Efficient 3D Object Recognition via Geometric Information Preservation

Hongsen Liu[a,b,c], Yang Cong[a,b,*], Chenguang Yang[d], Yandong Tang[a,b]

[a]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, 110016, China
[b]Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang, 110016, China
[c]University of Chinese Academy of Sciences, 100049, China
[d]Bristol Robotics Laboratory, University of the West of England, Bristol, BS16 1QY, UK

Abstract

Accurate 3D object recognition and 6-DOF pose estimation have been pervasively applied to a variety of applications, such as unmanned warehouse, cooperative robots, and manufacturing industry. How to extract a robust and representative feature from the point clouds is an inevitable and important issue. In this paper, an unsupervised feature learning network is introduced to extract 3D keypoint features from point clouds directly, rather than transforming point clouds to voxel grids or projected RGB images, which saves computational time while preserving the object geometric information as well. Specifically, the proposed network features in a stacked point feature encoder, which can stack the local discriminative features within its neighborhoods to the original point-wise feature counterparts. The main framework consists of both offline training phase and online testing phase. In the offline training phase, the stacked point feature encoder is trained first and then generate feature database of all keypoints, which are sampled from synthetic point clouds of multiple model views. In the online testing phase, each feature extracted from the unknown testing scene is matched among the database by using the K-D tree voting strategy. Afterwards, the matching results are achieved by using the hypothesis & verification strategy. The proposed method is extensively evaluated on four public datasets and the results show that ours deliver comparable or even superior performances than the state-of-the-arts in terms of F1-score, Average of the 3D distance (ADD) and Recognition rate.

Keywords: stacked 3D feature encoder, 3D object recognition, 6-DOF pose estimation, geometric information preservation.

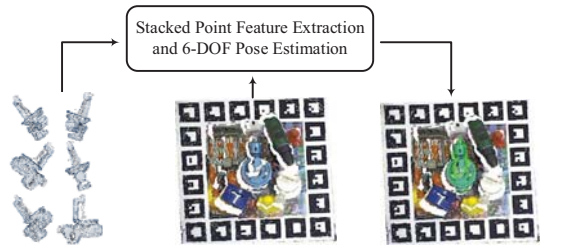
1. Introduction

3D object recognition and 6-DOF pose estimation are of great significance to many practical applications, e.g., unmanned warehouse, cooperative robots, and manufacturing industry [1, 2, 3, 4, 5, 6]. However, this is still a challenge due to the diverse attributes of objects, which results in the limited discrimination of handcrafted feature descriptors. In this paper, we focus on 3D object recognition and 6-DOF pose estimation of objects with texture-less or surface-smooth. The lack of interesting points make them intractable to extract robust descriptors. Most classical 2D [7, 8, 9, 10, 11, 12] and 3D [13, 14, 15, 16, 17, 18, 19, 20] local feature-based methods cannot perform well on such objects for the sake of weak keypoint descriptors. Template feature-based methods [21, 22, 23, 24] can achieve better recognition results for texture-less objects by extracting global features, but their performances could be deteriorated under heavy occlusion as well. While most existing patch feature-based methods [25, 26, 27] could extract various features from RGB-D images to solve such problems, they inevitably involve mapping the 3D real world into 2D image space and induce the loss of 3D spatial information accordingly. In comparison, [28] tries to handle this problem by converting the point clouds into regular voxel grids and extracting 3D patch features accordingly. However, this introduces unnecessary computational cost. Recently, a variety of CNN-based methods [29, 30, 31] have tried to learn features from large amounts of identically distributed training data, which depends heavily on large volume of data. The foregoing issues motivate us to extract a more efficient 3D features representation from raw point clouds directly and estimate the 6-DOF pose of the 3D objects depending on the hypotheses generation and verification strategy as shown in Fig. 1.

Generally, there are three main stages involved for 3D object recognition: 1) **3D feature extraction:** Unlike the existing methods [25, 26, 27, 28] and motivated by [32][33], we design a 3D feature encoder, which enables point interaction within a local neighborhood sphere, by stacking the locally sphere-wise aggregated feature on point-wise features. Stacking multiple encoder layers allows further learning deep sphere-wise features and point-wise features. Afterwards, the point-wise features are sequentially sent to the structurally similar decoder to reconstruct the input points. The output sphere-wise features of the middle layer are used for characterizing 3D keypoints descriptor. The mean squared error (MSE) is adopted as the

☆This work is supported by Nature Science Foundation of China under Grant (61722311, U1613214, 61821005, 61533015) and CAS-Youth Innovation Promotion Association Scholarship (2012163).

*Corresponding author
Email addresses: liuhongsen@sia.cn (Hongsen Liu), congyang81@gmail.com (Yang Cong), cyang@ieee.org (Chenguang Yang), ytang@sia.cn (Yandong Tang)



(A) The raw point clouds of model samples (B) The raw point clouds of real scenario (C) The result of 6-DOF pose estimation

Figure 1: The illustration of the proposed method that directly operates on the raw point clouds (no need for any structure conversion) and produces the 6-DOF object pose. (A) the raw point clouds of multi-view model samples. (B) the raw point clouds of real scenario. (C) the result of the 3D object recognition and 6-DOF pose estimation.

reconstruction loss. 2) **Similarity Search:** Using the proposed encoder, we create a feature database of point clouds spheres sampled from synthetic model views where each feature holds a local 6D pose annotation. During testing, we sample point clouds spheres from the raw point clouds of the testing scene, and extract the corresponding features afterwards. Then the features are matched against the database via an efficient K-D Tree search. The matching returns a collection of candidate votes, which are cast to generate hypotheses. 3) **Hypotheses Verification:** Casting votes can lead to a crowded voting space that requires further refinement. In order to keep detection computationally feasible, we first set a voting threshold to reject the false hypotheses with low score, and then refine the pose estimation via the Iterative Closest Point (ICP) [34]. At last, the non-maxima suppression is used to obtain the final refined results.

In summary, our main contributions are:

- We propose a simple but effective unsupervised feature encoder for point-clouds-based local 3D feature extraction, which avoids unnecessary computational cost and geometric information loss caused by data conversions.
- Benefiting from both our unsupervised feature encoder and the efficient verification strategy, we present an effective framework to recognize the 3D objects and estimate the corresponding 6D pose from the clutter environment accordingly.
- Extensive experiments on four public datasets are performed to validate the effectiveness of our method, where ours outperforms the state-of-the-arts.

2. Related Works

In this section, an overview of the existing works on 3D object recognition and 6-DOF pose estimation are presented. Here, these methods are classified into four categories as below:

Local feature-based methods: Earlier techniques extract texture-based local features from the 2D RGB image and then back project to 3D space [7, 8, 9, 10, 11]. These methods perform well on 3D objects with rich texture surface. However, many objects in our daily life are texture-less, especially in the industrial environment. Recently, the RGB-D sensors with low cost and acceptable accuracy, e.g., Kinect, become popular. Several point clouds based features are designed depending on various local 3D surface [13, 14, 15, 16, 17, 18, 19] without projecting feature points from 2D image to 3D space. The basic assumption of these methods is that the surface normal of the corresponding objects have rich variations. These methods may cause ambiguity for planar or self-symmetric objects due to various repeating local surfaces [20].

Template feature-based methods: Template features are achieved from the scanning model under multi-view, and the optimal matching is searched by sliding windows, which are commonly robust to texture-less objects. Line2D [22] merely employs the image contours to denote a 3D object with a limited set of templates and achieves efficient matching by linearizing the memory for parallelization. As an improvement, LineMod [21] performs robust 3D object detection by embedding quantized image contours and normal orientations on RGB-D images. However, these methods are being scaled linearly with the number of templates. To this end, R. Rioscabrera et al. [23] optimizes the matching via a cascaded classification scheme and gets 10 times speedup. W. Kehl et al. [24] proposes an improvement approach based on LineMod template features via hashing matching.

Patch feature-based methods: Recently, some feature representation methods based on local RGB-D patches are proposed, e.g., A. Tejani et al.[25] employ a manually designed feature inspired by [21] along with random forests based voting schemes for the estimation of the 6-DOF pose. A. Doumanoglou et al. [26] learn patch features via an unsupervised deep Sparse Autoencoder instead of manually designed. Given that the training classifier requires to learn the background as a negative class, the method is normally constrained as dataset-specific. Instead, W. Kehl et al. [27] train a Convolution Autoencoder to extract patch features and estimate 6-DOF pose based on K-nn search, which gives better performance. Liu et al. [28] present a 3D Voxel Autoencoder by converting the point clouds into voxel grids for fully using the 3D spatial structure information.

Trainable CNN-based methods: Even various deep learning based methods have justified their performance on object detection, classification and segmentation [32, 35, 36], these methods remain unable to accurately yield the 6-DOF object pose as a regression problem [37][38]. For 3D object recognition, the frequently-used strategy is first to segment and detect object on the RGB-D images and then back project them to 3D space to acquire rough location. Eventually, the Iterative closet point (ICP) is employed to refine the 6-DOF pose based on the approximate models [39, 40, 41]. Recently, some end-to-end methods are proposed [29, 30, 31] to predict the 2D bounding box in the image and compute the 6D pose using a PnP algorithm [42]. These methods inevitably rely on large amounts of identically distributed training data, which acquire extra cost in collecting these training datasets.

(A) Raw Point Clouds (B) Farthest Point Sampling (C) Random Sampling

Figure 3: The effect of different sampling methods. (A) The raw input point clouds. (B) The effect of the Farthest Point Sampling method (FPS). (C) The effect of the Random Sampling method (RS).

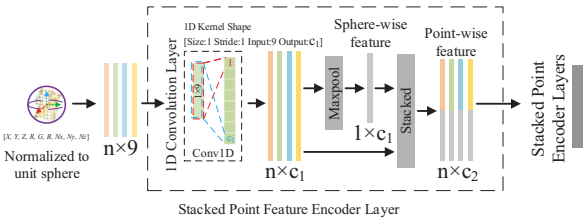


Figure 4: The architecture of the Stacked Point Feature Encoder Layer, which consists of a **1D Convolution Layer** for feature transformation, a **Maxpool Layer** for feature aggregation and a **Stacked Feature Layer** for feature concatenation. The following stacked encoder layers yield deep features layer by layer.

3. Methodology

In this section, the framework of our 3D object recognition and 6-DOF pose estimation method is introduced. The problem can be summarized as given a 3D model $\mathcal{M}$ of a specific object and a testing scene $\mathcal{S}$, which needs to estimate the 6-DOF pose of all the $\mathcal{M}$ existing in $\mathcal{S}$ at one time. Fig. 2 demonstrates the framework of our method. Generally, it consists of two main phases: the **Offline Training Phase** and **Online Testing Phase**. 1) In the offline training phase, the stacked feature encoder is trained first and then generate feature database of all keypoints, which are sampled from synthetic point clouds of multiple model views. Each feature holds a 6D pose annotation. 2) In the online testing phase, the features of the unknown scene are matched among the database by using the K-D tree searching. The matching results cast a collection of hypotheses that are refined via a verification strategy.

3.1. Sphere-wise data sampling and grouping

Typically, the point clouds of a low-cost depth sensor is composed of more than 30k points. Due to the density of the point clouds varies significantly in the whole space, operating directly on all points not only increases the memory/efficiency burden, but also disturb the detection accuracy. To this end, the Farthest Point Sampling (FPS) [43] method is adopted to sample keypoints as shown in Fig. 3, where FPS covers the entire surface shape better comparing with Random Sampling (RS) [44]. The sampling performance is compared in the experimental part. For a given point clouds and a support radius r, we first sample a fixed number t of keypoints $k_p = [p_1, ..., p_t]$, and then group the sphere-wise local point sets $l_p = [p_1, ..., p_n]$ with n points of r-nearest neighboring search radius around each keypoint. The model views and scenes use different sampling number of t. Specifically, for spheres with a point number more than and less than n, FPS and Repeated RS methods are used to sample n points respectively. In our case, the keypoints sampling number $t_m = 512$ for model views and $t_s = 4096$ for testing scenes. The local points sampling number $n = 256$ with a support radius r, which is set as $\frac{1}{3}$ of the shortest edge of the 3D model bounding box.

3.2. Stacked Point Feature Encoder

In this subsection, the process will be elaborated, which extracts the sphere-wise feature descriptor via the encoder layers. The offline training phase of Fig. 2 illustrates the hierarchical feature encoding process. The architecture details are described in the following paragraph as shown in Fig. 4.

Denote $S = \{l_{p_i} = [X_i, Y_i, Z_i, R_i, G_i, B_i, N_i]^T \in \mathbb{R}^9\}_{i=1...n}$ as a local sphere containing n points, where l_{p_i} contains XYZ coordinates and RGB values for the i-th point and N_i is the calculated normal direction, which consists of N_x, N_y, N_z. It needs to be explained that the normals are computed before the sampling and grouping of the spheres so as to ensure the continuity of the surface shape of the input point clouds. Firstly, each sphere is normalized, including centralization and normalization of coordinates, normalization of colors and unitization of normal vectors. Next, the normalized spheres with the size as $[b \times n \times 9]$ are transformed through the fully connected layers (FC) into aggregated feature space with the size as $[b \times n \times c_1]$, where b represents the training batch size (in our case, $b = 4$), n represents the number of points, 9 is the feature dimension of the initial point-wise features, c_1 is the dimension of transformed features. Specifically, in order to process 3D point clouds more efficiently, 1D convolution with $[1]$ kernel size and $[1]$ kernel stride are used to replace the fully connected layer to transform the input data into $[b \times n \times c_1]$. To extract sphere-wise features, which represent the global features of the local 3D points sphere, the max pooling layer is used as a symmetric function that aggregates information from all the point-wise features to achieve sphere-wise features with the size as $[b \times 1 \times c_1]$. The use of max pooling layer is not only to aggregate the spatial dimension of features, but also to ensure the permutations invariance of the unordered 3D points. Afterwards, the sphere-wise features are fed back to per point-wise features by **stacking** the sphere-wise features on each of the point-wise features with size as $[b \times n \times c_2]$, where $c_2 = 2 \times c_1$. Through this way, the new stacked point-wise features are able to preserve both the local and global information.

We use $[c^i_{in}, c^i_{out}]$ to represent the I/O of i-th encoder layer that transforms input features of dimension c_{in} into output features of dimension c_{out}. For the first encoder layer, c^1_{in} is 9 that represents there are 9 dimensions of the raw points clouds attributes. For each encoder layer, there are two output features, i.e., the sphere-wise feature c^{sphere}_{out} and the new stacked point-wise feature $c^{stacked}_{out}$. Only $c^{stacked}_{out}$ could be transformed through next encoder layer into deeper sphere-wise and point-wise features; the c^{sphere}_{out} could only be extracted from the end of the encoder layers.

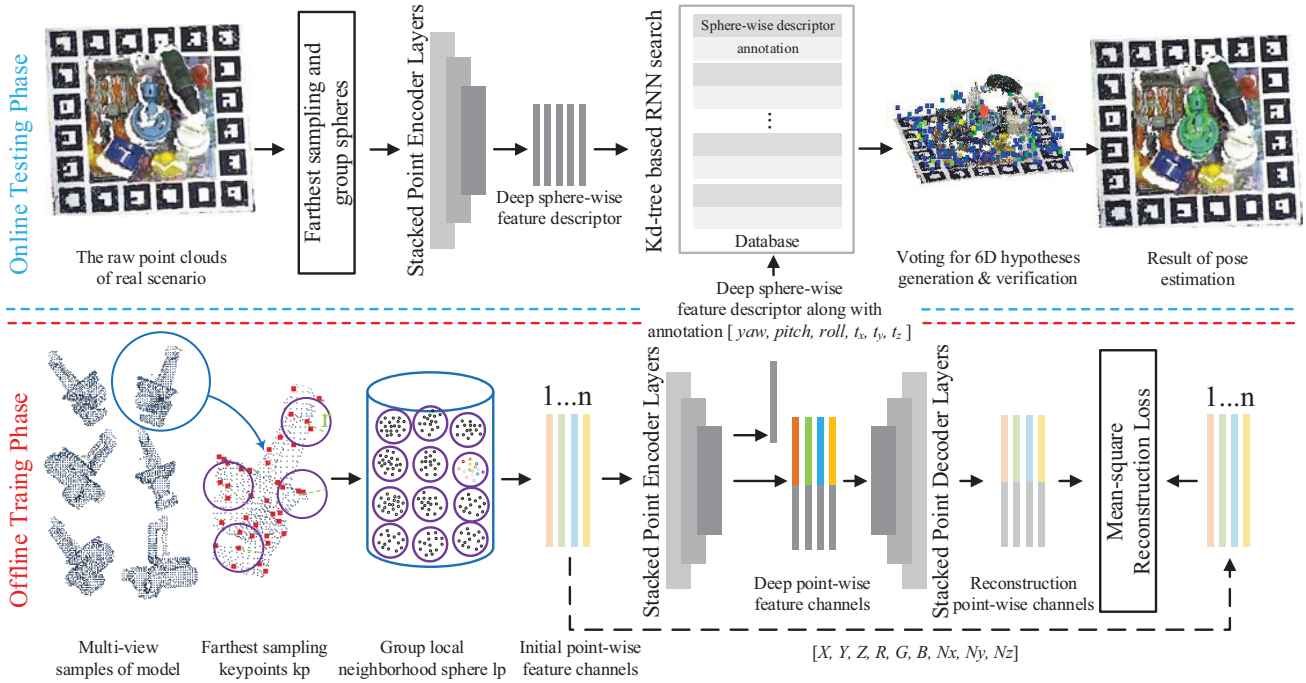


Figure 2: The framework of our 3D object recognition and 6-DOF pose estimation method. 1) In the offline training phase, the stacked point feature encoder is trained first and then generate feature database of all keypoints, which are sampled from synthetic point clouds of multiple model views. 2) In the online testing phase, the features from the unknown real scene are matched among the database by using the K-D tree searching method. Afterwards, the matching results are achieved by using the hypothesis & verification strategy.

3.3. The Offline Training Network

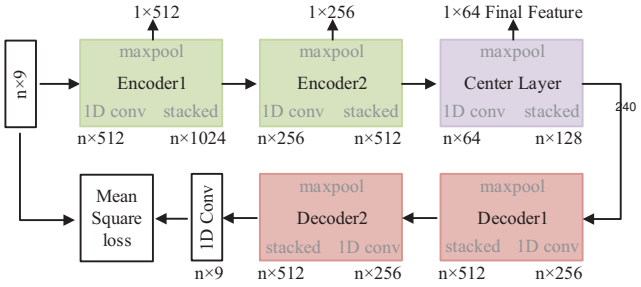


Figure 5: The complete framework and parameter configuration of the Autoencoder. It has three stacked encoder layers and three symmetric decoder layers. The center layer shared between the encoder and decoder. The output of the maxpool in the center layer is used as the final sphere-wise features. The mean-squared error (MSE) is used as the reconstruction loss.

For the offline training, an unsupervised feature encoder-decoder network is presented as shown in Fig. 5. The decoders have the same structure to the encoders, which are used to reconstruct the input point clouds features. Several papers have proved that the features extracted by unsupervised reconstruction have effective performances [26, 27, 28]. In our case, the network has three stacked encoder layers and three symmetric decoder layers. The complete network structure and parameter configuration are show in Fig. 5, where the center layer is shared by the encoders and decoders. The output features of the maxpool in the center layer are used as the final sphere-wise features (here the feature dimension is 64). Each encoder layer is composed of a 1D Convolution Layer, a maxpool layer and a stacked feature layer. The mean-squared error (MSE) is used as the reconstruction loss. For a visual impression of the reconstruction quality, the results of two random sampled spheres from testing scene are shown in Fig. 6, where both the reconstruction results of the $[XYZRGB]$ and the $[N_xN_yN_z]$ are shown almost similar. The $[N_xN_yN_z]$ shown is converted to the corresponding values within HSV color space.

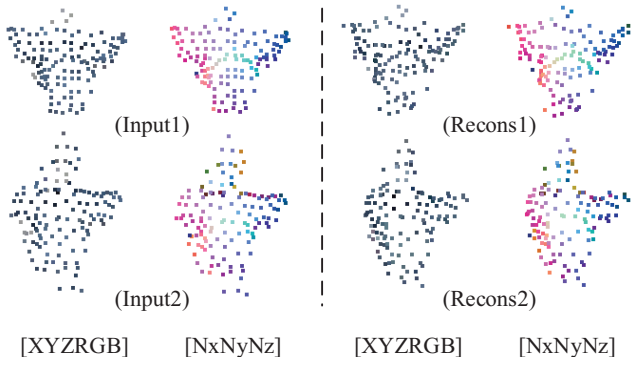


Figure 6: The visual impression of the reconstruction quality for two sampled spheres from testing scene, where both the reconstruction results of the $[XYZRGB]$ and the normal $[N_xN_yN_z]$ shown are almost similar. The normal shown is converted to HSV color.

After the completion of the training, the features of all 3D

Figure 7: 6D cast voting space with different τ values. The projected voting centroids are colored following their voting score, which is mapped by an HSV colormap. The Saturation (S) and Value (V) are fixed to maximum and mapping the score to the range of Hue (250→360) corresponding to (blue→red). It can be seen clearly that with the increase of τ, most false candidates are filtered out and the true candidates around the object centroid with a higher score is remained.

point clouds spheres y sampled from synthetic model views are used to create a feature database, where each feature holds a local 6D pose annotation $[yaw, pitch, roll, t_x, t_y, t_z] \in \mathbb{R}^6$. In our case, the annotation $[yaw, pitch, roll]$ represents the pose transformation of each model view under the model's coordinate frame and the annotation $[t_x, t_y, t_z]$ is the offset from the sphere center (x, y, z) to the model center $(0, 0, 0)$, where $(t_x = 0 - x, t_y = 0 - y, t_z = 0 - z)$. In this case, the model's coordinate frame is built on the model center.

3.4. The Online Hypotheses Generation and Verification

In this subsection, we intend to generate hypotheses of the candidate 6-DOF pose and refine them. For a given testing scene, the features of all 3D point clouds spheres sampled from the scene are exploited to find the pairwise correspondences from the feature database for 6-DOF pose estimation. Since the size of the training feature database is huge to cover all sampling views of the object model, the K-D Tree searching method is used to search the optimal correspondences for each sphere-wise feature efficiently. During testing, we sample the keypoint $s = (s_x, s_y, s_z)$ with associated sphere x from unknown scene first and then compute its feature $f(x)$ and search the nearest spheres $y_1, ..., y_m$ from database. Each neighbor casts a global vote $v(s, y) \Rightarrow (t_x + s_x, t_y + s_y, t_z + s_z, yaw, pitch, roll)$ with an associated weight as $w(v) = e^{-||f(x)-f(y)||}$ depending on the feature distance. This method is flexible enough to alter the number of possible vote candidates by tuning the search radius R, and the searched candidates will only be voted if they hold a similar feature distance. This reduces the impact of noise sensitivity on the method and is more easily constrained by the number of votes. For different objects, the value of R needs to be adjusted manually according to the actual feature output. We adjust this value by observing and sampling several most similar feature distances.

Due to noise sensitivity and feature ambiguity, the valid voting candidates can lead to a crowded voting space, which requires further refinement to make it computationally feasible. For the crowed 6-DOF voting space $[yaw, pitch, roll, t_x, t_y, t_z]$, we first group it into equal voxel grids and add cumulative candidate weight $w(v)$ to each grid. Then, by computing the weight histogram, a dynamic threshold $\tau \in [0, 1]$ is designed to reject 90% grids (in our case, $\tau = 0.9$) with low weight. As τ increases, more false candidates are filtered out. The filter results of different τ are shown in Fig. 7.

For the remaining candidates, each is refined by the Iterative Closest Point (ICP) to refine the transformation and calculate a matching score ε, where $\varepsilon > 0.8$ (in our case) means the overlap ratio between the object model and the scene surface. The non-maximum suppression is followed to find the local maximums depending on the voting weight, where each generates the final hypotheses $[R_c, T_c]$.

4. Experiments

In this section, we compare our method with several representative 3D object recognition methods, such as LineMod [21], SSD-6D [29], AE-HF [26], Spin image [15]. The experiments are evaluated on four publicly available datasets (the LC-HF dataset [25], the LineMod dataset [21], the AE-HF bin-picking dataset [26]) and the UWA dataset [20], which contain multiple objects with various interferences, e.g., occlusion, illumination change, cluttered background and no-colors. For the evaluation metric, we first adopt the F1-score defined in LC-HF [25]. The estimation is deemed correct if the mean distance m between the true pose $[R, T]$ of model $\mathcal{M}$ vertices and those estimated given the pose $[R_c, T_c]$ is less than λ (here is 15%) of the objects diameter [25]. Secondly, we adopt the Average of the 3D distance (ADD) metric defined in [21]. We take a pose estimate to be correct if the mean distance m is less than λ (here is 10%) of the object diameter [21]. Specifically, for rotationally symmetric objects, the mean distance is computed as Eq. 1:

$$m = avg \sum_{x \in M} \min_{M} ||(Rx + T) - (R_c x + T_c)||. \quad (1)$$

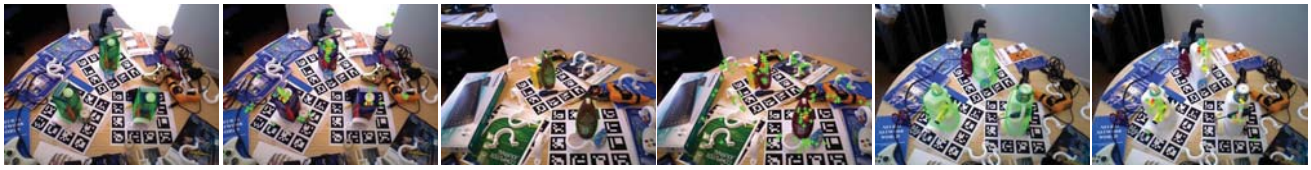
Thirdly, we adopt the Recognition rate under different Occlusion rate defined in [20]. The occlusion rate is defined as Eq. 2:

$$occlution = 1 - \frac{\text{model visible surface area in scene}}{\text{total model surface area}}. \quad (2)$$

4.1. Results On the LC-HF Dataset [25][27]

This dataset [25][27] contains 6 objects and each testing image has 2-3 same targets, which are placed on a cluttered round table. Each target is associated with the 3D mesh model and assigned a ground-truth $[R, T]$ matrix.

(A1) coffee (A2) coffee (B1) camera (B2) camera (C1) joystick (C2) joystick



(D1) juice (D2) juice (E1) shampoo (E2) shampoo (F1) milk (F2) milk

Figure 8: Some demo results and the voting maps of our method on the LC-HF dataset [25][27], where the recognition results are shown as the green transformed model overlaid the estimated location (A1-F1). Areas with the high voting score are also clearly clustered in the object centroid (A2-F2).

Table 1: The statistic results of average F1-score of the re-annotated LC-HF datasets [25][27] in comparison with LineMod [21], LC-HF [25], ConvAE [27], VoxelAE [28] and SSD-6D [29]. The best results and the second best results are represented with red and blue fonts.

Objects	[21]	[25]	[27]	[28]	[29]	OURS
Coffee	0.942	0.891	0.972	0.977	0.983	0.996
Shampoo	0.922	0.792	0.910	0.857	0.892	0.931
Joystick	0.846	0.549	0.892	0.739	0.997	0.958
Camera	0.589	0.394	0.383	0.681	0.741	0.949
Juice	0.595	0.883	0.866	0.866	0.919	0.970
Milk	0.558	0.397	0.463	0.493	0.780	0.831
Average	0.740	0.651	0.747	0.768	0.885	0.939

The statistic recognition results are shown in Tab. 1, where the overall average F1-score of our method is 93.9%, in comparison with LineMod (74.0%) [21], LC-HF (65.1%) [25], ConvAE (74.7%) [27], VoxelAE (76.8%) [28] and SSD-6D (88.5%) [29] respectively. Here, [29] counts a detection to be correct when the IoU score of a predicted bounding box with the groundtruth box is higher than 0.5. It is evident that LineMOD fares very well on most sequences with low occlusion (e.g., coffee, shampoo and joystick). It only shows problem where objects sometimes are partially visible (e.g., milk) or where the objects are confused by the background (e.g., camera and juice). LC-HF improves the inherent robustness to foreground occlusions by using patch representation, but its overall performance is not fully exploited due to the complex parameter adjustment of hough voting strategy. ConvAE improves the overall performance by using deep RGBD patch representation and combining simple and valid voting strategy. Due to the loss of geometric information in the process of projection from 3D to 2D for RGBD data, it decreases the precision by the cluttered environment. Although VoxelAE deals 3D points, the process of voxelization also leads to the loss of geometry information and introduces unnecessary memory expenditure. SSD-6D is a supervised CNN-based method, which requires a lot of scene distribution data and predicts the object 2D bounding box and a rough estimate of the objects orientation in RGB image. The final 6-DOF estimation is calculated by several stages of refinement and verification. This process will magnify the error of 2D prediction and then affect the 6-DOF pose estimation. In contrast to most existing methods, ours deals 3D points without any data conversion (e.g., voxelization), which makes full use of geometric information and does not dataset specific.

Ours also give rise to a good result for the camera model and the joystick model, where the camera model is small in size and looks similar to the background and the joystick model has thin and thick part. Especially for the milk model, while this model is texture-less, smooth-surface and contains other distracting objects on it, ours shows better results. It is evident that the learned features can handle various object appearance. Fig. 8 demonstrates our recognition results on the LC-HF dataset, where ours can accurately estimate the objects pose with amounts of clutter. We use the metric Eq. 1 when evaluating the pose accuracy for the rotationally invariant objects, *coffee, shampoo, camera and juice*.

4.2. Results On the LineMod Datasets [21]

This dataset [21] contains 15 objects and each testing image has only one target, which is placed on a desk with heavy amounts of occlusion and clutter. Each object is associated with the 3D mesh model and assigned a ground-truth $[R, T]$ matrix in more than 1k testing images. Since the mesh models of the bowl and the cup are missing, we test the other 13 models as well as [25]. The statistic recognition results of average F1-score and ADD metric are shown in Tab. 2.

Firstly, we also give the statistic recognition results of average F1-score in comparison with LineMod [21], LC-HF [25], ConvAE [27] and SSD-6D [29] respectively, where the average F1-score of our proposed method is 93.4% outperforming the state-of-the-arts, e.g., the second best one ConvAE (92.88%), the third best one SSD-6D (88.50%) and so on. Here, [29] counts a detection to be correct when the IoU score of a predicted bounding box with the groundtruth box is higher than 0.5. Specially, ours yields the best results for 3 out of all 13

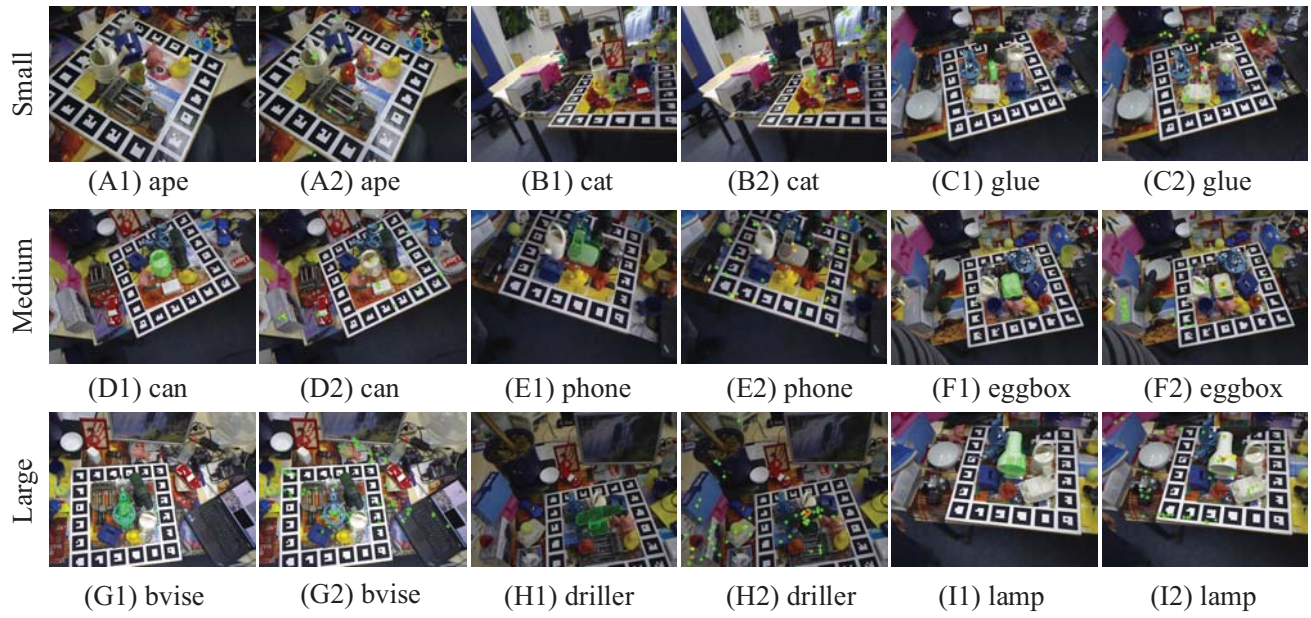


Figure 9: Some demo results and the voting maps of our method on the LineMod dataset [21], where the recognition results are shown as the green transformed model overlaid the estimated location (A1-I1). We roughly divided the objects into three scales: the small size (top row), the medium size (middle row) and the large size (bottom row). Areas with the high voting score are also clearly clustered in the object centroid (A2-I2).

Table 2: The statistic results of average F1-score and ADD metric on the LineMod dataset [21] in comparison with LineMod [21], LC-HF [25], ConvAE [27], SSD-6D [29], BB8 [30] and Seamless [31]. The best results and the second best results are expressed in red and blue fonts. The ADD metric of Seamless [31] is acquired without refinement and for reference only.

Metric	F1-Score ($\lambda = 0.15$)					ADD ($\lambda = 0.1$)		
Objects	[21]	[25]	[27]	[29]	OURS	[31]	[30]	OURS
ape	53.3	85.5	98.1	76.3	93.5	21.62	40.4	68.6
bvise	84.6	96.1	94.8	97.1	95.2	81.80	91.8	78.9
cam	64.0	71.8	93.4	92.2	96.3	36.57	55.7	87.2
can	51.2	70.9	82.6	93.1	94.3	68.80	64.1	83.0
cat	65.6	88.8	98.1	89.3	94.8	41.82	62.6	84.4
driller	69.1	90.5	96.5	97.8	93.2	63.51	74.4	80.5
duck	58.0	90.7	97.9	80.0	96.2	27.23	44.3	81.3
eggb	86.0	74.0	100	93.6	96.0	69.58	57.8	84.4
glue	43.8	67.8	74.1	76.3	88.2	80.02	41.2	62.8
holep	51.6	87.5	97.9	71.6	90.8	42.63	67.2	77.8
iron	68.3	73.5	91.0	98.2	92.3	74.97	84.7	81.3
lamp	67.5	92.1	98.2	93.0	95.7	71.11	76.5	85.8
phone	56.3	72.8	84.9	92.4	88.4	47.74	54.0	74.5
Average	83.44	81.69	92.88	88.50	93.40	55.95	62.7	79.3

models and all the others are getting the second best results. Although the results of most models are not the best, ours are the most balanced by integrating 3D geometric information.

Secondly, we give the statistic recognition results of ADD metric, where the ADD metric of our proposed method is 79.3%, in comparison with more CNN-based method, BB8 (62.7%) [30] and Seamless (55.95% without refinement for reference only) [31]. BB8 is made of one CNN to coarsely segment the object and another to predict the projections of the objects 3D bounding box given the segmentation in 2D image, which are then used to compute the 6D pose using a PnP [30] algorithm and further pose refinement. Unlike BB8 methods, which require multi-stage of processing, Seamless is a single-shot method that takes the image as input and directly detects the 2D projections of the 3D bounding box vertices. The objects 6D pose is then estimated using the PnP algorithm without any refinement. For these methods, the main problem is to predict in the 2D image space, and then get the 6-DOF estimation of the objects in the 3D space by spatial mapping. The error of 2D prediction is further magnified in the process of spatial mapping. In contrast to these methods, the overall ADD metric of ours shows better results.

Fig. 9 demonstrates our recognition results on the LineMod dataset, where ours can accurately estimate the objects pose with heavy amounts of occlusion, scale change and clutter. We use the metric Eq. 1 when evaluating the pose accuracy for the rotationally invariant objects, *glue, eggbox* as well as [21].

Table 3: The statistic results of average F1-score on the AE-HF bin-picking dataset [26]. The best results and the second best results are expressed in red and blue fonts.

Objects	LC-HF[25]	AE-HF [26]	OURS
CoffeeCup	0.314	0.361	0.469
Juice	0.248	0.290	0.412

4.3. Results On the AE-HF Bin-picking Datasets [26]

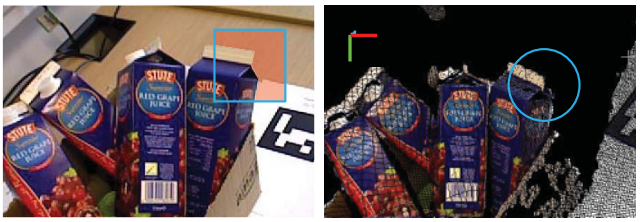
This dataset [26] constructs two bin-picking scenarios, where each contains multiple same targets, 16 for bin-coffee scenario and 5 for bin-juice scenario. Different from the two household datasets LC-HF datset [25] and LineMod dataset [21] with ob-

(A1) bin-coffee (A2) bin-coffee (B1) bin-juice (B2) bin-juice

Figure 10: Some demo results and the voting maps of our method on the AE-HF bin-picking dataset [26], where the recognition results are shown as the green transformed model overlaid the estimated location (A1, B1). As can be seen that ours can accurately recognize the objects pose even with heavy amounts of self-occlusion (A2, B2).

jects placed separately, the serious aliasing and self-occlusion between multiple objects make the dataset more challenging.

The statistic recognition results of average F1-Score are shown in Tab. 3, where our method outperforms LC-HF [25] and AE-HF [26] with about 30% improvement. For the most existing RGB-D patch-based methods (e.g., LC-HF, AE-HF), the features usually contain more interferences due to the local patches that inevitably cover part of the background, as shown in Fig. 11 (A), where the red part is the background. In contrast, our method extracts features from the local point clouds spheres, which contains relatively little interference because of the definite spatial location. As shown in Fig. 11 (B), the local sphere contains only the surface of the target without any background. Fig. 10 demonstrates our recognition results on the AE-HF bin-picking dataset, where the targets have accurate voting centers. As a result, ours are able to accurately recognize the object pose even with heavy amounts of self-occlusion.



(A) RGB-D Patch (B) Point Clouds Sphere

Figure 11: The demonstration of the local RGB-D patch and the local point clouds sphere. (A) The RGB-D patch contains part of the background. (B) The point clouds sphere contains only the surface of the target without any background.

4.4. Results On the UWA Datasets

The UWA dataset [20] contains 5 objects and 50 real testing scenes, where each scene is scanned by a Minolta VIVID 910 laser scanner by placing 4-5 different models in the scene without color information. Exclude the Rhino model due to it contains large holes and cannot be recognized well, other objects are associated with the 3D mesh model and assigned a ground-truth $[R, T]$ matrix.

Specifically, we use $[X, Y, Z, N_x, N_y, N_z]$ without RGB information as the only input of the feature encoders. The statistic recognition results are shown in Tab. 4, when the object occlusion rate is between 0 and 84%, the overall average recognition rate of our method is the second best 97.7%, in comparison with Tensor (96.6%) [15], Spin image (87.8%) [15], EM (97.5%) [16] and RoPS (98.8%) [17] respectively.

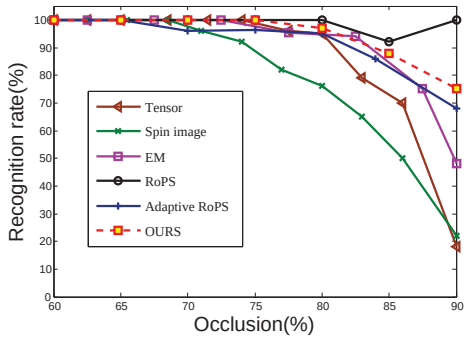
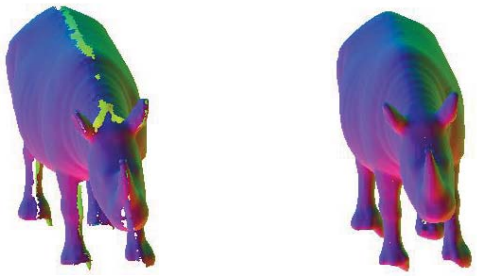


Figure 14: The recognition rate against different occlusion on the UWA dataset without Rhino model.

Table 4: The statistic results of average recognition rate on the UWA dataset [20] without Rhino model, where the the occlusion rate is between 0 and 84%. The best results and the second best results are expressed in red and blue fonts.

The occlusion rate is between 0 and 84%					
Methods	Tensor [15]	Spin image[15]	EM [16]	RoPS [17]	OURS
Avg	96.6	87.8	97.5	98.8	97.7

Although our method is designed to be better at dealing with scenes with color information, it is still applicable for such no-color dataset. Compared with other special methods, ours achieves 97.7% recognition rate when the occlusion rate is between 0 and 84%, which is close to the optimal RoPS method. As shown in Fig. 14, when the occlusion rate exceeds 84%, the recognition rate of ours is still better than most of the state-of-the-arts. In addition, we manually patched the Rhino model as shown in Fig. 12 and labeled all 23 scenes containing Rhino using ICP method. The overall average recognition rate of our method is 92.3%, when the occlusion rate of Rhino model



(A) Rhino model with holes (B) Rhino model after patched

Figure 12: Rhino model patched with Geomagic Studio. We convert the normal to HSV color space for visualization.

(A1) The first sample scene (A2) Our recognition result (B1) The second sample scene (B2) Our recognition result

Figure 13: Two sample scenes and our recognition results on the UWA dataset, the correctly recognized objects are covered by their 3D models. We convert the normal to HSV color space for visualization.

is between 0 and 74%. The overall average recognition rate is 52.2% in the whole occlusion range. Fig. 13 demonstrates our recognition results on two sample scenes on the UWA Dataset, where ours can also accurately recognize objects without color.

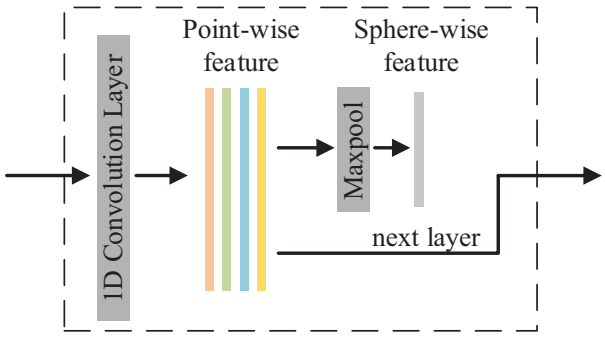
4.5. Results on the Average Running Time

We present the average time consumption of our method on the LC-HF dataset [25] and the comparison with ConvAE [27]. As shown in Tab. 5, we record the corresponding time of various stages, including Data sampling, Feature extraction, Hypotheses generation and Refinement. The total average running time for our method is 774ms, which is close to the 670ms usage of ConvAE. Specifically, we use different platforms for different phases and record the time separately.

Table 5: Comparisons of the average runtime of ours and ConvAE [27] on the LC-HF dataset [25].

Stage	ConvAE (ms)	OURS (ms)
Data Sampling	0.03	12.5
Feature Extraction	477.3	47.4
Hypothesis Generation	63	186.2
Refinement	130.5	528.4
Online testing	670.8	774.5

In our case, it contains data sampling (keypoints k_p and local points l_p sampling) and spheres grouping. We use GPU for parallel acceleration, which can sample and group the raw point clouds of more than 30K into local spheres of 4K with about 12ms. The feature extraction use about 47ms, which is less than ConvAE (477ms). We implement this phase on the Tensorflow framework based on Pointnet model [33] with a NVIDIA TITAN XP (12GB RAM). The Hypotheses generation and Refinement consume more time than ConvAE due to operations on point clouds directly. Both the Hypotheses generation and Refinement are executed on a standard PC with a general Intel CPU (i5-3470) at 3.20GHz, 16GB RAM.



Non-stacked Point Feature Encoder Layer

Figure 15: The architecture of the Non-stacked Point Feature Encoder Layer, which generates the features without stacked operations.

4.6. Comparisons on the Different Network Architecture

In order to justify the effectiveness of the stacked point feature encoder, we compare it with another architecture on the

AE-HF bin-picking dataset [26].

As shown in Fig. 15, we extract the sphere-wise features and point-wise features for this architecture without stacked operations. Given a normalized sphere with initial point-wise features, we transform it through the 1D convolution layer into a aggregated feature space and use maxpool layer to further aggregate information from the new point-wise features to sphere-wise features. The statistic recognition results of average F1-Score are shown in Fig. 16, where we compare two architectures with different number of encoder layers. We can see intuitively that for both the two different architectures, by increasing the number of encoder layers from 1∼3 (include the center layer), the recognition results increase obviously. By fixing the number of the encoder layers, the recognition result of the stacked point feature encoder in our case is more effective than the non-stacked point feature encoder, which can be attributed to the fact that the stacked operation augments the local sphere features by concatenating the locally aggregated features layer by layer.

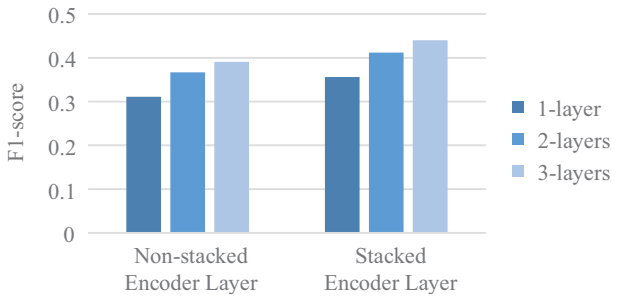


Figure 16: Comparisons on the performance of the two architecture with the different number of encoder layers.

4.7. Comparisons on the Different Sampling Methods

One of the most important processes for 3D point clouds in object recognition scenarios is to sample the input point clouds (∼30K) to decrease the memory/efficiency burden on the computing platform. The most commonly used sampling methods for point clouds are Voxel-based Uniform Sampling (VS) [45] and Random Sampling (RS) [44]. However, the VS method is unable to limit the sampling points number, which is not suitable for our network structure, and ours needs a fixed input size. The RS method is suitable for our network, but the sampling points cannot fully cover the surface of the input data, because of its inherent randomness during sampling. To this end, we adopt the Farthest Point Sampling (FPS) [43] method and present the comparison results with RS on the AE-HF bin-picking dataset [26] as shown in Fig. 17. By fixing the key-points number (kp = 4096) and the local points number (lp = 256), we can see that the recognition result of the FPS is more effective than the RS method due to the FPS can fully cover the surface of the input point clouds.

In addition, we present the recognition results with different FPS sampling number of the kp and lp as shown in Fig. 18, where the average F1-score increases obviously along with the

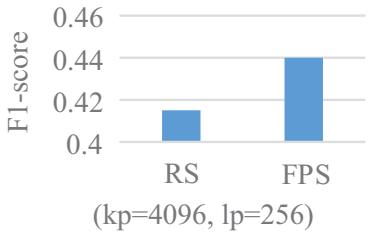


Figure 17: Compare the performance of RS [44] and FPS [43].

kp that is designed to increase from 1024 to 4096. By fixing the kp and increasing the lp, the average F1-score can also be improved. But when the lp exceeds 128 (in our case), it offers a slight improvement and comes at the expense of additional computational time. This is because the increase of kp augments the coverage of the target surface, and the increase of the lp can also enhance the expression of the local features, but when exceeding a threshold, it can fully express the local surface without more.

Figure 18: Comparisons on the performances of different sampling size of key-points k_p and sampling size of local points l_p.

4.8. Comparisons on the Different Support Radius r

The support radius r determines the range of the local sphere, which contains the local surface points for feature extraction. We present the recognition results of different size of the support radius r on the AE-HF bin-picking dataset [26].

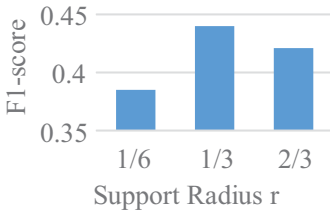


Figure 19: Comparisons on the performance of different support radius r.

We varied the size of the support radius r to $\frac{1}{6}$, $\frac{1}{3}$ and $\frac{2}{3}$, which means the proportion of the shortest object dimensions. As shown in Fig. 19, the $\frac{1}{3}$ shows the best performance. It also shows that an increase in the support radius r significantly improves the accuracy ($\frac{1}{6} \sim \frac{1}{3}$), while on the other hand, an excessive increase ($\frac{2}{3}$) of the r offers a slight decrease. This is because the appropriate support radius r is sufficient to express local features; instead, an oversize r will incur background interference.

5. Conclusions

In this paper, a deep stacked point feature is designed for 3D point clouds representation, which is able to preserve the original geometric information of objects to the greatest extent. We scrap the handcrafted feature engineering for 3D point clouds and propose a simple but effective unsupervised feature encoder that can be directly operated on a collection of local 3D point clouds spheres, which avoids the geometric information loss and reduces the computational costs. The feature extraction is enabled within a local neighborhood sphere, by stacking the locally sphere-wise aggregated feature on point-wise features. During offline training, the stacked point feature encoder is trained first and then generate a feature database of all keypoints, which are sampled from synthetic model views. During online testing, a number of scene features, which are sampled by farthest sampling method, match against the database of synthetic model views and cast 6D model votes. The votes are subsequently filtered to refine hypotheses. The proposed method is evaluated on four datasets and the results prove that ours can generalize well to multiple scenarios and deliver comparable or even superior performance than the state-of-the-arts. In the future work, we intend to extend the current pipeline to a supervised end-to-end network, which operates on pure point clouds and directly predicts the 6-DOF pose in 3D space.

References

[1] S. A. A. Shah, M. Bennamoun, F. Boussaid, Keypoints-based surface representation for 3d modeling and 3d object recognition, Pattern Recognition 64 (2017) 29–38.

[2] U. Asif, M. Bennamoun, F. A. Sohel, Rgb-d object recognition and grasp detection using hierarchical cascaded forests, IEEE Transactions on Robotics 33 (3) (2017) 547–564.

[3] M.-L. Torrente, S. Biasotti, B. Falcidieno, Recognition of feature curves on 3d shapes using an algebraic approach to hough transforms, Pattern Recognition 73 (2018) 111–130.

[4] M. Wang, Y. Gao, K. Lu, Y. Rui, View-based discriminative probabilistic modeling for 3d object retrieval and recognition, IEEE Transactions on Image Processing 22 (4) (2013) 1395–1407.

[5] X. Li, M. Fang, J. J. Zhang, J. Wu, Learning coupled classifiers with rgb images for rgb-d object recognition, Pattern Recognition 61 (2017) 433–446.

[6] Y. Cong, G. Sun, J. Liu, H. Yu, J. Luo, User attribute discovery with missing labels, Pattern Recognition 73 (2018) 33–46.

[7] J. M. Morel, G. Yu, Asift: A new framework for fully affine invariant image comparison, SIAM journal on imaging sciences 2 (2) (2009) 438–469.

[8] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[9] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.

[10] A. Collet, M. Martinez, S. S. Srinivasa, The moped framework: Object recognition and pose estimation for manipulation, International Journal of Robotics Research 30 (10) (2011) 1284–1306.

[11] J. Tang, S. Miller, A. Singh, P. Abbeel, A textured object recognition pipeline for color and depth image data, in: ICRA, 2012, pp. 3467–3474.

[12] T. Zhou, X. Jing, Surface-based detection and 6-dof pose estimation of 3-d objects in cluttered scenes, IEEE Transactions on Robotics PP (99) (2016) 1–15.

[13] A. E. Johnson, M. Hebert, Surface matching for object recognition in complex three-dimensional scenes, Image and Vision Computing 16 (9-10) (1998) 635–651.

[14] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, J. Wan, Trisi: A distinctive local surface descriptor for 3d modeling and object recognition, in: GRAPP/IVAPP, 2015.

[15] A. S. Mian, M. Bennamoun, R. Owens, Three-dimensional model-based object recognition and segmentation in cluttered scenes, IEEE transactions on pattern analysis and machine intelligence 28 (10) (2006) 1584–1601.

[16] P. Bariya, J. Novatnack, G. Schwartz, K. Nishino, 3d geometric scale variability in range images: Features and descriptors, International journal of computer vision 99 (2) (2012) 232–255.

[17] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, Wan, Rotational projection statistics for 3d local surface description and object recognition, International journal of computer vision 105 (1) (2013) 63–86.

[18] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, N. M. Kwok, A comprehensive performance evaluation of 3d local feature descriptors, International journal of computer vision 116 (1) (2016) 66–89.

[19] Y. Cong, D. Tian, Y. Feng, B. Fan, H. Yu, Speedup 3-d texture-less object recognition against self-occlusion for intelligent manufacturing, IEEE transactions on cybernetics 48 (99) (2018) 1–11.

[20] A. Mian, M. Bennamoun, R. Owens, On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes, International Journal of Computer Vision 89 (2-3) (2010) 348–361.

[21] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes, in: ACCV, 2012, pp. 548–562.

[22] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, V. Lepetit, Gradient response maps for real-time detection of textureless objects, IEEE transactions on pattern analysis and machine intelligence 34 (5) (2012) 876–888.

[23] R. Rioscabrera, T. Tuytelaars, Discriminatively trained templates for 3d object detection: A real time scalable approach, in: ICCV, 2014, pp. 2048–2055.

[24] W. Kehl, F. Tombari, N. Navab, S. Ilic, V. Lepetit, Hashmod: A hashing method for scalable 3d object detection, in: BMVC, 2016.

[25] A. Tejani, D. Tang, R. Kouskouridas, T. K. Kim, Latent-class hough forests for 3d object detection and pose estimation, in: ECCV, 2014, pp. 462–477.

[26] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, T. K. Kim, Recovering 6d object pose and predicting next-best-view in the crowd, in: CVPR, 2016, pp. 3583–3592.

[27] W. Kehl, F. Milletari, F. Tombari, S. Ilic, N. Navab, Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation, in: ECCV, 2016, pp. 205–220.

[28] H. Liu, Y. Cong, S. Wang, H. Fan, D. Tian, Y. Tang, Deep learning of directional truncated signed distance function for robust 3d object recognition, in: IROS, 2017, pp. 5934–5940.

[29] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, N. Navab, Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again, in: CVPR, 2018.

[30] M. Rad, V. Lepetit, Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth, in: ICCV, Vol. 1, 2017, p. 5.

[31] B. Tekin, S. N. Sinha, P. Fua, Real-time seamless single shot 6d object pose prediction, in: CVPR, 2018.

[32] R. Q. Charles, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: CVPR, 2017, pp. 77–85.

[33] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: NIPS, 2017, pp. 5099–5108.

[34] D. Chetverikov, D. Svirko, D. Stepanov, P. Krsek, The trimmed iterative closest point algorithm, in: ICPR, Vol. 3, 2002, pp. 545–548.

[35] R. Klokov, V. Lempitsky, Escape from cells: Deep kd-networks for the recognition of 3d point cloud models, in: ICCV, 2017, pp. 863–872.

[36] S. Song, J. Xiao, Sliding shapes for 3d object detection in depth images, in: ECCV, 2014, pp. 634–651.

[37] R. Girdhar, D. F. Fouhey, M. Rodriguez, A. Gupta, Learning a predictable and generative vector representation for objects, in: ECCV, 2016, pp. 484–499.

[38] S. Gupta, R. Girshick, P. Arbelez, J. Malik, Learning rich features from rgb-d images for object detection and segmentation, in: ECCV, Vol. 8695, 2014, pp. 345–360.

[39] S. Gupta, P. Arbelez, R. Girshick, J. Malik, Inferring 3d object pose in rgb-d images, ArXiv preprint arXiv:1502.04652.
[40] J. Leitner, A. W. Tow, N. Snderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. Mccool, The acrv picking benchmark: A robotic shelf picking benchmark to foster reproducible research, in: ICRA, 2017, pp. 4705–4712.
[41] A. Zeng, K. T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, J. Xiao, Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge, in: ICRA, 2017, pp. 1386–1383.
[42] V. Lepetit, F. Morenonoguer, P. Fua, Epnp: An accurate o(n) solution to the pnp problem, International Journal of Computer Vision 81 (2) (2009) 155–166.
[43] C. Moenning, N. A. Dodgson, Fast marching farthest point sampling for implicit surfaces and point clouds, Computer Laboratory Technical Report 565 (2003) 1–12.
[44] J. S. Vitter, Random sampling with a reservoir, ACM Transactions on Mathematical Software 11 (1) (1985) 37–57.
[45] R. B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: ICRA, IEEE, 2011, pp. 1–4.

Hongsen Liu is currently working toward the Ph.D. degree in pattern recognition and intelligent systems at the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His research interests include pattern recognition, image processing, and robotics.

Yang Cong (S'09-M'11-SM'15) is a full professor of Chinese Academy of Sciences. He won the NSFC outstanding youth fund, CAS-Youth Innovation Promotion Association Scholarship. He received the B.Sc. degree from Northeast University in 2004, and the Ph.D. degree from State Key Laboratory of Robotics, Chinese Academy of Sciences in 2009. He was a Research Fellow of National University of Singapore (NUS) and Nanyang Technological University (NTU) from 2009 to 2011, respectively; and a visiting scholar of University of Rochester. He has served on the editorial board of the Journal of Multimedia. His current research interests include image processing, compute vision, machine learning, multimedia, medical imaging, data mining and robot navigation. He has authored over 70 technical papers. He was the PI of more than 15 projects including 4 NSFC projects. He was a senior member of IEEE since 2015.

Chenguang Yang (M'10-SM'16) received the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010. From 2009 to 2010, he was a Post-Doctoral Researcher of human robotics with Imperial College London, London, U.K. He is currently a Professor with the Bristol Robotics Laboratory, Bristol, U.K. His research interests include human robot interaction and intelligent system design. Dr. Yang received the EU Marie Curie International Incoming Fellowship, U.K., the EPSRC UKRI Innovation Fellowship, and the Best Paper Award of the IEEE TRANSACTIONS ON ROBOTICS as well as over ten conference Best Paper Awards.

Yandong Tang received the B.S. and M.S.degrees in mathematics from Shandong University, Shandong, China, in 1984 and 1987, respectively, and the Doctors degree in applied mathematics from the University of Bremen, Bremen, Germany, in 2002.He is currently a Professor at the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His research interests include robot vision, image processing, and pattern recognition.