OstrichRL: A Musculoskeletal Ostrich Simulation to Study Bio-mechanical Locomotion

Vittorio La Barbera[*1] **Fabio Pardo**[*2] **Yuval Tassa**[3] **Monica A. Daley**[4]
Christopher T. Richards[1] **Petar Kormushev**[2] **John R. Hutchinson**[1]

[1]Royal Veterinary College, London, [2]Imperial College London
[3]DeepMind, London, [4]University of California, Irvine

`{vlabarbera, jhutchinson, ctrichards}@rvc.ac.uk,`
`{f.pardo, p.kormushev}@imperial.ac.uk,`
`tassa@deepmind.com, madaley@uci.edu`

Abstract

Muscle-actuated control is a research topic of interest spanning different fields, in particular biomechanics, robotics and graphics. This type of control is particularly challenging because models are often overactuated, and dynamics are delayed and non-linear. It is however a very well tested and tuned actuation model that has undergone millions of years of evolution and that involves interesting properties exploiting passive forces of muscle-tendon units and efficient energy storage and release. To facilitate research on muscle-actuated simulation, we release a 3D musculoskeletal simulation of an ostrich based on the MuJoCo simulator. Ostriches are one of the fastest bipeds on earth and are therefore an excellent model for studying muscle-actuated bipedal locomotion. The model is based on CT scans and dissections used to gather actual muscle data such as insertion sites, lengths and pennation angles. Along with this model, we also provide a set of reinforcement learning tasks, including reference motion tracking and a reaching task with the neck. The reference motion data are based on motion capture clips of various behaviors which we pre-processed and adapted to our model. This paper describes how the model was built and iteratively improved using the tasks. We evaluate the accuracy of the muscle actuation patterns by comparing them to experimentally collected electromyographic data from locomoting birds. We believe that this work can be a useful bridge between the biomechanics, reinforcement learning, graphics and robotics communities, by providing a fast and easy to use simulation.

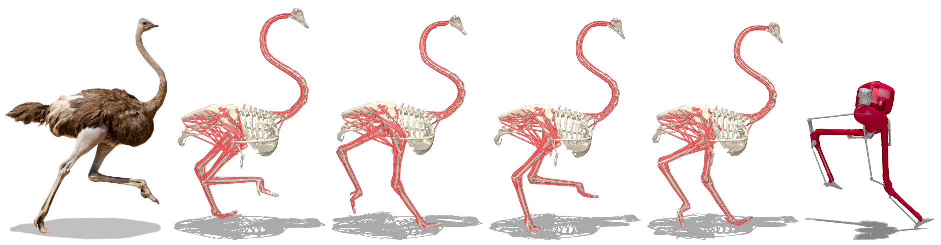

Figure 1: The models, tasks and data are available at `https://github.com/vittorione94/ostrichrl` and visualizations at `https://sites.google.com/view/ostrichrl`.

[*]Equal contribution.

1 Introduction and related work

Actuation means generating forces in order to create movement. In nature, movement in vertebrates is produced by contracting muscles which pull on skeletal bones, creating torques at the joints. Similarly, robots use motors, but motors and muscles have very different properties. A review of several different actuation methods is given by Peng & van de Panne (2017).

Understanding muscles is of interest in many different fields. From a biological point of view, we would like to understand how animals solve complex locomotor tasks. Muscles are also relevant in computer graphics to obtain accurate skin deformations in virtual characters and to obtain more natural-looking gaits for films and video games (Angles et al., 2019; Abdrashitov et al., 2021; Modi et al., 2021). In robotics, with the exception of soft robots, muscles have been studied relatively little despite their interesting energetic properties and the sophisticated movements they enable (Wang et al., 2021; Cotton et al., 2012). In sports medicine musculoskeletal simulations are used to understand sports injuries (Bulat et al., 2019).

Biomechanics is the study of how biological systems produce motion. Biological motion is affected by many factors such as the musculoskeletal geometry, internal states of the muscles, and force-length-velocity curves (discussed below). Understanding how complex biological systems such as humans and other animals use their muscles to move is quite challenging. Obtaining data from living animals is difficult and not neutral to animal welfare. To overcome this problem, biomechanics researchers often use computer-based simulations combined with numerical optimization to estimate how muscles are used.

However, most of the research that uses optimization techniques does not leverage the new available tools from machine learning, in particular those based on the reinforcement learning (RL) framework (Sutton & Barto, 2018). These tools have proven to be a valuable alternative to traditional optimization techniques to solve complex tasks in continuous and high-dimensional state and action spaces (Lillicrap et al., 2015; Fujimoto et al., 2018). The NeurIPS conference has been hosting recurrent competitions to bridge this gap [2], where the goal was to make a musculoskeletal human model walk. The challenge used the OpenSim (Delp et al., 2007; Seth et al., 2018) simulator. However, as documented in many of the solutions (Pavlov et al., 2018; Zhou et al., 2019; Akimov, 2019; Kolesnikov & Khrulkov, 2020), the engine was too slow to properly use traditional deep RL methods, which often require millions or billions of samples to find good solutions. Moreover, the model used in the latest challenge was in many ways quite simplistic, actuating only the legs with 22 muscles in total and removing the arms.

Some innovative research has used deep RL to solve complex locomotion tasks such as playing basketball (Liu & Hodgins, 2018), performing athletic jumps (Yin et al., 2021), boxing and fencing (Won et al., 2021). These studies used motion capture data for a rich reward signal during training (Merel et al., 2017; Hasenclever et al., 2020; Merel et al., 2018).

Wang et al. (2012) was one of the first studies in the graphics community to use biological actuators. Geijtenbeek et al. (2013) used evolution-based algorithms to control muscular bipeds and also optimized muscle routing. Jiang et al. (2019) tried to bridge torque-actuated models and muscle dynamics, using neural networks to map muscle activations from torque commands and achieved natural looking gaits using torque actuators. Lee et al. (2018) used volumetric muscles and trajectory optimization for juggling. Lee et al. (2019) combined RL and motion capture clips to control a human musculoskeletal model solving a variety of tasks from walking to weight lifting. Simulating musculoskeletal animal models is not common outside the field of biomechanics. Most of the efforts are focused on humans because of available resources in anatomical atlases and interest from the entertainment industry and sports. Available full-body animal musculoskeletal models include, for example the chimpanzee (Sellers et al., 2013) and dog (Stark et al., 2021). Moreover, for human models, researchers tend to have access to much higher quality motion capture data than for animals, but animal research often have superior empirical measurements of physiological function.

Here we present a new ostrich musculoskeletal model that uses a fast physics engine with reinforcement learning tasks. To the best of our knowledge, we are the first to apply RL to such a realistic

[2] `https://www.crowdai.org/challenges/nips-2017-learning-to-run`
`https://www.crowdai.org/challenges/nips-2018-ai-for-prosthetics-challenge`
`https://www.aicrowd.com/challenges/neurips-2019-learning-to-move-walk-around`

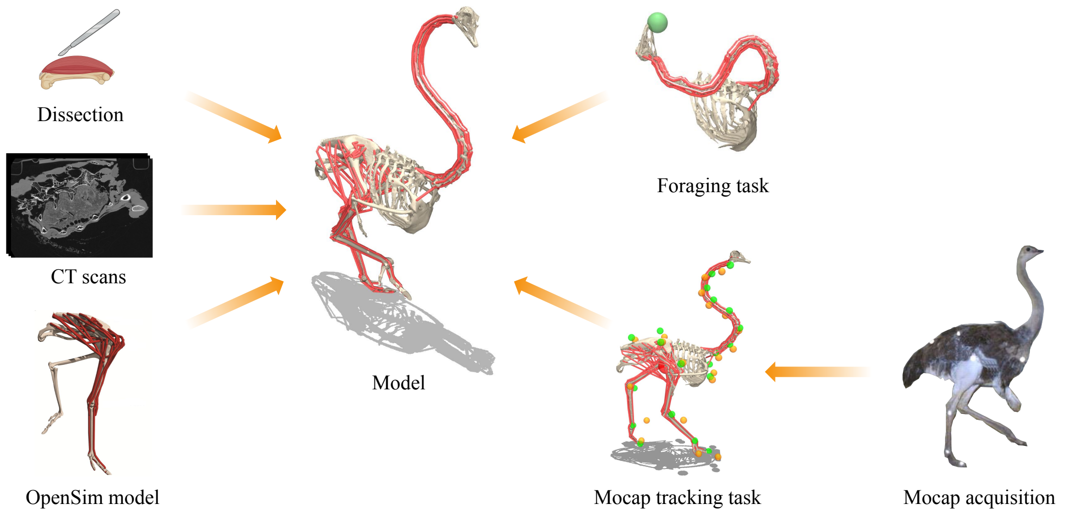

Figure 2: The workflow used to build the ostrich model. Various sources of data were blended together for the modeling part, and tasks helped in fine-tuning muscle strength and joint limits.

animal model, solving locomotion tasks. The main differences from prior human simulations are obviously the anatomy of the model but also the RL architecture; e.g. Lee et al. (2019) separated muscle coordination from trajectory mimicking through two different networks with privileged information and an intermediate PD controller. Ostriches are interesting to study because they are fast, economical bipedal runners (Alexander et al., 1979). With our new computational tools, which we release along with this paper, we hope to open new opportunities for researchers interested in accurate and fast muscle simulation provided by MuJoCo combined with RL.

First, we describe how we built the model by assembling various types of data and how we used MuJoCo's muscle model. Then, we detail the various tasks that we designed. The results from one of those tasks in particular are then used to evaluate the accuracy of the muscle excitation patterns by comparing against empirical electromyography data. Finally, we briefly explain how the similarity between the Cassie robot and ostriches allowed us to adapt our motion capture tracking pipeline to a simulated model of Cassie.

2 Model

2.1 Anatomical data acquisition

Building an anatomically accurate 3D musculoskeletal model required gathering data from real ostriches. In this subsection, we discuss how we collected the anatomical data necessary for our model.

Computed tomography scanning We first acquired multiple Computed tomography (CT) scans of a 96.5kg adult male ostrich specimen, donated by a local farm. The CT scans were then segmented (separating the bone from the rest of the x-ray slices) in Mimics (Materialise, Inc.; Leuven, Belgium) software to obtain 3D models (polygonal meshes) for each bone from the ribcage to the head, including the wings. These 3D geometries were important for two reasons: firstly they provided an accurate representation of the geometry of the bones, vital for defining muscle attachment points; and secondly they allowed inference of the inertia of body segments. To obtain the inertia of body segments we followed the steps provided in Hutchinson et al. (2007); the idea is to wrap the skeleton parts within meshes representing flesh and assume constant density (usually water density is used), from there, volumes, masses and the inertia tensors can be computed.

Muscle dissection Next, we performed a dissection in order to gather data for muscles of the neck and rib cage for our final model, this data is needed to correctly model muscles peak forces and lengths. Wings are not actuated in our model because air friction is not considered in our simulations and

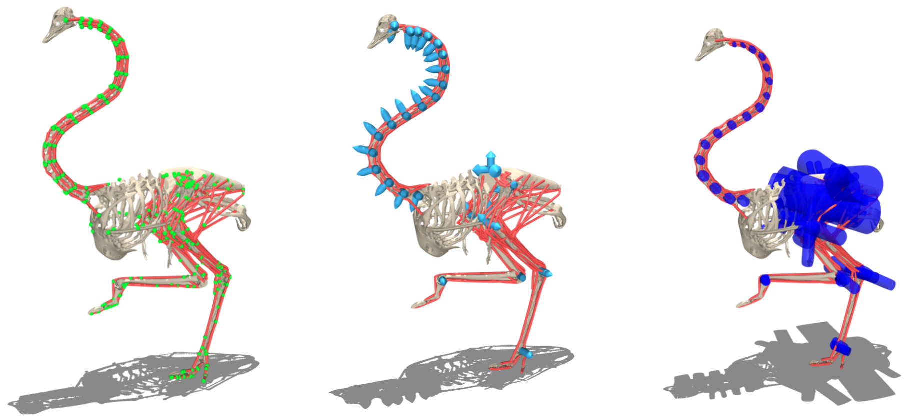
Figure 3: The skeleton of the model follows a tree structure of bodies and joints. The muscle routing is defined using way points, shown in green on the left. The joints are shown in light blue in the middle. The wrapping geometries are shown in blue on the right (some wrapping geometries have been omitted for clarity).

wings are mostly used when turning at high velocity. Dissection is important to properly understand muscle routing, where the muscles start (origin) and where they end (insertion). During the dissection, we used anatomical descriptions from Böhmer et al. (2019); Tsuihiji (2005, 2007) to identify the different muscles. We also acquired muscle-specific data: muscle mass, pennation angle, tendon length and muscle fiber lengths, used when simulating the muscle-tendon dynamics, as explained below.

2.2 Physics simulators

Here we discuss two computer modelling and simulation software packages commonly used by the biomechanics and reinforcement learning communities. The first one is OpenSim (Delp et al., 2007; Seth et al., 2018), an open-sourced engine that provides benchmarked physics-based muscle simulations. It is relevant to our study not only for its popularity but also because the legs of our ostrich model were originally modelled with this engine (Hutchinson et al., 2015), as we can see in figure 2. The same model was also used to run some tracking (inverse) simulations to estimate the forces, activations and mechanical work of the muscles involved in solving locomotion tasks (Rankin et al., 2016). OpenSim is quite slow in simulating each time step, as documented in previous competitions in NeurIPS that used it to simulate muscle dynamics (Kolesnikov & Khrulkov, 2020). The second simulator, MuJoCo (Todorov et al., 2012) is an open-source engine that is used in multiple RL domains such as dm_control (Tassa et al., 2020) and Gym (Brockman et al., 2016) because it provides fast and accurate rigid body dynamics. Specifically, tendon routing in OpenSim uses an iterative algorithm, while MuJoCo's tendon routing is closed-form, and therefore much faster. In a comparison made by Ikkala & Hämäläinen (2020), when calculating the average run time over 97 forward simulations MuJoCo is roughly 600 times faster than OpenSim.

Both software packages define their models in a hierarchical tree structure. Muscles attach to at least two bones, using two sites at the ends, respectively called the origin and the insertion. The muscle geometry can be more elaborate than just the origin and insertion sites; other sites called way points and wrapping geometries can also be specified. Way points are sites the muscle must pass through, which are useful to maintain an anatomically realistic 3D path for the muscles. Wrapping geometries are useful to prevent muscles from penetrating the bones, and are geometric primitives such as spheres or cylinders.

After converting the OpenSim leg model to MuJoCo's format, we completed the ostrich model by adding the ribcage, wings, neck and head geometries. We then actuated the neck by adding the muscles, matching the routing from the dissection; we also changed the feet morphology from being completely flat to match a more realistic pose. In our final model, each leg contains 7 hinge joints: 3 for the hip, 2 for the knee, 1 for the ankle, and 1 for the metatarsophalangeal (mtp), while the neck

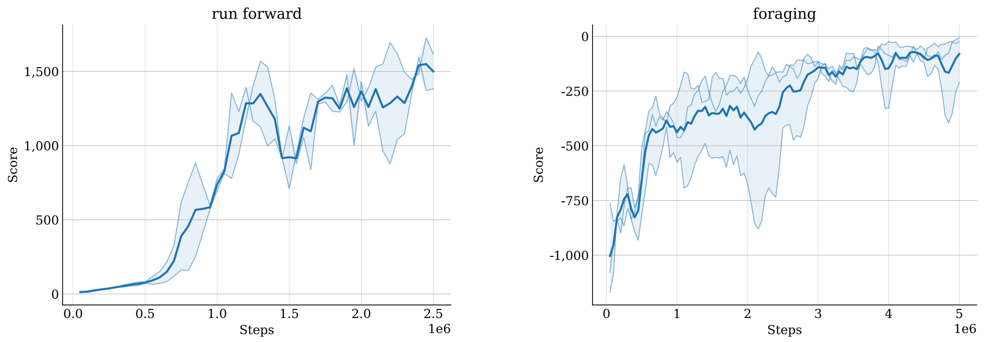

Figure 4: Score on the run forward and foraging tasks. Values are averaged over 5 seeds and smoothed with a sliding window of size 3.

contains 34 hinge joints; 2 for each pair of vertebrae, as can be seen in the middle image in figure 3. These 48 joints are actuated by 121 muscles, making this model very challenging to control. In comparison, the OpenSim model used for the NeurIPS 2019: Learn to Move - Walk Around challenge had only 22 muscles.

The muscle dynamics are explained in detail in the Appendix A. After converting the OpenSim model to MuJoCo, we found that fine tuning the muscle lengths was necessary to improve stability. To find the new muscle ranges, we randomized the model joint angles in their limits a large number of times and recorded the global minimum and maximum lengths for each muscle.

3 Tasks

The musculoskeletal model described above has been implemented using the MuJoCo physics engine (Todorov et al., 2012) which is used in a number of popular control domains for reinforcement learning research, in particular dm_control (Tassa et al., 2020) and OpenAI Gym (Brockman et al., 2016). Most of these domains use simplistic models, often inspired by animal morphologies, made up of basic geometric bodies such as capsules, and torque-controlled multiaxial joints. While such models are fast and fairly easy to control, they do not provide a realistic basis for studying animal movement. On the other hand, simulations used in biomechanics typically model animals more accurately (Sellers et al., 2013; Hutchinson et al., 2015; Stark et al., 2021) but are slow and often used in conjunction with relatively simple control techniques from the trajectory optimisation repertoire (e.g., direct collocation while minimising the sum of squared muscle activations).

While the previous sections described the proposed musculoskeletal ostrich model, this section describes a set of tasks implemented with dm_control (Tassa et al., 2020) that we used repeatedly throughout this work to test the model. These tasks could be used by other researchers for their own research. For example, new biomechanical data can be produced from the simulated behaviours, while RL algorithms can be tested on this unique and challenging set of environments.

We focused on two types of tasks in particular: locomotion and neck control. We found these to be particularly useful when designing the model because they used the two main groups of muscles present in the model. To obtain the policies we used the TD4 agent, a mixture of TD3 (Fujimoto et al., 2018), with its pair of critics, delayed actor training and target action noise, and D4PG (Barth-Maron et al., 2018), with its distributional value function (Bellemare et al., 2017) and n-step returns. We also found that Ornstein Uhlenbeck exploration noise (Lillicrap et al., 2015) was significantly better than Gaussian noise, probably due to the importance of correlated excitation when controlling muscles. We chose to use the Tonic deep RL library (Pardo, 2020), because it provided us with this state of the art agent and the flexibility we needed to quickly try, evaluate and visualize experiments while allowing custom dm_control tasks to be used. Details about the training hyperparameters and the experiments can be found in the Appendix and the policies obtained with the different tasks can be visualized on the website [3].

[3]https://sites.google.com/view/ostrichrl

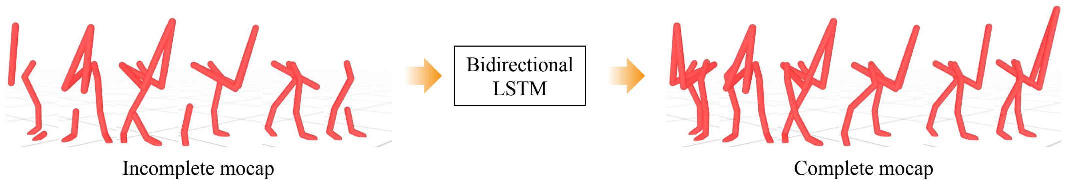

Figure 5: Visualization of the mocap prediction process. Left: the original incomplete data. Right: the complete data after predicting the missing markers with a bidirectional LSTM. Segments are drawn between pairs of existing markers.

3.1 Run forward

The first task we experimented with had a simple "move forward" objective. It was constrained to a vertical planar space, used torque-actuated joints, and the neck was kept rigidly attached to the thorax of the model. The goal was to ensure that the underlying model and simulation structure was working as expected before adding muscles. The obtained gait was similar to the one obtained when training on the 2D walker tasks from dm_control and Gym.

After adding muscles to the legs, we quickly found that the torque-actuated planar agent was not able to find satisfactory gaits anymore. Even after trying a large number of hyperparameter values, we noticed that the same gait was repeatedly emerging: the toes were completely (unrealistically) plantarflexed and the ankles remained straight in an unnatural way. This gait was deemed unsatisfactory because it was not optimal in terms of maximum possible speed but also because was obviously unnatural.

3.2 Motion capture tracking

A typical solution to better constrain the space of policies is to engineer a more sophisticated reward function. Reference motion tracking is a particularly well-suited option that has been used with motion capture data in numerous studies to produce natural-looking movements (Liu et al., 2010; Peng et al., 2018; Chentanez et al., 2018; Bergamin et al., 2019; Merel et al., 2018; Peng et al., 2019; Hasenclever et al., 2020). However, producing motion capture data of ostriches performing various movements was outside of the scope of this paper. Instead, as a proof of concept we started with reference motion data that we labelled by hand, frame by frame, using videos of running ostriches found online. The task was still planar and the quality of the results produced encouraged us to continue further in this direction.

3.2.1 Motion capture data

To move the simulated ostrich to a 3D space and to use more diverse behaviours, we started searching for proper motion capture data. We did not find any open source database but after contacting the authors of Jindrich et al. (2007), we obtained the data originally used to study joint kinematics of ostriches performing cutting maneuvers. The provided dataset was composed of 82 clips, recorded at 240 Hz, containing the 3D coordinates of 14 markers distributed as follows: 2 on the breasts, 1 on the spine, 2 on the hips, 2 on the knees, 2 on the ankles, 2 on the mtps, 2 on the toes and 1 on the head.

We selected the largest part of the clips where at least 10 markers were simultaneously present for at least 1 second, typically disregarding the beginning and the end of the clips when the ostrich was not in the field of view of the cameras. This selection left 35 clips. However, even in those clips, we found that a large number of markers were periodically missing, probably due to feathers and limbs occluding them from the cameras. To overcome this issue, we used a bidirectional LSTM trained in a similar way as a denoising autoencoder. With a probability 0.1, we randomly masked some markers and asked the model to predict their value given the context of a segment of 100 steps from the same clip.We then used this model to predict the actual missing markers. Since the model was trained and then used on the same, relatively small dataset, we anticipated some overfitting issues, but playing the clips with the predicted markers in place of the missing ones gave surprisingly good results (see Figure 5). The denoising is discussed in more details in the Appendix D.

After rescaling the marker coordinates to match the size of our model, the next step was to transform the set of marker coordinates to joint positions. Fortunately, enough markers were present over the different parts of the ostrich body to limit the space of solutions, except for the neck where only one marker was present on the head. We used gradient descent with the mean Euclidean distance between the reference markers and the ones produced by joint poses on two sets of parameters simultaneously. The first set described where to attach the markers on the body parts and is shared across all the clips. This was needed because we did not have the exact location of the markers used during the motion capture acquisition and how they would translate to locations with respect to the bones. The second set of parameters described the joint values at every keyframe. To reduce the space of solutions for the neck shape, we added a regularisation term, encouraging the neck to follow an S shape. We first tried to use a finite difference approximation of the gradient but found this approach to be too slow to be practical. We then decided to implement the kinematics function in a differentiable way, mapping joint values to body locations to create marker locations. This function was implemented using differentiable operations from the TensorFlow library (Abadi et al., 2016). We found this approach to be particularly fast thanks to the possibility to simultaneously optimise for all the steps across all the clips using the batch dimension.

3.2.2 Single clip tracking

During training, an initial time-step is uniformly sampled, the model is initialised in the pose corresponding to this step and a small amount of Gaussian noise is added to help diversify experiences. Since the control frequency used in our experiments was 40 Hz, the task used every sixth data points of motion capture data (240 Hz) during tracking. Also, to allow the policy to perform time-dependent behaviours and to deal with finite horizons, we provided the remaining number of steps in the clip to the observations following Pardo et al. (2018). Every clip was tracked by a different policy but using more mocap data could allow the creation of a reusable controller (Merel et al., 2018; Peng et al., 2019; Hasenclever et al., 2020). After tuning a number of task and agent hyperparameters, we managed to get very satisfactory results in most clips.

The tracking reward is defined as a product of the quantity $\exp\left(-w_p e_p - w_r e_r\right)$ measuring how much a body part's location and orientation match the reference. Given a body part with center of mass coordinates p and orientation matrix of inertia R, and the corresponding references $\bar{p}$ and $\bar{R}$, we use the Euclidean distance $e_p = \|\bar{p} - p\|$ and the angle of the difference rotation $e_r = \arccos\left((\mathrm{tr}(\bar{R}R^T) - 1)/2\right)$. The weights w_p and w_r control the wideness of the Gaussians, accounting for the different magnitudes and importance. The reward is bounded in (0, 1] and the multiplicative nature ensures that if one of the component is too far from the reference the entire reward decays to 0.

3.2.3 Cyclic running clip tracking

Considering that the unconstrained running task did not give satisfactory results, we decided to recreate a task with similar conditions but using mocap tracking. We therefore created a completely cyclic mocap clip from the middle section of one of the clips, that we repeated several times and smoothed to remove discrepancies at the boundaries. Similarly to the results obtained with the previous task, we found the tracking to be very satisfactory. A great advantage of this task is that it helped us produce simulated biomechanical data for a cyclic gait that could be directly compared to muscle excitations and lengths measured on living birds (see Section 4).

3.3 Neck control

The neck is composed of 17 vertebrae, allowing a large range of shapes to be produced. To reach a head position and orientation, an infinite number of solutions exist and when running, the weight and pose of the neck should influence balance and vision to a great extent. Controlling the neck with muscles that cross multiple vertebrae requires a very sophisticated policy.

To tune the neck part of the model, we found that full-body mocap tracking was not ideal due to the large number of moving pieces and the fact that the reference poses of the neck were artificial. We therefore created another task whose objective was to reach random targets with the beak. To sample feasible target locations, we used rejection sampling. Points were repeatedly sampled in a sphere centered at the base of the neck with a radius slightly smaller than the length of the neck, and rejected when inside a second smaller sphere roughly representing the torso of the ostrich. For this task, only the ribcage, the neck and the head were used and the ribcage was kept fixed in space. While efficient

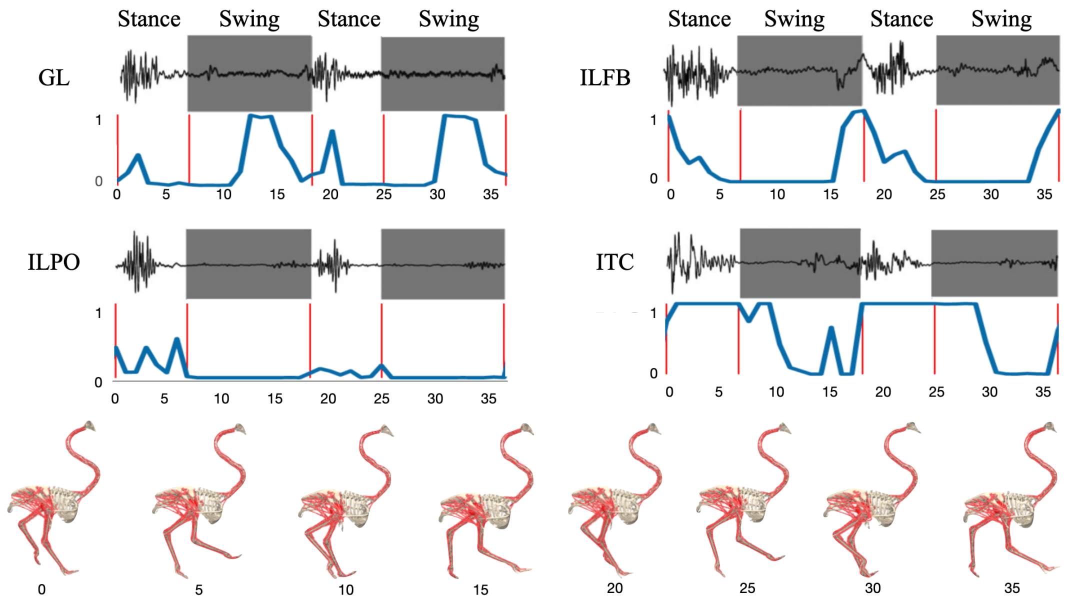

Figure 6: Comparison with experimental EMG data from emus. The first red vertical line indicates that the right foot was in contact with the ground (start of stance phase); the second one is when the foot leaves the ground (start of swing phase). EMG data from Cuff et al. (2019) are on top. Muscle excitations (ours) are below.

policies were easily obtained, initially neck shapes were very irregular, with many unnatural, abrupt changes. To reduce this phenomenon, we increased the stiffness of the neck joints until satisfactory smoothness was obtained. For the joint limits, we initially used values from Dzemski & Christian (2007) but then realized that the limits had to be increased to allow tighter neck curves which ostriches can perform.

4 Electromyography comparison

To demonstrate the realism of the proposed simulation, we chose to compare the muscle excitations outputted by a policy trained on the cyclic mocap tracking task, to Electromyography (EMG) data. One one side, EMG measures electrical activity in muscles, in response to a stimulation from motoneurons and on the other side, raw actions produced by a policy are used by MuJoCo to produce muscle activations A.

Collecting EMG data in animals is challenging, because their skin might be too thick for electrodes and surgery might be needed to implant them directly into the muscles. Since, no EMG data of ostriches have been collected yet, we decided to perform comparisons with data from emus, close relatives of ostriches. The EMG data were originally acquired by Cuff et al. (2019) to study muscle recruitment patterns across different avian species in order to understand neuromuscular control from an evolutionary point of view. Their results showed that walking/running birds generally use their hindlimb muscles in very similar ways, so emu EMG data are justifiable for comparisons with ostrich simulations.

To compare the data, following standard conventions in locomotion research, we divide the gait into two phases: stance (the foot is on the ground) and swing (the foot is off the ground). Once we divide the gait into these phases, we scale the timing accordingly between the two studies (e.g., the two birds might be moving at different speeds).

As show in the figure 6, we find broadly similar excitation patterns, except for the GL (Gastrocnemius Lateralis) which had an extra burst of swing phase excitation in these simulations; not usually found in any extant birds (Cuff et al., 2019). The ITC (Iliotrochantericus Caudalis) also had simulated excitation throughout more of the swing phase than in the EMG data; reminiscent of findings for simulated ostriches using the OpenSim model in Rankin et al. (2016). There are many more muscles

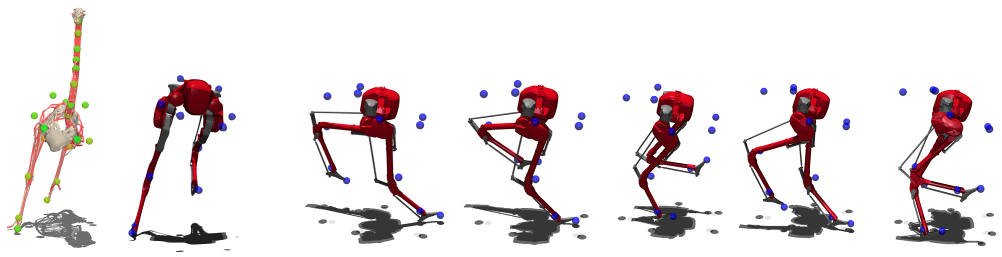

Figure 7: Ostrich and Cassie markers comparison, and poses from a motion capture clip.

than actual degrees of freedom in the legs, so there are many possible solutions (excitation patterns) that are able to match the kinematic data. Moreover, we do not penalize actions in our reward signal, so it is expected to see more excitations than in experimental data.

5 Compatibility with Cassie

Cassie is a bipedal robot that was produced by Agility Robotics and extensive work has been done on it using trajectory optimization (Reher et al., 2019; Li et al., 2020; Apgar et al., 2018) and reinforcement learning (Xie et al., 2018, 2020; Li et al., 2021). Its morphology is very similar to that of ostrich legs with short thighs, however, since Cassie's joints are actuated by electric motors, they are each limited to one degree of freedom. To create multiple degrees of freedom, multiple joints have to be added in series and an offset is required between them for mechanical reasons. This is a major difference between ball joints allowed by musculoskeletal bodies and traditional robots. Furthermore, Cassie couples the knee and ankle joints together. However, the similarity with the ostrich morphology seemed sufficient to try applying the mocap tracking to this robot. This provides an interesting opportunity to compare muscle and torque actuation with two different models of real world bodies with roughly similar morphologies.

We took the MuJoCo model of Cassie, provided by the Oregon State University Dynamic Robotics Laboratory [4]. The model had to be tuned to increase its stability and some constraints to the model had to be added to ensure that parts were correctly attached. These constraints were equality constraints to ensure that the rods on the back of the legs are always in contact. To satisfy these constraints we used MuJoCo's Jacobian function when performing the gradient descent (Buss, 2004). We then used the previously described mocap generation pipeline to obtain robot poses that corresponded to the mocap clips, excluding the neck. Surprisingly, the morphology of Cassie is sufficiently close to the one of an ostrich to allow the clips to be tracked accurately. This provides a set of interesting behaviors feasible with the Cassie robot, including more natural gaits than the one usually found in the literature and demos using this robot.

6 Conclusion

We presented a novel musculoskeletal ostrich model that we hope will be useful to researchers in the fields of reinforcement learning, neuroscience, biomechanics and computer graphics. There are still some future questions we can answer using our ostrich model. Firstly, we did not explore mechanical or metabolic efficiency. Animals including humans are very power-efficient when solving certain tasks, yet in our reward signal we did not have a penalty term for muscle usage and it is unclear how to insert it into the equation. The best approach might be to model directly fatigue (Potvin & Fuglevand, 2017) in the muscle simulation so that through time the muscle force output will decrease. Furthermore, obtaining satisfactory results without using motion capture would be desirable to obtain movement predictions produced in a broader set of situations such as terrain adaptation.

[4] `https://github.com/osudrl/cassie-mujoco-sim`

Acknowledgments and Disclosure of Funding

The research presented in this paper has been supported by the Royal Veterinary College and Dyson Technology Ltd.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Rinat Abdrashitov, Seungbae Bang, David IW Levin, Karan Singh, and Alec Jacobson. Interactive modelling of volumetric musculoskeletal anatomy. *arXiv preprint arXiv:2106.05161*, 2021.

Dmitry Akimov. Distributed soft actor-critic with multivariate reward representation and knowledge distillation. *arXiv preprint arXiv:1911.13056*, 2019.

R McN Alexander, GMO Maloiy, R Njau, and AS Jayes. Mechanics of running of the ostrich (struthio camelus). *Journal of Zoology*, 187(2):169–178, 1979.

Baptiste Angles, Daniel Rebain, Miles Macklin, Brian Wyvill, Loic Barthe, JP Lewis, Javier Von Der Pahlen, Shahram Izadi, Julien Valentin, Sofien Bouaziz, et al. Viper: Volume invariant position-based elastic rods. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2 (2):1–26, 2019.

Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan W Hurst. Fast online trajectory optimization for the bipedal robot cassie. In *Robotics: Science and Systems*, volume 101, pp. 14, 2018.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458. PMLR, 2017.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38(6): 1–11, 2019.

C Böhmer, J Prevoteau, O Duriez, and A Abourachid. Gulper, ripper and scrapper: anatomy of the neck in three species of vultures. *Journal of Anatomy*, 236(4):701–723, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Muge Bulat, Nuray Korkmaz Can, Yunus Ziya Arslan, and Walter Herzog. Musculoskeletal simulation tools for understanding mechanisms of lower-limb sports injuries. *Current Sports Medicine Reports*, 18(6):210–216, 2019.

Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004.

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th annual international conference on motion, interaction, and games*, pp. 1–10, 2018.

Sebastien Cotton, Ionut Mihai Constantin Olaru, Matthew Bellman, Tim van der Ven, Johnny Godowski, and Jerry Pratt. Fastrunner: A fast, efficient and robust bipedal robot. concept and planar simulation. In *2012 IEEE International Conference on Robotics and Automation*, pp. 2358–2364. IEEE, 2012.

Andrew R Cuff, Monica A Daley, Krijn B Michel, Vivian R Allen, Luis Pardon Lamas, Chiara Adami, Paolo Monticelli, Ludo Pelligand, and John R Hutchinson. Relating neuromuscular control to functional anatomy of limb muscles in extant archosaurs. *Journal of Morphology*, 280(5):666–680, 2019.

Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950, 2007.

Gordon Dzemski and Andreas Christian. Flexibility along the neck of the ostrich (struthio camelus) and consequences for the reconstruction of dinosaurs with extreme neck length. *Journal of Morphology*, 268(8):701–714, 2007.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Thomas Geijtenbeek, Michiel Van De Panne, and A Frank Van Der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32(6):1–11, 2013.

Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. Comic: Complementary task learning & mimicry for reusable skills. In *International Conference on Machine Learning*, pp. 4105–4115. PMLR, 2020.

John R Hutchinson, Victor Ng-Thow-Hing, and Frank C Anderson. A 3d interactive method for estimating body segmental parameters in animals: application to the turning and running performance of tyrannosaurus rex. *Journal of theoretical biology*, 246(4):660–680, 2007.

John R Hutchinson, Jeffery W Rankin, Jonas Rubenson, Kate H Rosenbluth, Robert A Siston, and Scott L Delp. Musculoskeletal modelling of an ostrich (struthio camelus) pelvic limb: influence of limb orientation on muscular capacity during locomotion. *PeerJ*, 3:e1001, 2015.

Aleksi Ikkala and Perttu Hämäläinen. Converting biomechanical models from opensim to mujoco. *arXiv preprint arXiv:2006.10618*, 2020.

Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. Synthesis of biologically realistic human motion using joint torque actuation. *ACM Transactions On Graphics (TOG)*, 38(4):1–12, 2019.

Devin L Jindrich, Nicola C Smith, Karin Jespers, and Alan M Wilson. Mechanics of cutting maneuvers by ostriches (struthio camelus). *Journal of Experimental Biology*, 210(8):1378–1390, 2007.

Sergey Kolesnikov and Valentin Khrulkov. Sample efficient ensemble learning with catalyst. rl. *arXiv preprint arXiv:2003.14210*, 2020.

Seunghwan Lee, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. Dexterous manipulation and control with volumetric muscles. *ACM Transactions on Graphics (TOG)*, 37 (4):1–13, 2018.

Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38(4):1–13, 2019.

Zhongyu Li, Christine Cummings, and Koushil Sreenath. Animated cassie: A dynamic relatable robotic character. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3739–3746. IEEE, 2020.

Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. *arXiv preprint arXiv:2103.14295*, 2021.

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

Libin Liu and Jessica Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*, pp. 1–10. 2010.

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018.

Matthew Millard, Thomas Uchida, Ajay Seth, and Scott L Delp. Flexing computational muscle: modeling and simulation of musculotendon dynamics. *Journal of biomechanical engineering*, 135 (2), 2013.

Vismay Modi, Lawson Fulton, Alec Jacobson, Shinjiro Sueda, and David IW Levin. Emu: Efficient muscle simulation in deformation space. In *Computer Graphics Forum*, volume 40, pp. 234–248. Wiley Online Library, 2021.

Fabio Pardo. Tonic: A deep reinforcement learning library for fast prototyping and benchmarking. *arXiv preprint arXiv:2011.07537*, 2020.

Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*, pp. 4045–4054. PMLR, 2018.

Mikhail Pavlov, Sergey Kolesnikov, and Sergey Plis. Run, skeleton, run: skeletal model in a physics-based simulation. In *2018 AAAI Spring Symposium Series*, 2018.

Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 1–13, 2017.

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

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019.

Jim R Potvin and Andrew J Fuglevand. A motor unit-based model of muscle fatigue. *PLoS computational biology*, 13(6):e1005581, 2017.

Jeffery W Rankin, Jonas Rubenson, and John R Hutchinson. Inferring muscle functional roles of the ostrich pelvic limb during walking and running using computer optimization. *Journal of the Royal Society Interface*, 13(118):20160035, 2016.

Jacob Reher, Wen-Loong Ma, and Aaron D Ames. Dynamic walking with compliance on a cassie bipedal robot. In *2019 18th European Control Conference (ECC)*, pp. 2589–2595. IEEE, 2019.

WI Sellers, L Margetts, KT Bates, and AT Chamberlain. Exploring diagonal gait using a forward dynamic three-dimensional chimpanzee simulation. *Folia Primatologica*, 84(3-5):180–200, 2013.

Ajay Seth, Jennifer L Hicks, Thomas K Uchida, Ayman Habib, Christopher L Dembia, James J Dunne, Carmichael F Ong, Matthew S DeMers, Apoorva Rajagopal, Matthew Millard, et al. Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS computational biology*, 14(7):e1006223, 2018.

Heiko Stark, Martin S Fischer, Alexander Hunt, Fletcher Young, Roger Quinn, and Emanuel Andrada. A three-dimensional musculoskeletal model of the dog. *Scientific reports*, 11(1):1–13, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Piotr Trochim, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, et al. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Takanobu Tsuihiji. Homologies of the transversospinalis muscles in the anterior presacral region of sauria (crown diapsida). *Journal of Morphology*, 263(2):151–178, 2005.

Takanobu Tsuihiji. Homologies of the longissimus, iliocostalis, and hypaxial muscles in the anterior presacral region of extant diapsida. *Journal of Morphology*, 268(11):986–1020, 2007.

Jack M Wang, Samuel R Hamner, Scott L Delp, and Vladlen Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

Jiangxin Wang, Dace Gao, and Pooi See Lee. Recent progress in artificial muscles for interactive soft robotics. *Advanced Materials*, 33(19):2003088, 2021.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Transactions on Graphics (TOG)*, 40 (4):1–11, 2021.

Zhaoming Xie, Glen Berseth, Patrick Clary, Jonathan Hurst, and Michiel van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1241–1246. IEEE, 2018.

Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pp. 317–329. PMLR, 2020.

Zhiqi Yin, Zeshi Yang, Michiel Van De Panne, and KangKang Yin. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)*, 40(4):1–17, 2021.

Felix E Zajac. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering*, 17(4):359–411, 1989.

Bo Zhou, Hongsheng Zeng, Fan Wang, Yunxiang Li, and Hao Tian. Efficient and robust reinforcement learning with uncertainty-based value expansion. *arXiv preprint arXiv:1912.05328*, 2019.

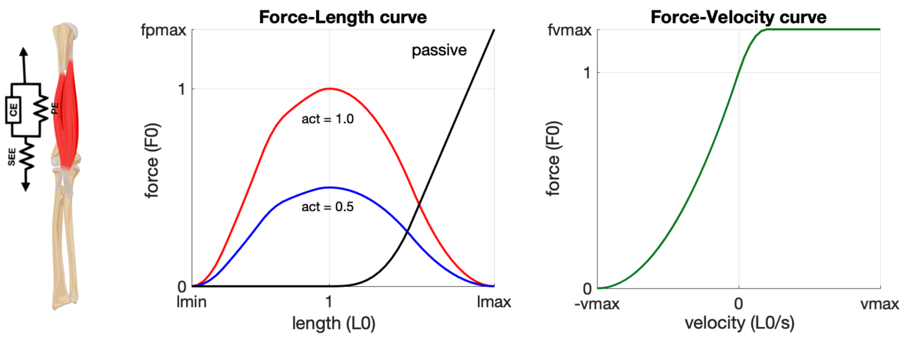

Figure 8: contraction dynamics

A Muscle dynamics

Difference between excitation and activation In biomechanics and neuroscience, researchers separate the excitation and activation signals. Neural excitation by the nervous system can be described as the discharge sequence of the efferent fibres. Neural excitation u is coupled with contraction via an intermediate state called activation a (Zajac, 1989). This intermediate state converts an electrochemical signal to mechanical force output. In MuJoCo, this conversion is modelled as a first-order nonlinear filter whose input is the control signal:

$$\frac{\partial a}{\partial t} = \frac{(u-a)}{\tau(u,a)}$$

where τ is defined as:

$$\tau(u,a) = \begin{cases} \tau_a(0.5+1.5a) & u > a \\ \tau_d/(0.5+1.5a) & u \leq a \end{cases}$$

τ_a, τ_d are time constants for activation and deactivation, equal by default to $10ms$ and $40ms$, respectively.

Contraction dynamics The contraction dynamics are implemented using the Hill-type model as shown in figure 8. This mechanical model computes the total force generated by a muscle by adding together the active contractile and passive properties of muscles. The contractile element models the muscle contraction force scaled by the activation signal a. The passive properties are modeled like springs: if we stretch a muscle it will generate a passive force to go back to its rest position.

The active contraction dynamics can be summarized by the formula:

$$f(l,\dot{l},a) = a \cdot f_l(l) \cdot f_v(\dot{l}) + f_p(l)$$

The functions f_l, f_p and f_v are plotted in figure 8. These functions are built into MuJoCo, capturing the contraction dynamics for basic purposes, and have been validated by the biomechanics community for some conditions (Millard et al., 2013).

All muscle-related quantities in MuJoCo are scaled by the muscle-tendon actuator's resting length; the advantage of this representation is that all muscles behave similarly. MuJoCo does not allow specification of the muscle resting length L_0 and tendon slack length LT directly (in contrast to OpenSim); this is due to the fact that MuJoCo does not include a stateful elastic component in the muscle model. This is a major compromise in MuJoCo when compared to OpenSim, that allows it to run faster. Instead, the actuator length range LR needs to be specified, which is the minimum and maximum of the sum of muscle and tendon lengths, along with a range R in units of L_0. In other words, the actuator length range LR defines the interval of values that the actuator is allowed to use during the simulation and the range R is a percentage expressing how much the actuator (here simply called "muscle") can shorten or extend.

At model compile time, MuJoCo solves these two equations where the unknowns are the muscle rest length L_0 and tendon length LT:

$$(LR_{min} - LT)/L_0 = R_{min}$$
$$(LR_{max} - LT)/L_0 = R_{max}$$

B Task details

Here we describe the various components of the tasks.

Table 1: Run forward task

Initialization	upright pose with random perturbation added to the legs, root rotation and height
Observation	head, pelvis and feet heights, joint positions (except x) and velocities, muscle activations, forces, lengths and velocities, horizontal velocity of the center of mass
Reward	forward velocity
Termination	head and pelvis height below threshold and the torso angle

Table 2: Mocap tracking task

Initialization	start at a random time step in the clip, with minimum number of steps (20) + small random perturbation
Observation	pelvis and feet heights, joint positions and velocities, muscle activations, forces, lengths and velocities, and the clip remaining time
Reward	product of Gaussian kernels measuring the similarity with the reference
Termination	when the reward drops below a threshold or when the end of the clip is reached (finite horizon)

Table 3: Foraging task

Model	ribcage and neck, ribcage fixed in space
Initialization	start with the neck pose from the previous episode, random target position in feasible area
Observations	joint positions and velocities, muscle activations, forces, lengths and velocities, the beak and target positions, the vector from the beak to the target
Reward	negative euclidean distance between the beak and the target
Termination	when the distance is below a threshold

C Hyperparameters

Here we describe the hyperparameters used in the experiments.

Table 4: Task parameters

Position weight	0.2
Rotation weight	0.1
Reward threshold	0.01
Initialization pose noise	0.02 (Gaussian scale)

Table 5: Training parameters

Model	MLP with 2x256 hidden units and ReLU
Distributional Atoms	51
Replay buffer size	1e6
Batch size	100
Steps before batches	5e4
Steps between batches	50
Number of batches	50
n-step return	1
Discount factor	0.99
Learning rate	1e-4
TD3 action noise scale	0.25
Exploration	Ornstein-Uhlenbeck
Action noise scale	0.25
Warm up random steps	1e4

D Motion capture data completion

Here we provide a visualization of the amount of missing data in the original mocap and representation of the model we used to predict the missing data.

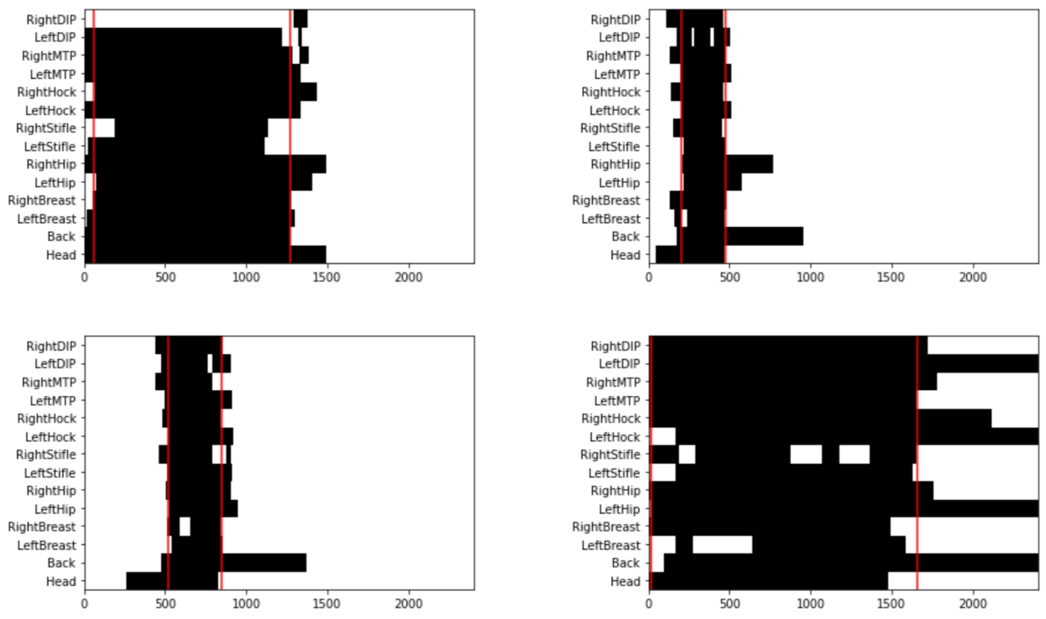

Figure 9: Binary mask showing the available data in black for 4 clips. The red lines indicate the intervals we used.

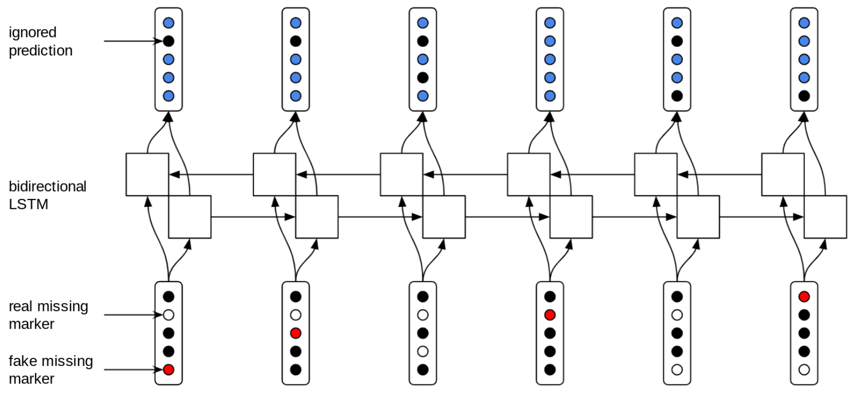

Figure 10: Bidirectional LSTM used to predict missing markers.

E Mocap tracking results

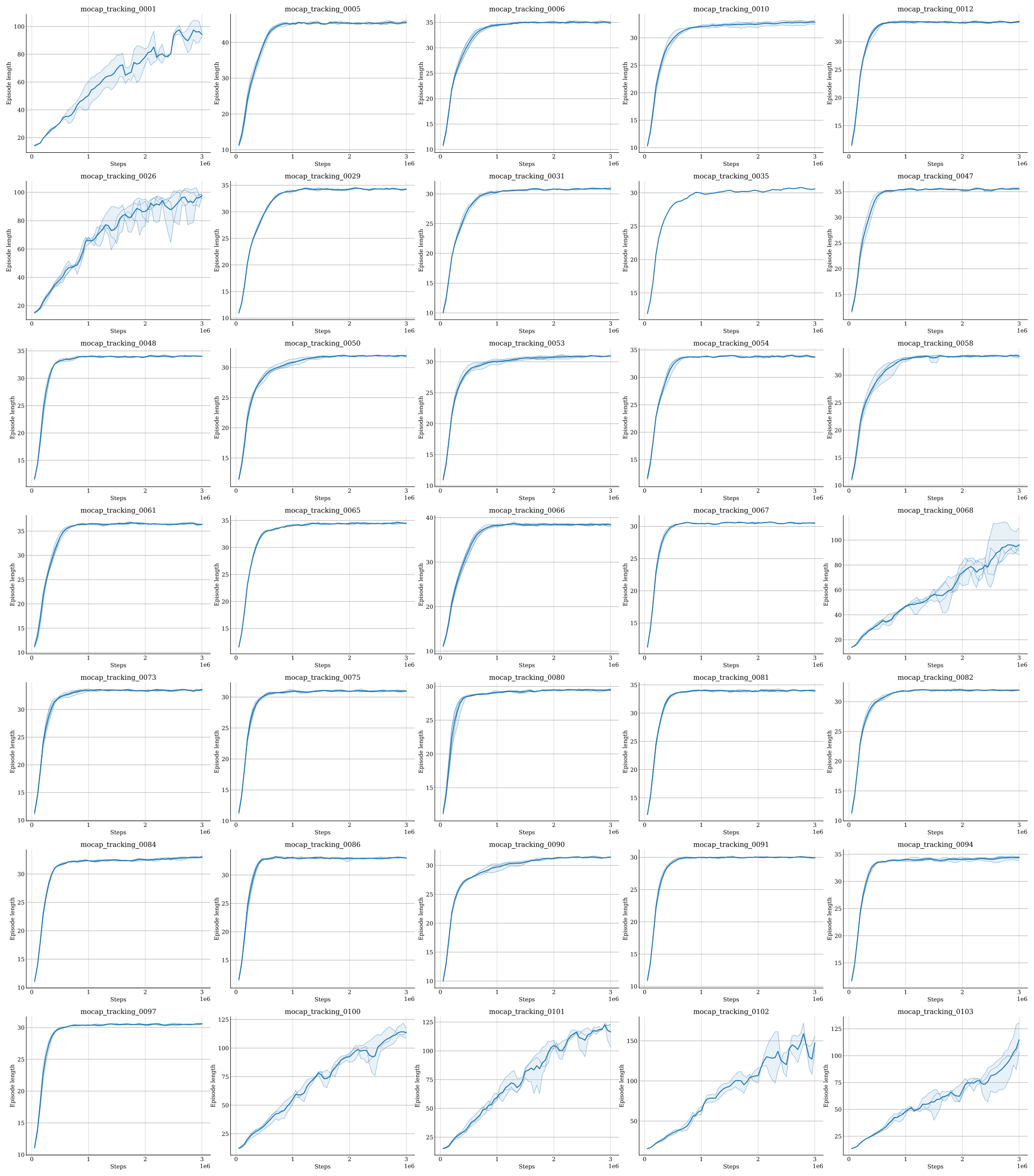

Figure 11: Average episode length for all the clips. As agents perform better at tracking, the episodes last longer. Values are averaged over 5 seeds and smoothed with a sliding window of size 3.