Data-Driven Reinforcement Learning for Virtual Character Animation Control

Vihanga Gamage
School of Computer Science
Technological University Dublin
Dublin, Ireland
vihanga.gamage@tudublin.ie

Cathy Ennis
School of Computer Science
Technological University Dublin
Dublin, Ireland
cathy.ennis@tudublin.ie

Robert Ross
School of Computer Science
Technological University Dublin
Dublin, Ireland
robert.ross@tudublin.ie

ABSTRACT

Virtual character animation control is a problem for which Reinforcement Learning (RL) is a viable approach. While current work have applied RL effectively to portray physics-based skills, social behaviours are challenging to design reward functions for, due to their lack of physical interaction with the world. On the other hand, data-driven implementations for these skills have been limited to supervised learning methods which require extensive training data and carry constraints on generalisability. In this paper, we propose RLAnimate, a novel data-driven deep RL approach to address this challenge, where we combine the strengths of RL together with an ability to learn from a motion dataset when creating agents. We formalise a mathematical structure for training agents by refining the conceptual roles of elements such as agents, environments, states and actions, in a way that leverages attributes of the character animation domain and model-based RL. An agent trained using our approach learns versatile animation dynamics to portray multiple behaviours, using an iterative RL training process, which becomes aware of valid behaviours via representations learnt from motion capture clips. We demonstrate, by training agents that portray realistic pointing and waving behaviours, that our approach requires a significantly lower training time, and substantially fewer sample episodes to be generated during training relative to state-of-the-art physics-based RL methods. Also, compared to existing supervised learning-based animation agents, RLAnimate needs a limited dataset of motion clips to generate representations of valid behaviours during training.

KEYWORDS

Virtual Characters, Reinforcement Learning, Animation

1 INTRODUCTION

Virtual character animation control is an area of great interest as engaging human characters have great potential as mediums for interaction in games and other applications [4, 5]. While motion capture methods can be used to create animation, it can be time-consuming and expensive if required to capture a wider, varied range of behaviours. As a result, data-driven animation synthesis has been a problem that has drawn a great deal of interest.

When exploring novel solutions to data-driven animation control, neural network-based approaches have been a popular choice. Earlier work used supervised learning to create agents that output robust, varied animation, while learning from a limited set of reference motions [11, 13, 15, 27]. However, due to their heavy reliance on the training dataset, these approaches offer limited flexibility, and can lead to unpredictable outputs in conditions different to the training set. As a result, recent work has explored reinforcement learning (RL) to address this issue, with physics-based simulation being leveraged to implement character animation agents, such as DeepMimic [18, 20]. Such methods rely heavily on interaction with physical surfaces and objects, as the feedback signals from the physics engine are required for agents to learn and function.

However, many applications using virtual humans require that they portray social, interactive behaviours, such as gestures, nods, exclamations or pointing. These behaviours do not elicit robust physics feedback signals, and therefore physics-simulation RL is not applicable. For portrayals of social behaviours to be effective, they need to be human-like, which poses a challenge from a modelling perspective. Taking pointing as an example, a function may be formulated that defines a successful pointing behaviour as one that returns to a starting pose after the direction of the index finger aligns with the vector to the target. Obtaining an optimal sequence out of the exponentially numerous possible trajectories is a problem without a straightforward solution, as the difference between a human-like portrayal and a more robotic one is difficult to define. Furthermore, for behaviours such as waving (in contrast to pointing) more dimensions would be required to articulate even a limited definition.

Recent work carried out using model-based RL has demonstrated that model-based agents can be trained to learn, within a compact latent state, dynamics required for agents to function robustly [8, 9, 24]. However, most state-of-the-art model-based RL approaches typically involve agents learning to solve tasks such as Cartpole and Walker from the DeepMind control suite [25]. Animation control allowing for the portrayal of human-like behaviour is a more challenging task given that the dimensionality of the action space is much higher, and overall domain is more complex.

In this paper, we present RLAnimate, a data-driven RL approach to create model-based agents for virtual character animation control. We use a latent dynamics model to learn dynamics for animation and portrayed behaviour in a way that allows for agents to be robustly trained to generate animation portraying versatile, human-like behaviour. We formalise a framework for modelling the problem in a way that allows for a dual information state. This allows for leverage to learn two set of latent dynamics: one on the behaviour portrayed that is deterministic in nature, and the other for character animation dynamics universally applicable to any behaviour portrayed, which consists of deterministic and stochastic components. Agents are trained to output animation by self-generating a description for the next pose as per an objective signal, after which learned dynamics are applied to obtain latent

representations used to calculate the rotations for the most optimal pose.

We summarise the contributions from our work as follows:

- **Novel modelling framework for character animation**: we show that the mathematical structure we present that splits information into a behaviour portrayal-focused objective and a globally valid description, allows for the efficient training of a model-based agent to portray multiple behaviours.
- **Latent dynamics model for human-like animation**: by maintaining two latent spaces, the model learns to represent dynamics disentangling those for the behaviour portrayed, from those for the animation, which is the actuation of the behaviour portrayal. Our evaluation shows that this has a key impact on agent performance, and sample efficiency.
- **Training algorithm for RL animation agents informed by motion data**: modelling precise definitions for human-like behaviour, is challenging in terms of formulating a RL reward function. We implement a mechanism that allows for motion data to be used to inform the training process of representations for valid behaviour portrayal.

2 MODELLING HUMAN-LIKE BEHAVIOUR PORTRAYAL

In our work, we are concerned with using RL to create virtual character animation control agents. An example of existing work applying RL to this problem is the work done by Peng et al. on DeepMimic learning physics-driven tasks [20]. They combine an imitation objective with a task objective to train agents to output a range of physics-based character skills such as running and backflips.

However, we are interested in agents generating animation that portray social behaviours such as gestures, that do not elicit requisite feedback from a physics engine to effectively train agents. Therefore, a novel approach that would enable RL agents to function, relying on signals that are ubiquitously present and applicable to animation regardless of behaviour, such as joint rotations and positions, is required.

2.1 Modelling Reinforcement Learning Problems

Problems that are to be resolved by the use of RL are generally described as a Markov Decision Process (MDP). Key components of a MDP are the state space $s \in S$, the set of actions $a \in A$, a state transition probability function $P(s_{t+1} = s' | s_t = s, a_t = a)$ representing the probability of the successor state at $t+1$ for a given action a taken in state s at time t, and a reward term $R_a(s, s')$ to represent the reward of a given action a that leads to the transition from a state s to a successor state s'. A RL algorithm then finds an optimal policy $\pi(s) \rightarrow a$ which is a distribution over actions given states that maximises cumulative reward.

In using a MDP to define the problem, an assumption is made that the state space follows the Markovian property $P(s_{t+1}|s_t) = P(s_{t+1}|s_1, ..., s_t)$, i.e., that the state at any given time depends only on its immediate predecessor. While assuming the Markov property is often needed to enable a theoretical structure that allows for convergence when learning optimal policies using RL algorithms, in some domains and settings, it may be more beneficial to explore alternate structures within which to apply RL algorithms. Model-free RL heavily relies on the assumption of the Markovian property which leads to the notion that all information required to obtain the ideal action is present in the representation of the state at the current time. An optimal model-free RL policy is a distribution that would allow for actions to be sampled per the state, to maximise reward without attempting to learn a model. Model-free RL is preferred in cases where a model is not available for the setting of the problem, or when not possible to learn an accurate model. Furthermore, the lack of reliance on a model makes model-free methods more adaptable to uncertainty or novelty during operation, relative to training. However, model-based RL offers many advantages such as sample efficiency if an accurate model for the dynamics relevant to the problem is present or can be learnt.

2.2 Generating Human-like Animation

With that in mind, we examine our problem domain of virtual human animation, where our goal is to generate optimal animation within a predictable range of behaviours, while the portrayals themselves are believable mimicry of human behaviour. To provide a brief overview of character animation, highlighting the key aspects required to understand the work presented in this paper, a virtual human character is a three-dimensional (3D) representation of a human, and animating a virtual character involves manipulating this 3D representation, typically consisting of a polygon mesh [23]. This manipulation is enabled by the mesh being rigged with a hierarchical rigid-body structure, usually referred to as the skeleton, and consists of a configuration of connected joints [19]. The rotation of these joints results in animation.

The credit assignment problem is a prominent challenge in RL modelling, with a solution addressing the question of which actions taken over a period were most relevant to obtaining a particular reward [16, 26]. In character animation, an output animation sequence needs to coherently portray a specific behaviour, and needs to be, out of millions of possible options, one that portrays a human-like version of the behaviour. An effective reward function would need to inform the training algorithm whether a particular shift in one trajectory makes the behaviour portrayed more accurate, more realistic, neither, or both. These two aspects, i.e., whether the overall behaviour is the right one, and whether this behaviour is human-like, are difficult to articulate mathematically.

2.3 Mathematical Framework for Character Animation

We address these challenges by following a model-based RL approach within a mathematical framework for modelling the virtual human animation tasks, that is refined to leverage attributes of the animation domain. While human understandable functions are not easy to define for behaviours such as gestures, motion capture data can be used to inform the agent based on latent definitions for ideal animation sequences. Compounded with the ability to learn accurate model for animation dynamics, this allows us to utilise the challenge of requiring consideration over a series of actions from

preceding steps to portray human-like behaviour, as it would allow us to enhance the sample efficiency of the learnt models.

A goal of an animation agent is to generate an animation sequence $\{a_t\}_{t=0}^{n}$, where a_t represents the pose for each timestep for the total length of the sequence n. We split the state space into two: $s_t = o_t \cdot d_t$, with the state s at a given time comprising of an objective o and description d. The objective contains information on the behaviour that is to be portrayed, whereas the description contains information on the current pose of the character. This allows for animation dynamics being learnt in a behaviour agnostic manner.

Agents are trained to maximise the idealness I of animation at each timestep, maintaining naturalness while successfully portraying the behaviour mandated via the objective state at each timestep. This overall training objective can be summarised as follows:

$$E\left[\sum_{t=0}^{n} I_t\left(a_0, ..., a_t, o_t, d_t\right)\right] \tag{1}$$

For this split state space, the Bellman equations can be written as follows [2]: the value of a state can be defined as $v_\pi(s)$ defined as $\mathbb{E}_\pi\left(I_{t+1} + v_\pi\left(S_{t+1} = O_{t+1} \cdot D_{t+1}\right) | S_t = o \cdot d\right)$, and $q_\pi(s, a)$ defined as $\mathbb{E}\left(I_{t+1} + v_\pi\left(S_{t+1} = O_{t+1} \cdot D_{t+1}\right) | S_t = o \cdot d, A_t = a\right)$, which is the value of a pair of state and actions.

3 DATA-DRIVEN REINFORCEMENT LEARNING

In the previous section, we presented the mathematical framework we use to model human-like behaviour portrayal tasks in RLAnimate. The key challenge faced when training RL agents to model social behaviours is that a traditional RL environment can not be created, as it is difficult to generate formulae to assign rewards based on observations, particularly when the requirement for realism is considered. RLAnimate addresses this via agents learning dynamics for behaviour, and animation portraying these behaviours, which are then used to learn representations for valid, human-like behaviour. Figure 1 contains an overview of the waving and pointing behaviours used to demonstrate our approach in this paper. In this section, we describe the models learnt by an RLAnimate agent to do this, and the methodology by which they are trained.

3.1 Agent Models

Figure 2 presents an overview of our approach, within which agent function is a result of three co-operating models. We first present the design for these models before examining our data-driven RL algorithm in detail.

3.1.1 Portrayal Model. At each time step, the agent receives an objective state signal o_t from the environment. The portrayal model $p(d_t|o_t)$ is responsible for generating a description state d_t, based on the objective. The description state is a human-understandable vector containing information about the pose of the character, and the role of the portrayal model is to express a likely description that would satisfy the current objective state. Details regarding the description and objective state definitions are presented later in this section. To update models, RLAnimate collects episode data in a data buffer B. The task module that provides the objective

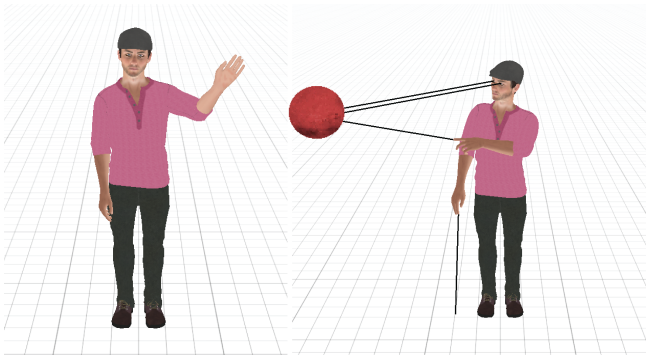


Figure 1: An overview of the waving (left), and pointing (right) behaviours used in this work. The vectors used in the description space are illustrated on the right.

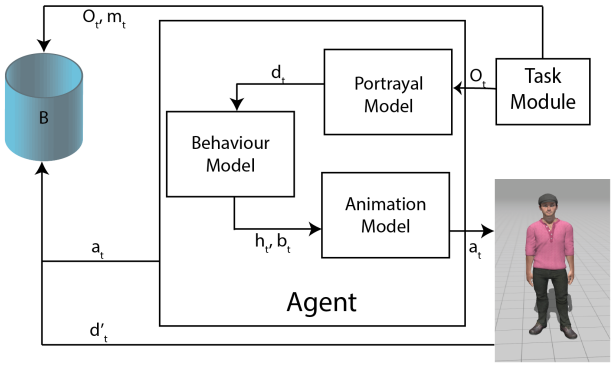


Figure 2: Overview of agent training and function. At each timestep, the agent recieves an objective signal o_t from the task module in the environment. Using the portrayal model, the agent self-generates a corresponding ideal description d_t for the next animation pose. This self-description in conjuction with the dynamics learnt allows an agent to obtain, from the behaviour model, latent states for task h_t and behaviour b_t. These latent states are the used to generate the animation action a_t. The environment generates a real description signal which is stored in the sample buffer along with the action, objective and ideal animation m_t per the relevant motion clip.

signals passes o_t to B as well as the ideal motion m_t to portray that objective obtained from the motion capture dataset.

3.1.2 Behaviour Model. The behaviour model is a latent state-space model that is responsible for learning latent animation dynamics. The design of this model was influenced by that of the recurrent state space model presented by Hafner et al. in Planet [8]. We subscribe to their thinking that for robust dynamics to be learnt, the hidden states of the model needs to be split into deterministic and stochastic components. However, to address the challenges of portraying multiple behaviours, we incorporate several augmentations.

Our behaviour model maintains a pair of latent states: the hidden task state h_t and behaviour state b_t. The hidden task state is modelled as purely deterministic, whereas the behaviour state is split into deterministic and stochastic components. The stochastic behaviour state is modelled using a normal distribution. The manner in which the behaviour model applies learned dynamics to update these states can be summarised as follows:

$$\begin{aligned} \text{Task state: } & h_t = f_1(h_{t-1}, o_t) \\ \text{Behaviour state: } & b_t \sim p(b_t | f_2(h_t, b_{t-1}, a_{t-1})) \end{aligned} \tag{2}$$

An overview of the behaviour model design is shown in figure 3, and we will examine the model design and role of the dynamics learnt later in this section.

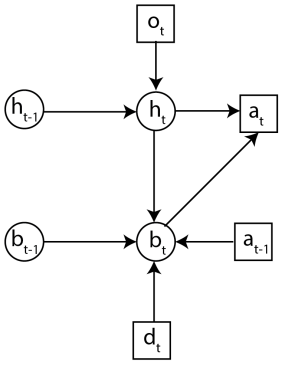


Figure 3: Latent dynamics learnt by RLAnimate agents. The deterministic task state h_t is obtained as a recurrent neural network (RNN) $f_1(h_{t-1}, o_t)$, and the deterministic component of the behaviour state as a second RNN $f_2(h_t, b_{t-1}, a_{t-1})$, which is used to generate the final behaviour state posterior b_t. h_t and b_t (computed as conditioned by d_t generated by the portrayal model)is used by the animation model to generate the next animation action a_t.

3.1.3 Animation Model. The animation model $p(a_t|h_t, b_t)$ receives the hidden task state h_t, and behaviour state posterior b_t and outputs an action a_t which are the rotations for the pose at that timestep of the animation sequence that are applied to the character model. The animation model output is parameterised as a Beta distribution, which is a class of continuous probability distributions that is defined within a bounded interval $[0, 1]$, parameterised by two shape parameters α and β. We made this choice for output layer parameterisation, as Beta distribution-based policies have been shown to be more effective in continuous control reinforcement learning, by addressing issues caused by the mismatch between the infinite support of the Gaussian distribution and bounded controls [3]. The a_t output by the agent is applied to the character, and stored in the episode buffer. When the animation is applied, the environment generates a real description d'_t which is also sent to the episode buffer.

Table 1: Objetive configuration. t is the current step of the sequence, and N the total number of steps. Note that for attributes, the signal for waving contains a repetition of the mood variable, to account for the differences in sizes.

	Type	Arm		Attributes			Time
Point	1	Left	Right	Unit vector to target			t/N
		0	1	x	y	z	
Wave	2	Left	Right	Exaggeration			t/N
		0	1	0 - 1	0 - 1	0 - 1	

3.2 Learning Animation Dynamics

A core element of RLAnimate is that of leveraging the nature of character animation. With the problem of portraying realistic character animation comes several complexities. One of which is the difference between realistic and unrealistic behaviour being ineffable in terms of defining a human-understandable function for computational purposes. This challenge is further enhanced by the nature of what character animation is portraying: human behaviour. Consider a portrayal where a human character points at a target straight ahead with the left arm. There are an exponentially large number of possible trajectories the arm could take, some more realistic than others.

However, the mechanics of the arm do not change, regardless of the target being pointed at, or even if the portrayal is of an entirely different behaviour such as waving. Hence, while human behaviour may very well be infinite in its complexity, our action space, which in this case is an approximation of the human body in the form of the joints used when modelling character animation, makes for a very structured medium for learning.

3.2.1 Objectives and Descriptions. This consistently structured action space of animation affords us leverage in order to address the challenges posed by the complexity of portraying human-like behaviours. To enable this, we provide input to the agents splitting the states into objectives and descriptions, both of which are human understandable form.

Objectives: The objective signal essentially tells the agent about the behaviour that it needs to portray. The configuration for objective we use in this work is as displayed in table 1.

Description: Based in a geometric space that is constant regardless of the behaviour, the description signal describes the current state of the character. It is a combination of information on the effectors and joint positions as denoted in table 2.

3.2.2 Latent Dynamics. The split objective and description states is the first step that enables us to create an approach to learn latent dynamics for character animation. Referring back to the behaviour model described earlier in this section, the duality is mirrored in the latent dynamics states. The task state h_t is purely deterministic, which allows it to capture the dynamics concerning the objective. Using our earlier analogy, there might be a large number of ways to point forward or wave, but determining whether the right action is being portrayed, or whether the target being pointed at is accurate, does not require stochastic dynamics.

Table 2: Description configuration. The description consists of information about the virtual human character, constructed using the unit directional vectors for the effector articles and positions for joint articles. These vectors are illustrated in figure 1

.

Attribute	Articles	Size
Effector Vector	left & right eyes, and index fingers.	12
Joint Position	left & right collars, shoulders, elbows, wrists, and index finger bases.	30

The behaviour state b_t is a more complex latent state that consists of deterministic and stochastic components. It is updated by applying learned dynamics to the deterministic hidden task state, using the current animation pose. The deterministic task state captures the dynamics related to the objective state, which concerns information with regards to the variations in behaviours portrayed. As a result, the dynamics learned to update the behaviour state involve universally applicable information on animation.

Due to the non-linear nature of the model, the hidden behaviour state cannot be directly computed. We infer a behaviour posterior state using an encoder from the predicted description provided by the portrayal vector, expressed by equation 3. Therefore, the behaviour state posterior can be considered as a representation of the most realistic animation poses for a given time step.

$$q(b_{1:T}|d_{1:T}, a_{1:T}) = \prod_{t=1}^{T} q(b_t|f_2(h_t, b_{t-1}, a_{t-1}), d_t) \tag{3}$$

3.3 Training Agents

In the previous sections, we presented how an RLAnimate agent uses latent dynamics to represent behaviour portrayals and generate animation. In addition to learning latent dynamics, to portray behaviour, an agent also needs to obtain an understanding of valid behaviour portrayals. As described in algorithm 1, RLAnimate trains agents by generating rollout episodes of portrayals imitating motion clips provided, and iteratively updating the model parameters. To train agents, C batches of episode chunks $\left\{\left(o_t, a_t, d'_t, m_t\right)_{t=k}^{L+k}\right\}_{i=1}^{C}$ of length L are drawn from the sample buffer where rollouts are collected in. The training objective seeking to maximise the ideality of animation consists of components $L1$ and $L2$ to learn latent dynamics for animation, and $L3$ to obtain an understanding of valid behaviour portrayals.

$$L(\theta) \triangleq E\left[\sum_{t=0}^{n} I_t\left(a_0, ..., a_t, o_t, d_t\right)\right] \triangleq L1(\theta) + L2(\theta) + L3(\theta) \tag{4}$$

The losses associated with learning the latent dynamics consist of the following description construction objective component $L1$ is calculated as the mean squared error, as expressed in equation 5, and KL divergence component ($L2$) based on the difference between the prior and posterior distributions for the behaviour state, obtained through the learned dynamics as expressed by equation 6 [10].

Algorithm 1 RLAnimate

```
Initialise episode buffer B.
Initialise models with random parameters θ.
while not converged do
    if training for waving and pointing then
        draw at random an example (m_t)_{t=1}^F for each from motion dataset M
    else
        draw a single example (m_t)_{t=1}^F from motion dataset M
    end if
    for step t = 1..F do
        Update deterministic task state h_t from o_t
        Infer behaviour posterior conditioned on d_t generated via the portrayal model
        Generate a_t and apply to character
        Collect real description d'_t
    end for
    B → B ∪ {(o_t, a_t, d'_t, m_t)_{t=1}^F}
    for model update step s = 1..T do
        Draw episode chunks {(o_t, a_t, d'_t, m_t)_{t=k}^{L+k}}_{i=1}^C ~ B at random from data buffer
        Compute loss L(θ) per (4)
        Update model parameters θ ← θ − α∇_θ L(θ)
    end for
end while
```

$$L1(\theta) = mse\left[d'_t - E\left[p\left(d_t|o_t\right)\right]\right] \tag{5}$$

$$L2(\theta) = KL\left[p\left(b_t|h_{t-1}, o_t, b_{t-1}, a_{t-1}\right) \parallel q\left(b_t|d_t, a_{t-1}\right)\right] \tag{6}$$

$L3$, as denoted by equation 7 is a Huber loss minimising the difference between the original animation from the motion clip, and the animation generated by applying the dynamics learnt by the portrayal and behaviour models [14]. If the losses are over the threshold δ, a linear function is used; otherwise, the function is quadratic.

$$L3(\theta) = \begin{cases} \frac{1}{2}\left(M_t - f\left(h_t, b_t\right)^2\right) & \text{for } |M_t - f\left(h_t, b_t\right)| \leq \delta \\ \delta\left|M_t - f\left(h_t, b_t\right)\right| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{7}$$

The losses calculated with $L3$ are backpropagated through the rollout sequences, gradients flowing through the behaviour model to the portrayal model, allowing the agent to learn representations for valid behaviours in correspondence with the latent dynamics model.

4 EXPERIMENTS AND EVALUATION

For our implementation, we use a game environment that builds on the the Panda3D engine [6]. From Adobe Mixamo, we obtained 50 motion clips for pointing that covered a wide range of targets using both arms, as well as 50 motion clips for waving at various levels of exaggeration for each arm [1]. To evaluate trained agents, we held back a selection of 10 motion clips that included 6 pointing behaviours and 4 waving behaviours.

These testing clips included 3 clips each of a pointing portrayal using each arm, and 2 each for portrayals of waving with different levels of exaggeration to the training clips. A detailed list of the motion clips used for training and evaluation, as well as an overview of the technical implementation is available in the supplementary material. When evaluating agents, we measure successful behaviour portrayal by using a metric that calculates the difference between joints positions generated playing the agent output and reference clips, to obtain a score out of 100 using 100−total error/frames. The error per frame is obtained as expressed in equation 8. The total error is calculated by adding a penalty term for per frame errors over 1, per equation 9.

$$error_t = \sum_{j=0}^{J} \left(\left| p_x^j - p_x^{j\prime} \right| + \left| p_y^j - p_y^{j\prime} \right| + \left| p_z^j - p_z^{j\prime} \right| \right) \tag{8}$$

$$\text{total error} = \sum_{t=0}^{T} error_t + max \left\{ 0, \log_{1.01} error \right\} \tag{9}$$

We also calculate a smoothness metric for final agent behaviour portrayals out of 100. Particularly with neural network-based approaches, data-driven animation output can contain artefacts that cause the motion to appear shaky. We use a Savitzky-Golay filter to obtain a smoothed version of the animation sequence, obtain a sum for the differences relative to the difference between the original motion clip and its smoothed version [22].

A video that contains a series of visual demonstrations and comparisons that emphasises the differences in performance and output quality of the agents and controls evaluated can be found at https://virtualcharacters.github.io/links/ALA2021. In our future work, we plan to carry out a perceptual evaluation using human participants. All our experiments were carried out using a workstation with an Intel Core i7-8750H 2.2GHz CPU and a Nvidia GeForce GTX 1070 GPU. We found the average throughput time for an RLAnimate agent rendered game to output animation was 0.0167 seconds per frame.

4.1 Ablation Studies

We carried out a series of experiments to ascertain the effect of key RLAnimate elements. In addition to evaluating using the test set of motion clips withheld from training, we also scored agent behaviours using a set of 10 clips that were included in the training set.

4.1.1 Spit observation and description Signals. Figure 4 compares the performance of RLAnimate to a controller that provides the agent an input of the objective as a single state. The encoder used to parameterise the behaviour posterior (Equation (3)) is still trained using the description, but the self-description of the next ideal pose carried out using the portrayal model is eliminated. While the single state controller achieves a serviceable empirical score, it is notable that, particularly for waving, the animation generation contains significant artefacts, in the form of shakiness. The single state agent has the lowest smoothness measurement of all the agents we evaluated (Table 3), demonstrating that self-description by agents plays a key role in RLAniamte agents being able to learn animation dynamics effectively.

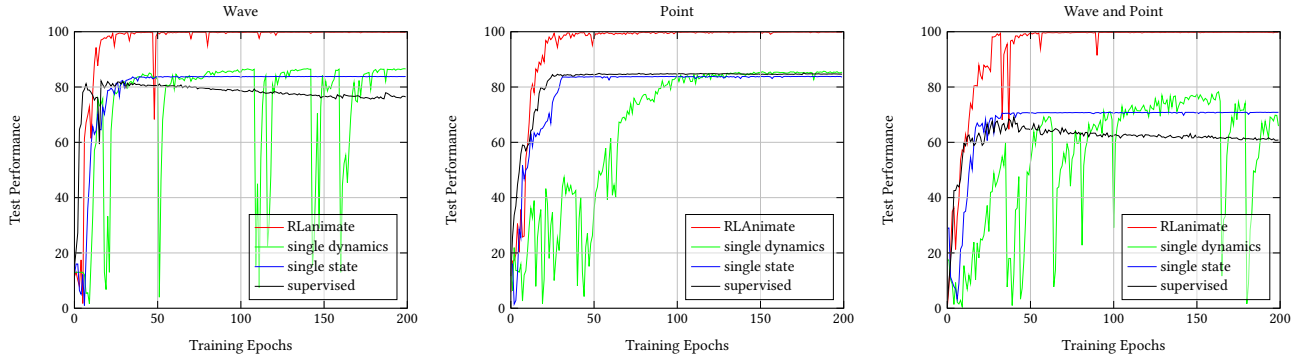
4.1.2 Latent dynamics space. We also compare the performance of RLAnimate to a control agent that used an alternate dynamics model that learnt animation dynamics using single latent space consisting of deterministic and stochastic components. This version is able to remain somewhat competitive relative to RLAnimate when portraying single behaviours, previously encountered in the test set. But when trained to portray either behaviour, there is a noticeable loss in performance. This demonstrates that using dual latent spaces to learn dynamics for tasks and animations separately allows agents to function more efficiently. Comparing the performance of an agent using the alternate dynamics model between imitating seen and unseen motions, while the difference is significant when trained to point or both, when waving, the difference in performance is much smaller, with the scores for the test set it had not encountered in training remaining within 4 points of imitating clips from the training set. This is due to the effect on animation due to the variation in degree of exaggeration being more predictable relative to the differences required to point at different targets. But, this agent is unable to maintain this performance when trained to perform either waving and pointing.

Additionally, the training for this agent appears significantly more unstable relative to the other agents. This affect is more pronounced and present throughout for waving and combined behaviours relative to pointing. This demonstrates further the benefit of using dual latent spaces to learn dynamics. Learning dynamics for animations via the descriptions universally, and separately learning dynamics for how objectives relate to ideal descriptions allows agents to function more effectively, particularly when portraying different behaviours. Even though this agent is able to learn to portray waving behaviours best, the mode of latent space does not allow for optimal dynamics to be captured, causing it to struggle. And while the agent seems to be able to learn dynamics in a more stable manner when trained to only point, it is unable to apply them effectively to portray unseen behaviours.

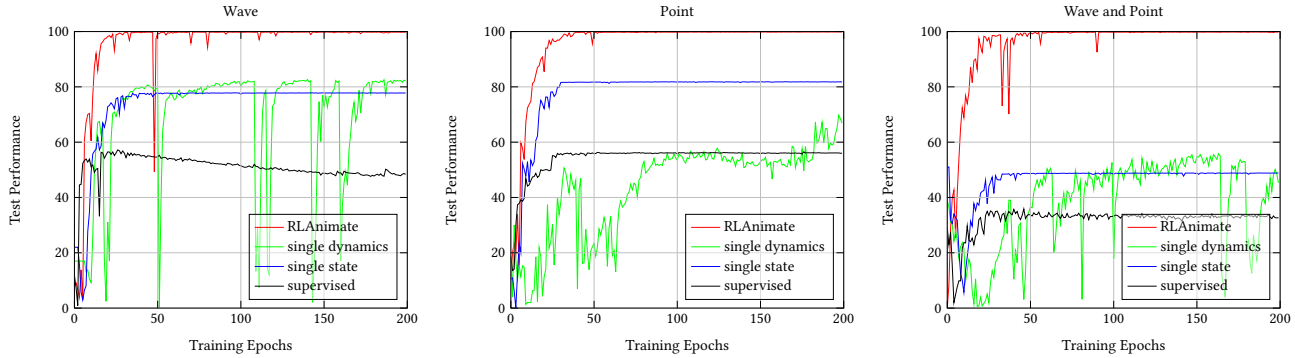
4.1.3 Learning objective. We also evaluated control agents trained with a completely supervised loss calculation. Their performance when imitating the test set was poor, containing a high amount of artefacts. As shown in Figure 4, this agent performs better when portraying behaviours seen during training relative to the unseen testing set motions. This demonstrates that the objective we devise for RLAnimate directly enables it to generate dynamic animation, by optimising for learning optimal animation dynamics via one component and using its understanding of dynamics to optimise it to generate animation portraying ideal behaviours.

4.2 Behaviour Portrayal Flexibility

While RLAnimate enables agents to effectively vary the length of the behaviour portrayal relative to original clips, a key factor is whether these flexible animation outputs contain artefacts. We generated alternate episodes varying the length of the behaviour portrayals relative to the original behaviour, and measured the effect on mean smoothness of the motion trajectory. We found that RLAnimate agents are able to adjust without artefacts motion clip a range of 0.4x to 1.5x times the average length for that class of behaviour, before the artefacts become noticeable and the smoothness



(a) Imitating clips from training set



(b) Imitating clips excluded from training set

Figure 4: Comparision of RLAnimate to other agent and model designs. The performance was measured over training epochs, with agents being trained for 50 epochs for single behaviour modes and 100 for agents learning to portray both pointing and waving. For the single behaviour modes, agents generated a single sample episode, whereas when training for both behaviours, agents generated two sample episodes, imitating a behaviour from each category.

score dips below the 98.5% threshold we identified. The supplementary video contains sequences of behaviours varying the length relative to the original motion length demonstrating this..

5 RELATED WORK

Some of the earliest work applying neural networks for animation trained Convolutional Neural Network (CNN)-based models to portray cyclical behaviours such as human locomotion [12, 13, 27]. However, performance of these models rely heavily on the training set, so they offer limited flexibility and ability to scale to portray a variety of behaviour types. Additionally, given the functions used to leverage the cyclical nature of behaviours resulted in increased computation during training, and final performance of agents required 30 hours.

More recent work has sought to generate animation portraying more dynamic behaviours by using physics simulation in conjunction with model-free reinforcement learning [18, 20, 21]. But results so far have only demonstrated that physics-driven RL methods can be applied to tasks that feature interactions with physical surfaces and objects in the virtual environment. According to the authors, DeepMimic requires between 50 to 140 million sample episodes to be generated to train agents [20]. RLAnimate requires episodes by a factor of about 0.5 million less; we believe this is due to RLAnimate being a model-based approach, requiring fewer sample episodes as an agent learns an effective model for animation dynamics. However, while DeepMimic agents are trained using a model-free RL algorithm, we point to their use of a physics engine at the core of the methodology. The role of the physics engine to provide feedback signals, is that of a model that governs the behaviour being portrayed. In RLAnimate, the latent dynamics model plays the same role, but the dynamics we learn depends on joint position and rotation-derived signals, which are ubiquitously applicable to character animation regardless of the behaviour being portrayed.

When exploring approaches to train agents portraying human-like behaviour, we chose model-based RL as an avenue of exploration due to the potential that would be afforded by learning latent dynamics applicable to multiple behaviours. Current work carried out has shown latent dynamics can be learnt to enable agents to function effectively [7, 8, 17]. We adopted a key conceptual element presented by Haffner et al. in PlaNet where they learnt latent dynamics models and used online planning to select actions [8]. They

Table 3: Final performance and smoothness of output generated imitating test set motions. After examining the output sequences,we identified 98.5% as a threshold for the smoothness score after which artefacts become noticeable.

	Wave		Point		Wave and Point	
Method	**Score**	**Smoothness**	**Score**	**Smoothness**	**Score**	**Smoothness**
RLAnimate	99.7	98.5	99.8	99.4	99.8	99.1
single state	82.1	96.2	66.9	97.7	45.2	97.0
single dynamics space	77.8	97.0	81.7	97.1	48.8	97.1
supervised loss	48.4	96.8	56.0	97.8	32.7	97.3

also introduced a latent dynamics model which learnt both deterministic and stochastic components, allowing agents to robustly learn to make predictions about multiple futures.

But, in our work, we use a latent dynamics model with a further augmentation to learn dynamics as a pair of separate latent states: one component providing agents with representations for the most suitable behaviour to portray, and providing agents with a representation for an estimate of the most human-like portrayal of that behaviour. The former component is entirely deterministic, allowing it to maintain information precisely over the entire span of the behaviour. As the objective of this component is to provide an agent of an understanding of the different between pointing and waving, what each behaviour entails, and how to adjust to different speeds of portrayal, there is no need for stochastic dynamics. In fact, the rigidity by which it is mandated to capture dynamics gleaned from the objective state, enables the latter component to be more effective in learning animation dynamics applicable regardless of behaviour portrayed. We also opt to use a neural-network parameterized output method, as planning would not be efficient given the high dimensional nature of the action space.

6 DISCUSSION

In this paper, we present RLAnimate, an approach for model-based animation control capable of portraying human-like behaviours. We observe that when applying RL for animation, the goal is the quality of the output animation, which needs to portray a finite, predictable range of behaviours, and we formalise a mathematical framework to model animation tasks along that line of thinking. We partition into objectives and descriptions, what typically would be the state space. During training, RLAnimate agents learn a model to self-generate descriptions from the objective signal, and learn an advanced dynamics model to that maintains latent representations that can be used to obtain an animation sequence portraying natural human behaviour.

Our evaluation shows that RLAnimate agents are able to learn to portray different behaviours, using 0.5M x fewer sample episodes generated relative to physics-based model-free RL methods. And to inform the training algorithm of valid behaviour portrayal, we use a limited set of motion data that is significantly smaller in comparison to supervised learning-based methods.

Further, we note the sample-efficiency of the design of the modelling structure and latent dynamics models allowed for; particularly, the effectiveness by which agents could be trained to self-generate description signals of 54 dimensions from an objective signal of 6 dimensions to generate action signals with a dimension size of 45, that make up a human-like animation sequence. Accordingly, we believe approaches that seek to leverage the problem domain to learn more powerful models can be a promising avenue when applying model-based RL to a number of real-world problems and applications.

Research carried out examining human perception of character animation has shown that humans can be sensitive to the even minor differences. In our future work, we plan to ascertain the impact of small imperfections, and explore how agents can be influenced to avoid those pitfalls. We also plan to examine how RLAnimate can be applied to generate animation portraying a wider range of behaviours, in particular more complex portrayal of beat gestures and other complex behaviours required to interact with a user.

REFERENCES

[1] Adobe. 2020. Mixamo. http://www.mixamo.com, Last accessed on 2021-01-30.

[2] EN Barron and H Ishii. 1989. The Bellman equation for minimizing the maximum cost. *Nonlinear Analysis: Theory, Methods & Applications* 13, 9 (1989), 1067–1090.

[3] Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. 2017. Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, International Convention Centre, Sydney, Australia, 834–843. http://proceedings.mlr.press/v70/chou17a.html

[4] Lee J Corrigan, Christopher Peters, Dennis Küster, and Ginevra Castellano. 2016. Engagement perception and generation for social robots and virtual agents. In *Toward Robotic Socially Believable Behaving Systems-Volume I*. Springer, 29–51.

[5] Vihanga Gamage and Cathy Ennis. 2018. Examining the Effects of a Virtual Character on Learning and Engagement in Serious Games. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (Limassol, Cyprus) *(MIG '18)*. Association for Computing Machinery, New York, NY, USA, Article 20, 9 pages. https://doi.org/10.1145/3274247.3274499

[6] Mike Goslin and Mark R Mine. 2004. The Panda3D graphics engine. *Computer* 37, 10 (2004), 112–114.

[7] David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122* (2018).

[8] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*. PMLR, 2555–2565.

[9] Mikael Henaff, William F Whitney, and Yann LeCun. 2017. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177* (2017).

[10] John R Hershey and Peder A Olsen. 2007. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, Vol. 4. IEEE, IV–317.

[11] Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 42.

[12] Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

[13] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*. 1–4.

[14] Peter J Huber. 1992. Robust estimation of a location parameter. In *Breakthroughs in statistics*. Springer, 492–518.

[15] Alex Klein, Zerrin Yumak, Arjen Beij, and A. Frank van der Stappen. 2019. Data-Driven Gaze Animation Using Recurrent Neural Networks. In *Proceedings of the 12th Annual International Conference on Motion, Interaction, and Games* (Newcastle upon Tyne, United Kingdom) *(MIG '19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3359566.3360054

[16] Jorgen Kornfeld, Michal Januszewski, Michale S Fee, Philipp Schubert, Viren Jain, and Winfried Denk. 2020. An anatomical substrate of credit assignment in reinforcement learning. (2020).

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[18] Libin Liu and Jessica Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 142.

[19] Rick Parent. 2012. *Computer animation: algorithms and techniques*. Newnes.

[20] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

[21] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

[22] William H Press and Saul A Teukolsky. 1990. Savitzky-Golay smoothing filters. *Computers in Physics* 4, 6 (1990), 669–672.

[23] Dominik Schillinger, Luca Dedè, Michael A. Scott, John A. Evans, Michael J. Borden, Ernst Rank, and Thomas J.R. Hughes. 2012. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Computer Methods in Applied Mechanics and Engineering* 249-252 (2012), 116–150. https://doi.org/10.1016/j.cma.2012.03.017

[24] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. 2020. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*. PMLR, 8583–8592.

[25] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690* (2018).

[26] Théophane Weber, Nicolas Heess, Lars Buesing, and David Silver. 2019. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2650–2660.

[27] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.