation for arbitrary 3d objects. *arXiv preprint arXiv:1906.05105*, 2019. 1

[47] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021. 2

[48] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *ICCV*, 2019. 2

[49] Kaifeng Zhang, Yang Fu, Shubhankar Borse, Hong Cai, Fatih Porikli, and Xiaolong Wang. Self-supervised geometric correspondence for category-level 6d object pose estimation in the wild. *arXiv preprint arXiv:2210.07199*, 2022. 2