Generating Upper-Body Motion for Real-Time Characters Making their Way through Dynamic Environments
Eduardo Alvarado, Damien Rohmer, Marie-Paule Cani

To cite this version:

HAL Id: hal-03757439
https://hal-ip-paris.archives-ouvertes.fr/hal-03757439
Submitted on 22 Aug 2022

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.
Generating Upper-Body Motion for Real-Time Characters Making their Way through Dynamic Environments

Eduardo Alvarado¹, Damien Rohmer¹, Marie-Paule Cani¹

¹ LIX, École Polytechnique/CNRS, Institut Polytechnique de Paris, Palaiseau, France

Abstract
Real-time character animation in dynamic environments requires the generation of plausible upper-body movements regardless of the nature of the environment, including non-rigid obstacles such as vegetation. We propose a flexible model for upper-body interactions, based on the anticipation of the character’s surroundings, and on antagonistic controllers to adapt the amount of muscular stiffness and response time to better deal with obstacles. Our solution relies on a hybrid method for character animation that couples a keyframe sequence with kinematic constraints and lightweight physics. The dynamic response of the character’s upper-limbs leverages antagonistic controllers, allowing us to tune tension/relaxation in the upper-body without diverging from the reference keyframe motion. A new sight model, controlled by procedural rules, enables high-level authoring of the way the character generates interactions by adapting its stiffness and reaction time. As results show, our real-time method offers precise and explicit control over the character’s behavior and style, while seamlessly adapting to new situations. Our model is therefore well suited for gaming applications.

CCS Concepts
• Computing methodologies → Animation;

1. Introduction
Being able to animate virtual characters navigating through complex, dynamic environments is of utmost importance for video games and virtual reality applications. Natural scenes that typically include tall plants, bushes, and trees of various types and sizes are particularly challenging.

In real life, we, humans, constantly anticipate and adapt our upper-body motions when making our way through complex environments. Roughly evaluating the expected stiffness and dynamics of obstacles helps us anticipate interactions, and in particular tune our muscular stiffness and reaction speed in order to efficiently push them out of the way or avoid their trajectories. For example, we stay relaxed making space around us in the middle of tall grass, but we tense our muscles to fend off a stiffer branch of a tree or quickly protect our head when a potentially dangerous object comes toward us. Moreover, the way we interact with obstacles...
Our technical contributions are threefold: driven by the input kinematic motion clip. Our model generates plausible action for a character, whatever its affecting the position and angular objectives of its limbs. Ultimately, control tension/relaxation during a character’s motion without a- interaction process relies on a dedicated reformulation of Neff and tions, through simple real-time query about their surroundings. The actors to react to any upcoming obstacle, and therefore to plan ac- tion and amount of muscular stiffness. Our solution allows char- tive that drives the character’s gestures with suitable synchroniza- tion mechanism is used to guide an Inverse Kinematics (IK) objec- bility as well as the predicted stiffness of obstacles. This anticipa- tions based on high-level rules that take into account ray-cast visi- upper-body animation. The character is able to anticipate interac- tions with a lightweight physics-inspired model for each upper- body limb separately. It makes the upper-body of kinematic-controlled characters able to interact through plausible dynamic responses (Sec. 3).

In this work, we propose a real-time animation model for the upper-body of human-like virtual characters; it provides high-level behavioral control in order to interact plausibly with dynamic ob- stacles. Composed of an anticipation mechanism based on sight, followed by a rule-based action module, our model makes the character responsive to its surroundings by allowing it to push away dynamic obstacles made of hierarchical articulated rigid bodies as- sociated to visual skinned-rigs, and make its way through the ob- served environment. To achieve this, we introduce a hybrid, layered character model that couples the input keyframe animation with a lightweight physics-inspired model, used to dynamically adapt the upper-body animation. The character is able to anticipate interactions based on high-level rules that take into account ray-cast visibil- ity as well as the predicted stiffness of obstacles. This anticipa- tion mechanism is used to guide an Inverse Kinematics (IK) objec- tive that drives the character’s gestures with suitable synchroniza- tion and amount of muscular stiffness. Our solution allows char- acters to react to any upcoming obstacle, and therefore to plan ac- tions, through simple real-time query about their surroundings. The interaction process relies on a dedicated reformulation of Neff and Fiume’s antagonist control method [NF02], which enables us to control tension/relaxation during a character’s motion without af- fecting the position and angular objectives of its limbs. Ultimately, our model generates plausible action for a character, whatever its current state and the nature of obstacles, while remaining loosely driven by the input kinematic motion clip.

Our technical contributions are threefold:

- A real-time, hybrid character model coupling keyframe anima- tions with a lightweight physics-inspired model for each upper- body limb separately. It makes the upper-body of kinematic-controlled characters able to interact through plausible dynamic responses (Sec. 3).
- An extension of antagonist controllers [NF02] allowing the intu- itive tuning of gestures and interaction styles by dissociating the position and orientation of the limbs from their degree of mus- cular rigidity (Sec. 4).
- An efficient, yet generic and customizable anticipation mecha- nism that enables our model to couple high-level procedural rules with metadata from observed obstacles. By driving the kinematic controllers and anticipating the amount of tension or reaction time required, this mechanism generates character ani- mations that adapt to the surroundings (Sec. 5).

Fig. 1 illustrates the application of our method to a real-time char- actor making its way through a variety of dynamic environments, including deformable obstacles.

2. Related Work

Generating the reactive motion for an agent in a dynamic envi- ronment is an active research topic. It was addressed in differ- ent fields, from Computer Graphics (CG) to robotics and bio- mechanics (e.g., [SP17]). For the sake of conciseness, we focus here on CG research tackling the reciprocal influence between char- acter motion and animation of its virtual surroundings.

2.1. Controlling Physically-Based Characters

Since the early years of CG animation, physically-based models have been explored to represent reactive characters in dynamic environments [RH91, HBW095]. Ragdoll models represent each limb as a constrained articulated rigid-body with prescribed mass and inertia. The character’s motion can be handled by applying a set of actuator forces and torques to the rigid bodies coupled with a numerical time integrator. However, computing coherent forces over time to achieve a given motion is a complex problem. A popular approach relies on the use of high-level controllers for joint actuation, providing a trade-off between motion plausibility and the complexity of user-control [GP12]. Proportional-Derivative (PD) controllers are in particular considered as common ground for character-animation methods [YLs07, WFH10]. However, despite their simplicity, instability and stiffness control are still recurrent problems when reproducing highly-dynamic and accurate anima- tions.

Stable Proportional-Derivative (SPD) controllers [TLT11, YY20] introduce the idea of incorporating the next simulation step into the computation of forces, thus improving numerical stability and performance. Simplified physical models [KH10, KHH17], stochastic optimal control [LYV10], or Model Predictive Control (MPC) [TET12, TMT14, EHSN19] have also been successfully used to animate virtual agents with PD controllers at their joints. In terms of motor parameterization, Abe et al. [ALP04] add momentum constraints to generate more plausible physical moves- ments. Proportional controllers mix stiffness (which models the appearance of tension/relaxation in the character) and the joint orientation at the equilibrium state. This dependence may ham- per the intuitiveness and precision that a user may want to ad- dress when parameterizing its character behavior. Neff and Fi- urne [NF02] introduce an antagonistic-based PD formulation, al- lowing them to decouple stiffness and equilibrium-state orientation.
In our work, we leverage and extend their antagonistic formulation for arbitrary 3D limb motions, and demonstrate that it can be used to dynamically edit keyframe animations in an online fashion and to decouple stiffness and position controls. To the best of our knowledge, joint-based stiffness control has shown benefits in robotics but has not yet been fully explored for character animation.

The use of Reinforcement Learning (RL) has become prevalent in recent years for optimizing physically-based controller parameters. These approaches can preserve character balance to achieve realistic locomotion up to complex acrobatic gestures. However, it is still a difficult task to define an accurate, yet general reward system that performs well in selecting the best action under a multitude of scenarios. Thanks to the democratization and availability of motion capture data, Deep Learning (DL) based motion synthesis has been very successful in reproducing lifelike character motions for prescribed scenarios. Most particularly, DL has been combined with RL to handle real-time dynamic simulated behaviors while preserving the naturalness of the training data. The effectiveness of such combination was demonstrated for characters maintaining their balance in new environments with moving solid obstacles. However, learning-based approaches still suffer from limitations that make them hard to use for efficient game-like setups in natural environments. First, precise authoring of learned behaviors is highly indirect and hard to predict; yet, this aspect is of utmost importance for game design. Second, natural environments are characterized by diverse deformable elements such as various types of vegetation or uneven terrains. Each natural element may be associated with different physical properties and thus, could trigger specific character behaviors. Pre-training all possible deformations and behaviors would be, at best, very challenging. This is also orthogonal to the process of current game development, where tuning of flexible behaviors and insertion of new environment assets have to be as lightweight as possible.

2.2. Hybrid Character Models using Kinematics

Kinematic-driven virtual characters are extremely efficient to compute and allow intuitive formulations for constraints and objectives. Hybrid models combine physical or data-driven representations with kinematics to offer a trade-off between automatic motion quality, automatic pose adaptation, and user-control. The use of space-time constraints is a common way to describe kinematics objectives while preserving character dynamics, but it involves an optimization procedure that is not applicable at run time. The use of Inverse Kinematics (IK) is an intuitive representation to specify end-effector objectives in a kinematic chain. It was used, for instance, in combination with physical constraints and hierarchical motion curve editing. It was also combined with short-term dynamical effects. Extending motion synthesis to the morphology of an arbitrary character was proposed in integrating procedural techniques with a gait generator in dynamic environments for both quadruped and multi-legged characters. A lightweight physically-based model reduced to a single inverse pendulum was combined with keyframe animations to generate a responsive, real-time character for augmented reality applications.

Similar to our approach, some work proposed local hybrid models addressing the motion of some parts of the upper-body, such as the arms. Jordan and Hodgins added contact constraints to locally modify motion capture data, and extended it further to integrate dynamic responses. Arm motion was also studied using learning-based approaches in sport applications, and used to infer lower-body motions in VR. In our work, we introduce a framework that helps the user define upper-body animations by locally combining keyframes and light physical control.

2.3. Controlling Characters in Natural Environments

Natural scenes are characterized by a rich set of dynamic and deformable elements, which might affect how the character actively behaves. Although visuomotor systems have been used to adapt the character based on external observations, generating plausible, yet general motions that remain controllable for virtual characters interacting with such environments – in real-time – is challenging, due to the computational cost of a full-scale physical simulation of both, characters and deformable materials that might constitute the ground or vegetation. Layered models embedding simplified representations of the environment’s response have been successfully used for real-time applications. For instance, a simplified fluid representation was used for swimming characters, or for windy environments. A simplified friction model was used to represent human locomotion efficiently on semi-flooded grounds or through dense vegetation represented using billboards. An essential aspect in plausible natural environments is the bi-directional interaction between the character and the scene. The character should not only adapt to its surroundings, but the environment itself should also be deformed by the character’s actions. In the resulting two-way interaction, the character’s behavior should dynamically adapt to the ongoing deformation. Such coupled interactions were proposed for lower-body motions on soft natural grounds, such as mud or snow. However, to the best of our knowledge, no work has yet tackled the real-time action-reaction of the character’s upper-body in interaction with a natural environment.

3. Hybrid Character Model for Upper-Body Interactions

We introduce a hybrid model that couples keyframe animation and lightweight physical simulation for human-like characters. Our novel system aims to be flexible enough to coexist with current gaming animation pipelines, such as in Unity 3D, by bringing together hand-crafted animations or motion-capture data, with on-the-fly actions and reactions to new situations. The hybrid model is able to switch between physical and kinematic spaces independently for the different body parts. This allows the game designer to improve the interactive motion from the predefined kinematic controllers, based on the effects that the dynamic, deforming environment might have upon specific body parts, as well as on the actions that the character physically exerts on its surroundings.

We define the character as a human (bipedal) model represented as a responsive, real-time character for augmented reality applications.
by a skin mesh, rigged to an articulated animation skeleton. This model can be associated to a set of predefined animations such as walking or running, either manually defined by keyframes or, alternatively, from motion-capture data. In this work, we call input skeleton the animated skeleton that follows these predefined movements. It will be used as a soft constraint to guide our hybrid model. In parallel, we associate a physically-based ragdoll model to a subset of the character’s limbs using an anchor system (that will be described later). Each physical limb is described as a rigid-body model defined by its mass, its center of mass, and its inertia tensor. Limbs are connected by joints, with angular limits for each degree of freedom. A physical simulator computes the movements of this skeleton model, and takes into account the various forces and torques acting on it, such as the action of the weight on each limb, any other external force such as response forces due to interactions, as well as the actuator torques computed by our approach to animate the skeleton.

Figure 2 gives an overview of our animation pipeline. At each time \( t \), we update the input skeleton regardless of the environment. Then, we consider the local surroundings of the character, thanks to a lightweight visibility model. Each obstacle type, its proximity, and its velocity are interpreted using a high-level rule-based system that provides kinematic objectives to the character’s upper-body, such as pushing obstacles away with one or both hands, or protecting itself. These objectives are handled using an IK solver, and leading to the generation of an intermediate kinematic skeleton. This skeleton can address high-level goals but does not integrate yet any dynamic response. To this end, we compute actuator torques on a specific subset of dynamic limbs defined by an anchor, using an antagonistic controller representation. The latter allows the physical model to move towards the time-varying kinematic skeleton, whatever our choice of stiffness (i.e., tension/relaxation) in the limbs. This stiffness is adapted in real time, either from our anticipation model before establishing contact with an obstacle, or from the current interaction forces during contact. As a result, the target motion defined by the kinematic skeleton drives the physically-based limbs selected by the anchor system, leading to the final responsive skeleton able to act on its dynamic surroundings, and to react to interactions in a plausible way.

We consider the following conventions to represent our skeletons (see Fig. 3): Each skeleton joint frame, located at the base of a bone, is encoded as a position \( p \) and an orientation \( q \) (unit quaternion), both w.r.t. its parent frame in the skeleton hierarchy. The local \( Y \) axis is assumed to be aligned with the bone. In order to limit the physically-based simulation to a local subset of the skeleton, the final upper-body responsive skeleton is partitioned into a set of kinematic parts and dynamic parts. Assuming that joint 0 corresponds to the root at the level of the hips, the partition between kinematic and dynamic bones is defined by the anchor \( a \in A \), where \( A \) is a set of admissible bones indicated in bold in Fig. 3. For a joint \( a \) and parent joint \( p(a) \), all ancestors \( (p_{\text{kin}}^a, q_{\text{kin}}^a), \ldots, (p_{\text{kin}}^0, q_{\text{kin}}^0) \) are considered as kinematics-driven, while the descendants \( (p_a, q_a), \ldots, (p_i, q_i) \) up to the end-effector \( i \) of the hierarchy are considered as dynamic ones with positions/orientations computed from the rigid-body simulator. Anchor \( a \) plays the role of a local physical-root, and the choice of \( a \) in the hierarchy depends on the nature of the current interaction.

4. Extension of Antagonistic-based Control

In this section, we first remind the general principle of antagonistic controllers [NF02] and their interest to control the dynamic part of the responsive skeleton, before describing our formulation, designed for joints with two or three degrees of freedom.

4.1. Antagonistic controllers principles

Controllers for physically-based character models generally rely on Proportional-Derivative (PD) controllers, which convert the angu-
Figure 4: Impact on the choice of an anchor on the responsive skeleton. Left: Input kinematic skeleton with yellow spheres representing the possible choices for an anchor. Right: responsive skeleton, where the weights of the limbs are only the external forces. The physical simulation takes control of the part of the character down the hierarchy, starting at the user-defined anchor point (where $a = 0$ is the hips join, and $a = 8$ is the wrist). The user may set an anchor on a single arm, or on both (shown for $a = 6$ and $a = 7$).

Figure 5: Antagonistic-based control for a 3-DOF joint. The system retrieves both, target and current orientation, and uses Swing-Twist Decomposition to get an orientation for each joint axis. Then, we convert the orientations to Axis-Angle representation and estimate the angular differences with respect to the user-defined spring-set points. Finally, these angular deviations are fed to the antagonistic controllers, which provide a torque that drives the physically-based model to the kinematic target orientation.

lar error in their proportional part to a spring-like force of prescribed stiffness. However, on the one hand, setting a fixed value for the stiffness does not allow a skeleton to reach precisely a target orientation when external torques are applied, such as the effect of weight. On the other hand, changing the stiffness by increasing or decreasing it over time to reach an objective angle affects the style of the motion, by making it more or less tense or relaxed. Derived from Feldman’s theory on bio-mechanical motor control [Fel66], the notion of antagonistic controller provides an elegant solution to stiffness control. First, it guarantees to reach the equilibrium at any arbitrary target orientation, specified within admissible bounds. Second, it preserves the joint tension, and therefore the motion style, throughout the animation. While antagonist controllers were already introduced in CG [NF02], the method was developed for pre-computed target motions only, and the original approach suffered from gimbal lock issues when dealing with the elbow joints. We therefore propose a new formulation, compatible with interactive motions where the target objective may change at run time, and expressed in local coordinates in order to avoid gimbal-lock issues.

Inspired from human anatomy, an antagonist controller models the action of a pair of antagonist muscles controlling the angle between two limbs at a given joint. The combined effect of two such muscles enables to reach any target angle with variable muscular stiffness. Considering a single rotational degree of freedom and the
The desired amount of tension/relaxation in our pose (see Fig. 6). where \( \theta \) is the target orientation for the character while still being able to tune the linear relation thus provides us with an intuitive way of modifying use while reaching the same equilibrium orientation. This simple linear relation thus provides us with an intuitive way to setup the stiffness values at the design stage of a game. At run time, our system further computes automatically the planar coordinates, with \( q \) being the unit conjugate quaternion of \( q \). Let us consider the initial T-pose of the character, and call \( b^\theta \) the unit quaternion representing the orientation of a joint-frame expressed with relative coordinates to the parent joint. We further attach in this relative coordinate system an orthogonal basis \((u_x, u_y, u_z)\) associated to the three degrees of freedom of this joint, and associate for each of them a low/high angular limit \((\theta^L, \theta^H)\). At run time, the articulated joint has an orientation given by the unit quaternion \( b \) expressed in relative coordinates, and \( q = b^\theta \) represents the rotation of by this joint in local coordinates, with \( \overline{b} \) being the unit conjugate quaternion of \( b^\theta \). Similarly, we consider the target rotation \( q^{kin} = b^{kin} \overline{b} \), \( b^{kin} \) being the target orientation also expressed in the relative coordinates system of its parent. Then, \( q \) (resp. \( q^{kin} \)) is decomposed as \( q = q_x q_y q_z \) (resp. \( q^{kin} = q_x^{kin} q_y^{kin} q_z^{kin} \)) using three consecutive Swing-Twist-Decomposition \([Dob15]\) along the axes \( u_x, u_y, \) and \( u_z \). As such, \( q_x \) represents a rotation around the axis \( u_x \), and similarly for the others. Converting these decomposed quaternions into an axis-angle representation leads to the three angles \((\theta_x, \theta_y, \theta_z)\) (resp. \((\theta_x^{kin}, \theta_y^{kin}, \theta_z^{kin})\)) corresponding to the local rotation along each individual degree of freedom. From these angles, the \((x, y, z)\) components of the torque \( \tau \) can be computed in this local frame using Eq. (1), and converted back to the global coordinate system before being used in the rigid-body simulator. Algorithm 1 summarizes this process.

Algorithm 1 Antagonistic Control

<p>| Input: current ( q ), target ( q^{kin} ), spring-set points ((\theta^L, \theta^H)) |</p>
<table>
<thead>
<tr>
<th>Output: torque ( \tau )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. function COMPUTETORQUE(q, ( q^{kin} ), ( \theta^L ), ( \theta^H ))</td>
</tr>
<tr>
<td>2. ( q_x q_y q_z \leftarrow \text{STD}(q) )</td>
</tr>
<tr>
<td>3. ( q_x^{kin} q_y^{kin} q_z^{kin} \leftarrow \text{STD}(q^{kin}) )</td>
</tr>
<tr>
<td>4. for each ( u \in \text{DOF} ) do</td>
</tr>
<tr>
<td>5. ( \theta_u \leftarrow q_x )</td>
</tr>
<tr>
<td>6. ( \theta_u^{kin} \leftarrow q_x^{kin} )</td>
</tr>
<tr>
<td>7. Compute ( \tau_u ) in equilibrium</td>
</tr>
<tr>
<td>8. Initialize ( k_L )</td>
</tr>
<tr>
<td>9. ( k_H \leftarrow k_L \frac{\theta^L - \theta^{kin}}{\theta^L - \theta^{kin}} - \frac{\theta^H - \theta^{kin}}{\theta^H - \theta^{kin}} )</td>
</tr>
<tr>
<td>10. ( \tau_u \leftarrow k_L (\theta_u^L - \theta_u) + k_H (\theta_u^H - \theta_u) - k_d \theta_u )</td>
</tr>
<tr>
<td>11. end for</td>
</tr>
<tr>
<td>12. return ( \tau )</td>
</tr>
<tr>
<td>13. end function</td>
</tr>
</tbody>
</table>

To avoid these artifacts, we propose to express this decomposition in a local frame where typical human-like gestures will be free from Gimble-lock issues (see Fig. 5).

Let us consider the initial T-pose of the character, and call \( b^\theta \) the unit quaternion representing the orientation of a joint-frame expressed with relative coordinates to the parent joint. We further attach in this relative coordinate system an orthogonal basis \((u_x, u_y, u_z)\) associated to the three degrees of freedom of this joint, and associate for each of them a low/high angular limit \((\theta^L, \theta^H)\). At run time, the articulated joint has an orientation given by the unit quaternion \( b \) expressed in relative coordinates, and \( q = b^\theta \) represents the rotation of by this joint in local coordinates, with \( \overline{b} \) being the unit conjugate quaternion of \( b^\theta \). Similarly, we consider the target rotation \( q^{kin} = b^{kin} \overline{b} \), \( b^{kin} \) being the target orientation also expressed in the relative coordinates system of its parent. Then, \( q \) (resp. \( q^{kin} \)) is decomposed as \( q = q_x q_y q_z \) (resp. \( q^{kin} = q_x^{kin} q_y^{kin} q_z^{kin} \)) using three consecutive Swing-Twist-Decomposition \([Dob15]\) along the axes \( u_x, u_y, \) and \( u_z \). As such, \( q_x \) represents a rotation around the axis \( u_x \), and similarly for the others. Converting these decomposed quaternions into an axis-angle representation leads to the three angles \((\theta_x, \theta_y, \theta_z)\) (resp. \((\theta_x^{kin}, \theta_y^{kin}, \theta_z^{kin})\)) corresponding to the local rotation along each individual degree of freedom. From these angles, the \((x, y, z)\) components of the torque \( \tau \) can be computed in this local frame using Eq. (1), and converted back to the global coordinate system before being used in the rigid-body simulator. Algorithm 1 summarizes this process.

To free the user from manually tuning the responsive character every time it should interact with a new obstacle, we provide a high-level, yet customizable anticipation and action method, enabling the character to use its upper-body to make its way through complex dynamic environments. To realize this, an anticipation mechanism extracts metadata from the obstacles in the field of view, and uses it to edit the responsive skeleton model based on a set of procedural rules. These rules generate action gestures for the upper-
The character anticipates a possible future collision with the obstacles in actively pushing the one at the closest distance from the center of a safety region. The choice of which arm to use to protect a particular part of the body is done according to which hand is at the closest distance to the obstacle, and the obstacle’s mass: If the difference between the distances of the obstacle to each hand is less than a threshold, or if the estimated weight of the object is perceived as greater than a certain value, the character uses both hands to interact with it. Otherwise, when different obstacles appear at the left and right sides of the character, reactions from the two arms can be generated independently from each other, and may overlap in time.

The general idea of the anticipation gesture is to move the character’s selected hand(s) to the closest point on the obstacle (or to the nearest accessible point to the obstacle, if the latter is not yet within reach), while adapting the motion speed and tension to the expected velocity and mass of the latter. To this end, we provide a set of procedural motion rules aimed at generating somewhat natural behaviour, and which the user may customize to better reflect the specific personality of the character to be animated. In addition, to model the possible uncertainty of the character’s prediction, we allow metadata to return false or disturbed information about the expected mass and velocity of the upcoming obstacle. Considering an obstacle $obs_i$, $m_i$ and $v_i$ stand for the expected (possibly fake) mass and velocity, while $\hat{m}_i$, $\hat{v}_i$ are the actual ones.

Let us consider $p_{0}$, $q_{0}$ the original position and orientation of the arm at time $t_0$ when the obstacle became the targeted one. The closest point on the obstacle is defined by its position $p_{c}$ and associated normal $n_{c}$. We aim to orient the hand such that the palm becomes tangent to the obstacle at position $p_{c}$. We thus define $q_{t}$, the objective orientation, as the rotation transforming the direction $b_x$ (orthogonal to the hand) into $n_{c}$, and the $b_y$ direction (aligned with the hand) into $b_y \times n_{c}$ (see vectors conventions in Fig. 7). At a given time $t$, we consider the following kinematics objectives $p^{kin}$, $q^{kin}$ for the hand trajectory:

$$
\begin{align*}
    p^{kin}(t) &= p_{0} + (p_{c} - p_{0}) \omega \frac{t - t_0}{t_r} \\
    q^{kin}(t) &= \text{SLERP}(q_{0}, q_{t}, \omega \frac{t - t_0}{t_r})
\end{align*}
$$

where $\omega$ is a smooth easing-function varying from 0 to 1, and $t_r$ is the character’s reaction time.

We propose a simple way to set the character’s reaction time, assuming that it only depends on the expected relative velocity of the obstacle $\|v_i\|$ with respect to the character’s root’s velocity:

$$
    t_r = \text{clamp} \left( \alpha_r \frac{d_l - r}{\|v_i\|}, t_{r_{min}}, t_{r_{max}} \right),
$$

where $d_l$ is the closest distance to the obstacle and $r$ is the character’s radius.
where $d_i$ is the closest distance to the obstacle, $r$ is the radius of the safety region, $\alpha_i \in [0,1]$ is a safety parameter ensuring that the hand should reach its final position slightly before the contact with the obstacle, and $(r_{min}, r_{max})$ are user-defined bounds.

Note that the trajectory we just defined is, by construction, free of external obstacles, otherwise the hand would switch to another target position during the planned reaction time, to anticipate a more sudden collision. In addition to these kinematics objectives, applied to either one or the two hands, the anticipation model also guides the stiffnesses $k_f$ and $k_{fp}$ of the antagonist controller - which are linked by the linear law of Eq. (3). Indeed, we make the assumption that the character adapts his tension/relaxation behavior based on the expected mass of the obstacle, leading to a relaxed motion on the expected mass of the obstacle, leading to a relaxed motion.

$\theta$ is handled using standard PD-controllers using a constant pre-gain value $k_p$ and then computed accordingly using Eq. (3).

Note that the previously defined bounds $r_{min}, r_{max}, k_{min}, k_{max}$ are customizable. The user can adapt them to a given character, but also, in all generality, set different bounds for different types of obstacles and safety regions. For instance, the character might always keep a slow motion and low stiffness when pushing away lightweight objects and more muscular tension when interacting with heavy ones. We model this behavior using the following linear relation between mass and stiffness:

$$k_L = \text{clamp} \left( k_{L_{\text{min}}} + (k_{L_{\text{max}}} - k_{L_{\text{min}}}) \frac{m}{m_{\text{max}}} k_{L_{\text{min}}}, k_{L_{\text{max}}} \right), \quad (6)$$

where $m_{\text{max}}$ is the extreme mass value that the character is expected to handle, and $k_{L_{\text{min}}}$ and $k_{L_{\text{max}}}$ are the limits for the lower gain $k_L$. The gain value $k_{L}$ is then computed accordingly using Eq. (3).

Note that the trajectory we just defined is, by construction, free of external obstacles, otherwise the hand would switch to another target position during the planned reaction time, to anticipate a more sudden collision. In addition to these kinematics objectives, applied to either one or the two hands, the anticipation model also guides the stiffnesses $k_f$ and $k_{fp}$ of the antagonist controller - which are linked by the linear law of Eq. (3). Indeed, we make the assumption that the character adapts his tension/relaxation behavior based on the expected mass of the obstacle, leading to a relaxed motion on the expected mass of the obstacle, leading to a relaxed motion.

$\theta$ is handled using standard PD-controllers using a constant pre-gain value $k_p$ and then computed accordingly using Eq. (3).

Note that the previously defined bounds $r_{min}, r_{max}, k_{min}, k_{max}$ are customizable. The user can adapt them to a given character, but also, in all generality, set different bounds for different types of obstacles and safety regions. For instance, the character might always keep a slow motion and low stiffness when pushing away lightweight objects and more muscular tension when interacting with heavy ones. We model this behavior using the following linear relation between mass and stiffness:

In our experiments, we consider the obstacles as dynamic articulated rigid bodies, visually represented as meshes deformed by skinning. The dynamic of these shapes is computed using a rigid body simulator, and the rest orientation of the articulated elements is handled using standard PD-controllers using a constant prescribed objective angle $\theta_0$. Therefore, as long as the contact lasts between the hand and the obstacle, the opposite response forces generated by the collision are used to apply the corresponding external torques to the simulated character on one hand, and to the obstacle on the other hand (see Fig. 9).

In addition, our method can handle different kinematic-driven behavior for the character’s arm, described as procedural rules. Our current implementation provides two specific scenarios: The first one is when a character walks along a slippery obstacle along which the hand remains in contact but can slide. To this end, the kinematic position of the hand is continuously adapted to target the updated closest point on the obstacle $p_i$ at the current frame, while remaining orthogonal to the normal at this position. The second scenario holds in non-sliding contact cases (eg. contact with an uneven wall), where the hand should remain at a fixed position $p_i^1$ relative to the obstacle as long as the arm can reach this position. To do so, the original contact position is stored in the local reference frame of the obstacle, and used as the kinematic objective of the responsive skeleton, until the hand needs to be moved to a new position. When this situation is detected, we trigger a temporary motion making the arm reaching a new updated closest point $p_i^2$. During the transition period begin parameterized by the time $t_r$, we consider a trajectory defined as a Cubic Hermite polynomial interpolating the two extreme positions $(p_i^1, p_i^2)$ with the two corresponding normals $(\alpha n_i^1, \alpha n_i^2)$, where $\alpha > 0$ controls how much the hand moves away from the obstacle. During this transition, the orientation of the hand is interpolated using SLERP in the quaternion space. We illustrate these two scenarios in Fig. 10.

Finally, when the targeted obstacle moves away of the safety region, the hand positions are interpolated, using a similar formulation than in Eq. (4), toward the next active obstacle in the priority list if it exists, or toward their current position in the kinematic skeleton otherwise.

5.4. Failing at anticipating or handling interactions
One of the key advantage of our method is that, as in real life, a character may miss-evaluate the nature of an obstacle or fail de-
tecting it in time, and therefore not handle it properly. This makes the generated behaviour more lively.

Figure 11: A character with constant tension in the arms (top) cannot maintain the prescribed posture when the mass of the red object increases, while an adaptation of tension (bottom) thanks to the antagonist stiffnesses in the controller enables to achieve it, within the limit of muscular strength.

First, the stiffnesses of the antagonist controllers only takes into account the expected obstacle mass and velocity \((m_i, v_i)\), which may be different from the real ones \((\hat{m}_i, \hat{v}_i)\). As a result, a heavier or faster obstacle than expected leads to a controller experiencing large angular displacement in order to absorb the momentum of the obstacle as illustrated in Fig. 11. Such miss-match will make the character look too relaxed, and hardly able to avoid the collision.

When the expected mass is too low, reaction time may be too slow. Moreover, an obstacle coming from some unseen orientation may actually collide with the character without being detected. In this case, the ragdoll model is dynamically updated in order to model a reaction to an impact, while trying to restore the current position of the kinematic skeleton. Let us consider that the obstacle collides with a given limb. Then the closest possible anchor position on this limb or one of its parent is selected to be the root of a temporary antagonistic controller. We then adapt Eq. (1) in order to take into account the additional torque exerted by the obstacle, while the equilibrium condition from Eq. (2) is set with the angle at the impact time. As illustrated in Fig. 12-middle for a collision on the character’s head, this approach allows to generate an adequate response to unexpected collisions as well.

6. Results and Discussion

We implemented our method as an interactive prototype in the game engine Unity 3D where the character is interactively controlled using a game-pad or a keyboard. The entire method is coded in high-level C# scripts, and all the presented examples run in real-time on a standard laptop (Intel Core i7, eight cores, running at 3.10 GHz). The main computational cost is spent on the rigid body simulation, while our additional anticipation model and procedurally-guided behavior do not bring any noticeable overhead. The default key-frame animation (with and without the IK)

runs at 6 ms/frame (155 FPS). Activating the ragdoll simulation on the character with a single spherical obstacle adds an additional 1 ms/frame (140 FPS). Our most complex scene (Fig. 14) includes 70 simulated plant assets and requires 25 ms/frame (40 FPS). Note that these measures could be optimized in skipping the simulation of the assets that are not interacted with.

In our experiments, the character is placed in an environment consisting on various dynamic elements, such as different vegetation types or hanging objects. The input kinematic skeleton is animated using a state-machine controller with keyframe animation of a walking gait. Antagonistic-based controllers are set at each degree of freedom of the shoulders, along with an initial stiffness \(k_2\) and angular limits \(\theta^u\) and \(\theta^l\). The character’s upper-body is protected using two spherical safety regions of radius \(r = 0.5 \text{ meter}\), located at the center of each shoulder joint. We use a default reaction time \(t_r = 0.5 \text{ s}\) with minimum and maximum bounds of \(t_{r_{\text{max}}} = 0.5 \text{ s}\) and \(t_{r_{\text{max}}} = 2 \text{ s}\). The maximum mass that the character is expected to handle is by default \(m_{\text{max}} = 15 \text{ kg}\). See Appendix 9 for additional information on the parameters used in our examples.

The animated results, described next, are provided in the companion video.

6.1. Interacting with Different Stiffnesses and Reaction Times

Fig. 1 illustrates a general situation where the character interacts with different types of outdoor elements. The diverse nature of the elements in terms of sizes or weights, are associated to different, but coherent, actions and reactions of the character.

A constant controller stiffness would lead to a fixed amount of tension during this journey. As such, the character would never adapt its muscular strength to the obstacle’s mass, leading to increased bending of the arms and difficulty to push heavier obstacles away, as shown in Fig. 11. Thanks to our local adaptation to the anticipated weight, from Eq. (3), the character is able to dynamically adapt to the current obstacle with a similar posture, in the limits of its muscular capacity.
The reaction time $t_r$ may affect the success of an interaction in certain cases (see Fig. 12-right and Fig. 13). If the character reacts too slowly when an obstacle is heading its way, its arms will not take an optimal position to stop the obstacle. Reducing reaction time increases the speed at which the reactive skeleton assumes the correct protective posture, making it easier to manage the interaction.

![Figure 13: Short reaction time (top) vs longer (bottom), with detection time and all other parameters unchanged. Note that by using the longer reaction time, the character did not have time to properly place his hand before impact.](image)

6.2. Anticipating Natural Surroundings

Table 1 summarises both internal factors and external metadata that can be retrieved by the anticipation system, along with their impact on the behaviour of the interaction. To better show these effects and experiment with interactions in dynamic environments, we built a 3D scene comprising multiple assets, as illustrated in Fig. 14. Appendix 9 provides the set of parameters used for our 3D models.

We tested different types of behaviors dependent on metadata in this environment, using our high-level procedural rules for cases such as quick anticipation movements, or long-term behavioral responses. The character adapts its muscular rigidity based on the expected mass retrieved from the active obstacles entering the safety region. A comparison between character interactions with, versus without, such adaptation is shown in Fig. 15, as well as in the companion video. Moreover, since the velocity of an incoming object is used to adapt reaction time $t_r$ (see Eq. (5)), the character is able to automatically generate quick anticipation movements to protect itself.

6.3. Discussion and Limitations

The previous results illustrated the benefits of our method in various scenarios, through real-time interactions with a variety of obstacles. While they demonstrate the benefits of automatic tension/relaxation adaptation, thanks to our new formulation of antagonist control, the current implementation is limited to the case of characters pushing obstacles away from their safety regions. To improve realism, more complex behaviors could be integrated, such as refining our anticipation process with better predicted hazards, modeling protective or avoidance gestures for obstacles with large momentum, and dynamically adapting the character’s response when the true mass of an obstacle is perceived, during interaction. We believe that these would be quite easy extensions of our method, thanks to our flexible procedural approach, guided by the observed environment.

As our prototype was developed as a generic proof of concept, some of the parameters and procedural rules should be refined to improve the plausibility of motions and contact modeling. For instance, the character’s wrist sometimes has too great a twist angle during the anticipation phase or rotates too slowly in relation to the arm. This could be addressed via dedicated IK constraints for the wrist. Additionally, some small gaps can also be seen between the hands and the mesh used to represent the vegetation. The use of finer collision proxy representation would avoid such visible artifacts. Also, the velocity (magnitude and direction) of the obstacles could be considered in the anticipation behavior to deal with multiple obstacles more plausibly, as our current system only handles the closest one.

The main limitation of our approach is obviously the independence between the upper-body motion and the orientation of the character, as well as the locomotion gait of the lower-body. Indeed, in real-life, the motion of our legs is linked to our actions. For instance, we stop moving when expecting an unavoidable hit, or we bend hips and knees in order to push heavy obstacles. Such coupling is not orthogonal to our approach, but would require - on top of additional rules - a more global simulation method including a locomotion controller.

Lastly, while enabling precise authoring of the character’s behavior thanks to procedural rules is usually desired in video-game development, the number of cases to explicitly define can be a burden. In addition, some of the parameter choices may result in limited plausibility, and the rules themselves may have a limited valid-

<table>
<thead>
<tr>
<th>Internal Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Region Radius</td>
<td>Reach of the anticipation</td>
</tr>
<tr>
<td>Reaction Time $t_r$</td>
<td>Rapidity of the interaction</td>
</tr>
<tr>
<td>Reaction Time Bounds $[t_{r\min}, t_{r\max}]$</td>
<td>Reaction capacity (faster/slower)</td>
</tr>
<tr>
<td>Stiffness Gain $k_L$</td>
<td>Muscle rigidity</td>
</tr>
<tr>
<td>Stiffness Gain Bounds $[k_{L\min}, k_{L\max}]$</td>
<td>Stiffness capacity (stronger/weaker)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Factor</th>
<th>Impact of Anticipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected Mass $m$</td>
<td>Increase/Decrease Stiffness $k_L$</td>
</tr>
<tr>
<td>Expected Velocity $v$</td>
<td>Increase/Decrease Reaction Time $t_r$</td>
</tr>
</tbody>
</table>

Table 1: Internal parameters and external factors that influence character behavior. Note that both, the expected mass and velocity returned by the metadata, might differ from real values. The character will then mis-adapt its behaviour, according to its current perception of the object.
Figure 14: Our test environment allows rich interactions with multiple dynamic elements, including flowers, flexible trees, a semi-rigid fence, and hanging objects.

Figure 15: While light vegetation like tall grass can be easily pushed away, heavier items like the wooden gate may require more muscle stiffness to be pushed back. When the character does not adapt its tension/relaxation to obstacles (top), it results in over-stiff movements for light plants, inaccurate contacts, and over-relaxed motions when facing heavier obstacles. With our method (middle), the character automatically adapts the stiffness of each arm while making its way through. Bottom: Depiction of the stiffness variation along the obstacles.

Figure 7. Conclusion and Future work
In this work, we have proposed a method to generate real-time upper body movements from simple keyframe locomotion inputs, such that characters make their way through dynamic environ-
ments. Our hybrid model for character animation combines kinematics, IK constraints, and lightweight physics to produce a responsive skeleton able to react to any kind of external obstacle. A local anchor system identifies the limbs that need to be dynamically simulated during the interaction, and leveraging antagonistic controllers to generate actuator torques that follow a reference animation, while controlling the level of tension/relaxation independently. A flexible anticipation mechanism allows the user to combine both, information from the surroundings and changes in the character’s stiffness and reaction time, enabling high-level authoring in the way the character handles the interactions. Overall, we believe our reactive character model provides a practical and flexible framework well suited to video game pipelines where precise control of behaviors and real-time computation are essential.

A promising future direction would be to explore the coupling of our approach with some non-supervised learning method, in order to improve the fine-grain plausibility of interactions while maintaining the same level of control. Procedural creation could then be used to define the coarse behavior, while reinforcement learning would help guide and enrich the local details of the animation, thanks to the fine-tuning of the antagonistic control for each joint, set to minimize the muscular effort undergone by the character.

8. Acknowledgments
This work has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 860768 (CLIPE project). We also thank Prof. Pierre Poulin (Université de Montréal) for his help and comments on the manuscript.

References


This work is published at Computer Graphics Forum, special issue of SCA 2022.
9. Appendix

9.1. Character and Environment Description

Table 2 gives the parameters used in our experiments for each rigid-body in the responsive skeleton.

<table>
<thead>
<tr>
<th>Rigid-body</th>
<th>Mass (kg)</th>
<th>Center of Mass ((x,y,z)) (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hips&amp;Spine</td>
<td>10.56</td>
<td>0.00 0.00 0.00</td>
</tr>
<tr>
<td>Chest</td>
<td>25.2</td>
<td>0.00 0.22 -0.03</td>
</tr>
<tr>
<td>Upper Chest</td>
<td>10</td>
<td>0.00 0.35 -0.04</td>
</tr>
<tr>
<td>Neck&amp;Head</td>
<td>4.8</td>
<td>0.00 0.60 0.00</td>
</tr>
<tr>
<td>Shoulder</td>
<td>1</td>
<td>±0.06 0.44 -0.04</td>
</tr>
<tr>
<td>Upper Arm</td>
<td>2.95</td>
<td>±0.19 0.42 -0.03</td>
</tr>
<tr>
<td>Forearm</td>
<td>1.59</td>
<td>±0.23 0.15 -0.02</td>
</tr>
<tr>
<td>Hand</td>
<td>0.5</td>
<td>±0.26 -0.13 -0.02</td>
</tr>
</tbody>
</table>

Table 2: Rigid-body settings for the upper-body model.

Most of the 3D obstacles in our environments (see Fig. 9) are rigged and defined using PD controllers with different rest-positions. Therefore, both fences and hanging plants follow similar principles, each set to a different dynamics. The main parameter values are given in Table 3.

<table>
<thead>
<tr>
<th>Asset</th>
<th>Mass (kg)</th>
<th>(k_p)</th>
<th>(k_d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunflower (Large)</td>
<td>1.25</td>
<td>20.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Sunflower (Small)</td>
<td>1.0</td>
<td>20.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Bush</td>
<td>0.2</td>
<td>200.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Banana Tree</td>
<td>10.0</td>
<td>50.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Tree Branch</td>
<td>3.0</td>
<td>500.0</td>
<td>15.0</td>
</tr>
<tr>
<td>Fence</td>
<td>10.0</td>
<td>100.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Fence w/ Door</td>
<td>10.0</td>
<td>100.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Hanging Bucket</td>
<td>1.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Swing</td>
<td>2.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hanging Fruit</td>
<td>0.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: Obstacles used in our natural environment.

Author’s version.  
This work is published at Computer Graphics Forum, special issue of SCA 2022.