Adaptive Spotting: Deep Reinforcement Object Search in 3D Point Clouds

Onkar Krishna, Go Irie, Xiaomeng Wu,
Takahito Kawanishi, and Kunio Kashino

NTT Communication Science Laboratories, NTT Corporation, Japan
{krishna.onkar.ru, xiaomeng.wu.px}@hco.ntt.co.jp goirie@ieee.org

Abstract. In this paper, we study the task of searching for a query object of unknown position and pose in a scene, both given in the form of 3D point cloud data. A straightforward approach that exhaustively scans the scene is often prohibitive due to computational inefficiencies. High-quality feature representation also needs to be learned to achieve accurate recognition and localization. Aiming to address these two fundamental problems in a unified framework, we propose Adaptive Spotting, a deep reinforcement learning approach that jointly learns both the features and the efficient search path. Our network is designed to directly take raw point cloud data of the query object and the search bounding box and to sequentially determine the next pose to be searched. This network is successfully trained in an end-to-end manner by integrating a contrastive loss and a reinforcement localization reward. Evaluations on ModelNet40 and Stanford 2D-3D-S datasets demonstrate the superiority of the proposed approach over several state-of-the-art baselines.

1 Introduction

Objects in the real world are three-dimensional (3D). Understanding their appearances and semantics naturally requires 3D information processing. The external surfaces of 3D objects can be captured as point clouds, which have become easily acquirable with the proliferation of modern range sensors such as LiDAR and RGBD cameras, and with the improvement of SLAM and SfM accuracy. While such data have been used mainly for environment modeling towards robot control or autonomous driving purposes, the recent development of deep neural networks such as PointNet [1], which can directly handle 3D point cloud data, has opened new research avenues including point cloud-based classification/segmentation [1–3], detection [4–8], and retrieval [9, 10].

In this paper, we consider the problem of point cloud-based object search. The task is to find a query object from a scene (reference map), both given in the form of 3D point cloud data; for example, we may want to find a chair specified by a user from an office room. It is different from the popular point cloud-based object recognition tasks mentioned earlier. In object classification, detection, and segmentation, the object classes to be classified/detected are known in advance, while in our task the object to search is specified online as a query. The task

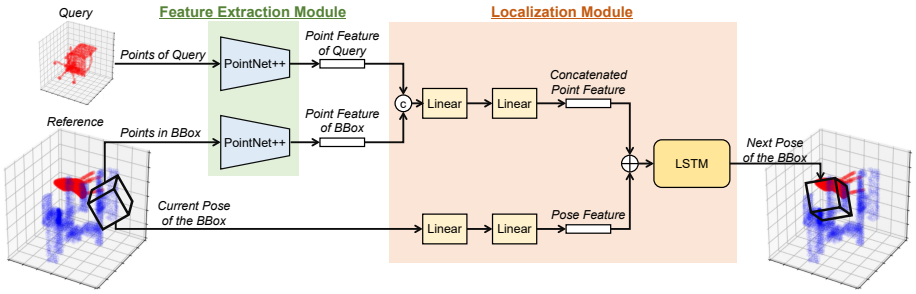

Fig. 1. Overview of our proposed network. Our model has two major modules: a feature extraction module and a localization module. Taking a query object and a current search bounding box in the reference map, the network sequentially determines the pose of the next bounding box. © and ⊕ indicate concatenation and element-wise addition, respectively.

most relevant to ours is object retrieval, where the common idea is to follow the successful approaches in 2D image retrieval. For example, the previous works [1, 9–11] use deep neural networks to represent each point cloud as a global feature, with which a query object is compared with multiple database objects. While the goal of object retrieval is to search a database, our task aims at finding a point cloud (query) that exists in a much larger point cloud (reference map) with unknown position and pose. One straightforward solution is to adapt existing object retrieval methods to our problem in a "sliding-window" manner, i.e., scanning the whole reference map using a small 3D bounding box with varying scale and orientation. However, this solution has to deal with a huge search space for accurate localization of the target object, making the overall search process fairly inefficient.

In this paper, we propose Adaptive Spotting, a point cloud search approach that significantly reduces the search space with greater matching accuracy. The idea is to learn efficient search path from data, i.e., using machine learning to pick and compare only highly prospective regions in the 3D reference map with a given query. Specifically, a recurrent neural network is employed to predict a sequence of 3D poses, each spotting a candidate 3D bounding box in the reference map and being expected to match with the query. The network is trained through reinforcement learning in an exploratory manner. No explicit supervision such as the 3D location of the target object or its class label is needed. In addition to the search path, our approach jointly learns point cloud feature representation in a unified framework, which provides a sharp insight into the similarity between the predicted target object and the query. To this end, we propose a novel loss configuration that integrates the feature contrastive loss with the reinforcement localization rewards. This configuration enables our model to learn more discriminative feature representation than traditional retrieval methods. We evaluate our approach on ModelNet40 and Stanford 2D-3D-S and

show that it produces remarkable accuracy improvement with a greatly reduced number of matching processes and a much shorter run time.

2 Related Work

We briefly review two relevant areas to this paper: deep feature representation techniques for point cloud data and point cloud-based object retrieval and search.

Deep Feature Representation for Point Cloud Data. While the feature representation of point clouds has long relied on handcrafted features, the recent success of CNNs in 2D image representation has led to researches on deep point cloud representation.

Unlike 2D images, regular convolution kernels cannot be applied to unstructured 3D point cloud data. Hence, early attempts rasterized the point cloud into dense and regular voxel grids to make use of 3D volumetric convolutions. Two pioneers are VoxNet [12] and 3D ShapeNet [13], which integrate volumetric occupancy grid representations with supervised 3D CNNs for point cloud representation learning. Following these two studies, many methods have been proposed to handle the sparsity problem of raw point clouds [14–17]. Another direction of deep point cloud representation is to project the set of 3D points onto 2D planes so that standard 2D CNNs can be applied directly [18–20]. For example, MultiView CNN [18, 19] tried to learn features using CNN in an end-to-end trainable fashion through multiple 2D projections of a 3D point cloud. This line of studies has achieved dominating performance on shape classification and retrieval tasks as reported in the competitions of the large scale 3D SHape REtrieval Contest (SHREC) [21]. However, it is nontrivial to extend them to our object search task or other 3D tasks such as point classification and shape completion.

More recently, a special type of neural network has been proposed that can be directly applied to raw point cloud data [1, 2, 22]. PointNet [1] is the pioneering network that directly processes point sets and uses a symmetric function to make the feature representation invariant to the order permutation of the input points. PointNet++ [2] is the extended version of PointNet and introduces a hierarchical architecture to exploit the local structures of the point set at multiple scales. Several other extensions have also been made to more effectively capture local to global shape characteristics of the input point cloud data [23–25]. In addition to these, several studies have been proposed that leverage the graph convolutional networks (GCNs) [26–28] for the point cloud convolution, given that the point cloud data can be readily modeled as a graph. Some studies such as [29] also introduced customized layers to capture the distribution of 3D points in a latent space.

In this study, we integrate PointNet++ [2] into our deep recurrent framework as a feature extraction module. The network is effectively trained in a fully end-to-end manner through our exploratory reinforcement learning and contrastive feature learning process.

Point Cloud-based Object Retrieval and Search. The common approach to point cloud-based object retrieval is the same as in the case of 2D images: it represents each point cloud by a feature vector and searches by feature matching. Traditional studies rely on handcrafted features to describe the shape of the point cloud data (often in the normal direction). A typical example is to use local 3D descriptors such as [30–32] to identify objects that match the query. Such methods often require dense and accurate point cloud data, and the signal-to-noise ratio of the target object needs to be small.

In response to the recent growth of deep point cloud representation, several studies have worked on point cloud object retrieval based on deep features. An early attempt [9] used a PointNet pre-trained for object recognition [1] to extract deep features for 3D shape retrieval. PointNetVLAD [10] adapted the NetVLAD architecture [33], which was originally designed for 2D image retrieval, by replacing the conventional 2D CNN with PointNet. Its performance was evaluated in a point cloud-based place recognition task that aims to identify the urban block corresponding to a given query from a database of pre-segmented urban blocks. PCAN [11] is an extension of PointNetVLAD that involves 3D attention map to exploit higher-level contextual information, and is also proposed for the same place recognition task.

Unfortunately, these techniques cannot be directly applied to the problem we are considering in this work. Our object search task is to accurately localize an object in a reference map with unknown position and pose, whereas the techniques described above do not take into account this localization requirement. Of course, it is possible to leverage these techniques for feature extraction and search for the object based on a sliding window (i.e., a 3D bounding box in the reference map). However, such an exhaustive strategy is often prohibitively time-consuming. By association, it is not easy to find an optimal quantization level for the position and size (scale) of the bounding box, such that sufficient accuracy and small search space can be guaranteed simultaneously. Another potentially related field is robotics, where object search has long been explored [34, 35]. However, most of the approaches are based on handcrafted features and heuristics.

Unlike the previous studies described above, our approach can predict the position and pose of the target object directly from the raw point cloud of the query and reference map. The conceptual idea of this study is somewhat similar to [36] and [37], where, however, only 2D images are taken into consideration. To the best of our knowledge, this is the first work that proposes an end-to-end reinforcement learning approach to object search on raw 3D point cloud data.

3 Adaptive Spotting

Given a pair of a query (target) object $\mathcal{Q}$ and a reference map $\mathcal{R}$ both represented as point clouds, our task is to find the correct pose of the target object in $\mathcal{R}$ that matches $\mathcal{Q}$.

Following previous studies for 3D object detection [4, 38], we represent the pose of the target object by a set of seven parameters, which are three for the

translation, three for the scaling, and one for the rotation (yaw) relative to $\mathcal{Q}$. This parameter set (or pose) is denoted by A_g. Given A_g, a 4×4 affine transformation matrix can be constructed, which represents the 3D mapping from $\mathcal{Q}$ to the target object in $\mathcal{R}$. We can bridge the geometric variation between the query and the target by applying the inverse of the above affine transformation to $\mathcal{R}$. Consequently, the points in the 3D bounding box, whose center is the origin and whose size is equal to the query, becomes the rectified target object and should match $\mathcal{Q}$ without viewpoint variation.

Given an arbitrary pose A, we use $\mathcal{R}(A)$ to denote the point cloud rectified from $\mathcal{R}$ according to A. For example, $\mathcal{R}(A_g)$ indicates the rectified target object. Our task is thus to find an affine matrix A such that $\mathcal{R}(A)$ matches $\mathcal{Q}$.

3.1 Overview

Different from the naïve solution that exhaustively scans $\mathcal{R}$, we approach to this problem by sequential search. Specifically, we aim at determining a sequence of poses $\{A_t\}_{t=0}^{T}$ such that the search process can correctly reach A_g within T steps. Here, t is the time step and T is the length of the sequence.

We propose a deep reinforcement learning approach with a neural network that embraces PointNet++ [2] and LSTM in a unified framework. The schematic overview of our network is illustrated in Fig. 1. Our network consists of two major modules called *feature extraction module* and *localization module*, respectively. In the test stage, the behaviors of our network at each time step t can be summarized as follows.

1. The feature extraction module extracts point features from $\mathcal{Q}$ and $\mathcal{R}(A_t)$, which are denoted by $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$, respectively. Here, $\mathcal{R}(A_t)$ indicates the predicted target object that has been rectified according to A_t.
2. The localization module receives the two sets of point features $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$ as well as the current pose A_t to determine the next pose A_{t+1}.

These two sub-processes are repeated sequentially until the maximum number of time steps T is reached. The detail of the two modules is given in Section 3.2.

Consider a realistic training scenario, where a trainer asks an autonomous agent to find an object in an unknown place, i.e., the ground truth pose A_g is unavailable. In this case, the only thing the trainer can do for training is to tell the agent whether the object localized by the agent is the target or not. Since the localization is achieved in a sequential manner, the decision made at the current time step relies on all the past decisions, and the quality of the current decision cannot be evaluated independently from the past ones. Therefore, typical supervised learning methods that assume i.i.d. samples cannot be applied to our scenario. Fortunately, the problem can be modeled as a Partially Observable Markov Decision Process (POMDP), which can be handled by reinforcement learning. We thus propose using a reinforcement learning algorithm that requires neither the explicit supervision nor the i.i.d. assumption. We give the detail of our training algorithm in Section 3.3.

Our idea is inspired by the Recurrent Attention Models (RAMs) primarily designed for image recognition [39–41]. In this study, we introduce this mechanism into point cloud-based object search by incorporating it with feature embedding of unstructured 3D data, leading to the simultaneous end-to-end learning of point features and efficient search paths.

3.2 Details of Modules

This section gives the details of the feature extraction module and the localization module one by one. Hereafter, we simply use the term "BBox" to indicate the 3D bounding box that covers the predicted target object in the space of $\mathcal{R}$ but has not been rectified according to A_t.

Feature Extraction Module. This module extracts point features $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$ from $\mathcal{Q}$ and $\mathcal{R}(A_t)$. Given a BBox that touches the true target object but does not fully overlap, the network is expected to move the BBox closer to the target at succeeding time steps. To make it possible, the network must know, e.g., in which direction/angle the BBox is offset from the target, and such information must be acquirable from $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$. Therefore, the point feature needs to be affine-variant: $f(\mathcal{R}(A_t))$ must change as the pose of $\mathcal{R}(A_t)$ changes.

In this study, we adopt PointNet++ [2] for this purpose but any deep, affine-variant feature extractor can be used here. Our feature extraction module consists of two identical PointNet++ networks with shared parameters; one for the query $\mathcal{Q}$ and the other for $\mathcal{R}(A_t)$. PointNet++ has a hierarchical architecture and alternately clusters and aggregates local points to extract a global feature vector that captures both global and local structures of the point cloud. This approach draws an analogy to convolution and pooling layers in standard CNNs. Specifically, we employ a three-level hierarchical architecture with 1024, 256, and 128 clusters, respectively. The final embedding is obtained by applying two fully connected (FC) layers with output dimensions of 512 and 64, respectively. The extracted features $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$ are both fed to the subsequent localization module to determine the pose of the next BBox.

Localization Module. The main component of the localization module is an LSTM that sequentially predicts the pose A_{t+1} of the next BBox from three external inputs, namely $f(\mathcal{Q})$, $f(\mathcal{R}(A_t))$, and the current pose A_t.

We first concatenate $f(\mathcal{Q})$ and $f(\mathcal{R}(A_t))$ and employ two FC layers to generate a single 128D point feature vector. This feature reflects the appearance of $\mathcal{Q}$ and $\mathcal{R}(A_t)$ as well as the *relative* spatial relationship between the two point clouds. Meanwhile, A_t is encoded, by two FC layers of output dimensions 64 and 128, into a 128D pose feature vector. This feature represents the *absolute* spatial pose of the current BBox. The point and pose features are summed and then fed to the LSTM, as shown in Fig. 1.

The LSTM outputs the next 128D hidden state h_{t+1} to which a linear projection is applied to obtain the expected value $\hat{A}_{t+1}$ of the seven pose parameters.

That is, we assume that A_{t+1} is a stochastic variable and follows a Gaussian distribution whose mean is given by $\hat{A}_{t+1}$. Once $\hat{A}_{t+1}$ is predicted, A_{t+1} is obtained as a sample drawn from the distribution $\mathcal{N}(\hat{A}_{t+1}, \lambda I)$, where I is the identity matrix and λ is a hyperparmeter controlling the standard deviation.

For initialization, the first pose A_0 is determined randomly. If at any time $\mathcal{R}(A_t)$ contains no points, we take the center point, i.e., $(0, 0, 0)$, as $\mathcal{R}(A_t)$ and feed it to PointNet++ to inform the network about the lack of objectness.

3.3 Model Training

Let $\Theta = \{\theta_f, \theta_l\}$ be the parameters of the whole network, where θ_f and θ_l correspond to feature extraction and localization modules, respectively. Reinforcement learning is used to tune Θ. For simplicity, we use $s_{t-1} = \{\{P_\tau\}_{\tau=1}^{t-1}, \mathcal{Q}, \mathcal{R}\}$ to denote the set containing all the past predicted poses as well as the query and the reference. The policy of the reinforcement learning can then be represented as a conditional distribution $\pi(A_t|s_{t-1}; \Theta)$. Our goal is to maximize the total reward $R = \sum_{t=1}^{T} r_t$ w.r.t. Θ, where r_t is the reward value, typically one or zero, at time t (The definition of r_t is given in Section 4.2). The expected value of the total reward is then

$$J(\Theta) = \mathbb{E}_{p(s_T;\Theta)}[R], \tag{1}$$

where $p(s_T; \Theta)$ is the probabilistic distribution of s_T, which depends on the policy. Although the gradient w.r.t. Θ is non-trivial, it can be approximately computed by sampling the sequence $\{A_t, s_{t-1}\}_{t=1}^{T}$ from the policy in a similar way to Monte-Carlo approximation, which gives

$$\nabla_\Theta J(\Theta) \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{t=1}^{T} \nabla_\Theta \log \pi(A_t^i|s_{t-1}^i; \Theta) R^i. \tag{2}$$

Here, M is the number of sampled sequences. With this equation, Θ can be iteratively updated through gradient ascent.

Training our model only with reinforcement learning often makes the learning process unstable. Hence, we involve another loss function suitable for the object search task to improve the stability. Specifically, we require the two point cloud features extracted by the feature extraction module to correctly reflect the similarity between $\mathcal{Q}$ and $\mathcal{R}(A_t)$, i.e., the distance between the two features should be small for matching pairs and large for non-matching pairs. To this end, we impose a loss function, which resembles contrastive loss, on the feature extraction module.

$$L(\theta_f) = \sum_{t=1}^{T} r_t d^2 + (1 - r_t) \max\{0, m - d\}^2, \tag{3}$$

where $d = ||f(\mathcal{Q}) - f(\mathcal{R}(A_t))||$ and m is the margin of the hinge loss. Unlike the standard contrastive loss, Eq. 3 exploits the reward r_t to determine the loss value. It is piecewise differentiable, and so can be easily optimized with gradient descent.

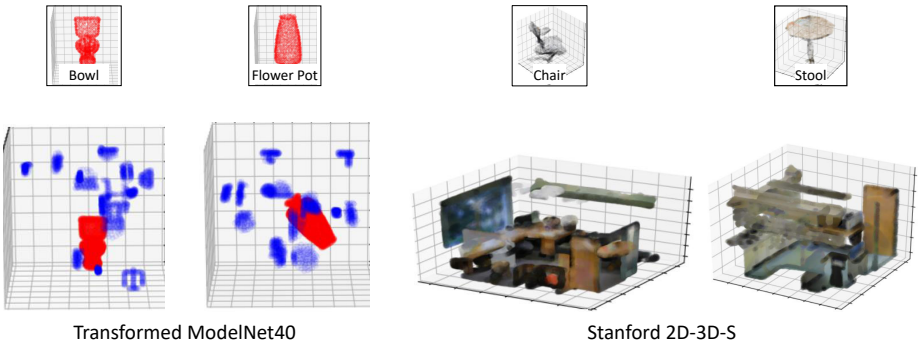

Fig. 2. Examples of our datasets. Pairs of query (top) and reference (bottom) are visualized. The examples of Stanford-2D-3D-S are textured for visualization purpose only. In our experiments, we use texture-less 3D points.

4 Experiments

4.1 Datasets

We used two benchmark datasets, ModelNet40[1] [13] and Stanford 2D-3D-S[2] [42], to evaluate the performance of our approach. Examples of the query/reference pairs are shown in Fig. 2.

Transformed ModelNet40. ModelNet40 consists of 40 object classes with varying numbers (from 64 to 889) of clean CAD instances per class, each having the size $2 \times 2 \times 2$ and $1,024$ points. The total number of objects is $12,311$. We used this dataset to analyze the performance of our approach under the controlled setup. The dataset was originally created for object recognition and shape completion [13] and cannot be directly used for object search evaluation. Hence, we conducted the following processing to generate query/reference pairs from the original dataset.

Given an object of size $2 \times 2 \times 2$, we randomly sample a set A_g of seven pose parameters from the ranges of $[-1, 1]$ for 3D translation, $[0.8, 1.0]$ for 3D scaling, and $[0, 360)$ for 1D rotation, and transform the object to a new reference space (reference map). The reference map is no larger than $4 \times 4 \times 4$. 10 patches, each having 128 points, are randomly extracted from objects of other categories and added to the reference map as distractors. The reference map thus has $2,304$ points. The object described above is then regarded as the query and combined with the reference to form a query/reference pair.

This processing was applied to all the objects in the dataset, leading to $12,311$ query/reference pairs. We followed the practice of [43] for obtaining the training and testing sets, each consisting of $9,843$ and $2,468$ pairs, respectively.

[1] https://modelnet.cs.princeton.edu/
[2] http://buildingparser.stanford.edu/dataset.html

Stanford 2D-3D-S. To evaluate our approach in more realistic scenes, we made use of the Stanford 2D-3D-S dataset to create query/reference pairs. The Stanford 2D-3D-S dataset consists of 247 different halls originating from 6 large-scale indoor areas that cover $6,000\ m^2$ in total. Instance-level annotations are provided for 13 object classes including structural elements (ceiling, floor, wall, beam, column, window, and door) and movable elements (table, chair, sofa, bookcase, board and clutter).

To generate query/reference pairs, we split each giant hall into $7^3 = 343$ regular partitions. Each partition is regarded as a reference map if it contains at least one object for which 70% of its points are inside the partition. All the movable elements, whose volumes are no less than 1/10 of the reference map, are taken as queries.

We selected one out of the six indoor areas and used all the query/reference pairs derived from it for testing. All the other pairs were used for training. The resulting training set consists of $2,919$ queries and $1,504$ references, while the test set consists of 312 queries and 285 references. For this dataset, a search is determined to be successful if the approach finds an object that has the same class as the query.

4.2 Experimental Setup

Performance Metrics. We evaluated our approach in terms of success rate and search time. Given a query/reference pair, our approach predicts a sequence of T poses over the reference map, which are used to evaluate the success rate. With each pose, we can easily identify the corners of the predicted BBox. A point cloud search is considered as being successful if the 3D IoU [38] between the ground truth BBox and the predicted BBox is greater than a certain threshold. The success rate is defined as the ratio of the number of the query/reference pairs, where the search is successful, over the total number of tested pairs. The search time (in seconds) is the time required for predicting the T poses for each query/reference pair.

Baselines. To the best of our knowledge, there is no existing techniques that predict a sequence of 3D poses over a reference map to search for a 3D point cloud object. In this study, we compared our approach with state-of-the-art models developed for 3D object retrieval and classification, namely PointNet [1], PointNet++ [2], and PointNetVlad [10], which were adapted to out task in a "sliding window" manner. For each baseline, we slide a 3D BBox of three different scales over all three dimensions of the reference map with a certain sampling rate. For example, we have $2^3 \times 3 = 24$ BBoxes if the sampling rate is two and $7^3 \times 3 = 1,029$ BBoxes if the rate is seven. We then use baseline models to extract point features from the sampled BBoxes, and compare them with the query to find the best match. All the baselines were trained using ModelNet40 and Stanford 2D-3D-S datasets, separately. Their network configurations are the same as those reported in the original papers.

Table 1. Results on Transformed ModelNet40. Values in the table are success rates along with the parenthesized average time (in seconds) required to process one query-reference pair. Results were obtained with the IoU threshold being 0.5 when a 64D PointNet++ feature vector was used.

No. of BBoxes	24	81	192	375	648	1029
PointNet++	0.12 (0.54)	0.16 (1.88)	0.16 (4.50)	0.26 (8.82)	0.28 (15.25)	- (36.70)
PointNet	0.13 (0.19)	0.24 (0.62)	0.31 (1.44)	0.29 (2.83)	0.30 (4.84)	0.32 (7.57)
PointNetVLAD	0.13 (1.05)	0.33 (3.55)	0.47 (8.35)	0.50 (16.69)	0.50 (28.94)	0.53 (45.67)
No. of BBoxes	4	6	8	10	12	24
Ours	0.44 (0.13)	0.52 (0.18)	0.57 (0.24)	0.65 (0.39)	0.68 (0.45)	0.74 (0.49)

Learning Configurations. We trained the proposed network from scratch. We used Adam with a batch size of five for Transformed Modelnet40 and one for Stanford 2D-3D-S. The learning rate was kept in the range $[10^{-4}, 10^{-3}]$ with exponential decay. The hyperparameter λ in Section 3.2 (Localization Module), which is used for sampling out the next pose with the predicted expected value, is fixed to 0.22. We evaluate our approach by varying the maximum number of time steps T, which is sometimes referred to as "No. of BBoxes". The reward r_t in our implementation is determined based on the success (one) or failure (zero) of the search at time t. The search is considered as being successful if the 3D IoU [38] between the ground truth BBox and the predicted BBoxes at time t is greater than a threshold. Note that this simulator is the same as the scheme used to calculate the success rate during evaluation.

In the following sections, we first report the quantitative performance of our approach evaluated in terms of success rate and search time (in seconds). We performed extensive evaluation of our network by varying different parameters to understand their impact over search performance. These parameters include number of BBoxes, IoU threshold, and output dimensionality of the feature extraction module (PointNet++). Finally, we show qualitative results to demonstrate the effectiveness of our network in learning the search path.

4.3 Accuracy vs Number of BBoxes

Results on Transformed ModelNet40. Table 1 shows the performance of our network obtained by varying the number of BBoxes from 4 to 24 on the Transformed ModelNet40 dataset. Because the baselines could not achieve satisfactory success rates with such a small number of BBoxes, we show their performance obtained when varying the parameter in the range of 24 to $1,029$. The procedure of BBox sampling for baseline methods has been described in Section 4.2.

Table 1 shows that the success rate increases as the number of search BBoxes increases for both our approach and the baselines. Our approach clearly outperformed all the baselines with a huge margin by processing only a very small number of BBoxes. For instance, with 24 BBoxes the success rate of our approach reached 0.74, while those of the baselines were less than 0.15. The baseline meth-

Table 2. Results on Stanford 2D-3D-S. Values in the table are success rates along with the parenthesized average time (in seconds) required to process one query-reference pair. Results were obtained with the IoU threshold being 0.4 when a 64D PointNet++ feature vector was used.

No. of BBoxes	81	192	375	648	1029
PointNet++	0.20 (1.38)	0.16 (3.30)	0.22 (6.55)	0.26 (11.40)	0.32 (18.15)
PointNet	0.14 (0.52)	0.14 (1.20)	0.12 (2.39)	0.12 (4.07)	0.18 (6.44)
PointNetVLAD	0.12 (3.16)	0.16 (7.53)	0.26 (14.67)	0.26 (25.47)	0.34 (40.95)
No. of BBoxes	55	65	75	85	95
Ours	0.29 (1.17)	0.33 (1.37)	0.35 (1.59)	0.35 (1.80)	0.42 (2.01)

ods could not achieve the same level of accuracy as ours, even with over $1,000$ search BBoxes. Moreover, our approach required much shorter search time than PointNet++ and PointNetVLAD. PointNet was the fastest among all the compared approaches because of its much simpler network architecture. Using the same network architecture, our approach achieved a huge gain in success rate (approximately six times) compared to PointNet++ without increasing the run time.

Results on Stanford 2D-3D-S. Results on Stanford 2D-3D-S dataset are shown in Table 2. This dataset is more challenging than ModelNet40, because it contains meaningful and much larger scale of distractors (i.e., realistic objects whose classes are different from the query), as shown in Fig. 2. Our approach is more likely to be "trapped" by the distractors.

The performances for varying the number of BBoxes are shown in Table 2. Due to the larger size of the reference map and realistic distractive noises, we use the larger number of BBoxes in the range of 55 to 95 for this dataset than Transformed ModelNet40. Our approach achieved a much better balance between success rate and search time than the baseline methods. For example, our approach achieved a success rate of 0.42 with 2.01 seconds in average, while the success rates of the baselines were no higher than 0.16 at comparable search times. Results on Stanford 2D-3D-S dataset confirms that our network is easily scalable for object search in 3D real-world indoor scenes by efficiently exploring only prospective regions in a sequential manner.

4.4 Accuracy vs Feature Dimensionality

We studied the relationship between the discriminative power of our network and the dimensionality of the point feature extracted from the feature extraction module (PointNet++). As shown in Table 3, our approach attained the highest success rates when the feature dimension was set at 64 or 128. The success rate decreased as we further increased the number of feature dimensions. The reason may be overfitting due to an excessive number of feature dimensions. In particular, as shown in Fig. 1, our network uses the pose feature obtained from the 7D pose parameters by projecting them to the same number of dimensions

Table 3. Success rate and search time (in seconds) obtained with varying feature dimensions of PointNet++ on both datasets. Results were obtained with 24 and 85 BBoxes, and the IoU threshold was set at 0.50 and 0.40 for Transformed ModelNet40 and Stanford 2D-3D-S datasets, respectively.

Feature Length	32	64	128	256	512
Transformed ModelNet40					
Ours	0.62	0.74	0.68	0.63	0.48
Stanford 2D-3D-S					
Ours	0.34	0.38	0.38	0.35	0.33

Table 4. Success rates obtained with different IoU thresholds on Transformed ModelNet40. The IoU threshold is used to judge whether a search is successful or not. Based on this judgment, the reinforcement localization reward is determined during training, while the same threshold is used for calculating the success rate during testing.

IoU	0.35	0.40	0.45	0.50	0.55
No. of BBoxes = 648					
PointNet++	0.64	0.52	0.40	0.28	0.20
PointNet	0.69	0.56	0.42	0.30	0.20
PointNetVLAD	0.84	0.76	0.66	0.50	0.36
No. of BBoxes = 24					
Ours	1.0	0.97	0.86	0.74	0.44

as the point feature. Using too large dimension of the pose feature, say 512D, may readily raise the risk of overfitting. Also, it might become less informative for pose estimation after combined with the point feature.

4.5 Accuracy vs IoU Threshold

As mentioned in Section 4.2, while training our network we provided the reward r_t based on the search outcome, i.e., the success (one) or failure (zero) of the search at each time step t. We consider a search is successful if the 3D IoU between the predicted and ground truth BBoxes is greater than a certain threshold. Here, we aim to analyze the sensitivity of the performance to this threshold. To this end, we first trained our network with different IoU thresholds, ranging from 0.35 to 0.55 on Transformed ModelNet40 and 0.25 to 0.50 on Stanford 2D-3D-S. Then we evaluated the trained network and calculated the success rate with the same IoU threshold as used during training.

The results on Transformed ModelNet40 are shown in Table 4. The success rate is as high as 1.0 for IoU threshold 0.35 even if only 24 BBoxes are compared with the query. The performance then decreases as we raises the threshold. All the baseline methods also followed the same trend, but in all cases, they could not achieve the same level of accuracy as ours, even if a much larger number

Table 5. Success rates obtained with different IoU thresholds on Stanford 2D-3D-S. The IoU threshold is used to judge whether a search is successful or not. Based on this judgment, the reinforcement localization reward is determined during training, while the same threshold is used for calculating the success rate during testing.

IoU	0.25	0.30	0.35	0.40	0.45	0.50
No. of BBoxes = 1029						
PointNet++	0.62	0.52	0.44	0.32	0.12	0.04
PointNet	0.56	0.42	0.30	0.18	0.14	0.06
PointNetVLAD	0.62	0.58	0.48	0.34	0.20	0.10
No. of BBoxes = 85						
Ours	0.61	0.56	0.52	0.35	0.26	0.17

of BBoxes were processed. Results on Stanford 2D-3D-S are given in Table 5, which shows the same trend as the case of Transformed ModelNet40.

4.6 Qualitative Analysis

Fig. 3 shows some examples of the search behavior of our approach on Stanford 2D-3D-S for given queries. In all the examples, our network is able to approach the target object within a small number of search steps. Some interesting behaviors can be observed. First, in all the examples, starting from a random initial position, our approach moves the BBox significantly at first, and then takes a finer step size for precise localization as soon as it touches the target object. This shows that our approach is able to learn a natural and efficient search strategy. Second, our approach prioritizes the areas where the query object is likely to be present. For example, in the case of (a), our approach moves the BBox quickly from a high initial position to the floor and then stays on the floor to look for the chair. In the example in (b), starting from a low initial position, our approach continues to search near the floor surface and successfully captures the target without searching unnecessarily high areas. These examples show that our approach can leverage the advantage of the inherent contextual information between the target object and the search space for lean object search.

5 Conclusions

In this study, we proposed Adaptive Spotting, a deep reinforcement learning approach to point cloud-based object search. This model takes the point clouds of a query and a reference map as input and leverages the internal state of the network to predict the next prospective position and pose to focus on. Using this model, the target object that corresponds to the query but is hidden in the reference can be efficiently localized in an exploratory way without being confused with clutter present in the reference. Our model significantly outperformed state-of-the-art deep point cloud networks and greatly reduced the number of

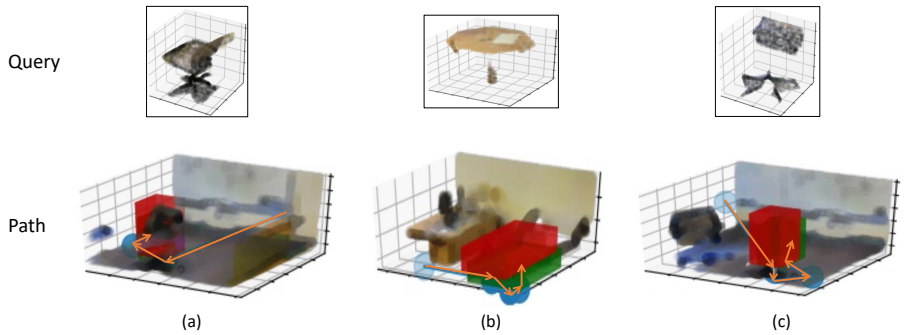

Fig. 3. Qualitative results on Stanford 2D-3D-S. For each query (top) and reference (bottom) pair, the target object and the final localization result by our approach are indicated by the red and green 3D BBoxes, respectively. Search paths are indicated by a sequence of orange arrows. The examples are textured for visualization purpose only. In our experiments, we use texture-less 3D points.

matching processes and execution times, thereby making a noticeable improvement in the trade-off between accuracy and efficiency.

As one of the few examples of research dealing with point cloud-based object search based on end-to-end deep reinforcement learning, we believe this work revitalizes problems at the intersection of computer vision and other related fields including robotics, sensing, and machine learning. Exploring how to terminate the search process at any time step is an interesting direction for further research. We shall also investigate the impact of different exploration/reward strategies on the learning performance of our model to further optimize the accuracy-efficiency trade-off.

References

1. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proc. CVPR. (2017) 77–85
2. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Proc. NeurIPS. (2017) 5099–5108
3. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on X-transformed points. In: Proc. NeurIPS. (2018) 828–838
4. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum PointNets for 3D object detection from RGB-D data. In: Proc. CVPR. (2018) 918–927
5. Xu, D., Anguelov, D., Jain, A.: PointFusion: Deep sensor fusion for 3D bounding box estimation. In: Proc. CVPR. (2018) 244–253
6. Shi, S., Wang, X., Li, H.: PointRCNN: 3D object proposal generation and detection from point cloud. In: Proc. CVPR. (2019) 770–779
7. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep Hough voting for 3D object detection in point clouds. In: Proc. ICCV. (2019) 9276–9285

8. Du, L., Ye, X., Tan, X., Feng, J., Xu, Z., Ding, E., Wen, S.: Associate-3Ddet: Perceptual-to-conceptual association for 3D point cloud object detection. In: Proc. CVPR. (2020)
9. Pham, Q.H., et al.: RGB-D object-to-CAD retrieval. In: Proc. Eurographics Workshop on 3D Object Retrieval. (2018) 45–52
10. Uy, M.A., Lee, G.H.: PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In: Proc. CVPR. (2018) 4470–4479
11. Zhang, W., Xiao, C.: PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In: Proc. CVPR. (2019) 12436–12445
12. Maturana, D., Scherer, S.A.: VoxNet: A 3D convolutional neural network for real-time object recognition. In: Proc. IROS. (2015) 922–928
13. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: Proc. CVPR. (2015) 1912–1920
14. Wang, D.Z., Posner, I.: Voting for voting in online point cloud object detection. In: Proc. Robotics: Science and Systems. (2015)
15. Riegler, G., Ulusoy, A.O., Geiger, A.: OctNet: Learning deep 3D representations at high resolutions. In: Proc. CVPR. (2017) 6620–6629
16. Engelcke, M., Rao, D., Wang, D.Z., Tong, C.H., Posner, I.: Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks. In: Proc. ICRA. (2017) 1355–1361
17. Le, T., Duan, Y.: PointGrid: A deep network for 3D shape understanding. In: Proc. CVPR. (2018) 9204–9214
18. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: Proc. ICCV. (2015) 945–953
19. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view CNNs for object classification on 3D data. In: Proc. CVPR. (2016) 5648–5656
20. Bai, S., Bai, X., Zhou, Z., Zhang, Z., Jan Latecki, L.: GIFT: A real-time and scalable 3D shape search engine. In: Proc. CVPR. (2016) 5023–5032
21. Savva, M., et al.: Large-scale 3D shape retrieval from ShapeNet Core55. In: Proc. Eurographics Workshop on 3D Object Retrieval. (2016)
22. Li, J., Chen, B.M., Lee, G.H.: SO-Net: Self-organizing network for point cloud analysis. In: Proc. CVPR. (2018) 9397–9406
23. Hua, B.S., Tran, M.K., Yeung, S.K.: Pointwise convolutional neural networks. In: Proc. CVPR. (2018) 984–993
24. Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3D. In: Proc. CVPR. (2018) 3887–3896
25. Wu, W., Qi, Z., Li, F.: PointConv: Deep convolutional networks on 3D point clouds. In: Proc. CVPR. (2019) 9621–9630
26. Qi, X., Liao, R., Jia, J., Fidler, S., Urtasun, R.: 3D graph neural networks for RGBD semantic segmentation. In: Proc. ICCV. (2017) 5209–5218
27. Yang, Y., Feng, C., Shen, Y., Tian, D.: FoldingNet: Point cloud auto-encoder via deep grid deformation. In: Proc. CVPR. (2018) 206–215
28. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Trans. Graphics (TOG) **38** (2019) 146:1–146:12
29. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: SPLATNet: Sparse lattice networks for point cloud processing. In: Proc. CVPR. (2018) 2530–2539
30. Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M.: Aligning point cloud views using persistent feature histograms. In: Proc. IROS. (2008) 3384–3391

31. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3D object recognition. In: Proc. CVPR. (2010) 998–1005
32. Samuele Salti, Federico Tombari, L.D.S.: SHOT: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding **125** (2014) 251–264
33. Arandjelovič, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: Proc. CVPR. (2016) 5297–5307
34. Saidi, F., Stasse, O., Yokoi, K., Kanehiro, F.: Online object search with a humanoid robot. In: Proc. IROS. (2007) 1677–1682
35. Aydemir, A., Pronobis, A., Gobelbecker, M., Jensfelt, P.: Active visual object search in unknown environments using uncertain semantics. IEEE Trans. Robotics **29** (2013) 986–1002
36. Nagaraja, V.K., Morariu, V.I., Davis, L.S.: Searching for objects using structure in indoor scenes. In: Proc. BMVC. (2015) 53.1–53.11
37. Krishna, O., Irie, G., Wu, X., Kawanishi, T., Kashino, K.: Learning search path for region-level image matching. In: Proc. ICASSP. (2019) 1967–1971
38. Zhou, D., Fang, J., Song, X., Guan, C., Yin, J., Dai, Y., Yang, R.: IoU loss for 2D/3D object detection. In: Proc. 3DV. (2019) 85–94
39. Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: Proc. NeurIPS. (2014) 2204–2212
40. Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. In: Proc. ICLR. (2015)
41. Ablavatski, A., Lu, S., Cai, J.: Enriched deep recurrent visual attention model for multiple object recognition. In: Proc. WACV. (2017) 971–978
42. Armeni, I., Sax, A., Zamir, A.R., Savarese, S.: Joint 2d-3d-semantic data for indoor scene understanding. arXiv:1702.01105 (2017)
43. Notchenko, A., Kapushev, Y., Burnaev, E.: Large-scale shape retrieval with sparse 3D convolutional neural networks. In: Proc. AIST. (2017) 245–254