2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)
Würzburg, Germany, September 2-4, 2019

# Bayesian-Optimized Impedance Control of an Aerial Robot for Safe Physical Interaction with the Environment

Asem Khattab, Ramy Rashad, Johan B.C. Engelen and Stefano Stramigioli

***Abstract*— Impedance control is a widely used interaction-control technique for aerial and ground robots. To achieve consistent performance during impedance control tasks, an a-priori knowledge of the environment parameters is needed to adjust the controller's impedance parameters accordingly. Concentrating on tasks requiring constant impedance parameters throughout operation, a model-free learning framework is proposed to autonomously find the suitable parameters values. The framework relies on Bayesian optimization and episodic reward calculation requiring the drone to repeatedly perform a predetermined task in the environment actively searching in the impedance parameters space. The sample-efficiency and safety of learning were improved by adding two novel modifications to standard Bayesian optimization. The proposed technique was validated in a high fidelity simulation environment. The results show that the proposed framework is able to automatically find suitable impedance parameters values in different situations given the same initial knowledge and that the learned parameters values can be generalized to similar interaction tasks.**

## I. Introduction

In the past two decades, unmanned aerial vehicles (UAVs) have been successfully deployed for several applications in the civilian sector. These applications are mostly limited to passive tasks, which require the UAV to act as a flying sensor. However, in recent years, there has been an increasing interest in the robotics community to extend applications of UAVs to active tasks that require manipulation and physical interaction with the environment [1].

One of the widely used interaction control techniques, both for UAVs and fixed-based manipulators, is impedance control where the interaction behavior of the robot is controlled instead of controlling the position/force of the robot independently [2]. By not relying on model-based environments, impedance control is able to cope with contact discontinuities and guarantees stability with arbitrary passive environments.

In impedance control, the contact force between the UAV's end-effector and the environment is directly related to the controller's stiffness and damping parameters. However, achieving a consistent performance of the UAV during interaction tasks requires a-priori knowledge of the surface geometry and contact properties. This contradicts the main advantage of impedance control to effectively work in unstructured and unmodeled environments. In general, the geometric properties of the environment can be acquired by the robot through visual perception. As for the mechanical contact properties, such as stiffness and friction, one approach is to automatically change the impedance parameters online to improve the interaction behavior.

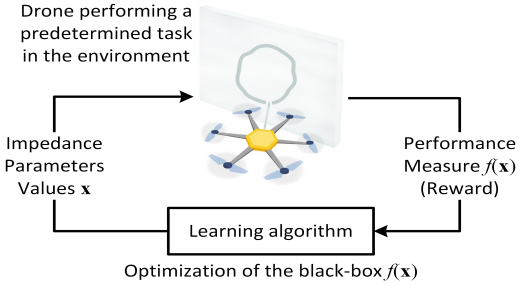



Fig. 1: Simple block diagram of the proposed learning problem formulation

In the literature, the approaches followed to automatically improve the impedance behavior of robots over time include adaptive control theory and machine learning. Adaptive impedance control has been demonstrated for rigid manipulators in [3] and for UAV-manipulator systems in [4]. Due to the approach's model-based nature, in order to guarantee the stability properties of the controller and design the adaptation laws, the environment was modeled in both [3], [4] as a linear spring, which only captures the steady state dynamics approximately. A supervised-learning approach has been demonstrated in [5] for medical robotic manipulators, which is irrelevant for UAV-based applications due to the requirement of human demonstrations. Unlike supervised approaches, reinforcement learning is more suited for general autonomous learning in unknown environments. In [6], [7], the $PI^2$ algorithm (policy improvement with path integrals) is used to learn impedance parameters trajectories for robots with high degrees-of-freedom doing tasks like flipping a light switch and opening a door. During such tasks, both the position trajectory and the impedance parameters trajectories are learned. While such methods have the advantage of being model-free and hence able to deal with different environments and unpredictable situations, they suffer from two main problems. The first one is that the learning is very specific to the task. Second, the state-action space is huge, which requires a large number of iterations for the learning to reach reasonable results.

This work has been funded by the cooperation program INTERREG Deutschland-Nederland as part of the SPECTORS project number 143081.

A. Khattab and J.B.C. Engelen were (previously), and R. Rashad is with the Robotics and Mechatronics group, University of Twente, Enschede, The Netherlands. Email: asem.khattab@gmail.com, jbc.engelen@gmail.com, r.a.m.rashadhashem@utwente.nl

S. Stramigioli is with the Robotics and Mechatronics group, University of Twente, and ITMO University, Saint Petersburg, Russia. Email: s.stramigioli@utwente.nl

978-1-7281-0778-3/19/$31.00 ©2019 IEEE

The focus in this work is on tasks requiring a fixed set of parameters values throughout operation. Examples of such tasks include surface contact tasks like grinding, cleaning, painting and writing. Considering such tasks, the optimal set of parameters values are greatly dependent on the combined mechanical properties of the surface and the UAV's end-effector. The aim is to perform efficient trial-and-error learning to automatically find the optimal parameters values. With reference to Fig. 1, the learning problem is formulated as the problem of finding the position of the global maximum of a black-box reward function $f(\boldsymbol{x})$ using as few evaluations as possible to minimize the learning cost, where the function being optimized represents a performance measure for the UAV under a selected set of impedance parameters values $\boldsymbol{x}$.

In this paper we propose a method that achieves safe and stable learning-based impedance controller for interactive aerial robots. Our approach consists of a model-based passive impedance controller that guarantees interaction stability, in addition to a data-driven learning algorithm that improves the closed loop performance over time. The learning algorithm uses the Bayesian optimization framework combined with Gaussian processes. The advantages of our approach compared to other techniques include: exploiting the system knowledge available, guarantying contact stability and enhancing sample efficiency. The issue of sample efficiency is crucial for aerial robots due to its very limited operation time. Another relevant issue is that the parameters space is not completely safe to explore. The safe learning is invoked in our approach by a novel modification to the standard technique of globally optimizing acquisition functions in Bayesian optimization. In the next section we discuss the details of our proposed method.

## II. Proposed Method

In this work, an episodic learning framework is proposed. At each episode, the drone receives a set of parameters values. The controller's impedance (i.e. parameters) is configured according to the received values and then the drone performs a task designed by the user to assess the performance of the chosen impedance settings. In our case, this task is following a prescribed trajectory (e.g. a circular trajectory). While performing the task, all relevant sensory measurements (from on-board sensors or off-board ones in the environment) are stored. At the end of the task, the stored sensors measurements are used to compute the reward (the performance measure) which can include different elements (e.g. the root-mean-square (RMS) of the position error) depending on the objective of the learning. This reward is sent back to the learning algorithm, which should intelligently select another set of parameters values to try in order to quickly find a satisfactory set of parameters values (i.e. ones that give a high reward).

The learning block in Fig 1 works on finding the position of the global maximum of the black-box reward function $\mathbf{x}^* = \arg\max f(\mathbf{x})$ where:

- $\mathbf{x}$ is a vector containing the selected impedance parameters values.
- $f(\mathbf{x})$ represents a performance measure of the drone in the interaction task under the selected impedance parameters values.

We propose in this work the use of the Bayesian optimization framework combined with Gaussian processes to optimize this expensive black-box function. In addition to being very sample-efficient, Bayesian optimization provides various schemes for effectively balancing exploration and exploitation as it searches for the position of the maximum. It also provides means for incorporating human expert knowledge or intuition, if available, in the learning.

From another perspective, our problem considered in this work can be formulated as a multi-dimensional continuum bandit problem with: 1) the multi-dimensional continuous action space corresponds to the impedance controller's parameters space, 2) the reward corresponds to the performance measure, 3) the state corresponds to the circumstance that determines the optimal action.

Continuum bandits are a class of reinforcement learning problems where the agent always stays in one state. Per our specification, as the interest is in a fixed set of parameters values throughout an interaction task, the optimal parameters values depend only on the environment and the task. Consequently, as long as the environment and the task are the same, the agent stays in one state throughout learning, and hence there is actually no need to formulate, estimate or detect the state, which is one of the main differences between the bandit problem and the full reinforcement learning problem.

Continuum bandit problems have been investigated in literature. However, most of the proposed approaches try to adapt solutions of the traditional discrete multi-armed bandit problem to continuum bandits [8]. This results in discretization of the action space and gives rise to the curse of dimensionality.

Bayesian optimization, as a solution to the continuum bandit problem can be then considered a reinforcement learning technique [9]. In contrast with the classical solutions of the continuum bandit problem, Bayesian optimization doesn't discretize the action space and hence is able to escape the curse of dimensionality and scale polynomially with the dimension of the action space.

## III. Bayesian Optimization Background

Bayesian optimization is a framework for sample-efficient search for the extremum of a black-box function. Bayesian optimization is particularly useful for situations where the target function has no closed-form expression, is difficult to get derivatives for, is very expensive to evaluate and/or is non-convex [9], [10].

Formally, the goal of Bayesian optimization is to find the position $\mathbf{x}^*$ of the global maximum (or minimum) of an unknown target function $f(\mathbf{x})$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\chi} f(\mathbf{x}) \tag{1}$$

where $\chi$ is the space of parameters.

Instead of relying on the density of evaluations, Bayesian optimization exploits all the evaluations it can get by building a surrogate model that represents the current belief about the black-box function. It uses this model to decide where to prop (evaluate) the function next. The decision of the next point to prop is carried out via *acquisition functions*, which provide a way to balance exploration and exploitation.

At the start, the user collects $w$ observations to form an initial surrogate model. In particular, the user selects points[1] $\mathbf{x}_{1:w}$ and samples the target function $f(\mathbf{x})$ at these points obtaining $y_{1:w}$, where $y_i = f(\mathbf{x}_i)+\epsilon_i$ is a (noisy) observation of the target function with noise $\epsilon_i$. The points and their corresponding observations combined in tuples are denoted $\mathcal{D}_{1:w}$ where $\mathcal{D}_i = (\mathbf{x}_i, y_i)$. Each iteration, a surrogate model is built, this model is used to find the best point to try next via acquisition functions. The selected point is sampled and the process continues. The procedures of Bayesian optimization on a simple 1D example are shown in Fig. 2 (a-c).

The surrogate model used in the presented work is a Gaussian Process $\mathcal{GP}(\mu_0, k)$; a non-parametric model, completely specified by a mean function $\mu_0(\mathbf{x})$ and and a *kernel* (covariance function) $k(\mathbf{x}, \mathbf{x}')$. In most cases, as well as the case in our work, the mean is chosen to be a fixed function, generally equal to zero [10].

Say we received $n$ (noisy) observations $\mathcal{D}_{1:n}$ of the target function. Then the posterior of the random variable $f(\mathbf{x})$ at an arbitrary point $\mathbf{x}$ given the observations $\mathcal{D}_{1:n}$ is normally distributed with the following mean and variance functions:

$$f(\mathbf{x}) \mid \mathcal{D}_{1:n} \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x})) \tag{2}$$

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}) \tag{3}$$

$$\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) + \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}) \tag{4}$$

where the mean vector $\mathbf{m}$ and the positive semi-definite covariance matrix $\mathbf{K}$ are defined elementwisely as $m_i := \mu_0(\mathbf{x}_i)$ and $K_{i,j} := k(\mathbf{x}_i, \mathbf{x}_j)$ respectively, and $\sigma^2$ is the given variance of the Gaussian noise. Further, $\mathbf{k}(\mathbf{x})$ represents a vector of covariance values between the arbitrary point $\mathbf{x}$ and the previous observations points $\mathbf{x}_{1:n}$.

The kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ selected to be used with the Gaussian process is the radial basis function (RBF) kernel expressed as:

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}[(\mathbf{x} - \mathbf{x}')^T\boldsymbol{\theta}(\mathbf{x} - \mathbf{x}')]\right) \tag{5}$$

where $\boldsymbol{\theta}$ is a diagonal matrix of $\dim(\mathbf{x})$ squared length scales $\theta_i^2$. Through the choice of hyperparameters $\boldsymbol{\theta}$, one can control how rapid are the changes in the functions predicted by the GP prior. Typically, the values of the hyperparameters are learned using the observations that were obtained so far through seeding random values and maximizing the marginal likelihood of the fitted GP model [10], [11].

As for the acquisition function, its role is to balance exploration and exploitation by providing a measure for the utility of obtaining a new observation at the point $\mathbf{x}_{n+1}$ based on the surrogate model built using the past observations $\mathcal{D}_{1:n}$. The next point to sample is then obtained by optimizing the acquisition function:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\chi} \alpha(\mathbf{x} \mid \mathcal{D}_{1:n}) \tag{6}$$

Different acquisition functions lead to different exploration/exploitation trade-offs and behaviors that are suitable for certain applications but not suitable for others. For this work, the Probability of Improvement (PI) function is used [9]. This function assesses, for a certain point $\mathbf{x}$, the probability of finding a target function value (or reward) that is higher than the maximum found observation by the hyperparameter $\xi$:

$$PI(\mathbf{x}) = P(f(\mathbf{x}) \geq \max(y_{1:n}) + \xi) \tag{7}$$

$$= \Phi\left(\frac{\mu_n(\mathbf{x}) - \max(y_{1:n}) - \xi}{\sigma_n(\mathbf{x})}\right) \tag{8}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. Maximizing the $PI$ function results in selecting points most likely to offer an improvement of at least $\xi$. This allows $\xi$ to affect how local/global the search is [10].

## IV. Local Optimization of Acquisition function

Even though the established theoretical proofs of the convergence of Bayesian optimization are only valid given that the position of the true global maximum of the acquisition function is found every iteration and is selected as the next point to observe the target black-box function at, in practice it is nearly impossible to ensure this condition [9]. In this work we give up the quest for finding this global maximum for a more local way of optimizing. We claim that the presented approach is more safe and efficient, especially for black-box functions related to the aerial robotic application at hand.

Working with robotic systems, using explorative acquisition functions is risky as the search is more probable to fall into areas of low rewards (corresponding to bad performance or even instability). While very local/exploitative settings do not suffer from this issue, they are more probable to get stuck in a local maximum. Instead of trying to find the global maximum of the acquisition function, only areas close to the positions of the already seen observations are considered in our approach. Specifically, at each iteration $n$ and for all $i \leq n$, a hyper-rectangle $B_i$ is virtually placed centered around each point $\mathbf{x}_i$ with dimensions equal to that of the domain of the target function scaled down by the hyperparameter $d < 1.0$ (distance).

The next point to sample is then selected as the position of the maximum of the acquisition function within hyper-rectangles $B_{1:n}$:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\bigcup_{i=1}^{n} B_i} \alpha(\mathbf{x} \mid \mathcal{D}_{1:n}) \tag{9}$$

The hyper-rectangle $B_i$ is formed as follows. Suppose $\chi$ the domain of the black-box function is given as a two-columns matrix such that row $j$ represents the lower and

[1] In this paper, we use the notation $x_{i:j} = \{x_i, x_{i+1}, \ldots, x_j\}$.

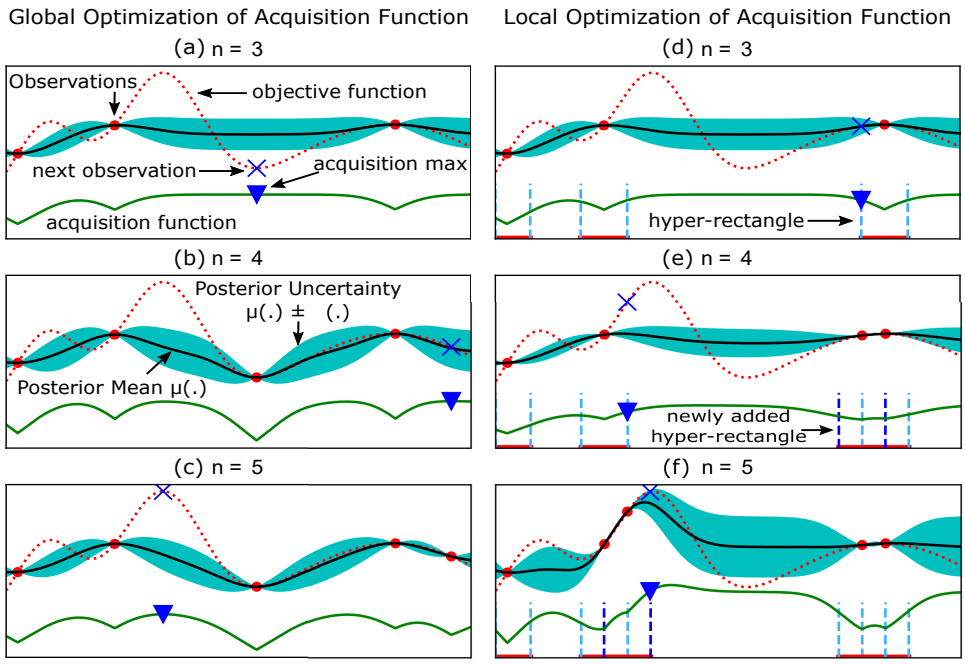



Fig. 2: Behavior of Bayesian optimization with global optimization of the acquisition function compared to that with the proposed local optimization on a simple 1D Example. The meaning of the different elements of the subfigures are indicated in the graphs. Theses meanings are the same across all subfigures.

upper limit of the input of the function along dimension $j$:

$$\chi^{j,1} \leq x^j \leq \chi^{j,2}$$

where superscripts here are used as row-based matrix indices and $x^j$ is the component of the input $\mathbf{x}$ along dimension $j$. Then $B_i$ is formed also as a two-columns matrix such that its row $j$ is:

$$B_i^{j,1:2} = [\max(\chi^{j,1}, x_i^j - \frac{D_j}{2}), \min(x_i^j + \frac{D_j}{2}, \chi^{j,2})] \quad (10)$$

where $D_j = d(\chi^{j,2} - \chi^{j,1})$ is the scaled distance along dimension $j$. The max and min operations above are used to ensure that the hyper-rectangles do not go outside the domain of the target function.

At each iteration of Bayesian optimization, a new hyper-rectangle is added as the number of obtained observations increases. Then the next point is chosen by optimizing the acquisition function in the areas within all the hyper-rectangles since the acquisition function within them changes its shape as the surrogate model evolves.

The idea is illustrated in Fig. 2 where the behavior of Bayesian optimization with global optimization of the acquisition function is compared to that with the proposed local optimization on a simple 1D Example. Both Bayesian optimization runs started with the same three initial observations (red dots in figures 2a and 2d) and are using the same UCB acquisition function. In the first iteration (upper row), the location of the global maximum of the acquisition function (Fig. 2a) offers a low value of the objective function, while with the locally optimized acquisition function (Fig. 2d), the search was able to avoid falling into this area of low reward by moving only in small steps.

With the proposed local optimization, at each new iteration (figures 2e and 2f), a new 1D hyper-rectangle is added intersecting with an existent one. The acquisition function is optimized within all hyper-rectangle to select the position of the next observation. This allows local optimization to get early warnings along paths leading to low rewards and to quickly find paths leading to higher rewards. After three iterations (bottom row), both global optimization and local optimization searches are about to find the global maximum. However, local optimization got higher rewards along the way. This can be seen by observing that, at each row, the value of the next observation (blue cross) resulting from local optimization (right graph) is higher than that resulting from global optimization (left graph).

## V. Simulation Setup

The simulations done in this work are centered around a fully-actuated hexarotor UAV [12] used for a task involving physical interaction with the environment. Simulations will be done for a surface sliding task. In this simple learning task, the drone learns the optimal impedance parameters for drawing a certain shape on a wall. The aim is to provide a clear proof-of-concept for using the proposed learning framework to find impedance parameters values. In this section, the task description is provided along with the choice of reward function, and the software implementation details for the Bayesian optimization algorithm.

### A. Sliding Task Description

For the learning of the impedance parameters values for drawing a certain shape on a wall, the predetermined task chosen to be done at each episode is simply following a trajectory to draw the shape on the wall. The end-effector of the drone is commanded to follow a trajectory while exerting a normal force on the surface using the impedance controller.

To investigate the ability of the proposed learning framework to find the suitable impedance values as the task changes, two different kinds of changes are included in the simulation. First, two different shapes (trajectories) will be used for training; a square and a circle. Second, surfaces of different materials will be put in the simulations. The different surface materials are simulated by changing the friction coefficient ($\mu$) between the end-effector of the UAV and the wall. Four different coefficients are used: 0.4, 0.8, 1.16, and 1.4.

### B. Reward Shaping

In the learning problem at hand, we consider a simple goal: maximizing the accuracy of drawing while minimizing the used impedance gains. Excessively large gains are associated with higher power consumption and more aggressive interaction behavior which is highly discouraged. However, higher gains also increase the accuracy of tracking the setpoint, which means that maximizing the accuracy actually contradicts minimizing the impedance gains. The reward function should manage this conflict.

The used reward function for this task, in terms of the impedance parameters values $\mathbf{x}$, is defined:

$$f(\mathbf{x}) = -c_1 E_t - c_2 E_r - c_3 \frac{\text{sum}(\mathbf{x})}{\dim(\mathbf{x})} \tag{11}$$

where $E_t = E_t^x + E_t^y + E_t^z$ is the sum of the translational root-mean-square errors along the three axes, $E_r = E_r^x + E_r^y + E_r^z$ is similarly the sum of the rotational errors, $\dim(\bullet)$ is the number of elements in a vector, and $c_1$, $c_2$ and $c_3$ are weighting factors.

For this reward function, as it is typically the case for many reward functions seen in literature, the units of the individual parts of the function are different. So, the definition of the function has no direct physical meaning. The weights have to balance the difference in units and set the priorities of optimization. For all simulations presented in this paper, $c_1 = 100$, $c_2 = 150$ and $c_3 = 0.15$.

The range of this suggested reward function is very difficult to predict in advance. This is inconvenient when using Bayesian optimization. For instance, the meaning the hyperparameter $\xi$ of the PI acquisition function (7) is directly related to the range of the optimized function. So, the same value for $\xi$ can lead to explorative behavior for one reward function but an exploitative behavior for another one with a much larger range. The same problem arises with the noise hyperparameter $\sigma^2$.

To solve this issue, we suggest limiting the range of the target function between zero and one using the Sigmoid logistic function defined:

$$S(x) = \frac{1}{1 + e^{-l(x-m)}} \tag{12}$$

where $m$ is the location of the midpoint of the function and $l$ is the steepness of the curve. Logistic functions have been used in different areas in machine learning. Most notably, in logistic regression and as the "activation" function for neural networks.

The midpoint $m$ of the Sigmoid function is selected by using the initial observations. Specifically, the maximum received reward in the initial points is considered the midpoint of the used logistic function. Consequently, the standard Bayesian optimization algorithm is modified with the highlighted modifications in Algorithm 1.

**Algorithm 1:** Bayesian Optimization with a Logistic Function

Get $w$ initial observations $\mathcal{D}_{1:w} = \{(\mathbf{x}_i, y_i)\}_{i=0}^{w}$
Calculate the midpoint of the logistic function $m = \max(y_{1:w})$.
The logistic function is then: $S(x) = 1/\left[1 + e^{-l(x-m)}\right]$
Transform the initial observations: $\mathcal{D}_{1:w} = \{(\mathbf{x}_i, S(y_i))\}_{i=0}^{w}$
**for** $n = w, w+1, w+2...$ **do**
    Build/update the surrogate statistical model $M(S(f(\mathbf{x}))|\mathcal{D}_{1:n})$
    Select $\mathbf{x}_{n+1}$ by optimizing the acquisition function $\alpha(\mathbf{x}|\mathcal{D}_{1:n})$
    Obtain a (noisy) new observation at $\mathbf{x}_{n+1}$: $y_{n+1} = f(\mathbf{x}_{n+1}) + \epsilon_{n+1}$
    Augment the data $\mathcal{D}_{1:n+1} = \{\mathcal{D}_{1:n}, (\mathbf{x}_{n+1}, S(y_{n+1}))\}$

### C. Implementation Details

The simulation environment used consisted of a simulated hexarotor UAV equipped with an impedance controller (details can be found in [12]). The simulation environment is built over `ROS` (Robotics Operating System) via `Gazebo` and `RotorS` simulators [13].

The learning algorithm was implemented by modifying the `BayesianOptimization` package [14] to add the possibility of forming the hyper-rectangles and optimizing the acquisition function within them. For limited page space, the reader is referred to [15].

Finally, Table I summarizes the selected values for the discussed settings in this section.

## VI. Simulation Results and Discussion

In this section, the simulation results of an agent learning four impedance parameters are shown. In all simulations, 50 observations were used of which 10 are initial observations and 40 are Bayesian optimization iterations. The used initial observations were the same for all simulations presented.

TABLE I: Summary of the used learning settings in the simulations.

| Setting | Notation | Selected Value |
|---|---|---|
| Acquisition Function | | PI |
| PI hyperparameter | $\xi$ | 0.01 |
| Domain of $f(\mathbf{x})$ along a dimension $i$ | $\chi_i$ | $(0.01, 50)$ |
| distance in local optimization | $d$ | 0.05 |
| GP noise hyperparameter | $\sigma^2$ | 0.0001 |
| Logistic function steepness | $l$ | 0.5 |

In the following discussions, we define $S_n^+$ the maximum obtained value of the target function after $n$ observations

$$S_n^+ = \max S(y_{1:n}) \tag{13}$$

which is known in literature as the *learning rate*. Similarly, $y_n^+ = \max y_{1:n}$ is the corresponding maximum of the noisy observations of the reward function defined in (11). We also define the *average of the received rewards* after $n$ observations as:

$$\bar{y}_n = \sum_{i=0}^{n} y/n \tag{14}$$

### A. Drawing a circle trajectory

Fig. 3 shows the learning rates (top) and the corresponding values of the maximum seen rewards (bottom) of the circle problem with different friction coefficient values. Most of the improvement happened in the first 15 Bayesian optimization iterations after the 10 initial observations. However, the amount of improvement increases as the friction coefficient increases.

Looking to the bottom graph in Fig. 3, it can be confirmed that, the reason for the improvement trend is that the initial observations produce better rewards for smoother surfaces. At the end of the 40 Bayesian optimization iterations, the maximum found reward converges to almost the same value of $-7.6$ for all values of $\mu$.

### B. Drawing a square trajectory

Having trained the drone to find suitable impedance parameters values for drawing circles with different frictions, it is worth investigating if the learned values are able to provide satisfactory performance for tasks involving drawing another trajectory with different smoothness properties.

*Using the Parameters Values Learned for the Circle:* Before training on the square trajectory, the impedance parameters values found in the circle learning were applied to the square task (denoted by □) with the respective friction coefficient values as in Table II. The previously shown results of using these parameters values in the circle tasks (denoted by ○) are also included in the table for comparability. It can be seen that the values produced lower rewards and higher translational errors in square task than in circle task. Note that a higher translational error (and hence a lower reward) is expected for square tasks because of the involved discontinuities in the trajectory. However, it is still unclear form that table if the difference in rewards is completely due to the added difficulty or due to the fact that square tasks need completely different values for the impedance parameters.

TABLE II: Learned parameters values for the circle task applied to the square task.

| $\mu$ | Shape | $y^+$ | $E_t$[cm] | $\bar{\mathbf{x}}$ |
|---|---|---|---|---|
| 0.40 | ○ | -7.52 | 1.83 | 7.89 |
| | □ | -8.74 | 2.47 | |
| 0.80 | ○ | -7.66 | 1.74 | 8.70 |
| | □ | -8.96 | 2.23 | |
| 1.16 | ○ | -7.62 | 1.73 | 8.30 |
| | □ | -8.91 | 2.24 | |
| 1.40 | ○ | -7.56 | 1.72 | 7.98 |
| | □ | -8.90 | 2.28 | |

*Square Learning Results:* To reach a conclusion about the aforementioned point, Bayesian optimization was run to find the suitable impedance parameters for the square problem. The same ten locations of the initial observations used in the circle task were used here. Figure 4 shows the learning curves. It can be seen from the top graph (showing the maximum seen value of the target function $S_n^+$) that the amounts of improvement exhibit a similar trend to these seen in the circle case.

Looking to the bottom graph in Fig. 4, it can be seen also that the maximum seen reward within the initial observations is higher for the two highest values of $\mu$ and lower for the two lowest values. At the end, the maximum found reward converges to almost the same value of around $-8.80$ for all values of $\mu$. This is more than one unit lower than the maximum rewards reached in the circle tasks. These differences between the square learning and the circle learning can be attributed to the increased difficulty in the square tasks (due to discontinuities) decreasing the room for improvement and limiting the maximum reward that can be found.

*Using the Parameters Values Learned for the Square:* To see how the performance obtained by the impedance parameters values learned for the square tasks differs from that of the values learned for the circle tasks, table III shows the results of using the learned parameters values of the square tasks on the circle and square tasks. Comparing these results with the ones shown in Table II, important conclusions can be drawn. Generally, it can be seen that the rewards resulting from impedance values obtained by training on a certain shape are greater than those resulting from training on the other shape. However, the difference between the two rewards is mostly small. This means that training for drawing one shape might be enough to find parameters values suitable for drawing different shapes.

The results in tables II and III show one of the key advantages of the problem formulation suggested in the presented work over other formulations in literature [6], [7] that concern learning a *trajectory* of impedance parameters values for a certain task. A learned impedance values trajectory for drawing a square, for example, cannot be used for drawing a circle. With the suggested method, training does not have to be repeated for similar tasks provided that the environment is the same.

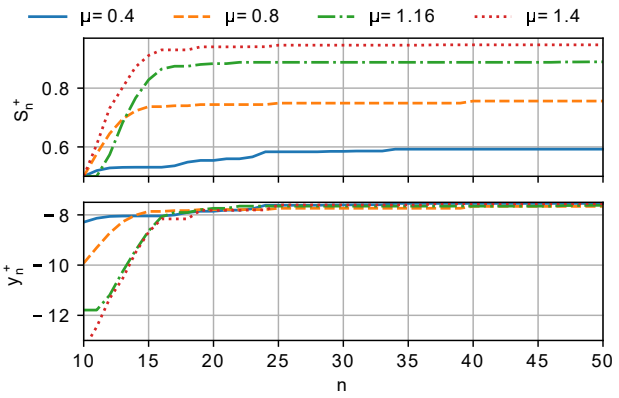



Fig. 3: Learning curves of the circle problems.

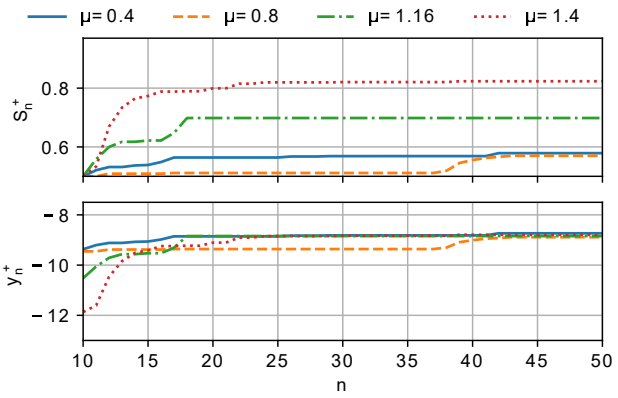



Fig. 4: Learning curves of the square problems.

TABLE III: Learned parameters values for the square tasks applied to the circle and square tasks.

| $\mu$ | Shape | $y^+$ | $E_t$[cm] | $\bar{\mathbf{x}}$ |
|---|---|---|---|---|
| 0.40 | ○ | -7.84 | 1.80 | 9.54 |
| | □ | -8.73 | 2.20 | |
| 0.80 | ○ | -7.71 | 1.72 | 9.23 |
| | □ | -8.89 | 2.19 | |
| 1.16 | ○ | -8.09 | 2.00 | 9.91 |
| | □ | -8.84 | 2.19 | |
| 1.40 | ○ | -7.70 | 1.72 | 8.96 |
| | □ | -8.79 | 2.15 | |

### C. Learning without the Proposed Modifications

All the simulations presented in this section so far were done with the two novel modifications proposed in sections IV and V-B. In this subsection, results are shown confirming the significance of these modifications when learning impedance parameters for an aerial robotic system.

*1) The Effect of Global Optimization:* Figure 5 compares the learning rates ($y_n^+$ curves) and the average received rewards ($\bar{y}_n$ curves) of the circle problems with a globally optimized acquisition function to the already seen results with a locally optimized acquisition function as suggested in section IV. Again, for the simulations done with a globally optimized acquisition function, the same settings and initial points offered for the other simulations, were used (including using a logistic function) except that PI was optimized globally with the same default global optimization method.

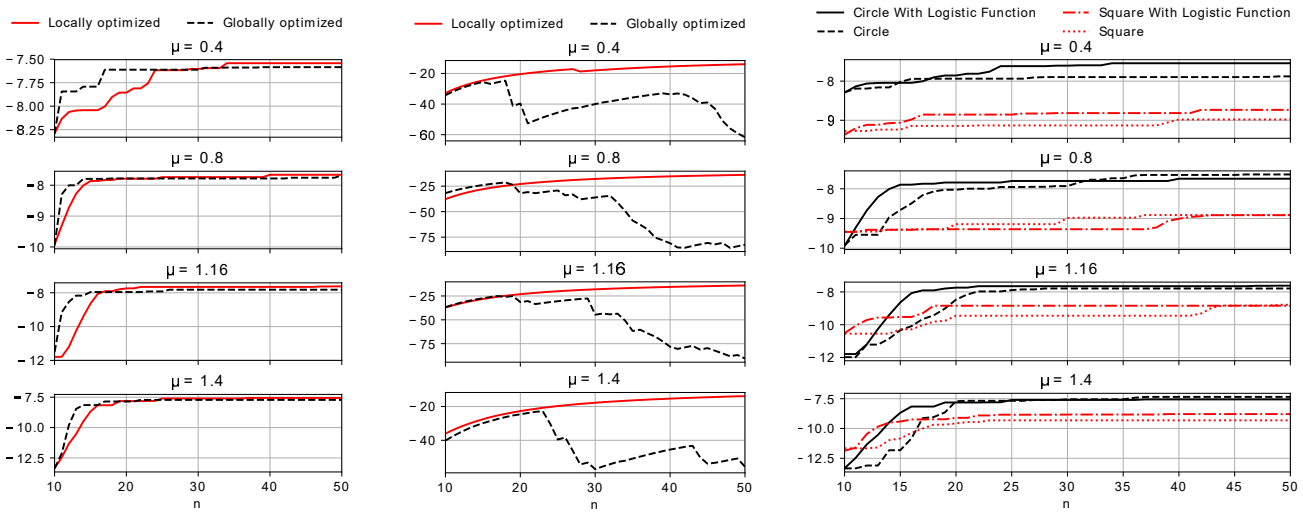
Looking at the learning rates ($y_n^+$ curves), it can be seen that learning with a globally optimized PI was able to reach high rewards quicker than it is with a local optimization. However, in all learning problems, local optimization of the acquisition function was able to find higher rewards at the end. This is because, for a globally optimized PI, setting the hyperparameter $\xi$ to 0.05 results in a more explorative behavior allowing quicker finding of places with higher rewards. However, this increased exploration resulted in the search being unable to fine-tune and reach the locations of the highest rewards. Local optimization, while still benefiting from the exploration by setting $\xi$ to 0.05, is able reach higher rewards slightly slower than global optimization.

What is more significant about local optimization is not its ability to find higher rewards, but its ability to make the learning much safer. Observing the averages of the received rewards during the different learning problems ($\bar{y}_n$ curves), it can be clearly seen that the search with local optimization almost always obtains rewards that are equal or higher than the already seen rewards ($\bar{y}_n$ are always increasing except for a minor exception for the learning problem with $\mu = 0.4$). This indicates that the search is safe as it doesn't fall in areas in the impedance parameters space with low rewards (corresponding to very unsatisfactory and dangerous performance or instability). This is because the search with local optimization is able to receive *early warnings* along paths leading to low rewards instead of jumping straight to them. On the other hand, the search with global optimization falls into areas with worse rewards than already obtained most of the time of the training, which indicates that the learning is very unsafe. Unless there is a very fast way of detecting failures, this training is disastrous if conducted with a physical UAV in the real world.

*2) The Effect of Not Using the Logistic Function:* Figure 5c compares the learning curves for the circle and square tasks without using a logistic function to the already seen learning curves obtained with using a logistic function as suggested in Algorithm 1.

In the simulations done without using the logistic function modification, the same locations for the initial observations were provided and the same learning settings were used except that Bayesian optimization was performed directly on the received rewards without inputting them to a logistic function as described in section V-B.

It can be seen form Fig. 5c that, in almost all cases, using the logistic function modification resulted in reaching higher rewards in lower numbers of iterations confirming the statements made in section V-B. Few exceptions exist. For the circle tasks with $\mu = 0.8$ and $\mu = 1.16$, using the logistic function modification resulted in getting higher rewards quicker. However, towards the end of the training (40 Bayesian optimization iterations), the learning curves




(a) Learning rates $y_n^+$ (b) Average received rewards $\bar{y}_n$ (c) Learning rates $y_n^+$

Fig. 5: Comparison of Learning with and without the Proposed Modifications. Fig. (a) and (b) shows effect of locally optimized (solid) vs. globally optimized (dashed) acquisition function for the circle problems. Fig. (c) shows effect of using the logistic function for the circle and square problems.

without using the logistic function could reach a slightly higher point in a way that doesn't affect the general trend. Also, for the square task with $\mu = 0.8$ the learning curve is faster without using the logistic function, although reaching the same point at the end of the training. The reason why the learning curve was faster in this only case is that the range of the reward function in that case was suitable for the selected learning settings. However, as the range of the reward function changes from one situation to another, Bayesian optimization without using the logistic function was not able to maintain the same performance.

These results clearly show that the suggested logistic function modification allowed Bayesian optimization to generalize the meaning of learning settings among black-box reward functions of different natures.

## VII. Conclusion

In this presented work, the concentration was on drone interaction tasks requiring a fixed set of impedance parameters. This allowed formulating the problem as an optimization of an expensive black-box function. A solution was then presented relying on Bayesian optimization with novel modifications significantly improving safety and sample-efficiency of learning as was demonstrated in the simulation results. Future work includes experimental results on a drone as well as applying the technique on different robots and for different interaction tasks.

## References


[1] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial Manipulation: A Literature Review," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1957–1964, jul 2018.

[2] L. Villani and J. De Schutter, "Force control," in *Springer handbook of robotics*. Springer, 2016, pp. 195–220.

[3] J. Duan, Y. Gan, M. Chen, and X. Dai, "Adaptive variable impedance control for dynamic contact force tracking in uncertain environment," *Robotics and Autonomous Systems*, vol. 102, pp. 54–65, 2018.

[4] M. Car, A. Ivanovic, M. Orsag, and S. Bogdan, "Position-based adaptive impedance control for a uav," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 957–963.

[5] N. Aghasadeghi, H. Zhao, L. J. Hargrove, A. D. Ames, E. J. Perreault, and T. Bretl, "Learning impedance controller parameters for lower-limb prostheses," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4268–4274.

[6] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research (ijrr)*, vol. 30, no. 7, pp. 820–833, 2011.

[7] F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. Theodorou, and S. Schaal, "Model-free reinforcement learning of impedance control in stochastic environments," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 4, pp. 330–341, 2012.

[8] O. Kroemer and J. Peters, "Active exploration for robot parameter selection in episodic reinforcement learning," in *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*. IEEE, 2011, pp. 25–31.

[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, Jan 2016.

[10] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *ArXiv e-prints*, Dec. 2010.

[11] D. Duvenaud, "Automatic model construction with gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[12] R. Rashad, J. B. Engelen, and S. Stramigioli, "Energy tank-based wrench/impedance control of a fully-actuated hexarotor: A geometric port-hamiltonian approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[13] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors - a modular gazebo mav simulator framework," in *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[14] Fernando Nogueira, "Bayesianoptimization, a python implementation of global optimization with gaussian processes." https://github.com/fmfn/BayesianOptimization, 2018, [Online; accessed 6-June-2018].

[15] A. Khattab, "Towards an interactive drone, a bayesian optimization approach," Master's thesis, University of Twente, 2018.