Hierarchical Optimal Control of a 7-DOF Arm Model

Dan Liu and Emanuel Todorov

***Abstract*— We apply a hierarchical control framework to a realistic arm model. With 7 degrees of freedom and 14 muscles, this arm model has complex nonlinear dynamics operating on 28-dimensional state space and 14-dimensional control space. A high-level controller is designed to capture the main features of the complex high-dimensional plant dynamics but with reduced dimensionality. This allows us to solve the original optimal control problem without running into the curse of dimensionality. We also provide a method to design a low-level controller to generate arm configurations that are consistent with high-level controls and at the same time satisfy biological constraints. To the best of our knowledge, this is the first feedback controller for a detailed 3-D model of a human arm.**

I. Introduction

The human body has over 250 muscles each with a distinct mechanical action at one or more joints [1], far more than the degrees of freedom needed to perform any particular task. Key to understanding biological motor control is thus to solve the control problem of complex redundant systems. This control problem, however, has not been well addressed because the nonlinear dynamics and high-dimensional state and control spaces of human body prevent the use of many traditional methods for controller design.

In an attempt to describe the control of redundant manipulators, [2] proposed a hierarchical control framework. This framework is inspired by two observations. First, from a computational viewpoint, hierarchical organization emerges in optimal feedback control for redundant systems even when such organization is not imposed by design [3], [4]. Secondly, from a biological viewpoint, it is known that sensorimotor control occurs simultaneously on many levels [5], [6]. Lower-level circuits (e.g., the spinal cord) interact with the musculoskeletal system directly by both receiving rich sensory input and generating corresponding motor outputs before the rest of the brain has had time to react to that input. Higher-level circuits (e.g., the motor cortex), on the other hand, operate on a more abstract and goal-related movement representation [7].

The proposed hierarchical framework in [2], which we implement here, is composed of two layers (Fig. 1), analogous to the motor cortex-spinal cord structure. The plant is augmented with a low-level feedback controller, which receives full information about the plant states $\mathbf{x}$, and sends to the high-level controller a lower-dimensional representation $\mathbf{y}(\mathbf{x})$ which captures the task-relevant aspects of plant dynamics. The high-level controller issues commands $\mathbf{v}(\mathbf{y})$ to achieve a goal defined in $\mathbf{y}$-space, without having full access to the $\mathbf{x}$ dynamics. Then the low-level controller computes energy-efficient controls $\mathbf{u}(\mathbf{v}, \mathbf{x})$ consistent with $\mathbf{v}$. In this way, the high-level controller solves the optimal control problem without considering all the details of the plant and thus avoids running into the curse of dimensionality; the low-level controller performs an instantaneous feedback transformation to deal with the details. Two methods, an explicit and an implicit one, for coupling the low-level and high-level controllers are described in [2] and implemented here.

Dan Liu and Emanuel Todorov are with the Department of Cognitive Science, UC San Diego, San Diego (email: {dliu, todorov}@cogsci.ucsd.edu).

This work was supported by NIH.

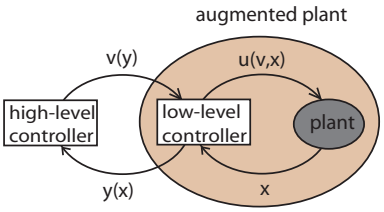

Fig. 1. Schematic illustration of a hierarchical control framework [2].

This hierarchical control framework has been tested on a simplified 2 degrees of freedom (DOF) planar arm model in [2]. The goal of this paper is to apply this framework to a more realistic 3-D arm model with 7-DOF and 14 muscles. Our focus is on designing the high-level controller to capture the low-level dynamics, and designing the low-lever controller to generate biologically plausible arm configurations.

The rest of this paper is organized as follows. Section II introduces the general framework of hierarchical control. Section III describes an arm model with 7-DOF and 14 muscles, whose dynamics closely resemble those of a human arm. Section IV and Section V explain the design of high-level and low-level controllers respectively. Simulation results on three tasks of reaching, orienting and drawing are presented in Section VI. Section VII concludes the paper with some final remarks.

II. General framework

Consider the dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t)) + B(\mathbf{x}(t))\mathbf{u}(t) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is the state vector, $\mathbf{u} \in \mathbb{R}^{n_u}$ is the control vector, $\mathbf{a}(\mathbf{x})$ are the passive dynamics, and $B(\mathbf{x})\mathbf{u}$ are the control-dependent dynamics assuming linear in the control. The high-level state vector $\mathbf{y} \in \mathbb{R}^{n_y}$ is a static function of the plant state:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{2}$$

$\mathbf{h}$ should allow the high-level states to contain enough information in computing the state-dependent cost $q(t, \mathbf{x})$ but with reduced dimensionality. In other words, $\mathbf{h}$ needs to satisfy $\exists \tilde{q}$ s.t. $\tilde{q}(t, \mathbf{y}) = q(t, \mathbf{x})$ and $n_{\mathbf{y}} < n_{\mathbf{x}}$.

In the high-level, we seek to construct some high-level dynamics that capture the main features of low-level dynamics but operate on lower-dimensional state and control space:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t)) + G(\mathbf{y}(t))\mathbf{v}(t) \tag{3}$$

Here $\mathbf{v} \in \mathbb{R}^{n_v}$ is the high-level control vector, functions $\mathbf{f}$ and G are the passive- and control-dependent dynamics from the high level. $\mathbf{f}$ and G should allow the high-level controller to choose control signals $\mathbf{v}$ to accomplish the task goal without considering all the details of the plant.

Differentiating Eq. 2 w.r.t. time t and using Eq. 1, the dynamics of $\mathbf{y}$ become

$$\dot{\mathbf{y}} = H(\mathbf{x})(\mathbf{a}(\mathbf{x}) + B(\mathbf{x})\mathbf{u}) \tag{4}$$

where $H(\mathbf{x}) = \partial\mathbf{h}(\mathbf{x})/\partial\mathbf{x}$ is the Jacobian of the function $\mathbf{h}$. The goal of the low-level controller is then to choose $\mathbf{u}(\mathbf{v}, \mathbf{x})$ so that the $\mathbf{y}$ dynamics from the low-level circuits (Eq. 4) match those from the high-level circuits (Eq. 3), or

$$H(\mathbf{x})\mathbf{a}(\mathbf{x}) + H(\mathbf{x})B(\mathbf{x})\mathbf{u} = \mathbf{f}(\mathbf{y}) + G(\mathbf{y})\mathbf{v} \tag{5}$$

The goals of the two levels can be archived using two methods:

1) Explicit modeling
   We can treat the high level as an autonomous system, explicitly model its dynamics, and then use standard optimization techniques to solve the optimal control problem in the high level. The low-level controller then seeks energy efficient $\mathbf{u}$ to satisfy Eq. 5. Ideally, we would like to have the high-level dynamics to mimic those from the low level, so

   $$\mathbf{f}(\mathbf{y}) \approx H(\mathbf{x})\mathbf{a}(\mathbf{x}) \tag{6}$$

   However, when the high-level controller considers only the end-effector (e.g. the fingertip) location as in most reaching movements, Eq. 6 cannot be guaranteed because different arm configurations may result in the same end-effector location. In other words,

   $$H(\mathbf{x_1})\mathbf{a}(\mathbf{x_1}) \neq H(\mathbf{x_2})\mathbf{a}(\mathbf{x_2}) \text{ even } \mathbf{h}(\mathbf{x_1}) = \mathbf{h}(\mathbf{x_2}) \tag{7}$$

   In this case, low-level dynamics cannot be fully captured on the high level. As a result, the optimal solution generated from the high-level controller may not be optimal with respect to the low-level dynamics.
2) Implicit modeling
   The drawback of explicit modeling can be avoided by not modeling the passive dynamics explicitly on the high level. Instead, the high-level controller gets online access to $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ at each time step, and uses it to update $\mathbf{y}$. The discrepancy between $\mathbf{f}(\mathbf{y})$ and $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ is no longer compensated on the low level. So the low-level controller only needs to satisfy

   $$H(\mathbf{x})B(\mathbf{x})\mathbf{u} = G(\mathbf{y})\mathbf{v} \tag{8}$$

   In this way, the high-level controller can exploit passive dynamics of the plant while operating on a lower-dimensional system. In addition, now we can apply Eq. 8 to match the low-level control cost $r(\mathbf{u}, \mathbf{x})$ exactly on the high level using some function $\tilde{r}(\mathbf{v}, \mathbf{y})$. Therefore, the high-level controller can seek to solve the exact optimal control problem with respect to both the true plant dynamics and the true cost function. However, since now the high level is no longer an autonomous system, regular optimization tools are not guaranteed to provide optimal solutions.

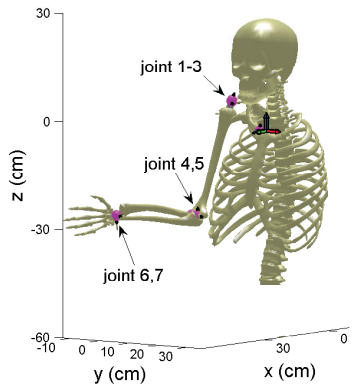

Fig. 2. A 7-DOF arm model.

III. Low-level dynamics

Here we consider a 7-DOF arm with the same skeletal structure as a human arm: the shoulder is modeled as a 3-DOF joint (only the glenohumeral joint is taken into account here), with flexion-extension, abduction-adduction and external-internal rotation; the elbow is modeled as a 2-DOF joint (humeroulnar joint and radioulnar joint), with flexion-extension and pronation-supination movements; the wrist is modeled as a 2-DOF joint, with flexion-extension and abduction-adduction movements.

Low-level dynamics are modeled as a third-order system resembling those in a human arm [8]. Low-level states include joint angles (θ), joint velocities ($\dot{\theta}$), and muscle activations ($\mathbf{s}$). Although each joint is actually controlled by a group of muscles which are also connected to other joints, here we consider a simplified case where each joint is independently controlled by two muscles acting in opposite directions. So the complete system has a 28-D state vector $\mathbf{x} \doteq (\theta_1, ..., \theta_7, \dot{\theta}_1, ..., \dot{\theta}_7, s_1, ...s_{14})^T$ and a 14-D control vector $\mathbf{u} \doteq (u_1, ..., u_{14})^T$.

Forward dynamics of the system can be expressed as

$$\ddot{\theta} = I(\theta)^{-1}\left(\tau(\theta, \dot{\theta}, \mathbf{s}) - \mathbf{n}(\theta, \dot{\theta})\right) \tag{9}$$

where $I(\theta) \in R^{7\times7}$ is a positive definite symmetric inertia matrix, $\tau(\theta, \dot{\theta}, \mathbf{s}) \in R^7$ represents joint torques, and $\mathbf{n}(\theta, \dot{\theta}) \in R^7$ are Centripetal, Coriolis, gravitational, and viscoelastic forces. Joint torques are generated by muscles following

$$\tau(\theta, \dot{\theta}, \mathbf{s}) = M(\theta)T(\mathbf{s}, \mathbf{l}(\theta), \dot{\mathbf{l}}(\theta, \dot{\theta})) \tag{10}$$

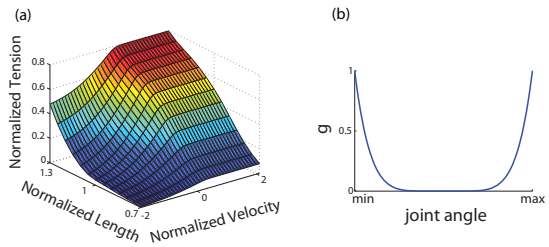

Fig. 3. **(a)**, The total force produced by a muscle fiber at a given level of activation depends on both its instantaneous length and velocity, which are independent kinematic variables. **(b)**, polynomial function to punish moving towards joint limits.

where $M(\theta) \in \mathbb{R}^{7\times 7}$ is the moment arm, defined as the perpendicular distance from a muscle's line of action to a joint's center of rotation. Although moment arms are posture dependent, here we consider them constant. $T(\mathbf{s}, \mathbf{l}(\theta), \dot{\mathbf{l}}(\theta, \dot{\theta}))$ represents muscle tension. The tension produced by muscle i depends on its physiological cross-sectional area (PCSA) and activation state s_i, as well as muscle length $\mathbf{l}$ and velocity $\dot{\mathbf{l}}$. The substantial length-and-velocity dependence is modeled based on the Virtual Muscle model [9] which provides a state-of-the-art fit to a range of physiological data, with slight simplification for computational purpose, illustrated in Fig. 3**(a)** for maximal activation $s_i = 1$. Muscle activations $\mathbf{s}$ have first-order low-pass filter dynamics

$$\dot{s}_i = \frac{1}{\alpha}(u_i - s_i) \tag{11}$$

for $i = 1, ..., 14$ with $\alpha = 40msec$. Each control u_i is constrained to the range $[0, 1]$. In this way, our arm model captures most key features of a human arm even after some simplifications.

The low-level dynamics described above can be summarized as follows:

$$\mathbf{a}(\mathbf{x}) = \begin{bmatrix} \dot{\theta} \\ I(\theta)^{-1}\left(\tau(\theta, \dot{\theta}, \mathbf{s}) - \mathbf{n}(\theta, \dot{\theta})\right) \\ -\frac{1}{\alpha}\mathbf{s} \end{bmatrix}, B(\mathbf{x}) = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \frac{\mathbf{1}}{\alpha} \end{bmatrix} \tag{12}$$

IV. High-level controller design

The goal of movement in everyday life (e.g. to pick up a cup of coffee) is, by most accounts, defined in spatial coordinates rather than in joint coordinates, and some evidence has shown that human movements are planned in Cartesian coordinates [10]. Therefore, in the high level, we only include 3-D position ($\mathbf{p}$) and velocity ($\dot{\mathbf{p}}$) of the end-effector (i.e., the fingertip) in Cartesian hand coordinates, so $\mathbf{y} \doteq [p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z]^T$. The high-level control, accordingly, is defined as a 3-D force $\mathbf{v} \doteq [v_x, v_y, v_z]^T$. Modeling the high-level dynamics with a second-order system, rather than a first-order system, is necessary because instantaneous velocity commands are too far removed from muscles that have to carry them out [2].

As introduced in Section II, there are two ways to design the high-level controller. Using the explicit modeling, we can simplify the entire arm into a point mass (m) and model its dynamics using a simple linear system:

$$\mathbf{f}(\mathbf{y}) = \begin{bmatrix} \dot{\mathbf{p}} \\ \mathbf{0} \end{bmatrix}, G(\mathbf{y}) = \begin{bmatrix} \mathbf{0} \\ \frac{1}{m} \end{bmatrix} \tag{13}$$

The transformation from low level to high level can be represented as

$$H(\mathbf{x}) = \begin{bmatrix} J(\theta) & \mathbf{0} & \mathbf{0} \\ \dot{J}(\theta) & J(\theta) & \mathbf{0} \end{bmatrix} \tag{14}$$

where J is the Jacobian $J(\theta) = \partial\mathbf{p}/\partial\theta$, and $\dot{J}(\theta) = \frac{d}{dt}(J(\theta))$.

We can also use the implicit modeling. Instead of explicitly modeling the high-level dynamics, we use $H(\mathbf{x})\mathbf{a}(\mathbf{x})$ to update the high-level dynamics at each time step. This is done by augmenting $\mathbf{y}$ into $\mathbf{y} \doteq [p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z, 1, 1, 1]^T$ and constructing the high-level dynamics as

$$\mathbf{f}(\mathbf{y}) = \begin{bmatrix} \dot{\mathbf{p}} \\ \mathbf{e} \\ \mathbf{0} \end{bmatrix}, G(\mathbf{y}) = \begin{bmatrix} \mathbf{0} \\ I(\mathbf{y})^{-1} \\ \mathbf{0} \end{bmatrix} \tag{15}$$

where $\mathbf{e}$ is the 4^{th} to 6^{th} rows of $H(\mathbf{x})\mathbf{a}(\mathbf{x})$, $I(\mathbf{y})$ is the end-effector inertial matrix with $I(\mathbf{y})^{-1} = J(\theta)I(\theta)^{-1}J(\theta)^T$. The transformation from low level to high level becomes

$$H(\mathbf{x}) = \begin{bmatrix} J(\theta) & \mathbf{0} & \mathbf{0} \\ \dot{J}(\theta) & J(\theta) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{16}$$

A. Dynamic compatibility between levels of control

Since the high-level dynamics are second order whereas the low-level dynamics are third order, dynamics from the two levels are not compatible. In particular, the low-level controls are muscle activations (see Eq. 12), which control the torque change and cannot affect acceleration on the high-level dynamics instantaneously (see Eq. 14 and Eq. 16), causing $H(\mathbf{x})B(\mathbf{x}) = 0$. However, the change of torque has a predicable effect when applied over time, suggesting that the "instantaneous" $\mathbf{a}$ and B should be replaced with some functions $\tilde{\mathbf{a}}$ and $\tilde{B}$ that can incorporate temporal predictions. Here we construct such functions using implicit integration as explained in [2] and turn $\tilde{B}$ into the form

$$\tilde{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{\Delta} \\ \frac{\mathbf{1}}{\alpha} \end{bmatrix} \tag{17}$$

resulting in $H\tilde{B} \neq 0$.

B. High-level controller optimization

Ideally, we would like to minimize a cost function that enforces both task performance and energy efficiency. That is,

$$\mathcal{L}(\mathbf{x}, \mathbf{u}) = \int_0^T \left(q(t, \mathbf{x}) + r(t, \mathbf{u})\right) dt \tag{18}$$

where $q(t, \mathbf{x})$ and $r(t, \mathbf{u})$ are instantaneous state-dependent (or task-dependent) cost and control-dependent cost respectively.

As assumed before, $\mathbf{y}$ contains enough information to satisfy $\exists\tilde{q}$ s.t. $\tilde{q}(t,\mathbf{y}) = q(t,\mathbf{x})$. Thus the state-dependent cost can be fully accounted for on the high level. Control-dependent control, on the other hand, may not be fully captured for on the high level. In particular, control-dependent cost is usually modeled using a quadratic form

$$r(t,\mathbf{u}) = \frac{\gamma}{2}\mathbf{u}(t)^T\mathbf{u}(t) \tag{19}$$

If we can represent low-level controls $\mathbf{u}$ using high-level controls $\mathbf{v}$ as

$$\mathbf{u}(t) = K(\mathbf{x},t)\mathbf{v}(t) \tag{20}$$

then we can reformulate the control-dependent cost $r(t,\mathbf{u})$ in the high level again in a quadratic form:

$$\tilde{r}(t,\mathbf{v}) = \frac{\gamma}{2}\mathbf{u}(t)^T K(\mathbf{x},t)^T K(\mathbf{x},t)\mathbf{u}(t) \tag{21}$$

This can be archived by the implicit modeling but not the explicit modeling. In particular, applying Eq. 15, Eq. 16, and Eq. 17 to Eq. 8, we have

$$P\mathbf{u} = I(\mathbf{y})^{-1}\mathbf{v} \tag{22}$$

where $P = J(\theta)\Delta$ and $P \in \mathbb{R}^{n_\mathbf{v}} \times \mathbb{R}^{n_\mathbf{u}}$. Since $n_\mathbf{v} = 3$ whereas $n_\mathbf{u} = 14$, Eq. 22 means that only a subspace of the low-level controls will affect task performance and is thus projected to the high level. Apply singular value decomposition to P and get $P = U\Lambda V^T$. V^T projects $\mathbf{u}$ into task-relevant space (Ω) and task-irrelevant space ($\bar{\Omega}$). Then minimizing $r(t,\mathbf{u})$ becomes equivalent to minimizing the controls in Ω and setting $\bar{\Omega}$ into 0. So we get

$$K = (U\tilde{\Lambda})^{-1}I(\mathbf{y})^{-1} \tag{23}$$

where $\tilde{\Lambda}$ is the first $n_\mathbf{v}$ columns of Λ.

Implicit modeling allows the high-level controller to capture both the passive dynamics and the true cost from the low level. However, since the high level is no longer an autonomous dynamical system, its optimization becomes difficult. Indeed, we apply the iterative LQG (iLQG) method [11], guaranteed to converge for autonomous system, to the high-level controller, and find that it does not yield good solutions. To make the high-level system slightly simpler but still capture the natural dynamics of the low level, we only include the skeletal structures of the low level in the high-level dynamics while ignoring the complex muscle properties. Under the assumption that muscles are strong enough to achieve the torques required by the high-level controller, the low-level controller can control the muscle activations to match the high-level controls exactly. After making this modification, the optimization process using iLQG can converge to good solutions after several iterations, which will be presented in numerical simulations.

V. Design of low-level controller

In addition to generating energy efficient control $\mathbf{u}$ to satisfy Eq. 5, the low-level controller also needs to take into account the constraints of biological movements. First, u_i is constrained to $u_i \in [0,1]$. Secondly, each joint can only move within a certain range, so $\theta_i \in [\theta_i^{\min}, \theta_i^{\max}]$. In order to punish moving towards the joint limits, we use a polynomial function:

$$g(\theta_i) = (\alpha_i\theta_i + \beta_i)^6/100^6 \text{ for } i = 1,2,...,7. \tag{24}$$

where

$$\alpha_i = \frac{200}{\theta_i^{\max} - \theta_i^{\min}}, \beta_i = 100 - \frac{200}{\theta_i^{\max} - \theta_i^{\min}}\theta_i^{\max}$$

Here, $\alpha_i\theta_i + \beta_i$ normalizes joint angle $\theta_i \in [\theta_i^{\min}, \theta_i^{\max}]$ into $[-100, 100]$, and 100^6 normalizes $g(\theta_i)$ into $[0,1]$. Thus $g(\theta_i)$ increase dramatically when the joint angle (θ_i) approaches its joint limits, shown in Fig. 3**(b)**.

Now, low-level controls $\mathbf{u}$ at each time t can be considered as the solution to the following constrained optimization problem: given $\mathbf{v}$ and $\mathbf{x}$, find $\mathbf{u}$ that minimize

$$\frac{\gamma}{2}\mathbf{u}^T\mathbf{u} + \eta\left(\mathbf{a}(\mathbf{x}) + B(\mathbf{x})\mathbf{u}\right)^T\nabla_\mathbf{x}\sum_{i=1}^{7} g(\theta_\mathbf{i}) \tag{25}$$

subject to

$$H(\mathbf{x})B(\mathbf{x})\mathbf{u} = \mathbf{f}(\mathbf{y}) + G(\mathbf{y})\mathbf{v} - H(\mathbf{x})\mathbf{a}(\mathbf{x})$$
$$0 \le u_i \le 1 \tag{26}$$

where η specifies the weight on moving away from joint limits. This optimization problem can be solved via quadratic programming. Note in implicit modeling, $\mathbf{f}(\mathbf{y}) = H(\mathbf{x})\mathbf{a}(\mathbf{x})$ is guaranteed automatically.

VI. Numerical simulations

Here we consider three tasks: reaching, orienting and drawing. Our focus is to test the validity of the proposed framework, and also explore the representation of high-level states to account for different task requirements.

A. Reaching task

The task is to start from some initial position, move and stop the end-effector (i.e., the fingertip) at a target $\mathbf{p}^*$ defined in Cartesian hand coordinates in a specified time interval T. The cost can be formed in the high level as follows:

$$\mathcal{L}(\mathbf{y},\mathbf{v}) = \frac{\gamma}{2}\int_0^T \mathbf{v}(t)^T\mathbf{v}(t)dt + \omega_1\|\mathbf{p}(T) - \mathbf{p}^*\|^2 + \omega_2\|\dot{\mathbf{p}}(T)\|^2 \tag{27}$$

where $\mathbf{p}(T)$ and $\dot{\mathbf{p}}(T)$ are the endpoint position and velocity of the end-effector in Cartesian hand coordinates. γ, ω_1 and ω_2 are the relative weights of the terms enforcing energy efficiency, endpoint accuracy and stopping at the target respectively. These three parameters are usually adjusted manually to fit movements with different distances. To automatically scale these parameters, we do the following normalizations. Accuracy and stopping requirements are assumed to be independent of movement distance, therefor their weights (ω_1 and ω_2) are kept constant. The weight of control cost (γ), on the other hand, needs to be scaled according to the distance so that the contribution of energy consumption is relatively the same with respect to that of accuracy and stopping. To get

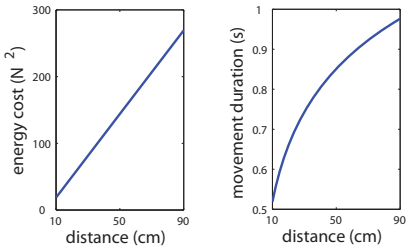

Fig. 4. Normalization. **(a)**, Total control as a function of movement distance. **(b)**, Time duration as a function of movement distance.

this scaling, we apply an optimal feedback control model [12] to reaching movements with different distances ranging from 10cm to 50cm, and fit the dependency of the total energy consumption ($\Gamma = \frac{1}{2}\int_0^T \mathbf{v}(t)^T\mathbf{v}(t)dt$) on distance ($d \geq 5cm$) and mass (m) on the moving direction as

$$\Gamma = (3.1d - 12.8)m^2 \tag{28}$$

Fig. 4**(a)** shows Γ as a function of distance. Then we can normalize the control cost by setting $\gamma = \frac{1}{\Gamma}$.

Movement duration T is determined based on Fitt's law [13], which states that the time taken to acquire a visual target should depend on both the movement distance (d) and the accuracy requirement (w) in the form $T = c_1 + c_2 \log_2(\frac{d}{w} + 1)$ where c_1, c_2 are experimentally determined constants. Assuming finishing a movement of 30cm with an accuracy of 1cm takes about 750msec, we set $c_1 = 0$ and $c_2 = 1.5$. Fig. 4**(b)** shows T as a function of distance.

We first compare explicit modeling and implicit modeling on a reaching movement from $\mathbf{p} = [48, -8, -15]^T$ to $\mathbf{p}^* = [21, -21, -30]^T$. In explicit modeling, high-level dynamics are modeled using a linear system and the optimization problem is solved using LQG [12]. In implicit modeling, on the other hand, the high level is not treated as an autonomous system but accounts for the low-level dynamics. The optimization problem is solved using iLQG. Since iLQG is a local method and may get trapped in local minima, initial controls are provided by LQG which treats the entire arm as a point mass. Fig. 5**(a)** shows how the final cost converges over iterations based on the two methods. As we expected, LQG converges within one iteration. However, cost estimated from the high level (blue star) does not match that from the low level (red star). Cost from iLQG, on the other hand, converges after 20 iterations, where the estimated cost from the high level truthfully captures the cost from the low level. Note iLQG guarantees converges only for autonomous dynamical systems, which is why the cost here increases occasionally rather than monotonically decreases. As we can see, the low-level cost after optimization is much smaller from implicit modeling than that from implicit modeling. Fig. 5**(b)** further compares the end-effector trajectory during both movement planning and movement execution from both methods. In explicit modeling, the planned trajectory (black line) is straight which is optimal with respect to the linear dynamical system in the high level. However, due to the discrepancy of the dynamics from the two levels, this planned trajectory cannot be executed by the low-level controller, causing big endpoint positional error (green line). The implicit method, in contrast, takes into account the true low-level dynamics and thus planned a curved movement (blue line). This trajectory is tracked exactly by the low-level controller during execution, leading to much better endpoint accuracy (megenda line). Therefore, the implicit method yields better performance although it takes more computational time. The rest of results are all based on implicit modeling.

Fig. 5**(c)** shows end-effector positions and velocities in the Cartesian hand coordinates, as well as joint angles and velocities in joint coordinates during the movement. As we can see, end-effector velocities in Cartesian hand coordinates are close to bell-curved profiles, and end-effector position reaches the target in the end of the movement. Fig. 5**(d)** shows the muscle activations during the movement. Note each joint is controlled by two muscles acting in opposite directions (solid lines vs. dashed lines). The two muscles in each muscle pair are activated at different times to either push the joint to move or to pull the joint to stop. Fig. 5**(e)** and **(f)** show the arm configurations at the beginning and the end of the movement. Note although there are many ways to accomplish the same end-effector position, the arm configuration in **(f)** is a natural posture in the sense that every joint stays in its movement limits. This is archived by including the term $g(\theta)$ on punishing movements approaching joint limits in computing low-level controls (Eq. 24 and Eq. 25). Fig. 5**(g)** compares the movement of joint 1 and joint 3 before (solid lines) and after (dashed lines) including $g(\theta)$. As we can see, function $g(\theta)$ pushes joint 1 to stay away from its joint limits, and moves joint 3 further to maintain the same end-effector position.

We further test this implicit modeling method with 77 targets evenly distributed in a 3-D space surrounding the initial end-effector location. Fig. 6**(a)** shows target locations (red circles) and the error vector on each target (blue line connecting the end-effector and the corresponding target). Each target location is determined by two parameters: height and angle on the x-y plane. Fig. 6**(b)** and **(c)** show how error vary over different heights (x axis) and angles (represented by different colors of lines) during both movement planning and movement execution. As we can see, since the high level captures the natural plant dynamics, the planned movement is well accomplished by the low-level controller. Also, the normalization of the control cost mentioned before makes it possible to generate movements ranging from 20cm to 50cm with similar accuracy (although errors for bigger movement are slightly bigger).

B. Orienting task

The task is to make a reaching movement and stop at a target, again defined in Cartesian hand coordinates, with a specific palm orientation. One way to solve this problem is to consider a fixed-length pointer attached to the palm with an orientation perpendicular to the palm. The goal of control is to move the two ends of the pointer to two desired positions

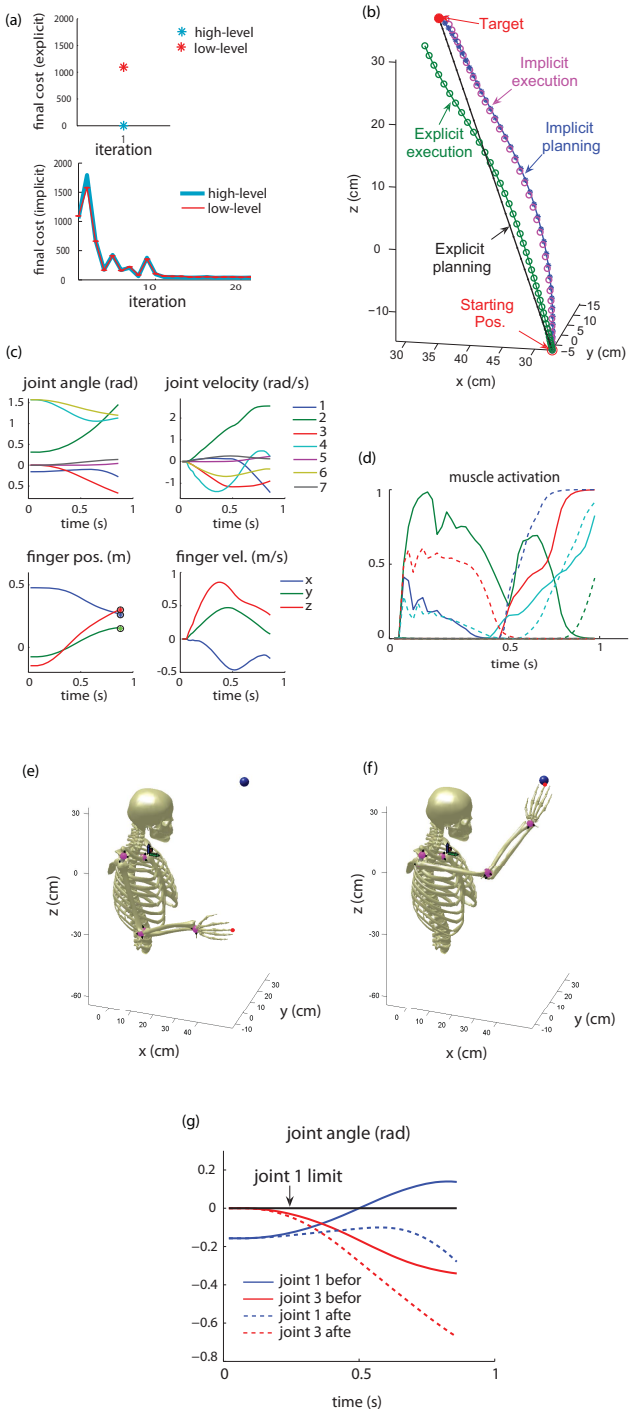

Fig. 5. Arm movement during reaching. **(a)**, Change of cost during optimization. **(b)**, Movement trajectory during movement planning and movement execution based on implicit modeling or explicit modeling. **(c)**, Joint angle, joint velocity, finger position, and finger velocity during a reaching movement. **(d)**, Muscle activations during a reaching movement. **(e)**, Arm configuration at the starting position. **(f)**, Arm configuration at the end position. **(g)**, Comparison of joint moments before and after including cost to punish hitting joint limits.

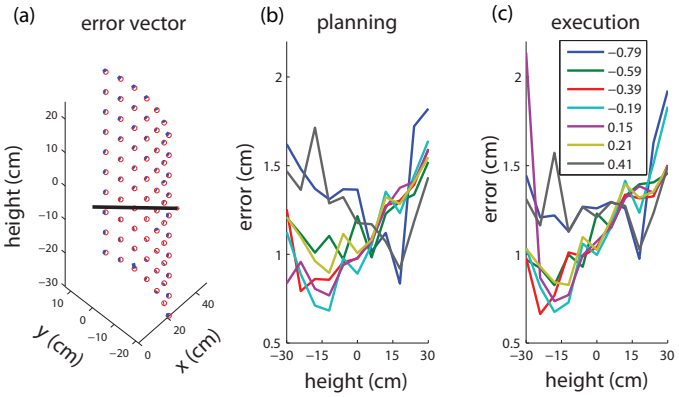

Fig. 6. Endpoint errors in reaching movements. **(a)**, Target locations and the corresponding error vectors. Each target location is determined by two parameters: height and angle on the x-y coordinate. **(b)**, Endpoint positional errors planned by the high-level controller. **(c)**, Endpoint positional error executed by the low-level controller. Lines with different colors represent errors from movements towards targets with different angles.

respectively. The high-level states thus include 18 dimensions with 9-D for each target defined as before. However, this control problem is really hard to solve since now the high-level dynamics become very complex and iLQG gets trapped in local minia so easily. To make the task slightly simpler, we directly provide the joint angles which will lead to different palm orientations, and define the cost in the high-level as follows

$$\begin{aligned}\mathcal{L}(\mathbf{y},\mathbf{v}) = \frac{\gamma}{2}\int_0^T \mathbf{v}(t)^T\mathbf{v}(t)dt \\ + \omega_1\|\mathbf{p}(T)-\mathbf{p}^*\|^2 + \omega_2\|\dot{\mathbf{p}}(T)\|^2 \\ + \omega_3\|\tilde{\mathbf{p}}(T)-\tilde{\mathbf{p}}^*\|^2 + \omega_4\|\dot{\tilde{\mathbf{p}}}(T)\|^2 \end{aligned} \quad (29)$$

where $\mathbf{p}(T)$ and $\dot{\mathbf{p}}(T)$ are the endpoint positions and velocities of the end-effector in Cartesian hand coordinate as defined in reaching tasks. $\tilde{\mathbf{p}}(T)$ and $\dot{\tilde{\mathbf{p}}}(T)$ are the endpoint joint angles and velocities of specific joints. $\mathbf{p}^*$ and $\tilde{\mathbf{p}}^*$ specify target position and desired joint angles. $\gamma, \omega_1, \omega_2, \omega_3, \omega_4$ are the relative weights of the terms enforcing energy efficiency, endpoint accuracy, stopping at the target, getting specific joint angles, and stopping moving the joints respectively. γ is normalized to account for different moving distances as explained before. $\omega_1, \omega_2, \omega_3, \omega_4$ are adjusted to get the best result.

Fig. 7 shows arm configurations in the end of a reaching movement aiming to the same target but with different palm orientations. Fig. 7**(a)** is the neutral posture where the wrist is not bended, Fig. 7**(b)(c)** are cases requesting for pronation-supination of the palm (movement of joint 5), Fig. 7**(d)(e)** are cases requiring flexion-extension of the palm (movement of joint 7). Fig. 8 shows how movement of joint 1 is adjusted automatically to accomplish different desired angles of joint 7, while keeping the endpoint end-effector position the same.

C. Drawing task

The task is to start from some point on a predefined ellipse and track this ellipse as fast as possible. Suppose this ellipse

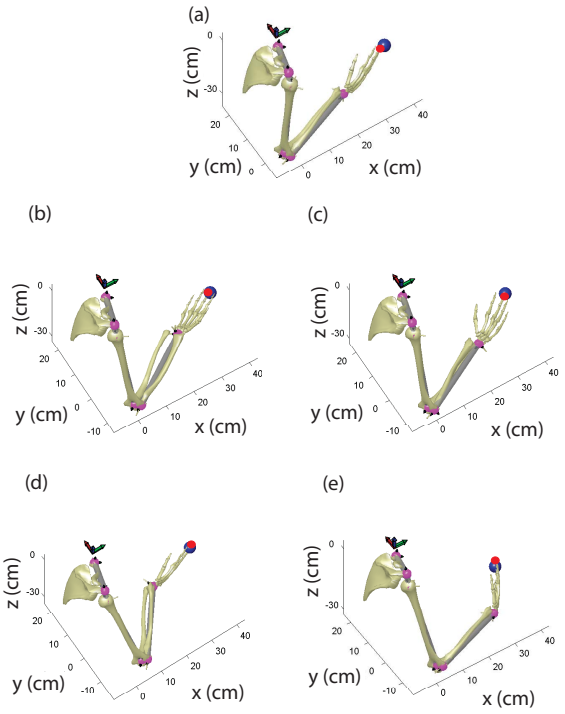

Fig. 7. Arm configuration with the same end-effector position but different palm orientations. **(a)**, wrist is not bended. **(b)**, positive pronation-supination of the wrist. **(c)**, negative pronation-supination of the wrist. **(d)**, positive flexion-extension of the wrist. **(e)**, negative flexion-extension of the wrist.

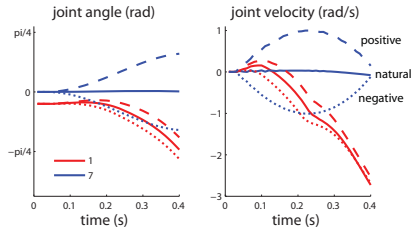

Fig. 8. Joint angle and joint velocity during a reaching movement with different palm orientations.

is defined in a 2-D plane (e.g., x-z coordinate in the Cartesian hand coordinates with constant value in y coordinate p_y^*), with the two foci $\mathbf{p}_1$ and $\mathbf{p}_2$ and the sum of the distances to these two foci a constant l. The control-dependent cost can be defined as in Eq. 19, and the state-dependent cost can be formed as follows:

$$q(t,\mathbf{y}) = \omega_1\|d(t)\|^2 + \omega_2\frac{\dot{\mathbf{p}}(t)\cdot\mathbf{p}_\perp(t)}{\|\mathbf{p}(t)-\mathbf{o}\|^2} + \omega_3\|p_y(t)-p_y^*\|^2 \tag{30}$$

where $d(t)$ is the tracking error measured as $d(t) = \|\mathbf{p}-\mathbf{p}_1\|+\|\mathbf{p}-\mathbf{p}_2\|-2l$, $\mathbf{o}$ is the center of the ellipse $\mathbf{o} = \frac{1}{2}(\mathbf{p}_1+\mathbf{p}_2)$, $\mathbf{p}(t) = [p_x(t), p_z(t)]^T$ and $\dot{\mathbf{p}}(t) = [\dot{p}_x(t), \dot{p}_z(t)]^T$ are the current position and velocity of the end-effector in the x-z coordinate, $\mathbf{p}_\perp(t)$ is a vector on the tangential direction defined as $\mathbf{p}_\perp(t) = [p_z(t), -p_x(t)]^T$, $\frac{\dot{\mathbf{p}}(t)\cdot\mathbf{p}_\perp(t)}{\|\mathbf{p}(t)-\mathbf{o}\|^2}$ computes the angular velocity. $\omega_1, \omega_2, \omega_3$ are the relative weights of the three terms enforcing tracking the ellipse, circling fast, and staying within the 2-D plane of the ellipse. As we can see, we do not specify the end-effector position at each time step as in a tracking task, which makes this drawing task even harder.

We apply implicit modeling on a drawing task where the ellipse is simply a circle with a radius of 10cm and a duration of 6s. Fig. 9 **(a)** and **(b)** illustrate the arm configurations before the movement and during the movement. Fig. 9 **(c)** shows the end-effector trajectories in Cartesian hand coordinates. As we can see, the planned trajectory (light blue line) tracks the desired circle (red line) closely even though the position at each time step is not explicitly specified in the task. This planned trajectory is also well accomplished by the low-level controller (dark blue line). Fig. 9 **(d)** shows the joint angles and velocities in joint coordinates, as well as the end-effector positions and velocities in Cartesian hand coordinates. The circular movement involves mainly the movements of joint 2, 4, and 6. Although there are slight oscillations in the beginning, movements become very stable after 0.5s.

VII. Discussion

A hierarchical control framework is designed for controlling a 7-DOF arm model, which captures most key features of a human arm. The high-level feedback controller solves the original control problem but operates on more abstract representations to avoid running into the curse of dimensionality; the low-level feedback controller performs an instantaneous feedback transformation to deal with the details. We show that by allowing the high-level feedback controller to mimic the low-level dynamics, the original optimal control problem can be better captured from the high level. To avoid making the high-level dynamics too complex, the high level considers only the skeletal structures of the joints rather than the full details of complex muscle properties. Simulation results suggest that taking into account the low-level dynamics in the high level generates better results than treating the high-level dynamics as linear. We also provide a scheme to shaping the task-irrelevant space of low-level controls to solve the redundancy problem. In particular, by punishing joint movements approaching their limits, the low-level feedback controller generates natural arm configurations to satisfy the task requirements monitored by the high-level controller. Satisfactory results in tasks such as reaching, orienting and drawing suggest that this hierarchical control framework may not only provide a solution to controlling complex redundant systems, but also shed some light on understanding the neural control of biological movements. Thus, this framework has the potential to be applied to control a paralyzed patient's arm via muscle stimulations.

This hierarchical frameworks is more general than other hierarchical schemes aiming to decouple task-level control from details of plant dynamics. For example, the operational space (OS) formulation [14] cannot handle systems other

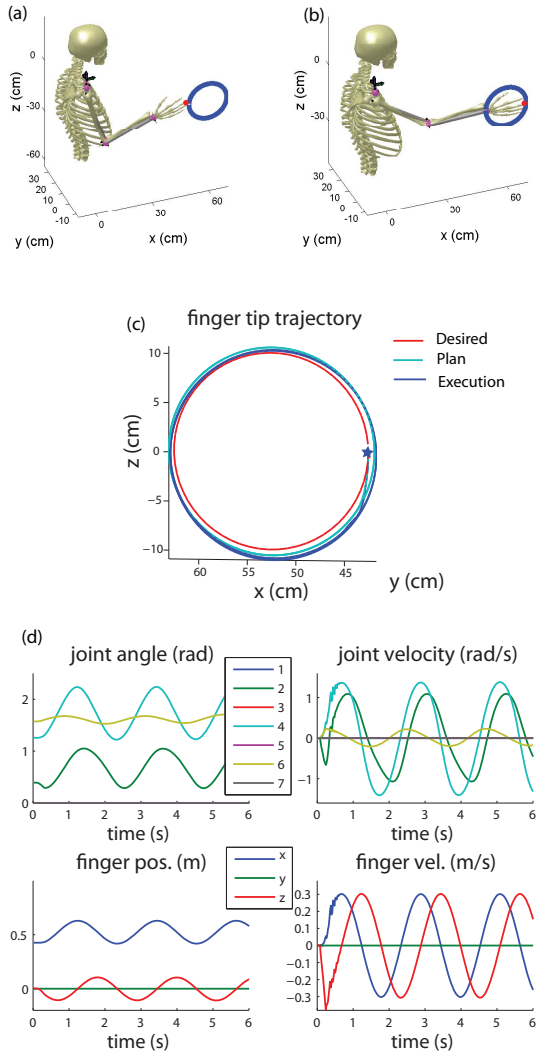

Fig. 9. Arm movements during drawing. **(a) (b)**, arm configuration at the beginning of and during drawing. **(c)**, end-effector trajectory during both planning and execution. **(d)**, end-effector position/velocity in Cartesian hand coordinates, and joint angle/velocity in joint coordinates.

than second order, whereas our hierarchical framework can deal with third-order systems without any modification. This makes it possible for our hierarchical framework to accommodate the true musculoskeletal dynamics of human body. Comparing with feedback linearization (FL) [15], [16] which usually assumes equal numbers of inputs and outputs, our hierarchical framework is able to deal with the mismatch in dimensionality and thus solve the redundancy problem with many more inputs than task-relevant outputs. In addition, instead of augmenting y to handle the mismatch between the high-level and low-level dynamics, our hierarchical controller uses the predictive version $\tilde{\mathbf{a}}$ and $\tilde{B}$ without increasing the dimensionality of the high-level states. Most importantly, OS usually assumes linear dynamics in the high-level and FL tries to linearize the high-level dynamics, both to some extend ignore the complex nonlinearity in the low-level dynamics. Our hierarchical framework, in contrast, aims to have the high-level controller exploit the natural plant dynamics and yet operate on reduced dimensionality to better pursue optimality.

REFERENCES

[1] E. Kandel, J. Schwartz, and T. Jessell, *Principles of Neural Science*. McGraw-Hill Education, 2000.

[2] E. Todorov, W. Li, and X. Pan, "From task parameters to motor synergies: A hierarchical framework for approximately-optimal feedback control of redundant manipulators," *Journal of Robotic Systems*, vol. 22, pp. 691–710, 2005.

[3] E. Todorov and M. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature Neuroscience*, vol. 5, no. 11, pp. 1226–1235, 2002.

[4] E. Todorov and M. Jordan, "A minimal intervention principle for coordinated movement," in *Advances in Neural Information Processing Systems 15* (S. Becker, S. Thrun, and K. Obermayer, eds.), pp. 27–34, Cambridge, MA: MIT Press, 2002.

[5] N. Bernstein, "Dexterity and its development," in *Dexterity and its development* (M. Latash and M. Turvey, eds.), Lawrence Erlbaum, 1996.

[6] G. Loeb, I. Brown, and E. Cheng, "A hierarchical foundation for models of sensorimotor control," *Exp Brain Res*, vol. 126, no. 1, pp. 1–18, 1999.

[7] J. Kalaska, L. Sergio, and P. Cisek, "Cortical control of whole-arm motor tasks," in *Sensory guidance of movement: Novartis Foundation Symposium* (M. Glickstein, ed.), Chichester, UK: John Wiley and Sons, 1998.

[8] W. Li and E. Todorov, "Iterative linear-quadratic regulator design for nonlinear biological movement systems," in *1st International Conference on Informatics in Control, Automation and Robotics*, 2004.

[9] I. Brown, S. Scott, and G. Loeb, "Mechanics of feline soleus: II. design and validation of a mathematical model," *Journal of Muscle Research and Cell Motility*, vol. 17, pp. 219–223, 1996.

[10] P. Morasso, "Spatial control of arm movements," *Exp Brain Res*, vol. 42, pp. 223–227, 1981.

[11] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference*, 2005.

[12] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," *American Control Conference*, 2005.

[13] P. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *Journal of Motor Behavior*, vol. 21, pp. 5–19, 1954.

[14] O. Khatib, "A unified approach to motion and force control of robotic manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 1, pp. 43–53, 1987.

[15] A. Isidori, *Nonlinear Control Systems*. London: Springer-Verlag, 1995.

[16] H. Khalil, *Nonlinear Systems*. New Jersey: Prentice-Hall, 2002.