A Pose Proposal and Refinement Network for Better 6D Object Pose Estimation

Ameni Trabelsi Mohamed Chaabane Nathaniel Blanchard Ross Beveridge

Department of Computer Science
Colorado State University
{amenit,chaabane, nathaniel.blanchard, ross.beveridge}@colostate.edu

Abstract

In this paper, we present a novel, end-to-end 6D object pose estimation method that operates on RGB inputs. Our approach is composed of 2 main components: the first component classifies the objects in the input image and proposes an initial 6D pose estimate through a multi-task, CNN-based encoder/multi-decoder module. The second component, a refinement module, includes a renderer and a multi-attentional pose refinement network, which iteratively refines the estimated poses by utilizing both appearance features and flow vectors. Our refiner takes advantage of the hybrid representation of the initial pose estimates to predict the relative errors with respect to the target poses. It is further augmented by a spatial multi-attention block that emphasizes objects' discriminative feature parts. Experiments on three benchmarks for 6D pose estimation show that our proposed pipeline outperforms state-of-the-art RGB-based methods with competitive runtime performance.

1. Introduction

Accurate 6D object pose estimation is crucial for many real-world applications, such as autonomous driving, robotic manipulation, and augmented reality. For instance, a 6D pose estimator for robot grasping needs to balance accuracy, robustness, and speed to be realistically deployable in real-world scenarios.

Some approaches [31, 34] have relied upon depth information in order to boost reliability and accuracy. However, depth sensors suffer a variety of failure cases, have high energy and monetary costs, and are less ubiquitous than their non-depth counterparts. Ultimately, pose estimation from RGB alone is a more challenging problem, but also a far more attractive option.

This paper presents a novel, end-to-end 6D pose estimation approach from RGB inputs. In Figure 1, we show an overview of our approach. First, the Pose Proposal Network

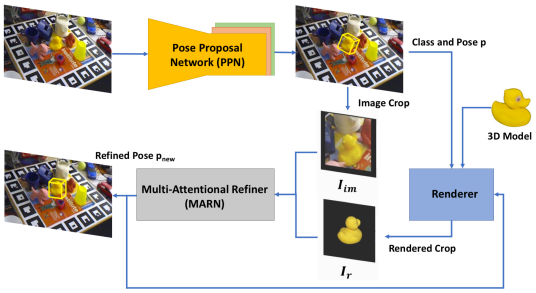

Figure 1. An overview of our proposed approach, consisting of a pose proposal module and a pose refinement module. The pose proposal module (PPN), outputs an object classification and an initial pose estimation from RGB inputs. The pose refinement module consists of a differentiable renderer and an iterative refiner called MARN. The renderer initializes the rendered crop of the detected object using its initial pose estimate and its 3D model. The refinement step utilizes a hybrid representation of the initial render and the input image, combining visual and flow features, and integrates a multi-attentional block to highlight important features, to learn an accurate transformation between the predicted pose and the actual, observed pose.

(PPN) extends the region proposal framework to classify and regress initial estimates of the rotations and translations of objects present in the RGB input. Notably, our proposed PPN method requires no additional steps, unlike methods that use PnP [9] or matching with pre-engineered codebook. Second, the pose refinement module consists of a differentiable renderer and a Multi-Attentional Refinement Network (MARN). MARN can be depicted as two main components: first, visual features from both the input crop and the rendered crop are fused using the flow vectors to learn better object representations. Second, a spatial multi-attention block highlights discriminative feature parts, insulating the network from adverse noise and occlusion effects. MARN is designed to allow iterative refinement; MARN outputs an estimated pose that directly maps to MARN's in-

put. In our experiments, we typically found that the greatest performance gains occurred within a couple of refinement iterations.

Finally, our entire pipeline is trained end-to-end and achieves state-of-the-art results across a range of experiments and datasets.

In summary, our work makes the following contributions:

1. An end-to-end 6D pose estimation approach that outperforms state-of-the-art RGB-based methods on three commonly used benchmarks.

2. A pose proposal network (PPN) that is fully-CNN-based, yielding fast and accurate pose estimations in a single pass from RGB images, without additional steps.

3. A pose refinement network (MARN) that uses a hybrid intermediate representation of the input image and the initial pose estimation by combining visual and flow features to learn an accurate transformation between the predicted object pose and the actual observed pose.

4. The integration of a spatial multi-attentional block that highlights important feature parts, making the refinement process more robust to noise and occlusion.

2. Related Work

6D pose estimation has a long and storied history [32], but, due to space constraints, we will limit this section to RGB based methods. Traditional RGB methods for pose estimation typically match detected local keypoints or handcrafted features with known object models [1, 6, 18, 30]. These methods maintain scale and rotation invariance, and hence are often faster and more robust to occlusion. However, they become unreliable with low-texture objects or low-resolution inputs. Deep learning methods tend to be more robust to these issues. More recent variants of these methods mostly rely on deep learning to either learn feature representations or create 2D-3D correspondences [2, 14].

Most existing RGB-based methods [13, 20, 21, 23] take advantage of deep learning techniques used for object detection [5, 10, 16, 25] or image segmentation [17] and leverage them for 6D pose estimation. For example, one technique involves utilizing CNNs to extract object keypoints, and solving the 6D poses using PnP [23, 29]. Sundermeyer *et al*. [27] utilizes an encoder-decoder that learns feature vectors and matches them to a pre-generated codebook of feature vectors to determine the 6D pose of an object. Our work is different from these methods in that we integrate a pose estimator based on a region proposal framework, that estimates object poses in a single forward pass through the network with no additional codebook matching steps. Kehl *et al*. [13] and Tekin *et al*. [28] use region proposal framework to detect objects within the input image and then use additional steps (such as PnP [9]) to solve their poses. Though our approach also integrates a pose estimator inspired from region proposal frameworks, our work leverages the framework in an encoder/multi-decoder network for 6D pose estimation and extends it into a novel end-to-end pose estimation and refinement network.

6D pose refinement has been utilized to improve the performance of several pose estimation methods [31, 33]. Recent refinement methods have been deep-learning based [31, 19], relying on CNNs to predict a relative transformation between the initial pose prediction and the target pose. Li *et al*. [15] relies on Flownet's [12] deep feature representation, extracted from the input image and the rendering of the estimated object pose to learn the pose residuals. Though our refiner was inspired from [15], our approach is fundamentally different as it relies on the synergy between the optical flow vectors and the appearance features to capture the pose transformation from the prediction to the target pose. Further, we employ a multi-attentional block that efficiently highlights discriminative feature parts, improving the robustness of our refiner to noise and occlusion.

3. Methods

In this paper, we estimate the 6D poses of a set of known objects present in an RGB image. We propose a novel two step approach: First, a pose proposal module (PPN) (§ 3.1) that regresses initial 6D pose proposals from different regions of objects in an RGB image. Second, a pose refinement module, which includes i) a differentiable renderer, that outputs a render of the detected object using its initial pose estimate and 3D model, and ii) a multi-attentional refinement network (MARN) (§ 3.2), to further refine the initial pose estimates.

In the following, the 6D pose is represented as a homogeneous transformation matrix, $p = [R|t] \in \mathcal{SE}(3)$, composed of a rotation matrix $R \in \mathcal{SO}(3)$ and a translation $t \in \mathbb{R}^3$. R can also be represented by a quaternion $q \in \mathbb{R}^4$.

3.1. Pose Proposal Network

We reframe the object pose estimation as a combined object classification and pose estimation problem, regressing from image pixels to region proposals of object centers and poses. Figure 2 illustrates our 6D object pose proposal network architecture. Our architecture has two stages: first, a backbone encoder, modeled on the YOLOv2 framework [26], extracts high-dimensional region feature representations from the input image. Second, the obtained feature representations are embedded into low-dimensional, task-specific features extracted from three decoders which output three sets of region proposals for translations, rotations, and object centers and classes.We note that, similar

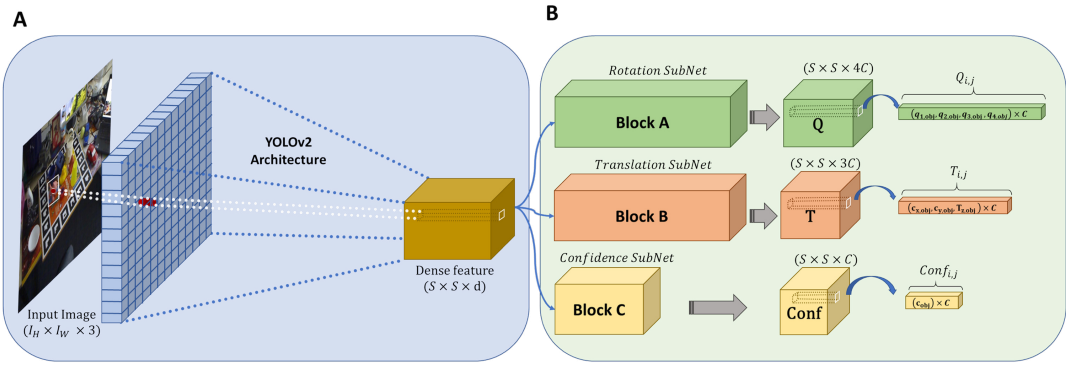

Figure 2. Our Pose Proposal Network (PPN) Architecture. The encoder/multi-decoder takes an RGB image, **A**. encodes it into high dimensional feature embedding, and **B**. decodes it into 3 task-specific outputs, which correspond to the rotation, translation, and confidence in the presence and class of the detected object. Detailed architecture of the three blocks is provided in supplemental material.

to [28], we rely on the YOLOv2 framework (§ 2) to extract feature representations from the input image. However, the application of the YOLOv2 network in our work is fundamentally different in the sense that it serves only the purpose of extracting appearance features from the input image which will be used as input to the second stage consisting of three decoders to ultimately estimate the objects poses. Specifically, the backbone encoder (Figure 2**A**) produces a dense feature representation $\mathcal{F}$ by dividing the input image into a $S \times S$ grid, each cell of which corresponds to an image block, that produces a set of high dimensional feature embeddings $\{\mathcal{F}_{i,j}\}$, with $\mathcal{F}_{i,j} \in \mathbb{R}^d$ for each grid cell $(i,j) \in \mathcal{G}^2$ s.t. $\mathcal{G} = \{1,\ldots,S\}$ and d is the embedding size. $\mathcal{F}$ is decoded by 3 parallel convolutional blocks (as shown in Figure 2**B**) that produce a fixed-size collection of region proposals $\{(Conf_{i,j}^{o_k}, T_{i,j}^{o_k}, Q_{i,j}^{o_k})\}$ for each object in the set of target objects $o_k \in \{o_1,\ldots,o_C\}$, where C is the number of target objects. The detailed architectures of the three blocks are depicted in supplemental material.

Block A: This block is a rotation proposal network that regresses a 4-dimensional quaternion vector $Q_{i,j}^{o_k}$ for each image region and object class.

Block B: This block is a translation proposal network that regresses a 3-dimensional translation vector $T_{i,j}^{o_k}$ for each image region and object class. Rather than predicting the full translation vector $T = [t_x, t_y, t_z]^T$, which can be cumbersome for training as discussed in [33], we regress the object center coordinates in the image space $c = (c_x, c_y)^T$ and the depth component t_z. The two remaining components of the translation vector are then easily computed with the camera intrinsics and the predicted information:

$$t_x = \frac{(c_x - p_x)t_z}{f_x}, \qquad t_y = \frac{(c_y - p_y)t_z}{f_y} \tag{1}$$

where f_x and f_y denote the focal lengths of the camera, and (p_x, p_y) is the principal point offset. To regress the object's center coordinate, we predict offsets for the 2D coordinates with respect to $(g_x, g_y) \in \mathcal{G}^2$, the top-left corner of the associated grid cell. We constrain this offset to lie between 0 and 1. The predicted center point (c_x, c_y) is defined as: $c_x = f(x) + g_x$ and $c_y = f(y) + g_y$ where $f(\cdot)$ is a 1-D sigmoid function.

Block C: This block is an object center proposal network, which returns high class confidence in regions where the object is present and low class confidence in regions where it is not. Specifically, for each image region, Block C predicts a confidence value for each object class corresponding to the presence or absence of that object's *center* in the corresponding region in the input image.

3.1.1 Duplication Removal:

After the inference of object center detection and pose estimation, which is done by one pass through our PPN, we apply non-maximal suppression to eliminate duplicated predictions when multiple cells have high confidence scores for the same object. Specifically, the inference step provides class-specific confidence scores, referring to the presence or absence of the class in the corresponding grid cell. Each grid cell produces predictions in one network evaluation, and cells with low confidence predictions are pruned using a confidence threshold. We then apply non-maximal suppression to eliminate duplicated predictions when multiple cells have high confidence scores for the same object and only consider the predictions with the highest confidence score, assuming either the object center lies at the intersection of two cells or the object is large enough to occupy multiple cells. We specifically measure the similarity of the projected bounding boxes of the 3D models given the predicted poses by computing the overlap score using intersection over union (IoU). Given two bounding boxes with high overlap score, we remove the bounding box that has

the lower confidence score. This step is repeated until all of the non-maximal bounding boxes has been removed for every class. Two projections are considered to be overlapping if the IoU score is larger than 0.3.

3.2. Multi-Attentional Refinement Network

Our proposed multi-attentional refinement network (MARN) iteratively corrects the 6D pose estimation error. Figure 3 depicts the MARN architecture and illustrates a typical refinement scenario. Two color crops (I_{im} and I_r), corresponding to an observed image and an initial pose estimate of the object in the image, are input into MARN, which outputs a pose residual estimate to update the initial predicted pose. This procedure can be applied iteratively, potentially generating finer pose estimation at each iteration.

3.2.1 Input Crops:

Input Crops are sampled from a given predicted 6D pose p. Crops circumvent the difficulty of extracting visual features from small objects. Two crops, a rendered and an RGB, are generated. Images are cropped under the assumption that only minor refinements are needed. Both crops will be used as input to the refinement network. The rendered crop is generated by rendering the 3D object model viewed according to the predicted pose p using a differentiable renderer provided by Pytorch3D library [24]. The RGB crop is generated from the original input image. We compute a bounding box, that bounds the object's 3D model, projected on the image space using the predicted pose p. We pad the bounding box by $epsilon$ pixels for each side to take into account the error introduced by the pose prediction. The enlarged bounding box is then used as a mask applied to the RGB image. Note that the mask cancels out the background, it does not crop the images. The images are cropped with a fixed size window $H \times W$, where the crop center corresponds to the object center, as defined by the 2D projection of the predicted pose p. Predicting $(\Delta c_x, \Delta c_y)$ consists of estimating how far the object center is from the image center.

3.2.2 Feature Extraction Block:

MARN refines the estimated pose by predicting the relative transformation to match the rendered view of the object to the observed view in the original image. To this end, MARN's feature extraction block is composed of two different networks: 1) a visual feature embedding network that captures visual features of the object, and 2) a flow estimation network that estimates the object "motion" between the rendered image and the observed image. The network takes two input crops: $I_r \in \mathbb{R}^{H\times W\times 3}$ and $I_{im} \in \mathbb{R}^{H\times W\times 3}$. Both crops are processed through the shared visual feature embedding network to extract visual feature representations $F_{im} \in \mathbb{R}^{H\times W\times d_{em}}$ for the image crop and $F_r \in \mathbb{R}^{H\times W\times d_{em}}$ for the render crop. Each pixel location of the embedding is a d_{em}-dimensional vector that represents the appearance information of the input image at the corresponding location. Simultaneously, the flow estimation network, based on the FlowNetSimple architecture [7], produces the optical flow between the rendered image and the observed image.

Subsequently, the visual feature map F_r, extracted from the render crop, is warped toward the visual feature map of the image crop F_{im}, guided by the flow information. Specifically, the warping function $\mathcal{W}$, extracted from the Flow estimation network, computes a new warped feature map F_w from the input F_r following the flow vectors $flow_{r\rightarrow im} \in \mathbb{R}^{H\times W\times 2}$:

$$F_w = \mathcal{W}(F_r, flow_{r\rightarrow im}) \tag{2}$$

Following [12], the warping operation is a bilinear function applied on all locations for each channel in the feature map. The warping in one channel l is performed as:

$$F_w^l(\boldsymbol{x_w}) = \sum_{\boldsymbol{x_r}} \mathcal{I}(\boldsymbol{x_r}, \boldsymbol{x_w} + \delta \boldsymbol{x_w}) F_r^l(\boldsymbol{x_r}) \tag{3}$$

where $\mathcal{I}$ is the bilinear interpolation kernel, $\boldsymbol{x_r} = (x_r, y_r)^T$ is the 2D coordinates in the visual feature embedding F_r, and $\boldsymbol{x_w} = (x_w, y_w)^T$ is the 2D coordinates in the visual feature embedding F_w. For backpropagation, gradients to the input CNN and flow features are computed as in [12]. Furthermore, the estimated optical flow $flow_{r\rightarrow im}$ is concatenated with the feature map extracted from the image crop F_{im} to produce $F_{im}^+ \in \mathbb{R}^{H\times W\times (d_{em}+2)}$.

3.2.3 Spatial Multi-Attention Block:

Estimating an object's relative transformation between two images first requires successful localization of the target object within the two inputs. MARN handles this in the spatial multi-attention block by localizing discriminative parts of the target object with spatial multi-attention maps, which robustly localize discriminative parts of the target. Therefore when the target is partially occluded, our multiple attention module can adaptively detect the visible parts while ignoring the occluded parts. Attention maps $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$, where $a_i \in \mathbb{R}^{H\times W}$ for $i \in \{1, \ldots, N\}$ and N is the number of attention maps, are extracted by generating summarized feature maps $s_i \in \mathbb{R}^{H\times W}$ for $i \in \{1, \ldots, N\}$ by applying two 1×1 convolutional operations to feature map F_w, extracted by the feature extraction block. Each attention map $a_i \in \mathcal{A}$, corresponding to a discriminative object part, is obtained by normalizing the summarized

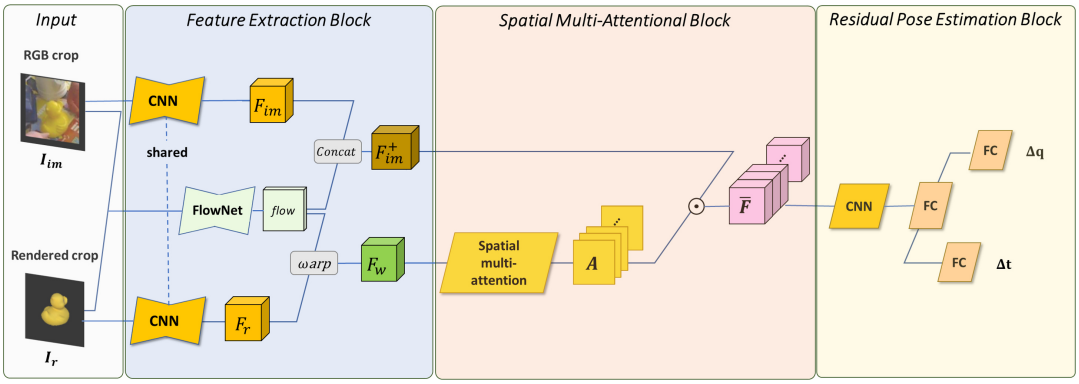

Figure 3. Our proposed multi-attentional refinement network (MARN) takes a proposed pose and iteratively refines it. Both the initial pose estimate, represented as a rendered image crop, and a real image crop are input into MARN. The network simultaneously extracts visual feature representations from the inputs, and an optical flow estimation between the inputs to warp the extracted features of the rendered crop towards the actual image crop (the Feature Extraction Block). Then, multiple attention maps, which correspond to different parts of the target object, are extracted from the warped features and applied to the feature representation of the real image crop, highlighting the important feature parts (the Spatial Multi-Attentional Block). Subsequently, the highlighted features are used to refine the pose estimate (the Residual Pose Estimation Block). The output refined pose estimate can be input into the refinement module for iterative refinement

feature map s_i using softmax:

$$a_i = \frac{\exp(s_i)}{\sum_{h=1}^{H}\sum_{w=1}^{W}\exp(s_{i,h,w})}, \quad i = 1,\dots,N \tag{4}$$

Finally, the attention map a_i and the feature map F^+_{im} are element-wisely multiplied to extract the attentional feature map $\bar{F}_i$:

$$\bar{F}_i = A_i \cdot F^+_{im}, \quad i = 1,\dots,N \tag{5}$$

where $A_i \in \mathbb{R}^{H\times W\times(d_{em}+2)}$ is the replication of the attention map a_i, $(d_{em}+2)$ times to match the dimensions of F^+_{im}. $\bar{F} \in \mathbb{R}^{H\times W\times(d_{em}+2)N}$ is the final extracted multi-attentional feature representation obtained by concatenating the attentional feature maps $\{\bar{F}_i\}_{i=1,\dots,N}$.

3.2.4 Residual Pose Estimation Block:

This block processes the residual pose estimation. First, the embedding space of the extracted feature map $\bar{F}$ is reduced the from $(d_{em}+2)N$ to 8 with three 3×3 convolutional operations. The resulting feature map is then fed into one fully connected layer, whose output is then fed into two separate fully connected and final output layers, one corresponding to the regressed rotation and the other corresponding to the translation. As explained in §3.2.1, MARN outputs an estimated relative rotation quaternion $\Delta q \in \mathbb{R}^4$ and a relative translation $[\Delta c_x, \Delta c_y, \Delta t_z]^T$. The refined pose prediction is then computed with regard to the the initial pose prediction $\hat{p} = [\hat{R}|\hat{t}]$ using $c_{x,new} = c_x + \Delta c_x$, $c_{y,new} = c_y + \Delta c_y$, $\hat{t}_{z,new} = \hat{t}_z + \Delta t_z$, and $\hat{R}_{new} = \Delta R * \hat{R}$, where (c_x, c_y) is the center of the object in the image space using $\hat{p}$, $*$ is the matrix multiplication and ΔR is the relative rotation matrix obtained from Δq. $\hat{t}_{x,new}$ and $\hat{t}_{y,new}$ are then computed using (1).

3.3. Losses:

In order to achieve accurate pose estimation, we must provide a criterion which quantifies the quality of the predicted pose. The different components of our approach are trained jointly in an end-to-end fashion with a multi-task learning objective:

$$\begin{aligned}\mathcal{L}_{total} &= \mathcal{L}_{PPN} + \mathcal{L}_{MARN} \\ &= \alpha\mathcal{L}_{pose} + \beta\mathcal{L}_{conf} + \gamma\mathcal{L}_{ref} + \kappa\mathcal{L}_{orth}\end{aligned} \tag{6}$$

where α, β, γ and κ are weight factors. Our multi-task learning objective is composed of four loss functions. First, a composite $\mathcal{L}_2$ loss function to optimize the PPN pose and center detection parameters:

$$\begin{aligned}&\mathcal{L}_{PPN} = \alpha\mathcal{L}_{pose} + \beta\mathcal{L}_{conf} \\ \text{where}\quad &\mathcal{L}_{pose} = \underset{x\in\mathcal{M}_s}{avg}\left\|(Rx+t) - (\hat{R}x+\hat{t})\right\|_2 \\ \text{and}\quad &\mathcal{L}_{conf} = \|conf_{gt} - conf_{pr}\|_2\end{aligned} \tag{7}$$

where $\|\cdot\|_2$ is the L_2 norm. $\mathcal{L}_{conf}$ is the loss term used to train the confidence block. $\mathcal{L}_{pose}$ is the loss term used to train the pose regression. $\mathcal{L}_{pose}$ is similar to the average distance (ADD) measure (further discussed in § 4). $p = [R|t]$ is the ground truth pose and $\hat{p} = [\hat{R}|\hat{t}]$ is the estimated pose. $\hat{R}$ and R are the rotation matrices computed from the predicted quaternion $\hat{q}$ and the ground truth quaternion q, respectively. $conf_{gt}$ and $conf_{pr}$ are the ground-truth and the

predicted confidence matrix, respectively. $\mathcal{M}_s \in \mathbb{R}^{M\times 3}$ is a set of points sampled from the CAD model. $\mathcal{L}_{pose}$ is only used for asymmetric objects. To handle symmetric objects, we instead use:

$$\mathcal{L}_{pose,sym} = \underset{x_1\in\mathcal{M}}{avg} \min_{x_2\in M} \left\| (Rx_1 + t) - (\hat{R}x_2 + \hat{t}) \right\|_2 \quad (8)$$

Second, MARN's loss function is defined as:

$$\mathcal{L}_{MARN} = \gamma\mathcal{L}_{ref} + \kappa\mathcal{L}_{orth}$$
$$\text{where} \quad \mathcal{L}_{ref} = \underset{x\in\mathcal{M}_s}{avg} \left\| (Rx + t) - (\hat{R}_{new}x + \hat{t}_{new}) \right\|_2 \quad (9)$$

$\mathcal{L}_{ref}$ is the same loss term used in PPN. Symmetric objects are handled similarly to PPN. $\hat{R}_{new}$ and $\hat{t}_{new}$ are the refined rotation and translation estimates.

$\mathcal{L}_{orth}$ is a regularization term used to discourage multiple attention maps locating the same discriminative object part. The regularization emphasizes orthogonality among the attention maps as proposed by [22]:

$$\mathcal{L}_{orth} = \left\| \tilde{A}^T\tilde{A} - I \right\|_2 \quad (10)$$

where $\tilde{A} = [\tilde{a}_1, \ldots, \tilde{a}_N] \in \mathbb{R}^{HW\times N}$ and $\tilde{a}_i \in \mathbb{R}^{HW}$ is the vectorized attention map of a_i.

3.4. Architectural and Training Details:

Below we present details about both our training procedures and system architecture. These details specifically pertain to experiments which follow.

Our model is optimized with Adam optimizer with weight factors (α, β , γ, κ) set to (0.1, 0.05, 0.1, 0.01).

3.4.1 PPN:

The backbone encoder in PPN consists of 23 convolution layers and 5 max-pooling layers, following the YOLOv2 architecture [26]. Additionally, we add a pass-through layer to transfer fine-grained features to higher layers. Our model is initialized with pre-trained weights from YOLOv2, with the remaining weights being randomly initialized. Input images are resized to 416×416 and split into 13×13 grids ($S = 13$). The feature embedding size of the backbone network, d, is set to be equal to 1024.

Initially, we use an additional weight factor, λ, that we apply to the confidence block output. Specifically, PPN is trained with λ set to 5 for the cells that contain target objects and 0.5 otherwise. This circumvents convergence issues with the confidence values because otherwise the early stages of training tend to converge on all zeros (since the number of cells that contain objects is likely to be much smaller than the cells that do not). In later training stages, λ is updated to penalize false negatives and false positives equally ($\lambda = 1$ for all cells). The number of points M, in the set of 3D model points $\mathcal{M}_s$, is set to $10,000$ points.

3.4.2 MARN:

For our visual feature embedding network, we use a Resnet18 encoder pre-trained on ImageNet followed by 4 up-sampling layers as the decoder. During training, the two networks are fine-tuned with shared weight parameters. We set the embedding size of the extracted features from the visual feature embedding network, d_{em}, to be equal to 32. The flow estimation network is the FlowNetS architecture populated with pre-trained weights following [7]. The network weights are frozen for the first two training epochs and unfrozen in later epochs. Once the weights are unfrozen, the component is trained in an end-to-end manner along with the other MARN components. The initial weight freeze increases training stability and ensures the output of the flow estimation network is meaningful. FlowNet output is up-sampled to match the input image crops. After a hyperparameter search, the padding offset for the mask ϵ was set to 10 pixels and the cropping window size is set to $H \times W = 256 \times 256$ applied to the original input image. Pose perturbations are used to create training data by adding angular perturbations ($5\deg$ to $45\deg$) and/or translational perturbations (0 to 1 relative to the object's diameter) to obtain a new noisy pose and rendering an image. The network is then trained to estimate the target output which is the relative transformation between the perturbed pose and the ground-truth pose.

4. Experiments:

The full model was implemented with PyTorch and all experiments were conducted on a Ubuntu server with a TITAN X GPU with 12 GB of memory. All models and code will be made publicly available soon.

In this section, our pose estimation models are compared against state-of-the-art RGB-based methods across three datasets, YCB-Video (§ 4.2), LINEMOD (§ 4.3), and LINEMOD Occlusion (§ 4.4), and obtain state-of-the-art results on all datasets, with competitive runtimes. Given a 480×640 input image, PPN alone runs at 50 fps and the full model runs at 10 fps, with two refinement iterations, which is efficient for real-time pose estimation. We also show in Supplemental material that our PPN alone has competitive performance when compared with methods that do use such information.

4.1. Evaluation Metrics:

Two standard performance metrics are used. First, the 2D-projection error, analogously to [21], measures the average distance between the 2D projections in the image space

Table 1. Comparison of our approach with state-of-the-art RGB-based methods on **YCB-Video dataset** in terms of 2D-Proj, ADD AUC and ADD(-S) metrics, averaged over all object classes for each method. We use a threshold of 2 cm for the ADD(-S) metric

Methods	HMap[20]	PVNet[21]	DeepIM†[15]	OURS†
2D-Proj	39.4	47.4	-	**55.6**
ADD AUC	72.8	73.4	81.9	**83.1**
ADD(-S) ($< 2cm$)	-	-	71.5	**73.6**

† denotes methods that deploy refinement steps.

Table 2. Results of the ablation study on different components of MARN on **YCB-Video dataset**. We use the same 2cm threshold for ADD(-S). AUC means ADD(-S) AUC. Each variant was refined with 4 iterations

Experiments	flow vectors	visual features	Attention maps	ADD(-S)	AUC
Variant 1	None	✓	None	63.7	77.2
Variant 2	✓	✓	None	68.9	79.8
Variant 3	✓	✓	single	71.2	81.9
Variant 4	✓	✓	multiple	**73.6**	**83.1**

of the 3D model points, transformed using the ground-truth pose and the predicted pose. The pose estimate is considered to be correct if it is within a selected threshold. 2D-Proj denotes the percentage of correctly estimated poses using a 2D Projection Error threshold set to 5 pixels. For symmetric objects, the 2D projection error is computed against all possible ground truth poses, and the lowest value is used. The second metric, Average 3D distance (ADD) [11], measures the average distance between the 3D model points transformed using the ground-truth pose and the predicted pose. For symmetric objects, we use the closet point distance, referred to as ADD-S in [33]. In our experiments, we denote as ADD(-S), following [33], the metric that measures the percentage of correctly estimated poses using a ADD(-S) threshold. Unless specified, in our experiments the threshold is set to 10% of the 3D model diameter. When evaluating on the YCB-Video dataset, we also report the ADD(-S) AUC as proposed in [33].

4.2. Evaluation on YCB-Video Dataset:

The YCB-Video dataset [33] has 21 objects [4] across 92 video sequences. In our experiments, we divide the data as in [33], using 80 sequences for training and 20 sequences for testing. We augment our training with 80k synthetically rendered images released by [33]. Pose predictions on the test set was refined with four MARN iterations.

4.2.1 Results:

The results in Table 1 suggest that our approach significantly outperforms state-of-the-art RGB-based methods with an average 2D-Proj accuracy of 55.6%. Compared to DeepIM [15], which also deploys refinement steps, our proposed approach achieves better performance by a margin of 1.2% and 2.1% in terms of ADD AUC and ADD(-S) respectively. Detailed results, broken down by object, can be found in the supplemental material. Our approach achieves the best results in 12 object classes out of 21 compared to other methods.

4.2.2 Ablation Study on The Refinement:

We performed an ablation study on MARN's components (detailed in § 3.2) to measure the effect of each of its components. In all, we test four variants: In variant 1, MARN only uses visual features extracted from the two input crops. In variant 2, MARN uses the flow estimation features but not the attention component, instead fusing the extracted feature map F^{+}_{im} and the warped feature map F_w with simple concatenation. In variant 3, spatial attention is added, but only a single attention map is used. Variant 4 is the production variant of MARN. Each variant refined the pose 4 times. We break down the results of the ablation study in Table 2. First, we notice that variant 1 refinement, though the simplest, still improves the pipeline performance significantly by a margin ADD(-S) of 5.2%. This finding proves that visual features help in capturing the relative transformation between two inputs, and thus helps refine the pose. Variant 2, which adds in optical flow estimation improves the performance of our refiner by 2.3% over variant 1. We conjecture that the predicted flow ensures that the network learns to exploit the relationship between both crops and thus capture the relative transformation of the object between them. Variants 3 and 4 show that the addition of attention maps helps to improve the performance of the refiner. The improvement of variant 4 over variant 3 demonstrates that multiple attention maps help achieve better performance than a single attention map. We suspect the ability of multiple attention maps to capture various salient parts of the objects helps the model highlight important features, and makes the refinement process robust to various degrees of occlusion in the dataset. We confirm these findings on the LINEMOD dataset (§ 4.3) in the supplemental material.

4.3. Evaluation on LINEMOD Dataset:

LINEMOD [11] contains 15,783 images of 13 objects, and includes 3D models of the different objects. Each image is associated with a ground truth pose for a single object of interest. The objects of interest are considered as texture-less objects, which makes the task of pose estimation challenging. The train/test split is chosen following [3] — 200 images per object are used in the training set and $1,000$ images per object in the testing set. When using the LINEMOD dataset, we opt for online data augmentation during training, to avoid over-fitting. Using this method, random in-plane translations and rotations are applied to the image along with random hues, saturations, and exposures.

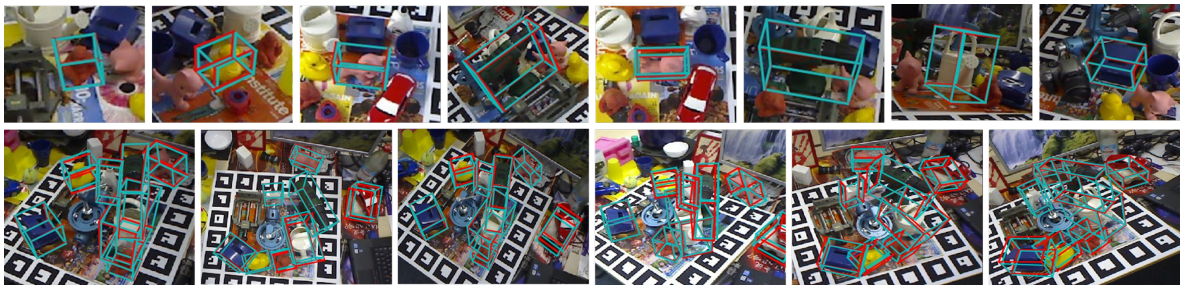

Figure 4. Pose estimation results using our method. The first row shows results from the LINEMOD dataset. The second row shows results from the LINEMOD Occlusion dataset. Cyan bounding boxes correspond to predicted poses and red bounding boxes correspond to ground-truth poses

Finally, we change the images by replacing the background with random images from the PASCAL VOC dataset [8]. Note that for testing on the LINEMOD dataset, two MARN iterations were used for refinement.

4.3.1 Results:

As shown in Table 3, our approach achieves better results than other RGB-based methods in terms of ADD(-S), with an average accuracy of 93.87% accuracy compared to an average accuracy of 88.6% for DeepIM, the second best performing method. Detailed results are shown in the supplemental material. Compared with other methods, our approach had the highest performance on 9 of the 13 object classes. Some examples of pose estimation results using the proposed approach on the LINEMOD dataset are shown in Figure 4. More qualitative results are shown in the supplemental material.

4.4. Evaluation on Occlusion Dataset:

The Occlusion dataset [2] is an extension of the LINEMOD dataset. Unlike LINEMOD, the dataset is multi-object — 8 different objects are annotated in each single image, with objects occluded by each other. Our models are trained with the same online data augmentation procedure described in the LINEMOD dataset (§ 4.3), further augmented by adding in image objects extracted from the LINEMOD dataset. Four MARN iterations were used for refinement on the Occlusion dataset.

Table 3. Results of our approach compared with state-of-the-art RGB-based methods on the **LINEMOD dataset** in terms of ADD(-S) and 2D-Proj metrics. We report percentages of correctly estimated poses averaged over all object classes

Method	Tekin[28]	PVNet[21]	SSD6D†[13]	DeepIM†[15]	OURS†
ADD(-S)	55.95	86.27	79	88.6	**93.87**
2D-Proj	90.37	99.0	-	97.5	**99.19**

† denotes methods that deploy refinement steps.

Table 4. Comparison of our approach with state-of-the-art RGB-based algorithms on **Occlusion Dataset** in terms of ADD(-S) and 2D-Proj metrics. We report percentages of correctly estimated poses averaged over all object classes

Method	HMap[20]	PVNet[21]	BB8†[23]	DeepIM†[15]	OURS†
ADD(-S)	30.4	40.77	33.88	55.5	**58.37**
2D-Proj	60.9	61.06	-	56.6	**65.46**

† denotes methods that deploy refinement steps.

4.4.1 Results:

Results in Table 4 show that, our approach achieves significant improvements over all state-of-the-art RGB-based methods. Specifically, our approach surpasses DeepIM by an ADD(-S) margin of 2.87% and PVNet by 17.6%. Furthermore, our approach significantly outperforms HMap, which was explicitly designed to handle occlusion, by an ADD(-S) margin of 27.97%. The significant improvement in performance on the Occlusion dataset, shows the importance of the different components of our MARN, and mainly the spatial multi-attentional block, in robustly recovering the poses of objects under severe occlusion. In Figure 4, we show examples of pose estimation results using the proposed approach on Occlusion dataset. Even when most objects are heavily occluded, our approach robustly recovers their poses.

5. Conclusion

We have proposed a novel end-to-end method for RGB-only 6D pose estimation. Specifically, our end-to-end approach is mainly composed of two modules. First, PPN, is a fully-CNN-based architecture that produces a one-pass pose estimates. Second, MARN, is a pose refinement network that combines visual and flow features to estimate accurate transformations between the predicted and actual object pose. Further, MARN utilizes a spatial multi-attentional block to emphasize important feature parts, making the method more robust. Our full end-to-end model achieves state-of-the-art results on three separate datasets.

References

[1] Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell, and Josef Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of cad models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3762–3769, 2014.

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[3] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6D pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3364–3372, 2016.

[4] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[5] Mohamed Chaabane, Lionel Gueguen, Ameni Trabelsi, Ross Beveridge, and Stephen O'Hara. End-to-end learning improves static object geo-localization in monocular video. *arXiv preprint arXiv:2004.05232*, 2020.

[6] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *The international journal of robotics research*, 30(10):1284–1306, 2011.

[7] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[11] 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. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[13] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[14] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *Proceedings of the IEEE international conference on computer vision*, pages 954–962, 2015.

[15] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[18] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[19] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6D pose refinement in RGB. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018.

[20] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3D object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[21] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6DoF pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019.

[22] Aaditya Prakash, James Storer, Dinei Florencio, and Cha Zhang. Repr: Improved training of convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10666–10675, 2019.

[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*, pages 3828–3836, 2017.

[24] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[26] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[27] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3D orientation learning for 6D object detection from RGB images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[28] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6D object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[29] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 124–132, 2017.

[30] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE, 2008.

[31] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6D object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[32] Christian Wöhler. *3D computer vision: efficient methods and applications*. Springer Science & Business Media, 2012.

[33] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *Robotics: Science and Systems (RSS)*, 2018.

[34] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3D bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.