Human-in-the-Loop Robot Planning with Non-Contextual Bandit Feedback

Yijie Zhou, Yan Zhang, Xusheng Luo, and Michael M. Zavlanos

***Abstract*— In this paper, we consider robot navigation problems in environments populated by humans. The goal is to determine safe trajectories that also maximize human satisfaction. In practice, human satisfaction is subjective and hard to describe mathematically. As a result, the planning problem we consider in this paper may lack important contextual information. To address this challenge, we propose a semi-supervised Bayesian Optimization (BO) method to design globally optimal robot trajectories using bandit human feedback, in the form of complaints or satisfaction ratings, that expresses how desirable a trajectory is. We demonstrate the efficiency of our proposed trajectory planning method in a simulated scenario where humans have diversified and unknown demands.**

I. INTRODUCTION

A growing number of current and future robotics applications, such as unmanned delivery [1, 2] and robotic nurse assistance [3, 4], require robots to operate in close proximity to humans. In these applications, besides meeting safety requirements, trajectory planning also needs to ensure that robots are available to assist humans as needed without causing discomfort. This may involve adapting to specific ways that specific humans perform their tasks or avoiding actions that cause discomfort. In practice, such requirements are subjective and hard to model mathematically. As a result, new planning methods are needed to design robot trajectories that maximize human satisfaction, using exclusively bandit human feedback that lacks contextual information typically captured by mathematical models.

Robot planning using bandit human feedback has been studied in [5]–[7]. Specifically, in [5], a robot simultaneously learns a policy and a reward model by actively querying a human expert for ratings. Similarly, [6] adopts a data-driven approach to design robot trajectories using human ratings on different roll outs. Nevertheless, these methods often require prohibitively many queries for human feedback under high dimensional settings. In [7] we proposed a new zeroth-order method that uses bandit human feedback to design robot trajectories that increase human satisfaction. However, since this problem is highly non-linear, zeroth-order methods can get trapped at local minima, reducing the quality of the resulting paths. Moreover, [7] only considers a single type of human feedback that may not be practical in the real world.

To plan globally optimal robot trajectories that maximize human satisfaction, we propose a Bayesian Optimization (BO) method that uses bandit human feedback in the form of complaints or satisfaction ratings to capture the quality of robot plans. However, since the BO search space can be high-dimensional, e.g., when each trajectory is parameterized by multiple waypoints, prohibitively many queries for human feedback may be needed to find a satisfactory solution [8]. To reduce the dimension of the state space, the methods in [9] assume that the objective function can be decomposed into the summation of several lower dimensional functions. However, this method assumes knowledge of the objective function, which is not available in our case. The approaches in [10, 11] project the original solution space into a low dimensional space using PCA or VAE. Since these methods do not utilize objective function information, the low dimensional space may not contain the optimal solution and, therefore, it can introduce bias. The authors in [12] combine BO with manifold Gaussian Processes (mGP) to find the optimal low dimensional space with objective function values obtained online. Nevertheless, this approach still requires prohibitively many queries for human feedback.

In this paper, we propose a semi-supervised method to improve the data efficiency of BO for human-in-the-loop trajectory planning. Specifically, we first utilize an autoencoder to compress the space of robot trajectories to a latent space. Since training the autoencoder does not require human feedback, this embedding can be learnt offline in an unsupervised way. Next, we employ BO on this latent space, which we combine with human feedback collected online, to iteratively optimize both the latent space and the robot trajectories. Moreover, to improve the exploration efficiency of BO, we bias the search for new trajectories towards dynamically feasible and collision-free trajectories. Our numerical experiments demonstrate that the semi-supervised BO framework, combined with biased exploration, can handle diversified human preferences and achieve better data efficiency compared to existing methods.

II. PROBLEM DEFINITION AND PRELIMINARIES

A. Problem Definition

Consider an environment $\mathcal{W} \subset \mathbb{R}^p$, where $p = 2$ or 3, that contains m obstacles $\mathcal{O} = \{o_1, ..., o_m\}$. In addition, consider a robot with state $q \in \mathcal{Q}$, where $\mathcal{Q}$ denotes the feasible state space. Assume also that the robot satisfies the following discrete-time dynamics

$$q_{t+1} = f(q_t, u_t), \tag{1}$$

where $u_t \in \mathcal{U}$ denotes the control input at time t and $\mathcal{U}$ is the set of feasible control inputs. Moreover, let $x_t = \Pi(q_t)$

The authors are with the Department of Mechanical Engineering and Material Science, Duke University, Durham, NC 27708, USA {e-mail: yijie.zhou,yan.zhang2,xusheng.luo, michael.zavlanos}@duke.edu. This work is supported in part by AFOSR under award #FA9550-19-1-0169 and by NSF under award CNS-1932011.

denotes the projection of the robot state q_t to its position x_t in the environment. Traditionally, the robot planning problem consists of designing a robot trajectory $\mathbf{x} = \{x_0, ..., x_T\} \in \mathcal{C} \cap \mathcal{D}$ to drive the robot from a starting location x_0 to a goal position x_T, where $\mathcal{C} = \mathcal{W} \backslash \mathcal{O}$ is the space of collision-free trajectories, and $\mathcal{D}$ is the space of dynamically feasible trajectories that satisfy the dynamics in (1).

In this paper, we consider a robot planning problem to design robot trajectories that are not only dynamically feasible and collision-free, but also maximize human satisfaction. Specifically, we assume that the environment $\mathcal{W} \backslash \mathcal{O}$ is populated by a group of humans in the set $\mathcal{H}$ that require different levels of assistance from the robot, which may involve adapting to specific ways that specific humans perform their tasks or avoiding actions that cause discomfort. While such requirements are subjective and hard to model mathematically in practice, the resulting levels of human satisfaction can be measured using bandit human feedback, in the form of complaints or satisfaction ratings, that expresses how desirable a robot trajectory is, without revealing the reason. To this end, we divide the humans in the set $\mathcal{H}$ into two subgroups, depending on the type of feedback they can provide. Particularly, we assume that $\mathcal{H} = \mathcal{H}^c \cup \mathcal{H}^f$, where humans $i \in \mathcal{H}^c$ provide feedback measured by the function $f_i^c : \mathbf{x} \to \{0, 1\}$ that captures complaints related to discomfort so that a value of 0 indicates no discomfort, while humans $i \in \mathcal{H}^f$ provide feedback measured by the function $f_i^f : \mathbf{x} \to \{0, 1, ..., k\}$ that captures satisfaction ratings on the level of assistance provided by the robot so that lower values indicate higher levels of satisfaction. Moreover, we let the function $\mathcal{F}(\mathbf{x}) := \sum_{i=1}^{|\mathcal{H}^c|} f_i^c(\mathbf{x}) + \lambda \sum_{i=1}^{|\mathcal{H}^f|} f_i^f(\mathbf{x})$ capture the total feedback on the trajectory $\mathbf{x}$, where λ is a weight used to ensure fairness between the two groups. Then, the problem we address in this paper is to design a collision-free and dynamically feasible trajectory $\mathbf{x}^*$ that maximizes human satisfaction, i.e.,

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{C} \cap \mathcal{D}} \mathcal{F}(\mathbf{x}). \tag{2}$$

Note that the form of the objective function $\mathcal{F}(\mathbf{x})$ is unknown to the agent. Only evaluations of $\mathcal{F}(\mathbf{x})$ are provided. Here the objective function is defined by the summation of two discrete functions, but it could also be any real valued function in other scenarios. In what follows, we employ BO to solve this planning problem. But first, we provide a brief overview of BO.

B. Bayesian Optimization

Given a dataset $D_N = \{[\mathbf{x}_i, \mathcal{F}(\mathbf{x}_i)], i \in \{1, ..., n\}\}$ consisting of N sampled trajectories and corresponding bandit feedback, BO can be used to estimate the shape of the unknown objective function $\mathcal{F}$ using a non-parametric model, e.g., Gaussian Process (GP). Specifically, using GPs assumes that the function value of an unobserved trajectory $\mathbf{x}$ is subject to a Normal distribution $\mathcal{F}(\mathbf{x})|D_N, \mathbf{x} \sim \text{Normal}(\mu_N(\mathbf{x}), \Sigma_N(\mathbf{x}))$, where $\mu_N(\mathbf{x}) = \Sigma_0(\mathbf{x}, \mathbf{x}_{1:N}) \Sigma_0(\mathbf{x}_{1:N}, \mathbf{x}_{1:N})^{-1} \mathcal{F}(\mathbf{x}_{1:N})$ is the mean, $\Sigma_N(\mathbf{x}) = \Sigma_0(\mathbf{x}, \mathbf{x}) - \Sigma_0(\mathbf{x}, \mathbf{x}_{1:N}) \Sigma_0(\mathbf{x}_{1:N}, \mathbf{x}_{1:N})^{-1} \Sigma_0(\mathbf{x}_{1:N}, \mathbf{x})$ is the variance, $\mathcal{F}(\mathbf{x}_{1:N}) = [\mathcal{F}(\mathbf{x}_1), ..., \mathcal{F}(\mathbf{x}_N)]$ is the collection of function evaluations, and $\Sigma_0(\mathbf{x}_{1:j}, \mathbf{x}_{1:k}) = [\Sigma_0(\mathbf{x}_1, \mathbf{x}_1), ..., \Sigma_0(\mathbf{x}_1, \mathbf{x}_j); ...; \Sigma_0(\mathbf{x}_k, \mathbf{x}_1), ..., \Sigma_0(\mathbf{x}_k, \mathbf{x}_j)]$, where $j, k \in \{1, 2, \dots, N\}$ is the kernel matrix. Note that, the kernel, Σ_0, is a covariance function that evaluates the similarity of two points. In this paper, we use a Gaussian kernel $\Sigma_0(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2l^2})$, where σ_f, l are parameters that can be tuned. Given the estimated shape of the unknown function $\mathcal{F}(\mathbf{x})$, BO then selects the next trajectory $\mathbf{x}_e$ to evaluate so that $\mathbf{x}_e$ optimizes the acquisition function, that is, $\mathbf{x}_e = \arg\min \mu_N(\mathbf{x}) - \kappa \sqrt{\Sigma_N(\mathbf{x})}$, where κ is the trade-off coefficient and $\mu_N(\mathbf{x}) - \kappa \sqrt{\Sigma_N(\mathbf{x})}$ represents the lower confidential bound of the objective function value at $\mathbf{x}$. Essentially, BO explores the trajectory $\mathbf{x}_e$ which minimizes the objective function composed of the estimated mean value $\mu_N(\mathbf{x})$ and the uncertainty of the estimation given by the variance $\Sigma_N(\mathbf{x})$ over $\mathbf{x}$.

Note that problem (2) is a constrained optimization problem, where the constraint set $\mathcal{C} \cap \mathcal{D}$ cannot be represented explicitly. Therefore, it is difficult to guide the trajectory exploration procedure within the set $\mathcal{C} \cap \mathcal{D}$ using the methods proposed for constrained BO in [13, 14]. Furthermore, since when the number of waypoints T in the trajectory $\mathbf{x}$ is large, the dimension of the solution space $D = Tp$ is also large. Consequently, BO requires a lot of exploration and human feedback in order to find an optimal solution. In the next section, we discuss ways to address these challenges.du

III. Algorithm Design

In this section, we develop a semi-supervised BO method for human-in-the-loop robot planning. Specifically, we reformulate problem (2) so that trajectory exploration is restricted to the set of dynamically feasible and collision-free trajectories $\mathcal{C} \cap \mathcal{D}$. Then, we modify the objective function in (2) with additional terms that bias the search for new trajectories towards short-length, dynamically feasible, and collision-free trajectories. This way, we improve the exploration efficiency of BO. Finally, we use an autoencoder to reduce the high-dimensional problem space into a low dimensional latent space, which we update using human feedback.

A. Constraint Satisfaction using Off-the-Shelf Motion Planners

As discussed in Section II.B, restricting the exploration of new trajectories in problem (2) to the safe set $\mathcal{C} \cap \mathcal{D}$ is difficult using existing BO techniques. To this end, we reformulate problem (2) as

$$\min_{\mathbf{x} \in \mathcal{X}} \mathcal{F}(m(\mathbf{x})), \tag{3}$$

where $m(\mathbf{x}) : \mathcal{X} \to \mathcal{X}$ represents the output of an off-the-shelf motion planner, e.g., Model Predictive Control (MPC) [15] or RRT [16], and $\mathcal{X} \in \mathbb{R}^D$ is the space of all possible robot trajectories. Using off-the-shelf motion planners, we can map any reference trajectory $\mathbf{x} \in \mathcal{X}$ that is sampled using existing BO methods to a trajectory $m(\mathbf{x}) \in \mathcal{C} \cap \mathcal{D}$ that

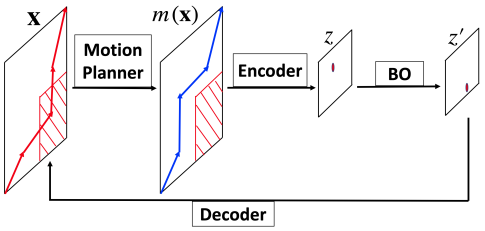

Fig. 1: A schematic of the proposed framework. The dashed red quadrilateral represents the obstacle in 2-d space. Given a reference trajectory $\mathbf{x}$, an off-the-shelf motion planner, e.g. MPC, can be used to generate a collision-free and dynamically feasible trajectory $m(\mathbf{x})$. Then, the trajectory $m(\mathbf{x})$ is encoded into a latent space to obtain a point z, which is used along with human feedback to conduct GP regression and obtain the next latent point z' to be evaluated. Finally, the latent variable z' is decoded to generate the next reference trajectory.

is dynamically feasible and collision-free. The operation of the motion planner $m(\mathbf{x})$ on a reference trajectory $\mathbf{x} \in \mathcal{X}$ is shown in the left part of Fig. 1. The reference trajectory $\mathbf{x}$ (red line) collides with the obstacle, whereas the trajectory $m(\mathbf{x})$ returned by the motion planner is collision-free. In what follows, we use MPC to obtain safe robot trajectories given reference samples $\mathbf{x}$, although other motion planners can also be used. Specifically, we solve the following MPC problem

$$\begin{aligned} &\{\tilde{x}_{t,k}, u^*_{t,k}\} = \operatorname{argmin} J(\{\tilde{x}_{t,k}, u_{t,k}\}) \\ &\text{s.t. } q_{t,k+1} = f(q_{t,k}, u_{t,k}),\ u_{t,k} \in \mathcal{U}, \\ &\quad \Pi(q_{t,0}) = x'_t,\ \Pi(q_{t,k}) \in \mathcal{C},\ \text{for } k = 1, \dots, K-1, \end{aligned} \tag{4}$$

where t, k represent the index for time step and planning step respectively, K is the control horizon, $\tilde{x}_{t,k} = \Pi(q_{t,k})$, is the projection of the current state of the robot to the space of robot positions, $J(\{\tilde{x}_{t,k}, u_{t,k}\}) := \sum_{k=0}^{K}(\tilde{x}_{t,k} - x_{t,k})^T Q_k (\tilde{x}_{t,k} - x_{t,k}) + \sum_{k=0}^{K-1} u_{t,k}^T R_k u_{t,k}$ represents the accumulated tracking error of the reference trajectory $\mathbf{x}$ and control energy cost over the future K steps, and x'_t denotes the true position of the robot at time step t. Note that since the dynamic feasibility and obstacle avoidance constraints are explicitly encoded in (4), the output of the motion planner $m(\mathbf{x})$ is guaranteed to belong to the set $\mathcal{C} \cap \mathcal{D}$. Other safety requirements can also be similarly encoded into problem (4).

B. Exploration Bias in Bayesian Optimization

Notice that the use of motion planners to map unsafe samples $\mathbf{x} \in \mathcal{X}$ to safe trajectories $m(\mathbf{x})$ may reduce the sampling efficiency of BO, since it is possible that multiple unsafe samples $\mathbf{x}$ and $\mathbf{x}'$ from the set $\mathcal{X}$ are mapped to the same safe trajectories, i.e., $m(\mathbf{x}) = m(\mathbf{x}')$. To improve the sample efficiency of BO to solve problem (3), in this section, we propose a way to bias the exploration of new trajectories $\mathbf{x} \in \mathcal{X}$ towards dynamically feasible and collision-free trajectories that are also feasible solutions of typical motion planners. This way, we can avoid sampling redundant unsafe trajectories $\mathbf{x} \in \mathcal{X}$ that are mapped to the same safe trajectory $m(\mathbf{x})$.

To bias trajectory exploration in problem (3), we introduce the term $\|\mathbf{x} - m(\mathbf{x})\|$ in the objective function which allows BO to learn the subspace in $\mathcal{X}$ that contains trajectories that can be tracked by the motion planner (4). Effectively, by selecting reference trajectories $\mathbf{x}$ that minimize the term $\|\mathbf{x} - m(\mathbf{x})\|$, we obtain that, different reference trajectories $\mathbf{x}$ and $\mathbf{x}'$ in this subspace will be mapped to different trajectories $m(\mathbf{x})$ and $m(\mathbf{x}')$ since both $\mathbf{x}$ and $\mathbf{x}'$ are feasible solutions of the motion planner. We note that the term $\|\mathbf{x} - m(\mathbf{x})\|$ does not affect the solution of the original problem (3). To see this, note that the global optimizer $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathcal{X}}\ \mathcal{F}(m(\mathbf{x}))$ also minimizes the modified problem $\min_{\mathbf{x}\in\mathcal{X}}\ \mathcal{F}(m(\mathbf{x})) + \rho\|\mathbf{x} - m(\mathbf{x})\|$, where ρ is a penalty parameter. This is because, if the planning problem is feasible, then there exists a minimizer $\mathbf{x}^* \in \mathcal{X}$ that can be perfectly tracked by the motion planner.

In practice, it may also be useful to add additional terms to the objective function in problem (3) that specify soft constraints on the robot's trajectories. For example, we can modify the objective function in (3) by adding the term $l(m(\mathbf{x}))$ denoting the total length of the trajectory $m(\mathbf{x})$. This modification will bias BO exploration towards trajectories with shorter length. However, unlike the tracking error $\|\mathbf{x} - m(\mathbf{x})\|$, the inclusion of the term $l(m(\mathbf{x}))$ in the objective of the problem (3) affects the final solution, because the global optimizer $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}\in\mathcal{X}}\ \mathcal{F}(m(\mathbf{x}))$ does not necessarily minimize the modified problem $\min_{\mathbf{x}\in\mathcal{X}}\ \mathcal{F}(m(\mathbf{x})) + \tau l(m(\mathbf{x}))$, where τ is a penalty parameter. Therefore, the parameter τ needs to be carefully chosen that human satisfaction is not significantly affected. In what follows, we reformulate problem (6) to the problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} Obj(\mathbf{x}), \tag{5}$$

where $Obj(\mathbf{x}) = \mathcal{F}(m(\mathbf{x})) + \rho||\mathbf{x} - m(\mathbf{x})|| + \tau l(m(\mathbf{x}))$ so that exploration in BO can be accelerated towards dynamically feasible, collision-free, and short-length trajectories.

C. Semi-Supervised Dimension Reduction and Trajectory Optimization

Problem (5) improves the exploration efficiency of BO within the constraint set $\mathcal{C} \cap \mathcal{D}$. However, the safe set $\mathcal{C} \cap \mathcal{D}$ is of the same high dimension as the original space $\mathcal{X}$. In this section, we first discuss how to reduce the dimension of the original space $\mathcal{X}$ by embedding it into a lower dimensional latent space $\mathcal{Z}$, which contains regular trajectories that are of certain degree of smoothness. This embedding is trained using trajectory data not labeled by human feedback. Therefore, this step can be completed offline and in an unsupervised way. Then, we run BO on the low dimensional latent space. Specifically, we sample new trajectories in an online way, query humans for feedback on each new trajectory, and use these data to update both the latent space and the GP model in an online way. Since this process uses human feedback, it is supervised in nature. The complete semi-supervised trajectory optimization framework is presented in Algorithm 1. Next, we explain each step in details.

Algorithm 1 Online Semi-supervised Bayesian Optimization

Input:
Objective function $f : \mathbb{R}^D \to \mathbb{R}$, unlabeled data set $\mathcal{D}_{un} \in \mathbb{R}^{N_u \times D}$, embedding dimension d, maximum query number N_{Query} and size of mini-batch N_m.

1: Randomly initialize θ_E, θ_D, θ_G;
2: Train autoencoder with $\mathcal{D}_{un}$ and update θ_E, θ_D;
3: Randomly generate an initial reference trajectory $\mathbf{x}_0$
4: Use motion planner to get $m(\mathbf{x}_0)$ and evaluate human feedback to get $\mathcal{D}_l = \{\{m(\mathbf{x}_0), y_0\}\}$
5: **for** $1 \leq i \leq N_{Query}$ **do**
6: Encode $\mathcal{D}_l$ to $\mathcal{D}_l^e$
7: Conduct GP regression with θ_G on $\mathcal{D}_l^e$ to get acquisition function $q : \mathbb{R}^d \to \mathbb{R}$
8: $z_i = \underset{z \in \mathbb{R}^d}{\operatorname{argmax}}\ q(z)$
9: Let $\mathbf{x}_i = D(z_i)$
10: Generate $m(\mathbf{x}_i)$ and evaluate $y_i = Obj(m(\mathbf{x}_i))$
11: $\mathcal{D}_l = \mathcal{D}_l \cup \{\{m(\mathbf{x}_i)), y_i\}\}$
12: **if** $i \% N_m = 0$ **then**
13: Update θ_E, θ_D and θ_G according to Algorithm 2;
14: **end if**
15: **end for**
16: Output $\mathbf{x}_{N_{Query}}$.

Given a pair of starting and goal robot positions, we can use off-the-shelf motion planners, e.g., RRT, to generate a dataset $\mathcal{D}_{un}$ containing a large number of random trajectories that respect the robot dynamics (1)[1]. Then, we use the autoencoder developed in [17] to embed these randomly generated trajectories to a low dimensional latent space. Specifically, consider an encoder function $E_{\theta_E} := \mathcal{X} \to \mathcal{Z}$, where θ_E is the parameter of the encoder function and $\mathcal{Z} \in \mathbb{R}^d$ is the latent variable space of dimension d, where $d \ll D$. In addition, consider a decoder function $D_{\theta_D} := \mathcal{Z} \to \mathcal{X}$, where θ_D is the decoder parameter. Then, we can train the encoder and decoder models by solving the problem

$$\min_{\theta_D, \theta_E} L_{re}(\theta_D, \theta_E; \mathcal{D}_{un}), \tag{6}$$

where $L_{re}(\theta_D, \theta_E; \mathcal{D}_{un}) := \sum_{\mathbf{x} \in \mathcal{D}_{un}} ||\mathbf{x} - D_{\theta_D}(E_{\theta_E}(\mathbf{x}))||_2^2$ measures the reconstruction loss between the original trajectory $\mathbf{x}$ and the output of the autoencoder $D_{\theta_D}(E_{\theta_E}(\mathbf{x}))$ (line 2 in Algorithm 1). Since training the autoencoder does not require human feedback on the trajectories in the set $\mathcal{D}_{un}$, this process is unsupervised.

Next, we run BO on the learnt latent space $\mathcal{Z}$. Specifically, instead of directly constructing a GP model of the unknown objective function Obj in (5) that depends on the trajectory $\mathbf{x}$, we construct a GP model that depends on the latent variable z. As shown in Fig. 1, given a newly sampled reference trajectory $\mathbf{x}$ (lines 3 and 9 in Algorithm 1), we first use the motion planner (4) to generate a safe robot trajectory $m(\mathbf{x})$ and collect the human feedback $\mathcal{F}(m(\mathbf{x}))$ (lines 4 and 10 in Algorithm 1). Then we encode the trajectory $m(\mathbf{x})$ into a point z in the latent space (line 6 in Algorithm 1). With the latent point z and the human feedback, we conduct GP regression to get the next latent point z' to be evaluated (lines 7 and 8 in Algorithm 1). Subsequently, we decode z' into the original trajectory space $\mathcal{X}$ to get a new reference trajectory (line 9 in Algorithm 1) and the process repeats.

[1]We note that the generated latent space can be used for any pair of starting and goal positions. If we also know the position of obstacles when training the autoencoder, we can use a motion planner to construct a dataset $\mathcal{D}_{un}$ that contains trajectories which are both dynamically feasible and collision free.

Algorithm 2 Parameter Update

Input:
Initial parameter value θ_E, θ_D, θ_G, labeled data set $\mathcal{D}_l = \{\cup_{i=1}^N \{\mathbf{x}_i, y_i\}\}$, number of epoch N_e, and step size α_E, α_D and α_G;

1: **for** $1 \leq k \leq N_e$ **do**
2: $\theta_E = \theta_E - \alpha_E \nabla_{\theta_E}(L_{re} + L_{nmll})$
3: $\theta_D = \theta_D - \alpha_D \nabla_{\theta_D} L_{re}$
4: $\theta_G = \theta_G - \alpha_G \nabla_{\theta_G} L_{nmll}$
5: **end for**
6: Output θ_E, θ_D and θ_G.

Since the initial latent space $\mathcal{Z}$ is learnt offline in an unsupervised way without human feedback, it does not necessarily contain the true optimal solution to problem (5). Therefore, as N_m new data samples have been collected using BO, they are used to update the parameters of the autoencoder, θ_E and θ_D, and the GP kernel parameter θ_G, i.e., $[\sigma_f, l]$ in Section II.B. Specifically, θ_G is updated by minimizing the negative marginal log-likelihood function

$$\begin{aligned} L_{nmll} &= -\log\, p(Y|\mathbf{X}, \theta_G, \theta_E) \\ &= Y^T \Sigma_0^{-1}(\mathbf{x}_{1:N_m}, \mathbf{x}_{1:N_m}) Y + \frac{1}{2} \log |\Sigma_0(\mathbf{x}_{1:N_m}, \mathbf{x}_{1:N_m})|, \end{aligned} \tag{7}$$

where $Y = [Obj(\mathbf{x}_1), ..., Obj(\mathbf{x}_{N_m})]^T$ and $X = [\mathbf{x}_1, ..., \mathbf{x}_{N_m}]$. Moreover, θ_E, θ_D are updated so that the reconstruction loss in (6) is minimized with respect to the new data in $\mathcal{D}_{N_m}$. We run N_e iterations of stochastic gradient decent to update these parameters according to Algorithm 2, and then run BO again to collect new data samples with the new autoencoder and GP parameters. Since the updates in Algorithm 2 are conducted using trajectory data that contain human feedback, this procedure is supervised.

Combining the unsupervised and supervised learning process described above, we obtain the semi-supervised trajectory optimization framework presented in Algorithm 1.

IV. Numerical Experiments

In this section, we corroborate the effectiveness of our semi-supervised trajectory optimization framework on the human-in-the-loop robot planning example discussed in Section II, specifically, a mobile robot interacting with humans in a 2D space. Moreover, we demonstrate the merit of exploration bias in Section III.B and semi-supervised learning in Section III.C through ablation studies.

Consider a workspace $\mathcal{W} \in \mathbb{R}^2$ of size 20×20. Two square obstacles of length 2 are located at locations $(5, 8)$

and $(11, 13)$ in the workspace. The goal is to drive a robot from position $(0, 0)$ to its goal position $(20, 20)$. At time t, the robot state is denoted by $q_t = [p_{x,t}, p_{y,t}, \theta_t]^T$, where $q_{x,t}$ and $q_{y,t}$ are the robot's coordinates in the workspace and θ_t is the orientation of the robot. The control signal of the robot at time t is denoted by $u_t = [v_t, \omega_t]^T$, where v_t represents the linear velocity and ω_t denotes the angular velocity. The dynamics of the robot are subject to the discrete-time unicycle model [7], i.e.,

$$q_{t+1} = q_t + \Delta t \begin{bmatrix} \cos\theta_t & 0 \\ \sin\theta_t & 0 \\ 0 & 1 \end{bmatrix} u_t, \tag{8}$$

where Δt is the time interval. Consider also a group of humans in the set $\mathcal{H} = \mathcal{H}^c \cup \mathcal{H}^f$ randomly distributed in the collision-free workspace $\mathcal{C}$. Their locations are fixed but unknown to the agent. We assume that for every human $i \in \mathcal{H}^f$, the feedback function $f_i^f(\mathbf{x}) = 1$ if the robot trajectory passes through a ball of radius r_f centered at that human's position p_i. Otherwise, $f_i^f(\mathbf{x}) = 0$. Essentially, humans in the set $\mathcal{H}^f$ complain if the robot trajectory comes too close to them. On the other hand, if $i \in \mathcal{H}^c$, then $f_i^c(\mathbf{x}) = \min\{\text{floor}(\text{dist}(\mathbf{x}, p_i)/r_c), k\}$, where $\text{dist}(\mathbf{x}, p_i)$ is the distance from p_i to the trajectory $\mathbf{x}$ and we select $k = 5$. This feedback function models a $(k+1)$-level rating on how satisfied a human is about robot's trajectory. If the robot is within distance r_c from the human, a rate 0 is given. On the contrary, if the robot trajectory is far from human, a rate 5 is sent. Essentially, humans in $\mathcal{H}^c$ want the robot to come close. Note that no contextual information on the feedback functions f_i^f and f_i^c, e.g., the human positions p_i or ranges r_f and r_c, is available to our semi-supervised BO method. All that is known is the feedback functions' evaluations.

We used a fully connected neural network with sigmoid activation functions as our Encoder and Decoder. The unsupervised training set $\mathcal{D}_{un}$ was generated using RRT [16, 18]–[20] so that the routes are in the collision-free and dynamically-feasible space $\mathcal{C} \cap \mathcal{D}$. Since RRT does not involve the optimization of any utility, it generates trajectories that span the entire domain when sufficiently many samples are collected. In what follows, we compare our semi-supervised BO method in Algorithm 1 to a *supervised learning* approach where the trajectory is optimized using only the online data associated with human feedback, without training the autoencoder using the unlabeled dataset. We also compare our method to an *unsupervised learning* approach where the autoencoder is trained using the unlabeled dataset beforehand, and BO is run on the fixed latent space given by the trained autoencoder. In our simulations, $|\mathcal{H}^c| = 5$ and $|\mathcal{H}^f| = 20$. We parameterize the robot trajectories by 20 waypoints and, therefore, the dimension of the original planning problem is 40. Moreover, the latent dimensionality is set to be 5. The number of initial random samples is 5.

In Fig. 2 we study the effect of our proposed biased exploration method. The result is averaged over 100 runs. Specifically, we compare the performance of supervised, unsupervised and semi-supervised learning with and without

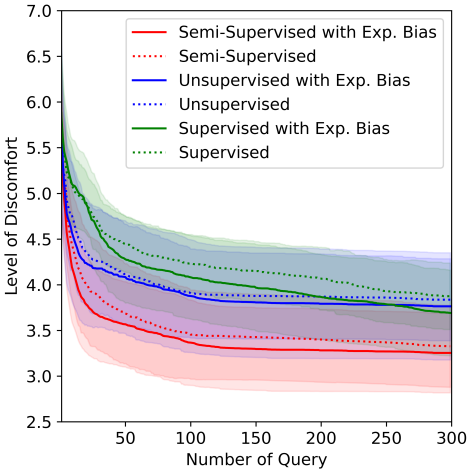

Fig. 2: Comparison of the human satisfaction $\mathcal{F}(m(\mathbf{x}))$ returned by supervised learning (green curves), unsupervised learning (blue curves), and semi-supervised learning (red curves). Solid curves represent implementation of these algorithms with exploration bias, while dashed curves are without exploration bias.

the exploration bias $\|\mathbf{x} - m(\mathbf{x})\|$. We observe that exploration bias improves the performance of all three learning methods compared to their counterparts without bias. The benefit of biased search in BO is more apparent in the case of supervised learning. This is because the latent space in the case of supervised learning is randomly initialized and, therefore, it is decoded into many trajectories that have collisions and are dynamically infeasible. In comparison, the latent spaces in the case of unsupervised and semi-supervised learning are trained using collision-free and dynamically feasible trajectories. Furthermore, we observe in Fig. 2 that semi-supervised learning (red curves) outperforms both supervised learning (green curves) and unsupervised learning (blue curves) with or without the exploration bias. Specifically, unsupervised learning gets trapped at suboptimal trajectories and is finally outperformed by the supervised approach. This is because unsupervised learning is initialized with a latent space trained using randomly generated collision-free and dynamically feasible trajectories, which accelerates the trajectory search in the beginning. However, since this latent space is not trained using all possible trajectories that satisfy the constraints and is not updated using online data, the final solution contains bias. On the other hand, the proposed semi-supervised learning method reduces the level of discomfort faster than the other two methods. The reason is that semi-supervised learning uses the same initial latent space as that used by the unsupervised method to achieve efficient exploration in the beginning, but it also updates this latent space using human feedback to remove the bias that is present in the supervised way. Therefore, the semi-supervised algorithm leads to faster convergence after around 10 queries, which is closely after processing the random samples. Note also that semi-supervised learning achieves better performance compared to the supervised and unsupervised methods using less than 30 queries for human feedback. This number is comparable to our recent zeroth-

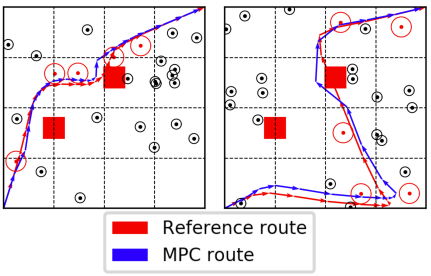

Fig. 3: Plots of the reference trajectories $\mathbf{x}$ (red line) and the actual robot trajectories $m(\mathbf{x})$ (blue line) returned by the MPC controller (4) for two different configurations of the humans in the workspace. The red circles represent humans in group $\mathcal{H}^c$ and the black circles represent humans in group $\mathcal{H}^f$.

order method [7] which, however, can not handle different types of human feedback and is prone to local minima, and much fewer than queries needed by typical RL methods like those discussed in Section I.

In Fig. 3, we present the reference trajectories $\mathbf{x}$ and the actual trajectories $m(\mathbf{x})$ followed by the robot for four different configurations of the humans $\mathcal{H}^c \cup \mathcal{H}^f$ in the workspace. These trajectories are returned using 50 rounds of human feedback. These trajectories satisfy the requirements of most humans except for very few cases due to the use of the trajectory length term in (5). As discussed in Section III.B, the objective in problem (5) strikes a balance between human satisfaction and trajectory length, which causes remaining complaints or poor ratings.

V. Conclusion

In this paper, we proposed a semi-supervised Bayesian Optimization (BO) method for human-in-the-loop motion planning that relies only on bandit human feedback, in the form of complaints or satisfaction ratings, that express how desirable a trajectory is without revealing the reason. We demonstrated the efficiency of our proposed trajectory planning method in a scenario with humans that have diversified and unknown demands and showed that it can find better solutions than competitive methods with very few queries for human feedback.

References

[1] S. Vyas, "Unmanned Secured Delivery System in International Journal on recent Innovation Trends in Computing and Communication, Volume 4,Issue 9, March 2016." *International Journal on recent Innovation Trends in Computing and Communication,*, vol. Volume 4, Mar. 2016.

[2] D. Bamburry, "Drones: Designed for Product Delivery," *Design Management Review*, vol. 26, no. 1, pp. 40–48, 2015, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/drev.10313. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/drev.10313

[3] M. Gombolay, X. Jessie Yang, B. Hayes, N. Seo, Z. Liu, S. Wadhwania, T. Yu, N. Shah, T. Golen, and J. Shah, "Robotic Assistance in Coordination of Patient Care," in *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016. [Online]. Available: http://www.roboticsproceedings.org/rss12/p26.pdf

[4] I. Mahrouche, "Robotic Nursing & Caring – Study Example : Robotic Nursing Assistant (RoNA) System," 2014, library Catalog: www.semanticscholar.org.

[5] C. Daniel, O. Kroemer, M. Viering, J. Metz, and J. Peters, "Active reward learning with a novel acquisition function," *Autonomous Robots*, vol. 39, no. 3, pp. 389–405, Oct. 2015. [Online]. Available: http://link.springer.com/10.1007/s10514-015-9454-z

[6] A. Menon, P. Kacker, and S. Chitta, "Towards a data-driven approach to human preferences in motion planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA: IEEE, May 2015, pp. 920–927. [Online]. Available: http://ieeexplore.ieee.org/document/7139287/

[7] X. Luo, Y. Zhang, and M. M. Zavlanos, "Socially-Aware Robot Planning via Bandit Human Feedback," *arXiv:2003.00658 [cs]*, Mar. 2020, arXiv: 2003.00658. [Online]. Available: http://arxiv.org/abs/2003.00658

[8] M. G. B. Blum, M. A. Nunes, D. Prangle, and S. A. Sisson, "A comparative review of dimension reduction methods in approximate bayesian computation," *Statistical Science*, vol. 28, no. 2, pp. 189–208, May 2013. [Online]. Available: http://dx.doi.org/10.1214/12-STS406

[9] K. Kandasamy, J. Schneider, and B. Poczos, "High Dimensional Bayesian Optimisation and Bandits via Additive Models," *arXiv:1503.01673 [cs, stat]*, May 2016, arXiv: 1503.01673. [Online]. Available: http://arxiv.org/abs/1503.01673

[10] E. Raponi, H. Wang, M. Bujny, S. Boria, and C. Doerr, "High Dimensional Bayesian Optimization Assisted by Principal Component Analysis," *arXiv:2007.00925 [cs]*, Jul. 2020, arXiv: 2007.00925. [Online]. Available: http://arxiv.org/abs/2007.00925

[11] R. Antonova, A. Rai, T. Li, and D. Kragic, "Bayesian Optimization in Variational Latent Spaces with Dynamic Compression," *arXiv:1907.04796 [cs]*, Jul. 2019, arXiv: 1907.04796. [Online]. Available: http://arxiv.org/abs/1907.04796

[12] R. Moriconi, M. P. Deisenroth, and K. S. S. Kumar, "High-dimensional Bayesian optimization using low-dimensional feature spaces," pp. 1–13, 2019. [Online]. Available: http://arxiv.org/abs/1902.10675

[13] J. M. Hernndez-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani, "A General Framework for Constrained Bayesian Optimization using Information-based Search," *Journal of Machine Learning Research*, vol. 17, no. 160, pp. 1–53, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-616.html

[14] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained Bayesian Optimization with Noisy Experiments," *Bayesian Analysis*, vol. 14, no. 2, pp. 495–519, Jun. 2019. [Online]. Available: https://projecteuclid.org/euclid.ba/1533866666

[15] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, May 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098135498003019

[16] Jongwoo Kim and J. P. Ostrowski, "Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, pp. 2200–2205 vol.2, iSSN: 1050-4729.

[17] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, ser. NIPS'93. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 3–10.

[18] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, Jun. 2015, arXiv: 1703.08944. [Online]. Available: http://arxiv.org/abs/1703.08944

[19] V. Vonásek, J. Faigl, T. Krajník, and L. Přeučil, "RRT-path – A Guided Rapidly Exploring Random Tree," in *Robot Motion and Control 2009*, ser. Lecture Notes in Control and Information Sciences, K. R. Kozłowski, Ed. London: Springer, 2009, pp. 307–316.

[20] Y. Abbasi-Yadkori, J. Modayil, and C. Szepesvari, "Extending rapidly-exploring random trees for asymptotically optimal anytime motion planning," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 127–132, iSSN: 2153-0866.