Real-Time Motor Control using Recurrent Neural Networks

Dongsung Huh, and Emanuel Todorov

***Abstract*—In this paper, we introduce a novel neural network architecture for motor control. Our general framework employs a recurrent neural network (*RNN*) to govern a dynamical system (*body*) in a closed loop fashion. This hybrid system is trained to behave as an interpreter that translates high-level motor plan into desired movement. Our method uses a variant of optimal control theory applicable to neural networks. When applied to biology, our method yields recurrent neural networks which are analogous to circuits in primary motor cortex (M1) or central pattern generators (CPGs) in the spinal cord.**

I. Introduction

In recent years, there has been much progress in the study of biological motor control, both experimentally and theoretically. Numerous neural recording experiments have yielded insight into the neural substrate of motor control. Optimality principles of sensory motor functions have been successful in explaining behavior [1,2]. However, there is a lack of understanding of how the two are related. The field may benefit from neural network (*NN*) modeling that bridges the two; neural mechanisms and optimal control. In the present work, we introduce a novel theoretical framework that yields recurrent neural network (*RNN*) controllers capable of real-time control of a simulated body (e.g. limb).

Previously, neural network modeling has been used to understand the simple reflex system of leeches, based on detailed neural activity data in response to sensory stimuli [3,4]. Their *RNN* was trained to reproduce the recorded input-output (stimulus-neural response) mapping. However, there are two shortcomings to such approach: one technical, and another conceptual. As we model more complex systems the number of neurons involved and the repertoire of movement grow larger, so that obtaining detailed neural activity data becomes unrealistic. Moreover, it gives little intuition about how such neural activity ends up accomplishing the final goal of motor control.

Here, we use an alternative approach. Instead of reproducing desired *neural response*, a *RNN* is trained to directly control a simulated body in a closed loop to generate desired *movement*. However, the conventional *RNN* training paradigm does not deal with such situation where *RNN* is connected with a foreign object in interactive way. It is time to merge NN with more general optimal control theory.

However, the conventional optimal control theory has its own shortcoming too, that it optimizes performance for a single task only. On the other hand, most motor areas, including the primary motor cortex, subserve multiple motor-tasks under changing task environment, where the task information (or command) is given from the decision making (or motor-planning) centers of the brain. Therefore, it is appropriate to model the motor area as performing a general mapping between command sequences to movements, not a single motor task (fig 1). In section 2, we introduce a generalized optimal control framework which properly deals with command-movement mapping.

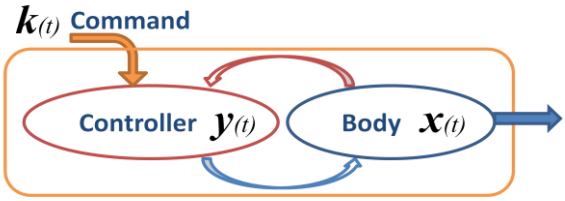


Fig 1. *Body* is a dynamical system with state $\boldsymbol{x}$. *Controller* receives time-varying command (task-information) $\boldsymbol{k}$ and controls *body* accordingly. The combined *body-controller* system can be seen as performing a mapping from command sequence $\{\boldsymbol{k}\}$ to *body*-state sequence $\{\boldsymbol{x}\}$, i.e. movement.

One of the major technical challenges for all mapping problems is that both the inputs and outputs may reside in a very high dimensional space. In order for the training to be possible, it is crucial to discover the underlying low dimensional structure of the data space. In section 4, we introduce *attractor-analysis-method* which reveals the low dimensional structure of the movement space. This method not only enables efficient training, but also an inspiring view point for understanding the motor control process.

TABLE I
Notation

Representation		meaning
bold, lower case	$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{q}$ $\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{k}$ $\boldsymbol{f}(\boldsymbol{x})$	State vector, or Signal vector, or Function with vector output
bold, upper case	$\boldsymbol{W}$	Matrix
normal, lower case	$V(\boldsymbol{q}), c(\boldsymbol{x}, \boldsymbol{u})$	Scalar, or scalar function
curly bracket	$\{\boldsymbol{x}(t)\}, \{\boldsymbol{x}\}$	Sequence of $\boldsymbol{x}(t)$ between initial and final time (t_i, t_f)
function with subscript	$\boldsymbol{f}_{\boldsymbol{x}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$	Partial derivative matrix, such that $d\boldsymbol{f} = \boldsymbol{f}_{\boldsymbol{x}} d\boldsymbol{x} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} d\boldsymbol{x}$

This work was supported by U.S. National Science Foundation.

Dongsung Huh is with University of California, San Diego, La Jolla, CA 92093 USA (corresponding author to provide phone: 858-822-3713; fax: 858-534-1128; e-mail: dhuh@ ucsd.edu).

Emanuel Todorov is with University of California, San Diego, La Jolla, CA 92093 USA (e-mail: todorov@cogsci.ucsd.edu).

In section 6, we show two examples of successful application of the method.

Our method yields recurrent neural networks which are analogous to circuits in primary motor cortex (M1) or central pattern generators (CPGs) in the spinal cord. Despite the biological motivation of this work, our methods are general and widely applicable to various engineering applications.

II. Modified Optimal Control Formulation

In this section, we formulate the mapping problem in the stochastic optimal control framework. It takes a dynamical equation of the *body-controller* system, and a cost function which describes the desired movement. We have modified the framework so that the desired motor task may change as function of high-level command, which is considered as stochastic event. Here, we choose a *RNN* as a model controller, but this framework applies to any parametric model in general. Note that a mapping problem is translation invariant in time, and therefore our cost function, dynamic equation and probability distribution do not have explicit time dependence.

A. General Model of Dynamical System

The model system is composed of a *body* and a *RNN* controller. *Body* is a stochastic dynamical system that can be described as a set of equations,

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) + noise && \text{state update} \\ \boldsymbol{s} &= \boldsymbol{s}(\boldsymbol{x}) + noise && \text{output} \end{aligned} \quad , \tag{1}$$

and *RNN* is also a stochastic dynamical system as described by the following equations,

$$\begin{bmatrix} \tau\dot{\boldsymbol{y}} + \boldsymbol{y} \\ \boldsymbol{u} \end{bmatrix} = \boldsymbol{\sigma}\left(\begin{bmatrix} & \boldsymbol{W} & \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{s} \\ \boldsymbol{k} \end{bmatrix} \right) + noise \quad , \tag{2}$$

where $\boldsymbol{x}$ is *body*-state, $\boldsymbol{y}$ is neural-state, $\boldsymbol{s}$ is sensory measurement from *body*, $\boldsymbol{k}$ is high-level command, $\boldsymbol{u}$ is low-level control signal, and $\boldsymbol{W}$ is a matrix of free parameters with appropriate dimension, $\boldsymbol{\sigma}$ is a transfer function that acts on each element of input vector one-by-one, $\boldsymbol{\sigma}(\boldsymbol{x})_i = \sigma(\boldsymbol{x}_i)$. $\boldsymbol{W}$ is usually referred to as the *synaptic weight matrix*.

Here, we will treat *body* and *RNN* as a single dynamical system with the following dynamical equation,

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{\omega};\boldsymbol{W})^{1} \tag{3}$$

where $\boldsymbol{q}_{(t)} = [\boldsymbol{x}_{(t)}; \boldsymbol{y}_{(t)}]$ is a combined state vector, and $\boldsymbol{\omega}_{(t)}$ is noise. This expression hides the fact that our controller is made of *RNN*. Thus, our method is independent from the nature of controller. Notice that we have taken the view that the system dynamics, which is originally stochastic, can be seen as deterministic once the noise is given as input [5].

B. Uncertainty

In order for a state trajectory $\{\boldsymbol{q}\}$ to be determined according to (3), initial state $\boldsymbol{q}_0$, command sequence $\{\boldsymbol{k}\}$, and noise sequence $\{\boldsymbol{\omega}\}$ should be given. The *controller-body* system operates in stochastic environment where these variables are probabilistically drawn from a distribution $\mathbb{P}(\boldsymbol{q}_0,\{\boldsymbol{k}\},\{\boldsymbol{\omega}\})$. Here we assume that $\mathbb{P}$ is known.

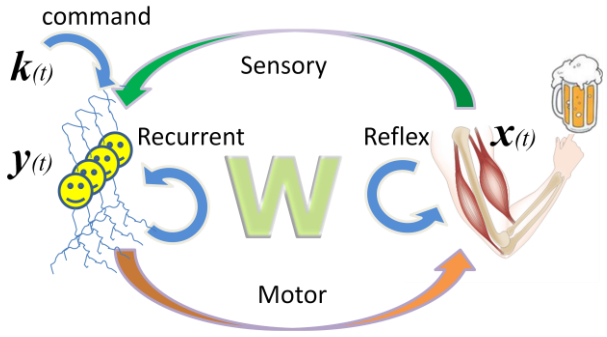


Fig. 2. Our model system consists of a dynamical body and a recurrent neural network, parameterized by a weight matrix W. There are five components of W: Recurrent, Sensory, Motor, Command, and Reflex connections.

C. Cost Function and Value Function

In optimal control framework, the desired mapping is described by an instantaneous cost function. A cost function evaluates current body-state and current control whose desirability is set by command:

$$\mathrm{c}(\boldsymbol{x},\boldsymbol{u};\boldsymbol{k})$$

which may be rewritten as $\mathrm{c}(\boldsymbol{q};\boldsymbol{k},\boldsymbol{W})$.

A value function, on the other hand, evaluates the entire movement trajectory. It is defined as the total accumulated cost along the trajectory $\{\boldsymbol{q}\}$ generated by (3):

$$V(\boldsymbol{q}_0,\{\boldsymbol{k}\},\{\boldsymbol{\omega}\};\boldsymbol{W}) = \int_{\{\boldsymbol{q}\}} c(\boldsymbol{q};\boldsymbol{k},\boldsymbol{W})dt. \tag{4}$$

Notice that value function is a function of $(\boldsymbol{q}_0,\{\boldsymbol{k}\},\{\boldsymbol{\omega}\})$ which are the determinants of trajectory $\{\boldsymbol{q}\}$.

D. Objective Function

For a deterministic problem, V *is* the objective function to be minimized. For a stochastic problem, however, an objective function should reflect the uncertainty $\mathbb{P}(\boldsymbol{q}_0,\{\boldsymbol{k}\},\{\boldsymbol{\omega}\})$. A popular objective function is the simple average $\mathbb{E}_{\mathbb{P}}[V]$. Such objective functions, however, only takes into account the mean but not the variability of V, and tend to leave some unoptimized outliers.

Instead, we define a risk-sensitive objective function, as introduced by [6],

$$A(\boldsymbol{W};\beta) = \frac{1}{\beta}\log \mathbb{E}_{\mathbb{P}}[\exp(\beta V)] \tag{5}$$

where β is a positive number which provides a link between optimality and robustness. In the limit $\beta \to 0$, $A(\boldsymbol{W})$ reduces to the usual average $\mathbb{E}_{\mathbb{P}}[V]$, and for $\beta \to \infty$, $A(\boldsymbol{W}) = \max_{\mathbb{P}}[V]$. Empirically, we have found that it is important to set β to an appropriate value for both faster optimization and robustness.

Our goal is finding W that minimizes A. This is exactly the same formulation as finding an optimal global control law. In

[1] Mathematicians would prefer more accurate notation $d\boldsymbol{q} = \boldsymbol{f}^{\boldsymbol{q}}(\boldsymbol{q};\boldsymbol{W})dt + d\boldsymbol{\omega}$. However, we consider $\boldsymbol{\omega}_{(t)}$ as one of the inputs to the dynamical system, and (3) emphasizes this view.

other words, we will directly approximate the global control law with *RNN*.

In practice, however, the expectation in (5) cannot be computed, because we do not have an explicit expression of V. Instead, we approximate it using sample average

$$\tilde{A}(\boldsymbol{W};\beta)=\frac{1}{\beta}\log\left[\frac{1}{N}\sum_{i=1}^{N}\exp(\beta V_i)\right], \qquad (5')$$

where $V_i = V(\boldsymbol{q}_{0i},\{\boldsymbol{k}\}_i,\{\boldsymbol{\omega}\}_i;\boldsymbol{W})$ is a sample of the value function drawn from $\mathbb{P}(\boldsymbol{q}_0,\{\boldsymbol{k}\},\{\boldsymbol{\omega}\})$. A large number of samples may be required for proper approximation, depending on the complexity of the body dynamics and dimensionality of $(\boldsymbol{q}_0,\{\boldsymbol{k}\},\{\boldsymbol{\omega}\})$. Note that the number of required samples is significantly decreased by focusing on the attractor dynamics, which has small dimensionality. (By focusing on the attractor dynamics, the number of required samples can be significantly decreased.)

III. Optimization

We use conjugate gradient descent method for optimization. First, we need to differentiate (5'):

$$\frac{d\tilde{A}}{d\boldsymbol{W}}=\frac{1}{N}\sum_{i=1}^{N}\exp\left(\beta V_i-\beta\tilde{A}\right)\frac{dV_i}{d\boldsymbol{W}}\cdot$$

Therefore, is $d\tilde{A}/d\boldsymbol{W}$ is a weighted sum of $dV_i/d\boldsymbol{W}$, where exponential weight. Here, $dV_i/d\boldsymbol{W}$ means differentiating (4) while holding all the random variables fixed at $(\boldsymbol{q}_{0i},\{\boldsymbol{k}\}_i,\{\boldsymbol{\omega}\}_i)$:

$$\frac{dV_i}{d\boldsymbol{W}}=\frac{d}{d\boldsymbol{W}}V(\boldsymbol{q}_{0i},\{\boldsymbol{k}\}_i,\{\boldsymbol{\omega}\}_i;\boldsymbol{W})\cdot$$

The differentiation in (9) is not simple to calculate, because change in W has complex effect in the state trajectory {q}. Pontryagin solved this problem using the *minimum action principle* of classical mechanics. Here we introduce a modified version.

A. Modified Pontryagin's Maximum Principle

Define Hamiltonian as

$$H(\boldsymbol{q},\boldsymbol{p};\boldsymbol{W})=\boldsymbol{p}^{\boldsymbol{T}}\cdot\boldsymbol{f}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{\omega};\boldsymbol{W})-c(\boldsymbol{q},\boldsymbol{k};\boldsymbol{W})$$

Then gradient of value function is

$$\frac{\partial V}{\partial \boldsymbol{W}}=\int_{\{\boldsymbol{q}^*,\boldsymbol{p}^*\}}-\frac{\partial H}{\partial \boldsymbol{W}}dt=\int_{\{\boldsymbol{q}^*,\boldsymbol{p}^*\}}\left[-\boldsymbol{p}^{\boldsymbol{T}}\bullet\boldsymbol{f}_{\boldsymbol{W}}+c_{\boldsymbol{W}}\right]dt$$

where integration is along the special trajectory $\{\boldsymbol{q}^*,\boldsymbol{p}^*\}$ which is determined by the following dynamics,

$$\boldsymbol{q}^* = \partial H/\partial \boldsymbol{p}^{\boldsymbol{T}}=\boldsymbol{f}\qquad,\ \boldsymbol{q}^*(t_i)=\boldsymbol{q}_0$$

$$-\dot{\boldsymbol{p}}^{*\boldsymbol{T}}=\partial H/\partial \boldsymbol{q}=\boldsymbol{p}^{*\boldsymbol{T}}\bullet\boldsymbol{f}_{\boldsymbol{q}}-c_{\boldsymbol{q}},\ \boldsymbol{p}^*(t_f)=\vec{\boldsymbol{0}}$$

Note that, {q*} is the actual trajectory as generated by (3). {p*} can be obtained by integrating the lower equation backward in time, and it represents backward propagation of cost gradient.

This method is called Pontryagin's Maximum Principle [7]. If the system of interest purely consists of *RNN*, this method reduces to the famous Back-Propagation-Through-Time method [ref:Werbos].

IV. Attractor Dynamics Analysis

One of the major technical challenges for all mapping problems is that both the inputs and outputs may reside in a very high dimensional space. In order for the training to be possible, it is crucial to discover the underlying low dimensional structure of the data space. We introduce the *attractor analysis method* (section 5), which mitigates the curse and leads to efficient optimization. It also sets a nice framework for understanding the movement generation process.

A. Size of Movement Space

The brute-force approach to the mapping problem would be to learn one-to-one match between all command sequence and the desired movement $(\{\boldsymbol{k}\},\{\boldsymbol{x}\})$. In this view, learning seems hopeless. Consider $\boldsymbol{X}(\ni \boldsymbol{x})$, the body-state space, and $\boldsymbol{M}(\ni\{\boldsymbol{x}\})$, the space of all possible movement between interval (t_i,t_f). The size, or cardinality of $\boldsymbol{M}$ grows exponentially in time, $|\boldsymbol{M}|=|\boldsymbol{X}|^{1+\Delta t/t_0}$, where $\Delta t=t_f-t_i$ is the movement duration, and $t_0(\cong 1\text{ s})$ is the movement correlation time. Then it would require huge number of sample pairs $(\{\boldsymbol{k}\},\{\boldsymbol{x}\})$ in order to approximate (5').

Fortunately, such exponential growth is illusionary, because it includes much redundancy. The actual complexity of mapping problem is much lower dimensional. In order to discover the low dimensional structure of the movement space, we analyze the attractor dynamics of the desired system.

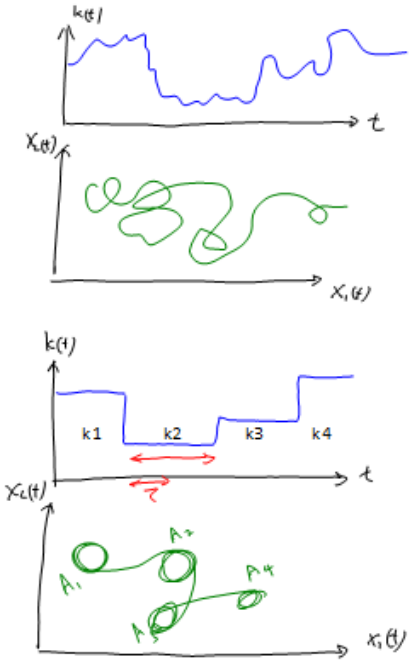


Fig. 3ab. (a) Command dynamics is continuous, which reflecting the actual dynamics. (b) Command dynamics is discretized in to a series of jumps, with duration longer than τ_c. This gives enough time for body-state to converge to attractors during each static period.

B. *Unique Attractor Property of Desired Mapping*

By default, we assume the desired mapping to have stability and memoryless properties for reasons explained below.

Stability means that if command input is held fixed k(t)=k0 for t>t0, then the body-state will converge to a set of states A(k0) after enough time (see fig 3b). Such a set is called an attractor. Let us assume that the convergence would take at most τ_c – the convergence time constant.

Memorylessness means an attractor is uniquely determined by the current command k0, regardless of the past command history or state history before t0.

Together, these properties guarantee that for each command, there exists a unique attractor state to which the body-state converges, and they create a one-to-one mapping between command and state, thereby keeping the mapping simple and intuitive.

Geometrically, an attractor can be a point (0d), a periodic orbit (1d, limit-cycle), a manifold (higher dimension), or chaotic (fractional dimension). In this paper, we deal only with the first two simple types.

C. *Reducing Dimensionality*

Rather than attacking the full mapping problem (fig 3a), we focused on a simplified version where the command sequence is discretized into a sequence of jumps and the intervals between jumps are constrained to be longer than τ_c (fig 3b). In this case, the desired movement is a series of transition from one attractor to another. Then, for $\boldsymbol{N}$ number of such transitions, the movement space has the size $|\boldsymbol{A}|^{N+1}$ for point-attractors, or $|\boldsymbol{A}|^{N+1}|\boldsymbol{\Theta}|^{\boldsymbol{N}}$ for limit-cycles, where $|\boldsymbol{A}|$ is the size of attractor space (related to dimensionality of $\boldsymbol{k}$) and $\boldsymbol{\Theta}$ is the space of limit-cycle phases (between 0 and 2π).

Now the redundancy becomes obvious here. Because of time invariance nature and memoryless property of each transition, optimizing a series of jumps between attractors is equivalent to optimizing a parallel batch of single jumps.[2] In other words, if we optimize the *controller* for single jumps, which we do (fig 4), it will behave optimally for series of multiple jumps, too. Therefore, the true complexity of the mapping problem is $|\boldsymbol{A}|^2$, or $|\boldsymbol{A}|^2|\boldsymbol{\Theta}|$.

This approach is analogous to impulse response analysis, or step response analysis for linear systems, which captures (most)(full) information of the system's dynamical properties in a compact manner. On the other hand, the brute-force approach is highly redundant which causes inefficient learning.

V. EXAMPLES

We trained *RNN* to control a realistic human arm-model for two types of attractor dynamics. The first type is reaching, which is interpreted as transition between point attractors. This simple dynamics is likely to underlie biological (human) arm control, and there are lots of neurophysiological data to be explained regarding this dynamics. Even more striking is revealed when the method is applied to learning the limit-cycle dynamics – circle drawing task. Such periodic dynamics would be more relevant to locomotive systems than an arm, but it clearly shows what our method is capable of.

[2] In terms of statistical physics, we are substituting time-average with ensemble-average.

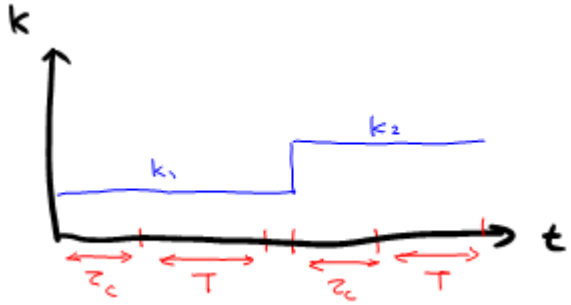


Fig. 4. Training movement sample :
A traning movement consists of two attractors and transition between them. For each attractor, the command signals is held fixed for duration longer than $\tau_c + T$, where τ_c is the convergence time constant, and T is the oscillation period for limit-cycle attractors, or 0.5 seconds for point-attractors. At initial time t=0, body-state is set to be on the first attractor, and neural-state is set randomly.

This approach focuses on accurately learning the attractor states, and learning the correct transition (convergence) movement toward the attractors.

A. *Model System - Specification*

1) *Two-Link Arm Model*

The body plant we use is a simulated, two degree-of-freedom arm restricted to horizontal movements (fig 5a). Actuation of the arm is performed by three pairs of opposing muscles (flexor/extensor). Two pairs of muscles individually actuate the shoulder and elbow joints, while the third pair actuates both joints simultaneously. Control of the arm is achieved by activating each muscle.

The arm state is represented by a 10 dimensional vector: $\boldsymbol{x} = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, a_1, a_2, a_3, a_4, a_5, a_6]^T$, where θ_1 is the shoulder joint angle, θ_2 the elbow angle, and a_i is activation level of each muscle. The activation level dynamics is approximated by a first order model: $\tau\dot{a}_i + a_i = \boldsymbol{u}_i$, where $\tau = 50ms$, and $\boldsymbol{u}$ is the control signal.

The tension generated by each muscle is a linear function of its activation level, and it also has non-linear dependence on the muscle length and speed (see fig 5b). $T_i = a_i \cdot T(l_i, \dot{l}_i)$.

Then joint torque is calculated as a function of muscle tension and joint angles. The actual arm movement is driven by the net joint torques plus the Coriolis force induced by movement velocity.

2) *Sensory Signal and Command*

In this model, sensory signal includes length, speed, and tension of each muscle: $\boldsymbol{s(x)} = \boldsymbol{[l; \dot{l}; T]}$.

In (2), note that there exist a direct route for the mapping from $\boldsymbol{s}$ and $\boldsymbol{k}$ to $\boldsymbol{u}$ that does not involves the *RNN*. The direct route from $\boldsymbol{s}$ to $\boldsymbol{u}$ imitates the mono-synaptic sensory-motor reflex loops. However, there is no biological evidence for a direct channel between $\boldsymbol{k}$ to $\boldsymbol{u}$ and thus it is removed from our model.

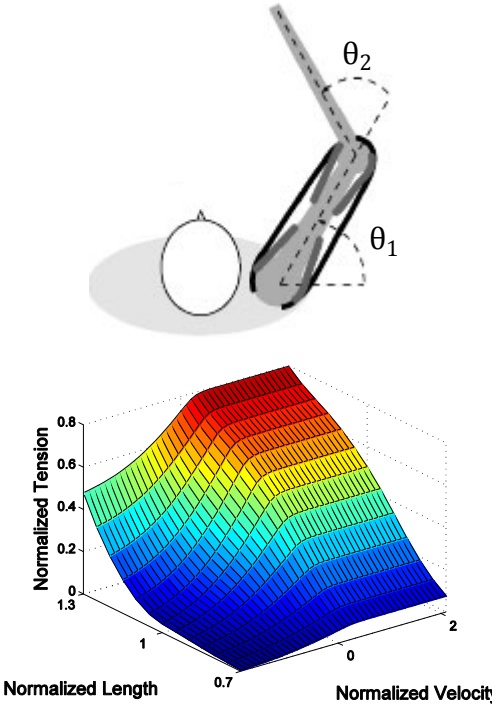


Fig 5. (a) arm model, showing how the muscles are connected to bones around the joints. (b) Tension generation function of muscle: Tension is a linear function of activation level but non-linear with respect to length and speed. The activation level was set to 1 for the above plot.

3) RNN Transfer Function $\boldsymbol{\sigma}$

In principle, neurons and muscles may process the information differently and may have different transfer functions: $\boldsymbol{\sigma_y}$, $\boldsymbol{\sigma_u}$. For muscles, we use soft-threshold-linear function $\boldsymbol{\sigma_u} = [\cdot]^+$, which properly rectifies negative output. There are more choices for the neuron transfer function $\boldsymbol{\sigma_y}$, such as sigmoid or other monotonically increasing functions. I found little difference in computational power and learning speed among those $\boldsymbol{\sigma_y}$'s given their dynamic range was properly normalized. In the end, I ended up using $\boldsymbol{\sigma_y} = [\cdot]^+$ because its piece-wise linear property makes the *RNN*'s dynamical properties simple to understand for future analysis.

4) Process Noise

In this model, process noise is modeled as multiplicative noise acting on sensory, command, and control signals. Therefore, each element of $\boldsymbol{u}$ is replaced with its noisy version $\boldsymbol{u}_i \cdot (1+\boldsymbol{\omega}_i^{\boldsymbol{u}})$, and $\boldsymbol{s}_i$, $\boldsymbol{k}_i$ with $\boldsymbol{s}_i \cdot (1+\boldsymbol{\omega}_i^{\boldsymbol{s}})$, $\boldsymbol{k}_i \cdot (1+\boldsymbol{\omega}_i^{\boldsymbol{k}})$. We modeled $\boldsymbol{\omega}_i$ as white Gaussian noise, with zero mean and 0.2 standard deviation (20% noise level).

B. Desired Dynamics

1) Point-Attractor: Reaching

We trained our system to perform reaching movements to randomly appearing target positions. Here, attractors are point-attractors.

The *RNN* receives a desired target location (in Cartesian coordinate) as command signal ($\boldsymbol{k} = \boldsymbol{TARGET}$). This is consistent with the experimental observation that neurons in motor cortex have cosine-tuning for target location, which implies Cartesian representation [8].

The instantaneous cost function is

$$c(\boldsymbol{x}, \boldsymbol{k}, \boldsymbol{u}) = 1 - \exp\left(-\frac{d^2}{2{d_0}^2}\right) + \frac{\alpha}{2}\frac{\|\boldsymbol{v}\|^2}{{v_0}^2}\exp\left(-\frac{d^2}{2{d_0}^2}\right) + \frac{\gamma}{2}\|\boldsymbol{u}\|^2, \quad (6)$$

where $d \equiv \|\boldsymbol{h} - \boldsymbol{TARGET}\|_2$ is the Euclidean distance between the hand ($\boldsymbol{h}$) and $\boldsymbol{TARGET}$, $d_0 \cong 0.04\,m$ is the target size, $\boldsymbol{v}$ is the hand velocity, $v_0 \cong 0.3\,m/s$ is the typical movement speed, and α, γ are appropriate mixing coefficients. For the distance cost, we used a saturating function (inverse of Gaussian) rather than the popular

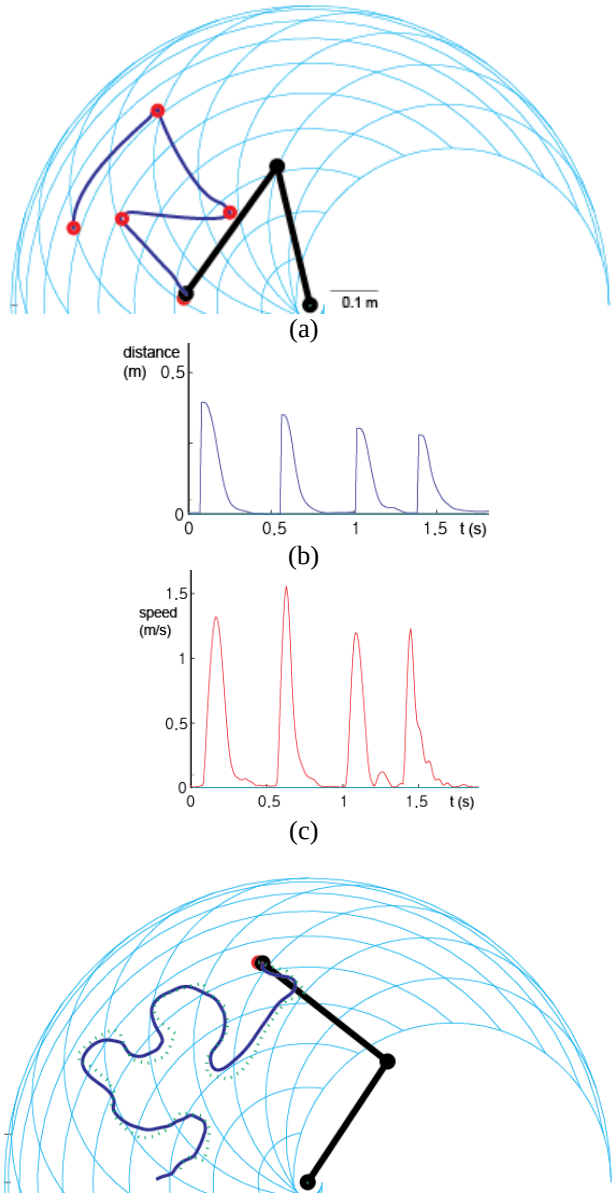


Fig 6. (a) Trajectory of reaching movement for four consecutive targets during testing. Open circles represent target locations. Thick line represent the arm. The background curves are joint-coordinate grids.
(b) Distance to target
(c) Speed of movement
(d) Tracking movement for continuously moving target. Target trajectory is drawn as a dotted line and hand trajectory is a solid line. Thick line represent the arm.

quadratic. Such saturating cost is necessary in order to describe reaching-task in a time-independent manner. The velocity cost is active only near ***TARGET***.

The space of point-attractors is 2 dimensional, and the space of reaching movements is 4 dimensional.

For batch training, 200 movement samples were used for approximating (5') with *RNN* of size 12 neurons. The larger the *RNN*, the more training samples were needed to prevent over-fitting. ***TARGET***s were uniformly distributed in space within reaching range.

We first tested the system with a series of discrete target jumps (fig 6(a,b,c)). The arm successfully generated a series of reaching movements to a given target locations with near straight movement. The curvature is due to arm dynamics and non-isotropic distribution of inertia. Distance to target indeed decreases to near zero, and the speed plot matches well with the bell-shaped profile recorded from human reaching movements.

When the target was allowed to make continuous movement, the *arm-RNN* system naturally generalized to perform trajectory following task (fig 6(d)).

2) Circle Drawing: limit-cycle

We also applied our method to limit-cycle mapping problem, where the task is to draw circles whose desired location, radius and rotation speed are given by the command signal: $\mathbf{k}=[\boldsymbol{CENTER}; radius; v_d]$. We define positive v_d as clockwise rotation and negative v_d counter-clock.

The instantaneous cost function is

$$c(\boldsymbol{x},\boldsymbol{k},\boldsymbol{u}) = \left(1-\exp\left(-\frac{d^2}{2{d_0}^2}\right)\right)\left(1+\frac{\alpha}{2}\frac{{v_d}^2}{{v_0}^2}\right) + \frac{\alpha}{2}\frac{\|\boldsymbol{v}-\boldsymbol{v_d}\|^2}{{v_0}^2}\exp\left(-\frac{d^2}{2{d_0}^2}\right)+\frac{\gamma}{2}\|\boldsymbol{u}\|^2 \qquad (7)$$

where $d \equiv \|\boldsymbol{h}-\boldsymbol{CENTER}\|_2 - radius$ is the "distance" between hand and circle, and $\boldsymbol{v_d}\,(=v_d\boldsymbol{n})$ is the desired velocity in the tangent direction ($\boldsymbol{n}$) of the circle. Notice that this cost function reduces to the reaching cost when both $radius$ and v_d are set to zero.

Notice that this cost function can easily be generalized to limit-cycles of different shapes, simply by redefining d to be the distance between the new orbit and hand, and $\boldsymbol{v_d}$ to be tangent to that orbit.

The space of all limit-cycle attractors is 4 dimensional, and the space of circle-drawing movements is 17 dimensional. This is much more difficult task than the reaching-task.

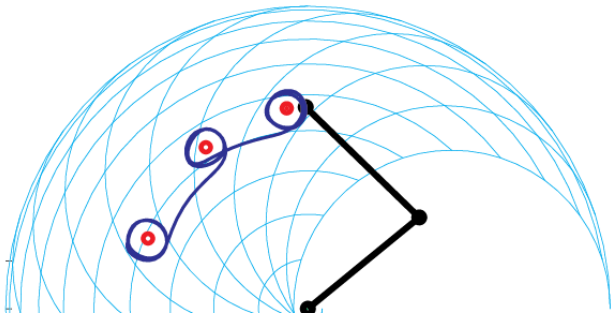
400 movement samples are used with *RNN* size of 36 neurons.

The *arm-RNN* system was tested for when the circle made a sequence of discrete jumps (fig 7(a)), when the radius and rotation speed input were varied (fig 7(b)). In all cases, the system successfully made desired circle drawing movements. Fig 7(b) different command sequences. It can also generate various complex shapes by appropriately planning the command signal – for example figure 8 shapes (fig 8)

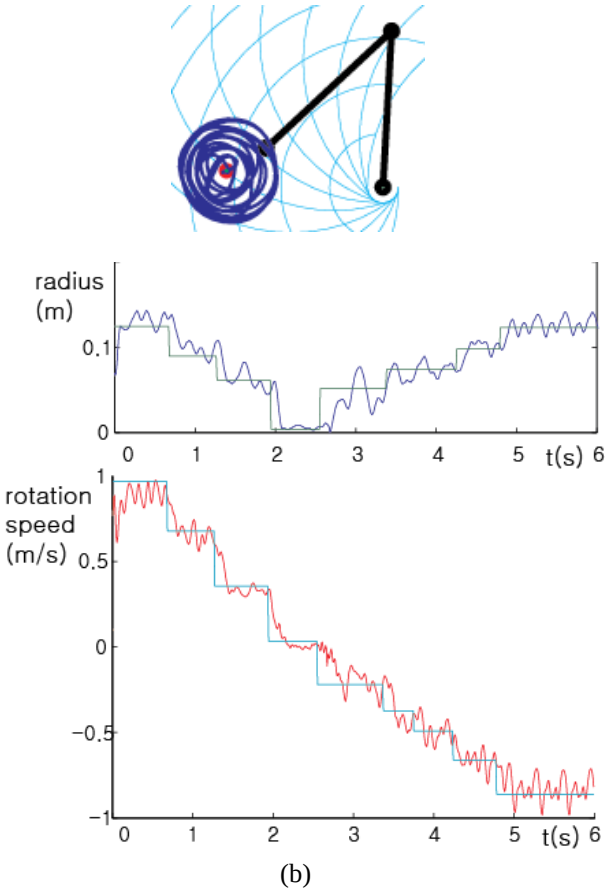
Limit-cycle attractors is more relevant to locomotion rather than arm control. Once we develop appropriate leg models, we would like to train the leg-RNN system for locomotive behavior. Indeed, locomotion would be an easier problem than the circle-drawing task, because its attractors will be uniquely determined by walking speed and step length, which is only 2 dimensional (and 5 dimensional movement space).

VI. Conclusion

In this paper, we have presented a novel use of *RNN*s as part of a hierarchical structure for controlling a non-linear *body*. Our novel optimization paradigm was efficient in training a *body-RNN* system with a desired mapping for both



(a)



(b)

Fig 7. Trajectory of circle drawing task (a) for multiple center locations. (b) varying radius and rotation speed condition, fixed center. Positive rotation sleep means clockwise rotation.

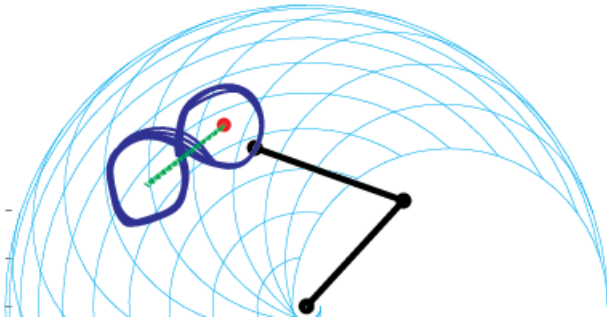


Fig 8. Figure 8 drawing task can be easily done by alternating between two center locations and opposite rotational directions.

point-attractor or limit-cycle based mapping.

When the command input is fixed, this *RNN* operates as an optimal controller that performs a single motor task (e.g. bringing the arm to a specific location). In more general view, however, it is better understood as an interpreter: the *RNN* translates a user-(homunculus)-friendly-neural-code into a muscle-friendly-code. The former is suitable for abstract motor planning, while the latter is more for detailed motor execution. For non-ambiguous translation, existence of a unique attractor for every command is a necessary feature, which is achieved by stability and memoryless properties.

This interpreter view then inevitably leads to another view that attractor dynamics is the basis for all movement generation. Is this true in biological systems?

Existence of limit-cycle attractors in spinal CPGs (such as the locomotive CPG) has been well recognized. In the primary motor cortex (M1, such as arm control area), micro-stimulation experiments have shown existence of point-attractor dynamics. Stimulating a population of neurons drives the limbs toward a unique postures (regardless of the previous body state), depending on the site of stimulation [9].

Moreover, in our preliminary comparison with monkey neurophysiological data, our RNN activity during reaching movement turns out to have very similar pattern to the data [to be published elsewhere]. Therefore, our RNN controller could be a good model of biological motor control systems.

Our view of attractor-based-control is related to the old equilibrium-point-hypothesis, which says that brain controls body movement by setting an equilibrium point at a desired limb position. In our model, the equilibrium point is substituted with an attractor that is elaborately generated by *RNN*.

Our model may also be interpreted in terms of synergy something., which says groups of muscles gets recruited together for certain motor tasks. In our model, such muscle groups may be related to the attractor basis.

This work is the first step toward understanding the whole hierarchy of biological motor control, and it may be comparable to modeling the hierarchy of sensory systems.

REFERENCES

[1] Optimality principles in sensorimotor control (review) Todorov E (2004). *Nature Neuroscience* 7(9): 907-915

[2] Optimal feedback control as a theory of motor coordination Todorov E and Jordan M (2002). *Nature Neuroscience* 5(11): 1226-1235

[3] Fang, Y.; Sejnowski, T. J.; Faster Learning for Dynamic Recurrent Backpropagation, Neural Computation, 2,270-273, 1990

[4] Lockery, S. R.; Fang, Y.; Sejnowski, T. J.; A Dynamic Neural Network Model of Sensorimotor Transformations in the Leech, Neural Computation, 2, 274-282, 1990

[5] PEGASUS: A policy search method for large MDPs and POMDPs, Andrew Y. Ng and Michael Jordan. In *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference*, 2000

[6] Jacobson, D.H. (1973). Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games. IEEE Trans. Automatic Control. AC-18: 124–131

[7] Iyanaga, S. and Kawada, Y. (Eds.). "Pontrjagin's Maximum Principle." *Encyclopedic Dictionary of Mathematics.* Cambridge, MA: MIT Press, pp. 295-296, 1980.

[8] Georgopoulos, A., Kalaska, J., Caminiti, R., & Massey, J. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience*, *2*(11), 1527–1537

[9] Graziano, M.S.A., Taylor, C.S.R. and Moore, T., 2002. Complex movements evoked by microstimulation of precentral cortex. *Neuron* **34**, pp. 841–851