A NOVEL POSE PROPOSAL NETWORK AND REFINEMENT PIPELINE FOR BETTER OBJECT POSE ESTIMATION

Ameni Trabelsi, Mohamed Chaabane, Nathaniel Blanchard, and Ross Beveridge
Department of Computer Science, Colorado State University

ABSTRACT

In this paper, we present a novel deep learning pipeline for 6D object pose estimation and refinement from RGB inputs. The first component of the pipeline leverages a region proposal framework to estimate multi-class single-shot 6D object poses directly from an RGB image and through a CNN-based encoder multi-decoders network. The second component, a multi-attentional pose refinement network (MARN), iteratively refines the estimated pose. MARN takes advantage of both visual and flow features to learn a relative transformation between an initially predicted pose and a target pose. MARN 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 the proposed pipeline outperforms state-of-the-art RGB-based methods with competitive runtime performance.

Keywords 6D object pose estimation · pose proposals · pose refinement

1 Introduction

Accurate 6D object pose estimation (i.e., object location and orientation) 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 [1, 2] 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 state-of-the-art pipeline for 6D pose estimation from RGB inputs.

Typically 6D pose estimation is performed in two stages: an initial pose estimation stage and a pose refinement stage. Following this template, we present a novel pose estimation pipeline, with a refinement stage designed to be iterative, for further refinement. Furthermore, each stage of the pipeline has novel design contributions. Our novel 6D object pose estimation method, which we call pose proposal network (PPN), is a one-step method that uses a region proposal framework to regress object rotations and translations from an RGB input. Notably, our proposed PPN method requires no additional steps, like PnP [3] and no codebook. PPN alone has competitive performance when compared with methods that do use such information (§ 4.5). The second stage of our pipeline is a Multi-Attentional Refinement Network (MARN). MARN has two novel components: visual features are fused with flow features to learn better object representations, and 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 (i.e., the output of MARN can be fed back into the input, further refining the refined pose). In our experiments, we typically saw that the greatest performance gains occurred within a couple of MARN iterations (four or less). In practice, our full pipeline achieves state-of-the-art results on three datasets.

In summary, our work makes the following contributions:

1. Our novel pose proposal fully-CNN-based architecture (PPN) yields fast and accurate pose estimations in a single-shot manner from RGB images, without additional steps.

2. Our novel pose refinement architecture (MARN) combines visual and flow features to learn an accurate transformation between the predicted object pose and the actual observed pose.
3. MARN integrates a spatial multi-attentional block that highlights important feature parts, making the refinement process more robust.

4. Our full pose estimation approach, a pipeline combining our PPN and MARN processes, outperforms state-of-the-art RGB-based methods on three commonly used benchmarks.

2 Related Work

6D pose estimation has a long and storied history [4], but, due to space constraints, we will focus this section on methods that use RGB inputs. Traditional RGB methods for pose estimation typically match detected local keypoints or hand-crafted features with known object models [5][6][7][8]. 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 [9][10].

Most existing RGB-based methods [11][12][13][14] take advantage of deep learning techniques used for object detection [15][16][17] or image segmentation [18] and leverage them for 6D pose estimation. [14][19] uses CNNs to extract object keypoints, then solves the 6D poses using PnP [3]. [20] 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 detector and pose estimator based on region proposals. [11][21] use region proposal framework to detect the object within the input image and then use additional steps to solve its pose. Our work extends the framework in a complete end-to-end network for 6D pose estimation.

6D pose refinement has been utilized to improve the performance of several pose estimation methods [1][22]. Recent refinement methods have been deep-learning based [23][24], relying on CNNs to predict a relative transformation between the initial pose prediction and the target pose. Our proposed refinement architecture also takes advantage of CNNs to extract visual and flow features in order to capture the pose transformation from the prediction to the target pose, and further employs a multi-attentional block that efficiently highlights discriminative feature parts, improving the robustness to noise and occlusion.

3 Methods

In this paper, we estimate the 6D pose of a set of known objects present in an RGB image. We propose a two-step approach: first, our novel end-to-end pose proposal network (PPN) (§3.1) regresses 6D pose proposals from different regions of an RGB image. Second, the PPN estimation is refined with our novel pose refinement network: multi-attentional refinement network (MARN) (§3.2). In the following, the 6D pose is represented as a homogeneous transformation matrix, \( p = [R|t] \in SE(3) \), composed of a rotation matrix \( R \in SO(3) \) 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

Inspired by region proposal frameworks like YOLO [17], we reframe the object pose estimation as a combined object detection and pose estimation problem, regressing from image pixels to region proposals of object centers and poses. Figure 1 illustrates our 6D object pose proposal network architecture. Our network has two stages: first, a backbone encoder, modeled on the YOLOv2 framework [25], extracts high-dimensional region feature representations from the input image. Second, the obtained feature representations, generated by the first stage, are embedded into low-dimensional, task-specific features. Then, the network outputs three sets of region proposals for object centers, translations, and rotations. Specifically, the backbone encoder (Figure 1A) 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 \( (i,j) \in \mathbb{G}^2 \) s.t. \( \mathbb{G} = \{1, ..., S \} \) and \( d \) is the embedding size. \( \mathcal{F} \) is decoded by 3 parallel convolutional blocks (Figure 1B) that produce a fixed-size collection of region proposals \( \{(\mathcal{C}o_{i,j}^{x}, T_{i,j}^{x}, Q_{i,j}^{x})\} \) for each object in the set of target objects \( o_k \in \{o_1, ..., o_C\} \), where \( C \) is the number of target objects. The detailed architectures of the three blocks are depicted in Figure 1C.

**Block A:** This block is a translation proposal network that regresses a 4-dimensional quaternion vector \( Q_{i,j}^{x} \) for each image region and object class.

**Block B:** This block is a translation proposal network that regresses a 3-dimensional translation vector \( T_{i,j}^{x} \) 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 [22], we regress the object center coordinates in the image space \( c = (c_x, c_y)^T \).
Figure 1: Our Pose Proposal Network (PPN) Architecture. The encoder/multi-decoder network 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 of the detected object. \( C \). Architectural details of blocks \( A \), \( B \), and \( C \) in our PPN 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:

\[
\begin{align*}
t_x &= \frac{(c_x - p_x)t_z}{f_x}, \quad t_y = \frac{(c_y - p_y)t_z}{f_y}
\end{align*}
\]

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 \((g_x, g_y)\) \( \in \mathbb{G}^2 \), 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:

\[
\begin{align*}
c_x &= f(x) + g_x \\
c_y &= f(y) + g_y
\end{align*}
\]

where \( f(\cdot) \) is a 1-D sigmoid function.

**Block C**: This block is a confidence proposal network, which should have high confidence in regions where the object is present and low 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.

**Duplication Removal**: After the inference of object 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
Figure 2: Our proposed multi-attentional refinement network (MARN) takes a proposed pose and iteratively refines it. In the context of our pipeline, the initial pose estimate, represented as a render image crop and a real image crop, are input into MARN. First, the network extracts visual feature representations from the inputs and an optical flow estimation between the two inputs (the Feature Extraction Block). Then, multiple attention maps, which correspond to different parts of the target object, are extracted from the flow and render crop 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 MARN for iterative refinement.

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.

PPN Loss: PPN parameters are optimized with an overall composite loss:

\[ L_{PPN} = \alpha L_{pose} + \beta L_{conf} \]

where

\[ L_{pose} = \text{avg}_{x \in \mathcal{M}} \| (Rx + t) - (\hat{R}x + \hat{t}) \|_1 \]

and

\[ L_{conf} = \| \text{conf}_{gt} - \text{conf}_{pr} \|_2 \]

where \( \| \cdot \|_2 \) is the \( L_2 \) norm and \( \| \cdot \|_1 \) is the \( L_1 \) norm. \( L_{conf} \) is the loss term used to train the confidence block. \( L_{pose} \) is the loss term used to train the pose regression. \( L_{pose} \) is similar to the average distance (ADD) measure (further discussed in §4) with a more robust \( L_1 \) norm. \( p = [R|t] \) 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. \( \text{conf}_{gt} \) and \( \text{conf}_{pr} \) are the ground-truth and the predicted confidence matrix respectively. \( \mathcal{M} \subseteq \mathbb{R}^{M \times 3} \) is a set of points sampled from the CAD model. \( \alpha \) and \( \beta \) are weight factors. \( L_{pose} \) is only used for asymmetric objects. To handle symmetric objects, we instead use:

\[ L_{pose,sym} = \text{avg}_{x_1 \in \mathcal{M}, x_2 \in \mathcal{M}} \| (Rx_1 + t) - (\hat{R}x_2 + \hat{t}) \|_1 \]

3.2 Multi-Attentional Refinement Network

Our proposed multi-attentional refinement network (MARN) iteratively corrects the 6D pose estimation error. Given the success of end-to-end trainable models [26, 27], we opt for an end-to-end refinement pipeline. Figure 2 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.

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 $p$. 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.

**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 [28], 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 $W$, extracted from the flow estimation network, computes a new warped feature map $F_w$ from the input $F_r$ following the flow vectors $flow_{r \rightarrow im} \in \mathbb{R}^{H \times W \times 2}$:

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

(4)

Following [29], 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:

$$F_w^l(x_w) = \sum_{x_r} I(x_r, x_u + \delta x_w) F_r^l(x_r)$$

(5)

where $I$ is the bilinear interpolation kernel, $x_r = (x_r, y_r)^T$ is the 2D coordinates in the visual feature embedding $F_r$, and $x_u = (x_u, y_u)^T$ 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 [29]. Furthermore, the estimated optical flow $flow_{r \rightarrow im}$ is concatenated with the feature map extracted from the image crop $F_{im}$ to produce $F_{im}^+ \in \mathbb{R}^{H \times W \times (d_{em}+2)}$.

**Spatial Multi-Attention Block:**

Estimating an object’s relative transformation between two images 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 $A = \{a_1, a_2, \ldots, a_N\}$, 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 $1 \times 1$ convolutional operations to feature map $F_w$, extracted by the feature extraction block. Each attention map $a_i \in A$, corresponding to a discriminative object part, is obtained by normalizing the summarized feature map $s_i$ using softmax:

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

(6)

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

$$\hat{F}_i = A_i \cdot F_{im}^+, \quad i = 1, \ldots, N$$

(7)

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 $F_{im}^+$. $\hat{F} \in \mathbb{R}^{H \times W \times (d_{em}+2)N}$ is the final extracted multi-attentional feature representation obtained by concatenating the attentional feature maps $\{\hat{F}_i\}_{i=1,\ldots,N}$. Inspired by [30], we add a regularization term to the total loss function
to discourage multiple attention maps locating the same discriminative object part. The regularization emphasizes orthogonality among the attention maps:

\[ \mathcal{L}_{\text{orth}} = \| \mathbf{A}^T \mathbf{A} - I \|_2 \]  

(8)

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

Residual Pose Estimation Block:

This block processes the residual pose estimation. First, the embedding space of the extracted feature map \( \bar{F} \) is reduced 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 [32], MARN outputs an estimated relative rotation quaternion \( \Delta q \in \mathbb{S}^3 \) 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 initial pose prediction \( \hat{p} = [\hat{R} | \hat{t}] \) using \( c_{x,new} = c_x + \Delta c_x \), \( c_{y,new} = c_y + \Delta c_y \), \( \hat{t}_{x,new} = \hat{t}_z + \Delta t_z \), and \( \hat{R}_{new} = \hat{R} * \hat{R} \), where \((c_x, c_y)\) is the center of the object in the image space using \( \hat{p} \), \( * \) is the matrix multiplication and \( \hat{R} \) is the relative rotation matrix obtained from \( \Delta q \). \( \hat{t}_{x,new} \) and \( \hat{t}_{y,new} \) are then computed using (1).

MARN Loss:

MARN’s loss function is defined as:

\[
\mathcal{L}_{\text{MARN}} = \lambda \mathcal{L}_{\text{pose}} + \mu \mathcal{L}_{\text{orth}}
\]

where

\[
\mathcal{L}_{\text{pose}} = \text{avg}_{x \in \mathcal{M}_s} \| (Rx + t) - (\hat{R}_{new}x + \hat{t}_{new}) \|_1
\]

\( \mathcal{L}_{\text{pose}} \) is the same loss term used in PPN. \( \hat{R}_{new} \) and \( \hat{t}_{new} \) are the refined rotation and translation estimate. \( \lambda \) and \( \mu \) are weight factors.

3.3 Architectural and Training Details:

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

PPN:

The backbone encoder in our final PPN consists of 23 convolution layers and 5 max-pooling layers, following the YOLOv2 architecture. 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 [25], 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, \( \gamma \), that we apply to the confidence block output. Specifically, PPN is trained with \( \gamma \) 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, \( \gamma \) is updated to penalize false negatives and false positives equally (\( \gamma = 1 \) for all cells). PPN is optimized with stochastic gradient descent with a weight factor \( \alpha \) set to 0.05 and \( \beta \) is set to 1. The number of points \( M \), in the set of 3D model points \( \mathcal{M}_s \), is set to 10,000 points.

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 [28]. 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 \( \epsilon \) 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. For this, the weight factors (\( \lambda, \mu \)) are set to (0.1, 0.05).


## Methods

| Methods | HMap[12] | PVNet[13] | DeepIM[23] | OURS† |
|---------|----------|-----------|------------|-------|
| 2D Proj | 39.3     | 41.4      | 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 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.

4 Experiments:

Both PPN and MARN were implemented with PyTorch and all experiments were conducted on a Ubuntu server with a TITAN X GPU with 12 GB of memory. All code will be made publicly available upon publication. We compare our pose estimation models 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 x 640 input image, PPN runs at 50 fps and MARN runs at 25 fps, per iteration, which is efficient for real-time pose estimation. Further, in § 4.5, we highlight a key contribution of our initial PPN stage: it is a fast, fully CNN-based network, that obtains competitive results against other existing methods.

4.1 Evaluation Metrics:

Two standard performance metrics are used. first, the 2D-projection error, analogously to [13], measures the average distance between the 2D projections in the image space 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) [31], 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 closest point distance, referred to as ADD-S in [22]. In our experiments, we denote as ADD(-S), following [22], 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 [22].

4.2 Evaluation on YCB-Video Dataset:

The YCB-Video dataset [22] has 21 objects [32] across 92 video sequences. YCB objects have varying shapes, sizes, and symmetries, as well as different levels of occlusion and lighting conditions of the scenes. In our experiments, we divide the data as in [22], using 80 sequences for training and 20 sequences for testing. We augment our training with 80k synthetically rendered images released by [22]. Pose predictions on the test set was refined with four MARN iterations.

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 [23], which also deploys refinement steps, the 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 on the YCB-Video Dataset: In Table 2, we show detailed pose estimation results on the YCB-Video dataset[22] in terms of ADD AUC. Our approach achieves the best results in 12 object classes out of 21 compared to other methods. DeepIM, surpasses other methods on 6 object classes out of 21, and HMap outperforms other methods on 4 object classes.

Ablation Study of The Refiner on YCB-Video Dataset: We performed an ablation study on MARN’s components (detailed in § 3.2) to measure the effect of our novel 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 3. 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
Table 2: Detailed results of our approach and other existing RGB-based methods on the different objects of the YCB-Video dataset in terms of ADD AUC

| Experiments | flow features | visual features | Attention maps | ADD(-S) | AUC |
|-------------|---------------|----------------|----------------|---------|-----|
| Variant 1   | None          | ✓              | None           | 63.7    | 77.4|
| Variant 2   | ✓             | ✓              | None           | 68.9    | 79.8|
| Variant 3   | ✓             | ✓              | single         | 71.2    | 81.9|
| Variant 4   | ✓             | ✓              | multiple       | 73.6    | 83.1|

Table 3: 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.

4.3 Evaluation on LINEMOD Dataset:

LINEMOD [31] is one of the standard datasets used for evaluation of pose estimation methods. It 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 textureless objects, which makes the task of pose estimation challenging. The train/test split is chosen following [33] — 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 overfitting. Using this method, random in-plane translations and rotations are applied to the image along with random hues, saturations, and exposures. Finally, we change the images by replacing the background with random images from the PASCAL VOC dataset [34]. Note that for testing on the LINEMOD dataset, two MARN iterations were used for refinement.

Table 4: 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* [21] | PVNet* [13] | SSD6D* [11] | DeepIM† [23] | OURS† |
|--------------|-------------|-------------|-------------|--------------|-------|
| ADD(-S)      | 55.95       | 59.75       | 86.27       | 88.6         | 90.81 |
| 2D-Proj      | 90.37       | 99.0        | -           | 97.5         | 98.84 |
Table 5: Detailed Results of our approach and other existing RGB-based methods on the different objects of the LINEMOD dataset in terms of ADD metric

| Method          | Tekin[21] | PVNet[13] | BB8†[14] | SSD6D†[11] | DeepIM†[23] | OURS† |
|-----------------|-----------|-----------|----------|------------|-------------|-------|
| ape             | 21.62     | 43.62     | 40.4     | 65         | 77          | 53.78 |
| benchvise       | 81.80     | 99.90     | 91.8     | 80         | 97.5        | 98.25 |
| cam             | 36.57     | 86.86     | 55.7     | 78         | 93.5        | 91.17 |
| can             | 68.80     | 95.47     | 64.1     | 86         | 96.5        | 97.06 |
| cat             | 41.82     | 79.34     | 62.6     | 70         | 82.1        | 83.64 |
| driller         | 63.51     | 96.43     | 74.4     | 73         | 95          | 96.63 |
| duck            | 27.23     | 52.58     | 44.30    | 66         | 77.7        | 79.34 |
| eggbox*         | 69.58     | 99.15     | 57.8     | 100        | 97.1        | 98.46 |
| glue*           | 80.02     | 95.66     | 41.2     | 100        | 99.4        | 99.7  |
| holepuncher     | 42.63     | 81.92     | 67.20    | 49         | 52.8        | 82.49 |
| iron            | 74.97     | 98.88     | 84.7     | 78         | 98.3        | 99.04 |
| lamp            | 71.11     | 99.33     | 76.5     | 73         | 97.5        | 97.61 |
| phone           | 47.74     | 92.41     | 54.0     | 79         | 87.7        | 91.37 |
| **MEAN**        | **55.95** | **86.27** | **62.7** | **79**     | **88.6**    | **90.81** |

† denotes methods that deploy refinement steps.
* denotes symmetric objects.

Table 6: Results of the ablation study on different components of our refinement network MARN on LINEMOD dataset

| Experiments | flow features | CNN features | Attention maps | ADD | 2D-Reproj |
|-------------|---------------|--------------|----------------|-----|-----------|
| Variant 1   | None          | ✓            | None           | 86.04 | 96.31 |
| Variant 2   | ✓             | ✓            | None           | 88.59 | 97.68 |
| Variant 3   | ✓             | ✓            | single         | 89.77 | 98.39 |
| Variant 4   | ✓             | ✓            | multiple       | 90.81 | 98.84 |

Table 7: Comparison of our approach with state-of-the-art RGB-based algorithms on Occlusion in terms of ADD(-S) and 2D-Proj metrics. We report percentages of correctly estimated poses averaged over all object classes

| Method          | HMap[12] | PVNet[13] | BB8†[14] | SSD6D†[11] | DeepIM†[23] | OURS† |
|-----------------|----------|-----------|----------|------------|-------------|-------|
| ADD(-S)         | 30.4     | 40.77     | 33.84    | 55.5       | 58.37       | 65.46 |
| 2D-Proj         | 60.9     | 61.06     | -        | 56.6       | 65.46       |       |

† denotes methods that deploy refinement steps.
Figure 3: Results of poses predicted using the proposed approach. The first row shows results from the LINEMOD dataset. The second row shows results from the LINEMOD Occlusion dataset. In both rows, the cyan bounding boxes correspond to predicted poses and red bounding boxes correspond to ground-truth poses.

Table 8: Evaluation Results of our PPN compared to other state-of-the-art RGB-based methods that do not use refinement on three datasets: YCB-Video, LINEMOD and Occlusion using the 2D-Proj metric.

| Methods     | PoseCNN [22] | HMap [12] | PVNet [13] | PPN (ours) |
|-------------|---------------|-----------|------------|------------|
| YCB-Video   | 3.72          | 39.3      | 47.4       | 49.3       |
| LINEMOD     | 62.7          | -         | 99.0       | 96.06      |
| Occlusion   | 17.2          | 60.9      | 61.06      | 61.10      |

Results: Results in Table 7 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 5, 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.

4.5 PPN Only: An Efficient Pose Estimator for Real Time Applications

We evaluate the performance of PPN, our pose estimation network without refinement, and compare it with state-of-the-art methods that do not use refinement. Results in Table 8 on three benchmarks suggest that PPN alone performs better than HMap and PoseCNN on all three datasets, and performs comparably to PVNet. Unlike these approaches, PPN has the highest speed (50 fps), is completely end-to-end, and does not require any additional steps such as the PnP algorithm. Thus, we suggest that PPN alone is fast and robust enough to be deployed in real-world applications.

5 Additional Qualitative Results

In Figure 4 to 6, we show qualitative results on the three datasets: YCB-Video [22], LINEMOD [31] and Occlusion [9] datasets. These examples show that our proposed method is robust to severe occlusions, scene clutter, different illumination and reflection.
Figure 4: Examples of 6D object pose estimation results on the YCB-Video dataset. Each row corresponds to images from one testing video. Red bounding boxes correspond to ground truth poses, cyan bounding boxes correspond to predicted poses using our approach.
6 Conclusion

We have proposed novel methods for RGB-only pose proposal and refinement. Specifically, we introduce PPN, our novel pose estimator, and MARN, a novel pose refiner. PPN is a fully-CNN-based architecture that produces single-shot pose estimates. MARN is an end-to-end pose refinement network that combines visual and flow features to estimate accurate transformation 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 pipeline combines our PPN and MARN processes, and achieves state-of-the-art results on three separate datasets.
Figure 6: Examples of 6D object pose estimation results on different objects from the Occlusion dataset. Red bounding boxes correspond to ground truth poses, cyan bounding boxes correspond to predicted poses using our approach.
References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[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] 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.

[18] 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.

[19] 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.
[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Sebastian Ruder. An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098, 2017.

[27] Mohamed Chaabane, Ameni Trabelsi, Nathaniel Blanchard, and Ross Beveridge. Looking ahead: Anticipating pedestrians crossing with future frames prediction. In The IEEE Winter Conference on Applications of Computer Vision (WACV), March 2020.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.