Muscle-Based Control For Character Animation
Ana Lucia Cruz Ruiz, Charles Pontonnier, Nicolas Pronost, Georges Dumont

To cite this version:

HAL Id: hal-01317881
https://inria.hal.science/hal-01317881
Submitted on 18 May 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Muscle-Based Control For Character Animation

A. L. Cruz Ruiz 1,2, C. Pontonnier 1,2,3, N. Pronost 4, G. Dumont 1,2
1IRISA/INRIA MimeTIC, Rennes, France
2ENS Rennes, Bruz, France
3Ecoles de Saint-Cyr Coëtquidan, Guer, France
4Université Claude Bernard Lyon 1 - LIRIS SAARA, Lyon, France

Abstract
Muscle-based control is transforming the field of physics-based character animation through the integration of knowledge from neuroscience, biomechanics, and robotics, which enhance motion realism. Since any physics-based animation system can be extended to a muscle actuated system, the possibilities of growth are tremendous. However, modeling muscles and their control remains a difficult challenge. We present an organized review of over a decade of research in muscle-based control for character animation, its fundamental concepts and future directions for development. The core of this review contains a classification of control methods, tables summarizing their key aspects, and popular neuromuscular functions used within these controllers, all with the purpose of providing the reader with an overview of the field.

1. Introduction
Character animation is the art of bringing virtual characters to life through the design of solutions, such as motion controllers, that allow the reproduction and/or synthesis of new motions. The way these solutions are designed depends on the requirements of the specific application for which the character will be used, such as the degree of realism. We define realism as the visual degree of similarity between the actions, motions and responses of virtual characters with those of their real counterparts, at both the dynamic and kinematic level. The desired degree of realism also varies depending on the application. For instance, in a game or simulation, a higher degree of realism is sought for the main characters as opposed to background characters.

In this review, we present a recent, but growing trend, for the production of realistic character animations: muscle-based control. This solution entails the use of more detailed character models, involving muscles and their controllers. Muscle-based models and controllers already span a variety of areas, such as: ergonomic design, rehabilitation therapies, prosthetics, medical diagnosis and even post-surgery predictions. In animation, their usefulness no doubt depends on the requirements described above, along with the time to setup the model, and the desired motion repertoire. Nevertheless, we will show that it is a promising direction for the field of character animation in terms of enhancing motion realism. Thus, the main objective of the review is to provide animators involved in motion control, and experts from other fields (such as robotics and biomechanics) with similar interests, with a wide overview of the field of muscle-based control for animation.

Let us first introduce muscle-based control from a historical point of view. Throughout the years, several solutions have been proposed to mimic how humans and animals control their motions in order to animate virtual characters [Wil87]. These solutions are split into two main approaches: kinematic-based methods and physics-based methods. The former involves animating characters by specifying limb or joint trajectories. The latter involves animating characters by specifying actuator trajectories, such as joint torques or muscle signals.

One of the earliest kinematics-based methods was keyframing. In this technique the user specified a sequence of positions and their corresponding times, later a computer made an interpolation (usually a splining technique) between the specified positions to generate motion. However, this implied a very low level control, where the user had to control each degree of freedom of the character. Another technique was the use of control functions, where the motion for each degree of freedom was specified via functions of position versus time, these functions were generally curves composed of a set of control points. This technique had the advantage that the changes could be more easily made on individual degree of freedoms, but the control was still very low
level. Later, with the arrival of motion capture systems, animators began using recorded human kinematic data to drive or enhance animations [PB02] [Gle98]. One major drawback of this technique was that motion diversity and quality were limited and conditioned by a motion database. Nevertheless, some approaches have extended and diversified the number of motions beyond those of the original databases by applying external forces on the characters and making dynamic corrections on the original motion [MKHK08] [PD07].

A different approach to animation is physics-based animation. This technique is based on the development of physics simulators, which aim at replicating real environments by modeling the physical laws and conditions that define it. This has freed animators from worrying about enforcing certain motion characteristics which come implicitly with the presence of physics, and has granted virtual characters with a freedom of motion, that is unrestricted but physically plausible. Once these virtual environments are created, the animator should choose what character will be used for the animation and a strategy for motion synthesis (the strategy for the design of a motion controller).

The character model can exhibit different levels of detail in terms of skeleton, actuators and tissues etc. (as will be explained in section 3). A common choice has been simple skeletons actuated by ideal servo-motors, and commanded by servo-based controllers. A very complete review and categorization of these controllers can be found in [GP12]. The animation systems discussed in the latter review featured motions ranging from 2D locomotion [Hod91] [vdPF93] and forward flips [HR90], 3D locomotion [RM01] [RH02] [Sim94], balance [YLvdP07] [CB-vdP10] [CKJ’11] [LKL10] [AdSP07] [MZS09] [WZ10], and navigation on uneven terrain [WP10] [MdLH10].

Despite these advances, several authors from areas, such as biomechanics, have shown the importance of increasing the level of detail in such models by including muscles as actuators (in the place of servos), and performing a muscle-based control. This has triggered an evolution from servo-based control, to servo-muscle-based control and finally to muscle-based control. Servo-based control assumes that the degrees of freedom (DoF) of the character are actuated by servo motors, and therefore produces torques. Servo-muscle-based control assumes that a set of degrees of freedom is actuated by servos, while another set is actuated by muscles, and consequently it produces a set of torques and a set muscle signals. Finally, muscle-based control assumes that all the joints of the character are actuated by muscles and produces muscle signals only.

Figure 1 depicts how muscle-based control is integrated in a physics-based framework. In this diagram, the physics simulator updates the state of the character and the environment as a result of the external forces in the environment and internal forces in the character. The external forces in the environment include gravity, force perturbations, and forces due to obstacles, slopes, and changes in the friction coefficient. The internal forces in the character are the muscle, ligament, and joint-contact forces, which are generated thanks to the muscle signals produced by the muscle-based control method. To design these control methods, knowledge in neuromusculoskeletal simulation is vital since it provides a basis on which to: model the characters, automate the motion generation process, and solve the motion redundancy problem. More information on these topics will be featured in section 3.

The simulator first gathers the internal and external forces, and executes a forward dynamics simulation that helps to compute the resulting state of the character. This state comprises several kinematic and muscle variables, such as the position and velocity of each link (or sub-body) in the character, and the length and velocity of shortening or lengthening of each muscle. Specifically, the simulator first computes the link accelerations. These are then fed to a numerical integrator that updates the link positions and velocities, which are finally used to update the state of the muscles.

As previously mentioned, the main contribution of this review is providing the reader with an overview of the trends in the field of muscle-based control for animation. We begin by explaining our motivation in section 2. In section 3 we present an overview of the neuromusculoskeletal simulation tools, used to design control laws, such as musculoskeletal modeling, simulated dynamics and muscle force estimation. The core of the review is section 4, where we feature a muscle-based control method classification consisting of two categories: controller optimization methods and trajectory optimization methods. This section also contains the main strengths and weaknesses of each method and examples of controllers found in the animation domain. Next, section 5 features comments on the current and future developments in the field of muscle-based animation. Finally, the review concludes with a set of appendixes including the most relevant aspects presented. Appendix 1 contains a table summarizing our classification and important control and model features. Appendix 2 contains a list of neuromuscular cost functions used by the controllers. Appendix 3 contains a short list of physics simulators and the controllers that use them.
Lastly, we would like to emphasize that the review aims at highlighting the contributions and novelties introduced by state of the art muscle-based controllers and the characters they use. Therefore, it provides brief but comprehensive descriptions of relevant controllers (with a special focus on their muscle-related components). It also provides a general discussion on musculoskeletal modeling, with thorough descriptions only of the models these controllers use. For more details on state of the art muscle modeling, the reader is invited to consult [LKG*10]. We would also like to underline the fact that most of the frameworks perform muscle-based control, however a few perform servo-muscle-based control, meaning that they involve characters actuated with both servos and muscles. In such frameworks we focus on the muscle controlled part.

2. Motivation

Several studies have shown the importance of considering internal forces when describing joint kinematics, specially in joints where complex interactions between muscle actions, soft tissues and cartilage exist (such as spinal disks, shoulders and knees) [ADR11]. Considering muscles, in particular, comprises several advantages: better stability properties and more realistic passive dynamics (section 2.1), better estimates of energy cost or fatigue (section 2.2), efficient control via motion mechanics (section 2.3) and an ease to simulate musculoskeletal defects, pathologies and physical fatigue (section 2.4). These advantages are a consequence of the non-linear properties present within the muscles, such as the force-length and force-velocity relationships (these properties are detailed in section 3.1).

Moreover, muscles are at the center of important motor control theories. For instance, the spring-like behavior of muscles has been a crucial part of motor control theories such as muscle synergies (both theories will be further described in section 3.4). Other authors have demonstrated that specific mechanical behaviors observed during walking can be encoded as muscle reflexes, which were used to reproduce human walking, without any higher level controller. These reflexes were inspired in spinal reflexes, which link sensory information directly into muscle activations, bypassing the inputs from the central nervous system (such reflexes will be further described in section 3.4). Other authors have demonstrated that specific mechanical behaviors observed during walking can be encoded in a single, simple muscle reflex [PGB97] [GBS03]. Nevertheless, more evidence of the performance of such legged mechanics principles, in terms of walking on uneven terrain and in various directions, is still needed. This evidence is needed to show the extent to which reflexes can deal with such tasks without a higher level controller.

2.2. Physiological feasibility and energy estimates

The inclusion of muscles motivates physiologically feasible motions. An example of this can be found in [KSK00], where the physiological infeasibility of interpolating user-input postures was shown, and it was later reduced based on muscle dynamics. Muscles also provide better estimates of energy expenditure. In [WHDK12], visual, kinematic, and dynamic comparisons evidenced that walking motions synthesized via energy estimates using muscles were closer to real human data than the estimates based on torques. Recently, comparisons have also been made between torque actuated and muscle actuated simulations of human swimming [Si13]. The former yielded plausible results but high control gains and a smaller numerical time step were necessary.

2.3. Control via motion mechanics

Thanks to the presence of muscles, the mechanical system of the character is granted with the ability to accomplish control functions, not only for counteracting perturbations, but also for tasks such as human walking. For instance, instead of trying to create control models that mimic complex neural circuits (for either torque or muscle-based characters), biomechanists have discovered that locomotion requires little control if certain principles of legged mechanics are used. In [GH10], these principles were encoded as muscle reflexes, which were used to reproduce human walking, without any higher level controller. These reflexes were inspired in spinal reflexes, which link sensory information directly into muscle activations, bypassing the inputs from the central nervous system (such reflexes will be further described in section 3.4). Other authors have demonstrated that specific mechanical behaviors observed during walking can be encoded in a single, simple muscle reflex [PGB97] [GBS03]. Nevertheless, more evidence of the performance of such legged mechanics principles, in terms of walking on uneven terrain and in various directions, is still needed. This evidence is needed to show the extent to which reflexes can deal with such tasks without a higher level controller.
2.4. Simulation of musculoskeletal defects, pathologies and physical fatigue

Among other advantages of muscle-based control is the fact that muscles provide a natural solution to the simulation of musculoskeletal defects, pathologies and physical fatigue. By taking into account an anatomical structure (musculoskeletal system), it is easier to simulate phenomena that derive from this structure. For instance, fatigue and recovery muscle models can be used to simulate a motion where a human gradually gets tired [KSK00] by limiting the maximal force that the muscle can produce with respect to the history of muscle force [GML93] [GML96]. Cost functions that minimize the force of a specific muscle can be used to synthesize motions with pain avoidance behaviors [LPKL14]. Changing muscle parameters and properties such as maximal strength, can be used to weaken muscles, and generate well known pathologies and defects [WHDK12]. Finally, injuries can also be simulated by displacing muscles [KSK00].

3. Neuromusculoskeletal simulation overview

Humans are actuated by muscles controlled by the central nervous system. These muscles produce forces that actuate joints to achieve a given motion. The motion is most of the time realized under external perturbations or forces (such as a voluntary pushes, the force of gravity, and ground reaction forces), which the central nervous system should also compensate for.

In the biomechanics field, the neuromusculoskeletal simulation can either consist in finding ways to estimate the muscle forces from a prescribed motion or directly generating motion from computed forces. The link with physics-based animation is straightforward: by applying to a musculoskeletal model (section 3.1), an optimal set of muscle forces (section 3.3) with regard to motion and other requirements, it should achieve realistically a specified motion.

The following section features a general review of important tools and concepts to understand in a neuromusculoskeletal simulation: musculoskeletal modeling, simulated dynamics, and muscle forces estimation. Most of the references cited here come from the field of biomechanics, as biomechanicians are the historical actors of development of the neuromusculoskeletal simulations. However, several openings and applications to animation are highlighted in order to show the strong link between both fields.

3.1. Musculoskeletal modeling

Simulating physics in a system means specifying the segments, joints, masses, inertias and actuation capacities of these systems. Musculoskeletal modeling consists in describing these different features in a convenient way in order to enable a physics simulation. This modeling is common to different types of virtual characters, such as animals, humans and even imaginary creatures.

3.1.1. Joints and segments

The core problem of musculoskeletal modeling is the definition of anatomically realistic segments and joints. In biomechanics, the models are often anatomically based [WSA∗02] [WvdHV∗05] and exhibit a higher level of detail than those used in animation, as they have to be accurate enough to provide clinically relevant biomechanical quantities. In many cases, kinematical closed chains appear in the structure of the model and make both kinematics and dynamics studies more complex [PSV07] [VDH94].

In the animation field, functional degrees of freedom (i.e. the resulting motion of the anatomical ones) are often used as a basis for the kinematical model [h-a] and segments directly link the considered articular centers. Nevertheless, some approaches have begun using detailed anatomically based models of certain body sections such as the neck [LT06], hand [SSB∗15], and the upper-body [Lee08]. Using complete body models still remains an area of growth, with only a handful of models presented for motion synthesis purposes, from which the models presented by [Si13] [LPKL14] stand out.

3.1.2. Lengths, masses and inertias

One of the main issues in musculoskeletal modeling is the scaling of the model. As motion is generally recorded to be analyzed (using tracking or motion capture systems), the lengths of the segments are easily computed from marker positions [LadZR11]. However, when this information is not available, regression laws based on cadaver measurements [Dem55] can be used. Cadavers were also used to scale masses and inertias [dL96] [DCV07]. However, advances in medical imaging, (e.g. scanner or MRI scanners) opened the door to subject-specific scaling of musculoskeletal models and showed interesting perspectives for clinical applications [BAGD07]. Moreover, other approaches have emerged which rely only on exterior measurements of the body, such as 3D point clouds, to determine subject-specific bone geometry and motion [ZHK15].

In the models used by the control methods in this review, the parameters are manually specified for every new character. However, some exceptions exist, such as [GvdPvdS13] and [HMOA03], which automatically computed the muscle parameters by including them in optimization procedures.

3.1.3. Servos and muscles

Historically, as shown in figure 2 (left), muscles were not represented and the simulation was mainly skeletal. Estimation of joint torques with servo actuated joints was the main goal, but provided no relevant information about muscle load and fatigue. Progressively, muscles were incorporated in the simulation models as non-direct actuators of the joints.
This has also been the case for animation, where characters have evolved in level of detail, and muscle models are finding their way into motion control frameworks. A more straightforward inclusion of muscles began with the use of mass-spring systems. Contrary to the usual angular-spring dampers and PD-controllers, these systems (like muscles) are non-direct actuators of the joints. Meaning, they produce first forces, not torques, which interact with the skeletal system to produce motion. Like muscles, these systems use force action lines, determined by their insertion or attachment site to the skeletal structure. Nevertheless, nowadays, more faithful muscle representations, such as biomechanical muscle models, are actively used.

One popular biomechanical muscle model that has been incorporated into virtual characters for force generation is the Hill muscle model [Hil38]. Although this model was developed decades ago, its current usefulness is evidenced by its various adaptations and implementations within the biomechanics community. These adaptations have come to be known as Hill-type models, such as the Hill-Stroeve model [Str96], and the widely used adaptation made in the late eighties by [Zaj89], for numerical simulations. As shown in figure 3, the model consists of a contractile element CE (non-linear visco-elastic relationship) in parallel with a passive element PE (non-linear spring). The contractile element represents the active tension, or forces, created by the contractile proteins in the muscle, while the passive element represents the passive tension or the force that results from the elongation of the connective tissue components in the musculotendon unit. The tendon is represented by a serial non-linear spring SE of length I_t, \( \alpha \) represents the pennation angle or the orientation of the fibers with regard to the tendon, \( I_m \) represents the muscle length, and \( I_{mt} \) the length of musculotendon unit. The latter is computed by adding the muscle \( I_m \) and tendon \( I_t \) lengths. This model has been widely used even if the numerous parameters necessary to completely define its behavior are difficult to obtain in vivo [HKvdH07] [IEC10].

The muscle force generation \( F_m \) of a musculotendon unit \( j \) can be summarized as the sum of the contractile and passive forces:

\[
F_{mj} = [f_p(I_m) + a_j \cdot f_t(I_m) \cdot f_v(I_{mt})] \cdot F_{0j} \]  

(1)

where \( f_p \) is the passive force relationship, \( a_j \) is the muscle activation, \( f_t \) is the force-length relationship, \( f_v \) the force-velocity relationship, \( F_{0j} \) the maximum isometric force, and \( I_{mt} \) the normalized length of the muscle unit (normalization is usually made using the resting length of the muscle). Several models have been proposed to approximate the \( f_t \) and \( f_v \) relationships with regard to experimental data [RAPC10]. Example models are presented in figure 4. The force-length relationship documents how muscle tension varies at different muscle lengths, and it is related to the "Sliding Filament Theory". At a microscopic level, muscle fibers are composed of smaller structures called actin and myosin filaments that make bindings to form muscle contractions. Peak muscle force can be generated when most of these bindings or cross-bridges are created. This event corresponds to the resting length of the muscle (usually near the middle of the range of motion) [Knu07]. The force-velocity relationship explains how the force of fully activated muscle varies with velocity. It states that the force the muscle can create decreases with increasing velocity of shortening (concentric actions), while the force the muscle can resist increases with increasing velocity of lengthening (eccentric actions) [Knu07].

The tendon force \( f_{jt} \), output of the musculotendon unit, is simply obtained by taking into account the pennation angle:

\[
f_{jt} = F_{mj} \cdot \cos \alpha_j \]  

(2)

However, in many studies the pennation angles are neglected.

Complete dynamics of the musculotendon unit also includes the activation dynamics, meaning that there is a non-linear temporal relationship between the neural excitation \( u_j \) and the effective activation of the muscle [BLMB04]. In many works [VYN05] [VYN06] [PD09], this non-linear relationship is approximated by a second order differential
equation, exhibiting different time constants for activation and deactivation:

\[
\dot{a}_j = \frac{(u_j - e_j)}{\tau_{ne}}, \quad e \geq a
\]

\[
\ddot{a}_j = \begin{cases} 
\frac{(e_j - a_j)}{\tau_{act}}, & e \geq a \\
\frac{(e_j - a_j)}{\tau_{deact}}, & e < a 
\end{cases}
\] (3)

Where \( u_j \) is the neural excitation, \( e_j \) the muscle activation, \( a_j \) an intermediate variable, \( \tau_{ne} \) the neural excitation constant time (often neglected), and \( \tau_{act} \) and \( \tau_{deact} \) the activation and deactivation time constants respectively. In animation, activation dynamics is sometimes modeled using equal activation and deactivation time constants [Gvd-PvdS13] [WHDK12].

The musculotendon unit modeling remains challenging, since changes in the tendon length affect the final muscle force, and vice-versa. A proper evaluation of the muscle length should be done [MD12]. However, such a computation is costly in terms of computation time and is often simplified. For example, the algorithm used in [DRC*06] just iterates once at each simulation time step, assuming that most of the tendon effect will be obtained with only one iteration.

Besides the mechanical model presented previously, other muscle models have emerged which encompass visual characteristics such as muscle deformation or both functional and visual characteristics [LGK*10]. The models can be grouped under three techniques: geometrically-based, physically-based and data-driven approaches. In geometrically-based approaches, muscle deformation is determined by the skeleton arrangement [CHP89] [WVG97] [TSC06]. In physically-based approaches both the contractile muscle forces and the changing muscle geometry are represented during contraction [NTH01] [TZT09] [TSB*05]. Finally, data-driven approaches directly model the skin shape that is deformed by the underlying muscle, thanks to data captured from the surface of subjects [ACP03] [PH06] [FLP14]. These models offer a next level of fidelity. However to this date they are not usually used for the control of virtual characters due to the fact that they would render the control computationally expensive.

3.2. Simulated dynamics

As it has been stated in [EMHvdB07], the musculoskeletal dynamics problem can be presented as follows: let us consider a musculoskeletal system with \( n \) degrees of freedom, actuated by \( m \) muscles. The degrees of freedom are the joint angles gathered in a vector called \( q \). The state of such a musculoskeletal model, from a dynamics point of view, can be expressed as \( (q, \dot{q}) \).

The relationship between motion and forces is given by the Newton’s second law of motion, that can be expressed in a matrix form as [Pan01]:

\[
M(q)\ddot{q} + C(q, \dot{q}) + G(q) + R(q)F_m + E = 0 \quad (4)
\]

Where \( M(q) \) is the mass matrix of the system, gathering masses and inertias of all the segments \( (n \times n) \), \( C(q, \dot{q}) \) represents the coriolis and centrifugal effects \( (n \times 1) \), \( G(q) \) represents the vector of gravity torques \( (n \times 1) \), and \( E \) represents the external forces. Finally, \( R(q)F_m \) represents the action of the muscles on the joints (muscular joint torques, \( n \times 1 \)), where \( R(q) \) is the matrix containing the muscular moment arms \( (n \times m) \) and \( F_m \) the muscle forces \( (m \times 1) \). In a dynamics simulation, such quantities can be automatically constructed by using algorithms such as the ones developed in [KL96] [Fea14].

From equation 4, two different problems can be derived. The first one, called inverse dynamics, consists in applying a specified motion and specified external forces to a musculoskeletal model and then computing the forces that generate the considered motion. The equation to solve is a reformulation of equation 4 and can be expressed as:

\[
R(q)F_m = -(M(q)\ddot{q} + C(q, \dot{q}) + G(q) + E) \quad (5)
\]

The output of equation 5 are the muscular joint torques that are used to define muscle forces. This equation is most of the time solved thanks to a top-down strategy also referred to as Newton-Euler algorithm [Win05] [Fea14] [RHWZ08], which considers each segment separately from distal to proximal. However, more robust methods and methods considering closed loops have also been validated to solve the inverse dynamics problem [Kuo98] [vdBS08]. This approach is widely used in motion analysis, as motion capture data is
a very common resource and can be used as a kinematical input in such problems.

The second problem is called forward dynamics and is the one that interests us the most, as it consists in generating motion from computed forces. Since no direct measurement of muscle forces is available, this approach is generally coupled with an optimization problem to compute a set of forces compatible with a given task. The equation to solve, issued from equation 4 can be written as follows:

$$\ddot{q} = M^{-1}(q)(-C(q, \dot{q}) - G(q) - R(q)F_m - E)$$  \hspace{1cm} (6)

In animation, the forward dynamics problem is incorporated in a physics simulation involving collision detection (that provides external forces to apply to the system) and a numerical integration (e.g. Runge-Kutta methods) to obtain the current system state $$(q, \dot{q})$$ from the computed accelerations $$\ddot{q}$$. In biomechanics, the problem is generally solved with real external force measurements (such as ground reaction forces from force plates), and it is formulated as the optimization problem presented in the next section.

### 3.3. Muscle forces estimation

Most musculoskeletal models exhibit actuation redundancy (\(m > n\)) that leads to a number of actuation solutions, as there are less equations (dynamics equations) than unknowns (muscle forces). The models may also exhibit underactuation, which stems from the fact that a single muscle can actuate several joints simultaneously, such as bi-articular muscles.

These challenges can be solved by defining what is the optimal actuation solution, through the modeling of known motor control laws associated to a motion. Motor control is the process through which humans and animals create motions by using their neuromuscular system. This process involves the computation of higher level commands by the central nervous system to achieve specific motion goals based on sensory information regarding the environment and the current body state. These commands later excite the muscular system, creating skeletal motion [Ros91]. Several models that mimic this process are actively studied and used in the fields of neuroscience, biomechanics, and robotics to generate muscle forces.

#### 3.3.1. Problem formulation

A popular motor control model is the minimization of a cost function \(f(X)\) encoding task goals and bio-inspired objectives that motivate natural motion. A common formulation of this model consists in a non-linear constrained optimization problem:

$$\text{Find } X \text{ which minimizes } f(X)$$ \hspace{1cm} (7)

subject to,

$$g_j(X) \leq 0, j = 1, 2, ..., m$$
$$h_k(X) = 0, k = 1, 2, ..., p$$

Where the constraints enforce that:

- the computed forces solve the dynamic equations;
- the muscles are only pulling and they have physiological-based force limits;
- the computed forces may respect any additional set of unilateral \((g)\) or bilateral constraints \((h)\).

The constraints \(g_j(X)\) and \(h_k(X)\) may be specified as hard constraints (as in the formulation above), or as soft constraints (as additional cost functions). The optimization can be a static or dynamic one (often an optimal control problem [ZDG*06]). A static optimization refers to the process of minimizing or maximizing an objective function at a time instant, while a dynamic optimization refers to the process of minimizing or maximizing an objective function over an interval of time of non-zero duration.

### 3.3.2. Forward, inverse and hybrid dynamics-based optimization

This model can be used both in a inverse dynamics and forward dynamics framework. General schemes of these frameworks are featured in figure 5 and in figure 6.

In the inverse dynamics-based optimization (figure 5), motion and external forces are applied to a musculoskeletal model. The generated joint torques are then used in an optimization procedure that computes the muscle forces that satisfy the task, motion objectives, and constraints. In biomechanics, the inverse dynamics problem is often solved by minimizing a cost function representing an energetic cost [TBB97] [Pan01] [EMHvdB07] [PD09]. Several cost functions have been tested. The sum of the squared and cubed muscle forces have been classically used for gait [CB81] and upper arm [AKCM84] [Cha97] motions as an image of the metabolic energy consumption, whereas a min/max criterion [RDV01] has been widely used in ergonomics applications as an image of the muscle fatigue. These examples are of importance as they are influencing the cost functions that are currently used in the muscle-based animation field. A widely used commercial software exploiting such a method for musculoskeletal analysis is AnyBody [DRC*06]. One of the purposes of using inverse dynamics-based optimization for animation is to provide characters with a certain degree of adaptability to perturbations. Computing muscle forces from motion capture data is also interesting because it allows the simulation of new motions (e.g. fatigued or injured motions), by replaying the computed muscle forces while altering muscle parameters [KSK00] [WHDK12] [LPKL14]. Moreover, recent developments allow the computation of these muscle forces in real-time [vdBG13]. Finally, it is worth noting that an alternative strategy to muscle excitation computation is
the muscle-based variation of jacobian transpose control [SADM94] [GvdPvdS13], which does not use the complete inverse dynamic model, but a simpler, static equivalent. Essentially, it consists in finding a set of forces and torques in the task space which achieve a target pose, and converting these into individual muscle torques and excitations via the jacobian transpose.

In the forward dynamics-based optimization problem (figure 6), only initial muscle excitations and external forces are applied to the model (no pre-recorded motion). The desired motion is instead directly provided to the optimization procedure to evaluate task achievement, and produce the required muscle excitations. Usual cost functions are the distance between computed kinematics (issued from the forward dynamics problem) and experimental kinematics data. A good example of a forward dynamics optimization is the Computed Muscle Control algorithm (CMC) implemented in OpenSim [DAA∗07]. Its forward simulation uses a cost function that can be based on the distance to experimental data, or on the distance to predefined poses issued from a planner.

Several Hybrid dynamics methods also exist, trying to use the advantages of both methods to be mechanistically and physiologically consistent. These methods basically consist of a inverse dynamics-based optimization, with the difference that (as in the forwards dynamics problem) initial excitations are provided for some muscles through the use of electromyographic measurements [BLMB04] [AM04] [ARB10]. The advantage of such approach is that the dimension of the original problem is reduced by removing variables from the optimization problem. Numerous optimal control methods have also been developed to obtain realistic motions from scratch through the actuation of a musculoskeletal model [Pan01] [AvdB10]. This is also of great interest for the animation field, as optimal control theory is especially well fitted to synthesize a motion between two body poses.

As previously shown, a central and important component in these frameworks is the choice of the objective function. For this reason, we have grouped the most relevant objective terms used in animation according to the categorization found in [ZW90]. In this categorization, a generalized performance criterion was proposed, which included three components: task specific (tracking a given trajectory, minimizing jerk), neuromuscular (minimizing muscle stress, neural effort), and bone joint (minimizing contact forces, avoiding certain ranges of motion) objectives. Appendix 2 contains a detailed description of each of these components and a table listing relevant cost functions. A special focus has been given to the neuromuscular objectives, which represent a novelty in muscle-based control. The reader is invited to consult the reference of each controller for more detailed descriptions on the remaining objective types.

To solve the optimization problem, popular algorithms such as sequential quadratic programming (SQP) [KSK00] [ZCCD06] [Si13] and simplex methods have been implemented. SQP consists in modeling the non-linear optimization problem as quadratic subproblems and to use the solutions of these subproblems to find better approximations that lead to the optimum [RR09]. If the optimization is unconstrained, other authors have opted for simpler methods such as the simplex method [GT95] [DZS08]. This iterative method uses a geometric shape or simplex to explore the solution space and find an optimum [RR09].

Evolutionary algorithms [dG04] have also been recently incorporated into the control of virtual characters [HMOA03] [WHDK12] [GvdPvdS13]. An example is the covariance matrix adaptation (CMA) [Han06], which uses a multi-variate normal distribution to explore the solution space in search of an optimum.

### 3.4. Motor Control Theories

The optimization itself can be used as a controller to generate muscle signals, or it can be used to optimize bio-inspired control laws based on motor control findings and theories (Section 4.1) such as: hierarchical systems, central pattern generators, equilibrium point theory, muscle reflexes and muscle synergies. The current section briefly presents these theories and some of the application cases found in both biomechanics and animation.

Hierarchical control systems, have been designed thanks to studies [KSJ∗00] that outline how the components or neural organs of the motor system work together to generate muscle excitations for voluntary and reflexive actions. This hierarchy has inspired multiple level controllers in animation, as in [ZCCD06].

Central pattern generators (CPGs) have been proposed for the generation of locomotion, and other kinds of behaviors (such as respiration and swallowing). CPGs are biological neural networks that produce rhythmic patterns without relying on sensory feedback or higher control centers [Hoo01]. This neural rhythmicity is generated from interactions between neurons or between currents within individual neurons. Although these networks do not rely on sensory feedback, higher control centers use this feedback to modulate the CPG outputs. The CPG models created by [Tag98] [TY91] for bipedal locomotion have been popular in the robotics domain [Ijs08] [AT05] [AT06] [ENMC05] and are also beginning to be present in the world of animation, specifically for the synthesis of human swimming [Si13] and walking [HMOA03]. Nevertheless, a limitation of these generators is the fact that they only produce a limited set of motions, mainly rhythmic or periodic patterns. Using CPG-based methods alone is not enough to provide a character with a rich motion repertoire. For this reason, in applications requiring periodic and non-periodic motions, it is necessary to use an additional control strategy to handle the non-periodic aspects [Si13].
Control laws based on the equilibrium point theory [Fel66] have been also implemented by animators [NF02]. This theory argues that the nervous system controls movement through the specification of final equilibrium positions of the limb. The equilibrium trajectory mirrors properties of the limb and neuromuscular system, such as inertia and viscoelasticity. This trajectory is specified by virtual positions corresponding to variations in muscular activations. The muscle activations move the limb from its real position to the virtual one.

Lower level control laws, such as muscle reflex models [GH10] have recently started to be incorporated into character motion synthesis, as we will see in the next section [GvdPvdS13] [WHDK12]. The models suggested that reflex inputs (which serve as mediators between the CNS or central nervous system and mechanical environment) dominate in contributions to muscle activations during locomotion. This supports the idea that the function of CPGs may be limited during normal locomotion. These reflexes have been included in animation as positive feedbacks of muscle fiber length and force [GBS03] that altered muscle activation. The effects of such reflexes was a reliance on compliant leg behavior, joint overextension avoidance, and improved gait stability.

Finally, a promising theory (somewhat close to the idea behind CPGs), that is yet to be actively applied within animation frameworks, is the theory of muscle synergies. This theory proposes that by combining a few modules, the CNS may learn new control policies fast and efficiently, for instance, to adapt to perturbations. Evidence of this modular organization has been found thanks to the low dimensionality of motor signals [dP10]. The existence of
such modules has been shown in human arm reaching motions [MBdF10], human postural responses [TOT07], overhead throwing [CRPSD15], and frog kicks [dSB03].

All of these theories are of interest since they have inspired many of the control methods used in animation. In section 4.1 we will see how central pattern generators, muscle reflexes, and the equilibrium point theory have inspired controllers in animation. In section 4.2 we will show other control methods which take more inspiration on the work done in the domain of control and robotics. Nevertheless, a handful of these methods have also been inspired by motor control theories, such as the theory of muscle synergies.

4. Muscle-based control methods

Once the character is designed using the musculoskeletal modeling and simulation dynamics concepts presented in section 3.1 and section 3.2, a muscle-based controller can be constructed using the force estimation frameworks and techniques featured in section 3.3, particularly, the forward dynamics-based optimization framework. As initially shown in figure 1, the purpose of the muscle-based controller is the computation of adequate muscle signals (muscle excitations or muscle forces) that allow the character to achieve a set of tasks and motion goals. Specifically, we are interested in defining a controller to determine the forces \( F_m \) in equation 6.

The motion goals for these controllers can consist of high-level goals, such as walking speed and direction, or task space and joint trajectories. Therefore, the specification of detailed motion data is not a necessity. In fact the purpose of designing controllers is to be as independent as possible from motion data. In the first case, one sole procedure computes both kinematics and muscle signals from the high-level goal. In the latter case, either a higher level controller computes the desired kinematics and a low-level muscle controller transforms it into muscle signals, or the animator provides the kinematic trajectories.

We have grouped the controllers into two categories: controller optimization methods (section 4.1) and trajectory optimization methods (section 4.2). In the controller optimization methods, the optimization seeks to determine the optimal control parameters that will allow a control law to produce muscle signals that satisfy specific motion goals. Once these parameters are determined, the controllers convert desired kinematic goals into muscle signals. These control laws are based on the motor control findings and theories discussed in section 3.4. In trajectory optimization methods, the optimization directly generates the muscle signal trajectories that accomplish the desired motion goals. Both methods attempt to optimize motions with sometimes similar cost functions (Table 2), but the controller optimization methods access the equations of motion implicitly through experience. In other words they see the character and its environment as a "black box", while the trajectory optimization methods access these equations explicitly. Furthermore, the controller optimization approaches have the characteristic, that (when needed) instead of computing the complete inverse dynamic model, many use a "simpler" equivalent, such as the jacobian transpose [SADM94] (section 3.3.2). Finally, it is worth noting that to compute the muscle signals, the controller optimization methods perform the optimization offline, while the trajectory optimization methods perform it online.

A section has been devoted to each control method. Each section consists of a description of the control method, its main strengths and weaknesses, and examples from the animation field. Appendix 1 resumes these characteristics for all controllers discussed in this review, and appendix 2 summarizes the details and formulas describing the neuromuscular cost functions used by each controller.

4.1. Controller optimization methods

Controller optimization methods seek to determine a set of control parameters that will yield the desired motion goals throughout an entire period of time. These parameters depend on the specific control law, but in general they can be summarized as: feedback control law gains (such as PD controller gains, force feedback gains and spring gains) and CPG unit weights. An overview of such methods is featured in figure 7. Animators specify the desired task \( f_{task}(t) \), neuromuscular \( f_{neuromuscular}(t) \) and bone joint objectives \( f_{bone, joint}(t) \), constraints \((q,s_m,q_s)\), external forces \( E \), initial control \( p_{ini} \) and character state \( s_m \), and finally an initial guess of the joint trajectories that fulfill the task. An optimization procedure continually updates the controller parameters and desired joint trajectories until the cost function is minimized. It is worth noting that the final desired joint trajectories might be specified by the animator, or synthesized by the optimization as in figure 7.

Once the control parameters and joint trajectories have been determined, the control law executes online and directly generates the actuation signals that accomplish natural looking motions, while satisfying task related objectives. Example of control laws include: antagonistic control, PD-controllers, muscle reflexes, and neural networks. The variety of motions that have been synthesized encompass human locomotion (ex. walking, running, hopping) [GvdPvdS13] [WHDK12] and postural adjustments [NF02].

4.1.1. PD controllers and muscle reflexes

Several approaches have made use of the fact that humans minimize muscle effort during locomotion [R76] within their controller optimization frameworks. Some notable examples are the controllers developed by [WHDK12] and [GvdPvdS13] for synthesizing locomotion in humanoids and imaginary bipedal creatures. These approaches used an
optimization procedure to determine the optimal control parameters of PD controllers (PD gains) and muscle reflexes (force and length feedback gains). The optimization had a time horizon of 10s or 20s, and was based on a muscle effort term called the rate of metabolic energy expenditure [And99], and soft constraints to track kinematic objectives and ensure stable gaits.

The PD control laws and muscle reflexes generated the muscle excitations to make the character fulfill the locomotion tasks. [GvdPvdS13] used task space PD controllers on different body segments, inspired by the jacobian transpose control of [SADM94], while [WHDK12] employed them on each muscle in the character.

Figure 7: Controller Optimization Scheme. The tuning process, which is usually made over a group of time steps [GvdPvdS13], is iterated until the cost function is minimized or the desired motion is obtained (the tuning process may also consist of a locally weighted regression [Si13] or manual parameter and target specification [NF02]). The optimal parameters and joint trajectories are then used in an online closed control loop (instead of joint trajectories [WHDK12] desired muscle lengths [Si13] might also be used).

4.1.2. PD controllers and neural networks

Recently, PD controllers have also been included for synthesizing human swimming motions. The authors of [SLST14] [Si13] used a controller optimization method for periodic motions in swimming, and trajectory optimization (consult section 4.2) for non-periodic motions. The controller optimization method consisted of neural networks (central pattern generators or CPGs) for specific body sections and PD controllers. As opposed to the PD controllers in the previous sections, in this approach, the control gains were fixed, and the parameters in the optimization were a set of weights in CPG model.

The character’s body was divided into 10 muscle groups (ex: right leg, left leg muscles) for the CPG modeling. Each CPG was modeled as a set of non-linear differential equations, that contained desirable properties such as trajectory reproduction, modulation, and external perturbation compensation. The networks produced desired time-
varying muscle lengths for specific swimming modes, and the PD controllers converted them into muscle activations. The learning process of these networks was carried out by an Incremental Locally Weighted Regression (ILWR) [SA98]. This process sought the minimization of an error criterion based on the desired muscle lengths obtained from kinematic data of swimming.

The use of muscle groups, and a higher level controller to modulate each CPG, simplified the control task. For instance, turns were induced by decreasing the activation amplitudes of the muscles on one side of the body relative to muscles on the opposite side.

4.1.4. Summary

Similarly to controller optimization methods in the joint space, these controllers have generated impressive results in terms of skills (list of possible motions of the character) and robustness to external perturbations. However, because muscles are used instead of servos, new cost functions (such as muscle effort) have been included within the parameter optimization procedure, which further motivate motion realism. These improvements have been evidenced at both the kinematic and dynamic level by [WHDK12].

Controller optimization methods have allowed the implementation of models of biomechanical mechanisms. The use of these laws, such as muscle reflexes [GH10], have generated well-known events during walking, such as joint compliance in stance phase, and dorsiflexion during the swing phase [WHDK12] [GvdPvdS13]. This represents an important advantage, since bio-inspired control laws can directly generate desired muscle and joint behavior at specific stages during the motion.

One of its main drawbacks is computational efficiency. Computation times still need to be improved, since to synthesize 10 seconds of animation, the controllers yielding the most impressive results require approximately 10 hours of tuning [WHDK12]. Nevertheless, we believe that considering and modeling muscle groups and synergies (section 3.4) could aid in overcoming this setback, by explicitly establishing relationships among muscles and reducing the number of control parameters [ADN*13].

Finally, another interesting aspect of some of these control frameworks was the strategy used for ensuring balance of the characters. For instance, in the case of [GvdPvdS13], trunk stability was maintained by using feedback rules to control orientation in sagittal and coronal planes with respect to the center of mass velocity and a target heading. Balance was also enforced thanks to the application of a SIMBICON-style balance correction [YLvdP07] to determine leg orientation in the sagittal and coronal plane. The authors of [WHDK12] also made use of the SIMBICON balance feedback laws to adjust desired hip target angles. Lastly, the approach developed by [NF02] used automatic balance controllers based on ideas from the balancing simulations of [Woo98].

4.2. Trajectory Optimization Methods

The objective of trajectory optimization methods is to generate control variable trajectories that minimize or maximize a measure of performance while also respecting a set of constraints. Such methods are for now mostly used in robotics. In the domain of character animation, these methods remain mostly a static optimization, solving a given set of equations at each discrete step. Alternatively, they can also be referred to as "model-predictive control methods" when they are used online and with a finite time horizon.

As detailed in section 3.3, these optimizations are usually formulated as non-linear constrained optimization problems that use bio-inspired cost functions, such as muscle effort or muscle fatigue. A schematic overview of this approach is featured in figure 9. Generally, the animator specifies the desired task ($f_{task}(t)$), neuromuscular ($f_{neuromuscular}(t)$) and bone joint objectives ($f_{bone, joint}(t)$), external forces ($E$), and initial muscle signals ($s_m$). The optimization iterates until the cost function is minimized, producing the optimal muscle signals that fulfill the task.

Among the variety of motions that have been synthesized through this approach, are locomotion patterns [Mil88] [GT95] [GTH98] [KSK97] [MWTK13] [TTL12], and human upper body movements such as breathing [CCD06] [DZS08], arm flexion [LST09], and hand movements [SKP08] [AHS03].

We have distinguished two types of trajectory optimization methods: those that rely on the assumption of a specific function (periodic or polynomial functions) as a control trajectory, and those that don’t. The following sections explain each type and provide further insight into how these controllers are used in animation.
where $f(t) = f_{\text{tail}}(t) + f_{\text{muscle}}(t) + f_{\text{joint}}(t)$

Subject to $0 \leq t \leq t_{\text{final}}$

$\phi_{\text{min}}(t) \leq \phi(t) \leq \phi_{\text{max}}(t)$

$\theta_{\text{lim}}(t) = 0$

$\dot{\theta}_{\text{lim}}(t) = 0$

Figure 9: Trajectory-Optimization methods. The optimization process directly computes the muscle control signals according to the minimization of a cost function and targets. These signals can be computed at each time step (static optimization) or in a defined simulation period (dynamic optimization or optimal control). Some approaches include additional components, such as neural networks to generate the desired joint stimuli [HMOA03] [LT06].

4.2.1. Trajectory optimization based on function primitives

One of the simplest motion control strategies consists in synthesizing motions through the generation and tuning of periodic signals. These controllers are mainly used for the generation of oscillatory motions, such as those seen in the locomotion of fishes, worms, snakes and in human chest motions. These signals generally drive spring-like muscles, which are gathered into muscle groups to reduce the number of controlled variables.

Early implementations manually tuned periodic functions to generate the desired motions [TT94] [Mii88] [ZCCD06]. [TT94] controlled artificial fishes by converting a desired swim speed into a spring contraction amplitude and frequency. This mechanism was based on the observation that the speed of most fishes can be proportional to the amplitude and frequency of the tail’s lateral oscillation.

The authors of [Mii88] used controllers that produced sine waves to generate waves of compression, which replicated the elastic deformation present in the locomotion of snakes and worms. Sine waves have also been used jointly with step functions to generate human motions. [ZCCD06] used these functions as varying parameters to synthesize human breathing. The first parameter was a contraction ratio, which was determined according to modeled and measured human muscle contraction ratios during breathing. The second parameter was a binary timing, which was defined by the desired breathing frequency. Finally, polynomial function primitives, such as spline curves, have also served as a means for human motion synthesis. [AHS03] synthesized hand motions in real-time by specifying muscle contraction values at keyframes and interpolating them via spline functions.

Recent implementations have automated the generation of periodic functions by using optimization procedures. An example is the torso controller of [DZS08] for synthesizing human breathing and laughing. In this approach, an optimization attempted to minimize the tracking error between a desired lung pressure, computed from an audio soundtrack, and the current pressure of the model. This process generated the parameters of a set of sine waves that were used directly as muscle activation signals.

More complex motions have also been synthesized with the use of function primitives. An example is the approach of [MWTK13], where control trajectories were encoded as splines, and trajectory optimization and spacetime constraints [WK88] were used for humanoid motion synthesis. The optimization generated joint coordinates, muscle activation signals and lengths, foot contact points, and ground reaction forces. This process was driven by a muscle effort term (metabolic energy expenditure [AP99]), and additional objectives that enforced the equations of motion and encoded other desired muscle behaviors. Although contact points and ground reaction forces, which usually introduce discontinuities into the optimization, were included in the optimization, this approach was still able to achieve impressive computation times. The reason behind this was the use of the Contact Invariant Optimization (CIO) framework [MTP12]. The framework smoothed out the discontinuities in the objective function by allowing foot contact points to gradually invoke ground reaction forces at a distance until a real contact was made.

Other approaches have begun incorporating motor control theories, such as muscle synergies (section 3.4), coupled with static optimization. This is the case of the synergy-based controller for throwing motions developed by [CR-PLD15]. Initial control signals (or synergies) were first ex-

submitted to COMPUTER GRAPHICS Forum (3/2016).
trated from human throwing motions and used to actuate the character. Next, a synergy-based forward dynamics pipeline ensured that the desired throwing motion was reproduced. The pipeline achieved this through two adaptation stages: the first stage consisted in determining the character’s unknown muscle parameters; while the second stage consisted in modifying the time-varying part (or shape) of the synergy via a static optimization, such that the desired throw was reproduced. An interesting feature of this controller, was that muscle redundancy was reduced, and that the initial synergies were able to reproduce general trends in the throwing motion.

4.2.2. Trajectory optimization without function primitives

The majority of trajectory optimization methods discussed in this review do not make assumptions on the trajectory of the control signal, and rely more heavily on biomechanical and motor control concepts, such as the minimization of effort or fatigue. We will first discuss the controllers whose sole purpose is synthesizing rigid body motions; next we will introduce the controllers that also model the effect of this motion on soft tissues; and we will finish by presenting controllers designed for purely soft bodies (no skeleton).

Important contributions have been made, which focus on synthesizing the rigid body motions of musculoskeletal systems. [KSK97] developed an open-loop feedforward controller for the animation of the lower limbs of muscle-based characters. This controller interpolated input postures and computed muscle forces via the inverse dynamics and prediction algorithm introduced by [CB81]. The same authors extended this approach in [KSK00], by converting input postures into feasible ones through an optimization smoothness term and an energy efficiency term that considered muscle power and rate of change of muscular tensions.

Feedback controllers have also been developed with an adaptability to different physiological and environmental conditions. For instance, the authors of [LPKL14] synthesized biped gaits, which were adaptable to conditions such as, muscle weakness, tightness, joint dislocation, external forces, and motion objectives, such as maximization of efficiency and pain reduction. The approach consisted of a muscle optimization and a trajectory optimization. From a given reference motion, the muscle optimization (which minimized muscle effort) found the optimal coordination of muscle activation levels to control the character in a per frame basis. The purpose of the trajectory optimization was to modulate the reference motion and its step locations [KH10] such that: an accurate and robust motion reproduction was achieved, or to adapt the motion to new conditions and requirements. The optimization was based on the minimization of an efficiency term (required) and a muscle force term to simulate a pain avoidance behavior (optional). It is also worth mentioning that this is one of the first approaches in animation to compare the synthesized muscle signals with human muscle data.

Other approaches couple trajectory optimization methods (as muscle-based controllers) and neuronal networks (as joint controllers). An example is the locomotion controller developed by [HMOA03], which employed a neuronal model and an optimization procedure. The neuronal model was composed of a central pattern generator (section 3.4) that computed the required joint stimuli based on sensory information such as the position of the center of gravity and joint displacements. This joint stimuli was later distributed as individual muscle forces through an online static optimization procedure that sought to minimize muscle fatigue [CB81]. Before running this model online, the optimal neural parameters were found through a genetic algorithm and evaluative criteria. This criteria contained a motion smoothness term and an energy efficiency term that considered muscle power and rate of change of muscular tensions.

Similarly, [LT06] synthesized neck motions by using neural networks to generate musculoskeletal stimuli and adjusting this stimuli through a static optimization. The networks generated neck poses and stiffness signals according to desired head orientations. While the optimization generated desired neck muscle strain (deformation) and strain rates that ensured that the head converged to the desired orientations via minimal joint displacements. The outputs of the networks were combined into one feedforward signal, which was converted into muscle activations by a PD controller that was constantly monitoring the error in muscle strain and strain rate. An innovation in their design was that the networks were able to control the pose and stiffness of the neck independently, thanks to their offline training process. For both networks, this offline training process consisted in minimizing a muscle effort term. However for the network controlling stiffness, the muscle forces were constrained to lie in the null space of the moment arm matrix, and therefore they did not contribute to the joint torque or affect the pose. Once the networks were trained offline, they performed their online tasks, faster than attempting to solve the corresponding optimal control problem online.

In addition to rigid body motion, some control meth-
ods also focus on synthesizing the movement of soft bodies (such as tendons and skin) and the interactions of these with rigid bodies. For instance, [TSF05] synthesized hand motions and muscle bulging through a controller which generated muscle activations for a set of desired joint orientations. The authors employed an optimization which minimized joint tracking error and muscle effort. Furthermore, a variety of hand motions were generated by using clinically motivated heuristics for Repetitive Strain Injuries diagnosis.

[SKP08] also synthesized hand motions and the movement of tendons and muscles under the skin, thanks to a novel biomechanical simulator that used target rigid body velocities. The simulator involved an optimization procedure to compute muscle activations and a skin deformation algorithm. The optimization was mainly led by a muscle effort term and task related objectives which ensured a good tracking of the desired rigid body velocities. These activations first produced muscle and tendon motion, and afterwards, rigid body or bone motion. It is worth noting that the transformation from activations to rigid body motion was enhanced by complex muscle routings or strands. These strands were modeled as cubic B-splines and were not constrained to pass through a fixed set of via points. Instead they were allowed to slide along predefined surfaces. Allowing this motion is important, because more realistic changes in muscle length and velocity are achieved, and these changes are known to affect the force generation properties of muscles.

The work of [SSB∗15] is closely related to that of [SKP08]. It consisted in a hierarchical control framework for hands and tendinous systems, which performed tasks such as writing, and could simulate clinical deformities of the hand by altering tendon parameters. At the highest level, kinematic controls were computed to track a fingertip reference trajectory, and at the lowest level, an activation controller transformed these controls into muscle activations, using the formulation of [SKP08]. A difference with the latter approach is the fact that the optimization parameters could be determined through a general-purpose learning-based approach requiring no previous system knowledge. Furthermore, the framework has an increased robustness when handling the constraints between tendons and bones, thanks to the application of the constrained strand framework of [SILP11]. Finally, another novelty is the ability to deal with highly stiff strands using larger time steps, thanks to the assumption that strain and stress propagate instantaneously through the strand.

Hand motion has not been the only area of development. The authors of [Lee08] [LST09], presented one of the most detailed biomechanical models of the human upper body and a controller to track a set of poses while achieving a desired level of muscle co-activation. The activations that satisfied these requirements were computed thanks to a static optimization procedure that minimized muscle effort, and was constrained by joint torques computed from input poses [Fea14]. The optimization was solved twice (once for each muscle in an antagonistic pair), with the difference that the antagonist activation was constrained by a torque of opposite sign. Finally, once the agonist and antagonist activations were defined, they were modified proportionally to the target co-activation. Secondary skin motion was also synthesized through the simulation technique of [Sift07]. This technique consisted in defining two meshes, a coarse one that simulated the elastic flesh deformation caused by muscle fibers, and a detailed one for rendering and collision handling.

Recently, even more complex and dynamic motions have been synthesized via trajectory optimization methods. As seen in the previous section, the authors of [Si13], synthesized human swimming through two types of controller on a very detailed musculoskeletal system. The first for periodic motions, and the second one for non-periodic motions. For the non-periodic motions, such as controlling body orientation, a static optimization was implemented for the purpose of tracking references poses, and imposing motion naturalness through Gaussian Process Dynamical Models (GDPM) and human motion data.

The locomotion of purely soft bodied characters has also been an area of study. One of the first examples is the controller created by [GT95] for the locomotion of highly flexible animals. The control was done at multiple levels of abstraction, which granted the characters with the ability to synthesize basic locomotion skills, remember the learned skills, and combine them efficiently to perform more complex tasks. The basic locomotion skills were learned through an optimization procedure containing task achievement objectives, such as the shortest distance to a goal, and a muscle effort objective. Next, the learned signals were converted into more compact representations and used within a second optimization procedure with the objective of finding a proper combination of skills to achieve more complex behaviors. Once the compact representations were created, the method could be made to work in real-time.

The computational efficiency of this controller was later improved in [GTH98]. The authors devised a simulation method that, unlike most optimal control methods, allowed the computation of gradients and fast gradient based optimization controller synthesis. The reason behind this was that neural networks were used instead of a physics simulator at each time step. This created a cascade network structure that allowed the use of a “backpropagation through time”, which adjusted the control signals using the chain rule of differentiation within each network.

More recently, the authors of [TTL12] also developed a trajectory-optimization based controller for soft bodied characters. This approach sought to control the shape of soft bodied characters (and therefore the shape of their muscles) to achieve specific locomotion tasks. The procedure
used an optimization at each time step to determine muscle 
lengths, and a contact planner, which predicted how the de-
sired changes in muscle contraction affected contact points, 
allowing the characters to slide and break contacts with the 
ground.

4.2.3. Summary

Trajectory optimization methods are until now the most 
common solution for muscle-based control. One of their ad-

cvantages is that they are more easily adaptable to different 
character morphologies. The reason for this is their central-
ized nature. Local controllers, which can be subject depen-
dent (due to the fact that they are usually assigned to spe-
cific muscles or body sections), are not used. Examples of 
this flexibility are the works of [GT95], who implemented a 
controller for a variety of virtual fishes, and [TTL12], who 
used a variety of alphabet letters with diverse muscle ar-
rangements.

One of the main drawbacks of trajectory optimization 
methods is their implementation. A considerable amount of 
knowledge is needed in optimization techniques and con-
strained dynamics. Another drawback is the fact that mod-
ing biomechanical mechanisms (such as muscle reflexes), 
which would certainly aid in synthesizing motions, is not 
straightforward. The effect of such mechanisms could be en-
coded within a cost function, but implementing a model of 
the mechanisms (as is done by controller optimization ap-
proaches) is a more natural and efficient alternative.

The computational requirements and efficiency of these 
methods depend on many factors, such as the number of 
objectives and complexity of the character [GP12]. Com-
putational efficiency still needs to be improved since the 
controllers have not reached real-time performance. How-
ever, some controllers have demonstrated impressive results, 
given the redundancy involved in the control problem. For 
example, [GTH98] and [MWTK13] achieved convergence 
times comprising a few seconds or minutes, due to the use 
of fast gradient based optimization methods. We believe that 
computation times could be further improved by considering 
muscle groups and muscle synergies, which would reduce 
the number of control parameters.

Finally, as with controller optimization frameworks, sev-
eral of the frameworks in this section were conceived for 
tasks, such as walking and jumping, which imply the addi-
tional and crucial challenge of balancing. Different strategies 
for balance were used. For instance, the authors of [TTL12], 
were inspired by the work of [MZS09], and regulated linear, 
angular momentum, and the area of support of their soft-
bodied characters to maintain balance. On the other hand, 
approaches such as [KSK00] used a stability function based 

on the ZMP (zero moment point) [Vuk90]. Another inter-

esting balance strategy used by [LPKL14] was based on 
the work of [KHK00] which consists in planning a balance-
recording reference motion instantaneously based on the 
estimated pendulum state.

5. Conclusions and Future Directions

Muscle-based animation is gradually enhancing character 
realism by introducing important biological factors through 
the use of muscle models and muscle-based controllers. 
However, the main challenge remains how to model these 
elements, while ensuring a set of desirable characteristics 
such as motion realism and an efficient computation time).

As seen in section 2 the beneficial effects of muscles, en-
compase: better stability properties, more realistic passive 
dynamics, better estimates of energy cost or fatigue, effi-
cient control via motion mechanics and an ease in simul-
ation of musculoskeletal defects, pathologies and fatigue. All 
of these benefits have been evidenced by comparative studies 
and theories from the fields of biomechanics and motor con-
trol. Moreover, the presence of viscoelastic and non-linear 
actuators (figure 4) supply the body with a limited amount 
of power. This viscoelasticity also tends to smooth the ac-
tion of the muscle on the joints, and the non-linearity tends 
to reduce drastically the efficiency of the muscles when they 
are contracted in extreme positions. Another benefit of the 
use of muscles is that their limited force generation abili-
ties tend to frame realistically the capabilities of the body in 
terms of motion. Finally, muscle routing better represents 
the action of the muscles on the joint than a servo, and generates 
a non-linear relationship between forces and torques.

We have presented a classification and description of the 
main techniques for muscle-based control in the anima-
tion field. This classification encompasses two general cate-
gories: controller optimization methods and trajectory opti-
mization methods. A synopsis of this classification and key 
characteristics of each control approach are featured in Ta-
ble 1. Moreover, in Table 2, we present the muscle-based 
terms used in each controller’s cost function, for detailed ex-
planations on non-muscle-based terms we invite the reader 
to consult the list of references.

Each control technique contributes differently to the ani-
mation field, at the expense of certain trade-offs. The main 
advantage of controller optimization methods is that they al-
low an easier implementation of well known biomechanical 
mechanisms and motor control theories through simple con-
trol laws that generate the required joint and muscle behav-
ior at different stages during the motion. Their main draw-
back is computational efficiency (section 4.1). As stated in 
section 4.2, two of the main advantages of trajectory opti-
mization methods are a higher flexibility to adapt to different 
character morphologies, and lower computation times with 
regard to controller optimization methods although most 
methods have not reached real-time capabilities yet. Among 
the main drawbacks are a higher difficulty for implementa-
tion and for encoding simple biomechanical mechanisms.
Muscle-based controllers are meant to impact and enhance the physics-based animation field. Any physics-based animation system that is currently using joint torques (or virtual forces etc.) has the potential to be extended to a muscle actuated system, leading to many possibilities. Multi-scale models and simulations will also benefit from the recent advances in musculoskeletal animation where entities from different scales can cohabit and interact within a common environment.

Acknowledgments

We would like to thank the reviewers, for their insightful comments and suggestions, which helped enrich this work. This study was funded by the French National Research Agency (ANR) project ENTRACTE (Grant agreement: ANR 13-CORD-002-01).

Appendix 1: Synopsis of muscle-based controllers for animation

A synopsis of the controller classification and key characteristics of each control approach are featured in Table 1. The controllers were classified according to control type, following the classification made in section 4. An additional “hybrid method” category was added, since the approach by [Si13] used both a controller optimization and a trajectory optimization.

The control methods are also classified by their control space, or the space in which their output signal is generated. This category is an indicator of the degree of detail with which the muscles are modeled. Motor space control methods generate excitations or activations to control characters with detailed muscle representations. Force space control methods generate forces to control characters with less detailed muscle representations (such as spring muscle models). Hybrid space control methods generate both motor signals and servo signals (torques) for characters that are actuated by muscles and servos.

The table also features the main tasks for which the controllers were employed. These can be: locomotion (L), kicking (K), jumping (J), balancing (B), gestures (G), posture adjustment (P), torso motion (T), neck motion (N), arm motion (A), hand motion (H), target tracking (Tr), and character interaction (CI).

The models commanded by these controllers are described by their degrees of freedom (DoF), number of muscles (Ms) and muscle groups (MGs). The muscle actuators are classified by the force generation model employed as either spring elements, Hill-type models or other. In parenthesis we also state the references on which the muscle models were based or extracted.

The most important characteristic is the control strategy,
which indicates the muscle control method as seen in section 4.1 and section 4.2. Next, we find the Main User Input. The objective of this characteristic is to denote what are the main user input commands to the control frameworks. For instance, the required user input can be compact goal (such as a desired walking speed or direction), it can also be more detailed motion data (recorded or specified by the animator), or sometimes less intuitive variables such as controller parameters directly.

Finally, the cost function column contains the objective functions used, if the strategy involved an optimization procedure. In the case that no cost function was used the field reads NA (not applicable).

Appendix 2: Cost functions used by muscle-based controllers for animation

Table 2 features a list of neuromuscular objective terms, which correspond to the categorization introduced in section 3.4, inspired by [ZW90]. We recall that this categorization is based on a generalized performance criterion which includes three components: task specific (tracking a giving trajectory, minimizing jerk), neuromuscular (minimizing muscle stress, neural effort) and bone joint (minimizing contact forces, avoiding certain ranges of motion) objectives. Task specific kinematics and bone joint objectives, are already popular among servo-based controllers. Therefore, we focus on the neuromuscular objectives, the novelty of muscle-based controllers.

Task specific kinematics and bone joint objectives

The task objectives mainly consist in reference tracking (pose, velocity, acceleration, moment and distance tracking) and stability control (to ensure balance and avoid sliding). Other objectives include those that attempt to motivate motion naturalness, by encouraged motion fluidity, smoothness and preferences for motions closer to the training data [Si13]. Finally, the bone joint objectives usually enforce joint limits and avoiding self collision.

Neuromuscular objectives

These objectives were divided into those that describe muscle fatigue and those that describe muscle effort (following the distinction made by [AvdB10]). Thus, cost functions presenting muscle volume weighting and lower exponents were classified as effort-like, while cost functions without muscle volume weighting and higher exponents were classified as fatigue-like. An additional category was added for the functions that did not fit these two categories. The final classification and details on these functions is featured in Table 2.

Fatigue-like functions Muscle fatigue occurs when there is a failure to maintain a required or expected force [Edw08], and it is related to the amount of synergy between the muscles. A high synergy implies that all muscles contribute during the motion in a way that the maximum relative load of any muscle remains as small as possible. In other words they work well together by helping each other and ensuring that no muscle works more than the rest. The authors of [KSK00] incorporated this function into an optimization to synthesize lower body motion. Another example is found [HMOA03], were the muscle fatigue term of [CB81] was used to synthesize locomotion.

Effort-like functions Muscle effort is a substitute for muscle energy expenditure. It is related to the volume of activated muscle tissue [AvdB10]. These functions have been more popular than fatigue-like functions, among animators. In [GT95] and [GTH98] a muscle effort objective was used, which penalized high amplitude and rapid variation in the actuation signals. [TSF05] [SKP08] [LT06] and [LST09] expressed effort through operations such as, weighting, normalizing and adding up muscle activations or forces. More recent approaches [WHDK12] [GvdPvdS13] [MWTK13] have preferred using the metabolic energy expenditure cost functions as described by [AP99]: the sum of heat released and mechanical work done by the muscles. Others [LPKL14] have used instead a measurement of energy consumption per unit moving distance [AP01].

Alternative muscle-based terms Some control methods have had as a basis muscle-based terms which do not fall into the two previous categories (effort and fatigue). An example is the approach of [MWTK13] who also ensured consistency between muscle forces and physiology. Another example is the controller of [HMOA03], who used an energy efficiency and motion smoothness term that took into account the rate of change of muscular tensions.

Appendix 3: Physics simulators

The physics simulator is one of the most important tools in muscle-based control frameworks. One of its main objectives is to perform forward dynamics calculations, such that the character’s state can be retrieved and updated once the controller is in action. Controller optimization frameworks such as [GvdPvdS13] [WHDK12] and [NF02], use simulators such as ODE (Open Dynamics Engine) [Sm06] and DANCE (Dynamic animation and control environment) [SFNTH05]. Trajectory optimization frameworks, such as [ZCCD06] also used ODE, while some [GTH98] [KSK97] opted for SD/FAST [HRS91], others [MWTK13] for MuJoCo [TET12] and others [CRPLD15] for MATLAB SimMechanics. Finally, some approaches have opted for home-made simulators, such as [Si13], who developed their own multiphysics simulation framework for human swimming.

submitted to COMPUTER GRAPHICS Forum (3/2016).
Table 1: Main muscle-based controllers for animation

<table>
<thead>
<tr>
<th>Control Type</th>
<th>Controller Reference</th>
<th>Control Space</th>
<th>Character</th>
<th>Model</th>
<th>Actuation Type</th>
<th>Control Strategy</th>
<th>Main User Input</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller Optimization Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force Space</td>
<td>[NF02]</td>
<td>[B,G,P]</td>
<td>Human, Human Torso</td>
<td>Human 47 DoFs 94 Ms</td>
<td>Springs</td>
<td>Antagonistic control</td>
<td>Joint positions</td>
<td>NA</td>
</tr>
<tr>
<td>Motor Space</td>
<td>[Gvd-PvdS13]</td>
<td>B.L.</td>
<td>Humanoids, Imaginary creatures</td>
<td>40 DoFs 27 Ms</td>
<td>Hill-type ([Zaj89])</td>
<td>PD controllers</td>
<td>Muscle reflexes</td>
<td>Constant excitations</td>
</tr>
<tr>
<td>Hybrid Space</td>
<td>[WHDK12]</td>
<td>B.L.</td>
<td>Humanoid</td>
<td>30 DoFs 16 Ms (legs)</td>
<td>Hill-type ([Zaj89]) (Lower-body)</td>
<td>PD controllers</td>
<td>Muscle reflexes</td>
<td>Constant excitations</td>
</tr>
<tr>
<td>Trajectory Optimization Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force Space</td>
<td>[Mi88]</td>
<td>L</td>
<td>Snakes, Worms</td>
<td>–</td>
<td>Springs</td>
<td>Function primitives</td>
<td>Function primitive parameters</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>[TT94]</td>
<td>CLL</td>
<td>Fishes</td>
<td>91 Ms</td>
<td>Springs</td>
<td>Function primitives</td>
<td>Pre-defined habits</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>[TTL12]</td>
<td>B.L.</td>
<td>Alphabet Letters</td>
<td>Ex. Letter I 104 DoFs 4 Ms</td>
<td>Springs</td>
<td>No function primitives</td>
<td>Task positions</td>
<td>Task velocities</td>
</tr>
<tr>
<td>Motor Space</td>
<td>[GT95]</td>
<td>L,Tr</td>
<td>Snakes, Marine animals</td>
<td>Snake 126 DoFs 40 Ms</td>
<td>Springs</td>
<td>No function primitives</td>
<td>Locomotion speed</td>
<td>Distance to goal</td>
</tr>
<tr>
<td></td>
<td>[GTH98]</td>
<td>L</td>
<td>Dolphin</td>
<td>12 Ms</td>
<td>Springs</td>
<td>No function primitives</td>
<td>Locomotion speed</td>
<td>Distance to goal</td>
</tr>
<tr>
<td></td>
<td>[KSK00]</td>
<td>B,K,L</td>
<td>Human</td>
<td>86 Ms (legs)</td>
<td>Hill-type ([DLH∗90]) ([Del90])</td>
<td>No function primitives</td>
<td>Joint positions</td>
<td>Stability</td>
</tr>
<tr>
<td></td>
<td>[AHS03]</td>
<td>H</td>
<td>Human hand &amp; Forearm</td>
<td>–</td>
<td>Other ([BH93])</td>
<td>Function primitives</td>
<td>Contraction values at keyframes</td>
<td>NA</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Control Type</th>
<th>Control Space</th>
<th>Controller Reference</th>
<th>Task</th>
<th>Character</th>
<th>Model</th>
<th>Actuation Type</th>
<th>Control Strategy</th>
<th>Main User Input</th>
<th>Cost Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[HMOA03]</td>
<td>B.L.</td>
<td>Human</td>
<td>19 DoFs 60 Ms</td>
<td>Other ([Hat77])</td>
<td>No function primitives</td>
<td>Locomotion speed Ratio of period of foot–ground contact</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[TSF05]</td>
<td>H</td>
<td>Human hand &amp; forearm</td>
<td>41 Ms 16 DoFs</td>
<td>Hill-type ([Zaj89])</td>
<td>No function primitives</td>
<td>Joint positions Joint velocities Muscle activations</td>
<td>Pose tracking Muscle effort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ZCCD06]</td>
<td>T</td>
<td>Human Torso</td>
<td>5 MGs</td>
<td>Hill-type ([Zaj89])</td>
<td>Function primitives</td>
<td>Function primitive parameters</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[LT06]</td>
<td>CLN</td>
<td>Human neck &amp; head</td>
<td>72 Ms</td>
<td>Linearized Hill-Type ([NTH01], [WC00])</td>
<td>No function primitives</td>
<td>Initial and final head positions</td>
<td>Offline: Muscle effort Online: Minimal joint displacement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[SKP08]</td>
<td>A.H</td>
<td>Human hand &amp; forearm</td>
<td>54 Ms</td>
<td>Hill-Type No FV-relationship ([SKP08])</td>
<td>No function primitives</td>
<td>Task velocities</td>
<td>Velocity tracking Muscle effort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[LST09]</td>
<td>A.T</td>
<td>Human upper body</td>
<td>147 DoFs 814 Ms</td>
<td>Linearized Hill-type ([LT06])</td>
<td>No function primitives</td>
<td>Joint positions Target coactivation</td>
<td>Muscle effort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[LPKL14]</td>
<td>B.L.</td>
<td>Human</td>
<td>25.39 DoFs 62-120 Ms</td>
<td>Hill-type ([Zaj89])</td>
<td>No function primitives</td>
<td>Joint positions Joint velocities Task positions</td>
<td>Pose tracking Velocity tracking Acceleration tracking Stability Muscle effort Energy efficiency Pain avoidance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[CR-PLD15]</td>
<td>A.H</td>
<td>Human arm &amp; hand</td>
<td>3 DoFs 6 Ms</td>
<td>Hill-Type ([Hil38], [RAPC10])</td>
<td>Function primitives</td>
<td>Joint positions</td>
<td>Pose tracking</td>
</tr>
<tr>
<td>Control Type</td>
<td>Control Space</td>
<td>Controller Reference</td>
<td>Task</td>
<td>Character</td>
<td>Model</td>
<td>Actuation Type</td>
<td>Control Strategy</td>
<td>Main User Input</td>
<td>Cost Function</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>------</td>
<td>-----------</td>
<td>-------</td>
<td>----------------</td>
<td>------------------</td>
<td>----------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| Hybrid Space | [SSB\textsuperscript{*15}] H | Human hand | – | Piece-wise linear muscle model (SSB\textsuperscript{*15}) | No function primitives | Task positions | Velocity tracking
| Motor Space  | [Si13] L | Human | 163 DoFs 823 Ms | Linearized Hill-Type ([LST09]) | Locomotion tasks: Controller optimization: Neural networks PD controllers Non-locomotion tasks: Trajectory optimization: No function primitives | Locomotion tasks: Joint positions High level CPG parameters Non-locomotion tasks: Trunk orientation Motion capture data | Locomotion tasks: Time-varying muscle length tracking Non-locomotion tasks: Pose tracking Motion naturalness Self collision avoidance | Locomotion tasks: Jumping height Kicking foot speed | Pose tracking Velocity tracking Virtual force reduction Dynamical consistency Muscle effort Muscle force Physiological consistency Self-collision avoidance Joint limit accordance | Locomotion tasks: Locomotion tasks: Pose tracking Motion naturalness Self collision avoidance |

Hybrid Method
(Using: Controller and Trajectory optimization)
Table 2: Neuromuscular objectives used in animation

<table>
<thead>
<tr>
<th>Type</th>
<th>Controller</th>
<th>Cost Function</th>
<th>Formula</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort-like</td>
<td>[WHDK12]</td>
<td>Muscle-effort</td>
<td>$J_M = 3J_A + W J_A + W J_L$</td>
<td>$J_M, J_A, J_L =$ Average rate of metabolic energy expenditure, sum of torques squared, sum of squared soft joint limit torques $A, M, S, W =$ Muscle activation, muscle maintenance, muscle shortening heat rates, positive mechanical work rate $w =$ Scalar weights</td>
</tr>
<tr>
<td>[GT95] [CTH98]</td>
<td>Muscle effort</td>
<td>$J_M = \frac{1}{2} \left( w_a</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>[TSF05]</td>
<td>Muscle effort</td>
<td>$\frac{1}{2}</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>[LTO06]</td>
<td>Muscle effort</td>
<td>$</td>
<td></td>
<td>W^{-1}f_i</td>
</tr>
<tr>
<td>[SKP08] [SSB*15]</td>
<td>Muscle effort</td>
<td>$w_a</td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>[LST09]</td>
<td>Muscle effort</td>
<td>$J_M = 3J_{upper}$</td>
<td>$J_{upper} =$ Upper-body effort, $f_{motor} =$ Active motor torques $W_a =$ Diagonal weight matrix, $q =$ Articular positions</td>
<td></td>
</tr>
<tr>
<td>[MWTK13]</td>
<td>Muscle effort</td>
<td>$J_M = 3J_{upper}$</td>
<td>$J_{upper} =$ Upper-body effort, $f_{motor} =$ Active motor torques $W_a =$ Diagonal weight matrix, $q =$ Articular positions</td>
<td></td>
</tr>
<tr>
<td>[LPKL14]</td>
<td>Muscle effort</td>
<td>$\frac{1}{2} \sum (w_i a_i)^2$</td>
<td>$a =$ Muscle activation</td>
<td></td>
</tr>
<tr>
<td>[SKS00] [HMOA03]</td>
<td>Muscle fatigue</td>
<td>$L_{i} = 3i$$ \sum \left( \frac{f_i}{f_{max}} \right) dt$</td>
<td>$i =$ ith-muscle, $f_i =$ Muscle force $f_{max} =$ Maximum muscle force, $i, f_i =$ Initial and final time</td>
<td></td>
</tr>
<tr>
<td>Alternative</td>
<td>[HMOA03]</td>
<td>Energy efficiency and motion smoothness</td>
<td>$1 \sum_{S=max,D}$</td>
<td>$S =$ Specific power $D =$ Rate of change of muscular tensions</td>
</tr>
<tr>
<td>muscle-based</td>
<td>[MWTK13]</td>
<td>Muscle force physiological consistency</td>
<td>$\sum (F_{SE} - F_{CE} - F_{PE} - mCE \dot{l}_{CE})$</td>
<td>$F_{SE}, F_{CE}, F_{PE} =$ Serial-elastic, contractile, passive forces $mCE =$ Pointmass between contractile element and tendon $l_{CE} =$ Length contractile element</td>
</tr>
<tr>
<td>objectives</td>
<td>[LPKL14]</td>
<td>Pain avoidance</td>
<td>$\sum \left( f_i \right)$</td>
<td>$f_i =$ ith-muscle force</td>
</tr>
</tbody>
</table>

Note: $J_A =$ Average rate of metabolic energy expenditure, $J_L =$ Soft joint limit energy expenditure, $J_{upper} =$ Upper-body effort, $f_{motor} =$ Active motor torques, $W_a =$ Diagonal weight matrix, $q =$ Articular positions, $w =$ Scalar weights, $a =$ Muscle activation, $w_a, w_d =$ Regularization and damping scalar weights, $w_i =$ Variable weight (regularizes muscle activation levels), $f_i =$ Muscle forces, $f_{max} =$ Maximum muscle force, $i, f_i =$ Initial and final time, $S =$ Specific power, $D =$ Rate of change of muscular tensions, $F_{SE}, F_{CE}, F_{PE} =$ Serial-elastic, contractile, passive forces, $mCE =$ Pointmass between contractile element and tendon, $l_{CE} =$ Length contractile element.


[PB02] Pullen K., Bregler C.: Motion capture assisted animation: Texturing and synthesis. ACM Trans. Graph. 21, 3 (July 2002), 501–508. 2


[PH06] Park S. I., Hodgins J. K.: Capturing and animating skin deformation in human motion. ACM Trans. Graph. 25, 3 (July 2006), 881–889. 6


submitted to COMPUTER GRAPHICS Forum (3/2016).
ward control of a musculoskeletal system. Avoidance.

Realistic biomechanical simulation and control of human swim-
ting and underwater movement. In Proceedings of the 21st An-
ual Conference on Computer Graphics and Interactive Tech-

Dynamic animation and control environment. In Proceedings of Graphics Interface 2005 (Victoria, British Columbia, 2005), GI ’05, Canadian Human-Computer Communications Society, pp. 61–70. 18


Sm06 SMITH R.: Open dynamics engine v0. 5 user guide. 18


Str96 STROEVE S.: Learning combined feedback and feedforward control of a musculoskeletal system. Biological Cybernetics 75, 1 (1996), 73–83. 5


submitted to COMPUTER GRAPHICS Forum (3/2016)
Submitted to COMPUTER GRAPHICS Forum (3/2016).