DeepRM: Deep Recurrent Matching for 6D Pose Refinement

Alexander Avery, Andreas Savakis
Rochester Institute of Technology
Rochester, NY
{aja9675, andreas.savakis}@rit.edu

Abstract

Precise 6D pose estimation of rigid objects from RGB images is a critical but challenging task in robotics, augmented reality and human-computer interaction. To address this problem, we propose DeepRM, a novel recurrent network architecture for 6D pose refinement. DeepRM leverages initial coarse pose estimates to render synthetic images of target objects. The rendered images are then matched with the observed images to predict a rigid transform for updating the previous pose estimate. This process is repeated to incrementally refine the estimate at each iteration. The DeepRM architecture incorporates LSTM units to propagate information through each refinement step, significantly improving overall performance. In contrast to current 2-stage Perspective-n-Point based solutions, DeepRM is trained end-to-end, and uses a scalable backbone that can be tuned via a single parameter for accuracy and efficiency. During training, a multi-scale optical flow head is added to predict the optical flow between the observed and synthetic images. Optical flow prediction stabilizes the training process, and enforces the learning of features that are relevant to the task of pose estimation. Our results demonstrate that DeepRM achieves state-of-the-art performance on two widely accepted challenging datasets.

1. Introduction

Detecting objects and estimating their 6 dimensional pose (x, y, z, roll, pitch, yaw) in 3D space is a fundamental task in the field of computer vision and robotics. As such, it has many applications, the most common of which is robotic manipulation. For a robot to be able to effectively interact with an object, it must know the object’s pose in relation to itself. In the case of robotic grasping, the object’s position is used to determine the input to the inverse kinematic solver, which can then calculate the joint states necessary to grasp the object. Augmented reality is another important field requiring very precise pose estimation [1]. In this setting, pose estimation enables humans to interact with both physical and virtual objects in a seamless manner. Applications range across industries such as healthcare, manufacturing, education, and gaming.

Estimating the 6D pose from a single RGB image is an ill-posed problem due to the projection of the 3D scene onto the 2D image sensor. Because of this loss of dimensionality, many solutions rely on depth sensors to recover the depth information. Depth sensors, however, can be noisy and are typically limited by factors such as cost, power, form factor, range, resolution, frame rate, and sensitivity to external factors, e.g. sunlight [18,31]. Furthermore, recent advancements in computer vision and AI are enabling RGB only solutions to approach the same levels of accuracy as those with RGB-D sensors. In the 2020 BOP Challenge on 6D Object Localisation [14], CosyPose [17], an extension of DeepIM [18], relied only on RGB data and outperformed all but two RGB-D approaches. Our intention in this work is to close the gap between RGB and RGB-D approaches by focusing on pose refinement with RGB only data, enabling our solution to be used across a wider range of applications.

In this paper, we introduce DeepRM, a 6D pose refinement technique for rigid objects. Figure 1 shows a representative example of the DeepRM pose refinement process. DeepRM uses an iterative render-and-compare approach to incrementally refine an initial pose estimate. Given an initial coarse pose, a target object can be rendered with the
same camera intrinsics as the original observation. The rendered image can then be matched with the observed image to predict the rigid transform that aligns the object in the two images. By leveraging the geometric information implicitly contained within the 3D model of the object, updates to the 6D pose can be inferred without external depth information.

The proposed DeepRM method improves upon DeepIM [18] with several innovations, such as high resolution cropping, disentangled loss, variable renderer brightness, a scalable backbone based on EfficientNet [28], and most notably a recurrent network architecture. DeepRM is the first work that both leverages a recurrent neural network to directly regress 6D pose of rigid objects and provides a scalable framework for this task. Utilizing a recurrent architecture allows additional information to be propagated through each refinement step, significantly improving performance over non-recurrent methods.

The main contributions of this paper are: 1) we present DeepRM, an end-to-end trainable recurrent neural network architecture for 6D object pose refinement, that requires only a single RGB image as input. 2) DeepRM offers a scalable solution that can be adapted based on computational constraints in real-world scenarios. 3) DeepRM achieves state-of-the-art results on the challenging YCB-Video [33] and Occlusion LINEMOD [2] datasets.

2. RELATED WORK

2.1. 6D Object Pose Estimation

The goal of 6D object pose estimation is to determine an object’s fully constrained pose within 3D space. As the field is vast, we limit the discussion of related works to methods based on RGB data. Traditional methods utilized template matching techniques [11] or matched hand crafted feature points to a 3D CAD model and solved the Perspective-N-Point (PnP) problem [5]. Early deep learning based methods built upon the two-stage approach of feature detection followed by PnP. BB8 [23] first used this technique to regress the 8 corners of the bounding cuboid in 2D, and then solved for pose via PnP. Similar methods followed the same approach, but addressed other limiting factors such as efficiency [30] and robustness to occlusion [21].

To further address the problem of occlusion, PVNet [22] introduced a pixel-wise voting network using RANSAC, resulting in an estimator that is capable of detecting keypoints, even when they are occluded. The best results were achieved with 8 keypoints similarly to methods using bounding boxes. However, the sparsity of the keypoints in such approaches limits functionality under high levels of occlusion and truncation. To address this, a different line of research attempts to predict 3D coordinates for every pixel in the target image. By drastically increasing the number of 2D-3D correspondences, performance is maintained even under high occlusion. To handle the additional noise inherent to the dense predictions, PnP+RANSAC is needed to achieve robustness to outliers. Dense correspondence methods include DPOD [35], EPOS [13], and ZebraPose [27].

Recent works such as PoseCNN [33] attempt to directly regress the pose of objects from RGB images. PoseCNN uses a VGG16 [26] backbone to extract high dimensional feature maps. These shared feature maps are then utilized by three downstream tasks: semantic segmentation to localize and distinguish objects, translation prediction, and rotation prediction. The translation and rotation predictions are directly regressed by passing flattened feature maps through fully connected layers. The benefit of direct approaches is that they can be fully trained end-to-end, without surrogate loss functions as in the two-stage approaches.

The Geometry-guided Direct Regression Network (GDR-Net) [32] aims to achieve the end-to-end differentiability of direct methods, the geometry-guided accuracy of PnP methods and the robustness of dense methods. GDR-Net predicts dense pixel-wise correspondences, but then instead of using a non-differentiable PnP solver, it uses a convolutional Patch-PnP network to directly regress pose. SO-Pose [7] further extends this approach by leveraging self occlusion information to enforce cross-layer consistencies across the correspondence field, self-occlusion information, and 6D pose, resulting in a direct method that performs comparably to many refinement based techniques.

2.2. 6D Object Pose Refinement

Although recent methods such as GDR-Net [32] and SO-Pose [7], achieve high levels of accuracy compared to prior works, the ill-posedness of the problem still makes this task very challenging for RGB-only methods. As a result, refinement techniques are necessary to achieve the performance requirements of high-precision applications. Similar to traditional pose estimation techniques, early methods used either hand crafted feature descriptors, or template based matching techniques for refinement. DeepIM [18] then introduced a novel neural network architecture to iteratively refine the pose of an object in a target image by matching it to a rendered image of the object’s initially estimated pose. DeepIM is based on the FlowNetS [9] optical flow architecture, and directly regresses the translational and rotational updates necessary to minimize the difference in the observed and rendered images.

Recent state-of-the-art works improve upon DeepIM by addressing a variety of factors, but virtually all of them follow the same basic render-and-compare approach. For example, CosyPose [17] replaces the FlowNetS backbone with EfficientNet; [28], removes the optical flow head, and directly regresses rotation in a 6D rotation parameterization [36] as opposed to a quaternion; and [31] introduces a combined pose proposal and refinement network. Focusing
Figure 2. Overview of the DeepRM method. An initial pose estimate is used to render a target object. The observed and rendered images are passed through a convolutional neural network to predict a $\text{se}(3)$ transformation that updates the previous pose estimate. This process is repeated multiple times to incrementally refine the estimate. In addition to the updated pose estimate, hidden states from recurrent LSTM modules are propagated to each iteration.

on the refinement network, [31] extracts and warps feature maps based on the optical flow between observed and rendered images. The warped feature maps then pass through a spatial multi-attention layer to highlight important features, before directly regressing the pose update.

RNNPose [34] is a recent work on RGB pose refinement that uses an architecture inspired by RAFT [29] for optical flow, but extends it significantly for the task of pose estimation. RNNPose is the first work to leverage Gated Recurrent Units (GRUs) during the iterative process of pose refinement. However, pose is optimized by a Levenberg-Marquardt (LM) algorithm on an estimated correspondence field, and therefore RNNPose is not considered a purely direct approach.

Following RNNPose, Lipson et al. [19] also use a RAFT inspired architecture, but solve for pose using a Bidirectional Depth-Augmented PnP (BD-PnP) solver. This technique extends the standard PnP process by additionally minimizing the reprojection errors of the rendered image, as well as the inverse depth. Like RNNPose, this method predicts a 2D-3D correspondence field and then solves for pose, therefore we do not consider it a direct approach.

Vision transformer architectures [8], [20] have recently gained popularity for many computer vision tasks, including fine grained classification, semantic segmentation, object tracking, and human pose estimation. Trans6D [34] and CRT-6D [4] utilize vision transformers for the task of 6D object pose estimation. However, while they utilize transformers, both methods require hybridized architectures consisting of both convolutional and attention layers to achieve state-of-the-art results. CRT-6D [4], for example, uses a ResNet34 [10] backbone for feature extraction, followed by multiple layers of deformable self and cross-attention. Additionally, both methods require an iterative refinement process to achieve improved results.

3. METHOD

An overview of the proposed DeepRM method is illustrated in Fig. 2. Inspired by DeepIM [18], it follows an iterative render-and-compare approach to refine the pose of an object in a single RGB input image. Given an initial pose estimate of a target object, an image of the target object is rendered. The rendered image is then matched with the real image of the object to predict an $\text{se}(3)$ transform to the initial pose estimate that better aligns the rendered object to the observed image. The $\text{se}(3)$ transform consists of a translation and rotation vector, where the rotation is represented as a normalized unit quaternion. $\text{se}(3)$ denotes the Special Euclidean group, which refers to the set of proper rigid transformations within the Euclidean group. Such transformations within the Euclidean group preserve the Euclidean distance between transformed points. Because each update reduces the error between the rendered and observed images, this process can be repeated iteratively to incrementally refine the result. This method compensates for lack of external information such as depth by leveraging pre-existing geometric and visual properties of target objects, i.e. textured CAD models. By rendering objects in a way that is geometrically consistent with the observed scene, 3-D spatial information can be recovered from the RGB only image data.

3.1. Network Architecture

The DeepRM neural network architecture is illustrated in Fig. 3. The observed and rendered RGB images are concatenated channel-wise to form a $240 \times 320 \times 6$ dimensional tensor. The 6-channel tensor is passed as input to the backbone convolutional neural network to extract fea-
Figure 3. Architecture of the proposed DeepRM method. The observed and rendered RGB images are concatenated to form a 6-channel tensor. The 6-channel tensor is then passed as input to the backbone network to extract feature maps. The final $8 \times 10 \times 384$ feature map is flattened and passed through three shared, fully-connected, LSTM layers before the final translation and rotation heads. The multi-scale feature maps from the backbone network are also used in the optical flow head during training.

3.2. Recurrent Fully Connected Layers

While many other works [17, 18, 31] in pose refinement leverage an iterative process to incrementally improve upon an initial coarse estimate, most do not leverage any type of recurrent network features. However, recurrent architectures have been successfully used to improve the iterative processes of other visual processing tasks, such as optical flow prediction [29], saliency detection [6], and instance segmentation [24]. Adding gated recurrent mechanisms, such as LSTMs or GRUs, to the iterative processes should generally maintain or improve their current levels of performance. Considering the case where all gated connections are disabled, we simply have the original network configuration, where each iteration is independent of the previous. We can then enable the recurrent connections to enforce continuity across iterations, improving performance with each iteration. Based on our hypothesis, we apply this theory to the task of 6D pose refinement and present a novel recurrent network architecture suited for this task.

3.3. High Resolution Cropping

To improve upon the cropping strategy of DeepIM [18], we choose to follow an approach similar to CosyPose [17]. This process consists of cropping the region of interest around the object, based on the estimated pose, and then resizing this crop to $320 \times 240$ before passing it to the network. This cropping strategy has several benefits: a) it reduces background clutter b) it leverages the higher input image resolution. c) it reduces the memory and computational requirements of the network. The only difference between our approach and [17] is that we generate the rendered image at the full $640 \times 480$ resolution, and use the same crop as the target image, rather than adjusting the camera parameters and rendering directly to $320 \times 240$.

3.4. Transformation Parameterization

Following DeepIM [18], the network does not directly predict the translational update as a vector in meters, but rather a 2D translation in pixel space, along with a relative...
change in depth, corresponding to the projected centerpoint of the target object. Given the initial pose of the object, and the pixel space displacements, the 3D translation can be recovered via the thin lens equation. This parameterization enables the network to perform simplified reasoning in 2D, as opposed to modeling the complex relationship between 3D object geometry and the camera intrinsics.

### 3.5. Rotation Parameterization

To regress rotation, the network predicts the four quaternion components, which are then normalized to form a unit quaternion. The advantage of normalizing the output is that the network only needs to learn the ratios between components.

#### 3.6. Disentangled Point Matching Loss

To learn 3D pose, we use the point matching loss (\(L_{PML}\)) function as in [18], but disentangle the translational components as in [25]. \(L_{PML}\) incorporates both rotational and translational error in a single scalar metric, conveniently eliminating the need to balance the separate elements. Additionally, the disentangled formulation isolates the influence of the \(xy\) translation with the relative change in depth. For a ground truth pose \(p = [R|T]\), and an estimated pose \(\tilde{p} = [\tilde{R}|\tilde{T}]\), the point matching loss is defined as the average \(\ell_1\) norm of a subset of \(n\) model points:

\[
L_{PML}(\tilde{R}, \tilde{T}) = \frac{1}{n} \sum_{i=1}^{n} \| (Rx_i + T) - (\tilde{R}x_i + \tilde{T}) \|_1 .
\]

where \(x_i\) denotes the \(i\)-th model point.

Extending the above equation to disentangle the translational components, we first split the ground truth translation and the predicted translation into their respective components, i.e. \(T = [x, y, z]\) and \(\tilde{T} = [\tilde{x}, \tilde{y}, \tilde{z}]\). We then utilize a combination of the ground truth and predicted translations as input to the \(L_{PML}\) function to create our disentangled pose loss, \(L_{DPML}\):

\[
L_{DPML} = \left[ L_{PML}(\tilde{R}, [\tilde{x}, \tilde{y}, \tilde{z}]) + L_{PML}(\tilde{R}, [\tilde{x}, \tilde{y}, z]) + L_{PML}(\tilde{R}, [x, \tilde{y}, \tilde{z}]) \right] / 3.
\]

Our formulation is slightly different than [25] and [17] in that it does not disentangle the rotation component. This was found experimentally to be much more stable during training, and provides better results than the fully disentangled representation. For the auxiliary optical flow head, we use the same multi-scale endpoint error loss (\(L_{MS-EPE}\)) as [9]. The disentangled point matching pose loss is then combined with the mask loss to obtain the total loss (\(L_{total}\)) as follows:

\[
L_{total} = L_{DPML} + \alpha \cdot L_{MS-EPE},
\]

where the balancing factor \(\alpha\) has been set to 0.1 following [18].

### 4. EXPERIMENTS

#### 4.1. Datasets

The YCB-Video dataset [33] is a a large scale dataset, with a total of 133,827 images over 92 unique scenes. Images contain labeled 6D poses of 21 target objects. The majority of images contain 4-5 objects in the scene, resulting in high levels of occlusion, as well as a variety of challenging lighting conditions. The 21 objects are a diverse selection of common household items, which include various levels of symmetry (i.e. non-symmetric, discretely symmetric, and continuously symmetric objects). For consistent comparison, we use the same exact real data, synthetic data, and data split as DeepIM [18].

The Occlusion LINEMOD dataset [2] is an extension upon the original LINEMOD dataset [12]. LINEMOD consists of 13 common household objects, split into 13 cluttered scenes. Roughly 1000 images are provided for each object. Many target objects are present in each image, however only a single object is labeled per image. The target object in each image is also generally very visible. To create a more challenging dataset, Occlusion LINEMOD was introduced. Occlusion LINEMOD provides ground truth labels for all objects in one of the 13 scenes. This results in high levels of partial occlusion, significantly increasing the difficulty of the dataset. Following the convention of other works such as [7, 32], we train on LINEMOD, and evaluate on Occlusion LINEMOD. Although, due to the limited amount of real data provided in LINEMOD, we additionally augment the training data with physically-based rendering (PBR) images that are publicly available from the 2020 BOP Challenge [14].

#### 4.2. Evaluation Metrics

To evaluate the performance against other state-of-the-art methods, we follow [7, 17, 18, 22, 31, 32] and use the ADD metric [12]. More specifically, we use two specific variations upon it, depending on the dataset, ADD(-S) 10% for Occlusion LINEMOD and area under the curve (AUC) ADD(-S) for YCB-Video. For the sake of brevity, we refer readers to prior works such as [12] and [33] for a more detailed description of these metrics.

#### 4.3. Implementation Details

DeepRM is implemented in PyTorch, and uses the same OpenGL based renderer as [18]. Unlike other works [17,18] that use a consistent light source, we manually tuned the
Table 1. Comparison to state-of-the-art on the YCB-V dataset. Ref. indicates that the method used to provide initial coarse estimates to DeepRM. * identifies the method used to provide initial coarse to DeepRM. In the P.E. column, M indicates a separate unique model is trained per object and 1 means a single model was trained for all objects.

<table>
<thead>
<tr>
<th>Method</th>
<th>P.E.</th>
<th>Ref.</th>
<th>AUC of ADD(-S) 10% ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoseCNN [33]</td>
<td>1</td>
<td></td>
<td>24.9</td>
</tr>
<tr>
<td>PVNet [22]</td>
<td>1</td>
<td></td>
<td>40.8</td>
</tr>
<tr>
<td>RePose [15]</td>
<td>M</td>
<td>✓</td>
<td>51.6</td>
</tr>
<tr>
<td>PPC [3]</td>
<td>1</td>
<td>✓</td>
<td>55.3</td>
</tr>
<tr>
<td>DeepIM [18]</td>
<td>1</td>
<td>✓</td>
<td>55.5</td>
</tr>
<tr>
<td>GDR-Net [32]</td>
<td>M</td>
<td>✓</td>
<td>56.1</td>
</tr>
<tr>
<td>Trans6D [34]</td>
<td>M</td>
<td>✓</td>
<td>57.9</td>
</tr>
<tr>
<td>Trabelsi [31]</td>
<td>1</td>
<td>✓</td>
<td>58.4</td>
</tr>
<tr>
<td>RNNPose [34]</td>
<td>M</td>
<td>✓</td>
<td>60.7</td>
</tr>
<tr>
<td>GDR-Net [32]</td>
<td>M</td>
<td>✓</td>
<td>62.2</td>
</tr>
<tr>
<td>SO-Pose [7]</td>
<td>1</td>
<td>✓</td>
<td>62.3</td>
</tr>
<tr>
<td>CRT-6D [4]</td>
<td>I</td>
<td>✓</td>
<td>66.3</td>
</tr>
<tr>
<td>ZebraPose [27]</td>
<td>M</td>
<td></td>
<td>76.9</td>
</tr>
<tr>
<td>DeepRM (Ours)</td>
<td>1</td>
<td>✓</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Table 2. Comparison to state-of-the-art on the LM-O dataset. Ref. indicates that the network includes refinement. * identifies the method used to provide initial coarse estimates to DeepRM. In the P.E. column, M indicates a separate unique model is trained per object and 1 means a single model was trained for all objects.

<table>
<thead>
<tr>
<th>Method</th>
<th>P.E.</th>
<th>Ref.</th>
<th>ADD(-S) 10% ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoseCNN [33]</td>
<td>1</td>
<td></td>
<td>61.31</td>
</tr>
<tr>
<td>PVNet [22]</td>
<td>1</td>
<td></td>
<td>73.4</td>
</tr>
<tr>
<td>RePose [15]</td>
<td>M</td>
<td>✓</td>
<td>77.2</td>
</tr>
<tr>
<td>GDR-Net [32]</td>
<td>1</td>
<td></td>
<td>80.2</td>
</tr>
<tr>
<td>DeepIM [18]</td>
<td>1</td>
<td>✓</td>
<td>81.9</td>
</tr>
<tr>
<td>RNNPose [34]</td>
<td>M</td>
<td>✓</td>
<td>83.1</td>
</tr>
<tr>
<td>Trabelsi [31]</td>
<td>I</td>
<td>✓</td>
<td>83.1</td>
</tr>
<tr>
<td>SO-Pose [7]</td>
<td>1</td>
<td></td>
<td>83.9</td>
</tr>
<tr>
<td>GDR-Net [32]</td>
<td>M</td>
<td></td>
<td>84.4</td>
</tr>
<tr>
<td>CosyPose [17]</td>
<td>1</td>
<td>✓</td>
<td>84.5</td>
</tr>
<tr>
<td>ZebraPose [27]</td>
<td>M</td>
<td></td>
<td>85.3</td>
</tr>
<tr>
<td>Trans6D [34]</td>
<td>M</td>
<td>✓</td>
<td>85.9</td>
</tr>
<tr>
<td>CRT-6D [4]</td>
<td>I</td>
<td>✓</td>
<td>87.5</td>
</tr>
<tr>
<td>DeepRM (Ours)</td>
<td>1</td>
<td>✓</td>
<td>87.0</td>
</tr>
</tbody>
</table>

Table 3 displays network performance in terms of accuracy and frames per second (FPS) as a function of various backbone architectures, fully-connected layer types, fully-connected layer dimensions, and number of trainable parameters for the YCB-Video dataset. Due to resource and time constraints, results are limited to 8 refinement iterations. All tests were performed on a desktop workstation with a single NVIDIA RTX 3060 GPU and an Intel i7-11700K CPU.

Highest accuracy is observed for the EfficientNet-B3
bachbone using 8 refinement iterations. This configuration achieves 86.8% at 9.5 FPS on the AUC ADD(-S) metric for YCB-Video. However, the number of refinement iterations can be decreased to 4 to achieve 18 FPS while still maintaining superior accuracy to all state-of-the-art methods.

Table 3 also demonstrates that the fully connected layers in our architecture can be scaled along with the EfficientNet backbone, using the same scaling parameter, $ϕ$. Using this technique, the model can be adapted to meet real-world execution time or resource constraints. This flexibility along with the accuracy and efficiency of our method provide a solution that is well-suited to practical robotics applications.

Finally, to support our claim that recurrent network features improve the performance of this task, Table 3 displays the impact of recurrent fully-connected layers compared to standard fully-connected ones. We find that LSTMs provide a significant increase of 1.8%, whereas GRUs provide a more moderate improvement of 0.5% over the standard fully-connected baseline.

### 4.6. Ablation Study on Refinement Iterations for YCB-Video

The process of iterative refinement is heavily dependent on the number of iterations performed. As such, we investigate the impact of training and testing on a variety of refinement iterations. All tests were performed with the EfficientNet-B3 backbone on the YCB-Video dataset. AUC ADD(-S) results are reported in Table 4. Best performance is achieved when training with 6 iterations, and testing with 12 iterations, although we find 8 testing iterations to provide the best balance of accuracy and execution time.

### 4.7. Ablation Study on Optical Flow

In addition to the recurrent structure, the auxiliary optical flow head is one of the main features that distinguishes our work from others such as CosyPose [17]. We find that the auxiliary optical flow head provides an accuracy improvement of 1.8% on the EfficientNet-B3 backbone configuration of our network, clearly demonstrating its benefit. Furthermore, this improvement only costs a 5% increase in parameters during training. At inference, this portion of the network is removed. Table 5 displays these results using 6 training iterations, and 8 testing iterations on the YCB-Video dataset.

### 5. CONCLUSIONS

In this work, we introduce DeepRM, a novel method for precise 6D pose estimation of rigid objects from RGB only data. DeepRM improves upon existing render-and-compare

### Table 3. Ablation Study on YCB-Video.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>FC Type</th>
<th>FC Layer Dims</th>
<th># Params</th>
<th>Metric</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepIM [18]</td>
<td>FlowNetS</td>
<td>MLP</td>
<td>256→256</td>
<td>60M</td>
<td>FPS ADD(-S)</td>
<td>12.0</td>
<td>81.9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>DeepRM (ours)</td>
<td>EfficientNet-B0 ($ϕ=0$)</td>
<td>LSTM</td>
<td>256→256→128</td>
<td>33M</td>
<td>FPS ADD(-S)</td>
<td>47.8</td>
<td>83.2</td>
<td>24.4</td>
<td>17.5</td>
</tr>
<tr>
<td>DeepRM (ours)</td>
<td>EfficientNet-B2 ($ϕ=2$)</td>
<td>LSTM</td>
<td>384→256→256</td>
<td>55M</td>
<td>FPS ADD(-S)</td>
<td>39.2</td>
<td>83.7</td>
<td>21.0</td>
<td>14.0</td>
</tr>
<tr>
<td>DeepRM (ours)</td>
<td>EfficientNet-B3 ($ϕ=3$)</td>
<td>LSTM</td>
<td>512→256→128</td>
<td>22M</td>
<td>FPS ADD(-S)</td>
<td>37.8</td>
<td>83.0</td>
<td>19.5</td>
<td>13.1</td>
</tr>
<tr>
<td>DeepRM (ours)</td>
<td>EfficientNet-B3 ($ϕ=3$)</td>
<td>MLP</td>
<td>512→256→128</td>
<td>22M</td>
<td>FPS ADD(-S)</td>
<td>37.8</td>
<td>83.0</td>
<td>19.5</td>
<td>13.1</td>
</tr>
<tr>
<td>DeepRM (ours)</td>
<td>EfficientNet-B3 ($ϕ=3$)</td>
<td>GRU</td>
<td>512→256→128</td>
<td>63M</td>
<td>FPS ADD(-S)</td>
<td>35.7</td>
<td>83.5</td>
<td>18.1</td>
<td>12.7</td>
</tr>
<tr>
<td>DeepRM (ours)</td>
<td>EfficientNet-B3 ($ϕ=3$)</td>
<td>LSTM</td>
<td>512→256→128</td>
<td>79M</td>
<td>FPS ADD(-S)</td>
<td>34.0</td>
<td>84.2</td>
<td>18.0</td>
<td>12.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADD(-S)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>train iters</th>
<th>init</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>test iters</td>
<td></td>
<td>60.0</td>
<td>82.3</td>
<td>82.5</td>
<td>82.2</td>
<td>84.2</td>
<td>85.5</td>
<td>85.5</td>
<td>85.4</td>
<td>84.2</td>
<td>86.2</td>
<td>86.6</td>
<td>86.8</td>
<td>87.0</td>
<td>87.0</td>
</tr>
</tbody>
</table>

Table 4. Ablation Study on Refinement Iterations for YCB-Video. ADD(-S) represents AUC ADD(-S). Best results are obtained when training with 6 iterations and testing with 12.

Table 5. Ablation Study on Optical flow for YCB-Video. Opti-

<table>
<thead>
<tr>
<th>Method</th>
<th># Params</th>
<th>AUC of ADD(-S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Flow</td>
<td>75 M</td>
<td>85.0</td>
</tr>
<tr>
<td>Flow</td>
<td>79 M</td>
<td>86.8</td>
</tr>
</tbody>
</table>

Table 5. Ablation Study on optical flow for YCB-Video. Optical flow reinforcement provides a 1.8% improvement, while only increasing the model size by 5%.
approaches by leveraging several unique elements, such as an optical flow enforced learning process, an efficient and scalable backbone, and an LSTM enhanced iterative refinement mechanism. DeepRM outperforms the majority of existing state-of-the-art methods on the challenging YCB-Video and Occlusion LINEMOD datasets.

References


[34] Yan Xu, Kwan-Yee Lin, Guofeng Zhang, Xiaogang Wang, and Hongsheng Li. RNNPose: Recurrent 6-DoF Object Pose Refinement with Robust Correspondence Field Estimation and Pose Optimization. *ArXiv*, 1, 2022.
