ISBN: 978-981-18-7950-0
2023 the 13th International Workshop on Computer Science and Engineering (WCSE 2023)
doi: 10.18178/wcse.2023.06.012

Self-Learning Target Posture Estimation Method Based on RGBD Data

Fu Hai, Zhao Xincan and Wu Depei +

School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China

Abstract. A self-learning target pose estimation method based on RGBD images is proposed to improve the robustness and applicability of 6DoF object pose estimation for augmented reality assembly guidance. The method is built on a self-supervised approach, using a dataset annotation module to create pseudo-labels for actual data and fine-tune the posture estimate model to accommodate changes in the realistic data distribution. The dataset annotation module uses an Iterative Closest Point (ICP) to solve for the target pose of a single frame and SLAM to locate the camera pose to infer the global pose of the target, in addition to the local pose in various camera spaces. The pose estimation model uses target detection techniques to coarsely segment the image and a dense fusion network to extract multisource features, predicting the target's pose and semantic segmentation results. The labeling rate of the dataset in real scenarios is 36 frames per minute, and the AUC of 6DoF pose estimation is 3.72% higher than that of existing algorithms. According to experimental findings, the self-learning pose estimate method can well adjust to new environments.

Keywords: 6DoF pose estimation, data annotation, deep learning, feature fusion.

1. Introduction

Six-degree-of-freedom (6DoF) object pose estimation is the basis of real-world applications such as robotic grasping and manipulation [1], autonomous driving [2], and augmented reality [3]. Most pose estimation methods perform well when dealing with objects with different shapes and textures, sensor noise, and variable illumination conditions. However, it is difficult to maintain an algorithm's robustness and real-time performance in highly dynamic augmented reality applications.

Traditional techniques use templates designed artificially to solve the 6DoF pose of objects [4], and are only suitable for scenes with apparent features and no dynamic objects. The deep learning convolutional neural network (CNN) approach extracts the sophisticated semantic properties of images by learning a vast quantity of label data. Its accuracy is far beyond that of the traditional artificial feature extraction method, and it provides a new idea for target pose estimation. YOLO-6D [5] uses RGB data as input, predicts the target's 2D keypoints, and uses a Perspective-n-Point (PnP) algorithm to complete 6DoF pose estimation. However, RGB data lack spatial geometric constraints, and the predicted keypoint locations may overlap. He et al. [6] proposed a method using a PVN3D network to extract target geometry information from a depth image, densely fuse texture and depth multisource features to predict target 3D keypoints, and match them with corresponding 3D model keypoints to obtain pose information. However, the keypoint filtering strategy affects pose prediction accuracy. Wang et al. [7] proposed the use of multisource fusion features by DenseFusion for end-to-end target position prediction, but the algorithm requires complex semantic segmentation training and preprocessing. Most of the existing methods is trained with full supervision, and its performance is closely related to the quality of the dataset [8]. When there is a significant difference between the distribution of the training set and the actual scene data, prediction ability decreases dramatically, and it is complicated to label the realistic pose dataset [9]. Self-supervised learning [10] uses assisted tasks to analyze the information of the dataset itself, automatically builds data pseudo-labels, and reduces labor consumption. It has the potential to replace fully supervised learning in characterizing learning. Mitash et al. [11] arranged multiple cameras to complete semantics segmentation and point cloud fusion of multiple perspectives, automatically generate posture data labels, and supervise model training. However, the

+ Corresponding author. Tel.: +86 18039293198.
E-mail address: dpwu@zzu.edu.cn.

method requires real-time calibration of the relative postures of multiple cameras. Ruan et al. [12] analyzed the motion constraints of continuous data frames and supervised the predicted posture results according to the epipolar geometry constraint between feature pairs. However, this scene is only suitable for camera posture tracking, and not the 6DoF object posture-solving task.

This paper presents a self-learning pose estimation method based on RGBD data, which uses a dataset construction method as an assistive technology to automatically build data pseudo-labels and optimize the posture estimation model. The self-learning pose estimate method can solve the complex problems of data annotation and pose solving in real scenarios, thus improving the accuracy and robustness of the target pose estimation algorithm in an AR assembly environment.

2. System Overview

Fig. 1 shows the workflow of the algorithm. The pose estimation method uses a self-supervised training strategy in the offline phase. The actual data labeling work is used as an auxiliary task to automatically construct pseudo-labels and provide data for the pose estimation model. The data annotation module uses a pretrained pose estimation model to predict the initial label values of the actual data, and a dataset construction method to reconstruct the predicted labels to optimally generate pseudo-labels. The 6DoF pose estimation module uses the actual data and pseudo-labels to fine-tune the pose estimation model parameters, update the extracted RGBD fusion features, and predict the data's pose and semantic segmentation results. Self-learning posture estimation can automatically label actual data, expand datasets, fine-tune model parameters, and continuously learn to adapt to the data distribution of new scenarios. The assembly guidance animation in the AR environment is displayed in the online phase based on the target position result, and the visual assembly guide is realized.

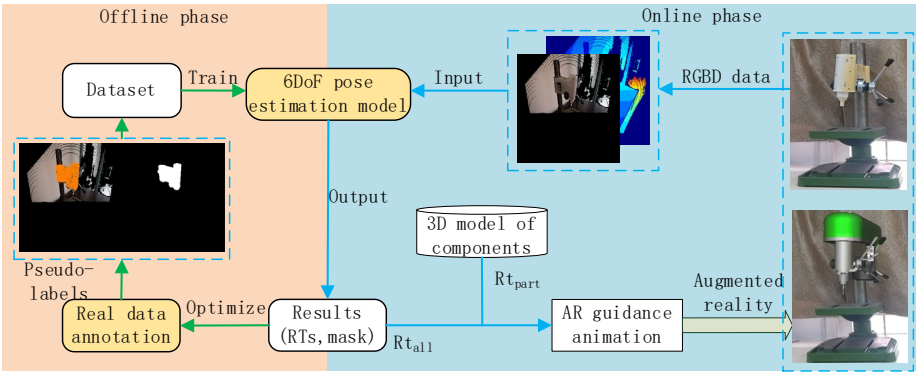

Fig. 1: Workflow of the algorithm.

2.1. Real data annotation module

The current semi-automated data annotation method for arranging and calibrating multiple sensors is time-consuming. Automatic positional localization replaces manual calibration for multiple sensors by combining a single camera with Simultaneous Localization and Mapping (SLAM) [13]. By calculating the target's pose in a single frame, with SLAM to locate the camera pose in different frames, the global pose of the target and the local pose labels in different frames can be inferred.

The real data annotation method takes the target local pose Rt_c and the semantic segmentation mask from the prediction results of the pretrained pose estimation module as initial values and uses the Iterative Closest Point (ICP) point cloud registration algorithm to calculate the pose transformation relationship between the target and the corresponding 3D model.

Fig. 2 shows the data annotation process. The method takes the initial values of the local pose Rt_c and the segmentation mask predicted from a single frame (corresponding to camera pose Rt_{c2w}), and uses the 3D model as the alignment target for ICP optimization to obtain the optimized local pose Rt_{c_icp},

$$Rt_{c\_icp} = ICP(Rt_c) \tag{1}$$

Considering the relative vagueness of the mask information mask and the corresponding point cloud information, it is impossible to determine the optimization results' accuracy. The semantic segmentation

results and corresponding camera poses are used to construct the whole multiview point cloud fusion information as the alignment target. Then, the optimized global posture Rt_{w_icp} is obtained after the second ICP optimization,

$$Rt_{w\_icp} = ICP(Rt_{c2w} * Rt_{c\_icp}) \tag{2}$$

The positional transformation relationship between the global pose Rt_{w_icp} and local pose Rt_{ci} is solved according to the camera pose Rt_i of the different frames,

$$Rt_{ci} = Rt_i^{-1} * Rt_{w\_icp} \tag{3}$$

Finally, the image's local pose, mask, and bounding box information are automatically labeled according to the 3D model and local pose information.

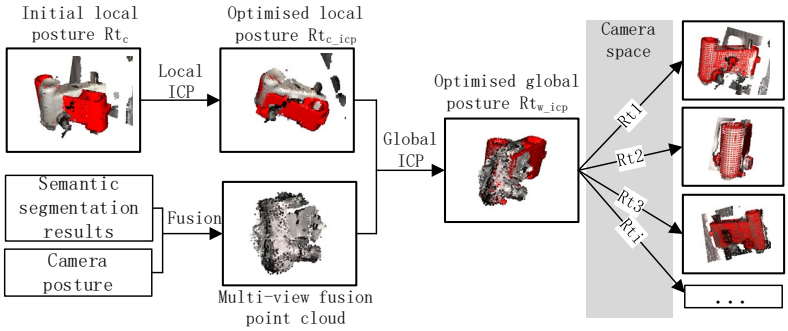

Fig. 2: Real data labeling process.

To initialize the model parameters in the pose estimation module requires the use of physical simulation technology to build a dataset training model for the first time, and to subsequently fine-tune the model using actual data and pseudo-labeling.

2.2. Target pose estimation module

The target pose estimation module is based on DenseFusion [7]. A pose estimation model, 6DPoseCNN, is constructed to meet the model's need for complex semantic segmentation preprocessing and a fixed input scale. The 6DPoseCNN model improves image data rough segmentation preprocessing, variable point cloud feature extraction, and semantic segmentation prediction.

(1) Model structure

Fig. 3 shows the structure of 6DPoseCNN. The network model includes target detection, feature extraction, posture prediction, and a posture optimization module.

Data segmentation preprocessing can filter redundant background information to make the model focus on extracting target features. The annotation information of the target detection box is more accessible than the semantic segmentation annotation information. Therefore, the target detection module (YOLO [14]) replaces the semantic segmentation module for data preprocessing to solve the problem of complex annotation of the semantic segmentation data.

The scale of the input data in the feature extraction module varies due to the range of the target detection results, and RandLA-Net [15] can extract geometric features from the input point cloud at various scales. Because global features in the feature fusion module tend to introduce background noise and interfere with the extraction of target features, only local features of target geometric information are extracted, using multilayer convolution. Batch Norm is added after the convolution layer to accelerate network training and prevent gradient explosion in backpropagation. However, the input data scale is inconsistent, and the batch scale can only be 1. A smaller batch scale increases the error rate [16], so Group Norm, which is not related to the batch scale, is used as the batch operation.

There is some background information in the target detection module's pre-segmentation results, so it is necessary to add semantic segmentation subtasks in the pose estimation model to eliminate background interference. Therefore, the semantic segmentation subtasks Ms are constructed using multilayer convolution to predict the category information of each pixel.

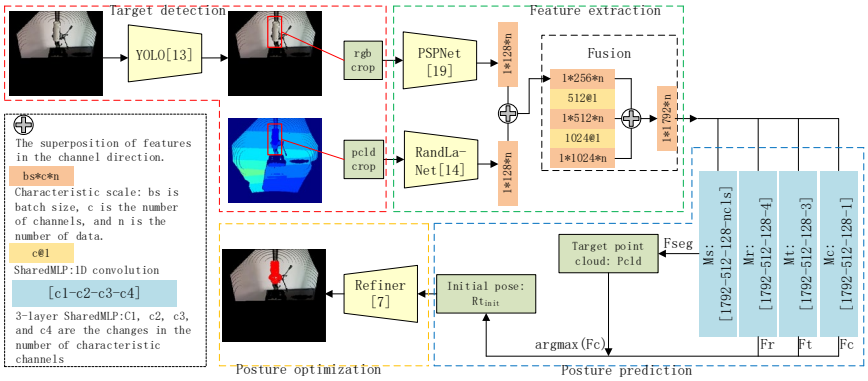

Fig. 3: Overview of pose estimation model.

(2) Loss function

Compared to the single-task supervision of DenseFusion, 6DPoseCNN adds a semantic segmentation subtask and requires multitask supervised training. To synchronously supervise the pose prediction and semantic segmentation sub-task, a 6DPoseCNN multitask loss function L is constructed,

$$L = w_1 Lct + w_2 Ls \tag{4}$$

where Lct is the loss function of the pose estimation task, Ls is the loss function of semantic segmentation tasks, and w_1 and w_2 are the weights of the subtasks, where we take w_1=w_2=1.0 [6,20].

To supervise the training of the positional estimation subtask, we use the DenseFusion loss function Lct,

$$\begin{cases} Lct = \dfrac{1}{N}\sum_{i=1}^{N}(L_i^{obj} c_i - w\log(c_i)) \\ L_i^{obj} = \dfrac{1}{M}\sum_{j=1}^{M} \| (Rx_j + t) - (R_i x_i + t_i) \| \end{cases} \tag{5}$$

where M denotes the point cloud of the category obj model that has been downsampled, N is the number of pixels belonging to category obj in the image, R and t denote the label poses, R_i and t_i denote the predicted poses of the corresponding pixel positions, c_i is the confidence level of the corresponding pixel prediction, and w is a hyperparameter.

To supervise the training of the semantic segmentation subtask, the semantic segmentation loss function Ls uses a standard multi-category loss function, Focal Loss [17],

$$Ls = -(1 - p_{obj})^{\gamma} \log(p_{obj}) \tag{6}$$

where p is the probability of the pixel prediction category obj, and γ is a hyperparameter.

3. Experimental Results and Analysis

In a pose estimation experiment, the RGBD camera in the Microsoft HoloLens2 helmet was used to collect actual scene data, where the resolutions of the depth image and RGB image were 320 × 288 and 760 × 428, respectively, and the resolution after alignment with the depth image was 320 × 288. An NVIDIA Quadro K2200 (4G) GPU was used for accelerated training of the deep learning network. The model was built using the deep learning framework PyTorch 1.8. The Adam optimizer was used, the batch size was 1, and the iteration cycle (Epoch) was 60..

3.1. Self-learning process performance evaluation

To evaluate the impact of the self-learning process on model performance, the experiment had three stages, collecting 389, 415, and 429 frames of data, with differences in light intensity. The data in these three stages were used to conduct two self-learning tests for changing scenes. The first self-learning process used all the data from the first stage to train the 6DPoseCNN model, while 2/3 of the data from the second stage

were randomly selected for fine-tuning, and the remaining 1/3 were used for testing. The second self-learning process used the data from the first and second stages to train the 6DPoseCNN model, randomly selecting 2/3 of the data from the third stage for fine-tuning, and the remaining 1/3 of the data for testing. The accuracy threshold curve with a maximum threshold of 0.1 m was drawn [18]. The area under the curve (AUC) was used as the evaluation criterion of pose estimation, whose increased significance indicated a better pose estimation effect. The impact of data fine-tuning on the performance of the pose estimation model in the self-learning process is shown in Table 1. "(No) fine-tuning" indicates whether fine-tuning data were used to optimize the model. It can be seen that the performance after fine-tuning was better than before. The self-learning process of the fine-tuning model could improve its adaptability to new scenes.

Table 1: Effect of data fine-tuning in the self-learning process on the performance of the pose estimation model.

Process	Data volume	Training set	Test set	AUC
Initialize model	389	389	--	--
First self-learning process (no fine-tuning)	415	--	139	87.30
First self-learning process (fine-tuning)	415	276	139	95.76
Second self-learning process (no fine-tuning)	429	--	143	87.98
Second self-learning process (fine-tuning)	429	286	143	95.75

3.2. Dataset construction performance evaluation

The construction of the actual scene dataset has seven stages: data collection, target detection box annotation, initial pose prediction, and semantic segmentation, manual selection, ICP local optimization, point cloud fusion and ICP global optimization, and positional solution for each viewpoint, where the first four involve data acquisition and preprocessing. Table 2 shows the time consumption of each execution phase, based on 400 actual data annotations. The manual data annotation method took 29.51 min, and the annotation rate was 13 frames per minute, where target detection box annotation (manual annotation) took the most time. To further improve the data annotation rate, the YOLO model was used instead of manual detection frame annotation to predict the results. Data annotation based on the target detection model took 11.01 minutes. The annotation speed was 36 frames per minute, which was 2.6 times greater than manual data annotation.

Table 2: Labeling process and time consumed/min

	Process	Manual data annotation method	Data annotation based on target detection
1	Data collection	2.00	2.00
2	Target detection box annotation	20.00	2.00
3	Initial pose prediction and semantic segmentation	4.00	4.00
4	Manual selection	0.50	0.00
5	ICP local optimization	0.01	0.01
6	Point cloud fusion and ICP global optimization	2.00	2.00
7	Positional solution for each viewpoint	1.00	1.00
--	Total time	29.51	11.01

To compare the effects of target detection box annotation methods on data annotation, different detection criteria were used to evaluate the similarity of the two label results. Using Intersection over Union (IOU) [14] to evaluate the effect of target detection box annotation, the similarity of such labels reached 97.89%; using AUC to evaluate the effect of pose annotation, the similarity reached 98.13%. The experimental data show

that target detection-based data annotation had excellent annotation results and a fast annotation rate, and could quickly provide label data for self-supervised training.

3.3. Pose estimation model performance evaluation

The data of pose estimation model evaluation experiment came from HoloLens2 to collect the RGBD data of the actual scene and used the dataset construction method to generate the corresponding real scene dataset. The dataset contained 1233 frames, with 986 frames for training, and 247 for testing. The loss convergence curve of the 6DPoseCNN training process is shown in Fig. 4. The training process has two stages: (1) prediction, where the posture optimization module does not participate in training and prediction; (2) pose optimization, where the posture optimization module is trained and the other modules are frozen. Fig. 4 can be seen that the pose optimization process saw rapid convergence of the loss curve after initialization. The Average Distance of Model Points (ADD) was used to evaluate pose error. If the ADD was < 2 cm, the pose estimation was correct [7]. The accuracy of 6DPoseCNN pose prediction was 98.8%. The pose prediction results of some data in Fig. 5 show that the model point cloud overlapped well with the target point cloud.

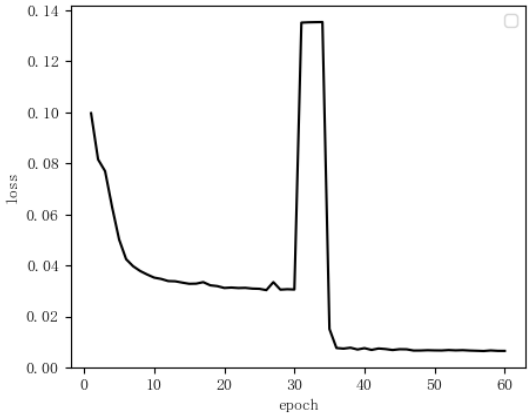

Fig. 4: Loss convergence curve of 6DPoseCNN.

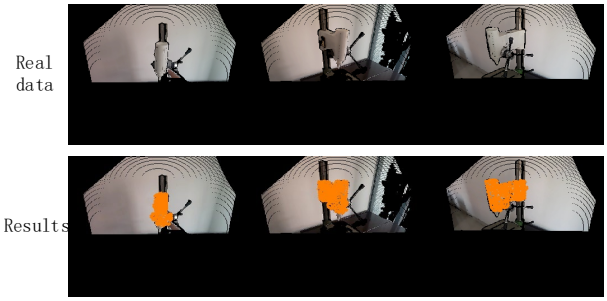

Fig. 5: Part of data, with corresponding pose prediction results.

To further evaluate the performance of pose estimation algorithms, the ADD less than 2 cm and AUC are used as the evaluation criteria for pose estimation of different algorithms. The comparison results are shown in Fig. 6 and Table 3, from which it can be seen that the 6DPoseCNN model had the best comprehensive effect, with prediction pose error superior to that of other algorithms under different thresholds. Compared with the DenseFusion model, the prediction accuracy AUC was increased by 3.72%. The experimental results of PVNet [20] were inferior to those of the pose prediction model with RGBD as input because it only used RGB as input, which lacks the depth geometry information, and the predicted 2D keypoints tend to overlap. As a pose estimation model with RGBD input, MaskedFusion [21] introduced geometric depth data and used this as an expanded dimension of texture data features without considering the influence of geometric feature information. PVN3D [6] used different CNNs to analyze texture and geometric features and used a multisource feature fusion mechanism to extract the target's keypoints and solve the pose-matching relationship with the keypoints of the 3D model. However, the keypoint extraction strategy affected the performance of the algorithm. DenseFusion [7] used the fusion features of texture and geometry

through a multisource fusion mechanism to directly regress the target poses, and an iterative module to optimize the prediction results. However, its fixed input was prone to missing key information during downsampling. The extraction of global features was prone to introducing background noise. The pose-prediction results relied heavily on the data preprocessing effect of semantic segmentation, and thus failed to obtain higher-quality results.

Table 3: Accuracy of pose estimation with different algorithms.

	PVNet [20]	MaskedFusion [21]	PVN3D [6]	DenseFusion [7]	Ours
ADD (< 2 cm)	53.84	86.63	74.09	97.16	98.80
AUC	77.53	86.07	84.94	90.91	94.63

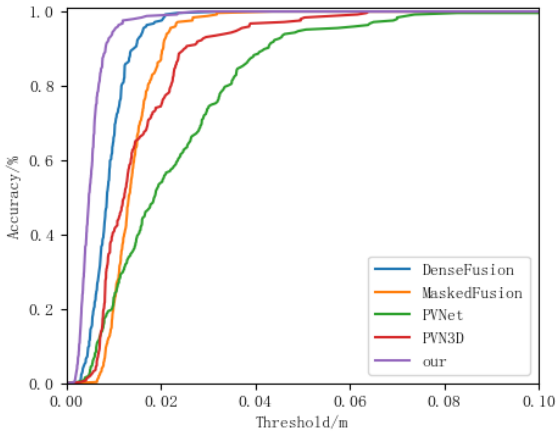

Fig. 6: Accuracy of posture estimation with different algorithms.

3.4. Enhanced assembly effect display for typical scenarios

With the AGILEX SCOUT MINI robot as the assembly target, an AR-based assembly assistant system was developed. The system used a Microsoft HeloLens2 helmet as an enhanced display device and HelmetSceneGraph (HSG) for three-dimensional graphics rendering and as an image processing engine. During the experiment, the robot was kept motionless, and the Hololens2 helmet was moved to obtain the display effect of assembly guidance information from different viewpoints and distances. The experimental results are shown in Fig. 7, where Fig. 7a shows the target robot, according to which the dataset was constructed, the pose estimation model was trained, and the poses in the augmented reality space were predicted. Fig. 7b shows the target's pose-estimation results, and overlays the target assembly information based on the target's 6DoF poses. The HoloLens2 was moved to change its viewing angle with the target to obtain the registration results of the guidance information under different viewing angles (as shown in Figs. 7c and 7d). The diagram shows that the posture estimation algorithm achieved good pose estimation results. Figs. 7e and 7f show the guidance screen of the disassembly process, which is marked with dynamic guidance information to determine the position of the tires and screws of the parts to be disassembled.

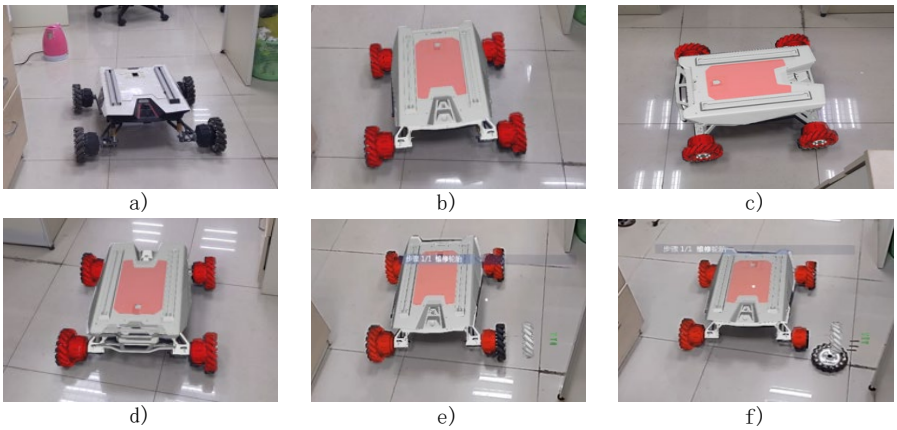

Fig. 7: Posture prediction results in augmented reality scenes.

4. Conclusion

In this paper, we presented a pose estimation method for an AR mechanical assembly assistant system. The method used a self-supervised training method, combined with a dataset construction method to quickly annotate data, in addition to using new data to optimize 6DPoseCNN to adapt to changes in the scene data distribution, for achieving the goal of continuous learning. The introduction of SLAM technology in the dataset construction phase overcomes the problem of multi-sensor placement and mutual calibration. We used 6DPoseCNN to solve the problem of assembly environment complexity and posture prediction accuracy. Experimental results showed that the self-learning pose estimation method had better adaptability to changing scenes, the AUC could reach 94.63% on real datasets, and the method could realize better accuracy in the disassembly process based on the AR assistant system. Because the dataset construction method needs to be analyzed using a 3D target model, this poses estimation method is limited to scenes with a 3D target model.

5. Acknowledgements

We are grateful to AGILEX(www.agilex.ai) for the 3D model and the technical support.

6. References

[1] Tremblay J, To T, Sundaralingam B, et al. Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects[C]//Conference on Robot Learning. PMLR, 2018: 306-316.

[2] Chen X, Ma H, Wan J, et al. Multi-view 3d object detection network for autonomous driving[C]//Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017: 1907-1915.

[3] Marchand E, Uchiyama H, Spindler F. Pose estimation for augmented reality: a hands-on survey[J]. IEEE transactions on visualization and computer graphics, 2015, 22(12): 2633-2651.

[4] Salti S, Tombari F, Di Stefano L. SHOT: Unique signa-tures of histograms for surface and texture description[J]. Computer Vision and Image Understanding, 2014, 125: 251-264.

[5] Bugra T, Sudipta N, Pascal F. Real-Time Seamless Single Shot 6D Object Pose Prediction[C]//Conference on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018: 292--301.

[6] He Y, Sun W, Huang H, et al. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation[C]//Conference on Computer Vision and Pattern Recognition. Seattle: IEEE, 2020: 11632-11641.

[7] Wang C, Xu D, Zhu Y, et al. Densefusion: 6d object pose estimation by iterative dense fusion[C]//Conference on Computer Vision and Pattern Recognition. Long Beach: IEEE, 2019: 3343-3352.

[8] PAN C Y, HUANG J, HAO J G, et al. Survey of weakly supervised learning integrating zero-shot and few-shot learning[J]. Systems Engineering and Electronics, 2020,42(10):2246-2256.

[9] LI S F, SHI Z L, ZHUANG C G. Deep learning-based 6D object pose estimation method from point clouds[J]. Com-puter Engineering, 2021,47(08):216-223.

[10] Hjelm R D, Fedorov A, Lavoie S, et al. Learning deep representations by mutual information estimation and maximization[C]// 7th International Conference on Learning Representations (ICLR). USA: OpenReview, 2019: 1-24.

[11] Mitash C, Bekris K E, Boularias A. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Canada: IEEE, 2017: 545-551.

[12] RUAN X G, LI A, HUANG J. Pose estimation method based on self-supervised recurrent convolutional neural networks[J]. Journal of Beijing University of Technology, 2021,47(12):1311-1320.

[13] Mur-Artal R, Tardós J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras[J]. IEEE transactions on robotics, 2017, 33(5): 1255-1262.

[14] Redmon J, Farhadi A. YOLO9000: better, faster, strong-er[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. Honolulu: IEEE, 2017: 7263-7271.

[15] Hu Q, Yang B, Xie L, et al. Randla-net: Efficient semantic segmentation of large-scale point clouds[C]// Conference on Computer Vision and Pattern Recognition. Seattle: IEEE, 2020: 11108-11117.

[16] Wu Y, He K. Group Normalization[J]. International Journal of Computer Vision, 2020, 128(3): 742-755.

[17] Lin T Y, Goyal P, Girshick R, et al. Focal Loss for Dense Object Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2020, 42(02): 318-327.

[18] Xiang Y, Schmidt T, Narayanan V, et al. Posecnn: A convolutional neural network for 6d object pose estima-tion in cluttered scenes[J]. Robotics: Science and Systems XIV: 2018.

[19] Zhao H, Shi J, Qi X, et al. Pyramid scene parsing net-work[C]//Proceedings of the IEEE conference on com-puter vision and pattern recognition. Honolulu: IEEE, 2017: 2881-2890.

[20] 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. 2019: 4561-4570.

[21] Pereira N, Alexandre L A. MaskedFusion: Mask-based 6D object pose estimation[C]//2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2020: 71-78.