REINFORCEMENT LEARNING BASED POINT-CLOUD ACQUISITION AND RECOGNITION USING EXPLORATION-CLASSIFICATION REWARD COMBINATION

Kevin Riou, Kevin Subrin, Patrick Le Callet

REINFORCEMENT LEARNING BASED POINT-CLOUD ACQUISITION AND RECOGNITION USING EXPLORATION-CLASSIFICATION REWARD COMBINATION.

Kevin Riou, Kevin Subrin, Patrick Le Callet

Nantes Université, Ecole Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France,
name.surname@univ-nantes.fr

ABSTRACT

3D points acquisitions based on robust sensors such as tactile or laser sensors are true alternatives to computer vision for 3D object recognition. In real life scenarios where robots are equipped with such sensors to acquire 3D data, only few points can be iteratively collected in a reasonable amount of time. However, existing Point-cloud classifiers are extremely sensitive to sparse points, missing parts and noise. To compensate for the sparsity of the data, some Reinforcement Learning (RL) based approaches have been proposed to learn a sparse yet efficient exploration of the target object regarding the 3D recognition objective. However, existing RL approaches only focus on classification performances to guide the training of the active acquisition-and-classification frameworks, and thus fail to dissociate poor exploration strategy (missing parts, noisy points) from actual classifier mistakes on proper data. In this study, we proposed a new RL framework that was rewarded regarding both the classification performances and the exploration quality. Our trained framework outperforms existing State-Of-The-Art models on 3D geometric objects classification. We further showed that our trained framework learnt to alternate between (1) a clean and broad exploration strategy, suitable for easily distinguishable categories, and (2) a specific local exploration strategy, facilitating the discrimination of similar categories.

Index Terms— Active point-clouds acquisition, 3D objects recognition, Reinforcement Learning

1. INTRODUCTION

Most existing deep learning architectures capable of reasoning about 3D point-cloud data are trained on offline/frozen datasets of dense and clean point-clouds [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. A recent study [11] proved that State-Of-The-Art (SOTA) point-cloud classifiers actually perform significantly worse on data including noise, missing parts and sparse points. On the contrary, many industrial applications require online/active acquisition of point-clouds, and often result in sparse and noisy data. Typical applications include environments where cameras are unoperable (e.g., dusty environments, poor luminosity conditions), requiring to use more robust data acquisition pipelines, such as iterative 3D points acquisitions using a laser sensor or a tactile sensor mounted on a polyarticulated robot. To maintain a decent pace in industrial context, only a limited number of points can be sampled for each object to process. To allow accurate 3D object classification from a limited number of actively sampled points, some recent works [12, 13] proposed to simultaneously learn the active sampling strategy, namely the exploration strategy, and the classifier. The insight behind such approach is that the exploration strategy training should be guided by the classification performances, so that each point acquisition provides the most information for the 3D recognition task. Both models leveraged an online RL algorithm rewarded by the classification performances to learn the exploration strategy, coupled with a classification loss to train the classifier. Such strategy basically compensates for the sparsity of the data by maximizing the efficiency of the exploration. However, such strategy doesn't explicitly penalize (1) missing parts on the explored object, e.g., due to a too narrow exploration, nor (2) noisy points, due to exploration trials that missed the object. In other words, such approach doesn't dissociate miss-classifications deriving from poor exploration strategy, from the ones caused by classifier mistakes. This leads to a potential unstable and sample-inefficient training.

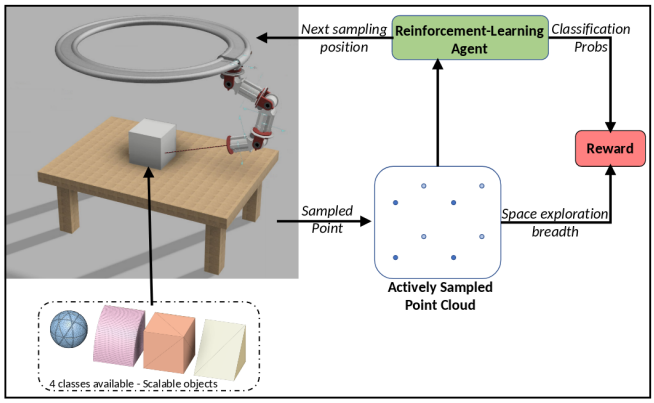

Fig. 1: Our RL based approach.

In this study, we investigate how an RL based framework can be used to learn an efficient active sampling strategy. We especially introduce a new exploration reward, that allows to improve the sampling strategy by providing cues on the exploration efficiency (exploration breadth, missing points) during training.

2. METHOD

Our proposed active sampling setup, depicted in Figure 1, consists of a robot equipped with a laser-based distance sensor that allows to recover 3D points from the work-place where an unknown 3D object is placed. The robot, fixed on the roof on a circular track, can turn around the work-piece and disposes of several degrees of freedom to enable the acquisition of 3D points all over the 3D object. This section (1) details the active sampling setup with the associated simulator proposed and (2) describes the RL framework proposed to learn an active sampling strategy that allows the recognition of the 3D object from few 3D points acquisitions.

2.1. Simulation of the active sampling hardware

An in-house simulator was leveraged to simulate the target use case and facilitate training and validation of the proposed solution. Figure 2 illustrates the simulator operation. The simulator consists of a 3D environment where 3D objects can be loaded and placed at the origin "O". 3D objects can be placed with any rotation around Z axis. A simulated laser sensor is integrated to the environment to allow the acquisition of 3D points on the loaded object. The object can be actively explored by launching successive laser acquisitions in the environment. The active exploration of the object is thus managed by iteratively controlling the position and the orientation of the simulated laser sensor.

To match the target use case, the laser sensor is revolvable around the 3D object. As depicted in Figure 2 c), the laser can be positioned anywhere on the side surface of a cylinder of which the diameter D is larger than the 3D object. The position of the laser sensor is noted P. P is defined with two variables, Zp and α, also depicted on Figure 2 c). Zp defines the height of the probe on the cylinder, between 0 and Zp_{max}. α defines the revolution angle around the object, between 0 and 360°. The laser propagates from P, and by default points towards Z axis and propagates parallel to the ground. On Figure 2 b) and c), vectors $\overrightarrow{u_1}$ and $\overrightarrow{u_2}$ represents default laser propagation respectively before and after applying revolution angle α.

Two more variables, φ and θ allow to deviate from the default orientation. φ, namely the "yaw angle", allows to deviate from the Z axis. After successively applying Zp, α and yaw angle φ in this order, the laser propagates as depicted by vector $\overrightarrow{u_3}$ on Figure 2. θ, namely the "pitch angle", allows to deviate from a propagation parallel to the ground. After successively applying Zp, α, φ and pitch angle θ in this order, the laser propagates as depicted by vector $\overrightarrow{u_4}$ on Figure 2 a). After receiving a query (Zp, α, φ, θ), the simulator propagates the laser with the query configuration ($\overrightarrow{u_4}$), until it touches an object, or reaches its maximum propagation length. Knowing the position, orientation and distance feedback of the laser sensor, the simulator computes the 3D position of the point reached during the acquisition. The acquired 3D point is computed in the coordinate system (O, X, Y, Z). The simulator finally returns the 3D coordinates reached, Xt, Yt, Zt, plus an additional Boolean "T", indicating whether the acquisition reached or missed the object.

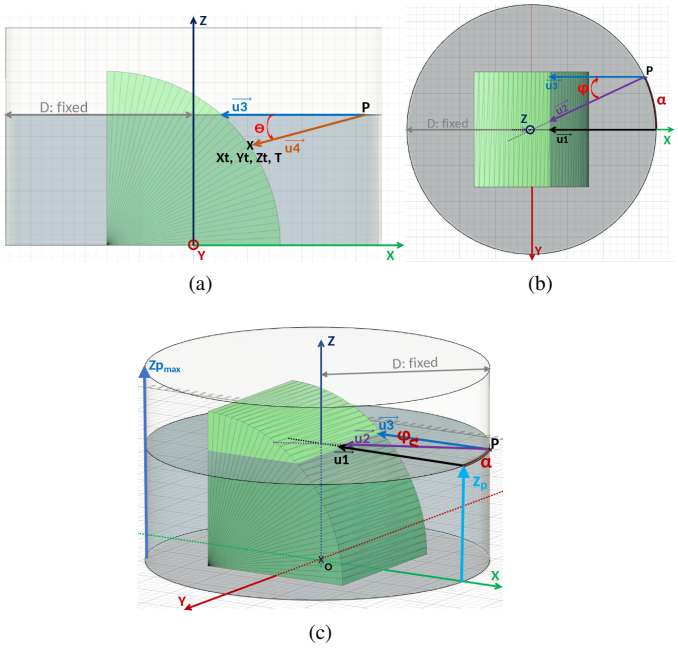

Fig. 2: Illustration of an object loaded in the simulator from 3 views ((a) Side; (b) Top; (c) 3D).

2.2. RL based active sampling

Similar to [12] and [13], we propose a RL based framework to jointly learn active exploration and classification of 3D objects positioned in the workspace. The Figure 3 gives an overview of the framework. The framework is made of (1) a point cloud storage (collected points), stacking the 3D points sampled from the workspace; (2) a Point Cloud Feature Extractor, that encodes the point cloud into a permutation invariant latent representation; (3) an action prediction module, that infers the configurations of the next points acquisitions (Zp, α, φ, θ) from the latent representation; and (4) a classifier, predicting the class of the object explored, from the latent representation. The whole framework is trained end-to-end using an offline RL algorithm, so that the learnt exploration strategy ensures the success of the 3D object recognition. The following sections detail (1) the modules of the proposed framework and (2) the RL algorithm and the proposed reward.

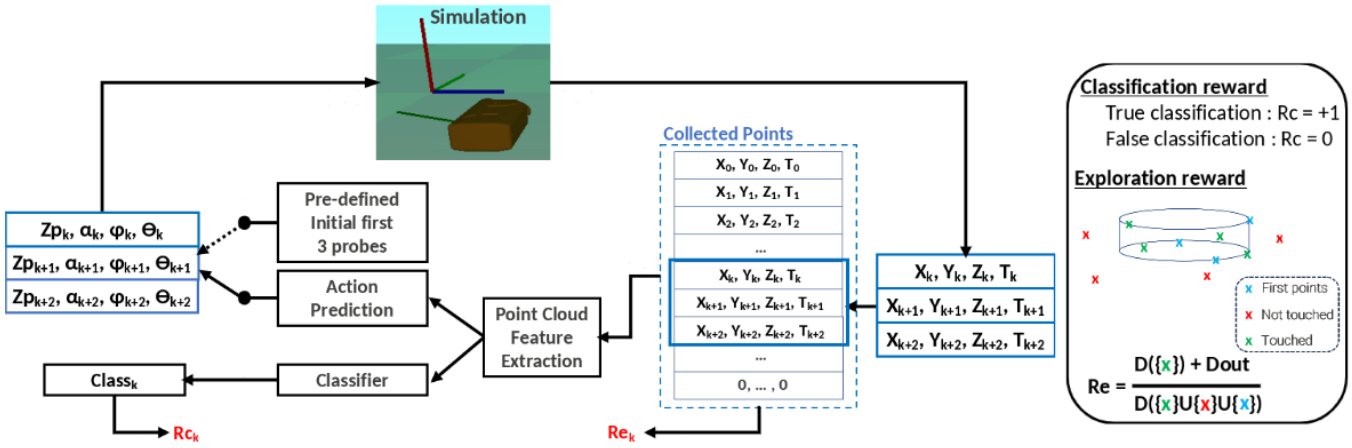

Fig. 3: Overview of the proposed policy network (left) and proposed reward function terms (right).

2.2.1. *Framework details*

The framework is trained to sample the workspace 3 points by 3 points, and to classify the object after each acquisition of 3 new points (empirical choice). N acquisition steps are performed successively, leading to a total of 3*N points sampled on the workspace at the end of the exploration. At the very beginning of the exploration, the point cloud storage is initialized with 3*N "empty points", filled with zero values. The exploration starts by requesting 3 pre-defined/hard-coded samplings to the simulator : $(Zp_{0:2}, \alpha_{0:2}, \varphi_{0:2}, \theta_{0:2})$. The point cloud is then updated with the 3 first acquired points : $(X_{0:2}, Y_{0:2}, Z_{0:2}, T_{0:2})$.

From this configuration, for each step $k \in \{3n, n \in [1, N]\}$, the framework (1) tries to classify the object; (2) predicts the next three samplings $(Zp_{k:k+2}, \alpha_{k:k+2}, \varphi_{k:k+2}, \theta_{k:k+2})$ and requests them to the simulator, and (3) stores the three new points acquired $(X_{k:k+2}, Y_{k:k+2}, Z_{k:k+2}, T_{k:k+2})$ in the point cloud. The whole process is illustrated on the left side of Figure 3.

The point cloud feature extraction module is a light version of the "Per point Context Aware Representation" proposed in [13]. It leverages "Context Aware" (CA) and "Self-Attention Context Awareness" (SACA-A) operations from PointGrow [14] architecture, that allow to relate each of the collected points to the preceding ones. SACA-A takes the 3*N point-cloud as input, and outputs 3*N embeddings, each reflecting the global context of the previous sampled points. SACA-A processes each point independently through Multi-Layer Perceptrons (MLP) and aggregates global context information from previous points using average pooling, a permutation invariant operation, which means that every permutation of the input point cloud should result in the same latent encoding. In other words, the sampling order doesn't impact the latent representation.

The classifier and the action prediction module both leverage the latent point cloud representation to predict respectively the class probabilities and the next sampling configuration. Both classifier and action prediction module architectures, illustrated in Figure 4 (top and middle), are similar to the PCRN-FC classifier proposed in [13] : several per-point MLP, followed by a point-wise average pooling which aggregates point-cloud information while maintaining permutation invariance, completed by several fully connected layers (FC). The classifier ends with a softmax layer to predict class probabilities.

The action prediction module is made stochastic by predicting the parameters μ and σ of a Gaussian distribution for each sampling parameters, instead of deterministically predicting their values :

$$\begin{aligned} z_k &\backsim \mathcal{N}(\mu_{z_k}, \sigma^2_{z_k}), \quad \varphi_k \backsim \mathcal{N}(\mu_{\varphi_k}, \sigma^2_{\varphi_k}) \\ \alpha_k &\backsim \mathcal{N}(\mu_{\alpha_k}, \sigma^2_{\alpha_k}), \quad \theta_k \backsim \mathcal{N}(\mu_{\theta_k}, \sigma^2_{\theta_k}) \end{aligned} \tag{1}$$

The stochasticity of the action decision module allows to use state-of-the-art RL algorithms, such as Soft Actor Critic (SAC) [15].

Finally, as illustrated in Figure 4, 3 sampling configurations are predicted at once, since the proposed framework explores the working-space 3 points by 3 points.

2.2.2. *Reinforcement learning algorithm and reward*

Reinforcement Learning is a category of machine learning algorithms adapted to sequential decision making problems. We use Soft Actor Critic (SAC) [15] RL algorithm to learn our sequential 3D point sampling and object recognition task. The objective of SAC is to maximize $J(\pi)$, defined by Equation 2, where the policy π is the decision making agent, predicting the next action a_t from the current state s_t.

$$J(\pi) = \mathop{\mathbb{E}}_{\tau \backsim \pi} \left[\sum_{t=0}^{T} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \Big) \right] \tag{2}$$

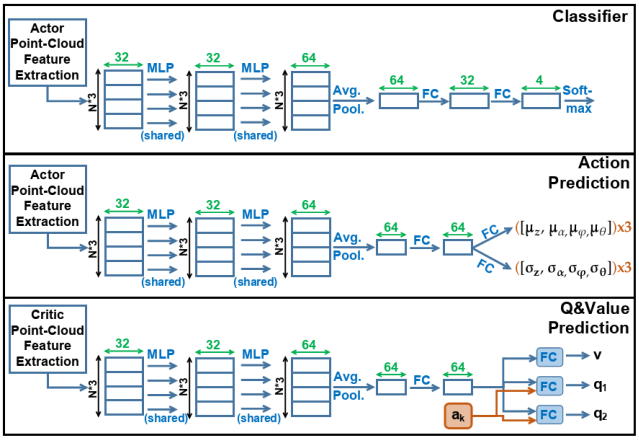

Fig. 4: Classifier, action prediction and critic architectures.

In our target use case, the state s_t is defined by the collected point cloud at time t. We define the action a_t as the concatenation of (1) the configurations of the next 3 samplings and (2) the classification probabilities. The whole framework represented in the left side of the Figure 3 and detailed in the previous section thus defines the policy. The SAC objective is divided in 2 terms, balanced by a tradeoff coefficient α : the reward $R(s_t, a_t, s_{t+1})$, that will be detailed thereafter, and the entropy of the policy $\mathcal{H}(\pi(\cdot|s_t))$, that encourages exploration of diverse policy strategies.

The reward R is a function that represents the usefulness of a transition (s_t, a_t, s_{t+1}) with regard to the target task. For the task of joint exploration and recognition of a 3D object, the reward is defined as the sum of a classification reward r_c with an exploration reward r_e.

r_c is representative of the ultimate goal of the framework: the 3D object recognition. Concretely, at each timestep $t \in [1, N]$, the policy tries to classify the object explored, from the collected point cloud. If it succeeds, $r_c = +1$. If it fails, $r_c = 0$. r_c thus ensures that (1) the classifier learns its recognition task; (2) the point cloud feature extraction module learns a representation that facilitates the discrimination of the objects geometries from few points; (3) the action prediction module learns to sample points as efficiently as possible regarding the classification task.

Additionally, r_e facilitates the training by providing information on the quality of the exploration strategy, by focusing on 2 aspects that are proven to affect the performances of 3D point-clouds classification : missing parts and noisy points (samplings that missed the object). Let B_t, G_t and R_t be respectively the set of 3 initial points, the set of points that touched the object and the set of points that missed the object. The 3 sets are respectively represented by Blue, Green and Red points on the right side of Figure 3. We define the sum of inter-points distances on a set of points A as

$$D(A) = \sum_{i=0}^{|A|} \sum_{j=0}^{i-1} d_e(A_i, A_j), \tag{3}$$

with $d_e(A_i, A_j)$ being the euclidean distance between A_i and A_j. A first version of r_e is introduced as the ratio :

$$r_e = \frac{D(G_t)}{D(G_t \cup R_t \cup B_t)} \tag{4}$$

This both encourages to sample points on the object and to broaden the exploration on the object.

Moreover, to further encourage the exploration of the borders of the object, we further introduce an additional term, D_{out} to give credit to samplings that missed but brushed the object. Let (d_k) be the sampling line defined by $(P_k, \overrightarrow{u_{4_k}})$, $\overrightarrow{u_{4_t}}$ being a direction vector of the sampling line and P_k a point on the sampling line, as defined on Figure 2. For each (d_k) corresponding to missed sampling, D_{out} takes into account the closest point to (d_k) in $(G_t \cup B_t)$, namely $A_{min}((d_k))$:

$$A_{min}((d_k)) = \underset{A \in (G_t \cup B_t)}{\operatorname{argmin}} (d_c(A, (d_k)), \tag{5}$$

$$with \quad d_c(A, (d_k)) = \frac{||\overrightarrow{PA} \wedge \overrightarrow{u_{4_k}}||}{||\overrightarrow{u_{4_k}}||} \tag{6}$$

D_{out} aims to give credit to missed samplings (d_k), regarding their value in terms of exploration breath if it reached corresponding $A_{min}((d_k))$ points, but normalized by the distance $d_c(A_{min}((d_k)), (d_k))$. The intuition is that missed samplings that brush the object may be interesting to explore the limits of the object, while samplings missing the object with a large margin are not interesting in terms of object delineation. The set of missed samplings that generated R_t is denoted as M_{R_t}, and D_{out_t} is defined as :

$$D_{out_t} = \sum_{(d_i) \in M_{R_t}} \sum_{A \in (G_t \cup B_t)} \frac{d_e(A_{min}((d_i)), A)}{d_c(A_{min}((d_i)), (d_i))}, \tag{7}$$

The final version of r_e, including D_{out} term, is defined as :

$$r_e = \frac{D(G_t) + D_{out_t}}{D(G_t \cup R_t \cup B_t)} \tag{8}$$

Note that on Figure 3, Rc_k and Re_k correspond to the classification and exploration rewards after collecting the last three points (k, k+1, k+2).

The last uncovered part of our framework is the critic network. SAC is an algorithm that concurrently learns an actor, the policy, and a critic, that tries to predict the sum of discounted future rewards from a given state-action pair. SAC especially leverages the double-Q trick from TD3 RL algorithm [16]. We use the Stable-Baselines-1 [17] implementation of SAC, which also needs a value-function prediction. This means that the trained framework needs an independent network to model two Q functions and one value function. The structure of the critic network is presented in the bottom part of the Figure 4.

3. EXPERIMENTS

Similar to [12], we built a dataset containing 4 3D geometric objects, represented in Figure 1. The objects were randomly chosen and loaded in the simulator at the origin with random rotation R_z around Z axis.In this study, we consider N=4 iterative acquisition steps, leading to a final point-cloud density of 12 points (similar to [13]).

We compared our proposed framework with state-of-the-art LSTM based tactile exploration model [12], adapted to our use case, which we denote "LSTM" model. We further compared our approach to (1) a lower-bound case, denoted "Rand. Explo + PointNet", where state-of-the-art point-cloud classifiers were fed with points randomly sampled from the simulator and (2) an upper-bound case, denoted "Hom. Explo + PointNet", where state-of-the-art point-cloud classifiers were fed with points homogeneously sampled on the 3D object. For the homogeneous sampling, θ and φ were fixed to 0, and Zp and α were uniformly chosen between their minimum and maximum values. The ideal homogeneous exploration, with all points on the object, was only manageable in the simulated setup, because the object position is fixed at the origin, and because we know the minimum dimensions of our objects. We chose PointNet [1] as state-of-the-art point-cloud classifier, as it proved to be robust to sparse and noisy point clouds [11]. Four independent point-cloud classifiers were trained for both Random and Homogeneous exploration, using respectively 3, 6, 9 and 12 points.

Compared to our approach, the independent classifiers have the advantage of being density-specific, while our model must handle all 3, 6, 9 and 12 points densities. An additional experiment was thus performed to directly compare the exploration efficiency of our trained framework to the exploration efficiency of random and homogeneous approaches. We used our trained framework to sample points from the simulator, and fed them to 4 PointNet classifiers, handling respectively 3, 6, 9 and 12 points densities. The experiment was denoted as "Ours + PointNet".

Our framework was trained using the stable-baselines-1 [17] SAC implementation, with default hyper-parameters, except for a batch size of 512. The LSTM model was trained with default hyper-parameters proposed in [12], except for a batch size of 512. PointNet models were all trained on 80 epochs, using 22000 training point clouds, balanced between the 4 classes. All models were evaluated by computing the accuracy of the best version obtained during training over 1400 objects per class.

The resulting accuracies, after different number of probes are summarized in the Table 1. Our proposed framework outperforms the LSTM model at each probes. It even outperforms the "Hom. Explo + PointNet" approach after the 6th probe, but performs slightly worse after 9th and 12th probes. Performances after 3rd glance are not comparable, since the first 3 points are pre-defined in our framework. However, our proposed framework learns an exploration strategy that appears to be more efficient than the homogeneous one, as training PointNet classifiers from point clouds sampled by our trained framework shows the best overall accuracies, at each probes.

Table 1: Accuracies of point-cloud classifiers with different exploration approaches.

Sampled Points	3	6	9	12
Rand. Explo + PointNet	58.48	71.52	78.68	83.20
Hom. Explo + PointNet	70.29	86.48	**100**	**100**
LSTM	56.31	81.10	87.07	90.30
Ours	90.52	98.85	99.92	99.95
Ours + PointNet	90.67	**99.44**	**100**	**100**

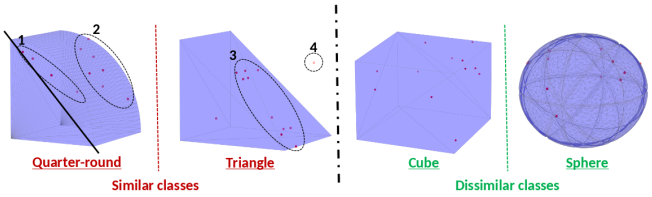

Fig. 5: Visualisation of 3D points sampled by our trained agent (red dots), projected on corresponding meshes.

Finally, the Figure 5 shows the points sampled by our trained framework, projected on the corresponding 3D objects. When there was no ambiguity on the class (sphere and cube), the sampled points broadly explored the surface of the objects. However, for similar classes, triangle and quarter-round, the exploration focused on relevant areas, such as the potential edges of a triangle in area "1", or the top surface of the objects (areas "2" and "3"), of which the curvature discriminates the two classes. Area "4" even showed that missed points are acceptable if they are useful to delineate critical parts of the 3D object.

4. CONCLUSION

In this work, we proposed a new RL based framework associated with a hand-crafted reward function, that enables a robot to efficiently sample few points in its workspace to recognize the 3D object presented. Our model learnt an exploration strategy that broadly explores the objects and avoids missed samplings, but is also able to focus on relevant local parts that allows to discriminate similar objects. Our model achieved better accuracy than existing models on end-to-end exploration-and-classification task. Moreover, the learnt exploration strategy enables to acquire points that, when fed to a SOTA point-cloud classifier, lead to better classification accuracy than the homogeneous exploration strategy.

5. REFERENCES

[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[2] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[3] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1588–1597.

[4] Saeid Asgari Taghanaki, Kaveh Hassani, Pradeep Kumar Jayaraman, Amir Hosein Khasahmadi, and Tonya Custis, "Pointmask: Towards interpretable and bias-resilient point cloud processing," *arXiv preprint arXiv:2007.04525*, 2020.

[5] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan, "Densepoint: Learning densely contextual representation for efficient point cloud processing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5239–5248.

[6] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen, "Pointcnn: Convolution on x-transformed points," *arXiv preprint arXiv:1801.07791*, 2018.

[7] Wenxuan Wu, Zhongang Qi, and Li Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[8] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8895–8904.

[9] Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov, "Effective rotation-invariant point cnn with spherical harmonics kernels," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 47–56.

[10] Yang You, Yujing Lou, Qi Liu, Yu-Wing Tai, Lizhuang Ma, Cewu Lu, and Weiming Wang, "Pointwise rotation-invariant network with adaptive sampling and 3d spherical voxel convolution," *arXiv preprint arXiv:1811.09361*, 2018.

[11] Saeid Asgari Taghanaki, Jieliang Luo, Ran Zhang, Ye Wang, Pradeep Kumar Jayaraman, and Krishna Murthy Jatavallabhula, "Robustpointset: A dataset for benchmarking robustness of point cloud classifiers," *arXiv preprint arXiv:2011.11572*, 2020.

[12] Sascha Fleer, Alexandra Moringen, Roberta L. Klatzky, and Helge Ritter, "Learning efficient haptic shape exploration with a rigid tactile sensor array," *PLoS ONE*, vol. 15, no. 1, pp. 1–22, 2020.

[13] Kevin Riou, Suiyi Ling, Guillaume Gallot, and Patrick Le Callet, "Seeing by haptic glance: Reinforcement learning based 3d object recognition," in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3637–3641.

[14] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma, "Pointgrow: Autoregressively learned point cloud generation with self-attention," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 61–70.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[16] Scott Fujimoto, Herke Hoof, and David Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[17] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.