NeuroAnimator

# Fast Neural Network Emulation and Control of Physics-Based Models


***Radek Grzeszczuk [1]***

***Demetri Terzopoulos [2,1]***

***Geoffrey Hinton [2]***

[1] ***Intel Corporation***
***Microcomputer Research Lab***

[2] ***University of Toronto,***
***Dept. of Computer Science***


intel

International Conference Graphicon 1998, Moscow, Russia, http://www.graphicon.ru/

# Physics-Based Animation

## *Animation through physical simulation*

- Inanimate objects:
  - *rigid models*
  - *articulated models*
  - *deformable models*
- Animate objects:
  - *animal models*
  - *human models*

- Pioneering work

*(Hahn88, Baraff89)*

*(Barzel88)*

*(Terzopoulos87, Platt88)*

*(Miller88, Tu95)*

*(Armstrong85, Wilhelms87, Hodgins95,)*

# Physics-based Models

## *Simulate Newtonian mechanics*

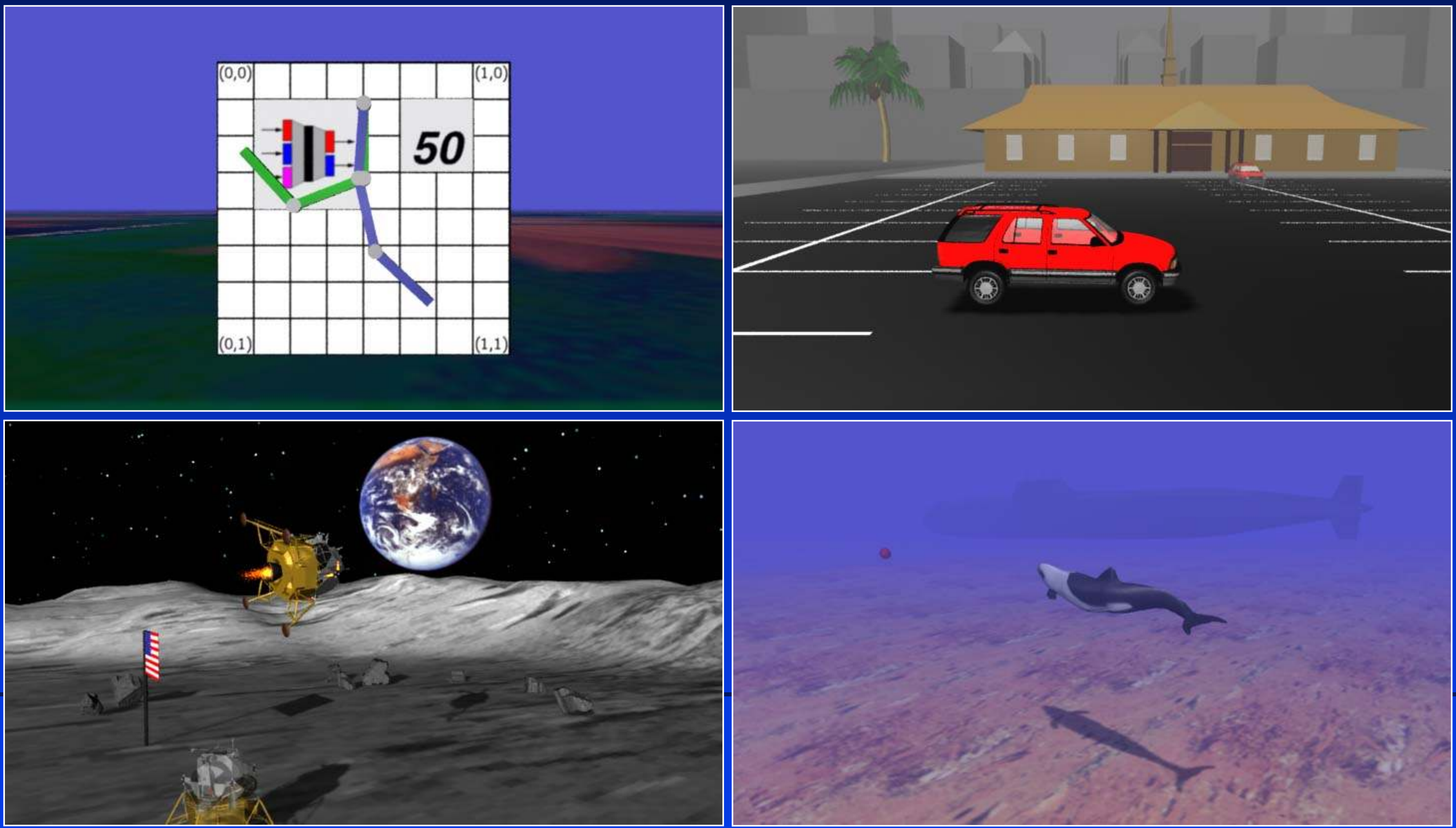
- Benefits
  - *offer unsurpassed realism*
  - *automate motion synthesis*
- Drawbacks
  - *incur high computational costs*
  - *difficult & expensive to control*
- Moore’s Law is on our side!

# NeuroAnimator

## *A neural network approach to physically realistic animation*

- Learns to approximate physical models by observing their actions
- Yields outstanding efficiency
  - *fast synthesis of physically realistic motion*
  - *fast synthesis of motion controllers for animation*

# Example NeuroAnimators



# Motivation

***Is there a more efficient alternative to animation by simulation?***

- Numerical simulation of a dynamical system evaluates a high-dimensional map $\Phi$ at every timestep
- In principle (Cybenko89), neural networks can learn to approximate arbitrary, complex maps $\Phi$
- NeuroAnimator: accurate and efficient neural network approximation of maps $\Phi$ associated with physics-based CG models

# Motivation

## *Animation through numerical simulation*

- Discrete-time dynamical systems

Map

$$\mathbf{s}_{t+\delta t} = \Phi(\mathbf{s}_t, \mathbf{u}_t, \mathbf{f}_t)$$

State

External Forces

Controls

$$\mathbf{s} = [\mathbf{x}\ \mathbf{v}]$$

- Example:

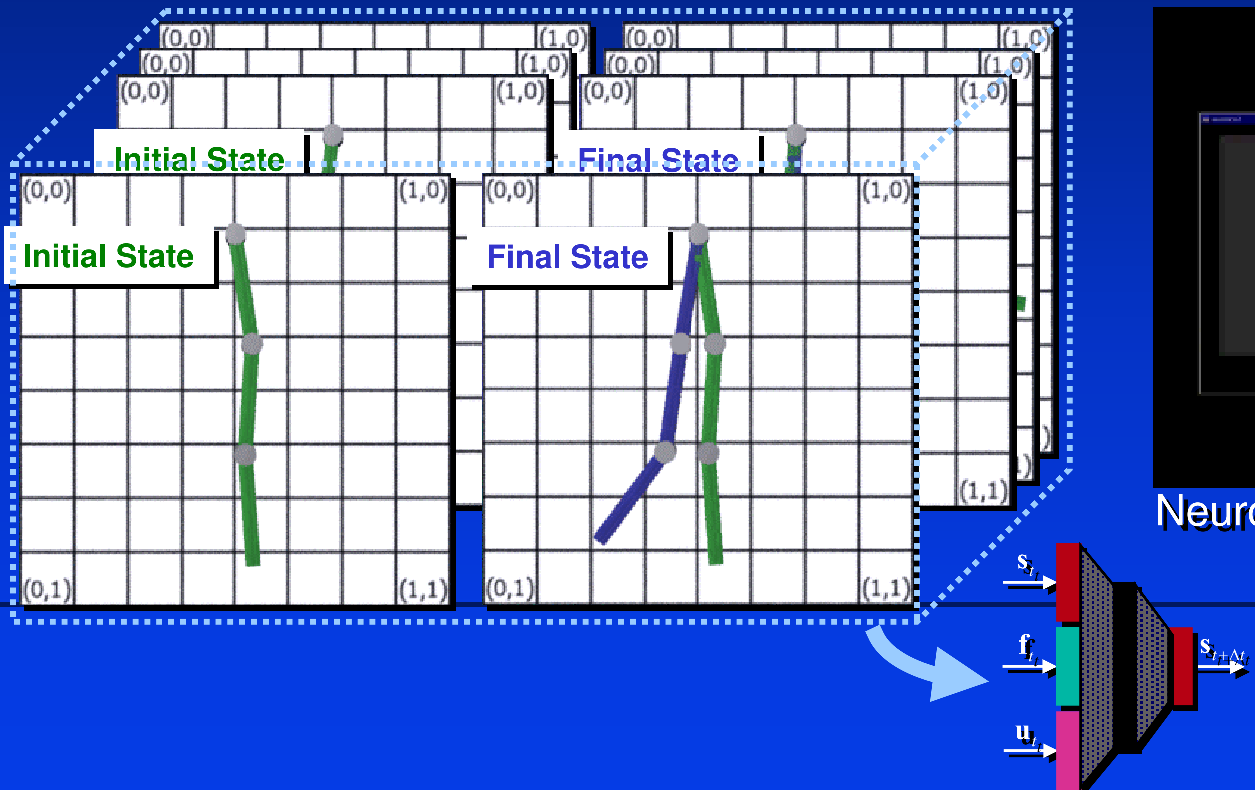
Implicit Euler time-integration method

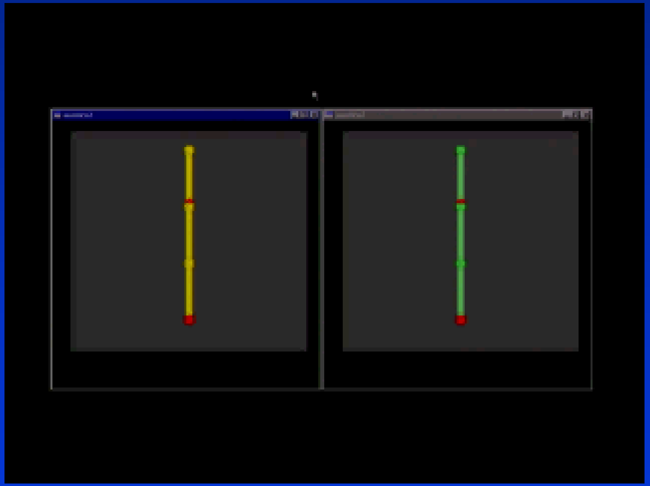
$$\begin{cases} \mathbf{A}_t \mathbf{v}_{t+\delta t} = \mathbf{v}_t + \mathbf{g}(\mathbf{u}_t, \mathbf{f}_t) \\ \mathbf{x}_{t+\delta t} = \mathbf{v}_t \delta t + \mathbf{x}_t \end{cases}$$

$\mathbf{s}_{t+\delta t}$ $\mathbf{s}_t$

# Learning Dynamics

## *The NeuroAnimator learns dynamics by observing sample state transitions*



NeuroAnimator Physical Model

# Emulation

super timestep
$\Delta t = n\,\delta t$

NeuroAnimator approximation of $\Phi$

$$\mathbf{s}_{t+\Delta t} = \mathbf{N}_{\Phi}(\mathbf{s}_t, \mathbf{u}_t, \mathbf{f}_t)$$

## *Why is the NeuroAnimator efficient?*

- The emulation step is relatively cheap
- The NeuroAnimator can emulate super timesteps
  - *up to 100 times faster than numerical simulation*
- $\mathbf{N}_{\Phi}$ is analytically differentiable
  - *dramatic efficiency for animation controller synthesis*

# Talk Overview

- Introduction
- Artificial neural networks
- From physical models to NeuroAnimators
- NeuroAnimator based controller synthesis
- Conclusion and future work

# Talk Overview

- Introduction
- Artificial neural networks
- From physical models to NeuroAnimators
- NeuroAnimator based controller synthesis
- Conclusion and future work

# Neural Networks

## *Seminal work in the field*

- Perceptrons

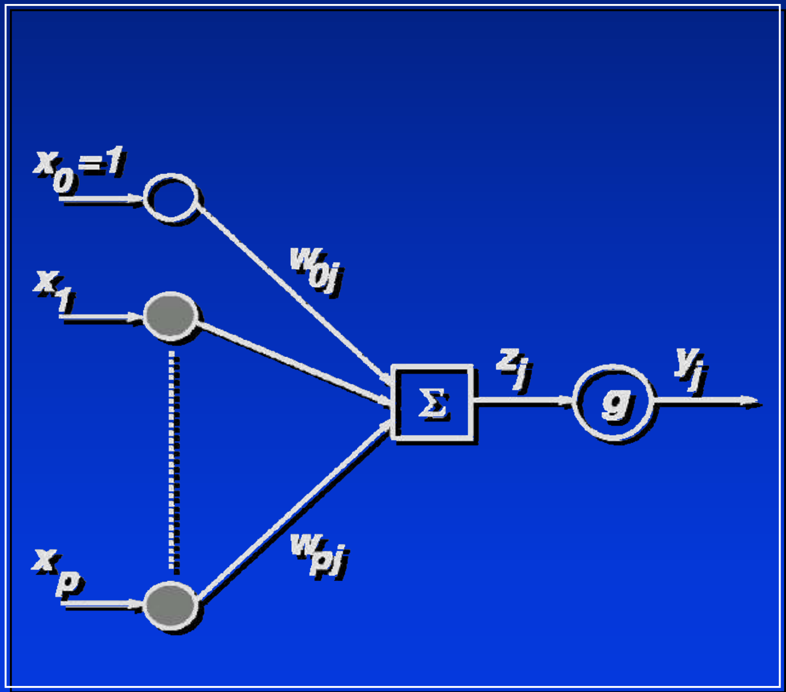
  (Widrow60, Rosenblatt62, Minsky69)

- Backpropagation learning algorithm
  (Rumelhart86) (Bryson69, Werbos74, Parker85)

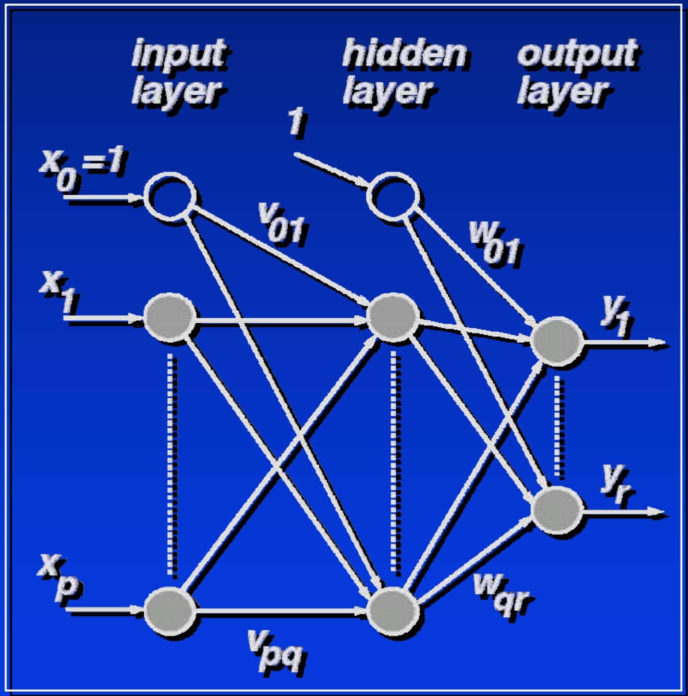
  – *backpropagation through time*
    *(Rumelhart86)*

# Artificial Neural Networks

## *Networks of simple computing elements*



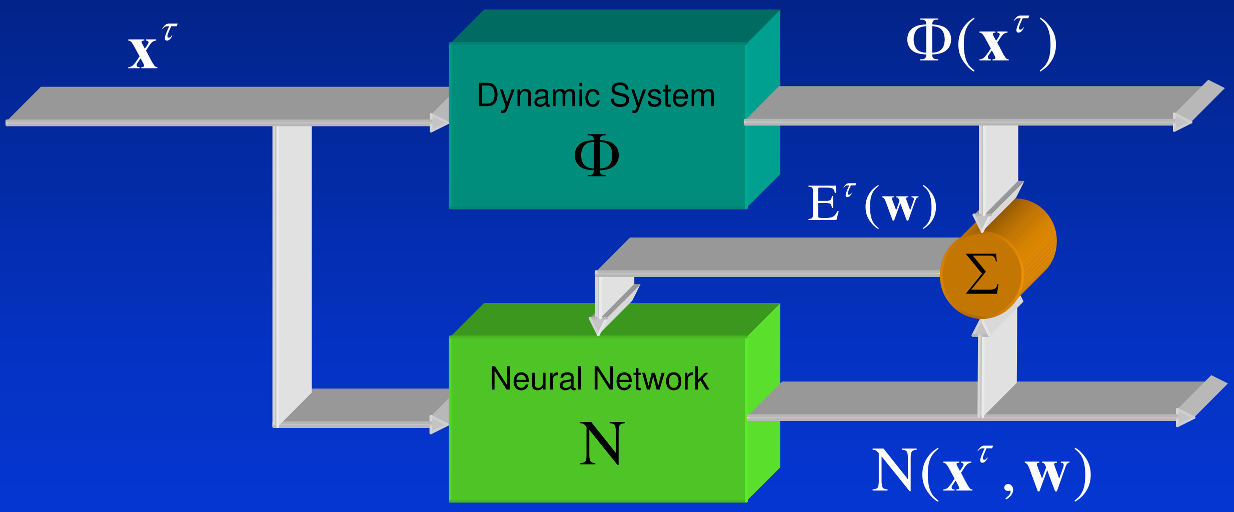

Neuron




Feedforward Network

# Backpropagation

## *Adjusts the weights of a neural network*



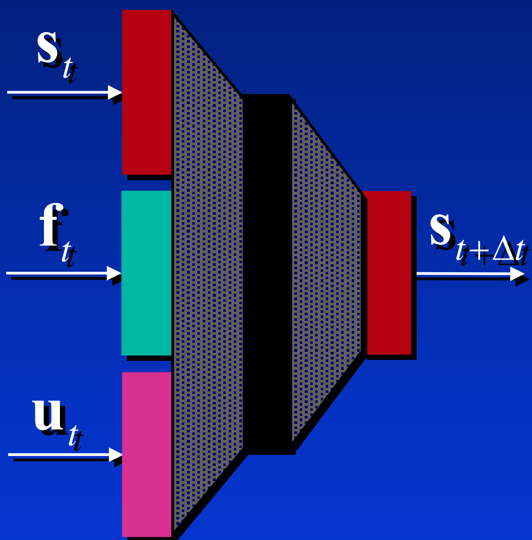

- Approximation error: $E^{\tau}(\mathbf{w}) = \left\| \Phi(\mathbf{x}^{\tau}) - N_{\Phi}(\mathbf{x}^{\tau}, \mathbf{w}) \right\|^2$
- Weights update formula: $\mathbf{w}^{l+1} = \mathbf{w}^{l} - \eta_{w} \nabla_{\mathbf{w}} E^{\tau}(\mathbf{w}^{l})$

# Talk Overview

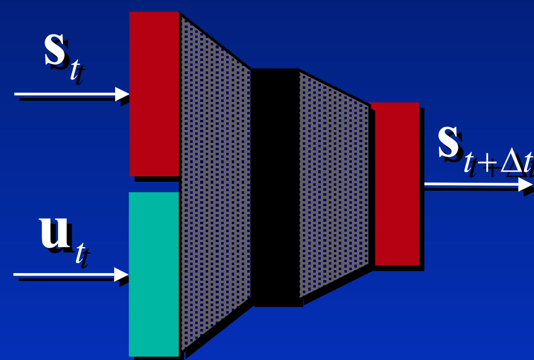
- Introduction
- Artificial neural networks
- From physical models to NeuroAnimators
  - *emulation results*
- NeuroAnimator based controller synthesis
- Conclusion and future work

# NeuroAnimator Structure

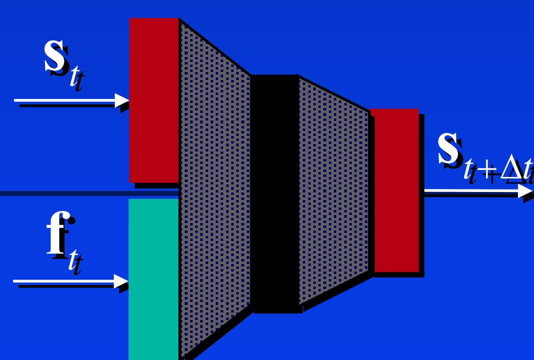
Active dynamic,
nondeterministic forces



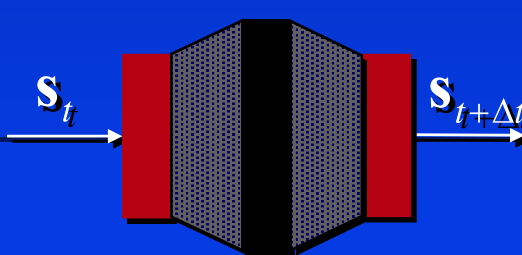

Active dynamic,
deterministic forces




Passive dynamic,
nondeterministic forces

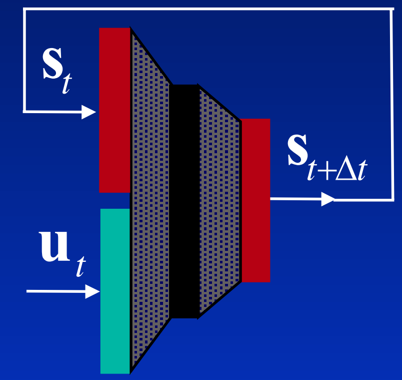



Passive dynamic,
deterministic forces

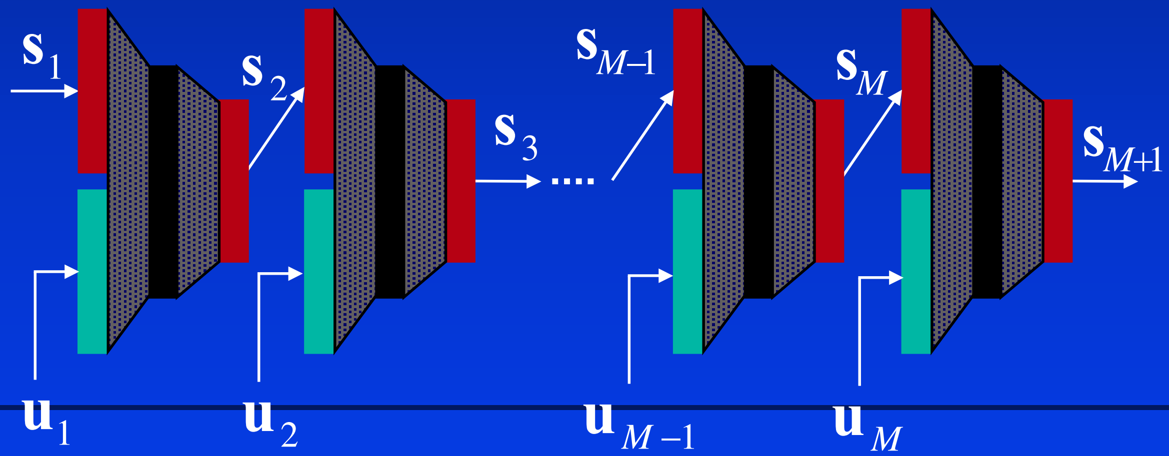


# Emulation

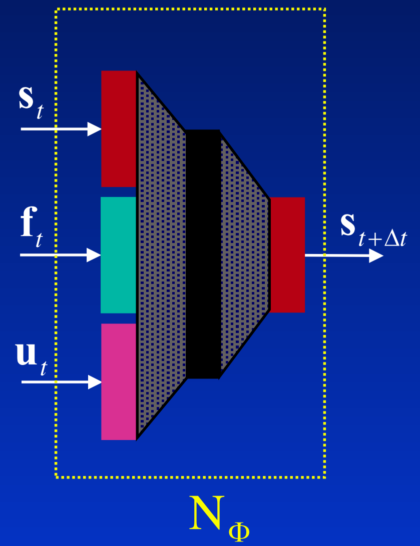
## *Sequence of network evaluations*



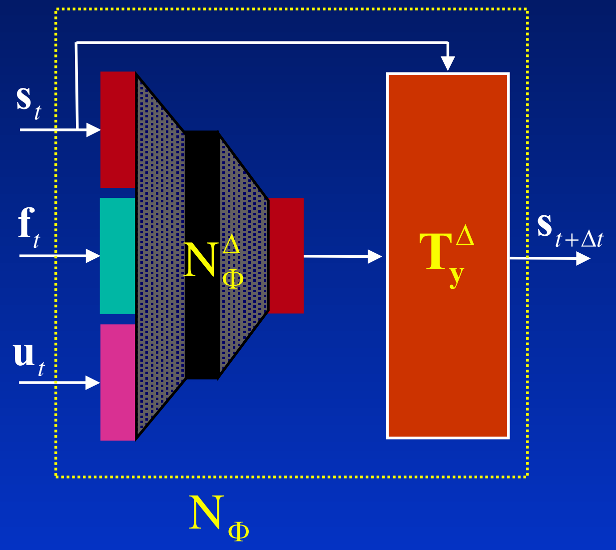

$$\mathbf{s}_{t+\Delta t} = \mathbf{N}_{\Phi}(\mathbf{s}_t, \mathbf{u}_t, \mathbf{f}_t)$$



# Input & Output Transformations

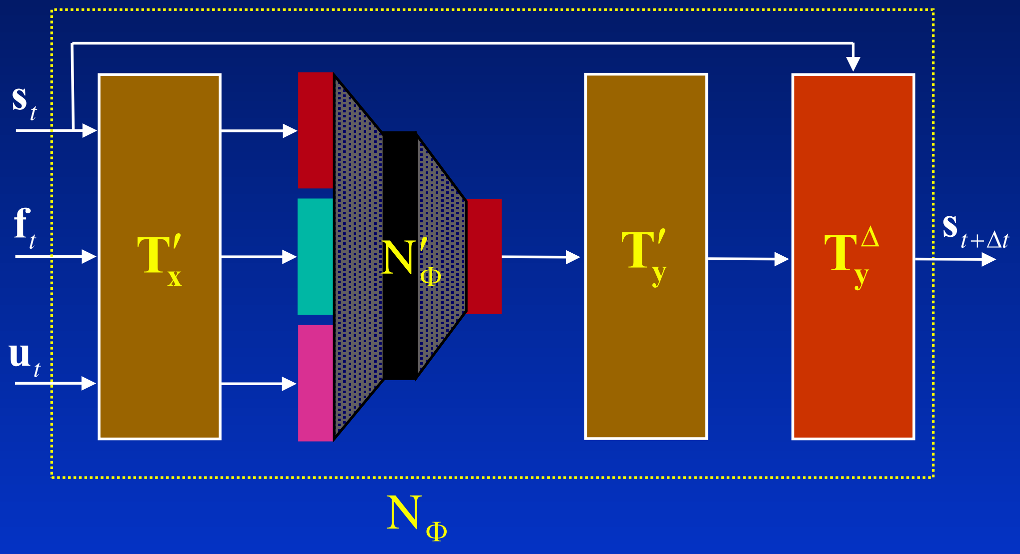


# Input & Output Transformations



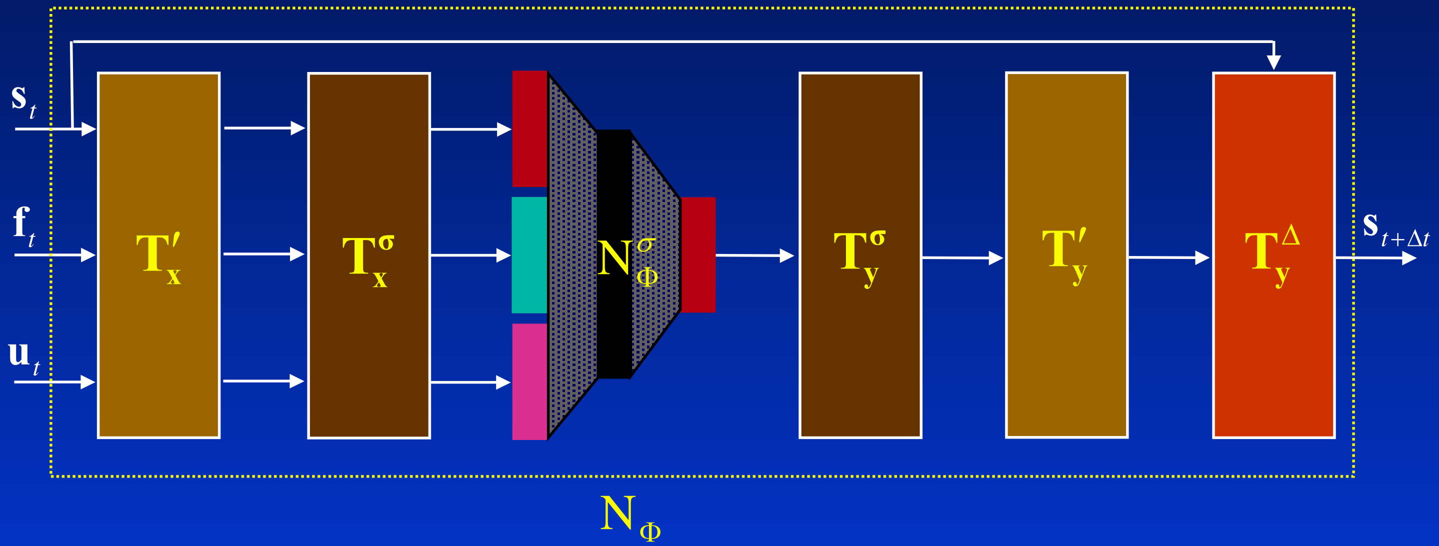

- Predict state changes

# Input & Output Transformations




- Predict state changes
- Invariance to translation and rotation

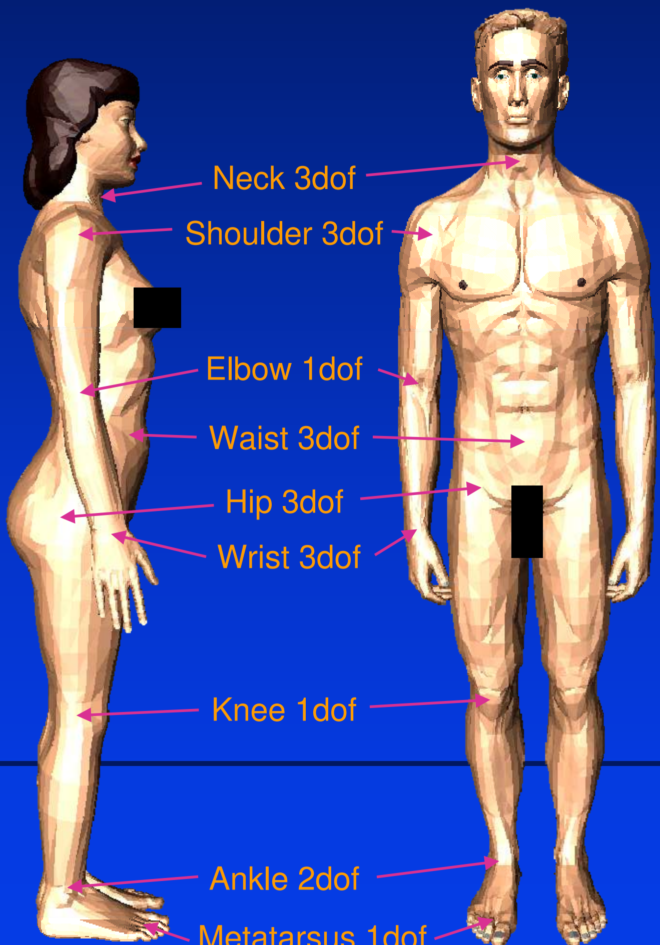
# Input & Output Transformations

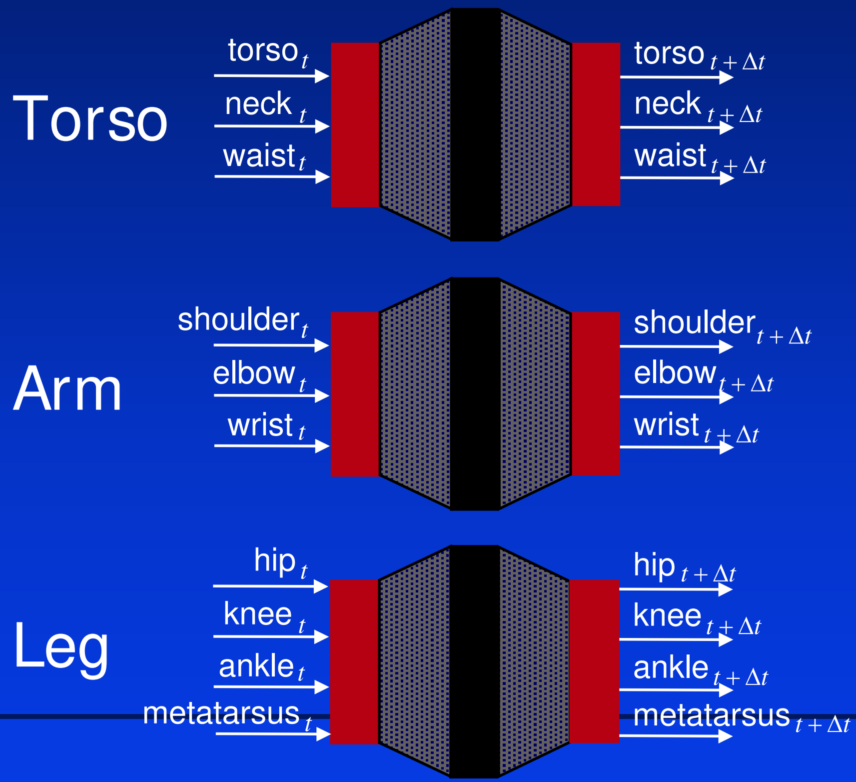



- Predict state changes
- Invariance to translation and rotation
- Normalize inputs and outputs

# Hierarchical Emulators

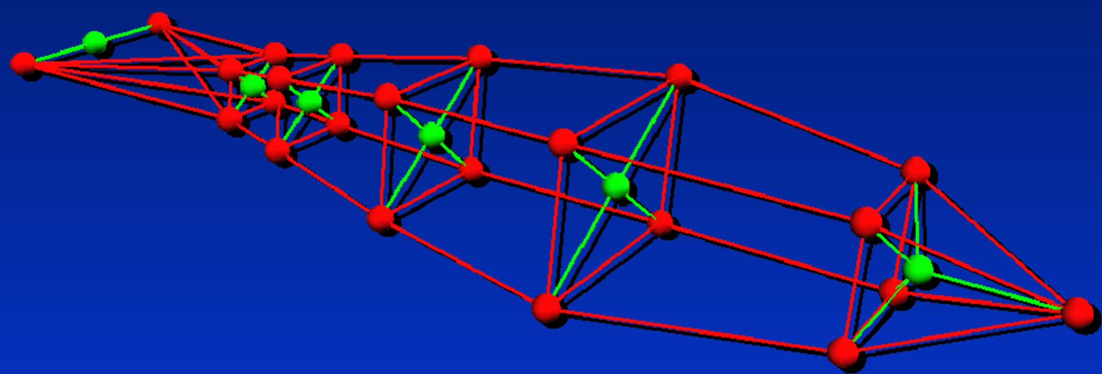
## *Human model*

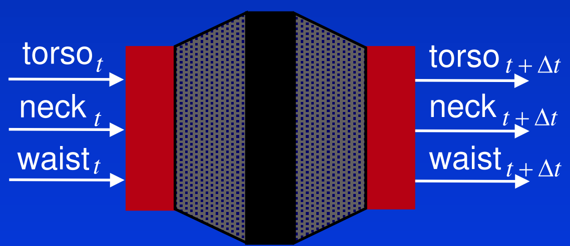


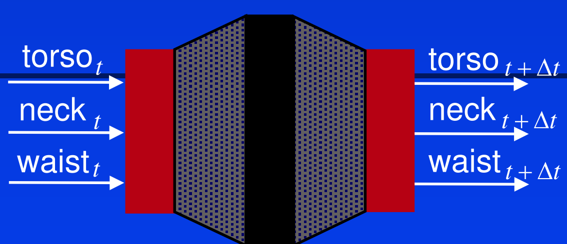



# Hierarchical Emulators

## *Dolphin model*





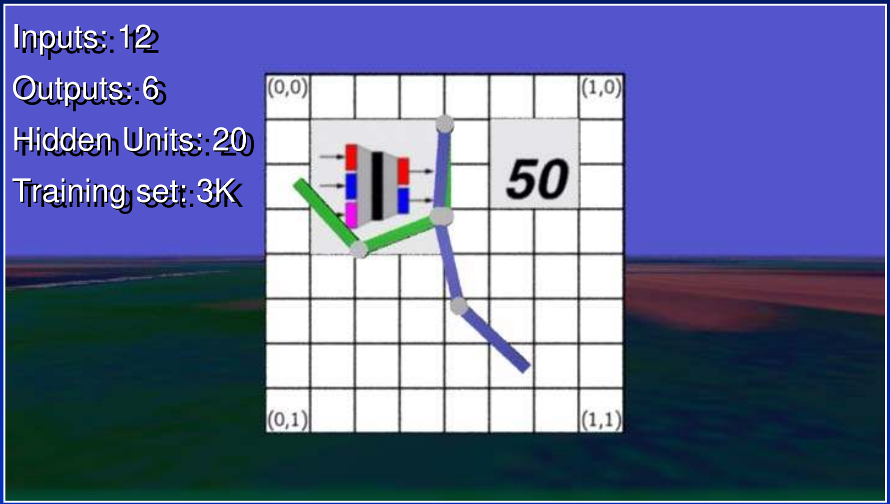



# Training NeuroAnimators

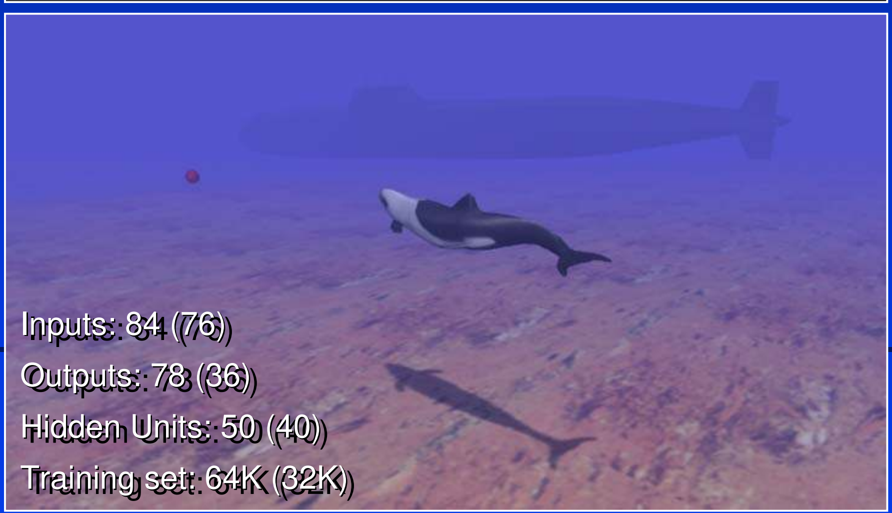
## *Offline backpropagation training of networks*

- *"Xerion"* public domain neural network simulator software from the University of Toronto
- Initialize networks with random weights
- Generate training examples with "short-time" physical model simulations from random initial conditions
  - *can reduce training times by sampling state, force, & control inputs that occur most often in practice*

# Example NeuroAnimators





# Emulation Examples

# Emulation Performance

## *Speedups for a NeuroAnimator with super-timestep $\Delta t = 50\,\delta t$*

- Passive pendulum 94.0x physical simulation
- Active pendulum 75.3x ”
- Truck 69.7x ”
- Lunar lander 53.7x ”
- Dolphin 66.3x ”
  - *approximation error holds ~steady with $\Delta t$*

# Talk Overview

- Introduction
- Artificial neural networks
- From physical models to NeuroAnimators
  - *emulation results*
- NeuroAnimator based controller synthesis
  - *control learning results*
- Conclusion and future work

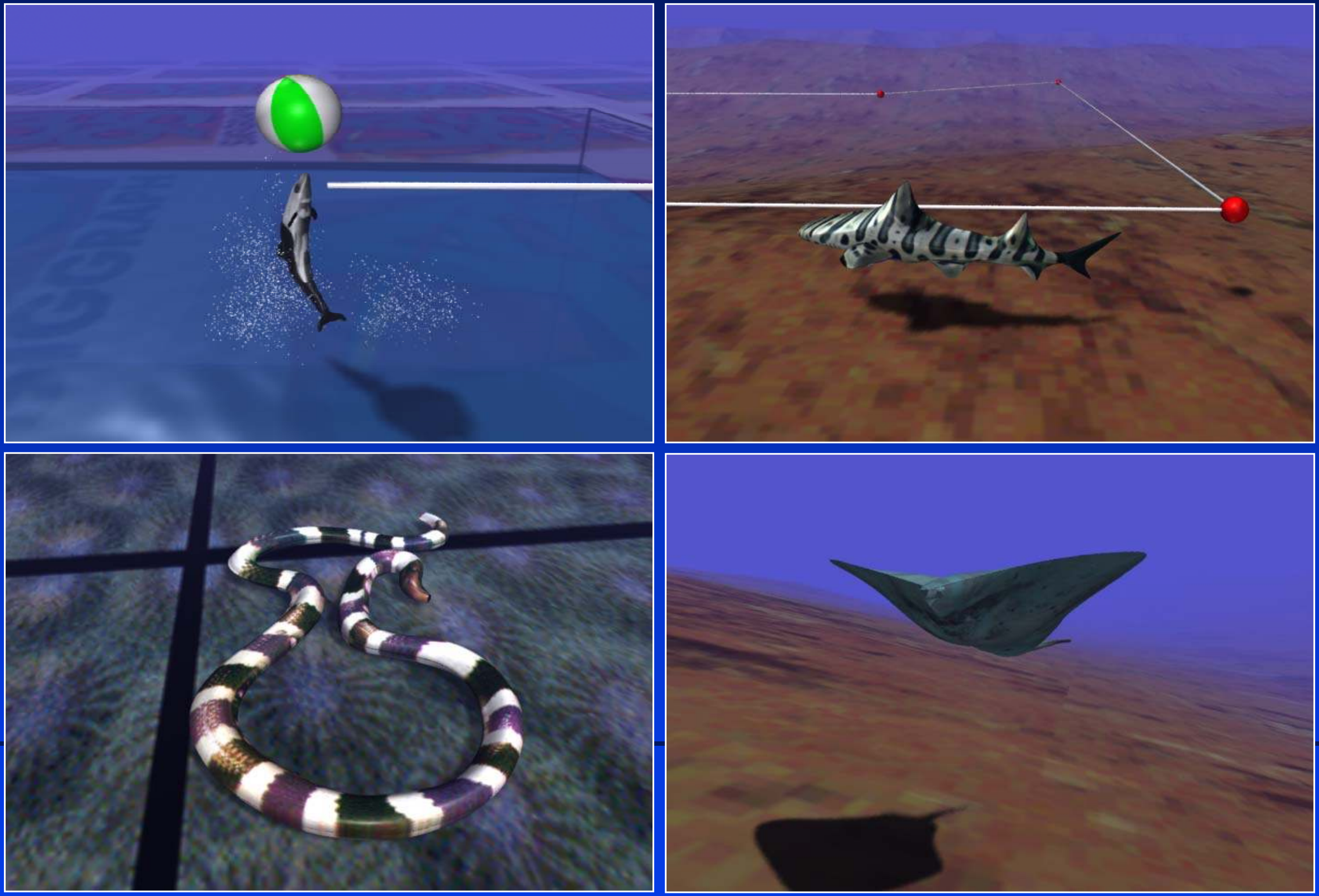
# Control of Physical Models

- Inverse dynamics
  (Isaacs87,Barzel88)
- Constraint optimization
  (Brotman88)
- Hand-crafted controllers
  (Miller88, Lee95, Tu94, Wilhelms87, Hodgins95)
- Controller synthesis
  (Goh88, Pandy92, Panne93, Ngo93, Grzeszczuk95)
- Connectionist robotic control
  (Mendel70, Werbos74, Barto87, Jordan88, Nguyen89 - “truck backer-upper”)

**Our approach**

# Controller Synthesis

## (Grzeszczuk & Terzopoulos 95)



# Controller Synthesis

## *Optimization of an objective function*

- Objective function

Controller quality

Motion quality

$$J(\mathbf{u}) = \mu_u J_u(\mathbf{u}) + \mu_s J_s(\mathbf{s})$$

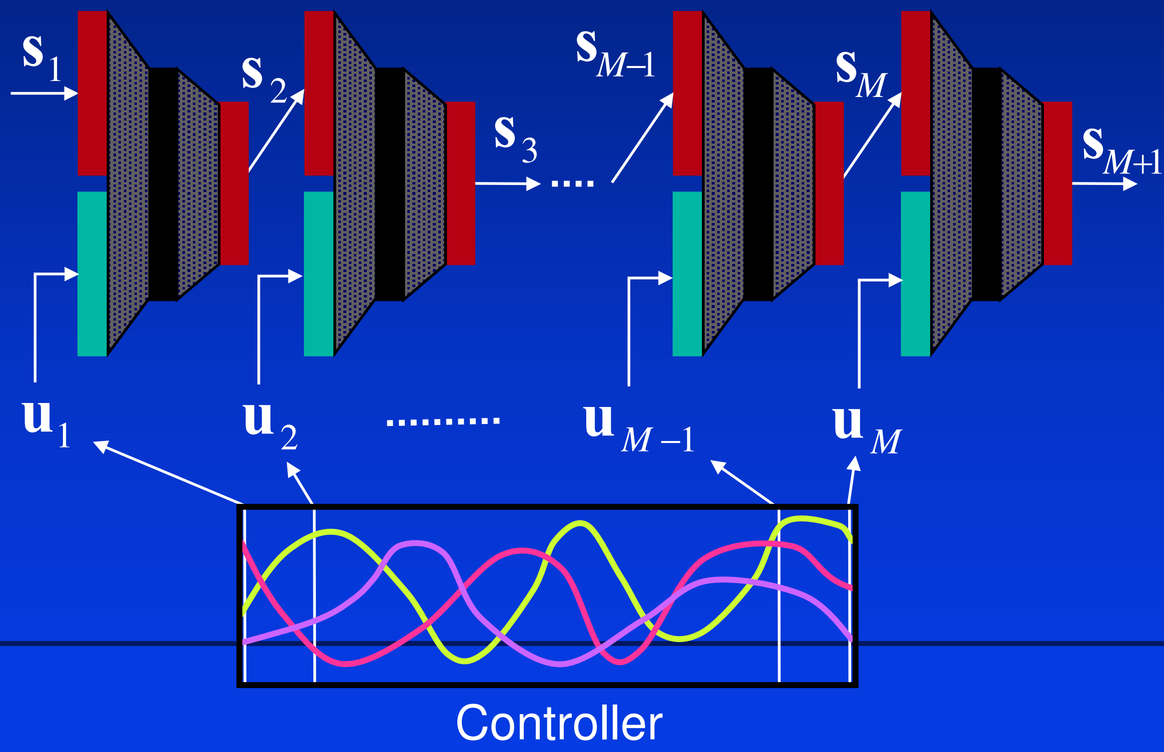
- Controller adjustment rule

$$\mathbf{u}^{l+1} = \mathbf{u}^{l} - \eta_x \underbrace{\nabla_{\mathbf{u}} J(\mathbf{u}^{l})}$$

- Trained NeuroAnimator yields gradient analytically
- Controller adjustment consists of two steps...

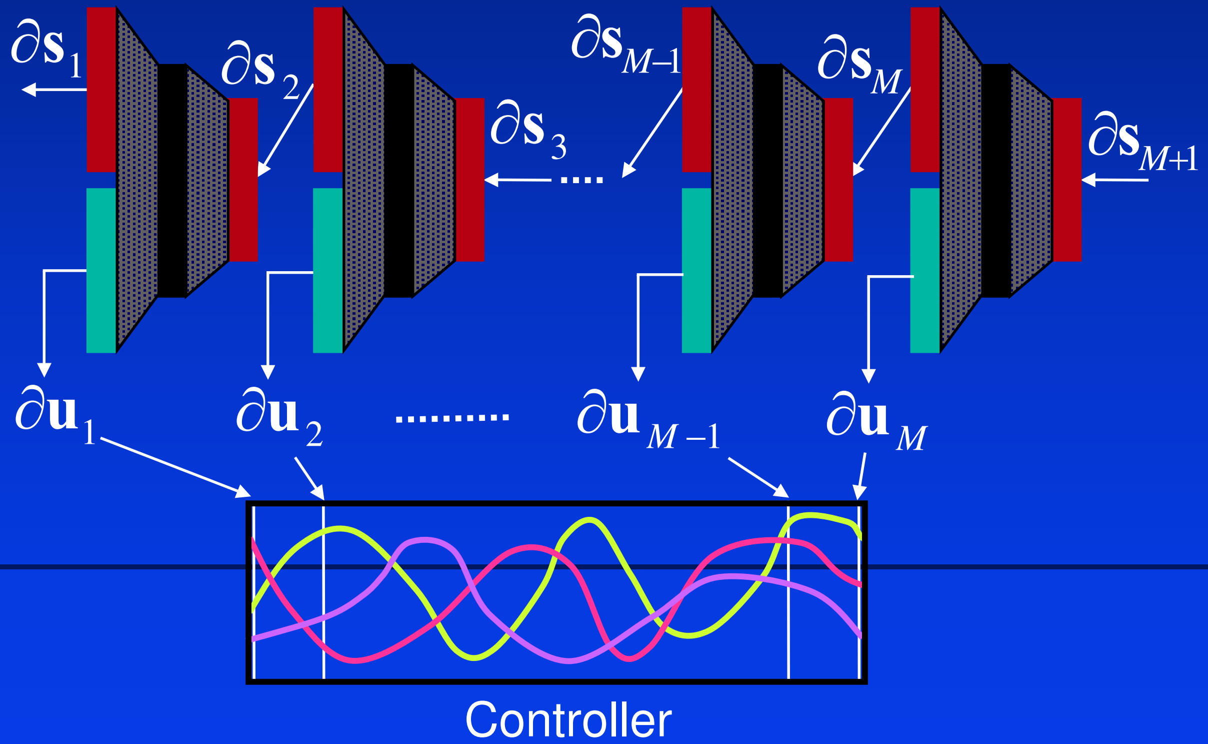
# 1) Forward Step

## *Emulates the forward dynamics*



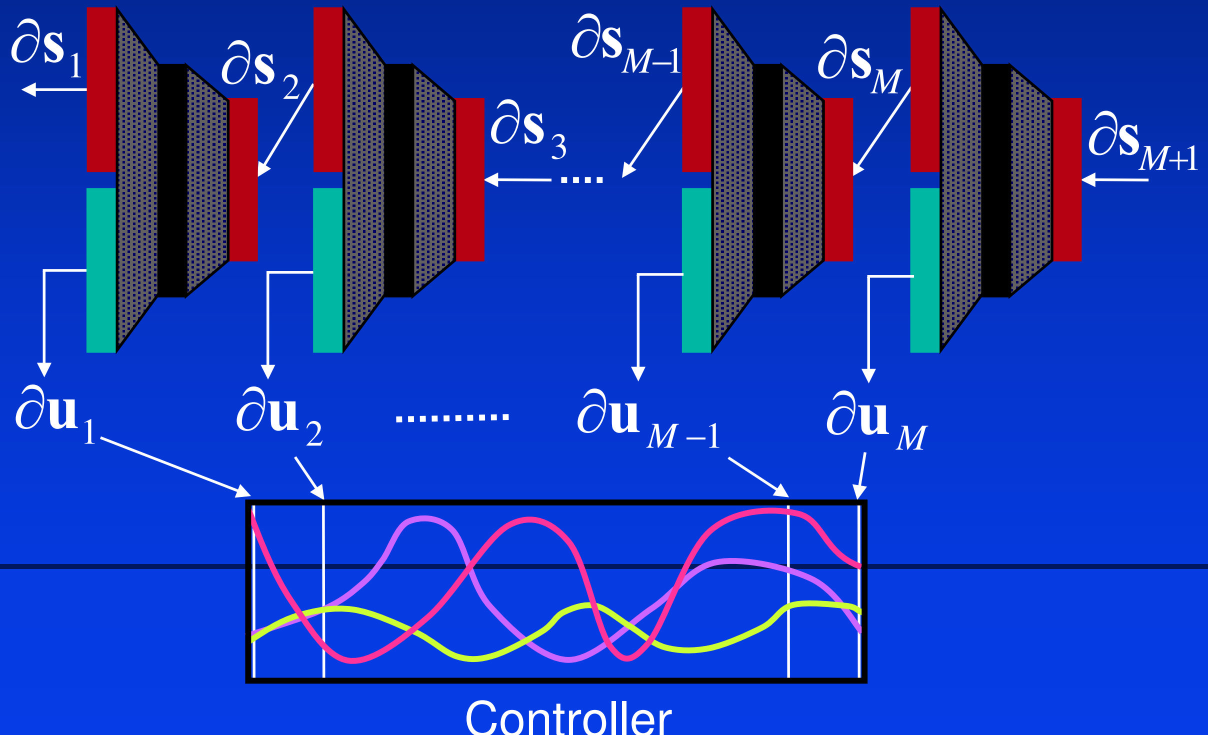
# 2) Backward Step

***Computes gradient using backpropagation through time***



# 2) Backward Step

***Computes gradient using backpropagation through time***



# Control Learning Results

# Controller Learning Performance

# Talk Overview

- Introduction
- Artificial neural networks
- From physical models to NeuroAnimators
  - *emulation results*
- NeuroAnimator based controller synthesis
  - *control learning results*
- Conclusion and future work

# Conclusion

***The NeuroAnimator can be a powerful complement to physics-based animation***

- NeuroAnimators accurately emulate various physical models up to 2 orders of magnitude faster than numerical simulation
- NeuroAnimator based controller learning algorithm synthesizes motions satisfying prescribed animation goals with up to 2 orders of magnitude fewer iterations

# Future Research

- NeuroAnimators for Artificial Life graphical characters
  - *acquiring “mental models” of dynamic worlds*
- NeuroAnimation by motion capture
  - *learning approximations of complex biomechanics*

- Connectionist controller representation
- Hierarchical emulation and control

# One More Thing...

***"The Eagle has Landed?"***

# Acknowledgements

## *Intel Corporation*

- Richard Wirt, Fellow & MRL Director
- Edward Langlois
- Steve Hunt
- Sonja Jeter
- Mike Gendimenico
- Michael Shantz, Dave Sprague
- Baining Guo
- John Funge
- Xiaoyuan Tu
- Alexander Reshetov
- Feng Xie
- Bob Liang

## *University of Toronto*

- Zoubin Ghahramani
- Michiel van de Panne
- Mike Revow
- Drew van Camp

## *Natural Sciences & Engineering Research Council of Canada*

Steacie Memorial Fellowship