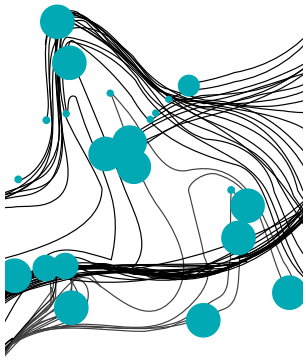
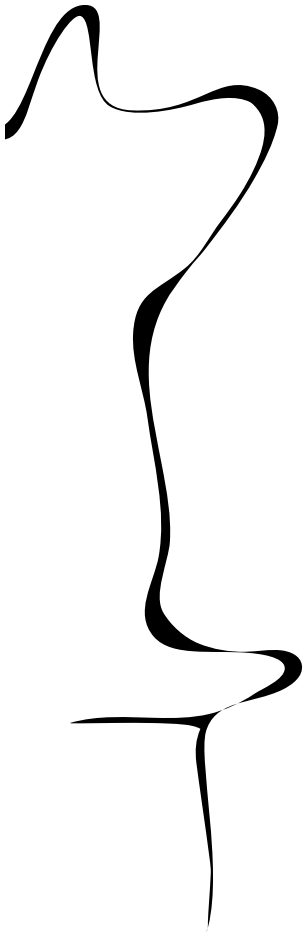


RAM ROBOTICS AND MECHATRONICS



BAYESIAN LEARNING-BASED IMPEDANCE CONTROL OF AN AERIAL ROBOT FOR WRITING

V.S.Y. (Yashwanth) Avvari

MSC ASSIGNMENT

Committee:
prof. dr. ir. G.J.M. Krijnen
dr. ir. R.A.M. Rashad Hashem
ir. A.N.M.G. Afifi
prof. dr. ir. A. Franchi
dr. ing. K.H. Chen

July, 2022

035RaM2022
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Contents

Summary

In recent years there has been an exploding interest in extending the current applications of multirotor UAVs to those that require aerial physical interactions, such as contact-based inspection, aerial writing, and tool handling on hard-to-reach surfaces. Impedance control is a widely used interaction-control technique for aerial and ground robots. To achieve consistent performance during the interaction tasks, an a-priori knowledge of the environment parameters is needed to adjust the impedance controller parameters accordingly. For the task of aerial writing on unknown surfaces, this unknown knowledge of the environment makes it challenging to achieve a consistent outcome for the interaction tasks. Therefore, a novel method of finding optimal impedance controller parameters for interacting unknown surfaces is proposed in this thesis.

The proposed method has two parts - use of pre-trained neural networks to predict the optimal controller parameters and generation of a custom dataset to train the neural networks. The work of this thesis is on the latter and involves a framework based on Bayesian Optimization (BO) to find the optimal parameters of an impedance-controlled aerial robot. Bayesian optimization is an iterative method of finding the maximum(or minimum) of an expensive black-box target function. A novel reward function is designed that depicts the accuracy and smoothness of the writing task by the aerial robot. Bayesian optimization is used to find the maximum of the target function formed by the impedance controller parameters and the rewards generated by the reward function. The corresponding controller parameters at the maximum reward are considered as optimal parameters. This is backed up by several simulations showing better accuracy in writing tasks with the optimal parameters predicted by the Bayesian optimization.

Since the real-world robotic experiments are extremely dangerous and costly, a virtual aerial robot with same specifications as that of a real-world robot is created in the Gazebo simulator. Using the techniques of Sim2Real transfer learning such as domain randomization, various simulation scenarios are created by varying the parameters of virtual environment and noise added to the simulation. Bayesian optimization is used for every such simulation scenario and the optimal controller parameters are collected to form the custom dataset. In future, this dataset will be used for training the neural networks so that it can predict the required optimal controller parameters when the real-world aerial robot performs a similar writing task on an unknown surface.

Chapter 1

Introduction

Originally designed for military operations, Unmanned Aerial Vehicles(UAVs) or drones have grown in popularity over the past few years. They are used in almost every field of technology and are being developed in all shapes and sizes. Affordable pricing of the consumer drones have gained a lot of interest among the young and are being used for recreational activities. They also became the starting point for the young enthusiasts who want to make a career in aerial photography. In an effort to capture new vantage points of the scenes, drones in film and videography industries are replacing the crane cameras, tracks and other traditional film making tools that usually come with a set of physical restrictions. Since the travel time by air is far less than that of land, drones are being employed in various cargo delivering services[1]. They are being used by fire department for routine monitoring[2] over the forests and mountains for any abnormalities and if required can even be equipped with water jets to put out fire at higher grounds. Transportation department use drones in traffic management and surveying the CO2 emissions[3] in the environment. Police departments are also using the drones to keep the suspected areas under surveillance and for various search and rescue operations[4]. Even agriculture field has started employing drones to study the plant life using imaging and vision techniques and help in spraying harmful pesticides[5] on to the crops. With the help of vision techniques, drones are used in mapping the terrains of various mountains and caves[6] where the environment is hazardous for human involvement. Drones are also being equipped with infrared cameras taking advantage of thermal images[7] to capture the details of bridges in low light conditions.

The scope for the applications of the drones is very wide[8]. Along with tasks like hovering and flying with payloads, visual monitoring or even grasping objects, drones are also being employed in complex aerial physical interaction tasks which is the focus of this thesis. These are very challenging tasks due to the complex aerodynamics[9] involved when the drone is close to the surface it is interacting with. With the latest developments, UAVs are able to execute a number of interaction tasks by exploiting the tilt rotor configuration of a hexarotor[10] or by using interconnection and damping assignment–passivity-based control(IDA-PBC) on a quadrotor[11]. The real world applications of aerial physical interactions are the tasks like inspection on high ground surfaces like windmill blades, radio towers, cleaning the windows of tall buildings, painting on walls, welding on high surfaces, inspecting tall industry chimneys, etc. These are the applications where human deployment is not preferable due to the dangerous working conditions. Even the deployment of land robots is not always possible due to extreme high ground situations. Therefore the employment of UAVs for these contact-based inspection jobs would be most beneficial as they bring efficiency and ability to reach places that humans cannot reach easily.

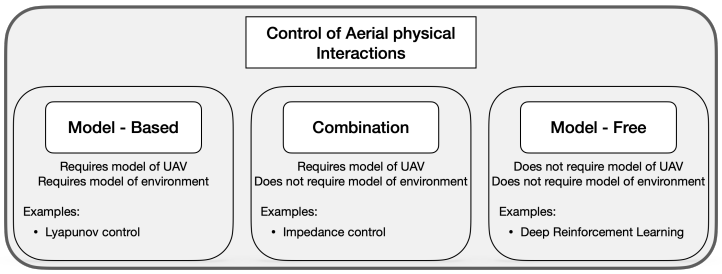


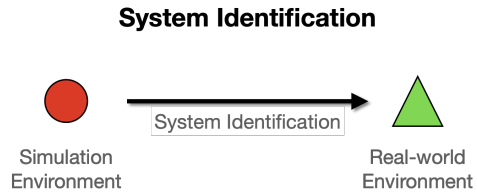
Figure 1.1: Block diagram showing various control techniques for aerial physical interactions

Depending on the task at hand, a model-based or model-free approach is usually chosen for designing a controller in-order to achieve successful aerial physical interactions with the environment. As illustrated in Figure 1.1, Lyapunov-based controllers can be model-based and depend on the model of the UAV as well as the model of the environment. Aerial physical interactions can also be achieved using the techniques of reinforcement learning(RL) for a model-free approach of tuning the controller parameters. The model of the UAV is obtainable either from the configuration of the UAV or through a series of experiments or directly from the vendor of the (commercially available)UAVs. However, it is not always the case that the model of the environment(target surface) UAV is supposed to interact with is known. For example when an UAV is deployed in an interaction task of inspecting a very old industrial chimney, it is most probably safe to assume that the mathematical model of the chimney is unknown. Therefore, an impedance control is suitable for the task of aerial writing on unknown surfaces. Although the model of the UAV is required, the impedance control does not need to know the model of the environment. Due to this, it has the challenge of tuning the controller parameters in-order to achieve a consistent performance.

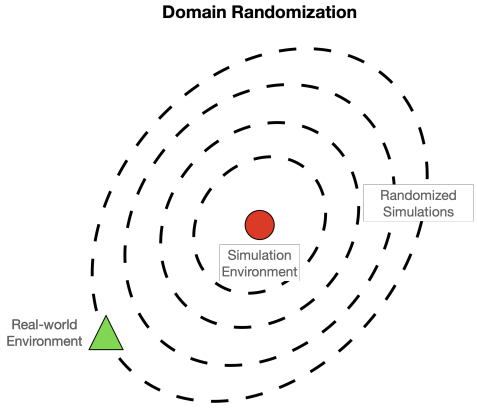
One-way of tuning the parameters of impedance controller is to use reinforcement learning and fine-tune the parameters through trial-and-errors. This come with a lot of safety concerns as there can be situations where the chosen controller parameters might result in harmful behaviours of the UAV and can cause damage to the environment. UAVs may also experience wear and tear due to repeated interactions with the environment and prolonged flight times leading to incorrect and unstable results. This real-world data is hard to collect and on top of that it is notoriously costly - financially, computationally and in terms of time and effort as well. On the other hand, virtual robots in a simulated world do not need to care about any of these concerns. Simulations may also act as a source for potentially infinite data if built(designed) accurately.

Sim2Real transfer[12, 13] aims at building simulations with accurate physics engines, designing virtual robots and virtual environments using the models of physical objects in an effort to reduce the gap between virtual and real world as much as possible. System identification, domain randomization, domain adaptation, simulation environment are some of the important concepts of Sim2Real transfer learning. As the name suggests, system identification methods are used for replicating the physical robots in a virtual simulated world. An illustration of system identification is shown in Figure 1.2a. Domain randomization is the process of randomizing various parameters of the simulation so that the real-world scenario might be of close resemblance to any one of them and is illustrated in Figure 1.2b. The data from virtual world and real world usually have different features. Domain adaptation uses the data from virtual world along with few samples from real-world to get a unified feature space through mapping or regularization techniques and uses this to predict the new samples of real-world data as illustrated in Figure 1.2c. While domain randomization relies completely on data from the simulations, domain adaptation on the other hand requires some data from the real-world. The performance of UAV in real-world depends on how well the techniques of Sim2Real transfer are implemented.

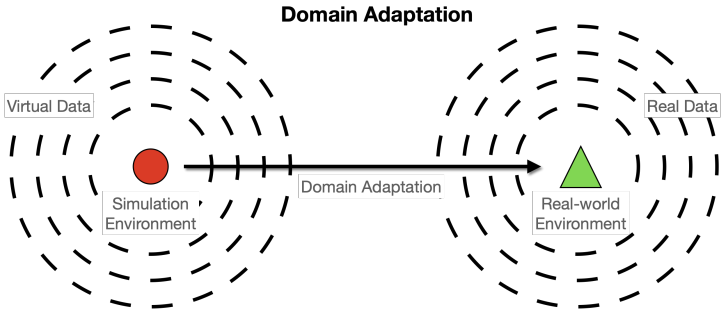
For a given hexarotor implemented with an impedance controller, this thesis proposes a novel 2-part method involving Sim2Real transfer to find the optimal impedance controller parameters required for the task of aerial writing on an unknown target surface.



(a) System Identification



(b) Domain Randomization



(c) Domain Adaptation

Figure 1.2: Illustrations of various techniques of Sim2Real transfer

1.1 Related Works

In an application of contact-based aerial inspections, a Lyapunov-based non-linear controller[14] is used for inspecting bridges from below. A quadrotor is equipped with a non-linear attitude, altitude and position controllers in-order to switch between free-flight and contact based configurations. A decrease in air pressure is observed as the UAV fly closer to the target surface reducing the power consumed by the rotors and thereby changing their velocity(RPM) and increasing the thrust of the UAV. This is called ceiling effect and the non-linear controllers are designed considering its influence on the UAV. When compared with a PID controller which does not consider the ceiling effect, the non-linear controller is able to make contact with the surface and revert back much quickly and safely. However, this approach is only valid for interactions with a horizontal surface and the interaction described here is a touch and leave scenario and the UAV does not move while making contact with the surface.

An interesting application of aerial physical interaction is showcased by a flying pantograph[15, 16]. The drone transfers the human hand drawing to a larger canvas(white board) relying only on the drawing and nothing else. The position commands are derived by tracking the human drawing gestures using vision techniques. Whereas a PID controller was used for a stable interaction with the canvas. However, the drawing is configured and restricted to a single surface. Most of the modern cities in developed countries are completely filled with high-rise buildings and are covered with glass windows. These windows are usually cleaned manually by humans and is very dangerous because of the high-grounds. A safer alternative is to use a hexarotor with one degrees of freedom(DOF)[17] to clean these glass windows. This hexarotor uses a force/position control for establishing and maintaining contact with the target surface. Although the controller produces respectable results in experiments of trajectory drawing, this method is also restricted to a single target surface which in this case a glass window.

A similar aerial physical interaction is described by *Asem Khattab*[18] where a hexarotor is tasked to draw a trajectory on an unknown target surface. An impedance controller is implemented here to take care of the interactions with the unknown target surface and Bayesian optimization is used to find the optimal gains of the controller. Unlike previous cases, this approach is not restricted to a single target surface and can be applied to any unknown surface. However, the disadvantage of this approach is that the drone has to draw a number of trajectories in-order to find the optimal controller gains. This is because Bayesian optimization is an iterative process and as the number of iterations increases, the behaviour of the trajectories is learned and the controller parameters are updated so that an accurate trajectory is drawn by the drone. Therefore, this method is only suitable for situations where the target surface is fixed but still unknown.

Sim2Real transfer learning is mostly used in tasks implemented with reinforcement learning as it requires several iterations of interactions with the real-world as part of learning which is costly and risk prone. Sim2Real transfer is implemented in the path planning of UAV for flying though narrow gaps[19] using deep reinforcement learning. A curriculum learning is used to deal with the flight trajectories whereas Sim2Real framework is used to transfer control commands to the real quadrotor. The trajectories are simulated in virtual world with virtual obstacles to generate appropriate control commands. Through domain randomization, the simulations are randomized by adding random Gaussian noise to the environmental variables like position, orientation, linear velocity and angular velocity. Noise is also applied to the initial parameters of the UAV such as initial linear velocity, initial angular velocity and initial position. While these random noises represent uncertainty of on-board sensors, the UAV model inaccuracy can be achieved by the addition of noise to the inertia matrix and max thrust of the rotors. Randomized to-Canonical Adaptation Networks(RCANs)[20] are used to achieve a Sim2Real transfer via Sim2Sim learning. This is done with a strong focus on domain adaptation

by converting the images generated from simulations to a canonical form before training. The images taken from real-world(very few compared to the number of images generated from simulations) are also converted into the canonical form and provided for training. The training is done on a 7-DoF robotic arm for a grasping task and achieved a 96% success rate in grasping. Similarly, a Sim2Real transfer is implemented in a 7-DoF robotic arm for the tasks of opening a drawer[21] and swinging a peg attached to the robot via a rope in to a hole where the simulation randomization is adapted from the real-world data.

1.2 Problem Description

Few applications of aerial physical interactions are contact-based inspections of chimneys and cleaning glass windows of high rise buildings. A hexarotor is has a task of drawing a circular trajectory on any unknown vertical target surface in an effort to replicate the inspection tasks. The hexarotor was implemented with an impedance control as the model of the environment(target surface) is unknown. The parameters of the impedance controller has to be chosen such that smooth and accurate trajectories are drawn everytime by the hexarotor. The tuning of controller parameters is a challenging task as it requires a few number of trial-and-errors before finalizing the optimal parameters. Since the task of the hexarotor is to draw circular trajectories on unknown surfaces, the optimal controller parameters chosen for a particular surface would result in deformed trajectories when used on different surfaces as illustrated in Figure 1.3.

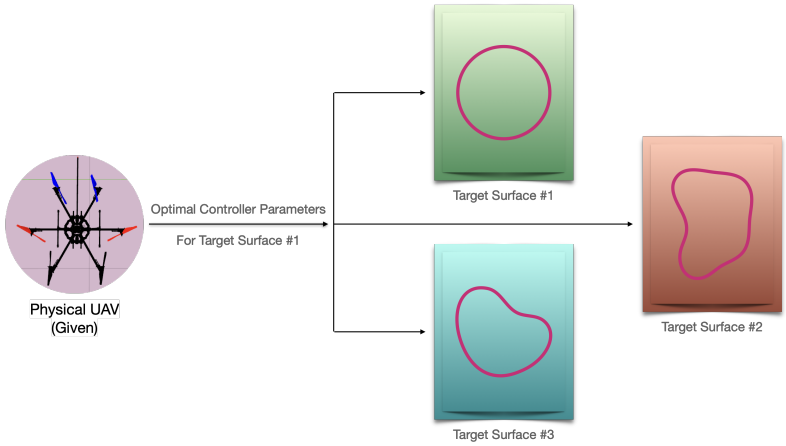


Figure 1.3: Hexarotor(UAV) drawing a circle on various surfaces with fixed optimal impedance controller parameters

Therefore, the impedance controller parameters has to be tuned for every surface the hexarotor is tasked to interact with. This comes with a huge risk factor of choosing harmful controller parameters that might damage the environment or the hexarotor itself. It is also a time taking process and might cause wear and tear of the hexarotor due to number of trial-and-errors in tuning the controller parameters. A safe alternative is to shift the process of tuning the controller parameters to simulations. The virtual models of the hexarotor and environment(target surface) can be built in the simulator and the optimal controller parameters can be calculated using as many iterations of trajectory drawings

as possible. The calculated optimal controller parameters from simulations can be implemented in a physical hexarotor for drawing trajectories on physical target surfaces using the techniques of Sim2Real transfer leaning. One problem involved here is finding the optimal controller parameters for the task of drawing circular trajectories on unknown target surfaces using the hexarotor. Second problem is how to transfer the optimal controller parameters calculated in simulations to a physical hexarotor for drawing trajectories on physical surfaces. The proposed method in next section focuses on solving these problems.

1.3 Proposed Method

For a given physical UAV(hexarotor) implemented with an impedance controller, this thesis proposes a novel method of predicting the optimal controller parameters required for drawing an accurate circle on an unknown target surface. This is a two-part method where one involves a pre-trained neural networks to predict the optimal controller parameters given the trajectory data and the other involves generation of a custom dataset to train the neural networks. The work of this thesis contributes to the second part while the first part is considered as a future work. A high level overview of this proposed method is illustrated in Figure 1.4.

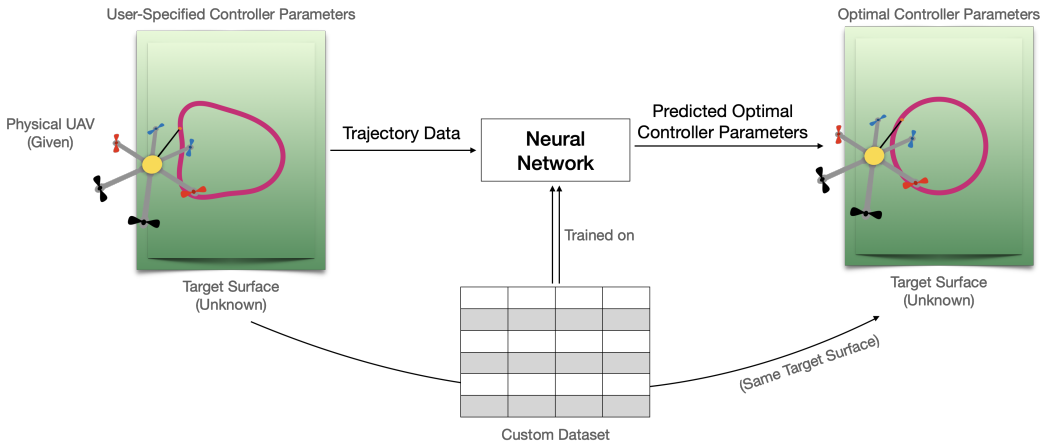


Figure 1.4: Block diagram of the proposed method

The idea is that a circular trajectory is drawn on the unknown target surface by an UAV with a user-specified safe set of controller parameters. As shown in Figure 1.4, the UAV may draw an incorrect trajectory resulting in a large error. This trajectory error along with the trajectory data are fed to a pre-trained neural network. The neural network is trained on a custom dataset with a large number of data samples related to various trajectories, trajectory data and corresponding optimal controller parameters. This converts the problem into a simple classification problem where the trajectory data is considered as inputs while the optimal controller parameters act as the corresponding labels. The pre-trained neural networks analyse the trajectory data and predicts the closest possible match of optimal controller parameters based on the custom dataset used for training. These optimal parameters are then fixed for the rest of the tasks on the current unknown surface resulting in an accurately drawn circle. If the surface is changed, then the UAV has to draw again with the user-specified safe controller parameters and the set of optimal parameters has to be predicted by the neural networks again.

The performance of the trained neural network is completely dependent on the quality of the dataset it is being trained on. A well classified dataset with a huge number of data samples can provide a one-shot Sim2Real transfer of the optimal controller gains. It is a Sim2Real transfer because the neural network is trained from simulation data and is predicting the optimal controller parameters for a real world data. Since this method rely solely on neural networks and requires a maximum of one to two iterations of circular trajectory drawing on the surface, this method can be deployed to draw on any target surface with ease. This thesis focuses on the topics of designing the reward function, domain randomization, Bayesian optimization and various design choices involved in them.

As mentioned earlier, the work of this thesis is on the second part which is the preparation of custom dataset itself. The dataset is a huge collection of various trajectories, their corresponding data and corresponding optimal controller parameters. It is obviously not feasible to generate this many data samples in real world and choosing the corresponding optimal controller parameters for each data sample. In order to overcome this challenge, the techniques of Sim2Real transfer learning are used. Meaning the entire experiment setup is recreated in a virtual world and the experiments of drawing circular trajectories on unknown surfaces are simulated to generate the required amount of data. This process eliminates the risk of choosing harmful controller parameters and unlocks the scope of several simulations which can not be done in real world due to the safely and stability issues. A virtual UAV with same specifications as its real world counterpart is tasked to draw the circular trajectory on an unknown virtual target surface. After drawing the trajectory, the optimal controller parameters are calculated and the corresponding data is stored in the custom dataset. This process is repeated for several times using domain randomization by changing the noise parameters, friction coefficients of the virtual surface and the shape of the drawn trajectory. This process is illustrated in Figure 1.5.

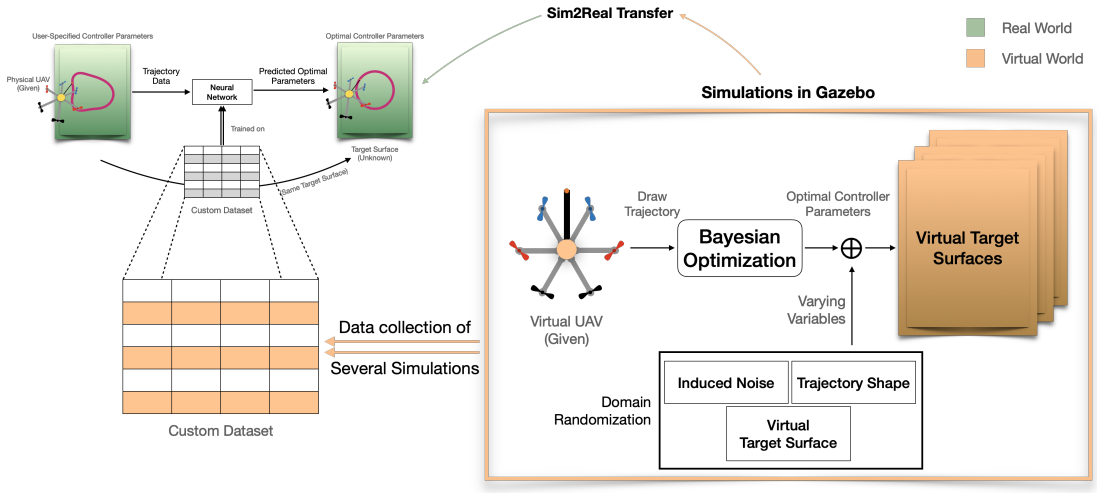


Figure 1.5: Block diagram representing the custom dataset of the proposed method (Focus of this thesis)

For every set of variables in domain randomization, the virtual UAV draws a circle on the virtual surface with random controller parameters. The optimal controller parameters for the given unknown virtual surface are calculated using the Bayesian optimization. It is an iterative process where the

algorithm learns the performance of past trajectories and chooses next controller parameters that would result in maximum performance of the trajectory or low error in drawn trajectory. A detailed explanation of Bayesian optimization is given in Chapter 2. This thesis also worked on designing the reward function which is one of the most important components of Bayesian learning. The trajectory performance that the Bayesian optimization is trying to maximize is actually the reward generated from the reward function. The process of designing the appropriate reward function is also known as reward shaping and is explained later in Chapter 3. Reward shaping is very crucial as it is the direct connection between the performance of the trajectory and optimal controller parameters. A greater reward means the drawn circle is accurate and the controller parameters chosen by Bayesian optimization that result in these rewards are considered as the optimal controller parameters.

1.4 Research Questions

The research questions for this thesis are as follows:

RQ-1 How to safely find the optimal impedance controller parameters of multi-rotor UAV for the task of drawing circular trajectories on unknown surfaces in as little iterations as possible?

The tuning of controller parameters requires a number of trial-and-errors which might result in harmful behaviour of the UAV if wrong parameters were chosen. A large number of iterations of drawing trajectories on the surface may lead to the wear and tear of the UAV and ultimately damaging it. Therefore, the optimal parameters need to be calculated in as few iterations as possible.

RQ-2 How to design the reward function such that the Bayesian optimization can find the optimal controller parameters for the task of drawing circular trajectories on unknown surfaces using an UAV?

The Bayesian optimization is used to find the extrema of an expensive target function. Therefore the task of drawing trajectories on surfaces using UAV has to be transformed into a target function. The controller parameters act as inputs to the target function where as the rewards act as output. These rewards are generated by a reward function.

RQ-3 How to speed up the search for optimal controller parameters by exploiting the expert knowledge of the task at hand?

The impedance controller requires tuning of a number of parameters. Using Bayesian optimization to find the optimal values for all the controller parameters would become expensive. Therefore based on the expert knowledge of the controller and task at hand, the controller parameters that needs to be optimized can be reduced in an effort to decrease the search space.

RQ-4 How to create multiple simulation scenarios to generate data for the custom dataset?

Neural networks require large amount of data to train on. Since they are being trained with the simulation data, a large number of unique simulation scenarios must be created. This is achieved with the help of domain randomization.

1.5 Report Layout

The theory behind the Bayesian optimization along with various design choices involved in the optimization algorithm such as surrogate model, kernel functions, acquisition functions and its hyperparameters are explained in this chapter 2. One of the most important components of the Bayesian optimization is the reward. This reward is generated from a function called reward function and the art of designing this reward function is called reward shaping. A novel reward function was designed for the task of drawing circular trajectory on unknown surfaces using an UAV and is described in Chapter 3. The simulation setup along with several simulations of optimizing 1D, 2D and 4D controller gains are shown in Chapter 4. These simulations are include varying the parameters of domain randomization variables for creating unique simulation scenarios. Finally the report ends with a conclusion and scope for future works in Chapter 5.

Chapter 2

Bayesian Optimization

Recapping the problem description and proposed method from previous section, a neural network is used to predict the optimal controller gains required for drawing a trajectory on an unknown target surface. However, a custom dataset needs to be generated first in-order to train the neural network. Bayesian optimization is used to find the optimal controller parameters. These optimal controller parameters along with the trajectory data of various trajectories drawn by the UAV form the custom dataset. This chapter describes various design choices involved in the Bayesian optimization. Starting with the choice of the surrogate model, the chapter continues onto various acquisition functions and finally concludes with a brief overview of the algorithm involved in finding the optimal controller parameters.

Bayesian optimization is an iterative process of finding the extrema of an expensive black-box function. For an unknown target function $f(\mathbf{x})$, the goal of the Bayesian optimization is to find the global maximum(or minimum) $\mathbf{x}^*$ in the domain χ of the target function and is expressed as follows:

$$\mathbf{x}^* = argmax_{\mathbf{x} \in \chi} \, f(\mathbf{x}) \tag{2.1}$$

Bayesian optimization does not need to know anything about the target function $f(\mathbf{x})$, as long as its domain χ is known and can be sampled at any point in time. A high-level overview of the Bayesian optimization algorithm is shown below in the Figure 2.1 and is used to find the maximum x^* of the unknown target function $f(x)$. Given a few initial observations, an initial surrogate model is built first and is represented as the prior distribution. The next best point to sample is then selected through the acquisition function. The point where the maximum of the acquisition function is observed is chosen as the next best point to sample and a new observation is made at this point. The surrogate model is updated for this new observation and the next best point is selected again for sampling. This process is repeated for n iterations leading to finding the maximum of the target function. The number of initial observations, surrogate model, acquisition function and the number of iterations n are all the design choices to be made by the user according to the needs of the task at hand and are individually explained in this chapter.

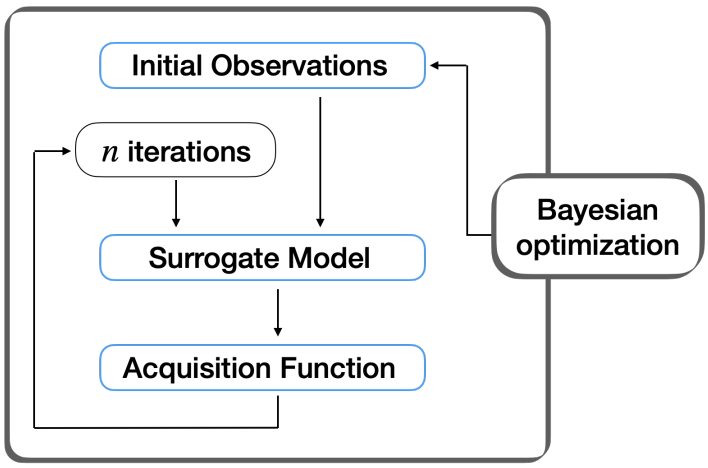


Figure 2.1: Block diagram of various steps involved in the Bayesian optimization

Bayesian optimization works using the 'Bayes theorem' as basis which states *"the posterior probability of a model* ***M*** *given evidence* ***E*** *is proportional to the likelihood of* ***E*** *given* ***M*** *multiplied by the prior probability of* ***M****"*. Hence the name 'Bayesian' and is expressed as:

$$P(M|E) \propto P(E|M)P(M) \tag{2.2}$$

The basis of every Bayesian statistical algorithm is *Posterior* $\propto$ *Prior* $\times$ *Likelihood* - a Bayes theorem. Therefore, it is important to know about the prior and posterior distributions of the target function. Prior distribution is the initial belief of the target function. It can be informative - containing relevant information about the target function or uninformative - no information about the target function. Likelihood function is the new belief that is obtained through the iterative process of the target function. Updating the prior distribution using Bayes theorem gives the posterior distribution which represents the updated belief of the target function after looking at the new data. Posterior is very much data-driven, if the prior is uninformative, posterior is calculated entirely from the data. If prior is informative, then posterior is calculated from both prior and data. Therefore, having a good prior will help determining a good posterior(explained in Section 2.2) thereby finding the global extrema of the target function quickly. Let's assume we have x samples of the target functions and x_i is the i^{th} sample with $f(x_i)$ being the observation of the target function at x_i. If $D_{1:t} = \{x_{1:t}, f(x_{1:t})\}$ is the collection of t observations, then the prior distribution $P(f)$ combined with the likelihood function if given by $P(D_{1:t}|f)$. The posterior distribution $P(f|D_{1:t})$ is then calculated by multiplying prior distribution with the likelihood function similar to Equation 2.2 and is given by:

$$P(f|D_{1:t}) \propto P(D_{1:t}|f)P(f) \tag{2.3}$$

The Bayesian optimization works using 3 important core topics *Surrogate model, Acquisition functions, Initial observations* and care must be taken in selecting these factors. Initial observations are used to get the prior belief of the unknown target function. This prior model is updated sequentially using a surrogate model as new observations are made. These new observations are taken at samples decided by the acquisition function depending on the need for the balance between exploration and exploitation. A detailed explanation of Bayesian Optimization and all the factors it is effected by are mentioned in *Eric Brochu's* work[22].

2.1 Surrogate Model

Bayesian optimization iteratively chooses new samples of the target function by fitting the surrogate model with the available samples. These include initial observations and samples that are obtained later as the model is updated. Any mistakes in the surrogate model results in not only being unable to locate the maximum of the target function, but also next point to sample might be chosen at areas of less informative. Therefore, choosing a valid surrogate model is an important part of Bayesian optimization. There are many methods that can provide accurate data when used as the surrogate model, such as the combination of a linear model encoding a tree-based dependency structure with an independent Gaussian processes[23], Preferential Bayesian Optimization(PBO)[24] that can only be queried through pairwise comparisons and automatic learning of warpings of the input space using the beta cumulative distribution function[25]. However, these are more sophisticated models and often require additional data in-order to reduce the uncertainties of the target function. It should be noted that the goal of the Bayesian optimization is to find the extrema of the target function but not to actually find the target function. A more popular and traditional method is to use Gaussian processes(GP) as the surrogate model. Recent studies also takes the advantage of neural networks by incorporating them as the surrogate models in Bayesian optimization[26], new methods like latent Gaussian processes[27] can deal with irreducible uncertainties that come with standard noise distributions and the use of ensemble(E) of GPs to adaptively select the surrogate model fit on-the-fly[28]. According to our problem at hand, the standard Gaussian processes(GP) is used as the surrogate model due to its natural way of providing uncertainty measures and a sample-efficient approach.

Gaussian process is a stochastic process(randomly changing over time) of random variables with a Gaussian distribution. Its called stochastic due its high uncertainty in the system, the processes can evolve into several directions if the starting point is known. The following definition is adapted from the book *Gaussian Processes for Machine Learning*[29]. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.* A Gaussian process is completely specified by its mean function and covariance function. The mean function $m(x)$ and the covariance function $k(x, x')$ of the target function is given by:

$$\begin{aligned} m(x) &= \mathrm{E}[f(x)], \\ k(x, x') &= \mathrm{E}[(f(x) - m(x))(f(x') - m(x'))] \end{aligned} \tag{2.4}$$

Gaussian processes are distributions over functions $f(x)$ of which the distribution is defined by a mean function $m(x)$ and a positive definite covariance function(also known as kernel function) $k(x, x')$, with the function values x and all possible pairs (x, x') in the input domain and can be written as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{2.5}$$

For any finite subset $X = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ of the domain of x, the marginal distribution is a multivariate Gaussian distribution with mean vector $\mu = m(X)$ and covariance matrix $\Sigma = K(X, X)$ is expressed as:

$$f(X) \sim \mathcal{N}(\mu, K(X, X)) \tag{2.6}$$

While the multivariate Gaussian captures a finite number of jointly distributed Gaussians, the Gaussian process doesn't have this limitation. Since functions can have an infinite input domain, the Gaussian process can be interpreted as an infinite dimensional Gaussian random variable. The posterior distribution can be calculated by treating the Gaussian process as a prior defined by the kernel function. This posterior distribution can then be used to predict the expected value and probability of the output variable y given input variables X. In most real-world situations, the actual function values are not available but only the noisy observations of the function therefore $y = f(x) + \varepsilon$. Assuming

additive independent identically distributed Gaussian noise ε with variance σ_n^2, the prior of the noisy observations is modified as $\Sigma = K(X, X) + \sigma_n^2 I$. If n_1 samples are observed at data points (X_1, y_1), then the predictions $y_2 = f(X_2)$ for n_2 new samples can be made using the Gaussian process prior. This can be done with the help of posterior distributions $p(y_2|y_1, X_1, X_2)$. Since both y_1, y_2 come from the same multivariate distribution(finite number of samples), they are jointly Gaussian and are expressed as:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} K(X_1, X_1) + \sigma_n^2 I & K(X_1, X_2) \\ K(X_2, X_1) & K(X_2, X_2) \end{bmatrix}\right) \tag{2.7}$$

The mean functions μ_1, μ_2 are chosen to be zero for simplicity although this can be set to any other values if required by the task at hand. Lastly, the predicted mean function $m(X_2)$ and covariance function $K(X_2, X_2)$ of the new samples n_2 are given by:

$$\begin{aligned} m(X_2) &\triangleq \mathrm{E}[y_2|y_1, X_1, X_2] = K(X_2, X_1)[K(X_1, X_1) + \sigma_n^2 I]^{-1} y_1, \\ k(X_2, X_2) &= K(X_2, K_2) - K(X_2, X_1)[K(X_1, X_1) + \sigma_n^2 I]^{-1} K(X_1, X_2) \end{aligned} \tag{2.8}$$

2.2 Initial knowledge

It is mentioned earlier that the Gaussian processes is specified by a mean function and covariance function. The mean function is set to zero for simplicity since there is no information about the target function. The covariance function models the joint variability of the Gaussian process random variables. In other words it provides the similarity measurement between any two points in the target function. The covariance function is also known as kernel function and there are several kernels to choose from(refer Chapter4 from book [29]). Kernel function should be a positive definite in order to be a valid covariance function. Every kernel functions results in different priors on the Gaussian process distribution and has to be chosen according to the problem at hand. A commonly used kernel known as *squared exponential(SE) kernel* also known as Radial-basis function(RBF) kernel with a characteristic length-scale l is used in the GP and is expressed as:

$$cov(f(x_1), f(x_2)) = k_{RBF}(x_1, x_2) = \exp\left(-\frac{1}{2\,l^2}|x_1 - x_2|^2\right) \tag{2.9}$$

Now that the kernel of the GP is fixed, the prior distribution is generated using the initial observations. The initial observations can be chosen at random by the algorithm or can be done manually. Through intuition, the initial observations can be chosen such that they provide hints to the Bayesian optimization about the location of global maximum of the target function. On the other hand, letting the algorithm decide the initial observations will automate the process with minimal human interference. Since we will be using the Bayesian optimization several times to generate data for the custom dataset, the initial observations are taken at random by the algorithm.

2.3 Acquisition Function

Once the initial observations are taken, an initial surrogate model is generated as prior. This model is updated constantly as new observations come in through several iterations. During these iterations, the surrogate model is updated and is used to find the next best point to sample through a utility function. This utility function is also called as acquisition function(also sometimes referred

to as infill function). Therefore, finding the next best point is a secondary optimization problem but is usually much easier to solve as the acquisition functions are chosen so that it is easy to evaluate while still being non-convex.

Acquisition functions are another flexible part of the Bayesian optimization with many design choices. They are generally defined such that high utility(acquisition) corresponds to potentially high values of the target function. This can be because the prediction is high, the uncertainty is great, or both. The maximum of acquisition function is used as the next point of the target function to evaluate(new observation). The main goal of acquisition functions is to guide the search for optimum while maintaining balance between exploration and exploitation. Exploration will choose the next point at an unexplored region and exploitation chooses the next point around the already explored region. While exploration helps in finding the global maximum(or minimum) of the target function quickly, exploitation helps in fine-tuning and finding the accurate maximum(or minimum) of the target function. Therefore, acquisition functions should consider a trade-off between exploration and exploitation depending on the problem at hand. The most commonly used acquisition functions are Probability of Improvement(POI), Expected Improvement(EI) and Upper Confidence Bound(UCB). The definitions of these functions(adapted from [22]) are given below for reference.

Probability of Improvement (POI)

For a function $f(x)$ that needs to be maximized, if $x^* = argmax_{x_i \in x_{1:t}} f(x_i)$ is the current maximum after i iterations then the improvement $I(x)$ is given as:

$$I(x) = max(f(x) - f(x^*), 0)$$

This results in zero if the associated value of $f(x)$ at x is less than $f(x^*)$. On the other hand, if $f(x)$ is larger than $f(x^*)$ then $I(x)$ depicts the improvement the new observation $f(x)$ at point x has over current best maximum $f(x^*)$. Therefore, the probability of improvement can be written as:

$$POI(x) = P(f(x) \geq f(x^*)) = \Phi\left(\frac{\mu(x) - f(x*) - \xi}{\sigma(x)}\right) \tag{2.10}$$

where, $\Phi(\bullet)$ is the normal cumulative distribution function. $\mu(x)$ and $\sigma(x)$ are the mean and variance generated from the Gaussian process. $\Phi \geq 0$ is the hyper parameter that helps in regulating the trade-off between exploration and exploitation. $\Phi = 0$ results in pure exploitation and the Bayesian optimization being struck in local maximum. POI function is also called Maximum Probability of Improvement(MPI) or P-algorithm.

Expected Improvement (EI)

While the probability of improvement of the current best maximum $f(x^*)$ is considered in POI, Expected Improvement(EI) considers the magnitude of this improvement over the current best maximum. Therefore, the Expected Improvement $EI(x)$ is given as:

$$EI(x) = \mathrm{E}\left[I(x)\right] = \int_{-\infty}^{\infty} I(x)\psi(z)\,dz$$

where, $\psi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-z^2}{2}\right)$ is the probability density function of the normal distribution $\mathcal{N}(0,1)$. This can be expressed in the form of predictive mean $\mu(x)$ and variance $\sigma(x)$ of Gaussian process as show below:

$$EI(x) = \begin{cases} (\mu(x) - f(x^*) - \xi)\Phi(Z) + \sigma(x)\phi(Z) & \text{for } \sigma(x) > 0 \\ 0 & \text{for } \sigma(x) = 0 \end{cases} \tag{2.11}$$

where $Z = \frac{\mu(x) - f(x^*) - \xi}{\sigma(x)}$, $\Phi(\bullet)$ is the standard normal cumulative distribution function(CDF), and $\phi(\bullet)$ is the standard normal probability density function(PDF). ξ is the hyper parameter and works similar to POI by maintaining trade-off between exploration and exploitation. The complete derivation of EI can be found in [30].

Upper Confidence Bound (UCB)

If $\mu(x)$ is the mean and $\sigma(x)$ is the uncertainity(variance) generated by the Gaussian process, then Upper Confidence Bound (UCB) is the weighted sum of these expected performances and is given by:

$$UCB(x) = \mu(x) + \lambda\sigma(x) \tag{2.12}$$

A lower λ will result in Bayesian optimization choosing next samples at high performing(high $\mu(x)$) regions while higher λ results in next samples landing in high uncertainty regions. Therefore by tuning $\lambda \geq 0$, the trade-off between exploitation and exploration can be controlled using UCB. While UCB is used in maximizing the function, the Lower Confidence Bound(LCB) is used in minimizing it and is given as follows:

$$UCB(x) = \mu(x) - \lambda\sigma(x)$$

2.4 Algorithm

For an unknown target function $f(x)$, the noisy observations y_i sampled at points x_i are given by:

$$y_i = f(x_i) + \varepsilon_i$$

The tuples of observations and the corresponding points are collected as $\mathcal{D}_i = (x_i, y_i)$. First, n initial observations are sampled randomly(or manually) from the target function. Since the kernel function is known, an initial surrogate model is generated using the initial observations $\mathcal{D}_{1:n}$. Based on this initial surrogate model $M(f(x)|\mathcal{D}_{1:n})$, the next best point x_j to sample is calculated by optimizing the acquisition function. The target function is sampled at this new point and the noisy observation y_j is calculated. The new observation is added to the initial observations and the data is updated $D_{1:j} = D_{1:n} + D_{(n+1):j}$. The GP(surrogate model) is updated with the new data $D_{1:j}$ and the process is repeated for several iterations as shown in Algorithm.1.

Algorithm 1 Framework of Bayesian Optimization

Require: Domain of the target function χ

1: Get n initial observations $\mathcal{D}_n = (x_n, y_n)$
2: **for** iterations $j = n, n+1, n+2, \ldots$ **do**
3: Build/update the GP(surrogate model) $M(f(x)|\mathcal{D}_{1:j})$
4: Find the next best point x_j to sample by optimizing acquisition function
5: Get the new observation $y_j = f(x_i) + \varepsilon_i$
6: Update the data $D_{1:j} = D_{1:n} + D_{(n+1):j}$
7: **end for**

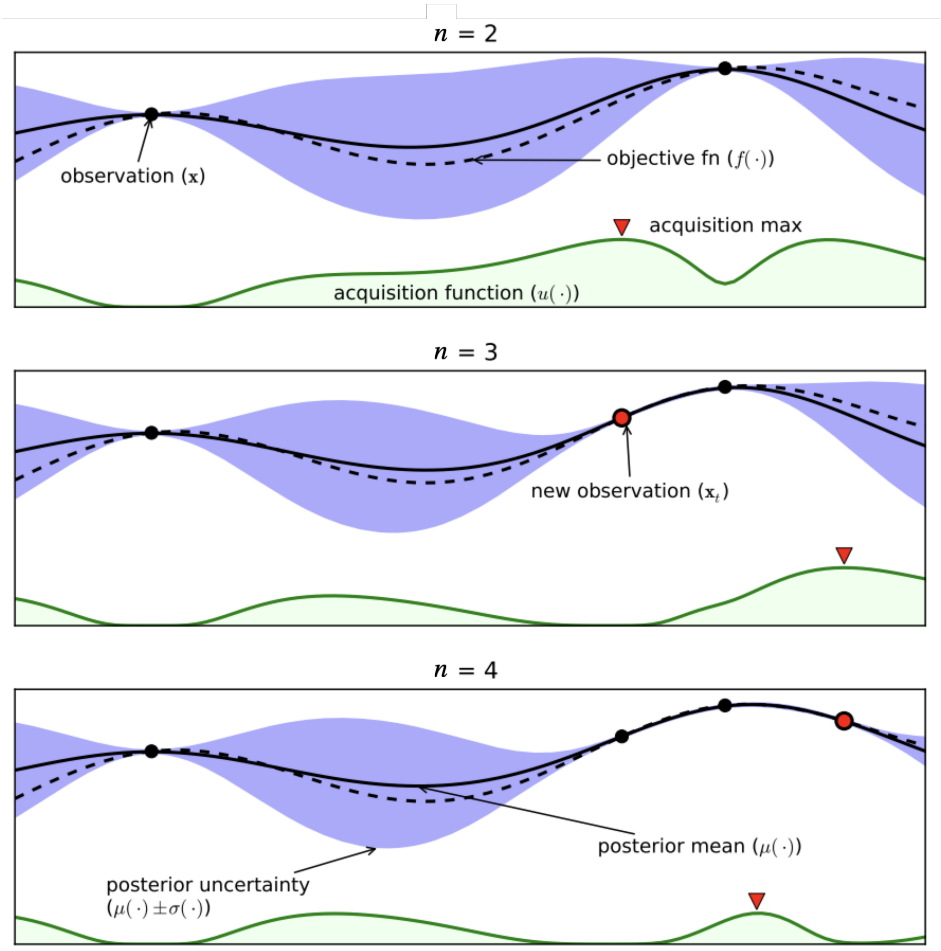


Figure 2.2: A representation of Bayesian optimization for a 1D toy problem. The figure shows the surrogate model(GP) along with its covariance(shaded blue region) and mean(continuous black line) functions. The acquisition function(green line) along with its maximum(red triangle) are shown at the bottom of each plot. The figure also shows the already observed points(black dots) and the next best point to sample(red dots).

A visual representation of the Bayesian optimization, specifically the process of choosing the next best points to sample and updating the surrogate model is shown in the Figure 2.2. The dotted line in plots represents the actual target function (groundtruth) $f(x)$ and the green line represent the acquisition function. The two black dots in the first (Top)plot are the points that are already sampled. The shaded blue regions are the regions the Bayesian optimization is uncertain about the target function. This shaded blue regions are zero at the two observations(black dots) because the Bayesian optimization has already sampled at these points and is certain about the target function at these points. Hence the zero uncertainty(zero shaded blue regions) at these sampled points. The continuous black line represents the mean of the surrogate model(GP) and is an approximation of the target function. The next point to sample is chosen by finding the maximum of the acquisition function represented by a red triangle in the first (Top)plot. The new observation is taken at this point in the second (Middle)plot and is represented by a red dot. It can be seen that the surrogate model(shaded blue region and its mean) has been updated accordingly to this new observation in the second (Middle)plot. The acquisition function has also updated and has a new maximum(next best point to sample). A new observation is taken at this point in the third (Bottom)plot. This process continuous for n iterations depending on the required accuracy of the user. As the new observations are made, the Bayesian optimization learns more about the target function. However, it should be noted that the goal of the Bayesian optimization is not to find the target function but to find its maximum. The choice of acquisition function and its hyperparameters help in finding the maximum of the target function in as few iterations as possible based on the requirements of the user.

2.5 Conclusion

This chapter describes the theoretical concepts involved in the Bayesian optimization. Bayesian optimization is an iterative process of finding the extrema of an expensive black-box function. Bayesian optimization has three important design choices that greatly effect its performance and outcomes. Surrogate model act as the base for the algorithm and a Gaussian process is chosen due its natural way of providing uncertainty. Another design choice involved in Bayesian optimization is the choice of acquisition function. Out of the three popular acquisition functions - Probability of Improvement (POI), Expected Improvement (EI) and Upper Confidence Bound (UCB) the first is chosen for the current task of drawing a circle on unknown target surface. This acquisition function and its hyperparameter were chosen based on the results of various experiments conducted by *Khattab*[31]. The Bayesian optimization also requires few initial observations that serve as a prior distribution. For simplicity, a few random observations were chosen to build the prior. Finally, in-order to apply Bayesian optimization to the task of trajectory drawing on unknown surface there is a need to formulate the task at hand into a black-box function. Since the impedance controller parameters act as the inputs there needs to be a variable that represents the performance of the task and act as output to the target function(black-box function). This output variable representing the performance of task at hand is called reward and is generated from reward function. The importance and challenges of designing this reward function also known as reward shaping is explained in the next chapter.

Chapter 3

Reward Shaping

The only assumptions the Bayesian optimization has on the target function is that its domain χ is known and that it can be sampled at any instance within its domain. The target function needs to have a reasonable output for every change in its input variables. The input can be multi-dimensional depending on the domain χ of the target function. However, the output should be a single dimensional variable otherwise it needs to be converted into one. Since this is not the case in most robotic applications, the art of designing a function(called reward function $R(x)$) that translates the performance of the task into a one dimensional variable(reward) is known as reward shaping. This chapter explains various challenges involved in designing a reward function and proposes a novel reward function overcoming those challenges.

The problem of using an UAV to draw circular trajectory on unknown target surface has to be formulated in terms of a target function $f(x)$ in order to be able to use Bayesian optimization for finding the optimal controller gains. Since the UAV is implemented with an impedance controller to deal with the interactions with the target surface, the controller parameters(gains) can be used as inputs to the target function. Tuning these impedance controller gains has an immediate effect on the interactions and therefore the accuracy of the trajectory. After all, our ultimate goal is to find these optimal impedance controller gains required for smooth interactions with any given unknown surface. The inputs of the target function can be one-dimensional or multi-dimensional but the output can only be one-dimensional in-order to be able to work with the Bayesian optimization. A one-dimensional variable called as reward is chosen to serve as the output to the target function. This reward represents the performance and accuracy of the trajectories drawn by the UAV.

The function that generates this reward is called reward function $R(x)$ and the process of designing this reward function is called reward shaping. Therefore, the target function $f(x)$ has the impedance controller gains as inputs and rewards generated by a reward function as outputs. This process is illustrated in a block diagram below shown in Figure 3.1. It can be seen that the reward $f(x)$(output of reward function $R(x)$) is the input to the Bayesian optimization algorithm. The goal of the Bayesian optimization is to find the maximum reward ($x^* = argmax\, f(x)$) by choosing the appropriate controller gains x. The controller gains associated with the maximum reward x^* are then considered as the optimal controller gains that yield the desired trajectories drawn by the UAV.

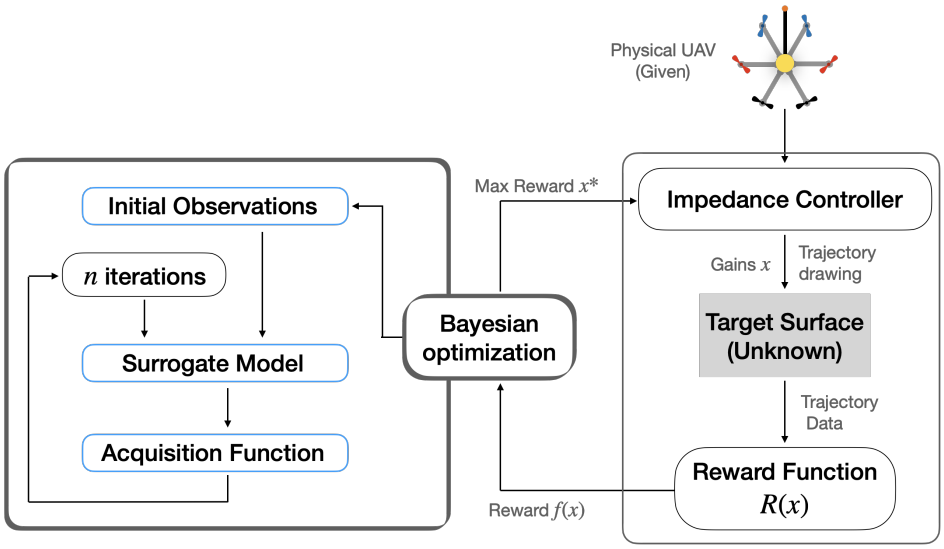


Figure 3.1: Block diagram representing the various steps involved in Bayesian optimization and the target function (Reward) $f(x)$

Reward shaping is one of the crucial parts of Bayesian optimization as the rewards(output) along with controller gains(input) are combined to form the target function $f(x)$. Designing the reward function is considered as an art because the reward should depict the features of the real-world application which in our case is the circular trajectory drawn on a surface by the UAV. This means a good reward function should reflect the low errors of the drawn trajectories in higher rewards and large errors in lower rewards. It is clear that the reward function is very unique to the problem at hand and one reward function does not apply to every problem. Therefore, one should have a clear understanding on the problem(task at hand) in order to design a related reward function.

3.1 Reward Function #1

As mentioned earlier in Section 1.1, a similar problem was discussed by *Khattab*[18] where the Bayesian optimization is used to find the optimal controller gains for a task of trajectory drawing on a vertical surface. The reward function described in this paper is based on trajectory errors. The impedance controller of the UAV assumes a virtual spatial spring connected between the end-effector of the UAV Ψ_E and the desired frame Ψ_D. While a virtual damper is connected between the body of the UAV Ψ_B to the inertial frame Ψ_I as shown in Figure 3.2.

If $P_E^I(t) \in \mathbb{R}^3$ and $R_E^I(t) \in so(3)$ are the actual position vector and orientation matrix of the UAV end-effector frame Ψ_E with respect to the inertial frame Ψ_I at time t respectfully and $P_D^I(t) \in \mathbb{R}^3$, $R_D^I(t) \in so(3)$ are the desired position vector and orientation matrix of the desired frame Ψ_D with respect to the inertial frame Ψ_I at time t respectfully. Then $E_T(t)$, $E_R(t) \in \mathbb{R}^3$ are the 3D vectors representing the translational position error and rotational orientation error respectfully along the three axes in the Euclidean space at time t expressed as follows:

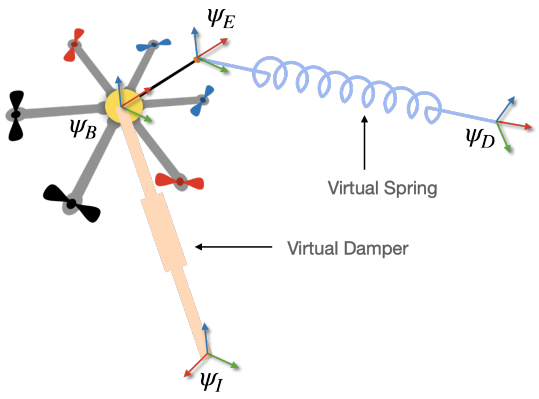


Figure 3.2: Coordinate frames associated with the UAV implemented with an impedance controller

$$E_T(t) = P_E^I(t) - P_D^I(t), \tag{3.1}$$

$$E_R(t) = \frac{1}{2}\left[R_D^I(t)^\top R_E^I(t) \; - \; R_E^I(t)^\top R_D^I(t)\right]^\vee, \tag{3.2}$$

where $\bullet^\vee$ is the map from 3D skew-symmetric matrix to 3D vectors($so(3) \to \mathbb{R}^3$). While $E_T(t)$ and $E_R(t)$ are the translational and rotational error at time t along the trajectory, E_t and E_r represent the root-mean-square(RMS) error of translational and rotational error along each axis $j \in \{x, y, z\}$ at time t for a given trajectory.

$$E_T = \lim_{T\to\infty} \sqrt{\frac{1}{2T}\int_{-T}^{T} (E_T^j(t))^2\, dt} \quad ; \quad E_R = \lim_{T\to\infty} \sqrt{\frac{1}{2T}\int_{-T}^{T} (E_R^j(t))^2\, dt} \tag{3.3}$$

where, T is the total time taken to complete drawing of one trajectory on the target surface. The RMS error is calculated over the duration of the trajectory in an effort to combat the dependency on the trajectory duration and thereby allowing for an apt comparison of trajectory errors over multiple iterations(trajectories). The reward function $R(x)$ thus formed is given by:

$$R(x) \; = \; -c_1 E_t \; - c_2 E_r \; - c_3 \frac{sum(x)}{dim(x)} \tag{3.4}$$

where, $sum(x)$ is the sum of all input controller parameters and $dim(x)$ is the of number of controller parameters that are considered as inputs to the target function $f(x)$. The presence of constants c_1 and c_2 is to maintain an equal importance between translational and rotational errors as they are usually of different scales. Since large controller gains are often associated with extremely high motor speeds and power consumption, care must be taken to minimize them as much as possible. Therefore, a third component - the mean of all the input controller gains x is added to the reward function and is regulated by constant c_3. The purpose of c_3 is to limit the maximum reward achieved to lower controller gains as much as possible. Finally, a negation is added to the reward function in order to find its maximum.

Drawbacks

Although this reward function does a decent job in maximizing the accuracy of the trajectories drawn while minimizing the required controller gains, it has a few drawbacks in the form of constants(weights) c_1, c_2, c_3. These weights can break the target function if not chosen properly and therefore plays a very important role in reward shaping. Besides having a basic idea on the relations between these weights through intuition, choosing the appropriate weights is not always an easy task. This would not be a big issue for situations where the environment remains unchanged[17, 18] and finding the optimal constants(weights) is feasible since it is a one-time process. But for our case of drawing circular trajectories on any unknown surface, it is very difficult and time consuming to find multiple sets of optimal weights as the environment is changed frequently during the process of generating data for the custom dataset(mentioned in Section 1.3). Any mistakes in these weights would result in neural networks being trained with imperfect dataset costing a lot of time and effort. As mentioned in Chapter 2, Bayesian optimization itself has many design choices in the form of acquisition functions and its hyperparameters and this choice of optimal weights adds unnecessary complexity to the already complex problem. Therefore the reward function mentioned in Equation 3.4 is not suitable for our proposed method and is demonstrated in an effort to explain to the mentioned drawbacks.

3.2 Reward Function #2

Unlike the reward function described in Equation 3.4, the proposed reward function aims to give a physical meaning to it by incorporating the energy of the impedance controller. If $E_{Kin}(t)$ is the Kinetic energy of the controller and $E_{Pot}(t)$ is the Potential energy, then the new reward function $R(x)$ is simply described as the summation of kinetic and potential energy over time t taken in drawing the trajectory and is given by:

$$R(x) \;=\; \int_t E_{Kin}(t)\,dt \;+\; \int_t E_{Pot}(t)\,dt \tag{3.5}$$

If $E_T(t) \in \mathbb{R}^3$ is the position error of the UAV end-effector with respect to its desired frame and K_t is a 3×3 matrix representing the translational stiffness coefficients of virtual spring then the potential energy can be written as:

$$E_{Pot}(t) \;=\; \frac{1}{2}\, E_T(t)^\top\, K_t\;\; E_T(t), \tag{3.6}$$

Similarly, let $T_{err}(t)$ be the twist error between the UAV end-effector and its desired frame. The kinetic energy can then be written as:

$$E_{Kin}(t) \;=\; \frac{1}{2}\, T_{err}(t)^\top\, I\; T_{err}(t) \;\;=\;\; \frac{1}{2}\, \omega_{err}(t)^\top\, J\; \omega_{err}(t) \;+\; \frac{1}{2}\, m\; v_{err}(t)^\top\, v_{err}(t) \tag{3.7}$$

where, I is the 3×3 identity matrix. m and J are the mass and inertia matrix of the UAV respectfully. $w_{err}(t)$ is the error in angular velocity and $v_{err}(t)$ is the error in linear velocity of the UAV. By looking at the various coordinate frames associated with the UAV shown in Figure 3.2, the twist error can

further be defined as $T_{err}(t) = T_B^{I,E}(t) - T_D^{I,E}(t)$. Where $T_B^{I,E}(t)$ is the twist at body frame Ψ_B of the UAV with respect to its end-effector frame Ψ_E expressed in inertial frame Ψ_I and $T_D^{I,E}(t)$ is the twist at desired frame Ψ_D with respect to end-effector frame Ψ_E expressed in inertial frame Ψ_I.

Potential energy is calculated between the end-effector of the UAV Ψ_E and the desired frame Ψ_D of the virtual setpoint of the trajectory as given in Equation 3.6. This means a higher potential energy is registered if the end-effector of the UAV is far away from the desired setpoint and vise-versa. The idea behind using this potential energy in the reward function is to check if the UAV is following the desired trajectory at all times. A lower potential energy means minimum trajectory tracking error translating into a maximum accuracy of the trajectory. However, the maximum rewards are always achieved at higher gains if the reward function consists of potential energy alone which results in unstable behaviour of the UAV. At higher gains, the impedance controller makes the propellers spin at high velocities in-order to minimize the trajectory tracking error resulting in high power consumption and unstable behaviour due to physical limitations of the UAV. Therefore, kinetic energy is added to the reward function in an effort to make the maximum rewards achievable at comparatively lower gains where UAV can show a stable behaviour. Kinetic energy is the twist error at the end-effector of the UAV and the desired frame of virtual setpoint of the trajectory as given in Equation 3.7. Since the later is zero, kinetic energy is directly related to the vibrations at the end-effector of the UAV and a lower kinetic energy results in smooth interactions with the surface. Therefore, a higher kinetic energy and a lower potential energy is observed at higher gains and are added together in an effort to maintain a balance between the accuracy and smoothness of the drawn trajectories.

However, the kinetic energy and potential energy are usually of different scales and adding them together in their original states may lead to one of them being dominant in the reward. For our case, the kinetic energy is multiplied by a constant c_1 in-order to bring it to the same scale as potential energy and is given by:

$$R(x) = c_1 \int_t E_{Kin}(t)\, dt + \int_t E_{Pot}(t)\, dt \tag{3.8}$$

This comes with its own problem of choosing this constant c_1. This is one of the reasons to move away from the reward function(Equation 3.4) mentioned in previous section only to come back to the same issue again. Although, the number of constants has been decreased from 3 (Equation 3.4) to 1 (Equation 3.8) it is still a concerning factor since our ultimate goal is to prepare a custom dataset with several data samples for training the neural network. Instead of trying to find the optimal value of c_1 for every trajectory, the constant c_1 is set to be a function of reward function itself.

Bayesian optimization discussed in Chapter 2 involves multiple iterations of trajectory drawing on the target surface in-order to find the maximum of unknown target function. This maximum reward is found after certain number of iterations(episodes) depending on the desired accuracy of the user. However, the algorithm needs good initial observations as the prior distribution before starting the learning process. During these initial observations, the kinetic energy and potential energy are calculates according to Equation 3.7 and Equation 3.6 respectfully. Before starting the learning process, the algorithm calculates the constant c_1 as the ratio of mean of potential energy and kinetic energy for the initial observations and is expressed as follows:

$$R(x) = c_1 \int_t E_{Kin}(t)\, dt + \int_t E_{Pot}(t)\, dt \;\; ; \qquad c_1 = \frac{\text{mean}(E_{Pot}(t))}{\text{mean}(E_{Kin}(t))} \tag{3.9}$$

Once the constant c_1 is calculated and fixed, the initial observations are again observed but this time the reward is calculated using Equation 3.9 and the prior distribution is built. The learning observations are also later observed using the same reward function. By making c_1 a function of initial observations, c_1 is automatically calculated for every trajectory drawn and is adaptable for any environment(target surface) with ease and eliminates the need for finding optimal values of c_1 for every case. The accuracy of c_1 and thereby the accuracy of scaling the kinetic energy to match that of potential energy can be adjusted by changing the number of initial observations. More initial observations lead to precise matching of scales of kinetic and potential energies. However, it should be noted that it may not always be the case where the kinetic energy is of smaller scale compared to potential energy. This reward function is specifically designed for the task of drawing trajectories on unknown surface using an UAV and is focused on balancing the accuracy and smoothness of the drawn trajectories. This is done by giving an equal importance to both kinetic energy and potential energy. This is a design choice and can be changed by multiplying the potential energy with a new constant c_2 in-order to prioritize one over the other if necessary.

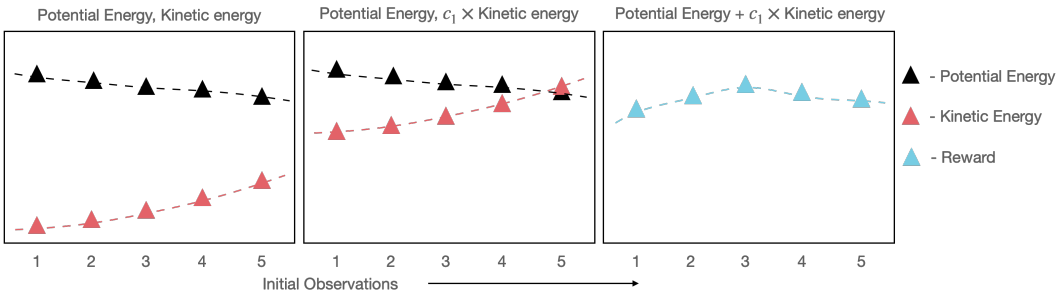


Figure 3.3: Illustration of reward function by up-scaling the kinetic energy

This process of up-scaling the kinetic energy to match the potential energy is illustrated using a 1D toy problem of 5 initial observations in Figure 3.3. It can be seen in fist subplot that the kinetic energy is of much smaller scale compared to that of potential energy. The c_1 is then calculated as the ratio of mean of potential energy and mean of kinetic energy for the given 5 initial observations. The up-scaled kinetic energy can be seen in second subplot where the kinetic energy has moved up to be in the same scale as the potential energy. Finally, the up-scaled kinetic energy is added to the potential energy resulting in the rewards that balance the accuracy and smoothness of the trajectories drawn and can be seen in the third subplot.

3.3 Conclusion

This chapter described the importance of designing a proper reward function. Reward shaping is a very important part of Bayesian optimization and an incorrect reward function would affect the optimization behaviour. A reward function based on position error and rotation error was described first in the chapter. This was followed by various drawbacks involved in the said reward function. These drawbacks include the presence of arbitrary constants whose values are to be set appropriately. This possess a major threat to the Bayesian optimization through altered rewards with the incorrect choice of the arbitrary constants. A novel reward function was later proposed in the chapter which eliminates the arbitrary constants. The new reward function is based on the potential energy and kinetic energy of the impedance controller. While the potential energy reflects in capturing the position error of the trajectories drawn by the UAV, the kinetic energy depicts the smoothness of the trajectory drawings. Adding both energies result in the reward balancing the accuracy and smoothness of the trajectories drawn by the UAV. Since the kinetic energy and potential energy usually belong to different scales, the kinetic energy is scaled up to the match that of the potential energy before adding together to form the reward. This up-scaling of kinetic energy is done by calculating the ratio of mean of potential energy and mean of kinetic energy of all initial observations of Bayesian optimization. The kinetic energy is then multiplied with this constant to bring it to the same scale as of the potential energy. This method of up-scaling eliminates the need for any arbitrary constants and can be applicable for any similar tasks and any target surface. Although a balanced approach of rewards was chosen in this thesis, the reward function can be customized to prioritize accuracy over smoothness of the drawn trajectories by multiplying the potential energy with an additional constant. A proper demonstration of rewards before and after scaling will be shown in next chapter.

Chapter 4

Simulations

This chapter talks about various simulations that incorporate the concepts of Bayesian optimization (Chapter 2) and the proposed reward function (Section 3.2) in an effort to find the optimal controller parameters. The optimal controller parameters found in this fashion for several scenarios are put together to form the custom dataset required for training the neural networks as part of the proposed method described in Section 1.3. The chapter starts with the simulation setup describing various design choices involved in the simulations, followed by various simulations related to the design of the reward function. A visual representation of finding the optimal controller parameters is shown through various simulations related to 1-Dimensional and 2-Dimensional optimization problems where 1 input parameter and 2 input parameters are involved as inputs to the target function respectively. After ensuring that the Bayesian optimization is successful in finding the optimal gains, domain randomization is used to randomize the simulation parameters. Later, the Bayesian optimization is used to find the optimal controller parameters for multi-dimensional optimization problem where more than 2 input parameters are optimized.

4.1 Simulation Setup

The proposed method discussed in Section 1.3 requires Sim2Real transfer as the neural networks would be trained with the data obtained through simulations. Therefore, it is essential to know about the various components involved in the simulation setup along with the concepts of Sim2Real transfer such as system identification, simulation environment and domain randomization. Figure 4.1 gives an overview of various packages involved in building and running the simulations.

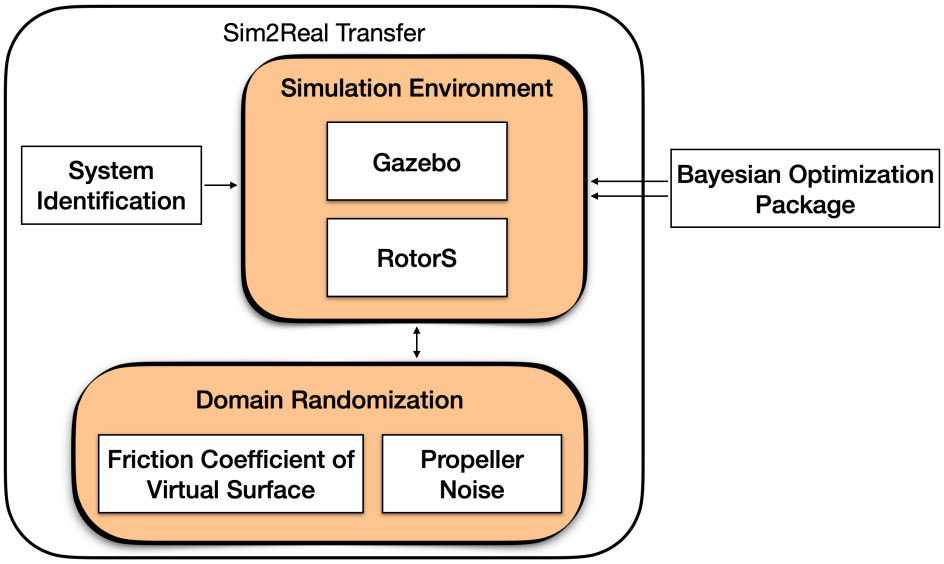


Figure 4.1: Block diagram illustrating various software components involved in setting up the simulations

System Identification

System identification is crucial for experiments with simulated environments and a considerable effort must be put into building the models of the physical systems in-order to bring the virtual simulation world as close as possible to the real-world. Thanks to the continuous research activities at Robotics and Mechatronics lab(RAM) in University of Twente, a fully-actuated hexarotor named *betaX*[32, 33] with its mathematical model was readily available and is therefore used in this thesis. This model was used to build a virtual version of *betaX* and is introduced in the simulation environment.

Simulation Environment

Another important aspect of the Sim2Real transfer is the simulation environment. Along with the system identification, the simulation environment should also be chosen carefully in order to minimize the gap between the virtual world and real-world. There are many choices in this area such as Unity3D, PyBullet, MuJoCo, ThreeDWorld[34], Gazebo, Nvidia Issac sim, etc. Since we are not using any deep learning or reinforcement learning libraries or gym environments, Gazebo simulator is used in this thesis as it is suitable for more complex scenarios. Gazebo also has the advantage for an easy integration with Robot Operating System(ROS) - a middleware enabling the use of realistic scenarios like RotorS[35] that is used in this thesis, AirSim[36], etc resulting in quick deployments into the real-world.

As part of Sim2Real transfer the process of drawing the circular trajectory on unknown surface using an UAV has to be repeated for several times in-order to generate the data large enough to train the neural networks. This consumes a lot of time when the simulation is run on default settings of Gazebo. Therefore, the parameters of the gazebo simulator are tweaked a bit in an effort to run the simulator faster than real-time. This is done by setting the *RealTimeUpdateRate* to 3000 hz and *MaxStepSize* to 0.001. The *RealTimeFactor* described as the multiple of MaxStepSize and RealTimeUpdateRate then becomes $RealTimeFactor = 0.001 \times 3000 = 3$. On a laptop configured with Intel core i7-9750H, Nvidia RTX-2070(mobile) and 32GB RAM, Gazebo was able to run the simulations at 3 times the speed of real-time.

All the simulation results mentioned in this thesis are obtained with these settings. If the *RealTimeUpdateRate* is set to zero, then the simulation runs as fast as possible depending on the hardware configuration. This is not advisable, as it is observed that the *RealTimeFactor* does not stay constant and varies rapidly through out the simulation leading to inaccurate results. Therefore depending on the configuration of the device, the *RealTimeUpdateRate* needs to be changed so that a constant *RealTimeFactor* is maintained. Figure 4.2 shows the virtual UAV(betaX) drawing a circular trajectory on the virtual target surface in the Gazebo simulator. The trajectory drawn can be of any shape and size, a circle was chosen in this task as it provides a challenge to the controller in compensating the changes in motion along the axes parallel to the plane of the target surface i.e along y-axis and z-axis(as shown in Figure 4.2). Depending on the task, the circular trajectory can be replaced with any shape that mimics the task. For example, if implemented in a window cleaning task the trajectory can be set to a square or rectangle to cover the areas of window more efficiently.

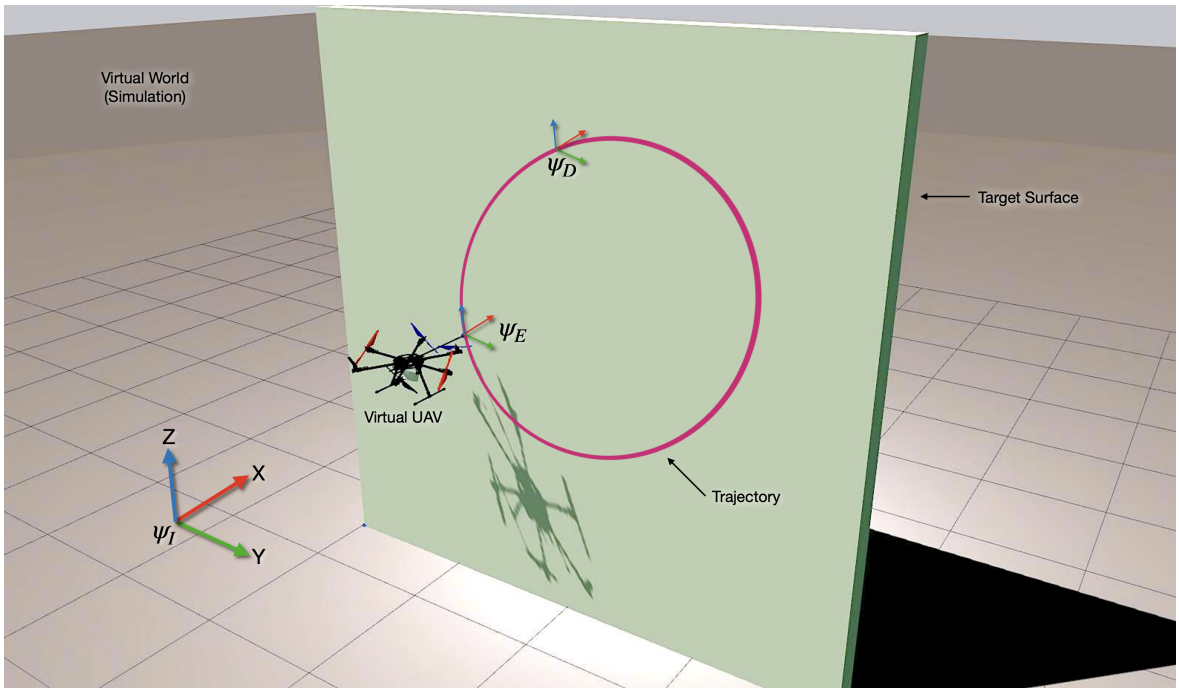


Figure 4.2: Graphical representation of UAV drawing a circular trajectory on the target surface in Gazebo simulator

Bayesian optimization

Once the simulation is setup with the virtual UAV(with same parameters as its physical counterpart) and the virtual target surface, Bayesian optimization is initiated to find the optimal controller gains. However, Bayesian optimization comes with its own set of design choices as described in Chapter 2. Firstly the surrogate model, a *gaussian process(GP)* with zero mean was chosen due to its natural way of providing uncertainty. A commonly used popular kernel function called *Radial-basis function(RBF)* is used as the kernel to the GP. Lastly as for the acquisition(utility) functions, *Probability of Improvement(POI)* is chosen as it provides better balance between exploration and exploitation behaviours compared to *Expected Improvement(EI)* and *Upper Confidence Bound(UCB)*. A summary of all the chosen parameters of Bayesian optimization is shown in Table.4.1.

Parameter	Value
Surrogate Model	Gaussian Process(GP)
Kernel Function	Radial-basis function(RBF)
Acquisition Function	Probability of Improvement(POI)
Hyperparameter - GP Noise σ^2	10^{-4}
Hyperparameter - POI ξ	0.01
Domain χ of target function $f(x)$	$[0.01, 100]$

Table 4.1: Table: Bayesian optimization parameters

The task of choosing appropriate acquisition function is much more complicated than it seems. The choice of POI along with its hyperpameter is highly inspired from the work of *Asem Khattab*[31]. The author used ten randomly generated smooth functions to compare and study various acquisition functions and how they are effected with changes in their hyperparameters. At lower dimensions, improvement based functions POI and EI favour exploration as long as there is uncertainty in some areas of the surrogate model. Therefore by controlling the desired improvement, they start with exploration and eventually turns towards exploitation. On the other hand, UCB starts with exploitation at low rewards and slowly progresses towards exploitation. This has a major drawback of getting stuck in local maxima. At higher dimensions, EI performs worse and requires more iterations to surpass the maximum given by POI, therefore the POI was considered to be a better option for the current task of trajectory drawing on unknown surfaces using an UAV. The maximum achieved by the POI at higher hyperparameter ξ values is much lower compared to that of with lower ξ. This is because of the search being global and fast with less steps(exploration) at higher ξ leading to the maximum being overlooked. While lower ξ results in search being slow due to more steps(exploitation), a mid value produces the best results by balancing the exploration and exploitation. This is just a very brief description of the reasoning behind the choices of acquisition functions and its hyperparameters ξ, for a detailed explanation refer to the work of *Khattab*[31].

4.2 Controller Parameters

An impedance controller was chosen to deal with the interactions between the betaX and the target surface. The impedance controller has a total of 24 parameters[1] that has to be chosen in-order to achieve smooth interactions with the surface and therefore achieve smooth trajectories drawn by the UAV. Since the target surface is unknown, the tuning of these controller parameters becomes a difficult task. However, not all controller parameters contribute to the task at hand. Therefore depending on the interaction task at hand, one can choose the controller parameters with greater

influence on executing the task and find a way to tune them for accurate results.

For the case of simplicity, the parameters of the impedance controller are divided into three - controlled parameters, fixed parameters and linked parameters. As their names suggests controlled parameters are the parameters that highly influence the trajectory drawing and are controlled via tuning. On the other hand, the fixed parameters are fixed with certain values and remain constant for all iterations of the task. Linked parameters are linked with other controlled or fixed parameters and have the same values as them. Since the co-stiffness gains can be derived from the stiffness gains, there is no need consider them for optimization. The coupling spring gains are set to zero for simplicity as the task at hand involves no rotation. Out of 24 impedance controller parameters, the task of drawing a circle on target surface using an UAV is influenced by 12 parameters - 3 translational stiffness gains k_t, 3 rotational stiffness gains k_r and 6 damping gains k_v.

Therefore, the 24 controller parameters are reduced to just 12 by including the expert knowledge of the task at hand. If the task is not related to a UAV drawing trajectories on surfaces then the 12 chosen controller parameters may not be true. The number of controller parameters that influence the task may be more or less than 12. The reason behind the inclusion of expert knowledge of the task at hand is to limit the search space of the maximum reward and therefore speed up the learning process of Bayesian optimization. The default values for the controller parameters based on the expert knowledge of the task are given in Table 4.2. where $diag(\bullet)$ are the diagonal terms of the matrix. k_t, k_r are the translational and rotational stiffness gains. k_v is the column vector representing the translational and rotational damping gains. μ_{TS} is the friction coefficient of the virtual target surface. F_N is the minimum normal force to be applied by the UAV on the target surface. The remaining parameters are set to zero for simplicity as they do not influence the current interaction task at hand.

Parameter	Value
$\text{diag}(k_t)$	$[6, 6, 6]$
$\text{diag}(k_r)$	$[10, 10, 10]$
k_v	$[5, 5, 5, 8, 8, 8]^\top$
μ_{TS}	1.16
ξ_D^E	0.15 m
F_N	2 N

Table 4.2: Table: Default controller parameters

[1]Impedance controller gains are as follows:

- 3 Translational stiffness gains
- 3 Rotational stiffness gains
- 6 Damping gains
- 3 Translational co-stiffness gains
- 3 Rotational co-stiffness gains
- 6 Coupling spring gains

4.3 1D Optimization

Using the expert knowledge of the task at hand, the 12 controller gains can further be reduced to a single controller gain with the help of linked parameters and fixed parameters. The reason behind optimizing a single controller parameter is not only to achieve a faster learning process but also to better understand the various concepts involved in the Bayesian optimization. The 1D optimization problem is especially used to make important design choices such as balancing the reward function, choice of acquisition functions and tuning of their hyperparameters. For this 1-Dimensional(1D) optimization, the translational stiffness gains along x-axis k_{tx} is considered as a fixed parameter and is set to a constant value of 20 so that a minimum normal force of 2N is always applied by the UAV on the target surface. This is achieved through a set of trial-and-errors as the normal force applied by the UAV is influenced by the translational stiffness gains along x-axis k_{tx} and also by the model of the target surface in the Gazebo simulator. Since the trajectories are drawn in YZ plane, the controller gains k_{ty} and k_{tz} has to be of equal importance in order to achieve an accurate circular trajectory. Therefore, the translational stiffness gain k_{ty} is considered as the controlled parameter along with a linked parameter k_{tz}. Not linking k_{tz} to k_{ty} results in restricted movement along z-axis if k_{tz} is set to a small value. Therefore, the goal of 1D optimization is to find the optimal controller gain K whose configuration is given by:

$$K = k_{ty} = k_{tz} \; ; \quad k_{tx} = \text{fixed} \tag{4.1}$$

4.3.1 Optimal gains of the groundtruth target function

The output(reward) of the target function is a one-dimensional metric that represents the performance of the task done. This reward is generated by a function called reward function $R(x)$ as described in Chapter 3. The reward function generates the reward by considering various parameters of the UAV or the task being done or both. It is a design choice to be made depending on the task at hand and the requirements of the user. For our case of trajectory drawing on unknown target surfaces by an UAV, the reward function considers the kinetic energy and potential energy(described in Section 3.2) of the impedance controller in-order to generate the appropriate reward. This process of designing the reward function according to the task at hand is called reward shaping.

It is already mentioned that the reward shaping is task dependent and needs to be changed/adjusted if the task changes. The proposed reward function for the task of trajectory drawing is given in Equation 3.9. A groundtruth scenario is developed where the UAV is tasked to draw the circular trajectory for every instance of input controller parameter $\mathbf{K}$ within its domain. The groundtruth is generated in an effort to verify the maximum reward found by the Bayesian optimization. This visualization of target function also used for refining the quality of reward function and in-turn the rewards itself. For this experiment, the input $K = k_{ty}$ is varied with a step size of 1 over its domain $[0.1, 100]$. The reward function mentioned in Equation 3.8 without any balancing factor is used by setting the constant c_1 to 1. The plot of target function with unbalanced rewards is shown in Figure 4.3. Thanks to the 1D optimization, the generated groundtruth target function is used to study the behaviour of the reward function and modify accordingly.

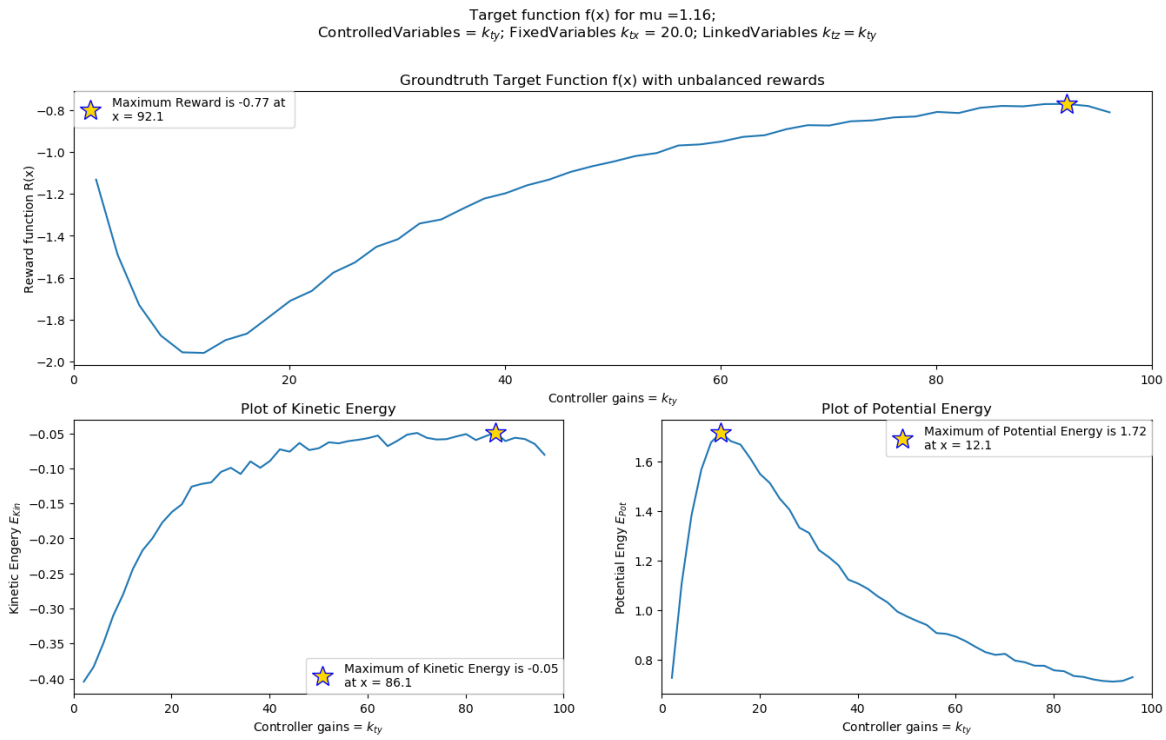
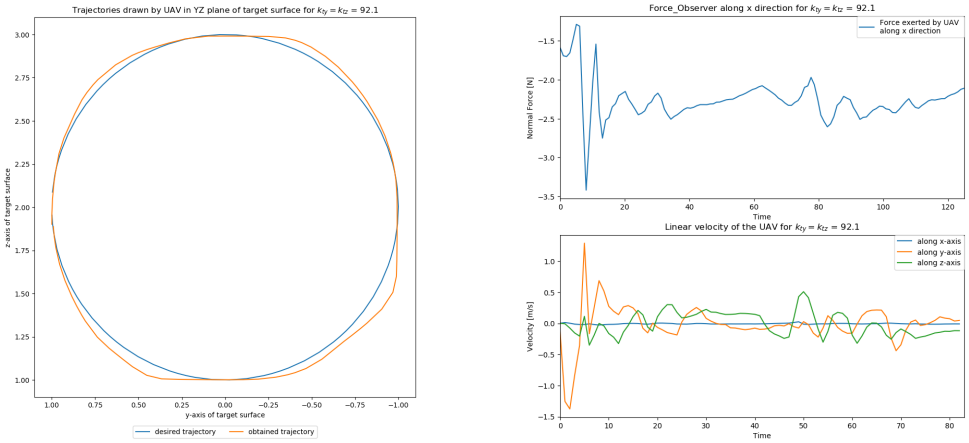


Figure 4.3: Plot of groundtruth target function with unbalanced rewards(Top) with maximum reward(Gold Star) observed at $k_{ty} = 92.1$; Plots of Kinetic Energy(Bottom Left) and Potential Energy(Bottom Right)

The figure shown above is the plot of the generated rewards(top) at corresponding controller gains $K = k_{ty}$ in the interval $[0.1, 100]$. It can be seen that the maximum reward is -0.77 and is observed at $k_{ty} = 92.1$ which is undesirable due to unstable behavior of UAV at higher gains. The trajectory drawn by the UAV with $k_{ty} = 92.1$ is shown in Figure 4.4a and is quite close to the desired trajectory. However, these higher gains come with a huge flaw that can be explained through the plot of normal force applied by the UAV and its linear velocity shown in Figure 4.4b. The controller tries to achieve an accurate trajectory at the expense of exerting high forces on the target surface wasting lot of energy. From Figure 4.4b, it is evident that the UAV had trouble maintaining a constant normal force on the surface. The spikes in linear velocity along y-axis during the same time states that the UAV was fluctuating and is highly unstable. It can also be seen that the target function(top) has the inverse shape as the potential energy(bottom right). This is because the scale of kinetic energy(bottom left) is very small compared to that of potential energy making it a tiny constant in comparison. This makes the influence of kinetic energy on the reward almost negligible.

In order push the influence of kinetic energy on to the reward, the kinetic energy is multiplied by a constant in an effort to upscale it to match the scale of the potential energy. The idea is that when the reward function is calculated as the sum of potential energy and up-scaled kinetic energy, the maximum reward achieved is pushed towards lower controller gains because of more vibrations(high kinetic energy) at higher controller gains. However as mentioned in Chapter 3, one of the main reasons to move away from the reward function(Equation 3.4) by *Khattab*[18] is the choice of arbitrary constants that needs to be optimized for every surface. These constants add another layer of optimization(optimal weights) problem to the already existing optimization(optimal controller gains) problem. Therefore in an effort to eliminate the constants/weights from the reward function, they



(a) Trajectory drawn in YZ plane for $k_{ty} = 92.1$ (b) Normal force, Linear velocity for $k_{ty} = 92.1$

Figure 4.4: Performance of UAV for $k_{ty} = 92.1$

are modified to be a function of reward per environment(target surface). For a given target surface, the mean of the potential energy and the mean of the kinetic energy over all the samples of target function is calculated separately. The constant c_1 is now represented as the ratio of mean of potential energy $E_{Pot}(t)$ and that of kinetic energy $E_{Kin}(t)$. The modified reward function is given as Equation 4.2 for a given target surface. The difference between this reward function and the one mentioned in Equation 3.9 is that the constant c_1 in the former is calculated for all observations in the domain $[0.1, 100]$ while only the initial observations are considered in the later.

$$R(x) = c_1 \int_t E_{Kin}(t)dt + \int_t E_{Pot}(t)dt \ ; \qquad c_1 = \frac{\text{mean}(E_{Pot}(t))}{\text{mean}(E_{Kin}(t))} \tag{4.2}$$

The plot of the new groundtruth target function with balanced rewards is shown above in Figure 4.5. As expected, the maximum reward is observed at a much lower gain of $k_{ty} = 46.1$ with the up-scaled kinetic energy compared to that of previous case(Figure 4.3) where the maximum reward is observed at $k_{ty} = 92.1$. The influence of the kinetic energy can also be seen in the target function by looking at the shape of the plot. This form of balanced reward function is immune to the changes in the trajectory shapes as well as the environments - the target surfaces. This plot of the target function also depicts the high vibrations(instability) that occur at higher controller gains. Although the kinetic energy and potential energy are considered with equal weights in this case, this balance of accuracy and smoothness of the trajectories drawn can be easily changed in the reward function(Equation 4.2) if required by the task. This plot is considered as the True-Data or groundtruth target function and is used to compare with the approximate target function produced by the Bayesian optimization. The goal of the Bayesian optimization is to find the maximum of this groundtruth target function which should be around $k_{ty} = 46.1$.

The trajectory drawn by the UAV at 1D controller gain $k_{ty} = 46.1$ is shown in Figure 4.6a. Although there is continuity in trajectory, it is not as error-free as the one shown in Figure 4.4a obtained for $k_{ty} = 96.1$. However, the UAV is more stable compared to the previous case shown in Figure 4.4b as there are no sudden fluctuations in the linear velocity and applied normal force on the target surface as seen in Figure 4.6b. Therefore, $k_{ty} = k_{tz} = 46.1$ are the optimal controller gains that provide the satisfactory trajectories by balancing the accuracy and smoothness of the trajectories drawn by the UAV. This is the exact motivation behind designing the reward function described in

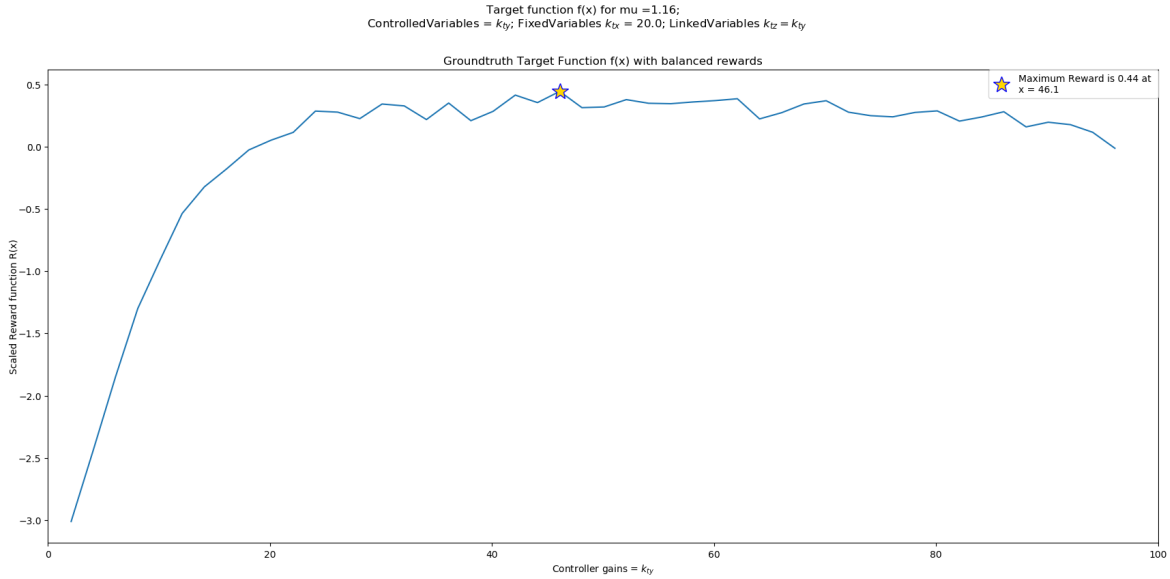
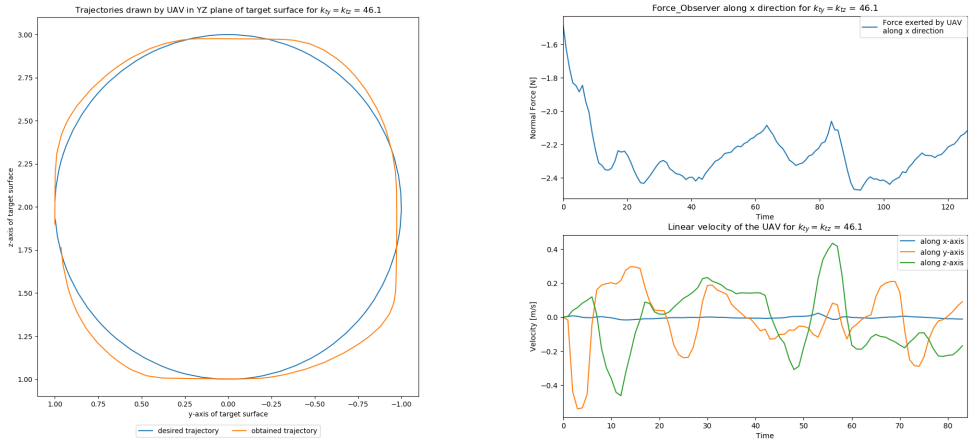


Figure 4.5: Plot of groundtruth target function with balanced rewards. This plot shows the maximum reward(Gold Star) is observed at $k_{ty} = 46..1$

Chapter 3. It should be noted that this may not be the true maximum of the target function because the target function is generated with the step size of 0.1. Decreasing this step size results in more accurate maximum reward and therefore more accurate optimal controller gains. While increasing the step size makes it inaccurate. However, it should be noted that the purpose of this groundtruth target function is to adjust the balancing factor of the reward function and to verify the maximum reward achieved by the Bayesian optimization.



(a) Trajectory drawn in YZ plane for $k_{ty} = 46.1$ (b) Normal force, Linear velocity for $k_{ty} = 46.1$

Figure 4.6: Performance of UAV for $k_{ty} = 46.1$

4.3.2 Optimal gains of Bayesian optimization

It is mentioned earlier in Chapter 2 that the Bayesian optimization requires few initial observations to build a prior distribution. Therefore, 10 random initial observations are chosen before the learning process begin. These can also be set manually which actually help in a better search for the maximum reward and thereby optimal controller gains. Since, we will need to repeat the process for a number of times for the preparation of the custom training dataset, random initial observations are chosen in order to reduce human involvement as much as possible. Based on these initial observations, the constant c_1 of the reward function is calculated according to Equation 3.9. The Bayesian optimization package has been modified in-order to accommodate the proposed reward function. Once all the initial observations are collected, the c_1 is calculated according to Equation 3.9 and the initial observations are sampled again but this time using the balanced(rescaled) rewards. These initial observations with balanced rewards are used to build the initial surrogate model and is shown in Figure 4.7. This initial surrogate model act as the prior distribution through which the next best point to sample is selected via the acquisition function(POI). The next observation is carried out at this point and the process continues for 40 iterations. These observations are called learning observations as the gaussian process(surrogate model) is learning about the regions of the target function with every observation. After 50 observations(10 initial and 40 learning observations), the observed rewards along with its maximum are plotted in Figure 4.9. An intermediate result of Bayesian optimization is shown in Figure 4.8 taken after 25 observations(10 initial and 15 learning observations). The groundtruth target function shown in Figure 4.5 is superimposed on all these observations for a comfortable visual comparisons. It can be seen that the target function predicted by the Bayesian optimization is being improved(achieving the shape of groundtruth target function) as the number of observations increase. The drawn trajectory(Figure 4.10a), linear velocities of the UAV and the applied normal force on the surface(Figure 4.10b) are all similar to that of groundtruth(Figure 4.6a and Figure 4.6b).

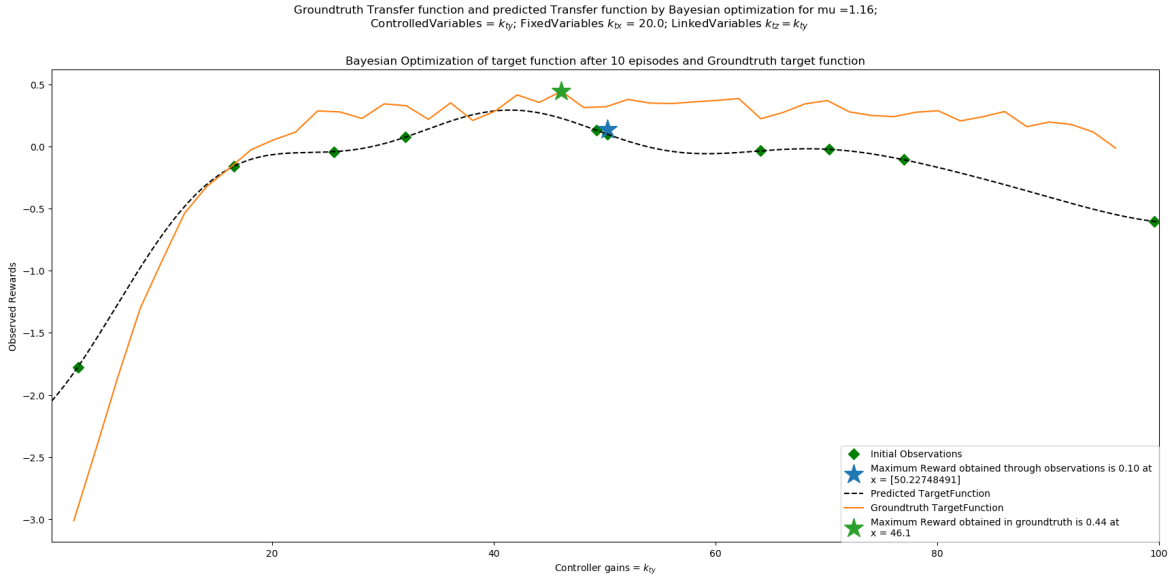


Figure 4.7: Plot of 10 initial observations of Bayesian optimization superimposed on groundtruth shown in Figure 4.5. This plot shows that the maximum reward(Blue Star) found by bayesian optimization is close to the maximum reward of the groundtruth(Green Star)

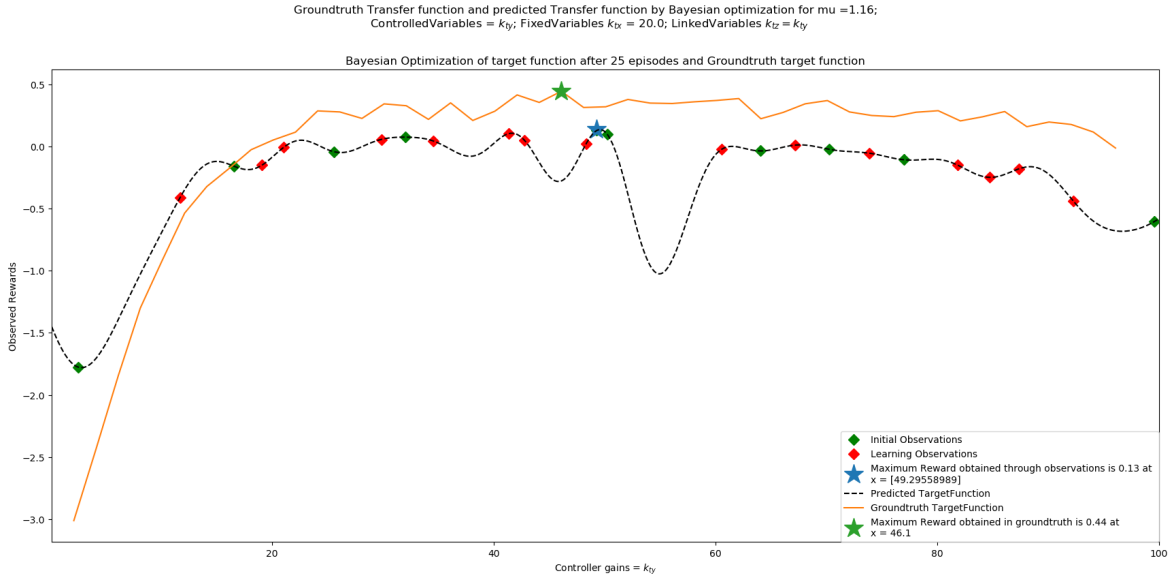


Figure 4.8: Plot of 25 observations of Bayesian optimization superimposed on groundtruth shown in Figure 4.5. This plot shows that the maximum reward(Blue Star) found by bayesian optimization is close to the maximum reward of the groundtruth(Green Star)

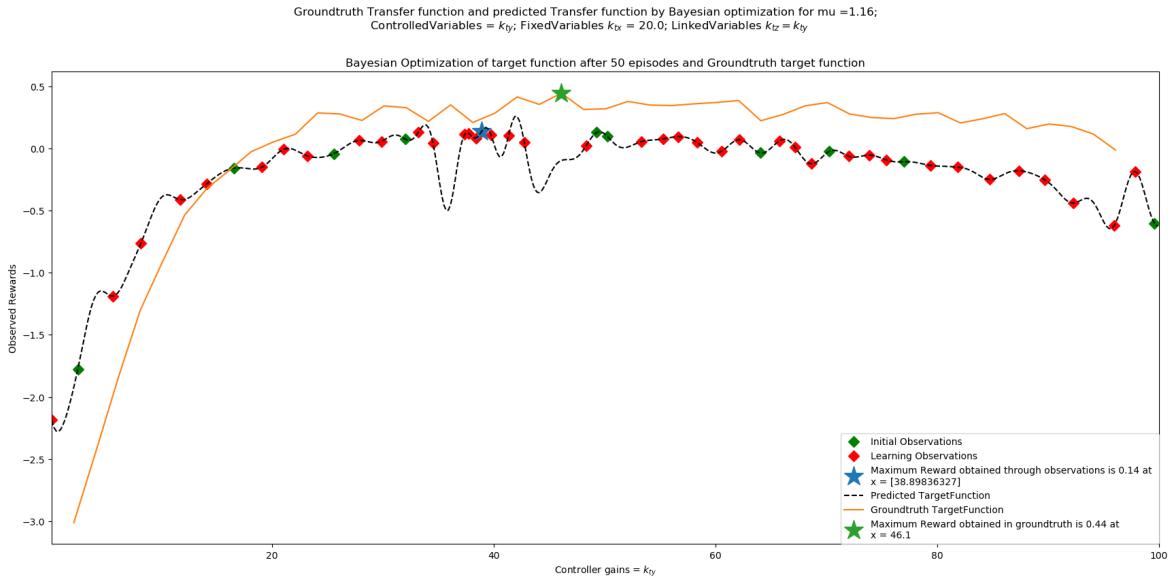
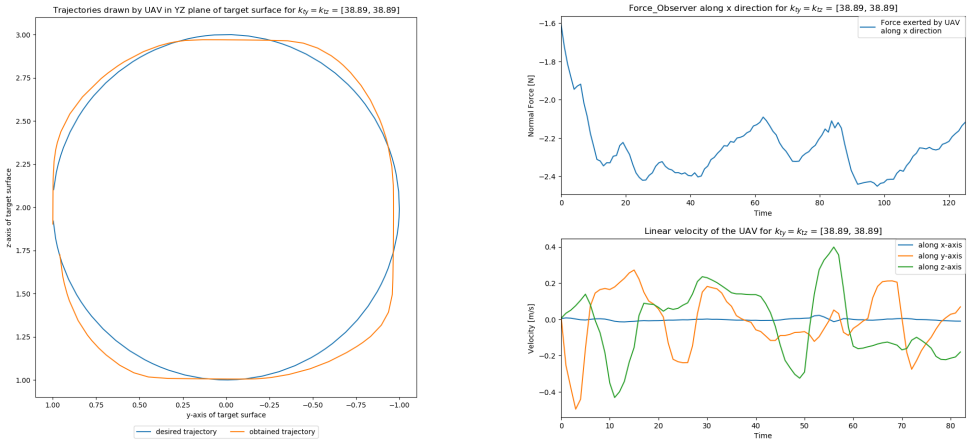


Figure 4.9: Plot of 50 observations of Bayesian optimization superimposed on groundtruth shown in Figure 4.5. This plot shows that the maximum reward(Blue Star) found by bayesian optimization is close to the maximum reward of the groundtruth(Green Star)



(a) Trajectory drawn in YZ plane for $k_{ty} = 38.89$ (b) Normal force, Linear velocity for $k_{ty} = 38.89$

Figure 4.10: Performance of UAV for $k_{ty} = 38.89$

It can be seen that the shape of the approximate target function produced by the Bayesian optimization(dotted curve) is similar to that of the groundtruth plot(orange curve) shown in Figure 4.5 with low rewards at low gains and gradually increasing rewards towards larger gains and then decreasing at the end. Although the learning observations were able to match the shape of the groundtruth curve, it can be seen that the observations are at an offset to the groundtruth. This is expected because the groundtruth is generated from the reward function where all the observations(100 observations) are considered for calculating c_1(Equation 4.2). Where as in Bayesian optimization, only the initial observations(10 observations) are considered for calculating c_1 to generate the approximate target function therefore the offset. The more the provided initial observations are, the closer the plot of Bayesian optimization target function get to that of groundtruth minimizing the offset. However, it should be remembered that the goal of the Bayesian optimization is not to reconstruct the target function, but to find the maximum(or minimum) of it. This is achieved through the learning observations since the maximum reward of Bayesian optimization is observed to be at $k_{ty}^{BO} = 38.89$ which is close to the one found in the groundtruth where the maximum reward is observed at $k_{ty}^{GT} = 46.1$. It should be noted that the groundtruth is sampled with a step-size of 1 therefore the maximum achieved by groundtruth(Figure 4.5) is not completely accurate but only an approximate location of the maximum reward. The same goes for the maximum found through Bayesian optimization(Figure 4.9) is an approximate location of maximum reward.

In order to analyse the performance of Bayesian optimization in finding the optimal controller gains, certain measures are to be observed. The maximum reward y_n^+ achieved in n observations(initial + learning) is given by $y_n^+ = max(y_{1:n})$. This measure of maximum reward describes how quickly the maximum reward is found and is useful in comparing the Bayesian optimizations with various parameters. An increasing y_n^+ plot shows that a new maximum reward is found for every iteration while a constant plot shows that no new maximum is found. The average of all the rewards $\bar{y_n}$ obtained so far suggests if the Bayesian optimization often goes into the areas of low rewards. More low rewards leads to a lower average reward which is not desirable as the Bayesian optimization is wasting time and effort trying to find the maximum reward in an obviously wrong region probably a local maximum. This average rewards is given by $\bar{y_n} = \sum_{i=1}^{n} \frac{y_i}{n}$. A decreasing $\bar{y_n}$ plot can be used as a warning sign and the learning parameters can be changed to avoid low rewards. The final one is the measure of euclidean distance between current observation and closest past observations. This is given by $d_n = min(\sqrt{x_n - x_i})_{i=1}^{n-1}$ and tells if the search for maximum reward is local or global. High

values d_n means the current observation is far away from its nearby observations therefore the search is global. While the low values of d_n translates to the search being local - exploitation behaviour, high values of d_n means the search is global - exploration behaviour. The plots of performance measures for the observations of Bayesian optimization are given in Figure 4.11.

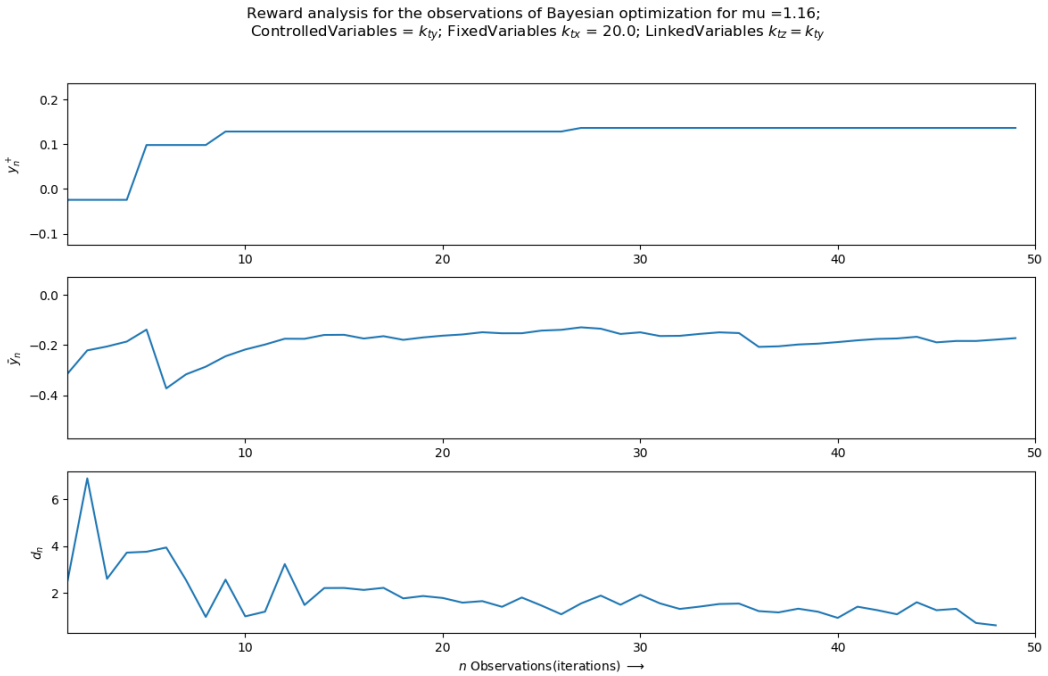


Figure 4.11: Plots for Bayesian optimization(without noise) **Top** - Maximum of rewards y_n^+ achieved so far, **Middle** - Average of rewards $\bar{y_n}$ achieved so far, **Bottom** - Euclidean distance d_n between current and closest previous observation

From Figure 4.11, it can be inferred that the maximum reward was achieved during the 10 initial observations itself and therefore the plot remains almost constant for the rest of the observations. However, a slightly better maximum reward is achieved at around 28^{th} observation. This constant plot is not a problem because it is not always the case that the maximum reward is achieved within the initial observations. With another set of random initial observations, a very actively increasing maximum reward y_n^+ plot might be observed. For this reason, we ignore the values from 0-10 initial observations as they are randomly chosen and no learning takes place at these observations. Looking at the plot of average rewards $\bar{y_n}$ after 10 initial observations, an increasing pattern can be seen. This means the Bayesian optimization algorithm is constantly looking for higher rewards and not getting struck in local minimum. Lastly, the plot of d_n has huge spikes at under initial observations and tries to become constant as the observations increase. This is because of exploratory behaviour(observing points away from the previous ones) at the first few observations until the algorithm is confident about maximum reward and then changes into exploitative behaviour(observing points close to previous observations for precision) towards the end. This analysis is followed in choosing the acquisition function and its hyperparameter for the current task of drawing the circle on unknown surfaces.

4.4 Domain Randomization

A brief description of Domain randomization is given in Chapter 1 and is implemented in simulations in this section. The idea is to generate as many unique simulation scenarios as possible to train the neural networks. This is because when the neural networks are fed with the data from real world, they can relate it to the data from any one of the simulation scenarios generated as part of domain randomization. The current implementation of gazebo simulation does not consider factors like noise, wind directions, and other aerodynamic factors that are obvious in real-world. All these factors can participate in domain randomization and their parameters can be varied to generate diverse simulation scenarios.

Since it is a challenging task to model these aerodynamic effects into the gazebo simulator, external forces can be added to the UAV in an effort to replicate some real world disturbances. These can either be applied as body forces of the UAV or the propeller forces. In this thesis, random noise is added to the propeller forces of the UAV and the noise parameters are varied as part of domain randomization. The rotor thrust force F_T is generated as given by:

$$F_T = \omega^2 C_T . e_{z_B} \tag{4.3}$$

where, ω is the angular velocity of the propeller, C_T is the rotor thrust constant and e_{z_B} is the unit vector pointing in the z-direction in the rotor's body frame.

This rotor thrust is implemented in the RotorS package as discussed in Section 4.1. To this rotor thrust a random Gaussian noise is added in-order to replicate the external disturbances in the simulations. This is implemented by mapping the propeller angular velocity to the variance of Gaussian distribution through an exponential curve as shown in Figure 4.12. Thanks to the experiments conducted at RAM lab in University of Twente, the thrust generated by the propeller of the betaX at several angular velocities was taken as reference. The variance of this thrust data is calculated and the maximum observed variance was set as the maximum variance of the exponential curve. This variance, along with a zero mean is used to generate random gaussian noise and is added to the rotor thrust given in Equation 4.3. Since the added noise is the function of angular velocity, a greater noise is added to the rotor thrust when the propellers spin at high speeds and a lower noise is added when the propellers spin at lower speeds. This would result in unstable behaviour of the UAV when the propellers spin at its maximum speed and would generate highly undesirable lower rewards for the bayesian optimization.

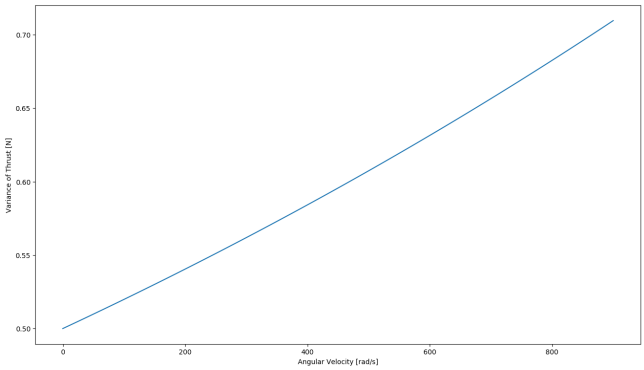


Figure 4.12: Plot of variance of thrust generated Vs angular velocity of the propeller that is used as a reference to generate random gaussian noise as a function of angular velocity of the propellers

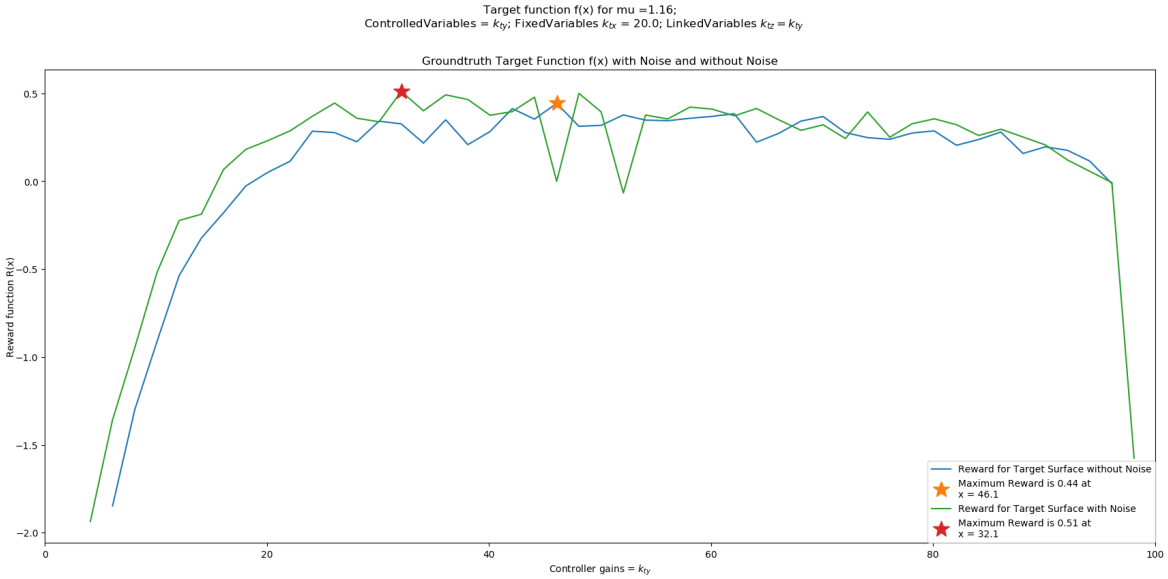
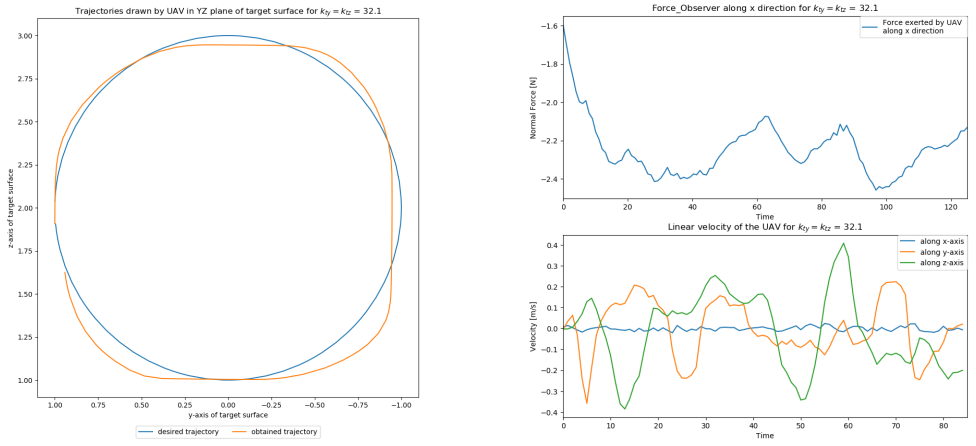


Figure 4.13: Plot of groundtruth(without Noise) target function superimposed on groundtruth(with Noise) target function

The groundtruth target function generated using the noise function mentioned above is shown in Figure 4.13. The original groundtruth target function without noise(Figure 4.5) is superimposed in-order to compare the curves of groundtruth with and without noise. As expected, the groundtruth curve with noise shows lower rewards at higher gains as more noise is added when the propellers of the UAV spin at higher RPM. The maximum reward of the groundtruth(with noise) target function is observed to be at $k_{ty} = 32.1$ and is lower than the one achieved by the groundtruth(without noise) which is at $k_{ty} = 46.1$. The trajectory drawn by the UAV is shown in Figure 4.14a whereas the normal force applied by the UAV and its linear velocities are shown in Figure 4.14b. The trajectories drawn at $k_{ty} = 32.1$ have slightly better accuracy and smoothness compared to the ones at $k_{ty} = 46.1$ shown in Figure 4.6.



(a) Trajectory drawn in YZ plane for $k_{ty} = 32.1$ (b) Normal force, Linear velocity for $k_{ty} = 32.1$

Figure 4.14: Performance of UAV for $k_{ty} = 32.1$

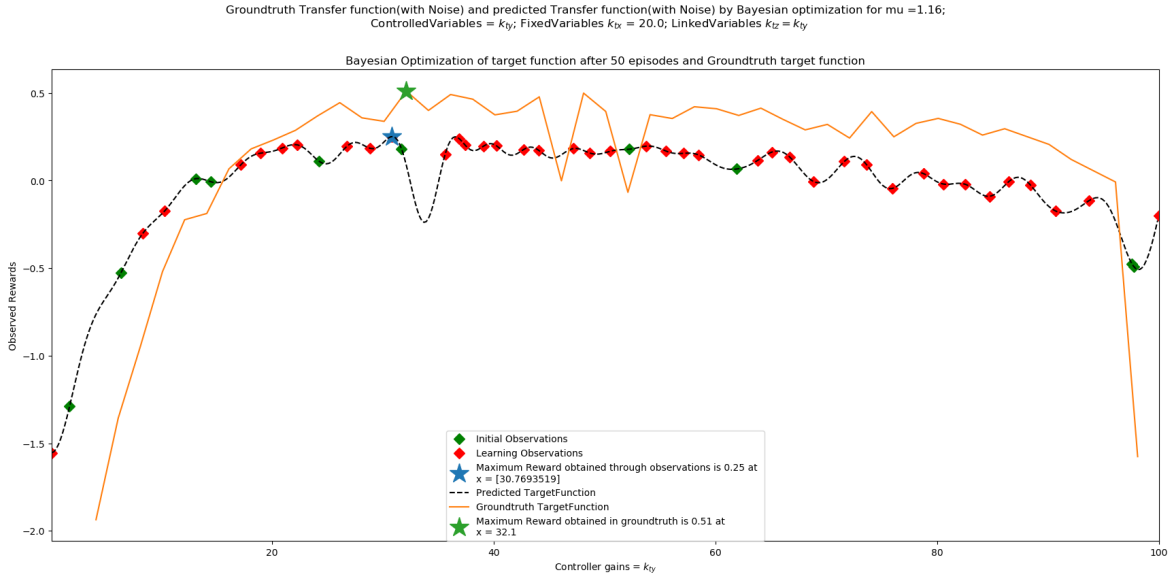
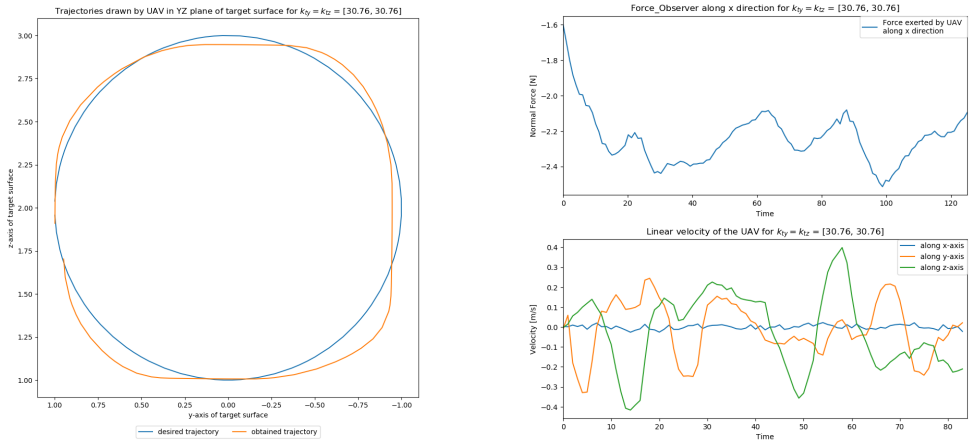


Figure 4.15: Plot of observations of Bayesian optimization superimposed on groundtruth with added noise function

Figure 4.15 shows the plot of groundtruth with added noise function(as shown in Figure 4.12) superimposed on the observations from Bayesian optimization which are also carried out with the same noise parameters. It can be seen that the Bayesian optimization still works with the added noise and the observations follow the curve of groundtruth(with noise). The maximum reward achieved at $k_{ty} = 30.76$ through Bayesian optimization is also close to that of groundtruth $k_{ty} = 32.1$. The trajectory drawn at $k_{ty} = 30.76$ and normal force applied on the surface are shown in Figure 4.16a and Figure 4.16b respectively. As expected, $k_{ty} = 30.76$ also maintains the balance between accuracy of the trajectory and stability of the UAV. Adding a noise function(Figure 4.15) also brought the maximum reward to be achieved at much lower gains $k_{ty} = 30.76$ compared to without noise $k_{ty} = 38.89$ in Figure 4.9.



(a) Trajectory drawn in YZ plane for $k_{ty} = 30.76$ (b) Normal force, Linear velocity for $k_{ty} = 30.76$

Figure 4.16: Performance of UAV for $k_{ty} = 30.76$

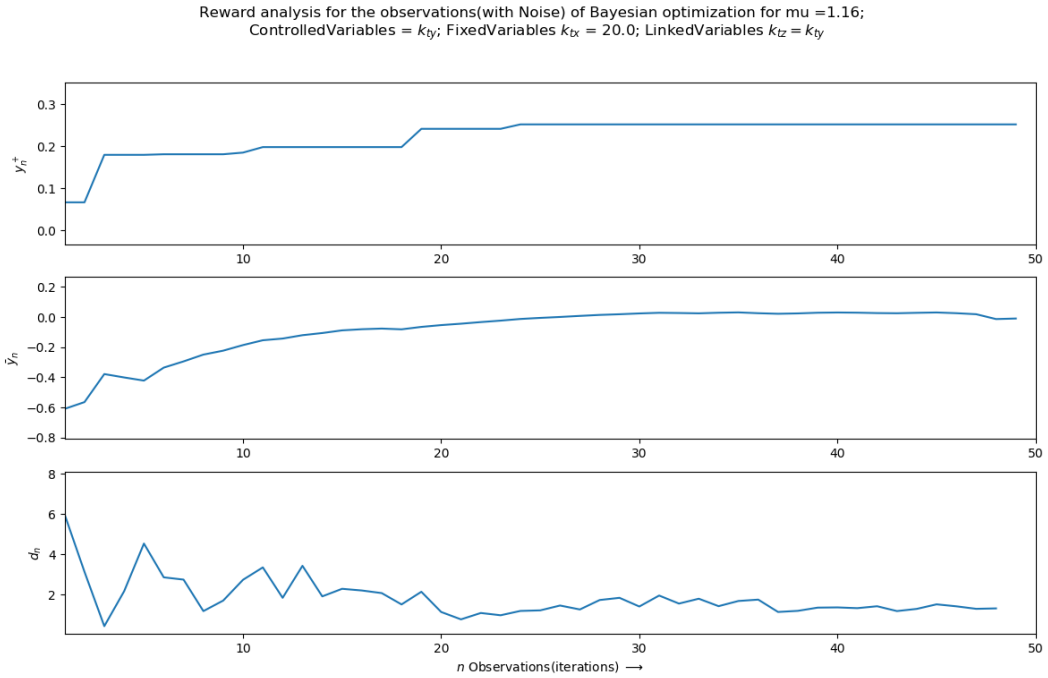


Figure 4.17: Plots for Bayesian optimization(with noise) **Top** - Maximum of rewards y_n^+ achieved so far, **Middle** - Average of rewards $\bar{y_n}$ achieved so far, **Bottom** - Euclidean distance d_n between current and closest previous observation

The performance of the Bayesian optimization can be analyzed through the measures of maximum rewards y_n^+, average rewards $\bar{y}_n$ and the Euclidean distance d_n between current observation and closest previous. The plots of these measures are shown in Figure 4.17. Similar to the Bayesian optimization without noise(Figure 4.11), the maximum reward y_n^+ is almost achieved in the initial observations with the more precise maximum achieved at around 24^{th} observation. The increasing $\bar{y}_n$ curve says that the Bayesian optimization is always choosing the points with higher rewards to sample. The highly fluctuating behaviour of the d_n curve until 20 observations says that the Bayesian optimization is showing exploratory behaviour meaning the next point to sample is selected far away from the current sampled point. The almost constant d_n curve after 30 observations depicts the exploitative behaviour meaning the next point to sample is selected close to the current sampled point.

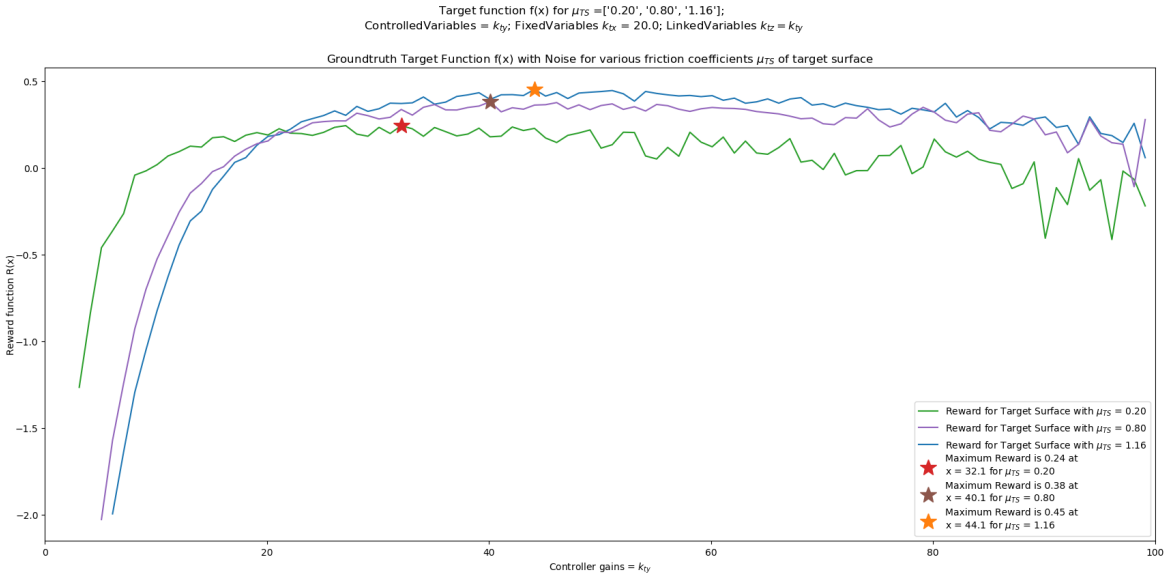


Figure 4.18: Plot of target functions for various friction coefficients of target surface and with added noise function

The noise function can further be tweaked by varying the extreme values of the variance and the steepness of the curve shown in Figure 4.12 contributing to the cause of domain randomization. Another parameter considered for domain randomization in this thesis is the friction coefficient μ_{TS} of the virtual target surface. Several virtual target surfaces are created in the Gazebo simulator by varying the friction coefficients of the target surface.

Figure 4.18 shows the plot of groundtruth target function $f(x)$ for various friction coefficients μ_{TS} of the virtual target surface. The simulations also include the noise function as shown in Figure 4.12. It can be seen that the curves with with lower μ_{TS} are able to reach the areas of maximum reward quickly whereas those with higher μ_{TS} reach slowly. Due to this, the maximum reward is achieved at much lower gains for target surfaces with lower friction coefficients compared to the surfaces with higher μ_{TS}. For $\mu_{TS} = 0.20$, the maximum reward is observed at $k_{ty} = 32.1$ whereas it is observed to be at $k_{ty} = 44.1$ for $\mu_{TS} = 1.6$. It can also be seen that all the curves experience huge fluctuations at higher gains due to the modelled noise at the propeller level of the UAV. This makes the higher gains highly undesirable.

In-order to compare the trajectories drawn by the UAV for various friction coefficients μ_{TS}, a trajectory error metric is calculated for the optimal controller gains from Figure 4.18 and is shown in Table 4.3. The trajectory error metric is the error between the desired trajectory and obtained trajectory in YZ plane. The norm of this error is calculated over the duration of the trajectory. Similarly, the norm of the linear velocities of the UAV along y-axis and z-axis are calculated and are mentioned using the open brackets '( )' in the Table 4.3. Since the noise function is added to the simulation, the results are different for every simulation iteration. Therefore, the simulations are repeated 5 times for each controller gain and friction coefficient and the average values are mentioned in the table. The gains corresponding to the diagonal values(blue colored) provide the maximum reward and hence are the optimal gains for the given μ_{TS}. Although the trajectory error metric is low for $k_{ty} = 40.1$ and 44.1 for $\mu_{TS} = 0.20$, the norm of linear velocities of the UAV is minimum for only $k_{ty} = 32.1$. Since the reward function provides a balanced approach towards the accuracy and smoothness of the trajectories draw, the maximum reward is achieved at $k_{ty} = 32.1$ for $\mu_{TS} = 0.20$.

The same philosophy is followed for the scenarios of $\mu_{TS} = 0.80$ and 1.16.

μ_{TS}	Controller Gains k_{ty}		
	32.1	40.1	44.1
0.20	0.16(0.178)	0.14(0.187)	0.14(0.205)
0.80	0.27(0.203)	0.23(0.220)	0.20(0.234)
1.16	0.30(0.231)	0.27(0.241)	0.24(0.254)

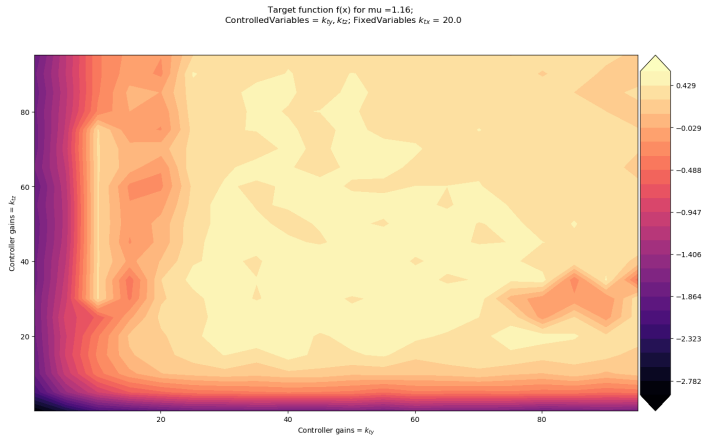
Table 4.3: Table depicting the errors of trajectories drawn by the UAV with various gains k_{ty} on surfaces with various friction coefficients μ_{TS}

It can be inferred from the Table 4.3 that the optimal controller gains calculated for one target surface are not suitable for other surfaces. These often result in inaccurate trajectories drawn by the UAV or heavy vibrations of the UAV while drawing the trajectories. These maximum rewards for surfaces with higher μ_{TS} can be shifted towards the preferred lower gains by tweaking the noise parameters accordingly. The data produced with various combinations of noise function parameters and surface friction coefficients act as domain randomization and contribute to the generation of the custom dataset.

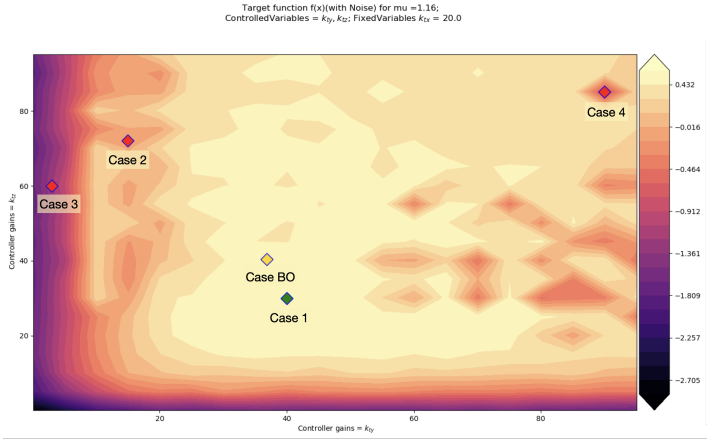
4.5 2D Optimization

The design choices of the Bayesian optimization such as the acquisition function and its hyperparameters are chosen with the help of various visualization plots of 1D optimization. Even the balancing of the reward function was done through the 1D optimization. Using the expert knowledge of the task at hand, a 2D optimization was done in this section where two controller parameters are chosen as the input of the target function. The idea is to use the Bayesian optimization with the same parameters used in 1D optimization and find the optimal values for two controller gains instead of one which was discussed in previous section. Instead of linking k_{tz} to k_{ty}, they are both set as controlled variables and are varied over the domain $[0.1, 100]$. This is done in-order to remove the movement restrictions along y-axis and z-axis while drawing the trajectories using the UAV. Varying both k_{ty} and k_{tz} will allow the Bayesian optimization to choose different values for both gains and prioritize one gain over the other in-order to achieve the most accurate trajectories. Therefore, the goal of 2D optimization is to find the optimal controller gains K_1, K_2 whose configuration is given by:

$$K_1 = k_{ty} \; ; \quad K_2 = k_{tz} \; ; \quad k_{tx} = \text{fixed} \tag{4.4}$$



(a) 2D groundtruth with out noise



(b) 2D groundtruth with noise

Figure 4.19: Groundtruth for 2D optimization problem with variables k_{ty}, k_{tz}

The groundtruth target function generated with a step size of 5 is shown in Figure 4.19 as a contour plot. Figure 4.19a shows the variant with out noise and Figure 4.19b shows the one with added noise function(according to Figure 4.12). It can be seen that a lot of local minima(darker regions) appeared in the target function with the addition of noise. In both variants of the 2D groundtruth target function, lower rewards are observed for the extreme values of gains k_{ty} and k_{tz}. The gains in the mid region of the domain $[0.1, 100]$ has the higher rewards meaning these gains result in the UAV drawing accurate and smooth trajectories on the target surface with friction coefficient $\mu_{TS} = 1.16$.

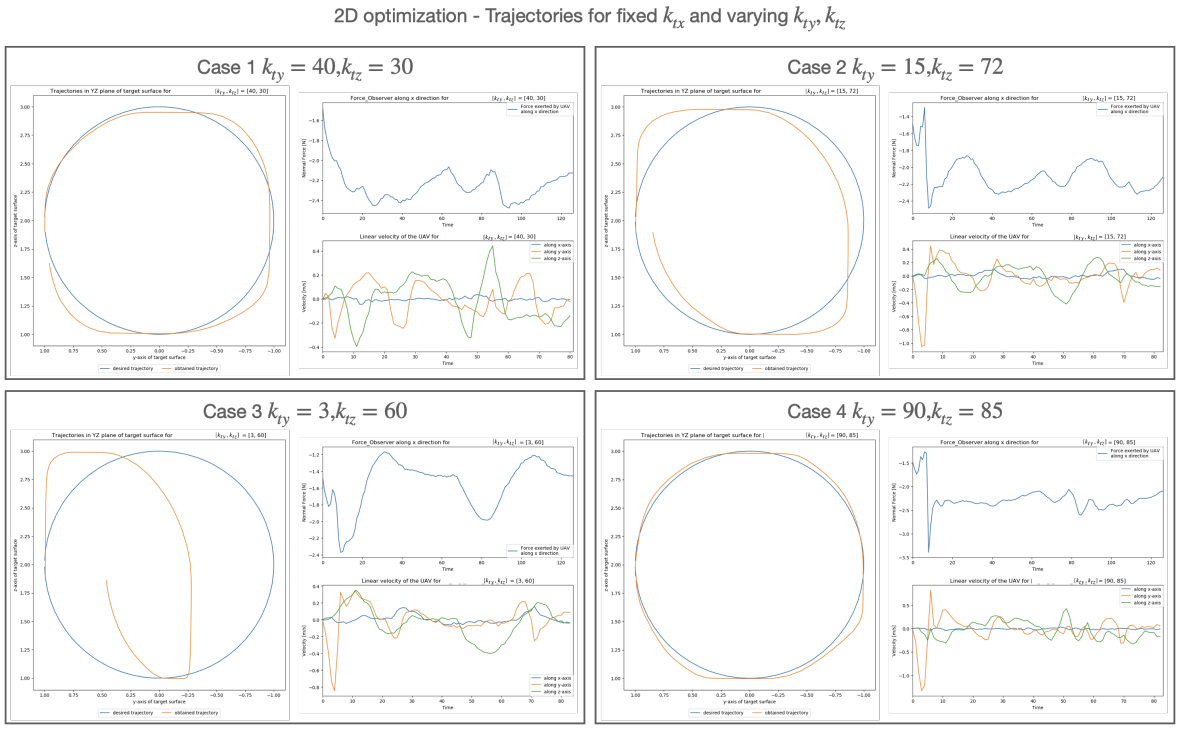
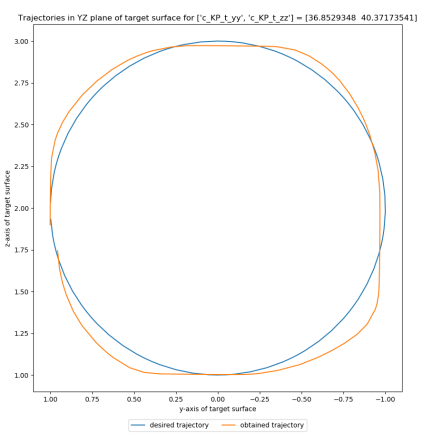


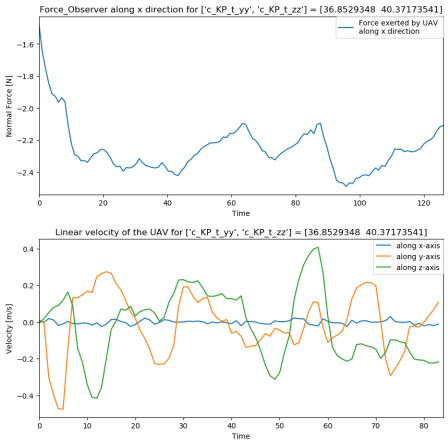
Figure 4.20: Various cases of random 2D controller gains demonstrating the drawn trajectories, applied normal forces and linear velocities of the UAV

Few random points are probed from the noisy groundtruth(Figure 4.19b) to check the corresponding trajectory and normal force being applied by the UAV onto the target surface and its linear velocity. These results are shown in Figure 4.20. Looking at the trajectories alone, case 4 with $k_{ty} = 90, k_{tz} = 85$ seems to be more accurate compared to the rest. However, after looking at the normal force and linear velocity of the UAV it can be seen that the UAV struggles to make contact with the surface and therefore results in more vibrations and high energy consumption(due to high gains) leading to unstable behaviour. Case 2 and Case 3 are self explanatory as the respective trajectories come with large errors. Case 1 with considerably lower gains is gives the best results by balancing between the accuracy of trajectory drawn and stability of the UAV during the execution of the task. The $k_{ty} = 40$ in Case 1 is of close proximity to the $k_{ty} = 30.76$ - the maximum reward achieved in 1D optimization problem with added noise function conveying that the plot shown in Figure 4.15 is accurate. Along with these cases probed from random points, the optimal controller gains found through the Bayesian optimization is marked in Figure 4.19b as Case BO with $k_{ty} = 36.85, k_{tz} = 40.37$. It can be seen that the coordinates of case BO lies in the vicinity of that of case 1 which are accepted to be satisfactory optimal controller gains. In order to verify if the the optimal gains provided by Bayesian optimization follows the same accuracy and stability, the corresponding trajectory, applied normal force and linear velocity plots are given in Figure 4.21. It indeed provide a balance between

the accuracy and smoothness of the trajectories drawn by the UAV.



(a) Trajectory drawn in YZ plane for $k_{ty} = 36.85, k_{tz} = 40.37$



(b) Normal force, Linear velocity for $k_{ty} = 36.85, k_{tz} = 40.37$

Figure 4.21: Performance of UAV for $k_{ty} = 36.85, k_{tz} = 40.37$

4.6 4D Optimization

1D optimization of controller gains is used to balance the rewards by visualizing the reward function. It is also used to choose the acquisition function and tune its hyperparameter by visualizing the observations of the Bayesian optimization. 2D optimization allowed the optimization two controller gains at the same time. It also showed the impact of each gains on the target function by visualizing the observed rewards. In both cases, the Bayesian optimization was able to predict the optimal controller gains accurately. In this section, a 4D optimization is conducted where four controller parameters are chosen as the inputs of the target function. Using the expert knowledge of the task, the translational stiffness gains k_{ty}, k_{tz} and damping gains k_{vy}, k_{vz} are chosen as controlled variables with k_{tx}, k_{vx} set to a constant. The components along x-axis are fixed because the task at hand involves drawing circular trajectories in YZ plane while maintaining a constant contact with the target surface at all times. The goal of this 4D optimization is to find the optimal controller gains K_1, K_2, K_3, K_4 whose configuration is given by:

$$\begin{aligned} K_1 = k_{ty}\ ; \quad K_2 = k_{tz}\ ; \\ K_3 = k_{vy}\ ; \quad K_4 = k_{vz}\ ; \\ k_{tx} = \text{fixed}\ ; \quad k_{vx} = \text{fixed} \end{aligned} \tag{4.5}$$

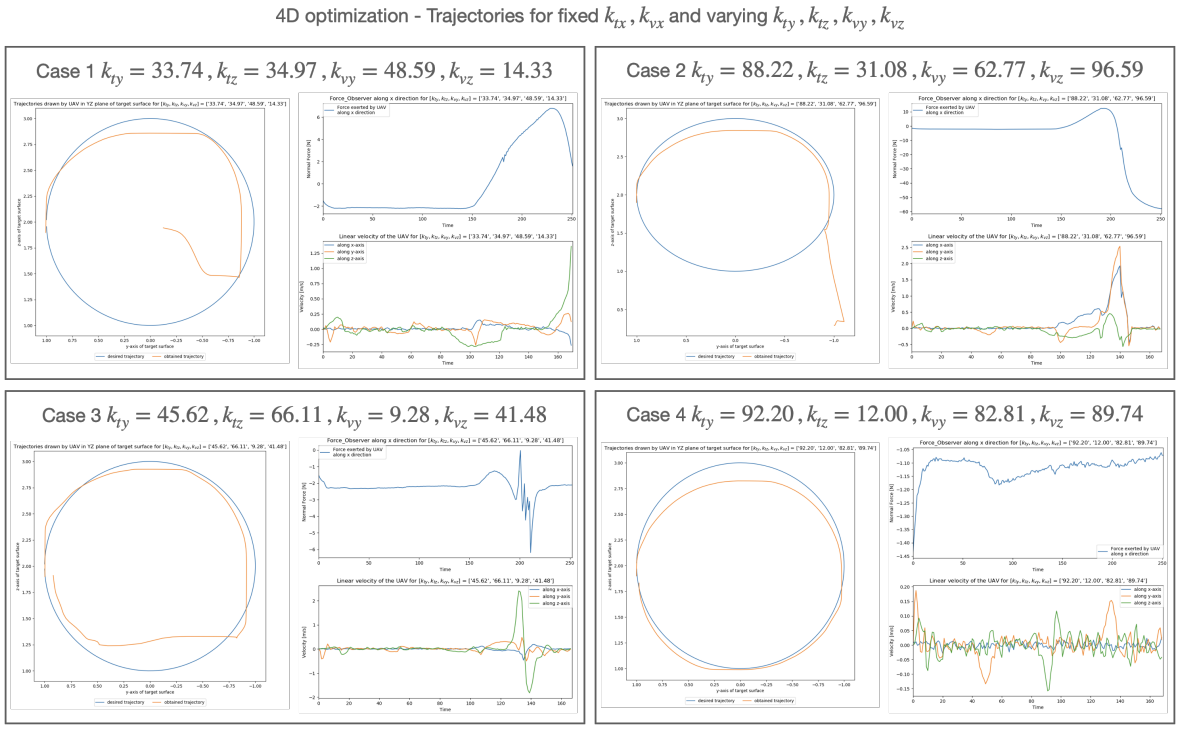
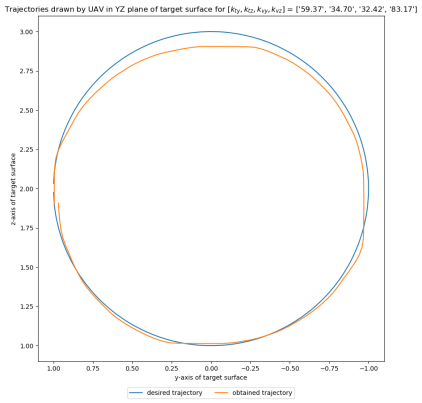


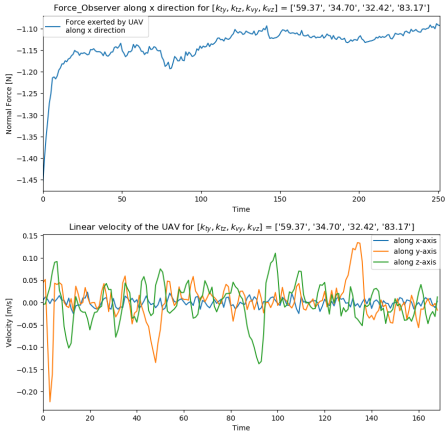
Figure 4.22: Various cases of random 4D controller gains demonstrating the drawn trajectories, applied normal forces and linear velocities of the UAV

The purpose of the groundtruth shown in 1D optimization and 2D optimization cases is only to verify if the maximum reward predicted by the Bayesian optimization aligns with that of the groundtruth. Since it is already proved through 1D optimization and 2D optimization that the Bayesian optimization is successful in finding the maximum reward, the groundtruth is no longer needed for higher dimensions of controller gains. It is also very difficult to visualize the groundtruth and observations of the Bayesian optimization for higher dimensions. The UAV is made to draw circular trajectories on the target surface by choosing random values for the four gains k_{ty}, k_{tz}, k_{vy}, k_{vz}. The drawn trajectories along with the normal force applied by the UAV and its linear velocities are shown in Figure 4.22. It can be seen that only the case-4 with $k_{ty} = 92.20$, $k_{tz} = 12.00$, $k_{vy} = 82.81$, $k_{vz} = 89.74$ was able to achieve a complete circular trajectory. Even though the achieved trajectory is not accurate, it is much better than the other cases. The pattern that is observed during these test cases is that the UAV is able to draw a complete circle if the z-component of translational stiffness gains is less than the y-component and the y-component of damping gains is less than the z-component i.e $k_{tz} < k_{ty}$ and $k_{vy} < k_{vz}$. Any other choice of gains results in completely deformed trajectories.

This is main reason for using the Bayesian optimization as it is very difficult to choose the impedance controller gains for high dimensions. The optimal 4D controller gains found by the Bayesian optimization are $k_{ty} = 59.37$, $k_{tz} = 34.70$, $k_{vy} = 32.42$, $k_{vz} = 83.17$. In order to verify the trajectories drawn by this choice of controller gains, the trajectory drawn by the UAV with these gains is shown in Figure 4.23a. While the normal force applied by the UAV along with its linear velocity are shown in Figure 4.23b. It can be seen that the trajectory drawn by the UAV is very accurate and much better than any trajectories from random controller gains shown in Figure 4.22. These simulations provide enough evidence that the Bayesian optimization is able to predict the optimal controller gains even for tasks of much higher dimensions.



(a) Trajectory drawn in YZ plane for $k_{ty} = 59.37$, $k_{tz} = 34.70$, $k_{vy} = 32.42$, $k_{vz} = 83.17$



(b) Normal force, Linear velocity for $k_{ty} = 59.37$, $k_{tz} = 34.70$, $k_{vy} = 32.42$, $k_{vz} = 83.17$

Figure 4.23: Performance of UAV for $k_{ty} = 59.37$, $k_{tz} = 34.70$, $k_{vy} = 32.42$, $k_{vz} = 83.17$

4.7 Conclusion

This chapter started off by describing various design choices that were made as part of simulation setup. These include building a virtual UAV in Gazebo simulator using the model of the physical UAV *betaX*. Using the expert knowledge of the task of drawing circular trajectories by an UAV, the 24 impedance controller gains are reduced to 12. Bayesian optimization was later used to find the optimal values of these controller gains. This was done by optimizing only few gains that have a greater influence on the task at hand using the expert knowledge. 1D optimization involves finding the optimal values of controller gain k_{ty} - translational stiffness gain along y-axis. The z-component of it k_{tz} was linked to it so that they both always have the same value giving equal importance to the changes in UAV motion along y-axis and z-axis. This 1D optimization was also used for generating a groundtruth target function in-order to design a balanced reward function. 1D optimization also allowed the visualization of the various observations involved in the Bayesian optimization. This visualization was used to choose the acquisition function of the Bayesian optimization and its hyperparameters.

As part of domain randomization, random Gaussian noise was added to the thrust generated by the rotors of the UAV. This was to replicate the external aerodynamic disturbances that occur at propeller level in the simulation. The Gaussian noise was modelled as an exponential function of rotor angular velocities so that more noise is added when the propellers spin at high RPM. Several simulations prove that the Bayesian optimization was able to find the optimal 1D controller gains even when random noise was introduced in the simulation. Apart from added noise, the friction coefficients μ_{TS} of the virtual target surface was also considered as part of domain randomization. A number of groundtruth target functions were generated with 1D controller gains by varying the friction coefficients of the target surface. It was observed that the optimal controller gains calculated for one target surface are not suitable for other surfaces. These optimal gains resulted in poor trajectories drawn by the UAV for other surfaces.

After finalizing the reward function, acquisition function, hyperparameters and other simulation parameters using the 1D optimization, the Bayesian optimization was used to find the optimal controller gains of higher dimensions. In 2D optimization, the controller gains k_{ty} and k_{tz} were no longer linked and a 2D groundtruth target function was generated to visualize the impact each controller gain has on the generated rewards. Later, Bayesian optimization was used to find the optimal 2D controller gains. The predicted optimal gains resulted in satisfactory trajectory drawn by the UAV maintaining balance between the accuracy and smoothness of the trajectory. Similarly, four controller gains k_{ty}, k_{tz}, k_{vy}, k_{vz} (where k_{vy}, k_{vz} are the damping gains) were selected using expert knowledge for a 4D optimization. Bayesian optimization was again successful in predicting the optimal 4D controller gains and reflected in accurate trajectories drawn by the UAV. All the simulations mentioned in this chapter prove that the chosen parameter of the Bayesian optimization can predict the optimal controller gains with out any issue. Therefore, the next step is to repeat the simulations several times utilizing the techniques of Sim2Real transfer such as domain randomization and generate the data required to build the custom dataset. An idea for the generation of data for custom dataset is proposed in Appendix A.

Chapter 5

Conclusion and Future works

Recapping the research questions from Section 1.4, **RQ-1:** '*How to safely find the optimal impedance controller parameters of multi-rotor UAV for the task of drawing circular trajectories on unknown surfaces in as little iterations as possible?*'. This is answered in the form of a two-part proposed method in Section 1.3. It is assumed that the model of the UAV is known and the model of the environment(target surface) is unknown. For a given physical UAV implemented with an impedance controller, the proposed method requires the UAV to draw a trajectory of circular shape on the target surface. During this trajectory, user-specified safe controller parameters are chosen. The data generated during this trajectory drawing is fed to a pre-trained neural network. The neural network considers this as a classification problem and predicts the optimal controller parameters suitable for the given target surface.

This raises a number of questions such as why does the UAV need to draw a trajectory with user-specified parameters? What data is used to train the neural networks? From where is the data collected from? Answering to these questions with the latest first - since the robot data is costly and hard to gather, the data for training the neural networks is generated in a virtual world through simulations. The techniques of Sim2Real transfer such as system identification and domain randomization are used . Gazebo simulator is considered for the simulation environment and the model of the physical UAV is used to build its virtual replica in the simulator. A number of simulation scenarios are generated(Figure 4.18) through domain randomization by varying the parameters of the virtual target surface and the induced noise at the propeller level of the UAV. It should be noted that only the part of generating the data for custom dataset is in the scope for this thesis. The training of neural networks is considered for future works.

In every simulation, the virtual UAV draws a circular trajectory on the virtual target surface similar to that of the real-world. The virtual UAV draws the circular trajectories in several randomized yet unique simulation scenarios created using the techniques of domain randomization. The generated trajectory data for every simulation are gathered together to form the custom dataset on which the neural network is trained in future. The trajectory data is also collected when the physical UAV draws the trajectory on the target surface in real world. The pre-trained neural network then compares this trajectory data(generated from real trajectory) to the several trajectory data(generated from virtual trajectories) from the custom dataset and predicts the appropriate optimal controller parameters for the given target surface in real-world.

The physical UAV is required to draw the circular trajectories on the unknown target surface so that the neural network could use the generated trajectory data to predict the optimal controller parameters. For a well trained neural network, the data from a single trajectory should be sufficient to make a valid prediction and is discussed in Appendix A. However, the prediction may need more than one trajectory error matrices if there are any faults in the training of neural networks. Poor representation of the custom dataset such as not enough data samples, or not many unique samples can also be the reasons for poor performance of neural networks. This reflects in the physical UAV requiring to draw multiple trajectories on the target surface before the neural network could predict the optimal controller parameters. All this may again raise new questions such as what are the controller parameters that are being predicted by the neural network? The neural network is being trained on a custom dataset with the trajectory error matrices being the inputs but what about the labels of this dataset?

There are 24 parameters of impedance controller whose values are to be chosen appropriately. However, exploiting the expert knowledge of the task at hand, it can be said that not all controller parameters have a huge influence on the task at hand. Only 12 parameters(Section 4.2) effect the performance of trajectories drawn on the vertical target surface. Since finding the optimal values for all 12 parameters is costly and not feasible. Few parameters namely translational stiffness gains k_t and damping gains k_v are chosen for optimizing as they have a greater impact on the performance of the trajectories drawn by the UAV. Four controller gains $[k_{ty}, k_{tz}, k_{vy}, k_{vz}]$ are optimized as part of a 4D optimization and the trajectories drawn with the optimal gains are shown in Figure 4.23a. With the help of controlled parameters, fixed parameters and linked parameters these gains are further reduced to a single parameter k_{ty} and two parameters $[k_{ty}, k_{tz}]$ for 1D optimization and 2D optimization respectfully. The 1D optimization is used to visualize the process of finding the optimal controller gains. The outcomes of these 1D, 2D and 4D optimizations are the required optimal controller gains for the given target surface. These optimal controller gains act as labels to the custom dataset while the trajectory error matrices take the role of inputs thereby answering the next question. This also answers one of the research questions, **RQ-3:** *'How to speed up the search for optimal controller parameters by exploiting the expert knowledge of the task at hand?'*. By reducing the 24 controller gains to just one, the search space for finding its optimal value for this gain is drastically reduced and therefore results in less simulation times. This may raise one final question on what exactly is happening in the optimizations of 1D, 2D and 4D controller parameters and how are the optimal controller parameters calculated?

Bayesian optimization - an iterative process of finding the maximum of an expensive black box target function is used for finding the optimal controller gains. The first step of implementing the Bayesian optimization was to formulate the task of trajectory drawing on unknown surfaces into a target function. A reward function was proposed and designed in Section 3.2 to generate rewards - a performance metric that depicts the accuracy and smoothness of the trajectories drawn by the UAV. The research question **RQ-2:** *How to design the reward function such that the Bayesian optimization can find the optimal controller parameters for the task of drawing circular trajectories on unknown surfaces using an UAV?* is answered here and is explained more in Section 3.2. The goal of the reward function is to describe the performance/accuracy of the trajectories drawn by the UAV. The kinetic energy and potential energy of the impedance controller are added together to generate one-dimensional rewards that depict the accuracy and smoothness of the trajectories drawn by the UAV. However, it should be noted that the reward generated from the reward function should be one-dimensional in-order to be able to work with the Bayesian optimization. The target function is therefore formulated with controller parameters as inputs and rewards as outputs.

The goal of Bayesian optimization is to find the maximum reward of this target function. With the help of few random initial observations(samples of target function), a surrogate model is built using gaussian processes(GP) and RBF kernel that act as a prior distribution. Through this surrogate model an acquisition function is optimized and its maximum is used to find the next best point to sample(Algorithm 1). The target function is sampled at this point and the observed reward is used to update the surrogate model. The maximum reward is found by repeating this process for several iterations. Since maximum reward translates into accurate and smooth trajectories drawn by the UAV, the controller parameters at which this maximum reward is observed are considered to be the optimal controller parameters. It should be noted that there is no convergence for the Bayesian optimization and its accuracy is a design choice and is controlled by the number of chosen iterations. This entire process of Bayesian optimization is applied to various cases of the target function with a 1D controller gain as input, 2D gains as input and 4D gains as input. The 1D optimization is used to visualize the groundtruth target function and design the balanced reward function. The observations of the Bayesian optimization are visualized to choose the appropriate acquisition function and its hyperparameters. Lastly, the 1D optimization and 2D optimization are used to verify if the optimal gains predicted by the Bayesian optimization are accurate. The Bayesian optimization is later used to find the optimal gains of 4D optimization which also gave accurate results shown in Figure 4.23.

As part of domain randomization, random Gaussian noise is added to the rotor thrust force in an effort to replicate external disturbances in the simulation. This noise was modelled according to an exponential curve as a function of angular velocities of the UAV rotors. Due to this, a higher noise is added to the thrust generated by the propeller when the rotor spin at high RPM. This results in low rewards when the propellers spin at maximum speed for higher controller gains making them undesirable. Apart from varying the parameters of the noise curve, the friction coefficients of the virtual target surface can also be varied to create several unique simulation scenarios(Figure 4.18) as part of domain randomization. The final research question is answered here **RQ-4** *How to create multiple simulation scenarios to generate data for the custom dataset?*

Future Works

The training of the neural networks was not intended for this thesis since this in itself could be an individual thesis topic because of huge number of factors to consider before initiating the training. These include the type of neural networks to be used, learning parameters, number of hidden layers, choice of activation layers etc. Through domain randomization, the parameters of noise and friction coefficients of target surface are varied in-order to generate several unique instances of simulation scenarios. However, more research can be done to see what other parameters can be added and varied like wind models, aerodynamic disturbances, etc in-order to make more realistic simulation scenarios. One can also change the shape of the trajectory and research if the neural network is able to predict the optimal gains for multiple trajectories. An even better option is to choose a unique shape that covers the strokes of commonly used application movements like square, rectangle, triangle, and spiral versions of them and train the neural network on drawing the trajectories of this unique shape. So that the UAV with pre-trained neural network can be immediately deployed into applications like window cleaning which can be done through the shapes like discontinuous horizontal/vertical lines or continuous spiral rectangle shape. The trajectory error matrix mentioned in Appendix A contains position error, orientation error and twist errors of the trajectories drawn by the UAV. This can be expanded to include more details that effect the performance of the trajectory and UAV. It is evident from the recent works[20, 21] that the use of little real world data helps a lot in the performance of the Sim2Real transfer. Therefore, domain adaptation methods can also be applied in future with the addition of few data samples from real world.

Appendix A

Custom Dataset

Since the Bayesian optimization is able to find the satisfactory maximum reward and therefore a satisfactory optimal controller gains, it is used to prepare the custom dataset for training the neural network eliminating the need for the groundtruth. On a higher level, the neural networks has to solve a classification problem where it has to predict the optimal controller gains given the data of a trajectory drawn by the physical UAV. The neural network are trained from a custom dataset which is a collection of several trajectory data drawn by the virtual UAV(simulation) along with the corresponding optimal controller gains found using Bayesian optimization. Therefore, one of the problems involved here is regarding the data itself that makes the custom dataset. The optimal controller gains act as the labels of the dataset but the data that act as input is a design choice that depends on the task at hand. For the task of trajectory drawing, the trajectory errors are considered as the input to the dataset.

If $P_x(t), P_y(t), P_z(t)$ are the positions of the UAV at time t along x, y, z axis respectfully, then $P_{act}(t) = [P_x(t), P_y(t), P_z(t)]$ is a 3D vector representing the actual (current)position of the UAV at time instance t. Similarly, $R_{act}(t)$ is a 3×3 matrix representing the actual (current)orientation of the UAV at time instance t. $v_{act}(t)$ and $\omega_{act}(t)$ are the actual linear and angular velocities of the UAV at time t. If P_{des}, R_{des} are the desired position and orientation matrices and v_{des}, ω_{des} are the desired linear and angular velocities respectfully then the corresponding errors are calculated as shown in Equation A.1.

$$\begin{aligned} \tilde{P}(t) &= P_{des}(t) - P_{act}(t) \\ \tilde{R}(t) &= tr(R_{des}^{\top}(t) R_{act}(t) - I) \\ \tilde{v}(t) &= v_{des}(t) - v_{act}(t) \\ \tilde{\omega}(t) &= \omega_{des}(t) - \omega_{act}(t) \end{aligned} \tag{A.1}$$

where $\tilde{P}(t)$ is the position error, $\tilde{R}(t)$ is the orientation error, $\tilde{v}(t)$ is the error in linear velocity and $\tilde{\omega}(t)$ is the error in angular velocity. $tr(\bullet)$ is the trace of the matrix. A trajectory error matrix Γ_{err} of size $4 \times T$ collects the position, orientation and velocity errors of a trajectory over time T and is represented by Equation A.2. Where T is the time taken to draw a trajectory.

$$\Gamma_{err} = \begin{array}{c} \\ \tilde{P}(t) \\ \tilde{R}(t) \\ \tilde{v}(t) \\ \tilde{\omega}(t) \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & . & . & . & T \end{array} \\ \begin{bmatrix} ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... \end{bmatrix} \\ 4 \times T \end{array} \tag{A.2}$$

The classification data has the trajectory error matrix Γ_{err} as input and optimal controller gains as label. However to generate a dataset for training the neural network, a large collection of data(inputs-labels) is required. This data collection is done through the process of domain randomization as mentioned in Section 4.4 through several combinations of friction coefficients μ_{TS} and parameters of noise function. Given a trajectory drawing task, the physical UAV draws a trajectory on the target surface with random(yet safe) controller gains in real world and the trajectory error matrix Γ_{err} is computed. The pre-trained neural network predicts the optimal controller gains suitable for the given unknown target surface by comparing the calculated trajectory error matrix with that of the custom dataset as illustrated in Figure A.1. Depending on the quality of the dataset i.e amount of data samples and uniqueness of the data samples a trained neural network would be able to predict the optimal gains with a single iteration of the trajectory in real-world. This is called one-shot Sim2Real transfer where the neural network only requires a single real world data sample for the prediction. This number may go up if the neural network cannot find any similar data samples from the dataset due to poor training and poor generation of custom dataset.

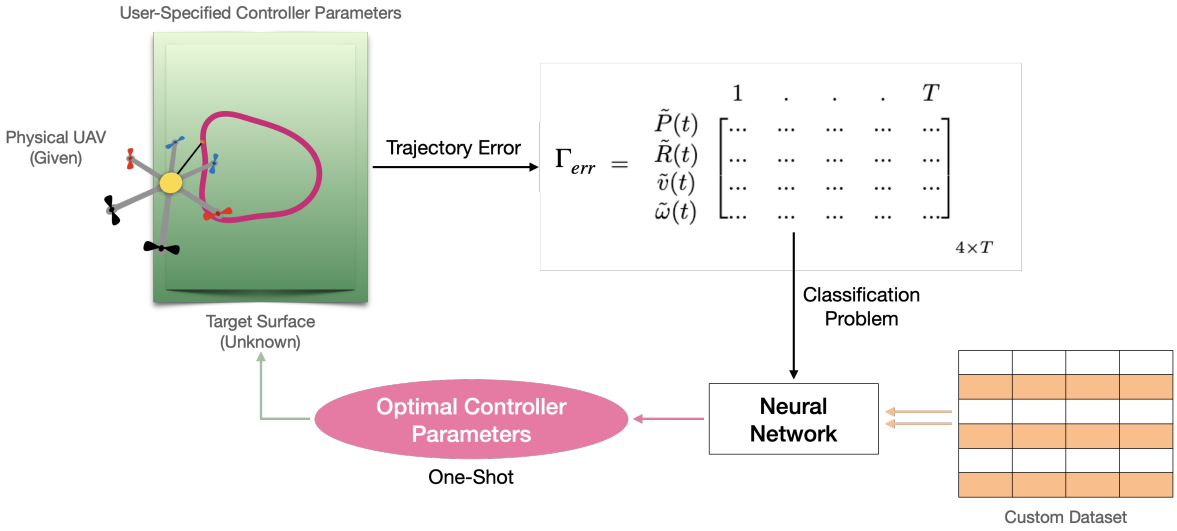


Figure A.1: Block diagram of One-shot Sim2Real transfer of the proposed method

Bibliography

[1] Aleksandra Faust et al. "Automated aerial suspended cargo delivery through reinforcement learning". In: *Artificial Intelligence* 247 (2017). Special Issue on AI and Robotics, pp. 381–398. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2014.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0004370214001416.

[2] Nathalie Guimarães et al. "Forestry Remote Sensing from Unmanned Aerial Vehicles: A Review Focusing on the Data, Processing and Potentialities". In: *Remote Sensing* 12.6 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12061046. URL: https://www.mdpi.com/2072-4292/12/6/1046.

[3] Anne Goodchild and Jordan Toy. "Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry". In: *Transportation Research Part D: Transport and Environment* 61 (2018). Innovative Approaches to Improve the Environmental Performance of Supply Chains and Freight Transportation Systems, pp. 58–67. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2017.02.017. URL: https://www.sciencedirect.com/science/article/pii/S136192091630133X.

[4] Sonia Waharte and Agathoniki Trigoni. "Supporting Search and Rescue Operations with UAVs". In: *2010 International Conference on Emerging Security Technologies* (2010), pp. 142–147.

[5] UM Rao Mogili and B B V L Deepak. "Review on Application of Drone Systems in Precision Agriculture". In: *Procedia Computer Science* 133 (2018). International Conference on Robotics and Smart Manufacturing (RoSMa2018), pp. 502–509. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.07.063. URL: https://www.sciencedirect.com/science/article/pii/S1877050918310081.

[6] Wennie Tabib et al. "Autonomous Cave Surveying With an Aerial Robot". In: *IEEE Transactions on Robotics* 38.2 (2022), pp. 1016–1032. DOI: 10.1109/TRO.2021.3104459.

[7] Chongsheng Cheng, Zhexiong Shang, and Zhigang Shen. "Automatic delamination segmentation for bridge deck based on encoder-decoder deep learning through UAV-based thermography". In: *NDT & E International* 116 (2020), p. 102341. ISSN: 0963-8695. DOI: https://doi.org/10.1016/j.ndteint.2020.102341. URL: https://www.sciencedirect.com/science/article/pii/S0963869520303224.

[8] Parham Nooralishahi et al. "Drone-Based Non-Destructive Inspection of Industrial Sites: A Review and Case Studies". In: *Drones* 5.4 (2021), p. 106.

[9] Stephen A Conyers. "Empirical evaluation of ground, ceiling, and wall effect for small-scale rotorcraft". PhD thesis. University of Denver, 2019.

[10] Markus Ryll et al. "6D Interaction Control with Aerial Robots: The Flying End-Effector Paradigm". In: *The International Journal of Robotics Research* 38 (2019), pp. 1045–1062. DOI: 10.1177/0278364919856694.

[11] Burak Yüksel et al. "Aerial Physical Interaction via IDA-PBC". In: *The International Journal of Robotics Research* 38 (2019), 403–421. DOI: 10.1177/0278364919835605.

[12] Melissa Mozifian et al. “Intervention design for effective sim2real transfer”. In: *arXiv preprint arXiv:2012.02055* (2020).

[13] Lilian Weng. “Domain Randomization for Sim2Real Transfer”. In: *lilianweng.github.io* (2019). URL: https://lilianweng.github.io/posts/2019-05-05-domain-randomization/.

[14] Antonio E. Jimenez-Cano et al. “Contact-Based Bridge Inspection Multirotors: Design, Modeling, and Control Considering the Ceiling Effect”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3561–3568. DOI: 10.1109/LRA.2019.2928206.

[15] Sang-won Leigh, Harshit Agrawal, and Pattie Maes. “A Flying Pantograph: Interleaving Expressivity of Human and Machine”. In: *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*. TEI '16. Eindhoven, Netherlands: Association for Computing Machinery, 2016, pp. 653–657. ISBN: 9781450335829. DOI: 10.1145/2839462.2856347.

[16] Sang-won Leigh, Harshit Agrawal, and Pattie Maes. “Z-Drawing: A Flying Agent System for Computer-Assisted Drawing”. In: *ACM SIGGRAPH 2015 Posters*. SIGGRAPH '15. Los Angeles, California: Association for Computing Machinery, 2015. ISBN: 9781450336321. DOI: 10.1145/2787626.2787652. URL: https://doi-org.ezproxy2.utwente.nl/10.1145/2787626.2787652.

[17] Xiangdong Meng, Yuqing He, and Jianda Han. “Design and Implementation of a Contact Aerial Manipulator System for Glass-Wall Inspection Tasks”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 215–220. DOI: 10.1109/IROS40897.2019.8968123.

[18] Asem Khattab et al. “Bayesian-Optimized Impedance Control of an Aerial Robot for Safe Physical Interaction with the Environment”. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2019, pp. 172–179. DOI: 10.1109/SSRR.2019.8848967.

[19] Chenxi Xiao, Peng Lu, and Qizhi He. “Flying Through a Narrow Gap Using End-to-End Deep Reinforcement Learning Augmented With Curriculum Learning and Sim2Real”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[20] Stephen James et al. “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12627–12637.

[21] Yevgen Chebotar et al. “Closing the sim-to-real loop: Adapting simulation randomization with real world experience”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

[22] Eric Brochu, Vlad M. Cora, and Nando de Freitas. “A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning”. In: *ArXiv* abs/1012.2599 (2010). URL: http://arxiv.org/pdf/1012.2599v1.pdf.

[23] Rodolphe Jenatton et al. “Bayesian Optimization with Tree-Structured Dependencies”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1655–1664.

[24] Javier González et al. “Preferential Bayesian Optimization”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1282–1291.

[25] Jasper Snoek et al. “Input Warping for Bayesian Optimization of Non-Stationary Functions”. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, June 2014, pp. 1674–1682. URL: https://proceedings.mlr.press/v32/snoek14.html.

[26] Jasper Snoek et al. “Scalable Bayesian Optimization Using Deep Neural Networks”. In: ICML’15. Lille, France: JMLR.org, 2015, pp. 2171–2180.

[27] Erik Bodin et al. “Modulating Surrogates for Bayesian Optimization”. In: *Proceedings of the 37th International Conference on Machine Learning*. ICML’20. JMLR.org, 2020.

[28] Qin Lu et al. “Surrogate modeling for Bayesian optimization beyond a single Gaussian process”. In: *arXiv e-prints*, arXiv:2205.14090 (May 2022), arXiv:2205.14090. arXiv: 2205.14090 [stat.ML].

[29] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X. URL: http://gaussianprocess.org/gpml/.

[30] Vu Nguyen et al. “Regret for Expected Improvement over the Best-Observed Value and Stopping Condition”. In: *Proceedings of the Ninth Asian Conference on Machine Learning*. Ed. by Min-Ling Zhang and Yung-Kyun Noh. Vol. 77. Proceedings of Machine Learning Research. Yonsei University, Seoul, Republic of Korea: PMLR, Nov. 2017, pp. 279–294. URL: https://proceedings.mlr.press/v77/nguyen17a.html.

[31] A. A. M. Khattab. *Towards an interactive drone : a Bayesian optimization approach*. Aug. 2018. URL: http://essay.utwente.nl/76246/.

[32] Ran Jiao et al. “Observer-based Geometric Impedance Control of a Fully-Actuated Hexarotor for Physical Sliding Interaction with Unknown Generic Surfaces”. In: *Journal of Intelligent & Robotic Systems* 102.4 (2021), pp. 1–18.

[33] Ramy Rashad et al. “Energy aware impedance control of a flying end-effector in the port-hamiltonian framework”. In: *IEEE transactions on robotics* (2022).

[34] Chuang Gan et al. “Threedworld: A platform for interactive multi-modal physical simulation”. In: *arXiv preprint arXiv:2007.04954* (2020).

[35] Fadri Furrer et al. “RotorS—A Modular Gazebo MAV Simulator Framework”. In: 2016.

[36] Shital Shah et al. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”. In: *Field and Service Robotics*. 2017. eprint: arXiv:1705.05065. URL: https://arxiv.org/abs/1705.05065.