2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561315

ReLMoGen: Integrating Motion Generation in Reinforcement Learning for Mobile Manipulation

Fei Xia*[1], Chengshu Li*[1], Roberto Martín-Martín[1], Or Litany[2], Alexander Toshev[3], Silvio Savarese[1]

***Abstract*— Many Reinforcement Learning (RL) approaches use joint control signals (positions, velocities, torques) as action space for continuous control tasks. We propose to lift the action space to a higher level in the form of subgoals for a motion generator (a combination of motion planner and trajectory executor). We argue that, by lifting the action space and by leveraging sampling-based motion planners, we can efficiently use RL to solve complex, long-horizon tasks that could not be solved with existing RL methods in the original action space. We propose ReLMoGen – a framework that combines a learned policy to predict subgoals and a motion generator to plan and execute the motion needed to reach these subgoals. To validate our method, we apply ReLMoGen to two types of tasks: 1) Interactive Navigation tasks, navigation problems where interactions with the environment are required to reach the destination, and 2) Mobile Manipulation tasks, manipulation tasks that require moving the robot base. These problems are challenging because they are usually long-horizon, hard to explore during training, and comprise alternating phases of navigation and interaction. Our method is benchmarked on a diverse set of seven robotics tasks in photo-realistic simulation environments. In all settings, ReLMoGen outperforms state-of-the-art RL and Hierarchical RL baselines. ReLMoGen also shows outstanding transferability between different motion generators at test time, indicating a great potential to transfer to real robots. For more information, please visit project website: http://svl.stanford.edu/projects/relmogen.**

I. Introduction

Many tasks in mobile manipulation are defined by a sequence of navigation and manipulation subgoals. Navigation moves the robot's base to a configuration where arm interaction can succeed. For example, when trying to access a closed room, the robot needs to navigate to the front of the door to push it with the arm or, alternatively, to press a button next to the door that activates its automatic opening mechanism. Such a sequence of subgoals is well parameterized as spatial points of interest in the environment to reach with the robot's base or end-effector. The path towards these points is mostly irrelevant as long as it is feasible for the robot's kinematics and does not incur collisions.

Collision-free feasible trajectories to points of interest can be efficiently computed and executed by a motion generator (MG) composed of a motion planner (MP) and a trajectory controller [1, 2]. MGs specialize in moving the robot's base or end-effector to a given short-range point, usually within the field of view so that they can use an accurate model of the environment. However, due to the sample complexity of large Euclidean space and the lack of accurate models of the entire environment, MGs cannot solve the problem of long-range planning to a point beyond sight. Moreover, MGs excel at answering "how" to move to a point, but not "where" to move to achieve the task based on current observations, where deep RL [3, 4] has shown strong results.

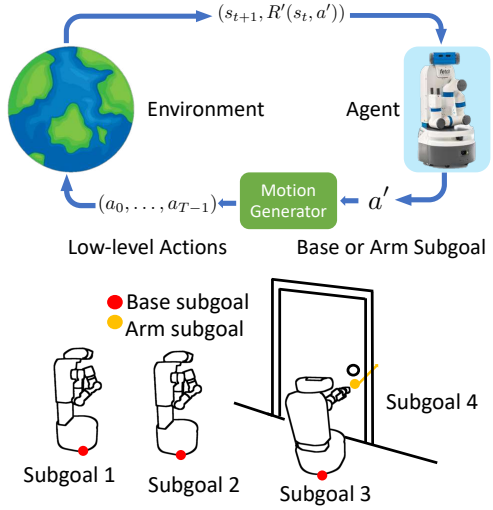


Fig. 1: (top) We propose to integrate motion generation into a reinforcement learning loop to lift the action space from low-level robot actions a to subgoals for the motion generator a' (bottom) The mobile manipulation tasks we can solve with ReLMoGen are composed by a sequence of base and arm subgoals (e.g. pushing open a door for Interactive Navigation).

RL has been successfully applied to solve visuo-motor tasks dealing with continuous control based on high dimensional observations [5, 6, 7, 8, 9, 10, 11]. However, this methodology falls short for mobile manipulation tasks, which involve long sequences of precise low-level actions to reach the aforementioned spatial points of interest. Often, the steps in free space do not return any reward, rendering mobile manipulation hard exploration problems [12, 13]. While the exploration challenge may be mitigated by a hierarchical structure [14, 15, 16, 17], the RL agent still dedicates a large portion of its training steps to learning to move towards spatial points of interest without collisions from scratch.

In this work, we present ReLMoGen (Reinforcement Learning + Motion Generation), a novel approach that combines the strengths of RL and MG to overcome their individual limitations in mobile manipulation domains. Specifically, we propose to employ RL to obtain policies that map observations to subgoals that indicate desired base or arm motion. These subgoals are then passed to a MG that solves for precise, collision-free robot control between consecutive subgoals. We refer to the resulting policy as Subgoal Generation Policy (SGP).

Considering the mobile manipulation task as a Markov Decision Processes (MDPs), ReLMoGen can be thought of as creating a *lifted* MDP, where the action space is re-defined

* Equal contribution. [1] Stanford University. [2] Nvidia. [3] Robotics at Google.

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.

to the space of MG subgoals. This presents a temporal abstraction where the policy learns to produce a shorter sequence of "subgoal actions", which makes exploration easier for the policy, as demonstrated in the experimental analysis. From a control perspective, ReLMoGen is a hierarchical controller, whose high-level module is a learned controller while the low-level module is a classical one.

The contributions of this paper are as follows. First, we demonstrate how to marry learning-based methods with classical methods to leverage their advantages. We thoroughly study the interplay between two RL algorithms, Deep Q Learning [18] and Soft-Actor Critic [19], and two established motion planners, Rapidly expanding Random Trees [20] and Probabilistic Random Maps [21]. Further, we demonstrate that ReLMoGen consistently achieves higher performance across a wide variety of long-horizon robotics tasks. We study the approach in the context of navigation, stationary manipulation and mobile manipulation. ReLMoGen is shown to explore more efficiently, converge faster and output highly interpretable subgoal actions. Finally, we show that the learned policies can be applied with different motion planners, even with those not used during training. This demonstrates the robustness and practicality of our approach and great potential in real-world deployment.

II. Related Work

ReLMoGen relates to previous efforts to combine robot learning and motion generation. At a conceptual level, it can also be thought of as a hierarchical RL approach with a stationary low-level policy. Therefore, we will relate to previous work in these areas.

Combining Learning and Motion Generation: Recently, researchers have attempted to overcome limitations of *classical* sampling- or optimization-based motion generators by combining them with machine learning techniques. There are two well-known limitations of classical MGs: 1) they depend on an accurate environment model, and 2) their computational complexity grows exponentially with the search space dimension. Researchers have proposed learning-based solutions that map partial observations to waypoints [22, 23, 24, 25] or trajectories [26], thus bypassing the trajectory searching problem. They rely on expert MG supervision to learn via imitation. In contrast, we do not attempt to improve MG but rather integrate it into a RL loop as is. The opposite has also been attempted: improving the exploration of RL agents using experiences from a MG [27, 28, 29, 30]. We only use MGs to map from our lifted action space to low-level motor control signals during training.

Closely related to our approach are works that integrate a planner or a motion generator as-it-is into RL procedure. For example, Jiang et al. [31] integrates a task and motion planner (TAMP) with RL: the TAMP planner provides solutions for room-to-room navigation that RL refines. Dragan et al. [32] learns to set goals for an optimization-based motion generator based on predefined features that describe the task. In a concurrent work [33], the authors propose to augment an RL agent with the option of using a motion planner and formulate the learning problem as a semi-MDP. Unlike their work, we propose to lift the action space completely instead of using a semi-MDP setup. We also tackle much more complex domain than the domain of 2D block pushing with stationary arm in Yamada et al. [33]. Wu et al. [34] is the most similar method to ours. They propose an approach for mobile manipulation that learns to set goals for a 2D navigation motion planner by selecting pixels on the local occupancy map. Their "spatial action maps" serve as a new action space for policy learning with DQN [18]. As we will see later, this approach is similar to our variant of ReLMoGen with Q-learning based RL (see ReLMoGen-D Sec. III-A). However, our solution enables both navigation and manipulation with a robotic arm. Moreover, we demonstrate with ReLMoGen-R (Sec. III-A) that our proposed method can be also applied to policy-gradient methods.

Hierarchical Reinforcement Learning: Often in HRL solutions, the main benefit comes from a better exploration thanks to a longer temporal commitment of a low-level policy towards the goal commanded by a high-level policy [16]. Therefore, in many HRL methods the high level learns to set subgoals for the low level [35, 36, 37, 15, 38, 14]. Notably, Konidaris et al. [39] applies HRL to Interactive Navigation (IN) tasks, problems that require the agent to interact with the environment to reach its goal. Their algorithms generate actions to solve subcomponents of the original task and reuses them to solve new task instances. Li et al. [17] propose an end-to-end HRL solution for IN that also decides on the different parts of the embodiment to use for each subgoal. HRL solutions often suffer from training instability because the high level and low level are learned simultaneously. Previous attempts to alleviate this include off-policy corrections [15], hindsight subgoal sampling [14] and low-level policy pre-training [35]. While ReLMoGen is not a full HRL solution, it is structurally similar: a high level sets subgoals for a low level. Therefore, ReLMoGen benefits from better exploration due to temporal abstraction while avoiding the aforementioned cold-start problem because our low level is not a learned policy but a predefined MG solution.

An orthogonal but related area of RL research is deep exploration. These methods typically rely on uncertainty modeling, random priors, or noisy data, and have proven to be effective in simple tasks such as Cartpole, DeepSea and Atari games [12, 40, 13]. Closest to our task setup, Ciosek et al. [41] proposes an Optimistic Actor Critic that approximates a lower and upper confidence bound on the Q-functions, and shows favorable results in MuJoCo environments. ReLMoGen can be thought of as improving exploration, not by relying on optimism of Q-functions, but by lifting the action space, circumventing the hard exploration problem with commitment towards an interaction point.

III. RL with Motion Generation

We formulate a visuo-motor mobile manipulation control task as a time-discrete Partially Observable Markov Decision Process (POMDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; $\mathcal{O}$ is the observation space; $\mathcal{T}(s'|s,a), s \in \mathcal{S}, a \in \mathcal{A}$, is the state transition model; $\mathcal{R}(s,a) \in \mathbb{R}$ is the reward function; γ is the discount factor. We assume that the state is not directly observable and we learn a policy

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.

$\pi(a|o)$ conditioned on observations $o \in \mathcal{O}$. Herein, the agent following the policy π obtains an observation o_t at time t and performs an action a_t, receiving from the environment an immediate reward r_t and a new observation o_{t+1}. The goal of RL is to learn an action selection policy π that maximizes the discounted sum of rewards.

We assume that $\mathcal{A}$ is the original action space for continuous control of the mobile manipulator, e.g. positions or velocities of each joint. Our main assumption in ReLMoGen is that, for the considered types of mobile manipulation tasks, a successful strategy can be described as a sequence of subgoals: $\tau_{succ} = \{a'_0, \ldots, a'_{\tau-1}\}$. Each subgoal a'_i corresponds to a goal configuration for a motion generator either to move the base to a desired location or to move the robot's end-effector to a position and perform a parameterized interaction. These subgoals are generated by a subgoal generation policy, π_{SGP}. As shown in Sec. IV, in this work we focus on mobile manipulation tasks that can be solved by applying a parameterized pushing interaction after positioning the arm at a subgoal location; however, we do not find any aspect of ReLMoGen that fundamentally restricts it from utilizing other parameterized interactions at the desired end-effector position (e.g. pull, pick, place, ...).

To generate collision-free trajectories, we propose to query at each policy step a motion generator, MG, a non-preemptable subroutine that attempts to find and execute an embodiment-compliant and collision-free trajectory to reach a given subgoal. The motion generator takes as input a subgoal from the subgoal generator policy, a', and outputs a sequence of variable length T of low-level actions that are executed, $MG(a') = (a_0, \ldots, a_{T-1})$. In case the MG fails to find a path, it returns a no-op. The proposed ReLMoGen solution is composed of two elements: the motion generator, MG, and the subgoal generation policy, π_{SGP}.

Based on the MG, we build with ReLMoGen a new *lifted* PODMP, $\mathcal{M} = (\mathcal{S}, \mathcal{A}', \mathcal{O}, \mathcal{T}', \mathcal{R}', \gamma)$, where $a' \in \mathcal{A}'$ is a new action space of subgoals to query the MG. $\mathcal{T}'(s'|s, a'), s, s' \in \mathcal{S}, a' \in \mathcal{A}'$ is the new transition function that corresponds to iteratively querying the original transition function $\mathcal{T}(s'|s, a)$ for T times starting at s_t, with the sequence of actions returned by the MG, $MG(a') = (a_t, \ldots, a_{t+T-1})$. Finally, the *lifted* reward is defined as the accumulated reward obtained from executing the sequence of actions from the MG, $\mathcal{R}'(s_t, a'_t) = \sum_{k=t}^{t+T-1} \mathcal{R}(s_k, a_k)$. The subgoal generator policy is trained to solve this lifted POMDP, taking in observations o and outputting actions a', subgoals for the MG. The composition of the trained subgoal generator policy and the MG is a policy that solves the original POMDP: $\pi = MG(\pi_{SGP})$. As a summary, ReLMoGen lifts the original POMDP problem into this new formulation that can be more easily solved using reinforcement learning.

A. ReLMoGen: RL with Motion Generation Action Space

In this section, we propose our solutions to the lifted POMDP created by ReLMoGen for mobile manipulation tasks. As explained above, ReLMoGen is a general procedure that comprises two elements, a subgoal generation policy (SGP) and a motion generator (MP). We show that ReLMoGen can be instantiated with continuous and discrete action parametrization with two alternative SGPs that we formalize.

Observations: Our subgoal generation policy, π_{SGP}, takes in sensor inputs and outputs MG subgoals (see Fig. 1). We assume three common sensor sources (an RGB image and a depth map from a robot's RGB-D camera, and a single-beam LiDAR scan), and, optionally, additional task information. For navigation and Interactive Navigation tasks, the task information is the final goal location together with the next N waypoints separated d meters apart on the shortest path to the final goal, both relative to the current robot's pose ($N = 10$ and $d = 0.2\,\mathrm{m}$ in our experiments). We assume the goal and the shortest path are provided by the environment and computed based on a floor plan that contains only stationary elements (e.g. walls), regardless of dynamic objects such as doors, and obstacles (see Fig. 3). For mobile manipulation (MM) tasks, there is no additional task information.

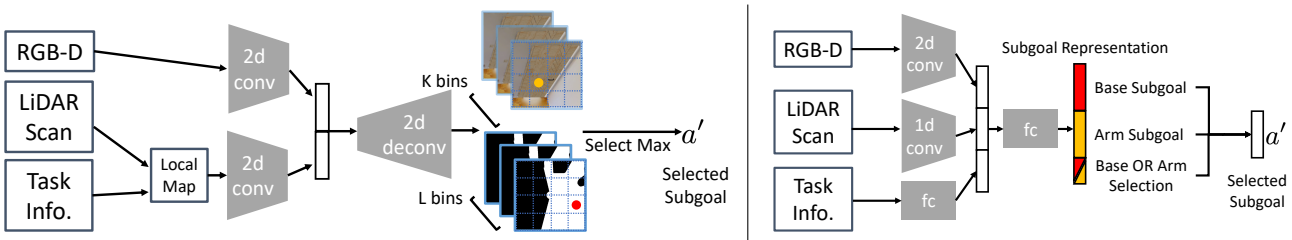
Continuous Action Parameterization Method - SGP-R: We call our subgoal generation policy for continuous action parameterization SGP-R, where "R" indicates regression. We denote this implementation of ReLMoGen as ReLMoGen-R. The high-level idea is to treat the space of subgoals as a continuous action space, in which the policy network predicts (regresses) one vector. Based on the observation, the policy outputs 1) a base subgoal: the desired base 2D location in polar coordinates and the desired orientation change, 2) an arm subgoal: the desired end-effector 3D location represented by a (u, v) coordinate on the RGB-D image to initiate the interaction, and a 2D interaction vector relative to this position that indicates the final end-effector position after the interaction, and 3) a binary variable that indicates whether the next step is a base-motion or an arm-motion phase (see Fig. 2b). These subgoals are executed by the motion generator introduced in the Section III-B. We train SGP-R using Soft Actor-Critic [19].

Discrete Action Parameterization Method - SGP-D: We call our subgoal generation policy for discrete action parameterization SGP-D, where "D" indicates dense prediction. We denote this implementation of ReLMoGen as ReLMoGen-D. This parameterization aligns the action space with the observation space, and produces dense Q-value maps. The policy action (subgoal) corresponds to the pixel with the maximum Q-value. This parametrization is similar to the "spatial action maps" by Wu et al. [34]. Unlike their policy, our SGP-D predicts two types of action maps: one for base subgoals spatially aligned with the local map and the other for arm subgoals spatially aligned with the RGB-D image from the head camera (see Fig. 2a). To represent the desired orientation of the base subgoal, we discretize the value into L bins per pixel for the base Q-value maps. Similarly, for the desired pushing direction of the arm subgoal, we have K bins per pixel for the arm Q-value maps ($K = L = 12$ in our experiments). We train SGP-D using Deep Q-learning [18].

B. Motion Generation for Base and Arm

We use a motion generator to lift the action space for robot learning from low-level motor actuation to high-level "subgoals". The motion generator consists of two modules: 1) a motion planner that searches for trajectories to a given subgoal using a model of the environment created based on current sensor information, and 2) a set of common low-level controllers (joint space) that execute the planned

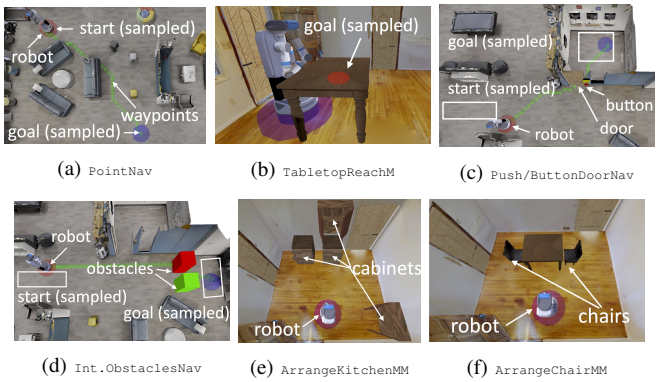
Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.



(a) Subgoal Generation Policy SGP-D

(b) Subgoal Generation Policy SGP-R

Fig. 2: Two types of action parameterization of ReLMoGen and network architecture of SGP-D and SGP-R.



(a) PointNav (b) TabletopReachM (c) Push/ButtonDoorNav

(d) Int.ObstaclesNav (e) ArrangeKitchenMM (f) ArrangeChairMM

Fig. 3: The simulation environments and tasks. (a)(b) navigation-only and manipulation-only tasks, (c)(d) three Interactive Navigation tasks, (e)(f) two Mobile Manipulation tasks.

trajectories. In our solution, we use a bidirectional rapidly-exploring random tree (RRT-Connect) [20] to plan the motion of the base and the arm, although we also experiment with probabilistic road-maps (PRM) [21] in our evaluation.

The motion planner for the base is a 2D Cartesian space RRT that searches for a collision-free path to the base subgoal location on the local map generated from the most recent LiDAR scan. The base subgoals are represented as the desired base 2D locations and orientations.

The motion planner for the arm comprises a 3D Cartesian space RRT and a simple Cartesian space inverse kinematics (IK) based planner. The arm motion is made of two phases: 1) the motion from the initial configuration to the selected subgoal location, and 2) the pushing interaction starting from the subgoal location. For the first phase, the 3D RRT searches for a collision-free path to reach the subgoal location. If the first phase succeeds, as the second phase, the simple IK-based planner is queried to find a sequence of joint configurations to move the end effector in a straight line from the subgoal location along the specified pushing direction. Since the intent of the second phase is to interact with the environment, the path is not collision-free. The arm subgoals are thus represented as the desired end-effector 3D locations and parameterized pushing actions. We hypothesize that the pushing actions can be replaced by other types of parameterized actions (e.g. grasping and pulling). More details about algorithm description, network structure, training procedure and hyperparameters can be found on our website.

IV. Experimental Evaluation

We evaluate our method on seven different tasks. These tasks include navigation, manipulation, Interactive Navigation, and Mobile Manipulation (see Fig. 3). We believe these tasks represent paradigmatic challenges encountered by robots operating in realistic environments.

Navigation-Only and Manipulation-Only Tasks: PointGoal navigation [42, 43] and tabletop tasks [44] are mature robotic benchmarks. In `PointNav`, the robot needs to move to a goal without collision. In `TabletopReachM`, the robot needs to touch a point on the table with its end-effector.

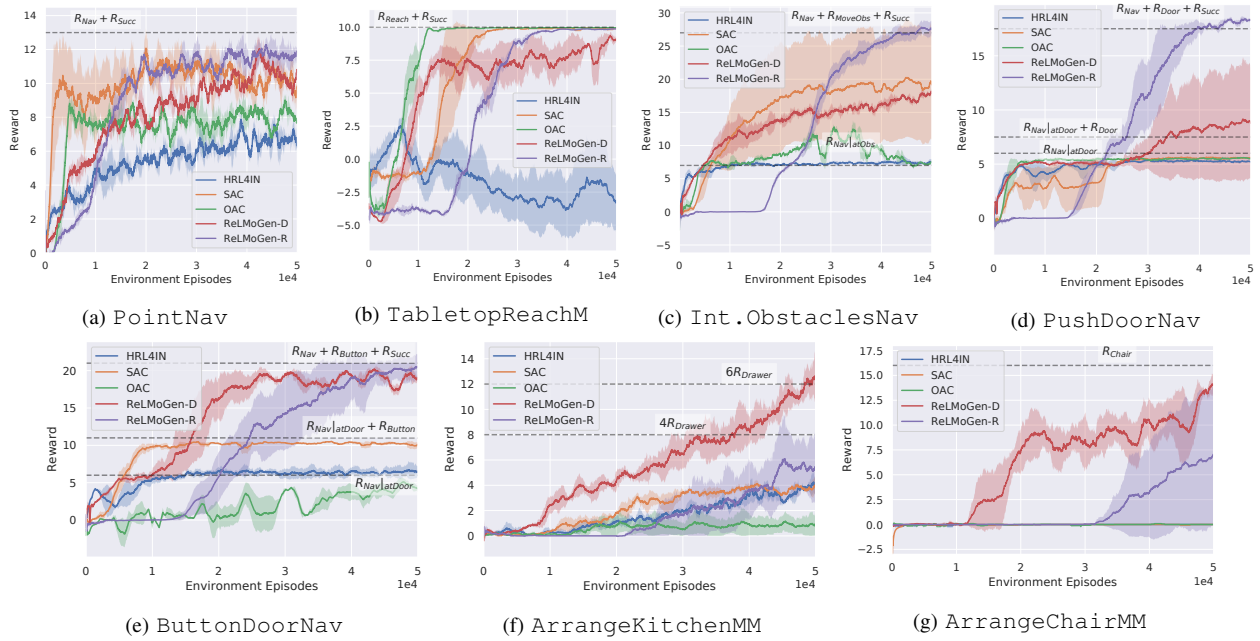
Interactive Navigation (IN) Tasks: In these tasks the robot needs to interact with the environment to change the environment's state in order to facilitate or enable navigation [45]. In `PushDoorNav` and `ButtonDoorNav`, the robot needs to enter a room behind a closed door, by pushing the door or pressing a button, respectively. In `InteractiveObstaclesNav` task, the robot is blocked by two objects and needs to push them aside to reach the goal. Only one of the objects can be pushed, and the agent needs to judge solely based on visual appearance (color). These tasks require the robot to place its base properly to interact with the objects [46, 47], and to infer where to interact based on a correct interpretation of the RGB-D camera information (e.g. finding the door button).

Mobile Manipulation (MM) Tasks: These are long-horizon tasks known to be difficult for RL [48, 17], making it a good test for our method. We created two MM tasks, `ArrangeKitchenMM` and `ArrangeChairMM`. In `ArrangeKitchenMM`, the robot needs to close cabinet drawers and doors randomly placed and opened. The challenge is that the robot needs to find the cabinets and drawers using the RGB-D information, and accurately actuate them along their degrees of freedom. In `ChairArrangeMM`, the robot needs to push chairs under a table. The opening under the table is small so the push needs to be accurate. Object locations are unknown to the robot. Both tasks can be thought of as an ObjectNav [42] task followed by a manipulation task. The reward is only given when the robot makes progress during the manipulation phase.

All experiments are conducted in iGibson Environment [45]. The Navigation and Interaction Navigation tasks are performed in a 3D reconstruction of an office building. The Mobile Manipulation and Tabletop tasks are performed in a model of a residential house (`Samuels`) from [45], populated with furniture from Motion Dataset [49] and ShapeNet Dataset [50]. We randomize the initial pose of the robot, objects and goals across training episodes so that the agent cannot simply memorize the solution.

For (Interactive) Navigation tasks, we have dense reward, R_{Nav}, that encourages the robot to minimize the geodesic

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.



(a) `PointNav` (b) `TabletopReachM` (c) `Int.ObstaclesNav` (d) `PushDoorNav`

(e) `ButtonDoorNav` (f) `ArrangeKitchenMM` (g) `ArrangeChairMM`

Fig. 4: Training curves for ReLMoGen and the baselines (SAC, OAC, and HRL4IN). ReLMoGen achieves higher reward with the same number of environment episodes and higher task completion for all seven tasks while the baselines often converge to sub-optimal solutions. The curve indicates the mean and standard deviation of the return across three random seeds. Note that the x-axis indicates environment episodes rather than steps to allow for a fair comparison between solutions that use actions with different time horizons.

distance to the goal, and success reward, R_{Succ}, for task completion. We have bonus reward for the robot to push obstacles, doors and buttons, denoted as $R_{MoveObs}$, R_{Door} and R_{Button}. For Mobile Manipulation tasks, we have dense reward for the robot to close drawers and cabinets, or to tuck chairs, denoted as R_{Drawer} and R_{Chair}. We don't provide reward for the robot to approach these objects. Episodes terminate when any part of the robot body other than the gripper collides with the environment. More detailed reward definition and evaluation metrics are on our website.

A. Baselines

SAC (on joint velocities): We run SAC [19] directly on joint velocities for all the joints on our robot (2 wheels, 1 torso joint, 7 arm joints), similar to previous work on visuomotor control [10].

OAC (on joint velocities): We run a variant of SAC called OAC presented by Ciosek et al. [41]. This work applies the principle of optimism in face of uncertainty to Q-functions and outperform SAC in several continuous control tasks [41].

HRL4IN: We run the hierarchical RL algorithm presented by Li et al. [17]. This work shows good performance for IN tasks. Similar to ours, a high-level policy produces base and arm subgoals and a variable to decide the part of the embodiment to use. Different from ours, this method uses a learned low-level policy instead of a motion generator. With this baseline we evaluate the effect of integrating RL and MG instead of learning a low-level policy from scratch.

The action space of ReLMoGen and the baselines have drastically different time horizons. For fair comparison, we set the episode length to be roughly equivalent in wall-clock time of simulation across algorithms: 25 subgoal steps for ReLMoGen and 750 joint motor steps for the baselines. To evaluate performance, we use success rate and SPL [42] for navigation tasks, and task completion (number of drawers/cabinets closed, chairs tucked within $10°/10\,\mathrm{cm}$ and $5°/5\,\mathrm{cm}$) for mobile manipulation tasks.

B. Analysis

We aim at answering the following research questions with our analysis in this subsection.

Can ReLMoGen solve a wide variety of robotic tasks involving navigation and manipulation? In Table I, we show the task completion metrics across all tasks for our methods and baselines. In a nutshell, our method achieves the highest performance across all seven tasks. It also exhibits better sample efficiency than our baselines (see Fig. 4).

SAC and OAC baseline have comparable performance to our methods for simpler tasks such as `PointNav` and `TabletopReachM` but fail completely for harder ones, such as `PushDoorNav` and `ChairArragementMM`, due to collisions or their inability to identify objects that are beneficial to interact with. OAC only outperforms SAC with a small margin in one task, which suggests that it remains an open research question on how to conduct deep exploration in robotics domain with high dimensional observation space and continuous action space. To our surprise, HRL4IN baseline perform worse than SAC baseline for several tasks. This is potentially caused by our deviation from the original task setup in [17] since we do not allow collisions with the robot base during exploration, while HRL4IN has a collision prone low-level policy. This is consistent with our insight that using MG instead of a learned low-level policy makes it easier to train the subgoal generation policy, and that RL is best suited to learn the mapping from observations to subgoals.

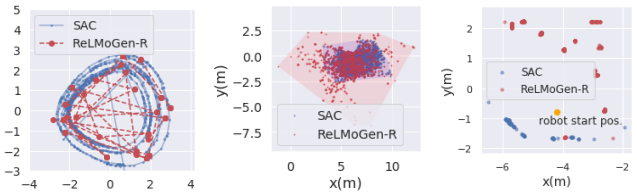
One common failure case for the baselines in IN tasks is that the agent harvests all the navigation reward by approaching the goal but gets stuck in front of doors or obstacles,

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.

Task	PointNav		TabletopReachM	ArrangeKitchenMM		ArrangeChairMM	
Metric	SPL	SR	SR	# Closed 5°/5 cm	# Closed 10°/10 cm	# Closed 5 cm	# Closed 10 cm
ReLMoGen-D (ours)	0.57/0.02/0.58	0.68/0.01/0.68	0.95/0.02/0.96	**4.35/1.20/5.72**	**6.10/1.05/7.3**	**0.21/0.03/0.23**	**0.36/0.06/0.43**
ReLMoGen-R (ours)	**0.63/0.09/0.67**	**0.72/0.06/0.77**	**1.0/0.0/1.0**	3.43/0.61/3.94	4.91/0.51/5.25	0.06/0.10/0.17	0.11/0.20/0.34
HRL4IN [17]	0.27/0.01/0.28	0.33/0.01/0.35	0.09/0.07/0.19	3.0/0.23/3.3	4.67/0.20/4.95	0.0/0.0/0.0	0.0/0.0/0.0
SAC (joint vel.) [19, 10]	0.60/0.04/0.65	0.60/0.04/0.65	**1.0/0.0/1.0**	3.42/0.19/3.6	4.95/0.29/5.24	0.0/0.0/0.0	0.0/0.0/0.0
OAC (joint vel.) [41]	0.45/0.01/0.46	0.46/0.01/0.47	**1.0/0.0/1.0**	1.99/0.61/2.60	3.55/0.48/4.02	0.0/0.0/0.0	0.0/0.0/0.0

Task	PushDoorNav		ButtonDoorNav		InteractiveObstaclesNav	
Metric	SPL	SR	SPL	SR	SPL	SR
ReLMoGen-D (ours)	0.36/0.36/0.72	0.41/0.40/0.80	0.42/0.17/0.57	0.50/0.19/0.66	0.54/0.011/0.55	0.58/0.02/0.60
ReLMoGen-R (ours)	**0.80/0.02/0.83**	**0.97/0.02/0.99**	**0.51/0.15/0.61**	**0.73/0.21/0.87**	**0.76/0.01/0.87**	**0.79/0.11/0.91**
HRL4IN [17]	0.0/0.0/0.0	0.0/0.0/0.0	0.0/0.0/0.0	0.0/0.0/0.0	0.0/0.0/0.0	0.0/0.0/0.0
SAC (joint vel.) [19, 10]	0.0/0.0/0.0	0.0/0.0/0.0	0.00/0.01/0.01	0.01/0.01/0.01	0.50/0.36/0.84	0.51/0.37/0.87
OAC (joint vel.) [41]	0.0/0.0/0.0	0.0/0.0/0.0	0.00/0.00/0.01	0.01/0.00/0.01	0.00/0.00/0.01	0.01/0.01/0.01

TABLE I: Task completion metrics for two version of ReLMoGen, one using DQN with discrete subgoal parameterization (ReLMoGen-D) and one using SAC with continous subgoal parameterization (ReLMoGen-R). We compare with two baselines (see Sec. IV-A). The entries of this table are in the format of mean/std/max over 3 random seeds and the method with the highest mean value is highlighted in bold.



(a) Latent Space (b) Cartesian Space (c) Interaction map

Fig. 5: Exploration of ReLMoGen-R and SAC. (a) shows the 2D projection of latent state space: SAC traverses nearby states with low-level actions, while ReLMoGen-R jumps between distant states linked by a motion plan. (b) shows the physical locations visited by ReLMoGen-R and SAC in 100 episodes: ReLMoGen-R covers a much larger area. (c) shows a top-down map of meaningful interactions (duration $\geq$1s) during exploration. ReLMoGen-R is able to interact with the environment more than SAC.

Base MP	Arm MP	Success rate
RRT-Connect	**RRT-Connect**	**0.99**
RRT-Connect	Lazy PRM	1.0 (+**0.01**)
Lazy PRM	RRT-Connect	0.99 (+**0.0**)
Lazy PRM	Lazy PRM	1.0 (+**0.01**)

(a) PushDoorNav Task

Base MP	Arm MP	# Closed (10°/10 cm)
RRT-Connect	**RRT-Connect**	**5.25**
RRT-Connect	Lazy PRM	5.0 (−**0.25**)
Lazy PRM	RRT-Connect	5.18 (−**0.07**)
Lazy PRM	Lazy PRM	5.09 (−**0.16**)

(b) ArrangeKitchenMM Task

TABLE II: Our policy trained with RRT-Connect as the motion planner for base and arm can perform equally well when changing to Lazy PRM at test time (the first row shows the training setup).

failing to learn meaningful interaction with them. On the other hand, both our ReLMoGen implementations with SGP-R and SGP-D are able to achieve significant success in tasks that involve precise manipulation (e.g. `ButtonDoorNav`), intermittent reward signal (e.g. `ArrangeChairMM` and `ArrangeKitchenMM`) and alternative phases of base and arm motion (all IN and MM tasks). Empirically, ReLMoGen-D outperforms ReLMoGen-R for tasks that involve more fine-grained manipulation due to its Q-value estimation at every single pixel, but seems to be less sample efficient than it for tasks that only require coarse manipulation. We argue that the main advantage of ReLMoGen is that it explores efficiently while maintaining high "subgoal success rates" thanks to its embedded motion generators, resulting in stable gradients during training. As a bonus, ReLMoGen performs an order of magnitude fewer gradient updates than the baselines, which translates to a much shorter wall-clock time for training (on average 7x times faster). Finally, our ReLMoGen-D model outputs highly interpretable Q-value maps: high Q-value pixels correspond to rewarding interactions, such as buttons, cabinet doors and chair backs. More visualizations can be found on our website.

Is ReLMoGen better at exploration? Fig. 5 shows the exploration pattern of a random policy for SAC baseline and for ReLMoGen-R. Specifically, we visualize the distribution of the states visited by the policy at the beginning of training. We project the neural network embedding of the visited states onto a 2D plane showing the first two principal components. For SAC and ReLMoGen-R, the trajectories of ten episodes are shown in Fig. 5(a). We can see that SAC baseline only travels between adjacent states in the feature space because it explores in **joint space** (considering wheels as joints). On the other hand, ReLMoGen can jump between distant states, as long as they can be connected by the motion generator, because it explores in **subgoal space**. The visited states by ReLMoGen are indicated in red dots connected with dashed lines. This is also evident when we plot the visited states in physical, Cartesian space in Fig. 5(b). From Fig. 5(c), we can see ReLMoGen have more meaningful interactions with the environment during exploration than SAC.

Can ReLMoGen generalize to different types of motion planners? During training, we used RRT-Connect as our motion planner. We want to test whether our method can zero-shot generalize to a new motion planner, namely Lazy PRM [21], during test time. We swapped base and/or arm motion planners and tried different parameters (e.g. number of trajectory optimization iterations) for our system, and observed minimal performance drop (see Table. II). Although different motion planners have different sampling schemas and timeout criteria, the subgoals generated by our policy can seamlessly transfer between them. This demonstrates strong practicality and flexibility of our approach.

V. Conclusion

We introduce ReLMoGen, a hierarchical framework that integrates classical motion generation with reinforcement learning to solve mobile manipulation tasks. ReLMoGen leverages the best from both worlds: learning complex subgoal prediction from high dimensional observations via RL and precise low-level action execution via MG. We demonstrate better task completion and higher training efficiency compared to other learning based approaches. The learned policies with ReLMoGen are also robust and can transfer to different motion planners after training.

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[5] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[6] H. Quan, Y. Li, and Y. Zhang, "A novel mobile robot navigation method based on deep reinforcement learning," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, p. 1729881420921672, 2020.

[7] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, 2020.

[8] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*, 2018, pp. 651–673.

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

[10] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[11] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.

[12] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Advances in neural information processing systems*, 2016, pp. 4026–4034.

[13] I. Osband, B. Van Roy, D. J. Russo, and Z. Wen, "Deep exploration via randomized value functions." *Journal of Machine Learning Research*, vol. 20, no. 124, pp. 1–62, 2019.

[14] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," *International Conference on Learning Representations*, 2019.

[15] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 3303–3313.

[16] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine, "Why does hierarchy (sometimes) work so well in reinforcement learning?" *arXiv preprint arXiv:1909.10618*, 2019.

[17] C. Li, F. Xia, R. Martín-Martín, and S. Savarese, "Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators," in *Conference on Robot Learning*, 2020, pp. 603–616.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[19] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[20] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[21] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 521–528.

[22] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *arXiv preprint arXiv:1804.09364*, 2018.

[23] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Beauty and the beast: Optimal methods meet learning for drone racing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 690–696.

[24] T. Jurgenson and A. Tamar, "Harnessing reinforcement learning for neural motion planning," in *Proceedings of Robotics: Science and Systems*, Freiburg im Breisgau, Germany, June 2019.

[25] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[26] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Conference on Robot Learning (CoRL)*, 2019.

[27] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[28] N. Jetchev and M. Toussaint, "Trajectory prediction in cluttered voxel environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2523–2528.

[29] M. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Intelligent robots and systems*, 2018.

[30] K. Ota, Y. Sasaki, D. K. Jha, Y. Yoshiyasu, and A. Kanezaki, "Efficient exploration in constrained environments with goal-oriented reference path," *arXiv*

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.

preprint arXiv:2003.01641, 2020.

[31] Y. Jiang, F. Yang, S. Zhang, and P. Stone, "Integrating task-motion planning with reinforcement learning for robust decision making in mobile robots," in *In Proceedings of the AAMAS*, 2019.

[32] A. Dragan, G. J. Gordon, and S. Srinivasa, "Learning from experience in manipulation planning: Setting the right goals," in *In Proceedings of the ISRR*, 2011.

[33] J. Yamada, G. Salhotra, Y. Lee, M. Pflueger, K. Pertsch, P. Englert, G. S. Sukhatme, and J. J. Lim, "Motion planner augmented action spaces for reinforcement learning," *RSS Workshop on Action Representations for Learning in Continuous Control*, 2020.

[34] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser, "Spatial Action Maps for Mobile Manipulation," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[35] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver, "Learning and transfer of modulated locomotor controllers," *arXiv preprint arXiv:1610.05182*, 2016.

[36] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in neural information processing systems*, 2016, pp. 3675–3683.

[37] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3540–3549.

[38] O. Nachum, S. Gu, H. Lee, and S. Levine, "Near-optimal representation learning for hierarchical reinforcement learning," *International Conference on Learning Representations*, 2018.

[39] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Autonomous skill acquisition on a mobile manipulator," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[40] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 8617–8629.

[41] K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann, "Better exploration with optimistic actor critic," in *Advances in Neural Information Processing Systems*, 2019, pp. 1787–1798.

[42] P. Anderson *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[43] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets *et al.*, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[44] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *arXiv preprint arXiv:1608.05742*, 2016.

[45] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 713–720, April 2020.

[46] D. Berenson, J. Kuffner, and H. Choset, "An optimization approach to planning for mobile manipulation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1187–1192.

[47] E. Klingbeil, A. Saxena, and A. Y. Ng, "Learning to open new doors," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2751–2757.

[48] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, "Learning mobile manipulation through deep reinforcement learning," *Sensors*, vol. 20, no. 3, p. 939, 2020.

[49] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2motion: Joint analysis of motion parts and attributes from 3d shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8876–8884.

[50] A. X. Chang *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

Authorized licensed use limited to: Wayne State University. Downloaded on January 17,2022 at 01:15:16 UTC from IEEE Xplore. Restrictions apply.