2022 2nd International Conference on Robotics, Automation and Artificial Intelligence

Lessons Learned from Utilizing Guided Policy Search for Human-Robot Handovers with a Collaborative Robot

Alap Kshirsagar*
Cornell University
Ithaca, USA
ak2458@cornell.edu

Tair Faibish*
Ben-Gurion University of the Negev
Be'er Sheva, Israel
tairse@post.bgu.ac.il

Guy Hoffman
Cornell University
Ithaca, USA
hoffman@cornell.edu

Armin Biess
Ben-Gurion University of the Negev
Be'er Sheva, Israel
abiess@bgu.ac.il

2022 2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI) | 978-1-6654-5944-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/RAAI56146.2022.10092989

***Abstract*—We evaluate the performance of Guided Policy Search (GPS), a model-based reinforcement learning method, for generating the handover reaching motions of a collaborative robot arm. In a previous work, we evaluated GPS for the same task but only in a simulated environment. This paper provides a replication of the findings in simulation, along with new insights on GPS when used on a physical robot platform. First, we find that a policy learned in simulation does not transfer readily to the physical robot due to differences in model parameters and existing safety constraints on the real robot. Second, in order to successfully train a GPS model, the robot's workspace needs to be severely reduced, owing to the joint-space limitations of the physical robot. Third, a policy trained with moving targets results in large worst-case errors even in regions spatially close to the training target locations. Our findings motivate further research towards utilizing GPS in human-robot interaction settings, especially where safety constraints are imposed.**

***Index Terms*—Physical Human-Robot Interaction, Reinforcement Learning, Manipulation Planning**

I. Introduction

In this work, we evaluate the potential of Guided Policy Search (GPS) for learning robot reaching motions in human-to-robot object handovers. GPS is a reinforcement learning (RL) algorithm that was initially proposed by Levine et al. [1]–[3]. Since then researchers have proposed several variations of the GPS algorithm [4], and have successfully utilized them for autonomous manipulation [5]–[8], and locomotion tasks [1], [2], [6], [9]. However, the potential of GPS for human-robot interaction tasks is less well-understood. Our work seeks to address this gap, as human-interactive tasks have several distinguishing features, which merit specific investigation. For example, as detailed in our previous work, human-robot handovers require large spatial variations in reach locations, moving targets, and generalizing over changes in the robot's mass induced by the object being handed over [10]. The variant of GPS that we use in this work, GPS-Bregman Alternating Direction Method of Multipliers (BADMM) [8], does not require prior knowledge of the robot or the environment dynamics.

Our previous work evaluated GPS-BADMM in a simulation environment [10]. We found that the policy learned with GPS does not perform well for test locations that are spatially distant from training locations, and requires the addition of more local controllers to cover high error regions. Further, a policy trained with static targets generates unreasonably high joint torques when tested with moving targets. More efficient reaching trajectories can be obtained by training on moving targets, although that decision results in higher worst-case errors. Despite providing important insights, this prior work is limited in its application because it was conducted in simulation only, and not evaluated on a physical robot.

The goal of the present work is twofold: first, to replicate our previous findings on a physical robot arm, and second, to provide new insights on the challenges associated with the use of GPS on a physical human-robot handover task.

II. Policy Search Formulation of Handovers

To formalize the reach phase of a handover task as an RL problem, we specify the state-action space along with a cost or reward function over the system states and control inputs. This largely builds on our previous work [10], and we refer the reader to that paper for more details.

A. State/Action Space

In our previous work [10], we found that a policy trained with a robot-centric state representation had the best overall performance. Thus, in this work, we use a robot-centric state representation which consists of the robot joint angles θ_r, the robot joint velocities $\dot{\theta}_r$, the positions and velocities of the robot-end effector in the world frame attached to the base of the robot ($\mathbf{p}_r$,$\dot{\mathbf{p}}_r$), and the positions and velocities of the human hand in the robot end-effector frame ($\mathbf{p}_h^r$,$\dot{\mathbf{p}}_h^r$).

$$\mathbf{x}_t = [\theta_r, \dot{\theta}_r, \mathbf{p}_r, \mathbf{p}_h^r, \dot{\mathbf{p}}_r, \dot{\mathbf{p}}_h^r]_t. \quad (1)$$

The robot's control input $\mathbf{u}_t$ consists of the robot joint torques τ: $\mathbf{u}_t = \tau_t$.

* Equal contribution.

This research was supported in part by the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Initiative and by the Marcus Endowment Fund both at Ben-Gurion University of the Negev. This research was supported by the Israel Science Foundation (Grant No. 1627/17).

Authorized licensed use limited to: Cornell University Library. Downloaded on November 05,2026 at 19:14:29 UTC from IEEE Xplore. Restrictions apply.

B. Cost Function

We use a cumulative error cost function to describe the reach-to-handover motion of a robot,

$$c_{reach} = \sum_{t=0}^{T} \left[||\mathbf{p}_r - \mathbf{p}_h||^2 + \ln(||\mathbf{p}_r - \mathbf{p}_h||^2 + \alpha_{reach}) \right]_t , \quad (2)$$

where T is the duration of each trial. This cost function penalizes the robot for a spatial distance away from the human's hand, and it encourages precise placement owing to its concave shape, as described in [7]. To be consistent with prior works [7], [10], we set $\alpha_{reach} = 10^{-5}$ in the evaluations described in the next section.

C. Overview of the GPS-BADMM Algorithm

The GPS-BADMM algorithm is described in detail in [8], and we provide its basic operation here for clarity.

GPS uses a set of linear Gaussian controllers ("local controllers" henceforth), each being optimized via the iterative Linear Quadratic Regulator (iLQR) algorithm [11] for a so-called *initial condition*. In our case, the "initial conditions" are handover location targets. The GPS algorithm alternates between generating optimal trajectories for each initial condition and training a global policy, which could be a deep neural network, supervised by the local controllers. The global policy's role, in turn, is to improve the local controllers: in each iteration, one of objectives of the local controller optimization is to stay close to the global policy. The BADMM variant of the GPS algorithm does not require knowledge of the dynamics model as it utilizes the training data with locally linear models to approximate the dynamics of the robot and the environment.

III. Implementation of GPS-BADMM on a Franka-Emika Robot

We train a collaborative robot, a Franka-Emika Panda research robot ("Panda robot" henceforth), to perform reach-to-handover motions with the GPS-BADMM algorithm[1]. This is the same robot that we used in the simulation in our previous work. The robot, shown in Fig. 2, is a 7 degrees-of-freedom (DoF) robot arm with torque sensors at each joint, allowing adjustable stiffness/compliance and torque control. We use an OptiTrack motion tracking system to sense the positions of the human's hand and the robot's end effector.

The robot is controlled via a distributed Robot Operating System (ROS) network across three personal computers (PC) as shown in Fig. 1: One PC runs the motion tracking software and streams the motion tracking data on a local network. The second PC converts the motion tracking data into the relevant coordinate frames, computes the state representations described in Section II-A, and runs the GPS training and testing procedures to generate controls for the robot. These are sent to a third PC, which is physically connected to the Panda robot. This third PC runs a real-time kernel as required by the robot manufacturer.

[1]The code is available at: https://github.com/alapkshirsagar/real-franka-gps

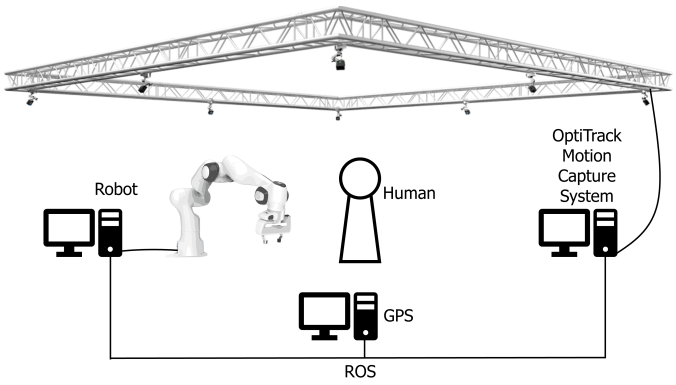


Fig. 1: The system setup includes a Panda robot (left), an Optitrack motion capture system (top), and three PCs communicating through ROS: one collecting motion tracking data (right), one responsible for RL (bottom center), and one connected to the robot (left).

We build on the GPS-BADMM implementation of White [12], which was designed for a Kuka LWR robot [13] in the Gazebo simulation environment. White's code, in turn, was based on the GPS implementation of Finn [14], designed for a Willow Garage PR2 robot [15]. In order to adapt the implementation of White [12] to the physical Panda robot, we had to tune several parameters, including the initial gains, variances, and stiffness parameters for the local iLQR controllers, and the PID parameters used to command the robot back to its initial position, after each trial. Tables I and II compare the values used by previous implementations and the one presented in this paper. For more details about these adaptations and parameters, we refer the reader to [16]. In general, we found that the high gains used in previous implementations resulted in the robot abruptly halting due to violations of joint trajectory or Cartesian trajectory requirements[2]. These requirements consist of safety limits on the robot's joints' positions, velocities, and torques, as well as on the robot end-effector's Cartesian position, velocity, acceleration, and jerk. Such inbuilt limits are common for robot arms designed for close-range human-robot interaction.

TABLE I: Comparison of PID parameters of position controller for resetting the LWR and the Panda robots.

Joint	LWR Values [12]				Panda Values			
	P	I	D	I_{clamp}	P	I	D	I_{clamp}
1	2400	0	18	4	6	3	3	1
2	1200	0	20	4	6	3	3	1
3	1000	0	6	4	6	3	3	1
4	700	0	4	4	6	3	3	1
5	300	0	6	2	2.5	1	1	1
6	300	0	4	2	2.5	1	1	1
7	300	0	2	2	2.5	1	1	1

IV. Evaluation of GPS-BADMM on a Physical Handover Task

The goals of the current work are, first, to examine whether our previous findings, which were performed only in sim-

[2]Franka-Emika Panda Specifications: https://frankaemika.github.io/docs/control_parameters.html

Authorized licensed use limited to: Cornell University Library. Downloaded on November 05,2026 at 19:14:29 UTC from IEEE Xplore. Restrictions apply.

TABLE II: Initial controller values of PR-2, LWR, and Panda robots.

Parameter	PR-2 [14]	LWR [12]	Panda
Joint 1 Gain	3.09	24	0.1
Joint 2 Gain	1.08	12	0.1
Joint 3 Gain	0.393	10	0.1
Joint 4 Gain	0.674	7	0.1
Joint 5 Gain	0.111	3	0.001
Joint 6 Gain	0.152	3	0.001
Joint 7 Gain	0.098	6	0.001
Initial Variance	1	30	0.5
Stiffness	0.5	60	1.0
Stiffness Velocity	0.25	0.25	0.5

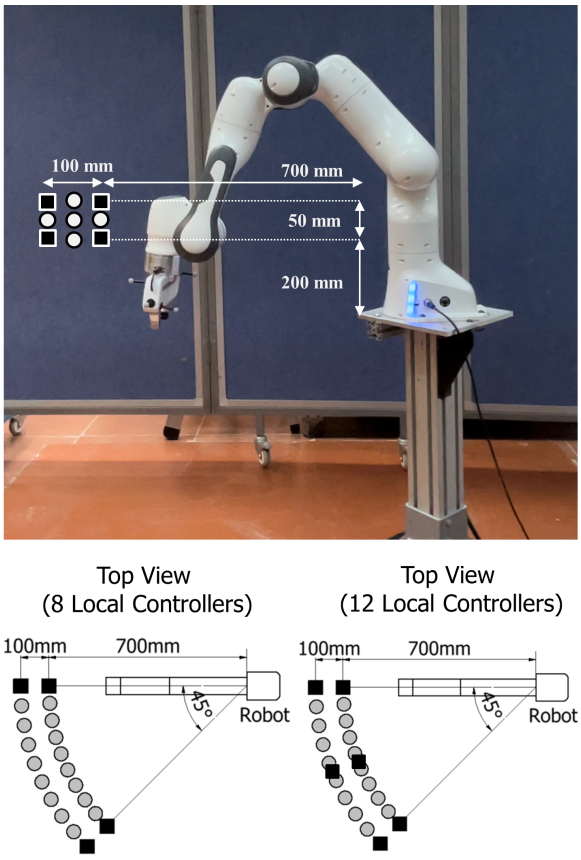


Fig. 2: The training and testing locations for 8 and 12 local controllers. The squares represent the initial training locations, and the circles represent the testing locations. This region was selected by trial and error to ensure that the robot does not run into joint/Cartesian limits in the training/testing process. For 12 local controllers, additional 4 training locations are located in a vertical plane dividing the workspace.

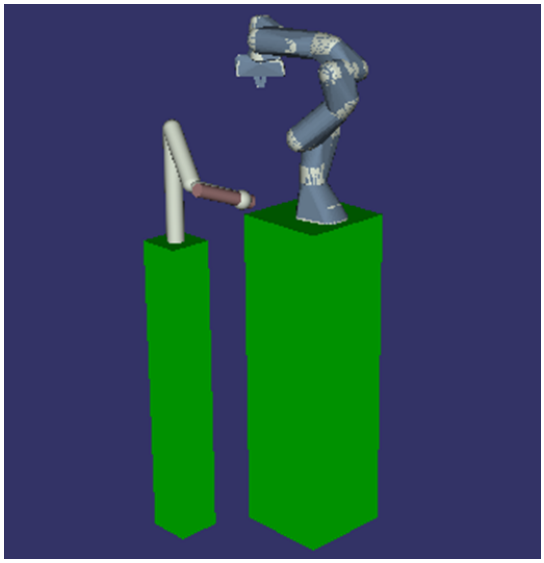


Fig. 3: MuJoCo (Multi-Joint dynamics with Contact) simulation environment for human handover tasks. A Panda robot (right) was trained in simulation on reaching movements in a human-to-robot handover task. The human operator is represented by a pseudo-robot (left).

ulation, replicate, and, second, to investigate the challenges associated with the use of GPS in a physical Human-Robot Interaction (HRI) setup. We do so in two stages: First, we test whether policies learned in simulation can be used in a physical setup ("Sim-to-Real Evaluation"), and second, we try to use GPS to train policies using the physical robot ("Real-to-Real Evaluation").

A. *Sim-to-Real Evaluation*

We previously used GPS to train a robot reach-to-handover in a simulation environment called *MuJoCo* (Multi-Joint dynamics with Contact) [17], as shown in Fig. 3. We trained a Panda robot for the task with a pseudo-robot arm with a mass rigidly attached to its end-effector, substituting the human operator [10]. In the first experiment of the present work, we ask whether the learned policy can be used on a physical Panda robot.

We find that the policy trained in simulation fails to generate correct trajectories on the physical Panda robot. This failure is due to two main reasons: differences in model parameters between the simulated and real robot and inbuilt safety constraints on the real robot.

While robot manufacturers provide some information about their hardware and software, some of it is proprietary or undocumented. As a result, simulation models almost always have to make assumptions about some important dynamics parameters. In this case, we used a MuJoCo model of the Panda robot created with the robot's meshes and specifications from *franka_ros*[3]. This model used the geometry provided by the manufacturer and any existing dynamics parameters. However, the manufacturer does not provide a damping factor

[3]Franka-Emika Panda MuJoCo Model: https://github.com/vikashplus/franka_sim

Authorized licensed use limited to: Cornell University Library. Downloaded on November 05,2026 at 19:14:29 UTC from IEEE Xplore. Restrictions apply.

for each joint, which was set arbitrarily, resulting in poor Sim-to-Real dynamics transfer.

Furthermore, the Panda robot, like most human-collaborative robots, has inbuilt safety constraints for joint positions, velocities, and torques (see also: Section III). In the Sim-to-Real transfer, many of the torques generated by the GPS policy learned in simulation exceeded these constraints.

It is worth noting here that the GPS algorithm does not readily incorporate hard constraints. We tried to impose a torque reduction in the simulation training by adding a penalization term in the RL cost function for out-of-limit velocities and out-of-limit torques. The torques were reduced slightly but not enough to run the global policy on the real robot. We also clamped the torques before the torques were sent to the robot's joints in MuJoCo. In that case, the robot could not learn a handover trajectory at all and barely moved from its initial position. In summary, due to the architecture of GPS training, undisclosed robot parameters, and the fact that human-collaborative robots have hard safety constraints, it proved difficult to train in simulation and execute trajectories on a physical robot.

B. Real-to-Real Evaluation

In the second stage of evaluation, we used GPS to directly train a physical Panda robot to perform reaching motions towards a human's hand, and tested the learned global policy for large variations in target locations and moving targets. To reiterate, we identified large target variations and moving targets as important characteristics of human-robot handovers that were difficult for GPS to generalize over.

In the remaining text, we denote the Panda robot the "learner", and the human the "trainer" or "tester" for the training and testing phase, respectively. Since it is not practical to have a human perform exactly the same reaching motion in all training iterations, we used recorded human hand motions and replayed them to the robot via ROS.

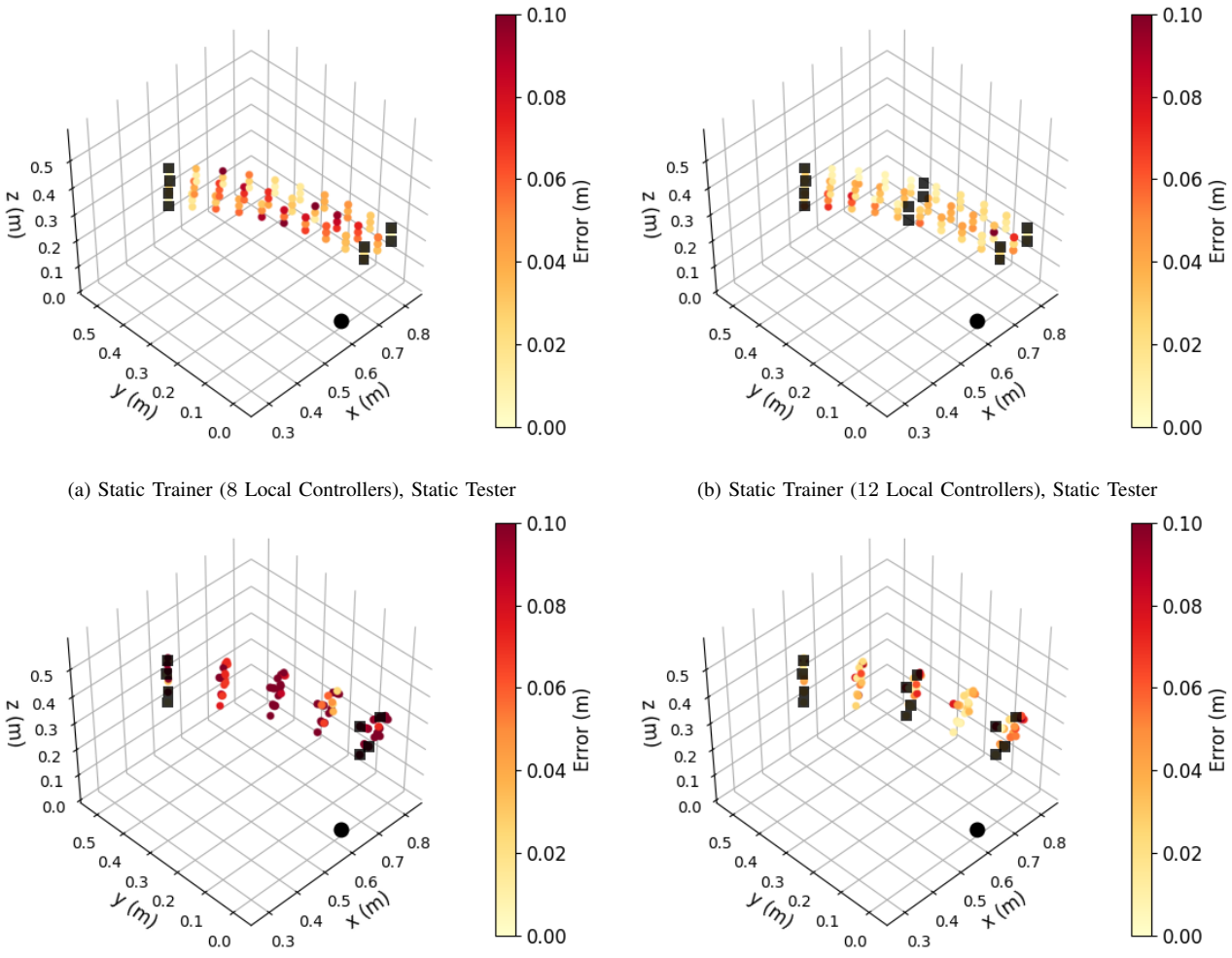
The first research question examined in our study is the spatial generalizability of the learned global policy, i.e., how does the global policy perform for large spatial differences between training and testing locations. To answer this question, we first tried to use the same training and testing locations as our previous work [10]. However, we found that the Panda robot ran into joint and Cartesian limit violations for those training locations. We had to reduce the robot's workspace by trial-and-error. The final workspace used in our evaluation is shown in Fig. 2. Replicating the conditions in the simulation study, we compared two scenarios of local controllers: one with 8 local controllers and another with 12 local controllers. The global policy was trained with these local controllers for 11 iterations. The learner's movement lasted for 5 seconds, while the trainer/tester remained static. The test performance was measured as the mean error between the learner's gripper position and the tester's hand position at the last time step of each trial.

The performance of the learned global policy is shown in Fig. 4a. The black circle represents the learner's gripper's initial position, and the black squares represent the training locations. The colored circles indicate the error at the last time step for each testing location. Mean error and its standard deviation are shown in Fig. 5 (left). The mean testing error (41.7mm) is about twice as large as the mean training error (22.7mm). If our previous studies replicate, this test error can be reduced by adding more local controllers in high error regions. In the current study, we similarly find that adding four additional local controllers in a vertical plane dividing the workspace (Fig. 4b), reduced the mean and standard deviation of the testing error. Trained on these 12 local controllers, the mean error of the global policy was reduced to 29mm.

Next, we investigated how GPS performs when the target is moving. First, we used the same global policy shown in Fig. 4a (static training), but instead of a static tester, we used a moving target, i.e., a recorded human reaching motion. The final position of the motion was in a region similar to the one shown in Fig. 2. Similar to our simulation studies, we find that the robot generated highly inefficient trajectories, and sometimes did not even execute these trajectories due to safety constraint violations. A possible way to address this issue is to train the controller with a moving target. To do so, we trained the robot with recorded human reaching motions and tested the policy on another set of recorded human reaching motions, as well as on real-time human handovers. In the recorded human reaching motions, the movement of the trainer/tester lasted 1 second, the peak speed was between 0.56m/s to 1.13m/s, and the range of motion was between 0.38m to 0.55m. The learner and the trainer/tester began the reaching motion at the same time, and the learner's motion lasted for 5 seconds. Fig. 4c and 4d show the performance of the global policy for various final positions of the tester's gripper, defined as in previous trials. Fig. 5 (right) shows error distributions.

For the global policy trained with a moving trainer and 8 local controllers (Fig. 4c), the mean testing error is 124.3mm. Although the test errors are high as compared to the static tester scenario, the robot stays within the joint and Cartesian limits. Moreover, the variance over the target location is high, and the worst-case error is 791.1mm, 442% higher than the maximum error for the static tester condition (179mm). Surprisingly, the tester's motion for the worst-case error is close to one of the training motions. This can be attributed to the highly non-linear nature of the global policy. Interestingly, GPS does not converge to a low training error for the moving trainer scenario, averaging at 123.2mm which is 544% higher than the training error for a static trainer 22.7mm. Training the global policy with a moving trainer and 12 local controllers (Fig. 4d) reduces the mean testing error to 37.9mm, which is comparable to a static trainer. The worst-case error also improves to 138.7 mm. Fig. 5 shows the distributions of training and testing performance for each target scenario.

Authorized licensed use limited to: Cornell University Library. Downloaded on November 05,2026 at 19:14:29 UTC from IEEE Xplore. Restrictions apply.



(a) Static Trainer (8 Local Controllers), Static Tester

(b) Static Trainer (12 Local Controllers), Static Tester

(c) Moving Trainer (8 Local Controllers), Moving Tester

(d) Moving Trainer (12 Local Controllers), Moving Tester

Fig. 4: Global policy evaluation for different types of trainers and testers. The black circle represents the learner's gripper's initial position, and the black squares represent the training locations. In the 'static' case, the trainer/tester stays in a fixed configuration. In the 'moving' case, the trainer/tester moves with a human-like trajectory (that were recorded in advanced) and reaches the locations given by colored dots. Thus, each point corresponds to the final position of the tester's gripper in a trial. Error between the learner's gripper position and the tester's gripper position is calculated over the last time step of each trial. Error values are clamped to the range [0.0, 0.1] for better visualization.

V. Discussion and Conclusion

Our work evaluates the feasibility of GPS as a learning method for generating robot reaching motions for human-robot handovers. To do so, we attempt to train a policy that is stable over relatively large variations in target locations as well as for moving targets. In the past, we evaluated these questions only in simulation. This work trains and tests the approach on a physical collaborative robot.

Attempting to transfer policies learned on a simulated robot to the physical setup proved infeasible. The robot runs into joint and Cartesian limits and halts almost immediately. This can be attributed to the differences between the simulation model's dynamics and the real robot's dynamics. Tuning the simulation model dynamics parameters to match the real robot's parameters is often not possible owing to a large number of possibilities and undisclosed manufacturer parameter settings. GPS has been shown to be robust to changes in the robot's dynamics within a certain range [10], but our findings suggest that GPS is not robust enough to directly transfer learning from simulation on a robot with unknown dynamics to a real robot. That said, a higher-precision simulation model might allow for better Sim-to-Real transfer and automatically generating such a model could be a fruitful area of future research.

Another hurdle for Sim-to-Real transfer of learned GPS policies lies in the prevalence of built-in safety limits found in human-collaborative robots, and the difficulty to impose such limits in the GPS training phase.

When GPS is used to directly train the physical robot, we find that the robot runs into these safety constraints during the training phase. To avoid these violations, we have to reduce the robot's workspace by trial-and-error. In this reduced workspace (Fig. 2), we find that most of our results

Authorized licensed use limited to: Cornell University Library. Downloaded on November 05,2026 at 19:14:29 UTC from IEEE Xplore. Restrictions apply.

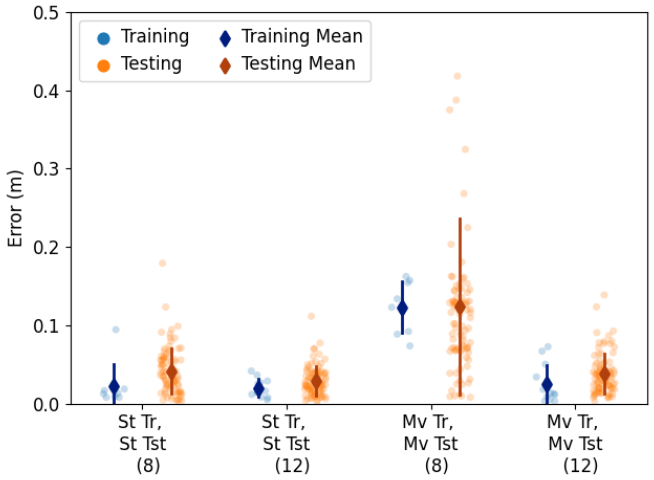


Fig. 5: Distributions of training and testing performance for each target scenario. Each point is the error between the learner's gripper position and the tester's hand position at the last time step of a trial. Error bars show one standard deviation around the mean of each distribution

from the simulation environment replicate. When the robot is trained to reach only static target locations, the global policy performance can be improved by adding local controllers in regions with highest test errors (Fig. 4a compared to Fig. 4b).

When evaluating the global policy trained with static targets on a moving test target, the robot generates highly inefficient trajectories, sometimes resulting in halts due to joint limit violations. To overcome this issue, we trained the global policy with moving targets. Nevertheless, this solution is not free of drawbacks. It successfully reduces the mean error and results in more efficient and low-torque trajectories, but also results in a high-variance (and therefore unreliable) global policy with significantly larger worst-case errors. This issue can be addressed by adding local controllers to the training phase, improving the global policy performance (Fig. 4d). These findings also support our previous findings in simulation, namely that GPS works best with variable moving targets when the space of possible end-positions is covered densely with the training end-positions.

We hope that the presented study can contribute to the understanding of the challenges and applicability of GPS in a real-world HRI context. Though we evaluated GPS only on one specific robot, the Franka-Emika Panda, our two key findings are relevant for other HRI applications of GPS with different robots. First, while GPS continues to be a promising approach, to be useful for safety-critical scenarios around humans, it needs to take into account the safety constraints required in such contexts. Second, while GPS-BADMM is theoretically agnostic to the robot's dynamics, there is a need for highly accurate dynamics models in order to allow training in simulation, which may often be required. For example, it would be infeasible to have a human repetitively provide training data around a physical robot, requiring good Sim-to-Real performance. These challenges provide a fertile ground for future research.

References

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

[2] ——, "Variational policy search via trajectory optimization," in *Advances in neural information processing systems*, 2013, pp. 207–215.

[3] ——, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, 2014, p. II–829–II–837.

[4] J. Du, J. Fu, and C. Li, "Guided policy search methods: A review," *Journal of Physics: Conference Series*, vol. 1748, no. 2, pp. 022–039, 2021.

[5] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," in *IEEE international conference on robotics and automation (ICRA)*, 2017, pp. 3381–3388.

[6] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[7] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 26–30.

[8] 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.

[9] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 528–535.

[10] A. Kshirsagar, G. Hoffman, and A. Biess, "Evaluating guided policy search for human-robot handovers," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3933–3940, 2021.

[11] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[12] J. White, "Guided policy search for a lightweight industrial robot arm," Master's thesis, Luleå University of Technology and Aalto University/Erasmus+, 2018.

[13] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald *et al.*, "The kuka-dlr lightweight robot arm-a new reference platform for robotics research and manufacturing," in *41st International Symposium on Robotics and 6th German Conference on Robotics*, 2010, pp. 1–8.

[14] C. Finn, M. Zhang, J. Fu, W. Montgomery, X. Yu Tan, Z. McCarthy, B. Stadie, E. Scharff, and S. Levine, "Guided policy search code implementation," Software available from rll.berkeley.edu/gps (2022/03/01).

[15] W. Garage, "Pr2 user manual," 2012.

[16] T. Faibish, "Human-robot handovers: Human preferences and robot learning," Master's thesis, Department of Industrial Engineering and Management, Ben-Gurion university of the Negev, Beer Sheva, Israel, 2022, [Online:https://in.bgu.ac.il/en/robotics/thesis/TairFaibish.pdf; accessed 22-July-2022].

[17] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

Authorized licensed use limited to: Cornell University Library. Downloaded on November 05,2026 at 19:14:29 UTC from IEEE Xplore. Restrictions apply.