Active Exploration for Robot Parameter Selection in Episodic Reinforcement Learning

Oliver Kroemer and Jan Peters
Max Planck Institute for Biological Cybernetics
38 Spemannstr. Tuebingen, Germany, 72012
email: {oliverkro, jan.peters}@tuebingen.mpg.de

Special session on **Active Reinforcement Learning**
Organizers: Lucian Buşoniu, Damien Ernst, Robert Babuška
SSCI Symposium: Adaptive Dynamic Programming and Reinforcement Learning

***Abstract*—As the complexity of robots and other autonomous systems increases, it becomes more important that these systems can adapt and optimize their settings actively. However, such optimization is rarely trivial. Sampling from the system is often expensive in terms of time and other costs, and excessive sampling should therefore be avoided. The parameter space is also usually continuous and multi-dimensional. Given the inherent exploration-exploitation dilemma of the problem, we propose treating it as an episodic reinforcment learning problem. In this reinforcement learning framework, the policy is defined by the system's parameters and the rewards are given by the system's performance. The rewards accumulate during each episode of a task. In this paper, we present a method for efficiently sampling and optimizing in continuous multi-dimensional spaces. The approach is based on Gaussian process regression, which can represent continuous non-linear mappings from parameters to system performance. We employ an upper confidence bound policy, which explicitly manages the trade-off between exploration and exploitation. Unlike many other policies for this kind of problem, we do not rely on a discretization of the action space. The presented method was evaluated on a real robot. The robot had to learn grasping parameters in order to adapt its grasping execution to different objects. The proposed method was also tested on a more general gain tuning problem. The results of the experiments show that the presented method can quickly determine suitable parameters and is applicable to real online learning applications.**

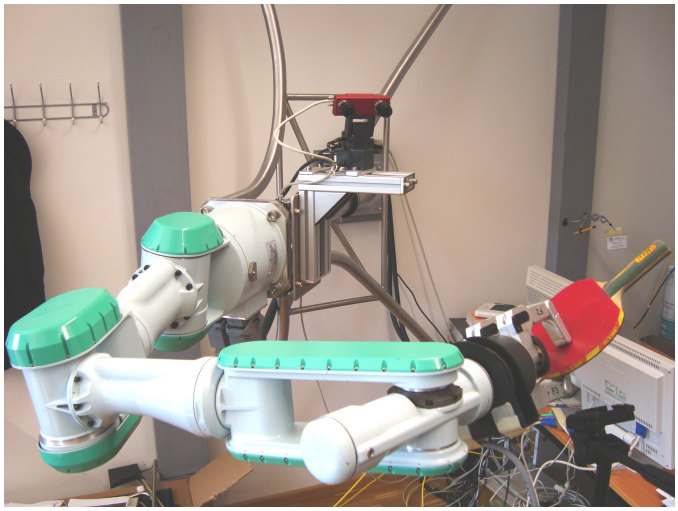

Figure 1. The robot consists of a Barrett hand, a Mitsubishi PA-10 arm, a pan-tilt unit, and a Videre stereo camera. The robot has just grasped a table tennis racket and lifted it from its stand, resulting in a successful grasp attempt.

I. Introduction

Parameter tuning represents a fundamental challenge in robotics and autonomous systems. In order to perform optimally, both task parameters and low-level control parameters must be selected correctly. The number of parameters, and their effect on performance, depends on the robot and the task it is performing. Robots are becoming more complex, and need to function autonomously in changing task environments. Therefore, there is an increasing need for robots to adapt their parameters autonomously. In order to achieve this goal, the robot must evaluate different parameter settings to determine their effects on performance. The robot must then use these experiences to optimize the parameters.

Given the multi-dimensional nature of the parameter tuning problems, applying a grid search to the entire parameter space will be highly inefficient due to the "curse of dimensionality" [5]. Instead, the space needs to be actively explored for optimal settings. Active exploration involves using the current knowledge of the system to select the next parameter setting to evaluate. The goal is not to explore the entire parameter space, but rather to focus on regions that lead to good performance. Therefore, the searching system must balance exploring new regions and exploiting its current knowledge of high-performance regions.

The autonomous tuning of parameters can be modeled as an episodic reinforcement learning task. Within each episode, the robot performs a task according to a parametrized policy $\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})$, where $\boldsymbol{a}$ is the action performed given that we are in state $\boldsymbol{s}$, and $\boldsymbol{\theta}$ is the vector of parameters. As the robot performs the task, it receives rewards $r(\boldsymbol{s},\boldsymbol{a})$ at each time step, which indicate its performance. The sum of these rewards over an episode is known as the return $R = \sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ where T is the finite horizon length of an episode. By changing the parameters $\boldsymbol{\theta}$, we create new policies π that will obtain different returns. The search through the parameter space should therefore try to maximize the return.

For discrete and finite sets of parameter configurations,

strategies that always evaluate the setting with the maximal upper confidence bound of the return have been highly successful [2, 3, 19]. The current methods designed to solve these problems in continuous spaces are largely based on discretizing the space [4, 9]. For high-dimensional domains, such as parameter tuning, any discrete segmenting will be affected by the "curse of dimensionality" [5] and hence scale badly. The hard segmentation will also result in unnatural borders and the use of prior knowledge becomes unnecessarily difficult. Hence, we propose a sample-based approach in this paper.

The method suggested in this paper comprises (i) a continuous approximation of the return's upper confidence bound using Gaussian process regression, and (ii) a technique for finding maxima of this approximation. The idea of using function approximation in this context was inspired by the work of [1]. Determining the maximum of the function is not trivial as it is nonlinear and has multiple local maxima. However, a mean-shift inspired algorithm can be used to find the local maxima of the function approximation, which represent suitable candidates for the next parameter setting to evaluate.

In Section II, we explain the proposed algorithm and discuss its properties and implementation. In the subsequent Section III, we evaluate the proposed algorithm through a series of experiments.

II. Upper Confidence Bound Policy Search for Episodic Reinforcement Learning

In this section, we present a method for optimizing a policy, over a series of episodes, using an upper confidence bound strategy. The first steps in developing the method are to model the expected returns for our policies (Section II-A) and determine a procedure for selecting the next policy to evaluate (Section II-B). In Sections II-C and II-D, we give information on the implementation of the method and show that the system is guaranteed to select a local maximum in every episode.

A. Gaussian Process Regression of Merit Function

We begin by modeling the expected return, and its upper confidence bound, as a function of the continuous policy parameters. An approach that satisfies these requirements is Gaussian process regression (GPR), see [15], which is a kernel-based non-parametric method. GPR also incorporates a prior, which keeps the mean and variance bounded in regions of no data. We employ standard Gaussian kernels of the form $k(\boldsymbol{x}, \boldsymbol{z}) = \sigma_a^2 \exp(-0.5(\boldsymbol{x}-\boldsymbol{z})^T \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{z}))$ where $\boldsymbol{W}$ is a diagonal matrix of kernel widths, and $\boldsymbol{x}$ and $\boldsymbol{z}$ define the parameters of two policies. The hyperparameter σ_a affects how quickly the search converges, with a greater value leading to more exploration, and sets the amplitude $k(\boldsymbol{x}, \boldsymbol{x}) = \sigma_a^2$.

The search strategy will actively select the next policy to attempt using the merit function defined as $M(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sigma(\boldsymbol{x})$, where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are the expected return and its standard deviation for the parameter configuration $\boldsymbol{x}$ respectively. This value is sometimes also called the upper confidence bound, but we introduce the term "merit" to avoid any ambiguities in meanings. The GPR model is then

$$\mu(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{Y})^T \left(\boldsymbol{K} + \sigma_s^2 \boldsymbol{I}\right)^{-1} \boldsymbol{t},$$
$$\sigma(\boldsymbol{x}) = \sqrt{v(\boldsymbol{x})},$$

where $v(\boldsymbol{x})$ is the variance of the process $v(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{Y})^T \left(\boldsymbol{K} + \sigma_s^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{Y})$, matrix $[\boldsymbol{K}]_{i,j} = k(\boldsymbol{y}_i, \boldsymbol{y}_j)$ is the Gram matrix, the kernel vector decomposes as $[\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{Y})]_j = k(\boldsymbol{x}, \boldsymbol{y}_j)$, and the N previous data points are stored in $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$ with corresponding returns $\boldsymbol{t} = [R_1, ..., R_N]^T$ [15]. The hyperparameter σ_s^2 regularizes the result. Both the mean and variance equations can be rewritten in a simpler form as the weighted sum of Gaussians, giving

$$\mu(\boldsymbol{x}) = \sum_{j=1}^{N} k\left(\boldsymbol{x}, \boldsymbol{y}_j\right) \alpha_j,$$
$$v(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i=1}^{N} \sum_{j=1}^{N} k'\left(\boldsymbol{x}, 0.5\left(\boldsymbol{y}_i + \boldsymbol{y}_j\right)\right) \gamma_{ij},$$

where $k'(\boldsymbol{x}, \boldsymbol{y}) = \sigma_a^2 \exp(-(\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{y}))$, and the constants are defined as $\alpha_j = [(\boldsymbol{K} + \sigma_s^2 \boldsymbol{I})^{-1} \boldsymbol{t}]_j$ and $\gamma_{ij} = [(\boldsymbol{K} + \sigma_s^2 \boldsymbol{I})^{-1}]_{i,j} \exp(-0.25(\boldsymbol{y}_i - \boldsymbol{y}_j)^T \boldsymbol{W}(\boldsymbol{y}_i - \boldsymbol{y}_j))$.

The variance indicates the system's uncertainty regarding the expected return, and not the variation of individual returns about the mean. A similar merit function has previously been employed for multi-armed bandits in metric spaces, wherein GPR was used to share knowledge between discrete bandits [18].

Having chosen to use an upper confidence bound framework and a GPR merit model, the implementation of the policy search has to be adapted to the merit function.

Algorithm 1 Parameter Tuning

Initialize: Store the N initial points in $\boldsymbol{Y}$ and $\boldsymbol{t}$

Loop:
```
Calculate coefficients α and γ
M_best = 0
for j = 1 to N
    x_o = y_j
    while not converged
        Calculate update step s
        x_{n+1} = s + x_n
    end
    if M(x) > M_best
        x_best = x_n
        M_best = M(x_best)
    end
end
Attempt and evaluate x_best
Store results in y' and t'
N = N + 1
```

B. Upper Confidence Bound Strategy for Selecting Next Policy

Given a model of the merit function, the system requires a suitable method for determining the policy with the highest merit from the infinite available ones. The merit function will most likely not be concave and will contain an unknown number of maxima with varying magnitudes [15].

Determining the global maximum of the merit function analytically is usually intractable [15]. However, numerically, we can determine a set of locally optimal points. This set of points will contain many of the maxima of the merit function, especially near the previous data points. Given the set of local maxima, the merit of each candidate is evaluated and the robot executes the policy with the highest merit.

The method for finding the local maxima was inspired by mean-shift [7], which is commonly used for both mode detection of kernel densities and clustering. Mean-shift converges onto a local maximum of a given point by iteratively applying

$$\boldsymbol{x}_{n+1} = \frac{\sum_{j=1}^{N} \boldsymbol{y}_j k\left(\boldsymbol{x}_n, \boldsymbol{y}_j\right)}{\sum_{j=1}^{N} k\left(\boldsymbol{x}_n, \boldsymbol{y}_j\right)}, \tag{1}$$

where $k(\boldsymbol{x}_n, \boldsymbol{y}_j)$ is the kernel function, and $\boldsymbol{y}_j$ are the N previously tested maxima candidates as before. The monotonic convergence via a smooth trajectory can be proven for mean-shift [7]. To find all of the local maxima, mean-shift initializes the update sequence with all previous data points. The global maximum is then determined from the set of local maxima, which is guaranteed to include the global maximum [12].

However, mean-shift is limited to kernel densities and is not directly applicable for regression, because the α_j and $\gamma_{i,j}$ weights are not always positive [7]. In particular, the standard update rule (1) cannot be used, nor can we guarantee that the global maximum will be amongst the detected local maxima. However, global maxima often do have a positively weighted point in their proximity, which would include them in the set of detected local maxima.

Due to Eq. (1) not being applicable to our regression framework, a new update step is required, which would monotonically converge upon the local maximum of our merit function.

C. Local Maxima Detection for Gaussian Process Regression

Given the model in Section II-A, the merit function now takes the form $M\left(\boldsymbol{x}\right) = \sum_{j=1}^{N} k(\boldsymbol{x}, \boldsymbol{y}_j)\alpha_j + \sqrt{k\left(\boldsymbol{x}, \boldsymbol{x}\right) - \sum_{i=1}^{N}\sum_{j=1}^{N} k'\left(\boldsymbol{x}, 0.5\left(\boldsymbol{y}_i + \boldsymbol{y}_j\right)\right)\gamma_{ij}}$. To use the policy described in Section II-B with this merit function, one requires a monotonically converging update rule, similar to that of mean-shift. To determine the local maxima of the merit function, we propose the iterative update rule

$$\boldsymbol{x}_{n+1} = \frac{\partial_x \mu + \partial_x \sigma}{q\left(\mu\right) + \frac{q(v)}{\sqrt{p(v)}}} + \boldsymbol{x}_n = \boldsymbol{s} + \boldsymbol{x}_n, \tag{2}$$

where $\partial_x \mu = \sum_{j=1}^{N} \boldsymbol{W}\left(\boldsymbol{y}_j - \boldsymbol{x}_n\right) k\left(\boldsymbol{x}_n, \boldsymbol{y}_j\right)\alpha_j$ and

$$\partial_x \sigma = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{2}{\sigma}\gamma_{ij}\boldsymbol{W}\left(\frac{\boldsymbol{y}_i + \boldsymbol{y}_j}{2} - \boldsymbol{x}_n\right) k'\left(\boldsymbol{x}, \frac{\boldsymbol{y}_i + \boldsymbol{y}_j}{2}\right).$$

The function $q(\cdot)$ returns a local upper bound on the absolute second derivative of the input within the $\boldsymbol{x}_n$ to $\boldsymbol{x}_{n+1}$ range. Similarly, $p(\cdot)$ returns a local lower bound on the absolute value of the input.

The update step described in Eq. (2) can be viewed as the current gradient of the merit function, divided by a local upper bound of the second derivative. This form of update rule displays the desired convergence qualities, as shown in Section II-D. The rule is only applicable because the Gaussian kernels have bounded derivatives resulting in finite $q\left(\mu\right)$ and $q\left(v\right)$, and any real system will have a positive variance giving a real non-zero $\sqrt{p\left(v\right)}$.

To calculate the local upper and lower bounds, we first define a region of possible $\boldsymbol{x}_{n+1}$ values to consider. Therefore, we introduce a maximum step size $m > 0$, where steps with larger magnitudes must be truncated; i.e., $\|x_{n+1} - x_n\| \leq m$. Having defined a local neighborhood, $q\left(\mu\right)$, $q\left(v\right)$, and $p\left(v\right)$ need to be evaluated.

In Section II-A, μ and v were represented as the linear weighted sums of Gaussians. Given a linear sum, the rules of superposition can be applied to evaluate $q\left(\mu\right)$, $q\left(v\right)$, and $p\left(v\right)$. Thus, the upper bound of a function in the region is given by the sum of the local upper bounds of each Gaussian, i.e.,

$$q_m\left(\sum\nolimits_{j=1}^{N} k\left(\boldsymbol{x}, \boldsymbol{y}_j\right)\alpha_j\right) \leq \sum\nolimits_{j=1}^{N} q_m\left(k\left(\boldsymbol{x}, \boldsymbol{y}_j\right)\alpha_j\right).$$

As Gaussians monotonically tend to zero with increasing distance from their mean, determining an upper bound value for them individually is trivial. In the cases of $q\left(\mu\right)$ and $q\left(v\right)$, the magnitudes of the second derivatives can be bounded by a Gaussian; i.e.,

$$\|\partial_x^2 k\left(\boldsymbol{x}, \boldsymbol{y}_j\right)\| < \sigma_a^2 \exp\left(-6^{-1}(\boldsymbol{x} - \boldsymbol{y}_j)^T \boldsymbol{W}(\boldsymbol{x} - \boldsymbol{y}_j)\right),$$

which can then be used to determine the local upper bound.

When working in multiple dimensions, it is advisable to first rescale the space of data points, such that the weight $\boldsymbol{W}$ is the identity matrix $\boldsymbol{I}$. In this manner, the Gaussians become isotropic, and the magnitude is only a function of the displacement from its center $\boldsymbol{y}_j$. Finding an upperbound for this form is straightforward.

We have thus defined an update step and its implementation, which can be used to detect the modes of a Gaussian process in a regression framework. The final algorithm is of order $O(N^3)$, as are all exact GPR methods [6]. However, this complexity scales linearly with the number of dimensions, while discretization methods scale exponentially, making the proposed GPR method computationally simpler when the problem dimensionality is greater than three. The mode detection algorithm can be easily parallelized for efficient implementations on multiple computers or GPUs as an anytime algorithm. Different upper confidence intervals σ, including heteroscedastic ones, can be implemented by simply modeling them with a second non-zero-mean GPR and neglecting the standard variance terms.

This concludes the implementation details of our proposed method, which is outlined in Alg. 1. The following section gives an analysis of the mode detection method's behavior.

D. Convergence Analysis

Given the proposed update rule, Lyapunov's direct method can be used to show that it converges monotonically to stationary points. The underlying principle is that an increased lower bound on the merit reduces the set of possible configurations, and therefore a continually increasing merit leads to

convergence. The following 1D analysis will show that only an upper bound on the magnitude of the second derivative is required for a converging update rule near a maximum.

The increase in merit is given by $M(x_{n+1}) - M(x_n)$. Given an upper bound u of the second derivative between x_n and x_{n+1}, and the gradient $g = \partial_x M(x_n)$, the gradient in the region can be linearly bounded as

$$g - \|x - x_n\| u \leq \partial_x M(x) \leq g + \|x - x_n\| u.$$

Considering the case where $g \geq 0$ and therefore $x_{n+1} \geq x_n$, the change in merit is lower bounded by

$$\begin{aligned} M(x_{n+1}) - M(x_n) &= \int_{x_n}^{x_{n+1}} \partial_x M(x)\, dx \\ &\geq \int_{x_n}^{x_{n+1}} g - (x - x_n)\, u\, dx. \end{aligned}$$

This integral is a maximum when the integrand reaches zero; i.e, $g - (x_{n+1} - x_n)u = 0$. This limit results in a shift of the form $s = x_{n+1} - x_n = u^{-1}g$, as was proposed in Eq. (2). The same update rule can be found by using a negative gradient and updating x in the negative direction. The merit has thus been shown to always increase, unless the local gradient is zero or u is infinite. A zero gradient indicates that a stationary point has been found, and variable u is finite for any practical GPR. The update rule guarantees that the gradient cannot shift sign within the update step, and thus ensures that the system will not overshoot nor oscillate about the stationary point. The update rule $x_{n+1} = u^{-1}g + x_n$ therefore guarantees that the algorithm monotonically converges on the local stationary point.

III. Experiments

In this section, we investigate the proposed method through a series of experiments. In the first experiment, we focus on a 1D benchmark problem. Using this benchmark problem, we compare the proposed method to other approaches.

The second and third experiments show applications of the proposed system. The second experiment demonstrates how the method can be used to tune the low level control of a robot in simulation. The system must actively tune parameters is an eight dimensional space. The final experiment investigates optimizing a grasping action for an object, and was performed on a real robot. The grasping problem is posed as a six dimensional problem. Therefore, both of the application experiments test the system if multi-dimensional spaces.

A. Immediate Reward Benchmark Experiment

In this experiment, we focus on a 1D benchmark example with immediate rewards. In this manner, we can compare our proposed method to standard methods for the continuum-armed bandits problem, a framework closely related to episodic reinforcement learning.

Along with our proposed method, we also tested UCBC [4], CAB1 [9], and Zooming [10], as described below. The tested methods were compared on the same set of 100 randomly generated 7^{th} order spline reward functions. The rewards were superimposed with uniform noise of width 0.1, but restricted to a range of $[0, 1]$. The space of parameters was also restricted to a range between 0 and 1. For a fair comparison, evaluating the initialization points counted towards the total number of attempts. Additionally, none of the policies were informed of the length of the experiment in advance, and efforts were made to tune the parameters of each policy to achieve high mean rewards.

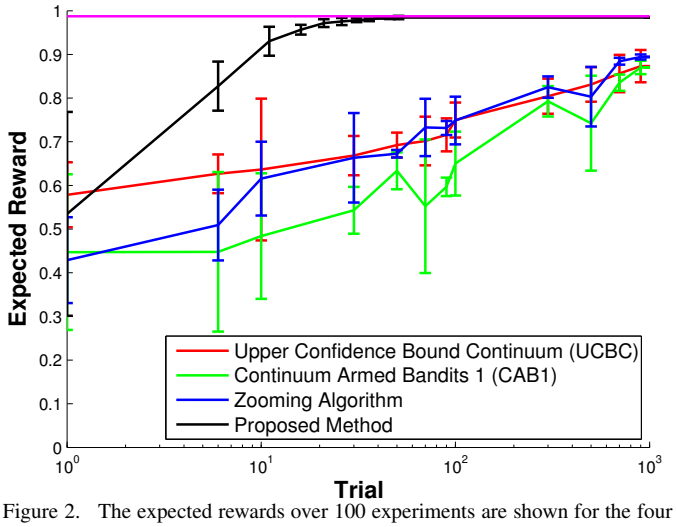

Figure 2. The expected rewards over 100 experiments are shown for the four compared methods. The results were filtered for clarity. Due to the differences in experiment lengths, the x-axis uses a logarithmic scale. The magenta line represents the maximum expected reward given the noise. The large drops in the green and blue plots indicate the resetting points of the respective policies. The error bars represent one standard deviation.

1) Compared Methods: A key issue for any policy that uses discretizations is selecting the number of discrete bandits to use. Employing a coarser structure will lead to faster convergence, but the expected rewards upon convergence are also further from the optimal.

The UCBC policy divides the bandits into regular intervals and treats each interval as a bandit in a discrete upper confidence bound policy. After choosing an interval, a uniform distribution over the region selects the bandit to attempt. The number of intervals sets the coarseness of the system, and was tuned to 10.

Instead of using entire intervals, the CAB1 policy selects specific bandits at uniform grid points. A discrete policy is then applied to this subset of bandits, for which the original paper [9] proposed UCB1 [3]. The discretization trade-off is dealt with by resetting the system at fixed intervals with larger numbers of bandits, thus ensuring that the grid of bandits become denser as the experiment continues.

The Zooming algorithm also uses a grid structure to discretize the bandits. In contrast to CAB1, the grid is not uniform and additional bandits are introduced in high rewarding regions to increase the density in these regions. A discrete policy is then applied to the set of active bandits. Similar to CAB1, the Zooming algorithm updates its grid sizes over time.

Our proposed method was initialized with 4 equispaced points before using the standard methodology, with a kernel width of $\boldsymbol{W}^{-1} = 0.01$. All four methods were initially run for 50 trials, as shown in Fig. 2. The proposed method was subsequently run for an additional 5 trials to show that it

had successfully converged, while the other methods were extended by 950 trials to demonstrate their convergence.

2) Results: The expected rewards for the four UCB policies during the experiment can be seen in Fig. 2, where our proposed method is displayed in black. The mean rewards over the initial 50 trials are:

UCBC	CAB1	Zoom	Proposed
0.6419	0.4987	0.6065	**0.9122**

Using a fixed set of bandits gave UCBC an advantage over CAB1 and Zoom during the first 50 trials. However, Zooming was the most successful of the competitors over the 1000 trials at achieving high rewards, due to its ability to adapt its grid to the reward function. Due to its continuous representation, the proposed GP-based method was the most adaptive approach and was capable of converging on maximal points of the reward functions.

The low dimensionality of this benchmark scenario puts the proposed GPR-based method at a disadvantage in terms of computational complexity. Implementations of GPR for large data sets do exist (e.g., Sparse GP [17]), which reduce this complexity. The loss of accuracy incurred by such implementations is comparable to the accuracy limits inherent to discretization methods, making these methods suitable alternatives to standard GPR. Ultimately, our proposed method was able to find suitable parameter configurations within a reasonable number of trials, and the standard GPR method was well within its limits.

B. Controller Parameter Tuning Evaluation

In this experiment, we demonstrate how the proposed method can be used to tune the parameters of a robot's controller. The robot is a simulated four degree of freedom arm, consisting of a ball-and-socket shoulder joint and an elbow joint. The simulation is based on the Mitsubishi PA-10 shown in Fig. 1.

1) Controller Experiment Setup: The robot's task involves following a 2Hz sinusoidal trajectory in the joint space. In order to ensure that the episodes are independent, the robot comes to rest at a home position between each trial. At each time step during the task, the robot receives a cost, or negative reward, proportional to its tracking error. However, the robot should avoid using excessively high gains and incurs a cost for using larger gains. The negative returns are mapped to the space $[0, 1]$ using an exponential function.

The robot is controlled using proportional-derivative (PD) controllers for each of the four joints. These controllers represent the parametrized policy. The robot must therefore tune eight parameters; i.e. proportional and derivative gains for each joint. Approaches based on discretizing this 8D space suffer from the "curse of dimensionally" [5].

The system is initialized by evaluating 30 initial parameter configurations. These initial configurations are distributed as small clusters in different regions of the parameter space. In this manner, we can seed the search in multiple regions. The maximal return that the robot can achieve in practice is unknown. Achieving the highest return of one is impossible, as it would require perfect tracking with the gains set to zero. We therefore set the hyperparameter $\sigma_a^2 = 0.5$ to avoid excessive exploration of low-performance regions.

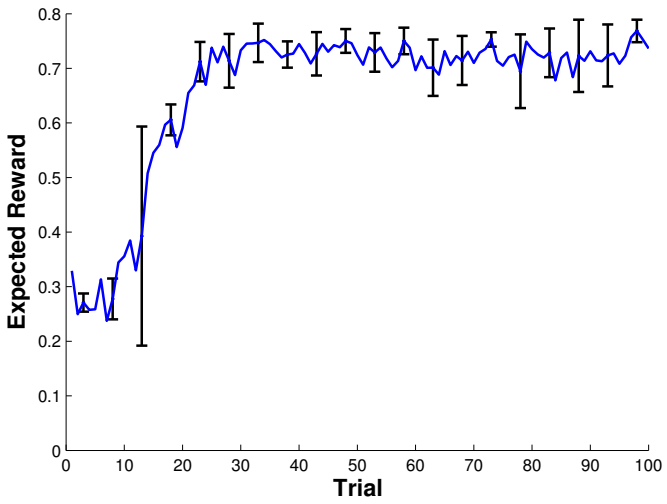

Figure 3. The expected rewards for each trial, or episode, during the PD controller tuning experiment. The rewards are averaged over the three experiments. The error bars represent a range of one standard deviation in each direction. The initialization trials are not shown. The highest reward obtained during initialization was only 0.393.

The experiment was run three times, with 100 episodes each. The highest rewards acquired during the initialization phase were 0.393, 0.367, and 0.379 for the three experiments respectively. The results of the experiments are shown in Fig. 3.

2) Controller Experiment Results: The results show that the automatic tuning method consistently found regions of high-performance parameters. Whether the global maximum was found is unfortunately not known. However, the final returns are approximately double the return acquired by the best initialization point, and represent a significant improvement in performance.

The majority of the performance improvement occurred during the first 30 trials. Although the rewards seem to have converged towards the end of the experiment, the system is actually continuing its search. The rewards remain high because the system is focusing on parameter configurations near the high-performance regions, and avoiding the low-performance regions. Therefore, the system is successfully executing the desired active exploration behavior for parameter tuning. This experiment also shows that the proposed method is applicable to multi-dimensional problems.

C. Robot Grasping Evaluation

In the previous section, we investigated how the proposed method can be used to tune the parameters of a low-level controller. In this section, we will demonstrate how the proposed approach can be used to tune task-level parameters. In particular, the robot will determine the parameters for successfully grasping an object.

Previous works on active learning for robot grasping have used K-nearest-neighbor [13] and classification tree [16] algorithms to determine areas where the model of grasping

points needs to be improved. As the focus was on gathering entire models of the object, the systems could rely on purely exploratory policies. However, the task of actively learning a good grasp of a given object is one of exploration vs. exploitation, wherein the robot needs to find and use good grasps without excessive exploration of the object.

1) Grasping Experiment Setup: The robot consists of a 7 degrees of freedom Mitsubishi PA-10 arm, a Barrett hand, a pan-tilt unit, and a Videre stereo camera, as shown in Fig. 1. Only stereo-vision and joint encoders are used as feedback in this experiment. The robot's task is to learn good grasps of a table tennis paddle through attempting and evaluating grasps, without the aid of a physics model.

The grasping action is defined by the six dimensional pose (3D position and 3D orientation) of the hand in the object's reference frame. During the experiment, the robot will be actively tuning these six parameters.

Each trial begins by computing the position and orientation of the object to be grasped. In this manner, the grasps can be defined relative to the object, and the object may be shifted between grasps. The stereo camera extracts the required information using the Early Cognitive Vision system of Pugeault [14] and Kraft et al. [11] with the pose estimation method by Detry et al. [8].

Once the object's pose is detected, our algorithm determines the grasp with the highest merit, which the robot subsequently executes. If it successfully grasps the object, the robot attempts to lift the object from the table, to ensure that the table is not applying additional support to the object. The result is evaluated and stored in the merit function for subsequent trials.

Although this trial structure is suitable for actively learning grasps, determining a good grasp in six dimensions without any prior knowledge is still an infeasible task. The system was therefore initialized with a search region defined by 25 demonstrated grasps. The width parameters $\boldsymbol{W}$ of the Gaussian kernel were optimized on these initial poses, while σ_s and σ_a were tuned using data from previous grasping experiments.

The return of an episode corresponds to the success of the grasp. Successful trials are given a reward depending on how little the fingers and object move while lifting the object, thereby encouraging more stable grasps. The values are restricted to the range zero to one, but are not deterministic due to errors in pose estimation and effects caused by the placement of the object. A reward of 0.7 or higher is generally suitable for standard pick and place tasks.

2) Grasping Experiment Results: The experiment was run until the robot had reliably converged upon a region, giving 55 trials. The experiment was performed five times to show the repeatability of the method. After terminating the experiment, the system had reached an overall success rate of 76.4% and a mean immediate reward of 0.701. These values would be expected to improve if the experiments were continued further.

The 25 imitation trials represented three distinct grasp areas: at the handle, at the edge of the paddle, and orthogonal to the plane of the paddle. This prior was intentionally chosen to contain only a few good robot grasps, to test for robustness. Despite the prior, the method converged upon a region with high rewards in each experiment. The majority of the successful grasps originate from the side of the paddle, even though only 20% of the imitation grasps recommended this region. No successful grasps were found in the region perpendicular to the paddle's plane. The handle had the most imitation data, as this is where a human would hold the object. This region yielded a few successful grasps, but of low reward due mainly to the hand's relatively large size.

Figure 4 shows the average rewards acquired over the course of the experiment and the exploration/exploitation trade-off. The system steadily shifts from exploration to exploitation, with the returns displaying a similar trend. A rapid change from exploration to exploitation could have caused premature convergence and a lower final reward. The uncertainty also steadily increases until trial 30, reflecting that small differences at the beginning of the experiment can have large effects on the system's overall development. However, once good grasps have been confidently determined, this uncertainty rapidly diminishes and increased exploitation confines the exploration to smaller regions.

The larger moves between attempted grasps, shown in Fig. 4(B), indicate shifting exploration between different grasp areas. All experiments began with trying the different regions. Some experiments failed to quickly ascertain high rewards, and continued to shift between the areas until trial 30, as can be seen by the large uncertainty bars. Beyond this trial, the system only displays decreasing amounts of shifting. Successes in a region increase exploitation, leading to a more localized search in the area. Some experiments initially succeeded at the handle, but the low rewards in this region made it unsuitable. Hence, exploration returned to finding higher rewarding areas. This funneling of shift sizes indicates that the system is converging onto smaller areas of higher expected gains. In this manner, the active policy search focuses on searching through high-performance regions.

IV. Conclusions

The episodic reinforcement learning method suggested in this paper was implemented for robot grasping and controller tuning, and consistently found good solutions. Accomplishing these multi-dimensional task was made possible by using a sample-based approach rather than a discretization of the continuous space. The method also performed well compared to standard approaches in a benchmark experiment.

Gaussian process regression was utilized by the algorithm to model the upper confidence bounds, which also allowed for prior knowledge to be easily incorporated into the system. By using local maxima, the policy can quickly converge onto higher rewarding regions. The usage of local maxima is feasible because of the mean-shift-inspired search technique described in this paper.

During the experiments, the system demonstrated the desired gradual shift from exploration to exploitation. The search regions also became more localized and refined, with a focus on regions of high-performance parameters. Although the focus of this paper has been on robot applications, the proposed method is suitable for a wide range of applications that require parameter tuning.

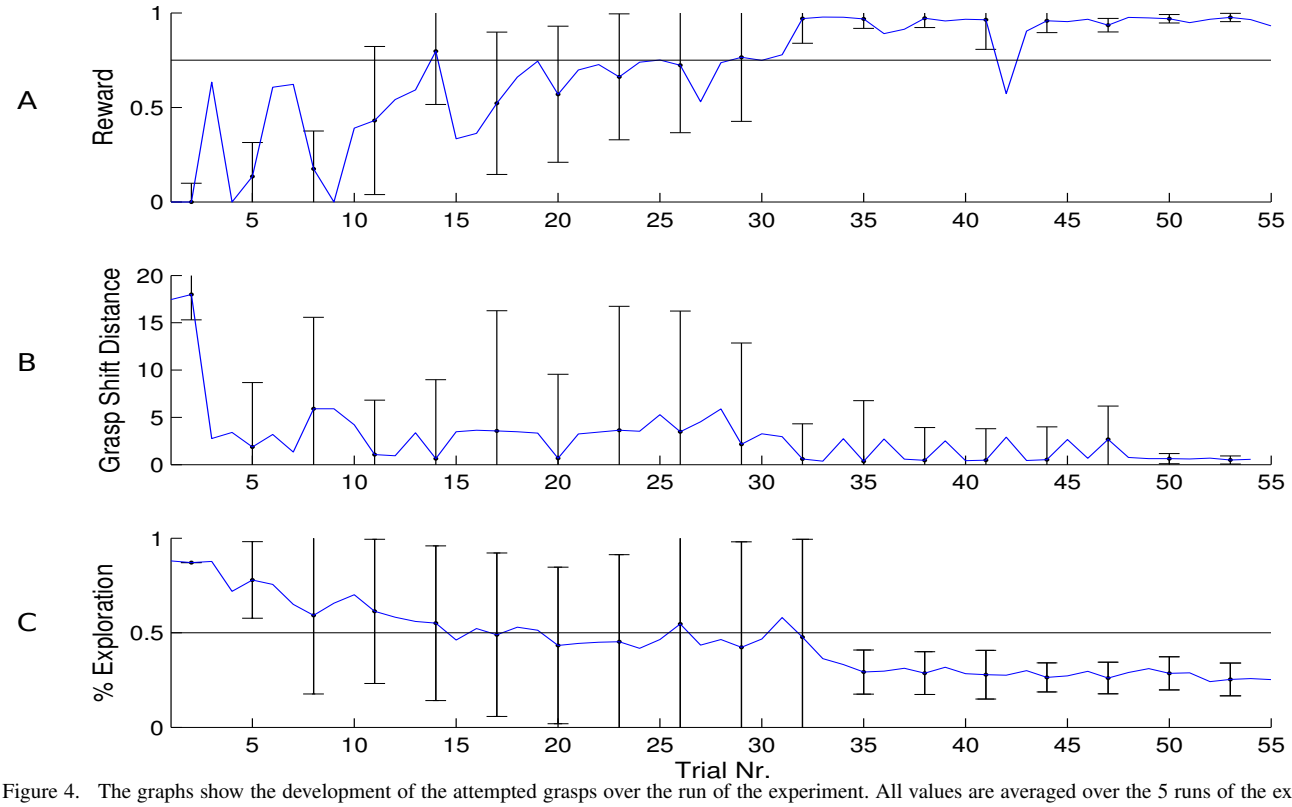

Figure 4. The graphs show the development of the attempted grasps over the run of the experiment. All values are averaged over the 5 runs of the experiment, with error bars of +/- two standard deviations. **Plot A** shows the immediate reward of the trials. The horizontal black line indicates the upper confidence bound of a point at infinity. **Plot B** shows the size in shift between subsequent grasps, with distances normalized by their respective width parameters. **Plot C** shows the percent of the selected grasp's merit induced by the confidence term. The horizontal black line indicates the 50% mark, below which grasps are usually classified as exploitative rather than exploratory.

References

[1] Rajeev Agrawal. The continuum-armed bandit problem, 1995.

[2] Peter Auer. Using upper confidence bounds for online learning. In *FOCS Proceedings*, 2000.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem, 2002.

[4] Peter Auer, Ronald Ortner, and Csaba Scepesvari. Improved rates for the stochastic continuum-armed bandit problem. In *COLT 2007 Proceedings*, 2007.

[5] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[7] D. Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. In *Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[8] R. Detry, N. Pugeault, and J. Piater. Probabilistic pose recovery using learned hierarchical object models. In *ICV Workshop*, 2008.

[9] Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *NIPS Proceedings*, 2004.

[10] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *ACM STOC Proceedings*, 2008.

[11] D. Kraft, N. Pugeault, E. Baeski, M. Popovic, D. Kragic, S. Kalkan, F. Woergoetter, and N. Krueger. Birth of the object: Detection of objectness and extraction of object shape through object action complexes. *International Journal of Humanoid Robotics*, pages 247–265, 2008.

[12] Ruben Martinez-Cantin. *Active Map Learning for Robots: Insights into Statistical Consistency*. PhD thesis, University of Zaragoza, 2008.

[13] A. Morales, E. Chinellato, A. H. Fagg, and A. P. Pobil. An active learning approach for assessing robot grasp reliability. In *IRS Proceedings*, 2004.

[14] N. Pugeault. *Early Cognitive Vision: Feedback Mechanisms for the Disambiguation of Early Visual Representation*. Vdm Verlag Dr. Mueller, 2008.

[15] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[16] M. Salganicoff, L.H. Ungar, and R. Bajcsy. Active learning for vision-based robot grasping. *Machine Learning*, 1996.

[17] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs, 2005.

[18] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995, 2009.

[19] Yizao Wang, Jean-Yves Audibert, and Remi Munos. Algorithms for infinitely many-armed bandits. In *NIPS Proceedings*, 2008.