Fitted Policy Iteration for a POMDP Peg-In-Hole search task

Guillaume de Chambrier[a,1,*], Aude Billard[a]

[a]*Learning Algorithms and Systems Laboratory (LASA), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland*

Abstract

Acting optimally under state uncertainty is necessary for robotic systems to achieve autonomy. We consider a Peg-In-Hole (PiH) search task in which both a human teacher and robot apprentice must localise an electric socket and establish a connection without vision, making the state space partially observable, and only relying on haptic and proprioceptive information. A search policy can be obtained by applying dynamic programming to the Partially Observable Markov Decision Process (POMDP) formulation of the task. This is infeasible when considering a continuous belief state and action space POMDP as the policy will have a large number of parameters and for which traditional exploration-exploitation strategies would result in a slow learning process.

We address this problem by demonstrating how human intuition can be leveraged in a Programming by Demonstration (PbD) and Fitted Policy Iteration (FPI) framework. We introduce a novel *fitted policy evaluation* and a Monte-Carlo Expectation-Maximisation (MC-EM) *policy improvement* step. A belief space value function is learned offline from trajectory data demonstrated by a group of blindfolded human teachers. The demonstrated location beliefs (Point Mass Filters) are compressed to their most likely state and entropy. Evaluations performed both in simulation and on the KUKA LWR robot showed that the proposed FPI algorithm outperforms both a myopic and a purely data driven policy in terms of distance travelled to localise the socket and when the socket has no distinctive features.

Keywords: Fitted Reinforcement Learning, POMDP, Gaussian Mixture Model, Programming by Demonstration

1. Introduction

The ability to act optimally under state uncertainty is paramount for all robotic systems acting in environments which are not fully observable. Not taking uncertainty into consideration in certain tasks can can lead to wasteful usage of resources and even failure. Given the potential adverse consequences (disastrous if considering a search and rescue task), it is important to design uncertainty robust policies and planners.

The generic solution to such a problem has been typically to formulate the task as a Partially Observable Markov Decision Process (POMDP) which is subsequently solved by dynamic programming or if the transition and observation models are unavailable by reinforcement learning. However, the insurmountable problem of solving a POMDP directly even for the simplest problems [35], has lead to the development of approximate methods.

Advances have been made in applying approximate POMDP algorithms to robotic applications [16], however the optimisation often requires a discretisation of the action space which is restrictive for tasks which are naturally continuous. In this case a local optimisation with quantifiable actions (macro) [49] or alternatively heuristic approaches [29] can be applied. However these approaches require the engineering of actions and are locally optimal.

An alternative approach are Actor-critic (AC) methods [44, Chap. 6.6]. The actor (policy) can be a smooth continuous parametric function ideal for robotics or non discretisable control systems whilst the critic (value function) can be any function approximator. ACs are advantageous over actor only methods (policy search [17]) as the estimated value function reduces the variance of the policy's parameter gradients which results in fast learning [21]. ACs are also advantageous over critic only methods as making the policy the derivative of the value function, even when small approximation errors are present, can lead to sub-optimal greedy policies [4]. The drawback of ACs is that when function approximators are combined with off-policy bootstrapping [44, Chap. 8.5] this can lead to divergence of the value function's parameters [3]. Only when both actor and critic have a linear architecture with respect to the parameters and share the same basis functions (but different parameters) is convergence guaranteed to a local optimum [45] which is restrictive. The divergence problem can be addressed by trading online for offline learning methods such as fitted/batch reinforcement learning.

In this paper we propose a Fitted Policy Iteration (FPI) for large continuous state and action space (PO)MDPs. In this framework the policy (Gaussian Mixture Model) and value functions have different functional representations and the learning is comprised of two steps: *fitted policy evaluation* and *policy improvement*. In fitted policy evaluation a value function is estimated offline by applying multiple passes of the Bellman backup operator over a dataset of episodes until convergence. By separating the learning of the value function from the pol-

*Corresponding author

Email addresses: chambrierg@gmail.com (Guillaume de Chambrier), aude.billard@epfl.ch (Aude Billard)

icy in an on-policy offline framework, the value function cannot diverge [20, 27]. In the policy improvement step, the Gaussian Mixture Model (GMM) parameters are updated to maximise the estimated value function. This is achieved by modifying the traditional GMM Expectation-Maximisation (EM) algorithm [8] by weighting each data point (state-action pair) by the temporal difference error. We refer to this modified EM as Q-EM. The similarity between Q-EM and the GMM-EM makes Q-EM intuitive and easy to implement and avoids the need of a learning rate required in gradient Policy Search methods.

We address the exploration-exploitation dil[illegible]a which affects all reinforcement learning methods by ta[illegible]a Programming by Demonstration (PbD) approach where only episodes demonstrated by human teachers are used in the FPI algorithm. It has been previously shown [15] that humans exhibit both risk-prone and risk-averse behaviour which constitutes an ideal training set (mixture of explorative-exploitative behaviour) for a reinforcement learning problem, removing the need for costly autonomous exploration. We will demonstrate that FPI is able to select the subset of relevant samples form the demonstrations in order to learn an efficient policy.

We consider a plug power-socket search and connection task, also known as Peg-in-Hole (PiH)[40] for the evaluation of the Fitted Policy Iteration method. In this task human teachers demonstrate to a robot apprentice (KUKA LWR) how to localise a power socket and establish a connection. For this task no vision or hearing information is used leaving only haptic information for both the teachers and robot apprentice. The haptic information is made available via a force-torque sensor mounted on the plug. This [illegible]k was chosen as having only haptic information makes it n[illegible]ivial to solve due to the high levels of state uncertainty present. There will be a large variance in behaviour demonstrated by the teachers which is ideal to compare FPI against a pure statistical controller as done in [15]. Finally this task is important to many industrial processes by reducing the cost of having a visual system.

This paper is organised as follows: Section 2 overviews the POMDP model and the Approximiate Dynamic Programming literature. Section 3 details the PiH-search task, the formulation of the belief space and the recorded data. Section 4 presents the Fitted Policy Iteration (FPI) algorithm. Section 5 details the control architecture. Section 6 describes the experiments conducted to evaluate the FPI in the PiH-search task. Section 7, provides a discussion and the conclusion.

2. Background

2.1. POMDP

A Partially Observable Markov Decision Process (POMDP) is a generic framework for formulating a temporal decision process given that the state space is not directly observable [41]. A POMDP is defined by the tuple $\{X, U, Y, \mathcal{T}, \Omega, R, \gamma\}$, where X, U and Y are the state, action and observation spaces (which can be discrete or continuous); $\mathcal{T} := p(x_t|x_{t-1}, u_{t-1})$ is the state transition probability distribution; $\Omega := p(y_t|x_t)$ is the observation model which gives the probability of a measurement $y_t \in Y$ given a state $x_t \in X$; $R(x_t) \in \mathbb{R}$ is the reward function which gives the utility of a state and $\gamma \in (0, 1]$ is the discount factor. As the state space is not observable the agent must consider the entire history $\mathcal{H} := \{y_{0:t}, u_{1:t-1}\}$ of measurements and actions when deciding which action $u_t \in U$ to take [38]. Instead of memorising the entire history $\mathcal{H}$, it can be iteratively integrated into a belief/information state $b_t := p(x_t|\mathcal{H})$, which is a probability distribution over the state space, without losing any information [23]. This leads to a reformulation of the tuple as a *belief*-MDP $\{\mathcal{B}, U, \tau, R_B, \gamma\}$, where $b_t \in \mathcal{B}$ is the set of all possible beliefs and τ is a belief state transition function $b_t = \tau(b_{t-1}, u_{t-1}, y_t)$ which can be any Bayesian state space filter (eg. an Extended Kalman Filter or a Particle Filter). Both the state $\mathcal{T}$ and observation Ω models become part of τ. The belief reward function R_B becomes a function of the state reward function:

$$R_B(b) = \sum_{x \in X} b(x)\, R(x) \tag{1}$$

The advantage of the *belief*-MDP formulation is that dynamic programming and reinforcement learning can be applied as $\mathcal{B}$ is observable. The objective is to find a policy, Equation 2, which maximises the infinite-horizon expected reward, Equation 3, where $r_t \in R_B$,

$$\pi_{\boldsymbol{\theta}} : b \mapsto u \tag{2}$$

$$V^{\pi_\theta}(b) = \mathbb{E}_{\pi_\theta}\left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| b_t = b \right\} \tag{3}$$

which can be iteratively evaluated with on-policy value iteration, Equation 4, and an optimal policy can be found by Generalized Policy Iteration [44, Chap. 4.6].

$$V^{\pi_\theta}(b) = R_B(b) + \sum_{u \in U} \pi_{\boldsymbol{\theta}}(b, u)\, \gamma \sum_{y \in Y} p(y|b, u)\, V^{\pi_\theta}(\tau(b, u, y)) \tag{4}$$

Solving a discrete state space POMDP problem is difficult as the number of parameters of the value function grows exponentially with respect to the decision horizon [47, Chap. 15][41], a result of preserving the Piece Wise Linear and Convex (PWLC) property of the value function. Recent advances for discrete POMDPs are Point-based Value Iteration (PBVI) methods [35], which focus on pruning, belief selection and exploration strategies, see [48, 18]. However these Value Iteration (VI) approaches do not naturally transfer to continuous domains [36]. An alternative approach is to represent the value function by a non-parametric function, parameterize the belief space and perform approximate dynamic programming.

2.2. Approximate dynamic programming

Approximate Dynamic Programming (ADP) methods[13] for solving (PO)MDPs, also known as fitted or batch reinforcement learning, do not utilise the PWLC property of the value

function and instead directly estimate the value function's parameters from a dataset of state-value samples. Once the policy or value function is computed a new batch of state-value samples are generated using the current policy and the process repeats until convergence. The ADP approach to solving a (PO)MDP is much simpler to understand when compared with PBVI as there is no need to implement pruning heuristics.

ADP has grown in popularity as it addresses stability issues when function approximators are combined with traditional online off-policy methods such as Q-learning, see [50][Chap. 2]. It was first proved by [20] that offline fitted value iteration is stable given that the function approximator belongs to the family of averages. The authors successfully learned the mountain car value function (locally weighted averaging) in which the state space was comprised of two consecutive images of the mountain car's simulation, resulting in a 2048 dimensional state space. In [11] the authors demonstrated that when traditional regression techniques (neural networks, locally weighted regression or linear regression) were naively combined with dynamic programming the results often lead to the value function diverging in common reinforcement learning problems. By separating the Bellman backup operator and the fitting of the regressor function, similarly as in [20], they demonstrated that locally weighted regression could be combined with dynamic programming to learn the mountain car's policy. The extension of the results given in [20] to a model-free approach with a kernel function approximator known as Kernel-Based Approximate Dynamic Programming (KBDP) [33] was proven to be globally optimal in a continuous-space framework. The guaranteed convergence of value function approximators have lead to an increase in the application of the ADP methodology to reinforcement learning. It is used for instance to control a UAV with fitted value iteration[10], and to other reinforcement learning architectures such as Fitted Natural Actor Critic (FNAC) [30] and Fitted Q-Iteration (FQI) [19, 32]. Even non averager regressor functions have been successfully combined with dynamic programming in the fitted reinforcement learning framework such as Neural Fitted Q-Iteration (NFQI) [37] which uses a multi-layer perceptron to represent the Q-function and has since been used in many extensions, [34, 1]. This has lead to the application of more sophisticated regression methods such as Deep Fitted Q-iteration (DFQ) [28, 31, 22].

An early successful example of the application of the ADP concept to a POMDP problem was Monte Carlo POMDP (MC-POMDP) [46]. A Q-value function (k-nearest-neighbour, which is an averager), is learned online whilst using experience replay (synonym for fitted/batch RL). The beliefs are represented by a particle filter (a point cloud of possible locations of the robot) and the KL-divergence between two Gaussian functions fitted to the particle sets is the distance metric used to compute the similarity between two point clouds in k-nearest neighbour (knn). This approach requires sampling the possible actions, performing forward simulation of the particle filter for each of the sampled actions and sampling the possible future measurements which all in all is computationally expensive.

In [12] a POMDP policy for a 2D navigation problem is learned with fitted value iteration. The belief states are the parameters of a Gaussian function and they are recursively updated by an Extended Kalman Filter. The authors first precalculate the state transition and observation models. Forward simulation and the Bellman backup operator is applied to compute the resulting belief state targets. The value function is then fitted to the belief parameter target value pairs with a Freudenthal triangulation function approximator. This model based fitted value iteration approach is suitable when actions can be sampled during the computation of the maximisation in Value Iteration (VI) which we consider infeasible in our case. When controlling a robotic system on-policy methods such as actor-critics are more suitable for the reasons we outlined in the introduction.

Approximate Policy Iteration (API) [7, 6] (part of ADP) is suited to actor-critic architectures. An early application in a batch setting was the Least-Squares Policy Iteration (LSPI) [27] where the associated Q-value function is represented by a linear weighted sum of radial basis functions and a policy is learned to control a simulated bike. Although being an API approach, the value function and control policy do not have a separate architecture and the Q-value function has to be linear in the parameters [45]. In [24] the authors introduce a non-parametric API approach for learning a vision to discrete action policy in which a Q-function (a regression Tree) is learned for each separate action. To the author's knowledge there are very few if any examples in which API as been applied to a continuous state and action space in which the policy and value function have distinct functional architectures.

This paper proposes a Fitted Policy Iteration approach suited to continuous state and action space. The main difference of our approach with the literature is within the policy improvement step. We introduce a Monte-Carlo Expectation Maximisation (MC-EM) algorithm tailored for generative distribution policies such as the Gaussian Mixture Model. The policy evaluation step resembles the fitted reinforcement learning approach in the literature with the difference being that the value function is learned instead of the Q-function. We do not give a review of EM-MC approaches but refer the reader to [17][p. 51] (Algorithm 11) which has inspired the FPI approach.

3. Methodology

3.1. PiH-search task

Figure 1 (*Top-right*), illustrates the PiH-search task. The orange area represents the teacher's starting area and is assumed prior knowledge. The sockets are always positioned at the center of a fake wall (wooden plank) which is clamped to a table. We consider one type of plug, Type J[1], and three different power sockets. Power *socket A*, has a ring around its holes, *socket B* has a funnel, which we hypothesize should make it easier to connect, and *socket C* has a flat elevated surface. See Figure 1 (*Top-left*) for an illustration.

To perform the PiH search tasks we recruited 10 student volunteers to be teachers (all male Master's and PhD students).

[1] http://www.iec.ch/worldplugs/typeJ.htm

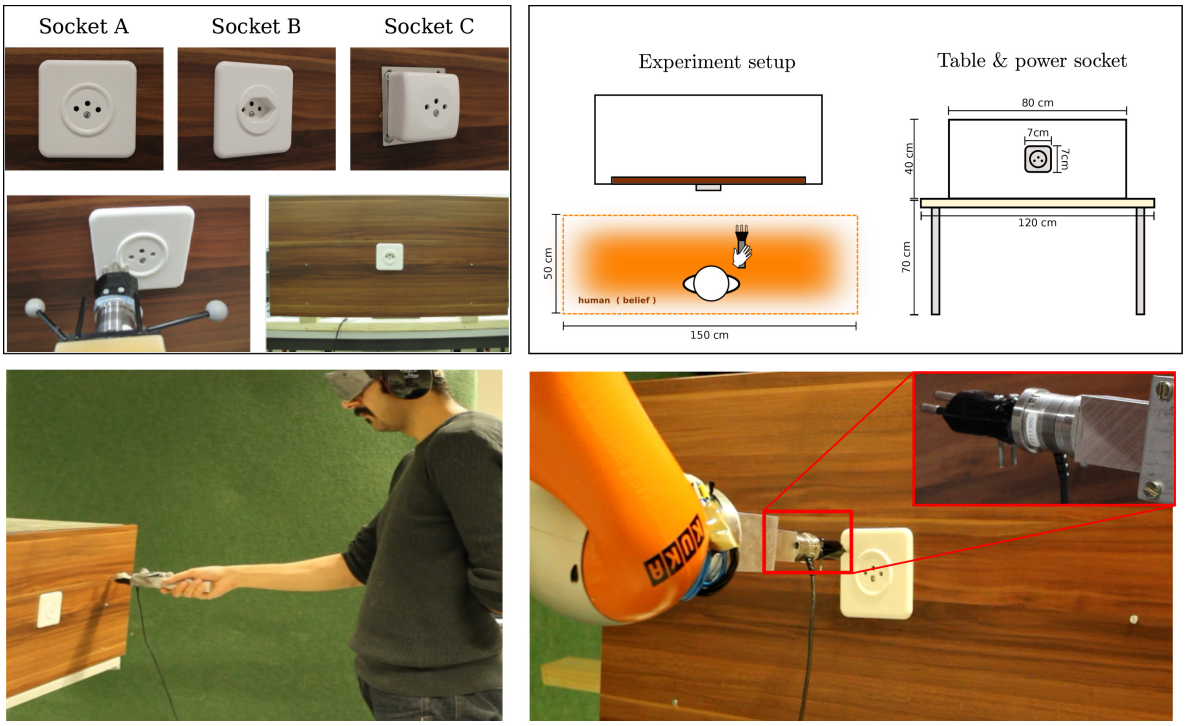


Figure 1: **Peg-in-Hole search task setup**. *Top-left*: Three different sockets are used, socket A will be only used to gather training data whilst socket B and C will be used for evaluation purposes. *Top-right:* Dimensions of the the wall and socket, the orange area illustrates the possible locations in which the human teacher will start the search. *Bottom-left:* A participant (human teacher) is blindfolded and placed within the orange rectangular area always facing the wall. He is holding a cylinder equipped with a peg and an ATI force torque sensor and OptiTrack markers. See Video 1 for an illustrate of a human subject performing the search task. *Bottom-right:* The KUKA LWR robot is equipped with a peg holder mounted with an ATI force torque sensor, it is reproducing a search and connection policy learned from the human demonstrations. See Video 2 for an illustration of the KUKA searching for the socket and then establishing a connection.

The participants were aged between 24 and 30 with an average age of 26 years and a standard deviation of 2.4 years. Each participant carried out 30 demonstrations of the PiH search-task and each session lasted approximately 50 minutes and never exceeded one hour. The 10 participants were divided equally in two groups, A and B. Each member of group A began by performing 15 PiH searches with socket A (AA), followed by a 10 minute break, finishing with an additional 15 searches with socket B (AB). The members of group B performed the same protocol starting with socket B (BB) and ending with socket A (BA). Figure 2 summarises a walk through of the experiment. The only exclusion criteria was the inability of the subject to accomplish the task. All participants gave written consent for taking part in this study. A total of 300 demonstrations were gathered. Figure 1 (*Bottom-left*) illustrates a human subject performing the search whilst blindfolded and wearing ear defenders and in Figure 1 (*Bottom-right*) the KUKA LWR robot is carrying out the PiH task that it learned from human demonstrations.

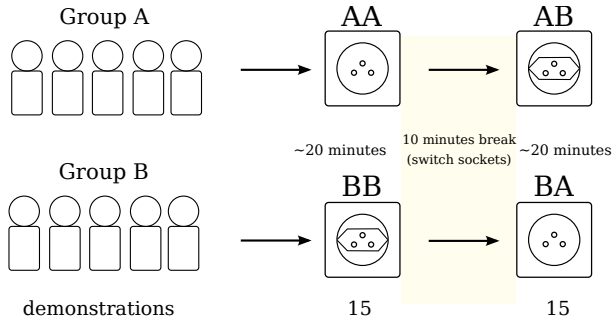


Figure 2: Experiment protocol. The participants are divided in two groups of 5, Group A begins with socket A and after a short break repeats the task with socket B. The same logic holds for Group B. For each socket 15 executions of the task are recorded.

3.2. Data collection

The human teacher holds the plug which is attached to a cylindrical handle with an ATI 6 axis force torque sensor (Nano25 [2]) which provides wrench $\phi \in \mathbb{R}^6$ measurements. We define the measurement used in the observation model Ω to be a binary multivariate feature vector which is a function of the wrench, $y = h(\phi)$. The feature vector encodes whether a contact is present and the direction in which it occurs, which is discretized to the four cardinalities.

On top of the cylinder there is a set of markers used by a motion capture system OptiTrack[3] (which has millimeter tracking accuracy) to measure both linear, $u \in \mathbb{R}^3$, and angular velocity, $\omega \in \mathbb{R}^3$, at each time step which is recorded at a rate of 100 Hz

[2] http://www.ati-ia.com/products/ft/sensors.aspx

[3] http://www.optitrack.com/

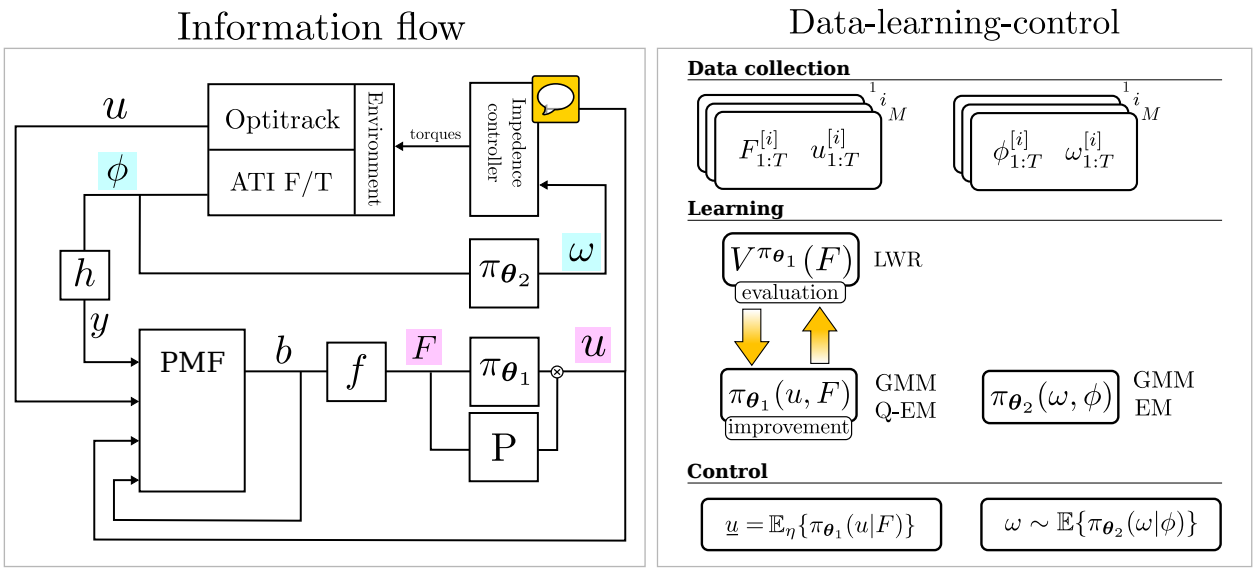


Figure 4: *Left:* The Point Mass Filter is updated with velocities $u \in \mathbb{R}^3$, from the Optitrack vision system when human demonstrations are being recorded and otherwise from the policy π_{θ_1} when the robot is performing the task. The sensed wrench $\phi \in \mathbb{R}^6$ is transformed to a multivariate binary feature vector y which is used in the measurement model of the PMF state estimator. The filtered probability density, b is compressed to a lower dimensional feature vector F before being given as input to the linear velocity policy π_{θ_1}. The second policy π_{θ_2} outputs angular velocities $\omega \in \mathbb{R}^3$ given a sensed wrench ϕ and is used during the insertion stage of the PiH. The output of the linear velocity policy π_{θ_1} is modulated by a position-force and proportional controller, denoted by the letter P in the figure. *Right:* Data pipeline. A set of M demonstrations are recorded and split into two datasets. The left dataset is used to learn policy π_{θ_1} with the Fitted Policy Iteration framework and the dataset on the right is used to directly learn the statistical policy π_{θ_2}. The desired linear search direction is obtained from a re-weighted expectation of the conditional $\pi_{\theta_1}(u|F)$ whilst angular velocities are sampled from the conditional of π_{θ_2}.

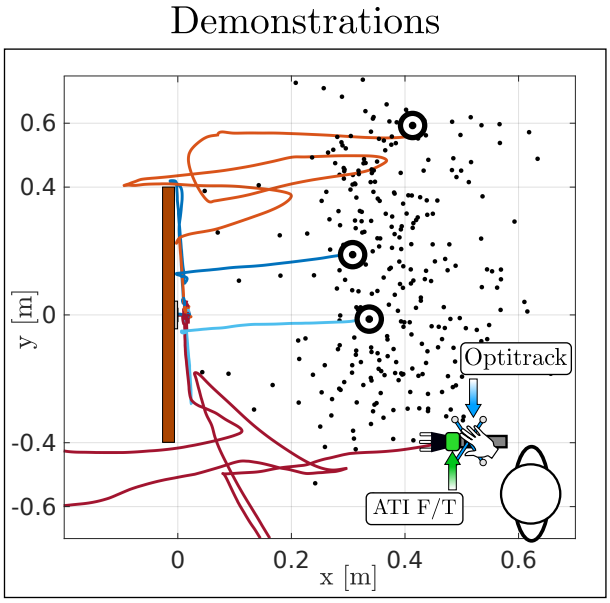


Figure 3: Black points represent the starting position of the end-effector for all the demonstrations. Four trajectories are illustrated.

along with the F/T information. See Figure 3 for an illustration of demonstrated trajectories from human teachers.

The location belief of the human teacher and robot apprentice is represented by a probability density function (pdf) $b_t := p(x_t|y_{0:t}, u_{1:t})$ which is represented by a Point Mass Filter (PMF) [5, p.87]. PMF is chosen to represent the believed location of the plug as the sensing likelihoods are non-gaussian and lead to multi-modal distributions. Both u and y are used in the state transition and measurement model defined in the POMDP. Figure 5 (*Left*) illustrates different time segments of the location belief recording during a demonstration. Figure 5 (*Right*) illustrates the output of the observation model Ω when an edge is sensed.

As the belief is high dimensional it is impractical to directly learn a statistical policy $\pi_\theta : b \mapsto u$ without some form of compression. We apply a compression function $f : b \mapsto F$ which transforms the belief to a lower dimensional feature vector $F = [\hat{x}, U]^{\mathrm{T}}$ which is composed of the Most Likely State (MLS) $\hat{x} := \arg\max_x b_t \in \mathbb{R}^3$ and differential entropy $U := H(b_t) \in \mathbb{R}$. Figure 4 (*Left*) gives an overview of the information flow and which variables are recoded.

The demonstrations of all the participants are combined into a dataset $D = \{u^{[i]}_{1:T}, F^{[i]}_{1:T}, \omega^{[i]}_{1:T}, \phi^{[i]}_{1:T}\}_{i=1:M}$ comprised of M episodes,where the upper index $[i]$ references the ith search trajectory (one episode) and subscript $1:T$ denotes the time steps during the trajectory from initialisation $t = 1$ until the end $t = T$. In Figure 4 (*Right* Data collection) we summarise the dataset, gathered from the human teachers, split into two subsets which will be used to learn two different control policies.

4. Learning

Two policies are learned from the dataset D acquired from the human teachers demonstrations: one maps a belief feature vector to linear velocity $\pi_{\theta_1} : F \mapsto u$ and the other maps a sensed wrench to angular velocity $\pi_{\theta_2} : \phi \mapsto \omega$. Both linear

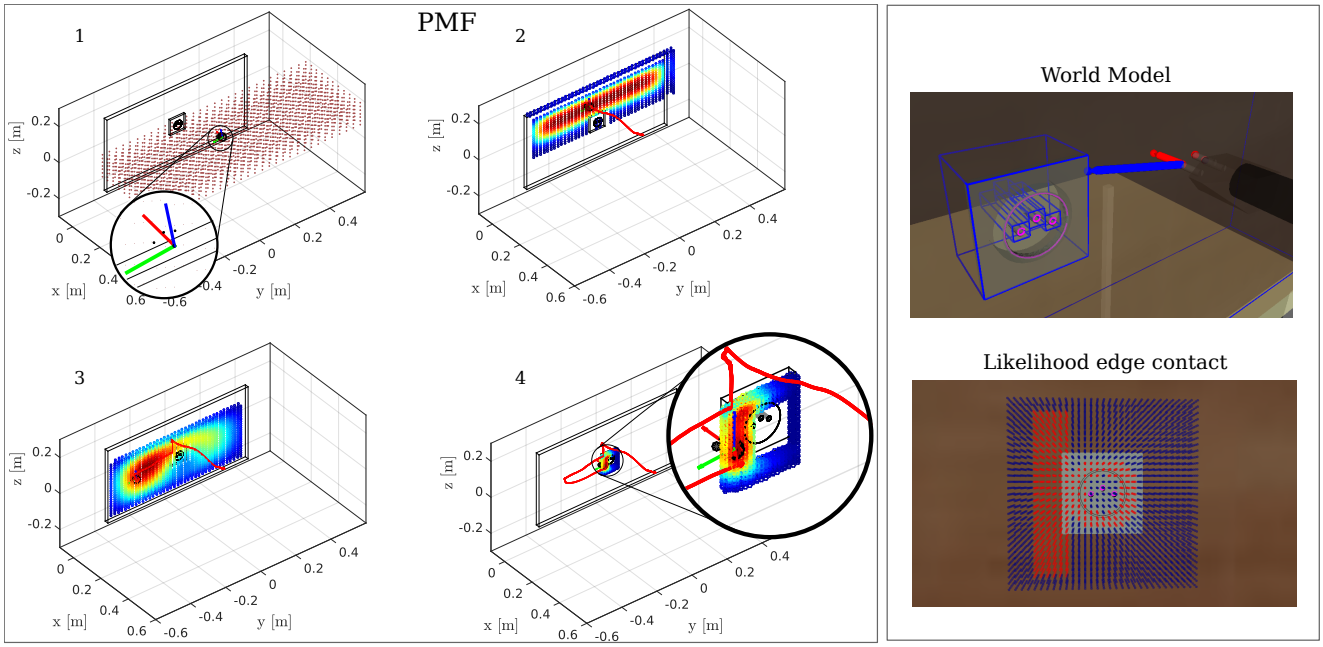


Figure 5: *Left:* Point Mass Filter (PMF) update of a particular human demonstration. (1) Initial uniform distribution spread over the starting region. Each grid cell represents a hypothetical position of the plug. The orientation is assumed to be known. (2) First contact, the distribution is spread across the surface of the wall. The red trace is the trajectory history. (3) motion noise increases the uncertainty. (4) The plug is in contact with a socket edge. See Video 3 for an illustration of the PMF for a subject's search. *Right*: **World model**: The plug is modelled by its three plug tips and the wall and sockets are fitted with bounding boxes. **Observation model**: The plug enters in contact with the left edge of the socket. As a result, the value of the likelihood in all the regions close the left edge take a value of one (red points) whilst the others have a value zero (blue points) and areas around the socket's central ring have a value of one.

and angular velocity policies are parameterised by a Gaussian Mixture Model (GMM), Equation 5.

$$\pi_{\boldsymbol{\theta}}(\xi_1, \xi_2) = \sum_{k=1}^{K} w^{[k]}\, g(\xi_1, \xi_2; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}) \tag{5}$$

The parameters $\boldsymbol{\theta} = \{w^{[k]}, \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}\}_{1:K}$, are the weights, means and covariances of the individual Gaussian functions $g(\cdot)$

$$\boldsymbol{\mu}^{[k]} = \begin{bmatrix} \boldsymbol{\mu}^{[k]}_{\xi_1} \\ \boldsymbol{\mu}^{[k]}_{\xi_2} \end{bmatrix} \qquad \boldsymbol{\Sigma}^{[k]} = \begin{bmatrix} \boldsymbol{\Sigma}^{[k]}_{\xi_1\xi_1} & \boldsymbol{\Sigma}^{[k]}_{\xi_1\xi_2} \\ \boldsymbol{\Sigma}^{[k]}_{\xi_2\xi_1} & \boldsymbol{\Sigma}^{[k]}_{\xi_2\xi_2} \end{bmatrix}$$

and $\sum_k w^{[k]} = 1$. For the linear policy π_{θ_1}: $\xi_1 = u$ and $\xi_2 = F$. For the angular policy π_{θ_2}: $\xi_1 = \omega$ and $\xi_2 = \phi$. In both cases the Bayesian Information Criterion is used to determine the number of Gaussian functions.

The parameters of the linear velocity policy π_{θ_1} are learned in our Fitted Policy Iteration framework whilst the parameters of the angular velocity policy π_{θ_2} are directly learned from the demonstrations following the methodology outlined in [26, Chap. 5]. It is important in the final stage of the task to consider applying angular rotations to the plug as the small clearance between the socket's holes and plug can cause jamming.

4.1. Fitted Policy Iteration

Fitted Policy Iteration is an on-policy offline reinforcement learning method which allows both value function and policy to converge to their optimal parameters (with respect to the reward function) through through successive iterations of policy *evaluation* and *improvement*.

Policy evaluation. A value function of the belief state feature vector F is learned from the trajectory data provided by the teachers. Given the continuous nature of the belief feature vector Locally Weighted Regression (LWR) [2] is used as a function approximator of the value function. To learn the value function a Fitted/Batch RL approach is taken[19]. An offline method applies multiple sweeps of the Bellman backup operator followed by fitting the value function through regression to a dataset of tuples $\{(F_t^{[i]}, r_t^{[i]}, F_{t+1}^{[i]})\}_{i=1:M}$ until the Bellman residual converges, see Algorithm 1.

Algorithm 1: Fitted Policy Evaluation

input : $\{(F_t^{[i]}, r_t^{[i]}, F_{t+1}^{[i]})\}_{i=1:M}$
output: $\hat{V}^{\pi}(F)$

while $\|\hat{V}^{\pi}_{k+1}(F) - \hat{V}^{\pi}_{k}(F)\| > \epsilon$ **do**
  $\hat{V}^{\pi}_{k+1}(F) = \text{Regress}(F_t,\, r_t + \gamma \hat{V}^{\pi}_{k}(F_{t+1}))$

Most Fitted RL methods have focused on learning the Q-value function directly (Fitted Q-Iteration) [32, 19, 37]. Although this solves the control problem it requires discretisation of the action space or assumes quantifiable actions, so that maximisation in the Q-Bellman backup $\max_{u_{t+1}} \hat{Q}(u_{t+1}, F_{t+1})$ is easily achievable. Given that FPI has to be applicable to non discretisable continuous control problems we opt for an on-policy approach.

Policy improvement. The actor is updated given the critic's value function through a modification of the Maximisation step

in Expectation-Maximisation (EM) for Gaussian Mixture Models. This modification is referred to as Q-EM which is strongly related to a Monte-Carlo EM-based policy search approach [17, p.50]. To highlight the difference in our approach we derive the Q-EM policy improvement step from first principals.

The reward of a demonstrated episode R_e is the sum of discounted one-step rewards r, Equation 6,

$$R_e(e_i) = \sum_{t=0}^{\mathrm{T}^{[i]}} \gamma^t \, r(u_t^{[i]}, F_t^{[i]}) \tag{6}$$

where $e_i = \{(u_0, F_0), \ldots, (u_T^{[i]}, F_T^{[i]})\}$ are the state-action samples of the ith episode. All policy gradient approaches seek to find a set of parameters $\boldsymbol{\theta}$ of the actor, which will maximise the expected reward, Equation 7,

$$J(\boldsymbol{\theta}) = \mathbb{E}_{p_\theta}\{R_e\} = \sum_{i=1}^{M} \underbrace{\left(\prod_{t=0}^{T^{[i]}} \pi_{\boldsymbol{\theta}}(u_t^{[i]}, F_t^{[i]}) \right)}_{p_{\boldsymbol{\theta}}(e_i)} R_e(e_i) \tag{7}$$

where $p_{\boldsymbol{\theta}}(e_i)$ is the probability of episode e_i. We differ with [17] by considering this probability to be the product of step-wise joint distributions $\pi_{\boldsymbol{\theta}}(u_t^{[i]}, F_t^{[i]})$ instead of conditionals $\pi_{\boldsymbol{\theta}}(u_t^{[i]}|F_t^{[i]})$. This is a valid decomposition of $p(e_i)$ and will result in an intuitive algorithm to maximise the parameters of the policy with respect to a value function. By setting the derivative of Equation 7 to zero, the parameters which maximise the cost function $\arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ can be found. This cannot be done directly, instead the logarithmic lower bound of the cost function Q is maximised which results in Equation 8. See [17, p.50] for a detailed derivation.

$$\nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{i=1}^{M} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(u_t^{[i]}, F_t^{[i]}) \, Q^{\pi_{\boldsymbol{\theta}'}}(u_t^{[i]}, F_t^{[i]}) \tag{8}$$

In the above equation, $\boldsymbol{\theta}'$ are the parameters used to generate the trajectories during the E-step. In our case these are the initial demonstrations provided by the teachers. In most policy search approaches the policy is conditioned on the state space, $\pi_{\boldsymbol{\theta}}(u|F)$. This would lead to a complex expression in the maximisation of Equation 8 and is restrictive in the case of the GMM as it fixes the state space parameters $\boldsymbol{\mu}_F^{[k]}$, $\boldsymbol{\Sigma}_{FF}^{[k]}$ (and partially $\boldsymbol{\Sigma}_{uF}^{[k]}$) and thus greatly constrains the solution. Equation 8 differs with [17, p.51] by the fact that we are using the joint distribution $\pi_{\boldsymbol{\theta}}(u, F)$ instead of the conditional distribution $\pi_{\boldsymbol{\theta}}(u|F)$, a result of maximising Equation 7. This has two benefits. Firstly, the input dimensions (the state space) are no longer fixed allowing the GMM basis functions to move and secondly the optimisation of GMM parameters is very similar to that of the traditional EM. Setting the derivative of Equation 8 to zero and solving for the parameters $\boldsymbol{\theta} = \{w, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ a new weighted Maximisation EM step is obtained which we refer to as Q-EM, see Figure 6.

The Q-EM algorithm depends on the $Q^{\pi_\theta}(u, F)$ function, however it is feasible to replace it with the advantage function $A^{\pi_\theta}(u, F)$ whose evaluation can be obtained from the gradient

$$\boldsymbol{\mu}_{\mathbf{new}}^{[k]} = \frac{\sum_{j=1}^{M} \alpha_k(\mathbf{x}^{[j]}) \, Q^{\pi_{\theta'}}(\mathbf{x}^{[j]}) \, \mathbf{x}^{[j]}}{\sum_{j=1}^{M} \alpha_k(\mathbf{x}^{[j]}) \, Q^{\pi_{\theta'}}(\mathbf{x}^{[j]})}$$

$$\boldsymbol{\Sigma}_{\mathbf{new}}^{[k]} = \frac{\sum_{j=1}^{M} \alpha_k(\mathbf{x}^{[j]}) \, Q^{\pi_{\theta'}}(\mathbf{x}^{[j]}) (\mathbf{x}^{[j]} - \boldsymbol{\mu}_{\mathbf{new}}^{[k]})(\mathbf{x}^{[j]} - \boldsymbol{\mu}_{\mathbf{new}}^{[k]})^{\mathrm{T}}}{\sum_{j=1}^{M} \alpha_k(\mathbf{x}^{[j]}) \, Q^{\pi_{\theta'}}(\mathbf{x}^{[j]})}$$

$$w_{\mathbf{new}}^{[k]} = \frac{\sum_{j=1}^{M} Q^{\pi_{\theta'}}(\mathbf{x}^{[j]}) \, \alpha_k(\mathbf{x}^{[j]})}{\sum_{j=1}^{M} Q^{\pi_{\theta'}}(\mathbf{x}^{[j]})}$$

Figure 6: Q-EM Maximisation of the GMM parameters. We used the same notation and derivation as in [9, Chap. 9.2.2], where $\alpha_k(\mathbf{x}^{[j]})$ is the responsibility factor, denoting the probability that data point $\mathbf{x}^{[j]} = [u^{[j]}, F^{[j]}]^{\mathrm{T}}$ belongs to Gaussian function k.

of the value function which is equal to the Temporal Difference (TD) error, see Equation 9.

$$A^{\pi_\theta}(u_t, F_t) = Q^{\pi_\theta}(u_t, F_t) - V^{\pi_\theta}(F_t) = \delta_t^{\pi_\theta} \tag{9}$$

By assuming that the estimated value function $\hat{V}^{\pi_\theta}$, obtained in the *fitted policy evaluation* step, is close to the true value function V^{π_θ}, the TD error δ^{π_θ} is an unbiased estimate of the advantage function. Using the advantage function as means of policy search is popular with methods such as Natural Actor Critic (NAC) [34].

By substituting Q for the TD, each jth state-action sample has an associated weight, $\delta^{\pi_\theta} \in \mathbb{R}$, where $\delta^{\pi_\theta} > 0$ means that the jth state action-pair lead to an increase in the value function and $\delta^{\pi_\theta} < 0$ lead to a decrease in the value function. The data log-likelihood is re-weighted accordingly, giving more importance to data points which lead to a gain. Since the Q-EM update steps cannot allow negative weights, the TD error is rescaled to lie between 0 and 1.

2D Fitted Policy Iteration example. To illustrate the mechanism of Fitted Policy Iteration, we give a 2D example of its application, see Figure 7. The *Top-left* subfigure depicts 10 trajectories demonstrated by two teachers going from start (white circle) to goal (orange star) state. The optimal path is a straight line passing in between the two obstacles. Neither teacher demonstrated the optimal straight path.

In the *Bottom-left*, a GMM is fitted $\pi_{\boldsymbol{\theta}}(\dot{x}, x)$ to the teachers' data, using the standard EM-algorithm. Taking the policy to be the output of Gaussian Mixture Regression (GMR) $\mathbb{E}\{\pi_{\boldsymbol{\theta}}(\dot{x}|x)\}$ different behaviours are obtained than those demonstrated by the human teachers. The GMR averages the different modes

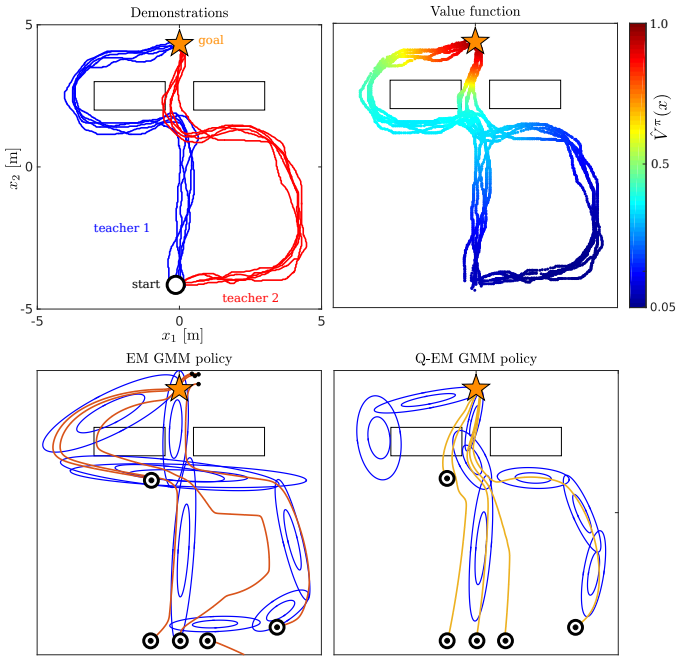


Figure 7: Fitted policy evaluation & improvement example. *Top-left:* The goal of the task is to reach the goal state. The first teacher (blue) demonstrates five trajectories which contour the obstacle in front of the goal. The second teacher (red) demonstrates 5 trajectories which initially deviate from the goal before passing between the two obstacles. *Bottom-left:* The EM algorithm is used to fit a GMM to the teachers' original data. The marginal $\pi_{\boldsymbol{\theta}}(x)$ is plotted in blue and trajectories generated by the policy $\mathbb{E}\{\pi_{\boldsymbol{\theta}}(\dot{x}|x)\}$ in orange. *Top-right Policy Evaluation:*. Value function after fitted policy evaluation terminated, the reward function is binary, $r_T = 1$ at the goal and zero otherwise, and a discount factor $\gamma = 0.99$ is used. *Bottom-right Policy Improvement:* the GMM is learned with the Q-EM algorithm in which each data point's weight is proportional to the advantage function.

encoded by the Gaussian functions which results in a mixing of the original demonstrated behaviours. No trajectories of the GMR policy truly replicate the demonstrated behaviour.

In the *Top-right* subfigure, we apply fitted policy evaluation to the original demonstrated data (discount factor $\gamma = 0.99$ and reward $r_T = 1$ when the goal is reached and zero otherwise) and compute the value function.

The *Bottom-right* subfigure illustrates the GMM policy learned with the Q-EM algorithm. As the temporal difference error δ^{π_θ} is highest along the start-goal axis, data points following this gradient will have a higher weight. This results in a policy with better rollouts (closer to the optimal path) compared with the trajectories generated by the policy learned via standard EM.

Belief state fitted policy evaluation. FPI is applied to episodes from dataset D which are examples of the PiH search with socket A. First $\hat{V}^{\pi_{\theta_1}}(F)$ is learned before using it in the policy improvement step to learn $\pi_{\boldsymbol{\theta}_1}$. For the PiH search task a reward of $r = 0$ is assigned at each time step until the goal (plug-socket connection) is achieved, where a reward of $r_T = 100$ is assigned. We also learn a linear search policy $\pi_{\boldsymbol{\theta}_1}$ by using EM-GMM as previously done in [15], which will serve as means of comparison against the policy learned via Q-EM. The angular velocity policy $\pi_{\boldsymbol{\theta}_2}$ is learned with the traditional EM-GMM algorithm as we do not consider state uncertainty in the orientation of the plug. Figure 8 (*Left*) illustrates the value function of the most likely state. As expected, the value function is highest close to the socket and around the axis $z = 0$ and $y = 0$. When Q-EM is applied the Gaussian functions of the GMM will favour these locations.

Figure 8 (*Middle*-right) illustrates the best and worst trajectories in terms of the accumulated value function. It can be seen that the best trajectories (red) tend to be aligned with the socket, whilst the worst trajectories are towards the edges of the wall and tend to follow spiralling motions.

In conclusion we learned two linear policies, one solely from the original human demonstrations which we call GMM and the second which is the result of **one iteration** of Fitted Policy Iteration which we call Q-EM. This ensures that both policies are given the same amount of information.

Figure 4 (*Right* Learning) summaries the learning process for both policies. For both GMM and Q-EM the angular velocity policy $\pi_{\boldsymbol{\theta}_2}$ is the same.

5. Control

The direction to search is given by the conditional, Equation 10,

$$\pi_{\boldsymbol{\theta}_1}(u|F) = \sum_{k=1}^{K} w_{u|F}^{[k]}\, g(u; \boldsymbol{\mu}_{u|F}^{[k]}, \boldsymbol{\Sigma}_{u|F}^{[k]}) \tag{10}$$

which is a distribution over the possible velocities. The function $g(\cdot)$ is a multivariate Gaussian function parameterised by mean $\boldsymbol{\mu}_{u|F}^{[k]} \in \mathbb{R}^{(3\times1)}$ and Covariance $\boldsymbol{\Sigma}_{u|F}^{[k]} \in \mathbb{R}^{(3\times3)}$. The subscript $u|F$ indicates that the parameters are the result of the conditional. The reader is referred to [14], [43] for a detailed derivation of the conditional of a GMM. The learned model is multi-modal, as different search velocities are possible in the same belief state. Figure 9 illustrates the multi-modal vector fields of the conditional, Equation 10. In autonomous dynamical systems control, the velocity is obtained from the expectation of the conditional, Equation 10. However, the expectation is a weighted combination of the modes which could result in no displacement as result of cancellation. To address this, we use a modified version of the expectation operator which favours the current direction, Equation 11 - 12.

$$\eta_k(\underline{u}) = w_{u|F}^{[k]} \cdot \exp(-\cos^{-1}(<\underline{u}, \boldsymbol{\mu}_{u|F}^{[k]}>)) \tag{11}$$

$$\underline{u}_t = \mathbb{E}_\eta\{\pi_{\boldsymbol{\theta}_1}(u|F)\} = \sum_{k=1}^{K} \eta_k(\underline{u}_{t-1}) \cdot \boldsymbol{\mu}_{u|F}^{[k]} \tag{12}$$

The GMM policy $\underline{u} = \mathbb{E}_\eta\{\pi_{\boldsymbol{\theta}_1}(u|F)\} \in \mathbb{R}^{(3\times1)}$ outputs a linear velocity which is normalised (denoted by the under bar). When the applied velocity mode is no longer present another direction is sampled. For example, when the robot enters in contact with a feature, greatly reducing the uncertainty, the current

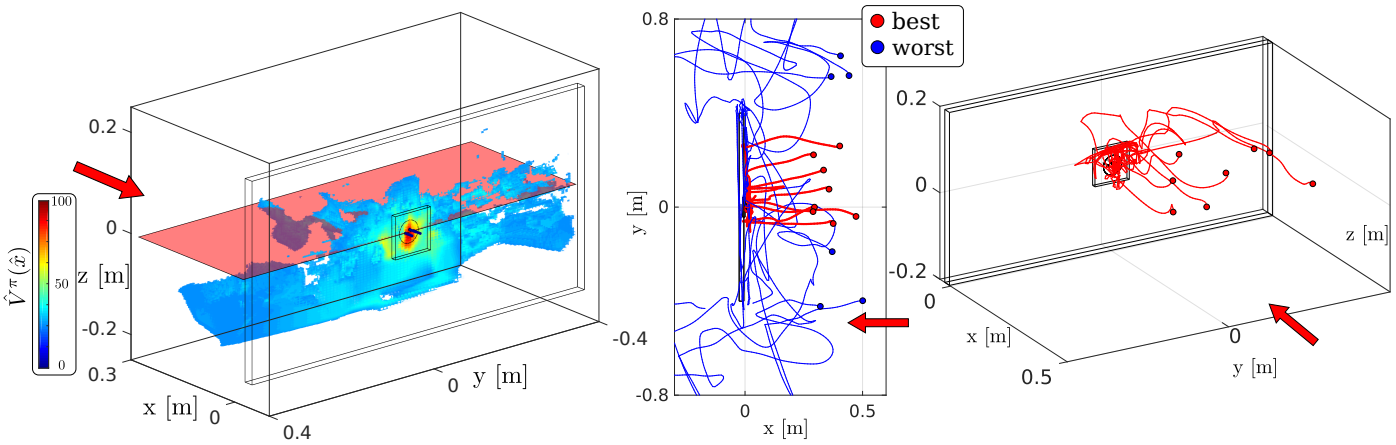


Figure 8: *Left*: LWR value function approximate $\hat{V}^{\pi_{\theta_1}}(\hat{x})$ for the most likely state $\hat{x}$. The red plane is to help visualise where the value function is above and below the axis $z = 0$. Only states with values above 0.25 are plotted. The red arrow indicates the heading of the human teacher when performing the search task. The discount factor was $\gamma = 0.99$ and the variance of the kernel variance of 1 [cm], which was set experimentally. *Middle-right*: Best and worst trajectories. The red demonstrated trajectories are the best in terms of the amount of value function gain whilst the blue are the worst. The red arrow indicates the teacher's heading. The blue trajectories tend towards the sides of the wall as the initial starting position is on the border of the wall. The red trajectories are centred along the y-axis of socket and tend to move in a straight line towards the wall whilst aligning themselves with the axis $z = 0$.

mode changes and a new search direction is computed. Figure 4 (*Right* Control) summaries how both linear and angular velocities are generated from π_{θ_1} and π_{θ_2}.

5.1. Robot Implementation

This search task is haptic and the end-effector of the robot is often in contact with the environment. To make the robot compliant with the environment we use an impedance controller in combination with a hybrid position-force controller. The hybrid controller targets a sensed force $\mathcal{F}_x$, in the x-axis, of 3N. The y and z velocity components of the direction vector are given by Equation 12. This is insufficient for the robot to reliably surmount the edges of the socket, hence the vector field of the GMM is modulated in y and z-axis, Equation 13.

$$\underline{u}_p = R_y(c(\mathcal{F}_z) \cdot \pi/2) \cdot R_z(c(\mathcal{F}_y) \cdot \pi/2) \cdot \underline{u} \qquad (13)$$

where R_y and R_z are (3×3) rotation matrices around the y and z-axis, and $c(\mathcal{F}) \in [-1, 1]$ is a truncated scaling function of the sensed force. When a force $\mathcal{F}_z$ of 5N is sensed, a rotation of $R_y(\pi/2)$ is applied to the original direction resulting in the robot getting over the edge. The direction velocity is always normalised up to this point. The amplitude of the velocity is a proportional controller based on the believed distance to the goal. Figure 4 (*Left*) illustrates the proportional controller and hybrid force-controller by P which multiples the normalised linear velocity $\underline{u}$ by Equation 13 and by a scaling factor which is a function of the believed distance to the goal results in the velocity u which is applied at the end-effector of the robot.

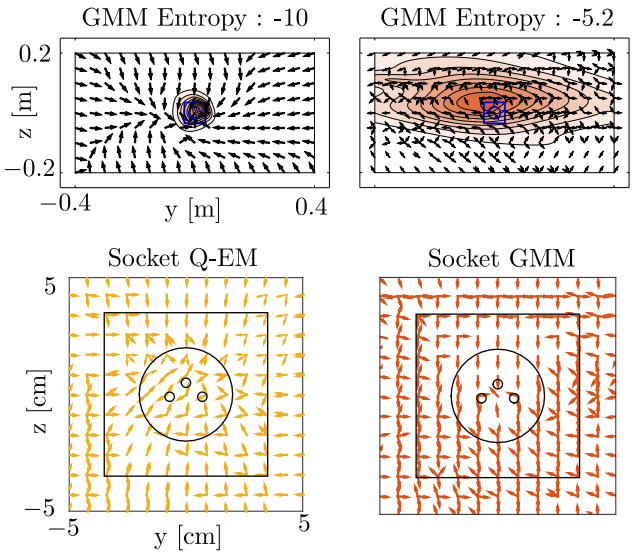


Figure 9: Q-EM and GMM policy vector fields. *Top*: The GMM policy is conditioned on an entropy of -10 and -5.2. For the lowest entropy level, most of the probability mass is close to the socket area since this level corresponds to very little uncertainty; we are already localised. We can see that the policy converges to the socket area regardless of the location of the believed state. For an entropy of -5.2 we can see that the likelihood of the policy is present across wall. The vector field directs the end-effector to go towards the left or right edge of the wall. *Bottom*: The entropy is marginalised out, the Q-EM being the yellow vector field and the GMM being the orange. The Q-EM vector field tends to be closer to a sink and there is less variation.

6. Results

We evaluate the following three aspects:

1. **Distance taken to accomplish the goal** (connect plug to socket). We compare the Q-EM policy with a GMM policy

learned through standard EM and a myopic Greedy policy. This highlights the difference between complicated and simplistic search algorithms and gives an appreciation of the problem's difficulty.

2. **Importance of data** provided by human teachers. We evaluate whether it is possible to learn an improved GMM policy from Greedy demonstrations. This policy which we call Q-Greedy is used to test whether indeed human demonstrations are necessary. We evaluate whether it is possible to obtain a good policy from the two worst teachers' demonstrations as not all teachers are necessarily proficient at the task in question.
3. **Generalisation**. We learn a policy to insert a plug into socket A which is located at the center of a wooden wall. We test the generalisation of the policy in finding a new socket location and whether the policy can generalise to sockets B and C, which were not used during the training phase.

We evaluate aspects 1) and 2) purely in simulation as finding the socket requires much less precision than establishing a connection and the physics of the interaction is simple. Aspect 3), the generalisation, is evaluated both in simulation, up to the point of localising the socket's edge, and on the KUKA LWR robotic platform for the connection phase of the task. The main reason for employing the robot is that the connection phase dynamics is complex and a simulation would be unrealistic. For the robot evaluation we consider the search starting already within the vicinity of the socket.

6.1. Distance taken to reach the socket's edge

We consider two search experiments which we refer to as **Experiment 1** and **2**, in order to evaluate the performance in terms of the distance travelled to reach the socket for the three search policies: GMM, Q-EM and Greedy. In these two experiments the task is considered accomplished when a search policy finds the socket's edge.

Experiment 1, three starting locations are chosen: *Center*, *Left* and *Right*. See Figure 10 *(Experiment 1 Top-left)*, for an illustration of the initial condition. This setup tests the effect of the starting positions. A total of 25 searches are carried out for each of the search policies. The trajectory results show a clear difference between the trajectories generated by the GMM and Q-EM policies (*Experiment 1 Bottom-left*). The orange GMM policy trajectories go straight towards the wall, whilst the yellow Q-EM policy trajectories drop in height making them closer to the socket. *Experiment 1 Bottom-right*, illustrates the distribution of the first contact with the wall for the *Center* initial condition. The distribution of the first contact of the Greedy method is uniform across the entire y-axis of the wall. It does not take into account the variance of the uncertainty. In contrast, the GMM policy remains centred with respect to the starting position and the Q-EM is even closer to the socket and there is much less variance in the location of the first contact.

Experiment 1 Top-right, illustrates the quantitative results of the distance taken to reach the socket for all three experiments. For the *Center* initial condition, the Q-EM policy travels far less than the other search policies. Considering that the initial position of the search is 0.45 [m] away from the wall, the Q-EM policy finds the socket very quickly once contact has been established with the wall. For the *Right* and *Left* starting conditions both the GMM and Q-EM policies travel less distance to reach the socket, with a smaller variance when compared with the Greedy search policy.

Experiment 2, Figure 10 (*Experiment 2*), the initial true starting positions of the end-effector are taken from a regular grid, within the red cube (see *Experiment 1*), covering the whole start region, also used as the initial distribution for the human demonstrations. A total of a 150 searches are carried out for each of the three policies. This experiment compares the search policies with the human teachers' demonstrations. The Human and GMM show similar distributions of searched locations. They cover the upper region of the wall and top corners, to some extent. These distributions are not identical for two reasons. The first is that the learning of the GMM is a local optimisation which is dependent on initialisation and number of parameters. The second reason is that the synthesis of trajectories from the GMM is a stochastic process.

For the Q-EM policy, the distribution of the searched locations is centred around the origin of the z-axis. The uncertainty is predominantly located in the x and y-axis. The Q-EM policy takes this uncertainty into consideration by restraining the search to the y-axis regardless of the starting position. The uncertainty is reduced when it is in the vicinity of the socket. The Greedy's policy search distribution is multi-modal and centred around the z-axis where the modes are above and below the socket. This shows that the Greedy policy acts according to the most likely state which changes from left to right of the socket, because of motion noise, resulting in left-right movements and little displacement. As a result the Greedy policy spends more time at these modes.

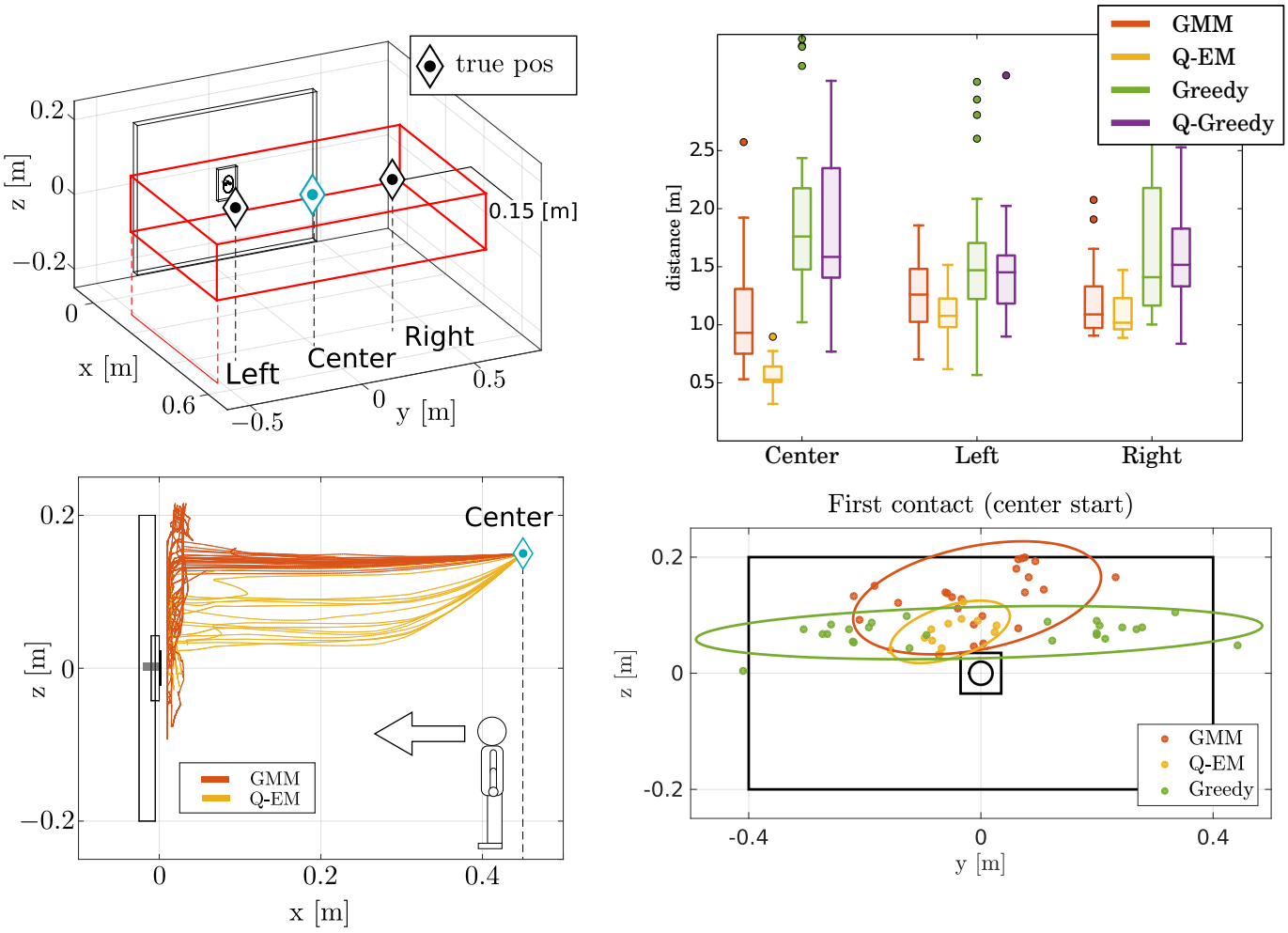
Experiment 2 Right, it is clear that all three search policies travel less to find the socket's edge compared with the teachers' demonstrations. All search policies are better than the human teachers with the exception of group BA, which is performing the task with socket A. The Q-EM policy remains the best.

We have shown that under three different experimental settings the Q-EM algorithm is predominantly the best in terms of distance taken to localise the socket. The GMM policy learned solely from the data provided by the human teachers also performs well in comparison with the human teachers and Greedy policy. A critical assumption was made however in order to be able to use this statistical policy approach. This **assumption** is that a human teacher is proficient in accomplishing the task. If a teacher is not able to accomplish the task in a repetitive and consistent way so that a search patter can be encoded by the GMM, the learned policy will perform poorly. Next we evaluate the validity of this assumption and the importance of the training data provided by the human teachers.

6.2. Importance of data

Two tests were performed to evaluate the importance of the teachers training data, referred to as **Experiment 3**. The worst two teachers in terms of distance taken to find the socket's edge

Experiment 1



Experiment 2

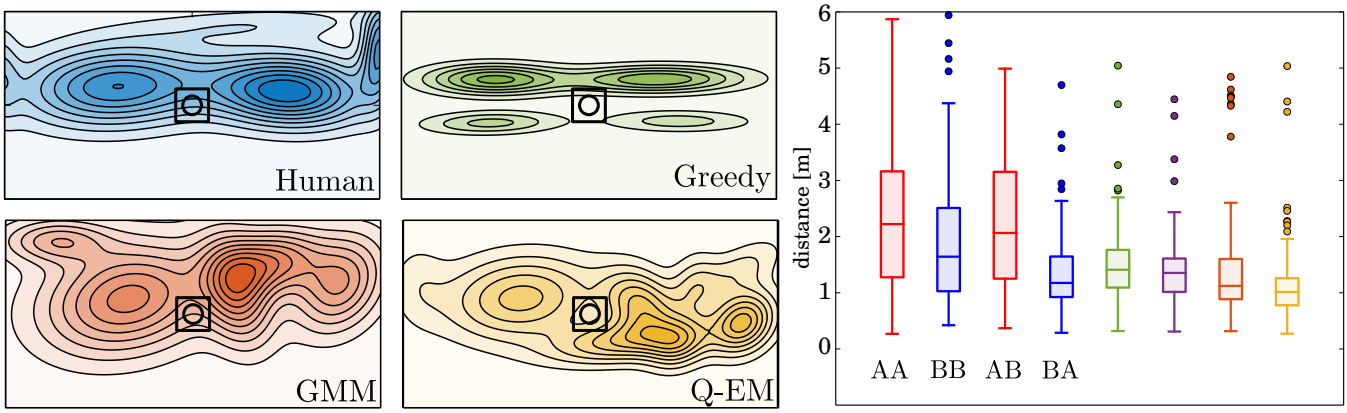


Figure 10: Two simulated search experiments. **Experiment 1:** *Top-left*: Three start positions are considered: *Left*, *Center* and *Right* in which the triangles depict true position of the end-effector. The red cube illustrates the extent of the uncertainty. *Bottom-left*: Trajectories of both the GMM (orange) and Q-EM (yellow) policies. For each start condition a total of 25 searches were performed for each search policy. *Bottom-right:* Distribution of first contact point giving the center initial starting condition. *Top-right*: Distribution of distance travelled until the socket's edge was localised. The Q-EM policy is always the best. **Experiment 2**: *Left*: Distribution of the visited regions during the search for the socket's edge. The Q-EM policy's distribution is more centred along the axis $z = 0$. *Right:* Time taken to find the socket, the search algorithms are better than the humans with the exception of group BA.

are used to learn a GMM and Q-EM policy separately, to evaluate whether it is possible to learn a successful policy given a few bad demonstrations (15 training trajectories for each policy). In the second test a noisy explorative Greedy policy was used as a teacher to gather demonstrations which in turn were then used to learn a new policy, which we call Q-Greedy.

Figure 11 (*Top-left*) illustrates 6 trajectories of teacher # 5. Once localised, the teacher would reposition himself in front of the socket and to try to achieve an insertion. This behaviour was not expected since by losing contact with the wall, the human teacher no longer has the sensory feedback necessary to maintain an accurate position estimate.

Figure 11 (*Bottom-left*) illustrates the value function of the belief state learned from the data of teacher # 5. The states with the highest value seem to create a path going from the socket towards the right edge of the wall. As before, to learn a GMM policy is learned from the raw data to produce a Q-EM policy in which the data points are weighted by the gradient of the value function. *Experiment 3 Middle-column* illustrates the resulting Marginalised Gaussian Mixture parameters for both the GMM and Q-EM policies where 25 rollouts are plotted of each policy starting at the *Center* initial condition also used in Experiment 1. It can be seen that the trajectories of the GMM policy have much greater variance in contrast to the Q-EM policy, resulting from an excess of variance in the 15 original demonstrations given by the teacher. Too much variance is not necessarily good, a random (uniform) policy in terms of generated trajectories will have the most variance and is as expected extremely inefficient in achieving a goal. Furthermore there is insufficient data to encode a pattern for the GMM model. In contrast, the Q-EM finds a pattern by combining multiple parts of the available data and as a result fewer data points are necessary to achieve a good policy. This effect is clear in Figure 12, showing the performance of the GMM and Q-EM algorithms under the same initial conditions as in Experiment 1. For all the conditions and for both teachers #5 and #7 the Q-EM policy always does better than the GMM.

We also tested whether we could use the Greedy policy as a means of gathering demonstrations in order to learn a value function and train a Q-Greedy policy. The Q-Greedy policy was used in combination with random perturbations applied to the velocity to act as a simple exploration technique. A maximum of 150 searches were performed, which terminated once the socket was found. These demonstrations were used to learn a value function and GMM policy which we refer to as Q-Greedy. Figure 10 *Experiment 1-2 (bar plot)*, illustrates the statistical results of the Q-Greedy policy for Experiment 1 and 2 (purple bar chart), showing that there is no difference between the two policies. This exploration method is probably too simplistic to discover meaningful search patterns and we could probably devise better search strategies which would result in a better policy. However this study has shown that human behaviour already does have a usable trade-off between exploration and exploitation which can be used to learn a new policy through our Fitted Policy Iteration framework.

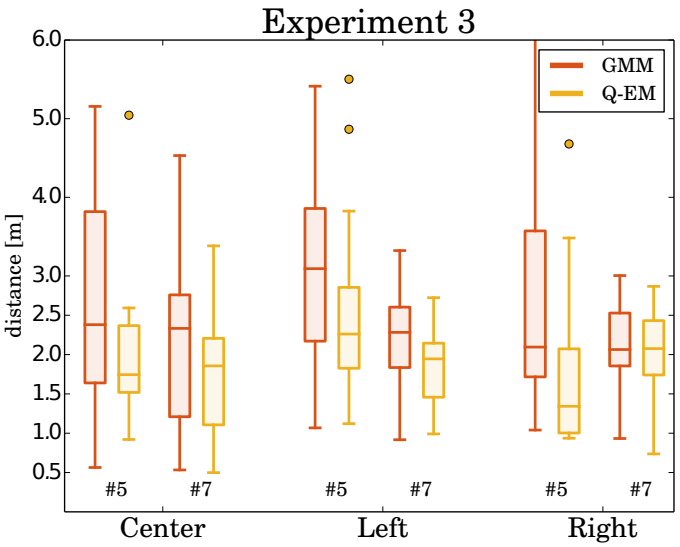


Figure 12: Distance taken to reach the goal for the GMM and Q-EM policies when trained with the worst two teachers. The initial starting conditions are as in Experiment 1. The Q-EM policy nearly always does much better than the GMM policy for both when trained with data from subject #5 or #7.

6.3. Generalisation

So far we have trained and evaluated our policies within the same environment. To test whether the GMM and Q-EM policies can generalise to a new setting the socket was moved to the upper right corner of the wall. The GMMs were trained in the frame of reference of the socket and when we translated the socket's location it also translated the policy.

The same initial conditions of Experiment 1 were used with an additional new configuration named *Fixed*, in which both the true and believed location are fixed, blue triangle and circle. Figure 13 *(Left)* illustrates the trajectories of the three search policies for the *Fixed* initial condition. The Greedy policy moves in a straight line towards the top right corner of the table. As the true position is to the right, it takes the Greedy policy longer to find the wall in contrast with both the GMM and Q-EM policies. From the statistical results shown in Figure 13 *(Right)* we can see that for the *Fixed* and *Right* initial condition, which are similar, both GMM and Q-EM are better. However, for the *Center* and *Left* initial condition this is no longer the case. The Greedy method is better under this condition since the socket is close to informative features (it is located close to the edges of the wall). Once the end-effector has entered in contact with the wall the actions of the Greedy policy always result in a decrease of uncertainty, which was not the case when the socket was located in the center of wall. Thus in both the *Fixed* and *Right* initial condition the Greedy method does worse because it takes longer to find the wall.

The GMM based policies are still able to generalise under different socket locations. In general, as the socket's location is moved further from the original frame of reference in which it was learned, the higher is the likelihood that the search quality degrades. We chose the upper right corner since it is the furthest point from the origin and the GMM and Q-EM policies were still able to find the socket. We note that the policy will

Experiment 3

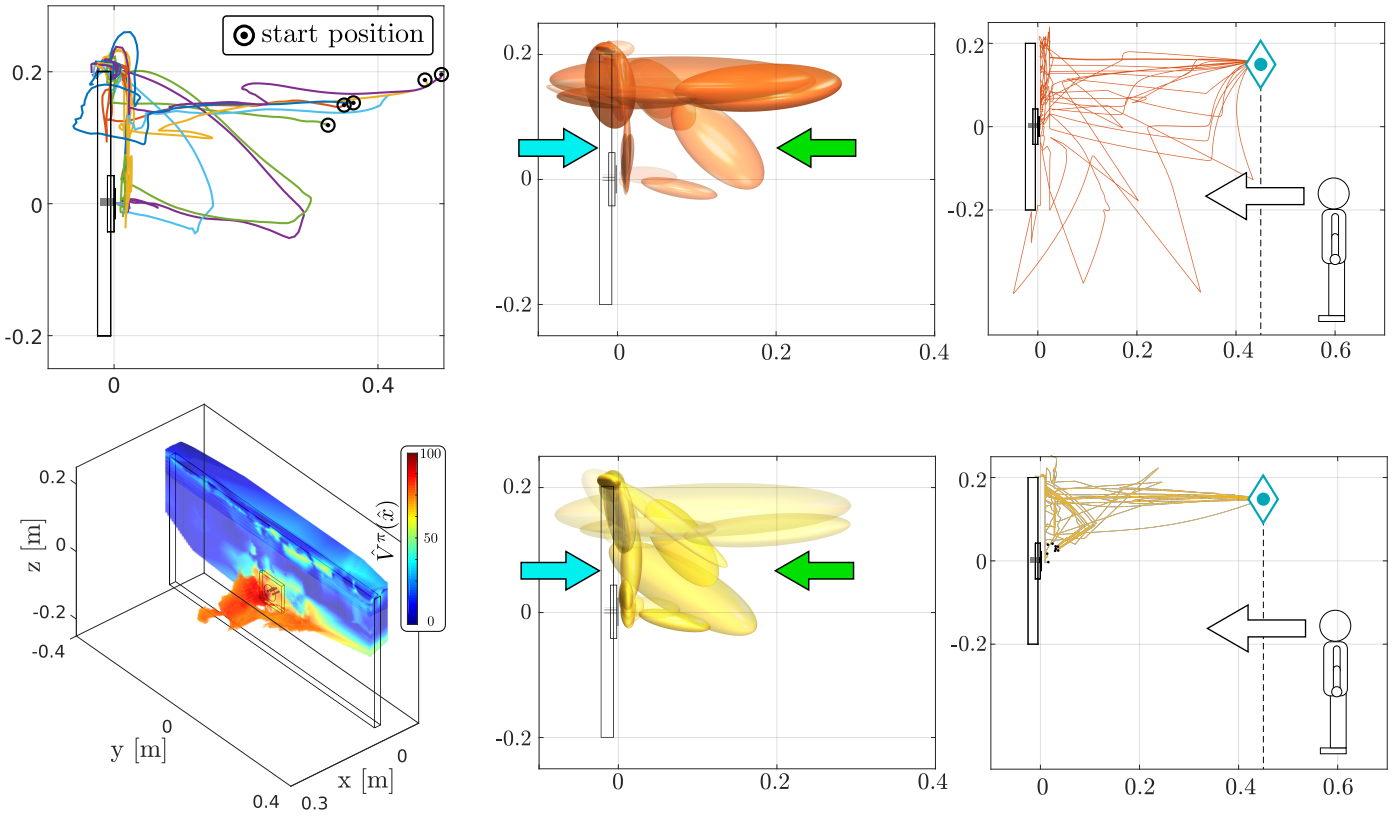


Figure 11: **Experiment 3** *Top-left*: Demonstrations of teacher # 5. *Bottom-left* Value function learned from the 15 demonstrations of teacher #5. The value of the most likely state is plotted. *Middle-column*: Most likely state parameters of the GMM and Q-EM learned from the demonstrations of teacher #5. *Right-column*: Rollouts of the policies learned from teacher #5. We can see that trajectories from the GMM policy have not really encoded a specific search pattern, whilst the Q-EM policy gives many more consistent trajectories replicating to some extent the pattern of making a jump (no contact with the wall) from the top right corner to the socket's edge.

Experiment 4

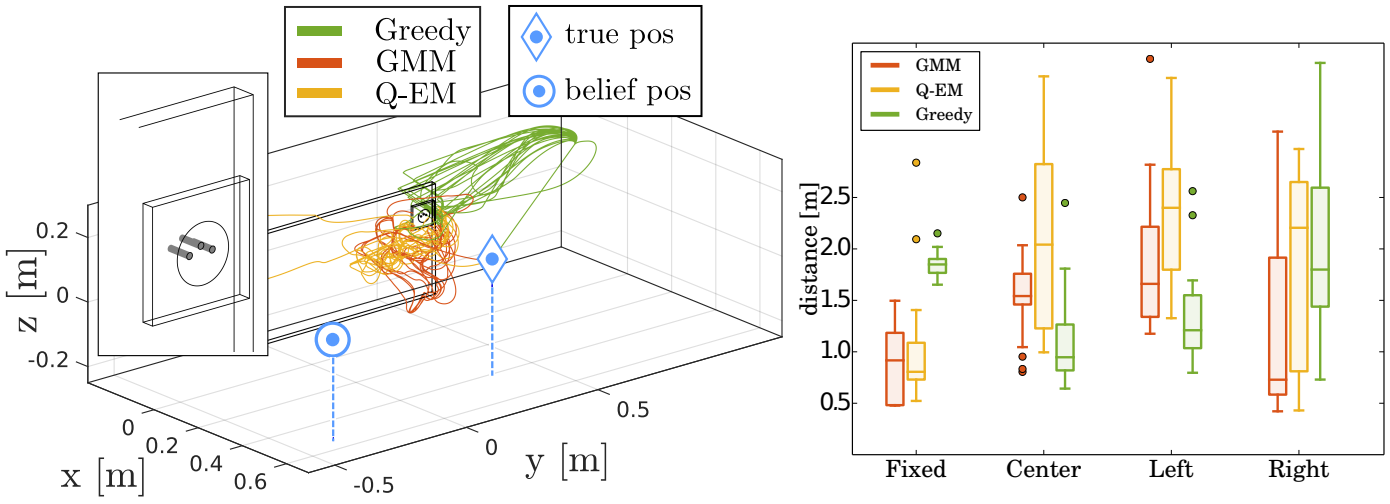


Figure 13: **Experiment 4** Evaluation of generalisation. The socket is located at the top right corner of the wall. We consider a *Fixed* starting location for both the true and believed locations (most likely state $\hat{x}_t$) of the end-effector. The red square depicted in Figure 10 is the extent of the initial uniform uncertainty. *Right:* Distance taken to reach the socket's edge for four initial starting conditions, left, centre and right of Experiment 1 and the fourth is the fixed condition as previously described. For the Fixed setup both the Q-EM and GMM significantly outperform the Greedy.

always be able to find the socket once it has localised itself. This can be seen from the vector field of the GMM policy when the uncertainty is low, see Figure 9 on page 9. In this case the policy is a sink function with a single point attractor.

6.4. Distance taken to connect the plug to the socket

This section evaluates the distance taken for the policies and humans to establish a connection, after the socket has been found. The distance is measured from the point that the plug enters in contact with the socket's edge until the plug is connected to the socket. All the following evaluations are done on a KUKA LWR4 robot. The robot's end-effector is equipped with a plug holder on which is attached a force-torque sensor, the same holders used during the human teachers' demonstrations. In this way both the teacher and robot apprentice share the same sensory interface.

We chose to have the robot's end-effector located to the right of the socket and a belief spread uniformly along the z-axis. See Figure 14 for an illustration of the initial starting condition. This initial configuration was used to evaluate the search policies for the three different sockets, see Figure 1 on page 4 for an illustration of the sockets. The same initial configuration for the evaluation of the three sockets was kept in order to observe the generalisation properties of the policies. Note that only the training data from demonstrations acquired during the search with socket A were used. Socket B has a funnel which should make it easier to connect whilst socket C should be more difficult as it has no informative features on its surface.

For each of the sockets 25 searches were performed starting from the same initial condition. In Figure 14 *(Left)* we plot the trajectories of each of the search methods for socket A. The GMM reproduces some of the behaviour exhibited by humans, such as first localising itself at the top of the socket before trying to attempt to make a connection. The Q-EM algorithm exhibits less variation than the GMM and tends to pass via the bottom of the socket to establish a connection. The Greedy method in contrast is much more stochastic since it does not take into consideration the variance of the uncertainty but instead tries to directly establish a connection. In Figure 14 *(Right)* illustrates a typical rollout of the GMM search policy for both socket A and C. Once a contact is made with the socket's edge the policy tends to stay close to informative features and tends to wander vertically up and down. Only when the uncertainty has been reduced does the GMM policy go towards the socket's connector.

The GMM and Q-EM policies are able to generalise to both socket B and C, as the geometric shape and connector interface of the two sockets are similar to socket A. The local force modulation of the policy's vector field, which is not learned, allows the end-effector to surmount edges and obstacles whilst trying to maintain a constant contact force in the x-axis. This modulation makes it possible for the plug to get on top of socket C.

Figure 15 (a) illustrates the statistics of the distance taken to establish a connection for all three sockets. For socket A both the Greedy and Q-EM are better than the GMM and the Q-EM has less variance in comparison to the Greedy searches. All three search methods are vastly superior, when compared to the human's performance see Figure 15 (b-c).

The interesting point is that both the GMM and Q-EM algorithms perform better than the Greedy approach for socket C. Socket C has no informative features on its surface and as a result myopic policies such as the Greedy policy will perform poorly. However for socket A and B, the Greedy policy performs better as both of these sockets have edges around their connector point allowing for easy localisation. It can also be seen that most search methods perform better on socket B than A, since the funnel shape connector helps in maintaining the plug within the vicinity of the socket's holes. We found that the insertion policy $\pi_{\theta_2}(\omega|\phi)$ only helped with the insertion when the wall was artificially rotated such that a direct insertion was not possible. Otherwise the insertion was always successful 100% of the time. In other more industrial applications the insertion step is more problematic than when considering an electric socket, see [26, Chap. 5] for further details on the jamming dilemma.

The discrepancy between the humans performance and the search policies can be attributed to many causes. One plausible reason is that the PMF probability density representation of the belief is more accurate than the human teachers' position belief. Also, the motion noise parameter was fixed to be proportional to the velocity and the robot moves at gentle pace ($\sim$ 1 cm/s) as opposed to some of the human teachers. In actuality, humans are far less precise than the KUKA which has sub-millimetre accuracy.

7. Discussion & Conclusion

In this work we introduced a Fitted Policy Iteration (FPI) algorithm for learning continuous state and action (PO)MDPs with a large number of parameters. By changing the objective function (Equation 7) so that the probability of a rollout is the product of individual state-action joint probabilities instead of conditionals, this has lead to an intuitive policy improvement step. For the case of a GMM, the improvement of its parameters is equivalent to a weighted EM maximisation of the data log-likelihood. This makes FPI intuitive to implement for the GMM and is also applicable to any generative model policy. The advantage of FPI over other methods is that it allows the basis functions to move and adapt in the input space. This is not the case for other policy architectures such as Dynamic Movement Primitives (DMP) [42] which directly model the conditional $\pi_{\boldsymbol{\theta}}(\dot{x}|x)$ and as a result, fix the input dimensions x of the basis functions.

The performance of FPI was evaluated on a PiH search task in which only haptic information is available making it non-trivial to solve efficiently. Two Gaussian Mixture Model policies were learned from the data recorded during the human teachers' demonstrations. The Q-EM policy was learned by FPI in which a belief space value function was estimated. The value function was then used to weight training datapoints in the M-step update of Expectation-Maximisation (Q-EM). The GMM policy was learned using the standard EM algorithm, and it considered all training data points equally, following in

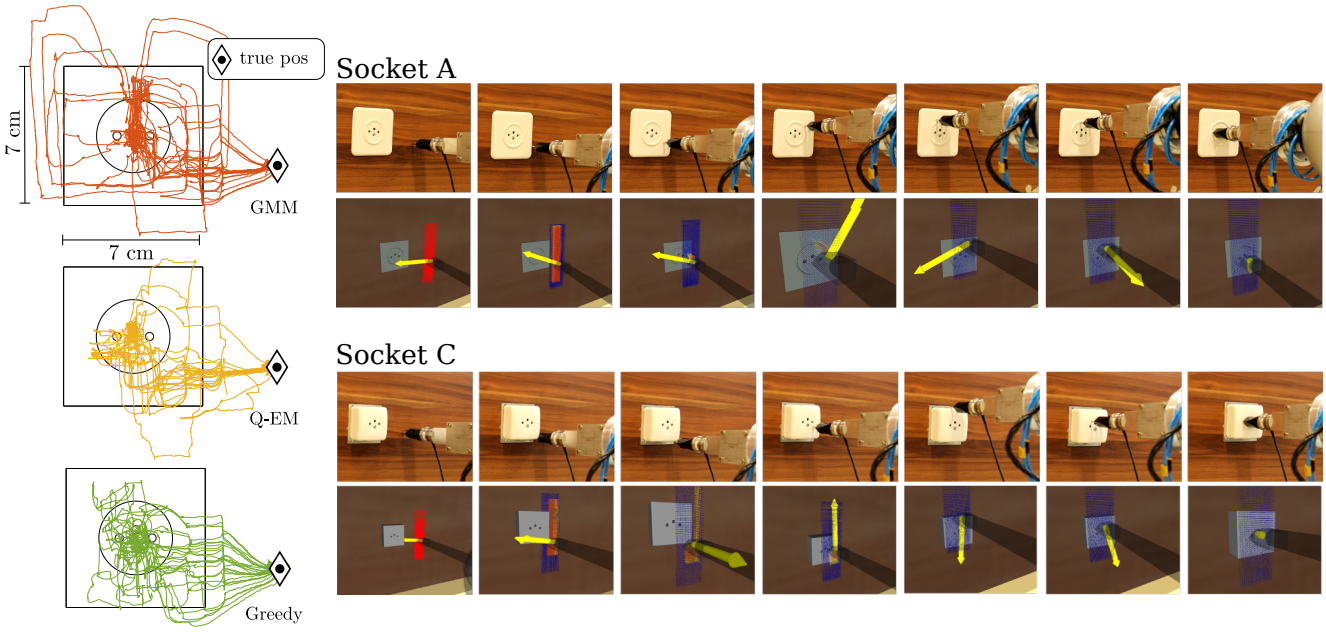


Figure 14: *Left*: 25 search trajectories for each of the three search policies for socket A. *Right* KUKA LWR4 equipped with a holder mounted with a ATI 6-axis force-torque sensor. *Socket A*: The robot's end-effector starts to the right of the socket. The second row shows screen captures taken of ROS Rviz data visualiser in which we see the Point Mass Filter (red particles) and a yellow arrow indicating the direction given by the policy. In this particular run, the plug remained in contact with the ring of the socket until the top was reached before making a connection. *Socket C*: Same initial condition as for socket A. The policy leads the plug down to the bottom corner of the socket before going the center of the top edge, localising itself, and then making a connection. See Video 4 and Video 5 for an illustration of the KUKA establishing a connection to socket B and C.

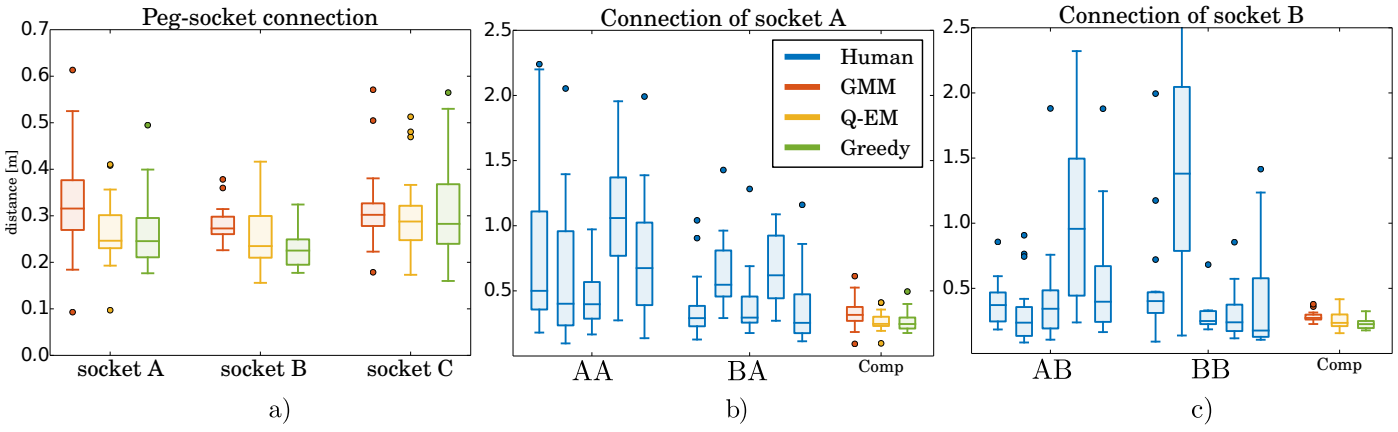


Figure 15: Distance taken to connect the plug to the socket (a) The Q-EM algorithm is the best for both socket A and C. For socket C, the Greedy algorithm does worse than the other two. This is because socket C has no informative features. (b) Group AA are the set of teachers who first started with socket A. They had no previous training on another socket beforehand. Group BA first gave demonstrations on Socket B before giving demonstrations on Socket A. Group BA is better than Group AA at doing the task. This is most likely a training effect. However all policy search methods are far better at connecting the plug to the socket. (c) Both Groups AB and BB are similar in terms of the distance they took to insert the plug into the socket, the search policies on the other hand travel less to accomplish the task.

the footsteps of our initial approach [15]. Both policies were trained with data solely from the human demonstrations of the search with socket A.

In nearly all experiments we demonstrated that the policy learned with FPI (Q-EM) always does better than the policy learned with EM. The benefit was made clear when data from the two worst teachers was used by both policies. The qualitative evaluation showed clearly that the Q-EM policy managed to extract a search pattern, which was not the case for the GMM policy. A Q-EM policy was also learned from the data provided by a Greedy policy with explorative noise and no improvement was found. From these results we conclude that the exploration and exploitation aspects of the trajectories provided by the human teachers is necessary.

We have shown that by simply adding a binary reward function in combination with data provided by human demonstrations in an on-policy offline fitted reinforcement learning framework a policy can be learned without the need of expensive exploration-exploitation rollouts traditionally associated with reinforcement learning. This is especially advantageous when only a few demonstrations are available.

In terms of future work it would be straightforward to combine Q-EM with SEDS [25] which would ensure that the GMM policy has a stable global attractor point. This would guarantee that at every iteration of FPI, the agent following the policy would always reach the goal. Another important area of research which has not received widespread attention is the development of an efficient belief compression technique, to improve on the traditional first and second moment approximations. Approaches such as E-PCA [39] require the discretisation of the state space and as such are not directly applicable to continuous state space problems.

[1] Agostini, A., Celaya, E., July 2010. Reinforcement learning with a gaussian mixture model. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8.
URL http://dx.doi.org/10.1109/IJCNN.2010.5596306

[2] Atkeson, C., Moore, A., Schaal, S., 1997. Locally weighted learning. Journal of Artificial Inteligence, 11–73.
URL http://dx.doi.org/10.1023/A:1006559212014

[3] Baird, L., 1995. Residual algorithms: Reinforcement learning with function approximation. In: International Conference on Machine Learning (ICML). pp. 30–37.

[4] Baxter, J., Bartlett, P., 2000. Reinforcement learning in pomdp's via direct gradient ascent. In: International Conference on Machine Learning. Vol. 17. pp. 41–48.

[5] Bergman, N., Bergman, N., 1999. Recursive bayesian estimation: Navigation and tracking applications. thesis no 579. Tech. rep., Linköping University, Linköping Studies in Science and Technology. Doctoral dissertation.

[6] Bertsekas, D., 2011. Approximate policy iteration: a survey and some new methods. Journal of Control Theory and Applications 9 (3), 310–335.
URL http://dx.doi.org/10.1007/s11768-011-1005-3

[7] Bertsekas, D., Tsitsiklis, J., 1996. Neuro-Dynamic Programming, 1st Edition.

[8] Bilmes, J., 1997. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Tech. rep.

[9] Bishop, C., 2006. Pattern Recognition and Machine Learning. Springer.

[10] Bou-Ammar, H., Voos, H., Ertel, W., Sept 2010. Controller design for quadrotor uavs using reinforcement learning. In: International Conference on Control Applications. pp. 2130–2135.
URL http://dx.doi.org/10.1109/CCA.2010.5611206

[11] Boyan, J., Moore, A., 1995. Generalization in reinforcement learning: Safely approximating the value function. In: Advances in Neural Information Processing Systems (NIPS). Vol. 7. pp. 369–376.

[12] Brooks, A., Makarenko, A., Williams, S., Durrant-Whyte, H., 2006. Parametric {POMDPs} for planning in continuous state spaces. Robotics and Autonomous Systems 54 (11), 887–897.
URL http://www.sciencedirect.com/science/article/pii/S0921889006000960

[13] Busoniu, L., Ernst, D., de Schutter, B., Babuska, R., April 2011. Approximate reinforcement learning: An overview. In: Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). pp. 1–8.
URL http://dx.doi.org/10.1109/ADPRL.2011.5967353

[14] Calinon, S., D'halluin, F., Sauser, E., Caldwell, D., Billard, A., jun 2010. Learning and reproduction of gestures by imitation. Robotics Automation Magazine 17 (2), 44–54.
URL http://dx.doi.org/10.1109/MRA.2010.936947

[15] Chambrier, G. d., Billard, A., 2014. Learning search policies from humans in a partially observable context. Journal of Robotics and Biomimetics 1 (1), 1–16.

[16] Cheng, H., Chen, H., Hao, L., Li, W., May 2014. Robot learning based on partial observable markov decision process in unstructured environment. In: International Conference on Robotics and Automation (ICRA). pp. 4399–4404.
URL http://dx.doi.org/10.1109/ICRA.2014.6907500

[17] Deisenroth, M. P., Neumann, G., Peters, J., 2013. A survey on policy search for robotics. Foundations and Trends in Robotics 2 (1-2), 1–142.
URL http://dx.doi.org/10.1561/2300000021

[18] Du, Y., Hsu, D., Kurniawati, H., Lee, W., Ong, S., Png, S., 2010. A pomdp approach to robot motion planning under uncertainty. In: International Conference on Automated Planning and Scheduling, Workshop on Solving Real-World POMDP Problems.

[19] Ernst, D., Geurts, P., Wehenkel, L., April 2005. Tree-based batch mode reinforcement learning. Journal of Machine Learning Research 6, 503–556.

[20] Gordon, G., 1995. Stable function approximation in dynamic programming. In: International Conference on Machine Learning (ICML).

[21] Grondman, I., Busoniu, L., Lopes, G., Babuska, R., Nov 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6), 1291–1307.
URL http://dx.doi.org/10.1109/TSMCC.2012.2218595

[22] Hausknecht, M., Stone, P., 2015. Deep recurrent q-learning for partially observable mdps. Association for the Advancement of Artificial Intelligence (AAAI).
URL https://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673

[23] Hauskrecht, M., Aug 2000. Value-function approximations for partially observable markov decision processes. Journal of Artificial Intelligence Research 13 (1), 33–94.
URL http://dl.acm.org/citation.cfm?id=1622262.1622264

[24] Jodogne, S., Briquet, C., Piater, J. H., 2006. Approximate Policy Iteration for Closed-Loop Learning of Visual Tasks. In: European Conference on Machine Learning. Vol. 17. pp. 210–221.
URL http://dx.doi.org/10.1007/11871842{_}23

[25] Khansari-Zadeh, M., Billard, A., 2011. Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. Transaction on Robotics (TRO).

[26] Kronander, K., 2015. Control and learning of compliant manipulation skills.

[27] Lagoudakis, M. G., Parr, R., dec 2003. Least-squares policy iteration. Journal of Machine Learning Research 4, 1107–1149.
URL http://dl.acm.org/citation.cfm?id=945365.964290

[28] Lange, S., Riedmiller, M., July 2010. Deep auto-encoder neural networks in reinforcement learning. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8.

[29] Lauri, M., Ritala, R., 2016. Planning for robotic exploration based on forward simulation. Robotics and Autonomous Systems.

[30] Melo, F. S., Lopes, M., 2008. Fitted Natural Actor-Critic: A New Algorithm for Continuous State-Action MDPs. pp. 66–81.
URL http://dx.doi.org/10.1007/978-3-540-87481-2{_}5

[31] Mnih, V., 02 2015. Human-level control through deep reinforcement

learning. Nature 518 (7540), 529–533.
URL http://dx.doi.org/10.1038/nature14236

[32] Neumann, G., Peters, J., Jun 2009. Fitted q-iteration by advantage weighted regression. In: Advances in Neural Information Processing Systems (NIPS). Vol. 21. pp. 1177–1184.

[33] Ormoneit, D., Glynn, P., Oct 2002. Kernel-based reinforcement learning in average-cost problems. Transactions on Automatic Control 47 (10), 1624–1636.
URL http://dx.doi.org/10.1109/TAC.2002.803530

[34] Peters, J., Schaal, S., mar 2008. Natural actor-critic. Neurocomputing 71 (7-9), 1180–1190.
URL http://dx.doi.org/10.1016/j.neucom.2007.11.026

[35] Pineau, J., Gordon, G., Thrun, S., Aug 2003. Point-based value iteration: an anytime algorithm for pomdps. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 1025–1030.
URL http://dl.acm.org/citation.cfm?id=1630659.1630806

[36] Porta, J., Vlassis, N., Spaan, M., Poupart, P., Dec 2006. Point-based value iteration for continuous pomdps. J. Mach. Learn. Res. 7, 2329–2367.
URL http://dl.acm.org/citation.cfm?id=1248547.1248630

[37] Riedmiller, M., 2005. Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. Vol. 16. pp. 317–328.
URL http://dx.doi.org/10.1007/11564096_32

[38] Ross, S., Pineau, J., Paquet, S., Chaib-draa, B., Jul 2008. Online planning algorithms for pomdps. Journal Artifcial Intelligence Research 32 (1), 663–704.
URL http://dl.acm.org/citation.cfm?id=1622673.1622690

[39] Roy, N., 2005. Finding Approximate POMDP solutions Through Belief Compression. Journal of Artificial Intelligence Research 23.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.6180

[40] Siddharth, C., Branicky, M., 2001. Search strategies for peg-in-hole assemblies with position uncertainty. In: International Conference on Intelligent Robots and Systems (ICRA). Vol. 3. pp. 1465–1470.
URL http://dx.doi.org/10.1109/IROS.2001.977187

[41] Smallwood, R., Sondik, E., Oct 1973. The optimal control of partially observable markov processes over a finite horizon. Journal of Operational Research 21 (5), 1071–1088.
URL http://dx.doi.org/10.1287/opre.21.5.1071

[42] Stulp, F., Theodorou, E., Schaal, S., 2012. Reinforcement Learning with Sequences of Motion Primitives for Robust Manipulation. Transactions on Robotics (TRO).

[43] Sung, H. G., 2004. Gaussian mixture regression and classification. Ph.D. thesis, Rice University.

[44] Sutton, R., Barto, A., 1998. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA.

[45] Sutton, R. S., Mcallester, D., Singh, S., Mansour, Y., 2000. Policy gradient methods for reinforcement learning with function approximation. In: Neural Information Processing Systems (NIPS). Vol. 12. pp. 1057–1063.

[46] Thrun, S., 2000. Monte Carlo POMDPs. In: Advances in Neural Information Processing Systems (NIPS). Vol. 12. pp. 1064–1070.
URL http://papers.nips.cc/paper/1772-monte-carlo-pomdps.pdf

[47] Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic Robotics.

[48] Veiga, T., Spaan, M., Lima, P., 2014. Point-based POMDP solving with factored value function approximation. In: Conference on Artificial Intelligence (AAAI). Vol. 28. pp. 2512–2518.

[49] Vien, N., Toussaint, M., Sept 2015. Pomdp manipulation via trajectory optimization. In: International Conference on Intelligent Robots and Systems (IROS). pp. 242–249.
URL http://dx.doi.org/10.1109/IROS.2015.7353381

[50] Wiering, M., van Otterio, M., 2012. Reinforcement Learning State-of-the-Art.