Finding the Optimal Haptic Assistance Policy for the Learning of a Motor Task using Deep Q-Learning

Zonghe Chua and Julia Di

***Abstract*— Haptic guidance and error amplification are two opposing paradigms for training users in Robot-assisted Minimally Invasive Surgery. Previous work has only focused on training users under a single paradigm with fixed gains on the amount of assistance or disturbance. Challenge point theory suggests that there is an optimal difficulty under which learning will happen most effectively. Human learning under each paradigm is likely to be non-linear and stochastic, hence we opt to use a model-free reinforcement algorithm, Deep Q-learning, to learn a training policy to maximize a user's task performance during evaluation. Because model-free methods are known to be data inefficient, we evaluate how many users or examples are required to learn on before training a policy that is better or close to a hand-designed benchmark policy. We found that the agent only requires a total of about 40 users to explore on before seeing consistent results that are close to the benchmark while lowering the amount of discomfort and effort required during training. We also evaluate the effects of each term in a negative reward function on the policy. We found that penalizing time to complete a trial leads to better mean performance, but that the other penalties lead to less effort required during training.**

I. INTRODUCTION

Haptic guidance describes the technique of using haptic feedback to guide the user of a device towards an optimal path or trajectory. However too much guidance could lead to an undesirable effect known as the "guidance hypothesis", which states that feedback that is too frequent and accessible lead to reduced learning. The opposite of haptic guidance is known as haptic error amplification. This type of feedback destabilizes a user and increases the task difficulty.

The use of haptic guidance and error amplification has been studied in the domain of robot-assisted minimally invasive surgery (RMIS) for skills training [1][2]. However, these experiments only trained users in a single paradigm (guidance or error amplification) at fixed force gains. Thus, the amount of guidance or disturbance rendered is not optimized for the ability of the user. For example, guidance initially would aid a naive user in learning the gross desired path or movement, and disturbance would help the user fine-tune their ability when they are at an intermediate level of skill. This concept is known as Challenge Point Theory and was investigated for haptic guidance in for RMIS [3]. The authors of this study developed a hand-designed algorithm for calculating the user's ability and adjusting the task difficulty by reducing the gain on the haptic guidance force. The effects of their algorithm were small and users who had assistance did not differ largely from those that had none.

It is unclear whether users respond to haptic guidance or disturbance in an intuitive way that lends itself to hand-designed algorithms. Reinforcement learning is a method of finding the optimal action for a given state of the system from data and exploration. In haptics, this has been explored in a hand-held RMIS training device that utilizes skin-shear to guide users through a desired path [4]. The authors used a variation of model-based reinforcement learning called Gaussian Process Regression Modeling to learn the state transitions and consequently used Monte Carlo Tree Search with Model Predictive Control online to compute the optimal action online.

We hypothesize that the transition dynamics of user skill are non-linear and complex when training under guidance or disturbance to learn a task. Model-free reinforcement learning algorithms, such as Deep Q-learning, excel in learning these representations but are not data-efficient. This lack of data-efficiency is a serious drawback for physical tasks that require human interaction. This work aims to investigate how effective a model-free reinforcement learning technique, specifically Deep Q-learning, will perform given a limited training set. We compare how performance scales with the number of training episodes and benchmark them against a hand-designed naive policy.

II. METHODOLOGY

A. Description of Trajectory Following Task

The trajectory following task used in this paper is modeled similarly to that in [2]. In this case we assume that a given user trains under conditions of haptic guidance or disturbance for a predefined number of training trials. They are then evaluated without haptics. This measures the current

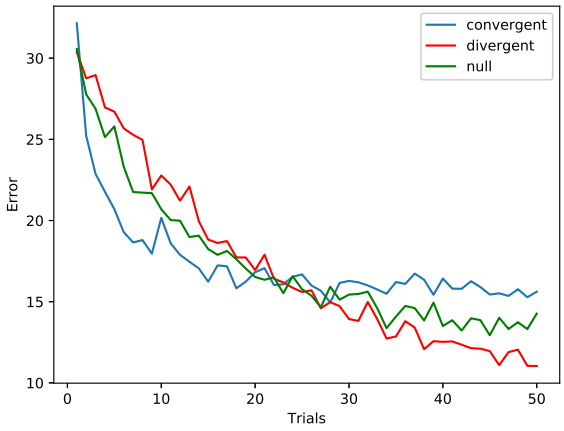

Fig. 1. Sample plot of errors during evaluation for each haptic training condition

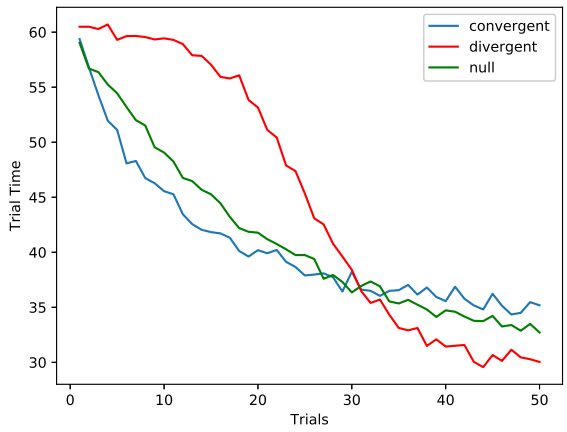

Fig. 2. Sample plot of trial times during evaluation for each haptic training condition

state of their skill. They then proceed to continue with another training session and consequence evaluation. This will continue until a number of cycles are completed. In this paper, the number of training and evaluation cycles were set to 50. Thus, all data points presented are "evaluation" data points.

B. MDP Model Description

The task is modeled as a Markov Decision Process (MDP) using the generative interface provided by POMDPs.jl.

1) State Space: The state of the MDP defines the user's skill level at a given evaluation cycle and is a continuous. It consists of 3 variables: (a) time taken to complete the task, (b) total path error, (c) the current trial number.

2) Action Space: The action space was discretized over a range of -15 to 15. These values represented the gains for the assistance or disturbance provided. Negative values represented convergent force fields that essentially pull a user into the desired path (guidance). Positive values represented divergent force fields that push a user away from the desired path (disturbance). A value of 0 represented a "null" field where there was no haptic guidance or disturbance rendered. This haptic paradigm can be described as:

$$F_{field} = a\,(||x_{actual} - x_{desired}||) \tag{1}$$

where $a < 0$ for convergent, $a > 0$ for divergent, and $a = 0$ for null.

3) Reward Function: The reward function consisted of negative rewards defined as

$$R = -time - error - 2|a| \tag{2}$$

where a is the action gain term from Eq. 1, such that a higher gain was penalized together with high error and time taken per trial. The action gain is multiplied by a factor of 2 to ensure that the magnitude of all three terms were relatively similar.

Additional reward functions were also evaluated because the reward function plays a major role in performance. We desired to investigate the role of each reward term n the system to better understand the relationship between reward and policy. We evaluated three reward functions that either penalized time strongly (Eq. 3), error strongly (Eq. 4), or action strongly (Eq. 5). The reward functions used were as follows

$$R = -(c \times time) - error - |a| \tag{3}$$

$$R = -time - (c \times error) - |a| \tag{4}$$

$$R = -time - error - |c \times a| \tag{5}$$

where c was set to 500 (an arbitrarily large number) to isolate the effects of each term in the reward function. The results are discussed in Sec. III.

C. Design of Black-box Generative Model

The generative model was constructed based on prior understanding of human performance under three different haptic conditions: convergent force fields, divergent force fields, and null. Under these conditions, we modeled both the time to complete a trial as well as the path error during each trial. These were the two parameters of interest because for a given task, we seek a policy that optimizes the trade-off between speed (trial time) and accuracy (path error).

For the convergent force field case, we modeled trial time using hyperbolic discounting (Eq. 8), and path error with hyperbolic-log discounting (Eq. 9). For the divergent force field case, we modeled trial time with a sigmoid function (Eq. 7), and path error with an exponential function (Eq. 6). For the null case, we modeled trial time as an exponential (Eq. 6), and path error with a hyperbolic discounting function (Eq. 8). These discounting functions were chosen so that the resultant black-box model emulates a known phenomena where a user has high upfront learning or rewards for the convergent haptic condition, but better long term rewards over a longer time horizon for the divergent haptic condition. We tuned these functions with an adjustable constant so that the curves were empirically plausible. A sample of the resultant learning curves generated for a policy that applies only a single type of field (convergent, divergent or null) at a fixed gain are shown in Fig. 1 for error and in Fig. 2 for trial time.

The equations for each type of discounting function or learning curves are

$$f(x) = (L+U)e^{c \times t_{eff}} \tag{6}$$

$$f(x) = \frac{L+U}{1+e^{c \times t_{eff}}} \tag{7}$$

$$f(x) = \frac{L+U}{1+c \times t_{eff}} \tag{8}$$

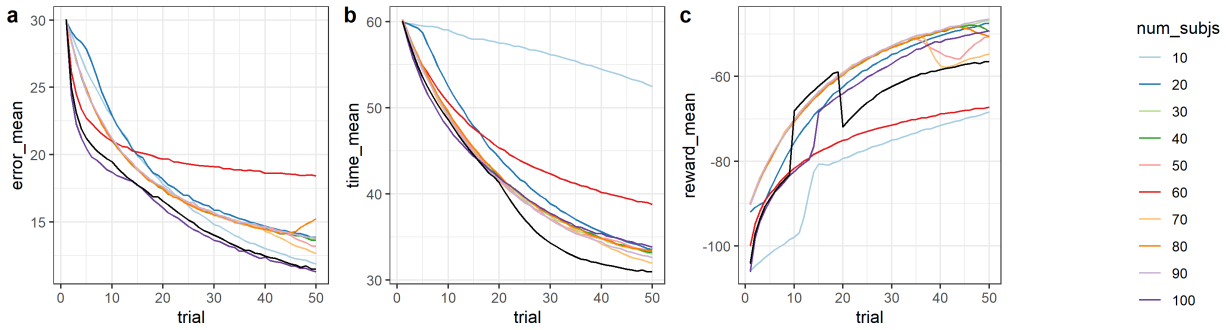

Fig. 3. Plots of (a) mean final evaluation trial error, (b) mean final evaluation trial time and (c) mean reward, for various numbers of subjects run (number of training examples to learn from). The black line indicates the benchmark policy.

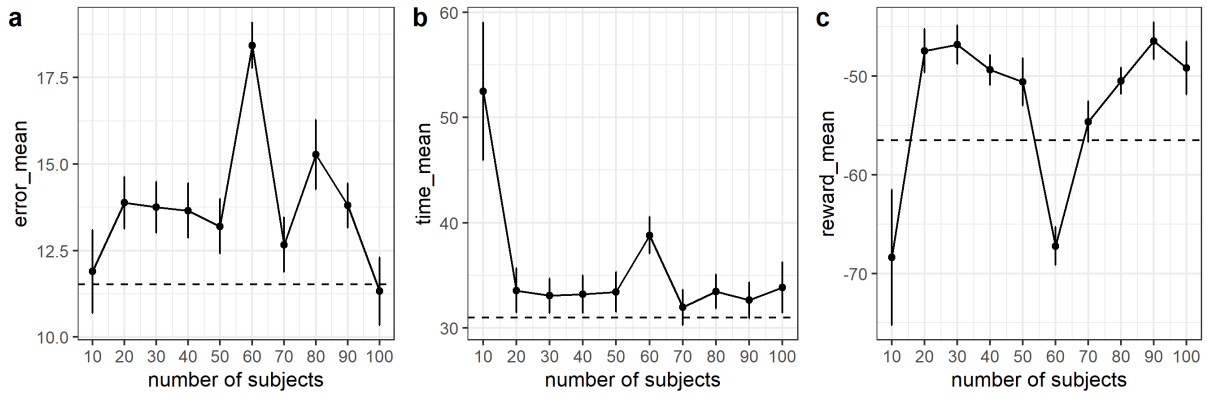

Fig. 4. Plots of (a) mean final evaluation trial error, (b) mean final evaluation trial time and (c) mean final reward, over various number of subjects run (number of training examples to learn from). The dotted line indicates the metric for the benchmark policy.

$$f(x) = \frac{L+U}{1+c\times\ln\left(1+t_{eff}\right)} \tag{9}$$

where t_{eff} refers to the effective trial, L refers to a baseline lower bound, U refers to a baseline upper bound, and c refers to a tuneable constant for each type of haptic condition. The effective trial, t_{eff}, is used to allow for transitions between haptic conditions, and is calculated by taking the current state and computing the inverse of the discounting function corresponding with the next action. The result is then used as effective starting point for computing the next state.

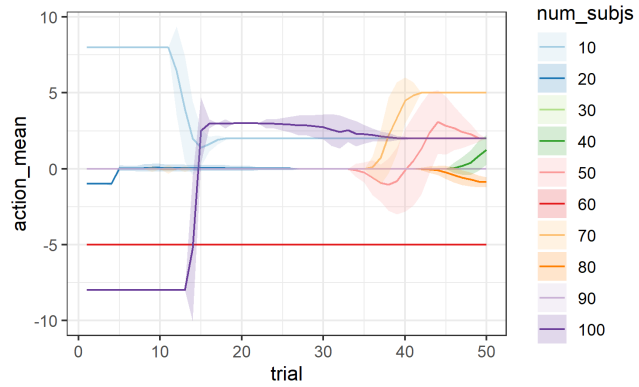

Fig. 5. Plot of mean actions over trials. Shaded ribbons denote standard deviations. Convergent actions are negative, and divergent actions are positive.

Gaussian process noise on both trial time and path error was injected into the system, and attenuated as trial number increases. The noise models the reduction in variability over repeated actions described in previous work by Sternad [5]. Similarly, we characterized both trial time and path error with normally distributed performance bounds (upper and lower bounds) that had a tuneable average and variance. These parameters are recorded in Table II in the Appendix.

D. RL Algorithm

The Julia POMDPs implementation of Deep Q-learning was used with a neural net consisting of a single affine layer of 100 neurons with a ReLU activation function. The Q-learning Solver utilized double Q-learning to prevent overestimation of the action values [6], a dueling network architecture for better policy evaluation of similar actions [7], and prioritized replay [8] to help learn more efficiently. The number of evaluations for each iteration of the Q-learning algorithm was set to two (100 evaluation instances). The exploration factor for an ε-greedy policy was set to 0.8 to boost initial exploration. A desirable outcome during learning of a motor task is to achieve a balance of a fast learning rate and also good long term gains. A discount factor of 0.95 was used to help the agent value the former consideration. Table I in the Appendix lists all the further hyperparameters used by the solver.

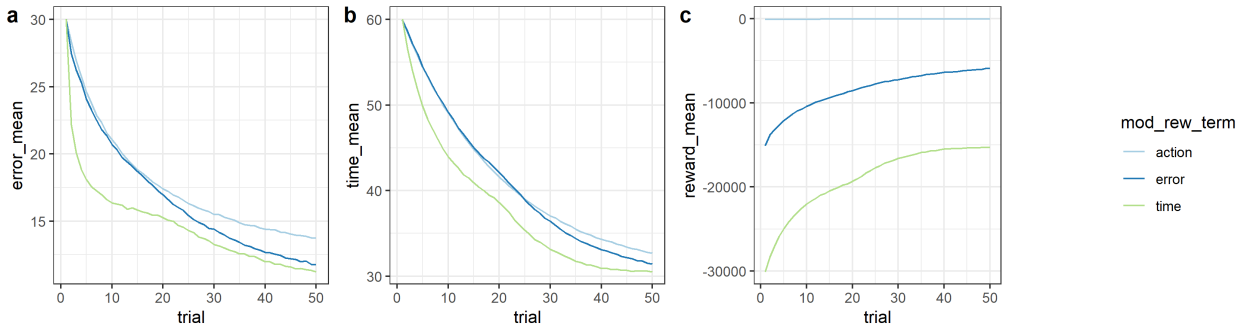

Fig. 6. Plots of (a) mean evaluation trial error, (b) mean evaluation trial time and (c) mean reward, for different weighting on the reward function.

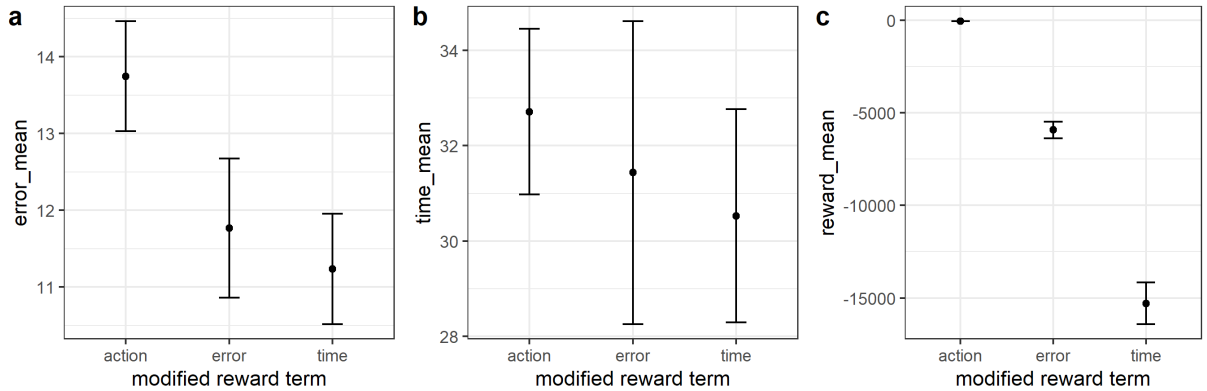

Fig. 7. Plots of (a) mean final evaluation trial error, (b) mean final evaluation trial time and (c) mean final reward, over different weightings on the reward function.

E. Benchmark Policy

The hand-designed policy was constructed in an intuitive manner and used as a benchmark for the Deep Q-learned policies. The hand-designed policy is as follows: a convergent force field with a gain of 7 was used for the first 10 trials to quickly learn the gross desired path, following which a null field was used for the next 10 trials. Once the user reached an intermediate level, a divergent field was used with a gain of 7. We did not opt to use the maximum gains of 15 so as to improve the reward of this policy, since higher gains were penalized by the action term.

F. Evaluating the Number of Required Training Examples

We evaluated how performance evolves with the number of training examples by training policies with an increasing number of training examples to learn from. All policies were then evaluated 100 times and the mean quantities of the reward, error and trial time computed. The performance of the policies was evaluated against the hand-designed policy.

III. RESULTS

A. Number of Required Training Examples

The mean error during evaluation decreased as the number of training examples increased with n = 100 achieving comparable performance to the benchmark policy. However there was some stochasticity in the learning of the policy as seen by how n = 10 achieved better performance than n = 60 (Fig. 3a). The mean time per trial during evaluation decreased with increasing n, however the improvement in performance is negligible after n = 30. The benchmark policy achieved better performance overall at the end of the 50 evaluation cycles (Fig. 3b). The learned policies had higher rewards compared to the benchmark policies mainly because it was also balancing achieving good performance with reducing the gains used to achieve that performance. This can be seen by how the learned policies regulated the gains towards zero (Fig. 5). In all cases, the rates of improvement over trials are quite similar to that of the benchmarks for both error and time. Fig. 4 shows how the number of subjects (or examples) the policy is trained on stabilizes after a training set of 20 subjects.

B. Variation of Performance due to Changes in the Reward Function

The reward function also plays a major role in learned policy performance, so three reward functions were evaluated as stated in Sec. II-C to investigate the role of individual reward parameters. The resultant policies were trained on n = 20 users because results of the previous section showed that the policy training stabilizes after 20 subjects in the training set.

Changes in the reward function led to clear changes in policy, as shown in Fig. 8. When the reward function heavily penalized trial time, the policy initially applied null, then applied a convergent field before switching to divergent field. This explains the performance metrics seen in Fig. 6, where mean trial time is at first drastically driven down

by the convergent field. Then when the user performance plateaus, the policy switches to divergent field to further drive down trial time. When the reward function heavily penalized path error as shown in Fig. 8, the policy becomes more conservative on accuracy and only applies convergence. As expected, penalizing error led to a trade-off in trial time, which is shown by the large deviation in Fig. 7b. However, this trade-off was not reciprocated in the reverse case where time was penalized, which may be due to the stochasticity in learning. When the reward function heavily penalized the action term, the policy took no actions (null) as expected.

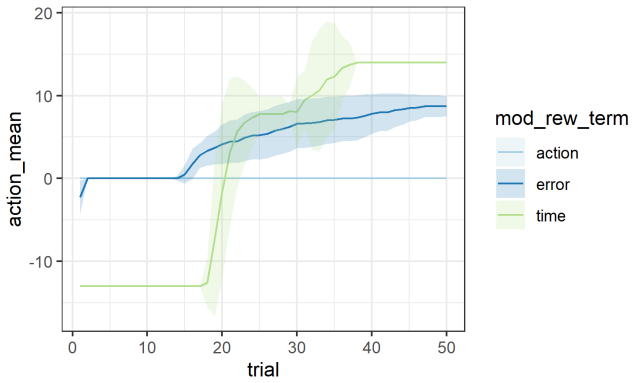

Fig. 8. Plot of mean actions over trials for different reward weightings. Shaded ribbons denote standard deviations. Convergent actions are denoted as negative, divergent actions are denoted as positive.

IV. DISCUSSION

The results indicate that the reinforcement learning algorithm only requires 20 users in the training set to before converging to a reasonable performance, with two users needed for policy evaluation after every two users that the Q-learning algorithm runs on. This results in a total of 40 users run needed before being able to reach the performance presented in the results. While typical haptics studies run about 20-30 users, 40 users is still within a reasonable scope in which to conduct an experiment to train a policy. While the performance is not as good as the benchmark policy, the magnitude of the forces used to achieve the performance for the policies, as seen by the reduced gains in Fig. 5 is lower. This would result in less actuator effort and possibly more comfort for the user.

The results also indicate that penalizing time to complete a trial leads to better mean performance, as seen in Fig. 6. Penalizing time appeared to result in both a lower mean trial time as well as lower mean path error value. However, closer look at the results shown in Fig. 8 show that the mean magnitude of the haptic guidance is also highest for the reward function that heavily penalizes time in comparison to the other reward functions tested. This means that the other penalties (error and action) help balance performance with actuator and user effort.

While the benchmark policies performed better than the learned policies, the generative model we used was fit to the rough trends seen in [3]. This meant that the true model of human learning could be more non-linear and stochastic than what we designed especially since we extrapolated the dynamics of the divergent force fields based on anecdotal observation and intuition from prior research studies. In this case, there is potential for the reinforcement learning algorithm to outperform the hand-designed benchmark algorithm. However the added complexity might also require more training examples to properly learn the dynamics of the system.

Another assumption that we made was that the abilities of the users do not deviate too much from each other. If the users that the policy is trained on differ quite a lot from each other, the policy could fail to converge to a good globally applicable policy.

V. CONCLUSIONS AND FUTURE WORK

In this work, we show that the number of users or training examples required to learn a Deep Q-learning policy that achieves comparable performance to a benchmark is reasonably within the scope of experiments in the haptics field. We also show that this policy achieves better energy efficiency and comfort than the benchmark, which is especially useful in a RMIS task, and that these benefits can be tuned with the error and action terms of the reward function at cost of raw performance.

It remains to be seen if the black-box model we designed truly reflects the dynamics of the actual task and captures the variability in human performance adequately. Higher true complexity could lead to better performance of the model-free algorithm compared to the benchmark with the possible downside of requiring more users to be presented to the agent for robust learning of a policy. Higher variability could reduce the generalizability of the learned policies and also require mode users to be presented to the agent to counteract.

We opted for a model-free method based on the assumption that the dynamics of the process were highly non-linear and stochastic such that learning a model for a model-based approach would prove to be unsuccessful. Future work would explore using a model-based reinforcement learning method and benchmarking this against the model-free method.

In this paper, only one deep reinforcement learning algorithm was implemented and evaluated. It would be useful to compare the performance of this algorithm to another, such as policy iteration, which would model the policy as a non-linear function of the state.

Lastly, we would want to verify our results experimentally through an actual user study which an experimental design informed by our findings.

VI. ACKNOWLEDGEMENTS

Zonghe Chua implemented the trial time transition dynamics of the black-box model. He also performed the analyses of how the number of subjects run on the reinforcement learning algorithm affected policy performance.

Julia Di implemented the path error transition dynamics of the black-box model. She also performed the analyses of how the each term in the reward affected the performance of the resultant policy.

REFERENCES

[1] M. M. Coad, A. M. Okamura, S. Wren, Y. Mintz, T. S. Lendvay, A. M. Jarc, and I. Nisky, "Training in divergent and convergent force fields during 6-dof teleoperation with a robot-assisted surgical system," in *2017 IEEE World Haptics Conference (WHC)*. IEEE, 2017, pp. 195–200.

[2] Y. A. Oquendo, Z. Chua, M. M. Coad, I. Nisky, A. Jarc, S. Wren, T. S. Lendvay, and A. M. Okamura, "Robot-assisted surgical training over several days in a virtual surgical environment with divergent and convergent force fields," in *Hamlyn Symposium*, 2019, pp. 81–82.

[3] N. Enayati, A. M. Okamura, A. Mariani, E. Pellegrini, M. M. Coad, G. Ferrigno, and E. De Momi, "Robotic assistance-as-needed for enhanced visuomotor learning in surgical robotics training: An experimental study," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6631–6636.

[4] J. M. Walker, A. M. Okamura, and M. J. Kochenderfer, "Gaussian process dynamic programming for optimizing ungrounded haptic guidance," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8758–8764.

[5] D. Sternad, "It's not (only) the mean that matters: variability, noise and exploration in skill learning," *Current opinion in behavioral sciences*, vol. 20, pp. 183–195, 2018.

[6] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[7] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[8] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

APPENDIX I
PARAMETERS

TABLE I

Option	**Value**
Learning Rate	1e-5
Max Steps	max number of episodes
Log Freq	500
Number of Episodes to Evaluate	100
ϵ fraction of training set	0.8
ϵ at end of exploration	0.01
Recurrence	false
Double Q	true
Dueling	true
Prioritized Replay	true

TABLE II

Parameter	**Value**
Baseline Process Noise Mean	0
Baseline Process Noise Var	0.5
Process Noise Attenuation Factor	$e^{-0.05 \times trialnumber}$
Process Noise Mean	0
Process Noise Variation	0.5
Trial Time Upper Bound	60
Trial Time Lower Bound	30
Trial Time Variance	1
Path Error Upper Bound	30
Path Error Lower Bound	10
Path Error Variance	0.5

APPENDIX II
CODE REPOSITORIES

The Github code repository is available upon request from the authors.