Automatic Acquisition of a Repertoire of Diverse Grasping Trajectories through Behavior Shaping and Novelty Search

Aurélien Morel, Yakumo Kunimoto, Alex Coninx, Stéphane Doncieux

Automatic Acquisition of a Repertoire of Diverse Grasping Trajectories through Behavior Shaping and Novelty Search

Aurélien Morel[1,*], Yakumo Kunimoto[2,*], Alex Coninx[3] and Stéphane Doncieux[4]

***Abstract*— Grasping a particular object may require a dedicated grasping movement that may also be specific to the robot end-effector. No generic and autonomous method does exist to generate these movements without making hypotheses on the robot or on the object. Learning methods could help to autonomously discover relevant grasping movements, but they face an important issue: grasping movements are so rare that a learning method based on exploration has little chance to ever observe an interesting movement, thus creating a bootstrap issue. We introduce an approach to generate diverse grasping movements in order to solve this problem. The movements are generated in simulation, for particular object positions. We test it on several simulated robots: Baxter, Pepper and a Kuka Iiwa arm. Although we show that generated movements actually work on a real Baxter robot, the aim is to use this method to create a large dataset to bootstrap deep learning methods.**

I. Introduction

Grasping is a mandatory step for many daily object manipulation tasks. We do it without even thinking about it, but despite this apparent simplicity, it is a challenging movement that has been studied for decades and is still not completely solved for robotic agents, in particular when the 3D model of the target objects are not known [1]. If robots are to be deployed in our every-day environment, for instance as home assistants, they will have to deal with very diverse situations and will thus need a strong adaptivity. In this context, grasping synthesis algorithms that are robust and efficient on a large set of objects having diverse shape and weight will be required. Learning methods are expected to help improve the generalization ability of grasping controllers [2], [3], [4], but learning to grasp requires to face a major issue: grasping is a hard exploration problem, where only a small subset of a large policy space is relevant. It creates a challenge for exploration-based learning algorithms, as without strong constraints or prior knowledge on the task or on the policy space to explore, it is unlikely that a purely random exploration could discover relevant grasping motions. A learning algorithm may thus fail to bootstrap and find solutions to improve on, may it be through gradient descent or through a gradient-free trial and error process.

(a) Kuka **(b)** Baxter **(c)** Pepper

Fig. 1: Examples of grasping behaviors found for each robot.

A possible solution is to decompose the task and start by estimating object pose before determining where to grasp it [4], but these approaches require to know object models. Another approach consists in exploiting available knowledge about the end-effector. The features of a parallel-jaw gripper, for instance, can be used to limit the number of grasping candidates [5], [6], [7]. Unfortunately, this strategy does not easily transfer to other kinds of end-effectors. Another strategy consists in constraining the grasping movements to an easier top-down grasping motion [8], but, by construction, it limits the range of possibilities. Human demonstrations can also be used to guide the learning process [9], [10], [11], [12], but generating large enough datasets is costly and would need to be repeated for every new robot.

In this work, we introduce a method to autonomously build grasping datasets. The method can be applied to any kind of robot arm and end-effectors with negligible adaptation. It has been tested in simulation on grippers (Kuka Iiwa and Baxter robots) as well as on a multi-fingered hand (Pepper robot). The transfer to a real robot has been studied on the Baxter. The proposed approach relies on a Quality Diversity algorithm [13], [14] built on top of Novelty Search [15]. Previous works with these algorithms have generated repertoires of legged-robots movements [16] or ball throwing and joystick manipulation movements on a Baxter [17]. Novelty Search has been shown to have an efficient exploration ability as it samples uniformly in a given behavior space [18], [19]. To deal with the sparsity of successful grasping movements, we introduce an improvement of its exploration skill through the simultaneous management of multiple behavior spaces. It drastically improves the number of grasping movements discovered. The approach generates sets of open-loop grasping movements for a given object at a given location (Fig. 1). It cannot be used to directly grasp unknown objects at any location, but can generate large-scale

*Both first authors contributed equally to this work.

[1]Aurelien Morel, Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France and Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Suisse aurelien.morel.arthur@gmail.com

[2]Yakumo Kunimoto, Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France yakunimoto@yahoo.fr

[3]Alex Coninx, Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France coninx@isir.upmc.fr

[4]Stéphane Doncieux, Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France doncieux@isir.upmc.fr

grasping datasets for deep learning approaches. Although the grasping movements are generated and tested in simulation, we show that the size and diversity of the dataset result in a large number of policies that transfer to the real world.

II. RELATED WORK

Large scale data collection can rely on human demonstrators observation [20], [21], but these methods are hard to scale since they require large human resources, are biased by human semantic priors and are specific to one type of robotic arm. Automating the data collection is challenging given the sparsity of grasping movements, but it has been achieved in restricted planar grasp conditions [22], [8]. These restrictions are sufficient to get a successful grasp rate of 10% to 30%, which allows to collect a sufficient dataset to train a neural network achieving grasp control from real video images [8]. Other approaches focus on parallel-plate grippers to define an autonomous grasping movement generator [7]. The approach introduced here does not constrain grasping movements and can be used directly on any kind of robot.

Discovering diverse solutions is an important challenge in a variety of Machine Learning fields. Novelty-based evolutionary methods [15], [13], [23], as well as related goal exploration process methods [24], were designed to tackle this problem and illuminate search spaces. Novelty search, one of these approaches [15], was actually shown to tend towards a uniform sampling in a user-defined behavior space [18], [19], a property that is hard to get given the complexity of the mapping the policy parameter space and the behavior space. These methods have been used to generate repertoires of primitive actions for robot locomotion [23], [16] or simple manipulations with a robot arm [17]. The generated repertoires have already been used to bootstrap deep learning approaches [25], [26]. We propose to extend novelty-based repertoire generation methods with exploration of multiple behavior spaces such as in [27], [28] but with an additional focus on setups with very sparse interactions. We then apply it to grasping movements generations.

III. METHOD

Fitness shaping is a convenient mean to turn a sparse reward into a dense reward in order to facilitate exploration [29]. Novelty-based algorithms algorithms are not driven by reward and therefore do not suffer from the sparse reward problem, but if the behaviors they look for are very sparse in the space of all possible behaviors, as is the case for grasping, a similar issue arises as the search will have trouble bootstrapping and discovering relevant solutions. In the previously studied robotics tasks such as locomotion, navigation and ball throwing, all policies (even randomly generated ones) yield a valid behavior: the robot reaches some final position, or the ball reaches some location. The first discovered behaviors are usually very simple (such as remaining very close to the start position or just dropping the ball), but they allow the algorithm to bootstrap and then iteratively discover more novel and higher quality policies. By contrast, trying to bootstrap grasping policies with random generation in a large policy space with little or no prior knowledge will mostly produce behaviors that do not engage the target object at all.

To mitigate this issue, we introduce an approach for *behavior shaping* that has a goal similar to fitness shaping, but in behavior spaces. The idea is to introduce new behavior spaces whose exploration aims at facilitating the generation of "target" behaviors, in a curriculum-like way. Building upon the ability of Novelty Search to continuously generate novel solutions [19], Novelty Search with Multiple Behavior Spaces (NSMBS) extends it to take into account multiple behavior spaces and thus propose a framework in which behavior shaping can be implemented.

A. Definitions and notations

In this paper, we consider policies π_θ parameterized by a vector $\theta \in \Theta \subset \mathbb{R}^{n_g}$, with n_g the policy dimension. The policies, as well as the transition model of the robot and environment, are assumed deterministic. Novelty-based methods rely on the policies' *behavior characterization*, which results from mapping the trajectory of the robot when applying the policy from an initial state s_0 into a smaller dimension vector in a carefully selected *behavior space* [18]. In the following and as usual in novelty-based algorithms, we assume that the initial state is fixed, and the behavior therefore only depends on the policy parameter θ.

NSMBS proposes a framework to explore in multiple behavior spaces, here called behavior components. A policy is thus be described by n_b behavior components. While in classic novelty-based algorithms, each policy is always associated to a behavior characterization, NSMBS relaxes this assumption and some policies may have undefined behavior components. It allows us to characterize grasping behaviors, for instance, and give an undefined grasping behavior component to policies that do not succeed in grasping the object. To this end, an eligibility criterion is associated to each behavior component. This gives rise to the following notations:

- $\mathcal{B}_i \subset \mathbb{R}^{n_{B_i}}$ is the i-th behavior component space;
- $\xi_i(\theta)$ is the eligibility criterion associated to $\mathcal{B}_i$. If $\xi_i(\theta) =$ True, the corresponding behavior is defined.
- $b_i(\theta) \in \mathcal{B}_i \cup \{\varnothing\}$ is the i-th component of the robot behavior when it follows policy π_θ. $b_i(\theta) = \varnothing$ i.f.f. $\xi_i(\theta) =$ False;
- n_b is the number of behavior components.

Although some recent work try to automatically learn the behavior spaces [30], [31], here the behavior components are supposed to be known and given by the experimenter.

B. NSMBS

NSMBS (Algorithm 1) derives from Novelty Search [15], [18]. As for other evolutionary algorithms, an initial population of solutions is randomly generated, evaluated, and a parent set is selected. The parents are then copied and modified by mutation and crossover operators to get new solutions, that are then evaluated and selected to constitute

the next generation of parents. In Novelty Search, an archive of explored behaviors is also maintained, and the selection process relies on maximizing a novelty objective, which is the average distance to the K-nearest neighbors in the behavior space among the archive and current population.

In NSMBS, the outcome of a policy π_θ is described by a list of behavior components: $(b_1(\theta), b_2(\theta), \ldots, b_{n_b}(\theta))$. The selection process in charge of finding the individuals of the next generation relies on these elements (Algorithm 2). Policies are selected one by one from a set of individuals $\mathcal{S}$ initialized with the current population and offspring. The individual selection starts by randomly choosing a behavior component among the ones that are defined at least for one policy in $\mathcal{S}$. The most novel individual in the selected component is then selected; it is removed from $\mathcal{S}$ and the process starts again until the new population is filled.

Algorithm 1: NSMBS

Input: population size μ, number of generations G, number of offsprings λ, evaluation function $eval()$, number of neighbours for novelty computation k
Result: archive of individuals
pop $\leftarrow$ $generateRandomPopulation(\mu)$;
archive $\leftarrow \varnothing$;
for a *in population* **do**
  $a.bd \leftarrow eval(a)$;
end
gen $= 0$;
while $gen < G$ **do**
  parents $\leftarrow$ $selectParents(pop, \lambda)$;
  offspring $\leftarrow$ $operate$(parents) ;
  for a *in offspring* **do**
    $a.bd \leftarrow eval(a)$;
  end
  refSet $\leftarrow$ pop $\cup$ offspring $\cup$ archive ;
  for a *in pop* $\cup$ *offspring* **do**
    $a.nov \leftarrow getNov(a,$ refSet$, k)$;
  end
  archive.$add(randomSample($offspring$))$;
  pop $\leftarrow$ $multiBCSel($pop $\cup$ offspring$, \mu)$;
  gen $=$ gen $+ 1$;
end

Algorithm 2: multiBCSel

Input: set of individuals $\mathcal{S}$, number of individuals to select μ
Result: set of individuals $\mathcal{P}$
$\mathcal{P} \leftarrow \varnothing$;
while $size(\mathcal{P}) < \mu$ **do**
  $i \leftarrow randomSelComponent(\mathcal{S})$;
  $\mathcal{T} \leftarrow \{\theta \in \mathcal{S} \mid \xi_i(\theta) =$ True$\}$;
  chosen $\leftarrow \arg\max_{\theta \in \mathcal{T}} nov_i(\theta)$;
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{$chosen$\}$; $\mathcal{S} \leftarrow \mathcal{S} - \{$chosen$\}$;
end

- *generateRandomPopulation* generates a population of size μ: $\{\theta_0, ...\theta_{\mu-1}\}$ by uniform sampling;
- *eval* evaluates an individual's policy θ and outputs its behavior components $bd = (b_1(\theta), b_2(\theta), \ldots, b_{n_b}(\theta))$, with $b_i(\theta) = \varnothing$ i.f.f. $\xi_i(\theta) =$ False
- *selectParents* is a random selection process of λ individuals in pop;
- *operate* copies the parents and applies gaussian bounded mutation and crossover operators to generate offsprings;
- *getNov* computes a list of per component novelties $\text{nov} = (\text{nov}_1, \text{nov}_2, \ldots, \text{nov}_{n_b}(\theta))$ for an individual. $\text{nov}_i = \sum_{k=0}^{K-1} (dist\,(b_i(\theta), b_i(\theta_k)))$ where $b_i(\theta_k)$ are the behavior components of the K nearest neighbors of b_i in refSet ($\varnothing$ values are ignored). If $b_i(\theta) = \varnothing$, nov_i is undefined.
- *multiBCSel* is the main selection process, described in Fig. 2 and Algorithm 2.
- *randomSelComponent* selects a behavior component by uniform selection among components for which at least one individual is eligible.

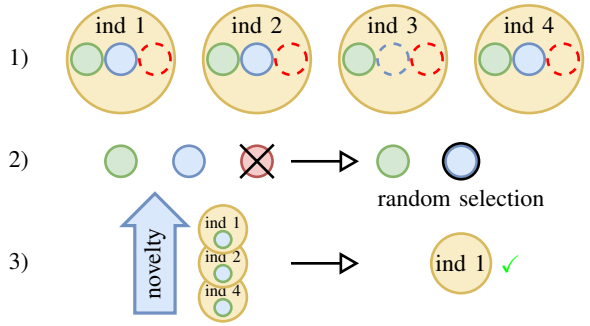


Fig. 2: Description of the *multiBCSel()* method for 4 individuals and 3 behavior components. The third (red) component cannot be selected because there is no eligible individual. Among the two remaining components, the second (blue) one is randomly selected, and the eligible individuals (inds 1, 2 and 4) are sorted according to nov_2. The most novel individual, ind 1, is selected.

IV. Experimental Setup

A. Simulated environments description

In order to demonstrate the algorithm's ability to generate rich, diverse grasping movements for multiple robots, we rely on simulated environments using the Gym framework [32]. A robot is placed in front of a table, where an object is spawned at a fixed position. Three robots are used: A Rethink Robotics Baxter using only the left arm, with 7 degrees of freedom and a parallel gripper; A Kuka Iiwa arm with 7 degrees of freedom and a two-fingers clamp gripper, both simulated using pyBullet [33]; and a SoftBank Robotics Pepper, using only the left arm, with 5 degrees of freedom and a multi-fingered hand, simulated using Qibullet [34]. The episode's length T is adapted to each robot and simulator's dynamics, all other experimental hyperparameters being the same between the three robots. Five target objects are used: a simple $5\,\text{cm}$ cube, a simple $5\,\text{cm}$ ball, a miniature plastic bowling pin, a mug with a handle, and a gamepad.

The policies are defined by three waypoints in the joint space, giving the robot pose at $T/3$, $2T/3$ and T. Smooth motions are interpolated using third order polynomials. An

extra parameter t_{grasp} defines the time when the gripper closes. The policy dimension n_g is therefore $3 \times n_{dof} + 1$.

Besides the length of the episode T, we define t_{touch} the time when the gripper first touches the object (undefined if the object is not touched), obj_t^{pos} and $grip_t^{pos}$ the position of resp. the object and the gripper at time t in the Cartesian space ($\mathbb{R}^3$), and $grip_t^{or} \in \mathbb{R}^4$ the orientation of the gripper at t, defined as a quaternion.

B. Behavior Descriptors

We consider the following four behavior descriptors:$b_1(\theta) = obj_T^{pos}$, $b_2(\theta) = grip_{T/2}^{or}$, $b_3(\theta) = grip_{t_{touch}}^{pos}$ and $b_4(\theta) = grip_{t_{touch}}^{or}$. Each b_i pushes the exploration in one direction and generates a useful incentive: b_1 pushes for trajectories that move the object around, b_2 promotes diverse movements during exploration, b_3 and b_4 explicitly push towards diverse grasping poses.

b_1 is always eligible. b_2 is only eligible if the gripper touched the object (we only search diverse trajectories that engage the object). b_3 and b_4 are only eligible if the object was successfully grasped. A successful grasp is defined as a trajectory where the gripper touches the object shortly after it is closed while the object is on the table, and that ends with the object in a stable position in the air, with minor conditions on penetrations to avoid unrealistic grasps.

C. Measures

To monitor the progress of the exploration and the resulting diversity, we use the following metrics:

- The number of successful grasping solutions found, and their *coverage* of the behavior space $\mathcal{B}_4$ [19]. This gives an estimation of the raw success of the repertoire generation process.
- The *sample efficiency*, defined as the proportion of all the evaluations that resulted in a successful grasp. This measures the ability to generate a rich repertoire while minimizing the computational cost.

In order to compare our method to the state of the art, we introduce two other measurements:

- The *first success generation*, which is the number of iterations after which the algorithm first discovers a successful grasp. This evaluates the ability of an algorithm to quickly bootstrap in this sparse, difficult task.
- The *successful run rate*, which measures the proportion of runs that generated at least a single grasping movement. This measures the reliability of an algorithm.

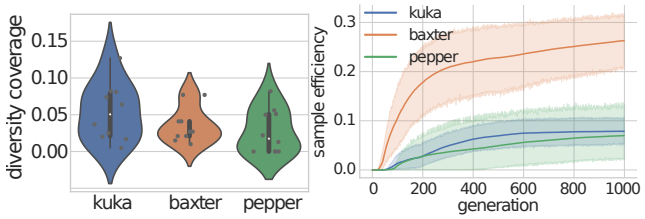
V. RESULTS

A. Grasping synthesis on multiple robots

The NSMBS algorithm is first run on all the three robots, with the mug object, for 1000 generations, a population size $\mu = 100$ individuals, an offspring size $\lambda = 50$ individuals, and a value of $K = 15$ for the KNN algorithm. The search is repeated 10 times for each condition.

The generated repertoires are very different between robots, with sizes of 13148 ± 4574 successful grasps for the Baxter, 3490 ± 4706 for the Pepper, and 3929 ± 2262 for the Kuka. This reflects the variable difficulty of the grasping task with different robots, the high variance for the Pepper and Kuka robots being due to the presence of failed runs where no grasping was discovered. Despite this, the search finds a variety of grasping behaviors for all robots, some of which are shown in Fig. 1.

Fig. 3 shows the final diversity coverage and the evolution of the sample efficiency for the three robots. We can see that the algorithm quickly manages to bootstrap and discover grasping motions, and then leverages those existing motions to improve diversity, as shown by the constantly growing sample efficiency during the process.



(a) Diversity coverage of NSMBS for all robots.

(b) Evolution of sample efficiency during the search.

Fig. 3: Coverage and sample efficiency of the search for grasping behavior with NSMBS for the three robots and the mug object.

Furthermore, we can sort the ways the mug object can be grasped in four different categories, which allows us to define four grasping styles, not all of them available to all robots due to their design:

- Grasping the handle (`handle` style): this grasping style is possible on all three robots;
- Grasping the mug from the top, with part of the gripper inside the mug and the other outside (`in-out` style): only the Kuka and Baxter robots are capable of this grasping style, the Pepper robot's weaker hand being unable to hold the mug this way;
- Grasping the mug from the sides, with the gripper grasping the outside of the mug (`out` style): only the Kuka robot is capable of this grasping style, thanks to its much larger gripper;
- Grasping the mug from the inside, by inserting the entire gripper (`in` style): only the Baxter robot with its thin parallel gripper is capable of this grasping style.

We categorize the discovered grasping motions on all three robots. The results (Fig. 4) show that NSMBS is general enough to fully make use of the capabilities of each robot, discovering all the possible styles for all robots.

B. Comparative analysis

In order to highlight the benefits of NSMBS, we compare the performance with the following variants and baselines:

- Our full NSMBS algorithm (NSMBS);
- The same algorithm without the object touching BD (NSMBS no BD2) and without the grasping position BD ((NSMBS no BD3);

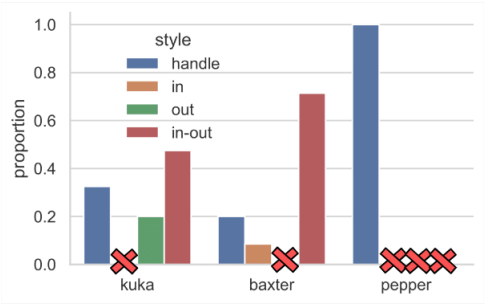


Fig. 4: Mug grasping styles proportions (10 runs, 1000 generations). Red cross means grasp style not generated for this robot.

- A classic Novelty Search (NS) algorithm [15] using a single behavior descriptor, which is built by concatenating the four b_i from NSMBS. If one of the components is not eligible, its values are set to 0.
- A Map-Elites algorithm [23], using the same concatenated behavior descriptor as NS with 1000 CVT cells [35]. Map-Elites uses a quality measure; it is not important here and set to a simple energetic criterion;
- A random search baseline where a similar number of individuals are generated by random uniform sampling in policy parameter space.

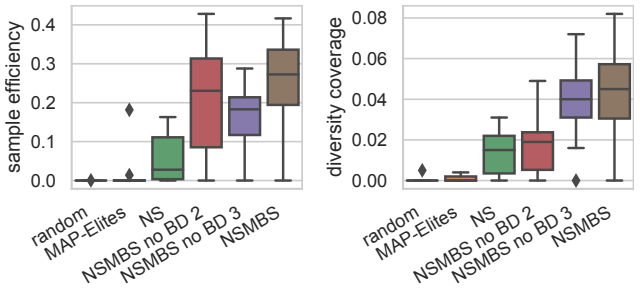
The algorithms are used on the Baxter environment with the mug object, with the same hyperparameters as above except the process is run for 2000 generations. Each condition is repeated 20 times. The main results are shown in Fig. 5. The NSMBS variants overperforms the baselines, yielding higher sample efficiency and diversity coverage. NS discovers some grasping motions, but is generally much less sample-efficient, resulting in a lower coverage. Map-elites is generally unsuccessful ; this may be due to the absence of a population combined with a random selection and a very large composite behavior space, which fails to focus the evolutionary budget on promising individuals [36].

Fig. 6 gives further insights into the benefits of our behavior shaping technique. Although most NS and NSMBS runs are eventually successful at generating at least some grasping behaviors (Fig. 6a), the full NSMBS algorithm with all the behavior components does so earlier in the evolutionary process (Fig. 6b). It also quickly achieves and maintains higher sample efficiency (Fig. 6c), with 25% of evaluated individuals (in average) being successful grasps after only 400 generations.

C. Transfer on real-world robot

The grasping motion repertoires are only relevant if at least some of the generated policies transfer to the real world. We therefore generate repertoires for five objects with the Baxter robot and evaluate them in real world experiments.

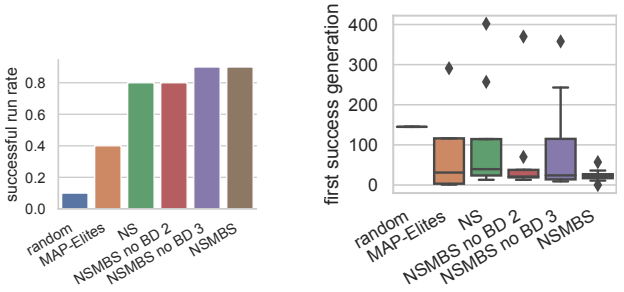
NSMBS is first run 10 times for 1000 generations in the simulated Baxter environment for each of the five objects. Repertoires are successfully generated for all objects (Fig. 8). Results for the mug were previously discussed in section V-B; sample efficiency is lower for other objects, but NSMBS still reliably generates large repertoires of successful grasps.



(a) Overall sample efficiency of each algorithm.

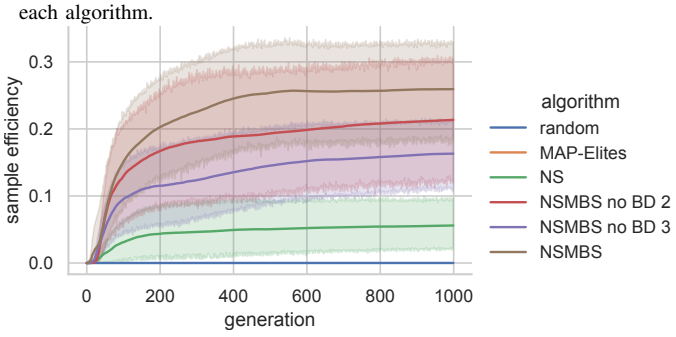
(b) Final diversity coverage of each algorithm.

Fig. 5: Sample efficiency and diversity coverage of the search for grasping behavior on the Baxter robot after 2000 generations.



(a) Proportion of successful runs (i.e.runs that discover at least one grasp within 2000 generations) for each algorithm.

(b) Generation at which the first grasp is found, for each algorithm. Unsuccessful runs are ignored.

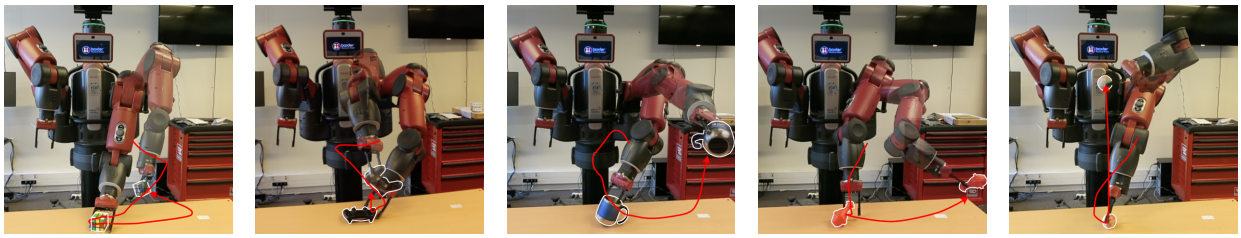


(c) Evolution of sample efficiency during the runs.

Fig. 6: Proportion of successful runs, bootstrap time for successful runs, and evolution of the sample efficiency for each algorithm (map-elites hidden behind random).

All 10 runs succeed in discovering grasping motions for the pin, 9 succeed for the mug, the sphere and the gamepad, and 7 succeed for the ball. For each successful run, 10 individuals are randomly sampled from the final repertoire, and evaluated on the robot, for a total of 70 to 100 real world evaluations depending on the object. Some successful grasps for each object are shown in Fig. 7 and in the video in annex.

The transfer success rates are reported in Fig. 9a. The highest transfer rate is achieved with the mug, where $30\,\%$ of policies succeeded, followed by the cube, the sphere, and finally the gamepad and the pin at $8\,\%$. It is noteworthy that this success rate is directly correlated with the sample efficiency of the process in simulation (Fig. 8). The lower sample efficiency for the gamepad and the pin can be explained by their characteristics : the gamepad is quite heavy ($0.21\,\mathrm{kg}$) compared to other objects, and it has a complex



(a) Cube **(b)** Gamepad **(c)** Mug **(d)** Pin **(e)** Sphere

Fig. 7: Grasping example on real Baxter robot for each object. The end effector trajectory, the pose at which the robot grasps the object and the final robot and object pose are shown.

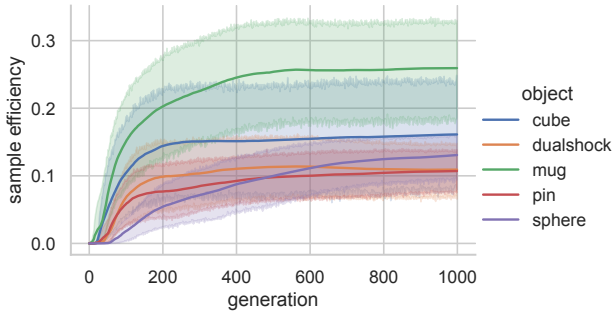
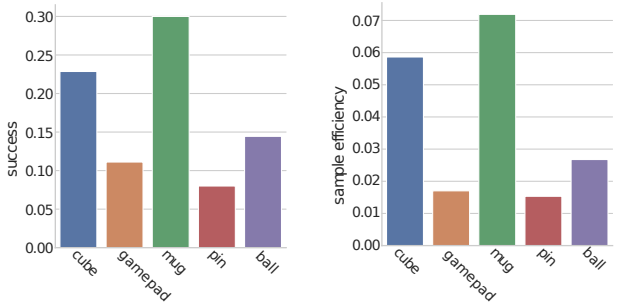


Fig. 8: Sample efficiency at each generation in simulation.



(a) Success rate of evaluated individuals in reality.

(b) Estimated sample efficiency for the full repertoire generation process in reality.

Fig. 9: Success rate and sample efficiency in reality on the robot Baxter. Repertoire generation is run 10 times per object, and 10 individuals are randomly selected from each successfully generated repertoire.

shape and moving parts that are not fully represented in the simulated environment. The pin is standing, thus it can easily fall and the dynamics then differ from the simulation.

We can use these results to estimate the sample efficiency of the full repertoire generation process, i.e. the expected proportion of all the evaluations in simulation that result in a grasp that transfers to reality, by multiplying the sample efficiency in simulation (as defined in section IV-C) by the transfer success rate. Results are shown in Fig. 9b. Values range from about $1.6\,\%$ for the pin to $7.1\,\%$ for the mug.

VI. CONCLUSION

The simulation results show that our NSMBS method is able to efficiently generate grasping behaviors for various robot arms, and that it outperforms the state of the art in both diversity coverage and sample efficiency. The comparison to ablated versions of the method with only some of the behavior components highlights the importance of choosing adequate behavior characterizations for the method to be able to bootstrap and learn, and the comparison to classic diversity methods showcases the ability of NSMBS to take advantage of those multiple components for behavior shaping.

Real world transfer rates range from $8\,\%$ to $30\,\%$ depending on the object. This is comparable to the technique used by Levine et al. [8], which makes strong hypotheses about the experimental setup and uses motor primitives that constrains the diversity of the possible grasps, whereas our method is applicable to a large variety of robots without any modification and promotes grasping diversity.

The real world sample efficiency ranges from $1.6\,\%$ to $7.1\,\%$, which corresponds to one out of 14 to one out of 62 simulated policies being a valid grasping policy in the real world. It may seem low, but it does not prevent the generation of a large, diverse repertoire, as simulated experiments are cheap enough that more than $100\,000$ evaluations can be performed in a few hours on a modern workstation, resulting in thousands of transferable grasps. An open question is how to filter the generated repertoire for individuals that transfer to the real world, as the transfer rate makes random sampling wasteful. Further work could address this by training a model to predict transferability from a limited real world dataset.

Our method allows for the generation of large-scale, diverse datasets for robotics, including for challenging tasks like grasping, without limitations about the applicable robots or policies. Instead, it requires defining relevant behavior spaces and the corresponding eligibility criteria. This is usually not a major issue, as it is often quite straightforward to define behavior components to promote diversity in a way that helps discover and explore the task solutions space.

The final test of NSMBS shall be the use of the generated repertoires as datasets to train closed loop policies [8], [25], and the evaluation of those policies on real robots. The large size and high diversity of the generated datasets, the good sample efficiency of the method and its applicability to various problems and robots without adaptation or parameter tweaking, make it promising in this regard.

ACKNOWLEDGMENT

This work was supported by ANR projects InDex and Learn2Grasp.

REFERENCES

[1] Billard, A., and Kragic, D., 2019. "Trends and challenges in robot manipulation". *Science,* **364**(6446).

[2] Sahbani, A., El-Khoury, S., and Bidaud, P., 2012. "An overview of 3d object grasp synthesis algorithms". *Robotics and Autonomous Systems,* **60**(3), pp. 326–336.

[3] Bohg, J., Morales, A., Asfour, T., and Kragic, D., 2013. "Data-driven grasp synthesis—a survey". *IEEE Transactions on Robotics,* **30**(2), pp. 289–309.

[4] Kleeberger, K., Bormann, R., Kraus, W., and Huber, M. F., 2020. "A survey on learning-based robotic grasping". *Current Robotics Reports*, pp. 1–11.

[5] Kraft, D., Detry, R., Pugeault, N., Başeski, E., Guerin, F., Piater, J. H., and Krüger, N., 2010. "Development of object and grasping knowledge by robot exploration". *IEEE Transactions on Autonomous Mental Development,* **2**(4), pp. 368–383.

[6] Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A., and Goldberg, K., 2017. "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". *arXiv preprint arXiv:1703.09312*.

[7] Depierre, A., Dellandréa, E., and Chen, L., 2018. "Jacquard: A large scale dataset for robotic grasp detection". In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 3511–3516.

[8] Levine, S., Pastor, P., Krizhevsky, A., and Quillen, D., 2016. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection.

[9] Lenz, I., Lee, H., and Saxena, A., 2015. "Deep learning for detecting robotic grasps". *The International Journal of Robotics Research,* **34**(4-5), pp. 705–724.

[10] Sharma, P., Mohan, L., Pinto, L., and Gupta, A., 2018. "Multiple interactions made easy (mime): Large scale demonstrations data for imitation". In Conference on robot learning, PMLR, pp. 906–915.

[11] Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., and Abbeel, P., 2018. "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation". In 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 5628–5635.

[12] Song, S., Zeng, A., Lee, J., and Funkhouser, T., 2020. "Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations". *IEEE Robotics and Automation Letters,* **5**(3), pp. 4978–4985.

[13] Pugh, J. K., Soros, L. B., and Stanley, K. O., 2016. "Quality diversity: A new frontier for evolutionary computation". *Frontiers in Robotics and AI,* **3**, p. 40.

[14] Cully, A., and Demiris, Y., 2017. "Quality and diversity optimization: A unifying modular framework". *IEEE Transactions on Evolutionary Computation,* **22**(2), pp. 245–259.

[15] Lehman, J., and Stanley, K. O., 2011. "Abandoning objectives: Evolution through the search for novelty alone". *Evolutionary Computation,* **19**(2), pp. 189–223.

[16] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B., 2015. "Robots that can adapt like animals". *Nature,* **521**(7553), May, pp. 503–507.

[17] Kim, S., Coninx, A., and Doncieux, S., 2021. "From exploration to control: learning object manipulation skills through novelty search and local adaptation". *Robotics Auton. Syst.,* **136**, p. 103710.

[18] Doncieux, S., Laflaquière, A., and Coninx, A., 2019. "Novelty search: a theoretical perspective". In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 99–106.

[19] Doncieux, S., Paolo, G., Laflaquière, A., and Coninx, A., 2020. "Novelty search makes evolvability inevitable". In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, GECCO '20, Association for Computing Machinery, pp. 85–93.

[20] Saudabayev, A., Rysbek, Z., Khassenova, R., and Varol, H. A., 2018. "Human grasping database for activities of daily living with depth, color and kinematic data streams". *Scientific Data,* **5**.

[21] Ekvall, S., and Kragic, D., 2007. "Learning and evaluation of the approach vector for automatic grasp generation and planning". In Proceedings 2007 IEEE International Conference on Robotics and Automation, pp. 4715–4720.

[22] Pinto, L., and Gupta, A., 2016. "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours". *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3406–3413.

[23] Mouret, J.-B., and Clune, J., 2015. Illuminating search spaces by mapping elites.

[24] Forestier, S., Mollard, Y., and Oudeyer, P.-Y., 2017. "Intrinsically motivated goal exploration processes with automatic curriculum learning". *ArXiv,* **abs/1708.02190**.

[25] Jegorova, M., Doncieux, S., and Hospedales, T. M., 2020. "Behavioral repertoire via generative adversarial policy networks". *IEEE Transactions on Cognitive and Developmental Systems*.

[26] Keller, L., Tanneberg, D., Stark, S., and Peters, J., 2020. "Model-Based Quality-Diversity Search for Efficient Robot Learning". *arXiv:2008.04589 [cs]*, Aug. arXiv: 2008.04589.

[27] Doncieux, S., and Coninx, A., 2018. "Open-ended evolution with multi-containers qd". In Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '18, Association for Computing Machinery, p. 107–108.

[28] Cazenille, L., 2021. *Ensemble Feature Extraction for Multi-Container Quality-Diversity Algorithms*. Association for Computing Machinery, New York, NY, USA, p. 75–83.

[29] Ng, A. Y., Harada, D., and Russell, S., 1999. "Policy invariance under reward transformations: Theory and application to reward shaping". In Icml, Vol. 99, pp. 278–287.

[30] Paolo, G., Laflaquiere, A., Coninx, A., and Doncieux, S., 2020. "Unsupervised learning and exploration of reachable outcome space". In 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 2379–2385.

[31] Cully, A., 2019. "Autonomous skill discovery with quality-diversity and unsupervised descriptors". In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 81–89.

[32] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., 2016. Openai gym.

[33] Coumans, E., and Bai, Y., 2016–2019. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`.

[34] Busy, M., and Caniot, M., 2019. "qibullet, a bullet-based simulator for the pepper and nao robots". *arXiv preprint arXiv:1909.00779*.

[35] Vassiliades, V., Chatzilygeroudis, K., and Mouret, J.-B., 2017. Using centroidal voronoi tessellations to scale up the multi-dimensional archive of phenotypic elites algorithm.

[36] Coninx, A., and Doncieux, S., 2021. "Younger is better: A simple and efficient selection strategy for map-elites". In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, p. 87–88.