# Reinforcement Learning Based Tactile Sensing for Active Point Cloud Acquisition, Recognition and Localization

Kevin Riou, Kaiwen Dong, Kevin Subrin, Patrick Le Callet

# Reinforcement Learning Based Tactile Sensing for Active point cloud Acquisition, Recognition and Localization

Kevin Riou, *Member, IEEE*, Kaiwen Dong, Kevin Subrin, *Member, IEEE*, Patrick Le Callet, *Fellow, IEEE*

***Abstract*—Traditional passive point cloud acquisition systems, such as lidars or stereo cameras, can be impractical in real-life and industrial use cases. Firstly, some extreme environments may preclude the use of these sensors. Secondly, they capture information from the entire scene instead of focusing on areas relevant to the end task, such as object recognition and localization. In contrast, we propose to train a Reinforcement Learning (RL) agent with dual objectives: i) control a robot equipped with a tactile (or laser) sensor to iteratively collect a few relevant points from the scene, and ii) recognize and localize objects from the sparse point cloud which has been collected. The iterative point sampling strategy, referred to as an active sampling strategy, is jointly trained with the classifier and the pose estimator to ensure efficient exploration that focuses on areas relevant to the recognition task. To achive these two objectives, we introduce three RL reward terms: classification, exploration, and pose estimation rewards. These rewards serve the purpose of offering guidance and supervision in their respective domain, allowing us to delve into their individual impacts and contributions. We compare the proposed framework to both active sampling strategies and passive hard-coded sampling strategies coupled with state-of-the-art point cloud classifiers. Furthermore, we evaluate our framework in realistic scenarios, considering realistic and similar objects, as well as accounting for uncertainty in the object's position in the workspace.**

***Index Terms*—Tactile Perception, Robotics, Extreme Environments, 3D Objects Recognition, Active Point-clouds Acquisition, Reinforcement Learning**

## I. Introduction

Vision-based 3D sensors, including Lidars and 3D cameras, enable the scanning of 3D scenes to recover dense 3D point-clouds from them. With the widespread deployment of such 3D sensors ranging from autonomous vehicles [1], mobile phones [2] and tablets [3], and industrial scenarios [4], [5], has led to a remarkable surge in the development of deep learning models designed for the processing of dense 3D point clouds. In particular, deep learning models have become increasingly accurate at localizing, recognizing, and delineating objects in the 3D scene from dense point clouds. However, extreme environments might not always permit the use of such vision-based sensors to acquire point cloud information. Fig 1 highlights a typical extreme use case where information is required for localizing and recognizing buried objects, e.g., buried mines, but where traditional vision-based sensors are not usable. Note that regarding buried mines, Ground Penetrating Radar (GPR) [6] is a crucial tool for rapid mine sweeping in conflict zones, however, manual tactile exploration remains the preferred method for post-conflict terrain cleaning, because of GPR limitations for certain types of mines or soil conditions [7]. Other extreme environment that might preclude the use of traditional vision-based sensors include those with poor lighting conditions, high levels of dust and troubled waters. In such scenarios, tactile perception, which is the ability to perceive objects/scenes through the sense of touch [8], can substitute for vision-based sensors in recovering point-cloud information from the target environment. In particular, in this work, we study how a system similar to the one highlighted in Fig. 1, can be used to localize and recognize objects in these kinds of extreme environments.

Fig. 1: In-house development of an autonomous probing system aimed at detecting, localizing and classifying underground objects.

The system samples one 3D point at a time by driving a rigid stick in a given direction, until contact with an object is detected using a contact sensor [9]. A point-cloud can be further constructed by iteratively sampling points from the scene. While this system can only generate extremely sparse point clouds from the environment, it provides the opportunity to strategically select point sampling locations to effectively localize and recognize target objects. The strategy employed to control the iterative 3D point sampling, achieved

Kévin Riou, Kévin Subrin and Patrick Le Callet are with Nantes Université, Ecole Centrale Nantes, CNRS, LS2N, UMR 6004, Nantes, France (e-mail: kevin.riou, kevin.subrin, patrick.lecallet@univ-nantes.fr)

Kaiwen Dong is with the IoT Perception Mine Research Center, China University of Mining and Technology, Xuzhou, China. (e-mail: dongkaiwen@cumt.edu.cn). Corresponding Author: Kévin Riou

Patrick Le Callet is also with Institut universitaire de France (IUF), France

by directing the tactile probe in specific directions, is the cornerstone of this work. We will refer to this control strategy as the "exploration strategy", because it defines how the environment will be explored. In a preliminary work [10], we introduced an active point cloud acquisition framework for 3D object recognition. The exploration strategy was denoted as "active" because each 3D point sampling was adaptively determined based on the context provided by the 3D points that had already been collected. This contrasted with "passive" exploration strategies, where the sequence of 3D points was always collected following the same sampling routine. The proposed framework relied on a deep learning model, that simultaneously learned an exploration strategy to control the robot and a 3D point cloud classifier, both aimed at maximizing classification performance. This way, the training of the exploration strategy was guided by the classification performance, ensuring that each collected point provided the most information for the 3D recognition task.

The framework was trained using a Reinforcement Learning (RL) algorithm, with the objective of maximizing two reward terms: the classification reward term, and the exploration reward term. The classification reward term provided a positive reward in the case of successful object classification. The exploration reward term was introduced to enhance the training of the sampling strategy. Its goal was to reduce "Missing Parts", which occur when the collected point-cloud doesn't uniformly cover the target object, and "Noisy Points", two types of disturbances that have been shown to affect state-of-the-art point cloud classifiers [11]. The exploration term encouraged the model to explore target objects broadly to avoid missing parts, while also ensuring as many samplings as possible on the object, as a sampling that misses the object results in a noisy point.

In this work, we build upon the original framework and propose the following novel contributions. i) We provide a detailed analysis of the contributions of the different reward terms (exploration and classification) to the overall framework's performance. ii) We introduce an additional reward term that estimates the object's position on the workspace and further investigate its impact on the classifier's performance. iii) We compare the behavior of hard-coded passive sampling strategies with the active sampling strategies learned by our framework under both simplified and more realistic scenarios, including realistic objects and varying object positions in the workspace. Our framework demonstrates higher classification accuracy compared to previous active exploration strategies and similar or better performance than a hard-coded "Homogeneous Exploration" strategy that uniformly samples the workspace. It achieves particularly strong classification results when all reward terms are used together.

This paper starts by reviewing related works in Section II. Section III describes our approach in details, as well as the in-house simulation developed to train and benchmark our models. Section IV provides experimental setup details and results. Finally, section V concludes and provides future related research directions.

## II. Related Works

In this section, we begin by surveying the existing landscape of solutions for object localization and recognition within dense fixed point clouds. Subsequently, we explore the realm of tactile perception solutions, which enable the active recovery of sparse 3D information from a given scene. In this context, we also delve into the integration strategies for training the previously mentioned object recognition and localization models within the framework of active exploration.

### A. Classifying and Locating objects from point clouds

Deep learning architectures proposed for 3D point cloud classification and segmentation [12] have exploded during the last decade. The architectures used for point cloud processing can be mainly divided into 3 categories [12]: graph-based methods, convolution-based methods and point-wise Multi-Layer-Perceptrons (MLP) methods.

Graph-based methods represent the point cloud as a graph, where each node represents a point, and where edges are generated based on the points neighborhood. Feature learning can further happen either in the spatial domain, e.g., using graph-convolutional neural networks [13], [14] or in the spectral domain [15].

Convolution-based methods apply convolution kernels to the point clouds based on the spatial distribution of the points [16]–[18].

The PointNet architecture [19] introduced the notion of point-wise MLP methods. PointNet processes point-wise features independently using MLP and fuses them using a permutation invariant function, e.g., a max-pooling layer. The permutation invariance guarantees that permuting the points in the point cloud doesn't affect the prediction results. Dozens of architectures were proposed based on PointNet, including PointNet++ [20], which allows to capture finer geometric strutures and PointWeb [21] that leverages local neighborhood context to improve point features. Sun et al. [22] further proposed the promising Context Awareness (CA) and Self Attention Context Awareness (SACA) operations, to incorporate global context information about the point cloud into the individual points-wise features. While they applied it to point cloud generation, these operations can be also be applied on classification models. PointNet classifiers have recently been proven to be particularly robust to sparse and noisy point clouds [11] compared to other baselines, which is particularly interesting for our use case. The sparsest point cloud evaluated by Taghanaki et al. [11] however still contained 128 points per object for classification, which exceeds the number of points considered in our use case.

Neverthless, most of the existing architectures are trained on pre-collected datasets, which are constructed using either vision based sensors such as 3D cameras [23]–[25] or lidars [24], or synthetic data [26]. These datasets tipically contain thousands of points per object or per scene.

Regarding object localization, existing architectures designed for dense point clouds operate in 2 steps : (1) they segment sub-parts/objects within the dense point cloud [27] or utilize mechanisms to focus on relevant smaller regions

of a large point cloud [28]; and (2) they predict the refined coordinates of target objects, either through an additional architecture [27], or dedicated prediction heads [28], [29]. In our case, as the exploration strategy's goal is to directly focus on the target object, we can skip the first step and equip our architecture with both a classification and a pose estimation head.

### B. Tactile Perception

Tactile Perception [8] is frequently utilized to assist robots in gaining a better understanding of their surroundings. It involves interpreting and representing tactile sensory information to observe the properties of objects.Luo et al. [8] categorize tactile sensors in 3 categories: (1) Single-point contact sensors, that can detect object-sensor contacts and eventually recover the 3D position of these contact points [30]. (2) High spatial resolution tactile sensors, including tactile arrays [31], artificial skins [32] and vision-based fingertip sensors [33], that can be easily embedded on robots grippers. (3) Large-area tactile sensor [34] that are designed to cover large, curved body parts of robots. In this study, we simulate a robust type of single-point contact sensor, similar to the one illustrated in Fig 1. This sensor allows us to retrieve 3D points, even in challenging environments, like when examining buried objects.

Many active tactile perception frameworks have been proposed to effectively explore and characterize an environment from touch. Gaussian processes were primarily used to model the explored space and further focus samplings on specific areas, depending on the uncertainty on the current environment modeling. Jamali et al. [35] utilized Gaussian processes in their work with fingertip sensors, and a similar approach was employed by Kaboli et al. [36] with multi-modal skin sensor to characterize objects in the target environment. Gaussian Processes were also applied to robots to simultaneously explore tactile information and refine grasp precision when dealing with unknown objects [37]. More recently, tactile exploration strategies learned through reinforcement learning (RL) algorithms, as discussed in studies such as [38], [39], have been proposed. These strategies aim to simultaneously learn an object exploration strategy and an object classifier, allowing both tasks to mutually enhance each other during the training phase. Further RL reward functions were also investigated [10] to improve the efficiency of the exploration strategy, in order to ultimately enhance the classifier's performance.

## III. Approach

We consider a simulated workspace, where unknown objects need to be quickly identified and localized, without access to a camera. The setup is depicted in Fig 2. A poly-articulated robot is equipped with a laser-based distance sensor that allows to simulate the acquisition of 3D points in the workspace from a single-point contact sensor, as shown in the Fig 1. The robot has sufficient degrees of freedom to acquire 3D points throughout the workspace. An RL agent is trained to sample the points that are the most informative for the classification and pose estimation tasks. The setup is simulated to facilitate the training and validation of the proposed solution. Two different sets of objects can be loaded into the simulation: the realistic and the geometric object sets. Detailed descriptions of these sets can be found in section . IV-A.

In this section, Part A first explains the simulated setup, which is similar to the one proposed in [10], but emphasizes novel features included to enable the evaluation of more realistic use cases. Secondly, it discusses the challenges associated to a deployment on a real setup, similar to the one presented in Fig 2. Part B explains the framework proposed to jointly learn active exploration, localization and classification of 3D objects positioned in the simulated workspace. Part C finally provides details about the employed reinforcement algorithm and the specific reward terms designed to train the framework.

### A. Active sampling setup

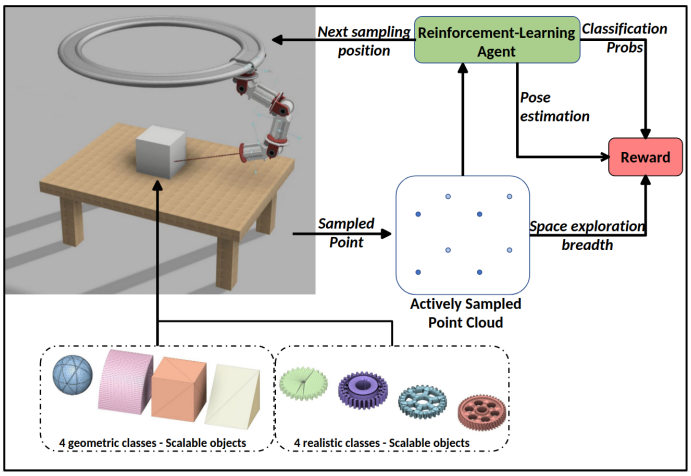


Fig. 2: Overview of the RL-based approach.

*1) Simulated setup:* The simulator consists of a 3D environment where 3D objects can be loaded at the origin "O", as depicted in Fig 3. The Z=0 plane in Fig 3 is equivalent to the top of the table in Fig 2. To facilitate the evaluation of the proposed models in more realistic scenarios, where objects could be randomly placed in the workspace, we can also introduce random offsets for the objects along both X and Y axes. The object position offsets along X and Y axis are respectively denoted as $X_o$ and $Y_o$. The objects can be rotated around Z axis. Note that if objects are both rotated and shifted with position offsets, the rotation is executed prior to the positional shift. The objects can further be re-scaled when they are loaded in the environment. 3D points can be acquired from the environment by activating the simulated laser sensor. Therefore, the 3D workspace and the loaded object can be actively explored by instructing the simulator to perform laser acquisition sequences. This setup allows active exploration of the workspace, where a model iteratively controls the position and orientation of the laser sensor, guided by previous acquisitions.

The Fig 3 highlights the inputs/outputs and the operation of the simulator. The simulator works by using input values that determine the laser's position and orientation to control

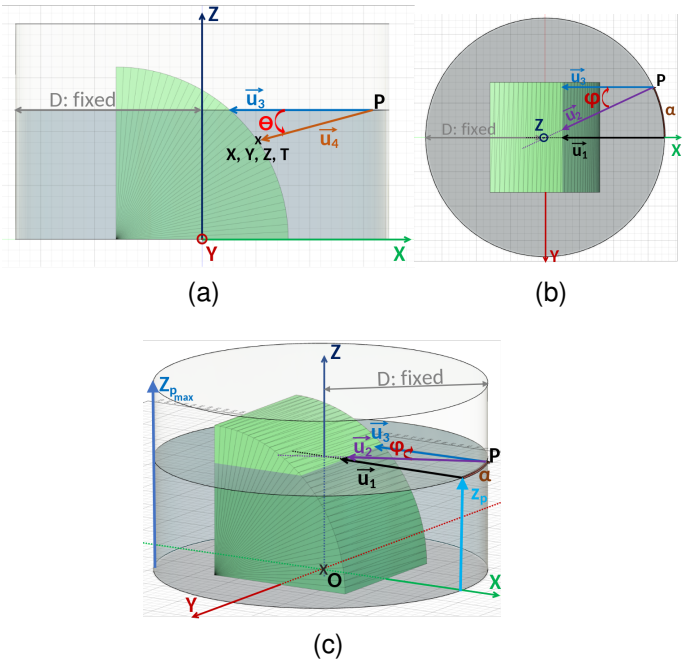


Fig. 3: Illustration of an object loaded in the simulator from 3 views ((a) Side; (b) Top; (c) 3D).

the laser acquisition system. It then records the 3D point that the laser reaches under this control.

More formally, the laser sensor can traverse the surface of a cylinder that encompasses the whole workspace. The diameter of the cylinder, noted as D, and the position of the laser on the cylinder, noted P, are both depicted in Fig 3(c). P is defined by two parameters: $Z_p \in [0, Z_{p_{max}}]$ and $\alpha \in [0, 360]$, prior one denotes the height of the probe on the cylinder, and later one denotes the revolution angle around the cylinder. As depicted in Fig 3(c), vector $\overrightarrow{u_1}$ and $\overrightarrow{u_2}$ represent two laser propagation routes that run parallel to the ground and towards Z axis, respectively before and after applying revolution angle $\alpha$.

Moreover, the orientation of the laser is determined by two variables: $\varphi$ and $\theta$, which represent the "yaw angle" and the "pitch angle", respectively. These angles allow the laser to deviate from its default propagation. $\varphi$ allows to deviate from default propagation towards Z axis, while $\theta$ allows to deviate from a propagation parallel to the ground. Concretely, vector $\overrightarrow{u_3}$ on Fig 3 represents the laser propagation after successively applying $Z_p$, $\alpha$ and $\varphi$ transformations in this order. Vector $\overrightarrow{u_4}$ represents the laser propagation after applying $\theta$ transformation to $\overrightarrow{u_3}$.

When the simulator receives a query set ($Z_p$, $\alpha$, $\varphi$, $\theta$), it propagates the laser with this configuration until it either touches the object or reaches the maximum propagation distance. It then computes the 3D point (X, Y, Z) reached with this configuration. The computed 3D point is returned, accompanied by a boolean value, T, indicating whether the laser actually made contact with an object or not. In the following part, we refer to this querying operation as "sampling" 3D points in the environment.

*2) Towards a real setup:* Since we began developing the hardware for our tactile exploration solutions, we've gathered insights to ease the transfer to a real setup. Unlike computer vision, our tactile strategy focuses on elements common to both simulation and reality, ignoring environment-specific backgrounds. The remaining gaps between simulation and reality stem from measurement errors and robotic hardware reachability limitations.

A sim-to-real transfer procedure is needed to model measurement error on the real hardware, incorporating uncertainties in 3D position due to factors like robot precision, repeatability, positioning error, or probe deformation (for solid mediums). It should also incorporates false positive rates in touch detection. These uncertainties must be integrated into the simulator and can be overestimated as a domain randomization strategy [40]. Uncertainties calibration ensures the simulation reflects real-world conditions, aiding generalization of trained models to real setups. For instance, with our real hardware, the average measurement error was estimated at about 2mm, peaking at 5mm, limiting target objects to sizes greater than 1cm. With our 6-degree-of-freedom robot, balancing parameters D and $\alpha$ (Section III-A1, Fig 3) is necessary. Our hardware restricted $\alpha$ to $[-\pi/2, \pi/2]$, restricting to probe only on surfaces facing the robot, which enabled exploration of up to approximately 1m wide surfaces.

Controlling the real robot based on the output of the trained model is straightforward. The parameters P and $\overrightarrow{u_4}$ represent the target position and orientation of the tool, respectively, which can be reached using the robot's inverse kinematics. The operation of the contact sensor at this position is managed by a dedicated controller, which moves the rigid stick until contact is made or the maximum range is reached.

## B. Framework architecture

*1) Overview of the proposed framework:* We propose a framework capable of sequentially predicting sampling configurations, so as to progressively construct a 3D point cloud. Consequently, this set of points will be used to classify and locate the 3D object in the workspace. The left side of Fig 4 gives an overview of the proposed framework. It iteratively predicts new points to sample based on the previously collected points. At each prediction step, the framework further predicts the class and location of the object using the acquired points. This framework is constructed with a deep neural network that can be trained end-to-end with an off-Policy RL algorithm. Importantly, the framework receives immediate rewards after each prediction, and these rewards are determined based on the classification and localization performances achieved using the current acquired point cloud. This way, each predicted sampling must provide the most relevant information for the classification and localization tasks. As a result, the exploration strategy is learned to optimize the success of 3D object recognition and localization. As illustrated in the left side of Fig 4, the proposed framework can be divided into 5 parts. (1) An action prediction module, that predicts the configuration of the next samplings from the latent representation of the already collected points; (2) a point

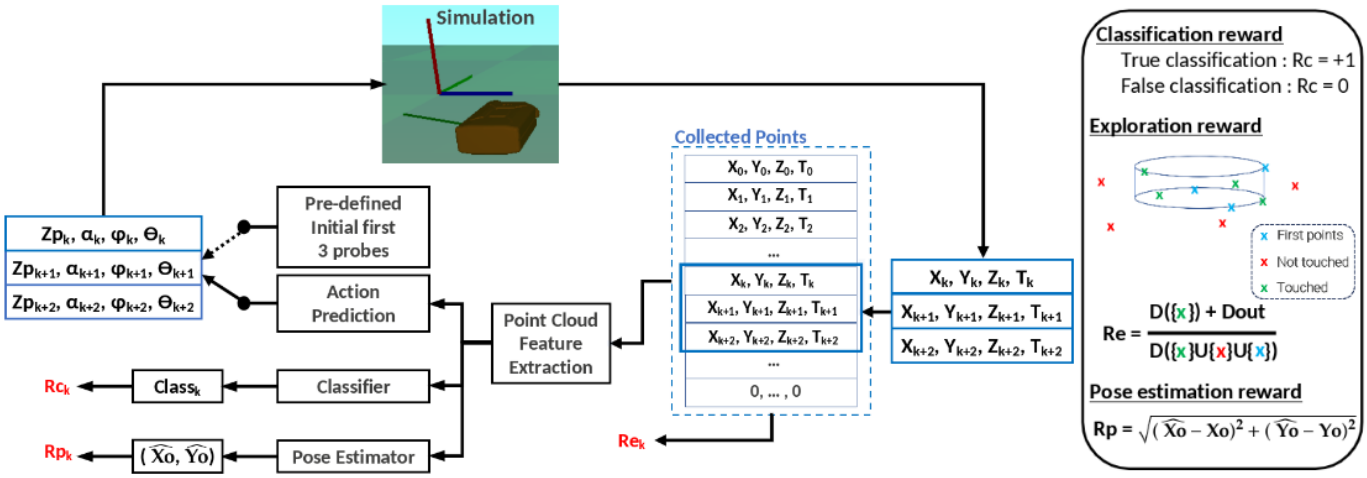


Fig. 4: Overview of the proposed policy network (left) and proposed reward function terms (right).

cloud storage, aiming to archive the 3D points collected by the samplings accumulated throughout all preceding steps; (3) a point cloud feature extractor, that encodes the accumulated point cloud from all the previous steps into a permutation invariant latent representation; (4) the classifier, that predicts the class of the object placed in the workspace from the latent representation of the accumulated point cloud; (5) the pose estimation module, that predicts the position $(X_o, Y_o)$ of the object in the workspace.

*2) Predicting series of samplings:* As illustrated on Fig 4 (left), it's important to acknowledge that the framework consistently predicts three samplings simultaneously. This choice represents a deliberate trade-off between two key considerations. Firstly, the framework receives a reward after each iteration. When a fixed total number of samplings is considered, predicting fewer samplings at once increases the frequency of reward reception, thus facilitating reinforcement learning optimization. However, predicting fewer samplings at once also leads to higher variation in point cloud densities supplied to the classifier and pose estimator. This variation can introduce challenges in their optimization processes, potentially increasing the noise in the associated reward terms. Moreover, when too few points are added to the point cloud at once, it may not significantly enhance classification and localization performance.

At the very beginning of the exploration, the point cloud storage is filled with zero values, which represents an empty point cloud. The exploration starts with 3 hard-coded sampling configurations, used to initialize the point cloud: $(Z_{p_{0:2}}, \alpha_{0:2}, \varphi_{0:2}, \theta_{0:2})$. The 3 initial samplings are chosen to cover uniformly the ranges of $\alpha$ and $Z_p$ parameters, with $\theta$ and $\varphi$ set to 0. This simple initialization provides an overview of the workspace to the model to start making its first predictions. After initialization, the point cloud storage is then updated with the 3 initially sampled points $(X_{0:2}, Y_{0:2}, Z_{0:2}, T_{0:2})$. Consequently, for each step $k \in \{3n, n \in [1, N]\}$, the framework (1) tries to classify and to localize the object; (2) predicts the next three samplings $(Z_{p_{k:k+2}}, \alpha_{k:k+2}, \varphi_{k:k+2}, \theta_{k:k+2})$ and requests them to the simulator, and (3) stores the three newly acquired points $(X_{k:k+2}, Y_{k:k+2}, Z_{k:k+2}, T_{k:k+2})$ in the point cloud storage. A total $3 \star N$ points are sampled from the object through the execution of $N$ acquisition steps. Details of various parameters can be found in the experiments section.

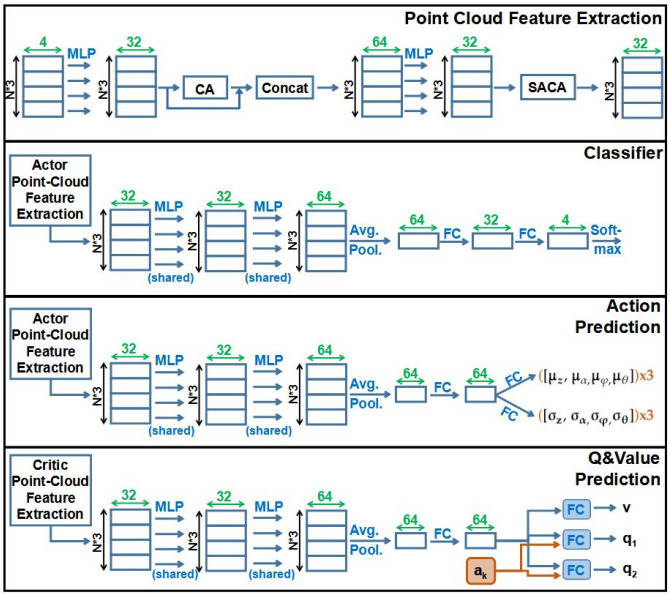


Fig. 5: Point cloud feature extraction, classifier, action prediction and Q&Value prediction architectures. Green values represents feature sizes.

*3) Point cloud feature extraction:* The point cloud feature extraction module is described in Fig 5. It is inspired from the "Per-point Context Aware Representation" proposed in [39]. The input of the module is the point cloud, that contains $3 \star N$ points. As described above, each point has a dimension of 4, prior 3 values representing the 3D point reached during the corresponding sampling, the last one is a boolean indicating if an object was actually touched during the sampling. These points are independently processed by a

set of MLPs sharing the same weights, in the PointNet [19] fashion. In the following, we refer to this operation as "per-point-MLP" operation. The features obtained from the first per-point-MLP are subsequently sent to a "Context Aware" operation (CA), overlapped by a Residual connection. The CA was introduced by Sun et al. [22] with the PointGrow architecture. The CA takes $3\star N$ point cloud features as input, and outputs $3 \star N$ embeddings. Each of these embeddings reflect the global context of the previously sampled points so far. Concretely, for each point in the cloud, CA returns the average of the features of the points collected previously. The resulting embeddings are finally sent to an additional per-point-MLP, followed by a "Self-Attention Context Awareness" operation (SACA-A). The SACA-A was also introduced in the PointGrow [22]. While the CA aggregates features of the previously collected points with a fixed average pooling, the SACA-A aggregates them with a weighted average pooling. The weights assigned to the previously collected points are predicted by a dedicated learnt sub-module.This sub-module itself is made of a CA operation overlapped by a Residual connection, followed by a per-point MLP. For both CA and SACA-A operations, we generally followed the configuration of PointGrow [22], except the feature sizes predicted by each of our MLP layers are set to 1/4 of original feature sizes. The reason behind this feature size reduction is the sparsity of the data used in our case. Additionally, we also added batch normalization after each MLP layer.

*4) Classification, action prediction and pose estimation:* The classifier and the action prediction module, also detailed on Fig 5, respectively infer the object class probabilities and the next samplings configurations from the current latent representation of the accumulated point cloud so far. Both the classifier and the action prediction module follow similar structures, except for the last layers. They are both made of two successive per-point-MLP layers, followed by an average pooling layer that aggregates information from the whole point cloud. Furthermore, several fully connected layers (FC) are used to reduce the aggregated features to the desired output dimensions. The very last layers structures will differ between classifier and action prediction branches. On one hand, the classifier predicts deterministic class probabilities through its softmax layer. On the other hand, the action prediction module is stochastic [41]. It predicts parameters $\mu$ and $\sigma$ of a Gaussian distribution for each component of the sampling configuration, instead of deterministically predicting their values :

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

Since the proposed framework explores the workspace 3 points by 3 points, the action prediction module predicts 3 sampling configurations at once, resulting in 3 sets of Gaussian distribution parameters predicted at once.

When the pose estimation module is employed, additional two sets of Gaussian Distribution parameters are predicted by the action prediction module, ($\mu_{Xo}$, $\sigma_{Xo}$) and ($\mu_{Yo}$, $\sigma_{Yo}$), so that :

$$X_{o_k} \backsim \mathcal{N}(\mu_{X_{o_k}}, \sigma^2_{X_{o_k}}), \quad Y_{o_k} \backsim \mathcal{N}(\mu_{Y_{o_k}}, \sigma^2_{Y_{o_k}}) \tag{2}$$

By doing so, the object position is modeled as a Gaussian noise added to the ideal position $(X_o = 0, Y_o = 0)$.

*5) Additional module dedicated to RL training:* The lower section of Fig 5 introduces an additional module dedicated to predicting Q-values ($q_1$ and $q_2$) and the state-value ($v$). These predictions play a crucial role in optimizing our architecture through the Soft Actor-Critic (SAC) reinforcement learning algorithm [42], detailed in Section III-C. Here, $a_k$ denotes predictions encompassing object class, object pose, and next sampling configuration. Conceptually, the Q-values are trained to estimate the advantage of these predictions with respect to the previously gathered point cloud, while the state-value assesses the quality of this point cloud for classification and localization tasks. The module's input is derived from a dedicated point cloud feature extraction module, which is structured identically to the one used for classification, action prediction, and pose estimation, but without weight sharing.

### C. RL algorithm and reward function terms

The proposed framework is trained using Soft Actor Critic (SAC) [42], an off-policy RL algorithm. SAC trains a policy $\pi$, the decision making agent, to take actions $a_t$ from current state $s_t$, so that it maximizes the objective $J(\pi)$ defined by Equation 3.

$$J(\pi) = \underset{\tau \backsim \pi}{\mathbb{E}} \left[ \sum_{t=0}^{T} \gamma^t \Big( R(s_t, a_t, s_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \Big) \right] \tag{3}$$

The objective represents the sum of rewards, denoted as $R(s_t, a_t, s_{t+1})$,that the agent receives for its decision-making throughout the whole sequence of visited states, spanning from step 0 to step T. The entropy term, $\mathcal{H}(\pi(\cdot|s_t))$, is added to encourage exploration of diverse policy strategies. The discount factor $\gamma < 1$ is an impatience term, favoring immediate rewards over late ones. In our use case, the state $s_t$ is defined as the point cloud collected at time $t$. The action $a_t$ is made of 3 components (1) the next 3 samplings configurations; (2) the classification probabilities; and (3) the object pose estimation $(\hat{X}_o, \hat{Y}_o)$. Equation 4 defines the reward as the weighted sum of three reward terms, namely, the classification reward $r_c$, the exploration reward $r_e$ and the pose estimation reward $r_p$. The weights $\beta_{r_c}$, $\beta_{r_e}$ and $\beta_{r_p}$ are part of hyperparameters of the framework.

$$R(s_t, a_t, s_{t+1}) = \beta_{r_c}.r^t_c + \beta_{r_e}.r^t_e + \beta_{r_p}.r^t_p \tag{4}$$

$r_c$ represents the classification performances of the framework. After each acquisition step $t \in [1, N]$, the model tries to classify the object. A successful classification is denoted as $r_c = 1$, while a failed one results in $r_c = 0$. This process serves 2 purposes: providing supervision for classifier training, and ensuring the action prediction module adheres to a sampling strategy that facilitates the classification.

The function of $r_e$ is to (1) ensure consistency in the exploration strategy during training, and (2) to enhance the exploration quality by penalizing missed parts and noisy points (samplings that missed the object). Formally, consider $B_t$ as the set of three initial points, $G_t$ as the set of points that made contact with the object, and $R_t$ as the set of points that

failed to touch the object. These sets are visually represented as Blue, Green, and Red points on the right side of Figure 4. We refer $D(A)$ which is defined by Equation 5, as the sum of inter-points distances of a set of points A.

$$D(A) = \sum_{i=0}^{|A|} \sum_{j=0}^{i-1} d_e(A_i, A_j), \tag{5}$$

where $d_e$ is the euclidean distance, and $|A|$ represents the number of points in $A$. On this basis, $r_e$ is defined by Equation 6. The left part of the equation encourages to sample points on the object of interest while simultaneously maximizing the distance between them, thereby broadening the exploration.

$$r_e^t = \frac{D(G_t)}{D(G_t \cup R_t \cup B_t)} + D_{out_t} \tag{6}$$

However, in certain cases. the samplings that failed to capture the object could hold relevance for the classification task. In particular, samplings that almost touched, or made slight contact with the object, could be valuable in outlining its boundaries. $r_e$ is thus enhanced with an additional term, namely $D_{out}$, that gives credit to missed samplings regarding the reward they would obtain if they reached the closest point in $(G_t \cup B_t)$. Noteworthy that the reward is normalized by their distance to this closest point. Formally, each sampling is represented as a line segment, denoted as $(d_k)$, characterized by $(P_k, \overrightarrow{u_{4_k}})$. Here, $\overrightarrow{u_{4_t}}$ represents the direction vector of the sampling line, and $P_k$ represents a point on the this line, as illustrated in Fig 3. The distance $d_c$ between a sampling $(d_k)$ and a point A is defined by Equation 7.

$$d_c(A, (d_k)) = \frac{||\overrightarrow{PA} \wedge \overrightarrow{u_{4_k}}||}{||\overrightarrow{u_{4_k}}||} \tag{7}$$

Therefore, the closest point to a sampling $(d_k)$ in $(G_t \cup B_t)$, denoted as $A_{min}((d_k))$, is defined by Equation 8.

$$A_{min}((d_k)) = \underset{A \in (G_t \cup B_t)}{\operatorname{argmin}} \ (d_c(A, (d_k)) \tag{8}$$

$D_{out_t}$ is finally defined by Equation 9, where $M_{R_t}$ is the set of missed samplings that resulted in $R_t$.

$$D_{out_t} = \sum_{(d_i) \in M_{R_t}} \sum_{A_j \in (G_t \cup B_t)} \frac{d_e(A_{min}((d_i)), A_j)}{d_c(A_{min}((d_i)), (d_i))}, \tag{9}$$

Finally, as defined on the right side of Fig 4, $r_p$ quantifies the Root Mean Square Error (RMSE) between the predicted and the actual pose of the object within the workspace.

## IV. Experiments

The experiments are divided into 6 parts. Part A describes the general experimental settings, common to all the experiments conducted. Part B recalls the main results of our preliminary work [10] and points out the limits of its experimental validations. Part C and D bring complementary studies on the initial results [10]. Part E investigates the usability of the framework in more realistic scenarios. Part F evaluates the novel pose estimator performances and its contributions to the overall framework.

### A. Experimental settings

In our preliminary work [10], we built a dataset with four 3D geometric objects to conduct experiments. In this study, we have extended this dataset with a second set of "realistic objects". These objects are characterized by greater complexity and a closer resemblance to each other, thus challenging the classification task even further. The geometric and realistic sets are respectively depicted in Fig 6 and Fig 7.

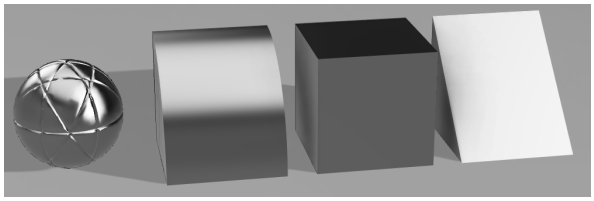

Fig. 6: Geometric objects : Sphere, Quarter-round, Cube and Triangle.

Fig. 7: Realistic objects : Gearwheel variations

For all the following experiments, during the framework training, the objects are randomly chosen among the used set, and placed with a random rotation $R_z$ around the $Z$ axis, which is illustrated in Fig 3. All experiments consider $N = 4$ iterative acquisition steps, leading to a total of 12 points sampled at the end of the exploration sequence, which is similar to [39]. The proposed RL-based framework is trained using the stable-baselines-1 [43] SAC implementation with default hyperparameters except for the batch size, which was set to 512. In particular, $\gamma = 0.99$, learning-rate=0.0003, and $\alpha$='auto', which means it is learnt during training. All RL-based trainings ran 600k steps, and were evaluated by computing the accuracy of the best model obtained during training on 1400 objects per class. Results corresponding to our framework are denoted as MTR-RL in result tables. It stands for Multi-Term-Reward Reinforcement Learning. Additionally, Table I details the simulation configuration and action ranges used in all the experiments. Finally, in experiments involving random object scaling, the loaded objects are rescaled by a randomly selected factor within the range of $[0.8, 1]$.

We compared our framework to both active-sampling-based and passive-sampling-based exploration methods for 3D object recognition. In the case of the active-sampling strategies, we compare to the state-of-the-art LSTM-based tactile exploration model [38], which we adapted to our use case. We denote it as "LSTM" model. For the passive-sampling strategies, we feed point clouds into a state-of-the-art classifier. These point clouds are sampled from the simulator using hard-coded sampling strategies. We employed two passive sampling strategies for the comparisons: (1) "Rand. Explo + PointNet", in which

TABLE I: Simulation parameters and action ranges (depicted in Fig 3). "Geom." and "Real." indicate parameter values respectively for the Geometric and the Realistic object sets.

| | min | max |
|---|---|---|
| $Z_p$ (Geom.) | 1 | 100 |
| $Z_p$ (Real.) | 0 | 50 |
| $\alpha$ | 0° | 360° |
| $\varphi$ | -30° | 30° |
| $\theta$ | -20° | 20° |
| D | 100mm | |
| Probe range | 200mm | |

point clouds are randomly sampled from the simulator, and (2) "Hom. Explo + PointNet", in which points are homogeneously sampled on the 3D object. For the homogeneous sampling, $\theta$ and $\varphi$ were set to 0, and $Z_p$ and $\alpha$ were uniformly chosen between their minimum and maximum values. The idea of the homogeneous sampling is to provide an exploration strategy that can scan the whole object to avoid "missing parts", known to affect point cloud classifiers. It is worth noting that the homogeneous sampling is an ideal scenario, which is possible in simulation. This is because the position of the object is known in advance, and the object is well positioned to facilitate such procedure, given the flexibility of the robotic arm movement around the object. On the contrary, our RL-based framework can be trained from scratch for any kind of scenario, adapting and learning an optimal exploration strategy tailored to each specific situation.

The proposed active-sampling based framework classifies the object after each 3 collected points, and for a total of 12 points. It means that the classification is successively made from 3, 6, 9, and 12 points. For fair comparisons, we also evaluated the passive-sampling strategies on 3, 6, 9 and 12 points densities. Concretely, we used the hard-coded sampling strategies to collect a point cloud dataset for each density and trained independent PointNet models [19] on each of these. PointNet has been chosen as it has been proven to be robust to sparse and noisy point clouds [11].

In contrast, our framework only uses one single classifier for all point cloud densities (3, 6, 9, and 12 points). Moreover, the classification utilizes a shared latent representation that is common to the classifier, the pose estimation head, and the action prediction head. This is not fair to compare our framework's classification results to the independent PointNet models trained for "Rand. Explo + PointNet" and "Hom. Explo + PointNet" experiments, since these models are density specific and fully dedicated to the classification task. We thus conducted further experiments to fairly compare the quality of the exploration strategy learned by our proposed framework. Similarly to passive models evaluation, we collected a point cloud dataset for each density, but using our trained active-sampling framework, and further trained independent PointNet models on each of these datasets. In experiments results, this scenario coupling our framework with independent PointNet classifiers is denoted "MTR-RL + PointNet".

Every PointNet models, for all "Rand. Explo + PointNet", "Hom. Explo + PointNet" and "MTR-RL + PointNet" approaches, were trained on 22000 training point clouds balanced between the 4 classes, and on 80 epochs. All PointNet models were also evaluated by computing the accuracy of the best version obtained during training over 1400 objects per class.

### *B. Comparison to prior works under ideal settings*

In our preliminary study [10], the proposed framework was evaluated on an ideal setup, where the position of the objects was frozen to $(X_o = 0, Y_o = 0)$, without rescaling and using only the geometric set. The resulting accuracies with both passive and active sampling-based approaches, after 3, 6, 9 and 12 points samplings, are summarized in Table II. Our framework (MTR-RL) outperforms LSTM model at each probe. It also outperforms the "Hom. Explo + PointNet" approach when only 6 points are sampled, and exhibits similar, yet slightly worse performances with 9 and 12 sampled points. However, training PointNet classifiers with point clouds sampled by our framework (MTR-RL + PointNet) allows to get better overall performances than training them with point clouds sampled using the homogeneous exploration "Hom. Explo + PointNet". The performance gap especially increases when decreasing the number of sampled points. Therefore, besides being trained from scratch, our framework allows to learn a more efficient exploration strategy than the hard-coded homogeneous version. Note that performances after the 3 first sampled points are not comparable, since these 3 initial points are pre-defined rather than learned in our framework.

TABLE II: Accuracies of point cloud classifiers on the geometric set, with frozen objects positions $(X_o = 0, Y_o = 0)$, without random object scaling, with different exploration approaches. MTR-RL models are trained using Classification and Exploration rewards ($\beta_c$=1, $\beta_e$=1 and $\beta_p$=0)

| | Sampled points | | | |
|---|---|---|---|---|
| | 3 | 6 | 9 | 12 |
| Passive sampling methods | | | | |
| Rand. Explo + PointNet | 58.48 | 71.52 | 78.68 | 83.20 |
| Hom. Explo + PointNet | 70.29 | 86.48 | **100** | **100** |
| Active sampling methods | | | | |
| LSTM | 56.31 | 81.10 | 87.07 | 90.30 |
| MTR-RL | 90.52 | 98.85 | 99.92 | 99.95 |
| MTR-RL + PointNet | 90.67 | **99.44** | **100** | **100** |

The Figure 8 shows points collected by our trained active sampling framework, projected on the 3D objects they were sampled from. On easily distinguishable shapes (cube, sphere), our model chooses to broadly explore the surface of the object, as intended, thanks to the exploration term incorporated into the proposed reward. However, on similar objects (Quarter-Round, Triangle), our model seems to seek for key areas on the objects, that would allow the classifier to discriminate them. For instance, the difference between the Quarter-Round and the Triangle comes from the curvature of the upper surface, area that the model seems to mainly focus on (areas "1", "2" and "3" on Figure 8). Moreover, the missed point in

area "4" seems to come from a sampling that brushed the upper surface of the triangle while searching for a potential "rounded surface", which is in line with the $D_{out}$ reward term expectations.

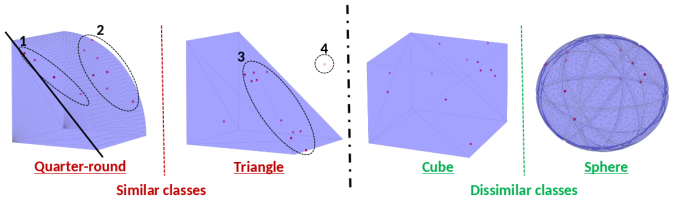


Fig. 8: Visualisation of 3D points sampled by our trained agent (red dots), projected on corresponding meshes. The agent was trained using both classification and full exploration reward and with no object positioning uncertainty.

The aforementioned results show that the proposed framework is able to learn an efficient exploration strategy but doesn't explicitly analyze which of its components contribute the most to its success. Moreover, such framework is of greatest interest in use cases where one needs to learn to discriminate new objects quickly without having to label large-scale datasets each time. With the proposed framework, only 3D models of the objects of interest are needed. The framework will learn to focus on relevant areas of the object and forget noisy information from rest of the workspace, which makes it fully trainable in simulation without sim-to-real gap, as long as the object positioning can be known. However, on a real case scenario, one can not expect to know the exact position of the object in the workspace. Therefore, the performances of the proposed framework should be evaluated under varying object positioning. Moreover, in such realistic scenario, it is interesting to investigate whether our framework could be extended with a pose estimation head, that could predict the actual position of the object on the workspace. Finally, an evaluation on realistic and more complex objects (e.g., the objects from the new set depicted in Figure 7) is needed to validate the usability of the framework on realistic scenarios. All the aforementioned questionings are addressed in the following sections.

### C. Reward Terms Contributions

Exploration rewards are aimed to give supervision to the exploration task by rewarding the model i) when it touches the object, ii) when it broadly explores the object, iii) if a missed sampling brushed the object. Fixing the position of the 3D object at the origin facilitates the exploration task, as the sampling hardware only needs to turn around it and to target Z axis to ensure a broad exploration that always reaches the object. On the contrary, a more realistic scenario where the position of the object is unknown, hardens the exploration task by i) favoring missed samplings and ii) adding uncertainties on the explored object. Intuitively, the second scenario gives more credit to the exploration rewards, as they help to robustify the exploration strategy learning. In this section, we experimentally study the contribution of the different reward terms under varying amount of uncertainty on the position of the object, relatively to its fixed position at the origin $(X_o = 0, Y_o = 0)$.

The Figure 9 shows the training curves of our proposed model, trained with classification reward only, namely 'Class Rwd', and trained with both classification and exploration rewards, namely 'Class + Touch Rwd'. Training curves with varying levels of uncertainty on the objects positions are reported. "0mm noise" corresponds to the experimental settings in section IV-B, where the objects are always placed at the origin $(X_o = 0, Y_o = 0)$. On the extreme opposite, "40mm noise" means that the object is randomly placed with deviations to the origin, uniformly chosen in $[-40mm, +40mm]$, on both X and Y axis.

Strikingly, the training curves with 0mm positioning noise appear to be really unstable. We emit the hypothesis that it is due to the choice of Soft Actor Critic as underlying RL Algorithm, which is an off-policy RL algorithm, known for its high sample-efficiency yet high instability [44]. Moreover, with 0mm position uncertainty, it is easier for the classifier to overfit on a given point cloud distribution corresponding to a specific exploration strategy. If the exploration strategy changes because of the RL algorithm unstability, the classifier will not be able to generalize to the corresponding new point cloud distribution. Using the exploration reward seems to reduce the accuracy drops during training with 0mm position uncertainty probably by ensuring a more consistent exploration strategy. On the contrary, with higher levels of position uncertainty, the trainings are slightly more unstable with the exploration reward, yet overall quite stable compared to the 0mm position uncertainty scenario, which tends to confirm that the uncertainty on the objects positions acts like a data augmentation strategy for the classifier.

More generally, the performance of the framework tends to decrease while increasing the amount of noise on the objects positions. However, as expected, the addition of the exploration term to the reward allows to maintain performances on par with the 0mm noise scenario, until 30mm position noise, and severely limits the accuracy drop with 40mm noise, compared to the 'Class Rwd' version.

### D. Passive vs Active acquisition methods under noisy object positioning

The Table III allows to fairly compare the exploration strategy learned by our framework with the hard-coded homogeneous sampling strategy, under varying levels of noise on the object position, on the geometric object set.

The exploration strategy allows to outreach 90% accuracy from only 6 points (2 acquisition steps), and outperforms the homogeneous exploration strategy on the sparser point cloud densities, for all position uncertainty levels. However, when denser point cloud are considered (12 points densities), the homogeneous exploration strategy gives better classification results.

However, after sampling 12 points, all results are close to 100% accuracy, regardless of the object position noise level considered. To be fairly representative of a realistic use case, more realistic objects needs to be considered in the models evaluation.

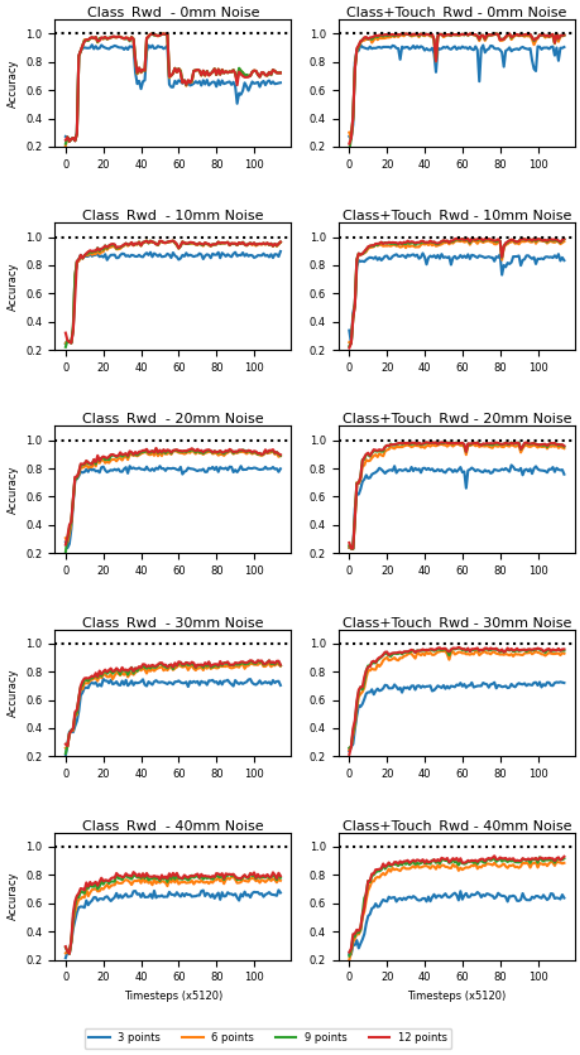


Fig. 9: Training curves of the proposed model without exploration reward (left column) and with exploration reward (right column) under varying uncertainty level on the object positioning (Noise). Training is done on the geometric set, without random objects scaling.

### *E. Evaluation of the framework on realistic use case*

Three levers were used to close the gap between the simulated scenario and a real-life industrial scenario : i) the uncertainty on the object positioning on the workspace; ii) the realism and the complexity of the considered objects and iii) the similarity between the objects to classify.

In this section, the realistic set of objects (Figure 7) was used as it contains 3D models of realistic and more complex objects Moreover, the four available objects share similar shapes, since they all represent variations of a gearwheel. To increase the difficulty of the recognition task, the objects can also be randomly rescaled when placed on the workspace,

TABLE III: Accuracies of point cloud classifiers with Hom. Explo and MTR-RL exploration strategies, under varying levels of position uncertainty (pos. noise), on the set of geometric objects. MTR-RL models are trained using Classification and Exploration rewards ($\beta_c$=1, $\beta_e$=1 and $\beta_p$=0).

| | Sampled points | | | |
|---|---|---|---|---|
| | 3 | 6 | 9 | 12 |
| Hom. Explo + PointNet | | | | |
| 0 mm pos. noise | 70.29 | 86.48 | 100 | **100** |
| 10 mm pos. noise | 58.48 | 86.46 | 98.55 | **100** |
| 20 mm pos. noise | 54.91 | 83.02 | 97.45 | **100** |
| 30 mm pos. noise | 54.69 | 80.51 | 96.02 | **99.90** |
| 40 mm pos. noise | 54.39 | 77.51 | 92.32 | **99.81** |
| MTR-RL + PointNet | | | | |
| 0 mm pos. noise | 90.67 | **99.44** | **100** | 100 |
| 10 mm pos. noise | 87.85 | **98.36** | **99.00** | 99.24 |
| 20 mm pos. noise | 80.62 | **98.68** | **99.48** | 99.65 |
| 30 mm pos. noise | 74.29 | **96.02** | **98.36** | 98.64 |
| 40 mm pos. noise | 68.5 | **91.56** | **95.28** | 96.65 |

which prevents the models to bias the classification on the objects sizes, and therefore forces them to seek for key semantic differences on the objects. Finally, uncertainty can also be added on the position of the object, as done in section IV-D on geometric objects. In this section, only extreme uncertainties are considered (0mm and 40mm).

TABLE IV: Accuracies of point cloud classifiers with different exploration approaches, on realistic objects, with 40mm position uncertainty, and random rescaling (Hardest Scenario). MTR-RL models are trained using Classification, Exploration and Localization rewards ($\beta_c$=1, $\beta_e$=0.5 and $\beta_p$=1).

| | Sampled points | | | |
|---|---|---|---|---|
| | 3 | 6 | 9 | 12 |
| Hom. Explo + PointNet | 56.27 | **80.38** | 80.3571 | 80.04 |
| MTR-RL | 46.0 | 71.66 | 72.52 | 72.52 |
| MTR-RL + PointNet | 56.92 | 78.37 | **80.63** | **82.05** |

The Table IV shows the performances of "Hom. Explo + PointNet", "MTR-RL" and "MTR-RL + PointNet" approaches under the most difficult scenario, namely, on realistic objects, with 40mm position uncertainty, and with random rescaling. The"MTR-RL + PointNet" approach shows the best results after 9 and 12 samplings, but is outperformed by the "Hom. Explo + PointNet" approach after 6 samplings. The choice of the hyperparameter $\beta_e$=0.5 is justified from Tables V and VI.

The Table V summarizes the performances of "Hom. Explo + PointNet", "MTR-RL" and "MTR-RL + PointNet" approaches with different combinations of the random object scaling and position uncertainty disturbances. The MTR-RL frameworks were trained with the hyperparameter settings formerly used on the geometric set ($\beta_c$=1, $\beta_e$=1 and $\beta_p$=0). First, we can notice that both "Homogeneous + PointNet" and "MTR-RL + PointNet" approaches show surprisingly high accuracies on the realistic set (nearing 100%), apart

from the hardest scenario where both position uncertainty and object rescaling are applied together (Scale, 40mm). Secondly, with both position uncertainty and object rescaling applied together, our framework (both "MTR-RL" and "MTR-RL + PointNet" approaches) shows significantly worse results than Hom. Explo + PointNet. However, by further tracking reward terms evolution during training, we noticed that the hardest scenario, "Scale, 40mm", increased the classification task difficulty way more than the exploration task, as the framework was still able to quickly increase $r_e$, while having difficulty to improve $r_c$. This resulted in an imbalance between $r_e$ and $r_c$, where $r_c$ became negligible. To compensate this effect, we tried to change the reward weights hyperparameters, which, by default, were set to $\beta_c$=$\beta_e$=1. (The location reward is voluntarily ignored in the first place, which means $\beta_p = 0$). We especially played with $\beta_e$ value to decrease $r_e$ preponderance. Table VI reports the evaluation results of our framework with $\beta_e$=0, $\beta_e$=0.5 and $\beta_e$=1 (3 values uniformly chosen between 0 and 1) on the "Scale, 40mm" scenario, while keeping $\beta_c$=1 and $\beta_p$=0.

The performances of our framework (MTR-RL + PointNet) varies from 66% to 79% accuracy (after 12 samplings) by only changing $\beta_e$ hyperparameter. Moreover, we can notice that the best performing end-to-end model (MTR-RL) doesn't always provide the best exploration strategy for the PointNet model. As a matter of fact, the end-to-end MTR-RL model performs better with $\beta_e$=0 (classification reward only), while the exploration strategy learned by the model with $\beta_e$=0.5 allows PointNet to achieve top accuracies, matching Homogeneous + PointNet accuracy on this scenario (Table V, "Scale, 40mm")

TABLE V: Accuracies of point cloud classifiers on realistic objects, processing point clouds from different exploration approaches. MTR-RL models are trained using Classification and Exploration rewards ($\beta_c$=1, $\beta_e$=1 and $\beta_p$=0)

| | Sampled points | | | |
|---|---|---|---|---|
| | 3 | 6 | 9 | 12 |
| Hom. Explo + PointNet | | | | |
| No scale, 0mm | 100 | 100 | 100 | 100 |
| Scale, 0mm | 96.28 | 99.44 | 99.39 | 99.27 |
| No scale, 40mm | 65.55 | 99.72 | 99.8 | 99.81 |
| Scale, 40mm | **56.27** | **80.38** | **80.3571** | **80.04** |
| MTR-RL + PointNet | | | | |
| No scale, 0mm | 100 | 99.98 | 100 | 100 |
| Scale, 0mm | 97.19 | 98.27 | .9896 | 99.29 |
| No scale, 40mm | 62.97 | 98.62 | 98.42 | 98.47 |
| Scale, 40mm | **56.45** | **65.83** | **65.64** | **66.31** |
| MTR-RL | | | | |
| No scale, 0mm | 100 | 100 | 100 | 100 |
| Scale, 0mm | 96.33 | 96.91 | 96.84 | 97.29 |
| No scale, 40mm | 56.69 | 95.49 | 96.85 | 96.2 |
| Scale, 40mm | **46.8** | **51.38** | **52.16** | **52.04** |

The overall best performances of our framework highlighted in Table IV are further obtained by adding the localization reward term during the model training ($\beta_p$=1). The contribution of this reward term is detailed in the next section.

TABLE VI: Impact of $\beta_e$ hyperparameter choice on MTR-RL and MTR-RL+PointNet classification accuracies, on the set of realistic objects, with 40mm position uncertainty and applied random rescaling . ($\beta_c$=1, and $\beta_p$=0)

| | Sampled points | | | |
|---|---|---|---|---|
| | 3 | 6 | 9 | 12 |
| MTR-RL + PointNet | | | | |
| $\beta_e$=0 (classif only) | 56.77 | 74.39 | 74.05 | 75.48 |
| $\beta_e$=0.5 | **56.34** | **79.11** | **79.8** | **79.8** |
| $\beta_e$=1 | 56.45 | 65.83 | 65.64 | 66.31 |
| MTR-RL | | | | |
| $\beta_e$=0 (classif only) | **48.62** | **72.133** | **73.06** | **73.42** |
| $\beta_e$=0.5 | 46.0 | 71.66 | 72.52 | 72.52 |
| $\beta_e$=1 | 46.8 | 51.38 | 52.16 | 52.04 |

### *F. Pose estimator performances and contribution to the overall framework*

TABLE VII: Contribution of the pose estimation reward on the classification performances of realistic objects, with 40mm range of position offset and applied random rescaling . ($\beta_c$=1, and $\beta_e$=0.5)

| | Sampled points | | | |
|---|---|---|---|---|
| | 3 | 6 | 9 | 12 |
| MTR-RL | | | | |
| Pos. est. ($\beta_p$=1) | 46.51 | 72.15 | 74.73 | 75.75 |
| No Pos. est. ($\beta_p$=0) | 46.0 | 71.66 | 72.52 | 72.52 |
| MTR-RL + PointNet | | | | |
| Pos. est. ($\beta_p$=1) | 56.92 | 78.37 | 80.63 | 82.05 |
| No Pos. est. ($\beta_p$=0) | 56.34 | 79.11 | 79.8 | 79.8 |

In this section, we investigate the value of adding the pose estimation term to the reinforcement learning reward. The hardest scenario is considered, that is, 40mm position uncertainty with object rescaling. As demonstrated in section IV-E, It has shown significant potential for improvement.

The Table VII compares the classification accuracy of our framework with and without pose estimation reward. The pose estimation reward gives more supervision during training and allows the framework to learn to figure out where the object is positioned, which results in an improvement on the accuracy of the framework, despite providing no additional information to the classification task.

Moreover, the pose estimation head, trained using the pose estimation reward, allows to estimate the position of the object on the workspace with 11mm, 10mm, 9mm and 10mm Root Mean Square Error, respectively after 3, 6, 9 and 12 points sampled.

## V. CONCLUSION

In this work, we proposed a framework for extremely sparse point cloud acquisition, classification and localization. The

main idea was to learn the three components together using a reinforcement learning algorithm, supervised by a 3-term reward, each term accounting for a respective component. Our framework learns an exploration strategy by itself, and doesn't need any further coding effort when considering new environments or new objects to recognize. Our solution outperforms similar learned exploration strategies adapted to our use case. Moreover, training point cloud classifiers on point clouds collected with our trained framework leads to better results than training them on randomly collected point clouds. It even leads to similar or better classification performances than training them on point clouds homogeneously sampled on the object of interest, while such hard-coded sampling strategy is not always possible, contrary to our learnt strategy. This highlighted the sensibility of point cloud classifiers to the sampling strategy when considering extremely sparse point clouds. We validated our solution both on a set of geometric shapes and on a set of realistic objects, with varying uncertainty on the object position on the workspace, and with random objects rescaling. We noticed significant classification accuracy drops when considering realistic objects with both object position uncertainty and random object rescaling. However, we further noticed that in this scenario, i) choosing the weights that balance the exploration and classification reward terms significantly affected classification performance; and ii) the addition of the pose estimation reward allowed to slightly increase the classification accuracy. In such a challenging scenario, we would encourage considering all exploration, classification, and pose estimation reward terms, but balancing them with weights to be determined through a hyperparameter search guided by the final performance of state-of-the-art point cloud classifiers used.

## VI. Future Works

The pose estimation reward was mainly used here to increase the final classification performances by helping the framework in learning to localize the objects. Few efforts have actually been spent on the actual pose estimation architecture.

Additionally, the investigation of sim-to-real transfer on our in-house robotic setup is a crucial area for future work. Specifically, we plan to examine how to incorporate the estimated measurement uncertainty of the real system into the simulation and study the impact of this additional noise during training on the performance in the real setup.

Evaluating the system in real life presents its own challenges, as acquiring a single evaluation example can take several minutes due to the time required for object positioning and exploration. Automating the evaluation involves a future plan with a two-robot setup: one for object positioning and the other for tactile exploration. Moreover, while our ultimate tactile exploration use case will involve a rigid stick-based contact sensor, we also plan to integrate a laser-based sensor into our hardware setup to prevent damage to the evaluation objects during repeated assessments.

## References


[1] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington, "3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception," IEEE Signal Processing Magazine, vol. 38, no. 1, pp. 68–86, 2021.

[2] Xuan Wang, Haiyang Lyu, Tianyi Mao, Weiji He, and Qian Chen, "Point cloud segmentation from iphone-based lidar sensors using the tensor feature," Applied Sciences, vol. 12, no. 4, pp. 1817, 2022.

[3] Alessandra Spreafico, Filiberto Chiabrando, L Teppati Losè, and F Giulio Tonolo, "The ipad pro built-in lidar sensor: 3d rapid mapping tests and quality assessment," The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 43, pp. 63–69, 2021.

[4] Vladimir Kuts, Tauno Otto, Toivo Tähemaa, Khuldoon Bukhari, and Tengiz Pataraia, "Adaptive industrial robots using machine vision," in ASME International Mechanical Engineering Congress and Exposition. American Society of Mechanical Engineers, 2018, vol. 52019.

[5] Roi Otero, Susana Lagüela, Iván Garrido, and Pedro Arias, "Mobile indoor mapping technologies: A review," Automation in Construction, vol. 120, pp. 103399, 2020.

[6] Olga Lucia Lopera Tellez and Bart Scheers, "Ground-penetrating radar for close-in mine detection," Mine Action-The Research Experience of the Royal Military Academy of Belgium, p. 82, 2017.

[7] David J Daniels, "A review of gpr for landmine detection," Sensing and imaging, vol. 7, no. 3, pp. 90, 2006.

[8] Shan Luo, Joao Bimbo, Ravinder Dahiya, and Hongbin Liu, "Robotic tactile perception of object properties: A review," Mechatronics, vol. 48, pp. 54–67, 2017.

[9] Antonio Bicchi, J Kenneth Salisbury, and David L Brock, "Contact sensing from force measurements," The International Journal of Robotics Research, vol. 12, no. 3, pp. 249–262, 1993.

[10] Kevin Riou, Kevin Subrin, and Patrick Le Callet, "Reinforcement learning based point-cloud acquisition and recognition using exploration-classification reward combination," in IEEE International Conference on Multimedia and Expo 2022, 2022.

[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] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun, "Deep learning for 3d point clouds: A survey," IEEE transactions on pattern analysis and machine intelligence, 2020.

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon, "Dynamic graph cnn for learning on point clouds," ACM Transactions on Graphics (tog), vol. 38, no. 5, pp. 1–12, 2019.

[14] Chaofan Chen, Shengsheng Qian, Quan Fang, and Changsheng Xu, "Hapgn: Hierarchical attentive pooling graph network for point cloud segmentation," IEEE Transactions on Multimedia, vol. 23, pp. 2335–2346, 2021.

[15] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo, "Rgcnn: Regularized graph cnn for point cloud segmentation," in Proceedings of the 26th ACM international conference on Multimedia, 2018, pp. 746–754.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5560–5568.

[22] 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.

[23] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.
[24] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361.
[25] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 567–576.
[26] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1912–1920.
[27] F. Hagelskjær and A. G. Buch, "Pointvotenet: Accurate object detection and 6 dof pose estimation in point clouds," in 2020 IEEE International Conference on Image Processing (ICIP). IEEE, 2020, pp. 2641–2645.
[28] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier, "Attentional pointnet for 3d-object detection in point clouds," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.
[29] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu, "Cia-ssd: Confident iou-aware single-stage object detector from point cloud," in Proceedings of the AAAI conference on artificial intelligence, 2021, vol. 35, pp. 3555–3562.
[30] Lucie A Huet, John W Rudnicki, and Mitra JZ Hartmann, "Tactile sensing with whiskers of various shapes: Determining the three-dimensional location of object contact based on mechanical signals at the whisker base," Soft robotics, vol. 4, no. 2, pp. 88–102, 2017.
[31] Yaroslav Tenzer, Leif P Jentoft, and Robert D Howe, "The feel of mems barometers: Inexpensive and easily customized tactile array sensors," IEEE Robotics & Automation Magazine, vol. 21, no. 3, pp. 89–95, 2014.
[32] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta, "Reskin: versatile, replaceable, lasting tactile skins," arXiv preprint arXiv:2111.00071, 2021.
[33] M. Lambeta, P. Chou, S. Tian, B. Yang, B. Maloon, V. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, et al., "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 3838–3845, 2020.
[34] Hadi Heidari, Nicoleta Wacker, and Ravinder Dahiya, "Bending induced electrical response variations in ultra-thin flexible chips and device modeling," Applied Physics Reviews, vol. 4, no. 3, 2017.
[35] Nawid Jamali, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale, "Active perception: Building objects' models using tactile exploration," in 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). IEEE, 2016, pp. 179–185.
[36] Mohsen Kaboli, Kunpeng Yao, Di Feng, and Gordon Cheng, "Tactile-based active object discrimination and target object search in an unknown workspace," Autonomous Robots, vol. 43, no. 1, pp. 123–152, 2019.
[37] Cristiana de Farias, Naresh Marturi, Rustam Stolkin, and Yasemin Bekiroglu, "Simultaneous tactile exploration and grasp refinement for unknown objects," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3349–3356, 2021.
[38] 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.
[39] 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.
[40] Xiaoyu Chen, Jiachen Hu, Chi Jin, Lihong Li, and Liwei Wang, "Understanding domain randomization for sim-to-real transfer," arXiv preprint arXiv:2110.03239, 2021.
[41] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath, "Deep reinforcement learning: A brief survey," IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26–38, 2017.
[42] 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.
[43] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.
[44] Matthew Hausknecht and Peter Stone, "On-policy vs. off-policy updates for deep reinforcement learning," in Deep Reinforcement Learning: Frontiers and Challenges, IJCAI 2016 Workshop, 2016.

**Kevin RIOU** received his Engineer degree in Electronics and Digital Technologies from the "École Polytechnique de l'Université de Nantes". He is currently conducting a phd on bi-manual tasks imitation learning. His current research interests include 3D scene understanding, imitation learning, reinforcement learning and robotics.

**Kaiwen DONG** received the B.S. degree from Nanjing Forestry University, Nanjing, China, in 2018, and the Ph.D. degree from the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China, in 2024. He is currently a Postdoctoral Researcher in the IoT Perception Mine Research Center, China University of Mining and Technology. His research interests include computer vision, human pose estimation, and robotics.

**Kevin SUBRIN** received his Ph.D. degree in science from the University of Clermont Auvergne in 2013. After a few years of robotics expertise for companies, he is an associate professor at Nantes University. His interests are in XXL robotics with applications in robotic tomography, large-scale 3D printing, and human-robot collaboration in industrial environments. He leads a team of engineers to bridge the gap between laboratory activities and companies, addressing industrial challenges in short timeframes. He is the co-author of around 30 publications and serves as a section editor for the journal Current Robotics Reports published by Springer Nature, focusing on the theme of Robotics in Manufacturing.

**Patrick LE CALLET** (Fellow, IEEE) is currently a Full Professor with the Polytech Nantes / Université de Nantes (Engineering School) in EE and CS. He is also a Senior Member of the Institut Universitaire de France (IUF), France. He is with the Steering Board of the CNRS LS2N lab (450 researchers). He is leading a multi-disciplinary team to conduct inter disciplinary research dealing with the application of human perception in media processing and cognitive computing, including all form of artificial intelligence. He is co-author of more than 350 publications and communications and co-inventor of 16 international patents in his research field, which include twin transition addressing quality of experience assessment for sustainable visual communication, quality of life measurement for better healthcare or inclusive technologies. He is or was an Associate Editor or Guest Editor for several Journals such as IEEE Signal Processing Magazine, IEEE Transactions on Image Processing, IEEE Journal of Selected Topics in Signal Processing, IEEE Transactions on Circuits and Systems for Video Technology, SPRINGER EURASIP Journal on Image and Video Processing, and SPIE JEI. He has been in IEEE IVMSP-TC since 2015 and IEEE MMSP-TC since 2015, and the Chair of EURASIP TAC (Technical Area Team) on Visual Image Processing. He is chairing activities in Standard (VQEG and IEEE-SA). He is the co-recipient of an Emmy Award in 2020 for his work on development of Perceptual metrics for video encoding optimization.