DeepIM: Deep Iterative Matching for 6D Pose Estimation

Yi Li[1], Gu Wang[1], Xiangyang Ji[1], Yu Xiang[2], and Dieter Fox[2]

[1] Tsinghua University and BNRist
[2] University of Washington and NVIDIA Research
yili.matrix@gmail.com, wangg16@mails.tsinghua.edu.cn,
xyji@tsinghua.edu.cn, {yux, dieterf}@nvidia.com

Abstract. Estimating the 6D pose of objects from images is an important problem in various applications such as robot manipulation and virtual reality. While direct regression of images to object poses has limited accuracy, matching rendered images of an object against the input image can produce accurate results. In this work, we propose a novel deep neural network for 6D pose matching named DeepIM. Given an initial pose estimation, our network is able to iteratively refine the pose by matching the rendered image against the observed image. The network is trained to predict a relative pose transformation using an untangled representation of 3D location and 3D orientation and an iterative training process. Experiments on two commonly used benchmarks for 6D pose estimation demonstrate that DeepIM achieves large improvements over state-of-the-art methods. We furthermore show that DeepIM is able to match previously unseen objects.

Keywords: 3D Object Recognition, 6D Object Pose Estimation

1 Introduction

Localizing objects in 3D from images is important in many real world applications. For instance, in a robot manipulation task, the ability to recognize the 6D pose of objects, i.e., 3D location and 3D orientation of objects, provides useful information for grasp and motion planning. In a virtual reality application, 6D object pose estimation enables virtual interactions between humans and objects. While several recent techniques have used depth cameras for object pose estimation, such cameras have limitations with respect to frame rate, field of view, resolution, and depth range, making it very difficult to detect small, thin, transparent, or fast moving objects. Unfortunately, RGB-only 6D object pose estimation is still a challenging problem, since the appearance of objects in the images changes according to a number of factors, such as lighting, pose variations, and occlusions between objects. Furthermore, a robust 6D pose estimation method needs to handle both textured and textureless objects.

Traditionally, the 6D pose estimation problem has been tackled by matching local features extracted from an image to features in a 3D model of the object [16,23,4]. By using the 2D-3D correspondences, the 6D pose of the object

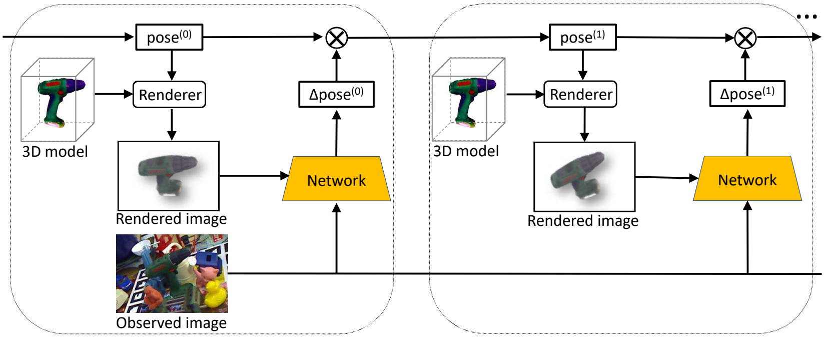


Fig. 1: We propose DeepIM, a deep iterative matching network for 6D object pose estimation. The network is trained to predict a relative SE(3) transformation that can be applied to an initial pose estimation for iterative pose refinement.

can be recovered. Unfortunately, such methods cannot handle textureless objects well since only few local features can be extracted for them. To handle textureless objects, two classes of approaches were proposed in the literature. Methods in the first class learn to estimate the 3D model coordinates of pixels or key points of the object in the input image. In this way, the 2D-3D correspondences are established for 6D pose estimation [1,20,26]. Methods in the second class convert the 6D pose estimation problem into a pose classification problem by discretizing the pose space [9] or into a pose regression problem [29]. These methods can deal with textureless objects, but they are not able to achieve highly accurate pose estimation, since small errors in the classification or regression stage directly lead to pose mismatches. A common way to improve the pose accuracy is pose refinement: Given an initial pose estimation, a synthetic RGB image can be rendered and used to match against the target input image. Then a new pose is computed to increase the matching score. Existing methods for pose refinement use either hand-crafted image features [27] or matching score functions [20].

In this work, we propose DeepIM, a new refinement technique based on a deep neural network for iterative 6D pose matching. Given an initial 6D pose estimation of an object in a test image, DeepIM predicts a relative SE(3) transformation that matches a rendered view of the object against the observed image. By iteratively re-rendering the object based on the improved pose estimates, the two input images to the network become more and more similar, thereby enabling the network to generate more and more accurate pose estimates. Fig. 1 illustrates the iterative matching procedure of our network for pose refinement.

This work makes the following main contributions. i) We introduce a deep network for iterative, image-based pose refinement that does not require any hand-crafted image features, automatically learning an internal refinement mechanism. ii) We propose an untangled representation of the SE(3) transformation between object poses to achieve accurate pose estimates. This representation also enables our approach to refine pose estimates of unseen objects. iii) We have conducted extensive experiments on the LINEMOD [9] and the Occlusion [1] datasets to evaluate the accuracy and various properties of DeepIM. These experiments show that our approach achieves large improvements over state-of-the-art RGB-only methods on both datasets. Furthermore, initial experiments demonstrate that DeepIM is able to accurately match poses for textureless objects (T-LESS [10]) and for unseen objects [28]. The rest of the paper is organized as follows. After reviewing related works in Section 2, we describe our approach for pose matching in Section 3. Experiments are presented in Section 4, and Section 5 concludes the paper.

2 Related work

RGB-D based 6D Pose Estimation: When depth information is available, it can be combined with RGB images to improve 6D pose estimation. A common strategy of using depth is to convert a depth image into a 3D point cloud, and then match the 3D model of an object against the 3D point cloud. For example, [9] render the 3D model of an object into templates of surface normals, and then match these templates against normals computed from the point cloud. [1,2,17] regress each pixel on the object in the input image to the 3D coordinate of that pixel on the 3D model. When depth images are available, the 3D coordinate regression establishes correspondences between 3D scene points and 3D model points, from which the 6D pose of the object can be computed by solving a least-squares problem. For pose refinement, the Iterative Closest Point (ICP) algorithm is widely used to refine an initial pose estimate [9,17,30]. However, ICP is sensitive to the initial estimate and may converge to local minima.

RGB based 6D Pose Estimation: Traditionally, pose estimation using RGB images is tackled by matching local features [16,23,4]. However, these methods cannot handle textureless objects very well. Recent approaches apply machine learning, especially deep learning, for 6D pose estimation using RGB images only [1,13]. The state-of-the-art methods [20,11,26,29] augment deep learning based object detection or segmentation methods [8,15,14,21] for 6D pose estimation. However, the performance of these methods is still not comparable to RGB-D based methods. We believe that this performance gap is so large due to the lack of an effective pose refinement procedure using RGB images only. Our work is complementary to existing 6D pose estimation methods by providing a novel iterative pose matching network for pose refinement on RGB images.

The approaches most relevant to ours are the object pose refinement network in [20] and the iterative hand pose estimation approaches in [3,19]. Compared to these techniques, our network is designed to directly regress to relative SE(3) transformations. We are able to do this due to our untangled representation of rotation and translation and the reference frame we used for rotation, which also allows our approach to match unseen objects. As shown in [18], the choice of reference frame is important to achieve good pose estimation results. Our work is also related to recent visual servoing methods based on deep neural networks [24,5] that estimate the relative camera pose between two image frames, while we focus on 6D pose refinement of objects.

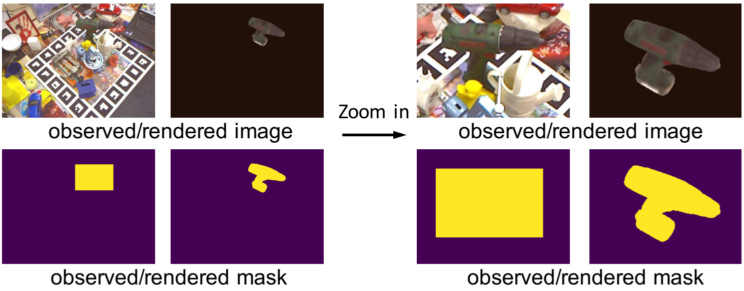


Fig. 2: DeepIM operates on a zoomed in, up-sampled input image, the rendered image, and the two object masks (480 × 640 in our case after zooming in).

3 DeepIM Framework

In this section, we describe our deep iterative matching network for 6D pose estimation. Given an observed image and an initial pose estimate of an object in the image, we design the network to directly output a relative SE(3) transformation that can be applied to the initial pose to improve the estimate. We first present our strategy of zooming in the observed image and the rendered image that are used as inputs of the network. Then we describe our network architecture for pose matching. After that, we introduce an untangled representation of the relative SE(3) transformation and a new loss function for pose regression. Finally, we describe our procedure for training and testing the network.

3.1 High-resolution Zoom In

It can be difficult to extract useful features for matching if objects in the input image are very small. To obtain enough details for pose matching, we zoom in the observed image and the rendered image before feeding them into the network, as shown in Fig. 2. Specifically, in the i-th stage of the iterative matching, given a 6D pose estimate $\mathbf{p}^{(i-1)}$ from the previous step, we render a synthetic image using the 3D object model viewed according to $\mathbf{p}^{(i-1)}$. We additionally generate one foreground mask for the observed image and rendered image. The four images are cropped using an enlarged bounding box according to the observed mask and the rendered mask, where we make sure the enlarged bounding box has the same aspect ratio as the input image and is centered at the 2D projection of the origin of the 3D object model. Finally, we zoom in and perform bilinear up-sampling to achieve the same size as the original image (480 × 640 in our experiments). Importantly, the aspect ratio of the object is not changed during this operation.

3.2 Network Structure

Fig. 3 illustrates the network architecture of DeepIM. The observed image, the rendered image, and the two masks, are concatenated into an eight-channel tensor input to the network (3 channels for observed/rendered image, 1 channel for

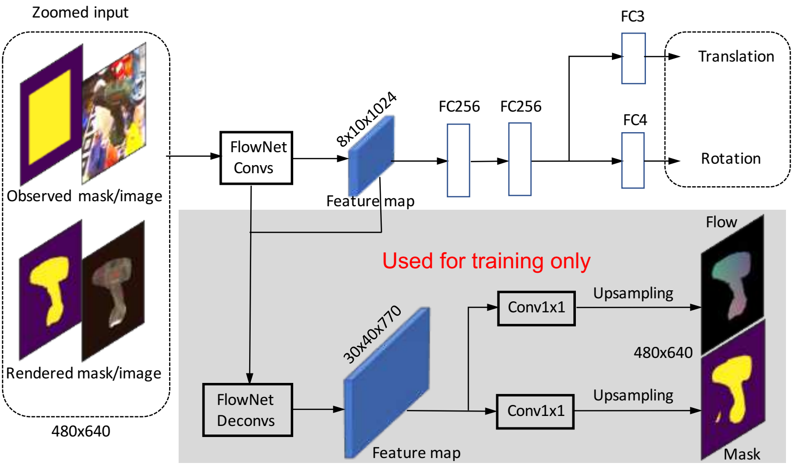


Fig. 3: DeepIM uses a FlowNetSimple backbone to predict a relative SE(3) transformation to match the observed and rendered image of an object.

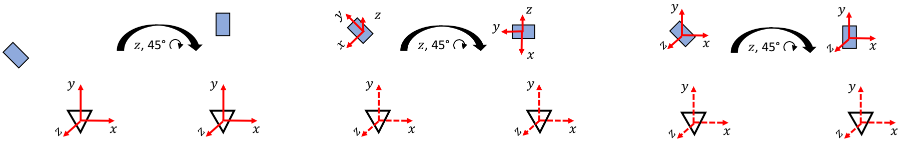
each mask). We use the FlowNetSimple architecture from [6] as the backbone network, which is trained to predict optical flow between two images. We tried using the VGG16 image classification network [25] as the backbone network, but the results were very poor, confirming the intuition that a representation related to optical flow is very useful for pose matching. The pose estimation branch takes the feature map after 11 convolution layers from FlowNetSimple as input. It contains two fully-connected layers each with dimension 256, followed by two additional fully-connected layers for predicting the quaternion of the 3D rotation and the 3D translation, respectively. During training, we also add two auxiliary branches to regularize the feature representation of the network and increase training stability. One branch is trained for predicting optical flow between the rendered image and the observed image, and the other branch for predicting the foreground mask of the object in the observed image.

3.3 Untangled Transformation Representation

The representation of the relative SE(3) transformation $\mathbf{\Delta p}$ between the current pose estimate and the target pose has important ramifications for the performance of the network. Consider we represent the object pose and transformation in the camera coordinate (Naive Coordinate in Fig. 4(a)). Denote the relative rotation and translation as $[\mathbf{R}_{\mathbf{\Delta}}|\mathbf{t}_{\Delta}]$. Given a source object pose $[\mathbf{R}_{\mathrm{src}}|\mathbf{t}_{\mathrm{src}}]$, the transformed target pose would be as follows:

$$\mathbf{R}_{\mathrm{tgt}} = \mathbf{R}_{\Delta}\mathbf{R}_{\mathrm{src}}, \quad \mathbf{t}_{\mathrm{tgt}} = \mathbf{R}_{\Delta}\mathbf{t}_{\mathrm{src}} + \mathbf{t}_{\Delta}, \tag{1}$$

where $[\mathbf{R}_{\mathrm{tgt}}|\mathbf{t}_{\mathrm{tgt}}]$ denotes the target pose. The $\mathbf{R}_{\Delta}\mathbf{t}_{\mathrm{src}}$ term indicates that a rotation will cause the object not only to rotate, but also translate in the image



(a) Naïve Coordinate (b) Model Coordinate (c) Camera Coordinate

Fig. 4: Three different coordinate systems for the relative rotation.

even if the translation vector $\mathbf{t}_\Delta$ equals to zero. Furthermore, the translation $\mathbf{t}_\Delta$ is in the metric of the 3D space (meter, for instance), which couples object size with distance in the metric space, thereby requiring the network to memorize the size of each object if it has to translate a mis-match in images to distance offset. It is obvious that such a representation is not appropriate. To eliminate such problems, we propose to decouple the estimation of $\mathbf{R}_\Delta$ and $\mathbf{t}_\Delta$. For rotation, we move the center of rotation from the origin of the camera to the center of the object in the camera frame, given by the current pose estimate. Then a rotation would not change the translation of the object in the camera frame. The remaining question is how to choose the axes of the coordinate frame for rotation. One way is to use the axes of the coordinate frame as specified in the 3D object model (Model Coordinate in Fig. 4(b)). However, such a representation would require the network to memorize the coordinate frames of each object, which makes the training more difficult and cannot be generalized to pose matching of unseen objects. Instead, we use axes parallel to the axes of the camera frame when computing the relative rotation (Camera Coordinate in Fig. 4(c)). By doing so, the network can be trained to estimate the relative rotation independently of the coordinate frame of the 3D object model.

To estimate the relative translation, let $\mathbf{t}_{\text{tgt}} = (x_{\text{tgt}}, y_{\text{tgt}}, z_{\text{tgt}})$ and $\mathbf{t}_{\text{src}} = (x_{\text{src}}, y_{\text{src}}, z_{\text{src}})$ be the target translation and the source translation. Then a straightforward way to represent it is $\mathbf{t}_\Delta = (\Delta_x, \Delta_y, \Delta_z) = \mathbf{t}_{\text{tgt}} - \mathbf{t}_{\text{src}}$. However, it is not easy for the network to estimate the relative translation in 3D space given only 2D images without depth information. The network has to recognize the size of the object, and map the translation in 2D space to 3D according to the object size. Such a representation is not only difficult for the network to learn, but also has problems when dealing with uknown objects or objects with similar appearance but different sizes. Instead of training the network to directly regress to the vector in the 3D space, we propose to regress to the object changes in the 2D image space. Specifically, we train the network to regress to the relative translation $\mathbf{t}_\Delta = (v_x, v_y, v_z)$, where v_x and v_y denote the number of pixels the object should move along the image x-axis and y-axis and v_z is the scale change of the object:

$$\begin{aligned} v_x &= f_x(x_{\text{tgt}}/z_{\text{tgt}} - x_{\text{src}}/z_{\text{src}}), \\ v_y &= f_y(y_{\text{tgt}}/z_{\text{tgt}} - y_{\text{src}}/z_{\text{src}}), \\ v_z &= \log(z_{\text{src}}/z_{\text{tgt}}), \end{aligned} \tag{2}$$

where f_x and f_y denote the focal lengths of the camera. The scale change v_z is defined to be independent of the absolute object size or distance by using the ratio between the distances of the rendered and observed object. We use logarithm for v_z to make sure that value zero corresponds to no change in scale or distance. Considering the fact that f_x and f_y are constant for a specific dataset, we simply fix it to 1 in training and testing the network.

Our representation of the relative transformation has several advantages. First, rotation does not influence the estimation of translation, so that the translation no longer needs to offset the movement caused by rotation around the camera center. Second, the intermediate variables v_x, v_y, v_z represent simple translations and scale change in the image space. Third, this representation does not require any prior knowledge of the object. Using such a representation, the DeepIM network can operate independently of the actual size of the object and its internal model coordinate framework. It only has to learn to transform the rendered image such that it becomes more similar to the observed image.

3.4 Matching Loss

A straightforward way to train the pose estimation network is to use separate loss functions for rotation and translation. For example, we can use the angular distance between two rotations to measure the rotation error and use the L2 distance to measure the translation error. However, using two different loss functions for rotation and translation suffers from the difficulty of balancing the two losses. [12] proposed a geometric reprojection error as the loss function for pose regression that computes the average distance between the 2D projections of 3D points in the scene using the ground truth pose and the estimated pose. Considering the fact that we want to accurately predict the object pose in 3D, we introduce a modified version of the geometric reprojection loss in [12], and we call it the Point Matching Loss. Given the ground truth pose $\mathbf{p} = [\mathbf{R}|\mathbf{t}]$ and the estimated pose $\hat{\mathbf{p}} = [\hat{\mathbf{R}}|\hat{\mathbf{t}}]$, the point matching loss is computed as:

$$L_{\text{pose}}(\mathbf{p}, \hat{\mathbf{p}}) = \frac{1}{n} \sum_{i=1}^{n} L_1\big((\mathbf{R}\mathbf{x}_i + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_i + \hat{\mathbf{t}})\big), \tag{3}$$

where $\mathbf{x}_i$ denotes a randomly selected 3D point on the object model and n is the total number of points (we choose 3,000 points in our experiments). The point matching loss computes the average L1 distance between 3D points transformed by the ground truth pose and the estimate pose. In this way, it measures how the transformed 3D models match against each other for pose estimation.

3.5 Training and Testing

In training, we assume that we have 3D object models and images annotated with ground truth 6D object poses. By adding noises to the ground truth poses as the initial poses, we can generate the required observed and rendered inputs to the network along with the pose target output that is the pose difference between the ground truth pose and the noisy pose. Then we can train the network to predict the relative transformation between the initial pose and the target pose.

During testing, we find that the iterative pose refinement can significantly improve the accuracy. To see, let $\mathbf{p}^{(i)}$ be the pose estimate after the i-th iteration of the network. If the initial pose estimate $\mathbf{p}^{(0)}$ is relatively far from the correct pose, the rendered image $\mathbf{x}_{\text{rend}}(\mathbf{p}^{(0)})$ may have only little viewpoint overlap with the observed image $\mathbf{x}_{\text{obs}}$. In such cases, it is very difficult to accurately estimate the relative pose transformation $\mathbf{\Delta p}^{(0)}$ directly. This task is even harder if the network has no priori knowledge about the object to be matched. In general, it is reasonable to assume that if the network improves the pose estimate $\mathbf{p}^{(i+1)}$ by updating $\mathbf{p}^{(i)}$ with $\mathbf{\Delta p}^{(i)}$ in the i-th iteration, then the image rendered according to this new estimate, $\mathbf{x}_{\text{rend}}(\mathbf{p}^{(i+1)})$ is also more similar to the observed image $\mathbf{x}_{\text{obs}}$ than $\mathbf{x}_{\text{rend}}(\mathbf{p}^{(i)})$ was in the previous iteration, thereby providing input that can be matched more accurately.

However, we found that, if we train the network to regress the relative pose in a single step, the estimates of the trained network do not improve over multiple iterations in testing. To generate a more realistic data distribution for training similar to testing, we perform multiple iterations during training as well. Specifically, for each training image and pose, we apply the transformation predicted from the network to the pose and use the transformed pose estimate as another training example for the network in the next iteration. By repeating this process multiple times, the training data better represents the test distribution and the trained network also achieves significantly better results during iterative testing (such an approach has also proven useful for iterative hand pose matching [19]).

4 Experiments

We conduct extensive experiments on the LINEMOD dataset [9] and the Occlusion LINEMOD dataset [2] to evaluate our DeepIM framework for 6D object pose estimation. We test different properties of DeepIM and show that it surpasses other RGB-only methods by a large margin. We also show that our network can be applied to pose matching of unseen objects during training.

4.1 Implementation Details

Training: We use the pre-trained FlowNetSimple [6] to initialize the weights in our network. Weights of the new layers are randomly initialized, except for the additional weights in the first conv layer that deals with the input masks and the fully-connected layer that predicts the translation, which are initialized with zeros. Other than predicting the pose transformation, the network also predicts the optical flow and the foreground mask. Although including the two additional losses in training does not increase the pose estimation performance, we found that they help to make the training more stable. Specifically, we use the optical flow loss L_{flow} as in FlowNet [6] and the sigmoid cross-entropy loss as the mask loss L_{mask}. Two deconvolutional blocks in FlowNet are inherited to produce the feature map used for the mask and the optical flow prediction, whose spatial scale is 0.0625. Two 1 × 1 convolutional layers with output channel 1 (mask prediction) and 2 (flow prediction) are appended after this feature map. The predictions are then bilinearly up-sampled to the original image size (480×640) to compute losses. The overall loss is $L = \alpha L_{\text{pose}} + \beta L_{\text{flow}} + \gamma L_{\text{mask}}$, where we use $\alpha = 0.1$, $\beta = 0.25$, $\gamma = 0.03$ throughout the experiments (except some of our ablation studies). Each training batch contains 16 images. We train the network with 4 GPUs where each GPU processes 4 images. We generate 4 items for each image as described in Sec. 3.1: two images and two masks. The observed mask is randomly dilated with no more than 10 pixels to avoid over-fitting.

Testing: The mask prediction branch and the optical flow branch are removed during testing. Since there is no ground truth segmentation of the object in testing, we use the tightest bounding box of the rendered mask $\mathbf{m}_{\text{rend}}$ instead, so the network searches the neighborhood near the estimated pose to find the target object to match. Unless specified, we use the pose estimates from PoseCNN [29] as the initial poses. Our DeepIM network runs at 12 fps per object using an NVIDIA 1080 Ti GPU with 2 iterations during testing.

4.2 Evaluation Metrics

We use the following three evaluation metrics for 6D object pose estimation. i) The *5°, 5cm* metric considers an estimated pose to be correct if its rotation error is within 5° and the translation error is below 5cm. ii) The *6D Pose* metric [9] computes the average distance between the 3D model points transformed using the estimated pose and the ground truth pose. For symmetric objects, we use the closet point distance in computing the average distance. An estimated pose is correct if the average distance is within 10% of the 3D model diameter. iii) The *2D Projection* metric computes the average distance of the 3D model points projected onto the image using the estimated pose and the ground truth pose. An estimated pose is correct if the average distance is smaller than 5 pixels.

4.3 Experiments on the LINEMOD Dataset

The LINEMOD dataset contains 15 objects. We train and test our method on 13 of them as other methods in the literature. We follow the procedure in [2] to split the dataset into the training and test sets, with around 200 images for each object in the training set and 1,000 images in the test set.

Training strategy: For every image, we generate 10 random poses near the ground truth pose, resulting in 2,000 training samples for each object in the training set. Furthermore, we generate 10,000 synthetic images for each object where the pose distribution is similar to the real training set. For each synthetic image, we generate 1 random pose near its ground truth pose. Thus, we have a total of 12,000 training samples for each object in training. The background of a synthetic image is replaced with a randomly chosen indoor image from PASCAL

Table 1: Ablation study of the number of iterations during training and testing.

train iter	init	1			2			4		
test iter		1	2	4	1	2	4	1	2	4
5cm 5°	19.4	57.4	58.8	54.6	76.3	86.2	86.7	70.2	83.7	85.2
6D Pose	62.7	77.9	79.0	76.1	83.1	88.7	89.1	80.9	87.6	88.6
Proj. 2D	70.2	92.4	92.6	89.7	96.1	97.8	97.6	94.6	97.4	97.5

VOC [7]. We train the networks for 8 epochs with initial learning rate 0.0001. The learning rate is divided by 10 after the 4th and 6th epoch, respectively.

Ablation study on iterative training and testing: Table 1 shows the results that use different numbers of iterations during training and testing. The networks with *train_iter* = 1 and *train_iter* = 2 are trained with 32 and 16 epochs respectively to keep the total number of updates the same as *train_iter* = 4. The table shows that without iterative training (*train_iter* = 1), multiple iteration testing does not improve, potentially even making the results worse (*test_iter* = 4). We believe that the reason is due to the fact that the network is not trained with enough rendered poses close to their ground truth poses. The table also shows that one more iteration during training and testing already improves the results by a large margin. The network trained with 2 iterations and tested with 2 iterations is slightly better than the one trained with 4 iterations and tested with 4 iterations. This may be because the LINEMOD dataset is not sufficiently difficult to generate further improvements by using 3 or 4 iterations. Since it is not straightforward to determine how many iterations to use in each dataset, we use 4 iterations during training and testing in all other experiments.

Ablation study on the zoom in strategy, network structures, transformation representations, and loss functions: Table 2 summarizes the ablation studies on various aspects of DeepIM. The "zoom" column indicates whether the network uses full images as its input or zoomed in bounding boxes up-sampled to the original image size. Comparing rows 5 and 7 shows that the higher resolution achieved via zooming in provides very significant improvements.

"Regressor": We train the DeepIM network jointly over all objects, generating a pose transformation independent of the specific input object (labeled "shared" in "regressor" column). Alternatively, we could train a different 6D pose regressor for each individual object by using a separate fully connected layer for each object after the final FC256 layer shown in Fig. 3. This setting is labeled as "sep." in Table 2. Comparing rows 3 and 7 shows that both approaches provide nearly indistinguishable results. But the shared network provides some efficiency gains.

"Network": Similarly, instead of training a single network over all objects, we could train separate networks, one for each object as in [20]. Comparing row 1 to 7 shows that a single, shared network provides better results than individual ones, which indicates that training on multiple objects can help the network learn a more general representation for matching.

Table 2: Ablation study on different design choices of the DeepIM network on the LINEMOD dataset.

Row	methods					5cm 5°	6D Pose	Proj. 2D
	zoom	regressor	network	coordinate	loss			
1	✓	-	sep.	camera	PM	83.3	87.6	96.2
2	✓	sep.	shared	model	PM	79.2	87.5	95.4
3	✓	sep.	shared	camera	PM	86.6	89.5	96.7
4		shared	shared	naive	PM	16.6	44.3	62.5
5		shared	shared	camera	PM	38.3	65.2	80.8
6	✓	shared	shared	camera	Dist	86.5	79.2	96.2
7	✓	shared	shared	camera	PM	85.2	88.6	97.5

"Coordinate": This column investigates the impact of our choice of coordinate frame to reason about object transformations, as described in Fig. 4. The row labeled "naive" provides results when choosing the camera frame of reference as the representation for the object pose, rows labeled "model" move the center of rotation to the object model and choose the object model coordinate frame to reason about rotations, and the "camera" rows provide our approach of moving the center into the object model while keeping the camera coordinate frame for rotations. Comparing rows 2 and 3 shows that reasoning in the camera rotation frame provides slight improvements. Furthermore, it should be noted that only our "camera" approach is able to operate on unseen objects. Comparing rows 4 and 5 shows the large improvements our representation achieves over the naive approach of reasoning fully in the camera frame of reference.

"Loss": The traditional loss for pose estimation is specified by the distance ("Dist") between the estimated and ground truth 6D pose coordinates, i.e., angular distance for rotation and euclidean distance for translation. Comparing rows 6 and 7 indicates that our point matching loss ("PM") provides significantly better results especially on the 6D pose metric, which is the most important measure for reasoning in 3D space.

Application to different initial pose estimation networks: Table 3 provides results when we initialize DeepIM with two different pose estimation networks. The first one is PoseCNN [29], and the second one is a simple 6D pose estimation method based on Faster R-CNN [22]. Specifically, we use the bounding box of the object from Faster R-CNN to estimate the 3D translation of the object. The center of the bounding box is treated as the center of the object. The distance of the object is estimated by maximizing the overlap of the projection of the 3D object model with the bounding box. To estimate the 3D rotation of the object, we add a rotation regression branch to Faster R-CNN as in PoseCNN. As we can see in Table 3, our network achieves very similar pose estimation accuracy even when initialized with the estimates from the extension of Faster R-CNN, which are not as accurate as those provided by PoseCNN [29].

Table 3: Ablation study on two different methods for generating initial poses on the LINEMOD dataset.

method	PoseCNN	PoseCNN +OURS	Faster R-CNN	Faster R-CNN +OURS
5cm 5°	19.4	85.2	11.9	83.4
6D Pose	62.7	88.6	33.1	86.9
Proj. 2D	70.2	97.5	20.9	95.7

Table 4: Comparison with state-of-the-art methods on the LINEMOD dataset

methods	[2]	BB8 w ref. [20]	SSD-6D w ref. [11]	Tekin et al. [26]	PoseCNN [29]	PoseCNN [29] +OURS
5cm 5°	40.6	69.0	-	-	19.4	**85.2**
6D Pose	50.2	62.7	79	55.95	62.7	**88.6**
Proj. 2D	73.7	89.3	-	90.37	70.2	**97.5**

Comparison with the state-of-the-art 6D pose estimation methods: Table 4 shows the comparison with the best color-only techniques on the LINEMOD dataset. DeepIM achieves very significant improvements over all prior methods, even those that also deploy refinement steps (BB8 [20] and SSD-6D [11]).

4.4 Experiments on the Occlusion LINEMOD Dataset

The Occlusion LINEMOD dataset proposed in [2] shares the same images used in LINEMOD [9], but annotated 8 objects in one video that are heavily occluded.

Training: For every real image, we generate 10 random poses as described in Sec. 4.3. Considering the fact that most of the training data lacks occlusions, we generated about 20,000 synthetic images with multiple objects in each image. By doing so, every object has around 12,000 images which are partially occluded, and a total of 22,000 images for each object in training. We perform the same background replacement and training procedure as in the LINEMOD dataset.

Comparison with the state-of-the-art methods: The comparison between our method and other RGB-only methods is shown in Fig. 5. We only show the plots with accuracies on the 2D Projection metric because these are the only results reported in [20] and [26] (results for eggbox and glue use a symmetric version of this accuracy). It can be seen that our method greatly improves the pose accuracy generated by PoseCNN and surpasses all other RGB-only methods by a large margin. It should be noted that BB8 [20] achieves the reported results only when using ground truth bounding boxes during testing. Our method is even competitive with the results that use depth information and ICP to refine the estimates of PoseCNN. Fig. 6 shows some pose refinement results from our method on the Occlusion LINEMOD dataset.

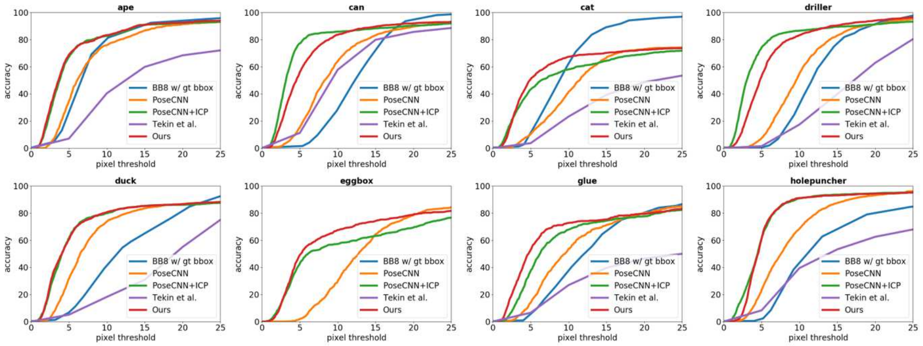


Fig. 5: Comparison with state-of-the-art methods on the Occlusion LINEMOD dataset [2]. Accuracies are measured via the Projection 2D metric.

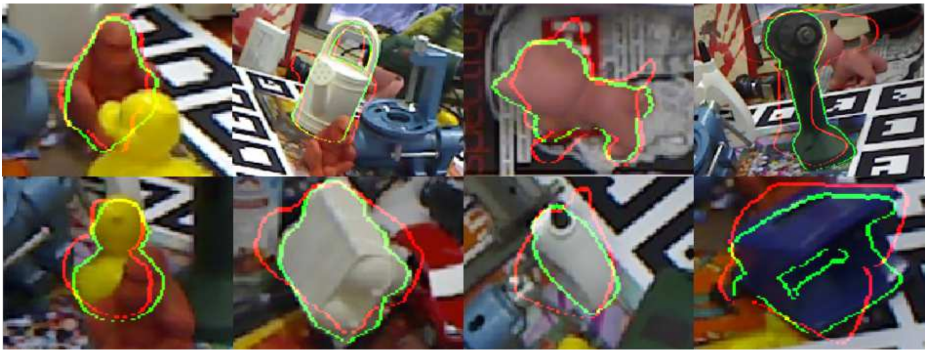


Fig. 6: Examples of refined poses on the Occlusion LILNEMOD dataset using the results from PoseCNN [29] as initial poses. The red and green lines represent the silhouettes of the initial estimates and our refined poses, respectively.

4.5 Application to Unseen Objects and Unseen Categories

As stated in Sec 3.3, we designed the untangled pose representation such that it is independent of the coordinate frame and the size of a specific 3D object model. Therefore, the pose transformations correspond to operations in the image space. This opens the question whether DeepIM can refine the poses of objects that are not included in the training set. In this experiment, we use the 3D models of airplanes, cars and chairs from the ModelNet dataset [28]. For each of these categories, we train a network on no more than 200 3D models and test its performance on 70 unseen 3D models from the same category. For training, we generate 50 images for each model and train the network for 4 epochs. We found that our network can perform accurate refinement on these unseen models. See Fig. 7 for example results. We also tested our framework on refining the poses of unseen object categories, where the training categories and the test categories are completely different. Please see the supplementary material for more details.

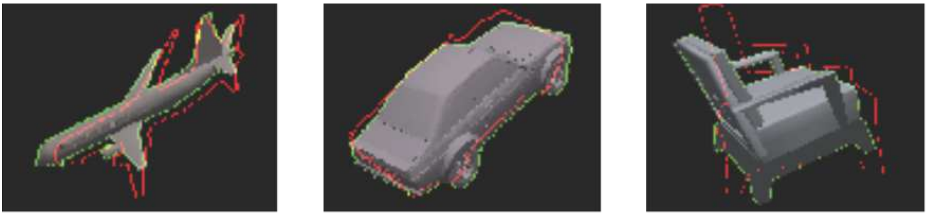


Fig. 7: Results on pose refinement of 3D models from the ModelNet dataset. These instances were not seen in training. The red and green lines represent the edges of the initial estimates and our refined poses.

5 Conclusion

In this work we introduce DeepIM, a novel framework for iterative pose matching using color images only. Given an initial 6D pose estimation of an object, we have designed a new deep neural network to directly output a *relative* pose transformation that improves the pose estimate. The network automatically learns to match object poses during training. We introduce an untangled pose representation that is also independent of the object size and the coordinate frame of the 3D object model. In this way, the network can even match poses of unseen objects, as shown in our experiments. Our method significantly outperforms state-of-the-art 6D pose estimation methods using color images only and provides performance close to methods that use depth images for pose refinement, such as using the iterative closest point algorithm. Example visualizations of our results on LINEMOD, ModelNet, and T-LESS can be found here: `https://rse-lab.cs.washington.edu/projects/deepim`.

This work opens up various directions for future research. For instance, we expect that a stereo version of DeepIM could further improve pose accuracy. Furthermore, DeepIM indicates that it is possible to produce accurate 6D pose estimates using color images only, enabling the use of cameras that capture high resolution images at high frame rates with a large field of view, providing estimates useful for applications such as robot manipulation.

Acknowledgments

This work was funded in part by a Siemens grant. We would also like to thank NVIDIA for generously providing the DGX station used for this research via the NVIDIA Robotics Lab and the UW NVIDIA AI Lab (NVAIL). This work was also Supported by National Key R&D Program of China 2017YFB1002202, NSFC Projects 61620106005, 61325003, Beijing Municipal Sci. & Tech. Commission Z181100008918014 and THU Initiative Scientific Research Program.

References

1. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6D object pose estimation using 3D object coordinates. In: European Conference on Computer Vision (ECCV) (2014)
2. Brachmann, E., Michel, F., Krull, A., Ying Yang, M., Gumhold, S., Rother, C.: Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3364–3372 (2016)
3. Carreira, J., Agrawal, P., Fragkiadaki, K., Malik, J.: Human pose estimation with iterative error feedback. In: IEEE conference on Computer Vision and Pattern Recognition (CVPR) (2016)
4. Collet, A., Martinez, M., Srinivasa, S.S.: The MOPED framework: Object recognition and pose estimation for manipulation. International Journal of Robotics Research (IJRR) **30**(10), 1284–1306 (2011)
5. Costante, G., Ciarfuglia, T.A.: LS-VO: Learning dense optical subspace for robust visual odometry estimation. IEEE Robotics and Automation Letters **3**(3), 1735–1742 (2018)
6. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: IEEE International Conference on Computer Vision (ICCV). pp. 2758–2766 (2015)
7. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision (ICCV) **88**(2), 303–338 (2010)
8. Girshick, R.: Fast R-CNN. In: IEEE International Conference on Computer Vision (ICCV). pp. 1440–1448 (2015)
9. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., , Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Asian Conference on Computer Vision (ACCV) (2012)
10. Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In: IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 880–888. IEEE (2017)
11. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: SSD-6D: Making rgb-based 3D detection and 6D pose estimation great again. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1521–1529 (2017)
12. Kendall, A., Cipolla, R.: Geometric loss functions for camera pose regression with deep learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
13. Krull, A., Brachmann, E., Michel, F., Ying Yang, M., Gumhold, S., Rother, C.: Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In: IEEE International Conference on Computer Vision (ICCV). pp. 954–962 (2015)
14. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision (ECCV). pp. 21–37 (2016)
15. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3431–3440 (2015)

16. Lowe, D.G.: Object recognition from local scale-invariant features. In: IEEE International Conference on Computer Vision (ICCV). vol. 2, pp. 1150–1157 (1999)
17. Michel, F., Kirillov, A., Brachmann, E., Krull, A., Gumhold, S., Savchynskyy, B., Rother, C.: Global hypothesis generation for 6D object pose estimation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
18. Mousavian, A., Anguelov, D., Flynn, J., Košecká, J.: 3D bounding box estimation using deep learning and geometry. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5632–5640 (2017)
19. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: IEEE International Conference on Computer Vision (ICCV) (2015)
20. Rad, M., Lepetit, V.: BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In: IEEE International Conference on Computer Vision (ICCV) (2017)
21. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (2016)
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS) (2015)
23. Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. International Journal of Computer Vision (IJCV) **66**(3), 231–259 (2006)
24. Saxena, A., Pandya, H., Kumar, G., Gaud, A., Krishna, K.M.: Exploring convolutional networks for end-to-end visual servoing. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 3817–3823 (2017)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
26. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6D object pose prediction. arXiv preprint arXiv:1711.08848 (2017)
27. Tjaden, H., Schwanecke, U., Schömer, E.: Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 124–132 (2017)
28. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D shapenets: A deep representation for volumetric shapes. In: IEEE conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015)
29. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017)
30. Zeng, A., Yu, K.T., Song, S., Suo, D., Walker, E., Rodriguez, A., Xiao, J.: Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 1386–1383 (2017)