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.

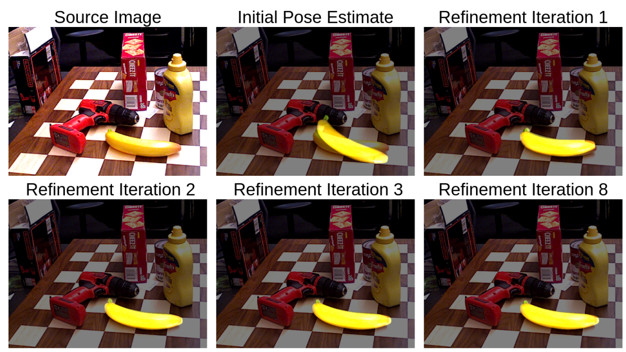

Figure 1. Example of DeepRM object pose refinement.

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 Localization [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 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

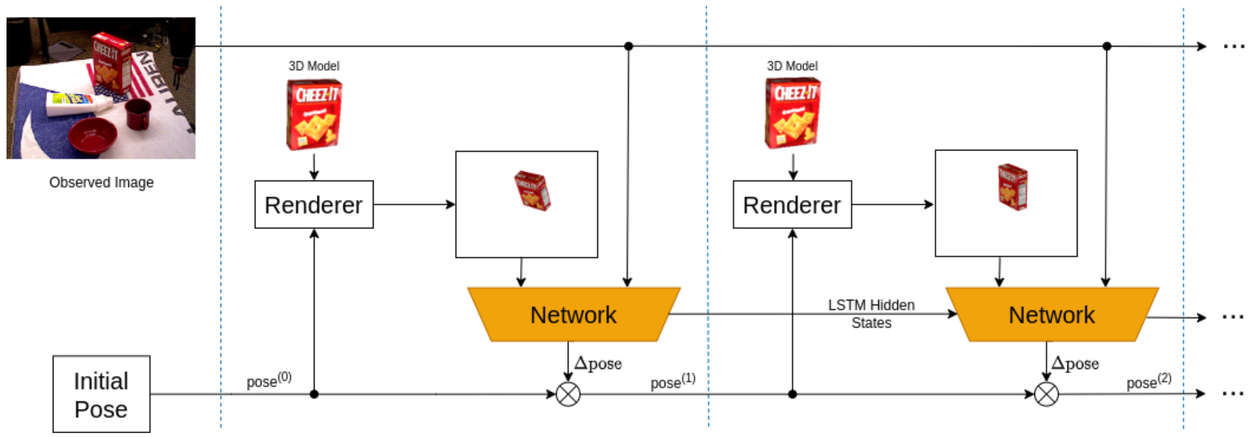

Figure 2. Overview of the DeepRM method. An initial pose estimate is used to render a target object. The observed and rendered images are passed through a convolutional neural network to predict a se(3) transformation that updates the previous pose estimate. This process is repeated multiple times to 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 se(3) transform to the initial pose estimate that better aligns the rendered object to the observed image. The se(3) transform consists of a translation and rotation vector, where the rotation is represented as a normalized unit quaternion. se(3) denotes the Special Euclidean group, which refers to the set of proper rigid transformations within the Euclidean group. Such transforms 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-

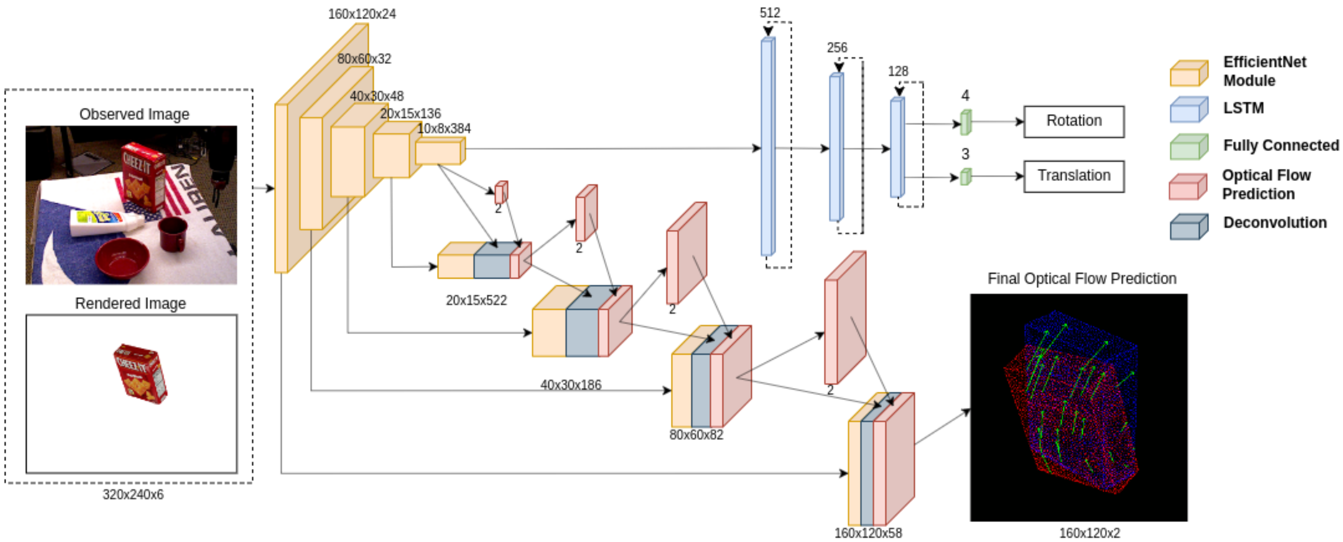

Figure 3. Architecture of the proposed DeepRM method. The observed and rendered RGB images are concatenated to form a 6-channel tensor. The 6-channel tensor is then passed as input to the backbone network to extract feature maps. The final $8\times10\times384$ feature map is flattened and passed through 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.

ture maps, where the final $8\times10\times384$ feature map from the backbone 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 an auxiliary optical flow head during training to stabilize the training process, and enforce the learning of features which are relevant to the task of pose estimation. The structure of the optical flow head is the same as FlowNetS [9], however both the spatial and channel dimensions are modified to match the corresponding layers in the backbone network. We use the B3 version of EfficientNet [28] as the backbone feature extractor to achieve the best balance between performance and model size, but also demonstrate state-of-the-art results with smaller versions of EfficientNet such as B0 and B2. Because we match the dimensions of the optical flow head to those of the backbone, our architecture can be scaled as a whole, using the same hyperparameter used by the EfficientNet backbone, called ϕ. The use of ϕ as a model scaling hyperparameter is further discussed in [28].

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 320×240 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 640×480 resolution, and use the same crop as the target image, rather than adjusting the camera parameters and rendering directly to 320×240.

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 ($\mathcal{L}_{PML}$) function as in [18], but disentangle the translational components as in [25]. $\mathcal{L}_{PML}$ 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 $\boldsymbol{p} = [\boldsymbol{R}|\boldsymbol{T}]$, and an estimated pose $\widetilde{\boldsymbol{p}} = [\widetilde{\boldsymbol{R}}|\widetilde{\boldsymbol{T}}]$, the point matching loss is defined as the average ℓ_1 norm of a subset of n model points:

$$\mathcal{L}_{PML}(\widetilde{R}, \widetilde{T}) = \frac{1}{n}\sum_{i=1}^{n}\left|\left|(Rx_i + T) - (\widetilde{R}x_i + \widetilde{T})\right|\right|_1. \quad (1)$$

where x_i denotes the $i\text{-}th$ 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. $\boldsymbol{T} = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$ and $\widetilde{\boldsymbol{T}} = [\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{y}}, \widetilde{\boldsymbol{z}}]$. We then utilize a combination of the ground truth and predicted translations as input to the $\mathcal{L}_{PML}$ function to create our disentangled pose loss, $\mathcal{L}_{DPML}$:

$$\begin{aligned}\mathcal{L}_{DPML} = \Big[&\mathcal{L}_{PML}(\widetilde{R},\, [\widetilde{x}, \widetilde{y}, \widetilde{z}]) + \\ &\mathcal{L}_{PML}(\widetilde{R},\, [\widetilde{x}, \widetilde{y}, z]) + \\ &\mathcal{L}_{PML}(\widetilde{R},\, [x, y, \widetilde{z}])\Big]/\,3.\end{aligned} \quad (2)$$

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 ($\mathcal{L}_{MS-EPE}$) as [9]. The disentangled point matching pose loss is then combined with the mask loss to obtain the total loss ($\mathcal{L}_{total}$) as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{DPML} + \alpha \cdot \mathcal{L}_{MS-EPE}, \quad (3)$$

where the balancing factor α 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

Method	P.E.	Ref.	AUC of ADD(-S) ↑
PoseCNN [33] ⋆	1		61.31
PVNet [22]	1		73.4
RePose [15]	M	✓	77.2
GDR-Net [32]	1		80.2
DeepIM [18]	1	✓	81.9
RNNPose [34]	M	✓	83.1
Trabelsi [31]	1	✓	83.1
SO-Pose [7]	1		83.9
GDR-Net [32]	M		84.4
CosyPose [17]	1	✓	84.5
ZebraPose [27]	M		85.3
Trans6D [34]	M	✓	85.9
CRT-6D [4]	1	✓	**87.5**
DeepRM (Ours)	1	✓	87.0

Table 1. **Comparison to state-of-the-art on the YCB-V 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.

Method	P.E.	Ref.	ADD(-S) 10% ↑
PoseCNN [33]	1		24.9
PVNet [22] ⋆	1		40.8
RePose [15]	M	✓	51.6
PPC [3]	1	✓	55.3
DeepIM [18]	1	✓	55.5
GDR-Net [32]	1		56.1
Trans6D [34]	M	✓	57.9
Trabelsi [31]	1	✓	58.4
RNNPose [34]	M	✓	60.7
GDR-Net [32]	M		62.2
SO-Pose [7]	1		62.3
CRT-6D [4]	1	✓	66.3
ZebraPose [27]	M		**76.9**
DeepRM (Ours)	1	✓	65.0

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.

renderer brightness so that it properly exposes each object that is matched in the target scene. This step can be automated by using an average metering algorithm, similar to what is used in digital photography for auto-exposure. Since the rendered object is always drawn on a black background, the render brightness for each object can be predetermined offline, and used throughout all training and testing. This simple modification improved the baseline results by 0.4%. For both YCB-Video and Occlusion LINEMOD datasets, we use the ADAM optimizer [16], with a base learning rate of 1e-4. Although due to the differences in each dataset, we use different batch sizes, training durations, and learning rate schedules for each dataset. For YCB-Video, 16 images are used per batch, with 4 objects per image, resulting in an effective batch size of 64. Similar to DeepIM [18], the model is trained for 20 epochs, with fixed learning rate decays of 0.1 at epochs 10 and 15. Although best results are obtained earlier at epoch 19 on YCB-Video. For Occlusion LINEMOD, 48 images are used per batch, with 1 object per image, resulting in an effective batch size of 48. Number of epochs are scaled up to 190, to account for the difference in the size of the dataset and batch size compared to YCB-Video. Additionally, for Occlusion LINEMOD only, a warmup period of 10% base learning rate is used in the first 4 epochs. Both datasets are trained with 6 refinement iterations during training. Then during testing, 12 iterations of refinement are used for YCB-Video, and 6 iterations are used for Occlusion LINEMOD.

4.4. Comparison to State-of-the-Art

Results on YCB-Video dataset. Table 1 presents the results of DeepRM compared to the current state-of-the-art on the YCB-Video dataset for the AUC ADD(-S) metric. Initial predictions are obtained from PoseCNN [33], where DeepRM outperforms all existing state-of-the-art methods except CRT-6D [4]. We note that CRT-6D was trained using synthetic, physically-based rendering (PBR), images for training, whereas DeepRM did not use this additional information. We speculate that re-training DeepRM on this improved dataset would close or surpass the 0.5% performance gap.

Results on LM-O dataset. Table 2 presents the results of DeepRM compared to the current state-of-the-art on the Occlusion LINEMOD dataset for the ADD(-S) 10% metric. Initial predictions are obtained from PVNet [22], where DeepRM outperforms all existing methods except for ZebraPose [27] and CRT-6D [4]. Although ZebraPose provides significantly superior performance, it requires a different model for each object and runs at a significantly reduced frame rate compared to DeepRM.

4.5. Ablation Study on YCB-Video

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

Method	Backbone	FC Type	FC Layer Dims	# Params	Metric	2	4	6	8
DeepIM [18]	FlowNetS	MLP	256→256	60M	FPS	12.0	N/A	N/A	N/A
					ADD(-S)	N/A	81.9	N/A	N/A
DeepRM (ours)	EfficientNet-B0 (ϕ=0)	LSTM	256→256→128	33M	FPS	47.8	24.4	17.5	13.0
					ADD(-S)	83.2	84.5	84.7	84.6
DeepRM (ours)	EfficientNet-B2 (ϕ=2)	LSTM	384→256→256	55M	FPS	39.2	21.0	14.0	10.3
					ADD(-S)	83.7	85.1	85.4	85.5
DeepRM (ours)	EfficientNet-B3 (ϕ=3)	MLP	512→256→128	22M	FPS	37.8	19.5	13.1	10.1
					ADD(-S)	83.0	84.6	84.8	85.0
DeepRM (ours)	EfficientNet-B3 (ϕ=3)	GRU	512→256→128	63M	FPS	35.7	18.1	12.7	9.3
					ADD(-S)	83.5	85.3	85.5	85.5
DeepRM (ours)	EfficientNet-B3 (ϕ=3)	LSTM	512→256→128	79M	FPS	34.0	18.0	12.0	9.5
					ADD(-S)	84.2	86.2	86.6	**86.8**

Table 3. **Ablation Study on YCB-Video.** FPS represents frames per second. ADD(-S) represents AUC ADD(-S). MLP represents multi-layer perceptron, i.e. standard fully connected layer. N/A represents 'Not Available'.

train iters	init	2			4				6						
test iters		2	4	6	2	4	6	8	2	4	6	8	10	12	14
ADD(-S)	60.0	82.3	82.5	82.2	84.2	85.5	85.5	85.4	84.2	86.2	86.6	86.8	86.9	**87.0**	87.0

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.

backbone 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.

Method	# Params	AUC of ADD(-S)
No Flow	75 M	85.0
Flow	79 M	**86.8**

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%.

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 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 datsets.

References

[1] Mark Billinghurst, Adrian Clark, and Gun Lee. A survey of augmented reality. *Foundations and Trends in Human-Computer Interaction*, 8(2-3):73–272, 2014. 1

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8690 LNCS, pages 536–551, 2014. 2, 5

[3] Lucas Brynte and Fredrik Kahl. Pose Proposal Critic: Robust Pose Refinement by Learning Reprojection Errors. In *Proceedings of the British Machine Vision Conference*, pages 1–16, 2020. 6

[4] Pedro Castro and Tae-Kyun Kim. Crt-6d: Fast 6d object pose estimation with cascaded refinement transformers. 10 2023. 3, 6

[5] Alvaro Collet and Manuel Martinez. MOPED: Object Recognition and Pose Estimation for Manipulation. *The International Journal of Robotics Research*, 30:1284–1306, 2011. 2

[6] Zijun Deng, Xiaowei Hu, Lei Zhu, Xuemiao Xu, Jing Qin, Guoqiang Han, and Pheng-ann Heng. R³Net: Recurrent Residual Refinement Network for Saliency Detection. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 684–690, 2018. 4

[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 International Conference on Computer Vision*, volume 1, 2021. 2, 5, 6

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani an Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 3

[9] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:2758–2766, 2015. 2, 4, 5

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 3

[11] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012. 2

[12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7724 LNCS(PART 1):548–562, 2013. 5

[13] Tomáš Hodaň, Dániel Baráth, and Jiř Matas. EPOS: Estimating 6D pose of objects with symmetries. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 11700–11709, 2020. 2

[14] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP Challenge 2020 on 6D Object Localization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12536 LNCS:577–594, 2020. 1, 5

[15] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M. Kitani. RePOSE: Real-Time Iterative Rendering and Refinement for 6D Object Pose Estimation. In *IEEE/CVF International Conference on Computer Vision*, 2021. 6

[16] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015. 6

[17] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose : Consistent Multi-view Multi-object 6D Pose Estimation. In *European Conference on Computer Vision*, volume 2, pages 574–591, 2020. 1, 2, 4, 5, 6, 7

[18] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. *International Journal of Computer Vision*, 128(3):657–678, oct 2020. 1, 2, 3, 4, 5, 6, 7

[19] Lahav Lipson, Zachary Teed, Ankit Goyal, and Jia Deng. Coupled Iterative Refinement for 6D Multi-Object Pose Estimation. 2022. 3

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021. 3

[21] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3D object pose estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11219 LNCS:125–141, 2018. 2

[22] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. PVNET: Pixel-wise voting network for 6dof pose estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4556–4565, dec 2019. 2, 5, 6

[23] 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 *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 3848–3856, 2017. 2

[24] Mengye Ren and Richard S. Zemel. End-to-end instance segmentation with recurrent attention. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:293–301, 2017. 4

[25] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel Lopez-Antequera, and Peter Kontschieder. Disentangling monocular 3D object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:1991–1999, 2019. 5

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015. 2

[27] Yongzhi Su, Mahdi Saleh, Torben Fetzer, Jason Rambach, Nassir Navab, Benjamin Busam, Didier Stricker, and Federico Tombari. ZebraPose: Coarse to Fine Surface Encoding for 6DoF Object Pose Estimation. 2022. 2, 6

[28] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:10691–10700, 2019. 2, 4

[29] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12347 LNCS:402–419, 2020. 3, 4

[30] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. 2

[31] Ameni Trabelsi, Mohamed Chaabane, Nathaniel Blanchard, and Ross Beveridge. A Pose Proposal and Refinement Network for Better 6D Object Pose Estimation. In *IEEE Winter Conference on Applications of Computer Vision*, pages 2381–2390, 2021. 1, 2, 3, 4, 5, 6

[32] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 16611–16621, 2021. 2, 5, 6

[33] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *ArXiv*, may 2017. 2, 5, 6

[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. 3, 6

[35] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. DPOD: 6D pose object detector and refiner. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 1941–1950, feb 2019. 2

[36] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 5738–5746, 2019. 2