University of Bristol
Department of Engineering Design
Interim Report

Systems for Independent Living

Reinforcement Learning for Adaptive Agents

Jerome Wynne

December 12, 2017

Executive Summary

This research project will compare two state-of-the-art reinforcement learning control systems. It will do so in anticipation of developing an assistive robot in Year 5. Two stages of experimentation will take place: In the first, the controllers will be scored on their performance in locomotion and balancing tasks, both to take place in simulation. In the second, the higher-scoring controller's robustness to actuation errors will be evaluated by exposing it to actuator faults. A reinforcement learning (RL) agent does not rely on a priori knowledge of its operating environment, and may therefore be used to create task-adaptive controllers. This project will evaluate the feasibility of using RL in Year 5, and will contribute to the RL literature by directly comparing two of the most successful RL methods for robotics to date.

Contents

List of Figures

List of Tables

1 Aims & Objectives

1.1 Group Objective

This group seeks to design an automated system that can provide physical assistance to a person in their own home, without oversight from a carer. The following section motivates this problem statement, and explains how our team's individual research projects will enable an assistive automated system to be realized in Year 5.

In the UK, 4 million people cite themselves as being substantially limited in their daily activities by disability, illness, or age [1]. To support them and others in need, 5.4 million friends, neighbours, and family members are informal carers [2]. They help the people they care for to perform the essential tasks of daily living such as getting out of bed, walking around the home, using the bathroom, and doing the laundry.

A person in need of care may have lost their independence due to either physical or mental health problems, or both. Approximately 90% of the cared for have some form of physical limitation, whereas just under half suffer from a mental illness [3]. This is reflected in the type of care provided: The NHS identifies 'practical help' around the home as the dominant form of informal care [3]. In view of this, our group has chosen to develop a system that can provide physical help within the home, excluding psychological and residential care from the scope of our work.

Robots are well-posed to solve the problem of physical assistance. A person in need of help usually requires either an energy or control input to accomplish their task: A robot can deliver either or both quantities. Robots can also be networked, providing a route for everyday or emergency communications. People in care are more vulnerable to falls and other health events, making this property particularly pertinent [4]. Moreover, working memory deteriorates with age, making complex user interfaces more difficult for the elderly to operate. Autonomous agents can facilitate complex tasks in exchange for little or no user input, shifting cognitive demands away from the user in the process.

Given the suitability of an automated solution, our group has agreed that our Individual Research Projects should align with the main subfields of robotics and automated systems (RAS). An incidence matrix between the team's individual research themes and the components of an automated system is shown in Table 1. The Table's headers map to the following research titles:

1. Hardware & power	Physical assistance through the use of actuation and form.
2. Mapping and navigation	Adaptable autonomous navigation for in-home environments.
3. Predictive sensing	Sensing and detection of fall events.
4. Human-robot interaction	Two-way communication between a robot and a user.
5. Safe control	Control schemes for safe physical robot-human contact.
6. Adaptive planning	Reinforcement learning for adaptive agents.

The interactions between these topics are developed in more detail over the course of this report, particularly in Sections 1.2 and 2.

As can be seen in the table, the research themes do not explicitly address the communications component of an automated system. There are two reasons for this. First, we felt that wireless technologies are sufficiently mature that, if needed in Year 5, they would not significantly constrain the final system's design. Second, we agreed that on-board communications and data management is best understood from the perspective of the components that use it: Computers, sensors, and actuators. Correspondingly, we have agreed to plan, execute, and critique our individual research projects with practical data transfer limitations in mind. As will be seen, this constraint is particularly relevant to this research project.

In summary, our group aims to deliver an automated system that can help a person to perform physical tasks in their own home. We have constrained the problem by specifying the domain in which the solution will be found, RAS, and by stipulating the class of task that will be solved - physical assistance of a single person. Our research projects will collectively profile what automated systems can currently be used to accomplish, allowing us to select an appropriate task to focus on together in Year 5.

	Research themes					
RAS requirements	Hardware & power	Human-robot interaction	Predictive sensing	Mapping and navigation	Safe control	Adaptive planning
Actuators	✓	-	-	-	-	-
User interface	-	✓	-	-	-	-
Sensing	-	-	✓	✓	✓	-
Communications	-	-	-	-	-	-
Control	-	-	-	✓	✓	✓
Power	✓	-	-	-	-	-
Structure	✓	-	-	-	-	-

Table 1: Correspondence between the essential components of a robotic system and our Individual Research Projects.

1.2 Individual Project Objectives

This section outlines the aims of this research, and what objectives will be fulfilled en route to meeting those aims.

Social care is often continuous - carers either cohabit with the people they support, or live just a few doors away [3]. This means that carers tend to be available to assist with a variety of tasks. Additionally, a person's care needs will change with respect to time: People age, and carers are able to adjust the help they provide accordingly. Consequently, if the system developed in Year 5 is to significantly reduce a person's dependence on their carer, it will need to be able to

- provide assistance across multiple distinct tasks,
- and adapt to changes in dynamics within a single task.

These two features are highlighted because they form the motivation for this research project. Bespoke control systems are unlikely to meaningfully reduce a care recipient's dependence on their carer. This project will develop controllers that can learn new tasks, and can adapt to changes in a given task over time. To do so, it will focus on techniques from the field of reinforcement learning, a discipline with roots in computer science, machine learning, and optimal control.

Reinforcement learning (RL) enables a system, also known as an agent, to learn a control protocol by observing and interacting with its environment [5]. RL controllers are usually used when an environment's dynamics are either unknown or change over time. In contrast to traditional control, in which the controller's response is tuned directly, an RL controller's behaviour is directed using a reward function. The agent selects its actions to maximize the reward received each time it undergoes a state transition.

An agent's action preferences are represented by a policy, a function mapping from states to actions. Robotics problems demand particular types of policies because a robot can be in many possible states, and can have access to many possible actions. As will be discussed in the literature review, if the agent is to perform well it is paramount that an appropriate policy parameterization is used.

This project will compare two state-of-the-art policy representations for adaptive robotic control. It will also evaluate how robust the better-performing representation is to actuation errors. The models developed over the course of this project will contribute the following properties and functionalities to the Year 5 project:

- Adaptivity to shifts in task dynamics, both between users, and for one user over time. RL controllers are capable of learning while in operation, a feature that is highly desirable in the context of a person's unique home and care needs.
- The ability to perform tasks for which the problem dynamics are unknown. This will permit complex behaviours to be elicited from the prospective Year 5 system, such as locomotion, manipulation, and high-level planning.
- The capacity to learn tasks under the guidance of a non-specialist user. Policy representations and reward functions can, in principle, be learnt by demonstration [6] [7] [8]. A generic manipulator that could be taught to interact with everyday items would gain access to a large subset of the tasks that care recipients need help with. A universal solution is preferable to the clutter and expense associated with multiple bespoke products.

In addition to the value offered to the group work, this research will also be useful to the reinforcement learning community. Its contributions will be threefold:

1. Experiment 1 will compare two different policy representations[1] on a pair of simple locomotion and balancing tasks. The representations chosen - dynamical motor primitives and deep neural networks - are both currently used for robotic control, but have not previously been compared directly.
2. The literature review suggested that a sensitivity analysis of RL controllers for robotics would be useful to roboticists looking to use RL in physical systems. Experiment 2 will evaluate the effects of actuation errors on the performance of the better-performing model of Experiment 1. In doing so, it will identify whether further work is needed to engineer RL models that are robust to imperfect actuators.
3. Practical, accessible advice regarding how to implement an RL controller for robotics is limited, but RL is cited as the likely candidate for next-generation robotic control [9] [6]. Consequently, this project will deliver a feasibility assessment of the models tested. This assessment will enable practising engineers to experiment with RL by explaining the technology's underlying assumptions and practical limitations.

The objectives described above are matched with deliverables in Table 2, where they are also split into sub-aims. The Work Plan in Section 3 specifies timescales and completion dates for each item. The next Section contrasts reinforcement learning against other methods for robotic control, frames the reinforcement learning problem, and justifies the tasks and models chosen for the experimental phases of this work.

[1]Note that 'parameterization' and 'representation' are used interchangeably throughout this document.

Table 2: Specification of research objectives and their associated tangible outcomes.

Research objective	Deliverable outcomes
(1) Experiment 1: Compare two RL models for robotics across more than one distinct task. Two of the most competitive RL models for robotics will be compared on a pair of simulated tasks. The mean and variance of their performance on the tasks will be used to determine which approach is superior, however each model's memory and time complexity will also be measured and reported. The outcomes from this objective will specify and justify the RL controller that would be more appropriate for use in Year 5.	(1.1) A data set containing the model scores on each of the tasks, aggregated across multiple runs. (1.2) The software code necessary to repeat the simulation that produced the dataset of (1.1), figures ued in the Final Report. (1.3) A critique of each model's practical properties and theoretical assumptions.
(2) Experiment 2: Assess how actuator errors affect the better-performing model. This experiment will profile the controller's sensitivity to actuator faults in a locomotion problem. The resulting simulation will serve as a proof-of-concept for a predictive design strategy in Year 5. The data gathered will also be used to quantify how quickly the system can recover from actuation faults, if it can.	(2.1) A data set containing the agent's score as a function of noise on the locomotion task (See the Summary of Related Work). (2.2) Accompanying code that enables another researcher to replicate the experiment's results. (2.3) A critique of the model's performance in terms of how it adapts to the simulated actuator faults.
(3) Evaluate the Model's Feasibility. To assess the practical and technical feasibility of the both models from Experiment 1 in the context of learning in an assistive robotic system, and to provide instruction as to how to implement these.	(3.1) A practical assessment that specifies the sensing, actuating, and computing requirements of the models tested and the learning algorithm used. (3.2) A technical assessment that evaluates the conceptual limitations of the models and learning algorithms tested, and suggest how these might be mitigated in practice. (3.3) A synthesis of (3.1) and (3.2) that reflects on what the model could be used to accomplish in Year 5.

2 Summary of Related Work

This section consists of four parts. The first contextualizes RL against the other dominant forms of robotic control, and explains why it may be more appropriate for certain domestic robots. The second then formulates the RL problem, and outlines how, in principle, it is solved. With the abstract solution in mind, the third part describes what the unique properties of robotics problems are, and critiques current algorithms and models for RL in robotics. The final part summarizes how the robustness of a model to actuation errors can be assessed.

2.1 Techniques for Robotic Control

Robotic control is currently achieved in one of three ways: Either by direct programming, imitation learning, or by reinforcement learning [9] [6]. While this research is focused on a developing a reinforcement learning system, the other approaches serve to highlight RL's benefits and limitations.

Direct programming is used in roboticized assembly lines, and partially enabled the era of automated manufacturing that has been ongoing since the 1960s [10]. The method consists of directing a robot to move through a specific series of low-level (i.e. positional) states. This approach was feasible in manufacturing because a factory floor could be closely controlled, and the tasks that took place on it were usually repetitive. Imitation learning emerged in the 1980s as a method by which a robot could be programmed by demonstration [11]. These demonstrations involved moving the robot through its path of operation manually, by teleoperation, or by tracking a demonstrator via machine vision [9]. Imitative techniques also introduced the idea that a robot could maintain a symbolic view of its world: Rather than treating a sequence of positions as the target, constraints relative to other objects (e.g. 'above', 'close to', 'concentric') began to be used. Fundamentally, programming by demonstration enabled a degree of dynamic planning that was not available to robots that were programmed directly.

Reinforcement learning as it is currently recognised took shape in the 1990s. It is only recently that RL agents have begun to be seen as viable control solutions in commercial systems [12] [13]. RL's core benefit relative to imitation learning and direct programming is that it enables a robot learn by trial and error. An RL agent is capable of learning dynamic behaviours by optimizing its long-term reward, as opposed to crudely combining demonstrations. This means that RL agents can perform tasks that can't be readily demonstrated, a feat that has been demonstrated repeatedly in the literature [14] [**?**] [13]. Moreover, an RL agent can adapt its behaviours in response to changes in task dynamics: It can respond to wear on its components, and changes in the task environment. These functionalities are attractive, particularly in a robot that aids with care tasks in the home. RL has been criticised for being more computationally expensive and conceptually complex than both imitation learning and direct programming [6] [15]. This being said, RL's hardware requirements are not usually prohibitive - demanding calculations can be offloaded to cheap remote servers, and an appropriate on-board computer is unlikely to cost more than a few hundred pounds. The complexity of RL solutions can be mitigated using best practices from software design, specifically careful planning and extensive testing.

2.2 The Reinforcement Learning Problem and its Solution

To explain why RL controllers enable task-adaptive robotics, it is first necessary to introduce the RL control problem. Informally, this can be summarized as *How can an agent learn to act when its environment is uncertain?* Formally, it is framed as a discrete-time Markov decision process [5] [16]. The agent's state and actions are represented using vectors $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ respectively, where $\mathcal{A}(s)$ highlights that the actions available to an agent may be state-dependent. The agent selects its actions at each time-step according to a *policy*, a function $\pi : \mathcal{S} \to \mathcal{A}$ that maps states to actions. The environment is described in terms of its transition dynamics[2], a set of probability distributions over possible state-action tuples $s_{t+1} \sim T(s_t, a_t) \in \mathcal{T}$. Each distribution $T \in \mathcal{T}$ specifies how likely the state $s_{t+1} = s \in \mathcal{S}$ is, given the agent's current state and chosen action.

In theory, the ensemble of objects $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \pi)$ is sufficient to describe the behaviour of an agent on a task: Given an initial state s_0, the agent can take an action $a_0 = \pi(s_0)$, transition to another state $s_1 \sim T(s_0, a_0)$, then select its next action. What is missing is a measure of the agent's performance, its reward function, $r(s_t, a_t, s_{t+1})$. In addition to indicating how well the agent performs, this function provides a feedback mechanism: The agent can adjust its policy to favour actions that generate large rewards.

To perform well on its task, the agent must move to states that yield large amounts of reward in the long term. Some actions may yield a high reward immediately, but may also make it difficult to earn reward in the future. It is often the case that RL agents attempt to estimate the *value* of each state, a function of the cumulative reward that would be gained by following the policy π from it. Typically, this function is taken to be a time-discounted average [5]. Symbolically, this can be expressed as

$$V^\pi(s) = E\Big[\sum_{k=0}^{\infty} \gamma^k r(s_t, a_t, s_{t+1}) | s_0 = s\Big] \tag{1}$$

Where π is the agent's policy, E is the expectation operator, $\gamma \in [0, 1]$ is a discounting parameter that weights immediate rewards against later ones, and s_0 is the agent's initial state. The expectation is necessary since the agent's policy and the environment's transition dynamics may be stochastic.

A sequential decision-making task and its prospective solution can then be represented by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \pi, V^\pi)$. In many situations, a designer will know $\mathcal{S}$ and $\mathcal{A}$, and may have some prior knowledge of $\mathcal{T}$. They must then choose r and π according to the task they want the agent to solve. RL models usually assume that V^π has the form shown in Equation (1). Depending on the learning method used, the value of $V^\pi(s)$ in may need to be estimated explicitly for each $s \in \mathcal{S}$. These sub-problems - choosing a reward function, selecting a policy, and estimating the value of each state - are the focus of contemporary research in reinforcement learning [5] [16].

Many techniques for improving the agent's policy and estimating the value function have been proposed. The approach taken usually involves initializing a policy π_0, estimating its value function $V^{\pi_0}(s)$ for all states, then updating the policy to select more valuable actions. This corresponds to the update rule

$$\pi_1(s) \leftarrow \underset{a \in \mathcal{A}(s)}{\arg\max}\, E\Big[r(s, a, s') + \gamma V^{\pi_0}(s')\Big] \tag{2}$$

Where π_1 is the agent's policy after the update, and s' is the states that can follow s. In words, the agent's policy is altered so that its selects the most valuable action in each state. This update is guaranteed to result in an agent that can generate more reward. Better still, the update can be repeated multiple times. A new value function, V^{π_1}, can be estimated, and a better policy π_2 can be formed by applying Equation 2 again. When the value function ceases to change between policy updates, a globally optimal policy, π^*, has been achieved. This policy maximizes the value of

[2]The environment is assumed to be Markov - each state transition is conditionally independent of all previous states apart from the current one. This assumption makes the problem computationally tractable, and can be relaxed by introducing 'memory' of past states into the state vector.

all states, resulting in a maximal value function $V^*(s)$, and an optimally performing agent - provided that that the reward function accurately reflects the intended task [5].

The algorithm described above is known as *generalized policy iteration*. In practice, it is implemented using an algorithm. Policy iteration algorithms have been developed that are particularly effective at fitting policies that are suitable for robotics problems. The next subsection explains how these algorithms have been specialized for robotics problems.

2.3 Reinforcement Learning for Robotics

2.3.1 Principles

Robotics problems make generalized policy iteration difficult for several reasons. These difficulties affect how a robot's RL model should be designed, and how the algorithm for optimizing the parameters of that model should operate.

1. The state and action spaces of a robotics problem are usually continuous. This means that either a parametric policy must be used, or the spaces must be discretized. Fine-resolution discretization can result in hundreds or thousands of states and actions, making the domains of π and V very large. This makes policy iteration computationally expensive, since obtaining an optimal policy requires all states and actions to be iterated over multiple times. Coarse discretization, on the other hand, results in jerky behaviour [15]. For these reasons, RL methods for robotics typically use a parametric policy.

2. Robots are commonly used for tasks that have objectives at several levels of abstraction. At the level of motion, short paths and low energy consumption may be desirable. Rewards for meeting these criteria alone, however, are not useful: They must be supplemented by task-level rewards, rewards for achieving things such as 'Placed teapot in dishwasher' or 'Elicited a smile from the user'.

3. The transition dynamics $\mathcal{T}$ of the environment may not be known, meaning that either
   (a) the agent must learn them,
   (b) or a policy iteration method must be used that doesn't rely on $\mathcal{T}$.

   Approach (a) is known as model-based reinforcement learning, and Approach (b) is called model-free. Learning a model of the environment means approximating the transition distributions in $\mathcal{T}$. The benefit of learning such a model is that it enables techniques from control theory to be applied to the problem: Controller gains can be chosen according to the estimated environment dynamics. These gains correspond to the parameters of the agent's policy [15] [6]. When applied in this way, model-based reinforcement learning is called adaptive control. This is the approach being taken by another member of the project team, who is focusing on its applications to safe robotics. For this reason, it is not discussed in detail here.

4. Learning may need to take place in real-time. This means that the computational demands of the chosen algorithm for policy iteration must be low, or the available computing power must be high.

5. The state information may be highly redundant. If the agent relies on a video feed, it is likely that a relation mapping from raw pixel intensities to relevant features will need to be learnt. This is a difficult task in its own right [12]. Another member of the project team is researching machine vision and navigation, and another again is focusing on predicting when a catastrophic health event has occurred. Their work means that it is reasonable to assume here that pertinent state variables will be readily available in Year 5.

2.3.2 Tasks, Policy Representations, and Learning Algorithms

The first experiment of this project will compare two RL models for robotic control using simulations. This subsection specifies those simulations and explains what models will be compared, and why. The simulations were chosen to highlight the versatility of RL control. The tasks that the agents must solve are taken directly from the literature: In one, an agent must learn to balance a pole atop a laterally movable cart, being scored according to the time elapsed before the pole topples [15]; In the other, an agent must learn to run, being scored according to the horizontal distance it covers before falling. Figure 1 depicts renderings of these tasks. The simulation environment is available through RL Lab, a testing interface maintained by researchers at the University of California at Berkeley. This framework was chosen over others [17] [18] for several reasons. First, the underlying physics engine, MuJoCo, is accurate and stable. Second, the tasks it offers are regarded as benchmarks within the RL research community [19]. Third, the tasks are quite different, both in terms of objective and number of joints actuated. Additionally, using a pre-built simulation package means that more time can be allocated towards designing, executing, and analysing the experiments themselves.

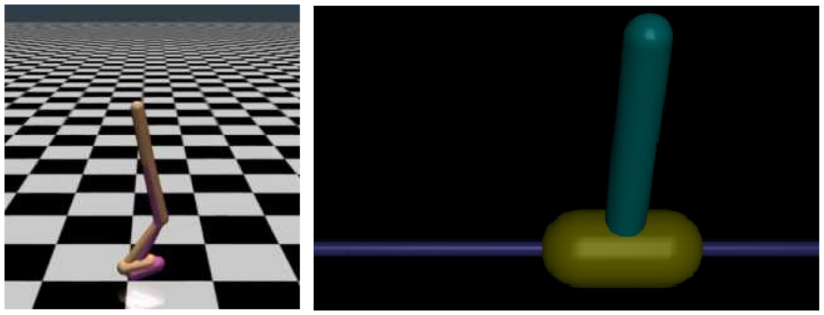

Figure 1: The RL models to be evaluated against two tasks - a locomotion problem (left) and a pole-balancing problem (right).

As has been mentioned, an agent's policy - its mapping from states to actions - must be parameterised if it is to function in continuous state and action spaces. Policy representations that have been applied in the literature include polynomial splines over the state space, mixtures of pre-defined nonlinear dynamical sequences known as 'motor primitives', radial basis function methods that interpolate between previous observations, and neural networks, which are weighted hierarchies of simple functions [15] [6] [20]. The central theme among all of these approaches is that they are methods for approximating differentiable nonlinear functions. Dynamical movement primitives (DMPs) are arguably the most successful representations applied in robotics literature to date: They have been used to teach robotic arms to play table tennis, flip pancakes, pour water, and play pool [21] [9]. It is also thought that they approximately embody how animals perform motor tasks [11]. While the empirical successes supporting DMPs are attractive, recent work in deep learning suggests that neural networks may enable better performance on locomotion and manipulation tasks [22] [23]. No direct comparison of the DMPs and neural networks yet exists in the literature: It is for this reason that Experiment 1 will compare the dynamic motor primitives model described in [21] with a fully-connected feedforward neural network (See [24] for a detailed introduction).

To learn, the parameters of the DMPs and NNs must be optimized with respect to the agent's reward function. Many optimisation algorithms for this problem have been proposed [9] [25], some of which are prohibitively complex for the scope of this project. Attempting to implement a sophisticated algorithm under strict time constraints would be a poor project management decision. The most widely implemented and cited technique in the papers studied is a gradient-based technique known as the REINFORCE algorithm [25] [15]. It is conceptually straightforward, has a low sample complexity[3], and verified implementations of it are readily available within RL Lab. REINFORCE estimates the gradient of the cumulative expected reward with respect to the policy parameters. This estimate is made using a sample of past trajectories, and is used to update the parameters such that the agent's future cumulative returns are increased. This process corresponds to an implicit version of the generalized policy iteration. For clarity, this update rule is

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta J\Big|_{\theta=\theta_k} \qquad \text{such that } J = E\Big[\sum_{k=0}^{\infty} \gamma^k r_k\Big]$$

Where θ_k is the k^{th} iteration of the policy's parameters, J is the cumulative expected reward, and α is a step-size hyperparameter for adjusting rate of convergence. The REINFORCE algorithm is used to estimate $\nabla_\theta J$. For an comprehensive description of how this is done, refer to page 2 of [25]. An alternative gradient estimator that was evaluated is a finite difference method: Run trials with the policy parameters perturbed by $\Delta\theta$, then regress the average returns J for each perturbation the onto $\Delta\theta$s to obtain an estimate of $\nabla_\theta J$. This approach is more difficult to tune for robotics since the appropriate magnitude for $\Delta\theta$ varies by the DOF parameterized, and some perturbations can result in unstable behaviours [25].

In summary, the first experiment will compare two of the most successful RL policy representations for robot locomotion: Dynamical movement primitives and a fully-connected neural network. Both models will use the same reward functions, and will have their parameters fit using the REINFORCE algorithm. The models will be evaluated across many trials according to:

- Their mean score at convergence (i.e. distance covered before falling, time elapsed before the pole topples).
- The variance of that score, a proxy for how reliably a good policy representation can be learnt[4].

2.4 Evaluating the Robustness of a Reinforcement Learning Model

No recent literature appears to report on the effects of actuation flaws on RL controllers for robotics. Presumably this is because research groups will tend to have access to high-grade equipment such as the Kinova arm or Baxter robot, which are sold on the basis that hardware problems are abstracted away from the user. Moreover, research efforts until now have focused on solving the algorithmic problems that RL for robotics faces. This may have meant that practical investigations were postponed for when strong algorithms were available.

Experiment 2 will take the models from Experiment 1 and will analyse how they respond to adverse operating conditions in the locomotion problem. For an RL controller to be successful in the Year 5 project, it must be robust to sensor noise and component wear. This experiment will focus specifically on how component wear might affect performance. Here wear is defined as drift in the effect associated with a control action a. For example, a torque of $a = 5$Nm might be requested by the agent, when only a torque of 4.5Nm is delivered. Note that this issue could be interpreted as a shift in task dynamics.

The experiment will be run according to the following procedure. The model will be fit under zero-noise conditions. This is plausible, since in practice many RL agents for robotics are fit via simulation. Once good performance has been obtained, the model parameters will be checkpointed. From this checkpoint onward, the agent's vector of chosen

[3]i.e. REINFORCE requires few sample trajectories to obtain a good estimate of the policy gradient.
[4]Note that the learning process is stochastic since the agent's policy function is probabilistic.

motor torques $a \in \mathbb{R}^n$ will be modulated by summing it with a randomly generated function of the form

$$\epsilon(t_c) = \beta_0 I_1(a) + \beta_1 t_c$$

Where $I_1(a)$ is a vector for which one element is randomly taken from a, and the remaining elements are set to 0. $\beta_0 \in [-1, 0]$ represents the fraction deducted from the element of a that is sampled. β_0 can be thought of as an absolute difference in behaviour between the simulation and the real system, as might be caused by friction or minor component faults. β_1, on the other hand, represents a small constant wear term across all components that grows larger with t_c, the time elapsed since the check point.

This experiment will be analysed by evaluating the relationship between the values of β_0 and β_1 and the agent's performance after checkpointing. Presumably, a larger step change and greater rate of wear will make it more difficult for the agent to perform well. It is also likely that faults in some actuators are more detrimental than faults in others. The specific questions that this Experiment will address are:

- Quantitatively, how does the agent's score deteriorate as a function of β_0 and β_1 in each component of a?
- Which actuators are performance-critical, and which can only be damaged slightly? Simulations of this type could in principle be used in Year 5 to focus advanced mechanical design on areas that need it most.

The results from both experiments will be synthesized in the Final Report, and will be accompanied by a feasibility study of whether the final models are appropriate for use in a physical robotic system. The order in which these tasks will be completed is outlined in the next Section.

3 Project Work Plan

The following Work Plan describes the planned project timeline. The work structure is sequential: The first experiment is planned and executed, then the second extends its outcomes.

Work package 1 - Preliminary research

1. Research how reinforcement learning problems are posed, and general principles for solving them (✓).
2. Review specific models and algorithms for solving a Markov decision process in robotics (✓).
3. Select a set of simulation tasks on which to run the experiments, and an appropriate framework through which to run them (✓).
4. Identify performance metrics for each stage of experimentation (✓).
5. Prepare the Interim Report - Conduct a risk assessment, map out the project, and summarize the outcomes of the literature review with respect to the project aim (✓).

- Work package 2 - Experiment 1: Which model performs better?

1. Design and validate Experiment 1 (In progress).
   (a) Specify how the experiment will function (by diagram), how its functionality will be validated (i.e. testing scripts, visualizations), what hyperparameter values will be input, what data will be output, and what the specific form of the models being used is (i.e. neural network architecture, or chosen primitives for DMPs).
   (b) Program the experiment according to specification using PyTorch and RL Lab.
   (c) Hire the server specified in the experiment plan from Amazon Web Services and install MuJoCo, RL Lab, and PyTorch.
   (d) Validate the experimental set-up over the course of several prototype runs.
2. Execute Experiment 1.
   (a) Freeze the code and capture metadata (Time, date, OS, library and language versions, GPU and CPU usage).
   (b) Run the experiment.
   (c) Back up the results and code to remote storage.
3. Analyse and review Experiment 1's results: Identify the conceptual and practical differences between the two models, and how these are reflected in the experiment's outcomes. Write the first experiment into the Final Report.

Work package 3 - Experiment 2: Is the better model robust to faulty actuators?

1. Design and validation - produce an experiment specification that adapts the tasks developed in Experiment 1 (Follow steps 1 - 6 in WP 3.1).
2. Execution - This will be similar to WP 2.2.
3. Review the experiment's results: Visualize the behaviour of the system as actuation error increases, and summarize what the outcomes imply for work in Year 5.
4. Write the experimental outcomes into the Final Report.

Work package 4 - Feasibility assessment and Final Report

1. Practical assessment - Define the sensing, computing, and actuating requirements of both models tested in Experiment 1, and describe how the models can be implemented.
2. Technical assessment - Evaluate the conceptual limitations of the models, and suggest how they might be mitigated in a practical robotics problem.
3. Synthesize the practical and technical assessments to form a specification of what tasks the RL models can be used to accomplish in Year 5.
4. Link the experimental reports and feasibility assessment to form the Final Report.

Work package 5 - Presentations and posters

1. Prepare a presentation summarizing the outcomes of Experiment 1 & 2 in preparation for a progress review meeting with Consequential Robotics.
2. Prepare the Group Poster, including a live demonstration of the first experiment's outcomes.
3. Viva voce examination.

4 Resource Requirements

The software requirements for this project are available under free licenses. The sole expense in this project is computing power: The simulations and models to be fit are computationally expensive. This resource will be sourced through Amazon's server rental service, AWS. The cost of this resource is quite low per hour (see Table 3), but could easily surpass budget if poorly managed, as will be discussed in the next section.

Resource	Supplier	Cost per unit	Total cost
High-performance computer	Amazon Web Services	\$1 - 3.3/hour	< \$20 - 65
Development language - Python 3.5	Python Software Foundation	Free	-
Optimization and linear algebra library - PyTorch	Open-source	Free	-
Physics engine and simulator - MuJoCo	University of Washington	Free	-
RL API for MuJoCo - RL Lab	UC Berkeley/Open-source	Free	-

Table 3: The project's resource requirements.

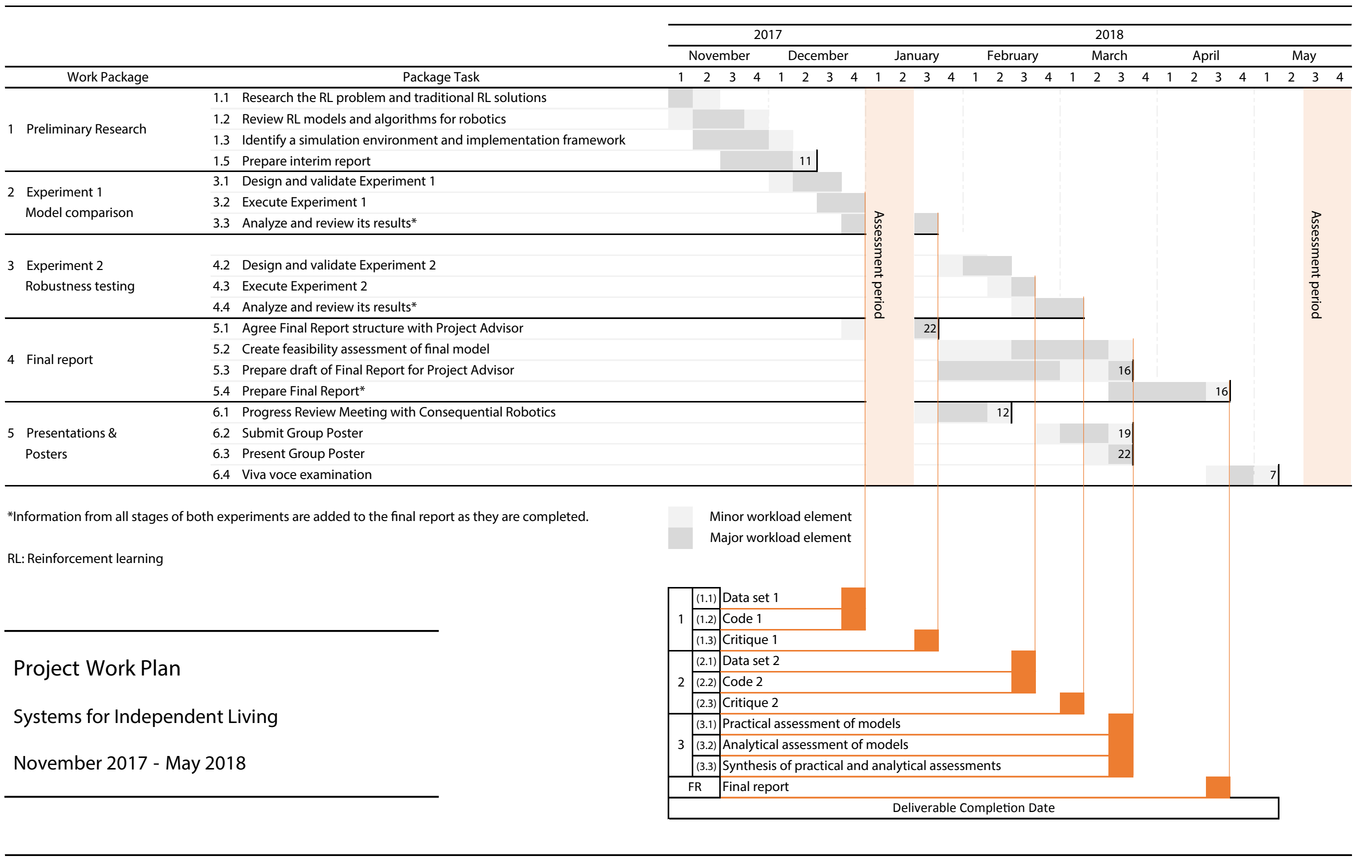

Figure 2: The planned project progression, including deliverable completion dates highlighted in orange, and University deadline dates specified as numbers. Each work package can only be completed if the outcomes of earlier work packages are available. Within a work package tasks must also be completed sequentially.

5 Project Risks

This section outlines the project risks that have been identified, and how they have been mitigated. Table 4 summarizes the issue discussed in detail below, and points to several other risks that the project faces: Data loss, late experimentation, exceeding the project budget, and failing to deliver a useful outcome. Rather than discuss those other problems further here, the reader is directed towards the passages in that Table.

The biggest risk to successfully completing the project is a knowledge shortfall. The experiments must be implemented in unfamiliar software, and the models to be used are conceptually demanding. The experiments will be written in Python, an accessible programming language, however extension libraries such as RL Lab and PyTorch are also needed. New libraries can be learnt within a reasonable timeframe, but configuring a computer on which to run them can be time-consuming and error-prone. To mitigate this risk, three months have been dedicated exclusively to experimental work: This includes time to set up the computers, verify them, and build familiarity with the simulation environment. In the worst-case scenario that set-up takes considerably longer than expected, the second experiment could, in principle, be dropped. However, this should be a last resort: If technical issues are obstructing progress, help will be sought through faculty in the University's Department of Computer Science.

The conceptual complexity of the field may also obstruct progress, a risk that applies to many advanced engineering topics. During the initial literature review, for instance, several learning algorithms were found that relied on graduate-level techniques in information theory and statistics. These techniques would have required months or years of intensive study to understand fully, and were passed over in favour of methods that would be simpler to explain and implement. Some aspects of the field also rely on a dense jargon that will need to be untangled in the Final Report. This will take time to do properly, and has been scheduled for accordingly in the Work Plan.

Risk	Likelihood	Impact	Current			Mitigation	Residual		
			L	I	S		L	I	S
Failure to close knowledge gap The simulation frameworks take longer to learn than planned, or the models are more difficult to implement than expected.	*Unlikely* The chosen models have previously been implemented, and the code for them is publicly available (albeit for different problems).	*Very high* Failing to implement the experiments would have similar consequences to *Late experimentation* - key deliverables would not be produced.	2	5	10	Schedule more time for experimentation. Select a learning algorithm that is readibly interpretable and explainable (i.e. REINFORCE). Start prototyping models now to become familiar with RL Lab and PyTorch. Closely monitor progress relative to Work Plan.	1	5	5
Loss of data/code Hard drive failures or file management mishaps may cause experimental data or code to be lost.	*Likely* These files will be used regularly over the course of half a year, making it likely that data loss will occur.	*Very high* Losing the project's deliverables would be catastrophic, requiring dozens or hundreds of hours to recreate.	3	5	15	Manage code and experimental data using Github. Commit work on a daily basis, minimizing the impact of data loss events. Back up data to a local hard drive weekly.	3	1	3
Late experimentation The experiments take longer to complete than expected, or are retrospectively invalidated.	*Very likely* Mistakes in poorly planned experiments and software development are common. Both could substantially affect the validity of the experiments and the research project.	*High* Debugging and validating experiments is time-consuming and unpredictable. Errors would probably require minor revisions of existing code, and would be less damaging than data loss events.	4	4	16	Schedule for detailed experiment planning, 'dummy' model testing, rehearsal experiments, and validation scripts. Provide a buffer of 1.5 months between analysis of the final experiment and the Final Report submission date.	2	3	6
Exceeding budget Renting high-performance computing services for longer than planned might take the project over-budget.	*Very likely* Since this will occur if either data is lost or experiments need re-running, it is likely that more hours may be needed than planned.	*Very high* Being unable to run experiments would prevent the project's deliverables from being produced.	4	5	20	Investigate University GPU computing resources. Consider computational cost when designing experiments - reduce problem size where possible. Set auto-expiry timers to ensure that servers are not left running overnight.	2	2	4
Outcome irrelevance The methods investigated are found to be impractical for the Year 5 project.	*Unlikely* There are many examples of RL systems being usefully applied in the literature, even if not in a domain transfer setting.	*Low* Another member of the team is researching control, and can therefore meet the project's needs if an RL system should prove unsuitable.	2	2	4	Ensure that the experimental tasks are mappable to known home social care tasks. Extend report critique to specify how the model's learning and optimization components could be used in isolation.	1	1	2

Project Risk Assessment

Systems for Independent Living

L	Likelihood of the risk becoming an issue.
I	Impact if the risk becomes an issue, as measured in time lost.
S	Severity of risk (Expected impact).
Current	Risk profile prior to implementing mitigating measures.
Residual	Risk profile after factoring in mitigating measures.

Key

	Likelihood				
Impact	5 - Certain	4 - Very Likely	3 - Likely	2 - Unlikely	1 - Very Unlikely
Very High (> 1 week) - 5	25	20	15	10	5
High (< 1 week) - 4	20	16	12	8	4
Medium (3 - 5 days) - 3	15	12	9	6	3
Low (1 - 2 days) - 2	10	8	6	4	2
Very Low (< 1 day) - 1	5	4	3	2	1

Table 4: A summary of the major project risks, their likelihoods of occurring, and the impact they would have on the project if they did occur.

References

[1] O. for National Statistics, “Disability in england & wales: 2011 and comparison with 2001,” 2011.

[2] N. A. Office, “Adult social care in england: Overview,” 2014.

[3] N. I. Centre, “Survey of carers in households,” 2010.

[4] U.-R. Network, “Robotics in social care: A connected care ecosystem for independent living,” 2017.

[5] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* A Bradford book, Bradford Book, 1998.

[6] S. Schaal and C. G. Atkeson, “Learning control in robotics,” *IEEE Robotics & Automation Magazine*, vol. 17, pp. 20–29, 2010.

[7] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[8] C. A. Rothkopf and C. Dimitrakakis, “Preference elicitation and inverse reinforcement learning,” in *ECML/PKDD*, 2011.

[9] C. C. C. D. Kormushev, P., “Reinforcement learning in robotics: Applications and real-world challenges,” vol. 2, no. 3, pp. 122–148, 2013.

[10] “Unimation, the first industrial robot.” https://www.robotics.org/joseph-engelberger/unimate.cfm. Accessed: 05/12/2017.

[11] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot Programming by Demonstration*, pp. 1371–1394. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[13] T. P. S. R. Barto, A., “Some recent applications of reinforcement learning,”

[14] S. Gu, E. Holly, T. P. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation,” *CoRR*, vol. abs/1610.00633, 2016.

[15] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *I. J. Robotics Res.*, vol. 32, pp. 1238–1274, 2013.

[16] A. Y. Ng, D. Harada, and S. J. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *ICML*, 1999.

[17] T. Erez, Y. Tassa, and E. Todorov, “Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx,” *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4397–4404, 2015.

[18] “Openai gym.” https://gym.openai.com/envs/. Accessed: 05/12/2017.

[19] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *ICML*, 2016.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Computing Research Repository*, vol. abs/1509.02971, 2015.

[21] y. v. p. Katharina Mülling and Jens Kober and Oliver Kroemer and Jan Peters, journal=I. J. Robotics Res., "Learning to select and generalize striking movements in robot table tennis,"

[22] S. Gu, T. P. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *ICML*, 2016.

[23] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *CoRR*, vol. abs/1709.10087, 2017.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[25] J. Peters and S. Schaal, "Policy gradient methods for robotics," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2219–2225, 2006.