Real-time Monocular Object Instance 6D Pose Estimation

Thanh-Toan Do[1]
thanh-toan.do@adelaide.edu.au
Trung Pham[2]
trungp@nvidia.com
Ming Cai[1]
ming.cai@adelaide.edu.au
Ian Reid[1]
ian.reid@adelaide.edu.au

[1] Australian Centre for Robotic Vision
University of Adelaide, Australia
[2] NVIDIA

Abstract

In this work, we present, LieNet, a novel deep learning framework that simultaneously detects, segments multiple object instances, and estimates their 6D poses from a single RGB image without requiring additional post-processing. Our system is accurate and fast (∼10 fps), which is well suited for real-time applications. In particular, LieNet detects and segments object instances in the image analogous to modern instance segmentation networks such as Mask R-CNN, but contains a novel additional sub-network for 6D pose estimation. LieNet estimates the rotation matrix of an object by regressing a Lie algebra based rotation representation, and estimates the translation vector by predicting the distance of the object to the camera center. The experiments on two standard pose benchmarking datasets show that LieNet greatly outperforms other recent CNN based pose prediction methods when they are used with monocular images and without post-refinements.

1 Introduction

Detecting objects and computing their 6D poses (3D locations and orientations) are crucial for many real-world applications including robotics (i.e., object manipulations), augmented reality, to name a few. In cluttered environments, object instances must be first localised, then their poses can be estimated. While the 2D object detection task has gained significant improvement thanks to the power of deep learning, estimating object poses in 3D remains a challenging problem.

Traditional methods match feature points between 2D images and the corresponding 3D models to estimate object 6D poses [1, 17, 18, 28]. However, feature matching is often not robust to poorly textured objects, leading poor pose estimations. Template-matching techniques [9, 12, 22, 26] are more robust, but highly sensitive to illuminations and occlusions. Pose estimation accuracies can be greatly improved by taking advantage of available depth information. However, depth sensors also admit several limitations such as power hungry, a limited working range, and less precise in outdoor environments.

Inspired by recent works [13, 20, 27], in this work we aim at addressing the above limitations by learning a deep network to predict object poses directly from RGB images. Unlike previous methods, which either return coarse object poses or only predict 2D projections of the 3D bounding box vertices, our proposed network directly outputs object poses from the images without a need of a posterior pose refinement or pose optimisation. As a result, our system is more elegant and efficient.

In this paper, we propose a novel deep CNN network, called LieNet, that takes an image input, and directly outputs object detections (represented by bounding boxes, labels, and segmentation masks) together with their 6D poses. Recently, Mask RCNN [8] has shown impressive performances on the instance detection and segmentation tasks. LieNet goes beyond Mask RCNN [8] by adding a new branch for pose estimation. The pose branch is in parallel with the object classification, bounding box regression, and mask prediction branches. Furthermore, the layers of LieNet is redesigned and optimised for real-time performances. An overview of LieNet is shown in Figure 1.

Our pose branch predicts poses by estimating translation and rotation parameters separately. The translation of an object can be estimated by simply predicting its distance to the camera center. Care must be taken when regressing the 3D rotation matrix as not all 3×3 matrices are valid rotation matrices. Fortunately, the Lie algebra representation of the 3D rotation matrix group parameterises a rotation with only three scalar values. Such a representation is unconstrainted and not over-parameterised, thus well suited for regression with deep learning. Although using Lie algebra to represent rotation has been widely used in robotics and vision problems [1, 23, 30], to our best knowledge, this is the first work which successfully applies the Lie algebra representation for deep learning-based object 6D pose estimation.

LieNet is very simple and easy to train in an end-to-end fashion, and does not require an expensive pose refinement post-process. LieNet allows fast inference at about 100ms per frame on a GPU. Evaluated on two standard pose benchmarking datasets, LieNet surpasses all the state-of-the-art RGB pose estimation methods that are used without post-refinements. LieNet is even better or on par with some methods that really depend on post pose refinement steps to boost their accuracies. The remainder of the paper is organized as follows: Section 2 discusses related work. Sections 3 and 4 present the proposed LieNet and experimental results, respectively. Section 5 concludes the paper.

2 Related Work

In this section, we review existing 6D object pose estimation methods ranging from traditional feature matching to modern deep learning based methods.

Traditional approaches. Early object pose estimation methods [7, 17, 18, 25, 28] are based on matching sparse feature points between 2D images and 3D object models. These methods, though simple and fast, do not work reliably with poorly textured objects. In such cases, templates matching methods [9, 10, 12, 22, 26] often work better. For instance, LINEMOD [9] used stable gradient and normal features for template matching. LINEMOD is, however, designed to work with RGB-D images, unlike our method which only needs RGB images. Moreover, template-based approaches are sensitive to the lighting and occlusion.

Feature learning approaches. Recent 6D pose estimation methods have relied on feature learning for dealing with poorly textured objects [4, 14, 15, 19]. In general, these methods compose of multiple stages. For example, in [4], a regression forest is trained to predict

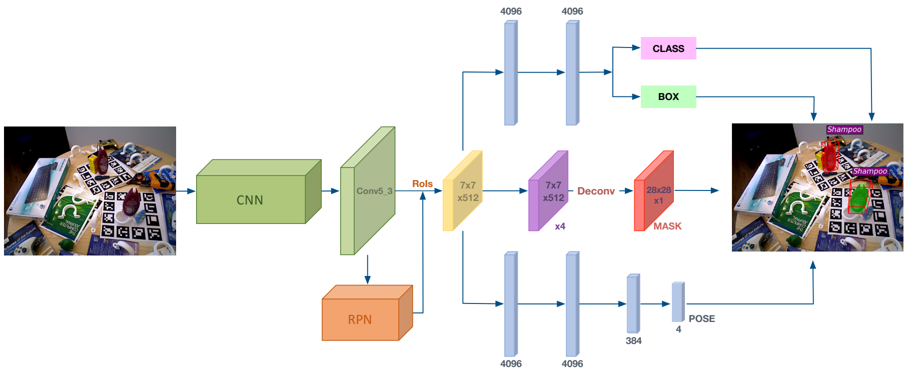

Figure 1: An overview of LieNet. LieNet takes a RGB image as input. A deep CNN backbone (i.e., a VGG net) is used to compute image features. A region proposal network (RPN) (attached to the last convolutional layer of VGG) predicts a number of regions of interest (RoIs). These RoIs will be passed to four independent branches for bounding box regression, bounding box classification, mask segmentation, and pose estimation. In the right most image, different object instances, detected by LieNet, are shown in different colors, and their predicted 6D poses are visualised using 3D boxes.

dense pixel semantic labels and pixel coordinates (with respect to object coordinate systems). A set of pose hypotheses is generated by using the predicted pixel labels, coordinates, and the input depth. Finally, an energy function is defined to evaluate and select the best pose hypothesis. Note that these works heavily depend on depth information. The work in [5] also follows a multi-stage approach as [4, 14] but is designed to work with RGB inputs. In order to deal with the missing depth information, the distribution of object coordinates is approximated as a mixture model when generating pose hypotheses.

CNN-based approaches. In recent years, deep learning based methods [13, 14, 15, 20, 27, 29] have been proposed to address the limitations of the traditional object pose estimation methods. Similar to the template-matching based methods, SSD-6D [13], based on a CNN, casts the object pose estimation as a pose classification problem by discretizing the 3D rotation space into a fixed number of "template" views. However, under-sampling the rotation space might lead to poor pose predictions, while denser-sampling will significantly increase computational cost. Furthermore, this method does not directly output translations — instead inferred from the predicted rotations and the detected 2D bounding boxes. In contrast, our framework LieNet directly regresses object poses from a single RGB image. LieNet outputs translation and rotation components independently, where the rotation component is parameterized using a Lie algebra representation. Recently, in [29], the authors have proposed a CNN-based approach to directly regresses 6D object poses, but they use quaternion representation for rotations. Unlike the Lie algebra, the 4-d unit quaternion representation is over parameterized, and normalization of the network output often results in worse performance.

Instead of directy regressing object poses, other recent methods [20, 27] train deep networks to predict 2D projections of 3D bounding box vertices, which are then used to infer object poses using a PnP algorithm. These methods often compose of a cascade of multiple CNNs for object localisation, predicting of box vertices, and pose refinement, are thus time-consuming for inference. In contrast, LieNet is end-to-end and runs in real-time.

3 LieNet

Our goal is to design an efficient end-to-end network that is capable of estimating 6D poses of multiple object instances in a RGB image. In order to estimate their poses, objects must be correctly detected (and segmented) from the image. Inspired by the impressive results of Mask R-CNN [8] for the object instance segmentation task, LieNet extends Mask R-CNN to simultaneously detect, segment, and estimate 6D poses of object instances presenting in the image. Besides the object detection and segmentation branches, LieNet contains a novel pose estimation branch composing of a few fully connected layers. For each detection, the pose branch, outputs four (real) values, where the first three elements represent the Lie algebra of the rotation matrix, and the last element represents the z component of the translation vector. Given the predicted z and the predicted bounding box from the box regression branch, we use projective geometry to recover the full translation vector.

3.1 Network Architecture

Figure 1 presents the schematic overview of LieNet. We differentiate two parts of the network: a backbone and head branches. The backbone is mainly used for extracting image features and shared between head branches. There are four head branches corresponding to the four different tasks including bounding box regression, bounding box classification, mask segmentation, and the 6D pose estimation. To ensure high acuracy and low latency, the backbone is adopted from the VGG network [24], which is followed by a region proposal network (RPN) similar to Faster RCNN [21]. For each region of interest (RoI) returned by RPN, a fixed-size 7×7 feature map is pooled from the *conv*5_3 feature map using the RoIAlign layer [8], which is then passed to head branches. Our bounding box regression, classification head branches follow exactly as Mask R-CNN [8], while our segmentation head branch is slightly different. We use four 3×3 consecutive convolutional layers (denoted as '$\times 4$' in Figure 1). A ReLu layer is inserted after each convolutional layer. A deconvolutional layer is used to upsample the feature map to 28×28 which becomes the segmentation prediction.

Our proposed pose branch is a small multilayer perceptron (MPL) composing of 4 fully connected layers whose sizes (number of nodes) are 4096, 4096, 384, and 4 respectively, where the last 4 nodes represent 6D poses (see Section 3.2). A ReLU layer is inserted after each fully connected layer, except the last one. We found that the above LieNet's architecture balances well the trade-off between accuracy and efficiency.

3.2 Pose Regressor

One of our main contributions is a novel pose branch that regresses 6D object poses directly from monocular images. Object poses are often decomposed into translation and rotation components respectively. Similarly, our pose branch predicts object translations and rotations separately.

Translation Regression Instead of regressing a full translation vector $t = [t_x, t_y, t_z]$, our network is trained to regress the t_z component only. The reason is that a 3D object model, when projected into a 2D image using two different translation vectors with the same t_z, may produce two object images with similar appearances and scales (at different positions in the image). This causes difficulty for the network to predict the x and y components by using only appearance information as input. Given a detection and its depth t_z component, its full translation vector can be recovered easily as follow:

$$t_x = \frac{(u - c_x)t_z}{f_x}, \quad t_y = \frac{(v - c_x)t_z}{f_y} \tag{1}$$

where$[u, v]$ is the bounding box center, and the matrix $[f_x, 0, c_x; 0, f_y, c_y; 0, 0, 1]$ is the camera intrinsic calibration matrix. The formulation (1) assumes that the object center in 3D will be projected to the object bounding box center in the 2D image.

Rotation Regression While regressing translations is quite straightforward, regressing rotations is more tricky. The Euler angles can describe any rotation using three angles, but wrap around at 2π radians, i.e., multiple values representing the same angle. This causes difficulty in learning a uni-modal scalar regression task. Furthermore, the Euler angles-based representation suffers from the problem of gimbal lock [3]. A rotation can also be presented by a 3×3 orthogonal matrix, which is unfortunately over-parametrised, and enforcing the orthogonality constraint during training is non-trivial. Recent works (e.g., [29]) represent rotations using unit length 4-dimensional quaternions. Such a representation is over-parameterised and the unit length constraint often results in worse performance.

In this work, we use the Lie algebra $so(3)$ associated with the Lie group $SO(3)$ ($3{\times}3$ rotation matrices are members of $SO(3)$) as our rotation representation. The Lie algebra $so(3)$ is known as the tangent space at the identity element of the Lie group $SO(3)$. An arbitrary element of $so(3)$ (parameterized by a vector in $\mathbb{R}^3$) admits a skew-symmetric matrix representation. Technically, any 3-dimensional vector can be easily mapped to a rotation matrix using the closed-form Rodrigues logarithm mapping formulation [2]. Effectively, our pose network needs to regress only three scalar numbers for a rotation, without any constraints. Such a representation is well suited for regression with deep learning. During training, we map ground-truth rotation matrices to their associated elements in $so(3)$, which are then used as regression targets for learning.

3.3 Multi-task loss function

In order to train the network, we define a multi-task loss to jointly optimise bounding box classification, bounding box regression, mask segmentation, and 6D object pose estimation. Formally, the loss function is defined as follows

$$L = \alpha_1 L_{cls} + \alpha_2 L_{box} + \alpha_3 L_{mask} + \alpha_4 L_{pose} \tag{2}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are weighting parameters controlling the importance of each loss. As in Mask RCNN [8], the classification loss L_{cls}, the bounding box regression loss L_{box}, and the segmentation loss L_{mask} are *softmax* loss, *smooth L1* loss, and *binary cross entropy* loss, respectively. Details of these losses can be found in [8]. The pose regression loss L_{pose} is defined as follows

$$L_{pose} = \|r - \hat{r}\|_p + \beta \|t_z - \hat{t}_z\|_p \tag{3}$$

where r and $\hat{r}$ are two 3-dimensional vectors representing regressed and ground-truth rotations; t_z and $\hat{t}_z$ are scalars representing regressed and ground-truth translations (z component only); p is a distance norm; β balances rotation and translation regression errors.

3.4 Training and inference

Training We implement LieNet using Caffe deep learning library [11]. The input to our network is a RGB image with the size 480×640. The RPN outputs RoIs at different sizes and shapes. We use 5 scales and 3 aspect ratios, resulting 15 anchors in the RPN. The 5 scales are 16×16, 32×32, 64×64, 128×128 and 256×256; the 3 aspect ratios are $2:1$, $1:1$, $1:2$. This design allows the network to detect small objects.

The $\alpha_1, \alpha_2, \alpha_3$, and α_4 in (2) are empirically set to 1, 1, 2, 2, respectively. β in (3) is empirically set to 1.5. An important choice for the pose loss (3) is the regression norm p. Typically, deep learning models use $p = 1$ or $p = 2$. We found that $p = 1$ give better results than $p = 2$ for our pose estimation problem.

We train the network in an end-to-end manner using stochastic gradient descent with 0.9 momentum and 0.0005 weight decay. The network is trained on a Titan X GPU for $350k$ iterations. Each mini batch has 1 image. The learning rate is set to 0.001 for the first $150k$ iterations and then decreased by 10 for the remaining iterations. The top 2000 RoIs from RPN (with a ratio of 1:3 of positive to negative) are subsequently used for computing the multi-task loss. A RoI is considered positive if it has an intersection over union (IoU) with a groundtruth box of at least 0.5 and negative otherwise. The losses L_{mask} and L_{pose} are defined for only positive RoIs.

Inference At the test phase, we run a forward pass on the input image. The top 1000 RoIs produced by the RPN are selected and fed into the box regression and classification branches, followed by non-maximum suppression [6]. Among the remained bounding boxes, we select bounding boxes which have classification scores higher than a certain threshold (i.e., 0.9) as the final detections. The segmentation branch and the pose branch are then applied to these detected boxes to output object segmentation masks and their 6D poses.

4 Experiments

We evaluate LieNet on two widely used datasets including the single object pose dataset LINEMOD provided by Hinterstoisser et al. [9] and the multiple object instance pose dataset provided by Tejani et al. [26]. We compare LieNet against the state-of-the-art RGB based 6D object pose estimation methods such as BB8 [20], SSD-6D [13], [5], and [27].

Metric: To evaluate the pose estimation accuracy, we use the *standard* metrics used in [5, 20, 27]. The **2D − projection** metric measures pose errors in 2D, in which we project the 3D object model into the image using the groundtruth pose and the estimated pose. The estimated pose is correct if the IoU between two project boxes is higher than 0.5. **5cm 5°** and **ADD** metrics measure pose errors directly in 3D. When **5cm 5°** metric is used, an estimated pose is correct if it is within $5cm$ translational error and 5° angular error of the ground truth pose. When **ADD** metric is used, an estimated pose is correct if the average distance between transformed model point clouds by the groundtruth pose and the estimated pose is smaller than 10% of the object's diameter. We also evaluate detection and segmentation results. A detection / segmentation is true-positive if its IoU with the groundtruth box / segmentation mask is higher than a threshold.

	Ape	Bvise	Cam	Can	Cat	Driller	Duck	Box	Glue	Holep	Iron	Lamp	Phone	Average
LieNet Det. $F_1^{0.5}$	99.8	100	99.7	100	99.5	100	99.8	99.5	99.2	99.0	100	99.8	100	99.7
LieNet Seg. $F_1^{0.5}$	99.5	99.8	99.7	100	99.1	100	99.4	99.5	99.0	98.6	99.2	99.4	99.7	99.4
LieNet Det. $F_1^{0.9}$	85.4	91.7	93.3	93.6	89.3	87.5	86.3	94.2	81.1	93.2	92.5	91.3	90.8	90.0
LieNet Seg. $F_1^{0.9}$	80.6	57.0	91.4	62.5	52.1	74.6	81.2	91.9	73.3	84.6	90.3	85.0	84.6	77.6
2D-projection metric														
LieNet	99.8	100	99.7	100	99.2	100	99.8	99.0	97.1	98.0	99.7	99.8	99.1	99.3
Tekin *et al.* [27]	99.8	99.9	100	99.8	99.9	100	100	99.9	99.8	99.9	100	100	100	**99.9**
Brachmann [6] (*)	98.2	97.9	96.9	97.9	98.0	98.6	97.4	98.4	96.6	95.2	99.2	97.1	96.0	97.5
SSD-6D [13] (*)	99.0	100	99.0	100	99.0	99.0	98.0	99.0	98.0	99.0	99.0	99.0	100	99.1
5cm 5° metric														
LieNet	57.8	72.9	75.6	70.1	70.3	72.9	67.1	68.4	64.6	70.4	60.7	70.9	69.7	**68.5**
Brachmann [6] (*)	34.4	40.6	30.5	48.4	34.6	54.5	22.0	57.1	23.6	47.3	58.7	49.3	26.8	40.6
BB8 [20] (*)	80.2	81.5	60.0	76.8	79.9	69.6	53.2	81.3	54.0	73.1	61.1	67.5	58.6	69.0
ADD metric														
LieNet	38.8	71.2	52.5	86.1	66.2	82.3	32.5	79.4	63.7	56.4	65.1	89.4	65.0	**65.2**
Brachmann [6]	-	-	-	-	-	-	-	-	-	-	-	-	-	32.2
BB8 [20]	27.9	62.0	40.1	48.1	45.2	58.6	32.8	40.0	27.0	42.4	67.0	39.9	35.2	43.6
SSD-6D [13]	-	-	-	-	-	-	-	-	-	-	-	-	-	2.42
Tekin *et al.* [27]	21.6	81.8	36.6	68.8	41.8	63.5	27.2	69.6	80.0	42.6	75.0	71.1	47.7	55.9
Brachmann [6] (*)	33.2	64.8	38.4	62.9	42.7	61.9	30.2	49.9	31.2	52.8	80.0	67.0	38.1	50.2
BB8 [20] (*)	40.4	91.8	55.7	64.1	62.6	74.4	44.3	57.8	41.2	67.2	84.7	76.5	54.0	62.7
SSD-6D [13] (*)	-	-	-	-	-	-	-	-	-	-	-	-	-	76.3

Table 1: Object detection and pose estimation accuracy on the LINEMOD dataset [9] for single object. F_1^a is F_1 score at IoU a. (*) indicates methods used with post-refinements. The result of SSD-6D [13] without post-refinements under ADD metric is cited from [27].

4.1 Single object pose estimation

In [9], the authors published a RGBD dataset named LINEMOD, which has become a standard benchmark for 6D pose estimation. The dataset contains 15 sequences with poorly textured objects in cluttered scenes. Following [6, 20], we only use RGB images for training and testing. The images in each sequence contain multiple objects, however, only one object is annotated with the ground-truth class label, bounding box, and 6D pose. The dataset also provides camera intrinsic matrix. Given 3D object models with ground-truth 6D poses and the camera matrix, we are able to compute the ground-truth object segmentation masks.

We randomly select 30% of the images from each sequence for training and validation. The remaining images serve as the test set. A complication when using this dataset for training arises because not all objects in each image are annotated, i.e., only one object instance is annotated per sequence, even though multiple object instances are present. In other words, one object, annotated as foreground in one sequence, is annotated as background in other sequences. Such annotation issue may make the network training difficult to converge properly. To overcome this problem, for each object sequence, we use the RefineNet [16], a state-of-the-art semantic segmentation algorithm, to train a semantic segmentation model. The trained model is applied on all training images in other sequences. The predicted masks in other sequences are then filtered out, so that the presences of objects without annotated information does not hinder the training.

Results: As done in previous works [6, 13, 20], we evaluate LieNet on 13 object sequences for which 3D models are available. We first study the object detection and segmentation performances of LieNet. Top rows in the Table 1 report the results. At an IoU 0.5, LieNet achieves impressive accuracies with F1 scores greater than 99% for all object categories. Even at an IoU 0.9, although the accuracies decrease, LieNet still performs quite well with average detection F1 score 90%.

Next we evaluate pose estimation performance of LieNet, which is the main focus of this

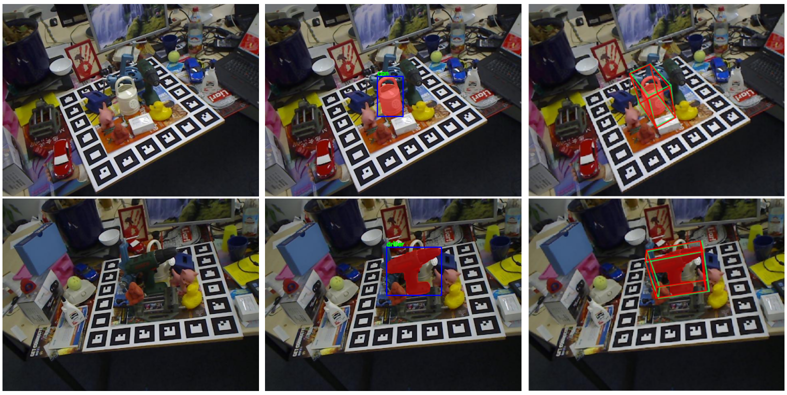

Figure 2: Some qualitative results of LieNet on the LINEMOD dataset [9]. Left: original images; Middle: predicted 2D boxes, classes, and segmentations; Right: green and red boxes are the groundtruth poses and predicted poses, respectively.

work. Table 1 reports the comparative pose estimation accuracies between LieNet and the state-of-the-art works including Brachmann et al. [5], BB8 [20], SSD-6D [13] and Tekin *et al.* [27]. All the methods only use RGB images as inputs to predict the poses. Note that except LieNet and [27], other methods comprise of multiple-stages including a 2D object detection, an initial pose estimation, and a pose refinement.

It can be seen that LieNet significantly outperforms all the considered competitors when they are used without a post-refinement under all evaluation metrics. The improvements are more significant when the errors are computed using the estimated poses directly (i.e., **ADD** and **5cm 5**°), and less significant when evaluated in 2D using IoU. Even when the competitor methods such as BB8 and SSD-6D further refine their estimated poses using a post-refinement step, LieNet is still very competitive. Note that the post-refinement cost is often expensive, for instance the method in [5] takes about 100ms per object. Figure 2 shows some qualitative results of LieNet for single object pose estimation on the LINEMOD dataset.

4.2 Multiple object instance pose estimation

We use the dataset published by Tejani *et al.* [26], which consists of six object sequences in which images in each sequence contain multiple instances of the same object class with different levels of occlusion. This dataset is considered more challenging than the LINEMOD dataset. The experimental settings are the same of the previous one (see Section 4.1).

Results: The pose estimation accuracies are reported in Table 2. We note that except SSD-6D [13], none of the previous RGB based pose object estimation works report their results using this data. Also SSD-6D [13] only reports their pose accuracies using the **2D − projection** metric. No results with **ADD** metric is reported. It can be seen from the Table 2 that LieNet performs better than SSD-6D even when SSD-6D uses a post-refinement. Furthermore, under more tricky **5cm 5**° and **ADD** metrics, LieNet still achieves impressive results with mean accuracies 64.5% and 62.0%, respectively.

Nonetheless, we found that LieNet is not very robust to rotationally symmetric objects such as Coffee mug. That is because such rotationally symmetric object looks the same after any rotations around Yaw axis. The network is not able to handle such confusions by using only appearance information. Figure 3 shows some qualitative results, where LieNet successfully detects and segments multiple object instances, as well as estimates their 6D poses. The last row of Figure 3 demonstrates the Coffee mug case where the object rotations are wrongly estimated. Nevertheless, the estimated poses are still useful, e.g., for picking task in robotic applications.

	Metric	Camera	Coffee	Joystick	Juice	Milk	Shampoo	Average
LieNet	**2D − projection**	99.2	100	99.6	98.4	99.5	99.1	**99.3**
SSD-6D [13]	**2D − projection**	-	-	-	-	-	-	98.8
LieNet	**5cm 5°**	76.5	18.7	60.2	85.6	73.5	72.4	64.5
LieNet	**ADD**	80.4	35.4	27.5	81.2	71.6	75.8	62.0

Table 2: Quantitative results on the dataset of Tejani *et al.* [26] for multiple object instances.

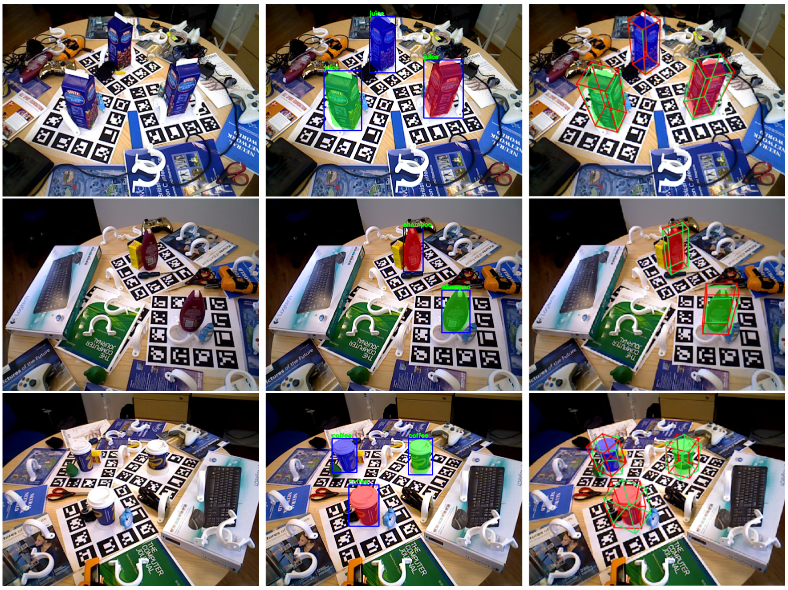

Figure 3: Some qualitative results of LieNet on the multiple object instance dataset of Tejani et al. [26]. Left: original images; Middle: predicted 2D boxes, classes, and segmentations; Right: green and red boxes are the groundtruth poses and predicted poses respectively.

Timing: When runing on a Titan X GPU, LieNet takes approximately 100ms to process one image, which is several times faster than BB8 [20] (300ms) and Brachman *et al.* [5] (450ms), and comparable with SSD-6D [13]. However, these methods report their running times using the LINEMODE dataset [9], which contains only one object instance in each image. Due to the post-refinement, their computational cost will increase rapidly when tested on images with multiple object instances such as the Tejani's dataset [26]. In contrast, the running time of LieNet stays almost the same regardless of the number of object instances.

5 Conclusion

In this paper, we propose LieNet, a deep learning approach for jointly detecting, segmenting, and importantly recovering 6D poses of object instances from a single RGB image. LieNet is end-to-end trainable and directly outputs 6D poses without any post-refinements. LieNet's novel pose head branch uses the Lie algebra based rotation representation, which is well suited for deep regression. LieNet outperforms the state-of-the-art RGB-based 6D object pose estimation methods when they are all used without post-refinements. Furthermore, LieNet also allows a fast inference which is around 10 fps. An interesting future work is to improve the network for handling with rotationally symmetric objects.

References

[1] Motilal Agrawal. A Lie algebraic approach for consistent pose registration for general euclidean motion. In *IROS*, 2006.

[2] C. Altafini. The De Casteljau algorithm on SE(3). In *Nonlinear control in the Year 2000*, 2000.

[3] S. L. Altmann. *Rotations, quaternions, and double groups*. Courier Corporation, 2005.

[4] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014.

[5] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016.

[6] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015.

[7] Iryna Gordon and David G. Lowe. What and where: 3D object recognition with accurate pose. In *Toward Category-Level Object Recognition*, 2006.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.

[9] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *ACCV*, 2012.

[10] Tomas Hodan, Xenophon Zabulis, Manolis I. A. Lourakis, Stepán Obdrzálek, and Jiri Matas. Detection and fine 3d pose estimation of texture-less objects in RGB-D images. In *IROS*, 2015.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[12] Wadim Kehl, Federico Tombari, Nassir Navab, Slobodan Ilic, and Vincent Lepetit. Hashmod: A hashing method for scalable 3d object detection. In *BMVC*, 2015.

[13] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: making rgb-based 3D detection and 6D pose estimation great again. In *ICCV*, 2017.

[14] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *ICCV*, 2015.

[15] Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. PoseAgent: Budget-constrained 6D object pose estimation via reinforcement learning. In *CVPR*, 2017.

[16] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, pages 91–110, 2004.

[18] Manuel Martinez, Alvaro Collet, and Siddhartha S. Srinivasa. MOPED: A scalable and low latency object recognition and pose estimation system. In *ICRA*, 2010.

[19] Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6D object pose estimation. In *CVPR*, 2017.

[20] 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 *ICCV*, 2017.

[21] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[22] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3D object detection: A real time scalable approach. In *ICCV*, 2013.

[23] German Ros, Julio Guerrero, Angel Domingo Sappa, Daniel Ponsa, and Antonio Manuel Lopez. VSLAM pose initialization via Lie groups and Lie algebras optimization. In *ICRA*, 2013.

[24] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, 2014.

[25] Jie Tang, Stephen Miller, Arjun Singh, and Pieter Abbeel. A textured object recognition pipeline for color and depth image data. In *ICRA*, 2012.

[26] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3D object detection and pose estimation. In *ECCV*, 2014.

[27] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6D object pose prediction. In *CVPR*, 2018.

[28] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, 2008.

[29] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *arXiv*, Nov, 2017.

[30] Chi Xu, Lakshmi Narasimhan Govindarajan, Yu Zhang, and Li Cheng. Lie-X: Depth image based articulated object pose estimation, tracking, and action recognition on Lie groups. *IJCV*, pages 454–478, 2017.