In Proceedings of the 25th Annual International Conference of the
IEEE Engineering in Medicine and Biology Society, IEEE (2003)

Unsupervised Learning of Sensory-Motor Primitives

E. Todorov[1], Z. Ghahramani[2]
[1]Department of Cognitive Science, University of California San Diego, USA
[2]Gatsby Computational Neuroscience Unit, University College London, UK

Abstract—**The search for motor primitives has captured the attention of researches in both biological and computational motor control. Yet a theory of how to construct such primitives from first principles is lacking. Here we propose to do that by building a compact forward model of the sensory-motor periphery via unsupervised learning. We also propose a method for probabilistic inversion of the forward model, which yields low-level feedback loops that can simplify control. The idea is applied to simulated biomechanical systems of varying levels of detail.**

Keywords—**Unsupervised learning, motor primitives**

I. INTRODUCTION

Investigators of motor behavior have long been looking for motor primitives, or building blocks of movement. Candidates for such primitives include muscle synergies, spinal force fields, basis functions for representing internal models, small pieces of endpoint trajectories. So far the search has been predominantly data-driven: experimenters collect data (muscle activity, finger position, hand posture), apply some variant of principal components analysis, and declare anything that comes out to be a "motor primitive". Unfortunately little independent evidence exists that the principal components correspond to anything real. Even if movements were generated by combining some low-level primitives, this bottom-up approach will not necessarily identify them unless they were sampled independently during the data collection period. But independent sampling is very unlikely: since the high-level control system is trying to achieve an overall behavioral goal, the primitives are most likely being recruited in a coordinated manner. Thus, decomposition algorithms in the absence of any prior knowledge are likely to confuse two sources of structure: one originating form the primitives themselves, and the other originating from the controller that coordinates them.

From a computational perspective building blocks of movement may seem less important, since in principle an optimal controller can be found using reinforcement learning, without any prior knowledge or control structure. In practice however such controllers cannot be found for high-dimensional continuous-state systems. Therefore a number of investigators have focused on building various low-level control structures before applying reinforcement learning. The problem is how to build such a structure in the first place. It cannot come from considerations of optimal control, since it has to exist before optimal control can even be attempted in a high-dimensional state space. Presently it comes from intuition about the specific problem. But what we really need is a way to automate that intuition and make it a part of the learning algorithm.

So in both biological and computational motor control, a theory is needed that explains how *good* motor primitives can be constructed from first principles. The presumed function of motor primitives is to simplify a complex control problem, by reducing the dimensionality of the space where control solutions are sought. This is only useful if we ensure that the reduction does not accidentally eliminate all the solutions to the control problem of interest. But how can such accidents be avoided, if the primitives are chosen before the control problem itself has been solved? In other words, what simplifying assumptions are safe to make for all tasks that may need to be performed in the future? The only thing that is perfectly safe to assume is that all tasks will be performed using the same musculo-skeletal system. Therefore we propose that the low-level primitives must reflect the regularities of that system.

We present an unsupervised learning algorithm designed to extract such regularities from sensory-motor interactions, and automatically transform them into higher-level control knobs. The unsupervised learning objective is to find a compact context-dependent representation of the correlations among motor commands and sensory feedback. We find that the resulting representations capture the agonist-antagonist organization of muscles and corresponding sensors, the distinction between muscle groups spanning separate joints, temporal filtering at appropriate time scales, etc. Once the primitives have been constructed, we apply reinforcement learning to obtain a high-level feedback controller. For the control problems investigated here, this form of learning is indeed faster than learning a feedback controller which does not utilize primitives.

II. FEEDBACK TRANSFORMATIONS

The pioneering works of Sherrington and Bernstein have emphasized that biological control hierarchies are composed of parallel feedback loops, whose processing delays increase with their level of sophistication. The low-level loops (e.g. the spinal cord) do not passively map high-level commands into control signals. Instead they receive sensory feedback and actively generate control signals even before the higher levels have had time to respond.

This can be formalized with the notion of a feedback transformation. Consider a dynamical system with state $\mathbf{x}$,

control **u**, dynamics $\Delta\mathbf{x} = \mathbf{f}(\mathbf{u},\mathbf{x})$, sensory observation $\mathbf{s}(\mathbf{u},\mathbf{x})$, and state estimator $\underline{\mathbf{x}}(\mathbf{u}...,\mathbf{s}...)$. Define a state transformation $\mathbf{h}=T(\underline{\mathbf{x}})$ and a control transformation $\mathbf{u}=G(\mathbf{v}, \underline{\mathbf{x}})$. This produces a new dynamical system with (observable) state **h** and control **v**, coupled to the original system via the functions T and G. Rather than learning a direct control policy $\mathbf{u}=P(\underline{\mathbf{x}};w)$ parameterized by w, we could instead try to learn a high-level policy $\mathbf{v}=Q(\mathbf{h};w)$ for the transformed system.

The best known example of this approach is the technique of feedback linearization: assuming for simplicity that $\mathbf{f}(\mathbf{u},\mathbf{x})$ is invertible w.r.t. **u**, the transformation $\mathbf{h}=\underline{\mathbf{x}}$, $u = \mathbf{f}^{-1}(\mathbf{v}, \underline{\mathbf{x}})$ produces a linear system with equivalent dynamics. Other successful application can be found in robotics where the feedback transformation is handcrafted using insight into the controlled system.

The question is, how can appropriate functions T and G be constructed by a learning algorithm instead of a human engineer. Before we address that we need to clarify the criterion for appropriateness of the feedback transformation. Generally speaking, it should make learning simpler. Simplicity however is not an objective property of the system to be controlled, but depends on what is "simple" from the learner's point of view. If for example we intend to build the high-level controller as an LQG regulator, the transformation should attempt to make the system linear and the cost quadratic. If instead we intend to apply reinforcement learning, sensible criteria include dimensionality reduction, state descriptions that predict the future (particularly the rewards in the future), and prepackaged control sequences providing temporal abstraction. The method proposed below appears to satisfy these criteria.

A. Unsupervised sensory-motor learning

Building a feedback transformation is an instance of the more general problem of building internal representations, which is naturally addressed via unsupervised learning. The idea is best illustrated by comparison with the perceptual system, where the importance of unsupervised learning has been appreciated for a long time. The perceptual analog to a motor primitive is a receptive field: a basic element used to decompose and represent the sensory input. As in the motor system, the receptive fields needed for perception cannot be easily derived from considerations of how an optimal perceptual system works. Instead receptive fields can be modeled by collecting sensory data and fitting a "generative model". The generative model as such is rather useless, since the job of the perceptual system is to perform exactly the opposite operation. It therefore needs to be "inverted" by a recognition model, mapping sensory input to internal states (Fig 1 Left).

We propose that motor primitives can be learned using the same principle – building an internal mirror image of the physical world (Fig 1 Right). The difference from perceptual

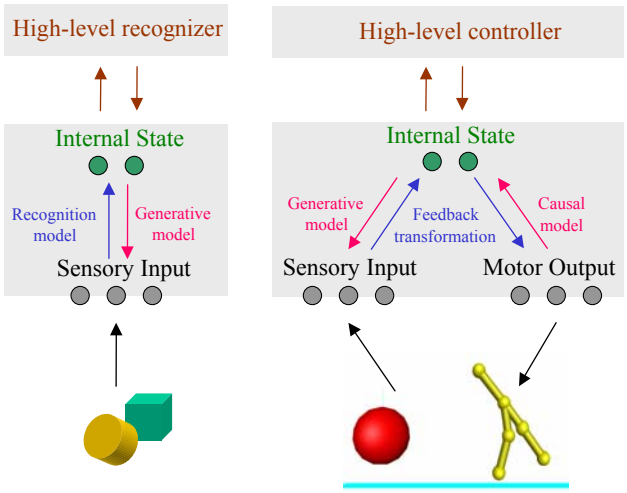

Fig. 1. Schematic illustration of forward and inverse models

learning is that we now have an input-output forward model, where the inputs (motor output) are under the control of the nervous system, and time cannot be neglected. The goal of learning is to find an internal state representation that explains the observed data (sensory input). As in perception, the forward model is exactly the opposite to what the motor system needs, but "inverting" it provides the feedback transformation we are looking for.

B. Constructing primitives

Suppose a sequence of control signals $\mathbf{u}_{...}$ is somehow generated, the plant dynamics is observed or simulated, and the corresponding sequence of sensory signals $\mathbf{s}_{...}$ obtained. Define the vectors $\mathbf{past}_t = [\mathbf{u}_{t-p}, \mathbf{s}_{t-p}, \ldots \mathbf{u}_{t-1}, \mathbf{s}_{t-1}]$, $\mathbf{control}_t = \mathbf{u}_t$ and $\mathbf{future}_t = [\mathbf{s}_t,\mathbf{u}_{t+1}, \mathbf{s}_{t+1}, \ldots \mathbf{u}_{t+f}, \mathbf{s}_{t+f}]$ where the past and future time horizons p and f can be inferred from temporal correlations in the data. The key step is fitting (see below) a conditional probability density model of the form:

$$p(\mathbf{future}, \mathbf{control} \mid \mathbf{past}) = \sum_{\mathbf{h}} p(\mathbf{future}, \mathbf{control} \mid \mathbf{h})\, p(\mathbf{h} \mid \mathbf{past}).$$

The "hidden" variable **h** is an internal state variable created by the unsupervised learning algorithm, and is used to communicate between the high and low level controllers. In particular, the low level controller computes at each time step the conditional expectation $\mathbf{h}_{past} = E(\mathbf{h} \mid \mathbf{past})$, using the conditional distribution $p(\mathbf{h} \mid \mathbf{past})$. Then $\mathbf{h}_{past}$ is sent to the high level controller, which applies a (task-specific) feedback law Q and returns $\mathbf{v} = Q(\mathbf{h}_{past})$. The high level control signal **v** is treated as a desired change in **h**. The actual control signal **u** is computed as the conditional expectation $\mathbf{u} = E(\mathbf{control} \mid \mathbf{h}_{past} + \mathbf{v})$, using the conditional distribution $p(\mathbf{future}, \mathbf{control} \mid \mathbf{h})$ and marginalizing over **future**. The resulting low level controllers can be expected to have several appealing properties:

- The internal state variable **h** captures the information about the past that is most useful in predicting the future. This reduces the control space accessible to the high level controller to the space of signals that have predictable consequences.
- The transformation E(**u** | $\mathbf{h}_{past}$ + **v**) can be thought of as a set of control synergies, which are driven by the high level control signal **v** as well as past sensory-motor activity. These synergies form a compact representation of the statistics of measured sensory-motor interactions, i.e. they capture the "modes" of the system.
- The control signals $\mathbf{u}_{...}$ used to generate the training data can be random, or alternatively they can be collected while an existing control scheme is being applied. In the latter case, the low level controller learns to mimic the control scheme used in the training phase, i.e. if the high level output is **v** = 0 the resulting control signal will be E(**u** | **past**). This provides a natural model of motor automation.

C. Fitting the model

While the general probabilistic model described above could be fit with a variety of methods, for the time being we have focused on the simplest method – which is a generalization of factor analysis to include inputs **i** as well as outputs **o**. Define **i** = **past**, **o** = [**future**, **control**], and the hidden variable is **h** as before. The generative model is

$$\mathbf{h} = B\,\mathbf{i} + \mathbf{w}; \quad \mathbf{o} = C\,\mathbf{h} + \mathbf{v}$$

where **w** and **v** are zero-mean Gaussian noise vectors with covariance Q and R respectively. The matrices B and C correspond to the (unknown) mappings from inputs to hidden states, and from hidden states to outputs. As in ordinary factor analysis, R is assumed to be diagonal. It is not difficult to derive an expectation-maximization (EM) algorithm for learning B, C, Q and R.

E-step: Define $K = Q\,C^T\,(R + C\,Q\,C^T)^{-1}$. Then the mean of **h** conditional on **i** and **o** is $\underline{\mathbf{h}} = B\,\mathbf{i} + K\,(\mathbf{o} - C\,B\,\mathbf{i})$. The covariance of **h** is $S = Q - K\,C\,Q$.

M-step: The model parameters are updated according to $C = \langle \mathbf{o}\,\underline{\mathbf{h}}^T \rangle S^{-1}$ and $B = \langle \underline{\mathbf{h}}\,\mathbf{i}^T \rangle \langle \mathbf{i}\,\mathbf{i}^T \rangle^{-1}$. The $\langle\,\rangle$ notation denotes an average over the training data. Similar updates can be obtained for Q and R.

D. Dynamical models

We applied this approach to simulated systems that capture many properties of musculo-skeletal dynamics (an example is shown in Fig 2 A). The physical simulator was built with the MathEngine Toolkit. The bodies of our creatures are made of cylinders with spherical ends, connected with hinge joints. For now they are restricted to a 2D plane. The physical simulation is actually in 3D, and we are producing extra forces to avoid deviations from the plane. The 2D constraint is used for easier visualization, and also because we have not yet incorporated a detailed model of 3D muscle wrapping.

Some of the spheres can be fixed in the world. The world can include gravity, ground, and external objects, i.e. spheres attached to spring-dampers whose other end is fixed. At the beginning of each (10 msec) time step the joint torques resulting from muscle activation are computed. All colliding pairs of objects are found, and the corresponding forces added. We used soft collisions (allowing some penetration) and friction in the directions orthogonal to the normal. The simulation is evolved using a semi-implicit integrator. An agonist-antagonist pair of muscles was present around each joint. The muscle model included a second-order linear filter ($\tau1=\tau2=50$ msec), constant moment arms, constant stiffness, and damping present only for shortening. Each simulated muscle was equipped with one static spindle (measuring length), one dynamic spindle (measuring velocity), one Golgi tendon organ (measuring force). The spindle sensitivity was adjusted by a (gamma) motor unit. Each joint had 2 limit sensors. Each sphere and cylinder (Fig 2) had one tactile sensor on each side, which responded whenever a contact occurred on the corresponding surface. Since contacts involved some penetration, the sensor had graded response. We simulated different systems driven by random control signals (2 per muscle), sampled at each time step from a uniform distribution. The low-pass filtering properties of muscles and the system impedance (inertia of the skeleton and muscle viscoelasticity) transformed the noise activation sequence into relatively smooth movements. The simulation continued for about 20000 simulated seconds.

III. RESULTS

A. Features of extracted primitives

The unsupervised learning algorithm implicitly discovered some of the basic properties of the dynamical systems being simulated. Examples are shown in Fig 2 where we have plotted the motor and sensory components of one primitive (by extracting the corresponding row and column of the B and C matrices, and reshaping them appropriately). The smooth progression of grayscale values in the vertical dimension corresponds to the fact that muscle forces change gradually (due to the low-pass filtering of muscles), and consequently the spindle inputs also change gradually. The fact that columns 1E and 2E are almost the opposites of columns 1F and 2F corresponds to the agonist-antagonist organization of the muscles around the two joints. Note also that 1E-1F are treated differently from 2E-2F (especially on the motor side). This particular primitive is thus more interested in the muscles acting around joint 1, i.e. the algorithm implicitly discovers the concept of a "joint".

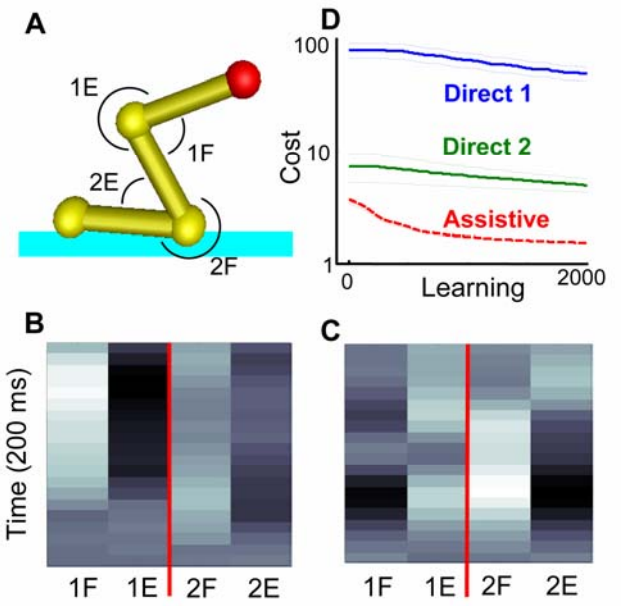

Fig. 2. Illustration of preliminary results. **A)** One of the dynamical systems we studied, with 2 joints and 4 muscles (1E, 1F, 2E, 2F). **B)** The learned motor weights (loadings) for one of the primitives. **C)** The sensory weights for the same primitive. **D)** Learning curves of different reinforcement learning algorithms (see text).

B. Using primitives to speed up learning

We also tested the hypothesis that using primitives as low-level controllers will speed up learning. This was done in 100 partially observable Linear-Quadratic-Gaussian (LQG) systems, where the dynamics and cost matrices were generated randomly. The task encoded by the cost matrices did not vary over time, so that the appropriate feedback control law was time invariant (and therefore easier to parameterize and learn). The reason for using LQG systems is that we can compute the optimal control law exactly, and therefore the minimum cost that can be achieved is known. Log costs for each system/task were scaled (see Fig 2D) so that the minimum achievable cost is always 1. We applied a policy gradient reinforcement learning method, to 3 different parameterized control laws. Control laws "Direct 1" and "Direct 2" acted directly on the dynamical system. "Direct 1" was driven by raw sensory inputs, while "Direct 2" was driven by the optimal estimate of the system state (and therefore it performed better than "Direct 1"). Note however that the "Assistive" controller, acting on the automatically extracted primitives, outperforms both of the direct methods. We should stress that the advantage of using primitives is likely to be much larger on a more difficult control problem.

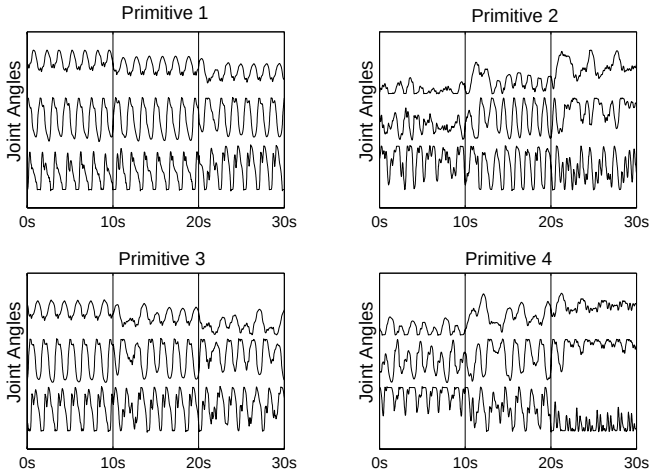

Fig. 3. Examples of rhythmic patterns of movement generated by a constant control signal to one primitive at a time. The magnitude of the control signal changes at the 10 sec time marks. These examples are from a dynamical system with 3 joint angles.

C. Autonomous pattern generation

An interesting feature of our primitives is that they have both a sensory and a motor component. In other words they act as tunable feedback controllers, and therefore can give rise to rich time-varying movement patterns even when the descending control is kept constant. This phenomenon is illustrated in Fig 3, where we send a constant control signal to a single primitive at a time. Of course these emergent patterns will not in themselves accomplish any useful task. But hopefully they will allow a high-level task controller to rapidly explore a large portion of space, and not waste its time trying to generate patterns that are "incompatible" with the natural dynamics of the controlled system.

IV. FUTURE WORK

While we are still exploring the properties of the proposed method, the preliminary results are encouraging. One future direction is to better understand what exactly the method extracts, i.e. what are the basic movement patterns that emerge from activating the primitives one at a time. This should perhaps be done in simpler dynamical systems. Another direction is to explore more advanced nonlinear methods for fitting the general probabilistic model. The speed-up in reinforcement learning we observed should also be investigated further, in more realistic biomechanical models performing specific motor tasks. The features of the extracted primitives should be compared to physiological responses in the spinal cord and other lower level motor areas. Finally, we hope that biomechanical models augmented with such automatically extracted primitives will prove more amenable to neuroprosthetic control.