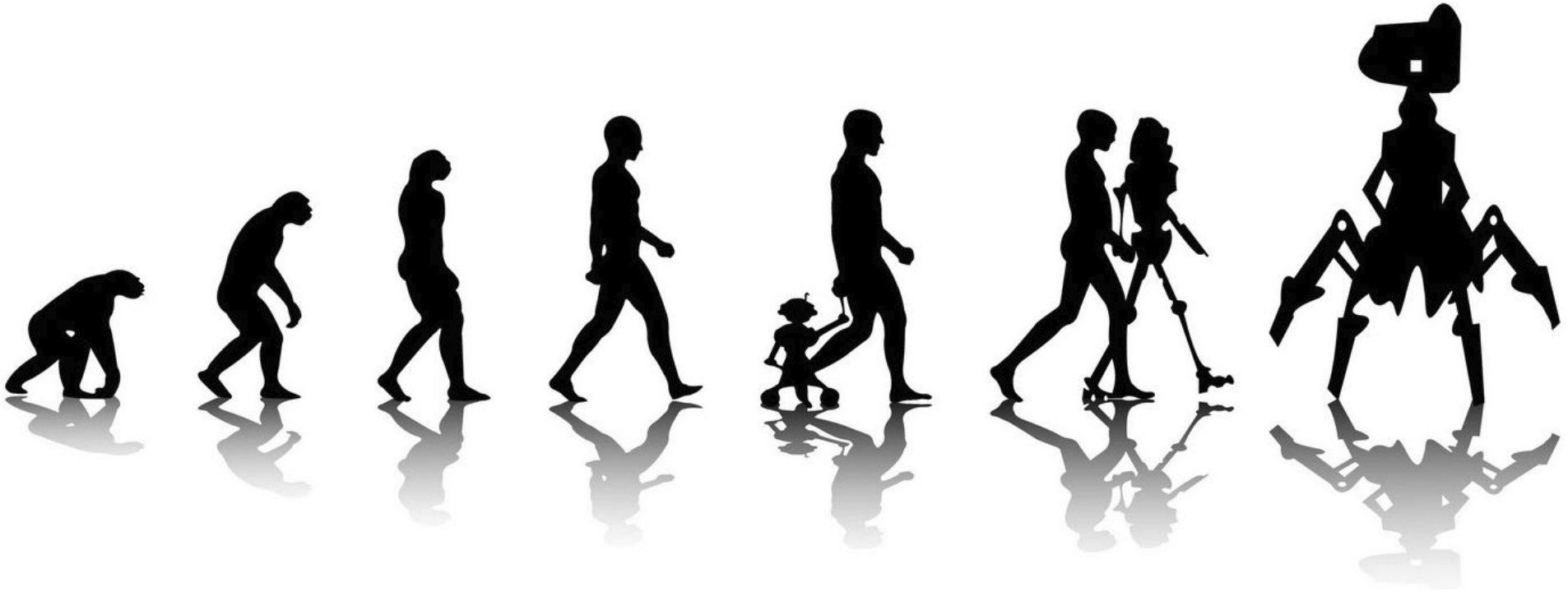


Modern Robotics: Evolutionary Robotics

COSC 4560 / COSC 5560

Professor Cheney

4/23/18

End-to-End Training of Deep Visuomotor Policies

Sergey Levine[†] SVLEVINE@EECS.BERKELEY.EDU
Chelsea Finn[†] CBFINN@EECS.BERKELEY.EDU
Trevor Darrell TREVOR@EECS.BERKELEY.EDU
Pieter Abbeel PABBEEL@EECS.BERKELEY.EDU
Division of Computer Science
University of California
Berkeley, CA 94720-1776, USA
[†]These authors contributed equally.

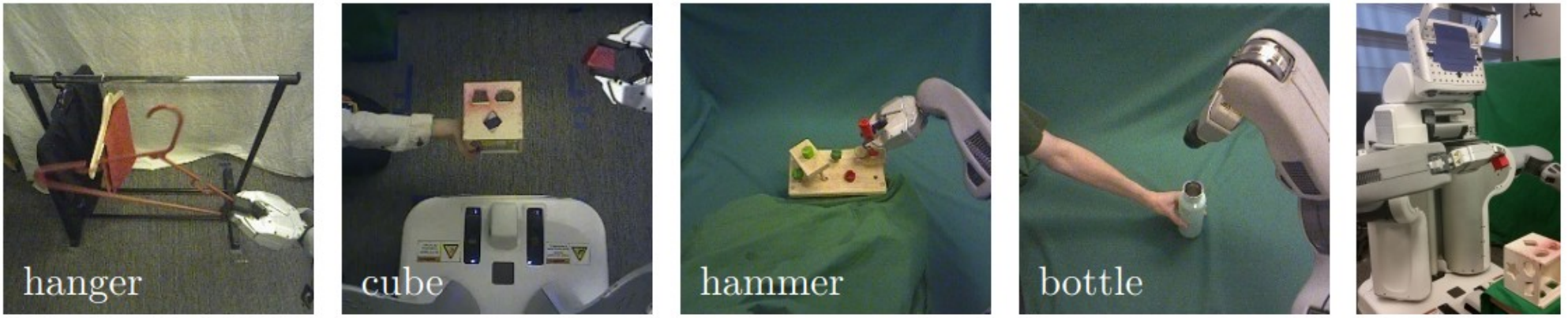


Figure 1: Our method learns visuomotor policies that directly use camera image observations (left) to set motor torques on a PR2 robot (right).

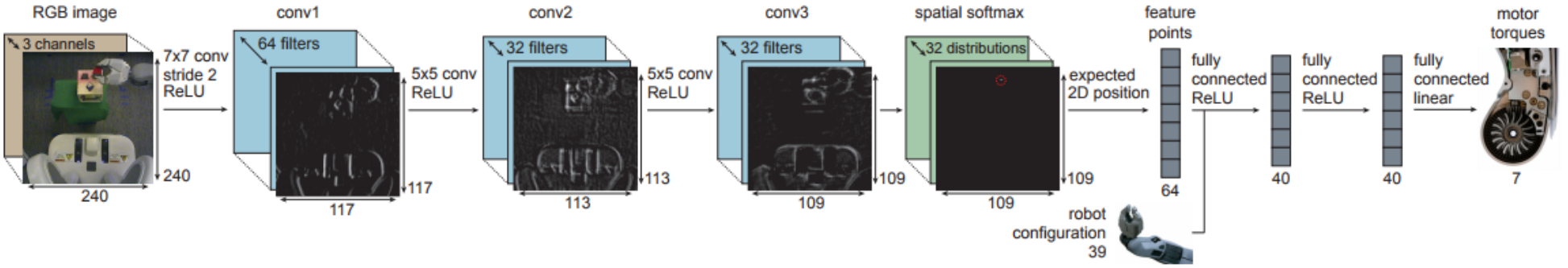


Figure 2: Visuomotor policy architecture. The network contains three convolutional layers, followed by a spatial softmax and an expected position layer that converts pixel-wise features to feature points, which are better suited for spatial computations. The points are concatenated with the robot configuration, then passed through three fully connected layers to produce the torques.

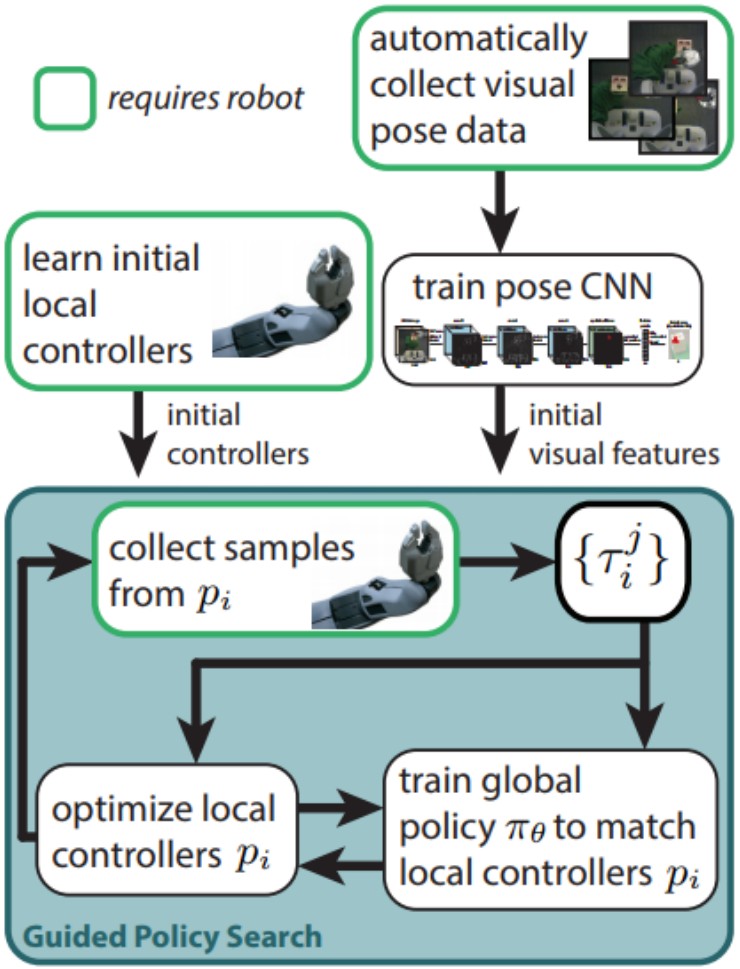


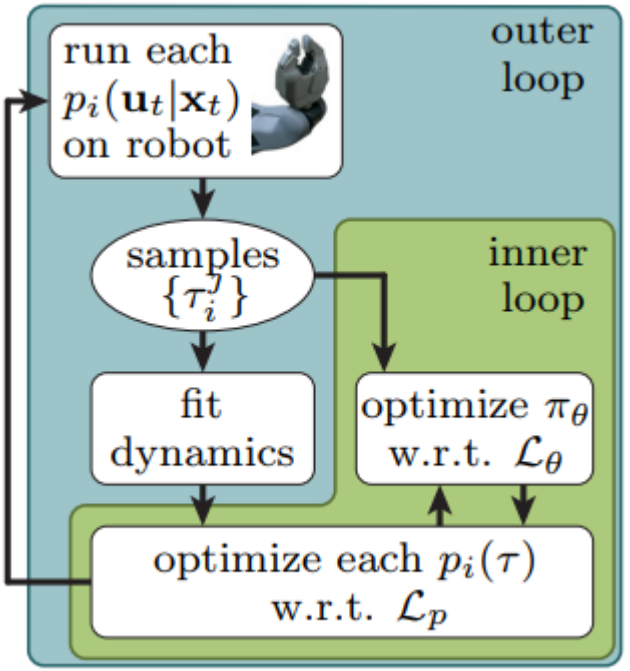
Figure 3: Diagram of our approach, including the main guided policy search phase and initialization phases.

$$\pi_\theta(\tau) = p(\mathbf{x}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t).$$

$$\tau = \{\mathbf{x}_1, \mathbf{u}_1, \mathbf{x}_2, \mathbf{u}_2, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$$

The goal of a task is given by a cost function $\ell(\mathbf{x}_t, \mathbf{u}_t)$, and the objective in policy search is to minimize the expectation $E_{\pi_\theta(\tau)}[\sum_{t=1}^{T} \ell(\mathbf{x}_t, \mathbf{u}_t)]$, which we will abbreviate as $E_{\pi_\theta(\tau)}[\ell(\tau)]$.

symbol	definition	example/details
$\mathbf{x}_t$	Markovian system state at time step $t \in [1, T]$	joint angles, end-effector pose, object positions, and their velocities; dimensionality: 14 to 32
$\mathbf{u}_t$	control or action at time step $t \in [1, T]$	joint motor torque commands; dimensionality: 7 (for the PR2 robot)
$\mathbf{o}_t$	observation at time step $t \in [1, T]$	RGB camera image, joint encoder readings & velocities, end-effector pose; dimensionality: around 200,000
τ	trajectory: $\tau = \{\mathbf{x}_1, \mathbf{u}_1, \mathbf{x}_2, \mathbf{u}_2, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$	notational shorthand for a sequence of states and actions
$\ell(\mathbf{x}_t, \mathbf{x}_t)$	cost function that defines the goal of the task	distance between an object in the gripper and the target
$p(\mathbf{x}_{t+1}\|\mathbf{x}_t, \mathbf{u}_t)$	unknown system dynamics	physics that govern the robot and any objects it interacts with
$p(\mathbf{o}_t\|\mathbf{x}_t)$	unknown observation distribution	stochastic process that produces camera images from system state
$\pi_\theta(\mathbf{u}_t\|\mathbf{o}_t)$	learned nonlinear global policy parameterized by weights θ	convolutional neural network, such as the one in Figure 2
$\pi_\theta(\mathbf{u}_t\|\mathbf{x}_t)$	$\int \pi_\theta(\mathbf{u}_t\|\mathbf{o}_t) p(\mathbf{o}_t\|\mathbf{x}_t) d\mathbf{o}_t$	notational shorthand for observation-based policy conditioned on state
$p_i(\mathbf{u}_t\|\mathbf{x}_t)$	learned local time-varying linear-Gaussian controller for initial state $\mathbf{x}_1^i$	time-varying linear-Gaussian controller has form $\mathcal{N}(\mathbf{K}_{ti}\mathbf{x}_t + \mathbf{k}_{ti}, \mathbf{C}_{ti})$
$\pi_\theta(\tau)$	trajectory distribution for $\pi_\theta(\mathbf{u}_t\|\mathbf{x}_t)$: $p(\mathbf{x}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{u}_t\|\mathbf{x}_t) p(\mathbf{x}_{t+1}\|\mathbf{x}_t, \mathbf{u}_t)$	notational shorthand for trajectory distribution induced by a policy



$$\mathcal{L}_\theta(\theta, p) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)] + E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] - E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$\mathcal{L}_p(p, \theta) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)] + E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] - E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^p(\theta, p),$$

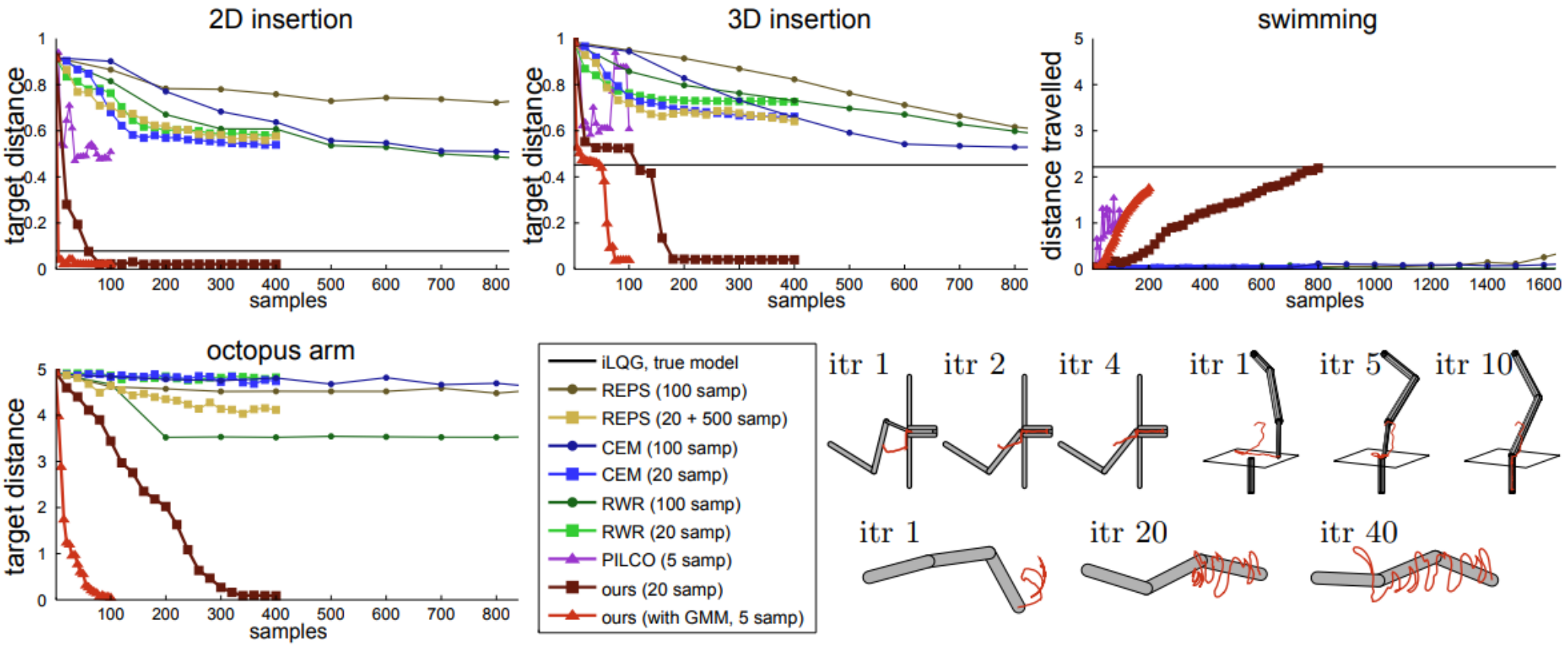


Figure 4: Results for learning linear-Gaussian controllers for 2D and 3D insertion, octopus arm, and swimming. Our approach uses fewer samples and finds better solutions than prior methods, and the GMM further reduces the required sample count. Images in the lower-right show the last time step for each system at several iterations of our method, with red lines indicating end effector trajectories.

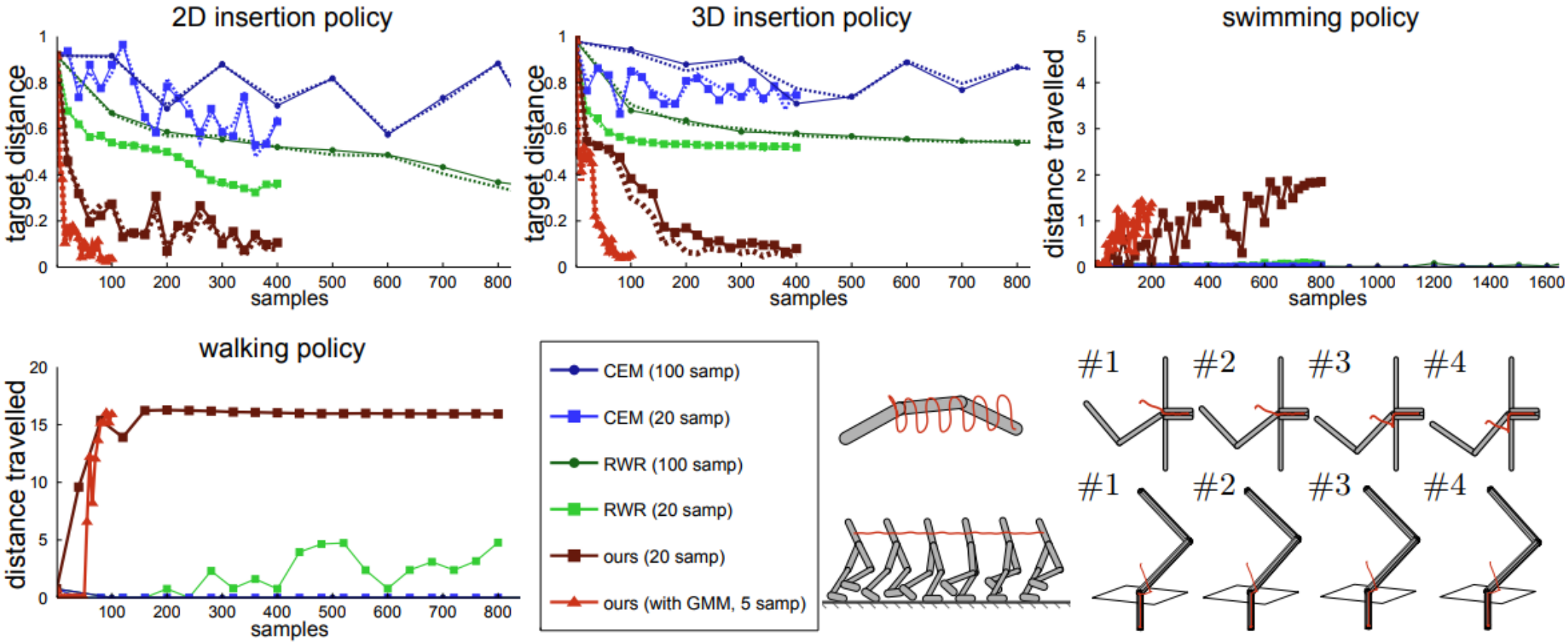


Figure 5: Comparison on neural network policies. For insertion, the policy was trained to search for an unknown slot position on four slot positions (shown above). Generalization to new positions is graphed with dashed lines. Note how the end effector (red) follows the surface to find the slot, and how the swimming gait is smoother due to the stationary policy.

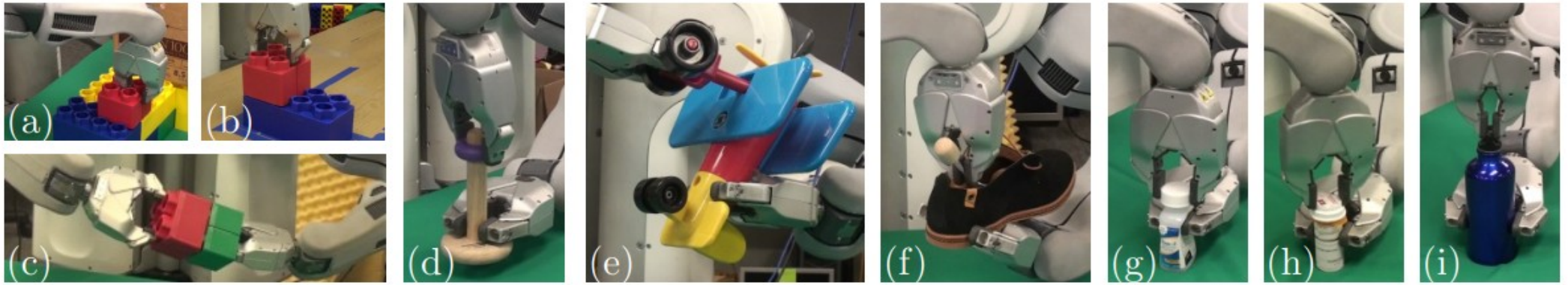


Figure 6: Tasks for linear-Gaussian controller evaluation: (a) stacking lego blocks on a fixed base, (b) onto a free-standing block, (c) held in both gripper; (d) threading wooden rings onto a peg; (e) attaching the wheels to a toy airplane; (f) inserting a shoe tree into a shoe; (g,h) screwing caps onto pill bottles and (i) onto a water bottle.

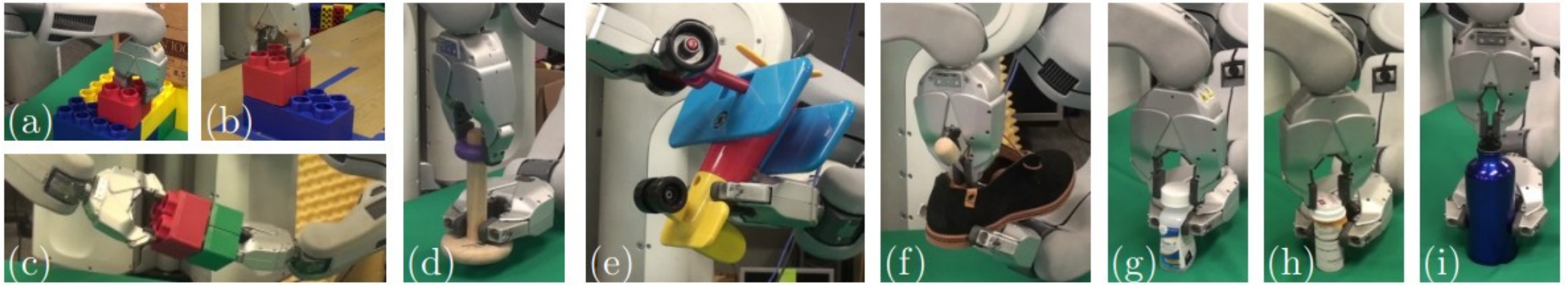


Figure 6: Tasks for linear-Gaussian controller evaluation: (a) stacking lego blocks on a fixed base, (b) onto a free-standing block, (c) held in both gripper; (d) threading wooden rings onto a peg; (e) attaching the wheels to a toy airplane; (f) inserting a shoe tree into a shoe; (g,h) screwing caps onto pill bottles and (i) onto a water bottle.

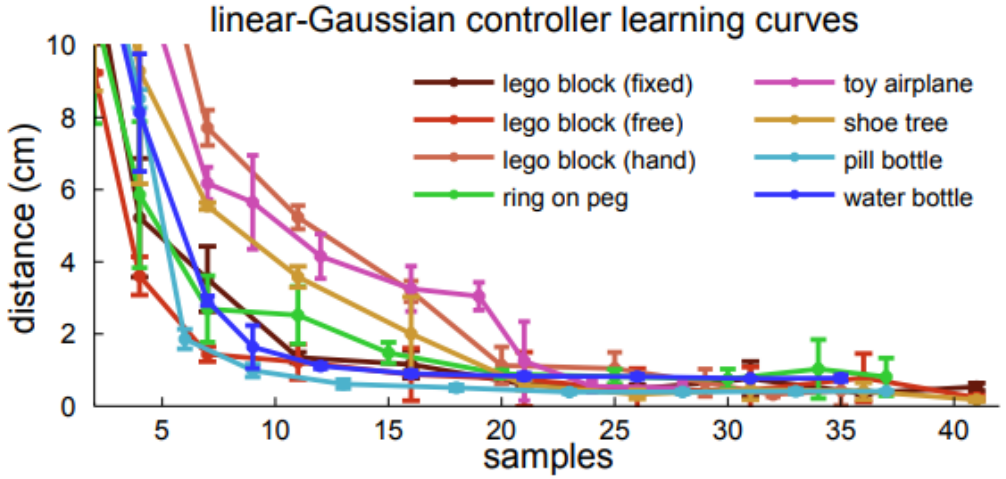


Figure 7: Distance to target point during training of linear-Gaussian controllers. The actual target may differ due to perturbations. Error bars indicate one standard deviation.

End-to-End Training of Deep Visuomotor Policies

With Audio

End-to-End Training of Deep Visuomotor Policies

Learned Visual Representations

With Audio

Terrain-Adaptive Locomotion Skills Using Deep Reinforcement Learning

Xue Bin Peng, Glen Berseth, Michiel van de Panne*
University of British Columbia

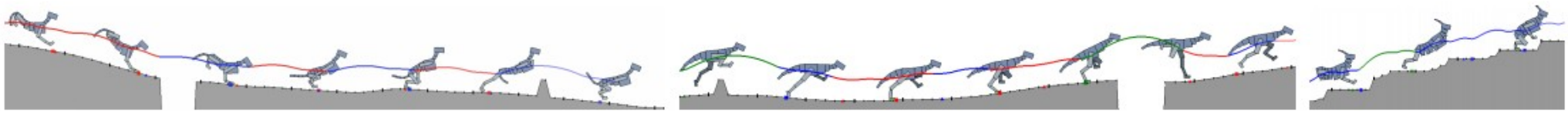


Figure 1: *Terrain traversal using a learned actor-critic ensemble. The color-coding of the center-of-mass trajectory indicates the choice of actor used for each leap.*

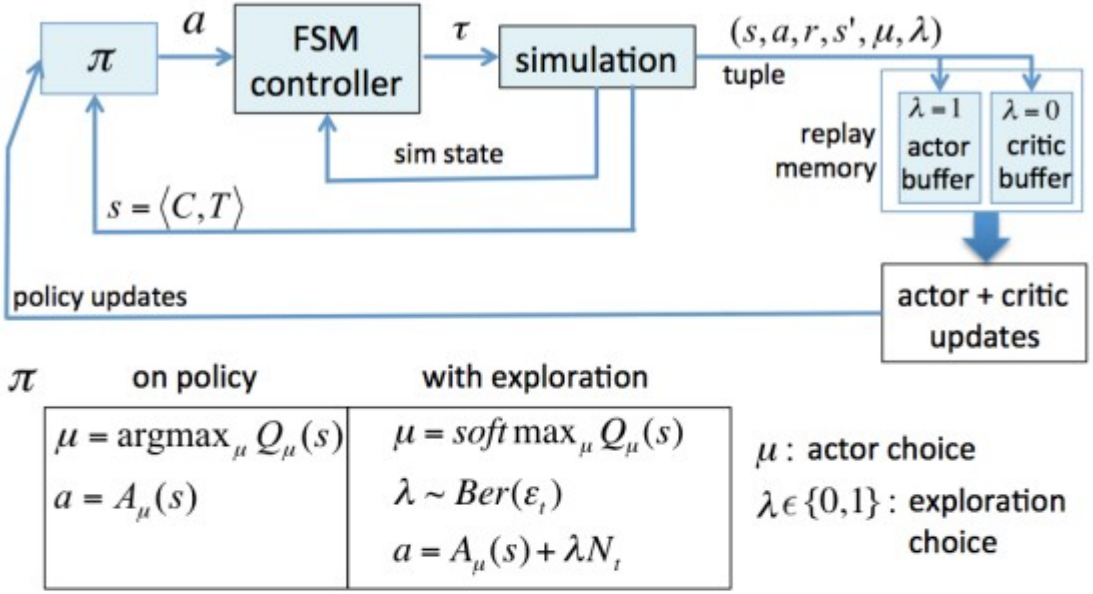


Figure 2: *System Overview*

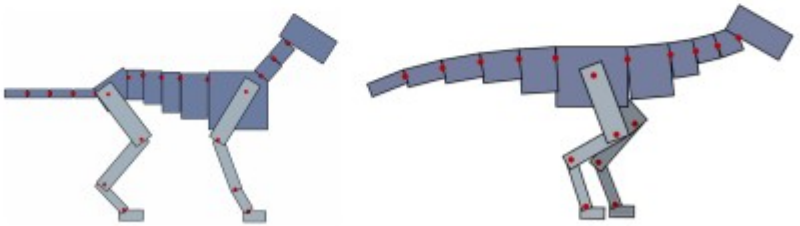


Figure 3: *21-link planar dog (left), and 19-link raptor (right).*

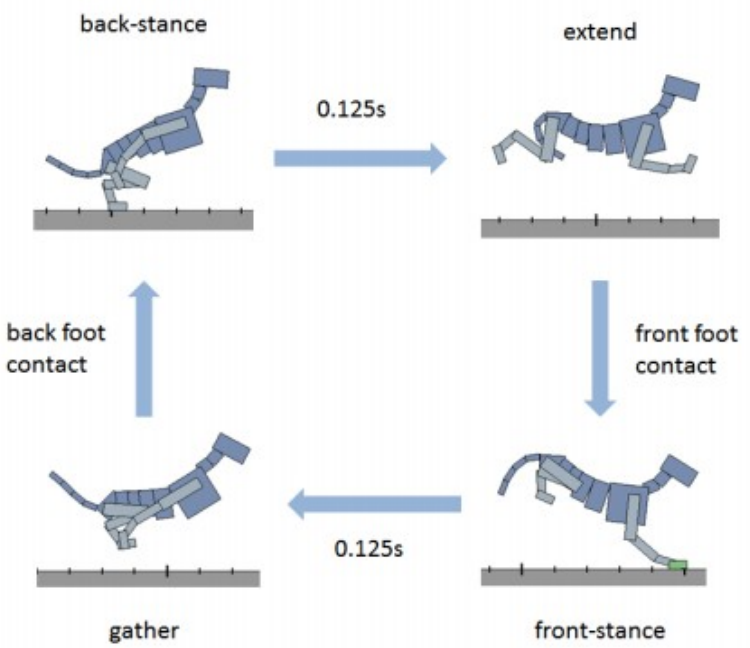


Figure 4: *Dog controller motion phases.*

Similar to much prior work in physics-based character animation, the motion is driven using joint torques and is guided by a finite state machine. Figure 4 shows the four phase-structure of the controller. In each motion phase, the applied torques can be decomposed into three components,

A policy is a mapping between a state space S and an action space A, i.e., $\pi(s) : S \mapsto A$. For our framework, S is a continuous space that describes the state of the character as well as the configuration of the upcoming terrain. The action space A is represented by a 29D continuous space where each action specifies a set of parameters to the FSM. The following sections provide further details about the policy representation.

Prior to learning the policy, a small set of initial actions are created which are used to seed the learning process. The set of actions consists of 4 runs and 4 leaps. All actions are synthesized using a derivative-free optimization process, CMA [Hansen 2006]. Two runs are produced that travel at approximately 4 m/s and 2 m/s, respectively. These two runs are then interpolated to produce 4 runs of varying speeds. Given a sequence of successive fast-run cycles, a single cycle of that fast-run is then optimized for distance traveled, yielding a 2.5 m leap that can then be executed from the fast run. The leap action is then interpolated with the fast run to generate 4 parametrized leaps that travel different distances.

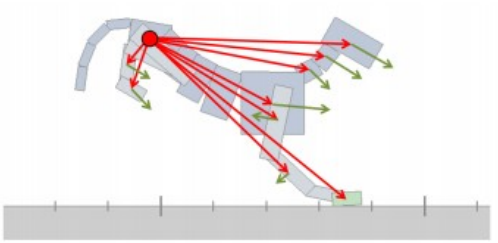


Figure 5: *The character features consist of the displacements of the centers of mass of all links relative to the root (red) and their linear velocities (green).*

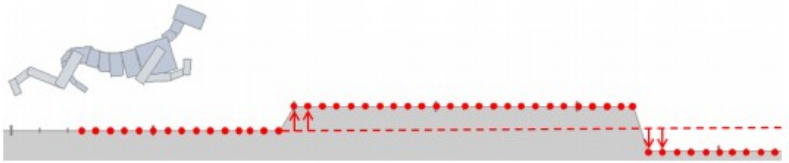


Figure 6: *Terrain features consist of height samples of the terrain in front of the character, evenly spaced 5cm apart. All heights are expressed relative to the height of the ground immediately under the root of the character.*

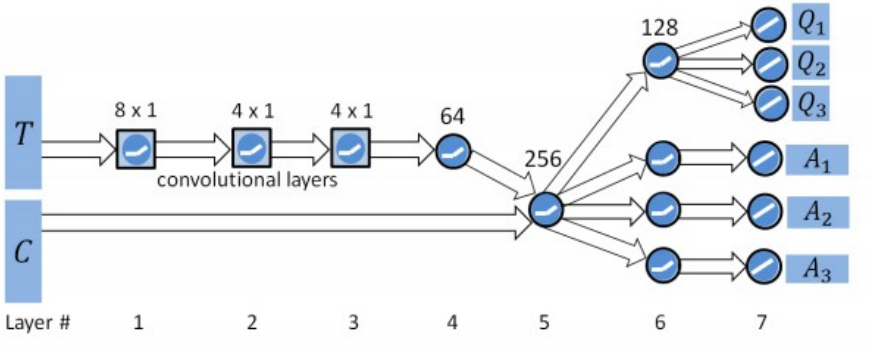


Figure 8: *Schematic illustration of the MACE convolutional neural network. T and C are the input terrain and character features. Each A_μ represents the proposed action of actor μ, and Q_μ is the critic's predicted reward when activating the corresponding actor.*

In reinforcement learning the reward function, $r(s, a, s')$, is used as a training signal to encourage or discourage behaviors in the context of a desired task. The reward provides a scalar value reflecting the desirability of a particular state transition that is observed by performing action a starting in the initial state s and resulting in a successor state s'. Figure 7 is an example of a sequence of state transitions for terrain traversal. For the terrain traversal task, the reward is provided by

$$r(s, a, s') = \begin{cases} 0, & \text{character falls during the cycle} \\ e^{-\omega(v^* - v)^2}, & \text{otherwise} \end{cases}$$

where a fall is defined as any link of the character's trunk making contact with the ground for an extended period of time, v is the average horizontal velocity of the center of mass during a cycle, $v^* = 4m/s$ is the desired velocity, and $\omega = 0.5$ is the weight for the velocity error. This simple reward is therefore designed to encourage the character to travel forward at a consistent speed without falling. If the character falls during a cycle, it is reset to a default state and the terrain is regenerated randomly.

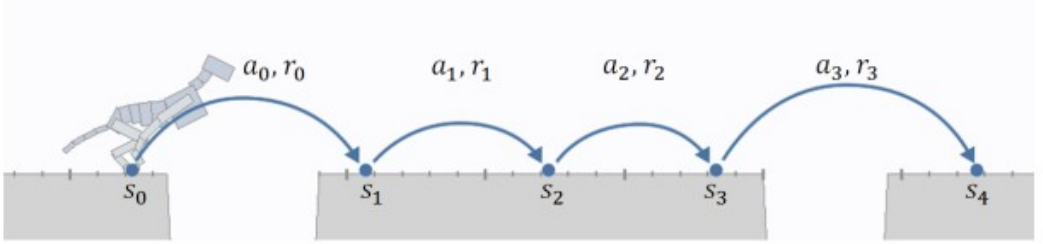
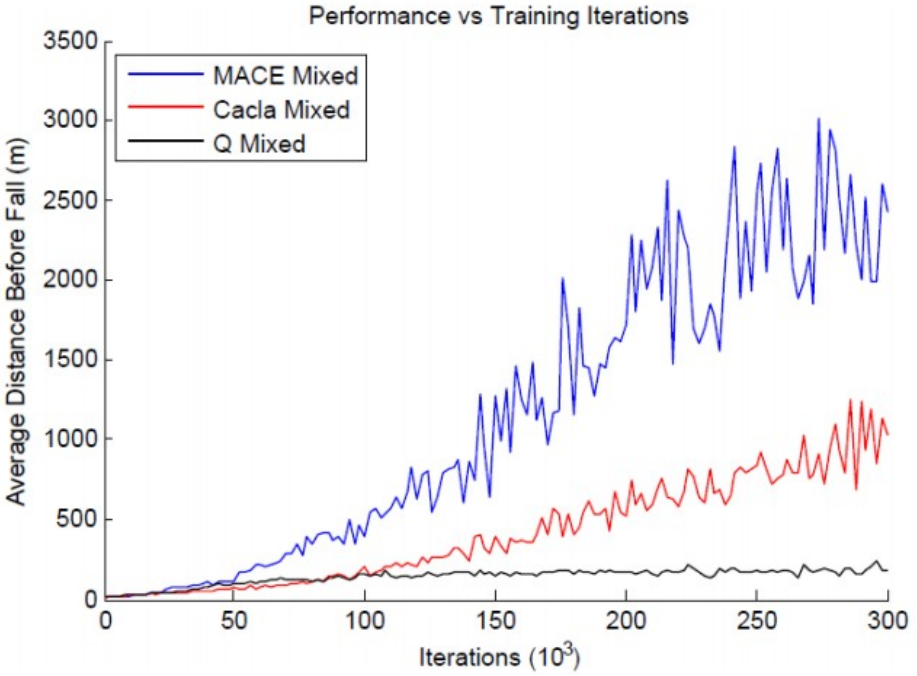


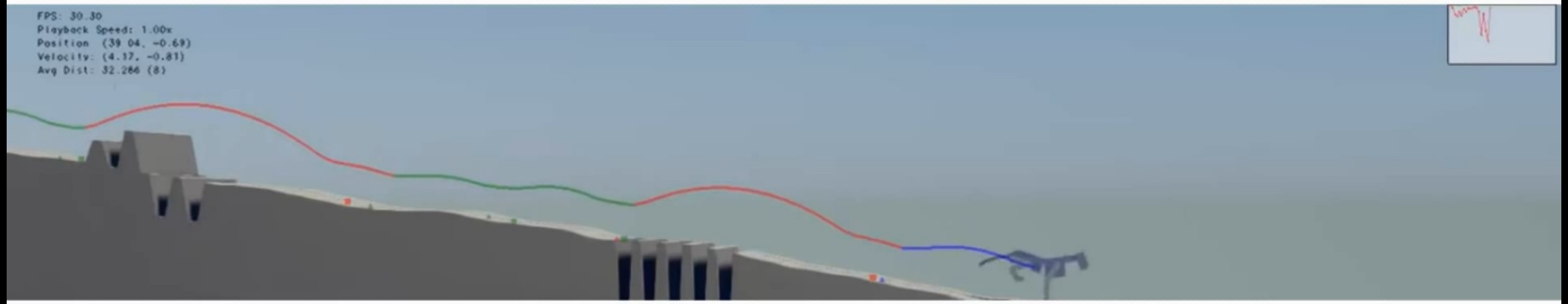
Figure 7: *Each state transition can be recorded as a tuple* $\tau_i = (s_i, a_i, r_i, s_i')$. s_i *is the initial state,* a_i *is the action taken,* s_i' *is the resulting state, and* r_i *is the reward received during the* i*th cycle.*



(a) MACE(3) vs CACLA and Q-learning on mixed terrain

Contributions: In this paper we use deep neural networks in combination with reinforcement learning (DeepRL) to address the above challenges. This allows for the design of control policies that operate directly on high-dimensional character state descriptions (83D) and an environment state that consists of a height-field image of the upcoming terrain (200D). We provide a parameterized action space (29D) that allows the control policy to operate at the level of bounds, leaps, and steps. We introduce a novel *mixture of actor-critic experts (MACE) architecture* to enable accelerated learning. MACE develops n individual control policies and their associated value functions, which each then specialize in particular regimes of the overall motion. During final policy execution, the policy associated with the highest value function is executed, in a fashion analogous to Q-learning with discrete actions. We show the benefits of *Boltzmann exploration* and various algorithmic features for our problem domain. We demonstrate improvements in motion quality and terrain abilities over previous work.

Terrain-Adaptive Locomotion Skills using Deep Reinforcement Learning



Xue Bin Peng, Glen Berseth, Michiel van de Panne
University of Biritish Columbia

includes audio

Flexible Muscle-Based Locomotion for Bipedal Creatures

Thomas Geijtenbeek*
Utrecht University

Michiel van de Panne
University of British Columbia

A. Frank van der Stappen
Utrecht University

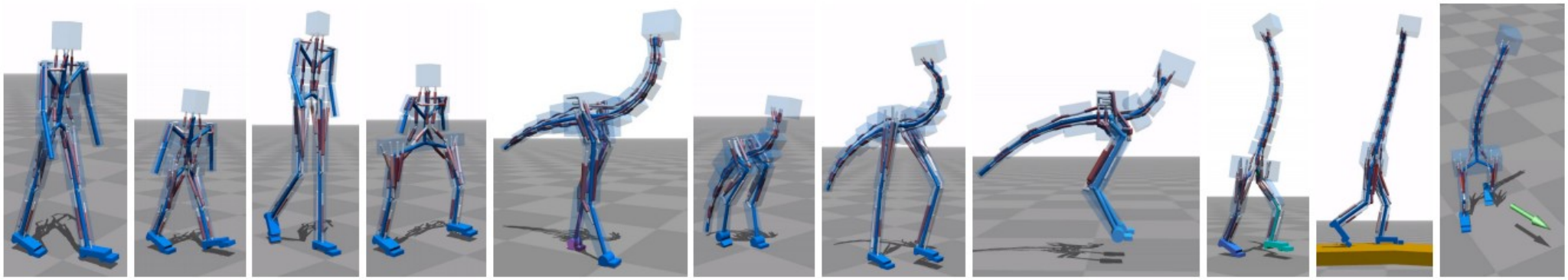


Figure 1: *Physics-based simulation of locomotion for a variety of creatures driven by 3D muscle-based control. The synthesized controllers can locomote in real time at a range of speeds, be steered to a target heading, and can traverse variable terrain.*

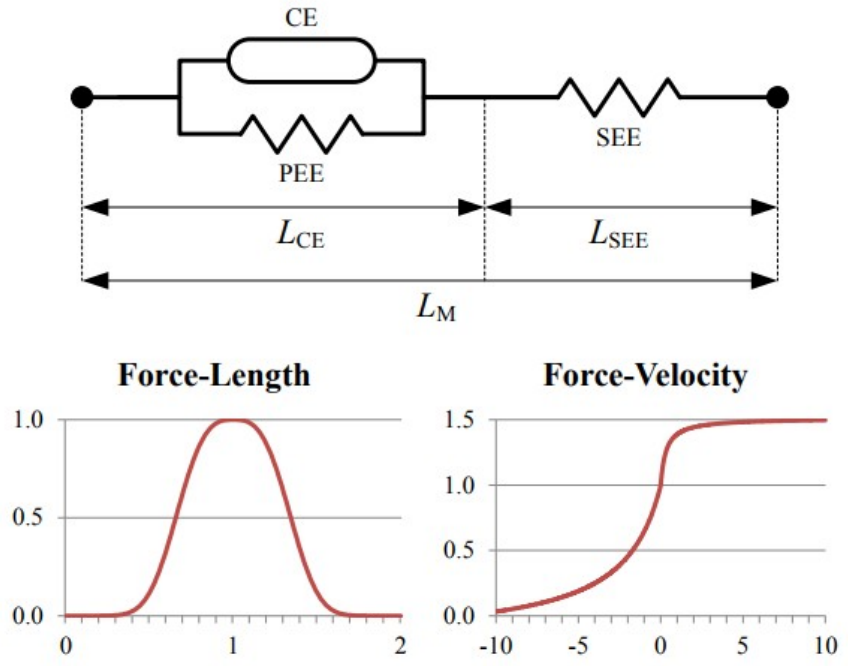


Figure 2: *Top: The three components of a Hill-type muscle. Bottom: Normalized force-length and force-velocity relations of the contractile element.*

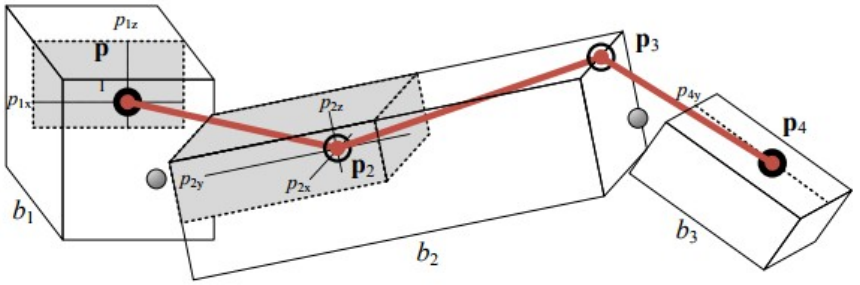


Figure 3: *Muscle attachment points that will be optimized within a constrained region. In this example, muscle point* $\mathbf{p}_1$ *is constrained to a 2D surface, muscle point* $\mathbf{p}_2$ *is constrained to a 3D volume,* $\mathbf{p}_3$ *is fixed, and* $\mathbf{p}_4$ *is constrained to a line. The actual areas used in our experiments are shown in Figure 8.*

Subject	Parameters	Section
Muscle physiology	3–30 *	3.1
Muscle geometry	12–39 *	3.3
State transition	3	4.1
Target features	14	4.2
Feedback control	14–63 *	4.3, 4.4
Initial character state	6	†

Table 1: *Parameters subject to optimization. The number of parameters marked * is model dependent (see Table 3). † The parameters for initial character state are: initial forward lean, and initial speeds for upper swing leg, lower swing leg (and foot), upper stance leg, lower stance leg (and foot), and other bodies.*

5 Optimization

Both our muscle model (Section 3) and control model (Section 4) introduce a large number of free parameters, which are determined through off-line optimization (see Table 1 for an overview). The total set of parameters, $\mathbf{K}$, is optimized using Covariance Matrix Adaptation [Hansen 2006], with step size $\sigma = 1$ and population size $\lambda = 20$.

Objective The goal of our optimization process is to minimize the error $\bar{E}(\mathbf{K})$, which consists of the following components:

$$\bar{E}(\mathbf{K}) = \bar{E}_{\text{speed}} + \bar{E}_{\text{headori}} + \bar{E}_{\text{headvel}} + \bar{E}_{\text{slide}} + \bar{E}_{\text{effort}} \quad (32)$$

Flexible Muscle-Based Locomotion for Bipedal Creatures

SIGGRAPH ASIA 2013

Thomas Geijtenbeek
Michiel van de Panne
Frank van der Stappen