Canonical 3D Pose Estimation via Object-Level Classification

Zhen Li
Simon Fraser University
zla247@sfu.ca

Abstract

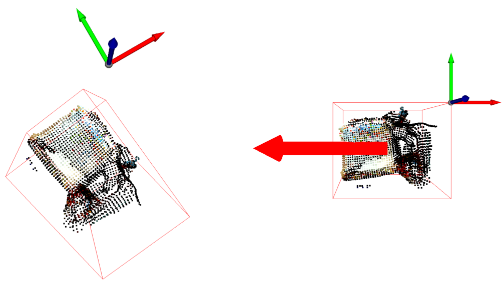
We present a Ca3DPose(Canonical 3D Pose) estimation method via object-level classification and reimplementation of a NPCS (Normalized Part Coordinate Space) based method to object level to do 3D pose estimation. We use the newly released 3D scene dataset Multiscan with over 200 scans. Our model relies on the MinkowskiEngine-powered U-Net backbone or PAConv backbone to get point-level features, voxelized or max pooled point-level features to get object-level features and does object-level classification. We formalize 3D Pose as a combination of the up-direction class, front-direction latitude class, and front-direction longitude class. As a result, we used the results of the NPCS method as a baseline, and our Ca3DPose outperformed the baseline method in the Multiscan dataset. We also found that PAConv backbone outperformed the U-Net backbone. Our source and data are publicly available at `https://github.com/Kaola-2115/MIN3dCaPose`

1 Introduction

3D object pose estimation is an essential task in robotics and 3D scene research. It is widely used as an essential benchmark in newly released 3D scene datasets and part of the segmentation and reconstruction work pipeline. 3D pose estimation is also closely related to the 6D or 9D pose estimation and bounding box predictions. Previously, there are instance-level works based on CAD models (Wang et al., 2019a; Nguyen et al., 2022a), and works based on a single RGB frame or consistent video(He et al., 2021). Nevertheless, those two kinds of pose estimation have limitations. Single RGB-based works usually have low performance. While the CAD model-based works also have limitations due to the preknowledge of CAD models, It's hard to generate them in real word datasets from scanning. GAPartNet also introduced works based on point clouds(Geng et al., 2022). Those works need some preprocesses of data, such as the reconstruction of 3D scenes and object-level annotations or segmentations.

The previous category-level pose estimation works can be classified as point-level regression methods GAPartNet (Geng et al., 2022), pointwise classification methods (Wang et al., 2019a), and methods combining classification and regression model as (Mahendran et al., 2018). While they also have limitations. GAPartNet is only trained on nine strictly defined part classes rather than objects. Wang's work only has nice performance, i.e., mAP within $(5°, 5cm)$ error, i.e., the prediction whose angle between the predicted direction and the real direction is less than $5°$ and the predicted distance and the real distance is less than 5cm is a correct prediction, is greater than 60, on the synthetical dataset with real background images and rendered foreground objects. Their best performance on real word RGB-D images is only 26.7 as mAP within $(5°, 5cm)$ error.

Those previous works and limitations motivate us to design a new object-level classification model architecture. Since Multiscan dataset is newly released with 230 scans of 108 indoor scenes containing 9458 objects and their dataset has an annotation of 3d object pose (Mao et al., 2022), we trained our model on the objects with the articulated part in Multiscan. Figure 1 shows visualizations of annotated data, fig. 1(i) is uncanonicalized chair, and fig.1 (ii) is the canonicalized chair, and red arrow shows annotated front direction. The chair is rotated to normalized space by aligning the front direction to x axis and aligning up direction to z axis. Since there is no baseline method in the new dataset, we re-implemented the NPCS method from GAPartNet and changed the part-level input to object-level as the baseline method. We use two backbones, the voxel based U-Net (Graham et al., 2017) and the point convolution-based PAConv(Xu et al., 2021), and then make comparisons. We use these backbones to get point-level features in both models. The first backbone network is powered by Minkowski Engine (Choy et al., 2019), and part of the implementation is from MINsu3D (Zhang et al., 2022). While the second one implements point convolution using tricks in PAConv.Finally, our model outperforms the baseline model.



(i) un-canonicalized pose (ii) canonicalized pose, red arrow shows front direction

Fig. 1: Annotated 3D Pose Visualization

In summary, the main contributions of this work are:

- A new Ca3DPose model based on the object-level classification to predict canonical 3d-object Pose from point cloud data and a new latitude/longitude/up-direction-based classification method.
- Reimplement an NPCS Pose estimation method and change the part-level model to object-level.
- Compare the voxel-based and point convolution-based backbones in this task.
- Get start-of-the-art 3d pose estimation results on the Multiscan dataset.

2 Related Work

This section reviews the related work on 3D object pose estimation, including point-level regression and classification models. Those works also have input data, such as CAD models, RGB-D images and video frames, and pre-generated point clouds. Some of the works also estimate 6D or 7D(i.e., 3D rotation, 3D translation, and 1D scale) poses beside the basic 3D Pose.

2.1 Instance-Level 3D Pose Estimation

Due to the importance of 3D Pose, there are a lot of instance-level works such as (He et al., 2021; Li et al., 2018; Labbé et al., 2020; Peng et al., 2019; Sundermeyer et al., 2018; Wang et al., 2019a). Instance-level means their works need the preknowledge of the CAD models of the objects. Most works predict the 6D Pose (3D rotation and 3D location) from input RGB-D images, single RGD images, or video frames. Although the start-of-art work achieves high accuracy, those works are still hard to be used in predictions of objects in real work scenes.

2.2 Category-Level 3D Object Detection

Recently, works have been doing a more challenging task, i.e., predicting 3D Pose without knowing the CAD models. Most of the works are used on real word 3D scene datasets. (Wang et al., 2019a) introduced a normalized object coordinate space, (Xiang et al., 2017) introduced the PoseCNN model, (Weng et al., 2021)does pose tracking for live point cloud streams, and (Chen et al., 2020) proposed CASS(Canonical Shape Space) which has some different from NOCS. (He et al., 2021) also introduced an NPCS(Normalized Part coordinate space) when proposing their domain-generalizable object perception. They got the start of the art 7D pose estimation results in their dataset.

Most previous works are based on point-level regression, using their model to predict coordinates per point in canonical space and regression methods to get 3D Pose, such as the (Umeyama, 1991) used by GAPartNet. Some of the works use point-level classification as the last step of prediction. (Wang et al., 2019a) and (Xiang et al., 2017) make the comparison between the classification and regression methods.

Both the point-level classification and regression have limitations. Direct regression has the potential to introduce instability during training, while point classification could introduce more parameters w.r.t. point size and class number, making training unpractical. Therefore, we introduced a new object-level classification method by formalizing 3d Pose as a combination of the up-direction class, front-direction latitude class, and front-direction longitude class. Also, we reimplemented the NPCS method as a baseline. In the Multiscan dataset, our Min3dCaPose method outperforms the baseline method significantly in Ac_5, AC_10, and AC_20.

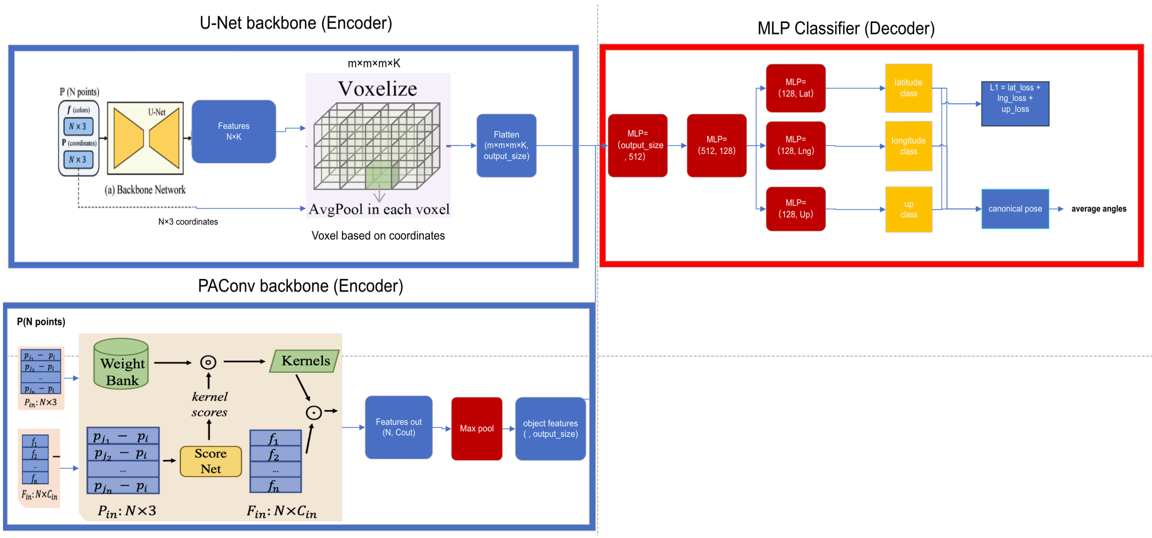


Fig. 2: **Our Ca3DCaPose Method Pipeline** In encoder side, m=30: voxel grid size, the input features is point position(N, 3) + colors(N, 3). In the decoder side Lat: latitude class number, Lng: longitude class number, Up: up class number.

3 Problem Formulation

Given the annotated or predicted object segmentation from a 3D scene, we investigate the problems of object-level 3d pose classification in our Ca3DPose model and the problem of regression in normalized object coordinate space in the reimplementation of the NPCS model.

3D Pose Classification. The input to our system is the point cloud of the object $P \in R^{N\times 6}$, where N denotes the number of points and six dimensions are the three dimension xyz coordinate and three dimension RGB color for each point. To predict the 3D Pose of the object, we try to predict the front direction $F = (x_1, y_1, z_1)$and up direction $UP = (x_2, y_2, z_2)$ in the world coordinate system.

l_a : latitude of the direction, it is computed from the x, y, z axes of the direction as: $l_a = arctan(\frac{z}{\sqrt{x^2+y^2}})$

L_a : formulated latitude class and N_a : the number of latitude classes, we formulate the latitude class L_a of objects as

$L_a = round(N_a \times \frac{l_a+\pi/2}{\pi})$

l_n : longitude of the direction, it is computed from the x and y axes of the direction as: $l_n = arctan(\frac{y}{x})$

L_n : formulated longitude class, and N_n : the number of longitude classes, we formulate the longitude class L_n of objects as

$L_n = round(N_n \times \frac{l_n+\pi}{2\pi})$

Latitude and Longitude class are calculated from the front direction. U : formulated up class and N : the number of up classes, we also formulate the up class of objects as

$U = round(N \times \frac{u+\pi}{2\pi})$, U : up level of the object, and it is computed from the y, z axes of the up direction as:

$u = arctan(\frac{y}{z})$,

Up class are calculated from the up direction; the three classes are independent of each other. Figure 3 illustrates the lng and lat classification given the front direction.

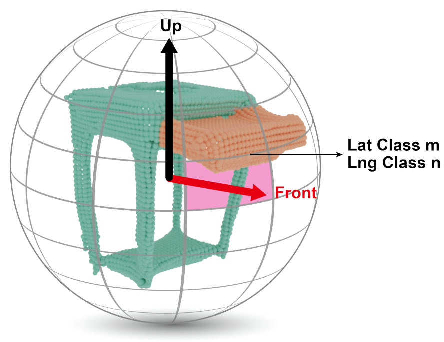


Fig. 3: Latitude and Longitude Class

Regression. Given the object coordinate $P \in R^{N\times 3}$ from coordinate+color input $P \in R^{N\times 6}$, the NPCS system tries to predict the object coordinate$P_{norm} \in R^{N_i\times 3}$ in canonical space, N_i denotes the number of points after removing outliers in RANSAC algorithm. And the 3D rotation $Rotation \in R^3$ is computed. The ground truth 3D rotation is computed from Euler angle rotation matrix $M_R \in R^{3\times3}$ By aligning front and up directions to the x and y axes in world coordinates.

4 Method

4.1 Ca3DPose: Object-level Classification

Architecture Overview. As shown in Figure 2. In U-Net backbone method, following Minsu3d's previous works, our object-level classification-based method leverage the Sparse U-Net as the backbone to produce point-wise feature F (NŒ16). We leverage the U-Net implementation from Minsu3d, which is powered by Minkowski. The backbone network is followed by a 3D 30Œ30Œ30 voxelization model to get a fixed size feature (30Œ30Œ30Œ16), and the empty voxel is assigned zero features to encode the geometry of the 3D object better. The voxelization model is flattened to get an object-level feature. In PAConv backbone method, we modify point convolution following tricks in PAConv and get point level features. We use max pool layers to get object-level features. At the decoder side of the Ca3DPose, 3 MLP classifiers are used to predict lng_class, lat_class, and up_class. And the front direction, up direction, and 3D rotation$Rotation^3$are computed by the three classes.

Loss Function: The pretrained backbone U-Net network uses the same loss function as described in Minsu3d. In our Ca3dPose model, we use 3 standard softmax loss functions for classification L_{lng}, L_{lat}, L_{up}. Then we assigned different weights to 3 classifications to get total $loss : L = W_1 * L_{lng} + W_2 * L_{lat} + W_3 * L_{up}$

4.2 Normalized Object Coordinate Space-based Regression

Architecture Overview . As shown in Figure 4. As the work GAPartNet introduced, their normalized Object Coordinate based method leverage the Sparse U-Net as a backbone to produce point-wise feature $F \in (N \times 16)$. Our reimplementation uses the U-Net implementation from MINSu3d, which is powered by Minkowski Engine. Their work use domain-generalizable object segmentation, while we use annotated object segmentation in Multiscan directly. As NOCS work suggested, the backbone network is followed by three MLPs to get point-wise NOCS regression. Then using RANSAC(Fischler et al., 1981)and Umeyama(Umeyama, 1991), the NOCS method can get the 3D rotation $Rotation \in R^3$and 3D translation$T \in R^3$Different from the original NOCS method, we set the 1D scale as 1 in the Umeyama algorithm rather than getting the 7D pose results. Finally, we use the 3D rotation as the 3D pose estimation results.

Loss Function: The pretrained backbone U-Net network uses the same loss function as described in MINSu3d. In the NOCS regression method, as GAPartNet suggested, we use three loss functions. The point-wise coordinate loss: $L(P, P_{norm}) = \frac{\sum_{(x,y,z)\in N_i}((x-x^*)^2+(y-y^*)^2+(z-z^*)^2)}{N_i}$, the rotation prediction loss: $L_R = sqrt(R, R^*)$, and translation prediction loss: $L_T = sqrt(T, T^*)$

Then we assigned different weights to three kinds of losses to get the total loss:
$L = W_1 * L(P, P_{norm}) + W_2 * L_R + W_3 * L_T$

Symmetry-aware Pose Estimation : Unlike the original NOCS work in GAPartNet, we use the original regression rather than the NPCS regression loss by their work. Therefore, we do not tolerate symmetries for object pose estimation.

5 Experiments and Results

Data Split and Statistics We randomly split the Multiscan dataset into train and test sets by ratio 5:1. Since each scene contains multiple scans, we split dataset by scene level to avoid the same objects appearing in both sets. We remove the objects with the semantic label "-1" and train on all the other objects, and Table 1 shows the objects and scenes number in train and test sets.

Dataset	Scenes	Objects
train	171	2941
test	37	604

Table 1: Dataset Statistics

Evaluataion Metrics: Following the previous 3D pose estimation work in GAPartNet, we compute the average error angel **Rerr** as the sum of the error angles of the front and up directions. we modify the widely-used metrics (5°,5cm), (10°,10cm) to our own metrics AC_5, AC_10 and AC_20 by removing the translation toleration part, e.g., AC_5 is the accuracy of prediction that the error angle is within 5°

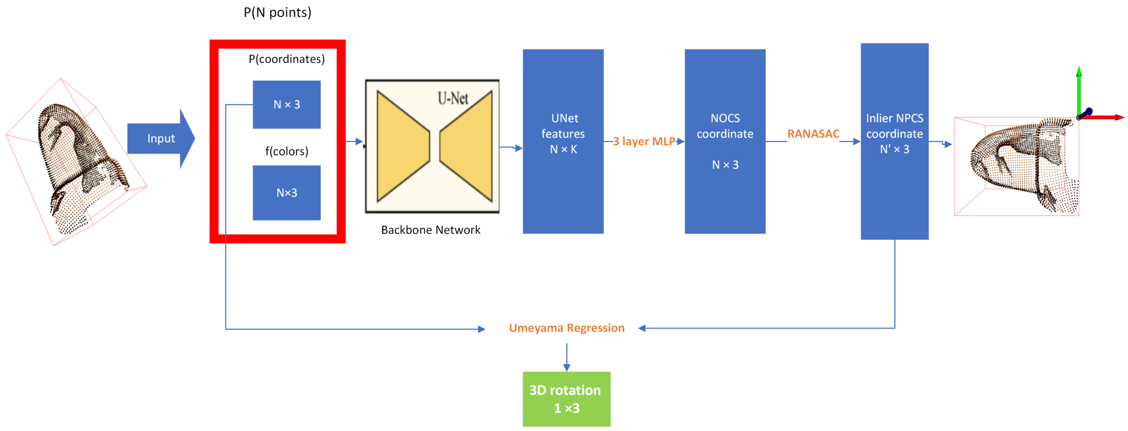


Fig. 4: Re-implemented NOCS Method Pipeline

Network Architecture and Computation Time. Details are shown in appendix sections A and B

Main Results. Fig5 shows the best results of Ca3DPose model using PAConv backbone. Fig6 shows other results as comparisons. It shows that we achieve an AC_5 of 0.631, AC_10 of 0.697 and AC_20 of 0.763, which outperforms the baseline method. Also, the PAConv backbone method outperforms the U-Net backbone method since point convolution keeps more input features compared with voxelization in U-Net.

What	AC_5	AC_10	AC_20	Rerr	Number
wall	0.764	0.809	0.828	0.543	157
door	0.610	0.659	0.683	0.937	41
table	0.517	0.583	0.633	0.849	60
chair	0.529	0.657	0.857	0.236	70
cabinet	0.652	0.710	0.783	0.595	69
window	0.824	0.941	1.000	0.046	17
sofa	0.636	0.727	0.864	0.274	22
microwave	0.500	0.667	0.667	1.061	6
pillow	0.727	0.788	0.939	0.157	33
tv_monitor	0.455	0.500	0.545	1.427	22
curtain	0.591	0.591	0.682	0.791	22
trash_can	0.875	0.875	0.875	0.393	8
suitcase	0.594	0.625	0.688	0.669	32
sink	0.286	0.500	0.500	1.479	14
backpack	0.000	0.250	0.250	1.808	4
bed	0.750	0.750	0.750	0.588	8
refrigerator	0.600	0.600	0.600	0.909	10
toilet	0.333	0.444	0.444	1.052	9
average	0.631	0.697	0.763	0.621	604

Fig. 5: Best Ca3DPose Results on Test Set

Method	AC_5	AC_10	AC_20	Rerr
Ca6DPose (U-Net)	0.328	0.349	0.387	1.812
Ca6DPose (PAConv)	0.631	0.697	0.763	0.621
NOCS (baseline)	0.004	0.040	0.175	1.359

Fig. 6: Comparison Results on Test Set

6 Conclusion

We presented a method for category-level 3D pose estimation based on object-level classification. We also reimplemented a Normalized Object Coordinate Space-based method as a baseline. We train and compare the results in the Multiscan dataset and show that our Ca3DPose models based on both backbones outperform the baseline method. Future work should investigate the performance in another 3D scene dataset.

References

WANG H, SRIDHAR S, HUANG J, et al. Normalized object coordinate space for category-level 6d object pose and size estimation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2019a.

NGUYEN V N, HU Y, XIAO Y, et al. Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2022a: 6771-6780.

HE Y, HUANG H, FAN H, et al. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2021: 3003-3013.

GENG H, XU H, ZHAO C, et al. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts[J/OL]. 2022. https://arxiv.org/abs/2211.05272.

MAHENDRAN S, ALI H, VIDAL R. A mixed classification-regression framework for 3d pose estimation from 2d images[J/OL]. CoRR, 2018, abs/1805.03225. http://arxiv.org/abs/1805.03225.

MAO Y, ZHANG Y, JIANG H, et al. Multiscan: Scalable rgbd scanning for 3d environments with articulated objects[C/OL]//Advances in Neural Information Processing Systems. 2022. https://github.com/smartscenes/multiscan.

GRAHAM B, VAN DER MAATEN L. Submanifold sparse convolutional networks[J/OL]. CoRR, 2017, abs/1706.01307. http://arxiv.org/abs/1706.01307.

XU M, DING R, ZHAO H, et al. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2021: 3173-3182.

CHOY C, GWAK J, SAVARESE S. 4d spatio-temporal convnets: Minkowski convolutional neural networks [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2019: 3075-3084.

ZHANG Y, QIRUIW, GONG Z, et al. Minsu3d [J/OL]. GitHub repository, 2022. https://github.com/3dlg-hcvc/minsu3d.

LI Y, WANG G, JI X, et al. Deepim: Deep iterative matching for 6d pose estimation[C]//Proceedings of the European Conference on Computer Vision (ECCV). [S.l.: s.n.], 2018: 683-698.

LABBÉ Y, CARPENTIER J, AUBRY M, et al. Cosypose: Consistent multi-view multi-object 6d pose estimation[C]//European Conference on Computer Vision. [S.l.]: Springer, 2020: 574-591.

PENG S, LIU Y, HUANG Q, et al. Pvnet: Pixel-wise voting network for 6dof pose estimation[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2019: 4561-4570.

SUNDERMEYER M, MARTON Z C, DURNER M, et al. Implicit 3d orientation learning for 6d object detection from rgb images[C]//Proceedings of the european conference on computer vision (ECCV). [S.l.: s.n.], 2018: 699-715.

XIANG Y, SCHMIDT T, NARAYANAN V, et al. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes[J]. arXiv preprint arXiv:1711.00199, 2017.

WENG Y, WANG H, ZHOU Q, et al. Captra: Category-level pose tracking for rigid and articulated objects from point clouds[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. [S.l.: s.n.], 2021: 13209-13218.

CHEN D, LI J, WANG Z, et al. Learning canonical shape space for category-level 6d object pose and size estimation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. [S.l.: s.n.], 2020: 11973-11982.

UMEYAMA S. Least-squares estimation of transformation parameters between two point patterns[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1991, 13(04):376-380.

FISCHLER M A, BOLLES R C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography[J/OL]. Commun. ACM, 1981, 24(6):381395. https://doi.org/10.1145/358669.358692.

MARTÍNEZ A, SCHMUCK C, PEREVERZYEV JR S, et al. A machine learning framework for customer purchase prediction in the non-contractual setting[J]. European Journal of Operational Research, 2020, 281(3):588-596.

WANG C, XU D, ZHU Y, et al. Densefusion: 6d object pose estimation by iterative dense fusion[C]// Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. [S.l.: s.n.], 2019b: 3343-3352.

NGUYEN V N, HU Y, XIAO Y, et al. Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2022b: 6771-6780.

A Network architecture

Ca3DPose In the U-Net backbone stage, we use the exact same network architecture as MINSu3d. In the voxelization stage, we use a $30 \times 30 \times 30$ voxel grid. In MLP classifier stage, we use three $(n_sizes \times 256), (256 \times lng/lat/up_classes)$linear layers, where N_sizes denotes the flattened object-level features. Table2 shows the parameter size.

Part(type)	Output Shape	Param#
backbone	-	7.5M
fc1(Linear)	(n_size, 256)	56.6M
dropout(Dropout)	(n_size, 256)	0
fc_lat(Linear)	(256, 128)	32.9K
fc_lng(Linear)	(256, 256)	65.8K
fc_up(Linear)	(256, 256)	65.8K

Table 2: Hyper-parameters of Ca3DPose Model

NOCS In U-Net backbone stage, we use the same network architecture as Ca3DPose. Then we use $(16 \times 9), (9 \times 3)$ linear layers in MLP to predict NOCS coordinates in three channels.

B Training Details and Computation Times

- Our network is implemented on Python 3.8 and Pytorch 1.8.2, and our environment is
- CPU: Intel Core i7-12700 @ 2.10-4.90GHz $\times$ 12
- RAM: 32GB
- GPU: NVIDIA GeForce RTX 3090 Ti 24GB
- System: Ubuntu 20.04.2 LTS

It takes us 12hr to train the Ca3DPose model with U-Net backbone, 4hr to train the Ca3DPose model with PAConv backbone and 30hr10min to train the NOCS model; the average inference time per object is 1.004s

C Visualization

We provide visualization of 3D pose estimation, the visualization contains current coordinates, object point cloud, and front direction. The default visualization code shows good results comparison within A_5 and predictions comparison over AC_10.