2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

MV6D: Multi-View 6D Pose Estimation on RGB-D Frames Using a Deep Point-wise Voting Network

Fabian Duffhauss[1], Tobias Demmler[2], and Gerhard Neumann[3]

***Abstract*— Estimating 6D poses of objects is an essential computer vision task. However, most conventional approaches rely on camera data from a single perspective and therefore suffer from occlusions. We overcome this issue with our novel multi-view 6D pose estimation method called MV6D which accurately predicts the 6D poses of all objects in a cluttered scene based on RGB-D images from multiple perspectives. We base our approach on the PVN3D network that uses a single RGB-D image to predict keypoints of the target objects. We extend this approach by using a combined point cloud from multiple views and fusing the images from each view with a DenseFusion layer. In contrast to current multi-view pose detection networks such as CosyPose, our MV6D can learn the fusion of multiple perspectives in an end-to-end manner and does not require multiple prediction stages or subsequent fine tuning of the prediction. Furthermore, we present three novel photorealistic datasets of cluttered scenes with heavy occlusions. All of them contain RGB-D images from multiple perspectives and the ground truth for instance semantic segmentation and 6D pose estimation. MV6D significantly outperforms the state-of-the-art in multi-view 6D pose estimation even in cases where the camera poses are known inaccurately. Furthermore, we show that our approach is robust towards dynamic camera setups and that its accuracy increases incrementally with an increasing number of perspectives.**

I. Introduction

6D pose estimation is a key technology for autonomous driving [1], [2], [3], [4], robot manipulation [5], [6], [7], [8], augmented reality [9], [10], and human machine interaction [11], [12]. It describes the prediction of position and orientation of objects in 3D space. Traditional pose estimation methods mostly rely on single RGB(-D) images [8], [13], [14], [7], [6] or point cloud data [15], [16], [17]. However, these methods suffer significantly from occlusions by other objects. To overcome this issue, we present our novel deep learning approach called MV6D which is illustrated in Fig. 1.

MV6D takes multiple RGB-D images depicting a cluttered scene from different viewpoints which are ideally very distinct. Whereas the RGB images are processed individually, we fuse all depth images to a joint point cloud. Similar to PVN3D [7] our approach predicts pre-defined 3D keypoints for each object using independent feature encoding networks for both modalities. In a final least-squares fitting [18] step, we predict the 6D poses of all objects in the scene. Since established 6D pose estimation datasets, such as YCB-Video [8], LineMOD [19], and T-LESS [20] do not provide many frames from very distinct perspectives, we created three photorealistic datasets with cluttered scenes using YCB objects [21].

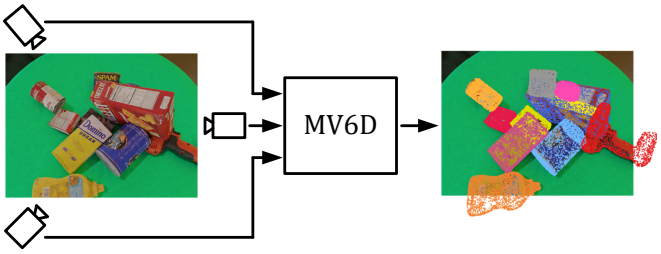

Fig. 1. Overview of our MV6D approach. MV6D takes multiple RGB-D input images and predicts the 6D poses of all objects in a cluttered scene.

We show that our multi-view approach outperforms the related single-view method PVN3D [7] and the state-of-the-art multi-view 6D pose estimator CosyPose [22] significantly. Even in the case of inaccurate camera positioning, the accuracy of our MV6D network is much higher than the baseline methods. Furthermore, we show that our approach copes with variable camera setups. We additionally evaluated the performance with different number of input images and show that employing a second view already leads to a large accuracy increase compared to the single-view setting while more views can further improve the results.

Our main contributions are:

- We present a novel deep learning approach for 6D pose estimation on RGB-D images which learns to fuse the information from multiple views in an end-to-end manner.
- We present three photorealistic datasets with multi-view RGB-D data and ground truth for instance semantic segmentation and 6D pose estimation.
- We show that our multi-view approach outperforms the corresponding single-view approach significantly even when the positions of the cameras are known inaccurately.
- Furthermore, we show that we considerably outperform current multi-view pose estimation approaches [22] which use a much more complex architecture including fusion of single view predictions and fine-tuning.

II. Related Work

There has been a lot of research recently about 6D pose estimation and related tasks like 3D object detection which we present in the following.

[1]Fabian Duffhauss is with the Bosch Center for Artificial Intelligence, Renningen, Germany, and the University of Tübingen, Germany. Fabian.Duffhauss@de.Bosch.com
[2]Tobias Demmler is with the Robert Bosch GmbH, Stuttgart, Germany. Tobias.Demmler@de.Bosch.com
[3]Gerhard Neumann is with the Karlsruhe Institute of Technology, Karlsruhe, Germany. Gerhard.Neumann@kit.edu

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9982268

A. Pose Estimation on Single RGB Images

Traditional pose estimation methods [23], [24], [25], [26], [27], [5], [28], [13] extract local features from the given RGB image and match them to the corresponding features in its 3D model. Based on the 2D-3D-correspondences a Perspective-n-Point (PnP) algorithm [29] can be applied to estimate the object's pose. Even though feature-based methods can handle occlusions up to a certain degree, the detection of 2D keypoints does not work well on objects without a distinctive texture [20].

In contrast, methods based on template-matching [30], [31], [32] can cope with textureless objects. Templates of an object can be generated by rendering its corresponding 3D model from diverse views. Finding a match between a rendered template and a part of the input image provides the 6D pose of the corresponding object. However, template-based methods suffer from occlusions as the matching becomes inaccurate.

There are also end-to-end trainable neural networks [33], [34], [8], [35], [36], [37] which directly predict the objects' poses based on a single RGB image.

However, often the generalization of direct methods is an issue due to the non-linearity of the rotation space [13]. [33], [38], [39] overcome this issue by discretizing the rotation space. Another common procedure is to refine the predicted poses, e.g., by applying the iterative closest point (ICP) algorithm [40] using additional depth data as in PoseCNN [8] or SSD-6D [36]. Alternatively, deep learning-based pose refining networks like DPOD [41] or DeepIM [34] are proposed for faster and more accurate refinement without the need of depth data.

B. Pose Estimation on Single Point Clouds

Recently, due to the rapid technological progress of depth and LiDAR sensors, many pose estimation methods were developed based on a single depth image or a single point cloud [42], [43]. In this area, there are methods [44], [45] that directly predict oriented 3D bounding boxes using 3D convolutions. However, 3D convolutions are computationally expensive which leads to high inference times. To reduce the computational complexity, it is common even in modern approaches [15], [16], [17] to apply feature encoding networks based on PointNet [46] or PointNet++ [47] which are able to extract geometric features. To do that, the point cloud is either divided into voxels [15], [16] or into vertical pillars [17].

Recently, Qi et al. [48] introduced deep Hough voting for end-to-end 3D object detection. Their network VoteNet generates votes to object centers which are fused to obtain object proposals. Building upon that, Xie et al. [49] further improved the feature encoding of seed points by an attentive multi-layer perceptron, a vote attraction loss, and vote weighting.

However, as methods based on depth images or point clouds cannot exploit texture, their performance is limited to applications where textures are not relevant.

C. Pose Estimation on Single RGB-D Images

RGB-D based approaches try to combine the advantage of both modalities. AVOD [2] and MV3D [1] use convolutional feature extractor networks followed by a 3D object proposal network for fusing RGB images and LiDAR point clouds. The latter is compressed into a bird's eye view and in case of MV3D an additional front view projection is used. However, these approaches are based on the assumption that all objects of interest are located on a plane.

PointFusion [3] proposes the usage of PointNet [46] for point cloud feature extraction and introduces a dense fusion module for combining point cloud features and RGB features which were created by a CNN. DenseFusion [14] transfers that concept from 3D object detection in autonomous driving to 6D pose estimation for robotics and introduces a neural network for iterative pose refinement.

PVN3D [7] further improves DenseFusion [14] by applying PointNet++ [47] and by introducing a deep Hough voting network for 3D keypoint detection. The 6D poses are estimated by a least-squares fitting algorithm [18]. FFB6D [6] enhances PVN3D [7] with a bidirectional fusion module that combines the features representing texture and geometric information in each encoding and decoding layer. Furthermore, they replaced PointNet++ [47] by a RandLA-Net [50] for point cloud feature encoding.

D. Multi-View Pose Estimation

Multi-view approaches aim to increase the detection accuracy by combining information from multiple perspectives. Zeng et al. [51] present a 6D pose estimation approach based on 15 to 18 RGB-D images that are recorded by a robot arm from very similar perspectives. They use a fully convolutional neural network to perform 2D object segmentation on each RGB image individually. Afterwards, the segmentation results are fused with the depth images into a single segmented point cloud. The 6D poses are estimated using an ICP algorithm [40].

Sock et al. [52] propose an active multi-view framework with next-best-view prediction and hypothesis accumulation. Based on previous single-shot pose hypotheses they predict the next best camera perspectives and select the most likely object poses. Li et al. [53] introduces an end-to-end trainable CNN-based architecture for 6D pose estimation based on a single RGB or RGB-D image which is used multiple times with images from different viewpoints. Afterwards, the best hypothesis is selected using a voting score that suppresses outliers.

Recently, Labbé et al. [22] present an approach for 6D pose estimation based on multiple RGB images. Their method CosyPose employs DeepIM [34] for predicting object candidates in each view separately. Subsequently, they match the object candidates that belong to the same object instance and refine the pose estimates by an object-level bundle adjustment [54]. However, the approach fails if an object is detected in just a single view.

All methods so far apply deep neural networks independently on each view which leads to high computational effort

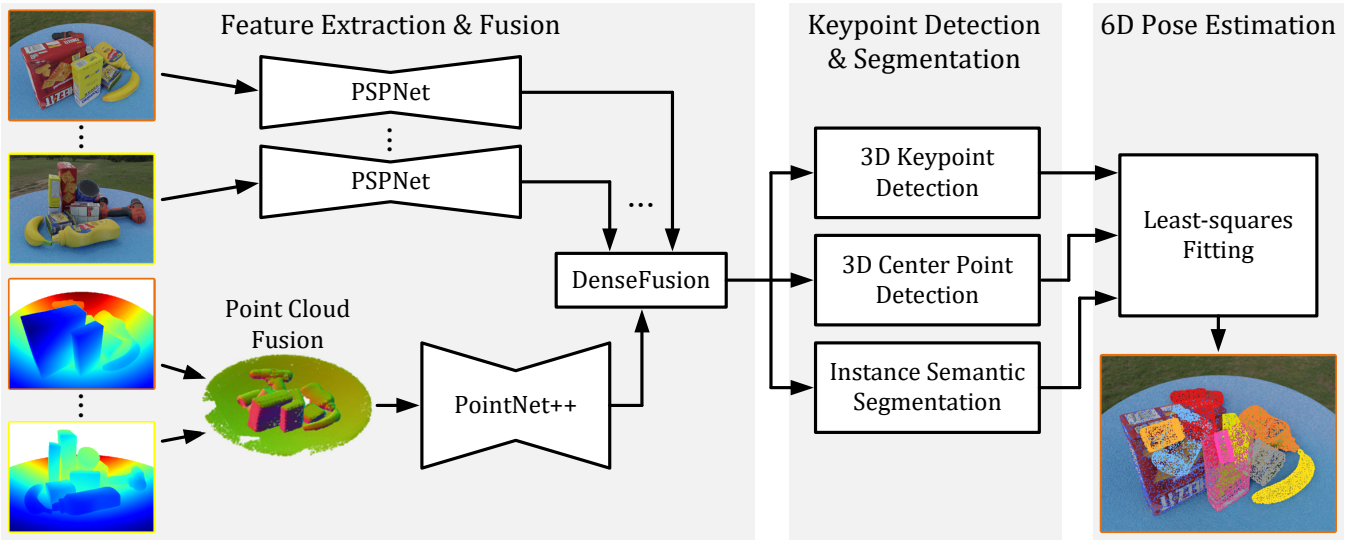

Fig. 2. Architecture of our MV6D network. Given multiple RGB-D images, MV6D extracts visual features from the RGB images and geometric features from a point cloud which is created by fusing all depth images. A DenseFusion network [14] fuses visual and geometric features. Based on a 3D keypoint detection, a 3D center point detection, and an instance semantic segmentation module, we predict 6D poses using least-squares fitting [18].

due to redundancy and sub-optimal use of information as there is no prediction that can use all information. To the best of our knowledge, our approach is the first that directly fuses the features from multiple RGB-D views before performing the pose estimation based on that.

III. Multi-View Fusion for 6D Pose Estimation

We propose MV6D, a deep neural network for 6D pose estimation based on multi-view RGB-D data. In the following, we define the problem of multi-view 6D pose estimation and present our deep learning approach to this.

Given an object that is observed by a camera, 6D pose estimation describes the task of predicting a rigid transformation $\boldsymbol{p} = [\boldsymbol{R}|\boldsymbol{t}]$ that transforms the object from the object coordinate system into the camera coordinate system. The 6D pose $\boldsymbol{p} \in SE(3)$ is composed of a rotation $\boldsymbol{R} \in SO(3)$ and a translation $\boldsymbol{t} \in \mathbb{R}^3$. Our goal is to predict the 6D poses of all objects in a given cluttered scene based on the RGB-D images of multiple cameras with known camera pose. We assume that the 3D models of the objects are known and not more than a single instance per object class occurs in each scene.

Fig. 2 illustrates the network architecture of our approach which is composed of three stages that are inspired by the single-view network PVN3D [7]. The first stage accepts a variable number of RGB-D frames, extracts relevant features, and fuses them to a joint feature representation of the entire input scene. The second stage contains network heads for instance semantic segmentation and 3D keypoint prediction for each object. The third stage estimates the 6D poses of all objects in the scene in a least-squares fitting manner. While PVN3D uses only a single view, we propose a new mechanism to fuse the point clouds as well as the RGB images of several views into a single consistent feature representation using a modified DenseFusion module [14].

A. Multi-View Fusion Architecture

Point Cloud Fusion: To extract geometric features, we first convert all depth images into point clouds and combine them to a single point cloud using the known camera poses. As previous methods [6], [7], we further process only a random subset of points and attach the corresponding RGB value as well as the surface normal to each remaining point. For feature extraction, we apply a PointNet++ [47] with multi-scale grouping.

RGB Image Fusion: For each RGB image, we independently extract pixel-wise visual features using modified PSPNets [55] which contain a ResNet-34 [56] pretrained on ImageNet [57] as backbone. All PSPNets share the same parameters. Each point in the processed point cloud is associated to a corresponding pixel of the RGB image from the view that generated the point. Similar to a DenseFusion network [14], we concatenate to each point in the point cloud the PSPNet feature vector of the associated pixel from the associated view as shown in Fig. 3. Hence, even if points are spatially close to each other, they can obtain the visual features from different RGB images if they have been generated from different views. Subsequently, we compute a global feature vector by an MLP followed by an average pooling layer that aggregates the information from the whole point cloud. This feature vector is then again appended to the geometric and RGB features of each point in the point cloud and used for further processing in the instance segmentation or keypoint detection heads.

B. Network Heads

Instance Semantic Segmentation: The instance semantic segmentation module in Fig. 2 consists of a semantic segmentation head and a center offset head as in [7]. Both submodules take the point-wise feature vectors (consisting of visual, geometric, and global features) and process them

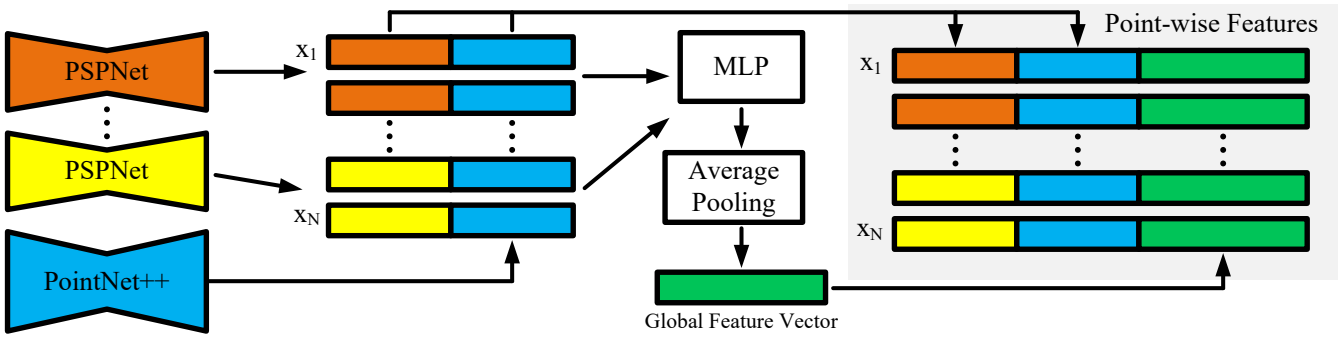

Fig. 3. DenseFusion module of our network architecture. While keeping a mapping between the visual features of the PSPNets [55] and the corresponding geometric features of the PointNet++ [47], we compute a global feature vector and concatenate the results to obtain point-wise feature vectors.

with shared multi-layer perceptrons (MLPs). The semantic segmentation module predicts an object class for each point. The center offset module estimates the translation offset from the given point to the center of the object that it belongs to. Following [7], we apply mean shift clustering [58] to obtain the final object center predictions. These are then used to further refine the segmentation map by rejecting points which are too far away from the object center.

3D Keypoint Detection: In advance of the training, we select eight target keypoints from the mesh of each object using the farthest point sampling (FPS) algorithm [59] as in [7]. Using a shared MLP, we predict the translation offset from each point to each target keypoint of the class that the object belongs to. Adding the predicted offsets to the corresponding points from the point cloud results in a set of keypoint predictions. All keypoint predictions belonging to an instance are clustered by mean shift clustering [58] in order to obtain the final 3D keypoint predictions as in [7].

C. Loss Function

The network is trained as in [7] by minimizing the multi-task loss function

$$L_{\text{multi-task}} = \lambda_1 L_{\text{keypoints}} + \lambda_2 L_{\text{semantic}} + \lambda_3 L_{\text{center}}, \quad (1)$$

where the 3D keypoints detection loss $L_{\text{keypoints}}$ and the center voting loss L_{center} are L1 losses, whereas the semantic segmentation loss L_{semantic} is a Focal loss [60]. $\lambda_1 = \lambda_3 = 12$ and $\lambda_2 = 1$ are manually chosen weights for the individual loss functions. These weights correspond to the reference implementation of He et al. [7] on GitHub but not to the values they published in their paper (i.e., $\lambda_1 = \lambda_2 = \lambda_3 = 1$). In their code, they actually add keypoint and center point losses for the entire batch using batch size 24 while averaging the segmentation loss over the entire batch. We made the implementation independent of the batch size by averaging over all losses individually which explains the high values for λ_1 and λ_3. We found that, in this way, the optimization was easier to tune, for example, making choice of the λ parameters and of the learning rate independent of the batch size.

Least-squares Fitting: Based on the target keypoints and the corresponding keypoint predictions, we use least-squares fitting [18] to compute the rotation $\boldsymbol{R}$ and the translation $\boldsymbol{t}$ of each object following [7].

The least-squares fitting algorithm [18] determines rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ of the 6D pose by minimizing the following squared loss as in [7]

$$L_{\text{Least-squares}} = \sum_{i=1}^{M} \left\| \widehat{\boldsymbol{k}}_{\boldsymbol{i}} - (\boldsymbol{R}\boldsymbol{k}_i + \boldsymbol{t}) \right\|_2^2, \quad (2)$$

where $M = 8$ is the number of keypoints per object, $\boldsymbol{k}_i$ are the target keypoints in the object coordinate system, and $\widehat{\boldsymbol{k}}_{\boldsymbol{i}}$ are the predicted keypoints in the camera coordinate system.

IV. Experimental Setup

A. Datasets

The most common datasets for 6D pose estimation are YCB-Video [8], LineMOD [19], Occlusion LineMOD [61], and T-LESS [20]. Unfortunately, all of these datasets provide annotated training data for only a few different scenes, e.g. 13 scenes in LineMOD, 80 scenes in YCB-Video, and just single-object training data in T-LESS. Besides, YCB-Video and LineMOD do not show a high amount of clutter and the scenes are only shown from similar perspectives instead of clearly distinct views. Therefore, these datasets are not suitable to train and evaluate our proposed method which is designed for very distinct views in heavily cluttered scenes.

Novel Photorealistic Datasets: Due to the drawbacks of the previously mentioned datasets, we created three novel photorealistic datasets of diverse cluttered scenes with heavy occlusions. All of these datasets contain RGB-D images from multiple very distinct perspectives which are annotated with 6D poses of all cameras and objects as well as ground truth for instance semantic segmentation.

Our datasets are composed of eleven non-symmetric objects from the YCB object set [21]. Many of these objects have a symmetric shape but are non-symmetric due to their texture. This requires the pose estimation approaches to exploit RGB information in order to correctly predict the objects' poses. Using Blender with physics, we created cluttered scenes by spawning the YCB objects above the ground in the center with a 3D normally distributed offset. Due to the similar spawning point of all objects, it is likely that objects rest on

top of each other. Instead of a flat ground plane, we use a shallow bowl for the objects to fall onto. The bowl prevents the objects from scattering in all directions. Consequently, the objects stay close together resulting in heavily occluded scenes.

In addition to the random composition of the scenes, we applied further domain randomization techniques in order to improve the generalization of our approach. For each scene, we selected a random 360×180 degree panorama photo from a set of 379 photos and a different bowl texture. Furthermore, we slightly vary intensity and color of the lighting to create different shadows and reflections.

For our three datasets, we collected RGB-D images from multiple views with different camera settings. Therefore, we call our dataset framework Multi-View YCB (MV-YCB). We generated ground truth for instance semantic segmentation and 6D pose estimation. Additionally, we provide the exact camera poses. All datasets are split into 90% training data and 10% test data.

For the first dataset, called MV-YCB FixCam, we generated 8,333 random scenes and placed three identical cameras at fixed positions equally distributed around the scene, i.e. in a circle in 120 degree intervals. This results in a total of 24,999 RGB-D frames with corresponding ground truth data.

For the second dataset, called MV-YCB WiggleCam, we use the same scenes and cameras as in MV-YCB FixCam, but add a 3D normally distributed offset to each camera independently. This reflects a typical robotic setting where the cameras are mounted slightly inaccurate around the scene.

For the third dataset, called MV-YCB MovingCam, we generated further 8,333 random scenes and placed four cameras around the scene, where each camera spawns in another quadrant in a sphere. This represents a scenario, where a single camera is moved around the scene recording four frames from four very distinct perspectives. This results in a total of 33,332 RGB-D frames with corresponding ground truth data.

B. Data Augmentation

We used a variety of data augmentations to boost the generalisation of our network. During training we followed [7] and applied a random combination of color jitter, sharpening filters, motion blur, Gaussian blur, and Gaussian noise on the RGB images. The depth maps and the corresponding point clouds are not augmented.

C. Training Procedure

For MV-YCB FixCam and MV-YCB WiggleCam, we train with all three camera views and increase the number of training samples by using all relevant camera combinations i.e. {[cam1, cam2, cam3], [cam2, cam3, cam1], [cam3, cam1, cam2]}. We do not need every possible permutation of this list since the exact order is irrelevant. Only the first camera in the list determines the camera coordinate system in which the 6D poses are predicted.

For MV-YCB MovingCam, we use a variable number of camera views. It allows for more flexibility when deploying the trained network as the network learns how to deal with a different number of views. We employ a set with all relevant camera combinations for each view count, e.g. {[cam1, cam2], [cam1, cam3], [cam2, cam3]} for a view count of two if we use a maximum of three cameras. The previous rotation step is then applied on each item.

Each sample in a batch must have the same view count. When training with a variable view count this is ensured by using a custom batch sampler. An equal amount of batches is sampled for each view count. A sample with a lower view count requires less memory and offers less information for network optimization. Therefore, we sample more samples per batch for lower view counts in order to further balance the optimization. Consequently, each batch has roughly the same memory size and information amount.

D. Implementation Details

For the training on MV-YCB FixCam and MV-YCB WiggleCam, we used eight GPUs of type NVIDIA Tesla V100 with 32GB of memory. We randomly sample 12,288 points from each depth map. For the training on MV-YCB MovingCam, we used three NVIDIA GeForce RTX 2080 Ti with 11GB of memory. Here we sample 6,144 points from each depth map. All networks were trained with an Adam optimizer [62] using a cyclical learning rate [63] with the triangular2 policy.

E. Evaluation Metrics

For evaluating our network and comparing it with other approaches, we follow previous works [7], [6], [14], [8] and use the area-under-curve (AUC) metrics for ADD-S and ADD(-S) as well as the percentage of predictions ADD-S and ADD(-S) $<$ 2cm.

The average distance metric ADD [64] is computed by

$$\text{ADD} = \frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{x} \in \mathcal{M}} \left\| (\widehat{\boldsymbol{R}}\boldsymbol{x} + \widehat{\boldsymbol{t}}) - (\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}) \right\|_2, \quad (3)$$

where $\mathcal{M}$ is the set of vertices of a given object mesh. $\boldsymbol{p} = [\boldsymbol{R}|\boldsymbol{t}]$ and $\widehat{\boldsymbol{p}} = [\widehat{\boldsymbol{R}}|\widehat{\boldsymbol{t}}]$ are the ground truth pose and the predicted pose respectively. The average distance is calculated between all corresponding vertices of the ground truth pose and the predicted pose.

The average closest point distance metric ADD-S [64] is a slightly relaxed adaptation of the ADD metric which is more suitable for symmetric objects. It is computed by

$$\text{ADD-S} = \frac{1}{|\mathcal{M}|} \sum_{\boldsymbol{x}_1 \in \mathcal{M}} \min_{\boldsymbol{x}_2 \in \mathcal{M}} \left\| (\widehat{\boldsymbol{R}}\boldsymbol{x}_1 + \widehat{\boldsymbol{t}}) - (\boldsymbol{R}\boldsymbol{x}_2 + \boldsymbol{t}) \right\|_2. \quad (4)$$

Here, the average distance is computed between each vertex of the predicted pose and the closest vertex on the ground truth.

The average (closest point) metric ADD(-S) [7] is a combination of the previous two metrics. If it is applied on a symmetric object, the relaxed ADD-S metric is used. If the object is non-symmetric, the stricter ADD metric is applied.

TABLE I

QUANTITATIVE RESULTS ON MV-YCB FIXCAM

	PVN3D	CosyPose	CosyPose	MV6D
Number of views	1	3	3	3
Known camera poses	✓	×	✓	✓
ADD-S AUC	81.3	90.8	91.9	**96.9**
ADD(-S) AUC	74.9	82.4	84.6	**94.8**
ADD-S < 2cm	82.1	92.9	93.0	**98.8**
ADD(-S) < 2cm	73.0	80.6	82.4	**96.5**

TABLE II

QUANTITATIVE RESULTS ON MV-YCB WIGGLECAM

	PVN3D	CosyPose	CosyPose	MV6D
Number of views	1	3	3	3
Known camera poses	✓	×	✓	✓
ADD-S AUC	80.8	90.0	91.3	**96.2**
ADD(-S) AUC	74.0	81.0	83.4	**93.0**
ADD-S < 2cm	82.0	92.3	92.6	**98.7**
ADD(-S) < 2cm	72.4	78.9	81.6	**96.0**

Based on the values of ADD-S and ADD(-S), we compute the area under the accuracy-threshold curve (AUC) and the percentage that is smaller than 2cm which is a typical threshold for successful robot manipulation.

F. Baseline Methods

We compare our approach with the single-view performance of PVN3D [7] (which was the best 6D pose estimator when we started this project) and the best multi-view 6D pose estimator CosyPose [22]. Originally, CosyPose is based on the single-view approach DeepIM [34], uses only RGB data, and can cope with unknown camera poses. However, CosyPose is already outperformed by PVN3D on LineMOD [19] and YCB-Video [8] as both papers suggest [22], [19]. In order to get a more challenging benchmark than PVN3D, we exchanged the DeepIM in CosyPose with PVN3D in order to create a multi-view 6D pose estimator based on RGB-D data. Since our approach assumes the camera poses to be known, we evaluated CosyPose not only with unknown camera poses as proposed in the paper [22] but also used the known camera poses in a second evaluation instead of their estimations to improve the hypothesis matching and to make the comparison with our approach fairer.

V. RESULTS

A. Quantitative Results on MV-YCB FixCam

Tab. I shows the 6D pose estimation accuracy of our MV6D network using the metrics presented in Sec. IV-E. It is compared with the single-view performance of PVN3D [7] and the multi-view performance of CosyPose [22] as described in Sec. IV-F. The best results are printed in bold.

We can see that MV6D outperforms PVN3D and CosyPose on all metrics significantly. Using the known camera poses instead of estimating them leads to a small improvement on CosyPose but its accuracy stays significantly lower that ours. Even though the MV-YCB FixCam dataset has heavy occlusions, more than 96% of all objects can be predicted within the 2cm robot manipulation threshold.

B. Quantitative Results on MV-YCB WiggleCam

Tab. II presents the results on the MV-YCB WiggleCam dataset where the known camera poses deviate slightly from the actual camera poses.

We see that the inaccurate camera positioning leads to a small accuracy decrease of all approaches. Nevertheless, our multi-view network still outperforms all other methods significantly.

C. Quantitative Results on MV-YCB MovingCam

Tab. III shows the AUC results of PVN3D [7] and MV6D on the MV-YCB MovingCam dataset for each object class individually. In this scenario, we trained MV6D with a varying number of views and evaluated the same model on one to four views.

The high accuracy of MV6D proves that our architecture copes very well with the dynamic camera setup. MV6D outperforms PVN3D vastly even when using just two views and the accuracy further increases with an increasing number of input images. Furthermore, MV6D almost matches PVN3D when inputting just a single view which is naturally not the designated use case of our method.

D. Qualitative Results

Fig. 4 shows qualitative results of our MV6D network on the MV-YCB FixCam dataset in comparison to PVN3D [7], CosyPose [22], and the ground-truth. We can see that our approach predicts the 6D poses of all objects very accurately even though some objects are heavily occluded. As PVN3D gets just the single RGB-D image from the depicted view, it cannot detect some objects at all, such as the tuna fish can and the gelatin box in the first row. CosyPose, which uses the PVN3D predictions on all three views, performs therefore usually better than PVN3D, but for heavily occluded objects, MV6D still outperforms it.

VI. CONCLUSION

We have presented MV6D, a deep learning approach for multi-view 6D pose estimation based on RGB-D data. Given multiple RGB-D recordings of a cluttered scene consisting of multiple known objects, our approach fuses the depth maps into a joint point cloud. The RGB images are processed independently by a CNN before their features are fused with the point cloud features. Based on the fused features, we predict 3D keypoints, 3D center points, and semantic labels, which are used to estimate the objects' 6D poses using a least-squares fitting algorithm.

For examining the strengths of our MV6D network, we created three novel photorealistic datasets with cluttered scenes recorded by multiple RGB-D cameras from different perspectives. Our experiments prove that our multi-view approach outperforms the baseline methods by a large margin even when the camera positions are known only inaccurately. Moreover, we prove that our method masters dynamic camera settings and that its accuracy incrementally rises with an increasing number of input images.

TABLE III

AUC RESULTS ON MV-YCB MOVINGCAM. THE BEST RESULTS ARE PRINTED IN BOLD.

Object classes	PVN3D 1 view		MV6D (variable number of views) 1 view		2 views		3 views		4 views	
	ADD-S	ADD(-S)	ADD-S	ADD(-S)	ADD-S	ADD(-S)	ADD-S	ADD(-S)	ADD-S	ADD(-S)
Banana	80.9	73.0	80.5	73.0	94.3	90.1	96.7	94.2	**97.5**	**95.9**
Cracker box	97.2	96.8	96.9	96.3	97.9	97.7	**98.2**	98.0	**98.2**	**98.1**
Gelatin box	78.1	73.4	76.3	72.2	93.4	90.4	96.5	94.6	**97.7**	**96.4**
Master chef can	94.5	91.5	94.6	91.4	98.1	97.1	**98.2**	**97.7**	**98.2**	**97.7**
Mustard bottle	91.2	87.2	90.2	86.8	97.2	95.5	97.7	96.9	**97.9**	**97.2**
Potted meat can	88.0	84.9	86.9	83.7	97.2	96.1	98.3	97.6	**98.4**	**97.9**
Power drill	94.4	92.6	93.6	91.5	97.1	96.3	97.2	96.8	**97.8**	**97.4**
Pudding box	86.4	82.6	85.4	82.0	95.8	94.3	97.7	96.8	**98.3**	**97.6**
Sugar box	93.1	90.5	91.9	89.5	97.5	96.7	98.0	97.6	**98.1**	**97.8**
Tomato soup can	87.7	83.6	87.3	83.3	97.3	95.4	98.2	97.2	**98.4**	**97.7**
Tuna fish can	78.6	71.6	77.3	70.8	93.1	88.5	97.5	94.5	**97.6**	**95.3**
ALL	88.2	84.4	87.4	83.7	96.3	94.4	97.7	96.5	**98.0**	**97.2**

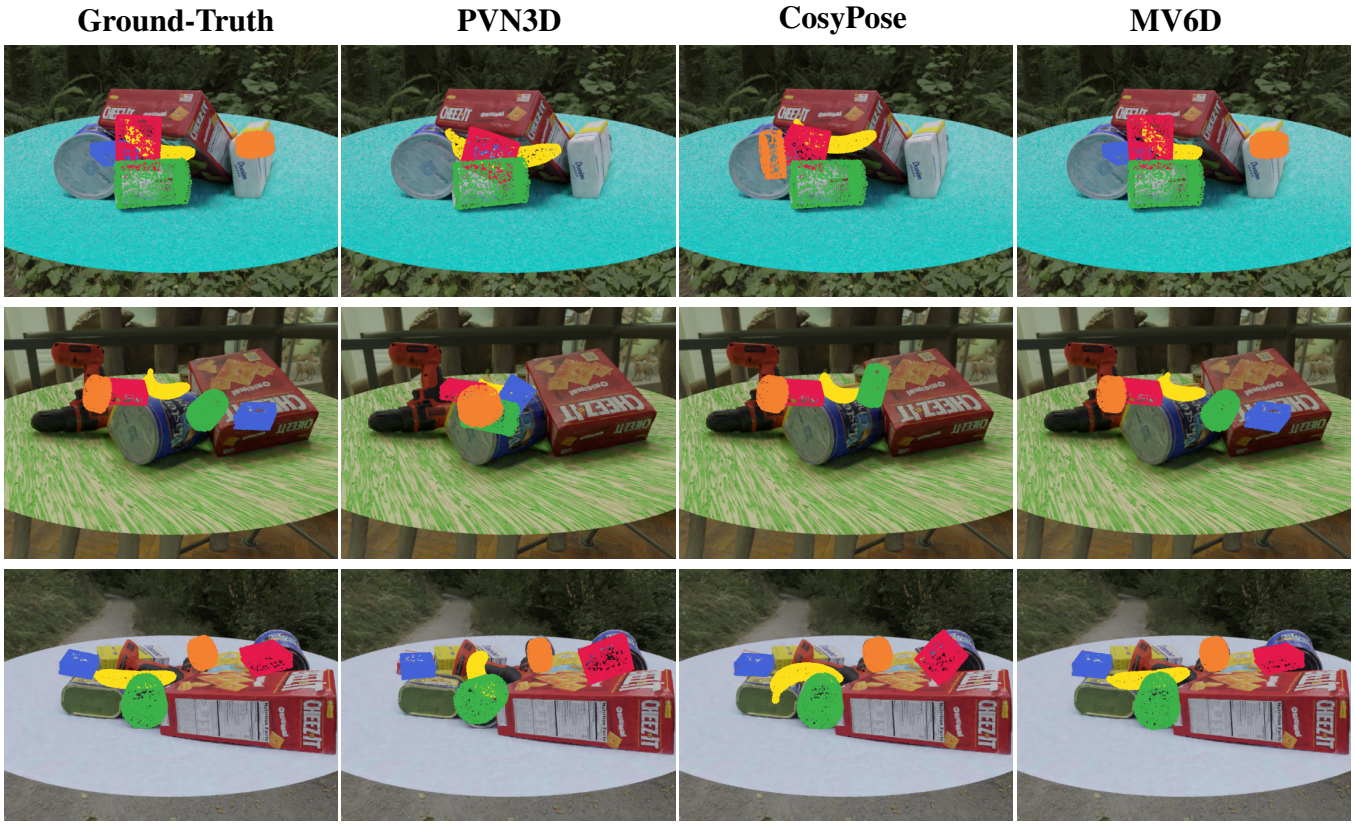

Fig. 4. 6D pose predictions of PVN3D [7], CosyPose [22], and MV6D in comparison to the ground-truth on MV-YCB FixCam. The three rows show three different example scenes that represent the typical performance of the networks. For clarity, only the poses of the five most difficult objects are visualized: tuna fish can (orange), banana (yellow), tomato soup can (green), gelatin box (blue), and pudding box (red).

REFERENCES

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *CVPR*, 2017, pp. 1907–1915.

[2] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *IROS*. IEEE, 2018, pp. 1–8.

[3] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep sensor fusion for 3D bounding box estimation," in *CVPR*, 2018, pp. 244–253.

[4] B. Gu, J. Liu, H. Xiong, T. Li, and Y. Pan, "ECPC-ICP: A 6D vehicle pose estimation method by fusing the roadside lidar point cloud and road feature," *Sensors*, vol. 21, no. 10, p. 3489, 2021.

[5] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *Int. J. Robotics Reseach*, vol. 30, no. 10, pp. 1284–1306, 2011.

[6] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, "FFB6D: A full flow bidirectional fusion network for 6D pose estimation," in *CVPR*, 2021, pp. 3003–3013.

[7] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation," in *CVPR*, 2020, pp. 11 632–11 641.

[8] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *Robotics: Science and Systems*, 2018.

[9] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for

augmented reality: a hands-on survey," *IEEE TVCG*, vol. 22, no. 12, pp. 2633–2651, 2015.
[10] Y. Su, J. Rambach, N. Minaskan, P. Lesur, A. Pagani, and D. Stricker, "Deep multi-state object pose estimation for augmented reality assembly," in *Int. Symp. Mixed Augm. Reality Adjunct*. IEEE, 2019, pp. 222–227.
[11] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DoF object pose from semantic keypoints," in *ICRA*. IEEE, 2017, pp. 2011–2018.
[12] F. Madrigal and F. Lerasle, "Robust head pose estimation based on key frames for human-machine interaction," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–19, 2020.
[13] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise voting network for 6DoF pose estimation," in *CVPR*, 2019, pp. 4561–4570.
[14] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "DenseFusion: 6D object pose estimation by iterative dense fusion," in *CVPR*, 2019, pp. 3343–3352.
[15] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *CVPR*, 2018, pp. 4490–4499.
[16] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.
[17] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *CVPR*, 2019, pp. 12 697–12 705.
[18] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE TPAMI*, no. 5, pp. 698–700, 1987.
[19] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *ICCV*, 2011, pp. 858–865.
[20] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *WACV*, 2017.
[21] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *ICAR*. IEEE, 2015, pp. 510–517.
[22] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "CosyPose: Consistent multi-view multi-object 6D pose estimation," in *ECCV*. Springer, 2020, pp. 574–591.
[23] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, vol. 2, 1999, pp. 1150–1157.
[24] ——, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[25] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *ECCV*. Springer, 2006, pp. 430–443.
[26] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *ECCV*. Springer, 2006, pp. 404–417.
[27] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *IJCV*, vol. 66, no. 3, pp. 231–259, 2006.
[28] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *ICRA*. IEEE, 2009, pp. 48–55.
[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[30] C. Gu and X. Ren, "Discriminative mixture-of-templates for viewpoint classification," in *ECCV*. Springer, 2010, pp. 408–421.
[31] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE TPAMI*, vol. 34, no. 5, pp. 876–888, 2011.
[32] Z. Cao, Y. Sheikh, and N. K. Banerjee, "Real-time scalable 6DOF pose estimation for textureless objects," in *ICRA*. IEEE, 2016, pp. 2441–2448.
[33] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *CVPR*, 2015, pp. 1510–1519.
[34] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6D pose estimation," in *ECCV*, 2018, pp. 683–698.
[35] K. Gupta, L. Petersson, and R. Hartley, "CullNet: Calibrated and pose aware confidence scores for object pose estimation," in *ICCVW*, 2019, pp. 2758–2766.
[36] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *ICCV*, 2017, pp. 1521–1529.
[37] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *CVPR*, 2018, pp. 292–301.
[38] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *ICCV*, 2015, pp. 2686–2694.
[39] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from RGB images," in *ECCV*, 2018, pp. 699–715.
[40] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE TPAMI*, vol. 14, no. 2, pp. 239–256, 1992.
[41] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: 6D pose object detector and refiner," in *ICCV*, 2019, pp. 1941–1950.
[42] J. Chen, L. Zhang, Y. Liu, and C. Xu, "Survey on 6D pose estimation of rigid object," in *Chinese Control Conf.* IEEE, 2020, pp. 7440–7445.
[43] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, "Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy," *Information Fusion*, vol. 68, pp. 161–191, 2021.
[44] S. Song and J. Xiao, "Sliding shapes for 3D object detection in depth images," in *ECCV*, 2014, pp. 634–651.
[45] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *IROS*. IEEE, 2017, pp. 1513–1518.
[46] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *CVPR*, July 2017, pp. 652–660.
[47] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, vol. 30, 2017, pp. 5099–5108.
[48] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3D object detection in point clouds," in *CVPR*, 2019, pp. 9277–9286.
[49] Q. Xie, Y.-K. Lai, J. Wu, Z. Wang, D. Lu, M. Wei, and J. Wang, "VENet: Voting enhancement network for 3D object detection," in *CVPR*, 2021, pp. 3712–3721.
[50] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *CVPR*, 2020, pp. 11 108–11 117.
[51] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge," in *ICRA*. IEEE, 2017, pp. 1386–1383.
[52] J. Sock, S. Hamidreza Kasaei, L. Seabra Lopes, and T.-K. Kim, "Multi-view 6D object pose estimation and camera motion planning using RGBD images," in *ICCVW*, 2017, pp. 2228–2235.
[53] C. Li, J. Bai, and G. D. Hager, "A unified framework for multi-view multi-class object pose estimation," in *ECCV*, 2018, pp. 254–269.
[54] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice*. Springer, 2000, pp. 298–372.
[55] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017, pp. 2881–2890.
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[57] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.
[58] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE TPAMI*, vol. 17, no. 8, pp. 790–799, 1995.
[59] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE TIP*, vol. 6, no. 9, pp. 1305–1315, 1997.
[60] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, Oct. 2017, pp. 2999–3007.
[61] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *ECCV*. Springer, 2014, pp. 536–551.
[62] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *ICLR*, May 2015.
[63] L. N. Smith, "Cyclical learning rates for training neural networks," in *WACV*. IEEE, 2017, pp. 464–472.
[64] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV*. Springer, 2012, pp. 548–562.