Combining Active Learning and Reactive Control for Robot Grasping

O.B. Kroemer[c,**], R. Detry[d,*], J. Piater[d,*], J. Peters[c,1,*]

[a] *Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany*
[b] *Université de Liège, INTELSIG Lab, Department of Electrical Engineering and Computer Science, Belgium*

Abstract

Grasping an object is a task that inherently needs to be treated in a hybrid fashion. The system must decide both where and how to grasp the object. While selecting where to grasp requires learning about the object as a whole, the execution only needs to reactively adapt to the context close to the grasp's location. We propose a hierarchical controller that reflects the structure of these two sub-problems, and attempts to learn solutions that work for both. A hybrid architecture is employed by the controller to make use of various machine learning methods that can cope with the large amount of uncertainty inherent to the task. The controller's upper level selects where to grasp the object using a reinforcement learner, while the lower level comprises an imitation learner and a vision-based reactive controller to determine appropriate grasping motions. The resulting system is able to quickly learn good grasps of a novel object in an unstructured environment, by executing smooth reaching motions and preshaping the hand depending on the object's geometry. The system was evaluated both in simulation and on a real robot.

Keywords: robot grasping, reinforcement learning, reactive motion control, imitation learning

*Corresponding author
**Principal corresponding author
The project receives funding from the European Community's Seventh Framework Programme under grant agreement no. ICT- 248273 GeRT.
Email addresses: oliverkro@tuebingen.mpg.de (O.B. Kroemer), renaud.detry@ulg.ac.be (R. Detry), Justus.Piater@ulg.ac.be (J. Piater), jan.peters@tuebingen.mpg.de (J. Peters)

Robotics and Autonomous Systems 58(9), 1105-1116 (09 2010), doi:10.1016/j.robot.2010.06.001

Combining Active Learning and Reactive Control for Robot Grasping

O.B. Kroemer[c,**], R. Detry[d,*], J. Piater[d,*], J. Peters[c,1,*]

[c] *Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany*
[d] *Université de Liège, INTELSIG Lab, Department of Electrical Engineering and Computer Science, Belgium*

1. Introduction

Robots possess great potential for being employed in domestic environments, where they could perform various tasks such as tidying up rooms, taking out the garbage, or serving dinner. Although these chores are variations of a basic pick-and-place task, robots still struggle with them.

One of the key challenges for roboticists is the large variability inherent in the tasks and environments that a robot may encounter. Preparing a robot completely beforehand for all possible situations is probably impossible as it is prohibitively difficult to foresee all scenarios. Such a preparation is also inefficient, as only a few of the situations will be required by the robot. Due to these limitations, it is important to design robots that can adapt and learn from their own experiences.

Grasping an unknown object is an example of a task that is made particularly difficult by the large variety of objects (see Figure 2). Many approaches have been proposed for robot grasping. Early work [6, 24] found analytical solutions to the problem, but these approaches require precise information about the environment (e.g., external forces, surface properties) that may not be accessible. Supervised learning can be used to train robots how to recognize good grasping points [36], but requires a considerable initial input from a human supervisor. Active and reinforcement learning methods have focused on exploring the object to acquire complete affordance model [35, 27], but not on optimizing grasps. However, finding good grasp locations is only a part of the problem.

The robot grasping task can be decomposed into two problems: (*i*) deciding where to grasp the object, and (*ii*) determining how to perform the grasping movement. These two sub-problems are closely related and must be addressed together in order to perform a successful grasp. The choice of where to grasp an object sets the context for determining how to grasp it. However, the execution of the grasp ultimately determines whether the grasp location was well-chosen.

In this paper, we propose a hierarchical controller that reflects the structure of these two task components, as shown in Figure 1. The upper level decides where to grasp the object, and the lower level determines how to perform the grasping movements given the context of these grasp parameters and the scene. The upper level subsequently receives a reward based on the grasp execution, and takes this into consideration when selecting future grasps.

*Corresponding author
**Principal corresponding author
The project receives funding from the European Community's Seventh Framework Programme under grant agreement no. ICT- 248273 GeRT.
Email addresses: oliverkro@tuebingen.mpg.de (O.B. Kroemer), renaud.detry@ulg.ac.be (R. Detry), Justus.Piater@ulg.ac.be (J. Piater), jan.peters@tuebingen.mpg.de (J. Peters)

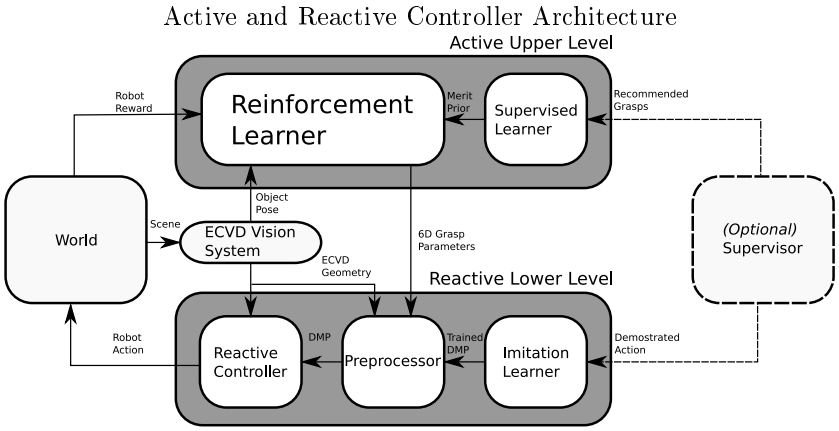


Figure 1: The controller architecture consists of a upper level based on reinforcement learning and a bottom level based on reactive control. Both levels are supported by supervised/imitation learning. The World and Supervisor are external elements of the system.

The system employs a hybrid architecture that uses reinforcement learning, imitation learning, and reactive control. The core of the upper level is a reinforcement learning approach that uses the successfulness of evaluated grasps to determine future grasps. It is crucial that its state-action space is low dimensional for faster convergence [42, 4], and that information from other sources (e.g., demonstrated grasps) can easily be incorporated. To reduce the action space, the reinforcement learner specifies a grasp as a six dimensional hand pose in the object's reference frame, and all remaining variables inherent to the grasping movements are handled by a lower level controller.

The lower level controller is responsible for action execution. A straightforward method of acquiring an arbitrary motion policy is by imitation learning. One approach to imitation learning is to transform a demonstrated trajectory into a standard dynamical systems motor primitive (DMP) [14, 37]. This policy is adapted, in a task specific manner, to the grasp parameters specified by the reinforcement learner. The resulting DMP is augmented by a reactive controller that takes the geometry of the object and scene into consideration. The resulting action is executed by the robot, which returns a corresponding reward to the upper level of the controller.

The complete hybrid controller is illustrated in Figure 1. It uses its own experiences to quickly converge on good grasping locations. The grasping motions are taught by demonstration and adapted to different grasp locations and the surrounding geometry. A key feature of this hybrid approach is that the reactive controller is incorporated in the reinforcement learner's action-reward feedback loop. Thus, the hybrid system will learn an appropriate grasping action together with a corresponding grasp location, and solve both of the sub-problems.

In the following sections, we discuss the proposed controller in a top-down manner. The active learner and the reactive bottom level of the controller are detailed in Sections 2 and 3 respectively. In Section 4, the system is evaluated both in simulation and on the robot platform shown in Figure 2.

2. High Level Active Learner

The high level controller chooses where on the object to apply the next grasp, and improves the grasp locations using the acquired data. The reinforcement learning approach is inspired by the grasp learning exhibited by infants [28, 29, 33], requiring relatively little prior knowledge and making few assumptions. Young infants have a grasp reflex that allows them to crudely grasp

objects [28]. They learn to improve their grasps through trial and error, allowing them to later be able to perform precision grips. The reactive controller of the hybrid system represents a vision-based grasp reflex. The initial grasps may be crude, but the learning system will adapt to the object and can learn to perform precision grasps.

To keep the number of assumptions low, we define the state as the object being grasped, and learn a model for each object. The robot's grasps are learned in the object's reference frame, allowing the object to be repositioned in the workspace. Similar to a young infant [28], learning to grasp an object is treated as context independent and only based on the task constraints it has encountered. Thus, if an object has always been presented as hanging on a string, both the robot and infant would initially not know that grasping it from below does not work when the object is on a table [28]. The robot will assign an expected reward to the grasp that reflects both situations and how often it has encountered each.

Another infant-like feature is that the robot has no vision-grasp mapping. Infants under nine months do not orientate their hands to the orientation of object parts [33]. The robot also does not assume that the geometry of a cup's handle will imply a certain orientation of the hand as appropriate. Instead, it will try different orientations and find one that is well-suited for it. Hence, several object properties do not need to be modeled explicitly, e.g., friction. Ultimately, the reinforcement learning approach is highly adaptive and is applicable to a wide range of situations.

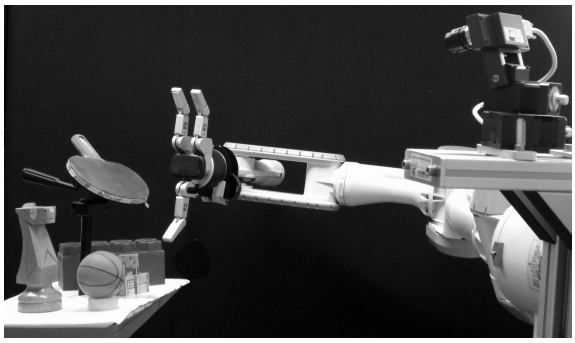


Figure 2: The robot used in our experiments and an example of a grasping task in a cluttered environment.

In contrast, supervised learning of grasps has focused on methods using internal models of the world [26, 20], or mappings between visual features of objects and grasps [36]. These approaches are more characteristic of adult human grasping, and thus require large amounts of prior information.

To converge quickly to high rewarding grasp locations, the system must balance the exploitation of good grasping points and the exploration of new, possibly better, ones. From a machine learning perspective, this selecting of grasps can be interpreted as a continuum-armed bandits problem [1].

The continuum-armed bandit problem is a generalization of the traditional n-armed bandit problem [42] where the agent must choose from a continuous range of locally dependent actions, instead of a finite number. Under this interpretation, the action is given by the grasp applied and the reward is a measure of the success of this grasp.

To date, most methods [3, 18] that solve the continuum-armed bandit problem are based on discretizing the space. For high-dimensional domains, such as robot grasping, any discrete segmenting will scale badly due to the "curse of dimensionality" [4]. The hard segmentation will result in unnatural borders and make the use of prior knowledge complicated. We propose a sample-based reinforcement learner that models the distribution of expected rewards over the continuous space of actions using Gaussian process regression (GPR) [32]. The proposed learner then searches for the most promising grasp to evaluate next, using a method inspired by Mean-shift [9]. The resulting policy is called Continuum Gaussian Bandits (CGB), and is outlined in Algorithm 1.

The following four sections detail the active learner and present the employed policy (Section 2.1), the modeling of the expected rewards (Section 2.2), how the learner selects the next

grasp (Sections 2.3), and then the method for implementing this selection on the reward model (Sections 2.4 and 2.5). Finally, Section 2.6 explains how supervised data can be incorporated into the active learner as prior knowledge.

2.1. Upper Confidence Bound Policy

Choosing where to grasp a novel object suffers from an exploration-exploitation problem. The traditional machine learning framework for studying this dilemma is the n-armed bandits problem, wherein an agent must repeatedly choose from a finite set of n possible actions to maximize the accumulated reward.

Among the more successful strategies [42] are upper confidence bound (UCB) policies. While there are different versions of UCB policies [42, 2], the principle idea is to assign each action two variables, i.e., the expected reward μ for taking that action, and a confidence bound $\pm\sigma$ indicating the range in which the actual mean reward is. Both μ and σ indicate how desirable executing the action is. A high expected reward μ is valuable in the sense of exploitation and receiving rewards, while a large confidence bound σ indicates an informative action that is good for exploration. Using the exploration variable σ leads to a more structured exploration than regular randomized policies (e.g., ϵ-greedy [42]). UCB policies also provide performance guarantees, and have an upperbound on the expected regret that scales only logarithmically with the number of trials [2].

The sum of the expected reward μ and the standard deviation σ indicates how desirable executing the action is overall. We call the value $\mu + \sigma$ the merit of an action. A UCB policy always selects the action for which this merit value is the greatest [2]. Intuitively, a UCB policy optimistically chooses the action which could be the best, and will thus only converge to an action when it knows that no other action could be better.

Adapting a UCB policy to the continuum-armed bandits requires a new approach that scales to the high dimensional spaces of grasping tasks. The first step towards realizing this approach is to create a sample-based model of the exploration σ and exploitation μ variables.

2.2. Expected Reward and Confidence Modeling with Gaussian Process Regression

Modeling the upper confidence bound for continuous actions requires the expected reward function and its standard deviation to be approximated. A well-suited approach that satisfies these requirements is Gaussian process regression (GPR) [32].

Rather than mapping inputs to specific output values, GPR returns a Gaussian distribution of the expected rewards. This Gaussian distribution is characterized by its mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$, where the standard deviation is a confidence bound on the expected reward. This technique is non-parametric, which implies that $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are functions that directly incorporate all previous samples. Non-parametric methods are very adaptable, and apply few constraints on the model. The GPR approach incorporates a prior that keeps the mean and variance bounded in regions without data. Unexplored regions will thus have a large confidence bound $\sigma(\mathbf{x})$ and small expected rewards $\mu(\mathbf{x})$. Sampling from these regions will shift $\mu(\mathbf{x})$ towards the actual expected reward at $\mathbf{x}$, but decrease the confidence bound $\sigma(\mathbf{x})$.

We employ the standard Gaussian kernels $k(\mathbf{x}, \mathbf{y}) = \sigma_a^2 \exp(-0.5(\mathbf{x} - \mathbf{y})^T \mathbf{W}(\mathbf{x} - \mathbf{y}))$ where $\mathbf{W}$ is a diagonal matrix of kernel widths. The parameter σ_a affects the convergence rate of the policy, as explained in Section 2.6.

For grasping, the vectors $\mathbf{x} \in \mathbb{R}^6$ and $\mathbf{y} \in \mathbb{R}^6$ each contain three position and three orientation parameters of grasps, which describe the final position of the hand in the object's reference frame. Working in the object's reference frame allows the object to be repositioned and reorientated in

the workspace without altering the grasp parameters. Additional grasp parameters are excluded to keep the number of parameters minimal, and thus allow for rapid learning. All of the other motion parameters are handled by the reactive low level controller, which modifies these parameters depending on the object and the scene, as well as the parameters in $\mathbf{x}$.

The proposed UCB policy will base its decisions on the merit function $M(\mathbf{x}) = \mu(\mathbf{x}) + \sigma(\mathbf{x})$, where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the expected reward and standard deviation at grasp $\mathbf{x}$ respectively. The standard GPR model [32] for the mean μ, variance σ^2, and standard deviation σ, are

$$\begin{aligned}\mu(\mathbf{x}) &= \mathbf{k}(\mathbf{x},\mathbf{Y})^T \left(\mathbf{K}+\sigma_s^2\mathbf{I}\right)^{-1}\mathbf{t},\\ \sigma(\mathbf{x}) &= \sqrt{\sigma^2(\mathbf{x})} = \sqrt{k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x},\mathbf{Y})^T\left(\mathbf{K}+\sigma_s^2\mathbf{I}\right)^{-1}\mathbf{k}(\mathbf{x},\mathbf{Y})},\end{aligned}$$

where $[\mathbf{K}]_{i,j} = k(\mathbf{y}_i,\mathbf{y}_j)$ is the Gram matrix, the kernel vector $\mathbf{k}$ decomposes as $[\mathbf{k}(\mathbf{x},\mathbf{Y})]_j = k(\mathbf{x},\mathbf{y}_j)$, the hyperparameter σ_s^2 indicate the noise variance, and the N previous data points are stored in $\mathbf{Y} = [\mathbf{y}_1,\ldots,\mathbf{y}_n]$ with corresponding rewards $\mathbf{t} = [t_1,\ldots,t_n]$.

Both the mean and variance equations can be rewritten as the weighted sum of Gaussians, giving

$$\begin{aligned}\mu(\mathbf{x}) &= \textstyle\sum_{j=1}^{N} k\left(\mathbf{x},\mathbf{y}_j\right)\alpha_j,\\ \sigma^2(\mathbf{x}) &= k(\mathbf{x},\mathbf{x}) - \textstyle\sum_{i=1}^{N}\sum_{j=1}^{N} k'\left(\mathbf{x}, 0.5\left(\mathbf{y}_i+\mathbf{y}_j\right)\right)\gamma_{ij},\end{aligned}$$

where $k'(\mathbf{x},\mathbf{y}) = \sigma_a^2\exp(-(\mathbf{x}-\mathbf{y})^T\mathbf{W}(\mathbf{x}-\mathbf{y}))$, and the constants are defined as $\alpha_j = [(\mathbf{K}+\sigma_s^2\mathbf{I})^{-1}\mathbf{t}]_j$ and $\gamma_{ij} = [(\mathbf{K}+\sigma_s^2\mathbf{I})^{-1}]_{i,j}\exp(-0.25(\mathbf{y}_i-\mathbf{y}_j)^T\mathbf{W}(\mathbf{y}_i-\mathbf{y}_j))$. Different upper confidence intervals σ have been used in UCB policies [44], and can be used by modeling them with a second GPR [7].

The previous rewards $\mathbf{t}$ occur in the exploitation term $\mu(\mathbf{x})$, but not in the standard deviation $\sigma(\mathbf{x})$ as it represents the exploration, which is independent of the rewards. 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 [40].

Having chosen a UCB policy framework and a GPR merit model, the implementation of the policy has to be adapted to the merit function.

2.3. UCB Policy for GPR Model

Given a model of the UCB merit function, the system requires a suitable method for determining the action with the highest merit. Executing this grasping action will acquire the greatest combination of reward and information.

The merit function will most likely not be concave and will contain an unknown number of maxima with varying magnitudes [32]. Determining the global maximum of the merit function analytically is therefore usually intractable [32]. However, numerically, we can determine a set of locally optimal grasps. Such sets of grasps will contain many maxima of the merit function, especially near the previous data points. Given a set of local maxima, the merit of each candidate grasp is evaluated and the robot executes the grasp with the highest merit.

The method for finding the local maxima was inspired by mean-shift [9], which is commonly used for both mode detection of kernel densities and clustering. Mean-shift converges onto the local maxima of a given point by iteratively applying

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

where $k(\mathbf{x}_n, \mathbf{y}_j)$ is the kernel function, and $\mathbf{y}_j$ are the N previously tested maxima candidates as before. The monotonic convergence via a smooth trajectory can be proven for mean-shift [9]. To find all of the local maxima, mean-shift initializes the update sequence with all previous data point. The global maximum is then determined from the set of local maxima, which is guaranteed to include the global maximum [23].

The intuition behind this approach for grasping is that all of the previous grasp attempts are locally re-optimized based on the current empirical knowledge, as modeled by the merit function. Subsequently, we choose the best of these optimized grasps to execute and evaluate.

Mean-shift is however limited to kernel densities and does not work directly in cases of regression, because the α_j and $\gamma_{i,j}$ weights are not always positive [9]. In particular, the standard update rule (1) can not be used, nor can we guarantee that the global maximum will be one of the detected maxima. However, the global maximum is only excluded from the set of found maxima if it is isolated from all previous samples by regions of low merit.

As Equation (1) is not applicable in our regression framework, a new update step had to be developed, which monotonically converges upon the local maximum of our merit function.

Algorithm 1:
Continuum Gaussian Bandits (CGB)

Initialize:
  Store N initial points in $\mathbf{Y}$ and $\mathbf{t}$
Loop:
  Calculate α and γ
  $M_{\text{best}} = 0$
  for $j = 1$ to N
    $\mathbf{x}_o = \mathbf{y}_j$
    while not converged
      Calculate update step $\mathbf{s}$
      $\mathbf{x}_{n+1} = \mathbf{s} + \mathbf{x}_n$
    end
    if $M(\mathbf{x}) > M_{\text{best}}$
      $\mathbf{x}_{\text{best}} = \mathbf{x}_n$
      $M_{\text{best}} = M(\mathbf{x}_{\text{best}})$
    end
  end
  Attempt and evaluate $\mathbf{x}_{\text{best}}$
  Store results in $\mathbf{y}_{N+1}$ and $\mathbf{t}_{N+1}$
  $N = N + 1$

Figure 3: The algorithm models the merit function with GPR, and finds a set of local maxima using a parallel search. The candidate action with the greatest merit is evaluated and the results are stored.

2.4. Local Maxima Detection for GPR

Given the model in Section 2.2, the merit function takes the form $M(\mathbf{x}) = \sum_{j=1}^{N} k(\mathbf{x}, \mathbf{y}_j)\alpha_j + \sqrt{k(\mathbf{x}, \mathbf{x}) - \sum_{i=1}^{N}\sum_{j=1}^{N} k'\left(\mathbf{x}, 0.5\left(\mathbf{y}_i + \mathbf{y}_j\right)\right)\gamma_{ij}}$. To use the policy described in Section 2.3 with this merit function, a monotonically converging update rule is required that can determine local maxima. We propose an update rule consisting of the current gradient of the merit function, divided by a local upper bound of the merit's second derivative; Specifically, we propose

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

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

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

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

This form of update rule displays the desired convergence qualities, as explained in Section 2.5. The rule is only applicable because the Gaussian kernels have bounded derivatives resulting in finite $q(\mu)$ and $q(v)$, and any real system will have a positive variance giving a real non-zero $\sqrt{p(v)}$.

To calculate the local upper and lower bounds, we first define a region of possible $\mathbf{x}_{n+1}$ 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, $q(\mu)$, $q(v)$, and $p(v)$ need to be evaluated.

In Section 2.2, 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 $q(\mu)$, $q(v)$, and $p(v)$. 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.,

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

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 $q(\mu)$ and $q(v)$, the magnitudes of the second derivatives can be bounded by a Gaussian; i.e.,

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

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

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 has a time complexity of $O(N^3)$, similar to all other exact GPR methods [7]. However, this complexity scales linearly with the number of dimensions, while discretization methods scale exponentially, making the proposed GPR method advantageous 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.

This section concludes the details of the proposed reinforcement learner, which is outlined in Algorithm 1. As shown, the final algorithm is quite compact and straightforward. It consists of modeling the expected rewards using GPR, and applying a parallel search to determine a maximum to evaluate next. The mode detection behavior is analyzed in the next section. Incorporating supervised data from other data sources is described in Section 2.6 which completes the upper level of the controller design.

2.5. Mode Detection Convergence Analysis

Having specified the method for determining maxima of a GPR in Section 2.4, Lyapunov's direct method can be used to show that the method converges monotonically to stationary points. The underlying principle is that an increased lower bound on the merit reduces the set of possible system states and, therefore, a continually increasing merit leads to convergence. The following one dimensional analysis will show that only an upper bound on the magnitude of the second derivative is required for a converging update rule.

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

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

This term is maximal when the linear 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 Equation (2). The same update rule can be found by using a negative gradient and updating x in the negative direction. The merit thus always increases, unless the local gradient is zero or u is infinite. A zero gradient indicates that the local stationary point has been found, and variable u is finite for any practical Gaussian process. In some cases, the initial point may be within the region of attraction of a point at infinity, which can be tested for by determining the distance from the previous data points.

The intuition underlying the results of the analysis is that at each step, the system assumes the gradient will shift towards zero at the maximum possible rate within the region. The estimate of the maximum is then moved to the first point where a zero gradient is possible. This concept can easily be generalized to higher dimensional problems. 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.

2.6. Incorporating Supervised data

Having fully designed the central reinforcement learner, the upper level controller still requires a method for allowing prior task information to be incorporated into the merit function to help reduce the search space.

Similar to how a child learns a new task by observing a parent before trying it themselves [28], a robot can use human demonstrations of good grasps to define its starting search region. However, whether these grasps are suitable for the robot is initially unknown.

GPR makes incorporating prior information fairly straightforward. If the supervised data has a reward associated to it, the data can be directly added to the data set. If the region suggested by the demonstration returns only low rewards, the system will begin searching neighboring areas where the merit is still high due to uncertainty. Thus, it defines an initial search region with soft boundaries that can move during the learning process.

The parameter σ_a of the merit function specifies how conservative the policy is in expanding these boundaries; i.e., a higher value will encourage more exploration, while a lower value will converge faster. Hence, it can be seen as a learning rate. With the rewards in the grasping task set to be within the range 0 to 1, the parameter is set to 0.75 to encourage exploration but also allow for a reasonable rate of convergence.

The robot experiment was initialized with search regions defined by 7, 10, and 25 demonstrated grasps for the box, watering can, and paddle respectively. The width parameters $\mathbf{W}$ of the Gaussian kernel were also optimized on these initial parameters.

This section concludes the discussion of the upper level controller. It takes the rewards of grasps, the pose of the object, and, optionally, demonstrated data as inputs, and returns the next grasp location to attempt. This grasp location is passed to the robot via a lower level controller, which generates the complete grasping motions based on these parameters.

3. Low Level Reactive Imitation Controller

While the upper level of the controller selected grasp locations, the lower level is responsible for the execution of the grasp, including the reaching and fingers' motions. It is important that the system is adaptive at this level, as the success of the grasps depend on the execution. The finger motions should particularly adapt to the geometry of the object, a process known as preshaping. The

robot's motions are learned from human demonstrations, and subsequently modified to incorporate the grasp information from the active learner and the scene geometry from the vision system.

A common approach to the grasp execution problem is to rely on specially designed sensors (e.g., laser scanner, ERFID) to get accurate and complete representations of the object and environment [26, 45], followed by lengthy planning phases in simulation [5]. We restrict the robot to only using stereo cameras, and a fast reactive sensor-based controller [39].

Although densely sampling sensors such as time-of-flight cameras and laser range finders are favored for reactive obstacle avoidance [17], the sparser information of stereo vision systems has also been used for these purposes [34, 22]. Robot grasping research has focused on coarse object representations of novel objects [43, 25, 30, 8], and using additional sensor arrays when in close proximity to the object [12, 41]. Learning to grasp objects is also often done in simulation [43, 20] which allows for many virtual grasp attempts on a model of the object. In contrast, the proposed hybrid system relies on relatively few real-world grasps and does not rely on having accurate dynamics and contact models.

For the lower level system, we propose a sensor-based robot controller that can perform human inspired motions, including preshaping of the hand, smooth and adaptive motion trajectories, and obstacle avoidance, using only stereo vision to detect the environment. Unlike previous approaches, we work with a sparse visual representation of objects, which maintains a high level of geometric details. The controller uses potential field methods [39], which treat the robot's state as a particle in a force field; i.e. the robot is attracted to a goal state, and repelled from obstacles.

The attractor field needs to be capable of encoding complex trajectories and adapting to different grasp locations. We therefore use the dynamical system motor primitive (DMP) [13, 37] framework. The DMPs are implemented as passive dynamical systems superimposed with external forces; i.e.,

$$\ddot{y} = \alpha_z(\beta_z\tau^{-2}(g-y) - \tau^{-1}\dot{y}) + a\tau^{-2}f(x), \tag{3}$$

where α_z and β_z are constants, τ controls the duration of the primitive, a is an amplitude, $f(x)$ is a nonlinear function, and g is the goal for the state variable y. The variable $x \in [0,1]$ is the state of a canonical system $\dot{x} = -\tau x$, which acts as a shared clock amongst different DMPs; i.e. it ensures that the finger and arm motions are synchronized. The function $f(x)$ encodes the trajectory for reaching the goal state, and takes the form

$$f(x) = \frac{\sum_{j=1}^{M} \psi_j(x)\, w_j x}{\sum_{i=1}^{M} \psi_i(x)},$$

where $\psi(x)$ are M Gaussian basis functions, and w are weights. The weights w are acquired by imitation learning, using locally weighted regression [13, 14]. The DMPs treat the goal state g as an adjustable variable and ensure that this state is always reached. However, their capability to generalize can be further improved by using a task-specific reference frame based on the active learner's grasp parameters, as detailed in Section 3.2. This adaptation of the action to different goals allows the object to be repositioned and reorientated in the robot's workspace.

More important is the choice of the scene's visual representation, which is used to augment the attractor field and forms the basis of the detractor field. The scene description needs to be in 3D, work at a fine scale to maintain geometric details, and represent the scenes sparsely to reduce the number of calculations required per time step. The Early Cognitive Vision system of Pugeault et al. [31, 11] (see Figure 4) fulfills these requirements by extracting edge features from the observed scene. The system subsequently localizes and orientates these edges in 3D space [21],

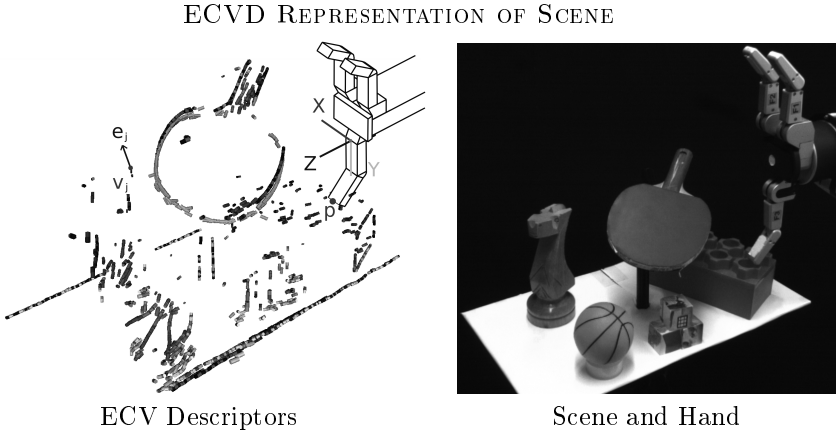


Figure 4: The left image shows the ECVD representation of the scene on the right. The paddle is the object to be grasped, while the surrounding objects clutter. The coordinate frame of the third finger of the lower finger in the image and the variables used in Section 3 are shown. The x-y-z coordinate system is located at the base of the finger, with z orthogonal to the palm, and y in the direction of the finger. The marked ECVD on the left signifies the j^{th} descriptor, with its position at $\mathbf{v}_j = (v_{jx}, v_{jy}, v_{jz})^T$, and edge direction $\mathbf{e}_j = (e_{jx}, e_{jy}, e_{jz})^T$ of unit length. The position of the finger tip is given by $\mathbf{p} = (p_x, p_y, p_z)^T$.

with the resulting features known as early cognitive vision descriptors (ECVD) [31]. By using a large amount of small ECVDs, any arbitrary object/scene can be represented. Given an ECVD model of an object, the object's position and orientation can be determined [10] and the ECVDs of the object model can be superimposed into the scene representation.

As a hybrid system, the lower level controller supplies a complex adaptive action policy that the upperlevel can indirectly modify. The top level controller only needs to modify the action for a given object, which can be done more efficiently than having to learn the entire action. To allow for quick learning, the actions given by the reactive controller should be repeatable, while still adaptive. By making the rewards for grasps depend on the reactive controller, the reinforcement learner finds both good grasp locations as well as matching grasp executions.

In Sections 3.1 and 3.2, we describe the DMPs for grasping, followed by their augmentation using the ECVD based detractor field in Section 3.3.

3.1. Attractor Fields based on Dynamical Systems Motor Primitives (DMPs)

Generating the grasp execution begins with defining an attractor field as a DMP, which encodes the desired movements given no obstacles. The principle features that need to be defined for these DMPs are the goal positions, and the generic shape of the trajectories.

The high level grasp controller gives the goal location and orientation of the hand, but not the fingers. Using the ECVDs, the goal position of each finger is approximated by first estimating a locally linearized contact plane for the object in the finger coordinate system (see Figure 4). The purpose of this step is to get the fingers close to the object's surface during preshaping to allow for more control of the object during grasping. It is not intended to infer exact surface properties or whether the grasp is suitable. If the selected surface is unsuitable for grasping, a low reward will be received and the upper level controller will adapt its policy accordingly.

A contact plane is approximated for each finger to allow for a range of object shapes. The influence of the i^{th} ECVD is weighted by $w_i = \exp(-\sigma_x^{-2} v_{ix}^2 - \sigma_y^{-2} v_{iy}^2 - \sigma_z^{-2} v_{iz}^2)$, where σ_x, σ_y, and σ_z are length constants that reflect the finger's length and width, and $\mathbf{v}_i$ is the position of the

ECVD in the finger reference frame. The hand orientation is such that the Z direction of the finger should be approximately parallel to the contact plane, which reduces the problem to describing the plane as a line in the 2D X-Y space. The X-Y gradient of the plane is approximated by $\phi = (\sum_{i=1}^{N} w_i)^{-1} \sum_{i=1}^{N} w_i \arctan(e_{iy}/e_{ix})$, where N is the number of vision descriptors, and $\mathbf{e}_i$ is the direction of the i^{th} edge. The desired Y position of the fingertip is then given by

$$\tilde{p}_y = \frac{\sum_{i=1}^{N}(w_i v_{iy} - \tan(\phi) w_i v_{ix})}{\sum_{i=1}^{N} w_i},$$

which can be converted to joint angles using the inverse kinematics of the hand. The proposed method selects the goal postures of the fingers in a deterministic manner, which depends on the object's geometry as well as the grasp parameters specified by the active learner. Thus, the hybrid system's active learner indirectly selects the posture of the fingers through a reactive mechanism based on the visual model of the object.

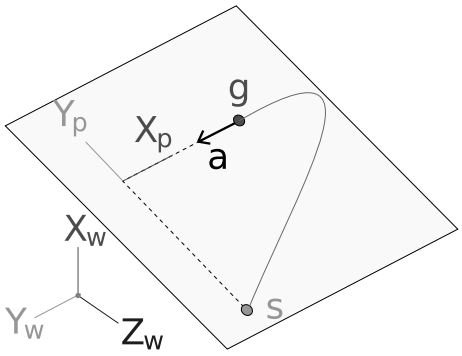


Figure 5: The diagram shows the the change in coordinate systems for the reaching DMPs. The axes X_w-Y_w-Z_w are the world coordinate system, and X_p-Y_p-Z_p is coordinate system in which the DMP is specified. The trajectory of the DMP is shown by the curved line, starting at **point s**, and ending at **point g**. X_p is parallel to the approach direction of the hand, the **arrow a**. The axis Y_p is perpendicular to X_p, and pointing from **s** towards **g**.

The next step defines the reaching and grasping trajectories. Many beneficial traits of human movements, including smooth motions and small overshoots for obstacle avoidance [16, 15, 29], can be transferred to the robot through imitation learning. To demonstrate grasping motions, we used a VICON motion tracking system to record human movements during a grasping task. The grasped object can be different to the robot's. VICON markers were only required at the hand and finger tips. The tracking system samples the human's motions, generating position $\mathbf{q}$, velocity $\dot{\mathbf{q}}$, and acceleration $\ddot{\mathbf{q}}$ data, as well as the samples' time stamps. The weights w_i of the DMP are then given by

$$w_i = \left(\sum_{k=1}^{T} \psi_i(x_k) x_k^2 \right)^{-1} \sum_{j=1}^{T} \psi_i(x_j) x_j \left(\tau^2 \ddot{q}_j - \alpha_z(\beta_z(g - q_j) - \tau^1 \dot{q}_j) \right) a^{-1},$$

where x_j is the state of the canonical system corresponding to the j^{th} time stamp. The solution is closed form and easily calculated. Further information on imitation learning of DMPs can be found in Ijspeert's paper [14]. As the reaching trajectories are encoded in task space the correspondence problem of the arm was not a problem.

The DMPs are provably stable [37] and the goal state, as specified by the upper level controller, will always be achieved. Alterations added by the reactive controllers must stay within the bounds of the framework to ensure that this stability is maintained.

3.2. Transformed Dynamical Motor Primitives for Grasping

While DMPs generalize to arbitrary goal positions, the grasps' approach direction can not be arbitrarily defined, and the amplitude of the trajectory is unnecessarily sensitive to changes in the start position y_0 and the goal position g if $y_0 \approx g$ during training. These limitations can be overcome by including a preprocessor that modifies the DMPs' hyperparameters.

The system can maintain the correct approach direction by using a task-specific coordinate system. Due to the translation invariance of DMPs, only a rotation $\mathbf{R} \in \mathbb{SO}(3)$ between the two coordinate systems needs to be determined. The majority of the reaching motions will lie in a plane defined by the start and goal locations, and the final approach direction. These components of the plane are supplied by the high level controller, with the approach direction defined by the final hand orientation.

The first new in-plane axis $\mathbf{x}_p$ is set to be along the approach direction of the grasp; i.e., $\mathbf{x}_p = -\mathbf{a}$ as shown in Figure 5. The approach direction is thus easily defined and only requires that the Y_p and Z_p DMPs reach their goal before the X_p primitive. The second axis, $\mathbf{y}_p$, must be orthogonal to $\mathbf{x}_p$ and also in the plane, as shown in Figure 5. It is set to $\mathbf{y}_p = b^{-1}((\mathbf{g}-\mathbf{s}) - \mathbf{x}_p(\mathbf{g}-\mathbf{s})^T\mathbf{x}_p)$, where b^{-1} is a normalization term, and $\mathbf{s}$ and $\mathbf{g}$ are the motion's 3D start and goal positions respectively. The third axis vector is given by $\mathbf{z}_p = \mathbf{x}_p \times \mathbf{y}_p$. The DMPs can thus be specified by the preprocessor in the X_p-Y_p-Z_p coordinate system, and mapped to the X_w-Y_w-Z_w world reference frame by multiplying by $\mathbf{R}^T = [\mathbf{x}_p, \mathbf{y}_p, \mathbf{z}_p]^T$.

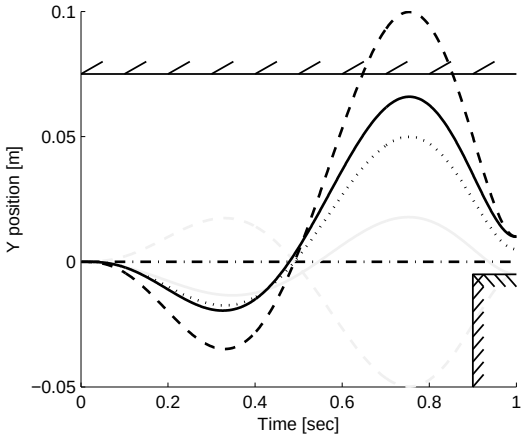


Figure 6: This is a demonstration of the effects of transforming the amplitude variable a of DMPs. The hashed black lines represent boundaries. The **dotted black** line shows the trained trajectory of the DMP going to 0.05. If goal is then placed at 0.1 and the workspace is limited to ± 0.075 (top boundary), the **dashed black** line is the standard generalization to a larger goal, while the **solid** plot uses the new amplitude. If the goal is -0.05, and needs to be reached from above (lower right boundary), then the **dashed grey** line is the standard generalization to a negative goal, and the **solid grey** trajectory uses the new amplitude. Both of the new trajectories were generated with $\eta = 0.25$.

The change of coordinate system is a fundamental step for the hybrid system. It places the reactive controller, together with all of its modifications, within the reinforcement learner's action-reward feedback loop. Therefore, the system learns pairings of grasp locations and grasp executions that lead to high rewards.

The second problem relates to the scaling of motions with ranges greater than $\|y_0 - g\|$, which are required to move around the outside of objects. In the standard form $a = g - y_0$ [13], which leads to motions that are overly sensitive to changes in g and y_0 if $g \approx y_0$ during training. The preprocessor can reduce the sensitivity by using a more robust scaling term, for which we propose the amplitude

$$a = \|\eta(g - y_0) + (1-\eta)(g_T - y_{0T})\|\,,$$

where g_T and y_{0T} are the goal and start positions of the training data respectively, and $\eta \in [0, 1]$ is a weighting hyperparameter. This amplitude is always between the training amplitude and the standard generalization value $a = g - y_0$, and η controls how conservative the generalization is to new goals (see Figure 6). By taking the absolute value of the amplitude, the approach direction is never reversed (see Figure 6). The amplitude previously proposed by Park et al. [30] corresponds to the special case of $\eta = 0$. Example generalizations of a reaching trajectory are shown in Figure 7.

The described transformations allow a single DMP to perform a larger range of grasps, which implies that fewer DMPs ares required in total. Using different DMPs for different sections of the object or workspace should be avoided as it creates unnecessary discontinuities in the rewards,

which can slow down the hybrid system's learning process. Only one grasp had to be learned for the entire robot experiment, which was then adapted to the various situations.

3.3. Detractor Fields based on ECVDs

Detractor fields refine the motions generated by the DMPs to avoid obstacles during the reaching motion and ensure that the finger tips do not collide with the object during the hand's approach.

The detractor field is based on ECVDs, which represent small line segments of an object's edges localized in 3D, as shown in Figure 4. The detractive forces of multiple ECVDs describing a single line should not superimpose, nor should the field stop DMPs from reaching their ultimate goals. The system therefore uses a Nadaraya-Watson model [7] of the form

$$u_a = -v(x)\frac{\sum_{i=1}^{N} r_i c_{ai}}{\sum_{j=1}^{N} r_j},$$

to generate a suitable detractor field, where r_i is a weight assigned to the i^{th} ECVD, s is the strength of the overall field, x is the state of the DMPs' canonical system, c_{ai} is the detracting force for a single descriptor, and subscript a specifies if the detractor field is for the finger motions or the reaching movements.

The weight of an ECVD for collision avoidance is given by $r_i = \exp(-(\mathbf{v}_i - \mathbf{p})^{\mathrm{T}}\mathbf{h}(\mathbf{v}_i - \mathbf{p}))$, where $\mathbf{v}_i$ is the position of the i^{th} ECVD in the local coordinate system, $\mathbf{h}$ is a vector of positive length scale hyperparameters, and $\mathbf{p}$ is the finger tip position, as shown in Figure 4. The detractor puts more importance on ECVDs in the vicinity of the finger.

The reaching and finger movements react differently to edges and employ different types of basis functions c_i for their respective potential fields. For the fingers, the individual potential fields are logistic sigmoid functions about the edge of each ECVD of the form $\rho(1 + \exp(d_i\sigma_c^{-2}))^{-1}$, where $d_i = \left\|(\mathbf{p} - \mathbf{v}_i) - \mathbf{e}_i(\mathbf{p} - \mathbf{v}_i)^{\mathrm{T}}\mathbf{e}_i\right\|$ is the distance from the finger to the edge, $\rho \geq 0$ is a scaling parameter, and $\sigma_c \geq 0$ is a length parameter. Differentiating the potential field results in a force of

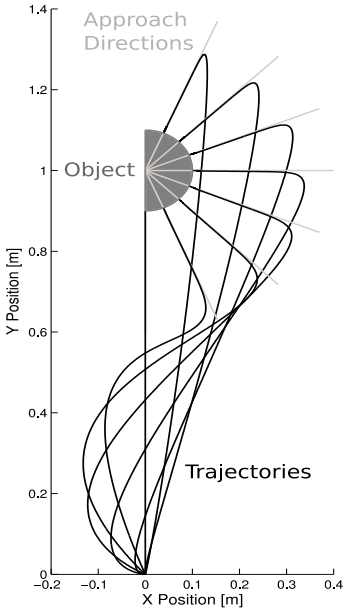


Figure 7: Workspace trajectories where the x and y values are governed by two synchronized DMPs. The semicircle indicates the goal positions, with desired approach directions indicated by the light gray straight lines. The approach direction DMP was trained on an amplitude of one, and $\eta = 0.25$.

$$c_{fi} = \rho\left(1 + \exp\left(d_i\sigma_c^{-2}\right)\right)^{-2}\exp\left(d_i\sigma_c^{-2}\right).$$

As the sigmoid is monotonically increasing, the detractor always forces the fingers open further to move their tips around the ECVDs and ensure that they approach the object from the outside. A similar potential function can be employed to force the hand closed when near ECVDs pertaining to the scene rather than the object.

The reaching motion uses the Gaussian basis functions of the form $\varrho\exp(-0.5\mathbf{d}_i^{\mathrm{T}}\mathbf{d}_i\sigma_d^{-2})$, where $\mathbf{d}_i = (\mathbf{q} - \mathbf{v}_i) - \mathbf{e}_i(\mathbf{q} - \mathbf{v}_i)^{\mathrm{T}}\mathbf{e}_i$ is the distance from the end effector position, $\mathbf{q}$, to the edge, and $\varrho \geq 0$ and $\sigma_d \geq 0$ are scale and length parameters respectively. Differentiating the potential with respect to $\mathbf{d}_i$ gives a force term in the Y direction of

$$c_{hi} = \varrho(\mathbf{d}_i.\mathbf{Y})\sigma_d^{-2}\exp(-0.5\mathbf{d}_i^{\mathrm{T}}\mathbf{d}_i\sigma_d^{-2}),$$

which thus apply a radial force from the edge with an exponentially decaying magnitude.

The strength factor $s(\mathbf{x})$ controls the precision of the movements, ensuring that the detractor forces tend to zero at the end of a movement and do not obstruct the DMPs from achieving its goal state. Therefore, the strength of the detractors is coupled to the canonical system of the DMP. Hence, $v(x) = (\sum_{j=1}^{M} \psi_j)^{-1} \sum_{i=1}^{M} \psi_i w_i x$, where x is the value of the canonical system, ψ are its basis functions, and w specify the varying strength of the field during the trajectory.

Modelling the human tendency towards more precise movements during the last 30% of a motion [16], the strength function, $v(x)$, was set to give the highest strengths during the first 70% of the motion for the reaching trajectories, and the last 30% for the finger movements. Setting the strength in this manner is also beneficial to the reinforcement learner. The reward of the learner depends mainly on the final position of the hand, and the closing of the fingers. If these parts of the motion are more repeatable, then it is easier for the upper level controller to learn.

The detractor fields of both the grasping and reaching components have been defined, and are superimposed into the DMP framework as

$$\ddot{y} = \left(\alpha_z(\beta_z \tau^{-2}(g - y) - \tau^{-1}\dot{y}) + a\tau^{-2} f(x)\right) - \tau^{-2} u_a,$$

which represents the entire ECVD and DMP based potential field.

Combining the ECVD based DMPs with the new coordinate system for reaching and motion amplitude, we have fully defined the low level controller. Its main contribution is to learn a grasping movement by imitation and then to reactively adapt these motions to new situations in a manner suited to the task and specified by the upper level controller.

4. Evaluations

The following sections evaluate the system both in simulation and on a real robot platform. The first part of the evaluation (Section 4.1) tests the upper level controller against other continuum UCB policies on a simulated benchmark problem. The real world evaluation, presented in Section 4.2, demonstrates the complete controller working on a real robot grasping novel objects in cluttered environments.

4.1. Comparative UCB Analysis

This section focuses on the reinforcement learner and shows that the CGB algorithm (see Algorithm 1) performs well in practice, and can be scaled to the more complex domain of grasp learning. The comparison is between four UCB policies, including our proposed method, on a 1D benchmark example of the continuum-armed bandits problem. The policies were tested 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 bandits was also restricted to a range between 0 and 1. None of the policies were informed of the length of the experiment in advance, and each policy was tuned to achieve high rewards.

4.1.1. Compared Methods

The tested competing policies are UCBC [3], CAB1 [18], and Zooming [19]. These algorithms represent standard UCB policy implementations for continuum bandits in the literature. 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. Balancing this trade-off is therefore important for a policy's success.

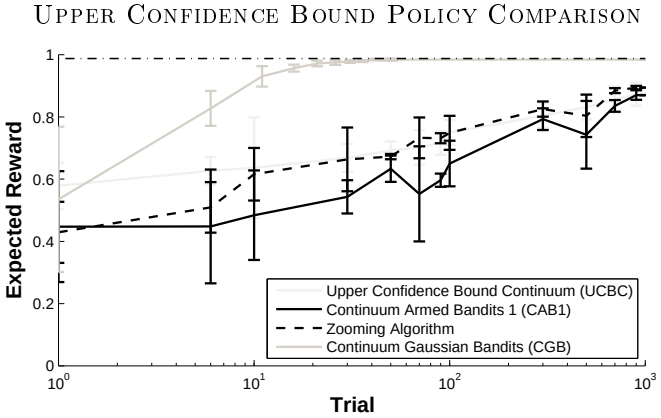


Figure 8: The expected rewards over 100 experiments are shown for the four compared methods. The results were filtered for clarity. Due to the differences in experiment lengths, the x-axis uses a logarithmic scale. The dashed horizontal line represents the maximum expected reward given the noise.

The UCBC policy of Auer [3] divides the bandits space into regular intervals and treats each interval as a bandit in a discrete UCB 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 of Kleinberg [18] selects specific grasps at uniform grid points. A discrete UCB policy is then applied to these points, for which we chose UCB1 [2], as suggested in [18]. The discretization trade-off is dealt with by resetting the system at fixed intervals with larger numbers of bandits, thus ensuring that the points becomes denser as the experiment continues.

The zooming algorithm, of Kleinberg et al. [19], also uses a grid structure to discretize the bandits. In contrast to CAB1, the grid is not uniform and additional bandits can be introduced at any time in high rewarding regions. A discrete policy is then applied to this set of active bandits. Similar to CAB1, the zooming algorithm works in time intervals and resets its grid after fixed numbers of trials.

Our proposed Continuum Gaussian Bandits (CGB) method was initialized with 4 equispaced points. Demonstrated data was not used in order to test its performance without the benefits of such data. All four methods were initially run for 55 trials, as shown in Figure 8. The CAB1, UCBC, and Zooming methods extended to 1000 trials to demonstrate their convergence behavior.

4.1.2. Results

The expected rewards for the four UCB policies during the experiment can be seen in Figure 8. The computation and run times were also acquired for the experiments for comparison, and estimated for the 6 dimensional problem, as shown in Table 1.

Apart from our proposed policy, Zooming was the most successful over the 1000 trials at achieving high rewards, as it adapts its grid to the reward function. However, only CGB consistently determined the high rewarding regions and converged on them. In several trails, the reward function had two distinct peaks with near-optimal rewards, and the CGB policy converged onto both.

The convergence of UCB policies is frequently described by the merit's percentage of exploitation $\mu(\mathbf{x}^*)/(\mu(\mathbf{x}^*)+\sigma(\mathbf{x}^*))$, where $\mathbf{x}^*$ is the current action selected by the policy. This value is initially

	UCBC	CAB1	Zoom	CGB
Mean Reward	0.6419	0.4987	0.6065	**0.9122**
1D computation time	46 μs	47 μs	**27 μs**	2.9 sec
6D computation time	4.6 sec	6.7 ms	**5.6ms**	17.6 sec
1D initialization run time	10 min	12 min	24 min	**4 min**
6D initialization run time	1.9 yrs	1.2 days	4.2 days	**24 min**

Table 1: These results pertain to the first 50 grasp attempts in the benchmark problem. The shows the mean computation times for the different algorithms, and how they would scale to six dimensions, given the computational complexity of the algorithms [18, 3, 19]. Similarly, the table shows the amount of time needed to initialize the systems by trying each of the initial grasps once.

zero and increases as the policy returns to previously explored actions with high rewards. The 97.5% exploitation mark was reached by the CGB policy on average at the 33^{rd} trial. Another measure of convergence is found by directly comparing the different maxima found by CGB. The policy converges when the expected value $\mu(\mathbf{x}^*)$ of the selected action is greater than the highest merit value $\mu(\mathbf{x}) + \sigma(\mathbf{x})$ of the other candidate actions. This criterion is based on the fact that the merit function $\mu(\mathbf{x}) + \sigma(\mathbf{x})$ tends to $\mu(\mathbf{x})$ as the exploration of an action is exhausted. Using this criteria, the policy converged on average at the 37^{th} trial.

As parametric policies, the standard methods assume that the optimal solution can be represented by their fixed features and corresponding parameters. These policies can therefore only converge to an optimal solution if it is representable by these features. Both CAB1 and the Zooming algorithm will converge onto the true optimum, but only as the number of samples tends to infinity, as indicated in Figure 8.

In terms of computation times, the previous methods were faster than the proposed method, although CGB and UCBC exhibit similar orders of magnitude. One reason for CGB being slower is that this implementation performs the parallel search for maxima sequentially. Parallelizing this search would reduce the expected 6D computation time of CGB to 0.65 seconds.

Most of the system's time is however used to perform the actions (i.e., the run times). For this comparison we focused on the time required to initialize the systems by trying each initial grasp once. Not only is the proposed method the fastest in terms of run times (see Table1), it also shows that implementing the other methods for grasping is not practical due to the curse of dimensionality.

The UCBC algorithm has both longer computation and running times than CAB1 and the Zooming algorithm. However, as CAB1 and the Zooming algorithm increase the number of active actions throughout the experiment, these would ultimately exhibit computation and run times greater than UCBC.

The memory requirements of the previous methods increases exponentially with the dimensionality, and CGB will only require more memory than UCBC once it has performed a million grasps. The memory requirements of CGB scale with the number of samples, and sufficient memory should be made available depending on the difficulty of the learned task.

In cases where large numbers of samples have been accumulated, suitable implementations of GPR (e.g., Sparse GP [38]) reduce the computational 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.

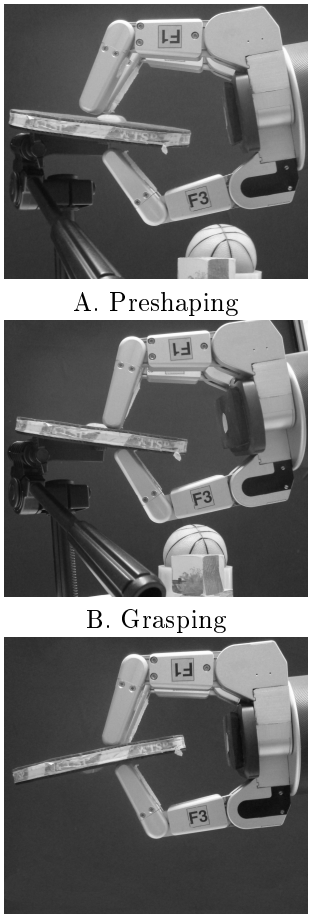
Ultimately the experiment shows that the proposed method outperforms the other methods in a low dimensional setting, and is the only practical method for higher dimensions due to the curse

of dimensionality.

4.2. Robot Grasping Task

Having shown that the proposed Gaussian Bandits algorithm is an efficient UCB policy, the robotics evaluation focuses on including the lower level controller for improved actions in a robot grasping scenario. This experiment involves the complete system being implemented on a real robot platform. The following sections detail the running of the experiment (Section 4.2.1) and the results of the experiment (Section 4.2.2).

In this experiment, we implement only the methods proposed in this paper. The methods described in Section 4.1.1 were not tested on the real system as their discretizations make them highly impractical.



A. Preshaping

B. Grasping

C. Lifting

Figure 9: The three main phases of a basic grasp are demonstrated. (A) Preshaping the hand poses the fingers to match the object's geometry. (B) Grasping closes the three fingers at the same rate to secure the object. (C) The object is lifted and the fingers adjust to the additional weight. The objects at the bottom of A and B are clutter.

4.2.1. Grasping Experiment

The robot is a basic hand-eye system consisting of a 7 degrees of freedom Mitsubishi PA-10 arm, a Barrett hand, and a Videre stereo camera. The robot only uses sensors essential for the task and forgoes additional hardware such as tactile sensors and laser rangefinders. The robot's task was to learn several good grasps of novel objects through trial and error. All grasps were executed on the real robot and not in simulation.

Each trial begins by estimating the object's position and orientation to convert between world and object reference frames, and to project the ECVD model of the object into the scene representation. The stereo camera allows the object position and orientation to be reliably estimated using the pose estimation method of Detry et al. [10].

The CGB algorithm then determines the parameters of the next grasp, which the reactive lower level controller uses to modify the grasping action. If the robot grasps the object, the robot attempts to lift the object from the table, thus ensuring that the table is not supporting to the object. Trials are given rewards depending on how little the fingers moved while lifting the object, thereby encouraging more stable grasps. The rewards are not deterministic due to errors in pose estimation and effects caused by the placement of the object.

The robot task was made more difficult by adding clutter to the scene. After each grasp attempt, the hand reverses along the same approach direction, but without employing the detractor fields or preshaping of the hand, to determine if collisions would have occurred if the reactive controller had not been used.

The system was run three times on a table tennis paddle to show that it is repeatable. To show that the system can adapt to various scenarios and objects, the experiment was also run twice on both a toy watering can and a wooden box.

The experiments for learning to grasp a paddle consisted of 55 trials, while only 40 trials were requried for the watering can and box experiments. Overall 325 different grasp attempts were executed with the combined active and reactive system.

4.2.2. Results

The active learner and reactive controller were successfully integrated and the complete system converged onto high-rewarding grasp regions in all of the trials. The imitation learning was straightforward, requiring only one demonstration and allowing for continuous smooth motions to be implemented. Examples of the estimated finger goal locations can be seen in Figure 10. The preshaping adapted to a range of geometries, and consistently placed the fingers close enough to the object for a controlled grasp to be executed. This preshaping gave more control over the object when grasping, leading to higher rewards and allowing for more advanced grasps to be performed (see Figure 11)

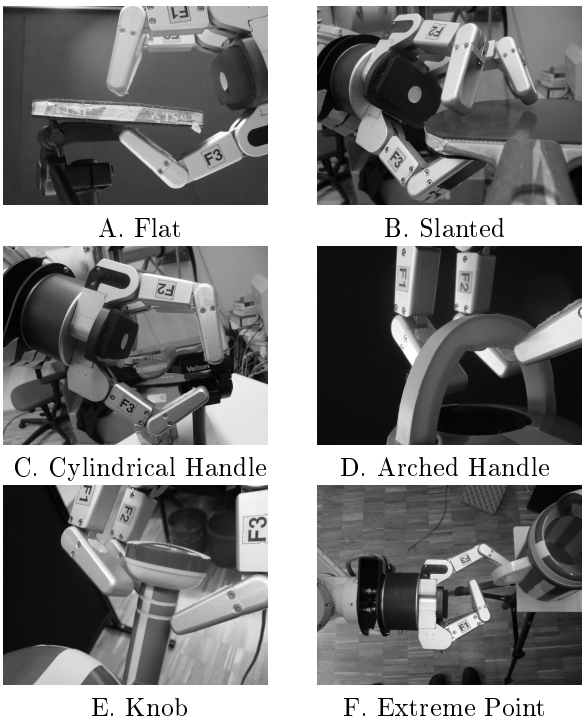


Figure 10: Various preshapes are shown. **A** and **B** show the system adjusting to different plane angles. **C** and **D** demonstrate the preshaping for different types of handles. **E** shows the preshaping for a circular disc structure, such as a door knob, and gets its fingers closely behind the object. **F** shows where the object was out of the reach of two fingers, but still hooks the object with one finger.

The detractor field and preshaping of the hand allowed the system to work in cluttered environments, which was not a trivial task. The hand came into contact with the clutter for an estimated 8.3% of the grasp attempts, but never more than a glancing contact. These contacts were usually with visually occulded parts of the objects, and thus not fully modelled by the ECVDs. Accumulating the scene representation from multiple views solves this problem. During the reversing phases, when the reactive controller is deactivated, the hand collided with one or more pieces of clutter during 85.4% of the attempts. Thus, the reactive control decreases the number of contacts with the clutter by a factor of ten. The fingers always opened sufficiently to accept the object without colliding with it.

The rewards during the experiment are shown in Figure 12. In all of the experiments, the proposed hybrid system found suitable grasps for the object. The watering can and box experiments converged faster than the paddle experiments, due to their initial search region being smaller. While all experiments acquired low rewards for the initial grasps, the soft boundaries allowed the system to explore beyond these regions and find neighbouring regions of better grasps.

Amongst the most important results of this experiment is that the central loop of the hybrid controller works in practice. The system did not just quickly learn a graspable location on an object, but rather the hybrid system quickly learned an entire fluid motion for grasping the object, including preshaping. The system took a single demonstrated action and learned modifications that

Demonstration of a Controlled Grasp

A. Preshaping B. Grasping

Figure 11: A controlled grasp, made possible by the hybrid system's preshaping ability. (A) The preshaping matchs the geometry of the object. When grasping, the two fingers on the left pinch the paddle. The finger on the right turns the paddle clockwise about the pinched point. (B) The grasping ends when the paddle has become aligned with all three finger tips.

generalized the action to three different objects. The learning process was significantly hastened by the hybrid approach, as the reactive controller allowed the dimensionality of the reinforcement learner to be kept relatively low, while simultaneously performing complicated grasping motions.

The upper and lower levels divide the grasping problem into two sub-problems: determining where to grasp an object and deciding how to correctly execute the grasp. By incorporating the reactive controller in the learning loop, the hybrid system learned an action that solves both of these sub-problems.

5. Conclusion

We have presented a hierarchical hybrid controller that can efficiently determine good grasps of objects and execute them. The upper level controller is based on reinforcement learning to allow the robot to learn from its own experiences, but capable of incorporating supervised data from other sources if available. Grasp execution is handled by a lower level controller based on imitation learning and reactive control. This hybrid structure allowed the system to learn both good grasp locations and corresponding grasp executions simultaneously, while keeping the dimensionality of the learning problem low.

We have shown that the presented algorithms and learning architectures work well both in simulation and on a real robot. In simulation, the active learner outperformed several standard UCB policies designed for the continuum-armed bandits problem. The entire system was successfully implemented on a real robot platform, which consistently found highly rewarding grasps for various objects.

References

[1] Agrawal, R., 1995. The continuum-armed bandit problem. SIAM J. Control Optim. 33 (6), 1926–1951.

[2] Auer, P., Cesa-Bianchi, N., Fischer, P., 2002. Finite-time analysis of the multiarmed bandit problem. Mach. Learn. 47 (2-3), 235–256.

[3] Auer, P., Ortner, R., Szepesvári, C., 2007. Improved rates for the stochastic continuum-armed bandit problem. In: Proc. of the Conference on Learning Theory. Springer-Verlag, Berlin, Heidelberg, pp. 454–468.

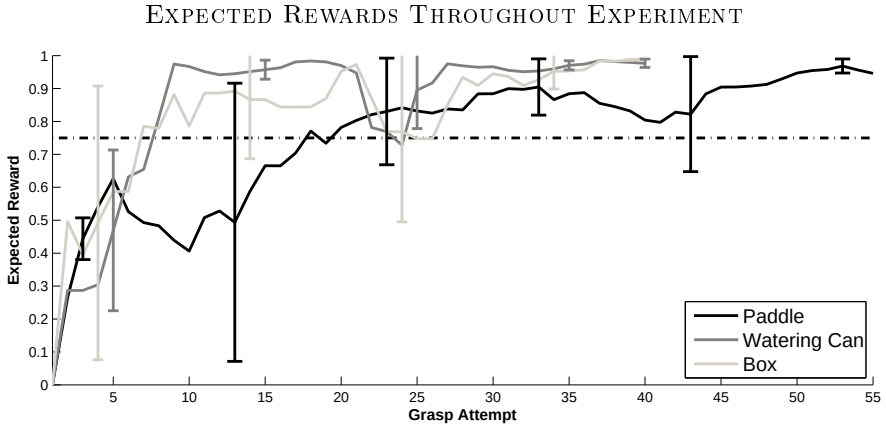


Figure 12: The graph shows the expected reward of the attempted grasps over the run of the experiment for the three different objects. All values are averaged over the runs of the experiment, with error bars of +/- two standard deviations. The dashed horizontal line indicates the upper confidence bound of a point at infinity.

[4] Bellman, R. E., 1961. Adaptive control processes - A guided tour. Princeton University Press, Princeton, New Jersey, U.S.A.

[5] Bertram, D., Kuffner, J., Dillmann, R., Asfour, T., 2006. An integrated approach to inverse kinematics and path planning for redundant manipulators. In: Proc. of International Conference on Robotics and Automation. IEEE, pp. 1874–1879.

[6] Bicchi, A., Kumar, V., August 2002. Robotic grasping and contact: a review. In: Proceedings of IEEE International Conference on Robotics and Automation. Vol. 1. IEEE, pp. 348–353.

[7] Bishop, C. M., 2007. Pattern Recognition and Machine Learning, 1st Edition. Springer.

[8] Bley, F., Schmirgel, V., Kraiss, K.-F., 2006. Mobile manipulation based on generic object knowledge. In: Proc. of Symposium on Robot and Human Interactive Communication. IEEE, pp. 411–416.

[9] Comaniciu, D., Meer, P., 2002. Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Analysis and Machine Intelligence 24 (5), 603–619.

[10] Detry, R., Pugeault, N., Piater, J., 2008. Probabilistic pose recovery using learned hierarchical object models. Springer-Verlag, Berlin, Heidelberg, pp. 107–120.

[11] Hartley, R., Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge University Press, New York, NY, USA.

[12] Hsiao, K., Nangeroni, P., Huber, M., Saxena, A., Ng, A. Y., 2009. Reactive grasping using optical proximity sensors. In: Proc. of International Conference on Robotics and Automation. IEEE Press, Piscataway, NJ, USA, pp. 4230–4237.

[13] Ijspeert, A., Nakanishi, J., Schaal, S., 2003. learning attractor landscapes for learning motor primitives. In: advances in neural information processing systems 15. cambridge, ma: mit press, pp. 1547–1554.

[14] Ijspeert, J. A., Nakanishi, J., Schaal, S., 2002. movement imitation with nonlinear dynamical systems in humanoid robots. In: Proc. of International Conference on Robotics and Automation. IEEE, pp. 1398–1403.

[15] Jeannerod, M., 1997. Perspectives of Motor Behaviour and Its Neural Basis. S Karger AG, Ch. Grasping Objects: The Hand as a Pattern Recognition Device, pp. 19–32.

[16] Jeannerod, M., 2009. Sensorimotor Control of Grasping: Physiology and Pathophysiology. Cambridge University Press, Cambridge, UK, Ch. The study of hand movements during grasping. A historical perspective, pp. 127–140.

[17] Khatib, M., 1996. Sensor-based motion control for mobile robots. Ph.D. thesis, LAAS-CNRS, Toulouse, France.

[18] Kleinberg, R., 2005. Nearly tight bounds for the continuum-armed bandit problem. In: Advances in Neural Information Processing Systems 17. MIT Press, pp. 697–704.

[19] Kleinberg, R., Slivkins, A., Upfal, E., 2008. Multi-armed bandits in metric spaces. In: Ladner, R. E., Dwork, C. (Eds.), Proc. of Symposium on Theory of Computing. ACM, pp. 681–690.

[20] Kragic, D., Miller, A. T., Allen, P. K., 2001. Real-time tracking meets online grasp planning. In: Proc. of International Conference on Robotics and Automation. IEEE, pp. 2460–2465.

[21] Krueger, N., Lappe, M., Woergoetter, F., 2004. Biologically motivated multi-modal processing of visual primitives. The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour, 1(5), 417–427.

[22] Lenser, S., Veloso, M., 2003. Visual sonar: Fast obstacle avoidance using monocular vision. In: Proc. of International Conference on Intelligent Robots and Systems.

[23] Martinez-Cantin, R., 2008. Active map learning for robots: Insights into statistical consistency. Ph.D. thesis, University of Zaragoza.

[24] Mason, M. T., Salisbury, J. K., 1985. Robot Hands and the Mechanics of Manipulation. MIT Press, Cambridge, MA.

[25] Miller, A. T., Knoop, S., Christensen, H. I., Allen, P. K., 2003. Automatic grasp planning using shape primitives. In: Proc. of International Conference on Robotics and Automation. IEEE, pp. 1824–1829.

[26] Morales, A., Asfour, T., Azad, P., Knoop, S., Dillmann, R., 2006. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In: Proc. of International Conference on Intelligent Robots and Systems. IEEE, pp. 5663–5668.

[27] Morales, A., Chinellato, E., Fagg, A. H., Pobil, A. P., 2004. An active learning approach for assessing robot grasp reliability. In: Proc. of International Conference on Intelligent Robots and Systems. IEEE, pp. 485–490.

[28] Oztop, E., Bradley, N. S., Arbib, M. A., 2004. Infant grasp learning: a computational model. Experimental Brain Research, 480–503.

[29] Oztop, E., Kawato, M., 2009. Sensorimotor Control of Grasping: Physiology and Pathophysiology. Cambridge University Press, Ch. Models for the control of grasping.

[30] Park, D.-H., Hoffmann, H., Pastor, P., Schaal, S., 2008. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: Proc. of International Conference on Humanoid Robots. IEEE.

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

[32] Rasmussen, C. E., Williams, C. K. I., December 2005. Gaussian Processes for Machine Learning. MIT Press.

[33] Rosenbaum, D. A., 1991. Human Motor Control. Academic Press, San Diego.

[34] Sabe, K., Fukuchi, M., Gutmann, J.-S., Ohashi, T., Kawamoto, K., Yoshigahara, T., 2004. Obstacle avoidance and path planning for humanoid robots using stereo vision. In: Proc. of International Conference on Robotics and Automation. IEEE, pp. 592–597.

[35] Salganicoff, M., Ungar, L. H., Bajcsy, R., 1996. Active learning for vision-based robot grasping. Mach. Learn. 23 (2-3), 251–278.

[36] Saxena, A., Driemeyer, J., Kearns, J., Osondu, C., Ng, A., 2008. Learning to Grasp Novel Objects Using Vision. Vol. 39 of Springer Tracts in Advanced Robotics. Springer, Berlin, Heidelberg, Ch. 4, pp. 33–42.

[37] Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A., 2004. learning movement primitives. In: Proc. of International Symposium on Robotics Research. Springer.

[38] Snelson, E., Ghahramani, Z., 2005. Sparse gaussian processes using pseudo-inputs. In: Proc. of Conference on Neural Information Processing Systems.

[39] Spong, M. W., Hutchinson, S., Vidyasagar, M., 2005. Robot Modeling and Control. John Wiley and Sons.

[40] Srinivas, N., Krause, A., Kakade, S. M., Seeger, M., 2009. Gaussian process bandits without regret: An experimental design approach. Computing Research Repository abs/0912.3995.

[41] Steffen, J., Haschke, R., Ritter, H., 2007. Experience-based and tactile-driven dynamic grasp control. In: Proc. of International Conference on Intelligent Robots and Systems. IEEE, pp. 2938–2943.

[42] Sutton, R. S., Barto, A. G., March 1998. Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). MIT Press.

[43] Tegin, J., Ekvall, S., Kragic, D., Wikander, J., Iliev, B., August 2008. Demonstration based learning and control for automatic grasping. Intelligent Service Robotics, 23–30.

[44] Teytaud, O., Gelly, S., Sebag, M., 2007. Anytime many-armed bandits. In: Proc. of Conference sur l'Apprentissage Automatique. Grenoble, France.

[45] Xue, Z., Kasper, A., Zoellner, J. M., Dillmann, R., July 2009. An automatic grasp planning system for service robots. In: Proc. of International Conference on Advanced Robotics.