Learning Needle Pick-and-Place Without Expert Demonstrations

Rokas Bendikas, Valerio Modugno, *Member, IEEE*, Dimitrios Kanoulas, *Member, IEEE*, Francisco Vasconcelos, and Danail Stoyanov, *Senior Member, IEEE*

***Abstract*—We introduce a novel approach for learning a complex multi-stage needle pick-and-place manipulation task for surgical applications using Reinforcement Learning without expert demonstrations or explicit curriculum. The proposed method is based on a recursive decomposition of the original task into a sequence of sub-tasks with increasing complexity and utilizes an actor-critic algorithm with deterministic policy output. In this work, exploratory bottlenecks have been used by a human expert as convenient boundary points for partitioning complex tasks into simpler subunits. Our method has successfully learnt a policy for the needle pick-and-place task, whereas the state-of-the-art TD3+HER method is unable to achieve success without the help of expert demonstrations. Comparison results show that our method achieves the highest performance with a 91% average success rate.**

***Index Terms*—Surgical robotics, autonomous agents, reinforcement learning, transfer learning.**

I. Introduction

END-TO-END control policies obtained via Reinforcement Learning (RL) methods are becoming an emerging alternative to traditional robot control approaches [1], [2]. Such control policy learning methods are well suited for short-horizon problems containing smooth state spaces and continuous reward functions, that can be effectively explored by an agent. Nevertheless, robotic manipulation settings often expect behaviours that lead to very specific goals, such as picking and placing an object at a desired location or inserting a peg in a hole [3]. While sparse reward functions describe such environments in a more natural way, they make learning much more challenging [4].

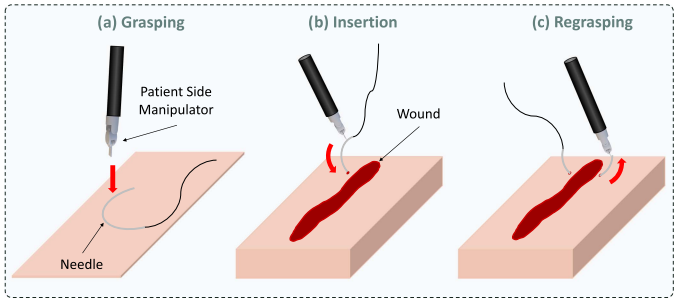


Fig. 1. Example of a sequential multi-stage manipulation required for a wound suturing procedure. It contains three steps: (a) Needle grasping, (b) Needle insertion into the tissue using Patient Side Manipulator, (c) Needle regrasping.

Learnt control policies can prove especially beneficial in addressing surgical manipulation problems. This is due to their ability to overcome typical obstacles that arise when using conventional non-learning approaches. Learnt methods often reflect much higher performance speed, which is essential when working in dynamic surgical environments [5]. Further, they have the ability to generalize and adapt their actions in real-time based on changing environmental conditions which are key toward the implementation of fully autonomous systems [6]. Therefore, learned control policies have the potential to enhance the effectiveness of surgical needle manipulation tasks, ultimately improving patient outcomes.

In the context of surgical needle manipulation, it can be very complex to learn such policy due to a range of problems, associated with the nature of the clinical setting. Such tasks often contain a multi-stage problem structure, where the execution of each stage has to be in the correct order [7]. For example, suturing tissue during robotic surgery to achieve anastomosis requires needle grasping, insertion, and needle regrasping [8], as shown in Fig. 1. Failing one of the stages often requires modifying the structure of the task. Furthermore, needle manipulation procedures require extremely accurate and precise control throughout the operation. Intuitively, grasping a needle is much more difficult than grasping a larger object, due to its size and geometry, requiring a very precise grasping policy. Furthermore, the material properties of the needle make it hard to grasp, as smooth metallic surface has very reduced frictional properties.

Manuscript received 17 December 2022; accepted 28 March 2023. Date of publication 12 April 2023; date of current version 25 April 2023. This letter was recommended for publication by Associate Editor J. Y. Wu and Editor P. Valdastri upon evaluation of the reviewers' comments. This work was supported in part by the Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS) under Grant 203145/Z/16/Z, in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grants EP/S021566/1 and EP/P012841/1, in part by the UKRI Future Leaders Fellowship (RoboHike) under Grant MR/V025333/1, and in part by the Royal Academy of Engineering Chair in Emerging Technologies Scheme. *(Corresponding author: Rokas Bendikas.)*

Francisco Vasconcelos and Danail Stoyanov are with the Surgical Robot Vision Group, WEISS, Department of Computer Science, University College London, WC1E 6BT London, U.K. (e-mail: f.vasconcelos@ucl.ac.uk; danail.stoyanov@ucl.ac.uk).

Rokas Bendikas is with the Surgical Robot Vision Group, WEISS, Department of Computer Science, University College London, WC1E 6BT London, U.K., and also with the Robot Perception and Learning Lab, Department of Computer Science, University College London, WC1E 6BT London, U.K. (e-mail: rokas.bendikas.21@ucl.ac.uk).

Valerio Modugno and Dimitrios Kanoulas are with the Robot Perception and Learning Lab, Department of Computer Science, University College London, WC1E 6BT London, U.K. (e-mail: valerio.modugno@diag.uniroma1.it; dkanoulas@gmail.com).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3266720, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3266720

The sequential multi-stage structure of such tasks, augmented with strict grasping requirements, introduces bottleneck regions that limit the capability of the learner to complete each subtask in the sequence. Following the definition introduced in [9], bottlenecks are regions in the state space always traversed by trajectories that solve the desired task. In the context of multi-stage sequential manipulation and especially for surgical applications, these regions can become very narrow, hence hindering the capability of the learner to solve the desired task. Here, we will refer to them as *exploratory bottlenecks* to stress their role as limiting factors in learning an effective control policy.

Recently, benchmarks have emerged, each aiming to solve a sub-part of a bigger surgical procedure. *NeedlePickAndPlace* is a common surgery-related benchmark, where the agent has to pick up a needle and place it in a different location. The task is crucial for solving autonomous suturing and thus, is greatly studied in the surgical robotics community [10], [11], [12].

In this letter, we propose a new framework for learning *NeedlePickAndPlace* task, without reward shaping, demonstrations, or the explicit definition of a curriculum. We break the task down into two sequential sub-tasks that encapsulate each other recursively. Defining each subtask as a combination of all the previous ones introduces a methodological difference with respect to classical approaches, such as Transfer Learning (TL) or Curriculum Learning (CL). In these methods, each subtask is learned in isolation and, successively, the acquired knowledge is transferred by exploiting specific similarity metrics among sub-problems [13].

We utilize a Patient Side Manipulator (PSM) to perform the manipulation task. PSM is a secondary component of a daVinci Research Kit (dVRK), constrained by a fulcrum point invariant to the joint configuration, i.e., a Remote Center of Motion (RCM), thus allowing to perform laparoscopic procedures [14], [15].

A. Contributions

In this work, we propose a novel method for learning the *NeedlePickAndPlace* tasks without using expert demonstration or explicit CL. To our knowledge, our approach has not been previously proposed in the surgical robotic control domain, and it achieves state-of-the-art performance. Summarizing our contributions, we introduce:

1) A learnt control policy that is able to perform a needle pick-and-place task using a Patient Side Manipulator.
2) A task restructuring and goal definition technique via implicit CL.
3) A novel approach to preload a replay buffer with partially completed trajectories that are bootstrapped from a prior actor.
4) A composite actor loss that leverages the knowledge from the prior critic using the Q-function transfer method.

II. Related Work

Automation of surgical robotics is an emerging area of research, that has gained significant attention in recent years [16]. Autonomous surgical needle manipulation is a sub-area of the field, that aims to develop advanced robotic systems capable of manipulating surgical needles with a high level of precision and accuracy, without requiring direct human intervention. [17], [18], [19], [20]. Such tasks often contain a multi-stage structure and are extremely hard to learn [10]. Nevertheless, the ability to accurately and precisely manipulate surgical needles is a critical step in various surgical procedures, and the development of autonomous systems that can perform this task has the potential to revolutionize the field of minimally invasive surgery [15]. Several researchers have explored different methods for autonomous surgical needle manipulation, including vision-based approaches [6], non-learning approaches [21], [22] as well as learning-based approaches [23].

Non-learning methods often exhibit higher operational accuracy, but lack performance speed [12], [24], due to the resource-intense path-planning algorithms. On the other hand, the state-of-art learning-based approaches employ the Learning from Demonstrations (LfD) paradigm, where the agents try to reproduce demonstrations from an expert [5], [6], [17]. Whilst LfD approaches offer great sample efficiency, the required data collection can be cumbersome. In the context of surgical robotics, this is also extremely resource-expensive, as generating good-quality demonstrations may require a trained surgeon or may vary across different surgeons.

Hierarchical Reinforcement Learning (HRL) is an effective learning paradigm for multi-stage tasks. However, it has not been applied in surgical settings. In [25], the authors describe policy tree approaches, a group of HRL methods consisting of two stages: subtask subdivision and optimal blending policy learning. HRL requires a set of pretrained policies to solve each subtask prior to learning the global policy. This reliance on pre-learned subtask policies can limit the optimality and adaptability of the final hierarchical policy. Our approach differs from policy tree HRL methods in that it uses a single end-to-end neural network for the final policy, whereas policy tree methods have multiple networks. This reduces memory footprint during operation, as all networks do not need to be stored in memory simultaneously.

Curriculum Learning (CL) is a widely used technique for learning goal-conditioned tasks with a sparse reward scheme. CL involves gradually increasing the complexity of the agent's experiences during the learning process, allowing it to learn simpler behaviors before moving on to harder ones [26]. CL methods can be classified as either *Explicit* or *Implicit*. *Explicit* CL requires the manual definition of an increasing difficulty strategy for the task, while *Implicit* CL methods allow the curriculum to emerge as a side effect of the training strategy [27].

In the field of robotics, *Explicit* CL is commonly used to define an increasing difficulty in the objective or operational environment [28], [29], [30]. This approach allows for better task learning, but it requires the manual definition of the entire training curriculum. In contrast, our method only requires the manual definition of an exploratory bottleneck.

Hindsight Experience Replay (HER) is a popular example of an *Implicit* curriculum learning method used in robotics [31]. HER assumes that easier-to-reach states are encountered first and can be used to gain knowledge from unsuccessful episodes. To accomplish this, HER reassigns the desired goal of each

episode to be one of the goals that were achieved during that episode. This allows the agent to learn from its unsuccessful experiences and as much as from the successful ones.

In our approach, we leverage an HER replay buffer to improve the agent's learning, but we do not explicitly generate a curriculum. Instead, a natural curriculum emerges from the recursive formulation of each subtask. As such, according to the classification introduced in [27], our proposed method falls under the *Implicit* CL category, where a CL technique is employed without explicitly defining a curriculum.

Our work is related to the Policy Distillation method [32], which allows the transfer of knowledge from multiple experts to a single agent. This technique can also compress the agent's capabilities while maintaining its performance. In our approach, we employ a form of Policy Distillation by transferring the combined knowledge of all previous subtasks to the next policy. This results in a new distilled policy that can solve all previous subtasks and any new ones that are added to extend the sequence. Additionally, since we always retrain the agent on the entire sequence of subtasks whenever a new one is added, we avoid the issue of catastrophic forgetting [33].

Transferring knowledge between policies necessitates employing a Transfer Learning (TL) strategy [34]. In our approach, we use a combination of Q-function [35] and replay buffer transfer. To transfer the replay buffer, we preload it with the transitions collected while executing the previous sub-task actor. Additionally, we achieve Q-function transfer by employing the previous sub-task critic to guide the currently learning actor.

III. Methods

A. Background

RL problems are formulated as Markov Decision Processes (MDP), defined by a tuple $\mathcal{M} = (\mathbb{S}, \mathbb{A}, P, r, \gamma)$. In this formulation, $\mathbb{S}$ is the state space, $\mathbb{A}$ is the action space, P is the transition probability, r is the immediate reward function, and γ is the discount factor. TD3 is an off-policy RL algorithm that learns a deterministic policy by back-propagating the gradient signal through the critic network to the actor network directly [36]. The algorithm contains two neural networks: Actor and Critic, that are trained iteratively.

The critic network is used to approximate the Q-function by learning a mapping $\mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$, parameterized by ϕ. The Q-function provides an agnostic measure of the state s_t quality when performing an action a_t from that state. The measure is expressed as the expected future reward, starting from that state and operating under policy π, and defined as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right] \quad (1)$$

In (1), s_t is the current state, and a_t is an action that is taken from s_t and $\mathbb{E}_{\pi}$ is the expected weighted total reward, obtained using policy π, starting from the state s_t and taking the first action a_t. The loss of the critic is defined using temporal-difference approximation, minimizing the difference between Q-value at timestep t and the sum of Q-value at s_{t+1} and instantaneous reward r_t with respect to ϕ:

$$L_Q(\phi, \mathcal{D}) = \underset{(s_t, a_t, r, s_{t+1}, d) \sim \mathcal{D}}{\mathbb{E}} \left[(Q_\phi(s_t, a_t) - T)^2\right] \quad (2)$$

Where the Target Q-value approximation (T) is defined as:

$$T = (r + \gamma(1-d) Q_\phi(s_{t+1}, \pi_\theta(s_{t+1})))$$

In (2), a replay buffer $\mathcal{D}$ is used to sample experiences in tuples that contain a current state representation s_t, an action a_t, an instantaneous reward r, the next state representation s_{t+1} and an episode termination indicator d. The actor-network represents a learnt policy π_θ that performs a mapping $\mathbb{S} \rightarrow \mathbb{A}$, parameterized by θ. An optimal policy π^* provides actions that maximize the total episode reward R. Therefore, the loss function of the actor-network is set to maximize the Q-value with respect to θ by encouraging the actor-network $\pi_\theta(s)$ to produce actions, that lead the agent to states with higher Q-values, such as:

$$L_\pi(\theta, \mathcal{D}) = \underset{s \sim \mathcal{D}}{\mathbb{E}} [(Q_\phi(s, \pi_\theta(s)))] \quad (3)$$

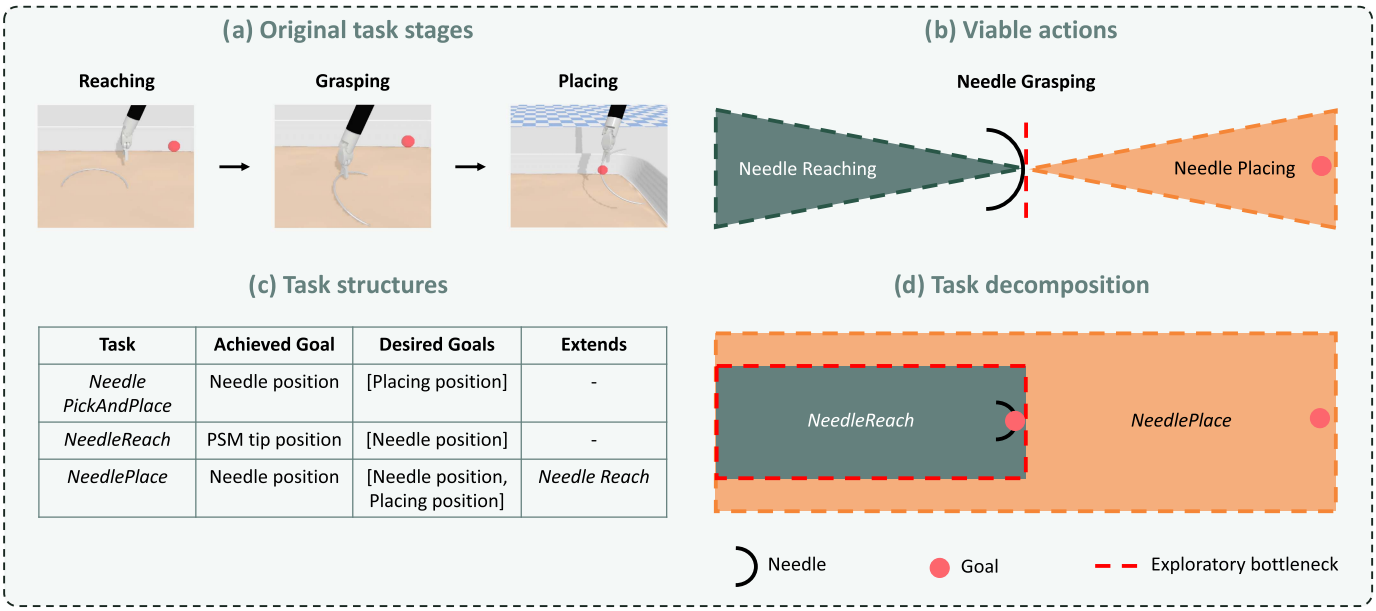
B. Task Decomposition

In this work, we focus on solving a complex multi-stage manipulation task that consists of picking and placing a small needle to a designated location and which we will call from now on for simplicity *NeedlePickAndPlace*.

The *NeedlePickAndPlace* task cannot be learnt directly via random exploration, as the trajectory goal is located beyond an exploratory bottleneck. This bottleneck is located at the instance of needle grasping and is caused because only a very narrow set of actions can be performed in order to cross it. Furthermore, state representations just before and after the grasping instance are extremely similar, making it difficult for an agent to reason about the grasping status.

In our method, we introduce an intermediate goal that allows learning the overall policy by implicitly increasing the complexity of the curriculum. We first split the *NeedlePickAndPlace* task into 3 distinct manipulation phases: *Reaching*, *Grasping*, and *Placing*. In this formulation, the second task contains the exploratory bottleneck, however, it also requires the shortest manipulation trajectory to complete. We then define a two sub-task system, defining a two-level curriculum, each containing a trajectory goal. The first sub-task requires an agent to reach the needle grasping point and has a goal defined at the needle grasping point. The second sub-task requires the completion of all three stages and has a goal defined at the expected placing position. We do not explicitly learn the grasping phase, since we use it to isolate the bottleneck in the shortest possible manipulation trajectory segment. Nevertheless, grasping is learnt implicitly when learning to complete the manipulation task.

Intuitively, both tasks are strongly related, and manipulation trajectories that are performed in the first sub-task provide a partial completion of the second sub-task. Therefore, such decomposition induces an implicit curriculum structure, allowing to learn the shorter manipulation task before learning the longer one. Our method is presented in Fig. 2.



Task	Achieved Goal	Desired Goals	Extends
Needle PickAndPlace	Needle position	[Placing position]	-
NeedleReach	PSM tip position	[Needle position]	-
NeedlePlace	Needle position	[Needle position, Placing position]	*Needle Reach*

Fig. 2. Semantic decomposition of *NeedlePickAndPlace* task: (a) The overall task is divided into three phases: *Reaching*, *Grasping*, and *Placing*; (b) Grasping stage is the exploratory bottleneck, therefore the set of actions that allow to cross it gets reduced the closer an agent gets to the bottleneck; (c) The complete task is refactored into *NeedleReach* and *NeedlePlace* sub-tasks, where the latter contains a partial expected trajectory of the former; (d) Summary of the task definition structure.

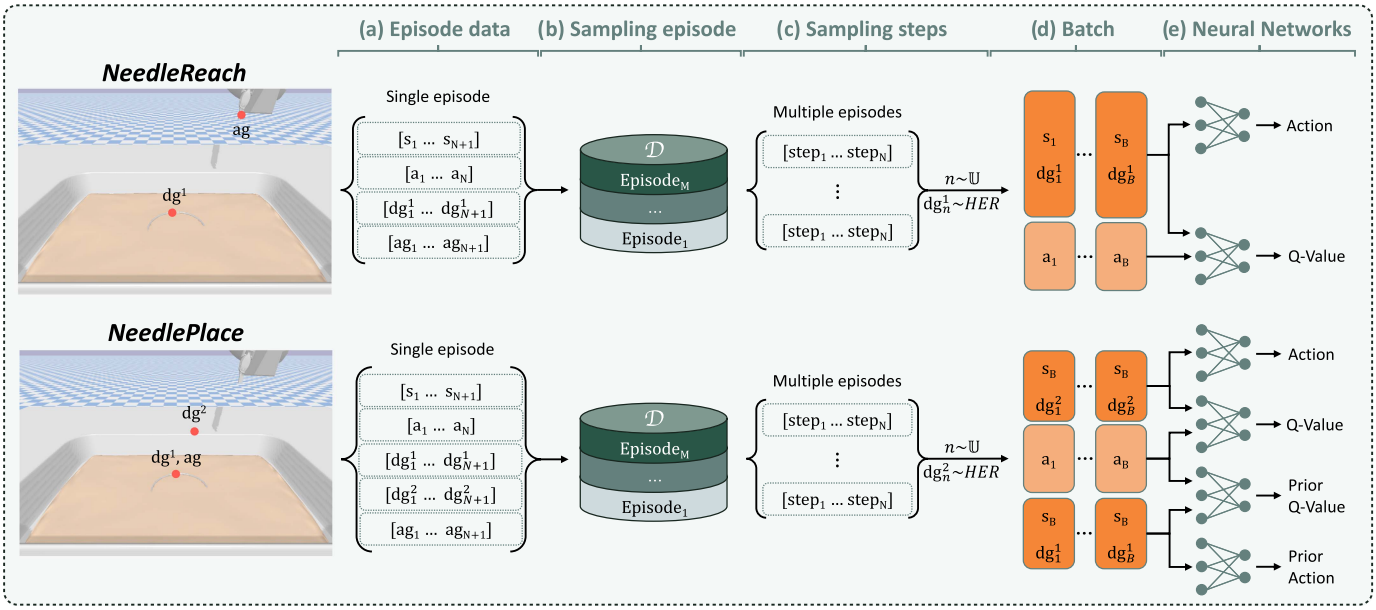


Fig. 3. Complete model pipeline, describing the data processing operations. The data, containing state representations (s), actions (a), desired goals 1 (dg^1), desired goals 2 (dg^2), and achieved goals (ag) are collected from the simulator one episode at a time and stored in a replay buffer $\mathcal{D}$. A number of episodes, equal to batch size B, is sampled from the buffer and a single step is sampled from each episode, uniformly. Hindsight Experience Replay (HER) sampling scheme is used to re-sample the desired goals of the selected samples, forming a batch of data. *NeedleReach* models employ the inputs conditioned on desired goal 1, whereas *NeedlePlace* models employ inputs, conditioned on the desired goal 2.

C. State and Reward Spaces

Our method employs a composite state space representation. We condition the task-invariant scene observation s_t with a task-dependent desired goal dg^n, that is to be achieved in the n^{th} task. Such a scheme allows for an easy state representation generation, that provides intra-task compatibility support when using our method. Furthermore, the state representation is normalized before feeding the data to the neural network. We use a custom input normalization layer, that collects and updates running mean and standard deviation measures during training. These measures are then used during the inference passes.

Furthermore, both sub-tasks are trained using sparse reward schemes. Any action results in a reward of -1, and only a transition that leads to a successful episode obtains 0 reward. The success conditions are explained in section III-E.

D. Learning the Policy

The policy for *NeedlePickAndPlace* task is learnt in two training stages, utilizing both sub-task as a curriculum with increasing difficulty. During the first stage, we train the agent from scratch using a vanilla TD3 + HER algorithm for the *NeedleReach* task, with the goal defined at the needle grasping point. The agent is able to learn such policy via random exploration due to goal resampling, facilitated by the use of HER replay buffer.

During the second stage, we are training the agent to complete the full *NeedlePickAndPlace* with the goal defined in the placing location. We further use the agent trained for the first sub-task to leverage a useful prior, allowing to overcome the exploratory bottleneck. In order to achieve a successful transfer, the prior agent must have converged. Using an agent that has not converged to successfully complete the *NeedleReach* task will lead to complete failure in learning *NeedlePickAndPlace* task. We determined empirically the convergence point of the prior to happen at 500 k timesteps, achieving 94% success rate during evaluation, as illustrated in Fig. 4.

Before the training starts, we preload a newly initialized HER buffer with partially completed trajectories, provided by the *NeedleReach* agent. This ensures that the actor and critic networks of the second sub-task are immediately exposed to trajectories that they are expected to perform. Further, pre-trained actor occasionally grasps the needle and moves it to a different location due to uncertainty of the correct actions after reaching is completed and added exploratory noise. Such trajectories provide a direct learning signal after HER goal resampling is applied.

Furthermore, we redefine the *NeedlePlace* actor loss formulation, allowing to exploit the *NeedleReach* critic knowledge when learning *NeedlePlace* task. This is accomplished by obtaining a compound loss for the *NeedlePlace* actor, generated by both *NeedlePlace* and *NeedleReach* critics. Therefore, we update the actor loss to maximize the combination of both critics with respect to θ, as:

$$L_\pi(\epsilon, \theta, \mathcal{D}) = \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[(1-\epsilon)\, C_2 + \epsilon\, C_1 \right] \tag{4}$$

$$C_1 = Q^n_{\phi'}\left(s^n, \pi_\theta(s)\right)$$

$$C_2 = Q^{n+1}_{\phi}\left(s^{n+1}, \pi_\theta(s)\right)$$

In (4), ϵ is a linearly decaying parameter, and it decays from 1 to 0 as the training of the second stage progresses. Values s^n and s^{n+1} are state representation for the *NeedleReach* and *NeedlePlace* task, respectively. Lastly, C1 and C2 represent outputs of the critics of each task, parameterized by ϕ and ϕ'. The pseudocode of the algorithm is presented in Algorithm 1.

Algorithm 1: Training Policy for Sub-Task $n+1$ Via Prior n.

Input: Trained prior actor A^n and critic Q^n, initialised environment Env

Output: Initialise HER buffer $\mathcal{D}$, current agent actor A^{n+1}_θ, critic Q^{n+1}_ϕ and target critic Q^{n+1}_{target}

While Preloading $\mathcal{D}$ **do**
  $episode \leftarrow Env, A^n(s^n)$
  Append $episode$ to $\mathcal{D}$
end while
while training **do**
  $episode \leftarrow Env, A^{n+1}(s^{n+1})$
  Append $episode$ to $\mathcal{D}$
  $batch \sim \mathcal{D}$
  $r \leftarrow Env.get\_rewards(batch)$
  $d \leftarrow Env.get\_dones(batch)$
  $T \leftarrow r + \gamma(1-d)Q^{n+1}_{target}(s^{n+1}_{t+1}, A^{n+1}(s^{n+1}_{t+1}))$
  $L_Q(\phi, batch) = \mathop{\mathbb{E}}_{batch}[(Q^{n+1}_\phi(s^{n+1}_t, a) - T)^2]$
  $Q^{n+1}_\phi \leftarrow Q^{n+1}_\phi + \nabla_\phi L_Q$
  if policy update step **then**
    $C_1 \leftarrow Q^n(s^n_t, A_\theta(s^{n+1}_t))$
    $C_2 \leftarrow Q^{n+1}_\phi(s^{n+1}_t, A_\theta(s^{n+1}_t))$
    $L_A(\theta, batch, \epsilon) = \mathop{\mathbb{E}}_{batch}[(1-\epsilon)C_2 + \epsilon C_1]$
    $A^{n+1}_\theta \leftarrow A^{n+1}_\theta - \nabla_\theta L_A$
  end if
  $Q^{n+1}_{target} = (1-\gamma)Q^{n+1}_{target} + \gamma Q^{n+1}_\phi$
end while

E. Environment Setting

We perform the training of both sub-task in a modified SurRoL training environment [10]. SurRoL employs a physical interaction and friction-based grasping approach called "interact", reducing the reality gap between the simulation and the real world. Both sub-tasks contain two objects of importance: a PSM and a needle that is placed on a surgical tray. The PSM is initialized so that the End-Effector tip pose is $[(2.5, 0.25, 3.6), (0, \frac{pi}{2}, \pi)]$ when an episode starts. The needle is randomly sampled on the tray with a deviation of 5 cm from the tray centre in parallel to the tray surface. Furthermore, its yaw angle is randomly set in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

We further define the state representation using the data provided by the environment. The simulated environment returns a raw state representation, containing three contextual objects: a task-invariant scene observation, an achieved goal, and a list of desired goals. The scene observation contains robot and needle pose information, ensuring a fully-observable domain and is shared between both sub-tasks.

For the *NeedleReach* task, the achieved goal is defined to be the end-effector tip position and the desired goal contains the needle body central point position. To complete the episode successfully, the L1 norm between the achieved goal and desired goal has to be below 4 mm. The task has a maximum episode length of 50 steps. For the *NeedlePlace* task, the achieved goal is defined to be the needle body central point position and the

desired goal list contains two goals: the needle body central point position and a randomly sampled position in the operational space, where the object has to be placed. To complete the episode successfully, the L1 norm between the achieved goal and the second desired goal has to be below 2.5 cm. The task has a maximum episode length of 100 steps. Both settings are presented in Fig. 3.

F. Control Setting

We utilize a model of the Da Vinci Research Kit (dVRK) surgical robotic system [14] for minimally invasive laparoscopy. We employ a single PSM arm, containing 7 degrees of freedom. The dVRK system comes with a quality kinematics framework included through the CIST library, therefore allowing for effective control through high-level control inputs.

We utilize delta-pose method to control the manipulator. Delta pose method refers to a positional control approach, where spatial change values of the End-Effector Cartesian position and orientation are provided by the agent. In our setting, these values are bounded in the range of $[-0.5, 0.5]$. The first three control values represented delta End-Effector (EE) position in the world coordinate frame, allowing for the gripper translations in the operational space. Furthermore, a single delta-yaw value is used to achieve a gripper tip rotation, which allows fitting the curvature of the manipulated object. Lastly, a single-valued gripper action is used, where a value ≥ 0.0 represents an open gripper state.

IV. Training and Results

A. Comparison With the Baseline

We compared our agent's performance against four baseline methods: TD3 with HER replay buffer (HER), TD3 with HER replay buffer and Behavioral Cloning (BC) loss (HER + BC), TD3 with HER replay buffer that was preloaded with expert demonstrations (HER+DEMO) and TD3 with HER replay buffer that was preloaded with expert demonstrations and Behavioral Cloning (BC) loss (HER + DEMO + BC). The first two baselines showed that pure exploration is not enough to learn *NeedlePickAndPlace* task from scratch, even with BC loss being applied directly to the actor. Whereas the latter two both employ expert demonstrations and are the current state-of-art solution for learning end-to-end policy for *NeedlePickAndPlace* task, as demonstrated by [10]. Our method required also obtaining a prior agent, that was trained for 500 k time steps, preloading 10 k time steps of experience collected via random exploration.

We trained all five models for 2 million time steps whilst maintaining the hyperparameters as similar to each other as possible in order to allow for a reasonable comparison. All five instances were preloaded with 50 k time steps of transitions, coming from the random exploration, path planner, and prior actor for HER and HER+BC, HER+DEMO and HER+DEMO+BC, and our methods respectively. Then each agent was trained for the remaining 1.95 million time steps, using ADAM optimiser with a learning rate of $1e^{-4}$ and batch size of 2048 samples. For BC

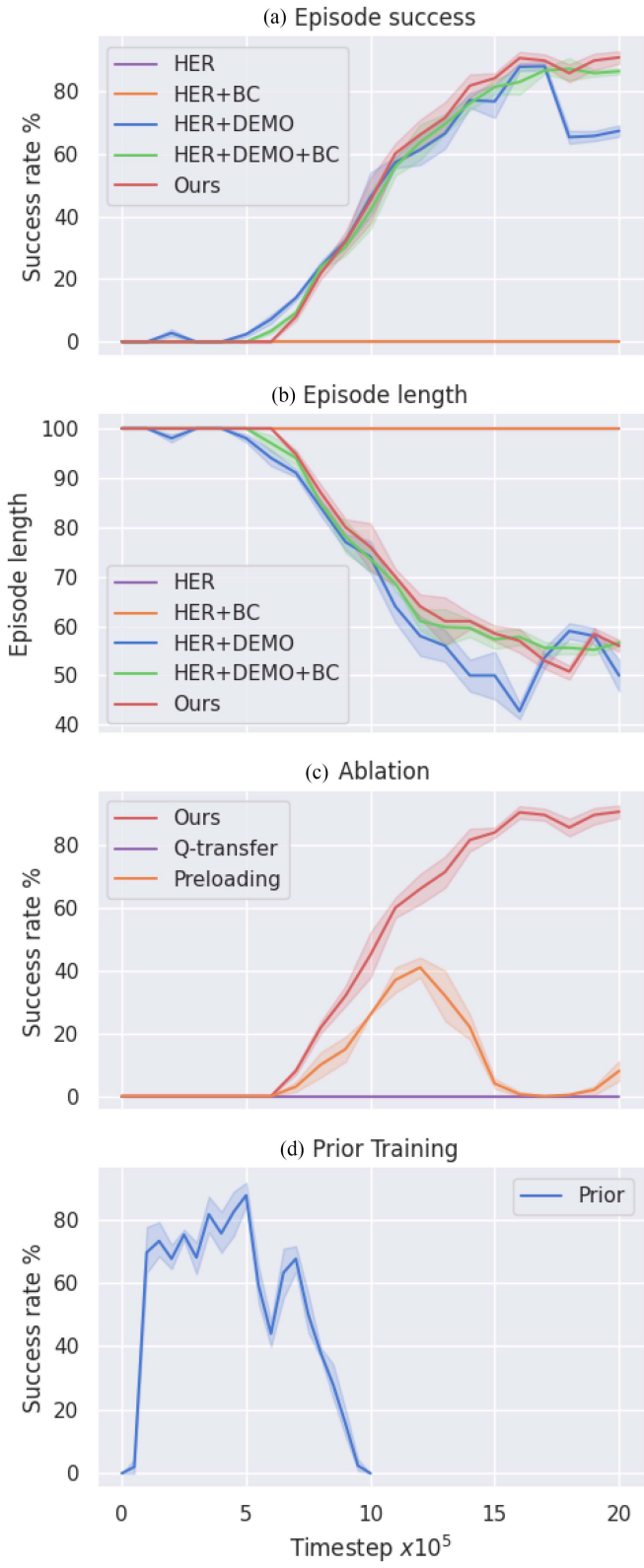


Fig. 4. (a) Success rate of our method, versus baselines; (b) Episode length of our method, versus baseline; (c) Ablation study: the success rate of our complete method, versus only buffer preloading with partially correct trajectories, versus only critic transfer; (d) Prior agent training progress. The bold line represents the average performance values, whereas the shaded boundary demonstrates the 95% confidence interval.

methods, we implemented a separate replay buffer, that was preloaded with the same expert demons as the main buffer. During each policy training step, an expert demonstration state-action pairs (s^*, a^*) were sampled from the buffer and the actor training signal was obtained by combining the critic loss as well as a BC loss, defined as $||a^* - \pi(s^*)||_2$.

Our policy achieved a peak average success rate of 91.1% across five random seeds. Therefore, it outperformed HER+DEMO and HER+DEMO+BC approaches, which achieved 87.4% and 88.2% success rate over the same five seeds. The HER and HER+BC agents did not manage to learn the policy, thus achieving the success rate of 0%. However, HER+DEMO method outperformed our agent in the average episode length comparison, completing an episode using 42.26 steps on average. Our agent performed significantly slower, requiring 50 steps on average to complete an episode. Nevertheless, our method reflected a much more stable episode length convergence. We evaluated each agent after every 100 k time steps of training to provide a better insight into the training performance. The results are presented in Fig. 4, parts (a) and (b). It can be observed that whilst it takes longer to train our agent, the peak performance is higher at the end of the training.

B. Ablation Study

We further investigated the importance of both method components: buffer preloading with *NeedleReach* actor and *NeedlePlace* actor guidance using *NeedleReach* critic. It was determined that only a combination of both components allows an agent to learn the desired policy. Buffer preloading alone allows the agent to learn a suboptimal policy, peaking at a 42% success rate. We noticed that the agent learns to reach the object quite quickly. However, the policy starts diverging shortly after, due to many unstable grasps being loaded to the buffer, thus making policy assign equal rewards for (unstably) grasped needle states and random states. On the other hand, actor guidance does not manage to learn a successful policy in isolation. By visually inspecting the training process, we observed that the actor is being guided in the right direction. However, the small scale of the object and its physically rich interaction nature prevents the agent from grasping the needle successfully, thus preventing the agent from obtaining any reward. The ablated performance is presented in Fig. 4, part (c).

V. Conclusion

In this letter, we present a novel learning approach, that allows learning a multi-stage *NeedlePickAndPlace* manipulation task with sparse rewards without using explicit export demonstrations. Our agent successfully outperformed a HER+DEMO trained agent, achieving a state-of-the-art performance of 91% success rate.

Whilst our method was not tested in a real-world setting, previous work has shown that such transfer should be possible [5], [10]. SurRoL platform interface is directly compatible with dVRK control inputs, as explained in [10] by following the transfer protocol defined in [37]. It was shown that object of interest based state-space representation can be transferred by placing objects of interest in predetermined poses in World Cartesian coordinates. On the other hand, visual tracking methods have also been successful and used in determining accurate tool poses in the World Coordinate Frame [5], [38]. Such methods can be applied to estimate the poses of the objects of interest, allowing to transfer our policy directly to the real robot. Nevertheless, there is a performance drop to be expected between the simulation and the real world due to the noise induced by a state estimator [5].

Our method allows learning end-to-end control policy without expert demonstrated trajectories. Nevertheless, it still requires performing a manual task decomposition by a human entity. Furthermore, the person has to be familiar with the mechanics of the problem, therefore, they might be referred to as an expert. Nevertheless, the expertise required to break down the task is significantly reduced, as it only requires a strong understanding of the manipulation process, rather than specific domain knowledge and surgical dexterity. Furthermore, such decomposition raises a one-off cost, that is temporarily much cheaper than performing a number of trajectories manually.

Further, adaptations of the method can potentially cause negative TL issues. Our problem setting meets the basic assumptions of TL: 1) domain similarity, 2) data sampled from both tasks comes from the same distribution, and 3) an appropriate model can be fit to both domains [39]. This is further experimentally confirmed by a successive transfer. However, changing the problem setting might break the set of assumptions, making the method unfeasible for such tasks.

Lastly, our method is also less sample efficient compared to expert demonstration-based approaches, which means that a higher number of trajectories needs to be collected compared to HER+DEMO approach. Our method allows for increasing the sample efficiency by saving the replay buffer after each stage and loading it directly for the subsequent training stages since all the goals are saved simultaneously. Nevertheless, we found that a freshly initialized buffer for each stage provides less noisy data and thus more stable training.

References

[1] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Efficient adaptation for end-to-end vision-based robotic manipulation," Jun. 2020. [Online]. Available: https://openreview.net/forum?id=CVN5cZBFFFG

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[3] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3019–3026, Apr. 2020.

[4] J. Hare, "Dealing with sparse rewards in reinforcement learning," 2019, *arXiv:1910.09281*.

[5] Z.-Y. Chiu, F. Richter, E. K. Funk, R. K. Orosco, and M. C. Yip, "Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7737–7743.

[6] A. Wilcox et al., "Learning to localize, grasp, and hand over unmodified surgical needles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 9637–9643.

[7] S. A. Pedram, C. Shin, P. W. Ferguson, J. Ma, E. P. Dutson, and J. Rosen, "Autonomous suturing framework and quantification using a cable-driven surgical robot," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 404–417, Apr. 2021.

[8] V. M. Varier, D. K. Rajamani, N. Goldfarb, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots," in *Proc. 29th IEEE Int. Conf. Robot Hum. Interactive Commun.*, 2020, pp. 1380–1386.

[9] M. Stolle and D. Precup, "Learning options in reinforcement learning," in *Proc. 5th Int. Symp. Abstraction, Reformulation Approximation*. 2002, pp. 212–223.

[10] J. Xu, B. Li, B. Lu, Y.-H. Liu, Q. Dou, and P.-A. Heng, "SurRoL: An open-source reinforcement learning centered and dVRK compatible platform for surgical robot learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1821–1828.

[11] T. Huang, K. Chen, B. Li, Y.-H. Liu, and Q. Dou, "Guided reinforcement learning with efficient exploration for task automation of surgical robot," 2023, *arXiv:2302.09772*.

[12] C. D'Ettorre et al., "Automated pick-up of suturing needles for robotic surgical assistance," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1370–1377.

[13] F. L. D. Silva and A. H. R. Costa, "Object-oriented curriculum generation for reinforcement learning," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst., ser. AAMAS '18. Richland, SC: Int. Found. Auton. Agents Multiagent Syst.*, 2018, pp. 1026–1034.

[14] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da Vinci Surgical System," in *Proc. IEEE Int. Conf. Robot. Automat*. 2014, pp. 6434–6439. [Online]. Available: https://ieeexplore.ieee.org/document/6907809/

[15] C. D'Ettorre et al., "Accelerating surgical robotics research: A review of 10 years with the da Vinci research kit," *IEEE Robot. Automat. Mag.*, vol. 28, no. 4, pp. 56–78, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9531355/

[16] A. Attanasio, B. Scaglioni, E. De Momi, P. Fiorini, and P. Valdastri, "Autonomy in surgical robotics," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 4, pp. 651–679, 2021.

[17] K. L. Schwaner, D. Dall'Alba, P. T. Jensen, P. Fiorini, and T. R. Savarimuthu, "Autonomous needle manipulation for robotic surgical suturing based on skills learned from demonstration," in *Proc. IEEE 17th Int. Conf. Automat. Sci. Eng.*, 2021, pp. 235–241.

[18] S. A. Pedram, P. Ferguson, J. Ma, E. Dutson, and J. Rosen, "Autonomous suturing via surgical robot: An algorithm for optimal selection of needle diameter, shape, and path," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 2391–2398.

[19] F. Zhong, Y. Wang, Z. Wang, and Y.-H. Liu, "Dual-arm robotic needle insertion with active tissue deformation for autonomous suturing," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2669–2676, Jul. 2019.

[20] M. Hwang et al., "Automating surgical peg transfer: Calibration with deep learning can exceed speed, accuracy, and consistency of humans," *IEEE Trans. Automat. Sci. Eng.*, vol. 20, no. 2, pp. 909–922, Apr. 2023.

[21] B. Thananjeyan et al., "Optimizing robot-assisted surgery suture plans to avoid joint limits and singularities," in *Proc. IEEE Int. Symp. Med. Robot.*, 2019, pp. 1–7.

[22] A. Avinash, A. E. Abdelaal, P. Mathur, and S. E. Salcudean, "A "pickup" stereoscopic camera with visual-motor aligned control for the da Vinci surgical system: A preliminary study," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 7, pp. 1197–1206, Jul. 2019. [Online]. Available: http://link.springer.com/10.1007/s11548-019-01955-9

[23] M. Yip and N. Das, "Robot autonomy for surgery," in *Proc. Encyclopedia Med. Robot.*, 2018, pp. 281–313.

[24] S. Lu, T. Shkurti, and M. C. Çavuşoğlu, "Dual-arm needle manipulation with the da Vinci Surgical Robot," in *Proc. IEEE Int. Symp. Med. Robot.*, 2020, pp. 43–49.

[25] S. Pateria, B. Subagdja, A.-H. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Comput. Surv.*, vol. 54, pp. 1–35, 2021.

[26] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48, doi: 10.1145/1553374.1553380.

[27] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *Int. J. Comput. Vis.*, vol. 130, no. 6, pp. 1526–1565, Apr. 2022, doi: 10.1007/s11263-022-01611-x.

[28] S. Luo, H. Kasaei, and L. Schomaker, "Accelerating reinforcement learning for reaching using continuous curriculum learning," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2020, pp. 1–8.

[29] M. Kerzel, H. B. Mohammadi, M. A. Zamani, and S. Wermter, "Accelerating deep continuous reinforcement learning through task simplification," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2018, pp. 1–6.

[30] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proc. Conf. Robot Learn.*, 2022, pp. 91–100.

[31] M. Andrychowicz et al., "Hindsight experience replay," *Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., vol. 30, 2017.

[32] A. Rusu et al., "Policy distillation," 2015, *arXiv:1511.06295*.

[33] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," 2013.

[34] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Dec. 2009.

[35] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," in *Proc. 18th Eur. Conf. Mach. Learn.*, 2007, pp. 188–200.

[36] T. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016.

[37] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 561–591 . [Online]. Available: https://proceedings.mlr.press/v87/mahmood18a.html

[38] J. Lu, A. Jayakumari, F. Richter, Y. Li, and M. C. Yip, "SuPer deep: A surgical perception framework for robotic tissue manipulation using deep learning for feature extraction," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4783–4789 .

[39] W. Zhang, L. Deng, L. Zhang, and D. Wu, "A survey on negative transfer," *IEEE/CAA J. Automatica Sinica*, 2022.