G2L-Net: Global to Local Network for Real-time 6D Pose Estimation with Embedding Vector Features

Wei Chen [1,2]   Xi Jia[1]   Hyung Jin Chang[1]   Jinming Duan[1]   Ales Leonardis[1]

[1] School of Computer Science, University of Birmingham
[2] School of Computer Science, National University of Defense Technology

{✉wxc795,X.Jia.1,h.j.chang,j.duan,a.leonardis}@cs.bham.ac.uk

Abstract

In this paper, we propose a novel real-time 6D object pose estimation framework, named G2L-Net. Our network operates on point clouds from RGB-D detection in a divide-and-conquer fashion. Specifically, our network consists of three steps. First, we extract the coarse object point cloud from the RGB-D image by 2D detection. Second, we feed the coarse object point cloud to a translation localization network to perform 3D segmentation and object translation prediction. Third, via the predicted segmentation and translation, we transfer the fine object point cloud into a local canonical coordinate, in which we train a rotation localization network to estimate initial object rotation. In the third step, we define point-wise embedding vector features to capture viewpoint-aware information. To calculate more accurate rotation, we adopt a rotation residual estimator to estimate the residual between initial rotation and ground truth, which can boost initial pose estimation performance. Our proposed G2L-Net is real-time despite the fact multiple steps are stacked via the proposed coarse-to-fine framework. Extensive experiments on two benchmark datasets show that G2L-Net achieves state-of-the-art performance in terms of both accuracy and speed. [1]

1. Introduction

Real-time performance is important in many computer vision tasks, such as, object detection [34, 21], semantic segmentation [35, 9], object tracking [5, 10], and pose estimation [28, 38, 15]. In this paper, we are interested in real-time 6D object pose estimation, which has significant impacts on augmented reality [23, 24], smart medical and robotic manipulation [47, 39].
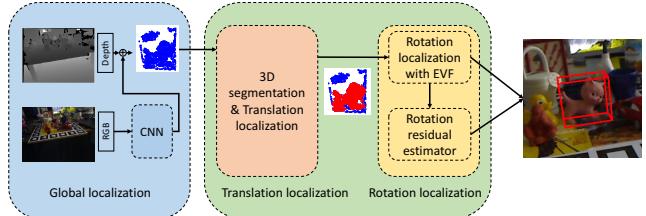

Figure 1. **Three steps of G2L-Net.** We propose a novel real-time point cloud based network for 6D object pose estimation called G2L-Net. Our G2L-Net contains global localization, object translation localization and rotation localization. For the rotation localization, we propose point-wise embedding vector features (EVF) and rotation residual estimator to access accurate rotation.

Deep learning methods have shown the state-of-the-art performance in the pose estimation tasks, but many of them [42, 32, 27, 19] cannot run in real-time. While there exist some real-time deep learning methods [28, 33, 38, 45] (> 20fps), they use only RGB information from an image. One major limitation of using RGB only is that features learned from such information are sensitive to occlusion and illumination changes, which precludes these methods from being applied to complicated scenes. Deep learning methods based on depth information [16, 19] are more suitable for realistically complicated scenes, but they are usually computation-intensive. One common issue for these RGBD-based methods is that exploiting viewpoint information from depth information is not very effective, thus reducing their pose estimation accuracy. To overcome this, these methods tend to use a post-refinement mechanism or a hypotheses generation/verification mechanism to enhance pose estimation accuracy. This, however, reduces the inference speed for pose estimation.

In this paper, to overcome the existing problems in depth-based methods, we propose a global to local real-

---

[1]Our code is available at https://github.com/DC1991/G2L_Net.

time network (G2L-Net), with two added modules which are point-wise embedding vector features extractor and rotation residual estimator. Built on [29], our method has three major novelties: i) instead of locating the object point cloud by a frustum, we locate the object point cloud by a 3D sphere, which can limit the 3D search range in a more compact space (see Section 3.1 for details), ii) instead of directly regressing the global point feature to estimate the pose, we propose the point-wise embedding vector features to effectively capture the viewpoint information, and iii) we estimate the rotation residual between predicted rotation and the ground truth. The rotation residual estimator further boosts the pose estimation accuracy. We evaluate our method on two widely-used 6D object pose estimation datasets, *i.e.* LINEMOD [11] and YCB-Video [42] dataset. Experimental results show that G2L-Net outperforms state-of-the-art depth-based methods in terms of both accuracy and speed on the LINEMOD dataset, and that G2L-Net achieves comparable accuracy while is the fastest method on the YCB-Video dataset.

In summary, the main contributions of this paper are as follows:

- We propose a novel real-time framework to estimate 6D object pose from RGB-D data in a global to local (G2L) way. Due to efficient feature extraction, the framework runs at over 20fps on a GTX 1080 Ti GPU, which is fast enough for many applications.

- We propose orientation-based point-wise embedding vector features (EVF) which better utilize viewpoint information than the conventional global point features.

- We propose a rotation residual estimator to estimate the residual between predicted rotation and ground truth, which further improves the accuracy of rotation prediction.

2. Related work

Pose estimation from RGB image: Traditional methods [11, 18, 22] compute 6D object pose by matching RGB features between a 3D model and test image. These methods use handcrafted features that are not robust to background clutter and image variations [44, 36, 28]. Learning-based methods [32, 28, 33, 27, 14, 38] alleviate this problem by training their model to predict 2D keypoints and compute the object pose by the PnP algorithm [8, 26]. [42, 20, 19] decouple the pose estimation into two sub-tasks: translation estimation and rotation estimation. More concretely, they regarded the translation and rotation estimation as a classification problem and trained neural networks to classify the image feature into a discretized pose space. However, the RGB image features may be affected by illumination

changes which result in pose estimation from RGB image more sensitive to illumination changes.

Pose estimation from RGB image with depth information: When depth information is available, previous approaches [2, 37, 41, 12] learned features from the input RGB-D data and adopted correspondence grouping and hypothesis verification. However, some papers [44, 36] found that the methods are sensitive to image variations and background clutter. Besides, correspondence grouping and hypothesis verification further increase the inference time presenting real-time applications. Some methods [42, 15] employed the depth information in a post-refinement procedure by highly customized Iterative Closest Point (ICP) [6, 1] into deep learning frameworks, which would significantly increase the running time of the algorithms. Recently, several deep learning methods [16, 19] utilized the depth input as an extra channel along with the RGB channels. However, combining depth and RGB information in this way cannot make full use of geometric information in data and makes it difficult to integrate information across viewpoints [25]. Instead, in this paper, we transfer depth maps to 3D point clouds and directly process the 3D point clouds by Point-Nets [30, 29] which extract 3D geometric features more efficiently than CNN-based architectures.

Pose estimation from point cloud: PointNets [30, 29], Qi *et al.* have shown that employing depth information in 3D space via the point cloud representation could achieve better performance than that in 2.5D space. Based on that, some PointNet-based methods [29, 40, 46, 43, 4] presented to directly estimate 6D object pose. They adopted a PoinNet-like [30] architecture to access pose from point cloud. In this work, we also make use of PointNet-like architecture but in a different way. Different from 2D methods [42, 19], we decouple 6D object pose estimation into three sub-tasks: global localization, translation localization, and rotation localization. For the first two sub-tasks, we use similar methods in [29] but with some improvements described in Section 3. For the third sub-task, we propose point-wise embedding vector features that exploit the viewpoint information more effectively and we also propose a rotation residual estimator that further improves the pose estimation accuracy. We show that with these improvements, the proposed G2L-net achieves higher accuracy than state-of-the-art methods and runs at real-time speed.

3. Proposed method

In Figure 2, we show the inference pipeline of our proposing G2L-Net which estimates the 6D object pose in three steps: global localization, translation localization, and rotation localization. In the global localization step, we use a 3D sphere to fast locate the object position in 3D space. In the translation localization step, we train a PointNet to perform 3D segmentation and estimate object translation. In
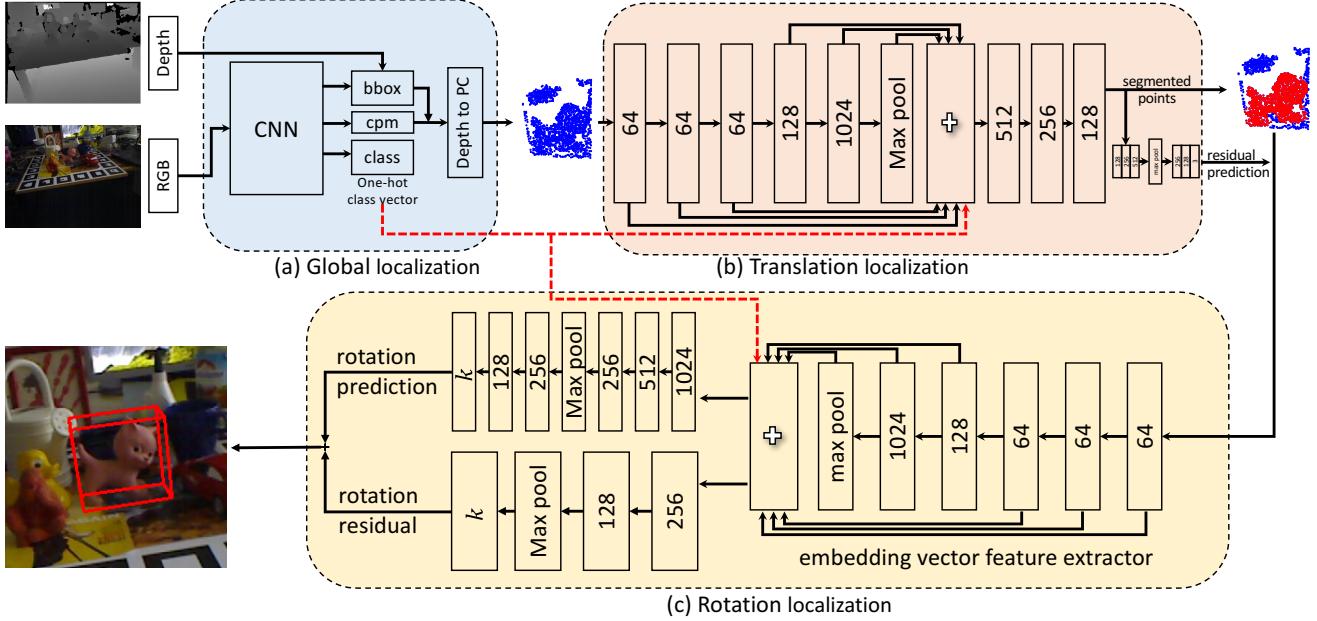
Figure 2. **Inference pipeline of the proposed G2L-Net.** (a) For RGB image, we use a 2D detector to detect the bounding box (bbox) of the target object and the object label which is used as a one-hot feature for the following networks. Also, we additionally choose the maximum probability location in class probability map (cpm) as the sphere center (we transfer this 2D location to 3D with known camera parameters and corresponding depth value) which is used to further reduce the 3D search space. (b) Given the point clouds in the object sphere, we use a translation localization network to perform 3D segmentation and translation residual prediction. Then we use the 3D segmentation mask and the predicted translation to transfer the object point cloud into a local canonical coordinate. (c) In the rotation localization network, we first use the point-wise embedding vector feature extractor to extract embedding vector features. Then we feed this feature into two-point clouds decoders: the top decoder directly outputs the rotation of the input point cloud and the bottom one outputs the residual of the output of the top one between the ground truth. k is the dimension of the output vector. "$+$" denotes feature concatenation.

the third step, we estimate rotation with the proposed point-wise embedding vector features and rotation residual estimator. Please note, this rotation residual estimator is different from the post-refinement component in previous methods [42, 15], it outputs rotation residual with initial rotation synchronously. In the following subsections, we describe each step in detail.

3.1. Global localization

To fast locate the global position of the target object in the whole scene, we train a 2D CNN detector, YOLO-V3 [34], to detect the object bounding box in RGB image, and output object label which is used as one-hot class vector for better point cloud instance segmentation, translation and rotation estimation. In [29], they use the 2D bounding box to generate frustum proposals which only reduce the 3D search space of two axes (x,y). Differently, rather than only using a 2D bounding box, we propose to employ a 3D sphere to further reduce the 3D search space in the third axis (z) (see Figure 3 for details). The center of the 3D sphere is transferred from the 2D location which has the maximum value in the class probability map with known camera parameters and corresponding depth value. The radius of this

3D sphere is the diameter of the detected object. We only choose points in this compact 3D sphere, which makes the learning task easier for the following steps.

3.2. Translation localization

Although the extracted point cloud is tight, there are still two issues remained: 1) the point cloud in this 3D space contains both object points and non-object points, and 2) the object points cannot be transferred to a local canonical coordinate due to unknown translation. To cope with the issues, similar to [29], we train a two PointNets [30] to perform 3D segmentation and output the residual distance $||T - \bar{T}||_2$ between the mean value $\bar{T}$ of the segmented points and object translation T. This residual can be used to calculate the translation of the object.

3.3. Rotation localization with embedding vector feature

From the first two steps, we transfer the point cloud of the object to a local canonical space where the viewpoint information is more evident. Theoretically, we need at least four different viewpoints to cover all points of an object (see Figure 4) in 3D space [?]. For the pose estimation task, we
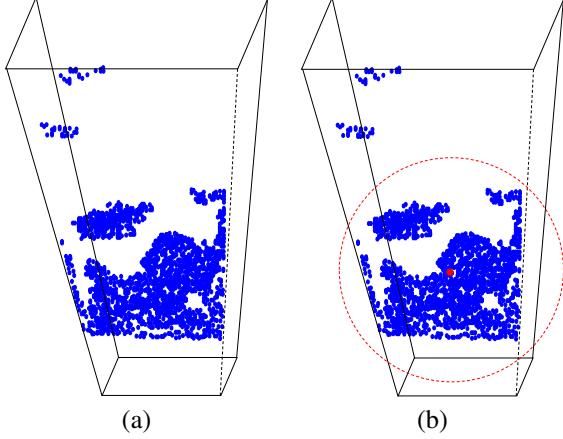
Figure 3. **Global 3D sphere.** In the global localization step, we locate the object point clouds by bounding box as well as a 3D sphere. (a) Locate the object point cloud only by bounding box. In this case, it can only locate the object in two-dimensional space, some points can still very far away from the object in the third axis. (b) Locate the object point cloud by both the bounding box and 3D sphere. All points lay in a more compact space.
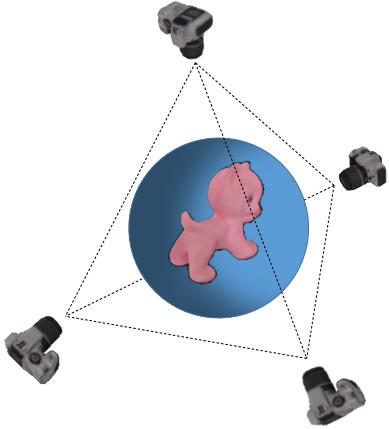

Figure 4. **Different viewpoints.** For a 3D object, we need at least four viewpoints to cover all the points of the 3D object.

usually have hundreds of different viewpoints for one object during training. Then our goal is to use the viewpoint information adequately. In [29], they use PointNets [30, 31] to extract the global feature from the whole point cloud. However, in our experiments, we found global point features extracted from point clouds under similar viewpoints are highly correlated, which limit the generalization performance (see Figure 9 in the experiment section).

To overcome the limitation of global point features, we propose point-wise embedding vector features. Specifically, we design the rotation localization network architecture as shown in Figure 5 to predict point-wise unit vectors pointing to keypoints (illustrated in Figure 6). The keypoints are
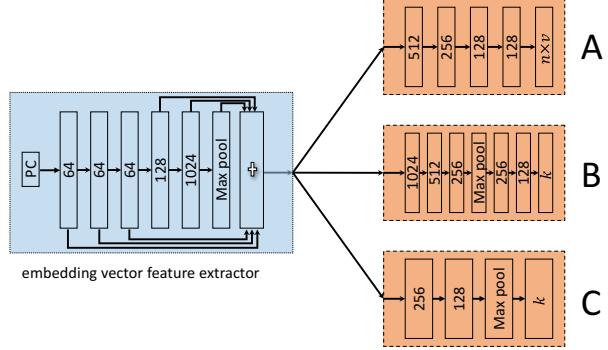

Figure 5. **Architecture of the rotation localization network.** In the training stage, there are three blocks in rotation localization network. We train block A to predict the unit vectors pointing to the keypoints, the loss function of this block is the mean square error between the predicted and ground truth directional vectors. By training this block, the network can learn how to extract point-wise embedding vector features from the input point cloud. Note that, block A is not deployed in the inference stage. Then we use block B to integrate the point-wise embedding vector features to predict object rotation. The loss function of this block is the mean square error between the predicted rotation and ground truth. For rotation residual estimator block C, we use the Euclidean distance between the predicted 3D keypoints position (output of block B) and ground truth as ground truth. k is the dimension of the output rotation vector and v is the dimension of the output directional vector. "$+$" denotes feature concatenation.
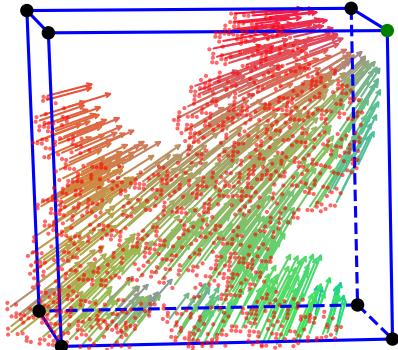

Figure 6. **Point-wise vectors.** Here we show point-wise vectors pointing to one keypoint which is shown in green color, and other keypoints are shown in black color. We train our network to predict such directional vectors

some pre-defined 3D points based on each 3D object model. Two aspects need to be decided for the keypoints: number and location. A simple way is to use the 8 corners of the 3D bounding box of object model as keypoints which we show in Figure 7 (a). This definition is widely used by many CNN based methods in 2D cases [32, 33, 27, 38]. Another way is, as proposed in [28], to use the farthest point sampling (FPS) algorithm to sample the keypoints in each object model. Figure 7 shows examples of different keypoint
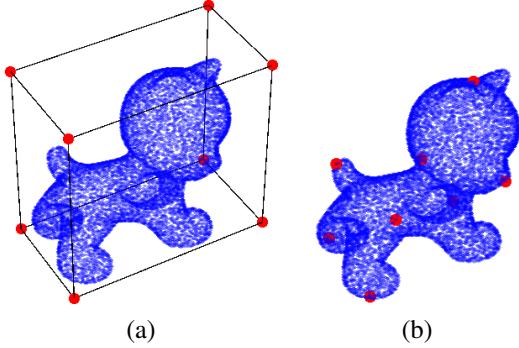
(a)   (b)

Figure 7. **Visualization of different keypoint selection schemes**.
The left image is a 3D object point cloud and its 3D bounding
box; the right image is the keypoint selected by FPS algorithm.
The keypoints are shown in red color.

selection schemes. In Section 4.4, we show how the number
and location of the keypoints influence the pose estimation
results.

Similar to [4], our proposed rotation localization net-
work takes object point cloud in the local canonical space
and outputs point-wise unit vectors pointing keypoints. The
loss function is defined as follows:

$$\ell(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{K|\mathcal{X}|} \sum_{k=1}^{K} \sum_{i} \|\widetilde{v}_k(\mathcal{X}_i; \boldsymbol{\theta}) - v_k(\mathcal{X}_i)\|_2^2, \quad (1)$$

where K is the number of keypoints. $\boldsymbol{\theta}$ is the network pa-
rameters. $\widetilde{v}_k(\mathcal{X}_i; \boldsymbol{\theta})$ and $v_k(\mathcal{X}_i)$ are the predicted vectors
and the ground truth, respectively. $\mathcal{X} \in \mathbb{R}^{n \times 3}$ represents
the object points in the local coordinate space. $|\mathcal{X}|$ is the
number of object points.

Different from other state-of-the-art methods [28, 42, 4],
we adopt a multilayer perceptron (MLP) that takes point-
wise embedding vector features as input and outputs the
rotation of object as shown in Figure 5. Please note, dur-
ing inference, we use the rotation matrix to represent the
rotation which is computed from the keypoint positions us-
ing the Kabsch algorithm. Over the training process, as per
the definition of point-wise vectors, we used only the key-
point positions to represent rotation. In experiments, we
have found that our proposed method can make faster and
more accurate predictions than the methods [28, 42, 4].

Rotation residual estimator: To better utilize the
viewpoint information in the point-wise embedding vector
features, we add an extra network branch (block C in Fig-
ure 5) to estimate the residual between estimated rotation
(block B in Figure 5) and ground truth. However, we do not
have ground truth for this residual estimator. To address this
problem we train this estimator in online fashion. Assuming
that the ground truth for block B of rotation localization net-
work is $\mathcal{P}$ and the output of block B is $\widetilde{\mathcal{P}}$, then the target of

our rotation residual estimator is $\|\mathcal{P} - \widetilde{\mathcal{P}}\|_2$. As the rotation
network converging, it becomes harder to learn the resid-
ual. If the rotation localization network can fully exploit the
embedding vector feature, the role of rotation residual esti-
mator can be ignored. However, when the rotation network
cannot fully exploit the embedding vector feature, the rota-
tion residual estimator will have a big impact on the final
results, we show this property of rotation residual estimator
in Figure 9 (b). Please note, our proposed rotation resid-
ual estimator is different from the post-refinement module
in the previous state-of-the-art methods [42, 40, 20]. Our
proposed rotation residual estimator outputs rotation resid-
ual with estimated rotation synchronously, which saves the
running time.

4. Experiments

There are two parts in this experiments section. Firstly,
we do ablation studies on keypoints selection schemes and
empirically validate the three innovations introduced in our
new frame: 3D sphere ("SP), point-wise embedding vector
features ("EVF) and rotation residual estimator ("RRE").
Then we test our proposed G2L-Net on two benchmark
datasets, _i.e._ LINEMOD and YCB-Video datasets. Our
method achieves state-of-the-art performance in real-time
on both datasets.

4.1. Implementation details

We implement our framework using Pytorch. We have
conducted all the experiments on an Intel i7-4930K 3.4GHz
CPU with one GTX 1080 Ti GPU. First, we fine-tune the
YOLO-V3 [34] architecture which is pre-trained on the Im-
ageNet [7] to locate the 2D region of interest and access the
class probability map. Then we jointly train our proposed
translation localization and rotation localization networks
using PointNet [30] as our backbone network. The archi-
tectures of these networks are shown in Figure 2. Note that,
other point-cloud network architectures [31, 46] can also be
adopted as our backbone network. For point cloud segmen-
tation we use cross-entropy as the loss function. For trans-
lation residual prediction, we employ the mean square error
and the unit in our experiment is mm. We train our rota-
tion localization network as described in Figure 5. We use
Adam [17] to optimize the proposed G2L-Net. We set the
initial learning rate as 0.001 and halve it every 50 epochs.
The maximum epoch is 200.

4.2. Datasets

LINEMOD [11] is a widely used dataset for 6D object pose
estimation. There are 13 objects in this dataset. For each ob-
ject, there are about 1100-1300 annotated images and each
has only one annotated object. This dataset exhibits many
challenges for pose estimation: texture-less objects, clut-
tered scenes, and lighting condition variations.
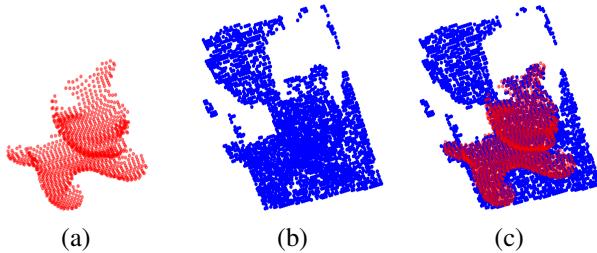
(a)   (b)   (c)

Figure 8. **Point cloud labeling**. (a) The object model of *cat* in LINEMOD dataset; (b) the point cloud from the depth images in object region; (c) the transformed object model is overlapped on the point cloud. We label each point according to the shortest distance between the point and the corresponding transformed object model.

YCB-Video [42] contains 92 real video sequences for 21 YCB object instances [3]. This dataset is challenging due to the image noise and occlusions.

However, both LINEMOD and YCB-Video datasets do not contain the label for each point of the point cloud. To train G2L-Net in a supervised fashion, we adopt an automatic way to label each point of the point cloud of [4]. As described in [4], we label each point in two steps First, for the 3D model of an object, we transform it into the camera coordinate using the corresponding ground truth. We adopt the implementation provided by [13] for this process. Second, for each point on the point cloud in the target region, we compute its nearest distance to the transformed object model. If the distance is less than a value $\epsilon = 8mm$, we label the point as 1 (belonging to the object), otherwise 0. Figure 8 shows the labeling procedure.

4.3. Evaluation metrics

We employ the ADD metric [11] to evaluate our G2L-Net on LINEMOD dataset:

$$\frac{1}{|\mathcal{M}|} \sum_{\mathbf{x} \in \mathcal{M}} \|(\mathbf{R} \cdot \mathbf{x} + \mathbf{T}) - (\widetilde{\mathbf{R}} \cdot \mathbf{x} + \widetilde{\mathbf{T}})\|, \qquad (2)$$

where $|\mathcal{M}|$ is the number of points in the object model. $\mathbf{x}$ represents the point in object 3D model. $\mathbf{R}$ and $\mathbf{T}$ are the ground truth pose, and $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{T}}$ are the estimated pose. In this metric, the mean distance between the two transformed point sets is computed. When the average distance is less than 10% of the 3D object model diameter, we consider that estimated 6D pose as correct. For symmetric objects, we employ ADD-S metric [11], where the average distance is calculated using the shortest point distance:

$$\frac{1}{|\mathcal{M}|} \sum_{\mathbf{x_1} \in \mathcal{M}} \min_{\mathbf{x_2} \in \mathcal{M}} \|(\mathbf{R} \cdot \mathbf{x_1} + \mathbf{T}) - (\widetilde{\mathbf{R}} \cdot \mathbf{x_2} + \widetilde{\mathbf{T}})\|. \qquad (3)$$

When evaluating on YCB-Video dataset, same as [42, 28, 19], we use the ADD-S AUC metric proposed in [42],

Table 1. Ablation studies of different novelties on LINEMOD dataset. The metric we used to measure performance is ADD(-S) metric. "SP" means 3D sphere, "EVF" means embedding vector feature, and "RRE" denotes rotation residual estimator.

Method	SP	EVF	RRE	Acc	Speed(fps)
EXP1	×	×	×	93.4%	25
EXP2	✓	×	×	95.8%	25
EXP3	✓	✓	×	98.4%	23
EXP4	✓	✓	✓	98.7%	23

Table 2. Ablation studies of different keypoints parameters on LINEMOD dataset. The metric we used to measure performance is ADD(-S) metric. **BBX-8** means using the 8 corners of 3D bounding box as keypoints. **FPS-K** denotes that we adopt K keypoints generated by the FPS algorithm.

Method	BBX-8	FPS-4	FPS-8	FPS-12
Acc	98.7%	98.5%	98.4%	98.6%
Speed (fps)	23	23	23	23

which is the area under the accuracy-threshold curve. The maximum threshold is set to 10cm [42].

4.4. Ablation studies

Compared to the baseline method [29], our proposed method has three novelties. First, we fast locate the object point clouds by a 3D sphere which is different from the frustum method in [29]. Second, we use the proposed point-wise embedding vector features to estimate rotation of the point cloud which can better utilize the viewpoint information. Third, we propose a rotation residual estimator to estimate the rotation residual between ground truth and predicted rotation. From Table 1, we can see that the proposed three improvements can boost performance.

We also compare the different keypoints selection schemes in Table 2, however, it shows that different keypoints selection schemes make little difference in the final results. For simplicity, we use the 8 corners of 3D bounding box as keypoints in our experiments.

4.5. Generalization performance

In this section, we evaluate the generalization performance of our G2L-Net. We gradually reduce the size of training data to see how the performance of the algorithm can be affected on LINEMOD dataset. From Figure 9 (a), we can see that even only 5% of the training data, which is 1/3 of the normal setting, is used for the network training, the performance (88.5%) is still comparable.

4.6. Comparison with the state-of-the-art methods

Object 6D pose estimation on LINEMOD: Same as other state-of-the-art methods, we use 15% of each object sequence to train and the rest of the sequence to test on

Table 3. 6D pose estimation accuracy on LINEMOD dataset. We use ADD metric to evaluate the methods. For symmetric objects *Egg Box* and *Glue*, we use ADD-S metric. Note that, we summarize the pose estimation results reported in the original papers on LINEMOD dataset.

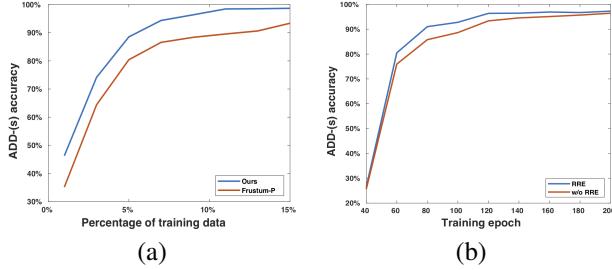| Method	PVNet [28]	PoseCNN + DeepIM [42, 19]	DPOD [45]	Frustum-P [29]	Hinterstoisser [12]	DenseFusion [40]	Ours
Input	RGB	RGB	RGB	RGB+Depth	Depth	RGB+Depth	RGB+Depth
Refinement	×	✓	✓(×)	×	✓	✓(×)	×
Ape	43.6%	77.0%	87.7% (53.3%)	85.5%	**98.5%**	92.3% (79.5%)	96.8%
Bench Vise	**99.9%**	97.5%	98.5% (95.3%)	93.2%	99.0 %	93.2%(84.2%)	96.1%
Camera	86.9%	93.5	96.0% (90.4%)	90.0%	**99.3%**	94.4%(76.5%)	98.2%
Can	95.5%	96.5%	99.7% (94.1%)	91.4%	**98.7%**	93.1%(86.6%)	98.0%
Cat	79.3%	82.1%	94.7% (60.4%)	96.5%	**99.9%**	96.5%(88.8%)	99.2%
Driller	96.4%	95.0%	98.8% (97.7%)	96.8%	93.4%	87.0%(77.7%)	**99.8%**
Duck	52.6%	77.7%	86.3% (66.0%)	82.9%	**98.2%**	92.3%(76.3%)	97.7%
Egg Box	99.2%	97.1%	99.9% (99.7%)	99.9%	98.8%	99.8%(99.9%)	**100%**
Glue	95.7%	99.4%	96.8% (93.8%)	99.2%	75.4%	**100%** (99.4%)	**100%**
Hole Puncher	81.9%	52.8%	86.9% (65.8%)	92.2%	98.1%	92.1%(79.0%)	**99.0%**
Iron	98.9%	98.3%	**100%** (99.8%)	93.7%	98.3%	97.0% (92.1%)	99.3%
Lamp	99.3%	97.5%	96.8% (88.1%)	98.2%	96.0%	95.3%(92.3%)	**99.5%**
Phone	92.4%	87.7%	94.7% (74.2%)	94.2%	98.6%	92.8%(88.0%)	**98.9%**
Speed(FPS)	25	5	33(40)	12	8	16(20)	23
Average	86.3%	88.6%	95.2% (83.0%)	93.4%	96.3 %	94.3 %(86.2%)	**98.7 %**

(a)   (b)

Figure 9. **Visualization of method performance on LINEMOD dataset.** (a) Influence of training data size using the ADD metric. When using the same training size, compared to Frustum-P [29], our method improves the performance significantly. For simplicity, here we provide ground truth 2D bounding box and randomly choose an object point as 3D sphere center for evaluation. (b) As the rotation localization network converging, the impacts of rotation residual estimator (RRE) decreases.

Table 4. 6D Pose estimation accuracy on the YCB-V dataset. We use ADD-S AUC metric to evaluate the methods.

Method(RGB+Depth)	PoseCNN [42] + ICP	MCN [19]	DenseFusion [40] (no refinement)	Ours
002_master_chef_can	95.8%	**96.2%**	95.2%	94.0%
003_cracker_box	91.8%	90.9 %	**92.5%**	88.7%
004_sugar_box	**98.2%**	95.3%	95.1%	96.0%
005_tomato_soup_can	94.5%	**97.5%**	93.7%	86.4%
006_mustard_bottle	**98.4%**	97.0%	95.9%	95.9%
007_tuna_fish_can	**97.1%**	95.1%	94.9%	96.0%
008_pudding_box	**97.9%**	94.5%	94.7%	93.5%
009_gelatin_box	**98.8%**	96.0%	95.8%	96.8%
010_potted_meat_can	92.8%	**96.7%**	90.1%	86.2%
011_banana	**96.9%**	94.4%	91.5%	96.3%
019_pitcher_base	**97.8%**	96.2%	94.6%	91.8%
021_bleach_cleanser	**96.8%**	95.4%	94.3%	92.0%
024_bowl	78.3%	82.0%	86.6%	**86.7%**
025_mug	95.1%	**96.8%**	95.5%	95.4%
035_power_drill	**98.0%**	93.1%	92.4%	95.2%
036_wood_block	90.5%	**93.6%**	85.5%	86.2%
037_scissors	92.2%	94.2%	**96.4%**	83.8%
040_large_marker	**97.2%**	95.4%	94.7%	96.8%
051_large_clamp	75.4%	93.3%	71.6%	**94.4%**
052_extra_large_clamp	65.3%	90.9%	69.0%	**92.3%**
061_foam_brick	**97.1%**	95.9%	92.4%	94.7%
Average	93.0%	**94.3 %**	91.2 %	92.4 %
Speed (fps)	< 0.1	< 10	20	**21**

LINEMOD dataset. In Table 3, we compare our method with state-of-the-art RGB and RGB-D methods. The numbers in brackets are the results without post-refinement. We use Frustum-P [29] as our baseline. We re-implement Frustum-P to regress 3D bounding box corners of the objects. From Table 3, we can see that our method outperforms the baseline by 5.4% in ADD accuracy and runs 2 times faster than the baseline method. Comparing to the second-best method [12] that using depth information, our method outperforms it by 2.4% in ADD accuracy and runs about 3 times faster than it. Although DPOD and PVNet are faster than our method, they only take RGB image as input. When using depth information, our method achieves

the fastest inference speed. In Figure 10, we provide a visual comparison of predict pose versus ground truth pose.

Object 6D pose estimation on YCB-Video:

Different from LIMEMOD dataset, in YCB-Video dataset, each frame may contain multiple target objects. Our method can also estimate 6D pose for multiple objects in fast speed. Table 4 compares our method with other state-of-the-art methods [42, 19, 40] on YCB-Video dataset under ADD-S AUC metric. From Table 4, we can see that our method achieves a comparable accuracy (92.4%) and is the
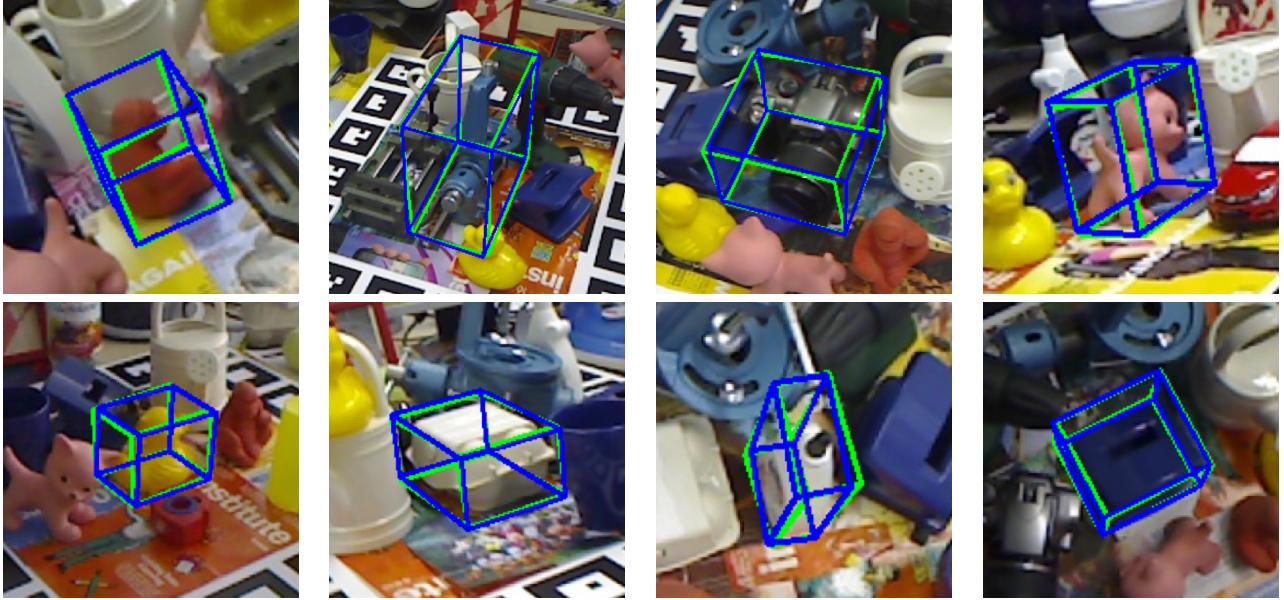
Figure 10. **Qualitative pose estimation results on LINEMOD dataset**. Green 3D bounding boxes denote ground truth. Blue 3D bounding boxes represent our results. Our results match ground truth well.
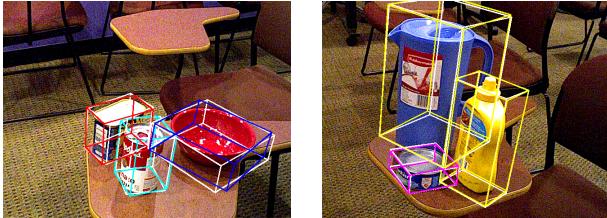

Figure 11. **Visualizing pose estimation results on YCB-Video**. White 3D bounding boxes are ground truth. Colorful 3D bounding boxes represent our results. For different objects, our prediction matches ground truth well.

fastest one (21fps) among all comparisons. In Figure 11, we also provide visualization results on this dataset.

4.7. Running time

For a single object, given a 480×640 RGB-D image, our method runs at 23fps on a PC environment (an Intel i7-4930K 3.4GHz CPU and one GTX 1080 Ti GPU). Specifically, the 2D detector takes $11ms$ for object location, and pose estimation part which includes translation localization and rotation localization takes $32ms$. The rotation residual estimator takes less than $1ms$.

5. Conclusion

In this paper, we propose a novel real-time 6D object pose estimation framework. Our G2L-Net decouples the object pose estimation into three sub-tasks: global localization, translation localization and rotation localization with embedding vector features. In the global localization, we use a 3D sphere to constrain the 3D search space into a more compact space than 3D frustum. Then we perform 3D segmentation and object translation estimation. We use the 3D segmentation mask and the estimated object translation to transfer the object points into local coordinate space. Since viewpoint information is more evident in this canonical space, our network can better capture the viewpoint information with our proposed point-wise embedding vector features. In addition, to fully utilize the viewpoint information, we add the rotation estimation estimator, which learns the residual between the estimated rotation and ground truth. In experiments, we demonstrate that our method achieves state-of-the-art performance in real-time.

Although our G2L-Net achieves state-of-the-art performance, there are some limitations to our framework. First, our G2L-Net relies on a robust 2D detector to detect the region of interest. Second, while our network exploits viewpoint information from the object point cloud, the texture information is not well adopted. In future work, we have a plan to overcome these limitations.

References

[1] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 2

[2] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. 2

[3] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015. 6

[4] Wei Chen, Jinming Duan, Hector Basevi, Hyung Jin Chang, and Ales Leonardis. Ponitposenet: Point pose network for robust 6d object pose estimation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2824–2833, 2020. 2, 5, 6

[5] Wei Chen, Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Appearance changes detection during tracking. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1821–1826. IEEE, 2016. 1

[6] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[8] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003. 2

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[10] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014. 1

[11] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. 2, 5, 6

[12] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *European Conference on Computer Vision*, pages 834–848. Springer, 2016. 2, 7

[13] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On evaluation of 6d object pose estimation. In *European Conference on Computer Vision*, pages 606–619. Springer, 2016. 6

[14] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[15] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017. 1, 2, 3

[16] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220. Springer, 2016. 1, 2

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[18] Vincent Lepetit, Pascal Fua, et al. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 1(1):1–89, 2005. 2

[19] Chi Li, Jin Bai, and Gregory D. Hager. A unified framework for multi-view multi-class object pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 2, 6, 7

[20] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2, 5

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[22] David G Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. IEEE, 1999. 2

[23] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2016. 1

[24] Eitan Marder-Eppstein. Project tango. In *ACM SIGGRAPH 2016 Real-Time Live!*, page 40. ACM, 2016. 1

[25] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 2

[26] Francesc Moreno-noguer. EPnP: An Accurate O(n) Solution to the PnP Problem. *Iccv*, 2007. 2

[27] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018. 1, 2, 4

[28] Sida Peng, Yuan Liu, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Pvnet: Pixel-wise voting network for 6dof pose estimation. *arXiv preprint arXiv:1812.11788*, 2018. 1, 2, 4, 5, 6, 7

[29] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *The IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), June 2018. 2, 3, 4, 6, 7

[30] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 3, 4, 5

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 4, 5

[32] Mahdi Rad and Vincent Lepetit. Bb8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017. 1, 2, 4

[33] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4663–4672, 2018. 1, 2, 4

[34] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 3, 5

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[36] Younghak Shin and Ilangko Balasingham. Comparison of hand-craft feature based svm and cnn based deep learning framework for automatic polyp classification. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3277–3280. IEEE, 2017. 2

[37] Alykhan Tejani. Latent-Class Hough Forests for 3D Object Detection and Pose Estimation of Rigid Objects. (November), 2014. 2

[38] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2, 4

[39] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. 1

[40] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. 2019. 2, 5, 7

[41] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015. 2

[42] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 1, 2, 3, 5, 6, 7

[43] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 2

[44] Yuan Yuan, Jia Wan, and Qi Wang. Congested scene classification via efficient unsupervised feature learning and density estimation. *Pattern Recognition*, 56:159–169, 2016. 2

[45] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1941–1950, 2019. 1, 7

[46] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2, 5

[47] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3936–3943. IEEE, 2014. 1