UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

Gaussian Process Optimization for Self-Tuning Control

Author:
Alonso MARCO

Supervisors:
Dr. Sebastian TRIMPE
Dr. Cecilio ANGULO

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Universitat Politècnica de Catalunya
Automatic Control Department

developed at

Max Planck Institute for Intelligent Systems
Autonomous Motion Department

October 9, 2015

ETSEIB

MAX-PLANCK-GESELLSCHAFT

"Ohne Musik wäre das Leben ein Irrtum."

Friedrich Wilhelm Nietzsche

SUMMARY

Robotic setups often need fine-tuned controller parameters both at low- and task-levels. Finding an appropriate set of parameters through simplistic protocols, such as manual tuning or grid search, can be highly time-consuming. This thesis proposes an automatic controller tuning framework based on linear optimal control combined with Bayesian optimization. With this framework, an initial set of controller gains is automatically improved according to the performance observed in experiments on the physical plant.

In the tuning scenario that we propose, we assume we can measure the performance of the control system in experiments through an appropriate cost. However, we only allow a limited number of experimental evaluations (e.g. due to their intrinsic monetary cost or effort). The goal is to globally explore a given range of controller parameters in an efficient way, and return the best known controller at the end of this exploration.

At each iteration, a new controller is generated and tested on a closed-loop experiment in the real plant. Then, the recorded data is used to evaluate the system performance using a quadratic cost. We re-iterate in order to solve a global optimization problem, whose goal is to learn most about the location of the global minimum from the limited number of experiments.

We use the Linear Quadratic Regulator (LQR) formulation as a standard way to compute optimal multivariate controllers given a linear plant model and quadratic cost. We parametrize the LQR weights in order to obtain controllers with appropriate robustness guarantees.

The underlying Bayesian optimization algorithm is Entropy Search (ES), which represents the latent objective as a Gaussian process and constructs an explicit belief over the location of the objective minimum. This method maximizes the information gain from each experimental evaluation. Thus, this framework shall yield improved controllers with fewer evaluations compared to alternative approaches.

A seven-degree-of-freedom robot arm balancing an inverted pole is used as the experimental demonstrator. Results of a two-dimensional tuning problem are shown in two different contexts: in the first setting, a wrong linear model is used to compute a nominal controller, which destabilizes the actual plant. The automatic tuning framework is still able to find a stabilizing controller after a few iterations. In the second setting, a fairly good linear model is used to compute a nominal controller. Even then, the framework can still improve the initial performance by about 30%. In addition, successful results on a four-dimensional tuning problem indicate that the method can scale to higher dimensions.

The main and novel contribution of this thesis is the development of an automatic controller tuning framework combining ES with LQR tuning. Albeit ES has been tested on simulated numerical optimization problems before, this work is the first to employ the algorithm for controller tuning and apply it on a physical robot platform. Bayesian optimization has recently gained a lot of interest in the research community as a principled way for global optimization of noisy, black-box functions using probabilistic methods. Thus, this work is an important contribution toward making these methods available for automatic controller tuning for robots.

In conclusion, we demonstrate in experiments on a complex robotic platform that Bayesian optimization is useful for automatic controller tuning. Applying Gaussian process optimization for controller tuning is an emerging novel area. The promising results of this work open up many interesting research directions for the future.

In future work, we aim at scaling this framework further than this problem to higher dimensional systems such as balancing of a full humanoid robot for which a rough linear model can be obtained. In addition, comparing ES with other global optimizers from the literature may be of interest. Investigating safety considerations such as avoiding unstable controllers during the search is also part of future research.

Acknowledgements

To my parents, who unconditionally supported all the important decisions I made, even when those hurt.
To my sister, because she is, and she will always be the path to follow.
I thank my advisor Sebastian Trimpe for his great support from the first minute I met him, and for his deep degree of involvement during the development of this thesis. For his unconditional and unlimited patience and attention, and for his altruist help, even when I didn't ask for it. For patiently waiting until I made a decision about where to start my Ph.D., and for dedicating his time to listen to my hesitations, and answer my wide questions. I also want to thank him for being an inexhaustible source of knowledge, and for naming me a researcher.
I would like to thank Jeannete Bohg, for making me feel part of a whole from the first instant, for treating me as a friend when I just arrived, and for being there inside and outside the lab.
I thank Ludovic Righetti for his advice regarding the low-level tracking controllers, and for advising me when I felt lost.
I also thank Alexander Herzog for his great help with the motion tracking system.
I am grateful to Heiko Ott and Felix Grimminger for their support with the robot hardware and the design of the pole, and because with them, working is just fun.
I thank Philipp Hennig for pointing out that Friday Beer cannot be found in many places when I doubted about where to start my Ph.D.
I want to thank Akshara Rai for having pronounced the right words in difficult moments.
I thank all the members of the lab, for discussing spontaneous questions, and for many more dinners, card games and visits to Chez Michel.
To Javier, for always being ready to sing a new song, and for inviting me to delicious dishes from our land.
To all my new friends in Tübingen, but specially to Gerard, Laura and Cristina.
To Francisco Ramos De la Flor, because I owe him the proud of writing this words, at this moment, and in this research institute.
To Cecilio Angulo for his support with this thesis and for always replying my long e-mails.
To my whole family, who represent *the joy*.
To all my friends, who are happily uncountable.

To my grandma. . .

Contents

Chapter 1

INTRODUCTION

In this chapter we present the goal and motivation for this thesis. In addition, we present its scientific contributions and review the related literature.

Preliminary results of this approach [1] (paper) were recently shown in oral presentation at the *Second Machine Learning in Planning and Control of Robot Motion Workshop* at IEEE International Conference on Intelligent Robots and Systems (iROS). This thesis also shows further discussions and extended results, which were submitted to the 2016 IEEE International Conference on Robotics and Automation (ICRA) [2]. The contents of this thesis are an extension of these two previous works.

1.1 Motivation

Robotic setups often need fine-tuned controller parameters both at low- and task-levels. Finding an appropriate set of parameters through simplistic protocols, such as manual tuning or grid search, can be highly time-consuming. We seek to automate the process of fine tuning a nominal controller based on performance observed in experiments on the physical plant. We aim for information-efficient approaches, where only few experiments are needed to obtain improved performance.

Designing controllers for balancing systems such as in [3] or [4] are typical examples for such a scenario. Often, one can without much effort obtain a rough linear model of the system dynamics around an equilibrium configuration, for example, from first principles modeling. Given the linear model, it is then relatively straightforward to compute a stabilizing controller, for instance, using optimal control. When testing this nominal controller on the physical plant, however, one may find the balancing performance unsatisfactory, e.g. due to unmodeled dynamics, parametric uncertainties of the linear model, sensor noise, or imprecise actuation. Thus, fine-tuning the controller gains in experiments on the real system is desirable in order to partly mitigate these effects and obtain improved balancing performance.

1.2 Goal and scientific contributions

We have a tuning scenario in mind, where a limited budget of experimental evaluations is allowed (e.g. due to limited experimental time on the plant, or costly experiments). The automatic tuning shall globally explore a given range of controllers and return the best known controller after a fixed number of experiments. During exploration, we assume that it is acceptable for the controller to fail, for example, because other safety mechanisms are in place [5], or it is uncritical to stop an experiment when reaching safety limits (as is the case in experiment considered herein).

For this scenario, we propose a controller tuning framework extending previous work [6]. Therein, a Linear Quadratic Regulator (LQR) is iteratively improved based on control performance observed in experiments. The controller parameters of the LQR design are adjusted using Simultaneous Perturbation Stochastic Approximation (SPSA) [7] as optimizer of the experimental cost. It obtains a very rough estimate of the cost function gradient from few cost evaluations, and then updates the parameters in its negative direction. While control performance could be improved in experiments on a balancing platform in [6], this approach does not exploit the available data as much as could be done. Additionally, rather than exploring the space globally, it only approximates locally the location of the minimum.

In contrast to [6], we propose the use of Entropy Search (ES) [8], a recent algorithm for global Bayesian optimization, as the minimizer for the LQR tuning problem. ES employs a Gaussian process (GP) as a non-parametric model capturing the knowledge about the unknown cost function. At every iteration, the algorithm exploits all past data to infer the shape of the cost function. Furthermore, in the spirit of an active learning algorithm, it suggests the next evaluation such as to learn most about the location of the minimum. Thus, we expect ES to be more data-efficient than simple gradient-based approaches as in [6]; that is, to yield better controllers with fewer experiments.

The main and novel contribution of this thesis is the development of an automatic controller tuning framework combining ES [8] with LQR tuning [6]. Albeit ES has been tested on simulated numerical optimization problems before, it had never been tried on a real setting, and also never used for automatic controller tuning. Such recently developed algorithm is gaining interest, as Bayesian optimization is emerging as an exciting subfield of machine learning, concerned with the global optimization of noisy, black-box functions using probabilistic methods.

The effectiveness of the proposed auto-tuning method is demonstrated in experiments of a humanoid robot balancing a pole (see Figure 1.1). We present successful auto-tuning experiments for parameter spaces of different dimensions (2D and 4D), as well as for initialization with relatively good, but also poor initial controllers.

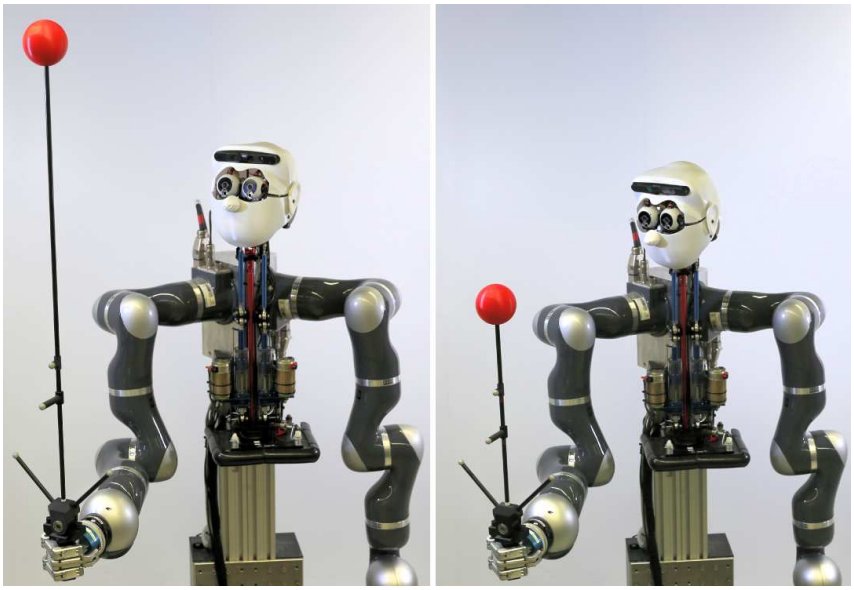


FIGURE 1.1: Robot Apollo balancing two inverted poles. These experimental platforms are used as a demonstrators of the automatic tuning framework.

1.3 Related work

Automatic tuning of an LQR is also considered in [9–11], for example. In these references, the tuning typically happens by first identifying model parameters from data, and then computing a controller from the updated model. In contrast, we tune the controller gain directly thus bypassing the model identification step. Albeit we exploit a nominal model in the LQR design, this model is not updated during tuning and merely serves to pre-structure the controller parameters.

Using Gaussian processes for controller tuning has recently been proposed in [12]. Therein, the parameter space is explored by selecting next evaluation points of maximum uncertainty (i.e. maximum variance of the GP). In contrast, ES uses a more sophisticated selection criterion maximizing the expected information gain of a new experiment aiming at eventually finding the global minimum, while the objective in [12] is on exploration only. A particular focus of the method in [12] is on safe exploration, and a GP classifying safe and unsafe regions is learnt alongside.

The task of learning a controller from experimental rewards (i.e. negative cost) is also considered in the rather large area of reinforcement learning (RL), see [13] for a survey. However, the tools used here (GP-based optimization) differ from the classical methods in RL. In [14], controller parameterization in terms of LQR weights is explored in the context of RL. While the authors find this choice to be inefficient for solving a reaching task, we find LQR parametrization suitable for improving feedback controllers (similar to the results in [6]).

We use a robot balancing a pole as the experimental demonstrator for our method. A similar

example is used in [12], albeit in simulation only. Pole balancing problems are commonly used as simulation or experimental examples also in other learning control approaches, see [15–19], for example.

The LQR tuning problem is described in Sec. 2.1. The use of ES for automating the tuning is outlined in Sec. 2.2. The experimental results are presented in Chapter 4, and concluding remarks in Chapter 7. Supplementary information such as technical details about the robotic platform can be found in Chapter 3. Additional information about the logistics of this work can be found in Chapter 5 and Chapter 6.

Chapter 2

AUTOMATIC LQR TUNING

This chapter is one of the core parts of the thesis. It is compounded by a complete description of the proposed framework. The other core part is Chapter 4, which shows experimental results. Sec. 2.1 addresses the LQR tuning problem, and Sec. 2.2 explains the use of ES as optimizer.

2.1 LQR tuning problem

In this section, we formulate the LQR tuning problem following the approach proposed in [6]. First, the considered control design problem is introduced in Sec. 2.1.1. Then, the parametrization of the controller is presented in Sec. 2.1.2, and ultimately the corresponding optimization problem is stated in Sec. 2.1.3.

2.1.1 Control design problem

We consider a system that follows a discrete-time non-linear dynamic model

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \tag{2.1}$$

with system states $\boldsymbol{x}_k$, control input $\boldsymbol{u}_k$, and zero-mean process noise $\boldsymbol{w}_k$ at time instant k. We assume that (2.1) has an equilibrium at $\boldsymbol{x}_k = \boldsymbol{0}$, $\boldsymbol{u}_k = \boldsymbol{0}$ and $\boldsymbol{w}_k = \boldsymbol{0}$, which we want to keep the system at. We also assume that $\boldsymbol{x}_k$ can be measured and, if not, an appropriate state estimator is used.

For regulation problems such as balancing about an equilibrium, a linear model is often sufficient for control design. Thus, we consider a scenario, where a linear model

$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_{\mathrm{n}}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}_{\mathrm{n}}\boldsymbol{u}_k + \boldsymbol{w}_k \tag{2.2}$$

is given as an approximation of the dynamics (2.1) about the equilibrium at zero. We refer to (2.2) as the *nominal model*, while (2.1) are the true system dynamics, which are unknown.

A common way to measure the performance of a control system is through a quadratic cost function such as

$$J = \lim_{K\to\infty} \frac{1}{K}\mathbb{E}\left[\sum_{k=0}^{K} \boldsymbol{x}_k^T \boldsymbol{Q}\boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R}\boldsymbol{u}_k\right] \tag{2.3}$$

with positive-definite weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, and $\mathbb{E}[\cdot]$ the expected value. The cost (2.3) captures a trade-off between control performance (keeping $\boldsymbol{x}_k$ small) and control effort (keeping $\boldsymbol{u}_k$ small), which we seek to achieve with the control design.

Ideally, we would like to obtain a state feedback controller for the non-linear plant (2.1) that minimized (2.3). Yet, this non-linear control design problem is intractable in general. Instead, a straightforward approach that yields a locally optimal solution is to compute the optimal controller minimizing (2.3) for the nominal model (2.2). This controller is given by the well-known Linear Quadratic Regulator (LQR) [20, Sec. 2.4]

$$\boldsymbol{u}_k = \boldsymbol{F}\boldsymbol{x}_k \tag{2.4}$$

whose static gain matrix $\boldsymbol{F}$ can readily be computed by solving the discrete-time infinite-horizon LQR problem for the nominal model $(\boldsymbol{A}_\mathrm{n}, \boldsymbol{B}_\mathrm{n})$ and the weights $(\boldsymbol{Q}, \boldsymbol{R})$. For simplicity, we write

$$\boldsymbol{F} = \mathrm{lqr}(\boldsymbol{A}_\mathrm{n}, \boldsymbol{B}_\mathrm{n}, \boldsymbol{Q}, \boldsymbol{R}). \tag{2.5}$$

If (2.2) perfectly captured the true system dynamics (2.1), then (2.5) would be the optimal controller for the problem at hand. However, in practice, there can be several reasons why the controller (2.5) is suboptimal: the true dynamics are non-linear, the nominal linear model (2.2) involves parametric uncertainty, or the state is not perfectly measurable (e.g. noisy or incomplete state measurements). While still adhering to the controller structure (2.4), it is thus beneficial to fine tune the nominal design (the gain $\boldsymbol{F}$) based on experimental data to partly compensate for these effects. This is the goal of the automatic tuning approach, which is detailed next.

2.1.2 LQR tuning problem

Following the approach in [6], we parametrize the controller gains $\boldsymbol{F}$ in (2.4) as

$$\boldsymbol{F}(\boldsymbol{\theta}) = \mathrm{lqr}(\boldsymbol{A}_\mathrm{n}, \boldsymbol{B}_\mathrm{n}, \bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta})) \tag{2.6}$$

where $\bar{\boldsymbol{Q}}(\boldsymbol{\theta})$ and $\bar{\boldsymbol{R}}(\boldsymbol{\theta})$ are *design weights* parametrized in $\boldsymbol{\theta} \in \mathbb{R}^D$, which are to be varied in the automatic tuning procedure. For instance, $\bar{\boldsymbol{Q}}(\boldsymbol{\theta})$ and $\bar{\boldsymbol{R}}(\boldsymbol{\theta})$ can be diagonal matrices with $\theta_j > 0$, $j = 1, \ldots, D$, as diagonal entries.

Parametrizing controllers in the LQR weights $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$ as in (2.6), instead of varying the controller gains $\boldsymbol{F}$ directly, serves to discard destabilizing or otherwise undesirable controllers.

It is a classical result of control theory [21][1] that any stabilizing feedback controller (2.4) that yields a *return difference* greater one (in magnitude) can be obtained for some $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$ as the solution to the LQR problem. The return difference is an important quantity in the analysis of feedback loops [20], and its magnitude exceeding one means favorable robustness properties. Therefore, parametrizing controllers in terms of the LQR design (2.6) does not discard any controllers except those that are undesirable because they destabilize the nominal plant, or have poor robustness properties. Of course, further parametrizing $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$ in $\boldsymbol{\theta}$ (e.g. diagonal) can possibly restrict the search space. However, such restrictions may be desirable to focus on most relevant parameters or to ease the optimization problem.

When varying $\boldsymbol{\theta}$, different controller gains $\boldsymbol{F}(\boldsymbol{\theta})$ are obtained. These will affect the system performance through (2.4), thus resulting in a different cost value from (2.3) in each experiment. To make the parameter dependence of (2.3) explicit, we write

$$J = J(\boldsymbol{\theta}). \tag{2.7}$$

The goal of the automatic LQR tuning is to vary the parameters $\boldsymbol{\theta}$ such as to minimize the cost (2.3).

Remark: The weights $(\boldsymbol{Q}, \boldsymbol{R})$ in (2.3) are referred to as *performance weights*. Note that, while the *design weights* $\big(\bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta})\big)$ in (2.6) change during the tuning procedure, the performance weights remain unchanged.

2.1.3 Optimization problem

The above LQR tuning problem is summarized as the optimization problem

$$\arg\min J(\boldsymbol{\theta}) \quad \text{s.t. } \boldsymbol{\theta} \in \mathcal{D} \tag{2.8}$$

where we restrict the search of parameters to a bounded domain $\mathcal{D} \subset \mathbb{R}^D$. The domain $\mathcal{D}$ typically represents a region around the nominal design, where performance improvements are to be expected or exploration is considered to be safe.

The shape of the cost function in (2.8) is unknown. Neither gradient information is available nor guarantees of convexity can be expected. Furthermore, (2.3) cannot be computed from experimental data in practice as it represents an infinite-horizon problem. As is also done in [6], we thus consider the approximate cost

$$\hat{J} = \frac{1}{K}\left[\sum_{k=0}^{K} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k\right] \tag{2.9}$$

[1]The result in [21] is stated for continuous-time systems with a single control variable (as is the case in the experiments considered in Chapter 4). Similar results exist for discrete-time systems [22], and for multi-input systems (see references in [23]).

with a finite, yet long enough horizon K. The cost (2.9) can be considered a noisy evaluation of (2.3). Such an evaluation is expensive as it involves conducting an experiment, which lasts few minutes in the considered balancing application.

2.2 LQR tuning with Entropy Search

In this section, we introduce Entropy Search (ES) [8] as the optimizer to address problem (2.8). The key characteristics of ES are explained in Sec. 2.2.1 to 2.2.3, the resulting framework for automatic LQR tuning is summarized in Sec. 2.2.4, and Sec. 2.2.5 briefly discusses related methods. Here, we present only the high-level ideas of ES from a practical standpoint. The reader interested in the mathematical details, as well as further explanations, is referred to [8].

2.2.1 Underlying cost function as a Gaussian process

ES is one out of several popular formulations of Bayesian Optimization [24–26], a framework for global optimization in which uncertainty over the objective function J is represented by a probability measure $p(J)$, typically a Gaussian process (GP) [27]. Note that the shape of the cost function (2.3) is unknown; only noisy evaluations (2.9) are available. A GP can be understood as a probability measure over a function space. Thus, the GP encodes the knowledge that we have about the cost function. New evaluations are incorporated through conditioning the GP on these data. With more data points, the cost function shape thus becomes better known. GP regression is a common way in machine learning for inferring an unknown function from noisy evaluations; refer to [27] for more details.

We model prior knowledge about J as the GP

$$J(\boldsymbol{\theta}) \sim \mathcal{GP}\left(\mu(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta}_*)\right) \tag{2.10}$$

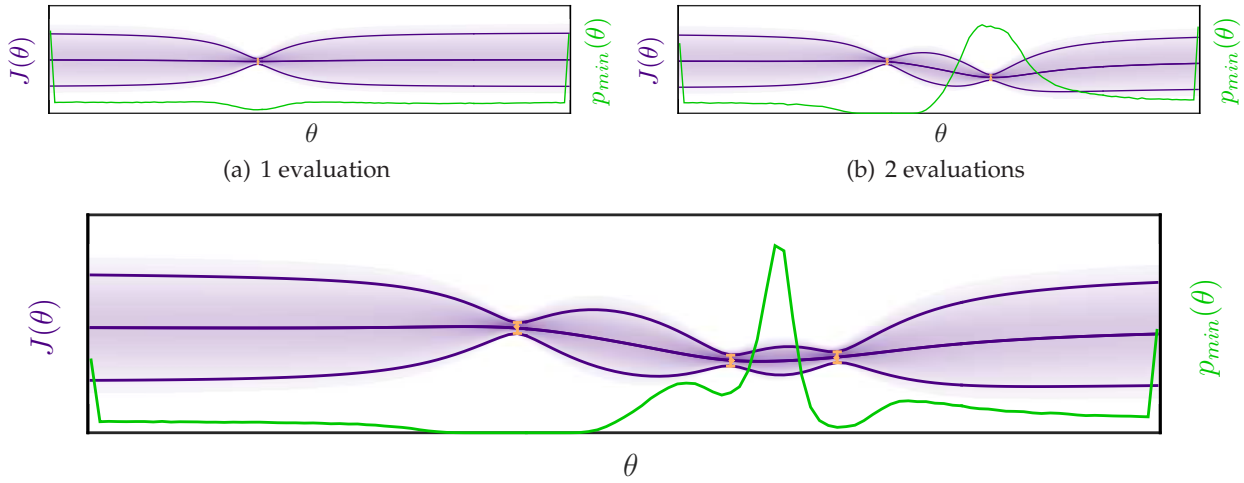
with mean function $\mu(\boldsymbol{\theta})$ and covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}_*)$. Common choices are a zero mean function ($\mu(\boldsymbol{\theta}) = \mathbf{0}$ for all $\boldsymbol{\theta}$), and the squared exponential (SE) covariance function

$$k_{\mathrm{SE}}(\boldsymbol{\theta}, \boldsymbol{\theta}_*) = \sigma^2 \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_*)^{\mathrm{T}} \boldsymbol{S} (\boldsymbol{\theta} - \boldsymbol{\theta}_*)\right] \tag{2.11}$$

which we also use herein. The covariance function $k(\boldsymbol{\theta}, \boldsymbol{\theta}_*)$ generally measures covariance between $J(\boldsymbol{\theta})$ and $J(\boldsymbol{\theta}_*)$. It can thus be used to encode assumptions about properties of J such as smoothness, characteristic length-scales, and signal variance. In particular, the SE covariance function (2.11) models smooth functions with signal variance σ^2 and length-scales $\boldsymbol{S} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$, $\lambda_j > 0$.

We assume that the noisy evaluations (2.9) of (2.3) can be modeled as

$$\hat{J} = J(\boldsymbol{\theta}) + \varepsilon \tag{2.12}$$



(a) 1 evaluation

(b) 2 evaluations

(c) 3 evaluations

FIGURE 2.1: Evolution of an example Gaussian process for three successive function evaluations (orange dots), reproduced with slight alterations from [8]. The posterior mean $\bar{\mu}(\boldsymbol{\theta})$ is shown in thick violet, two standard deviations $2\bar{\sigma}(\boldsymbol{\theta})$ in thin violet, and the probability density as a gradient of color that decreases away from the mean. Two standard deviations of the likelihood noise $2\sigma_{\mathrm{n}}$ are represented as orange vertical bars at each evaluation. Approximated probability distribution over the location of the minimum $p_{\mathrm{min}}(\boldsymbol{\theta})$ in green. These plots use arbitrary scales for each object. They were generated using [28].

where the independent identically distributed noise ε describes a Gaussian likelihood $\mathcal{N}(J(\boldsymbol{\theta}), \sigma_{\mathrm{n}}^2)$.

To simplify notation, we write $\boldsymbol{y} = \{\hat{J}^i\}_{i=1}^N$ for N evaluations at locations $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^i\}_{i=1}^N$. Conditioning the GP on the data $\{\boldsymbol{y}, \boldsymbol{\Theta}\}$ then yields another GP with posterior mean $\bar{\mu}(\boldsymbol{\theta})$ and a posterior variance $\bar{k}(\boldsymbol{\theta}, \boldsymbol{\theta}_*)$.

Figure 2.1 provides an example for a one-dimensional cost function and three successive function evaluations. As can be seen, the shape of the mean is adjusted to fit the data points, and the uncertainty (standard deviation) is reduced around the evaluations. In regions where no evaluations have been made, the uncertainty is still large. Thus, the GP provides a mean approximation of the underlying cost function, as well as a description of the uncertainty associated with this approximation.

We gather the hyperparameters of the GP in the set $\mathcal{H} = \{\lambda_1, \lambda_2, \ldots, \lambda_D, \sigma, \sigma_n\}$. An initial choice of $\mathcal{H}$ can be improved with every new data point $\hat{J}^i$ by adapting the hyperparameters. As commonly done, the marginal likelihood is maximized with respect to $\mathcal{H}$ at each iteration of ES.

Typically, the cost can have different sensitivity to different parameters θ_j; that is, the cost may change more significantly with some dimensions than with others. We use automatic relevance determination [27, Sec. 5.1] in the covariance function (2.11), which removes those dimensions with low influence on the cost as more data points become available.

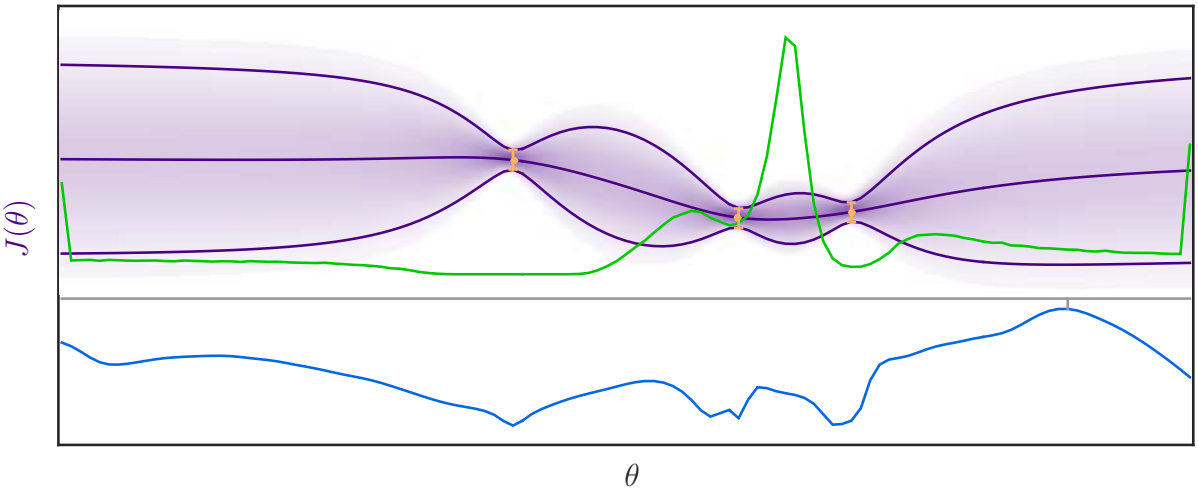


FIGURE 2.2: The blue line represents the expected gain in information $\mathbb{E}[\Delta H](\theta)$. The probability of the location of the minimum $p_{\min}(\boldsymbol{\theta})$ and the GP posterior over $J(\theta)$ from Fig. 2.1(c) are repeated for reference. The grey line is used as a separator. These plots use arbitrary scales for each object. They were generated using [28].

2.2.2 Probability measure over the location of the minimum

A key idea of ES (see [8, Sec. 1.1]) is to explicitly represent the probability $p_{\min}(\boldsymbol{\theta})$ for the minimum location over the domain $\mathcal{D}$:

$$p_{\min}(\boldsymbol{\theta}) \equiv p(\boldsymbol{\theta} = \arg\min J(\boldsymbol{\theta})), \quad \boldsymbol{\theta} \in \mathcal{D}. \tag{2.13}$$

The probability $p_{\min}(\boldsymbol{\theta})$ is induced by the GP for J: given a distribution of cost functions J as described by the GP, one can in principle compute the probability of any $\boldsymbol{\theta}$ being the minimum of J. For the example GPs in Fig. 2.1, $p_{\min}(\boldsymbol{\theta})$ is shown in green.

To obtain a tractable algorithm, ES approximates $p_{\min}(\boldsymbol{\theta})$ with finitely many points on a non-uniform grid that puts higher resolution in regions of greater influence.

2.2.3 Information-efficient evaluation decision

The key feature of ES is the suggestion of new locations $\boldsymbol{\theta}$, where (2.9) should be evaluated to learn most about the location of the global minimum. This is done by, first, computing the current amount of information H about the minimum, second, approximating the expected gain in information $\mathbb{E}[\Delta H](\boldsymbol{\theta})$ if we evaluated at a certain location $\boldsymbol{\theta}$, and third, suggesting as next evaluation point the location where $\mathbb{E}[\Delta H](\boldsymbol{\theta})$ is maximal.

Algorithm 1 Automatic LQR Tuning. As its inputs, ENTROPYSEARCH takes the type of covariance function k, the likelihood l, a fixed number of evaluations N, and data points $\{\boldsymbol{\Theta}, \boldsymbol{y}\}$. Alternative stopping criteria instead of stopping after N iterations can be used.

initialize $\boldsymbol{\theta}^0$; typically $\bar{\boldsymbol{Q}}(\boldsymbol{\theta}^0) = \boldsymbol{Q}$, $\bar{\boldsymbol{R}}(\boldsymbol{\theta}^0) = \boldsymbol{R}$
$\hat{J}^0 \leftarrow$ COSTEVALUATION($\boldsymbol{\theta}^0$) ▷ Cost evaluation
$\{\boldsymbol{\Theta}, \boldsymbol{y}\} \leftarrow \{\boldsymbol{\theta}^0, \hat{J}^0\}$
procedure ENTROPYSEARCH(k,l,N,$\{\boldsymbol{\Theta}, \boldsymbol{y}\}$)
  for $i = 1$ to N **do**
    $[\bar{\mu}, \bar{k}] \leftarrow \mathrm{GP}(k, l, \{\boldsymbol{\Theta}, \boldsymbol{y}\})$ ▷ GP posterior
    $p_{\min} \leftarrow \mathrm{approx}(\bar{\mu}, \bar{k})$ ▷ Approximate $p_{\min}$
    $\boldsymbol{\theta}^i \leftarrow \arg\max \Delta H$ ▷ Next location to evaluate at
    $\hat{J}^i \leftarrow$ COSTEVALUATION($\boldsymbol{\theta}^i$) ▷ Cost evaluation
    $\{\boldsymbol{\Theta}, \boldsymbol{y}\} \leftarrow \{\boldsymbol{\Theta}, \boldsymbol{y}\} \cup \{\boldsymbol{\theta}^i, \hat{J}^i\}$
    $\boldsymbol{\theta}^{\mathrm{BG}} \leftarrow \arg\max p_{\min}$ ▷ Update current "Best Guess"
  end for
  return $\boldsymbol{\theta}^{\mathrm{BG}}$
end procedure

function COSTEVALUATION($\boldsymbol{\theta}$)
  LQR design: $\bar{\boldsymbol{F}} \leftarrow \mathrm{lqr}(\boldsymbol{A}_{\mathrm{n}}, \boldsymbol{B}_{\mathrm{n}}, \bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta}))$
  update control law (2.4) with $\boldsymbol{F} = \bar{\boldsymbol{F}}$
  perform experiment and record $\{\boldsymbol{x}_k\}, \{\boldsymbol{u}_k\}$
  Evaluate cost: $\hat{J} \leftarrow \frac{1}{K}\left[\sum_{k=0}^{K} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k\right]$
  return $\hat{J}$
end function

The current amount of information about the location of the global minimum is computed with the relative entropy

$$H = \int_{\mathcal{D}} p_{\min}(\boldsymbol{\theta}) \log \frac{p_{\min}(\boldsymbol{\theta})}{b(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} \tag{2.14}$$

between $p_{\min}(\boldsymbol{\theta})$ and the uniform distribution $b(\boldsymbol{\theta})$ over the bounded domain $\mathcal{D}$. The rationale for this is that the uniform distribution essentially has no information about the location of the minimum, while a very "peaked" distribution would be desirable to obtain distinct potential minima. In fact, H diverges toward plus infinity as $p_{\min}$ approaches to a Dirac point distribution.

Relying on (2.14), ES approximates the expected gain in information $\mathbb{E}[\Delta H](\boldsymbol{\theta})$ if we selected a certain location $\boldsymbol{\theta}$ as next evaluation point. In Fig. 2.2 we show $\mathbb{E}[\Delta H](\boldsymbol{\theta})$, computed for the GP posterior of Fig. 2.1(c). ES would select as next evaluation point the location where the blue line reaches its maximum.

In addition to suggesting the next evaluation, ES also outputs its current best guess of the minimum location; that is, the maximum of its approximation to $p_{\min}(\boldsymbol{\theta})$.

2.2.4 Automatic LQR tuning

The proposed method for automatic LQR tuning is obtained by combining the LQR tuning framework from Section 2.1 with ES; that is, using ES to solve (2.8). At every iteration, ES suggests a new controller (through $\boldsymbol{\theta}$ with (2.6)), which is then tested in an experiment to obtain a new cost evaluation (2.9). Through this iterative procedure, the framework is expected to explore relevant regions of the cost (2.3), infer the shape of the cost function, and eventually yield the global minimum within $\mathcal{D}$. The automatic LQR tuning method is summarized in Algorithm 1.

The performance weights $(\boldsymbol{Q}, \boldsymbol{R})$ encode the desired performance for the system (2.1). Thus, a reasonable initial choice of the parameters $\boldsymbol{\theta}$ is such that the design weights $\left(\bar{\boldsymbol{Q}}(\boldsymbol{\theta}), \bar{\boldsymbol{R}}(\boldsymbol{\theta})\right)$ equal $(\boldsymbol{Q}, \boldsymbol{R})$. The obtained initial gain $\boldsymbol{F}$ would be optimal if (2.2) were the true dynamics. After N evaluations, ES is expected to improve this initial gain based on experimental data representing the true dynamics (2.1).

2.2.5 Relation to other GP-based optimizers

Similar to ES, there exist a number of GP-based optimization algorithms, which infer the objective function from evaluations to search for the function's optimum. However, these methods do not have an explicit notion of the function's extremum as in (2.13). Typically, they aim to collect low function values, rather than maximizing information gain.

GP optimization methods like *probability of improvement* (PI) [25] or *expected improvement* (EI) [24] also construct a GP posterior from function evaluations in the same way as ES. Instead of maximizing information gain about the minimum location, however, they maximize heuristic utility functions that are designed to have high values close to the function's minimum. In contrast to $p_{\min}$, which is a global measure, these utilities represent local quantities. This leads PI and EI to typically evaluate in regions of low function values, while ES might evaluate in high cost regions if this leads to *learning* most about the location of the minimum.

Another type of algorithms treat optimization in the continuous bandit setting, like [29] relying on *Gaussian process upper confidence bound* (GP-UCB) [30]. In the bandit setting, the goal is to minimize regret; that is, the sum of function values at each iteration. Similar to EI and PI, this typically leads to evaluations in a local region of the expected minimum location to avoid incurring high costs, while ES is able to explore globally, disregarding the function value itself.

The bottom line is that ES is designed for problems where, after a finite number of experiments, the best guess for the minimum is to be made. This corresponds to the controller tuning scenario discussed in the introduction. Instead, algorithms like EI, PI and GP-UCB, combined with the continuous bandit setting, aim to collect a sequence of low function values, which may be relevant in other tuning scenarios where performance during exploration matters. Numerical experiments in [8] demonstrate that ES outperforms these three methods in the sense of obtaining better guesses for the true minimum with fewer evaluations.

Chapter 3

ROBOTIC PLATFORM

This chapter merely summarizes representative aspects of the hardware (Sec. 3.1) and software (Sec. 3.2) that conform the robotic platform used as experimental demonstrator for testing the proposed framework. In addition, Sec. 3.3 briefly describes the underlying low-level controllers structures, which were critical for achieving and improving the pole balancing, and thus, reaching the goals of this thesis. However, readers not very interested in technical details about the robotic platform, but more in understanding the core parts of this thesis (Chapter 2 and Chapter 4) can safely skip this part.

3.1 Hardware

This section briefly explains the four pillars of the robotic platform: the robot, the pole, the motion tracking system, and the real-time workstation.

Robot Apollo

The robot Apollo consists of two humanoid arms, to which two robotic hands are mounted as end-effectors, and a SARCOS head.

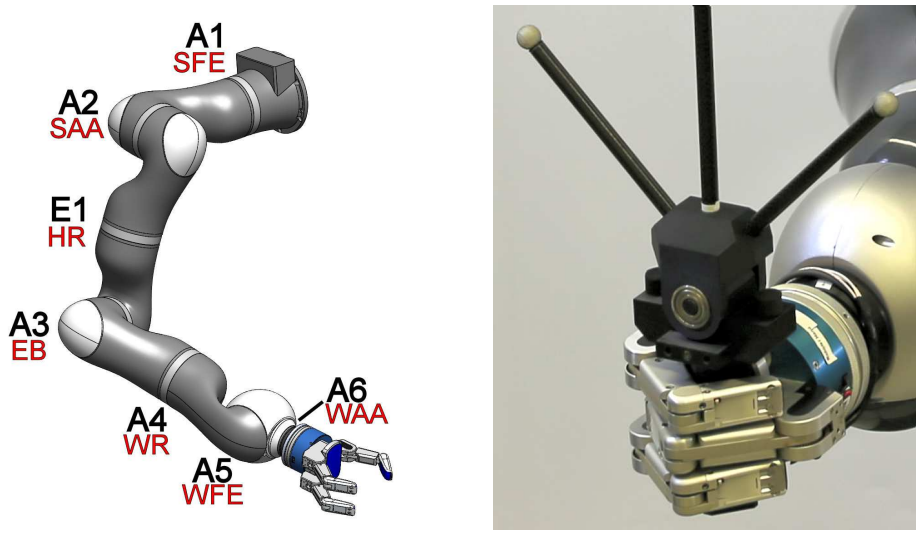
Each humanoid arm is a seven degree-of-freedom (DOF) robot arm LBR4+ from KUKA Laboratories (see Fig. 3.1 (a)).

A BH8-series BarrettHand is mounted as end-effector on each arm. One of the end-effectors is used to hold the handle of the pole (see Fig. 3.1 (b)).

Inverted pole

The pole used on the balancing problem consists of a stick linked to a handle through a rotatory joint with one DOF, as can be seen in Fig. 3.2. The red ball on the top is used to increase its inertia. Several visual markers (grey balls) are attached to it to allow motion tracking. A detailed sketch of the design of the pole can be found in Appendix A.

FIGURE 3.2: Short and long poles used on the balancing experiments.



(a) Robot arm (b) Robot hand

FIGURE 3.1: In (a) is shown a computer-aided design (CAD) model of a LBR4+ Kuka arm, with a BarretHand mounted as end-effector. In black, KUKA naming scheme. In red, Simulation Laboratory (SL) naming scheme [31]. In (b) is shown a detail of the end-effector holding the inverted pole.

VICON System

The motion tracking is achieved with four VICON cameras distributed around the robot. These cameras are connected to a workstation that provides for real-time position and orientation measurements. These data are passed trough a real-time safe communication bridge to the Real-time workstation.

Real-time workstation

The robot is controlled from a 8-cores 3.5GHz workstation under Ubuntu 12.04 extended by Xenomai to guarantee real-time constraints. The data from the motion tracking system and the readings from the joint encoders of the are sent to this machine, which accordingly generates and sends torque signals to the robot.

3.2 Software

Simulation Library (SL)

The actual communication is realized by Simulation Laboratory (SL), a soft-ware package for simulation and real-time robot communication and control, written by Stefan Schaal [32]. SL consists of different processes, running in parallel, performing various tasks. Running with 1 kHz on Apollo, the motor servo receives and filters measurements from the joint angle

encoders, numerically derives angular velocities and accelerations, and sends torques to the joints. The measurements are passed to the task servo, possibly running at 1 kHz, too. Based on the current joint states measured by the motor servo, the task servo computes new desired joint states. Additionally, SL offers the possibility to run the same configuration on the simulation servo, instead of on the real robot. Featuring a precise model of Apollo's dynamics and a powerful visualization tool, user tasks can easily be tested in simulation first [31].

Motion tracking program

The angular position of the pole is computed, with respect to a fixed coordinate frame, by processing the data from the commercial VICON software. This measured angle is then sent to SL through a real-time communication bridge, programmed using ROS.

Entropy Search

Entropy Search, used to solve the optimization problem, runs in MATLAB, and is publicly available [8].

3.3 Low-level control of the robot arm

For the pole balancing problem, we assume we can control the pole by commanding end-effector accelerations $u(t)$. In reality, these end-effector accelerations are realized through an appropriate tracking controller for the end effector. For this tracking controller, we follow a similar control structure as the one proposed in [33] except for the force feedback control loop, which is not needed here.

The desired torques τ_l for each joint $l = \{1, \ldots, 7\}$ are computed as the combination of a feedback plus a feedforward term:

$$\tau_l(t) = \tau_l^{\text{ff}}(t) + \tau_l^{\text{fb}}(t)$$

Feedback term

The feedback term τ_l^{fb} is obtained by applying PD position control on the joint space and PID position control at the end-effector space, i.e. the cartesian space. We briefly describe next each one of these two control strategies.

The position control at each joint is achieved as

$$\tau_l^{\text{fb},1}(t) = P_l(\vartheta_l^{\text{des}}(t) - \vartheta_l^{\text{cur}}(t)) + D_l(\dot{\vartheta}_l^{\text{des}}(t) - \dot{\vartheta}_l^{\text{cur}}(t))$$

where P_l and D_l represent the proportional and derivative gains of each joint l, $\vartheta_l^{\text{des}}(t)$ represents the desired joint angle and $\vartheta_l^{\text{cur}}(t)$ represents the current joint angle, measured by the joint encoders. The corresponding derivatives are also measured using the joint encoders.

This layer of control by itself ensures certain level of position control at the end-effector. In fact, fairly good tracking can be achieved in manipulation tasks, for which only slow movements of the end-effector are required.

Certainly, raising up the gain P_l can increase the end-effector positioning precision, but at expenses of having a stiff arm. The stiffer the arm, the higher is the risk of the motors getting broken if the arm hits a rigid obstacle.

In addition, increasing D_l might damp the tracking of the end-effector. However, noisy sensor measurements limit how much this gain can be increased, i.e., how much the measurement noise gets amplified.

Increasing the accuracy of the end-effector tracking, while preserving low PD gains (i.e., high compliance) on the robot joints, is nonetheless possible. In the approach described in [33], a cartesian feedback control loop is implemented in top of the typical position control layer, described above. The cartesian position error is directly mapped into the joint torques as shown below:

$$\tau_l^{\text{fb},2}(t) = \boldsymbol{j}_l^T(t) \left(\boldsymbol{P} \left(\boldsymbol{r}^{\text{des}}(t) - \boldsymbol{r}^{\text{cur}}(t) \right) + \boldsymbol{D} \left(\dot{\boldsymbol{r}}^{\text{des}}(t) - \dot{\boldsymbol{r}}^{\text{cur}}(t) \right) + \boldsymbol{I} \int \left(\boldsymbol{r}^{\text{des}}(t) - \boldsymbol{r}^{\text{cur}}(t) \right) \mathrm{d}t \right)$$

where $\boldsymbol{j}_l^T$ represents the row l of the transpose jacobian at time t. Proportional, derivative and integral matrix gains are represented by $\boldsymbol{P}$, $\boldsymbol{D}$ and $\boldsymbol{I}$ respectively. Finally, $\boldsymbol{r}^{\text{des}}(t) \in \mathbb{R}^3$ represents the desired position in the cartesian space, and $\boldsymbol{r}^{\text{cur}} \in \mathbb{R}^3$ represents the current position, obtained the forward kinematics.

The whole feedback term is computed as the superposition of these two control layers $\tau_l^{\text{fb}}(t) = \tau_l^{\text{fb},1}(t) + \tau_l^{\text{fb},2}(t)$.

Although the pole balancing was already achieved only with the first layer of control, we believe that applying the second layer has been a critical part for achieving the goals of this thesis.

Feedforward term

The feedforward term $\tau_l^{\text{ff}}(t)$ accounts for the gravity compensation and the inverse dynamics of the robot arm. An approximate non-linear model of the robot arm is used for computing it.

Chapter 4

EXPERIMENTAL RESULTS

In this chapter, we present auto-tuning experiments for the pole balancing problem shown in Fig. 1.1 that serve to demonstrate the automatic controller tuning framework on a complex robotic platform. The system is described in Sec. 4.1, the implementation choices are presented in Sec. 4.2, and the results are discussed in Sec. 4.3 and Sec. 4.4.

4.1 System description

We consider a one-dimensional balancing problem: a pole linked to a handle through a rotatory joint with one degree of freedom (DOF) is kept upright by controlling the acceleration of the end-effector of a humanoid arm. Figure 1.1 shows the set-up for two poles of different length.

The angle of the pole is tracked using an external motion capture system. We assume we can control the pole balancing by commanding end-effector accelerations $u(t)$ (see Chapter 3 for more information about the hardware and software).

The continuous-time dynamics of the balancing problem (similar to [34]) are described by:

$$\begin{aligned} mr^2\ddot{\psi}(t) - mgr\sin\psi(t) + mr\cos\psi(t)u(t) + \xi\dot{\psi}(t) = 0 \\ \ddot{s}(t) = u(t) \end{aligned} \tag{4.1}$$

where $\psi(t)$ is the pole angle with respect to the gravity axis, $s(t)$ is the deviation of the end-effector from the zero position, and $u(t)$ is the end-effector acceleration.

Two poles with different lengths are used in the experiments. The center of mass of the short pole lies at $r \simeq 0.33$ m from the axis of the rotatory joint, its mass is $m \simeq 0.27$ kg, the friction coefficient is $\xi \simeq 0.012$ Nms, and the gravity constant is $g = 9.81$ m/s^2. For the long pole, we have $r \simeq 0.64$ m and $m \simeq 0.29$ kg.

A model (2.2) of the system is obtained by linearization of (4.1) about the equilibrium $\psi = 0$, $s = 0$ and discretization with a sampling time of $T_s = 1$ ms. Using the parameters of the short

pole, we obtain its nominal model 4.2.

$$\boldsymbol{A}_{\mathrm{n}} = \begin{bmatrix} 1.0 & .001 & 0 & 0 \\ .029 & .99 & 0 & 0 \\ 0 & 0 & 1.0 & .001 \\ 0 & 0 & 0 & 1.00 \end{bmatrix}, \boldsymbol{B}_{\mathrm{n}} = \begin{bmatrix} 0 \\ -.003 \\ 0 \\ .001 \end{bmatrix} \tag{4.2}$$

The non-linear model (4.1) assumes that we can command a discretized end-effector acceleration u_k as control input to the system.

The estimated end-effector position s_k and velocity $\dot{s}_k$ are computed at a sampling rate of 1kHz from the robot's joint encoders using forward kinematics. The pole orientation is captured at 200 Hz by the motion capture system. From these data, we obtain estimates of pole angle ψ_k and angular velocity $\dot{\psi}_k$ through numerical differentiation and low-pass filtering (2nd-order Butterworth, 10 Hz cutoff). With this scheme, no model is required to obtain estimates of all states (in contrast to the Kalman filter used in [1]), and it can be used irrespective of which balancing pole is used. The complete state vector of (2.2) is given by $\boldsymbol{x}_k = [\psi_k, \dot{\psi}_k, s_k, \dot{s}_k]^{\mathrm{T}}$.

To compensate for any steady-state error in the end-effector position, we augment the LQR controller by integral action on s_k. That is, we implement the control law $\boldsymbol{u}_k = \boldsymbol{F}\boldsymbol{x}_k + F_z z_k$ instead of (2.4), where z_k is the integrator state and the gain $F_z = -0.3$ is fixed. The gain $\boldsymbol{F}$ is obtained by the LQR tuning as previously described. Since the integrator state is an artificial state, it is not included in the computation of the experimental cost (2.9).

4.2 Automatic LQR tuning: Implementation choices

We choose the performance weights to be

$$\boldsymbol{Q} = \mathrm{diag}(1, 100, 10, 200), \;\; \boldsymbol{R} = 10 \tag{4.3}$$

where $\mathrm{diag}(\cdot)$ denotes the diagonal matrix with the arguments on the diagonal. We desire to have a quiet overall motion in the system. Therefore, we penalize the velocities $\dot{\psi}_k$ and $\dot{s}_k$ more than the position states.

We conducted two types of tuning experiments, one with two parameters and another one with four. The corresponding design weights are

- 2D tuning experiments:

$$\bar{\boldsymbol{Q}}(\boldsymbol{\theta}) = \mathrm{diag}(1, 50\theta_1, 10, 50\theta_2), \;\; \bar{\boldsymbol{R}}(\boldsymbol{\theta}) = 10 \tag{4.4}$$

  where the parameters $\boldsymbol{\theta} = [\theta_1, \theta_2]$ can vary in $[0.01, 10]$, and $\boldsymbol{\theta}^0 = [2, 4]$ is chosen as initial value.

- 4D tuning experiments:

$$\bar{\boldsymbol{Q}}(\boldsymbol{\theta}) = \text{diag}(\theta_1, 25\theta_2, 10\theta_3, 25\theta_4), \;\; \bar{\boldsymbol{R}}(\boldsymbol{\theta}) = 10 \tag{4.5}$$

  with $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4]$, $\theta_j \in [0.01, 10]$, and $\boldsymbol{\theta}^0 = [1, 4, 1, 8]$.

In both cases, the initial choice $\boldsymbol{\theta}^0$ is such that the design weights equal the performance weights. That is, the first controller tested corresponds to the nominal LQR design (2.5).

Balancing experiments were run for 2 minutes, i.e. a discrete time horizon of $K = 1.2 \cdot 10^5$ steps. Because the nominal model does not capture the true dynamics, some LQR controller obtained during the tuning procedure may destabilize the system. This means that the system exceeded either acceleration bounds or safety constraints on the end-effector position. In these cases, the experiment was stopped and a fixed heuristic cost J_u was assigned to the experiment. Values for J_u are typically chosen slightly larger than the performance of a stable but poor controller. We used $J_u = 3.0$ and $J_u = 5.0$ for the 2D and 4D experiments, respectively.

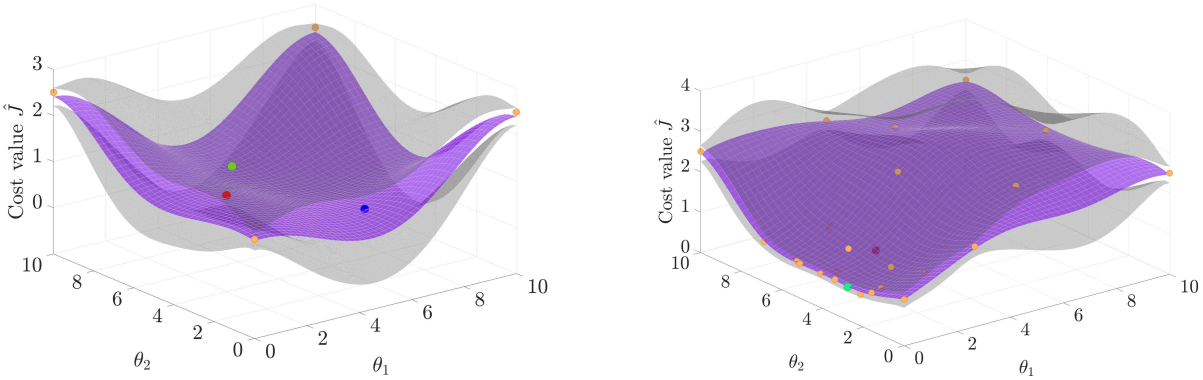
Before running ES, a few experiments were done to acquire knowledge about the hyperparameters $\mathcal{H}$. Accordingly, a Gamma prior distribution was assumed over each hyperparameter with expected values and variances gathered in Table 4.1. For the first iteration of ES, we use these expectations as initial set $\mathcal{H}$. After each iteration, $\mathcal{H}$ is updated as the result of maximizing the GP marginal likelihood.

4.3 Results from 2D experiments

For the 2D experiments (4.4), we first use a short pole (Fig. 1.1, right) and the best available linear model , showing that the framework is able to improve the initial controller. Secondly, we exchange the pole with one of double length (Fig. 1.1, left), but keep the same nominal model . We show, for the latter case, that even with a 50% underestimated model, the framework finds a stable controller with good performance. In both cases, we use the design weights (4.4).

TABLE 4.1: Characterization of the gamma prior over $\mathcal{H}$

	2D exploration		4D exploration	
	$\mathbb{E}[\cdot]$	Std $[\cdot]$	$\mathbb{E}[\cdot]$	Std $[\cdot]$
Lengthscale λ_j	2.5	0.11	2.00	0.63
Signal variance σ	0.2	0.02	0.75	0.075
Likelihood noise σ_{n}	0.033	0.0033	0.033	0.010



(a) ES initialization

(b) GP posterior after 20 iterations

FIGURE 4.1: GPs at (a) the start and (b) the end of the first tuning experiment. The GP mean is represented in violet and $\pm$ two standard deviations in grey. The red dot corresponds to the initial controller, computed at location $\boldsymbol{\theta}^0 = [2, 4]$. The green dot represents the current best guess for the location of the minimum. The blue dot is the location suggested by ES to evaluate next, and orange dots represent other evaluations. The best guess found after 20 iterations (green dot in (b)) has significantly lower cost than the initial controller (red dot).

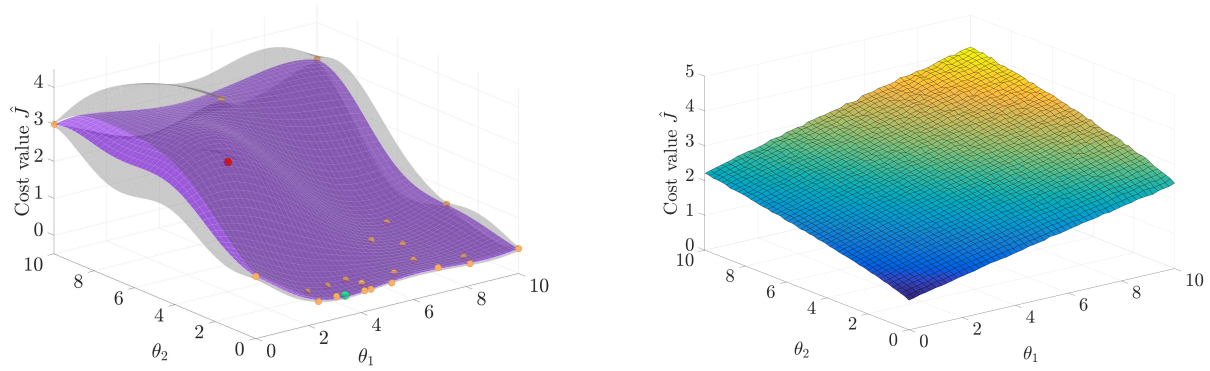
4.3.1 Using an accurate nominal model

ES was initialized with five evaluations, i.e. the initial controller $\boldsymbol{\theta}^0$, and evaluations at the four corners of the domain $[0.01, 10]^2$. Figure 4.1 (a) shows the 2D Gaussian process including the five initial data points. The algorithm can also work without these initial evaluations; however, we found that they provide useful prestructuring of the GP and tend to speed up the learning. This way, the algorithm focuses on interesting regions more quickly.

Executing Algorithm 1 for 20 iterations (i.e. 20 balancing experiments) resulted in the posterior GP shown in Figure 4.1 (b). The "best guess" $\boldsymbol{\theta}^{BG} = [0.01, 2.80]$ (green dot) is what ES suggests to be the location of the minimum of the underlying cost (2.3).

In order to evaluate the result of the automatic LQR tuning, we computed the cost of the resulting controller (best guess after 20 iterations) in five separate balancing experiments. The average and standard deviation of these experiments are shown in Table 4.2 (left, bottom), together with the average and standard deviation of the initial controller, computed in the same way before starting the exploration (left, top). Even though the initial controller was obtained from the best linear model we had, the performance was still improved by 31.9%.

The direction of change of the parameters suggests that a less aggressive penalization in the pole angular velocity was necessary to calm down the motion of the balancing. This apparent contradiction could be explained following the next reasoning: first, the lower are the penalization in the LQR weights, the lower are the resulting LQR gains, in general. Second, any noise in the states signals will be amplified by their corresponding LQR gains, and transmitted to the control input u_k, i.e., the commanded end-effector accelerations, following (2.4). A noisy control input u_k will make the balancing shaky, providing a poorer performance and a higher



(a) GP posterior after 20 evaluations

(b) Cost function in simulation

FIGURE 4.2: In (a) is shown the final GP posterior for the second tuning experiment using a wrong nominal model. The color scheme is the same as in Fig. 4.1. In (b) can be seen the theoretical cost function reproduced in simulation assuming a linear plant and artificial noise

cost. Therefore, lowering the LQR gains (i.e., lowering the LQR weights) might be beneficial for the case we have noisy measurements, like is the case of the pole angular velocity.

4.3.2 Using a wrong nominal model

In this experiment, we take the same nominal model as in the previous case, but we use a longer pole in the experimental demonstrator (Fig. 1.1, left). The initial controller, computed with $\boldsymbol{\theta}^0$, destabilizes the system, which can be explained by the nominal model significantly misrepresenting the true dynamics. As shown in Figure 4.2, after 20 iterations, ES suggested $\boldsymbol{\theta}^{BG} = [3.25, 0.01]$ as the best controller. The results of evaluating this controller five times, in comparison to the initial controller, are shown in Table 4.2 (middle columns).

The posterior GPs of both experiments (Fig. 4.1(b) and Fig. 4.2(a)) shows that the entire parameter region is explored, which is a distinguishing feature of ES (cf. Sec. 2.2.5).

The resulting set of parameters implies a smaller penalization of the end-effector velocity, which suggests a less aggressive damping as the right direction for stabilizing the long pole. Since the LQR gains use the model of the short pole (which requires higher damping to be

TABLE 4.2: Cost values for three tuning experiments

	2D exploration				4D exploration	
	$\hat{J}$ (Best model)		$\hat{J}$ (Under. model)		$\hat{J}$ (Under. model)	
	mean	std	mean	std	mean	std
$\boldsymbol{\theta}^0$	1.12	0.11	3.00	-	5.00	-
$\boldsymbol{\theta}^{\text{BG}}$	0.76	0.058	0.058	0.012	0.040	0.0031

balanced) the direction of the parameters shows how the framework, by tuning gains, automatically stabilizes the system, skipping any additional system identification step.

A comparison with a theoretical cost function is also shown in Figure 4.2 (b). The balancing system with the long pole was simulated following a linear plant, computed using its physical parameters. Measurement noise is sampled from a Gaussian distribution with zero mean and standard deviation previously computed from the motion tracking system. The same nominal model used in the corresponding experiment (i.e., the model of the short pole) is used to compute the LQR controllers at each location through (2.6). This analysis shows that unmodeled nonlinearities of the real plant are reflected in the cost function during the exploration, making it look significantly different from the simulated one. This reinforces the use of a global optimizer, even in a low-dimensional space (2D), for solving such a non-convex problem, rather than a local optimizer. In addition, it indicates that further system identification steps for improving the linear model may not help to improve the balancing performance. This difference might show up more remarkably in higher dimensional problems, e.g., the balancing of a humanoid robot.

Video demonstration

The results presented in Sec. 4.3.2 are also shown in a video demonstration, attached as complementary material to this thesis. It can also be visualized on-line following this link.

4.4 Results from 4D experiments

The 4D tuning experiments, realized with the long pole, uses the same nominal model as in the previous experiments (i.e., a wrong linear model for the real plant), and the design weights (4.5). We show that the framework is able to improve the controller found during the 2D experiments with the long pole, in a higher dimensional space.

The first controller $\boldsymbol{\theta}^0$ destabilizes the system. After 46 iterations, ES suggests $\boldsymbol{\theta}_{BG} = [4.21, 7.47, 0.43, 0.01]$, which in comparison with the 2D experiments with the long pole, performs about a 31.7% better (see Table 4.2). We actually ran this experiment until iteration 50, however, the algorithm did not lead to further improvements.

Figure 4.3 shows the cost function evaluations over the course of the tuning experiment. The fact that unstable controllers are obtained throughout the experiment reflects how the global search tends to cover all areas.

Before starting the auto-tuning in 2D experiments, we spent some effort selecting the method's parameters, such as hyperparameters and parameter ranges. In contrast, we started the 4D experiments without such prior tuning. In particular, we kept the same performance weights, chose similar design weights, and started with the same values for the hyperparameters $\mathcal{H}$ and

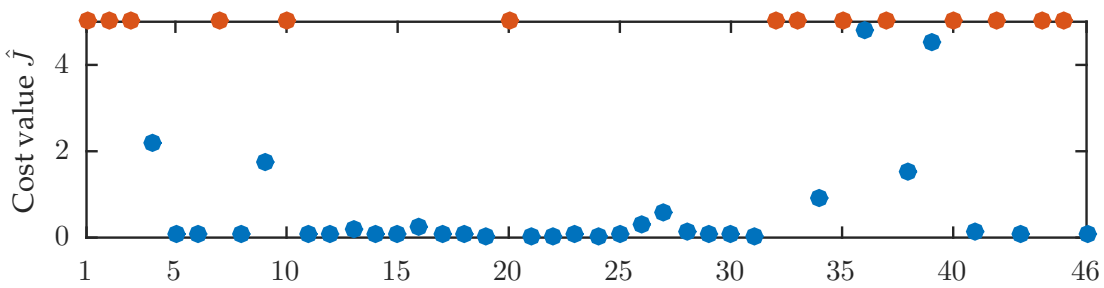


FIGURE 4.3: Cost values obtained at each experiment differentiating stable controllers (blue dots), and unstable controllers (red dots).

penalty J_u. However, slight adaptations of these parameters were found necessary in response to issues encountered. We briefly describe these next.

We chose initially a penalization for unstable controllers $J_u = 3.0$. After a few runs of the auto-tuning, we found a stable controller (with poor performance), which yielded a cost around 3.5. Since a stable controller should not be penalized more than an unstable one, we stopped the tuning, selected $J_u = 5.0$, and restarted the experiment. Furthermore, we realized after a few iterations of ES that hyperparameters were not changing significantly in response to the obtained data. Therefore, after about 10 iterations we decided to stop the tuning and changed, in particular, the gamma prior variances to those gathered in Table 4.1.

In general, any method reasoning about functions on continuous domains from a finite number of data points relies on prior assumptions (see [8, Sec. 1.1] for a discussion). We were quite pleased with the outcome of the tuning experiments and, in particular, that not much had to be changed moving from the 2D to 4D experiment. Nonetheless, developing general rules for choosing the parameters of GP-based optimizers like ES (maybe specific for certain problems) seems important for future developments.

Chapter 5

SCHEDULES AND COSTS

This chapter gathers supplementary information related to logistical aspects about the development of this thesis. We summarize the planned scientific method followed for its development and the amount of resources involved. Without entering in many technical details, we pretend to reflect the importance of decision making to accomplish such work in a limited amount of time.

5.1 Initial plan, real path and mismatches

This section describes the initially proposed plan, the problems encountered during the execution of this, and the eventually followed path. In spite of the mismatch between the initial and final plans, we show that the final goal of this thesis was nevertheless reached. Furthermore, extra results out of the initial scope of this work, were achieved.

5.1.1 Initial plan

Thesis proposal

This work was publicly announced from beginning of 2014, in the webpage of the Institute for Dynamic Systems and Control at ETH Zürich (idsc.ethz.ch) under the title *Gaussian Process Optimization for Self-Tuning Control*.

Initial time schedule

The initially planned duration for this work was about six months, from end of January 2015 till end of July 2015, working about 40 hours per week.

Original goal

The original goal was to employ ES for automatic tuning controllers using the LQR formulation (see Chapter 2 for a detailed description) using Apollo balancing an inverted pole as experimental demonstrator. In case the investigation was successful, it was proposed to record

a video demonstration showing the effectiveness of the framework by parametrizing in 2 dimensions, using the aforementioned robotic platform. The video had to show the framework dealing with a case in which only a wrong linearized model of the plant was available. The motivation for this idea to work in reality was supported by previous research done S. Trimpe and others on 2014.

Precedent status

Right before starting this work, the robot Apollo was programmed to balance a Furuta pendulum [35, 36], which implies a different setting than the one considered in this thesis. While in this balancing problem they only allow rotation in one of the joints of the robot arm, we consider the movement of the end-effector onto a line, for which all the joints are involved.

Visual feedback was obtained using an ASUS Xtion PRO LIVE device mounted on Apollo's head. It measured an RGB image and a depth cloud at a rate of 30 Hz with the purpose of estimating the pole tilting on real-time through a vision algorithm. This was done by, first identifying the centres of each ball with respect to the coordinate frame of the camera, and then computing the rotation angle of the pole assuming that it is balanced within a plane (see Fig. 5.1).

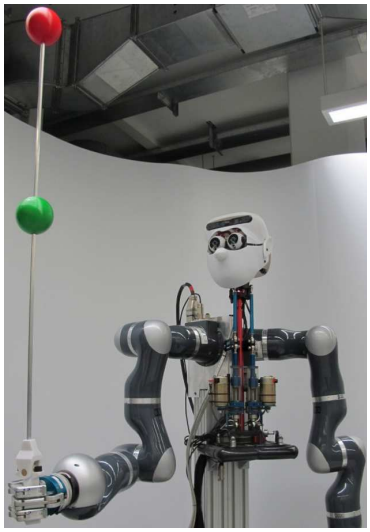


FIGURE 5.1: Original pole. Picture extracted from [31]

An LQR controller (implemented in SL) was used to map the measured pole and end-effector states into the desired accelerations of the actuated joint. These accelerations were commanded to the robot at a rate of 1 kHz in the closed-loop experiments.

This previous work was carried out by another master student. The goal of his thesis [31] was to balance the pole during an undefined amount of time, using the aforementioned setup. However, several non-solved issues avoided him reaching this goal. Instead, only balancing experiments about 6-7 seconds were achieved at maximum.

In general, using a depth cloud and RGB image to balance an inverted pole with a humanoid arm is not a trivial problem. It has never been solved before, but only close approaches like [34], that use stereo vision for obtaining measurements with higher acquisition rate (60 Hz), and a lower signal to noise ratio (SNR).

Originally, the goal of my thesis assumed the aforementioned set-up to be an experimental demonstrator working out-of-the-box for testing a theoretical approach. However, this was not the case. Therefore, it was a necessary condition to solve the pole-balancing problem on Apollo before starting to work on the actual thesis.

Sources of error

We identified four (possibly coupled) sources of errors. We summarize them next:

- The assumption of the pole moving in a plane might be too strong for the Fututa pendulum configuration (which assumes arc movements at the end-effector). Thus, we proposed to balance the pole in a vertical plane, i.e., by constraining the movement of the end-effector onto a line.
- Neither the tracking of the end-effector position (i.e., how well the end-effector tracks the commanded trajectory) nor of the joint positions and velocities are accurate enough for the bandwidth of frequencies involved in the balancing.
- The measurement noise of the ASUS device might be too high for such a task even after a filtering stage that has to deal gently with the induced measurement delays.
- The delay of the ASUS device itself (around 60ms [31]) plus de computational effort of the vision program (around 30ms [31]) might be too high for such control problem.

Planed steps

The original plan is divided in two parts: first, solve the issues related to the balancing; second, apply ES to automatically improve an LQR controller.

Steps for solving the issues related to the balancing (to be solved in about three months):

1. State-of-the-art: understand and implement in simulation the algorithm proposed in [6]. Read previous work about cart-pole balancing problems [34–38]. Read, understand and use Entropy Search [8] on a toy-example in MATLAB. Read [27] to understand better how non-parametric non-linear regression with Gaussian Process works, both from a theoretical and from a practical point of view.
2. Get familiar with SL, through a series of initial examples in simulation, as training for working with the real robot.
3. Implement an LQR controller for cart-pole balancing in SL, in simulation, and test it on the real robot.
4. Implement a Kalman Filter (KF) in SL, in simulation, that estimates the pole angular position and velocities when measurements are not available. Since the pole angular position is delayed (about 90ms), a state prediction stage helps to have an estimate of the state at the current time relying on a linearized model of the system.
5. Double-check with MATLAB the previous results by implementing a KF with prediction stage using the simulated data of the states and control input.

6. Identify a linearized model of the pole through standard system identification techniques.
7. Test the implemented KF with the identified model of the pole on the real robot. If the balancing works, then the major problem was the ASUS device's delay. If not, continue.
8. Improve the tracking of the end-effector and joints positions of the robotic arm by commanding steps and sinusoidal trajectories[1].
9. Test the balancing again on the real robot. If the balancing works, then the major problem was the end-effector tracking.

Steps to test the automatic tuning framework using ES as optimizer (to be started after steps 1 to 9, and completed in about three months):

10. Run ES in MATLAB for a toy example: locate the global minimum of a parabola function in one dimension trough evaluations corrupted by noise. Understand more in deep how ES and the Gaussian Process regression works for a one dimensional case.
11. Simulate the cart-pole balancing problem in MATLAB with process noise and measurement noise and integrate it with ES.
12. Integrate ES with Apollo balancing a pole in SL, in simulation.
13. Run an ES exploration parametrizing in two dimensions.
14. Record a video demonstration.
15. Write this thesis.

5.1.2 Followed plan:

Solving the initial problems

After following the steps from 1 to 9, the balancing was still unsatisfactory. Further efforts, like increasing the acquisition rate of the ASUS device to 60 Hz, at the expense of frames with half the resolution, were still unsatisfactory. One of the main reasons is that frames with half the resolution imply higher SNR at the pole angle measurement.

In order to decouple possible sources of error, we exchanged the ASUS device by a motion tracking system (VICON) to estimate the pole angular position with higher precision. With such a ground truth at the vision part of the problem, potential inaccuracies at the end-effector tracking would show up.

[1] This is a time-consuming trial-and-error process that requires increasing the PD gains of each joint individually and compare the desired signal with the actual signal aiming at reducing the error between them. However, it is desirable to maintain the robot with a certain level of compliance. Low compliance (i.e., high stiffness) can improve tracking, but also involves some risks: if the robot hits an static object the motors might break.

After adjusting the KF parameters and the LQR gains, the pole was balancing for an undefined amount of time.

Still, the tracking of the end-effector could be improved. For this, we referred to the literature, where [33] implemented a cartesian feedback control loop, in top of the typical position control layer, for manipulation tasks. This solution made the increased significantly the accuracy of the end-effector tracking, while high compliance on the robot joints was preserved (see Sec. 3.3 for a detailed explanation).

The aforementioned solutions were run in parallel with steps 10 and 11. In total, around 10 weeks were necessary to achieve the pole balancing, and around 5 weeks more to improve the end-effector tracking and polish low-level controllers. This left 9 weeks for accomplishing steps 12 to 14. Nonetheless, some intermediate steps (not planned at the beginning of this work) turned out to be necessary to reach the proposed goals. This made the remaining time unrealistic to finish the work. These intermediate steps are enumerated below.

Needed intermediate steps

15. Identify the linearized model of the new pole using system identification techniques.

16. Measure the delay of the VICON system. For this, we placed a VICON object on top of the hand of the robot. Then, we commanded sinusoidal and step movements to the wrist. By comparing the phase lag between the readings from the joint encoders of the wrist and the angle of the VICON object, we measured it (about 15 ms).

17. Understand the role of the hyperparameters of the kernel, and the kernel itself with real data. For this, several pole balancing runs were necessary to acquire experimental data.

18. Use a model-free filter, such as Butterworth filter, instead of the previously implemented KF, that is model-dependent. The automatic LQR tuning framework needs an approximate linearized model of the system. However, the approaches used for measuring the system variables should be model-agnostic, regarding the principles of this framework.

Reached goal

The original goal of this work was successfully reached. In fact, we went further and run an exploration parametrizing in 4D, being able to achieve better performance than in 2D. See Chapter 4 for a detailed explanation.

Final schedule

Understanding, designing, building up and testing this framework to demonstrate its validity required a certain amount of time, resources and assistance. The total amount of time spent

is roughly 7 months and a half: from end of January 2015 until middle of October 2015, discounting the whole month of August. On average, 10 hours per day including weekends were dedicated to this thesis.

5.1.3 Mismatches

None of the steps proposed in the initial plan was skipped. However, some intermediate steps (15 to 19) were needed to reach the proposed target.

The proposed goals were reached, exactly as stated at the initial schedule of this work. In addition, the results were extended to higher dimensional problems, albeit out of the scope of the initial plan.

We also showed this work during its development to the scientific community through one poster presentation, one oral presentation in a conference and a submitted conference paper (see Sec. 6.2 for details). All of this extra effort and dedication, together with solving the balancing problem on Apollo are the main reasons for the increase on the amount of time put on this work.

Summarizing, 6 weeks more were needed to complete this thesis, with an extra of 30 hours per week, and 2 more working days. In total, the initially planned time effort was increased by about 120% in order to achieve the desired goals.

5.2 Cost of the solution and other alternatives

The cost of the solution itself is null, since this thesis proposes a methodology for automatic controller tuning, independent of the experimental platform. The methodology is a compendium of algorithms, ideas, and theory, which belongs to the public knowledge as much as the stated contributions to the scientific community. Although these methods are tested in

TABLE 5.1: Approximated cost of each item (in euros)

Initial plan		Followed plan	
Item	Price	Item	Price
Workstation	8,000	Workstation	8,000
Screen (x2)	2,000	Screen (x2)	2,000
Real-time workstation	6,000	Real-time workstation	6,000
SARCOS Head	60,000	SARCOS Head	60,000
Barrent hand (x2)	50,000	Barrent hand (x2)	50,000
Kuka arm (x2)	188,000	Kuka arm (x2)	188,000
ASUS Xtion PRO LIVE	200	VICON camera (x4)	60,000
		VICON Workstation	10,000
Total	314,200	**Total**	384,000

a robot demonstrator, the proposed framework is not limited to it. In fact, we believe it can be scaled to more complex settings such as the industry of robots or water supply (see Sec. 6.2 for more information), or to other more complex humanoid robots.

We summarize in Table 5.1 the cost of the core resources needed for testing this framework. We distinguish between the cost of the resources if the initial plan had been followed in contrast with the cost of the actual followed plan. Using a VICON system instead of an ASUS Xtion increases the costs about 22 %. Therefore, trying to accomplish the pole balancing and test the proposed automatic tuning framework with Xtion is definitely desirable to reduce costs. Furthermore, from a practical standpoint, Xtion is easier to handle and to transport that VICON.

Chapter 6

IMPACT OF THIS WORK

6.1 Environmental impact

Regarding the directive 2011/92/EU of the European Parliament and of the council of 13 December 2011 *on the assessment of the effects of certain public and private projects on the environment*, published on 28.1.2012 on the Official Journal of the European Union (L26), we notice in article 1, point 1 that specifications about the environmental impact only applies to public projects:

> The directive shall apply to the assessment of the environmental effects of those public and private projects which are likely to have significant effects on the environment.

More in detail, in Article 1, point 2 (a), a *project* is defined within this context:

> project means the execution of construction works or of other installations or schemes, or, other interventions in the natural surroundings and landscape including those involving the extraction of mineral resources.

Since the present work is not a project, but a thesis, the aforementioned regulation does not apply.

6.2 Social impact

The automatic tuning controller proposed in this thesis does currently not have any impact on the society else than four demonstrations to the research community. However, there exists a prospective impact in the future.

Immediate social impact

This work is presented to the scientific community in four occasions:

- **1st Symposium on Intelligent Systems in Science and Industry (Germany, July 2015):** poster presentation of early experimental results.

- **Machine Learning in Motion Planning and Control of Robot Motion Workshop at iROS (Germany, September 2015):** oral presentation.
- **First Max Planck ETH Workshop on Learning Control (Germany, November 2015):** poster presentation.
- **ICRA (Sweden, May 2016):** paper submitted.

Prospective social impact

A draft of a potential social impact of this thesis in the near future is presented next. However, it must be merely understood as a hypothetical and even speculative set of ideas without any solid scientific base.

In the robotic industry

- Suppress the need of an operator to hand-tune controllers for robots. Only a supervisor at a high-level might be needed.
- Reduce the number of experiments and time needed for finding a suitable controller, in comparison to a human.

In the water supply industry

Efficiently tune the parameters that control water flow and supply time of the involved dams for a city. The observations would be the water offer-demand signals, and the cost to minimize would be the supply surplus.

In the drone industry

Automatically tune the controller involved in the steering system of a drone based on simple experiments. This idea has already applied on a research work with a drone by [39], but with different approaches to the ones of this thesis.

Chapter 7

CONCLUDING REMARKS

7.1 Achievements and future work

In this thesis, we introduce Bayesian optimization for automatic controller tuning. We develop, and successfully demonstrate in experiments on a robotic platform, a framework based on LQR tuning [6] and Entropy Search (ES) [8].

The main and novel contribution of this work is to be the first on employing ES for controller tuning and applying it on a physical robot platform. Bayesian optimization has recently gained a lot of interest in the research community as a principled way for global optimization of noisy, black-box functions using probabilistic methods. Thus, this work is an important contribution toward making these methods available for automatic controller tuning for robots.

The auto-tuning algorithm was demonstrated on a 2D and a 4D experiment, both when the method was initialized with an unstable and with a stable controller. While the 2D experiment could presumably also be handled by grid search or manual tuning, and thus mostly served as a proof of concept, the 4D tuning problem can already be considered difficult for a human. A key question for the development of truly automatic tuning methods is the amount of "prior engineering" that has to be spent to get the method to work. In particular, the 4D experiments were promising since not a lot of tuning, and only few restarts were necessary. However, questions pertaining to the prior choice or automatic adjustment of the method's parameters are relevant for future work.

This framework has been tested on the classical cart-pole balancing setting, as an entry level problem. Scaling this set-up to a higher dimensional problem, such as the balancing of a humanoid robot, is part of the future work.

Since the ES algorithm reasons about where to evaluate next in order to maximize the information gain of an experiment, we expect it to make better use of the available data and yield improved controllers more quickly than alternative approaches. Although ES has been shown to have superior performance on numerical problems (cf. discussion in Sec. 2.2.5), investigating whether this claim holds true in practice is future work.

Other interesting directions for future investigations concern extensions to different controller tuning scenarios. Herein, we considered a scenario, where a controller is allowed to fail during exploration (e.g., the robot was simply stopped when exceeding safety limits). In

other scenarios, it may be desirable or necessary to avoid unstable controllers; that is, to keep the system operational during the entire tuning procedure. While ES seeks to maximize the information obtained from an experiment, it seems reasonable to also include safety considerations such as avoiding unstable controllers. Safe learning in the context of control is an area of increasing interest and has recently been considered, e.g., in [5, 12, 40, 41] in slightly different contexts.

7.2 Critical decisions

In Sec. 3.3 we describe the low-level control layers applied for the tracking of the end-effector. Without the second layer, the pole balancing task can be achieved for an undefined time. However, the performance was poorer and the results less repeatable. At the time of evaluating the performance of the system through the approximate cost (2.9), we need certain level of repeatability. On the contrary, the SNR might be too high, and the observed cost function might be pure noise. For this reason, we believe that a more robust balancing, obtained with the aforementioned second layer of control, has been a critical part for reaching the goals of this thesis.

Another critical decision was to switch from ASUS Xtion to VICON in order to obtain the pole angle measurement. Based on our intuition, we could predict that the amount of time required to achieve the pole balancing using ASUS Xtion could have been easily in the same order of magnitude of the thesis itself. In the interest of reaching the planned goals of this thesis, we decided to rely on a ground truth regarding the motion feedback in order to focus on improving the end-effector tracking.

7.3 General conclusion

This thesis had an equilibrated amount of theoretical insights and practical work. Rather than testing the automatic tuning framework only in simulation, we were able to see it working on a real system. This made us learn from a practical standpoint how to deal with the limitations and unexpected problems of such complex scenario.

The novel nature of the work itself, and the promising results allowed us to show this work to the scientific community in several occasions.

Beyond opening research branches, this work might have a potential impact in the industry, were there is a clear need of reducing time spent on tuning controllers. In fact, a recent publication [12] from Bosch GmbH, in a similar topic, shows the interest of the industry on these kind of methodologies nowadays

Bibliography

[1] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. "Automatic LQR Tuning Based on Gaussian Process Optimization: Early Experimental Results". In: *Second Machine Learning in Planning and Control of Robot Motion Workshop at International Conference on Intelligent Robots and Systems*. 2015.

[2] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. "Automatic LQR Tuning Based on Gaussian Process Global Optimization". In: *IEEE International Conference on Robotics and Automation*. submitted. 2016.

[3] Sebastian Trimpe and Raffaello D'Andrea. "The Balancing Cube: A dynamic sculpture as test bed for distributed estimation and control". In: *IEEE Control Systems Magazine* 32.6 (2012), pp. 48–75.

[4] Sean Mason, Ludovic Righetti, and Stefan Schaal. "Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting". In: *14th IEEE-RAS International Conference on Humanoid Robots*. 2014, pp. 374–379.

[5] Anayo K Akametalu, Shahab Kaynama, Jaime F. Fisac, Melanie N. Zeilinger, Jeremy H. Gillula, and Claire J. Tomlin. "Reachability-based safe learning with Gaussian processes". In: *IEEE 53rd Annual Conference on Decision and Control*. 2014, pp. 1424–1431.

[6] Sebastian Trimpe, Alexander Millane, Simon Doessegger, and Raffaello D'Andrea. "A Self-Tuning LQR Approach Demonstrated on an Inverted Pendulum". In: *IFAC World Congress*. 2014, pp. 11281–11287.

[7] James C. Spall. "Simultaneous Perturbation Stochastic Approximation". In: *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control* (2003), pp. 176–207.

[8] Philipp Hennig and Christian J. Schuler. "Entropy Search for Information-Efficient Global Optimization". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 1809–1837.

[9] K. J. Åström and B. Wittenmark. *Adaptive Control*. 2nd. Dover, 2008.

[10] M. J. Grimble. "Implicit and explicit LQG self-tuning controllers". In: *Automatica* 20.5 (1984), pp. 661–669.

[11] D. W. Clarke, P. P. Kanjilal, and C. Mohtadi. "A generalized LQG approach to self-tuning control – Part I. Aspects of design". In: *International Journal of Control* 41.6 (1985), pp. 1509–1523.

[12] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. "Safe Exploration for Active Learning with Gaussian Processes". In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. 2015.

[13] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement Learning in Robotics: A Survey". In: *The International Journal of Robotics Research* (2013).

[14] John W Roberts, Ian R Manchester, and Russ Tedrake. "Feedback Controller Parameterizations for Reinforcement Learning". In: *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning*. 2011, pp. 310–317.

[15] Jan Peters and Stefan Schaal. "Natural Actor-Critic". In: *Neurocomputing* 71.7 (2008), pp. 1180–1190.

[16] Angela P. Schoellig and Raffaello D'Andrea. "Optimization-based iterative learning control for trajectory tracking". In: *Proc. of the European Control Conference*. Budapest, Hungary, Aug. 2009, pp. 1505–1510.

[17] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. "Online least-squares policy iteration for reinforcement learning control". In: *Proc. of the American Control Conference*. Baltimore, MD, USA, July 2010, pp. 486–491.

[18] Seul Jung and Sung Su Kim. "Control Experiment of a Wheel-Driven Mobile Inverted Pendulum Using Neural Network". In: *IEEE Transactions on Control Systems Technology* 16.2 (Mar. 2008), pp. 297–303.

[19] Stefan Schaal and Christopher G. Atkeson. "Learning Control in Robotics". In: *IEEE Robotics Automation Magazine* 17.2 (June 2010), pp. 20–29.

[20] Brian D.O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, 1990.

[21] Rudolf Emil Kalman. "When is a linear control system optimal?" In: *Journal of Fluids Engineering* 86.1 (1964), pp. 51–60.

[22] Jacques L. Willems and Herman van de Voorde. "The return difference for discrete-time optimal feedback systems". In: *Automatica* 14.5 (1978), pp. 511 –513. ISSN: 0005-1098.

[23] He Kong, Graham Goodwin, and Maria Seron. "A revisit to inverse optimality of linear systems". In: *International Journal of Control* 85.10 (2012), pp. 1506–1514.

[24] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[25] Daniel James Lizotte. "Practical Bayesian optimization". PhD thesis. Edmonton, Alta., Canada, 2008.

[26] Michael A Osborne, Roman Garnett, and Stephen J Roberts. "Gaussian processes for global optimization". In: *3rd International Conference on Learning and Intelligent Optimization*. 2009, pp. 1–15.

[27] Carl Edward Rasmussen. *Gaussian processes for machine learning*. Citeseer, 2006.

[28] Philipp Hennig. *Animating Samples from Gaussian Distributions*. Technical Report 8. Spemannstraße, 72076 Tübingen, Germany: Max Planck Institute for Intelligent Systems, 2013.

[29] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. "Gaussian process optimization in the bandit setting: No regret and experimental design". In: *International Conference on Machine Learning*. 2010.

[30] Peter Auer. "Using confidence bounds for exploitation-exploration trade-offs". In: *The Journal of Machine Learning Research* 3 (2003), pp. 397–422.

[31] Holger Kaden. "Pole balancing with Apollo". Diploma Thesis. 2014.

[32] Stefan Schaal. *The SL simulation and real-time control software package*. Tech. rep. 2009. URL: http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf.

[33] Ludovic Righetti, Mrinal Kalakrishnan, Peter Pastor, Jonathan Binney, Jonathan Kelly, Randolph C Voorhies, Gaurav S Sukhatme, and Stefan Schaal. "An autonomous manipulation system based on force control and optimization". In: *Autonomous Robots* 36.1-2 (2014), pp. 11–30.

[34] Stefan Schaal. "Learning from demonstration". In: *Advances in Neural Information Processing Systems* (1997), pp. 1040–1046.

[35] Katsuhisa Furuta, M Yamakita, and S Kobayashi. "Swing-up control of inverted pendulum using pseudo-state feedback". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 206.4 (1992), pp. 263–269.

[36] Karl Johan Åström and Katsuhisa Furuta. "Swinging up a pendulum by energy control". In: *Automatica* 36.2 (2000), pp. 287–295.

[37] Markus Hehn and Raffaello D'Andrea. "A flying inverted pendulum". In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 763–770.

[38] Christopher G. Atkeson and Stefan Schaal. "Robot learning from demonstration". In: *International Conference on Machine Learning*. Vol. 97. 1997, pp. 12–20.

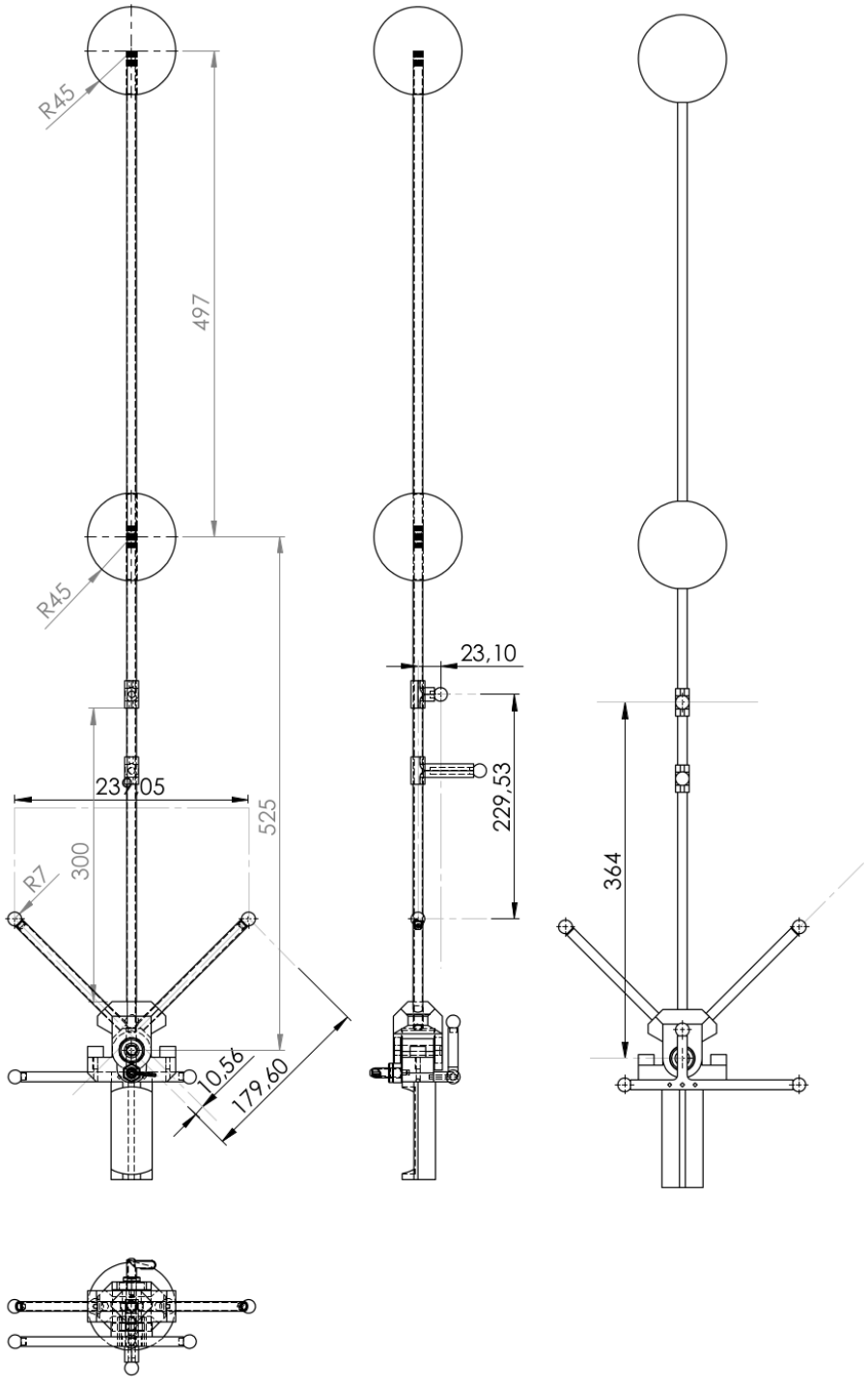
[39] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. "Safe Controller Optimization for Quadrotors with Gaussian Processes". In: *IEEE International Conference on Robotics and Automation*. submitted. 2016.

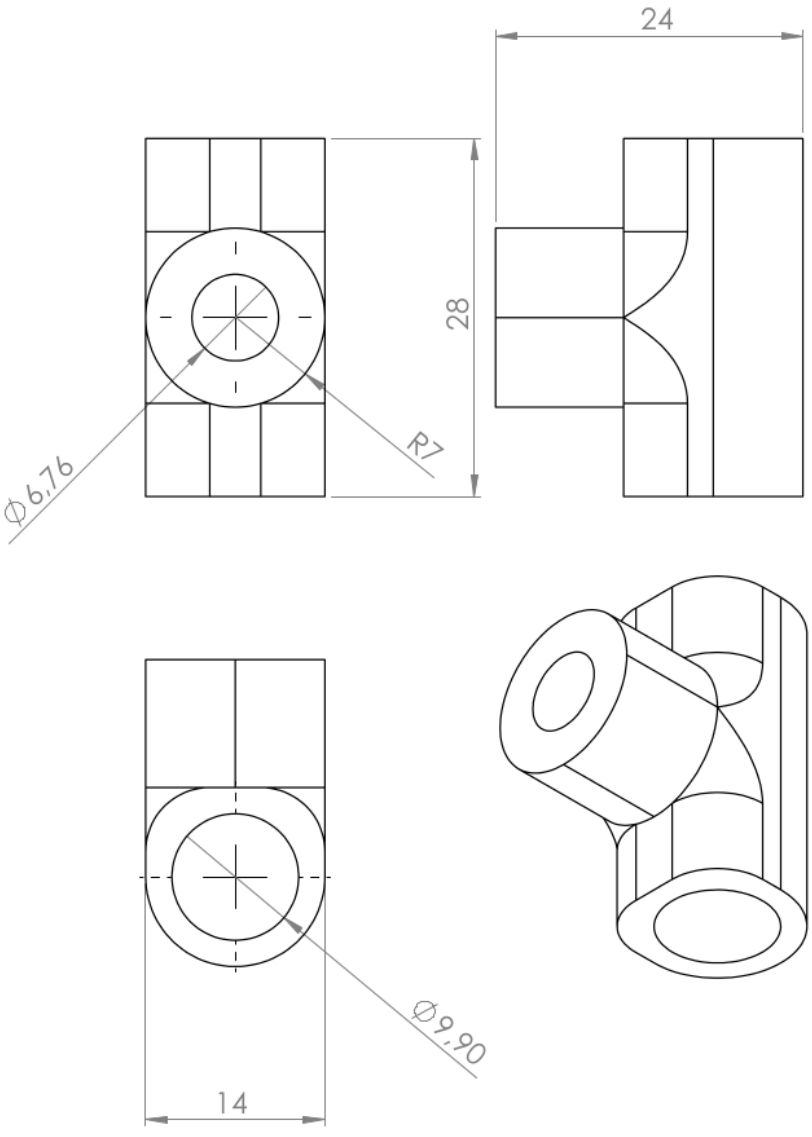
[40] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. "Safe Exploration for Optimization with Gaussian Processes". In: *The 32nd International Conference on Machine Learning*. 2015, pp. 997–1005.

[41] Felix Berkenkamp and Angela P. Schoellig. “Safe and robust learning control with Gaussian processes”. In: *European Control Conference*. 2015.

Appendix A

Pole design




24
28
Ø6,76
R7
Ø9,90
14