Chapter 4

# Peg in hole

In Chapter 3, we demonstrated that we could learn a search policy and successfully transfer it to a robot apprentice from demonstrations of human teachers for a task consisting of locating a wooden object on a table. In these search tasks our approach is based on the fact that intuition and knowledge exhibited by the human teachers, during the search, gives a good balance between exploration and exploitation actions which can then be encapsulated in a generative Gaussian Mixture Model (GMM) and be subsequently used as a control policy. The approach is satisfactory when extracting the many different behaviours demonstrated by the human teachers and reproducing them. However for learning an optimal or near optial policy with a unique behaviour the using of this approach will not necessarily result in a efficient policy. For the GMM will model both the good and the bad search strategies exhibited by the human teachers. If the task is difficult and many possible solutions exist, such as in the previously discussed blindfolded search task in Chapter 3, many demonstrations will be required for search patters to emerge and be encoded in the GMM. Otherwise the GMM needs to be combined with another policy as previously shown (Hybrid GMM-Greedy policy).

GMM does not discriminate between behaviours, as the Expectation-Maximisation (EM) algorithm used to learn the GMM policy ignores the quality of the demonstrated data. The EM algorithm does not contain a cost or reward function, encoding the objective of the task, which is common practice in Planning and Reinforcement Learning (RL). In the case of a difficult task where there are mostly bad demonstrations, the GMM search policy will reproduce suboptimal behaviour. Additionally there may be that no good teachers, however by combining different components from individual demonstrations a good search strategy can be extracted.

To overcome the above mentioned limitations, we introduce a binary cost function as a means of ranking the human teachers' demonstrations. To this end, we combine our PbD-POMDP approach with an Actor-Critic Reinforcement Learning (RL) framework which is close to Fitted-Value Iteration(FVI) and other Experience replay methods. This new method we refer to as RL-PbD-POMDP. Our objective is to avoid the noisy explorative rollouts, a weakness common to all RL approaches, and only rely on the data provided by the human

teachers. Autonomous exploration in RL can be seen to have three problem areas.

Firstly it is time consuming and is typically only applicable where an exhaustive exploration of the entire state or parameter space is feasible, such as in the inverted pendulum or mountain cart type problems. The universal exploration method, used throughout RL, is state independent (sometimes state dependent) white noise which results in an entire exploration of the state space. This is neither practical nor feasible for the type of search problems we are considering. Secondly in this search problem as we are using a physical robotic system the exploration cannot be random as this would be dangerous. Finally it is imperative that both the PbD-POMDP and RL-PbD-POMDP receive the same information. This would enable a fair comparison between the two algorithms and support our hypothesis that the RL-PbD-POMDP provides an improved policy with only human demonstrations as input.

We analyse our RL-PbD-POMDP approach on a power-socket Peg-in-Hole (PiH) search task. In this task, human teachers must demonstrate to a robot apprentice how to search for and connect a plug to a power socket, see Figure 4.1 (*Left*). The first part of the task, the search for the socket, is similar to the table wooden-block setup in the previous chapter. However the connection of the plug to the power socket, the PiH component, requires a higher level of precision. In Figure 4.1 (*Right*) the robot reproduces the behaviour demonstrated by the teacher.

**Figure 4.1:** Peg-in-hole (PiH) search task. *Left:* A teacher is wearing ear defenders (to impede any hearing) and a blindfold. He first searches for the socket's location and then attempts to establish a connection. Force and torque information is obtained from an ATI 6-axis force torque sensor at the end-effector of the tool held by the teacher. *Right:* The KUKA LWR4 robot equipped with the same force torque sensor and plug reproducing the teacher's demonstrated behaviour.

## 4.1 Outline

- 4.2 Background
  We review aspects of the literature related to the Peg-in-hole problem and Actor-Critic Reinforcement Learning with an emphasis on Fitted methods (also known as Batch or Experience replay), which we adapt to our GMM policy.

- 4.3 Experiment methods
  We detail the Peg-in-Hole search task, the number of participants (teachers) which provide demonstrations, the type of data which is recorded and the representation of the human's location belief.

- 4.4 Learning Actor and Critic
  We detail the representation of the Actor and the Critic and how they are learned in a Fitted Policy Evaluation framework.

- 4.5 Control architecture
  The learned behaviour is demonstrated on a 7 Degree of Freedom (DoF) articulated robot, named KUKA LWR4. We detail the hybrid position-force controller and dynamical field modulation heuristic which is used in combination with the learned behaviour policy from the human teachers.

- 4.6 Results
  We perform a set of 5 simulated experiments to test the generalisation, the importance of data provided by the human teachers and the performance against simple search approaches to find the location of the socket. We further perform 3 experiments on the real robotic platform to test the generalisation of the learned policy on different power sockets.

- 4.7 Conclusion
  The RL-PbD-POMDP policy achieves an important improvement over the previous PbD-POMDP approach by learning a value function over the belief space using approximate dynamic programming (part of FVI) and using it to update the parameters of our GMM policy. We evaluate the ability of the policies to generalise to novel sockets and different socket locations, both in simulation and on the KUKA LWR robot. The RL-PbD-POMDP approach consistently proves to be better. More importantly the RL-PbD-POMDP approach performs significantly better when it is used on the worst teacher's demonstrations, which mitigates the **original assumption** that teachers have to be consistently efficient at the task.

## 4.2 Background

### 4.2.1 Peg-in-hole

The Peg-in-Hole (PiH) task is one of the most widespread steps in industrial assembly and manipulations processes, with examples including the assembly of vehicular transmission components Chhatpar and Branicky (2001) and valves Cheng and Chen (2014). To be successful, the estimated position of the robot's end-effector and workpiece must be precise. Typically, the clearance the[en]

and plug and the workpiece's hole is very small leaving little room for error. As a result, variations in the assembly's components in combination with position uncertainty can result in either jamming during the insertion process or in failure for the plug finding the hole. This created a need for adaptive search and insertion policies for PiH, which has been driving research in this area.

From the literature, we identified the different components in PiH solutions.

All approaches use to some extent a vision system to estimate the position of the workpiece. For instance in Meeussen et al. (2010) a PR2 is equipped with a checkerboard to facilitate pose estimation of the plug with respect to a power outlet whose position is extracted through a vision processing pipeline. An initial connection is attempted by visual servoing which is successful 10% of the time. Given an estimate of the workpiece's position, a common approach is to follow either a blind increasing spiral Cartesian trajectory or parametrised policies which guarantee that all positions on the workpiece have been visited. In Meeussen et al. (2010), if the PR2 initially fails to connect the plug to the socket a spiralling outward motion is carried out with 2mm increments which obtains an overall success rate of 95%. For this approach to be applicable to a generic robot, it would require the addition of an external camera and checkerboard to the robot in question which might be cumbersome. In our work we consider a vision free system.

Another approach (which has been confined to academic circles) follows the data driven Programming by Demonstration (PbD) framework. Teleoperated or kinesthetic demonstrations by a human teacher are recorded and a policy is learned and fine-tuned so as to reproduce the same (F)orce/(T)orque profile as that demonstrated by the human teacher.

In Yang et al. (2014) the authors learn a PiH policy for the Cranfield benchmark object. A vision system obtains the pose parameters of the object whilst a human teacher demonstrates trajectories, through teleoperation, in the frame of reference of the object. A time-dependent policy represented with Dynamic Movement Primitives (DMP) Schaal et al. (2004) encodes the recorded Cartesian end-effector pose. A F/T profile is encoded separately by a regressor parameterised by radial basis functions. Successive refinements of the DMP policy are achieved through using force feedback to adapt the parameters of an admittance controller. This results in the policy having similar force profiles to the human teachers. Such an approach was proposed by Nemec et al. (2013) and further applications based on this method have been performed Abu-Dakka et al. (2014) with the incorporation of a disturbance rejection policy. Reproducing exactly the same force torque profile for the full trajectory which is encoded in a time dependent dynamical system might be unnecessary as the force torque profile is predominantly useful during the final stage of the PiH task, where the insertion can cause jamming. The force torque information can be used to rectify this problem Kronander (2015). A hybrid control paradigm Fisher and Mujtaba (1992) can also be used to control the sensed force feedback with

the environment. We make use of the hybrid control paradigm in this work in combination with a time-independent dynamical system.

Reinforcement learning has also been used in combination with DMP to learn PiH policies. In Kalakrishnan et al. (2011) an DMP policy is initialised with kinesthetic demonstrations of a door opening and pen pick up task. The recorded Cartesian trajectories are encoded in a parameterised DMP policy and augmented with a F/T regressor profile. A reward function is designed, encoding desirable properties of the F/T profile such as smoothness and continuity. After 110 trials the policy was found to be a 100% successful. In Gullapalli et al. (1994) a 18 dimensional input (sensed position, previous position and force) and a 6 dimensional output (linear and angular velocity) neural network is learned by associative reinforcement learning. During the learning process the plug is randomly positioned within the vicinity of the hole. After a 100 executions and updates, the policy was shown to be successful and was able to generalise across different geometries and clearances. This approach is similar in spirit to our work, however we will not be considering autonomous rollouts common in RL, but will solely rely on the initial data provided by human teachers with no additional rollouts.

All the above policies were learned from human demonstrations and encoded by a regressor function and optimised to reproduce a desired F/T profile. Other approaches to the PiH problem are predominantly based on heuristic search mechanisms and compliant controllers.

In Chhatpar and Branicky (2001) different blind search policies are analysed for the insertion of a spline toothed hub into a forward clutch. The state space is discretised into points so that the distance between two neighbours is smaller than the clearance of the hole, which is known as a spray point coverage. Different search strategies are evaluated which ensure that all the points are visited. It is found that paths following concentric circles gradually spiralling inwards are the most effective method for finding the hole. This concentric circle search strategy has been applied in many PiH tasks. For instance in Bdiwi et al. (2015), a PiH heuristic policy was developed to connect a 5-pin waterproof industrial charger to an electric socket. The authors estimated the pose of the socket through a vision system and used a force controller in combination with a spiral search policy to achieve a connection and demonstrated their approach to be reliable. These blind search strategies do not consider actual state uncertainty and only work well when the plug or peg is within the vicinity of the socket. In our work we consider no visual information which leads to high state uncertainty making the direct application of such blind search methods ill-suited.

In Park et al. (2013) the authors observe that humans lack the precision and sensing accuracy of robotic systems, but nevertheless, are more proficient than robots at PiH. The authors state that when humans try to connect a square plug to a socket, they rub the plug against the socket's outlet without looking. It is thought that the inherent compliance in humans' motor control

is the key to our success at PiH tasks kook Yun (2008). The authors introduce an Intuitive Assembly Strategy (IAS) inspired by the above observation which does not require the hole to be precisely localised. The IAS search strategy is based on compliant spiral motion and the execution of the search trajectory is performed with a hybrid force/position controller. We also have observed that humans are good at accomplishing such tasks and we exploit this in our own PiH policy. We further consider different types of geometric objects whilst only considering haptic information.

The spiral strategy is widely used in industrial applications due to its simplicity, however, it is a blind search method. Another approach when dealing with the assembly process consists of fine-tuning parameters of predefined policies. In Cheng and Chen (2014) the authors develop an online Gaussian Process policy optimisation of an assembly task. They demonstrate that by learning the dynamical model of the task during execution, it is faster than offline methods, such as Design of Experiment (DOE) or Genetic Algorithms.

### 4.2.2 Actor-Critic & Fitted Reinforcement Learning

In our PiH-search task, to learn a POMDP policy, we consider RL approaches which naturally handle continuous belief-state and action spaces, such as policy search/gradient methods. Chapter 2 gives an overview of such policy search methods. However, policy gradient methods are time consuming when the number of parameters is large compared with the state space, Zhao et al. (2015). This results from the large variance of the policies' gradient, making the stochastic gradient ascent learning slow. In Chapter 3, the learned PbD-POMDP policy has over 80 Gaussian functions, each of dimension 7, and we expect the number of parameters of this RL policy to be of the same order. So instead we opt for an Actor-critic (AC) approach which has been reported and proven Grondman et al. (2012b) and with this approach the variance of the gradient update in AC methods has a smaller variance compared with actor methods (policy gradient). This results in a faster learning of the policy.

Actor-critic (Sutton and Barto, 1998a, Chap. 6.6) is an RL approach in which the policy (actor) and critic (value function) have separate parameterization and can be represented by different functions, for instance the value function could be a decision tree and the policy a neural network. The advantage of an AC is that the policy can be chosen such that it is computationally efficient in evaluating actions and the value function does not necessarily have to have the same input space as the policy.

The policy gradient theorem Sutton et al. (2000) states that if the regressor functions of both the actor and critic share the same basis functions (also called *compatible* features) and their parameters are linear, then an unbiased estimate of the policies' gradient can be obtained. The drawback of this approach is

that both the value function and policy have to be defined over the state-action space. This is restrictive and in addition it has been shown that Function Approximators (FA), such as linear models or neural networks, when combined with temporal difference learning can diverge, Boyan and Moore (1995).

As we are working in belief-space we seek a framework in which both the actor and critic can have their own parameterisation whilst guaranteeing convergence of the value function. All value function approximators, such as tile coding, state aggregation, k-nearest-neighbour, locally weighted averaging and grid discretisation are all averagers and are guaranteed to converge in model based RL Gordon (1995) when used with temporal difference learning. The key idea presented in Gordon (1995) and which lead to the increase in popularity of batch and fitted method, is to separate the Bellman and value function in a synchronous update, that is, to first compute the Bellman update for all the sample states and then fit a value function via standard supervised learning techniques. The extension to a model-free approach with a kernel function approximator (locally weighted averaging, the kernel is a Gaussian function) known as Kernel-Based Approximate Dynamic Programming (KBDP) Ormoneit and Glynn (2002) has proven to be globally optimal in a continuous-space framework. This leads to the wider application of Batch RL methods such as Fitted Value Iteration (FVI) Bou-Ammar et al. (2010) and Fitted Q-Iteration (FQI) Ernst et al. (2005a) (Q-approximator is a random forest ensemble), Neumann and Peters (2009a) to the RL community. By remembering all the state transition pairs and by applying multiple synchronous Dynamic Programming (DP) and function approximation updates, the problem of diverging value function approximators is resolved.

Retaining all the data makes it in practice easy to apply function approximators which are not averagers, such as neural networks, to RL problems. A successful example was Neural Fitted Q-Iteration (NFQI) Riedmiller (2005b) which uses a multi-layer perceptron to represent the Q-function for the cart-pole and mountain car problems and shows rapid convergence to optimal policies. It has since been used in many extensions, Peters and Schaal (2008b), Agostini and Celaya (2010). This has lead to the application of more sophisticated regression methods such the increasingly popular Deep Learning methods which are known as Deep Fitted Q-iteration (DFQ)Lange and Riedmiller (2010) (used to learn visual control policies) and with recent work including learning to play ATRI and ping-pong games Mnih et al. (2015), Hausknecht and Stone (2015) (reviewed in Chapter 2).

The reader is referred to Buşoniu et al. (2011) for a literature review which includes a taxonomy of Batch RL methods and to (Wiering and van Otterio, 2012, Chap 2) for a concise description of the origins of Batch RL from its origins to how it became popular with Fitted RL approaches and its continuation into Deep Learning.

The RL-PbD-POMDP framework which we will use in this chapter is also

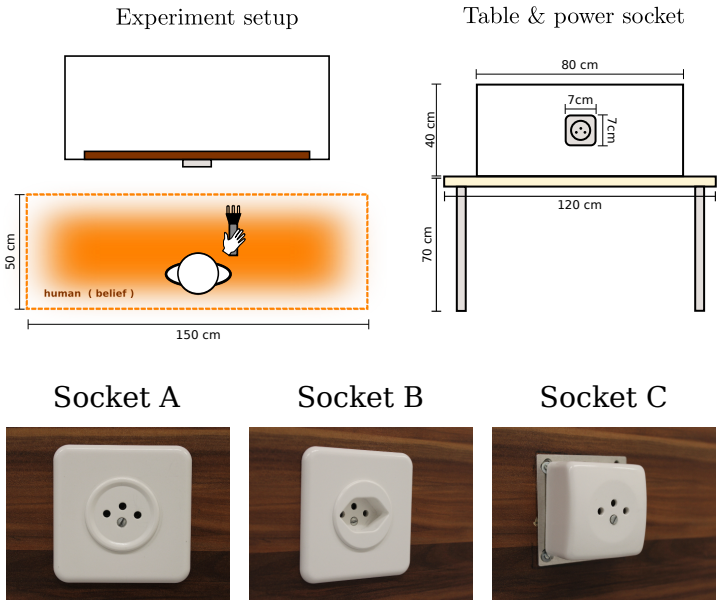



**Figure 4.2:** The experimental setup. *Top-left:* A participant (human teacher) is blindfolded and placed within the orange rectangular area always facing the wall. *Top-right:* Dimensions of the the wall and socket. *Bottom:* Three different power sockets, only socket A and B are used for data collection, socket C is purely used for evaluating the generalisation of the learned policy.

based on a Fitted approach, however we avoid performing the expensive maximisation over the continuous actions space, as in the FVI and FQI approaches, by doing a fitted policy evaluation followed by policy improvement. We use a Gaussian Mixture Model to parameterise the policy and a Locally Weighted Regression (LWR) as the value function approximator.

## 4.3 Experiment methods

Figure 4.2 (*Top-left*), illustrates the PiH-search experiment setup. The orange area represents the teachers 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, see Figure 4.2 (*Top-right*) for an illustration.

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 4.2 (*Bottom*) for an illustration.

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 **raw** wrench $\phi \in \mathbb{R}^6$ measurements. We define the **actual** measurement to be a function of the raw wrench, $\tilde{y}_t = h(\phi_t)$, which is a binary feature vector. The feature vector

[1] http://www.iec.ch/worldplugs/typeJ.htm
[2] http://www.ati-ia.com/products/ft/sensors.aspx


**Figure 4.3:** Human holding the cylinder plug holder, which is equipped with OptiTrack markers.

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), see Figure 4.3, to measure both linear, $\dot{x} \in \mathbb{R}^3$, and angular velocity, $\omega \in \mathbb{R}^3$, at each time step which is recorded at a rate of 100 Hz. The force and torque information from the ATI sensor is recorded at the same rate.

In this task, the human's location belief is represented by a probability distribution function. The participants (teachers) initial belief is assumed to be uniformly distributed as depicted in orange area of Figure 4.2 and that all subsequent beliefs can be inferred from the measured velocity and measurements provided by the ATI and OptiTrack sensors. The following section describes how the belief can be represented, computed and compressed.

### Belief state

For the task at hand, the belief probability density function, $p(x_t|y_{0:t}, \dot{x}_{0:t})$, is a Point Mass Filter (PMF) (Bergman and Bergman, 1999, p.87), which is a Bayesian filter. It is parametrised by a set of grid cells which contain valid probabilities and is recursively updated by the application of a **motion**, $p(x_t|x_{t-1}, \dot{x}_t)$ and **measurement**, $p(y_t|x_t)$ model. The motion model updates the position of the probability density function and subsequently increases the uncertainty of the position. The measurement model indicates areas of the state space from which a measurement $\tilde{y}_t$ could have originated. In Figure 4.4 (*Bottom-right*) we illustrate the likelihood when an edge is sensed.

A PMF was chosen to represent the believed location of the plug as the sensing the sensing likelihoods are non-gaussian and lead to multi-modal distributions. A PMF is able to capture such non-gaussianity whilst remaining fully deterministic (which is not the case for a particle filter).

The probability density function $p(x_t|y_{0:t}, \dot{x}_{0:t})$ is high dimensional and thus it is impractical to directly learn a statistical policy $\pi_\theta : p(x_t|y_{0:t}, \dot{x}_{0:t}) \rightarrow \dot{x}_t$ without some form of compression. One possibility would be E-PCA Roy and Gordon (2003b) which extracts a set of representative basis functions which

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

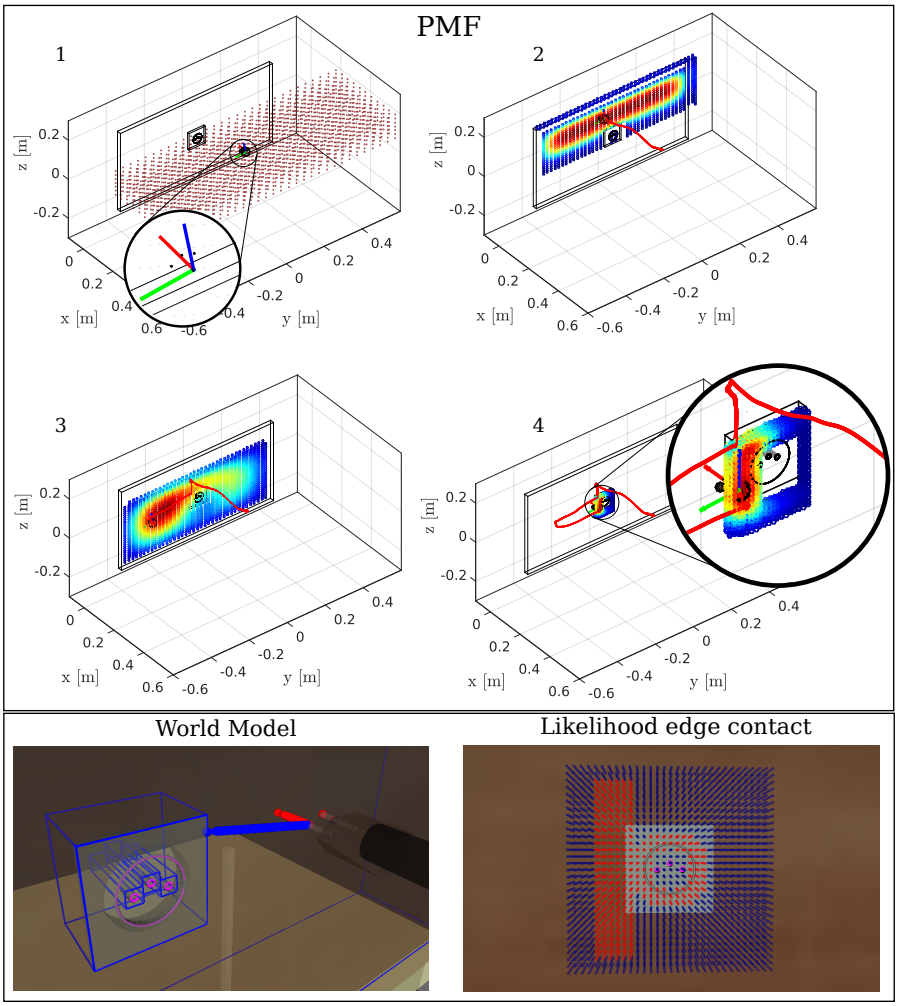



**Figure 4.4:** *Left:* Point Mass Filter (PMF) update of a particular human demonstration. (1) Initial uniform distribution spread over the starting region. Each grid cell represents a hypothetical position of the plug, the orientation is assumed to be known. (2) First contact, the distribution is spread 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. *Right*: **World model**: The plug is presented by its three plug tips and the wall and sockets are fitted with bounding boxes. **Likelihood**: The plug enters in contact with the left edge of the socket. As a result, the value of the likelihood in all the regions, $x_t$, 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.

are aslo probability distributions. Although elegant this method requires a discretisation of the belief space which is computationally expensive. Instead we chose to compress the pdf to a belief space vector composed of the maximum a posteriori, $\hat{x}_t^{\text{MAP}} = \text{argmax}_{x_t} p(x_t|y_{0:t}, \dot{x}_{0:t}) \in \mathbb{R}^3$, and the differentiation entropy, $U = H\{p(x_t|y_{0:t}, \dot{x}_{0:t})\} \in \mathbb{R}$. All pdfs in our recorded data set $D$ are transformed to a belief space feature vector, $b_t = [\hat{x}_t^{\text{MAP}}, U]^{\text{T}}$.

Each demonstration of the task by a participant results in a dataset $D = \{\dot{x}_{1:T}^{[i]}, \omega_{1:T}^{[i]}, \phi_{1:T}^{[i]}, b_{1:T}^{[i]}\}$, where the upper index $[i]$ references the ith search trajectory (also one execution of the task or one episode) and subscript $1:T$ denotes the time steps during the trajectory from initialisation $t = 1$ until the end $t = T$.

### 4.3.1 Participants and experiment protocol

To perform the PiH search tasks we recruited 10 student volunteers to be teachers (all male Master's and PhD students). 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 repetitions of the PiH searches task with socket A, followed by a 10 minute break. Then they performed an additional 15 searches with socket B. The members of group B performed the same protocol starting with socket B and ending with socket A. Figure 4.5 summarises a walk through of the experiment. The only criteria of exclusion was the inability of the subject to accomplish the task. All participants gave written consent for taking part in this study.

The next section describes in detail the protocol for the search task:

1. Participant signs a form of consent before starting the experiment.

2. Each participant is given the opportunity to familiarise himself with the environment and become comfortable in wearing the sensor deprivation apparatus. During this time the participant is allowed to practice connecting the plug to the socket whilst standing within its vicinity.

3. Once the participant feels sufficiently ready to carry out the task to the best of his ability, the experimenter proceeded to disorient him through the usage of swivel chair. The disorientation process takes 30 seconds and included both translation and rotation motions. After disorientation, the participant is signalled to stand up. The participant is reminded that he is facing the direction of the wall and that his the starting location is within the orange rectangular area demarcated on the floor. Once the participant is standing he is signalled by a light touch to the shoulder that he can start the task.

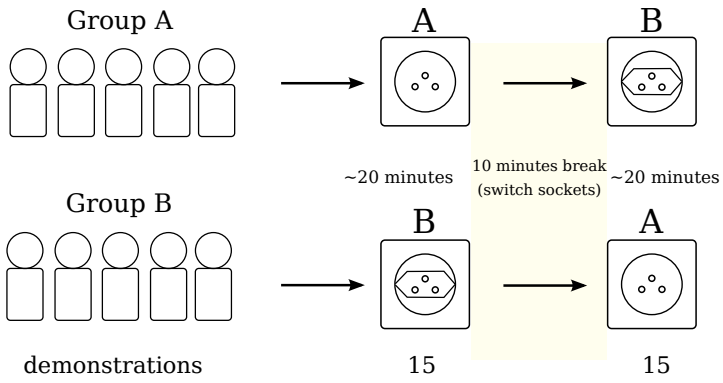



**Figure 4.5**: Experiment protocol. The participants are divided in two groups of 5, Group A first does the search experiment with socket A and after a short break repeat the task with socket B and the same logic holds for Group B. For each a socket 15 executions of the task is recorded.

4. At task completion, the subject is once again disoriented and the this process is repeated a total of 15 times. After 15 trials, the subject is given a 10 minute break to rest whilst the experimenter changes the type of socket (A or B). A participant of group A will now continue with socket B. Similarly a participant of group B will continue after the break with socket A.

Each participant carried out a total of 30 PiH-search experiments, giving a total of 300 demonstrations.

#### Preliminary results

Both groups A and B took $9 \pm 10$s to find the socket's edge, regardless of the type. This is to be expected since the sockets are at the same location. However after the socket was found it took a further $8 \pm 7$s on average for group A to connect socket A and $12 \pm 10$s on average for group B to connect socket A. As we can see this is not a straight forward task when considering the sensory deprivation. See Figure 4.6 (*Bottom*) the time taken to connect the plug to the socket. In Appendix 4.8.1 we report the results of an analyse of variance (ANOVA) on the time taken to connect the socket the sockets and we found that it took more time (4 seconds more) to connect socket A than socket B. This is somewhat expected as socket B has a funnel which can help to contain the subject to within the vicinity of the holes.

As socket A is more difficult we will be only using the demonstrations from this socket as training data to learn a policy. Both socket B and C will be used solely to evaluate the generalisation of the policy.

## 4.4 Learning Actor and Critic

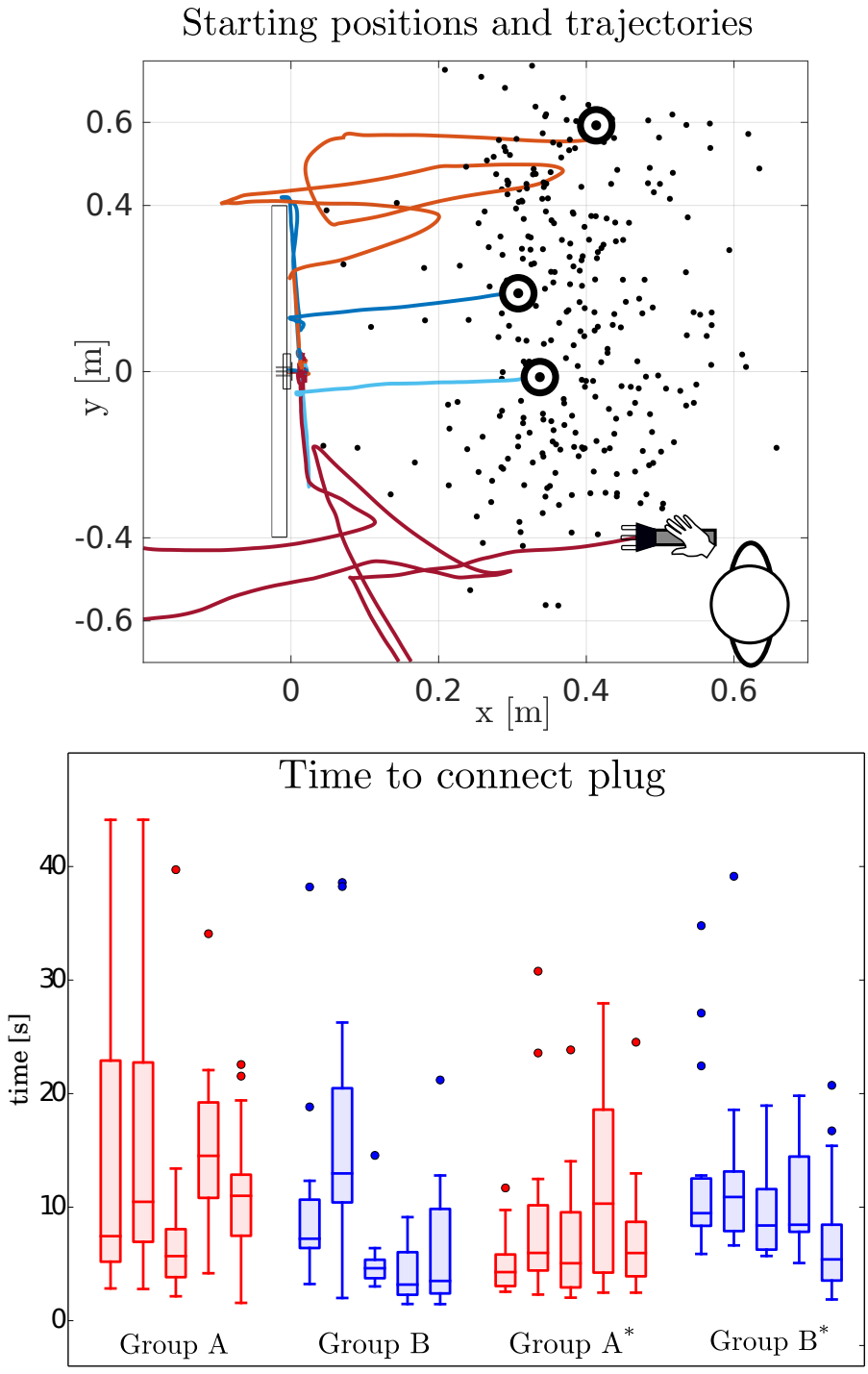



**Figure 4.6:** *Top*: Black points represent the starting position of the end-effector for all the demonstrations. Four trajectories are illustrated. *Bottom:* Time taken for the teachers to accomplish the PiH once the socket is localised. Group A and B are depicted in red and blue. The asterisk indicates that the group has changed sockets, so Group A$^*$ means that Group A is now performing the task with socket B and Group B$^*$ means that group B is now performing the task with socket A.

In our approach we learn two data driven policies. The first policy maps from belief space to linear velocity $\pi_{\boldsymbol{\theta}_1} : b_t \mapsto \dot{x}_t$ and the second from angular sensed wrench to angular velocity, $\pi_{\boldsymbol{\theta}_2} : \phi_t \mapsto \omega_t$. We chose to learn the belief policy $\pi_{\boldsymbol{\theta}_1}$ in a Actor-Critic RL framework and the wrench policy $\pi_{\boldsymbol{\theta}_2}$ directly from the demonstrated data as was done in Krona (2015), which proved to be efficient in overcoming jamming during the PiH. A POMDP solver's objective is to find a policy (Actor), $\pi_{\boldsymbol{\theta}_1} : b \mapsto u$, which maximises the value function (Critic) $V^{\pi_{\boldsymbol{\theta}_1}} : b \mapsto \mathbb{R}$ for an initial belief, $b_0$. The value function is the expected reward over an infinite time horizon.

$$V^{\pi_{\boldsymbol{\theta}_1}}(b_t) = \mathbb{E}\left\{ \sum_{t=0}^{\infty} \gamma^t r_{t+1} | b_0 = b, \pi_{\boldsymbol{\theta}_1} \right\} \tag{4.4.1}$$

In an Actor-Critic setting, the temporal difference error, $\delta_t \in \mathbb{R}$, of the value function (the Critic) is used both as a learning signal to update simultaneously itself and the actor (the policy). In our setting we will learn two separate policies, one for the linear velocity and the other for the angular velocity, as the orientation remains the same during most of the search until it is time to connect the plug to the socket. When the plug has to be connected it is necessary to control for orientation to avoid jamming.

### 4.4.1 Actor & Critic

Both actors/policies are parametrised by a Gaussian Mixture Model (GMM), Equation 4.4.2.

$$\pi_{\boldsymbol{\theta}}(\dot{x}, b) = \sum_{k=1}^{K} w^{[k]} \cdot g(\dot{x}, b; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]}) \tag{4.4.2}$$

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

$$\boldsymbol{\mu}^{[k]} = \begin{bmatrix} \boldsymbol{\mu}_{\dot{x}}^{[k]} \\ \boldsymbol{\mu}_{b}^{[k]} \end{bmatrix}, \; \boldsymbol{\Sigma}^{[k]} = \begin{bmatrix} \boldsymbol{\Sigma}_{\dot{x}\dot{x}}^{[k]} & \boldsymbol{\Sigma}_{\dot{x}b}^{[k]} \\ \boldsymbol{\Sigma}_{b\dot{x}}^{[k]} & \boldsymbol{\Sigma}_{bb}^{[k]} \end{bmatrix}$$

where $\sum_k w^{[k]} = 1$, $\boldsymbol{\mu}_{\dot{x}}^{[k]} \in \mathbb{R}^3$ and $\boldsymbol{\mu}_{b}^{[k]} \in \mathbb{R}^4$.

A generative model of the angular velocity and wrench $\pi_{\boldsymbol{\theta}_2}(\omega, \theta)$ and a generative model of the linear velocity and belief state $\pi_{\boldsymbol{\theta}_1}(\dot{x}, b)$ are learned. In both cases we use the Bayesian Information Criterion to determine the number of Gaussian functions. In the next section, we will show how the parameters of $\pi_{\boldsymbol{\theta}_1}$ can be adapted by the value function of the Critic.

The Critic (the value function, Eq. 4.4.1) evaluates the performance of the current policy. It is the expected future reward given the current belief state and policy. In our setting a reward of $r = 0$ is received at each time step until the goal (plug-socket connection) is achieved, where a reward of 100 is given, $r_T = 100$. Given the continuous nature and dimensionality of the

belief space we use Locally Weighted Regression Atkeson et al. (1997) (LWR) as a function approximator of the value function, $V^{\pi}(b)$. LWR is a memory-based non-parametric function approximator. It keeps a set of input-target pairs $\{(b, r)\}$ as parameters. When a value, $b$, is queried, a set of $p$ neighbouring points are chosen from the input space and are weighted according to a distance metric. The predicted output is then the result of a weighted least square of the $p$ points. Equation 4.4.3 is the distance function used where $D$ is a diagonal matrix.

$$W_{i,i} = \exp\left(-\frac{1}{2}(b - b_i)^{\mathrm{T}} D^{-1} (b - b_i)\right) \tag{4.4.3}$$

A new value is queried according to Equation 4.4.4,

$$V^{\pi}(b) = b\,(B^{\mathrm{T}} W B)^{-1} B^{\mathrm{T}} W \mathbf{r} \tag{4.4.4}$$

where $B = (b_1, \ldots, b_p)^{\mathrm{T}} \in \mathbb{R}^{(D \times p)}$, $W \in \mathbb{R}^{(p \times p)}$ is a diagonal matrix, $\mathbf{r} = (r_1, \cdots, r_p)^{\mathrm{T}} \in \mathbb{R}^{(p \times 1)}$

### 4.4.2 Fitted policy evaluation and improvement

#### Policy evaluation

To learn the value function we make use of batch reinforcement learning Ernst et al. (2005a), also known as Experience replay. This is an offline method which applies multiple sweeps of the Bellman backup operator over a dataset of tuples $\{(b_t^{[i]}, \dot{x}_t^{[i]}, r_t^{[i]}, b_{t+1}^{[i]})\}_{i=1,\cdots,M}$ until the Bellman residual, $||V_{k+1}^{\pi}(b) - V_k^{\pi}(b)||$, converges.

**Algorithm 1:** Fitted Policy Evaluation

**input** : $\epsilon$, $\{(b_t^{[i]}, r^{[i]}, b_{t+1}^{[i]})\}_{i=1,\cdots,M}$
**output**: $\hat{V}_k^{\pi}(b_t)$

**while** $||\hat{V}_{k+1}^{\pi}(b) - \hat{V}_k^{\pi}(b)|| < \epsilon$ **do**
  $\hat{V}_{k+1}^{\pi}(b_t) = \text{Regress}(b,\, r_t + \gamma \hat{V}_k^{\pi}(b_{t+1}))$

A broad spectrum of research has made use of batch RL methods to learn policies. Most of them have focused on learning the Q-value function directly (Fitted Q-Iteration) Neumann and Peters (2009b); Ernst et al. (2005a); Riedmiller (2005a). Although learning the Q-value function directly solves the control problem it often requires discretisation of the action space or assumes quantifiable actions. The reason is that the Q-Bellman backups, $\hat{Q}(b_t, \dot{x}_t) \leftarrow \gamma \max_{\dot{x}_{t+1}} \hat{Q}(\dot{x}_{t+1}, b_{t+1})$, requires an optimisation over the action space, $\dot{x}_{t+1}$, to find the best applicable action. Given the dimensionality and continuity of

our problem we opt for an on-policy evaluation method which requires multiple *policy evaluation* and *policy improvement* iterations to achieve an optimal policy. In order for the RL-PbD-POMDP and PbD-POMDP to be comparable we will only be performing one iteration of policy evaluation and improvement, hence Algorithm 1 is applied only once to the dataset.

#### Policy improvement

The Temporal Difference (TD) error $\delta_t^\pi = r_{t+1} + \gamma V^\pi(b_{t+1}) - V^\pi(b_t)$ given by the critic is used to update the actor (Sutton and Barto, 1998b, Chap. 6). In our offline approach we first estimate value function of the belief state, $\hat{V}^\pi(b)$, until convergence and then use it to update the actor. This offline batch method has the advantage that no divergence will occur during the learning process.

We proceed to update the Actor given the Critic through a modification of the Maximisation step in Expectation-Maximisation (EM) for Gaussian Mixture Models. We refer to this modification as Q-EM which is strongly related to a Monte-Carlo EM-based policy search approach (Deisenroth et al., 2013b, p.50).

The reward of a demonstrated trajectory (one episode) is given by the discounted return, Equation 4.4.5.

$$R(b^{[i]}, \dot{x}^{[i]}) = \sum_{t=0}^{\mathrm{T}^{[i]}} \gamma^t \, r(b_t^{[i]}, \dot{x}_t^{[i]}) \tag{4.4.5}$$

All policy gradient approaches seek to find a set of parameters, $\boldsymbol{\theta}$, of the Actor, which will maximise the expected reward, equivalent to maximising Equation 4.4.6.

$$\begin{aligned} J(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\dot{x},b)}\{R(b,\dot{x})\} \\ &= \sum_{i=1}^{N} \underbrace{\left( \prod_{t=0}^{T^{[i]}} \pi_{\boldsymbol{\theta}}(\dot{x}_t^{[i]}, b_t^{[i]}) \right)}_{\pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})} R(b^{[i]}, \dot{x}^{[i]}) \end{aligned} \tag{4.4.6}$$

To find the parameters which maximise the cost function, $\operatorname{argmax}_{\boldsymbol{\theta}'} J(\boldsymbol{\theta}')$, its derivative is set to zero. As this cannot be done directly, we maximise the logarithmic lower bound of the cost function which results in Equation 4.4.7.

$$\nabla_{\boldsymbol{\theta}'} \log(J(\boldsymbol{\theta}')) = \sum_{i=1}^{N} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}'} \log \pi_{\boldsymbol{\theta}'}(\dot{x}_t^{[i]}, b_t^{[i]}) \, Q^\pi(\dot{x}_t^{[i]}, b_t^{[i]}) \tag{4.4.7}$$

Setting the derivative of Equation 4.4.7 to zero and solving for the parameters $\boldsymbol{\theta} = \{w, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ leads to a Maximisation update step of EM which is weighted by $Q^\pi$, see Appendix 4.8.2 for a more complete derivation. We use critic's TD error as a substitute for $Q^\pi$. Assuming that our estimated value function, $\hat{V}^{\pi_{\boldsymbol{\theta}}}$,

is close to the true value function $V^{\pi_{\boldsymbol{\theta}}}$, the TD error $\delta^{\pi}$ is an unbiased estimate of the advantage function, Equation 4.4.8 (see Appendix 4.8.4).

$$A^{\pi}(b_t, u_t) = Q^{\pi}(b_t, u_t) - V^{\pi}(b_t) = \delta_t^{\pi} \tag{4.4.8}$$

Using the advantage function as means of policy search is popular with examples such as Natural Actor Critic (NAC) Peters and Schaal (2008a).

Each data point has an associated weight, $\delta \in \mathbb{R}$, where $\delta^{[m]} \geq 0$ means that the state action-pair $x^{[m]}$ leads to an increase in the value function and $\delta^{[m]} \leq 0$ leads to a decrease in the value function. The 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 be between 0 and 1.

The reader is referred to Appendix 4.8.3 for the new Maximisation update step of Q-EM for a GMM parameterization of the policy.

**2D example fitted policy evaluation and improvement**

To illustrate the mechanism of fitted policy evaluation and improvement, we give a 2D example of its application, see Figure 4.7. 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 two obstacles. Neither teacher demonstrated the optimal straight path. If we fit a GMM $\pi_{\boldsymbol{\theta}}(\dot{x}, x)$ to the teachers' data and take the policy to be the output of Gaussian Mixture Regression (GMR) $\mathbb{E}\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\}$ we obtain the behaviour illustrated in the *Bottom-left* subfigure. As different behaviours were demonstrated and encoded in the GMM, the trajectories generated from the GMR policy are an average of the original demonstrated behaviour. 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 $r = 1$ 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 advantage function $A^{\pi}(x, \dot{x})$ is highest along the start-goal axis, data points which follow this gradient will be weighted higher. This results in a policy with better rollouts (closer to the optimal path) than the trajectories generated by the policy learned via standard EM. The *Bottom-right* subfigure illustrates of the Q-EM policy and rollout trajectories.

#### Belief state fitted policy evaluation

The Fitted policy evaluation (FPE) Algorithm 1 is applied to the demonstrations of the PiH-search task with socket A. In Figure 5.1 we illustrate the value function of the most likely state after the FPE algorithm converged. As

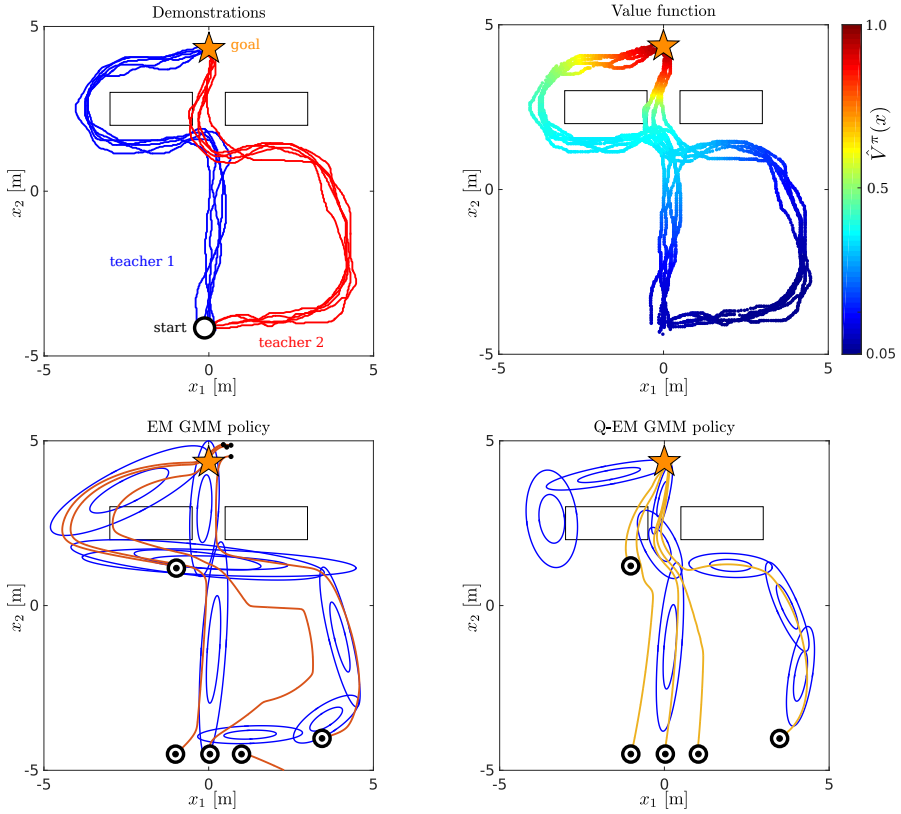



**Figure 4.7:** Fitted policy evaluation & improvement example. *Top-left:* The goal of the task is to reach the goal state. The first teacher (blue) demonstrates five trajectories which contours the obstacle in front of the goal. The second teacher (red) demonstrates 5 trajectories which initially deviate 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, $r = 1$ 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 proportional to the advantage function.

expected, the value function is high closest to the socket and low further away and is highest around the axis $z = 0$ and $y = 0$. When policy improvement via Q-EM is applied the Gaussian functions of the GMM will favour these locations.

In Figure 4.9 we illustrate the best and worst trajectories in terms of the accumulated value function. We can see that the five best trajectories (red) tend to be aligned with the socket (star position in front of socket), whilst the worst are towards the edges of the wall and tend to follow spiralling movements.

We learned two 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 evaluation and improvement which we call Q-EM. In the Result section we compare these two policies to the improvements which can be achieved with the RL-PbD-POMDP framework.

## 4.5 Control architecture

A Gaussian Mixture Model was learned for both linear and angular velocity, although only the linear control policy is active until the plug is within the socket's hole. The orientation is kept constant. The direction to search is given by conditional, Equation 4.5.1,

$$\pi_{\boldsymbol{\theta}}(\dot{x}|b) = \sum_{k=1}^{K} w_{\dot{x}|b}^{[k]} \cdot g(\dot{x}; \boldsymbol{\mu}_{\dot{x}|b}^{[k]}, \boldsymbol{\Sigma}_{\dot{x}|b}^{[k]}) \tag{4.5.1}$$

which is a distribution over the possible normalised velocities. The function $g(\cdot)$ is a multivariate Gaussian function parameterised by mean $\boldsymbol{\mu}_{\dot{x}|b}^{[k]} \in \mathbb{R}^{(3\times1)}$ and Covariance $\boldsymbol{\Sigma}_{\dot{x}|b}^{[k]} \in \mathbb{R}^{(3\times3)}$. The subscript $\dot{x}|b$ indicates that the parameters are the result of the conditional. The reader is referred to Calinon et al. (2010), Sung (2004) 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. In Figure 4.10 we illustrate the multi-modal vector fields of the conditional, Equation 4.5.1. In autonomous dynamical systems control, the velocity to is taken from the expectation of Equation 4.5.1. Taking the expectation, which is weighted linear combination of the modes could result in unobserved behaviour or no movement if the velocities cancel out. As a result we use a modified version of the expectation operator which favours the current direction, Equation 4.5.2 - 4.5.3.

$$\alpha(\dot{x}) = w_{\dot{x}|b}^{[k]} \cdot \exp(-\cos^{-1}(< \dot{x}, \boldsymbol{\mu}_{\dot{x}|b}^{[k]} >)) \tag{4.5.2}$$

$$\dot{x} = \mathbb{E}_{\alpha}\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\} = \sum_{k=1}^{K} \alpha_k(\dot{x}) \cdot \boldsymbol{\mu}_{\dot{x}|b}^{[k]} \tag{4.5.3}$$

When the applied velocity mode is no longer present another direction is

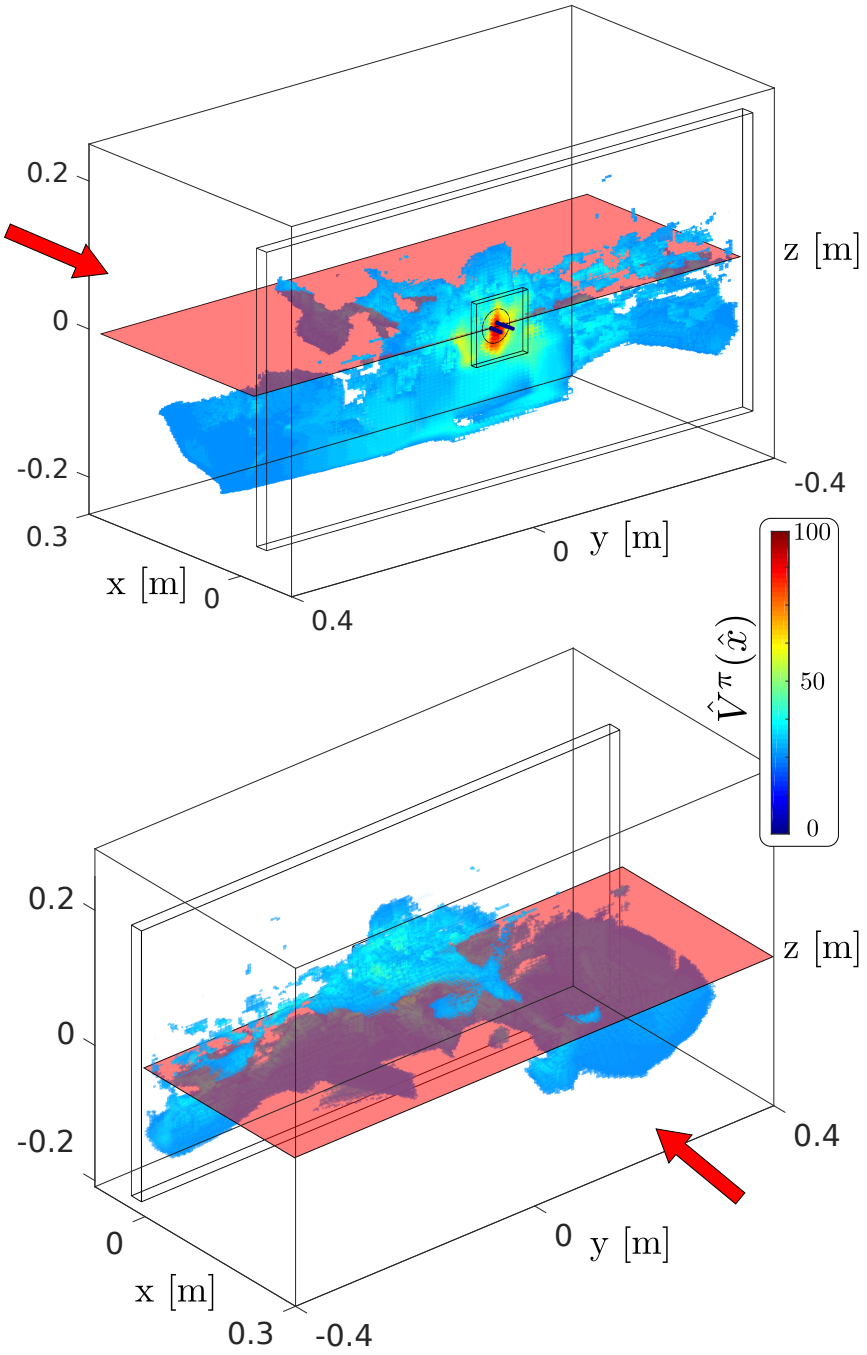



**Figure 4.8:** LWR value function approximate $\hat{V}^{\pi}(\hat{x})$ for the most likely state $\hat{x}$. The red plane is to help visualise where the value function is above and below the axis $z = 0$. Only states with values above 0.25 are plotted. The red arrow indicates the heading of the human 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.

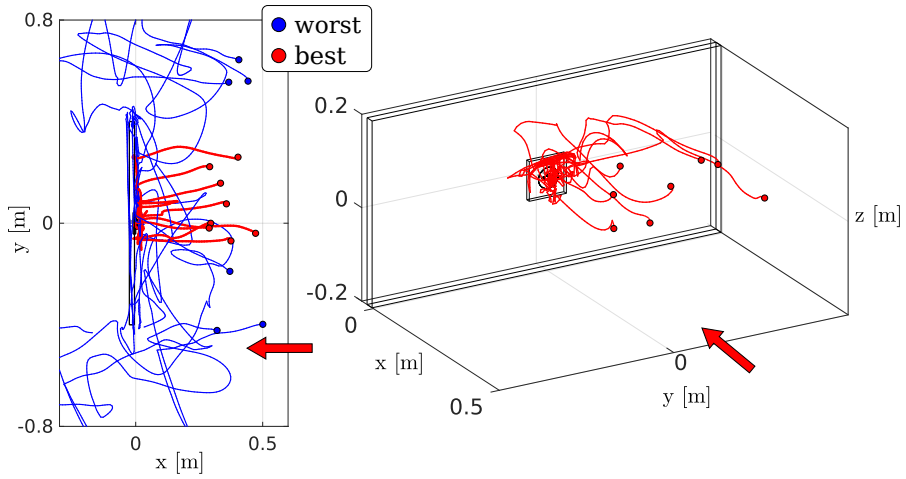



**Figure 4.9:** Best and worst trajectories. The red demonstrated trajectories are the best in terms of the amount of value function gain whilst the blue are the worst. The red arrow indicates the teacher's heading. The blue trajectories tend towards the sides of the wall as the initial starting position is on the boarders 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 $z = 0$.

sampled. For example, when the robot enters in contact with a feature, greatly reducing the uncertainty, the current mode changes and a new search direction is computed. Figure 4.10 illustrates the policy vector field for GMM and Q-EM, both learned from teachers demonstrations.

### 4.5.1 Robot Implementation

The GMM policy $\dot{\underline{x}} = \mathbb{E}_\alpha\{\pi_{\boldsymbol{\theta}}(\dot{x}|b)\}$ outputs a linear velocity which is normalised, $\dot{\underline{x}} \in \mathbb{R}^{(3\times 1)}$. The amplitude of the velocity is computed separately but first we modulate the velocity according to forces sensed on the end-effector. This search task is haptic and as a result the end-effector of the robot will always be 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. Our hybrid controller targets a sensed force $F_x$, in the $x$-axis, of 3N. The other two velocity components of the direction vector are given by Equation 4.5.3. This force by itself is insufficient for the robot to reliably surmount the edges of the socket. To overcome the edges the vector field of the GMM is modulated in $y$ and $z$-axis, Equation 4.5.4.

$$\dot{\underline{x}} = R_y(c(F_z)\cdot\pi/2)\cdot R_z(c(F_y)\cdot\pi/2)\cdot\dot{\underline{x}} \tag{4.5.4}$$

$R_y$ and $R_z$ are $(3\times 3)$ rotation matrices around the $y$ and $z$-axis, $c(F) \in [-1, 1]$ is a truncated scaling function of the sensed force. When a force $F_z$ of 5N is sensed, a rotation of $R_y(\pi/2)$ is applied to the original direction resulting in

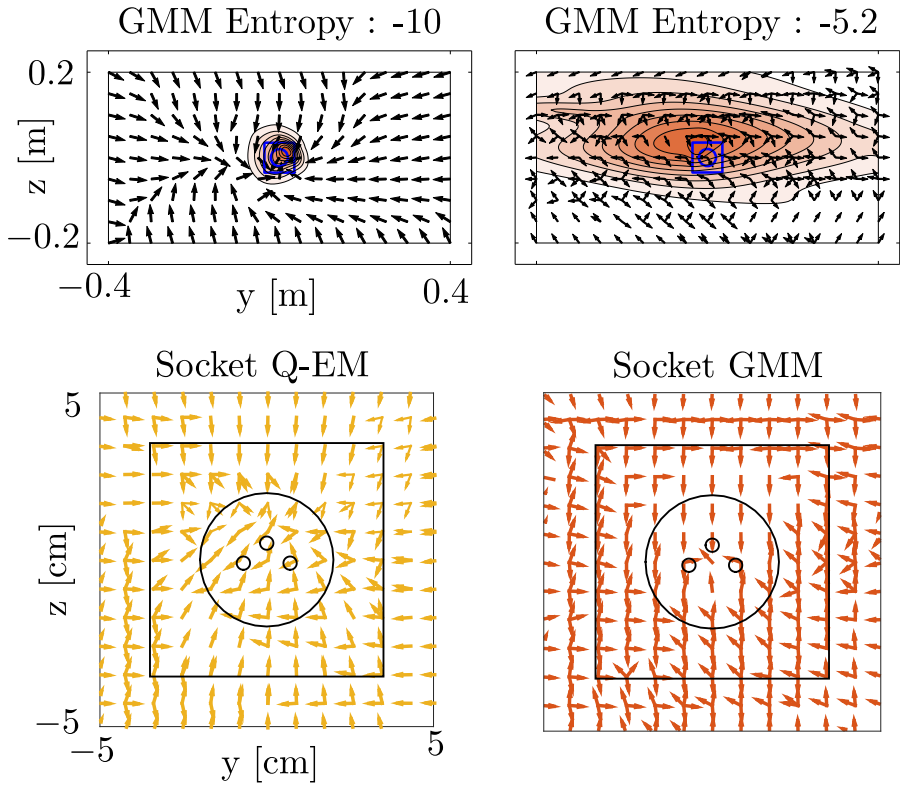



**Figure 4.10:** Q-EM and GMM policy vector fields. *Top*: The GMM policy is conditioned on an entropy of $-10$ and $-5.2$. For the lowest entropy level, most of the probability mass is close to the socket area since this level corresponds to very little uncertainty; we are already localised. We can 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 yellow vector field is of the Q-EM and orange of the GMM. The Q-EM vector field tends to be closer to a sink and there is less variation.

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,

$$\begin{aligned} \nu &= \max(\min(\beta_1, K_p(x_g - \hat{x}), \beta_2) \\ \dot{x} &= \nu \dot{\underline{\mathrm{x}}} \end{aligned} \tag{4.5.5}$$

where the lower and upper amplitude limits are given by $\beta_1$ and $\beta_2$, $x_g$ is the position of the goal, and $K_p$ the proportional gain which was tuned through trials.

The above procedure can control the general behaviour of the search but is insufficient for a successful implementation on a robotic system such as the 7 Degree of Freedom $q \in \mathbb{R}^7$ KUKA LWR, which we illustrate in Figure 4.11. The GMM policy $\dot{x} = \mathbb{E}_\alpha\{\pi_{\boldsymbol{\theta}_1}(\dot{x}|b)\}$ outputs a linear velocity and during most of the search, until the insertion of the peg into the socket in which case the angular velocity is the output of samples drawn from the conditional $\omega \sim \pi_{\boldsymbol{\theta}_2}(\omega|\phi)$, a reference orientation $\mathbf{R}^r = I$ is kept constant throughout the task. An angular velocity $\omega_t = \text{angleaxis}(\mathbf{R}^{\mathrm{T}}\mathbf{R}^r)$ is computed from the current and desired orientations. Given the kinematic chain of the robot, the inverse of the Jacobian $J(q) \in \mathbb{R}^{6\times 7}$ is used in an impedance control to transform the Cartesian control $u_t = [\dot{x}_t, \omega_t]^{\mathrm{T}}$ to torque commands $\tau_t \in \mathbb{R}^7$, Equation 4.5.6.

$$\tau_t = J^{\mathrm{T}}(q_t)\left(-Ku_t - D\dot{u}_t\right) + g(q) \tag{4.5.6}$$

Where $K, D \in \mathbb{R}^{6\times 6}$ are diagonal stiffness and damping matrices and $g(q)$ compensates for gravity. Given an applied torque there is a resulting joint velocity $\dot{q}_t$ from which we can compute the measured Cartesian end-effector velocity used in the motion model of the PMF. Figure 4.12 illustrates the complete control flow.

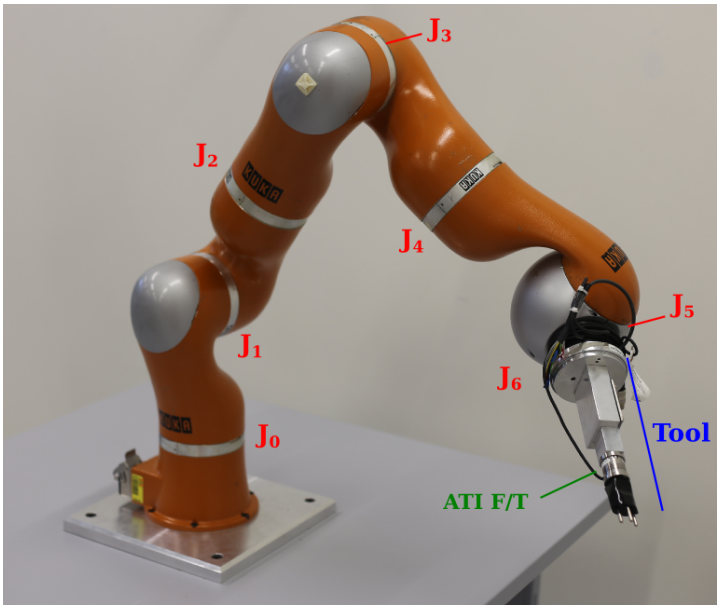



**Figure 4.11:** The KUKA LWR is a 7 Degree Of Freedom (DoF) robot, we illustrate in red each joint, which is controlled at a rate of 1kHz via an ethernet cable. The KUKA API provides a command interface to the stiffness, damping, position and torque variables of each joint.

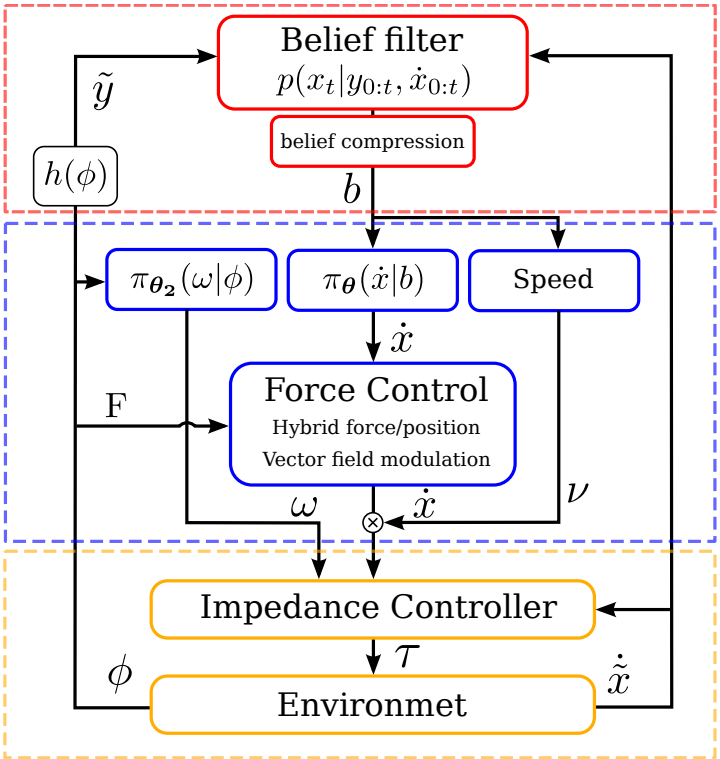



**Figure 4.12:** Control architecture. The PMF (belief) received a measured velocity, $\dot{\tilde{x}}$, and sensor feature $\tilde{y}$ and gets updated via Bayes rule. The belief is compressed and used by both the GMM policy and the proportional speed controller, Equation 4.5.5.

## 4.6 Results

We evaluate the following three properties of the policy learned in our Actor-Critic framework:

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 as a result 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. We call this policy Q-Greedy and it is used to test whether human demonstrations are necessary. We evaluate whether it is possible to obtain a good policy from the worst two teachers. Not all teachers are necessarily proficient at the task in question and we want to test whether our methodology can be applied in these cases. We evaluate if we are able to obtain an improved policy from the worst two teachers.

3. **Generalisation**. We learn a policy to insert a plug into socket A which was located at the center of the wooden wall. We test the generalisation of the policy in finding a new socket location. We further test whether the policy can generalise to two new sockets which were not used during the training phase.

We evaluate the above properties under two separate conditions. In the **first condition** we consider the period between the start of the search until the socket is localised. In the **second condition** we consider the period from the point the socket is found until a connection has been established. In the first condition the evaluation is done in simulation whilst in the second condition, when the socket is found, we perform the evaluation with a physical robot, the KUKA LWR4.

This choice is motivated by the fact that the two parts of the task require different levels of precision. Finding the socket requires much less precision than establishing a connection. It is thus more informative to consider the performance of these two parts separately. Another aspect is that the search for the socket can be reliably evaluated in simulation since the physics of the interaction is simple. The connection phase is more complicated and a simulation would be unrealistic. For the evaluation of the connection of the plug to the socket we consider the search start point already within the vicinity of the socket.

### 4.6.1 Distance taken to reach the socket's edge (Qualitative)

We consider three search experiments which we refer to as **Experiment 1, 2** and **3**, in order to evaluate the performance (distance travelled to reach the socket) of three search policies: GMM, Q-EM and Greedy. In these three experiments the task is considered accomplished when a search policy finds the socket's edge.

In **Experiment 1**, three starting locations are chosen: *Center*, *Left* and *Right*, see Figure 4.13, *Experiment 1*, 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.

In **Experiment 2**, two *Cases* are chosen in which the believed state (most likely state of the PMF) and the true position of the end-effector are relatively far apart. The location of the beliefs are chosen to be symmetric, see the Figure 4.13, *Experiment 2*. A total of 25 searches are carried for each of the two conditions.

In **Experiment 3**, Figure 4.13, *Experiment 3*, the initial true starting positions of the end-effector are taken from a regular grid 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.

We evaluate the performance of the three experiments in terms of the trajectories and their distribution in reaching the edge of the socket.

We can see a clear difference between the trajectories generated by the GMM and Q-EM policies in Experiment 1, see Figure 4.13 *Experiment 1, second row*. 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. The same effect can be seen in Experiment 2 (*second row*). The Q-EM trajectories follow a downward trend towards the location of the socket. The gradient is less due to the initial starting condition being lower than in Experiment 1.

The trajectories of the Greedy policy depend on the chosen believed location (most likely state of the PMF). In the second experiment there is no variance in the Greedy's trajectories until it reaches the edge of the red square, where the branching occurs as the believed location is disqualified. This happens as no sensation is registered when the believed location reaches the wall as the true location is before the believed location, see Figure 4.13, *Experiment 2, second row*.

In Figure 4.13 *Experiment 3, second row*, both Human and GMM distributions of searched locations are similar. 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.

The distribution of the searched locations of the Q-EM policy is centred around the origin of the $z$-axis, see Figure 4.13 *Experiment 3, second row*. The

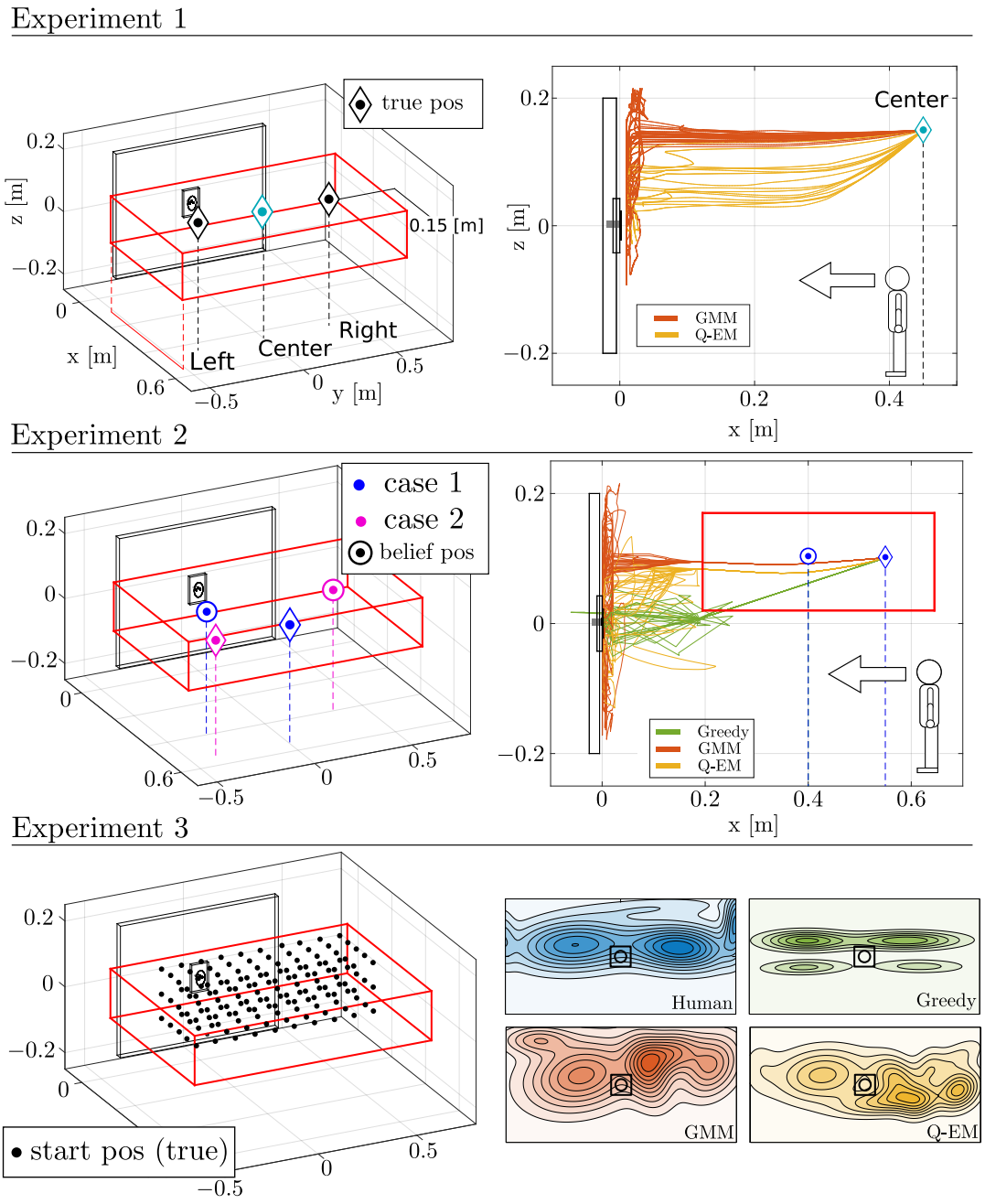



**Figure 4.13:** Three simulated search experiments. **Experiment 1:** Three start positions are considered: *Left*, *Center* and *Right* in which the triangles depict true position of the end-effector. The red cube illustrates the extent of the uncertainty. In the second row of Experiment 1, we illustrate the 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. **Experiment 2:** Two cases are considered: *Case 1* blue, the initial belief state (circle) is fixed facing the left edge of the wall and the true location (diamond) is facing the socket. *Case 2* pink, the initial belief state (circle) is fixed to the right facing the edge of the wall and the true location is the left edge of the wall. In the second row, the trajectories are plotted for *Case 1*. **Experiment 3:** A 150 start locations are deterministically generated from a grid in the start area. In the second row, we plot the distribution of the areas visited by the true position during the search.

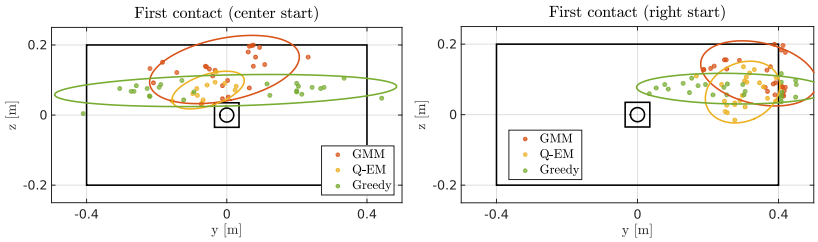



**Figure 4.14:** First contact with the wall, during experiment 1. (a) Contact distribution for initial condition "Center" . (b) Contact distribution for initial condition was "Right". The ellipses correspond to two standard deviations of a fitted Gaussian function.

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 whilst remaining 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.

In Figure 4.14 (*Top-left*), we illustrate the distribution of the first contact with the wall during Experiment 1 for the *Center* initial conditions. 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.

### 4.6.2 Distance taken to reach the socket's edge (Quantitative)

In Figure 4.15 we illustrate the quantitative results of the distance taken to reach the socket for all three experiments. In **Experiment 1**, 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.

In **Experiment 2**, the Q-EM search policy is the most efficient. For *Case 1* of Experiment 2, the initial most likely state is fixed to the left and the true position is facing the socket. As the belief is chosen to be to the left, upon contact with the wall the policy takes a left action since it is more likely to result

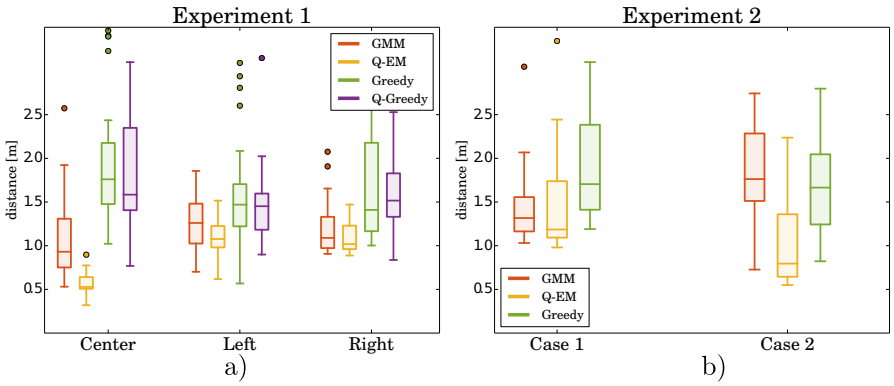



**Figure 4.15:** Distance travelled until the socket's edge is reached. a) Three groups correspond to the initial conditions: Center, Left and Right depicted in Figure 4.13, *top left*. The Q-EM method is always better than the other methods, in terms of distance. b) Results of the two initial conditions depicted in Figure 4.13, *top middle*, both the true position and most likely state are fixed. The Q-EM method always improves on the GMM.

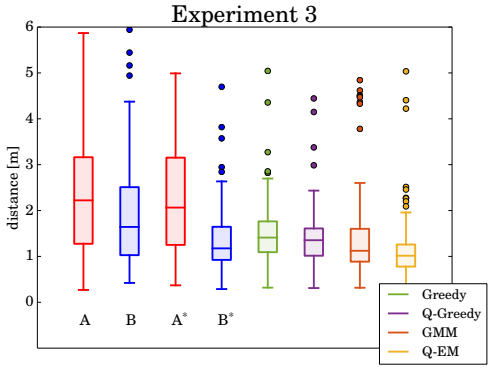



**Figure 4.16:** Distance travelled until the socket's edge is reached. Results corresponding to Experiment 3, Figure 4.13, *top right*. Again the Q-EM method is better, but at a less significant level.

in a localisation, given that the left edge of the wall is within close proximity. This on average results in an exploration in the upper left area of the wall, which explains why *Case 1* does worse than Experiment 1 for the *Center* initial condition. In *Case 2* however, where the true state is facing the left edge and the believed position is facing the right edge, less distance is taken to find the socket than it does for Case 1, Figure 4.15 (b), as reason for the improvement over Case 1, is that in *Case 2* the true location of the end-effector is close to an edge which is an informative feature and results in a much faster localisation.

From **Experiment 3**, Figure 4.16, it is clear that the three search policies have less variation in the distance travelled to find the socket's edge than the human teachers. All search policies are better than the human teachers with the exception of group $B^*$, 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 to the human teachers and Greedy policy. We made, however a critical assumption in order to be able to use our (RL-)PbD-POMDP 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. We next evaluate the validity of this assumption and the importance of the training data provided by the human teachers.

### 4.6.3 Importance of data

We perform two tests to evaluate the importance of the teachers training data for learning a search policy. Firstly we take the worst two teachers in terms of distance taken to find the socket's edge and learn a GMM and Q-EM policy separately from their demonstrations. In this way we can evaluate whether it is possible to learn a successful policy given a few bad demonstrations (15 training trajectories for each policy). Our second evaluation consists of using a noisy explorative Greedy policy as a teacher to gather demonstrations which can then be used to learn a new policy, which we call Q-Greedy.

Figure 4.17 illustrates 6 trajectories of teacher # 5. The human teacher preferred to localise himself at the top of the wall before either proceeding to a corner or going directly towards the socket. Once localised, the teacher would reposition himself in front of the socket and try to achieve an insertion. This behaviour was not expected since by losing contact with the wall, the human teacher no longer has sensory feedback which is necessary to maintain an accurate position estimate.

Figure 4.18 illustrates the value function of the belief state learned from the data of teacher # 5. The states with the highest values seem to create a path going from the socket towards the right edge of the wall. We proceed as before to learn a GMM policy from the raw data and a Q-EM policy in which the data points are weighted by the gradient of the value function. In Figure 4.19, we illustrate the resulting Marginalised Gaussian Mixture parameters for both the GMM and Q-EM policies and we plot 25 rollouts of each policy starting at the *Center* initial condition used in Experiment 1. We note that the trajectories of the GMM policy seem to have a lot of variance in contrast to the Q-EM policy, resulting from an access of variance amongst the 15 original demonstrations given by the teacher. To much variance is not necessarily good, a random policy will have to most variance (uniform) in terms of produced trajectories and will be extremely poor and achieving the goal. Furthermore there is insufficient

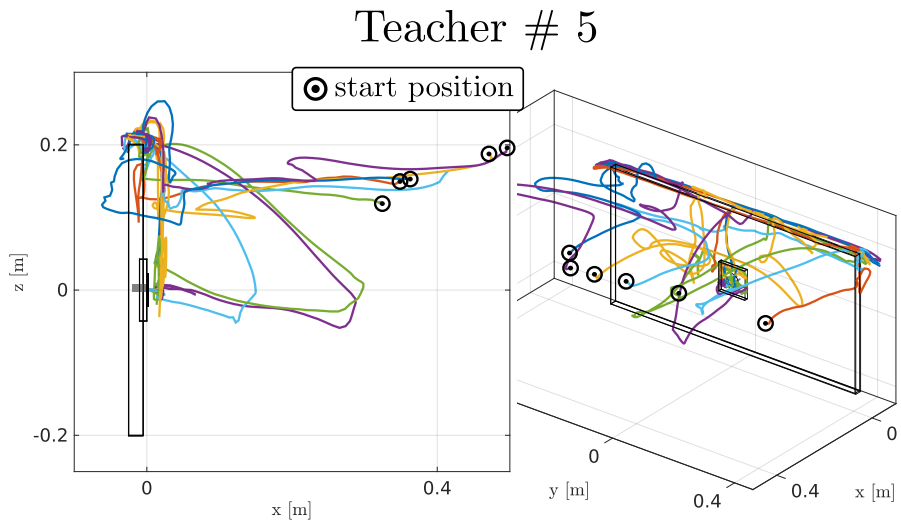



**Figure 4.17:** Demonstrations of teacher # 5. The teacher demonstrates a preference

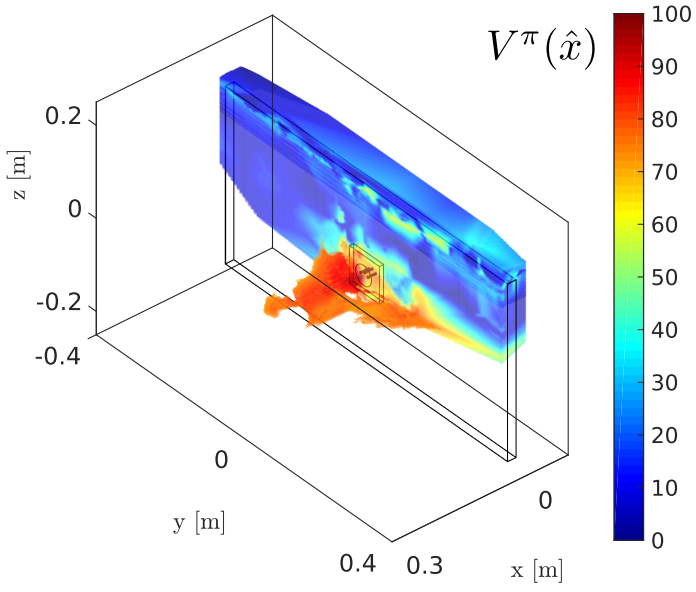



**Figure 4.18:** Value function learned from the 15 demonstrations of teacher #5. The value of the the most likely state is plotted.

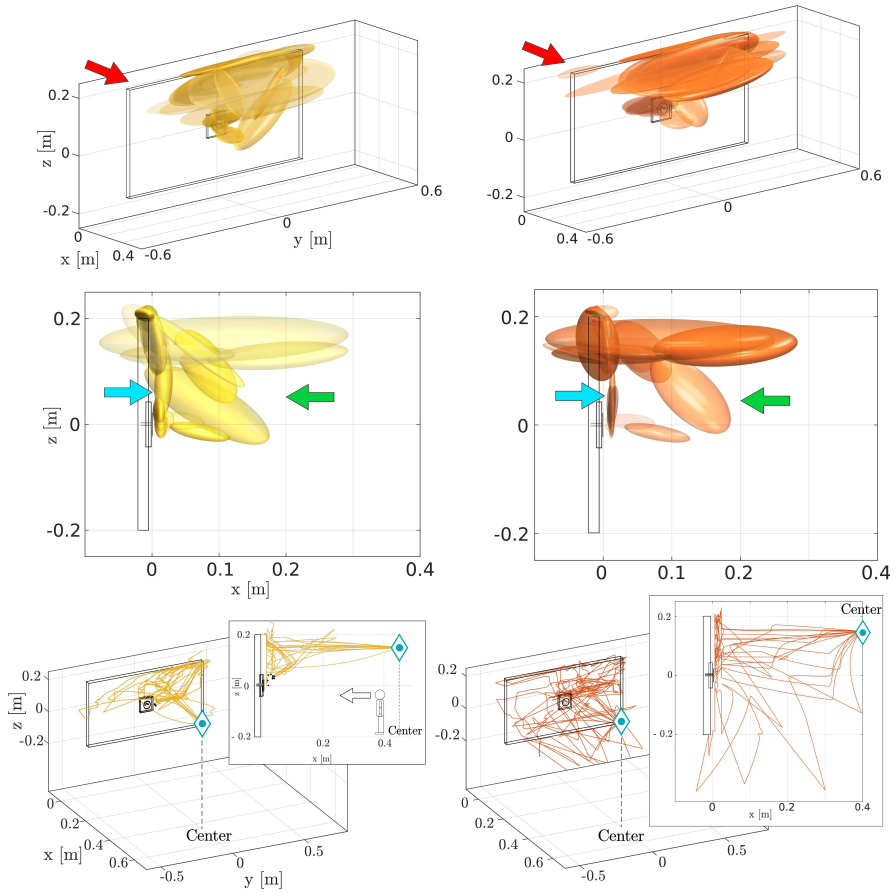



**Figure 4.19:** Marginalised Gaussian Mixture parameters of the GMM and Q-EM learned from the demonstrations of teacher #5. The illustrated transparency of the Gaussian functions is proportional to their weight. *Left column*: The Gaussian functions of the Q-EM have shifted from the left corner to the right. This is a result of the value function being higher in the top right corner region, see Figure 4.18. *Center column*: The original data of the teacher went quite far back which results in a Gaussian function given a direction which moves away from the wall (green arrow), whilst in the case of the Q-EM parameters this effect is reduced and moved closer towards the wall. We can also see from the two plots of the Q-EM parameters that they then follow the paths encoded by the value function. *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 patter, whilst the Q-EM policy gives many more consistent trajectories which replicate to some extent the pattern of making a jump (no contact with the wall) from the top right corner to the socket's edge.

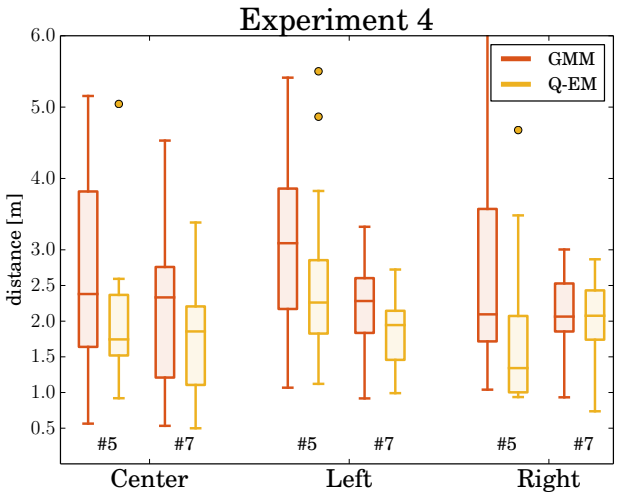


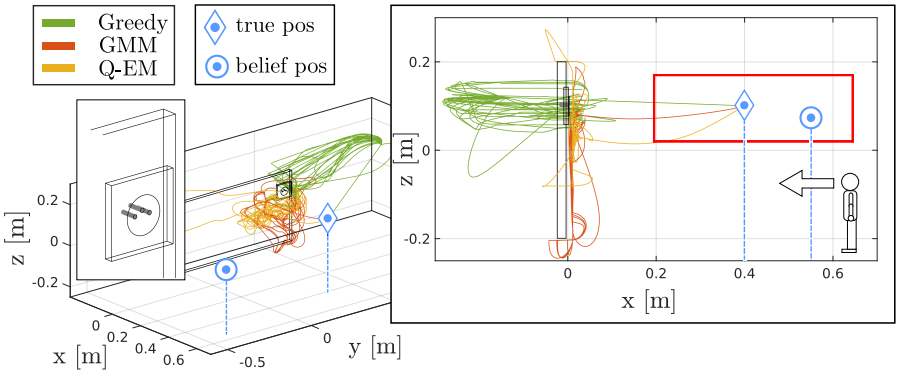

**Figure 4.20:** Results of a GMM and Q-EM policy under the same test conditions as Experiment 1. The Q-EM policy nearly always does much better than the GMM policy.

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 4.20, 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. We used the Q-Greedy algorithm in combination with random perturbations applied to the Greedy velocity, to act as a simple exploration technique. We performed a maximum of 150 searches, which terminated once the socket was found and used these demonstrations to learn a value function and GMM policy which we refer to as Q-Greedy. Figure 4.15 illustrates the statistical results of the Q-Greedy policy for Experiment 1 and 3, showing that there is no difference between two policies. Our exploration method is probably too simplistic to discover meaningful search patterns and we could probably devise better search strategies which would result in a good policy. However we have shown that human behaviour does already have a usable trade-off between exploration and exploitation which can be used to learn a new policy through our RL-PbD-POMDP framework.

### 4.6.4 Generalisation

An important aspect of a policy or any machine learning methodology is to be able to generalise. So far we have trained and evaluated our policy within



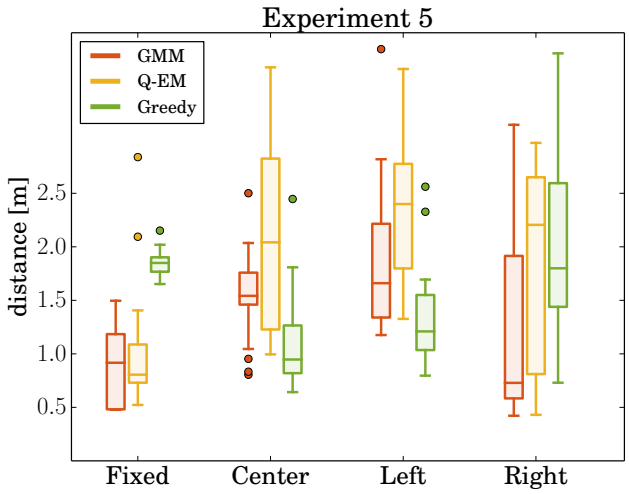

**(a)**

**Figure 4.21:** Evaluation of generalisation. The socket is located in at the top right corner of the wall. We consider a *Fixed* starting location for both the true and believed location of the end-effector. The red square depicts the extent of the initial uncertainty, which is uniform. (b) Distance taken to reach the socket's edge. For the Fixed setup (see (a) for the initial condition), both the Q-EM and GMM significantly outperform the Greedy. The other three conditions are the same as for Experiment 1.

the same environment. To test whether our GMM policies can generalise to a new setting we changed the location of the socket to the upper right corner of the wall. The GMM was trained in the frame of reference of the socket and when we translated the socket's location it also translated the policy.

To evaluate the generalisation of our learned policy we use the same initial conditions of Experiment 1 with an additional new configuration named *Fixed*, in which both the true and believed location are fixed, blue triangle and circle, see Figure 4.21, which 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 to both the GMM and Q-EM policies. From the statistical results shown in Figure 4.22 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 more likely will the search quality degrade. 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. The




(a)

**Figure 4.22:** Distance taken to reach the socket's edge. For the Fixed setup (see Figure 4.21) for the initial condition), both the Q-EM and GMM significantly outperform the Greedy.

policy will always be able to find the socket once it has localised itself. This is can be seen from the vector field of the policy, see Figure 4.10, when the entropy is low. In this case the policy acts like a point attractor.

### 4.6.5 Distance taken to connect the plug to the socket

In this section we evaluate the distance taken for the policies and humans to establish a connection, after the socket has been found. We start measuring the distance 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 demonstration of the human teachers. 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 4.24 for an illustration of the initial starting condition. This initial configuration was used to evaluate the search policies for three different sockets, see Figure 4.23 (a) for an illustration of the sockets. We kept the same initial configuration for the evaluation of the three sockets so that we can observe the generalisation properties of the policies. As a reminder we only used the training data from demonstrations acquired during the search with socket A. Socket B has a funnel which should make it easier to connect whilst socket C should be harder as it has no

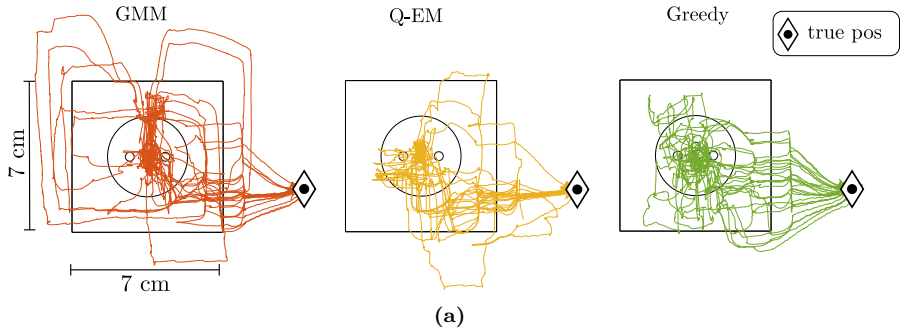
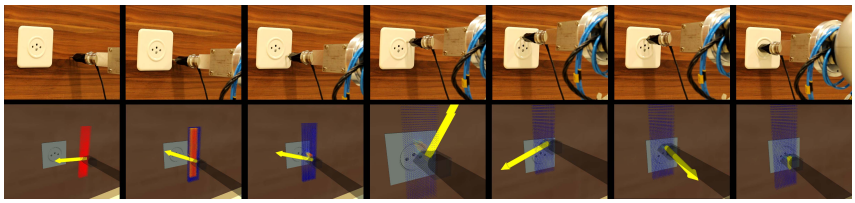



**Figure 4.23:** 25 search trajectories for each of the three search policies for socket A.

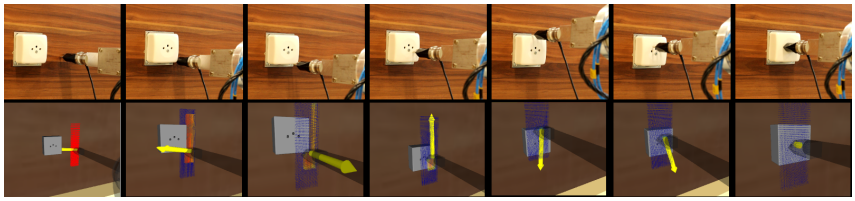
informative features on its surface.

For each of the sockets we performed 25 searches starting from the same initial condition. In Figure 4.23 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 4.25 (c) we can see that 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. When compared to the human's performance, all three search methods are vastly superior, see Figure 4.25. In Figure 4.24 we illustrate 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 vertical up and down motions. Only when the uncertainty has been reduced does the GMM policy try to 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 isn't 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. In Figure 4.25 (c) we illustrate the statistics of the distance taken to establish a connection for all three sockets. The point of interest is that both the GMM and Q-EM algorithms do 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 in the Greedy case 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



**(a)**



**(b)**

**Figure 4.24:** KUKA LWR4 equipped with a holder mounted with a ATI 6-axis force-torque sensor. (a) The robot's end-effector starts to the right of socket A. The second row are screen captures of the 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. (b) Same initial condition as in (a) but with socket C. 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 makes a connection.

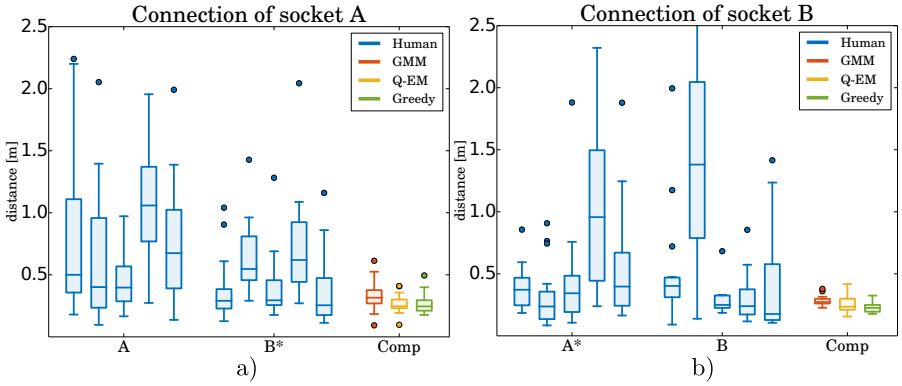



**Figure 4.25:** Distance taken to connect plug to socket once the socket is localised. (a) **Socket A**. The human Group A are the set of teachers who first started with socket A. They had no previous training on another socket beforehand. Group B$^*$ first gave demonstrations on Socket B before giving demonstrations on Socket A. Group B$^*$ is better than Group A 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. (b) **Socket B**. Both Groups A$^*$ and B are similar in terms of the distance they took to insert the plug into the socket and as was the case for (a), the search policies travel less to accomplish the task.

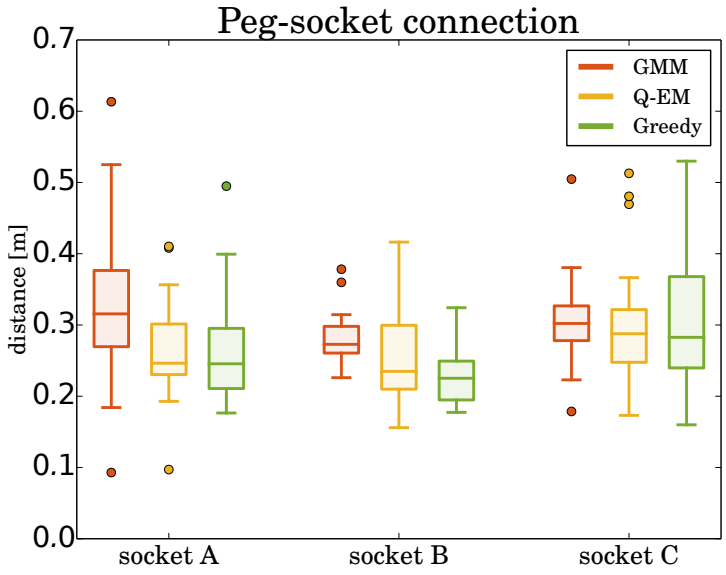



**Figure 4.26:** Distance taken (measured from point of contact of plug with socket edge) to connect the plug to the socket.

on socket B than A, since the funnel shape connector helps in maintaining the plug within the vicinity of the socket's holes.

The search discrepancy between the performance of the humans and 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. 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, we are far less precise than the KUKA which has sub-millimetre accuracy.

## 4.7 Discussion & Conclusion

In this work we learned search policies from demonstrations provided by human teachers for a task which consisted of first localising a power socket (either socket A, B or C) and then connecting it with a plug. Only haptic information was available as the teachers were blindfolded. We made the assumption that the position belief of the human teachers was initially uniformly distributed in a fixed rectangular region of which they were informed and is considered prior knowledge. All subsequent beliefs were then updated in a Bayesian recursion using the measured velocity obtained from a vision tracking system, and wrench acquired from a force torque sensor attached to the plug. The filtered probability density function, represented by a Point Mass Filter, was then compressed to the most likely state and entropy.

Two Gaussian Mixture Model policies where learned from the data recordedj

during the teaching by the human teachers . The first policy, called Q-EM, was learned in an Actor-Critic RL framework in which a value function was learned over the belief space. This was then used to weigh training datapoints in the M-step update of Expectation-Maximisation (EM). The second policy, called GMM, was learned using the standard EM algorithm, considering all training data points equally, following in the footsteps of our initial approach Chambrier and Billard (2014). Both the Q-EM and GMM policies were trained with data solely from the demonstrations of the search with socket A.

We evaluated 4 different aspects of the learned policies. Firstly, we evaluated which of three policies, Q-EM, GMM and a Greedy policy, took the least distance to find the socket. We concluded that across three different Experiments the Q-EM algorithm was always the best. It was clear that the Q-EM policy was less random and more consistent than the GMM policy as it tried to enter in contact with the wall at the same height as the socket thus increasing the chances of finding the socket.

Secondly, we tested the importance of the data provided by the human teachers. We took the worst two teachers and trained an individual GMM and Q-EM policy for each of them. We found that the performance of the Q-EM was better than the GMM in terms of distance travelled to find the socket. When qualitatively evaluating the trajectories of the GMM with respect to the Q-EM for the worst teacher, it is clear that the Q-EM policy managed to extract a search pattern, which was not the case for the GMM policy. We also tried to learn a Q-EM policy from the data provided by a Greedy policy with explorative noise and we found no improvement. From these results we conclude that the exploration and exploitation aspects of the trajectories provided by the human teachers is necessary.

Thirdly we tested whether the two policies were able to generalise to a different socket location. Under a specific condition, which we called *Fixed*, both policies were significantly better than the Greedy policy. However for the *Center* and *Left* initial conditions the Greedy policy was better. For the initial conditions in which the Greedy policy enters in contact with the wall at an early stage, it performs better than the GMM and Q-EM. The reason for this is that the actions taken by the Greedy policy in this setting will always result in a decrease of entropy when the location of the socket is close to a corner, as opposed to being in the center of the wall.

Fourthly we evaluated the three policies on the KUKA LWR4 robot. First all the policies did better than the human teachers. For socket A, on which both the GMM and Q-EM policies were trained, there is no clear distinction between the Q-EM and Greedy policy. On socket B, which was novel, the Greedy policy performed better than the statistical controllers, which we hypothesize was a result of a funnel which would make it easier for a myopic policy. For socket C, both the GMM and Q-EM policies do better than the Greedy, as socket C has no features on its surface, this being a disadvantage for a myopic policy.

We concluded by making the observation that by simply adding a binary reward function in combination with data provided by human demonstrations, with Fitted reinforcement learning, we can learn a better policy without the need of doing expensive exploration-exploitation rollouts traditionally associated with reinforcement learning and designing complicated reward functions. This is especially advantageous when only a few demonstrations are available.

## 4.8 Appendix

### 4.8.1 Time to connect socket (ANOVA)

The null hypothesis is that there is no difference in the time taken to connect socket A or B. The result of the one-way anova is given in Table 4.1. The null hypothesis is rejected at high significant level (***) indicating that there is a difference between the sockets. According to the linear model it takes on average 4 seconds less to connect socket B than it does for socket A.

| | F | Pr(>F) |
|---|---|---|
| socket | 13.9 | 0.000232 *** |

**Table 4.1:** One way anova of the time taken to connect two sockets A and B. There is a significant difference.

We tested whether the group order had an effect on the connection time. We fitted a linear model with the predictor being time and the factors being the group (One or Two), the socket type (A or B) and the subject's ID (1 to 10):

$$time = \beta_0 + \beta_1\, group + \beta_2\, socket + \beta_3\, subject \tag{4.8.1}$$

and did the corresponding anova analysis on this linear model. We found that the difference between the sockets remained significant.

### 4.8.2 EM policy search

Steps taken to make a policy $\pi_{\boldsymbol{\theta}}(\dot{x}, b)$ maximise the objective function, $J(\boldsymbol{\theta})$. The policy will be maximised with respect to the lower bound of the cost function $J(\boldsymbol{\theta})$:

$$\begin{aligned} J(\boldsymbol{\theta}') &= \sum_{i \in \mathbb{T}} \pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]}) \\ &= \sum_{i \in \mathbb{T}} \frac{\pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]})}{\pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})}\, \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]}) \end{aligned} \tag{4.8.2}$$

where $\mathbb{T}$ is the set of all rollouts. Next we take the logarithm and make use of Jensen's inequality and move the logarithm into the summation.

$$\begin{aligned}\log(J(\boldsymbol{\theta}')) &= \log \sum_{i \in \mathbb{T}} \frac{\pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]})}{\pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})} \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]}) \\ &\geq \sum_{i \in \mathbb{T}} \log \left( \frac{\pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]})}{\pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})} \right) \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]}) \end{aligned} \tag{4.8.3}$$

We take the derivative of the lower bound of $\log(J(\boldsymbol{\theta}'))$, Equation 4.8.3, with respect to $\boldsymbol{\theta}'$ and set it to zero so as to maximise the cost function.

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}'} \log(J(\boldsymbol{\theta}')) = \\ &\sum_{i \in \mathbb{T}} \nabla_{\boldsymbol{\theta}'} \log \left( \pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]}) \right) \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]}) \\ &- \underbrace{\nabla_{\boldsymbol{\theta}'} \log \left( \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]}) \right) \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]})}_{=0} \\ &= \sum_{i \in \mathbb{T}} \nabla_{\boldsymbol{\theta}'} \log \left( \pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]}) \right) \pi_{\boldsymbol{\theta}}(\dot{x}^{[i]}, b^{[i]})\, R(\dot{x}^{[i]}, b^{[i]}) \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\dot{x}, b)} \Big\{ \nabla_{\boldsymbol{\theta}'} \log \left( \pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]}) \right) R(\dot{x}^{[i]}, b^{[i]}) \Big\} \end{aligned} \tag{4.8.4}$$

$$\nabla_{\boldsymbol{\theta}'} \log(J(\boldsymbol{\theta}')) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(\dot{x}, b)} \Big\{ R(b^{[i]}, \dot{x}^{[i]}) \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}'} \log \pi_{\boldsymbol{\theta}'}(\dot{x}^{[i]}, b^{[i]}) \Big\} \tag{4.8.5}$$

$$= \sum_{i=1}^{N} \sum_{t=0}^{T^{[i]}} \nabla_{\boldsymbol{\theta}'} \log \pi_{\boldsymbol{\theta}'}(\dot{x}_t^{[i]}, b_t^{[i]})\, Q^{\pi}(\dot{x}_t^{[i]}, b_t^{[i]}) \tag{4.8.6}$$

The reader is referred to Deisenroth et al. (2013a) for more details regarding Expectation-Maximisation and policy search in reinforcement learning.

### 4.8.3 Q-EM for GMM derivation

Making the substitution $x = (\dot{x}, b)^{\mathrm{T}}$ (small abuse of the notation) and insuring a positive Q-function, $Q^{\pi}(x^{[m]}) \geq 0$ and by setting the derivative of Equation 4.8.6 to zero and solving for the parameters $\boldsymbol{\theta} = \{w, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ we get a new weighted Maximisation update step in EM:

$$\nabla_{\boldsymbol{\mu}^{[k]}} \log J(\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha(z_{mk})\, Q(x^{[m]})\, {\boldsymbol{\Sigma}^{[k]}}^{-1} (x^{[m]} - \boldsymbol{\mu}^{[k]}) = 0$$

$$\boldsymbol{\mu}_{\mathbf{new}}^{[k]} = \frac{\sum_{m=1}^{M} \alpha(z_{mk})\, Q(x^{[m]})\, x^{[m]}}{\sum_{j=1}^{M} \alpha(z_{jk})\, Q(x^{[j]})} \tag{4.8.7}$$

where $\alpha(z_{mk})$ is the responsibility factor, denoting the probability that data point $m$ is a member of the Gaussian function $k$.

$$\alpha(z_{mk}) = \frac{w^{[k]} \cdot g(x^{[m]}; \boldsymbol{\mu}^{[k]}, \boldsymbol{\Sigma}^{[k]})}{\sum_{j=1}^{K} w^{[j]} \cdot g(x^{[m]}; \boldsymbol{\mu}^{[j]}, \boldsymbol{\Sigma}^{[j]})} \tag{4.8.8}$$

$$\boldsymbol{\Sigma}_{\mathbf{new}}^{[k]} = \frac{\sum_{m=1}^{M} Q(x^{[m]}) \alpha(z_{mk}) (x^{[m]} - \boldsymbol{\mu}^{[k]})(x^{[m]} - \boldsymbol{\mu}^{[k]})^{\mathrm{T}}}{\sum_{j=1}^{M} Q(x^{[j]})\, \alpha(z_{jk})} \tag{4.8.9}$$

$$w_{\mathbf{new}}^{[k]} = \frac{\sum_{m=1}^{M} Q(x^{[m]})\, \alpha(mk)}{\sum_{j=1}^{M} Q(x^{[j]})} \tag{4.8.10}$$

### 4.8.4 Unbiased estimator

The temporal difference error is an unbiased estimate of the advantage function:

$$\begin{aligned} \mathop{\mathbb{E}}_{\pi_{\boldsymbol{\theta}}} \{\delta_t^{\pi} | b_t, u_t\} &= \mathop{\mathbb{E}}_{\pi_{\boldsymbol{\theta}}} \{r_{t+1} + \gamma V^{\pi}(b_{t+1}) | b_t, u_t\} - V^{\pi}(b_t) \\ &= Q^{\pi}(b_t, u_t) - V^{\pi}(b_t) \\ &= A^{\pi}(b_t, u_t) \end{aligned} \tag{4.8.11}$$