Human demonstrations for fast and safe exploration in reinforcement learning

Schonebaum, Gerben; Junell, Jaime; van Kampen, Erik Jan

DOI
10.2514/6.2017-1069

Publication date
2017

Document Version
Accepted author manuscript

Published in
AIAA Information Systems-AIAA Infotech at Aerospace, 2017

Postprint of: Gerben Schonebaum, Jaime Junell, and Erik-Jan Van Kampen. "Human Demonstrations for Fast and Safe Exploration in Reinforcement Learning", AIAA Information Systems-AIAA Infotech @ Aerospace, AIAA SciTech Forum, (AIAA 2017-1069)
https://doi.org/10.2514/6.2017-1069

Human Demonstrations for Fast and Safe Exploration in Reinforcement Learning

Gerben Schonebaum*, Jaime Junell† and Erik-Jan van Kampen‡
Delft University of Technology, Delft, The Netherlands

Reinforcement learning is a promising framework for controlling complex vehicles with a high level of autonomy, since it does not need a dynamic model of the vehicle, and it is able to adapt to changing conditions. When learning from scratch, the performance of a reinforcement learning controller may initially be poor and –for real life applications– unsafe. In this paper the effects of using human demonstrations on the performance of reinforcement learning is investigated, using a combination of offline and online least squares policy iteration. It is found that using the human as an efficient explorer improves learning time and performance for a benchmark reinforcement learning problem. The benefit of the human demonstration is larger for problems where the human can make use of its understanding of the problem to efficiently explore the state space. Applied to a simplified quadrotor slung load drop off problem, the use of human demonstrations reduces the number of crashes during learning. As such, this paper contributes to safer and faster learning for model-free, adaptive control problems.

Nomenclature

Symbols

s	State
a	Action
π	Policy
r	Immediate reward
$f(s,a)$	Transition function
$\rho(s,a)$	Reward function
R	Return
ϵ	Exploration rate
γ	Discount factor
$Q(s,a)$	State-action value
$\phi(s,a)$	Basis function
w	Basis function weight
η	Forget factor
K_θ	Update interval
Γ, Λ, z	Matrices containing sample information

Abbreviations

MAV	Micro aerial vehicle
RL	Reinforcement learning
LSPI	Least squares policy iteration
IOLSPI	Informed online least squares policy iteration
HD	Human demonstration
BF	Basis function
RBF	Radial basis function

I. Introduction

In recent years the need for automation and autonomy in all sorts of vehicles and systems has been growing. Especially in the development of controllers for Micro Aerial Vehicles (MAVs) autonomy has large benefits.[1] Making such vehicles fully autonomous is difficult because it means that they have to perform their missions with no human intervention. As a result, it is necessary that the vehicle and its control system are inherently safe, and are able to adapt to changing conditions. The dynamics of some of these vehicles, in particular small and flapping wing MAVs such as the Delfly,[2] are complex. For these vehicles, there is a need for model-free controllers to circumvent the problems arising in model-based controllers because of errors in modeling.[3]

*Master Student, TU Delft Aerospace Engineering, Control & Simulation. Delft. The Netherlands.
†PhD Candidate, TU Delft Aerospace Engineering, Control & Simulation. Delft. The Netherlands, Student Member AIAA.
‡Assistant Professor, TU Delft Aerospace Engineering, Control & Simulation. Delft. The Netherlands, Member AIAA.

Reinforcement Learning (RL) is a promising method to meet these demands, since it can be applied without a model of the system dynamics. In addition, when used online, RL controllers can adapt to changing conditions.[4] In RL the controller receives scalar rewards from the environment and uses these to improve its control policy. The goal for the controller is to develop a policy that maximizes the return, which is the sum of all rewards, received from its current state onward. RL is generally referred to as a method which is applied without using prior knowledge. However, learning a policy from scratch can be a time consuming task, in which there is little control over the parts of the state space that will be visited. In many real life applications, such as flying with MAVs, this exploration could lead to unsafe situations while the controller is still learning towards a good control policy, and in the worst-case result in a crash.[5]

The formalization of RL allows for using prior knowledge to accelerate the learning. Successful attempts have been made to include knowledge on the dynamics of the system to speed up the learning.[6,7] These attempts used different kinds of knowledge on the system dynamics to do so. Another method to improve the speed of learning is the use of human demonstrations to explore the state space of a system. Several attempts have been made to do so,[7,8] however, these attempts did use a model of the dynamics of the system at hand. A method that accelerates the reinforcement learning in a model-free and online fashion through the use of human demonstrations, has not yet been found.

The goal of this research is to find out in which way human demonstrations can be used to reduce the learning time, and improve the safety of online, model-free reinforcement learning algorithms. The RL algorithm proposed in this research is purely model-free, which is useful for vehicles with complex dynamics such as MAVs. In addition, it uses basis functions to approximate the state-action values which allows for the application to continuous state systems.[9] The algorithm uses a combination of two state of the art RL techniques: Offline Least Squares Policy Iteration[10] (LSPI) that is able to use the information of the human demonstration sample, and online LSPI[11] which continues learning online, and hence allows to adapt to changing conditions. This combined algorithm is called Informed Online Least Squares Policy Iteration (IOLSPI).

The contribution of this research is the implementation of this model-free and online IOLSPI algorithm, resulting in faster and safer exploration in RL. To test the proposed algorithm, a standard RL benchmark task is selected: the rotational pendulum. In this standard problem it is shown that the learning time could be significantly reduced by using human demonstrations in IOLSPI. Furthermore, it is shown that while using human demonstrations, the algorithm is still able to adapt to changing conditions. However, it is found that the benefit of the human demonstrations is not the same for each problem.

The rotational pendulum problem includes no intuitive aspect of safety (there is no clear unsafe part of the state space), therefore, an additional problem is reviewed: the control of a quadrotor which carries a slung load. This combined vehicle has complex dynamics. In a task of dropping a load in a target, a capable human expert could demonstrate good behavior. In this example the relevance of safety in exploring becomes more apparent, since faulty policies result in dropping the load at the wrong location, or crashing the quadrotor. It is shown that using human demonstrations the number of crashes while learning is reduced for this quadrotor slung load problem. Reinforcement learning has already been applied to quadrotors with a slung load.[3,12–14] All these studies use a model of the system dynamics. In this paper, a two-dimensional version of this slung load problem is used with a model-free reinforcement learning controller.

This paper is structured as follows: In section II, a short background on reinforcement learning and some of its basic concepts and notations are introduced. Section III describes the used reinforcement learning algorithm and methodology. Section IV gives an overview of the most important results, and section V presents the conclusions of the paper.

II. Reinforcement Learning Preliminaries

In reinforcement learning a Markov Decision Process is considered, with a state space S and an action space A. The RL agent starts in state s_k and takes action a_k according to its current policy π_l:

$$a_k = \pi_l(s_k) \tag{1}$$

The transition function f, which describes the dynamics of the system, gives the next state s_{k+1}:

$$s_{k+1} = f(s_k, a_k) \tag{2}$$

Furthermore, the agent receives a reward r_{k+1} which follows from the reward function ρ:

$$r_{k+1} = \rho(s_k, a_k) \tag{3}$$

The goal for the agent is to find a policy π that maximizes the expected return R in each state. In Eq. (4) the return is defined.

$$R^{\pi_l}(s_0) = \sum_{k=0}^{K} \gamma^k \rho(s_k, \pi_l(s_k)) \tag{4}$$

Where K is the number of time steps and $\gamma \in [0, 1]$ is the discount factor, ensuring that the further in the future a reward is received, the less it contributes to the return at the current state. The expected return for a certain policy π_l, starting from state s_k, applying action a and using policy π_l onward is given by the Q-function:

$$Q^{\pi}(s_k, a) = E\{R^{\pi}(s_k)|a\} \tag{5}$$

Using the Q-function, the greedy policy for that Q-function can be found with the following equation:

$$\pi_{l+1}(s_k) = \arg\max_a Q^{\pi_l}(s_k, a) \tag{6}$$

In many practical applications, the state and action spaces are large or uncountable, and as such it becomes intractable to describe the Q-value for each combination of state and action separately.[9] In these cases an approximation of the Q-value is made using a function approximator. Usually a linear combination of basis functions (BFs) $\phi(s, a)$ is used. These BFs are weighted with a tuning vector w_l to approximate the Q-values $\hat{Q}(s, a)$:

$$\hat{Q}(s, a) = \phi^T(s, a) w_l \tag{7}$$

In policy iteration algorithms two steps are alternated. First the policy is evaluated, i.e. the Q-function is determined. This is done by solving the Bellman equation (Eq. (8)).[15] After this evaluation step the policy is updated using equation (6). This new policy is then evaluated again, and so on.

$$Q^{\pi_l}(s_k, a) = \rho(s_k, a) + \gamma Q^{\pi_l}(f(s_k, a), \pi_l(f(s_k, a))) \tag{8}$$

In a model-free algorithm, the Bellman equation cannot be solved since the reward function $\rho(s_k, a_k)$ and transition function $f(s_k, a_k)$ are unknown. However, the transition and reward information contained in a set of samples D collected for the system can be used to approximate the Bellman equation. D contains n_s samples constituted of $\{s_k, a_k, r_{k+1}, s_{k+1}\}$. The Bellman equation can be formulated in matrix sense for a set of samples D (Eq. (9)). For a more elaborate discussion on this approach we refer to the paper by Lagoudakis.[10]

$$\Gamma w_l = z + \gamma \Lambda w_l \tag{9}$$

The matrices Γ and Λ and vector z in Eq. (9) are defined by Eqs. (10) - (12), and contain the information in the transition samples.

$$\Gamma_{k+1} \leftarrow \Gamma_k + \phi(s_k, a_k)\phi^T(s_k, a_k) \tag{10}$$

$$\Lambda_{k+1} \leftarrow \Lambda_k + \phi(s_k, a_k)\phi^T(s_{k+1}, \pi_l(s_{k+1})) \tag{11}$$

$$z_{k+1} \leftarrow z_k + \phi(s_k, a_k) r_{k+1} \tag{12}$$

To approximate the Q-function for a certain policy, the least square error in the modified Bellman equation (Eq. (9)) is minimized.[15] A policy evaluation algorithm called LSTD-Q[10] (Least Squares Temporal Difference for Q-functions) solves Eq. (9), which can be slightly modified to Eq. (13):

$$\frac{1}{n_s}\Gamma w_l = \frac{1}{n_s} z + \frac{1}{n_s}\gamma \Lambda w_l \tag{13}$$

This modified equation is mathematically not different from Eq. (9), but improves the numerical stability.[9]

After the policy evaluation, the policy is updated using Eq. (6). The new policy is evaluated again using LSTD-Q. This policy iteration method is called Least Squares Policy Iteration.[10]

The approach described has an interesting characteristic. The policy is defined implicitly by the Q-function: for each state the greedy action maximizes the Q-value (Eq. (6)). In practice the greedy action only needs be calculated for the state in which the action has to be taken, and not for the whole state space. This approach makes a method to explicitly define the policy for each state redundant. Since the policy is defined with respect to the Q-value (Eq. (6)) and an approximation of the Q-value with BFs is used, the policy is at any moment fully defined by the current BF weights w_l.

III. Least Squares Policy Iteration

In this paper the Least Squares Policy Iteration (LSPI) algorithm is used. There are different ways described in literature in which this algorithm has been implemented. Lagoudakis and Parr[10] invented the algorithm for offline use. Busoniu et al. altered the algorithm to use it online. This online algorithm will be referred to as Online Least Squares Policy Iteration[11] (OLSPI). Both these approaches will form the foundation for the integrated algorithm that is proposed in this paper, which uses offline LSPI to learn from a batch of pre-collected samples, and afterwards continues with gathering samples online. This combined algorithm is called Informed Online Least Squares Policy Iteration (IOLSPI).

A. Offline LSPI algorithm

Algorithm 1 shows the offline LSPI algorithm.[10] This algorithm uses a batch of samples to learn a new policy. These samples can be generated using a simulator, or they can be obtained as recordings during the use of the physical system at hand. This algorithm starts with no knowledge of the problem or the policy in advance. The matrix Λ_0 and vector z_0 are thus initialized with zeros. The matrix Γ_0 is the identity matrix multiplied with a small constant δ, to make sure it is invertible. In line 3, the policy weights are randomly initialized. This random initialization gives the need to run the algorithm for several random seeds and average its results. In lines 6-8, the complete batch of samples is processed according to equations (10) - (12). After the whole batch is processed, the policy update follows in line 10. This process is repeated until either the policy has converged, such that the difference in weights has dropped below the stopping criterion χ, or a maximum number of iterations l_{max} is reached. The output of this algorithm is a policy which uses the information of the presented sample batch. Note that the number of samples is directly linked to the computation time needed to run the algorithm.

Algorithm 1 Offline LSPI[10]

1: **Input:** n BFs $\phi_1(s,a),\ldots,\phi_n(s,a)$; γ; δ; Sample set D with n_s samples; χ
2: $\Gamma_0 \leftarrow \delta I_{nxn}$; $\Lambda_0 \leftarrow 0_{nxn}$; $z_o \leftarrow 0_{nx1}$
3: $l \leftarrow 0$; initialize weights (policy) w_0
4: **while** $l < l_{max}$ and $\Delta w > \chi$ **do**
5: **for** $k = 0 : n_s$ **do**
6: $\Gamma_{k+1} \leftarrow \Gamma_k + \phi(s_k,a_k)\phi^T(s_k,a_k)$
7: $\Lambda_{k+1} \leftarrow \Lambda_k + \phi(s_k,a_k)\phi^T(s_{k+1},\pi_l(s_{k+1}))$
8: $z_{k+1} \leftarrow z_k + \phi(s_k,a_k)r_{k+1}$
9: **end for**
10: $w_{l+1} \leftarrow$ solve $\frac{1}{k+1}\Gamma_{k+1}w_l = \gamma\frac{1}{k+1}\Lambda_{k+1}w_l + \frac{1}{k+1}z_{k+1}$
11: $\Delta w = ||w_l - w_{l+1}||$
12: $l \leftarrow l + 1$
13: **end while**

B. Online LSPI algorithm

Busoniu et al. have extended the offline LSPI algorithm to a variant that can be used online.[11] In online LSPI the algorithm gathers its samples online to update the policy, contrary to offline LSPI. In Algorithm 2 the working principle of Online LSPI is shown. It is similar to Algorithm 1, but some clear differences are

present. Since the algorithm runs online, it starts to update its policy after having processed a couple of samples, contrary to offline LSPI, which updates its policy only after having processed all of its samples. In online LSPI this policy update happens every K_θ time steps (line 10). The larger this update interval K_θ is, the longer the algorithm waits before the policy is updated. On the other hand, when K_θ becomes small, ultimately 1, this means that after each time step the policy is updated, which makes the algorithm slower. With a K_θ of 1 the update is called fully optimistic.[16] In general a partially optimistic update is used.[12]

Another feature of the online LSPI algorithm is that there is the need for exploration, as for all online RL algorithms.[17] Without exploration, the agent would always take the same action at the same state, and as such the estimate of the Q-function would be poor. Furthermore, exploration allows gathering samples in all parts of the state space, which improves the Q-function estimate. As can be seen in line 5 of algorithm 2, at each time step, with a probability of ϵ a random (exploratory) action is taken. Such a policy is called an ϵ-greedy policy.[4] If $\epsilon = 1$, the action is chosen fully at random, if $\epsilon = 0$, the action is chosen fully greedy. Different schemes for this exploration rate can be used. In online LSPI it makes sense to start with a high exploration rate, and make it decay over time. In this way at the beginning, when the policy is not good yet, many different samples are gathered to improve the Q-function estimate. Gradually it becomes more exploiting, to get better performance. The exploration rate should not decay to zero (fully greedy) however, since then the algorithm is not able to adapt to changes in the conditions. An example of an exponential decaying exploration rate that could be used is presented in Eq. (14).[11]

$$\epsilon_k = \epsilon_0 \epsilon_d^{kT_s} \tag{14}$$

In this equation ϵ_0 is the initial exploration rate, $\epsilon_d \in [0, 1]$ the exponential decay factor and T_s the sampling time.

Following from the need for adaptation to changing conditions, in addition to the algorithm presented in [11], a forget factor η is added (line 7-9). This forget factor makes sure that recently gathered samples have a higher impact on the policy than old ones. This factor is added to make sure that the algorithm is able to change along with changes in conditions by not carrying along too much outdated information.

Algorithm 2 Online LSPI[11, 12] with ϵ-greedy exploration and forget factor

1: **Input:** n BFs $\phi_1(s,a), \ldots, \phi_n(s,a)$; γ; $\{K_\theta\}_{k\geq 0}$; $\{\epsilon_k\}_{k\geq 0}$; Γ_0; Λ_0; z_0; w_0
2: $l \leftarrow 0$
3: Measure initial state s_0
4: **for** every time step $k = 0, 1, 2, \ldots$ **do**
5: $a_k \leftarrow \begin{cases} \text{exploit: } \arg\max_a \phi^T(s,a) w_l & \text{with prob } 1-\epsilon_k \\ \text{explore: random action} & \text{with prob } \epsilon_k \end{cases}$
6: Apply a_k and observe next state s_{k+1} and reward r_{k+1}
7: $\Gamma_{k+1} \leftarrow \eta\Gamma_k + \phi(s_k, a_k)\phi^T(s_k, a_k)$
8: $\Lambda_{k+1} \leftarrow \eta\Lambda_k + \phi(s_k, a_k)\phi^T(s_{k+1}, \pi_l(s_{k+1}))$
9: $z_{k+1} \leftarrow \eta z_k + \phi(s_k, a_k) r_{k+1}$
10: **if** $k = (l+1)K_\theta$ **then**
11: $w_l \leftarrow$ solve $\frac{1}{k+1}\Gamma_{k+1} w_l = \gamma \frac{1}{k+1}\Lambda_{k+1} w_l + \frac{1}{k+1} z_{k+1}$
12: $l \leftarrow l + 1$
13: **end if**
14: **end for**

An advantage of the online LSPI algorithm is that the computational effort to update the policy is not dependent on the number of samples processed. As such the time the online LSPI algorithm has been running has no influence on the required computation time for the policy updates.

It should be noted that this algorithm works under the assumption that the Q-function does not change too much for subsequent policies.[11] This means that the previous values of the matrices Λ and Γ and the z vector are also representative for the next policy. Using the forget factor does change this assumption a little, since the part in these matrices corresponding to the samples that already have been forgotten, does not influence the Q-function anymore.

C. Informed Online LSPI algorithm

In this paper the online and offline LSPI algorithms are combined. Using a set of samples, which can either be generated with a simulator, or obtained from a human demonstration on the physical system, the offline algorithm is run to find a satisfactory policy. Additionally, the samples processed offline are contained in the Λ and Γ matrix and the z vector. After running the offline LSPI algorithm, the online LSPI algorithm is used. In this Informed Online Least Squares Policy Iteration (IOLSPI) algorithm, contrary to regular online LSPI, the policy found using offline LSPI is used as initial policy for the online part of the algorithm. Furthermore, the Λ and Γ matrix and the z vector that were found after having run the offline LSPI are used instead of an empty initialization as is done in regular online LSPI. The information of the offline samples is contained in the matrices, and as such it does not start to learn from scratch. Moreover, the online algorithm still is able to learn from the samples gathered online, and as such is able to adapt to changing conditions.

IV. Experimental Study

The IOLSPI algorithm is tested on two experimental studies. The first problem setup studied is a rotational pendulum, with two state and one action variable. This is a common benchmark problem for reinforcement learning controllers. The second application is a simplified representation of a quadrotor with a slung load, which has four state and one action variable. This application is used to show the safety aspect in learning and to evaluate the performance of the algorithm for high-dimensional problems. The performance for the example problems is evaluated in two ways. Firstly, the intermediate policies are evaluated using a simulator, while learning and exploration are turned off. The discounted return is calculated, for each of the initial conditions. The discounted return for all different initial conditions is averaged. Secondly, the sum of the rewards gathered per episode while learning is presented to show the performance of the controller online. The results show the quality of the found policies. In the quadrotor problem, a third performance measure is added: The number of crashes while learning, which quantifies the safety.

A. Rotational Pendulum

The rotational pendulum is the same system as described by Busoniu et al.,[9] see figure 1. It consists of a DC motor with a horizontal spinning axis, with an eccentric mass attached. The goal is to make the pendulum swing up the mass and stabilize it pointing upwards. The state contains the angle and the angular rate: $s = [\alpha, \dot{\alpha}]^T$. The angle α is zero radians when the mass is pointing up, and π rad when it is hanging down in the middle. At the exact bottom, α transitions from π rad to $-\pi$ rad. The maximum angular rate of the rotational pendulum is 15 π rad/s.

There are three possible actions for this system: Apply a clockwise torque (+3V), a counterclockwise torque (-3V), or no torque (0V). Not for all states the torque of the motor is sufficient to push the pendulum upright in one go. For some states, for example hanging down, back and forth swings are needed to swing up the mass to the upright position. The rotational pendulum has the following properties: The mass moment of inertia of the disc J is 1.91 e-4 kgm^2, the mass of the pendulum m is 0.055 kg. The distance between the shaft and the mass l is 0.042 m, friction coefficient b = 3 e-6 Nms/rad, torque constant $K = 0.0536$ Nm/A and the internal resistance of the motor $R = 9.5\ \Omega$. The gravitational acceleration g is assumed to be 9.81 m/s^2. The equation of motion of the rotational pendulum is given by Eq. (15).

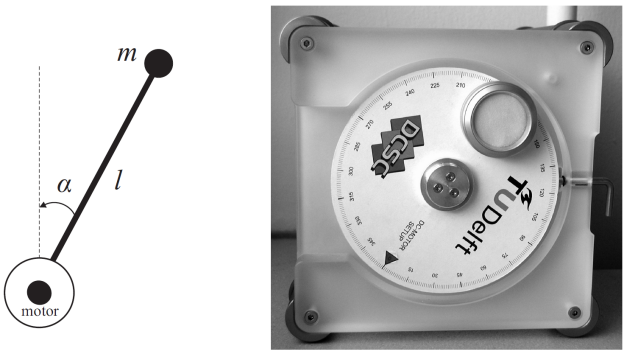

Figure 1: Schematic rotational pendulum (left), real rotational pendulum (right)[11]

$$\ddot{\alpha} = \frac{m \cdot g \cdot l \sin\alpha - b\dot{\alpha} - \frac{K^2\dot{\alpha}}{R} + \frac{K \cdot a}{R}}{J} \tag{15}$$

This problem will be simulated for 600 s, with episodes of 1.5 s, after which the state is reinitialized.

The sampling time is 0.005 s. The initial conditions (ICs) α_0 and $\dot{\alpha}_0$ used during learning are randomly selected from the following sets: $\alpha_0 = \{\pi - 0.1, \pi, \pi + 0.1\}$ rad, and $\dot{\alpha}_0 = \{$-π, $\frac{-2\pi}{3}$, $\frac{-\pi}{3}$, 0, $\frac{\pi}{3}$, $\frac{2\pi}{3}$, $\pi\}$ rad/s. With this set of initial conditions the pendulum always starts around the hanging down position with a low angular velocity, and as such always needs multiple swings to reach the goal state. Because of the partly random character of the algorithm, each result in this section is repeated 50 times with different random initializations.

In this problem a reward structure is used that penalizes an offset of the angle with respect to the goal, the angular rate and the control action (Eq. (16)). The values in this reward function were found trying different combinations, and shows a behavior that is mostly inclined to limit the error in angle.

$$r_{k+1} = s^T \begin{bmatrix} 5 & 0 \\ 0 & 0.01 \end{bmatrix} s + 0.01a^2 \tag{16}$$

To approximate the Q-function, an equidistant grid of 11x11 radial basis functions (RBFs) is used. This means that the vector of RBFs is $\bar{\phi}(s) = [\phi_1(s), \ldots, \phi_{121}(s)]^T$. To make state-action BFs, the RBFs are repeated for the three different actions. The RBFs are evaluated for the current action and zero for the RBFs associated with the other actions. As such the vector of 121x3 state-action BFs can be represented as follows: $\phi(s,a) = [\mathcal{I}(a=-3)\cdot\phi^T(s), \mathcal{I}(a=0)\cdot\phi^T(s), \mathcal{I}(a=3)\cdot\phi^T(s)]^T$. Here the indicator function $\mathcal{I}$ is 1 when its argument is true, and zero otherwise.

1. *Effect of tuning parameters for the online LSPI algorithm*

First the online LSPI algorithm is studied with no use of human demonstration or other prior information to get a baseline comparison. Figures 2 and 3 show the influence of different constant exploration rates. The exploration rates are kept constant for clear comparison with the Informed Online LSPI results presented later on. In figure 2 the discounted return when evaluating the intermediate policies is presented. In this figure it can be seen that to reach the optimal policy, at least some exploration is needed. Furthermore, it shows that greediness also pays off, since for this problem it results in becoming able to swing up the pendulum. As a result, samples are also gathered in the swung up part of the state space, which is beneficial for the Q-function approximation. In figure 3 the learning score is displayed, which is the summed reward per episode. Each episode is 1.5 s. In this figure it can be seen that a high exploration rate yields low learning scores, which is caused by the high percentage of random actions. Some exploration is beneficial though. As was seen in figure 2, this pays off for finding a good policy.

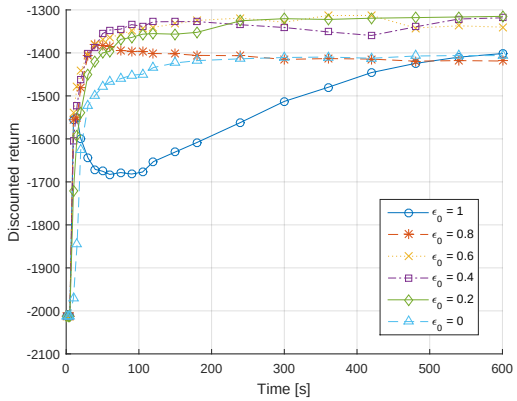

Figure 2: Online LSPI: Discounted return as a function of time for different, constant exploration rates, $K_\theta = 1000$

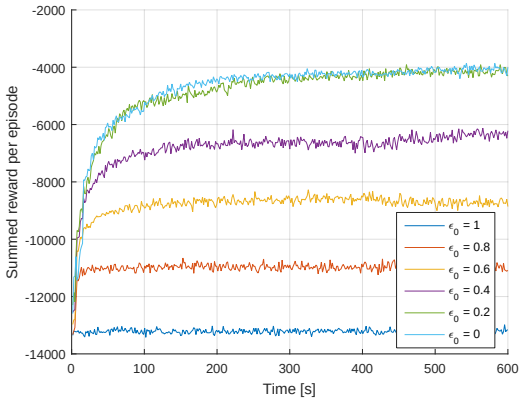

Figure 3: Online LSPI: Learning score as a function of time for different, constant exploration rates, $K_\theta = 1000$

The behavior of the policies for $\epsilon = 1$ follows a different trend. After gathering more samples, the performance first decreases, after which it increases. An explanation for the behavior is as follows. The first samples gathered contribute to finding a better policy. However, a big clutter of samples will emerge around

the hanging down position, since it is hard for a purely random controller to reach the region of the state space where the pendulum is pointing up. As a result, the fitting of the Q-function will be poor because of the minimal variation in data points. Over time the chance of ending up in the upright part of the state space grows. As such the policy gradually goes up. Moreover, full randomness is not a real interesting case to consider, since in practice it would not be used anyway as can be seen in figure 3: it performs poor online.

2. Description of human demonstrations

For the Informed Online LSPI, five different sets of human demonstrations (HDs) are used to see whether human demonstrations could improve the performance, and which type of human demonstration works best. The human demonstration samples are gathered in 21 episodes of 1.5 s, starting at each of the possible initial conditions from the previously defined grid. With a sampling time of 0.005 s, this results in sets of 6300 samples each. The human demonstrator has practiced with controlling the pendulum, and as such can be considered an "expert". Table 1 shows the details on the different sets. In some of the sets some corruption on the action is added. This means that during the human demonstration, a percentage of the actions of the human is replaced by a random action ($a \in \{-3, 0, 3\}$). This corruption can be seen as a forced exploration within the human demonstration set. A human tends to make long streaks of the same action. To swing the pendulum up for example, a human would create a large streak of samples with a clockwise action, after which a large streak of counterclockwise actions is executed, and so on. This results in a low variety of samples, since most adjacent samples contain the same action. As such, this corruption results in an increase in variety in the sample set. Furthermore, two different modes of human demonstration are used: Trying to perform as good as possible, and trying to efficiently explore the state space. The case of having 50% corruption while trying to explore was not included, since the high corruption makes it impossible to efficiently explore the whole state space.

Table 1: Overview of human demonstration sample sets, used for the results in figures 4 and 5.

Sample set number	Corruption	Description
1 (figure 6)	0%	Perform
2	0%	Explore
3	50%	Perform
4	33%	Perform
5 (figure 7)	33%	Explore

Figures 4 and 5 show the results for the different human demonstration sets used in the Informed Online Least Squares Policy Iteration (IOLSPI) algorithm. It can be seen that human demonstration set number 5 (33% percent corrupted, exploratory) performs best. Set number 3 (50% corrupted) also performs well. Both of them converge to a good policy, and show that they perform well from the start, in contrast to the non-initialized online LSPI algorithm that needs some time to learn a good policy. The uncorrupted sets perform significantly worse.

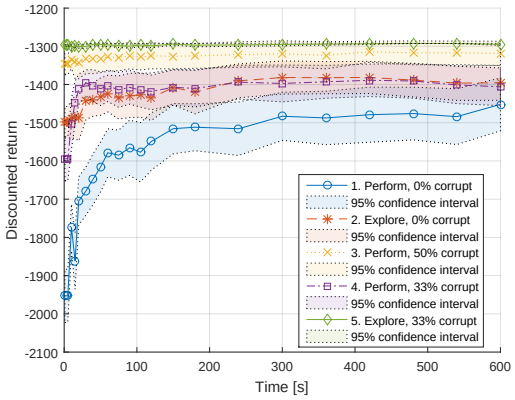

Figure 4: IOLSPI: Discounted return as a function of time for different human demonstration sets, $K_\theta = 1000$, $\epsilon = 0.2$

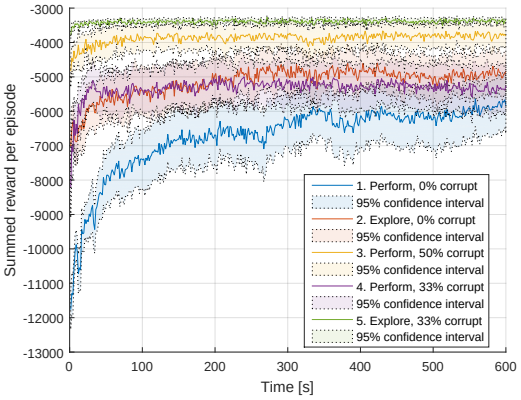

Figure 5: IOLSPI: Summed reward per episode as a function of time for different human demonstration sets, $K_\theta = 1000$, $\epsilon = 0.2$

Figures 6 and 7 show the distribution of samples for human demonstration set 1 and 5. Both the corruption in the actions and the exploration can clearly be distinguished in the latter, whereas the former shows a limited distribution of samples, both in action and in state. It can be concluded that the spread in the sample set is an important factor for the performance of a human demonstration.

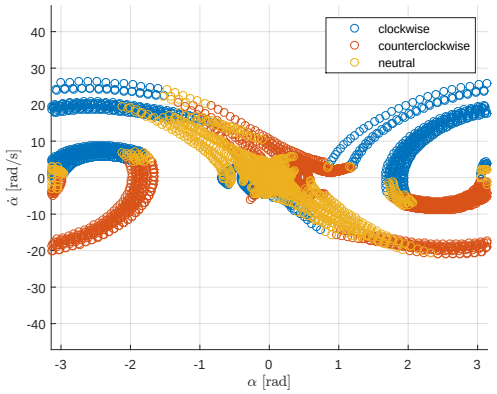

Figure 6: Sample distribution for human demonstration sample set number 1 (0% corruption, perform, 6300 samples)

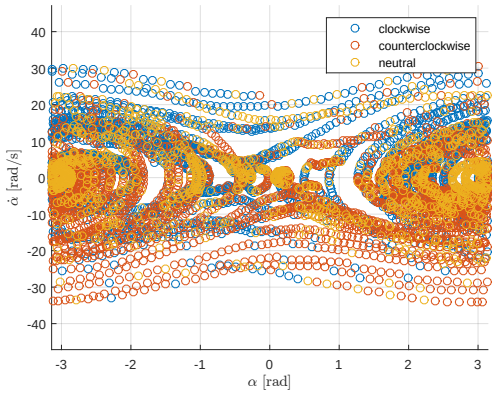

Figure 7: Sample distribution for human demonstration sample set number 5 (33% corruption, explore, 6300 samples)

To study the benefit of using human demonstrations, the best human demonstration set (number 5) is compared to non-initialized online LSPI, and to two other types of sample sets containing the same number of samples. The first is a set of non-sequential random generated samples, which requires a simulator. The sample distribution is depicted in figure 9. The other sample set is obtained using a purely random policy throughout complete episodes, resulting in sequential samples. This approach does not require a simulator, since this can be performed on the physical system. The spread in samples for this random sequential set is limited, as can be seen in figure 8. The difference of this set with the human demonstration sample sets (figures 6 and 7) and the non-sequential random sample set (figure 9) is clear. This is because using purely random actions, the chance is small to reach the swung up region of the state space, while humans use their prior intuition to reach that region.

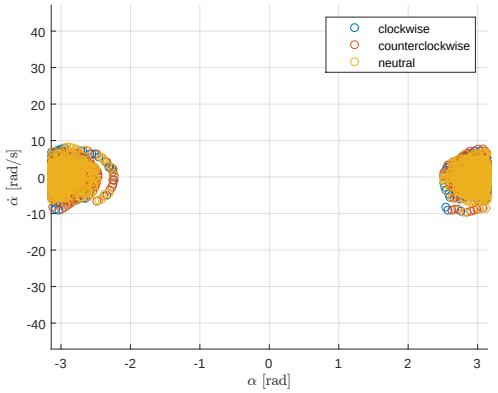

Figure 8: Sample distribution for the sample set generated using a random policy and thus sequential states (6300 samples)

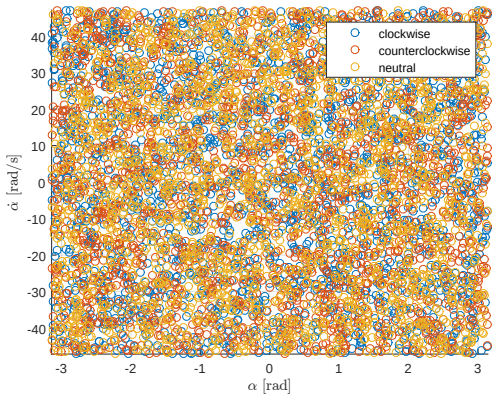

Figure 9: Sample distribution for random generated non-sequential samples (6300 samples)

Each sample set is tested with the IOLSPI algorithm. Their performance is compared to the non-initialized online LSPI algorithm. In the results of figures 10 and 11 it can be seen that the non-sequential random sample set, together with the human demonstration sample set perform best (note that the non-sequential random sample set requires a simulator). A good policy is followed from the start on, and the converged policy is also better than the converged policy found with the sequential random sample set and the non-initialized online LSPI algorithm. This graph hence shows the benefit of the use of the human demonstration for this rotational pendulum problem. Comparing the human demonstration sample set (figure 7) to the set of samples using a purely random sequence of actions in each episode (figure 8) shows why the human scores better than a random policy. The human is able to use its intuitive understanding of the problem to efficiently reach a large portion of the state space, which improves the Q-function approximation and hence the policy.

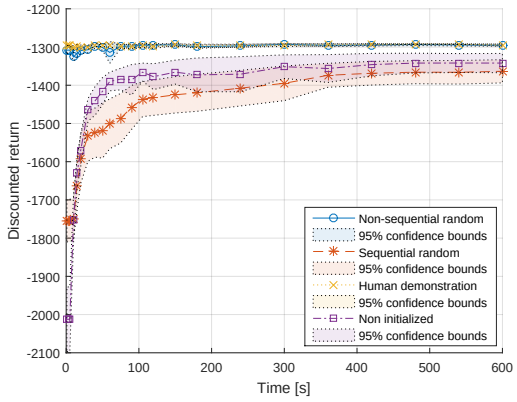

Figure 10: (Informed) Online LSPI: Discounted return as a function of time for different types of sample sets, $K_\theta = 1000$, $\epsilon = 0.2$, using "hanging down ICs"

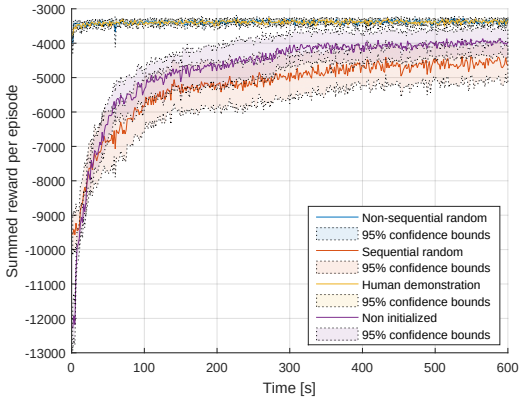

Figure 11: (Informed) Online LSPI: Learning score as a function of time for different types of sample sets, $K_\theta = 1000$, $\epsilon = 0.2$, using "hanging down ICs"

3. *Results for the rotational pendulum problem with distributed initial conditions*

In the previous section it was shown that for the rotational pendulum problem, when an episode always starts at a hanging down position, the human demonstration has a great benefit since the human can use its intuitive understanding of the problem to efficiently explore the state space. In problems where the entire state space is easily visited anyway, the benefit of the human demonstrations might be less pronounced. This hypothesis is tested using the same pendulum setup, but with different initial conditions. Instead of always starting from a hanging down position, the pendulum starts at different positions throughout the whole state space.

The new initial conditions (ICs) used during learning are a random combination of angle and angular velocity, selected from the following subsets: $\alpha_0 = \{-\pi, -\pi/2, 0, \pi/2\}$ rad, and $\dot{\alpha}_0 = \{-10\pi, -3\pi, -\pi, 0, \pi, 3\pi, 10\pi\}$ rad/s.

For this modified, "distributed ICs" problem, regardless of the policy followed, samples will be gathered throughout the entire state space. To study the impact of using human demonstrations on the IOLSPI algorithm for this modified problem, the same analysis is done as was shown in figures 10 and 11. The sequential random sample set, the non-sequential random sample set and human demonstration set number 5 are used in the IOLSPI algorithm and compared to the non-initialized online LSPI algorithm. The results in figures 12 and 13 show the difference in trend of this modified problem with the original "hanging down ICs" problem which was depicted in figures 10 and 11. It can be seen that the benefit of the human demonstration is a lot less. For the random policy and the non-initialized online LSPI the performance reaches the performance of the human demonstration sample set within 100 s. The final policy performs a little less, the difference is small however. Studying the learning score, it can be seen that within 100 s the score is for all different sample sets similar. It can be concluded that, for a problem that does not need intelligent exploration, the benefit of the human demonstration is small.

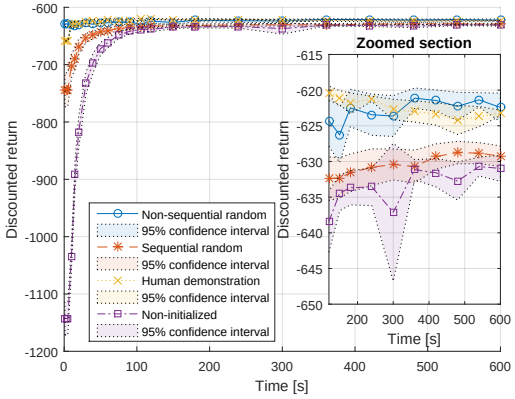

Figure 12: (Informed) Online LSPI: Discounted return as a function of time for different types of sample sets, $K_\theta = 1000$, $\epsilon = 0.2$, "distributed ICs" (Note the zoomed section from 120 s to 600 s, included to emphasize the different converged policies)

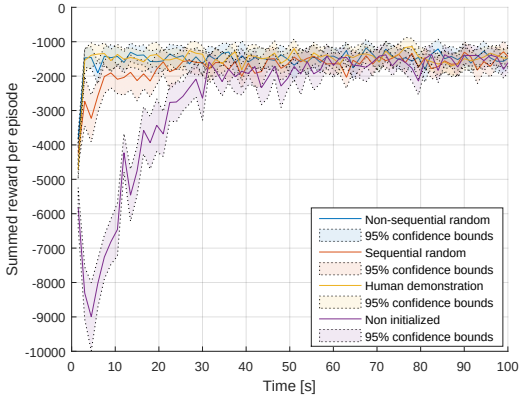

Figure 13: (Informed) Online LSPI: Learning score as a function of time for different types of sample sets, $K_\theta = 1000$, $\epsilon = 0.2$, "distributed ICs" (Only first 100 s are plotted, since after this no significant changes occur)

4. *Algorithm behavior in changing conditions*

One of the benefits of using RL for control is that it is able to cope with changing conditions. With the use of pre-generated samples within the LSPI however, there is a risk that when the conditions change, the algorithm will use too much information of previous conditions. Samples gathered in outdated conditions are no longer valid, and might prevent the algorithm from adequately keep up with the changing conditions. To check this, a sudden change in conditions has been applied to the rotational pendulum problem, both for

online LSPI and for IOLSPI using a human demonstration sample set. To simulate a change in conditions, the gravitational acceleration was suddenly changed after 300 s from 9.81 m/s^2 to 19.81 m/s^2.

Figures 14 and 15 show the performance with this change in conditions. It can be seen that after the change in conditions, the score drops drastically. For both the online LSPI and the IOLSPI using the human demonstration samples, the scores go up after the change again, so it is clear that the agent is adapting to the change in conditions and develops a policy suited for the new situation.

For the IOLSPI it can be seen that after the change some jumps in the summed reward per episode occur. The policy fluctuates severely between 300 s and 340 s, which results in the summed reward per episode going up and down up to 1000 units, which is reflected in figure 15. In this region, every now and then the pendulum reaches the upright regime, at which it already has a fine policy. For the upright regime, the influence of the gravitational acceleration is lower than in the hanging down regime, since it does not includes swinging up. As such, the required change in policy when g changes is low in this regime. After gathering more and more samples over time in the new conditions, the samples gathered in the new conditions become more dominant, reshaping the policy over the whole state space. As a result, the policy is also good at the swing up regime, allowing the pendulum to always swing up. This fluctuating behavior is not seen for the online LSPI, since that policy is not yet good enough to swing up to the upright regime anyway.

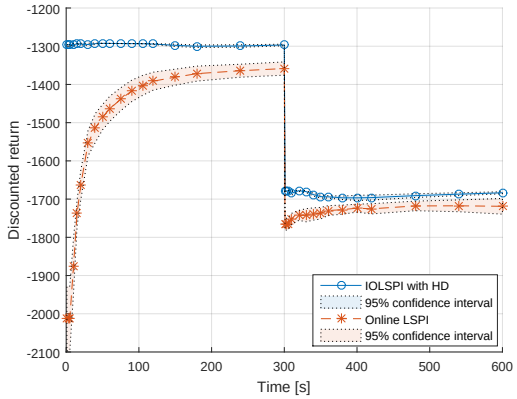

Figure 14: Discounted return as a function of time with a jump in g from 9.81 to 19.81 m/s^2 at 300 s, $K_\theta = 1000$, $\epsilon = 0.05$, using "hanging down ICs"

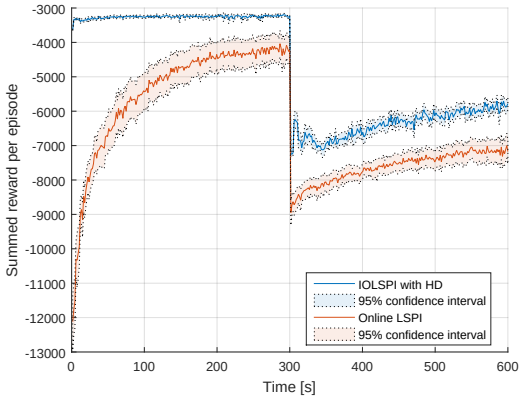

Figure 15: Learning score as a function of time with a jump in g from 9.81 to 19.81 m/s^2 at 300 s, $K_\theta = 1000$, $\epsilon = 0.05$, using "hanging down ICs"

In figures 16 and 17 the time history of the angle α and the reward is shown, corresponding to the results in figures 14 and 15. It can be seen that after the conditions change at 300 s, at first the controller performs bad, and does not succeed in swinging up the pendulum. Looking at the final two episodes however (from 597 s to 600 s), it can be seen that a satisfactory policy was learned. This clearly shows the learning effect, even after changing the conditions.

Comparing figures 16 and 17, the difference between the online LSPI with and without human demonstrations can be seen. First of all it can be seen that before the conditions change, the IOLSPI finds a good policy. The online LSPI on the other hand succeeds in only one of the two displayed episodes, and hence is not yet at an optimal policy. This conclusion is supported by figure 15 which shows that the performance of online LSPI is lower than IOLSPI. After the conditions change, at the last two episodes (from 597 s to 600 s), it is seen that both algorithms succeed in swinging up the pendulum. However, the IOLSPI needs one swing less, and reaches the goal state faster.

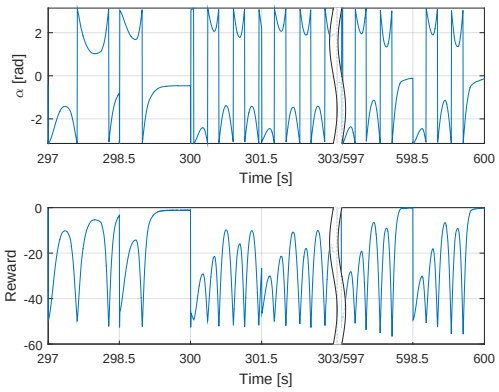

Figure 16: Online LSPI: Time history of the angle and reward, with a jump in g from 9.81 to 19.81 m/s^2 at 300 s, episodes last 1.5 s, $K_\theta = 1000$, $\epsilon = 0.05$, using "hanging down ICs", note the gap in the x-axis

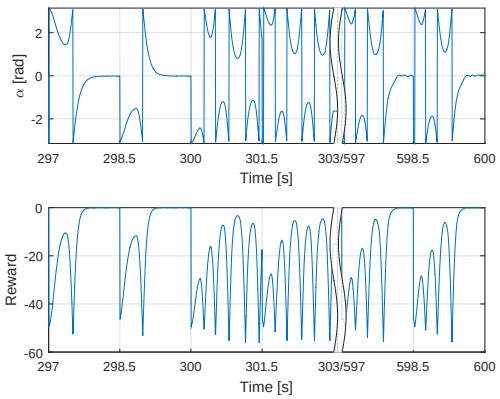

Figure 17: IOLSPI using HD: Time history of the angle and reward, with a jump in g from 9.81 to 19.81 m/s^2 at 300 s, episodes last 1.5 s, $K_\theta = 1000$, $\epsilon = 0.05$, using "hanging down ICs", note the gap in the x-axis

5. *Forget factor implementation*

When the conditions change, the set of samples the controller uses to update the policy still largely consists of samples gathered in the old conditions. As such it is expected that when the conditions change, a forget factor η will help. It makes the controller value recent samples more, and forget older ones. After k time steps, a sample will have been reduced to a fraction of η^k of its original value (see algorithm 2). Depending on the forget factor, after a certain number of time steps, the weight of a sample drops below 1 % of its original value and as such is practically discarded. The following equation calculates which forget factor corresponds to a number of samples $N_{0.01}$ that are more than 1% of their original value.

$$\eta = 0.01^{\frac{1}{N_{0.01}}} \tag{17}$$

Three different forget factors are compared: η =1 (no forgetting), $\eta = 0.99991$ (some forgetting, $N_{0.01} = 50000$ samples) and $\eta = 0.9995$ (much forgetting, $N_{0.01} = 10000$ samples). Figures 18 and 19 show the performance for these different forget factors for online LSPI and IOLSPI using human demonstrations respectively. First of all it can be concluded that a too low forget factor (i.e. much forgetting) yields poor results. This can be explained by looking at the forget factor of 0.9995. With this forget factor only 10000 samples are used, since all samples older than 10000 time steps are practically discarded. It is found that this set of only 10000 samples is not enough to uphold a good value function.

Second, it can be seen that for online LSPI, a forget factor of 0.99991 yields better performance than a forget factor of 1. This can be explained by the fact that the samples at the beginning are forgotten after some time. These samples do not contribute too much, since they are all around the hanging down position. After the change in conditions, the forget factor of 0.99991 also clearly performs better. It forgets the old conditions gradually, and by that allows for a better policy in the new condition.

For the IOLSPI using HDs in figure 19, a little different behavior is seen. With the forget factor of 0.99991, in the first 300 s, the HD samples are being forgotten more and more. This means that the spread in the used samples decreases, and hence the performance also decreases. After the change in conditions, when the HD samples are not of too much interest anymore, since they were obtained under different conditions, the forget factor of 0.99991 shows better results than a forget factor of 1.

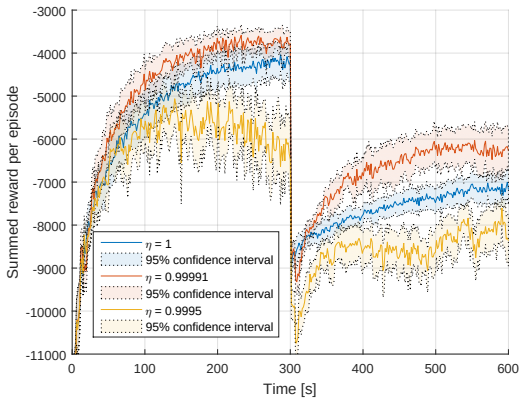

Figure 18: Online LSPI: Summed reward per episode as a function of time for different forget factors, $K_\theta = 1000$, $\epsilon = 0.05$, using "hanging down ICs"

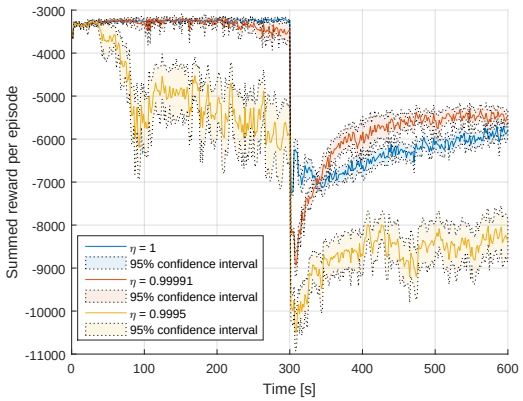

Figure 19: IOLSPI with HD: Summed reward per episode as a function of time for different forget factors, $K_\theta = 1000$, $\epsilon = 0.05$, using "hanging down ICs"

B. Quadrotor slung load system

Having studied the rotational pendulum, the safety aspect in the exploration has not become apparent yet. As such, a second system is studied in which safety plays a more pronounced role. This system is a quadrotor carrying a load suspended from a cable, where the goal is to drop the load in a basket. When the controller is exploring, the quadrotor could either hit a wall or hit its own rotors with the suspension cable. Both events can be considered a crash, with damage to the quadrotor as a consequence. Furthermore, the load could be dropped outside the basket, damaging the (possibly fragile) load. Another interesting aspect of this quadrotor slung load problem is that it has more dimensions than the rotational pendulum. As such the effect of increasing dimension on the performance of the algorithm can be studied.

In this paper, a 2D simplification of the quadrotor system is studied with two degrees of freedom: the lateral position of the quadrotor x, and the swing angle of the load θ. This 2D simplification has more dimensions than the rotational pendulum problem, but still is straightforward to implement. The goal is to drop the slung load at the target as quickly as possible. A snapshot of the simulation is presented in figure 20. Since the quadrotor cannot move up or down, it has to swing the load up to get it in the target basket.

The position of the quadrotor is defined in an inertial frame, with x pointing right and z pointing downwards. The location of the pendulum is described in a radial reference frame, centered in the c.o.m. of the quadrotor. When the pendulum is hanging down, $\theta = 0$ rad. Swinging to the right corresponds to positive θ. The state is defined as $s = [x, \dot{x}, \theta, \dot{\theta}]^T$. The action represents a force exerted on the quadrotor which induces an acceleration $a \in \{-3, 0, 3\}$ m/s^2. A fourth action is present, which is releasing the load.

Considering the dynamics of the system, first the motion of the quadrotor is considered. The influence of the slung load on the quadrotor is neglected. The control input for the system is the acceleration of the quadrotor. It is assumed that no external forces act on the quadrotor, so it can be assumed that $\ddot{x} = a$. The angular acceleration of the slung load is calculated using Eq. (18).[3]

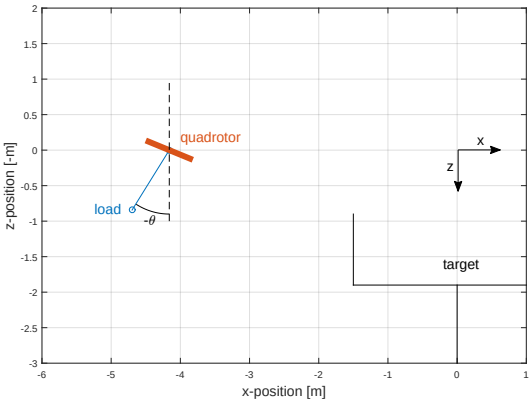

Figure 20: Snapshot of the 2D quadrotor simulation

$$\ddot{\theta} = \begin{bmatrix} -\cos\theta & \frac{\sin\theta}{L} \end{bmatrix} \cdot \begin{bmatrix} \ddot{x} \\ -g \end{bmatrix} \tag{18}$$

In this equation L is the length of the suspension cable. The state is propagated every time step using a simple Euler integration scheme with sampling time T_s:

$$s \leftarrow s + \dot{s} \cdot T_s \tag{19}$$

A number of assumptions are made concerning the slung load. The cable is assumed to be inelastic and massless. Furthermore, it is assumed that the cable is suspended from the c.o.m. of the quadrotor. The system is assumed to be undamped, so aerodynamic forces are neglected, as well as friction of the cable in the suspension point. Finally, it should be noted that the cable is actually modeled as a rod, since the simulation does not allow the cable to become slack.
There are three conditions under which an episode ends:

- The maximum simulation time T_{end} is reached
- A crash occurs (either the quadrotor hits the wall or the cable hits the rotor)
- The load is dropped (either a miss or a dunk)

To score a goal, the following conditions should be fulfilled at the same time:

- The x position of the load is within $x_{goal} \pm x_{g_{margin}}$ where x_{goal} is the basket center, and $x_{g_{margin}}$ defines its width.
- The z position of the load is higher than z_{goal} but lower than $z_{goal} + z_{margin}$
- The load is moving downwards ($\dot{z}_l \geq 0$)

In short, the load must fall in the basket from the top. In table 2 the constants used in this simulation are presented.

Table 2: Dimensions and constants for the quadrotor simulation

Variable	Value	Unit	Description
g	9.81	m/s^2	gravitational acceleration
L	1	m	Cable length
T_s	0.1	s	Sampling time
T_{end}	5	s	Simulation time
z_{start}	0	m	Starting position
x_g	0	m	Goal position
z_g	0.9	m	Goal position
$x_{g_{margin}}$	1.5	m	Goal width

A reward is given when the load is dropped in the goal and a penalty is given when the goal is missed. A slung angle of higher than 90 deg is not desirable, because this would cause the cable to tangle up with the propellers. Therefore, a too high slung angle is penalized in the reward structure. The sides of the domain are represented by walls, so when the quadrotor leaves the domain (it hits the wall) a penalty is given. Furthermore, a penalty is given depending on the distance to the goal, in order to stimulate fast progression to the goal. The reward structure is formalized in Eq. (20).

$$r = \begin{cases} 100 \text{ if dropped in basket} \\ -10 \text{ if dropped next to basket} \\ -50 \text{ if } \theta > 90 \text{ deg} \\ -150 \text{ if quadrotor hits a wall} \\ -0.5 \cdot ||x - x_{goal}|| \end{cases} \tag{20}$$

During learning, the initial conditions are picked at random from the following subsets: $x \in [-6, 0]$ m, $\dot{x} = 0$ m/s, $\theta \in [\frac{-\pi}{3}, \frac{\pi}{3}]$ rad and $\dot{\theta} = 0$ rad/s. This rather distributed spread of initial conditions is chosen to speed up the learning. This makes sure that the spread of the states visited will naturally be a lot bigger than when the initial conditions are always around the same point in the state space.

Similar to the rotational pendulum problem, results are repeated for fifty different random seeds, to account for the randomness. To approximate the Q-function, an equidistant grid of radial basis functions is used. The issue encountered in a system with a higher dimensional state is that the number of basis functions using this an equidistant grid grows exponentially with the number of state dimensions. To make the problem solvable from a computational point of view, the number of RBF centers is limited to the x dimension and the θ dimension, since the velocity and angular rate are believed to be less critical for this problem. For this problem this means that an equidistant grid of 9x1x9x1 RBF centers is used. Since there are four possible actions, a total of 324 RBFs is used.

1. *Results for the quadrotor slung load problem*

Figure 21 shows the learning effect for the quadrotor slung load problem for IOLSPI with a human demonstration and for online LSPI. It can be seen that the performance of IOLSPI is for the first 15000 episodes better than online LSPI. From a safety point of view, figure 22 is especially interesting. It shows the percentage of crashes, misses, and the number of dunks (dropping the load in the goal), as a function of the number of elapsed episodes. It can be seen that the longer the non-initialized LSPI algorithm learns, the lower the amount of misses and the higher the amount of dunks. This is a result of the policy which is improving, and the exploration rate which decays exponentially. The amount of crashes drops in the beginning, and from 10000 episodes onward shows a gradually increasing behavior. This increase in crashes can be attributed to the decreasing exploration: While the exploration is still high, the chance of accidentally dropping the load in the beginning of an episode is large. This dropping of the load ends the episode before a crash could have occurred. After some time, when the exploration rate is lower, the load will remain attached longer, increasing the chance of hitting a wall. It can be concluded that the policy found by the non-initialized LSPI succeeds in reducing the number of misses, but fails to keep the number of crashes low. When human demonstrations are used, it is seen that the performance in terms of dunks and misses is already better from the beginning. The number of crashes is also lower when using human demonstrations. Combining the reduction in misses and crashes over the whole time span, it can be concluded that using IOLSPI with human demonstrations is safer than the non-initialized online LSPI.

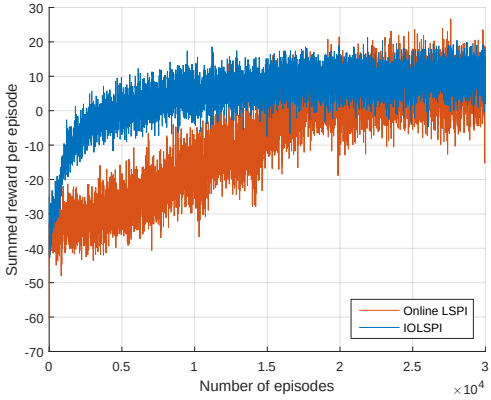

Figure 21: Summed reward per episode as a function of number of episodes for the quadrotor slung load problem

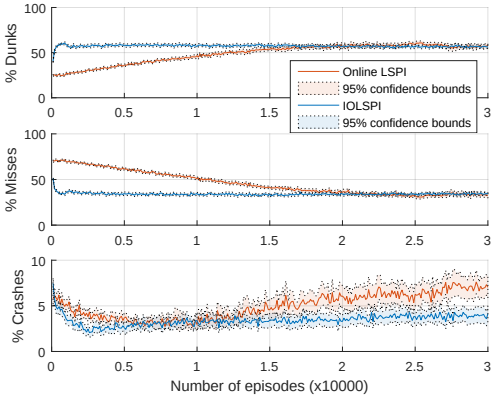

Figure 22: Safety aspects: Percentage of crashes, misses and dunks as a function of number of episodes

For this higher dimensional problem, the execution time becomes a big issue. The time to complete 1000 episodes is compared for different numbers of radial basis functions. Table 3 shows the results. It can be seen that for higher numbers of RBF centers, the computational effort goes up quickly. This can be explained by the fact that within the algorithm, some vector multiplications are done with the vector

of basis functions and also with the Γ and Λ matrices, which are square matrices with dimension n (n is the number of basis functions). In the policy improvement step, a linear system of order n is solved which has $O(n^3)$ complexity. This calculation dominates the execution time and explains the strong increase in execution time with increasing number of RBFs.

Table 3: Execution time for different number of RBFs

Total RBFs	x-dimension	$\dot{x}$-dimension	θ-dimension	$\dot{\theta}$-dimension	Computation time [s]
36	9	1	1	1	10.2
324	9	1	9	1	18.5
2916	9	9	9	1	349.5
26244	9	9	9	9	Size of matrices causes memory problems *

* Computations were executed in Matlab R2015a, which reaches its memory limits.

V. Conclusions and discussion

This paper shows that for some systems, human demonstrations can improve the performance of reinforcement learning in terms of learning time and safety. To show this, a combination of two state of the art reinforcement learning (RL) algorithms is used. Offline Least Squares Policy Iteration (LSPI) and online LSPI are combined to an algorithm named Informed Online Least Squares Policy Iteration (IOLSPI). This algorithm is able to use human demonstration sample sets to improve its performance. The IOLSPI algorithm uses the information present in a batch of samples, which can be obtained by a human demonstration, to kick start the online LSPI algorithm, which continues learning online.

The example studied in this paper is a rotational pendulum problem, where a mass has to be swung up from a hanging down position. It is found that using human demonstrations, a good policy is found straight away, whereas non-initialized online LSPI needs 60000 time steps (300 s) to converge. In addition the final policy found using human demonstrations is 10% better in terms of cumulative reward than the non-initialized online LSPI.

It is found that the benefit of the use of human demonstrations is largest when the human uses its understanding of the problem to efficiently explore the state space, rather than only giving perfect demonstrations of the task at hand. This exploration results in a large variation in samples throughout the state space, which allows for better Q-function approximation. In addition, it is found that human demonstrations work better in IOLSPI when every third action of the human demonstrator is corrupted (replaced by a random action) in order to increase the variety in the demonstration sample set even more.

The benefit of using human demonstration in combination with IOLSPI is larger for problems in which some understanding of the problem is needed to reach a larger portion of the state space. This is the case in the aforementioned swing up pendulum problem. When the rotational pendulum problem is changed in such a way that each episode of the problem is started at a random state, no understanding of the problem is needed to explore the whole state space, in contrast to the original problem, which always starts hanging down. For this modified problem, it is found that the IOLSPI using human demonstrations finds a good policy straight away as well. However, in this modified problem the non-initialized online LSPI algorithm finds a good policy already after 20000 time steps (100s), significantly faster than in the original problem. Furthermore, the found policy scores only 1% worse than the policy found by the IOLSPI algorithm using human demonstrations.

In addition, it is shown that the IOLSPI algorithm fulfills the demands for an autonomous controller: It is fully model-free and it can adapt to changing conditions. To improve the adaptability of the algorithm, a forget factor is implemented, to value outdated samples less than more recently gathered ones, to focus on the current conditions. It is found that a mild forgetting behavior results in better performance in changing conditions. With a mild forget factor, the IOLPSI algorithm needs over 40000 time steps less to obtain a similar performance after a change in conditions, compared to the algorithm without the use of a forget factor.

With a second studied system the benefit in terms of safety of the IOLSPI algorithm is shown. In this quadrotor slung load drop off task it is shown that the number of crashes while learning is reduced by 10% to 50% using the IOLSPI algorithm, compared to a non-initialized reinforcement learning algorithm.

With the second case study, the limitation of the algorithm becomes apparent. A uniformly distributed grid of radial basis functions is used to approximate the Q-function. With a growing number of state parameters, the number of basis functions grows exponentially. For higher dimensional problems, the computational effort becomes too high for present day computers.

In future research, improvements could be made by combining the IOLSPI algorithm with methods that choose the radial basis functions in an intelligent way, or use other basis functions. When the number of basis functions is limited, the computational effort stays low. With this addition in place, the algorithm can be applied to multiple actuator systems, with continuous actions. With these extensions, the algorithm can be applied to a 3D quadrotor simulation, and eventually on a real quadrotor. This will be a next step towards model-free and adaptive control supporting the autonomy of these complex vehicles.

References

[1]Kumar, V. and Michael, N., "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, Vol. 31, No. 11, 2012, pp. 1279–1291.

[2]de Croon, G. C. H. E., Groen, M. A., De Wagter, C., Remes, B., Ruijsink, R., and van Oudheusden, B. W., "Design, aerodynamics and autonomy of the DelFly," *Bioinspiration & Biomimetics*, Vol. 7, No. 2, 2012.

[3]Faust, A., Palunko, I., Cruz, P., Fierro, R., and Tapia, L., "Learning swing-free trajectories for UAVs with a suspended load," *IEEE International Conference on Robotics and Automation, 2013*, 2013, pp. 4902–4909.

[4]Barto, A. G. and Sutton, R. S., *Reinforcement learning: An introduction*, The MIT Press, Cambridge, Massachusetts, 1998.

[5]Mannucci, T., van Kampen, E., de Visser, C., and Chu, Q., "SHERPA: a safe exploration algorithm for Reinforcement Learning controllers," *AIAA SciTech Guidance, Navigation, and Control Conference*, 2015, pp. 1–15.

[6]Busoniu, L., De Schutter, B., Babuska, R., and Ernst, D., "Exploiting policy knowledge in online least-squares policy iteration: An empirical study," *Automation, Computers, Applied Mathematics*, Vol. 4, 2010, pp. 521–529.

[7]Vaandrager, M., Babuska, R., Busoniu, L., and Lopes, G., "Imitation learning with non-parametric regression," *IEEE International Conference on on Automation, Quality and Testing, Robotics, 2012*, 2012, pp. 91–96.

[8]Abbeel, P., Coates, A., and Ng, A. Y., "Autonomous Helicopter Aerobatics through Apprenticeship Learning," *The International Journal of Robotics Research*, Vol. 29, No. 13, 2010, pp. 1608–1639.

[9]Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., *Reinforcement learning and dynamic programming using function approximators*, CRC Press, Boca Raton, Florida, 2010.

[10]Lagoudakis, M. G., "Least-squares policy iteration," *Journal of Machine Learning Research*, Vol. 4, 2003, pp. 1107–1149.

[11]Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R., "Online least-squares policy iteration for reinforcement learning control," *American Control Conference , 2010*, 2010, pp. 486–491.

[12]Palunko, I., Faust, A., Cruz, P., Tapia, L., and Fierro, R., "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," *IEEE International Conference on Robotics and Automation, 2013*, 2013, pp. 4896–4901.

[13]Palunko, I., Donner, P., Buss, M., and Hirche, S., "Cooperative suspended object manipulation using reinforcement learning and energy-based control," *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 885–861.

[14]Faust, A., Palunko, I., Cruz, P., Fierro, R., and Tapia, L., "Automated aerial suspended cargo delivery through reinforcement learning," *Artificial Intelligence*, Vol. 1, 2014, pp. 1–18.

[15]Bellman, R. E., *Dynamic Programming*, Princeton University Press, New Jersey, 1957.

[16]Bertsekas, D. P., *Dynamic Programming and optimal control 3rd ed.*, Athena Scientific, Bellmont, Massachusetts, 2011.

[17]Li, L., , Littman, M. L., and Mansley, C. R., "Online exploration in least-squares policy iteration," *International Conference on Autonomous Agents and Multiagent Systems, 2009*, Vol. 2, 2009, pp. 733–739.