ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (2022), Number 8

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

Eduardo Alvarado[1], Damien Rohmer[1], Marie-Paule Cani[1]

{eduardo.alvarado-pinero,damien.rohmer,marie-paule.cani}@polytechnique.edu
[1] LIX, École Polytechnique/CNRS, Institut Polytechnique de Paris, Palaiseau, France

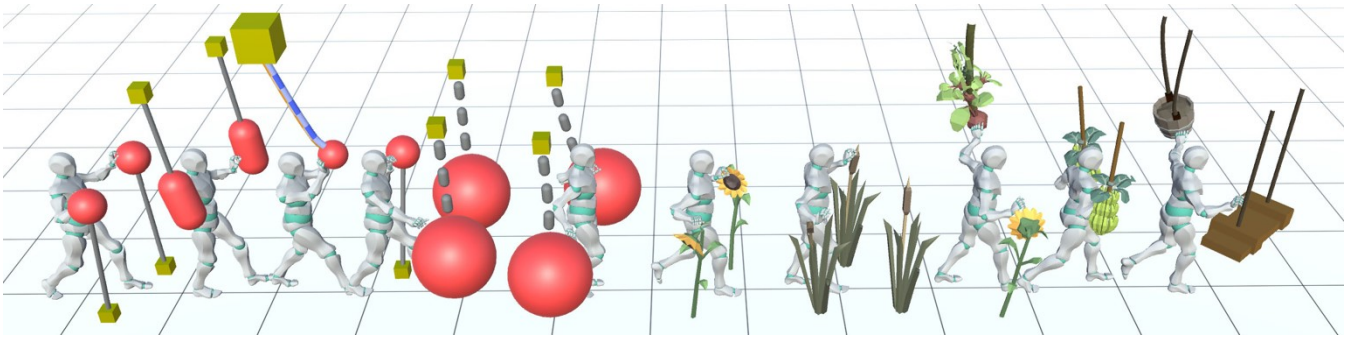

Figure 1: *Thanks to a hybrid physical-kinematic model allowing the adjustment of tension/relaxation in the upper-body, our responsive character is able to anticipate interactions in order to make its way through a variety of obstacles, while being driven by an input walking motion.*

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

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

does not only depend on our prediction but also adapts dynamically to the environment's response, especially if it differs from our expectations.

While recent video games produce highly realistic images of natural scenes [Koj19, Gue22], the range of dynamic interactions they can handle between a character and its environment remains quite limited. Indeed, the combined need of predictable overall motion and high performance usually prevents the use of fully physically-based models in gaming applications. Therefore, modeling interactions usually requires the dynamic selection of a best animation clip from a motion library, and coupling it in real time with a suitable response of the environment, as was done, for instance, to model characters walking on loose grounds such as mud or snow [Roc18]. Leveraging such lightweight animation clips with simple adaptation to various environments would tremendously ease the design of video games. Ideally, a game designer should be able to use standard pre-recorded animations, and refine and adapt these efficiently in a semi-automatic way to different behaviors. To this end, the gestures of the character should (i) automatically adapt in a plausible way to its surrounding environment when needed (i.e., a character making their way through dynamic environments such as in dense vegetation), (ii) still follow, while doing so, the high-level intent set by the designer, while (iii) relying on the initial animation clip for the general motion.

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 obstacles. 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 associated to visual skinned-rigs, and make its way through the observed 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 visibility as well as the predicted stiffness of obstacles. This anticipation mechanism is used to guide an Inverse Kinematics (IK) objective that drives the character's gestures with suitable synchronization and amount of muscular stiffness. Our solution allows characters to react to any upcoming obstacle, and therefore to plan actions, 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 affecting 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 animations 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 intuitive tuning of gestures and interaction styles by dissociating the position and orientation of the limbs from their degree of muscular rigidity (Sec. 4).
- An efficient, yet generic and customizable anticipation mechanism 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 animations that adapt to the surroundings (Sec. 5).

Fig. 1 illustrates the application of our method to a real-time character 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 environment is an active research topic. It was addressed in different fields, from Computer Graphics (CG) to robotics and biomechanics (e.g., [SP17]). For the sake of conciseness, we focus here on CG research tackling the reciprocal influence between character 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, HBWO95]. *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 [YLv07, WFH10]. However, despite their simplicity, instability and stiffness control are still recurrent problems when reproducing highly-dynamic and accurate animations.

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 [LYV*10], 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 movements. 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 hamper the intuitiveness and precision that a user may want to address when parameterizing its character behavior. Neff and Fiume [NF02] introduce an antagonistic-based PD formulation, allowing 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 [BSTS11, BN15, MMLG*19], 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 [KAK*22]. These approaches can preserve character balance to achieve realistic locomotion up to complex acrobatic gestures [CMM*18,PALvdP18,PCZ*20]. 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 [HKS17, HFO*20, MHL*21]. 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 [BCHF19, WGSF20]. 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 vegetations 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 [WK88] 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 [ALCS18]. It was used, for instance, in combination with physical constraints [BMT96] and hierarchical motion curve editing [LS99]; it was also combined with short-term dynamical effects [RRSK12]. 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 [KMG*, KGM*12]. 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 [MAA*09].

Similar to our approach, some work proposed local hybrid models addressing the motion of some parts of the upper-body, such as the arms. Zordan and Hodgins [ZH99] added contact constraints to locally modify motion capture data, and extended it further to integrate dynamic responses [ZH02]. Arm motion was also studied using learning-based approaches in sport applications [LH18], and used to infer lower-body motions in VR [YKL21]. 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 [EHSN19], 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 [YLS04], or for windy environments [LGS*11]. A simplified friction model was used to represent human locomotion efficiently on semi-flooded grounds [BTZZ18] or through dense vegetation represented using billboards [PARC21]. 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 [APRC22]. 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

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 $((p_0^{kin}, q_0^{kin}), \cdots, (p_{p(a)}^{kin}, q_{p(a)}^{kin}))$ are considered as kinematics-driven, while the descendants $((p_a, q_a), \cdots, (p_n, q_n))$ up to the end-effector n 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.

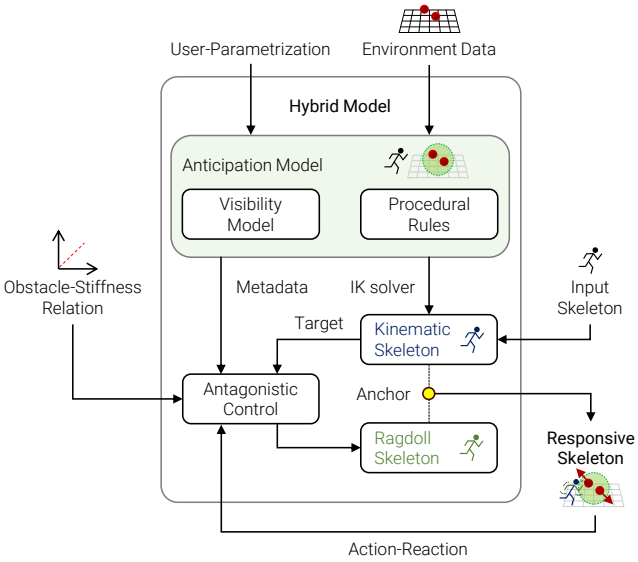

Figure 2: *Overview of the animation pipeline for the hybrid model.*

Fig. 4 shows the effect of choosing different nodes for a for a static character with no external, upper-body interactions, where only the weights of the limbs are applied to the rigid-body simulation. Note that several anchors can be set at the same time, for instance for having both left and right arms be driven by dynamics, while the rest of the body remains fully kinematic.

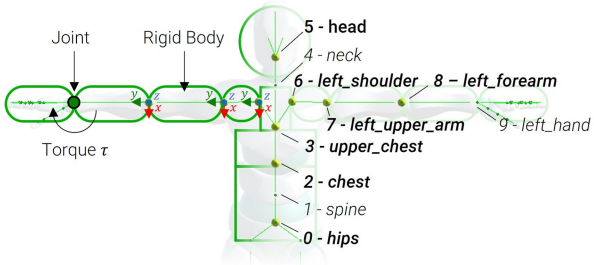

Figure 3: *Physical model of a character. The anchor is chosen among the possible positions $a \in A$, shown as spheres at the joints. The anchor a partitions the skeleton hierarchy into a set of kinematic versus dynamic bones.*

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-

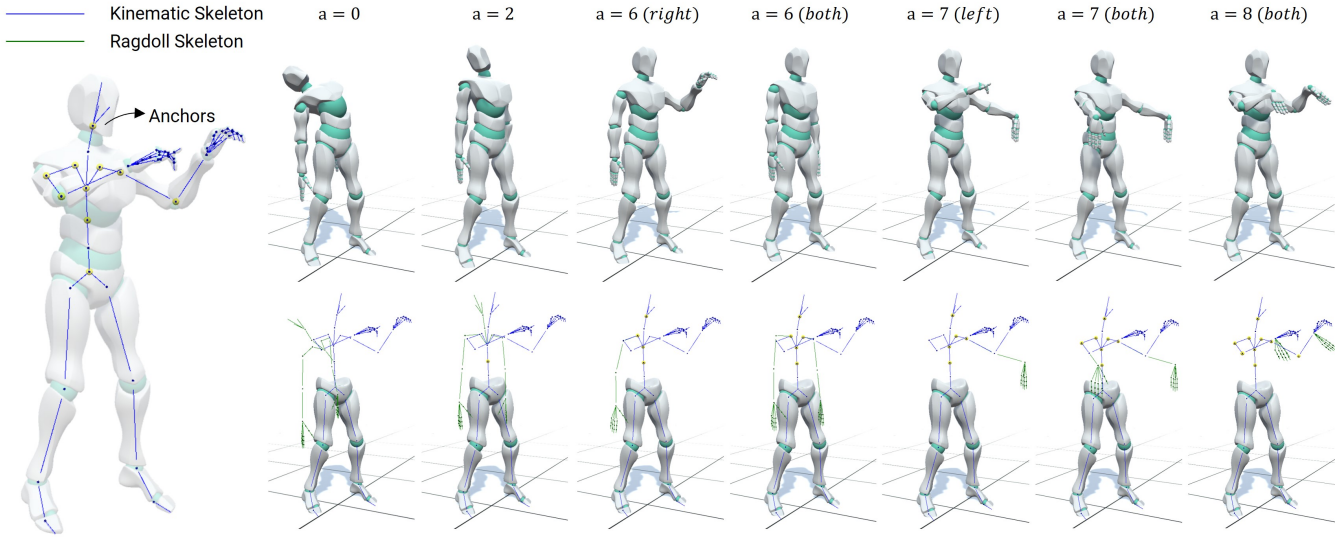

Figure 4: *Impact on the choice of an anchor on the responsive skeleton. Left: Input kinematic skeleton with yellow spheres representing the possible choices for an anchor. Right: responsive skeleton, where the weights of the limbs are only the external forces. The physical simulation takes control of 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 $a = 7$).*

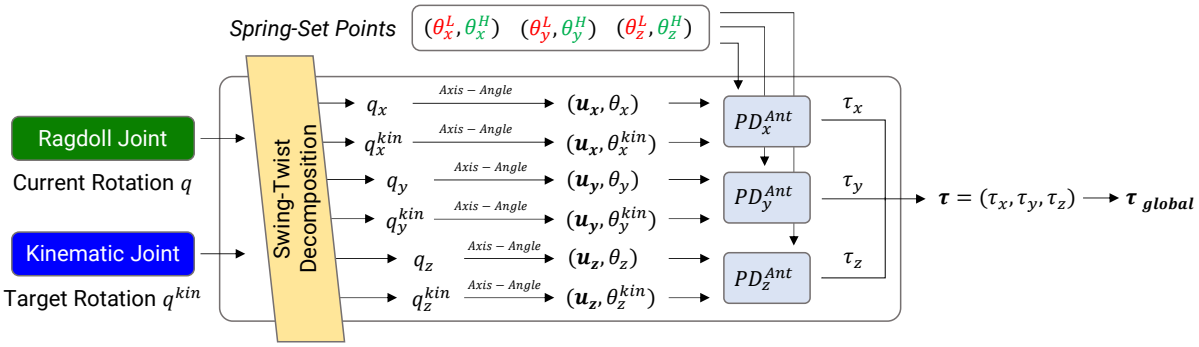

Figure 5: *Antagonistic-based control for a 3-DOF joint. The system retrieves both, target and current orientation, and uses Swing-Twist Decomposition to get an orientation for each joint axis. Then, we convert the orientations to Axis-Angle representation and estimate the angular differences with respect 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

relative angle θ between the limbs with respect to the initial T-pose, the dynamics of the antagonist controller is parameterized by a pair of lower and upper stiffnesses (or proportional gains) (k_L, k_H) associated to two angular soft limits (θ_L, θ_H) at the joint, and a derivative gain k_d such that

$$\tau + \tau_{ext} = k_L(\theta^L - \theta) + k_H(\theta^H - \theta) - k_d\dot{\theta} \ , \tag{1}$$

where τ is the current actuator torque exerted by the controller and τ_{ext} is the torque resulting from the external forces.

Let the motion of the articulation be guided by an objective target angle θ^{kin}, obtained from the kinematic skeleton. When θ reaches θ^{kin}, the antagonistic controller is at static equilibrium ($\tau = 0$), leading to:

$$\tau^{kin}_{ext} = k_L(\theta^L - \theta^{kin}) + k_H(\theta^H - \theta^{kin}), \tag{2}$$

where τ^{kin}_{ext} is the total torque in this state.

This equilibrium constraint implicitly links the two stiffness values, and can be rewritten as:

$$k_H = k_L \frac{\theta^L - \theta^{kin}}{\theta^{kin} - \theta^H} - \frac{\tau^{kin}_{ext}}{\theta^H - \theta^{kin}}. \tag{3}$$

This relation describes the so-called isoline of the controller, whose slope is independent from the external torques exerted on the joint. In addition to compensating for the respective error in the orientation, each pair of gains k_L and k_H lying on the isoline represents all the possible relaxation/tension configurations that the joint may use while reaching the same equilibrium orientation. This simple linear relation thus provides us with an intuitive way of modifying the target orientation for the character while still being able to tune the desired amount of tension/relaxation in our pose (see Fig. 6).

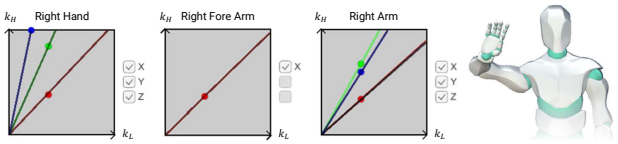

Figure 6: *Isolines between antagonist stiffnesses, for each limb of the responsive skeleton of the right arm. The slopes of the depicted line-segments are function of the target angular pose. The isolines represent the degree of freedom of the stiffness to reach this specific pose. Navigating interactively along these lines allows an intuitive way to setup the stiffness values at the design stage of a game. At run time, our system further computes automatically the corresponding coordinates along these lines based on a procedural rule taking into account the parameter of the obstacles (see Sec. 5).*

4.2. Adaptation to joints with multiple degree of freedom

Antagonistic controller must be formulated in relation to each individual degree of freedom of a joint as it depends on extremal values θ^L,θ^H. A naive approach to extend this formulation to a joint with two of three degrees of freedom would be to decompose the joint orientation along Euler angle representation. However, decomposing a joint orientation expressed in a global frame system using Euler angles would lead to gimble lock artifacts during the animation.

Algorithm 1 Antagonistic Control

Input: current q, target q^{kin}, spring-set points (θ^L, θ^H)
Output: torque τ

function COMPUTETORQUE(q, q^{kin}, θ^L, θ^H)
  $q_z\, q_y\, q_x \leftarrow \text{STD}(q)$
  $q^{kin}_z\, q^{kin}_y\, q^{kin}_x \leftarrow \text{STD}(q^{kin})$
  for each $u \in$ DOF **do**
    $\theta_u \leftarrow q_u$
    $\theta^{kin}_u \leftarrow q^{kin}_u$
    Compute τ^{eq}_{ext} in equilibrium
    Initialize k_L
    $k_H \leftarrow k_L \frac{\theta^L_u - \theta^{kin}_u}{\theta^{kin}_u - \theta^H_u} - \frac{\tau^{eq}_{ext}}{\theta^H_u - \theta^{kin}_u}$
    $\tau_u \leftarrow k_L(\theta^L_u - \theta_u) + k_H(\theta^H_u - \theta_u) - k_d\dot{\theta}_u$
  end for
  return τ
end function

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^0 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 $(\theta^L_{x/y/z}, \theta^H_{x/y/z})$. At run time, the articulated joint has an orientation given by the unit quaternion b expressed in relative coordinates, and $q = b\,\overline{b^0}$ represents the rotation of by this joint in local coordinates, with $\overline{b^0}$ being the unit conjugate quaternion of b^0. Similarly, we consider the target rotation $q^{kin} = b^{kin}\,\overline{b^0}$, b^{kin} being the target orientation also expressed in the relative coordinates system of its parent. Then, q (resp. q^{kin}) is decomposed as $q = q_z\, q_y\, q_x$ (resp. $q^{kin} = q^{kin}_z\, q^{kin}_y\, q^{kin}_x$) 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. $(\theta^{kin}_x, \theta^{kin}_y, \theta^{kin}_z)$) corresponding to the local rotation along each individual degree of freedom. From these angles, the (x, y, z) components of the torque τ 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.

5. Anticipation and Action on Upcoming Obstacles

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-

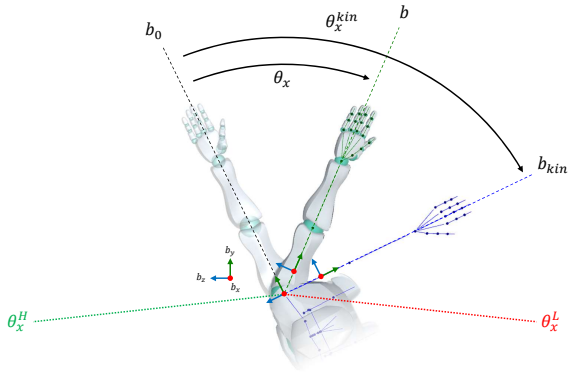

Figure 7: *Left upper-arm configuration for the degree of freedom corresponding to the rotation around the local x-axis (vertical direction) at the shoulder. All angles are measured with respect to the initial T-pose orientation b_0. θ_x describes the current orientation, θ_x^{kin} the target one, and (θ_x^H, θ_x^L) are the high/low soft-limits of the antagonistic controller.*

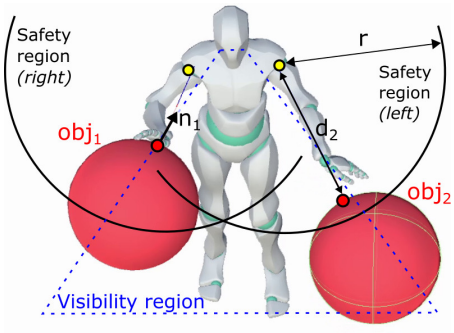

Figure 8: *Visibility and safety Regions. The approximated frustum-like volume keeps track of the obstacles within the visual bounds of the character, while the safety regions are used to target the closest obstacles.*

body's kinematic skeleton so that the character protects itself and responds differently to obstacles of different nature. They also automatically tune the antagonist stiffnesses of the ragdoll skeleton, so that obstacles can effectively be pushed out of the way. In practice, this is achieved through one or several *safety region(s)* around the character, which it will always try to keep free from obstacles; this is detailed next.

5.1. Detecting Upcoming Obstacles

We use a frustum-like *visibility region* starting from the character's head and propagating along the sight direction to make our character aware of his environment, and in particular of the obstacles to come. We further define a set of *safety regions* representing parts of the body that a character may try to protect with its arms while walking in some dense, dynamic environment. We approximate these safety regions as spheres around some body parts such as the left and right shoulders. Their radius r can be defined from the character's size and arm length (note that using a slightly larger radius than the arm length is relevant, since it will enable the character to anticipate a collision by taking the right posture in advance, before eventually catching the object). Our system considers that an obstacle must be stopped or pushed by the character's hand if it is both in the *visibility region* and intersects with at least one of the *safety regions* (see Fig. 8).

5.2. Anticipation behavior

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_c, 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:

$$\begin{aligned} p^{kin}(t) &= p_0 + (p_c - p_0)\,\omega\left(\tfrac{t-t_0}{t_r}\right) \\ q^{kin}(t) &= \mathrm{SLERP}\left(q_0, q_c, \omega\left(\tfrac{t-t_0}{t_r}\right)\right) \end{aligned}, \tag{4}$$

where ω is a smooth easing-function varying from 0 to 1, and tr 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:

$$t_r = clamp\left(\alpha_{tr}\,\frac{d_i - r}{\|v_i\|}, t_{rmin}, t_{rmax}\right), \tag{5}$$

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

where d_i is the closest distance to the obstacle, r is the radius of the safety region, $\alpha_{t_r} \in [0,1]$ is a safety parameter ensuring that the hand should reach its final position slightly before the contact with the obstacle, and (t_{rmin}, t_{rmax}) 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_L and k_H 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 for lightweight objects and more muscular tension when interacting with heavy ones. We model this behavior using the following linear relation between mass and stiffness:

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

where m_{max} is the extreme mass value that the character is expected to handle, and k_{Lmin} and k_{Lmax} are the limits for the lower gain k_L. The gain value k_H is then computed accordingly using Eq. (3).

Note that the previously defined bounds $t_{rmin}, t_{rmax}, k_{Lmin}, k_{Lmax}$ 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 flexible and spiky vegetation such as brambles, while being able to move faster and exert a higher tension toward other type of long-anticipated obstacles, such as a fence to be opened. Moreover, an obstacle moving towards the eyes of the character can thus be associated with a faster movement of anticipation than if it were moving towards his chest.

5.3. Behavior during the contact phase

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 θ_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

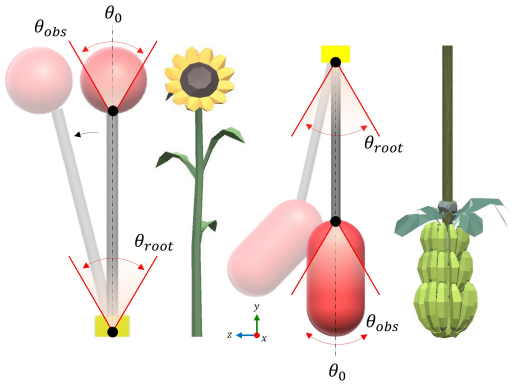

Figure 9: *Obstacles such as flowers (left) or hanging fruits (right) are rigged and their background dynamics are defined by self-balancing poles that are kept upright using PD controllers. The red object (defined by the corolla, or by the fruits, respectively) is used as a proxy for computing the interaction with the character.*

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.

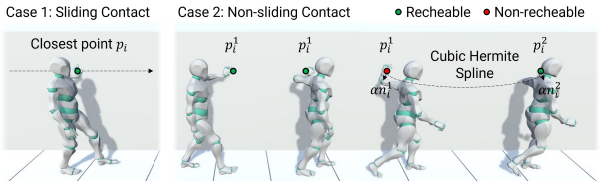

Figure 10: *Depending on the user's choice, the character's interactions can be described as a continuous, sliding contact (left) or as a discretely updated contact position depending on the reach of the character's arms (right).*

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.

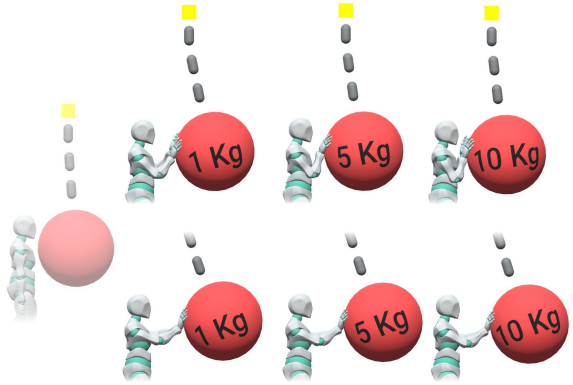

Figure 11: *A character with constant tension in the arms (top) cannot maintain the prescribed posture when the mass of the red object increases, while an adaptation of tension (bottom) thanks to the antagonist stiffnesses in the controller enables to achieve it, within the limit of muscular strength.*

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)

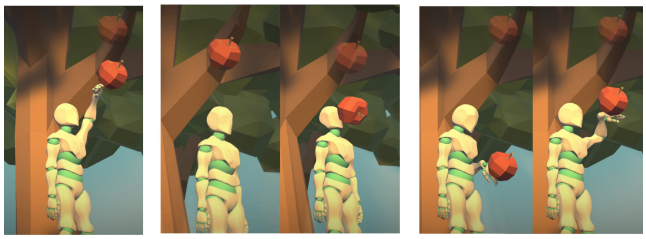

Figure 12: *When the character fails anticipating a collision, our hybrid system system dynamically updates to generate an appropriate responds. Left: Expected correct handling of the falling apple. Middle: The apple fall outside of the visibility region leading to a collision with the head which is pushed back by the impact thanks to the dynamic update of the anchor point on the responsive skeleton. Then our antagonistic-based formulation helps it recover its initial position. Right: The apple falls too fast compared to the character's reaction time leading to non-optimal position to stop it.*

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 25ms/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_L and angular limits θ^L and θ^H. The character's upper-body is protected using two spherical safety regions of radius $r = 0.5\ meter$, located at the center of each shoulder joint. We use a default reaction time $t_r = 0.5\ s$ with minimum and maximum bounds of $t_{rmin} = 0.5\ s$ and $t_{rmax} = 2\ s$. The maximum mass that the character is expected to handle is by default $m_{max} = 15\ kg$. 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.

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

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.

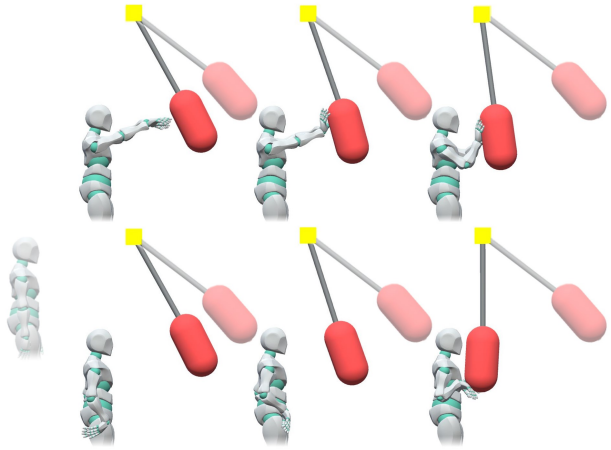

Figure 13: *Short reaction time (top) vs longer (bottom), with detection time and all other parameters unchanged. Note that by using the longer reaction time, the character did not have time to properly place his hand before impact.*

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

Internal Factor	***Description***
Safety Region Radius r	Reach of the anticipation
Reaction Time t_r	Rapidity of the interaction
Reaction Time Bounds $[t_{rmin}, t_{rmax}]$	Reaction capacity (faster/slower)
Stiffness Gain k_L	Muscle rigidity
Stiffness Gain Bounds $[k_{Lmin}, k_{Lmax}]$	Stiffness capacity (stronger/weaker)
External Factor	***Impact of Anticipation***
Expected Mass m	Increase/Decrease Stiffness k_L
Expected Velocity v	Increase/Decrease Reaction Time t_r

Table 1: *Internal parameters and external factors that influence character behavior. Note that both, expected the 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.*

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 approached, 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-

Figure 14: *Our test environment allows rich interactions with multiple dynamic elements, including flowers, flexible trees, a semi-rigid fence, and hanging objects.*

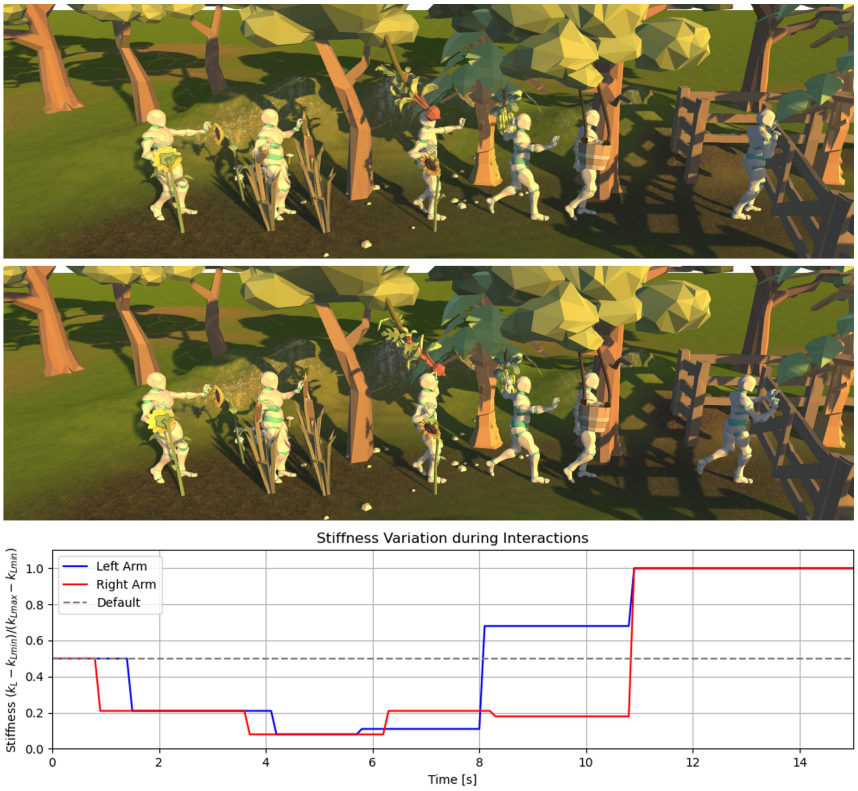

Figure 15: *While light vegetation like tall grass can be easily pushed away, heavier items like the wooden gate may require more muscle stiffness to be pushed back. When the character does not adapt its tension/relaxation to obstacles (top), it results in over-stiff movements for light plants, inaccurate 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.*

ity range. For instance, defining the controller's stiffness as a linear function of the obstacle mass may not be a valid approximation anymore for a wide range of masses, or for objects moving at high velocities. This choice may be a limitation in some cases compared to more complex, learning-based approaches. Using such methods could allow to tune tension/relaxation not only in anticipation before a collision, but also continually during the contact phase, resulting in more convincing interactions.

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

[ALCS18] ARISTIDOU A., LASEBRY J., CHRYSANTHOU Y., SHAMIR A.: Inverse kinematics techniques in computer graphics: A survey. *Computer Graphics Forum 37*, 6 (2018), 35–58. 3

[ALP04] ABE Y., LIU C. K., POPOVIĆ Z.: Momentum-based parameterization of dynamic character motion. *Computer Animation 2004 - ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (aug 2004), 173–182. 2

[APRC22] ALVARADO E., PALIARD C., ROHMER D., CANI M.-P.: Real-Time Locomotion on Soft Grounds With Dynamic Footprints. *Frontiers in Virtual Reality 3* (mar 2022), 3. 3

[BCHF19] BERGAMIN K., CLAVET S., HOLDEN D., FORBES R. J.: DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Trans. Graph. 38*, 6 (2019), 11. 3

[BMT96] BOULIC R., MAS R., THALMANN D.: A robust approach for the control of the center of mass with inverse kinetics. *Computer & Graphics 20*, 5 (1996), 693–701. 3

[BN15] BHATTACHARJEE T., NIEMEYER G.: Antagonistic muscle based robot control for physical interactions. In *Proceedings - IEEE International Conference on Robotics and Automation* (2015), vol. 2015-June, pp. 298–303. 3

[BSTS11] BUCHLI J., STULP F., THEODOROU E., SCHAAL S.: Learning Variable Impedance Control. *The International Journal of Robotics Research 30*, 7 (2011), 820–833. 3

[BTZZ18] BERMUDEZ L., TESSENDORF J., ZIMMERMANN D., ZORDAN V.: Real-time locomotion with character-fluid interactions. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games (MIG)* (2018), ACM. 3

[CMM*18] CHENTANEZ N., MÜLLER M., MACKLIN M., MAKOVIYCHUK V., JESCHKE S.: Physics-based Motion Capture Imitation with Deep Reinforcement Learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games (MIG)* (2018), ACM. 3

[Dob15] DOBROWOLSKI P.: Swing-Twist Decomposition in Clifford Algebra. *arXiv:1506.05481* (jun 2015). 6

[EHSN19] EOM H., HAN D., SHIN J. S., NOH J.: Model predictive control with a visuomotor system for physics-based character animation. *ACM Trans. Graph. 39*, 1 (oct 2019). 2, 3

[Fel66] FELDMAN A.: Functional tuning of the nervous system with control of movement or maintenance of a steady posture. *Biophysics* (1966). 5

[GP12] GEIJTENBEEK T., PRONOST N.: Interactive character animation using simulated physics: A state-of-the-art review. *Computer Graphics Forum 31*, 8 (dec 2012), 2492–2515. 2

[Gue22] GUERRILLA GAMES: Horizon Forbidden West, 2022. 2

[HBWO95] HODGINS J. K., BROGAN D. C., WOOTEN W. L., O'BRIEN J. F.: Animating human athletics. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (1995), 71–78. 2

[HFO*20] HOLDEN D., FORGE L., OUSSAMA KANOUN C., UBISOFT C., BLVD S. L., KANOUN O., PEREPICHKA M., POPA T.: Learned motion matching. *ACM Transactions on Graphics (TOG) 39*, 4 (jul 2020), 13. 3

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Trans. Graph 36*, 4 (2017). 3

[KAK*22] KWIATKOWSKI A., ALVARADO E., KALOGEITON V., LIU C. K., PETTRÉ J., VAN DE PANNE M., CANI M.-P.: A Survey on Reinforcement Learning Methods in Character Animation. *Computer Graphics Forum 41* (mar 2022). 3

[KGM*12] KARIM A. A., GAUDIN T., MEYER A., BUENDIA A., BOUAKAZ S.: Procedural Locomotion of Multi-Legged Characters in Dynamic Environments. In *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (2012). 3

[KH10] KWON T., HODGINS J.: Control Systems for Human Running using an Inverted Pendulum Model and a Reference Motion Capture Sequence. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), p. 129–138. 2

[KHH17] KWON T., HODGINS J. K., HODGINS J. K.: Momentum-Mapped Inverted Pendulum Models for Controlling Dynamic Human Motions. *ACM Transactions on Graphics 36*, 4 (jul 2017), 1. 2

[KMG*] KARIM A. A., MEYER A., GAUDIN T., BUENDIA A., BOUAKAZ S.: Generic Spine Model with Simple Physics for Life-Like Quadrupeds and Reptiles. In *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*. 3

[Koj19] KOJIMA PRODUCTIONS: Death Stranding, 2019. 2

[LGS*11] LENTINE M., GRÉTARSSONGR O., SCHROEDER C., ROBINSON-MOSHER A., FEDKIW R., GRÉTARSSON J. T., SCHROEDER C., ROBINSON-MOSHER A., FEDKIW R.: Creature Control in a Fluid Environment. *IEEE Transactions on Visualization and Computer Graphics 17*, 05 (may 2011), 682–693. 3

[LH18] LIU L., HODGINS J.: Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Trans. Graph 37* (2018), 14. 3

[LS99] LEE J., SHIN S. Y.: A Hierarchical Approach to Interactive Motion Editing for Human-like Figures. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* (1999), p. 39–48. 3

[LYV*10] LIU L., YIN K., VAN DE PANNE M., SHAO T., XU W.: Sampling-based contact-rich motion control. *ACM Transactions on Graphics (TOG)* (jul 2010). 2

[MAA*09] MITAKE H., ASANO K., AOKI T., MARC S., SATO M., HASEGAWA S.: Physics-driven Multi Dimensional Keyframe Animation for Artist-directable Interactive Character. *Computer Graphics Forum 28*, 2 (2009). 3

[MHL*21] MOUROT L., HOYET L., LE CLERC F., SCHNITZLER F., HELLIER P., HELLIER P. A., MOUROT L., HOYET L., LE CLERC F., SCHNITZLER F., HELLIER P.: A Survey on Deep Learning for Skeleton-Based Human Animation. *Computer Graphics Forum 41*, 1 (nov 2021), 1–32. 3

[MMLG*19] MARTÍN-MARTÍN R., LEE M. A., GARDNER R., SAVARESE S., BOHG J., GARG A.: Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks. *IROS 2019* (2019). 3

[NF02] NEFF M., FIUME E.: Modeling Tension and Relaxation for Computer Animation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA* (2002), 81. 2, 4, 5

[PALvdP18] PENG X. B., ABBEEL P., LEVINE S., VAN DE PANNE M.: DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Transactions on Graphics 37*, 4 (apr 2018), 18. 3

[PARC21] PALIARD C., ALVARADO E., ROHMER D., CANI M.-P.: Soft Walks: Real-Time, Two-Ways Interaction between a Character and Loose Grounds. In *Eurographics (short papers)* (2021). 3

[PCZ*20] PENG X. B., COUMANS E., ZHANG T., LEE T.-W., TAN J., LEVINE S.: Learning Agile Robotic Locomotion Skills by Imitating Animals. 3

[RH91] RAIBERT M. H., HODGINS J. K.: Animation of dynamic legged locomotion. *ACM SIGGRAPH Computer Graphics 25*, 4 (jul 1991), 349–358. 2

[Roc18] ROCKSTAR STUDIOS: Red Dead Redemption 2, 2018. 2

[RRSK12] RAHGOSHAY C., RABBANI A., SINGH K., KRY P. G.: Inverse Kinodynamics: Editing and Constraining Kinematic Approximations of Dynamic Motion. In *Proceedings of Graphics Interface 2012* (2012), p. 185–192. 3

[SP17] SHAHABPOOR E., PAVIC A.: Measurement of Walking Ground Reactions in Real-Life Environments: A Systematic Review of Techniques and Technologies. *Sensors 17*, 9 (2017). 2

[TET12] TASSA Y., EREZ T., TODOROV E.: Synthesis and stabilization of complex behaviors through online trajectory optimization. *IEEE International Conference on Intelligent Robots and Systems* (2012), 4906–4913. 2

[TLT11] TAN J., LIU K., TURK G.: Stable Proportional-Derivative Controllers. *IEEE Computer Graphics and Applications 31*, 4 (2011), 34–44. 2

[TMT14] TASSA Y., MANSARD N., TODOROV E.: Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 1168–1175. 2

[WFH10] WANG J. M., FLEET D. J., HERTZMANN A.: Optimizing walking controllers for uncertain inputs and environments. *ACM SIGGRAPH 2010 Papers, SIGGRAPH 2010* (2010). 2

[WGSF20] WANG T., GUO Y., SHUGRINA M., FIDLER S.: UniCon: Universal Neural Controller For Physics-based Character Motion. 3

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *ACM SIGGRAPH* (1988), vol. 22. 3

[YKL21] YANG D., KIM D., LEE S.-H.: LoBSTr Real-time Lower-body Pose Prediction from Sparse Upper-body Tracking Signals. *Computer Graphics Forum 40* (June 2021), 265–275. 3

[YLS04] YANG P.-F., LASZLO J., SINGH K.: *Layered Dynamic Control for Interactive Character Swimming*. Tech. rep., 2004. 3

[YLv07] YIN K., LOKEN K., VAN DE PANNE M.: SIMBICON: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3 (jul 2007), 105. 2

[YY20] YIN Z., YIN K.: Linear Time Stable PD Controllers for Physics-based Character Animation. *Computer Graphics Forum 39*, 8 (Dec. 2020), 191–200. 2

[ZH99] ZORDAN V. B., HODGINS J. K.: *Tracking and Modifying Upper-Body Human Motion Data with Dynamic Simulation*. Tech. rep., Georgia Institute of Technology, 1999. 3

[ZH02] ZORDAN V. B., HODGINS J. K.: Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), pp. 89–96. 3

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.

Rigid-body	***Mass (kg)***	***Center of Mass (x,y,z) (m)***		
Hips&Spine	10.56	0.00	0.00	0.00
Chest	25.2	0.00	0.22	-0.03
Upper Chest	10	0.00	0.35	-0.04
Neck&Head	4.8	0.00	0.60	0.00
Shoulder	1	$\pm$ 0.06	0.44	-0.04
Upper Arm	2.95	$\pm$ 0.19	0.42	-0.03
Forearm	1.59	$\pm$ 0.23	0.15	-0.02
Hand	0.5	$\pm$ 0.26	-0.13	-0.02

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.

Asset	***Mass (kg)***	k_p	k_d
Sunflower (Large)	1.25	20	10
Sunflower (Small)	1	20	10
Bush	0.2	200	10
Banana Tree	10	50	1
Tree Branch	3	500	15
Fence	10	100	10
Fence w/ Door	10	100	20
Hanging Bucket	1	-	-
Swing	2	-	-
Hanging Fruit	0.5	-	-

Table 3: *Obstacles used in our natural environment.*

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