Flexible Motion Optimization with Modulated Assistive Forces

NAM HEE KIM†, HUNG YU LING†, ZHAOMING XIE, and MICHIEL VAN DE PANNE,
University of British Columbia, Canada

Fig. 1. Our method takes sparse and crudely specified keyframe sequences as motion sketches, and outputs physically plausible motions. Dynamic movements, contact-rich behaviors, and motion timing emerge from the computed solutions.

Animated motions should be simple to direct while also being plausible. We present a flexible keyframe-based character animation system that generates plausible simulated motions for both physically-feasible and physically-infeasible motion specifications. We introduce a novel control parameterization, optimizing over internal actions, external assistive-force modulation, and keyframe timing. Our method allows for emergent behaviors between keyframes, does not require advance knowledge of contacts or exact motion timing, supports the creation of physically impossible motions, and allows for near-interactive motion creation. The use of a shooting method allows for the use of any black-box simulator. We present results for a variety of 2D and 3D characters and motions, using sparse and dense keyframes. We compare our control parameterization scheme against other possible approaches for incorporating external assistive forces.

CCS Concepts: • **Computing methodologies → Physical simulation**.

Additional Key Words and Phrases: Keyframing, Physics-Based Character Animation, Rigid Bodies

ACM Reference Format:
Nam Hee Kim, Hung Yu Ling, Zhaoming Xie, and Michiel van de Panne. 2021. Flexible Motion Optimization with Modulated Assistive Forces. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3 (September 2021), 25 pages. https://doi.org/10.1145/3480144

1 INTRODUCTION

How can plausible whole-body animated motions be created using a simple interface? This basic question underlies a large body of work in character animation. Spacetime constraint methods [Witkin and Kass 1988] are a promising approach to this, based on trajectory optimization.

*Equal contribution.

Authors' address: Nam Hee Kim, nhgk@cs.ubc.ca; Hung Yu Ling, hyuling@cs.ubc.ca; Zhaoming Xie, zxie47@cs.ubc.ca; Michiel van de Panne, van@cs.ubc.ca,
University of British Columbia, Vancouver, Canada.

2577-6193/2021/9-ART $15.00
https://doi.org/10.1145/3480144

Given a set of input keyframes, including timing, the goal is to find a trajectory that satisfies the keyframe constraints as well as the constraints of physics. There remain many challenges, however. Collocation-based methods for trajectory optimization typically require advance knowledge of the contact locations and timing, which are exactly the kind of detail that require significant iterative effort by animators. Alternatively, solutions based on forward simulations, i.e., shooting methods, allow contacts to naturally emerge during the movement. However, these suffer from being constrained to the space of physically-feasible motions. This is thus incompatible with the idea that animators often want to create physically-impossible movements, and should not be burdened with trying to determine the physical feasibility of a set of keyframes.

In this paper, we develop a robust and flexible trajectory optimization method through the introduction of auxiliary assistive forces, whose use is modulated and minimized over time through regularization. Intuitively, auxiliary assistive forces help to smooth the optimization landscape, thereby facilitating more efficient generation of natural motions. We use a shooting-based solver, realized with a derivative-free optimization method. Taken together, these choices allow for a method that is flexible in multiple respects: motions can be physically realizable or not; emergent behavior can be realized in between keyframes where feasible; motions can be specified using sparse or dense keyframes; keyframe timing can be optimized; black-box simulators can be exploited; the system produces predictable results via the keyframe-based objective; and the system allows for near-interactive motion design.

Our contributions are as follows:

- We introduce a skeleton-driven character animation framework that allows crude keyframe input to be transformed into plausible simulation-driven movements. It is particularly capable in its support of: (a) physically-impossible motions; (b) optimization of keyframe timing; (c) emergent contact behavior; and (d) emergent behaviors in between sparse keyframes.
- We introduce a novel and compact parameterization of external assistive forces, consisting of a time-indexed scalar modulation of the set of keypoint forces. These forces are defined by a set of linear spring-and-dampers that pull a character towards a target pose that is a simple linear interpolation of the keyframes.

2 RELATED WORK

The generation of physics-based movement is commonly cast as an optimization problem, involving the known physics, the applied control actions, and an objective function for the motion. One possible solution involves computing a controller or control policy, which computes the control actions as a function of the state and therefore can dynamically react to changes in the environment. An alternative approach is to perform trajectory optimization, which computes the best motion trajectory forward from a fixed initial state, typically for some fixed time horizon. While we review related work for both of these approaches, we emphasize trajectory optimization methods as our work belongs to this latter category.

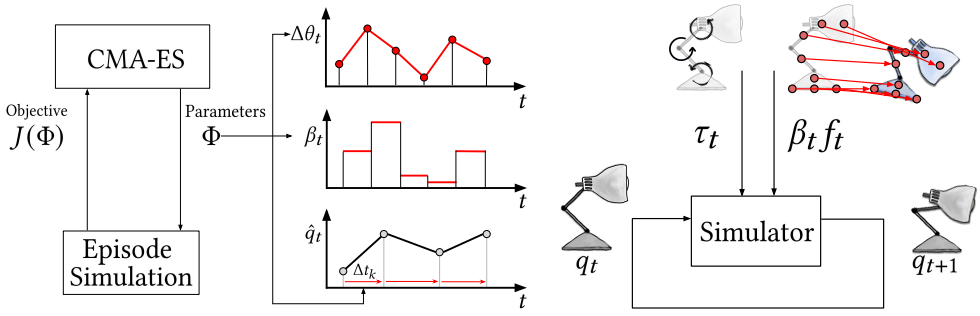
Controllers. Controllers for physics-based motions are often authored with the help of motion capture data, which can be treated as an imitation objective. This has a long history in many reference-motion tracking controllers, e.g., [Ding et al. 2015; Faloutsos et al. 2001; Ha et al. 2012; Yin et al. 2008; Zordan et al. 2014]. Recent work has seen a significant adoption in deep reinforcement learning (RL) based approaches. Learned control policies can either be specific to a given reference motion [Peng et al. 2018, 2017] or be conditioned on a near-future window of the upcoming reference motion [Bergamin et al. 2019; Chentanez et al. 2018; Park et al. 2019; Won et al. 2020]. Motion control trained on motion capture data can also be abstracted into learned latent variable representations which can then be reused in new ways even in the absence of an input reference

motion, e.g., [Merel et al. 2020]. There is a long history of reference motion tracking controllers in the literature. Human coaching hints can also be leveraged [Ha and Liu 2014]. Training deep RL controllers to produce high-quality motions based on non-imitation objectives takes extra care and attention, e.g. using a curriculum that may use external assistance during training, e.g. [Yu et al. 2018]. It is also typically a slow off-line process.

Assisted Control. The use of moderate external assistance has been explored in the design of motion-tracking controllers, in ways which do not necessitate offline learning. World-space joint control, can be combined with moderate external torque control of the root link [Wrotek et al. 2006; Zordan and Hodgins 2002] (or the feet [da Silva et al. 2017]), and is shown to provide reasonable adaptations to terrain changes during locomotion. A related approach is demonstrated for use in film production [Chapman et al. 2020]. The general approach of allowing for a degree of external assistance is made more adaptable in [Levine and Popović 2012], where a quadratic programming problem is solved at every timestep in order to allow for better response and local anticipation of the next frame of the reference motion. These methods are strictly reactive in nature, with a focus on remaining balanced, and therefore they are not directly applicable to many of the motions that we seek to generate. Robotics controllers can also be designed to leverage external assistance as needed [Maekawa et al. 2018]. The perceptual impact of physics violations has been explored in some detail for animation, e.g., [Bai et al. 2016; Barzel et al. 1996; Reitsma and Pollard 2003].

Trajectory Optimization. Since the original work on spacetime constraints [Witkin and Kass 1988], a wide variety of methods have been proposed towards making trajectory optimization approaches more flexible and practical. The complexities to be overcome are manifold: handling of contacts and contact timing, access to equations of motion vs. black-box simulation, scalability to full-body dynamics; complex optimization landscapes, and fragile behavior in the face of ill-posed or infeasible problems. Multiple efforts introduce the use of simplified models to produce a better-conditioned, simpler problem, e.g., [Bai et al. 2016; Jain et al. 2009; Kry et al. 2012; Kwon et al. 2020; Liu and Popović 2002; Liu et al. 1994; Mordatch et al. 2012]. This is often based on a centroidal dynamics model, where the movement can be well-characterized at any point in time by the center of mass position, an overall orientation, and the total linear and angular momentum. Contact timings are often assumed to be known in advance, are optimized as a separate phase, or are introduced with a motion-phase structure. Simplified models can also be leveraged in editing motion capture data via momentum and force edits [Sok et al. 2010]. This leverages a trajectory optimization technique based on normalized dynamics and knowledge of contacts and inverse dynamics. A related model is the PhysIK system [Rabbani and Kry 2016], which allows for physically-plausible keyframing by extending inverse kinematics-based keyframes to include center-of-mass and intertia handles in a 2D setting. The keyframes fully specify the contact configurations and there is no optimization involved.

Trajectory optimization methods that have access to the equations of motion and derivatives have the opportunity to be uniquely efficient, although they need to circumvent the challenges posed by contact discontinuities. One approach is to have a derivative-free outer loop optimization that defines the contact modes, wrapped around an inner-loop trajectory optimization using the now-known contact structure [Wampler and Popović 2009]. An alternate treatment of the hybrid contact optimization problem uses sequential quadratic programming and includes contact forces as optimization parameters [Posa et al. 2014]. Intermediate optimization phases that relax the constraints are found to be important for avoiding problematic local minima. Trajectory optimization through contact discontinuities can be tackled using numerical differentiation and smoothed contact dynamics, with further measures taken to achieve greater computational efficiency [Han et al. 2016]. This method can adjust footstep contacts and timings, and is demonstrated in the



(a) System Overview (b) Simulation Step

Fig. 2. (a) Visualization of our optimization pipeline, the optimization parameters Φ, and their interpolation. They correspond to the per-timestep residual PD target $\Delta\theta_t$, assistive force modulation β_t, and the keyframe timings, Δt_k. (b) Internal joint actuation and modulated external assistive forces are used in combination.

context of tracking existing reference motions. Short-horizon trajectory optimization problems are shown to be solved at near-interactive rates using an iterative linear quadratic Gaussian (iLQG) method [Tassa et al. 2012], together with finite-difference derivatives, suitable contact approximations, and update regularization. Solutions to trajectory optimization for multi-contact settings continue to improve with dedicated packages such as Crocoddyl [Mastalli et al. 2020] for humanoid robotics.

Shooting methods offer an alternative approach to collocation-based trajectory optimization. They can leverage black-box simulations and can avoid many of the issues related to discontinuities, as long as they can be simulated. External forces were used in conjunction with shooting methods over 25 years ago, limited to basic walk cycles [van de Panne and Lamouret 1995]. Sample-based solutions for contact-rich motions have been shown to be a feasible solution for motion-capture imitation tasks [Liu et al. 2010] and online short-horizon trajectory optimization solutions for tasks defined by locomotion and balance objectives [Hämäläinen et al. 2014; Rajamäki and Hämäläinen 2018]. Derivative-free optimization methods such as covariance-matrix adaptation (CMA) have been used offline to compute optimized motion trajectories for a variety of highly dynamic motions, e.g., [Al Borno et al. 2012, 2017]. It is further possible to optimize for a diverse range of motions for given tasks [Agrawal et al. 2013]. Physically impossible motions are generally outside of the scope of the optimization methods described above. Our work considers keyframe-based motions that may be physically-infeasible, as enabled by the proposed assistive force modulation.

3 METHOD

The core components of our proposed method are: (1) the use of external assist as realized via forces, modulated over time, that pull a character towards its corresponding keyframe-interpolated pose; (2) a framework that simultaneously optimizes actions, the external assist modulations, and keyframe timing; and (3) a cost function that regularizes keyframe errors, internal actions, external assist used, and keyframe timing. Figure 2a provides a high-level overview of our system, including the simulation, the optimization, and the free parameters of the optimization. Figure 2b shows how the external forces are integrated into the simulation.

3.1 Notation

We describe our method in a 2D setting to keep the notation compact. However, our system implements the straightforward 3D generalization of the method. We begin with a rigid-body character equipped with J joints. In 2D settings, J is equal to the size of the character's action space, since all joints must only rotate along a single axis. To animate this character, the user defines a sequence of $K+1$ keyframes[1] $\hat{\boldsymbol{q}} = \{\hat{q}^0, \hat{q}^1, \cdots, \hat{q}^K\}$ that roughly sketches the desired *character poses* over time. A pose $q \in \mathcal{Q} = \mathbb{R}^{J+2+1}$ consists of the global position $x \in \mathbb{R}^2$, orientation $\psi \in [-\pi, \pi)$, and joint angles $\theta \in \mathbb{R}^J$. Note that joint velocities are not included in the keyframe definition.

The task of our system is to compute a trajectory $\boldsymbol{q} = \{q_0, q_1, \cdots, q_T\}$ over the horizon of $T+1$ timesteps, such that the sequence of input keyframes $\hat{\boldsymbol{q}}$ is realized over the course of the trajectory. The timing of each keyframe can be specified in advance or optimized as a free parameter, as we discuss later. For example, one can specify for keyframe $\hat{q}^k$ to be realized at timestep t_k, such that the output trajectory satisfies $q_{t_k} = \hat{q}^k$. We assume that keyframes also describe the desired start and end states, meaning $q_0 = \hat{q}^0$ and $q_T = \hat{q}^K$.

A linearly-interpolated trajectory $\hat{\boldsymbol{q}}_{0,1,\ldots,T} = \{\hat{q}_0, \hat{q}_1, \cdots, \hat{q}_T\}$ given specified keyframes will have $\hat{q}_{t_k} = \hat{q}^k$ as required and

$$\hat{q}_t = \frac{(t_{k+1} - t)\hat{q}_{t_k} + (t - t_k)\hat{q}_{t_{k+1}}}{t_{k+1} - t_k} \tag{1}$$

for $t_k < t < t_{k+1}$. Figure 2a illustrates this interpolation strategy.

To make the motion look physically plausible, an animator often needs to spend significant effort designing a dense set of keyframes carefully. This motivates the use of trajectory optimization to generate a suitable trajectory:

$$\begin{aligned} \text{minimize } & J(Q, U) = \sum_{t=0}^{T} l(q_t, u_t) \\ \text{subject to } & q_{t+1}, \dot{q}_{t+1} = f(q_t, u_t, \dot{q}_t) \\ & q_{t_k} = q^k \text{ for } k \in 0, 1, \cdots, K \\ & \dot{q}_0 = 0 \end{aligned}$$

where $u \in \mathbb{U}$ is the actuation from the set of actuators attached on the joints of the characters, $l : \mathcal{Q} \times U \to \mathbb{R}$ is a user defined cost function and $f : \mathcal{Q} \times T\mathcal{Q} \times U \to \mathcal{Q} \times T\mathcal{Q}$ describes the physical laws the characters will follow to evolve the state. Keyframe timing can also be optimized by treating t_k as free variables.

3.2 Internal Action Parameterization

The internal actions, $\Delta\theta_t$, form a key part of the optimization, and consists of values for each joint, sampled at regular time intervals, δ_t, for a total of n_a values per joint. n_a and δ_t are parameters that the user can adjust depending on the length of the input keyframes and the desired granularity of the motion. We assume that the total number of internal actions is more than that of horizon length $T+1$, otherwise the optimized motion may not cover the entirety of the input keyframes. In our experiments, we fix δ_t to 20, as we find higher frequency actions not needed for our keyframe scenarios. n_a is set to be larger than needed, as unused internal actions are ignored when the last keyframe checkpoint has been reached. Please refer to Table 1 for details on how n_a is set for different motions.

[1]We use $K+1$ instead of K to simplify notation. Similarly, our horizon is $T+1$ steps instead of T steps.

The internal actions are then linearly interpolated to yield a continuous function $\Delta\theta(t)$, as shown in Figure 2a. Importantly, the number of free parameters with respect to internal actions is independent of keyframe timings. This allows both actions and timings to be optimized simultaneously without an outer optimization loop. We use proportional-derivative (PD) control to actuate the character. The actions, $\Delta\theta_t \in [-\pi, \pi)^J$ then specify residual PD target angles. At each timestep t, torques are computed as

$$\tau_t = k_p^A(\hat{\theta}_t + \Delta\theta_t - \theta_t) - k_d^A\dot{\theta}_t, \tag{2}$$

where k_p^A and k_d^A are joint PD gains for the PD controller and θ and $\dot{\theta}$ are the joint angles and joint velocities of the simulated character.

3.3 Modulated Assistive Forces

External assistive forces pull a character towards it corresponding the target pose as defined by a simple linear interpolation of the keyframes over time. This is achieved by pulling each character *keypoint* towards its counterpart on the target pose, as shown in Figure 2b. The forces are computed using a linear spring-damper model. Using the assistive forces, a character can be animated as a ragdoll, loosely tracking the linearly interpolated reference trajectory.

More formally, we use a set of N keypoints position on the links of the articulated character, $[p_0, p_1, \cdots, p_{N-1}]$ where $p_i \in \mathbb{R}^2$. These can be computed from a pose q via forward kinematics. The external force $f_{i,t} \in \mathbb{R}^2$ for keypoint i at timestep t is computed as a result of simulating a spring-damper system based on the distance between reference keypoint position $\hat{p}_{i,t} \in \mathbb{R}^2$ and current keypoint position $p_{i,t} \in \mathbb{R}^2$. Finally, this force is modulated by a scalar, β_t:

$$f_{i,t} = \beta_t \left(k_p^L(\hat{p}_{i,t} - p_{i,t}) - k_d^L\dot{p}_{i,t}\right), \tag{3}$$

where k_p^L and k_d^L are linear spring and damper stiffness coefficients respectively, and β_t is a modulation parameter, which we discuss below. The k_p and k_d constants are set to be proportional to the mass of the link that the keypoint belongs to. During each timestep, we compute the spring-damper force for each keypoint of the character and apply all of the external forces $f_t \in \mathbb{R}^{2N}$ together, along with joint torques $\tau_t \in \mathbb{R}^J$ of the character. We note that while a single 6-DoF linear-plus-angular springs-and-dampers could also be used to bring two links into correspondence, this has the extra complexity of needing to introduce another hyperparameter to balance linear and angular forces. Our set of linear keypoint springs-and-dampers achieves the same objective in a simpler fashion. Furthermore, the optimization process has some flexibility with respect to the placement of the keypoints. For all our results, the keypoints are placed at the center of mass of each link, as a simple default placement. In order to understand the impact of keypoint placement, we also experimented with as few as two keypoints fox Luxo2D, at the head and the foot. The optimization procedure generates similar solution modes and comparable motion quality as the default keypoint placements for most scenarios.

The scalar $\beta_t \in [0, 1]$, defines the *assistive force coefficient* that modulates the magnitude of the assistive forces as a form of action. Similar to actions, they are anchored in time and we optimize for $\boldsymbol{\beta} = [\beta_0, \beta_{\delta_t}, \cdots, \beta_{(n_a-1)\delta_t}]$, where β_t is piecewise constant for each corresponding time interval, δ_t, as shown in Figure 2a.

3.4 Keyframe Timing

In our framework, keyframe timing can either be defined by the animator or optimized as a free parameter. Our method automatically finds the most suitable keyframe timing for a given interval if it is not specified. More formally, we introduce variables $\boldsymbol{\Delta t} = [\Delta t_1, \Delta t_2, \cdots, \Delta t_K]$ where we refer

to each entry Δt_k as the *time-to-arrival*. These time-to-arrival values represent the time between the k–1^{th} and k^{th} keyframes, i.e. $t_k = t_{k-1} + \Delta t_k$. The first keyframe timing is defined to be $t_0 = 0$ and we set the character's initial pose to match the first keyframe. By default, the user can provide lower and upper bounds for Δt_k to limit the search space. When designing more delicate keyframe motions, the user can optionally provide a precise time-to-arrival value for each individual keyframe interval. In contrast to spacetime constraints methods, keyframe timing is used to formulate an objective minimization problem in our system, as opposed to serving as hard constraints.

3.5 Objective Function

Our trajectory optimization problem treats the keyframes as *soft spacetime constraints*. We use weighted keyframe errors to penalize trajectories that deviate from the input keyframes $\hat{\boldsymbol{q}}$. We also introduce penalty terms associated with parameters to achieve the following desiderata: excessive assists should be avoided to prevent completely non-physical motions; character actions should have moderate magnitude lest the resulting action might be unnatural; and, finally, excessively long-duration keyframe spacing should be avoided.

More formally, we associate weight $w_k \in [0, 1]$ for every keyframe such that the importance of each keyframe can be taken into account. For example, the user may wish to set the first keyframe's weight w_0 to 0, given that $q_0 = \hat{q}^0$ is always true since we initialize our character at the first keyframe. Our free variables for the optimization are $\Phi = \{\boldsymbol{\Delta\theta}, \boldsymbol{\beta}, \boldsymbol{\Delta t}\}$ and the final cost function expressing the aforementioned desiderata is written as follows:

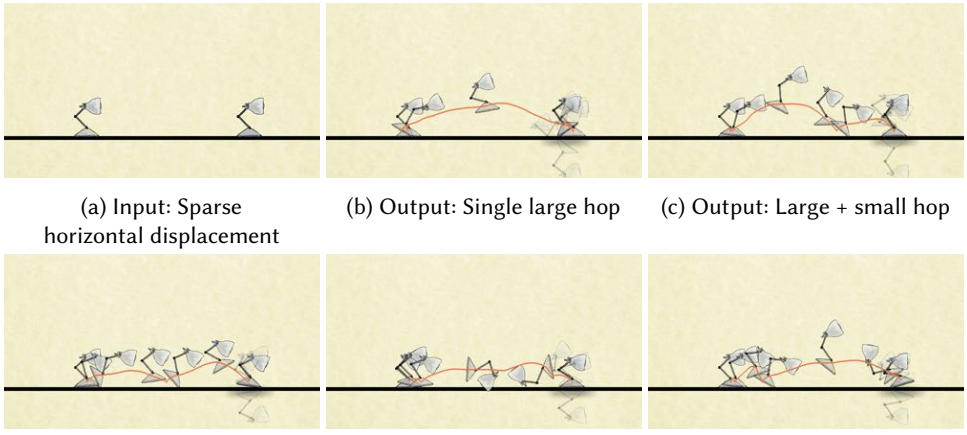
$$\begin{aligned} J(\Phi) &= J(\boldsymbol{\Delta\theta}, \boldsymbol{\beta}, \boldsymbol{\Delta t}) \\ &= \underbrace{\sum_{k=0}^{K} w_k ||\hat{q}^k - q_{t_k}||^2}_{E_{\text{KF}}} + \underbrace{\lambda_\beta \sum_{t=0}^{T} ||\beta_t||^2}_{E_{\text{assist}}} + \underbrace{\lambda_{\Delta\theta} \sum_{t=0}^{T} ||\Delta\theta_t||^2}_{E_{\text{internal}}} + \underbrace{\lambda_{\Delta t} \sum_{k=1}^{K} ||\Delta t_k||^2}_{E_{\text{timing}}} \end{aligned} \tag{4}$$

E_{KF} is the keyframe error. E_{assist} is a regularizer for external assist. The E_{internal} term regularizes large actions so as to discourage rapid change. Finally, E_{timing} regularizes the keyframe timing. Note that keyframe timing affects the objective function in multiple ways. Aside from the E_{timing} term, it also impacts the timing of the keyframe targets $\hat{\boldsymbol{q}}$. λ_β, $\lambda_{\Delta\theta}$, and $\lambda_{\Delta t}$ are non-negative penalty weights corresponding to the regularization terms.

3.6 Optimization

We use a derivative-free continuous optimization method, covariance matrix adaptation (CMA-ES). To facilitate a more interactive motion design process, we further introduce a horizon-based curriculum. This allows for immediate display of initial results, with updates being shown as the optimization proceeds. In this curriculum, the trajectory optimization is performed in stages with growing number of keyframes. The first stage consists of the initial and the second keyframe, i.e. the first keyframe interval. The physics simulation is terminated when the timing corresponding to the second keyframe has been reached. At this point, the first optimized motion segment is ready to be displayed to the user.

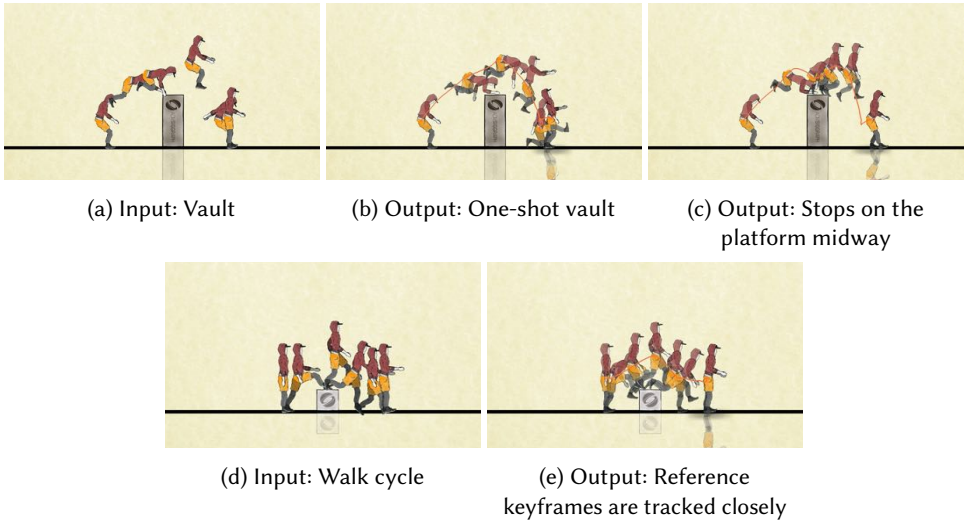
The curriculum advances to the next stage when the CMA-ES variance drops below a predefined threshold. The threshold is a hyperparameter which we set to be one-fourth of the initial variance in all of our experiments. Intuitively, after the variance converges to a small value, it is unlikely to discover a significantly better motion in the current optimization round. In the new curriculum stage, we set the optimization parameters associated with the previous stages to have a lower



(a) Input: Sparse horizontal displacement (b) Output: Single large hop (c) Output: Large + small hop

(d) Output: Two medium hops (e) Output: Forward roll (f) Output: Small + large hop

Fig. 3. An example of generating diverse motions from a single set of sparse horizontal displacement keyframes.



(a) Input: Vault (b) Output: One-shot vault (c) Output: Stops on the platform midway

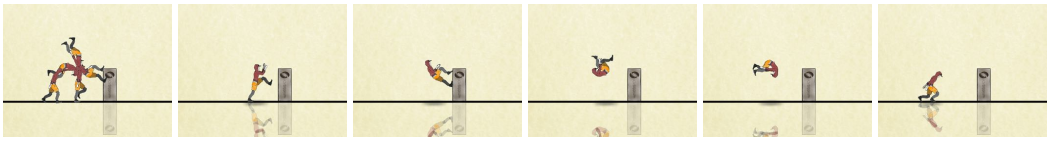
(d) Input: Walk cycle (e) Output: Reference keyframes are tracked closely

Fig. 4. Folk2D walk cycle and vault motions specified with dense keyframes.

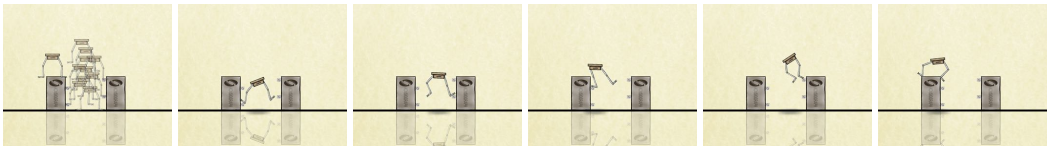
variance, so that the optimizer builds on top of the existing solution. The entire optimization process is completed when the final keyframe is covered by the curriculum.

4 RESULTS

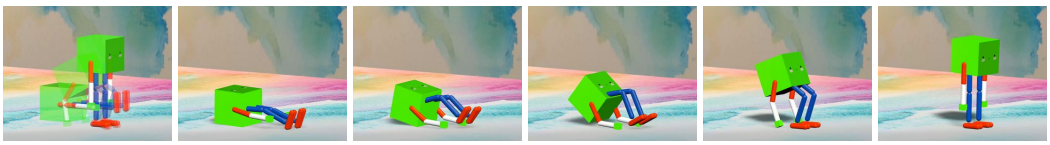
We present a large variety of examples in order to demonstrate emergent behaviors, physical and infeasible motions, and contact-rich movements. The supplementary video provides the best way to view the results. Additional details are given in the appendices for hyperparameters (A), user interface (B), the character models (C), implementation details (D), the horizon curriculum (E),



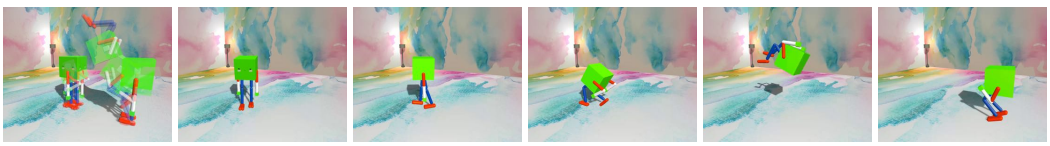
(a) Folk2D Wall Flip



(b) Desk2D Wall Jump



(c) Boxy Supine Getup



(d) Boxy Rotate Back Flip

(e) Boxy Sit Down on SIGGRAPH Letters

Fig. 5. Our method can generate highly dynamic motions and 3D motions. The leftmost column shows the keyframes for each motion. The optimized trajectories are shown from left to right in each row.

sensitivity to CMA-ES population size (F), assist-vs-tracking regularization (G), and a visualization of the assist modulation over time (H).

4.1 Emergent Behaviors

Given crude and sparse keyframes, our trajectory optimization method is capable of generating compatible motions. Since sparse keyframes may specify an under-constrained optimization problem, the optimized solutions can be imaginative and diverse. In many cases, these motions contain complex contact patterns, which can be difficult to find for existing keyframe-based methods without meticulous keyframe design.

In Figure 3, the input is a set of sparse horizontal displacement keyframes for Luxo2D. The result of optimization is a diverse set of emergent and natural behaviors. Additional variations can be synthesized by adjusting the time-to-arrival and assist penalty parameters. These motions are shown in Figure 3d-3f. A smaller time-to-arrival generally promotes faster motions and a larger assist penalty encourages more natural motions. The forward roll strategy in Figure 3e

demonstrates the flexibility of our method in handling complex contact patterns. Figure 4a-4c show another example of emergent behavior. In the second variation, Folk discovers the strategy of using the vaulting platform as a intermediate foothold, in order to perform a more balanced landing.

Our method also extends to dense-keyframe setups. When necessary, this allows artists to provide additional keyframes to precisely control the character behavior during particular motion segments. Figure 4 shows a six-keyframe walk cycle for the Folk character over a medium sized block. Through this example, we note that our method is capable of generating trajectories which track the carefully designed input keyframe sequences.

4.2 Impossible Motions

Artists often require animated characters to perform motions exaggerated beyond the limitations of physical laws. This is particularly challenging for existing optimization methods which are often designed to synthesize physically feasible motions. In contrast, in addition to physically feasible motions, our method is able to generate trajectories that are not entirely physical. Although these *impossible* motions can only be performed under the assistance of external forces, the overall motion quality remains reasonable due to the regularization effecct of the assist penalty in the objective function. Figure 7 and Figure 8 show Luxo2D jumping onto platforms beyond their physical capabilities. For physically feasible motions, we observe that our framework generally produces motions that are not overly dependent on the external assistive forces. In some cases, the optimized trajectories are not impacted at all when we disable the assistive forces at run-time. Section 5 discusses the difference between impossible and physically feasible motions in detail.

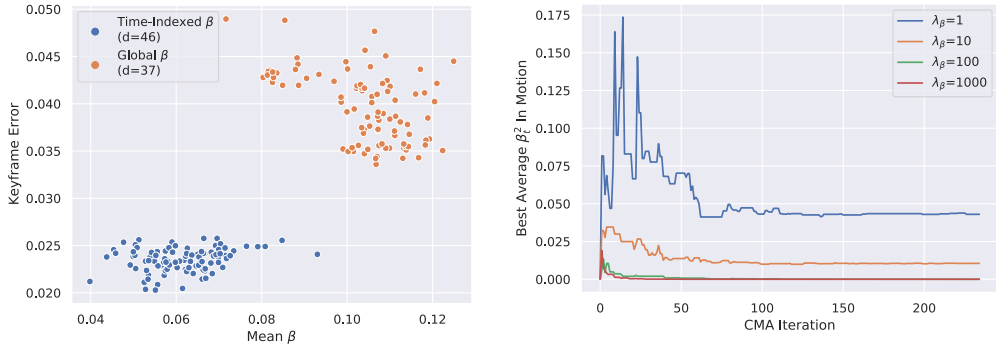
4.3 Contact-Rich Movements

We are able to generate highly dynamic contact-rich motions using our framework. We synthesize various wall flip and wall jump trajectories using sparse keyframes. Figure 5a and 5b are two examples of complex movements. Note that in Figure 5a, the second keyframe intersects the wall. Despite this, the output trajectory is a clean collision as a result of the physics simulation. Figure 5b shows an example where the Desk2D character is able to use the mini-footholds to escape the confinement of the walls. These motions typically require meticulous fine-tuning in traditional kinematic keyframe animations. In contrast, our method seamlessly integrates physical interactions with the environment into the optimization process. Our method also compares favorably against other methods that learn physically-based controllers. For example, reward-based reinforcement learning is likely to produce unnatural motions or to fail entirely in the absence of significant reward engineering.

4.4 3D Motions

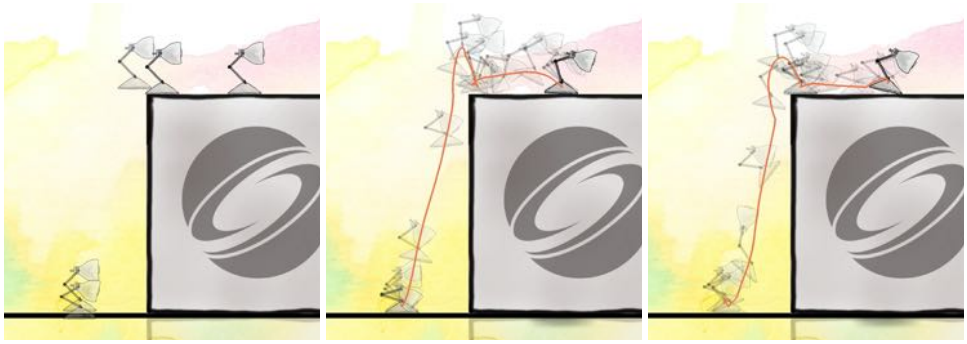
We also synthesize multiple motions for 3D characters. Even with relatively few keyframes, we can generate various getting-up, sitting-down, and flipping motions for Boxy, as shown in Figure 5. These motions contain an additional yaw-rotation parameter as part of the keyframe specification, which allows the character to navigate in the xy-plane. Note that this is not the same as fully-parameterized 3D motions, which would allow roll-rotations along with pitch and yaw. Please refer to the supplementary video for visual demonstrations of additional motions.

While our system can generate 3D motions, it has a number of limitations. Firstly, it takes substantially longer to generate 3D motions compared to 2D motions. This is problematic especially when interactive use of the system is desired. In 3D, the search space for the optimization problem grows as additional degrees of freedom are introduced. The motion trajectories that can be collected in the same amount of time represent a smaller fraction of the search space. The induced sparsity can be problematic for statistical optimization methods such as CMA, and is commonly referred to



(a) Assist Parameterization Comparison (b) Evolution of Assistive Forces

Fig. 6. (a) Comparison of performance between global β and time-indexed β. Each point represents a solution. Ideal solutions (small assist, small error) reside on the bottom-left corner. (b) Evolution of assistive force usage during optimization, as measured by the mean value of β_t^2 used throughout the motion. The optimization initially explores large values of β_t and then gradually decreases the reliance on assistive force.



(a) Keyframes (b) Tβ approach (d=47) (c) Gβ approach (d=36)

Fig. 7. Comparison of the character performing the "jump-hop" task according to different external assist parameters. This scenario involves an physically impossible jump followed by a physically-feasible hop.

as the *dimensionality curse*. We can potentially resolve this by using a smarter optimization scheme, such as dividing the optimization window into smaller segments and solve them independently. Secondly, large 3D rotations could be problematic to guide with keypoint-based linear springs, additional keypoints or different assist mechanism might be needed in such scenarios.

5 EVALUATION & DISCUSSION

We perform a number of evaluations and ablations in order to validate and understand the key elements of our approach.

5.1 Importance of a Compact Assist Parameterization

To assess the importance of our novel control parameterization scheme, we present quantitative and qualitative analyses of results comparing our approach with other baselines. We design three approaches in total:

- Time-indexed β with spring-damper system (Tβ): use piecewise constant β as described in Section 3.
- Global β with spring-damper system (Gβ): use the same spring-damper assistive forces based on linearly interpolated reference motion. However, instead of modulating the influence of external forces, apply a constant β value throughout the motion.
- Time-indexed root-force-and-torque (RFT): use external force and torque on character's root, independent of reference motion.

In what follows, we show that our Tβ approach is sufficiently expressive for scenarios that require both physically feasible and infeasible motions, which Gβ does not handle well. We also show that the structure and compactness of our Tβ approach compares favorably to the less-compact RFT parameterization .

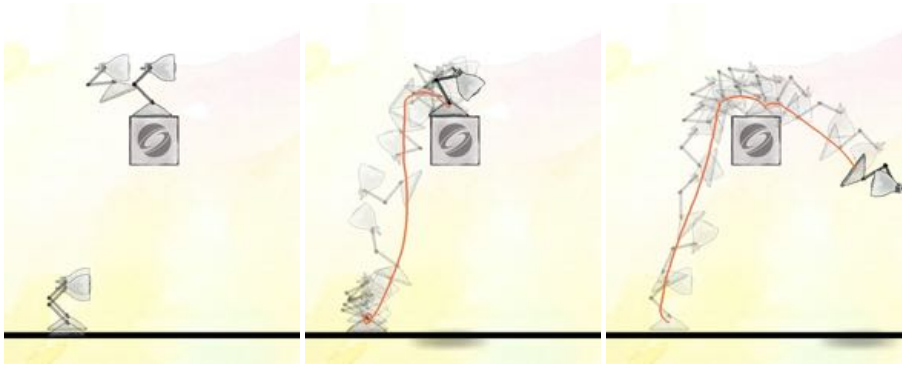
Time-Indexed β vs. Global β. To showcase the benefits of our Tβ approach, we show that our scheme can handle physically feasible and infeasible motions appearing in the same task. Intuitively, choosing a constant reliance on the assistive forces can lead to excessive use of assist in easy motions or insufficient use of assist in difficult motions. Since we aim to create a system that is agnostic to the physical feasibility of input motion sketch, the control parameters must be sufficiently expressive to handle these scenarios.

The test case depicted in Figure 7a captures an example use case where the artist specifies an impossible high-jump motion and an easy forward-hopping motion appearing together. We use our system to produce optimized motions with tuned hyper-parameters such as assist regularization weight λ_β and the range of time-to-arrival values. Once tuned, these hyper-parameters are fixed across the two parameterization schemes. For each approach, we obtain 100 independent converged solutions using the compute budget of 15,000 samples, and then we examine the resulting keyframe errors as well as the magnitude of β, which corresponds to the degree of external assist.

Figure 6a highlights how the performance of our system differs based on whether Tβ or Gβ is used. Figure 7 compares the optimized motions side-by-side. We observe that the Tβ approach yields solutions that succeed in achieving lower keyframe errors while keeping the use of external assist moderate. On the other hand, learning a global β yields either high keyframe errors due to insufficient use of assist during the difficult motion or excessive use of assist to drive the keyframe errors low. However, we note that the reduction in parameter dimensionality for the Gβ approach tends to yield high-quality actions, which generally coordinate well with the external assistive forces. As seen in Figure 7c, the character produces a plausible jumping motion for the initial high jump, although it fails to produce good actions for the later hopping motion, due to the excessive use of external assist. On the other hand, Figure 7b shows the character relying mostly on external assist for the high jump, refraining from producing internal torques, which is expected since actions are regularized. The character produces a plausible hopping motion afterwards, being able to suppress the use of external assist when not necessary.

Time-Indexed β vs. Time-Indexed Root-Force-and-Torque. Another alternative to using the Tβ scheme is to directly treat external forces as parameters to be optimized. We can introduce time-indexed forces $f_{\text{ext},t}$ and torques $\tau_{\text{ext},t}$ to be applied at the character's root link, following the same piecewise constant time discretization as β shown in Figure 2a. However, this approach greatly increases the parameter dimensionality. For the 2D case, time-indexed forces $f_{\text{ext},t} \in \mathbb{R}^2$ and torques $\tau_{\text{ext},t} \in \mathbb{R}$ requires three parameters, as opposed to one parameter associated with each time-indexed $\beta_t \in [0, 1]$, and an additional hyperparameters to regularize the external force and torque in the objective.

Given the same compute budget of 15,000 samples, the time-indexed root-force-and-torque approach struggles to perform on tasks requiring a significant amount of external assist. We use



(a) Keyframes (b) Tβ approach (d=44) (c) RFT approach (d=64)

Fig. 8. Comparison of the character performing the "high box jump" task using different external assist parameters. This scenario requires a significant amount of external assist to produce the desired motion.

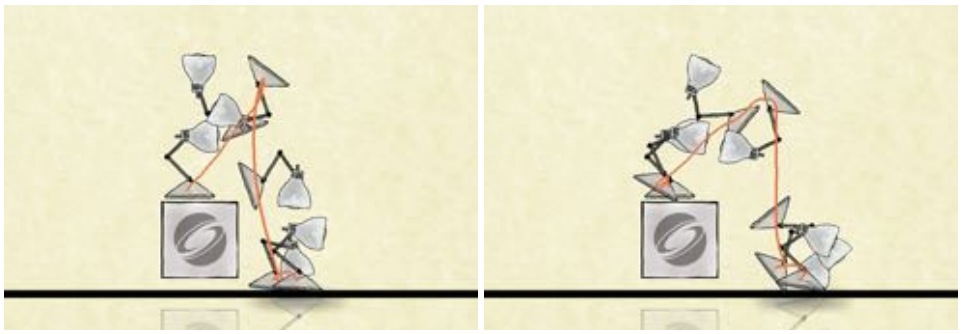
a high box jump task in Figure 8a to demonstrate this. Again, we use our system to produce optimized motions after tuning the hyper-parameters. This time, we tune the regularization weight λ_β separately to take into account the scale differences in the two parameterization schemes.

In Figure 8, we provide a comparison of one of the optimized motions. The parameter dimensionality differs by 20, making the search space for optimal control much larger for the RFT approach. Figure 8c shows an example case of a bad local minimum encountered when using the RFT approach, which happens frequently among converged solutions. On the other hand, Figure 8b shows an example of a good local minimum encountered when using the Tβ approach, yielding an emergent solution of using a somersault motion to synergize with the spring-damper forces.

5.2 Assistive Forces as Constraint Relaxation

It is clear that external assistive forces are required for the characters to perform impossible motions. However, assistive forces may also help with optimization even when the keyframes are physically feasible, due to forces in effect acting as a temporary constraint-relaxation method that may speed the optimization. In our experiments, we consistently find that using assistive forces leads to higher quality results than the alternative, given equal amount of compute budget. Figure 9 compares two trajectories of the same back flip motion optimized with and without assistive forces. In the case where high assist penalty ($\lambda_\beta = 10^3$) is used, the result is a full back flip rotation with a stable landing, even when assistive forces are disabled at run-time. In contrast, the optimization without assistive forces is incapable of performing a stable landing.

Since every optimization parameter has an initial mean value of zero, our system starts the optimization iterations assuming that the assistive force coefficients are close to zero, i.e. $\boldsymbol{\beta} \approx 0$. However, as shown in Figure 6b, the optimizer tends to explore higher values of assistive force coefficients in the beginning and leverage assistive forces for finding viable solution modes. After some iterations, the reliance on assistive force is gradually reduced and eventually converges to a fixed value depending on the cost function penalty λ_β. This exploration behavior is analogous to a learning curriculum, similar to [Yu et al. 2018], and may explain the discrepancy observed in Figure 9.



(a) Motion optimized with high assist penalty and replayed without assist

(b) Motion optimized without assist

Fig. 9. Comparison of optimized trajectories with and without assist during training. Training with assist yields a more stable trajectory, even with assistive forces disabled at run-time. Note that in (b), the trajectory performs a partial rotation and unstable landing.

5.3 Reusing Solutions

While our system incorporates optimization into the animator's workflow, offline precomputation can be used to yield real-time response to keyframe edits. For example, to make a character jump onto a box, one may procedurally generate instances of the jumping scenario by varying keyframe configurations and the box's height. For each of these instances, solutions can be pre-computed. Then, to display desired motions according to the animator's input keyframes, we can return precomputed solutions or an interpolation thereof. While time-indexed controls generally may not generalize well, any use of the assistive forces in the computed solutions will naturally act to provide stabilizing feedback around the target keyframe trajectory. To prototype this capability, we prepare a simple scenario where the Luxo character jumps onto boxes of different heights. We obtain a solution for one particular instance (box height=2.0m) and observe that the solution readily transfers to other instances (box height=[1.0m - 2.2m]) of box jump (see Figures 10).

For a more quantitative look, Figure 11 shows the deployment performance of solutions learned on different box heights by plotting final keyframe error as a function of tested box height. Instead of cumulative keyframe error, we evaluate the keyframe error associated with the final keyframe as a heuristic for whether the trajectory completes the motion as intended, while not taking intermediate keyframes into account. Intuitively, deployment performance is best when the tested box height and the original box height are equal. Solutions perform robustly when tested box heights are reasonably close to their original box heights. With multiple precomputed solutions, we can leverage the emerging best-error envelope (highlighted in magenta in Figure 11) to deploy the best available solution given a box height within range, bypassing the optimization process. This shows the promise of solution re-use, although testing these ideas on more complex motions is left as future work.

5.4 Limitations

While we have made significant progress, our work still has unsolved issues. The system requires some experience in order to be able to quickly debug motions that do not behave as intended. The ability of CMA-ES to find multiple solution modes can be seen as a feature (emergent behavior) or as a a problem (unpredictable behavior). The optimized results can be sensitive to various hyperparameters, such as the CMA-ES population size and the assistive force penalty. We have

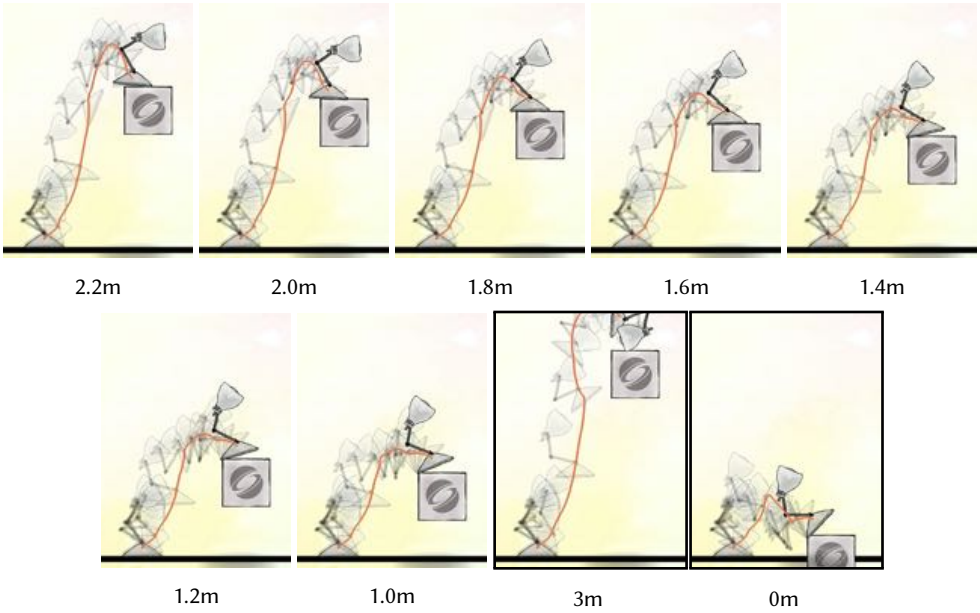


Fig. 10. The converged solution for a particular 2.0m box jump instance is successfully deployed to boxes of other heights without any additional optimization. Outlined scenarios are failure cases.

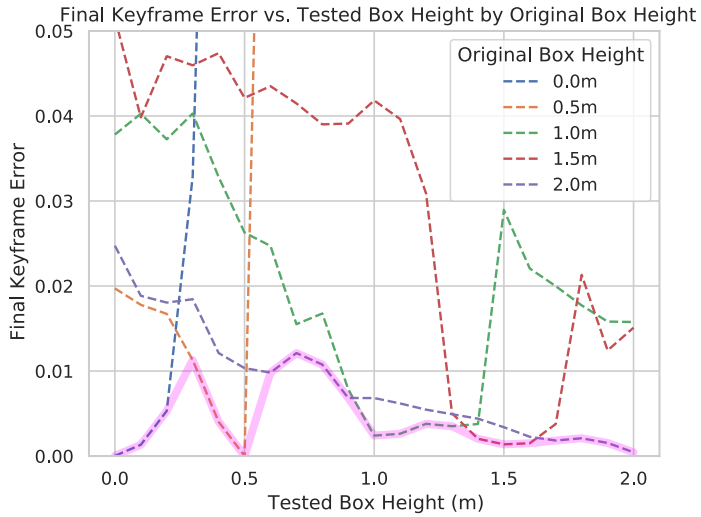


Fig. 11. Deployment performance for solutions learned on different box heights. The best-error envelope is highlighted in a magenta line.

yet to show that the method could scale to complex 3D human movements, although we note that existing full-body trajectory optimization methods have many constraints of their own, i.e., having long compute times, fixed contact timing, or do not extend to impossible motions. The current system

is not yet fully interactive. We do not yet demonstrate long duration keyframe sequences, although we expect that these could be efficiently achieved using a windowed optimization. Underdamped settings for the PD-based joint control can result in minor wobbles, visible in some of our motions.

6 CONCLUSIONS

We have presented a flexible motion design system for generating plausible simulated motions from crude keyframe inputs. Our method provides a novel combination of internal actuation, external assistive forces, and black-box simulation. We believe that the simplicity of the method will facilitate its adoption.

We apply our method to four distinct characters on a wide variety of motions. In many cases, our system produces natural motions and emergent behaviours from sparse keyframe inputs. The method effectively subsumes physics-as-a-hard-constraint and rag-doll trajectory tracking as special cases at opposite ends of the assistive spectrum. For the case of physically-feasible motions, we show that the introduction of assistive forces can act as a constraint relaxation method that enables faster optimization. The compact modulation-based assistive force parameterization is sufficiently expressive to handle both physically feasible and infeasible motions occuring in the same motion; the system is agnostic to the physical feasibility of the input keyframes.

Many avenues exist for future work. We envision partial keyframing to be a straightforward yet rewarding extension. For example, one may fix the height and pose of a keyframe but otherwise leave its location free to be optimized. More flexible motions will emerge, at the cost of a slight increase in the dimensionality of the optimization. We also envision opportunities for data-driven methods to improve performance, e.g. using a supervised model as a caching mechanism to allow for re-use and warm-starts for optimization. Meanwhile, differentiable physics (e.g. [Heiden et al. 2019]), GPU-accelerated simulators (e.g. [Monteiro et al. 2019]), and sampling and particle filter-based optimization (e.g. [Hämäläinen et al. 2014]) all show promise towards scaling optimization for more complex character morphologies and movements. Other improvements include multi-resolution optimization; understanding the benefits of higher-order interpolants, if any; and an improved user interface. Finally, by considering state-indexed control methods as an orthogonal direction, we may bridge the gap between trajectory optimization and closed-loop control for directable physics-based character animation. Success in these tasks will help combine the strengths of keyframing-driven techniques with the benefits of RL-based techniques (e.g. real-time inference and robustness to perturbations).

ACKNOWLEDGEMENTS

This work is partly funded by NSERC Canada Graduate Scholarship-Master's (CGS-M) and NSERC Discovery RGPIN-2020-05929. The authors would like to thank Jenny Boucher for contributing original artwork for rendering, Dinesh K. Pai for providing helpful comments on improving this work, and the anonymous reviewers at ACM SIGGRAPH Symposium on Computer Animation for their kind feedback.

REFERENCES

Shailen Agrawal, Shuo Shen, and Michiel van de Panne. 2013. Diverse motion variations for physics-based character animation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 37–44.

Mazen Al Borno, Martin De Lasa, and Aaron Hertzmann. 2012. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics* 19, 8 (2012), 1405–1414.

Mazen Al Borno, Michiel van de Panne, and Eugene Fiume. 2017. Domain of attraction expansion for physics-based character control. *ACM Transactions on Graphics (TOG)* 36, 2 (2017), 1–11.

Yunfei Bai, Danny M Kaufman, C Karen Liu, and Jovan Popović. 2016. Artist-directed dynamics for 2D animation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

Ronen Barzel, John R Hughes, and Daniel N Wood. 1996. Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation'96*. Springer, 183–197.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–11.

Danny Chapman, Tim Daoust, Andras Ormos, and Joseph Lewis. 2020. WeightShift: Accelerating Animation at Framestore with Physics. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation, showcase proceedings*. 1–2.

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. 1–10.

Erwin Coumans and Yunfei Bai. 2016–2020. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Danilo Borges da Silva, Rubens Fernandes Nunes, Creto Augusto Vidal, Joaquim B Cavalcante-Neto, Paul G Kry, and Victor B Zordan. 2017. Tunable robustness: An artificial contact strategy with virtual actuator control for balance. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 499–510.

Kai Ding, Libin Liu, Michiel Van de Panne, and KangKang Yin. 2015. Learning reduced-order feedback policies for motion skills. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 83–92.

Petros Faloutsos, Michiel Van de Panne, and Demetri Terzopoulos. 2001. Composable controllers for physics-based character animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 251–260.

Sehoon Ha and C Karen Liu. 2014. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–11.

Sehoon Ha, Yuting Ye, and C Karen Liu. 2012. Falling and landing motion control for character animation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–9.

Perttu Hämäläinen, Sebastian Eriksson, Esa Tanskanen, Ville Kyrki, and Jaakko Lehtinen. 2014. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

Daseong Han, Haegwang Eom, Junyong Noh, and Joseph S Shin. 2016. Data-guided model predictive control based on smoothed contact dynamics. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 533–543.

Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

Nikolaus Hansen, Youhei Akimoto, yoshihikoueno, Dimo Brockhoff, and Matthew Chan. 2020. *CMA-ES/pycma: r3.0.3*. https://doi.org/10.5281/zenodo.3764210

Eric Heiden, David Millard, Hejia Zhang, and Gaurav S Sukhatme. 2019. Interactive differentiable simulation. *arXiv preprint arXiv:1905.10706* (2019).

Sumit Jain, Yuting Ye, and C Karen Liu. 2009. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 1–12.

Paul G Kry, Cyrus Rahgoshay, Amir Rabbani, and Karan Singh. 2012. Inverse kinodynamics: Editing and constraining kinematic approximations of dynamic motion. *Computers & Graphics* 36, 8 (2012), 904–915.

Taesoo Kwon, Yoonsang Lee, and Michiel Van De Panne. 2020. Fast and flexible multilegged locomotion using learned centroidal dynamics. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 46–1.

Sergey Levine and Jovan Popović. 2012. Physically plausible simulation for character animation. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 221–230.

C Karen Liu and Zoran Popović. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 408–416.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. In *ACM SIGGRAPH 2010 papers*. 1–10.

Zicheng Liu, Steven J Gortler, and Michael F Cohen. 1994. Hierarchical spacetime control. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 35–42.

Azumi Maekawa, Ryuma Niiyama, and Shunji Yamanaka. 2018. Pseudo-Locomotion Design with a Quadrotor-Assisted Biped Robot. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2462–2466.

Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. 2020. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2536–2542.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. 2020. Catch & Carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 39–1.

Filipe Figueredo Monteiro, Andre Luiz Buarque Vieira, João Marcelo Xavier Natário Teixeira, Veronica Teichrieb, et al. 2019. Simulating real robots in virtual environments using NVIDIA's Isaac SDK. In *Anais Estendidos do XXI Simpósio de Realidade Virtual e Aumentada*. SBC, 47–48.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–11.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Michael Posa, Cecilia Cantu, and Russ Tedrake. 2014. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research* 33, 1 (2014), 69–81.

Amir H Rabbani and Paul G Kry. 2016. PhysIK: Physically Plausible and Intuitive Keyframing.. In *Graphics Interface*. 153–161.

Joose Rajamäki and Perttu Hämäläinen. 2018. Continuous control monte carlo tree search informed by multiple experts. *IEEE transactions on visualization and computer graphics* 25, 8 (2018), 2540–2553.

Paul SA Reitsma and Nancy S Pollard. 2003. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. In *ACM SIGGRAPH 2003 Papers*. 537–542.

Kwang Won Sok, Katsu Yamane, Jehee Lee, and Jessica Hodgins. 2010. Editing dynamic human motions via momentum and force. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer animation*. 11–20.

Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4906–4913.

Michiel van de Panne and Alexis Lamouret. 1995. Guided optimization for balanced locomotion. In *Computer Animation and Simulation'95*. Springer, 165–177.

Kevin Wampler and Zoran Popović. 2009. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.

Andrew Witkin and Michael Kass. 1988. Spacetime constraints. *ACM Siggraph Computer Graphics* 22, 4 (1988), 159–168.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 33–1.

Pawel Wrotek, Odest Chadwicke Jenkins, and Morgan McGuire. 2006. Dynamo: dynamic, data-driven character control with adjustable balance. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*. 61–70.

KangKang Yin, Stelian Coros, Philippe Beaudoin, and Michiel Van de Panne. 2008. Continuation methods for adapting simulated skills. In *ACM SIGGRAPH 2008 papers*. 1–7.

Wenhao Yu, Greg Turk, and C Karen Liu. 2018. Learning symmetric and low-energy locomotion. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Victor Zordan, David Brown, Adriano Macchietto, and KangKang Yin. 2014. Control of rotational dynamics for ground and aerial behavior. *IEEE transactions on visualization and computer graphics* 20, 10 (2014), 1356–1366.

Victor Brian Zordan and Jessica K. Hodgins. 2002. Motion Capture-Driven Simulations That Hit and React. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/545261.545276

APPENDICES

A TABLE OF HYPERPARAMETERS

In Table 1 we provide a table of hyperparameters used for generating our results. Tuning these hyperparameters is a straightforward process, where we change the values a little at a time and observe the quality of generated motions. We find the number of action pieces, the range of allowed time-to-arrival values, and the regularization weight for external assist modulation to play significant roles in generating visually pleasing results.

Table 1. The table of hyperparameters for the motions appearing in this paper and the supplementary video.

Motion	# action pieces	min. Δt	max. Δt	λ_β (10^x)
Luxo reverse somersault	10	5	40	+2.00
Luxo clamber	10	10	60	+0.25
Luxo jump-hop	10	5	30	+2.00
Luxo high box jump	10	5	45	+3.00
Luxo wall flip	10	10	30	+2.00
Luxo cliff-walk	30	15	40	+1.00
Luxo cliff-jump	13	10	30	+0.20
Desk cliff-walk	20	5	40	+0.85
Desk cliff-jump	13	5	30	+1.50
Desk wall jump	10	15	30	+0.65
Folk forward roll	10	15	30	+0.50
Folk backward roll	14	15	40	-1.00
Folk long jump	12	5	35	+1.00
Folk parkour	16	5	25	-0.50
Folk wall flip	10	10	55	+1.75
Folk valut	13	5	30	+1.50
Folk handstand	15	10	45	+0.70
Boxy supine getup	10	30	65	-2.00
Boxy prone getup	10	40	60	-3.00
Boxy wall flip	10	10	30	+1.00
Boxy backtumble	10	10	30	-2.00
Boxy turn-and-flip	15	5	40	-1.00
Boxy high jump	13	5	20	+0.00
Boxy walk and sit	20	15	30	-2.00

B USER INTERFACE

Our user interface prototype is built to enable rapid prototyping of user-directed physics-based character animation, borrowing from traditional keyframing-based animation workflow. Instead of kinematic interpolation, we offer the capability to launch an optimization job based on the parameters specified by the control panel. Figure 12 shows the full view of our DearPyGui-based control panel, which is used in conjunction with the display of the simulation environment.

Please refer to the supplementary video for demonstrations of the graphical user interface.

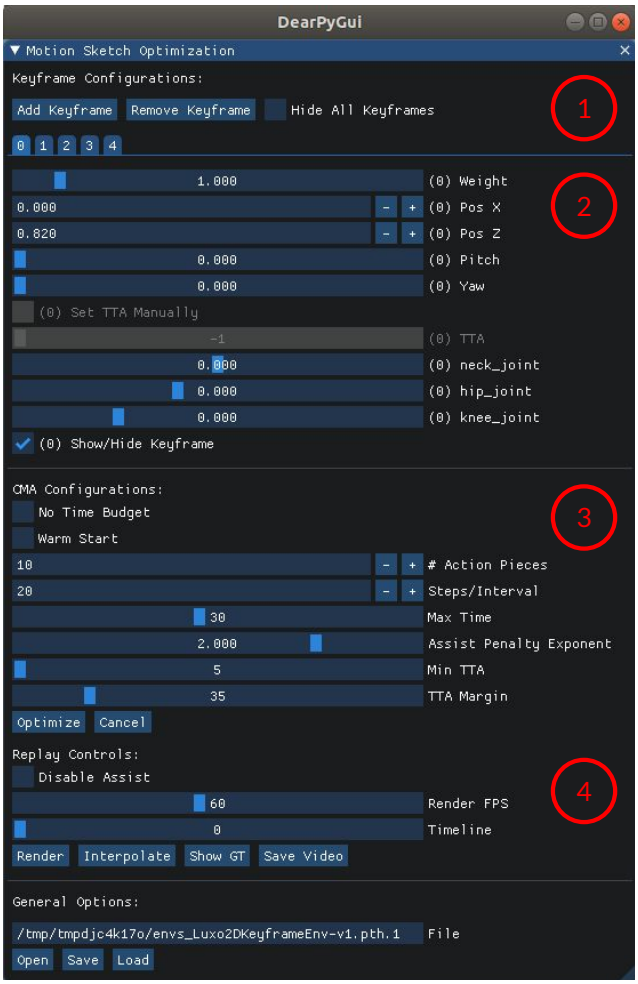


Fig. 12. The full view of the control panel of our user interface. **1:** keyframes are displayed as tabs that can be added and removed. **2:** keyframe configurations, such as root position, orientation, joint angles, and keyframe weights can be manipulated with sliders and button modules. **3:** CMA configurations are exposed to the user, including action lengths, time-to-arrival ranges, penalty weights, and computation budget parameters. **4:** the user can display optimized motions as well as reference motions induced by input keyframes.

C ANIMATED CHARACTERS

We prepare four rigid-body characters to demonstrate the robustness and capacity of the optimized motions. Our method generalizes across 2D and 3D characters. In this section, we describe three 2D characters and one 3D character. Their representative figures are shown in Figure 13.

Luxo2D. The classic 2D Luxo is a 6-DoF character equipped with neck, hip, and knee joints. The primary locomotion mode for the Luxo character is hopping.

Desk2D. The desk character is a biped with a wide base of support capable of sideways crab-like jumps and locomotion.

Fig. 13. Representative figures of the animated characters. From left to right: *Luxo2D*, *Desk2D*, *Folk2D*, and *Boxy*.

Table 2. Evaluation of Horizon Curriculum. Each value shows the mean and standard deviation of five independent optimization runs with different random seeds. The *Time* columns are the overall optimization wall-clock time in seconds. The *Cost* columns show the scaled ($\times 10^{-2}$) objective function values when optimization is completed.

	Curriculum		No Curriculum	
	Time (s)	Cost	Time (s)	Cost
Luxo Back Flip	17.9 ± 4.1	0.9 ± 0.2	27.6 ± 4.1	0.8 ± 0.4
Desk Wall Jump	112.8 ± 8.9	7.5 ± 0.4	54.2 ± 4.6	5.3 ± 0.4
Folk Wall Flip	78.3 ± 3.1	8.9 ± 2.5	101.1 ± 16.5	3.6 ± 0.3
Boxy Jump	50.8 ± 7.2	3.0 ± 1.1	74.0 ± 22.9	2.7 ± 0.5

Folk2D. The 2D humanoid character is equipped with the typical number of joints as 3D humanoid settings, with all rotations constrained to the sagittal plane, hence only using y-axis rotation.

Boxy: A 3D Biped. Boxy is a simplified 3D humanoid biped character with a cubic torso. We constrain the motion of all limb joints to the sagittal plane, while allowing for abdominal rotation into both the sagittal and the horizontal planes.

D IMPLEMENTATION

We use PyBullet [Coumans and Bai 2020] to simulate the physical interactions between the characters and the environment. The flexible programming interface of PyBullet allows character keyframes to be dynamically added and removed at run-time. We use a publicly available implementation of CMA-ES [Hansen 2016; Hansen et al. 2020] for optimization.

E EVALUATION OF HORIZON CURRICULUM

We evaluate the effect of horizon curriculum on the optimization time and objective function cost at convergence. For experiments with horizon curriculum disabled, we set the initial curriculum stage to span all keyframes. Table 2 summarizes the result. A caveat is that a lower objective function value is not always indicative of better motion. For example, we find the quality of wall flip motions for values 8.9×10^{-2} and 3.6×10^{-2} to be visually similar.

The horizon curriculum serves to reduce the perceived latency of the optimization process by immediately having short-horizon results to display. However, this can sometimes come at the cost of slowing the overall optimization. In some cases, such as Desk2D wall jump, we find that

Table 3. Effects of CMA-ES population size on final objective function values. Time and cost are reported the same way as Table 2.

Population Size	P=8	P=16	P=32	P=48
Luxo Back Flip				
Time (s)	9.4 ± 0.8	8.0 ± 0.6	11.9 ± 0.9	17.9 ± 4.1
Cost	5.9 ± 1.7	3.1 ± 1.6	2.1 ± 1.0	0.9 ± 0.2
Folk Wall Flip				
Time (s)	58.8 ± 2.9	54.1 ± 4.7	63.0 ± 2.8	78.3 ± 3.1
Cost	18.3 ± 5.0	13.3 ± 3.5	9.9 ± 2.4	8.9 ± 2.5

horizon curriculum can degrade the convergence speed when the optimized motions are complex. This behavior is expected since the growing horizon approach of the curriculum means that earlier optimization stages cannot consider the impact of later keyframes. Consequently the earlier solutions may fall in *local minima*, and thus need to be re-optimized as new keyframes become available. For simpler motions, the total optimization times are comparable between training with and without horizon curriculum. In the case of Luxo2D back flip and Boxy jump, the optimization time with horizon curriculum is on average faster than training without curriculum.

F SENSITIVITY TO POPULATION SIZE

The optimized motions can exhibit some sensitivity to the population size of CMA-ES. In general, evolution strategies tend to find better solutions when using larger populations. A larger population size alleviates the problem of local minima by maintaining a more diverse set of individuals throughout the optimization process. We observe this phenomenon in motion sketch optimization, as shown in Table 3. Since the experiments were performed on a 24-core machine, we expect the optimization time to be at a minimum when population size is equal to 24. We also note that better solutions are achieved at convergence with a larger population size. In the back flip and wall flip examples, the difference in motion quality between $P = 8$ and $P = 48$ is noticeable. Using lower population size is more likely to produce motions with incomplete rotations, which inhibit the characters from performing a stable landing. Thus, when considering the trade-off between optimization time and motion quality, we set the population size to 48 for all our experiments.

G TRADE-OFF: TRACKING AND ASSIST USE

As with most optimization problems involving regularization, there is a trade-off between keyframe tracking performance and the amount of external assist used, induced by the choice of regularization strength λ_β as appears in Equation 4. Intuitively, larger values of λ_β discourage the use of external assistive forces and thus encourages more physically-based solutions to emerge, which may have worse keyframe tracking performance when the specified motion is difficult. Smaller values of λ_β, on the other hand, may result in solutions that rely completely on external assist, which makes the result visually unappealing due to ragdoll-like behaviors.

This trade-off is visualized in Figure 14, where varying levels of λ_β result in solutions lying on different parts of the plot, implicitly forming a trade-off frontier. For this experiment, for each value of λ_β, we budget 15,000 samples per run and obtain 5 independent runs of CMA-ES using our time-indexed β approach. We use a simple box jump motion (see Figure 15) that is difficult but

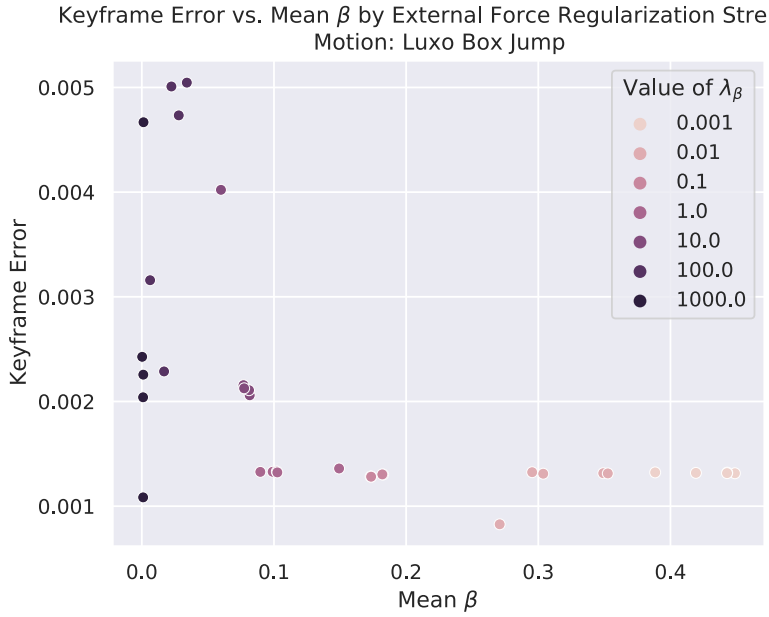


Fig. 14. Comparison of performance across 7 different values of regularization strength λ_β. Each point represents a solution. Ideal solutions (small assist, small error) reside on the bottom-left corner.

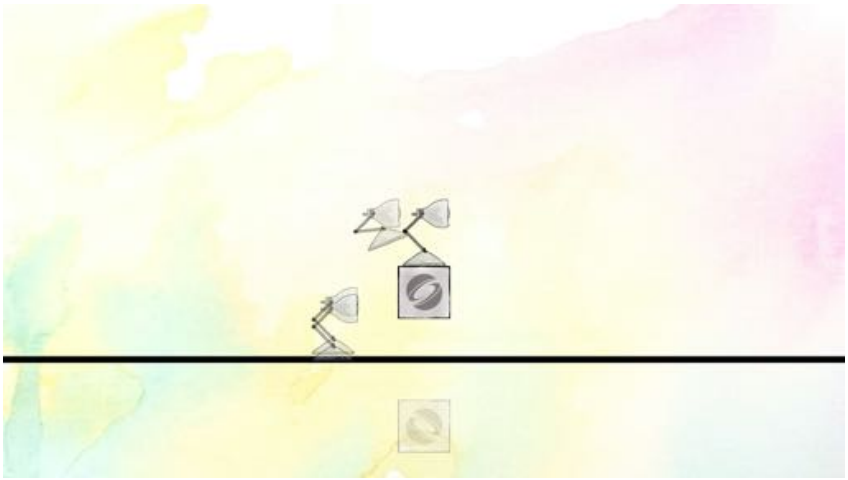


Fig. 15. The “low box jump" scenario that is difficult but possible to produce the desired motion without external assist.

possible to perform without external assist. Figure 16 shows that a high λ_β results in physically-based motions incurring high keyframe error while a low λ_β results in passive and ragdoll-like motions incurring low keyframe errors.

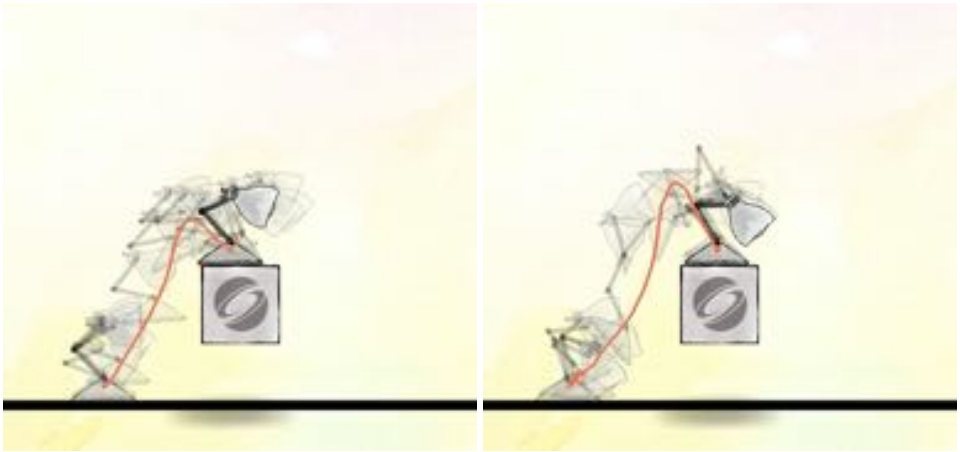
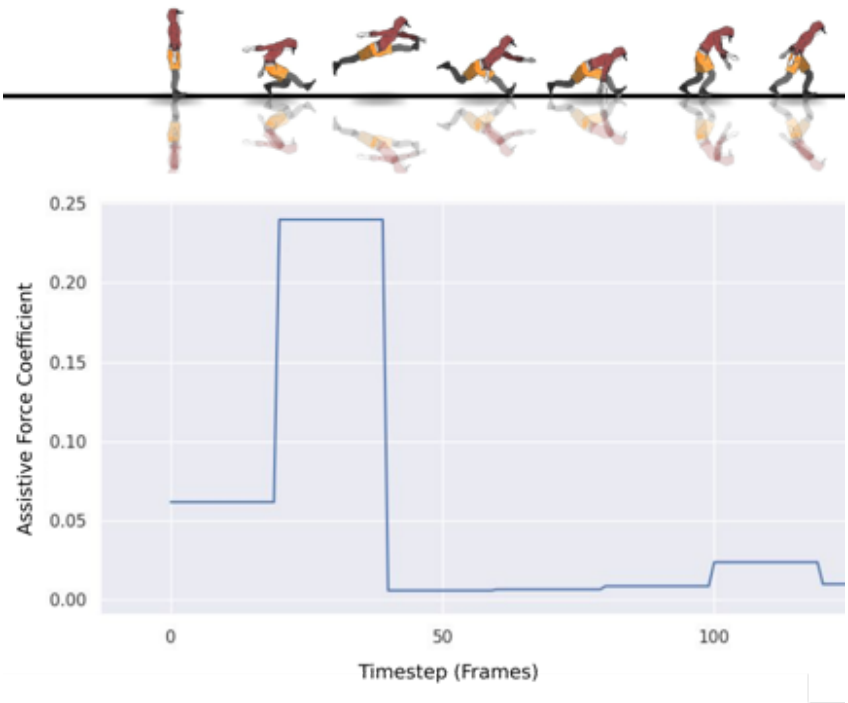


Fig. 16. A side-by-side comparison of the character performing the "low box jump" task (keyframes shown in Figure 15) according to different regularization strengths. **Left:** optimized with $\lambda_\beta = 1e3$. **Right:** optimized with $\lambda_\beta = 1e-3$. Note the emergent backflip which helps the character stay on a physically-based trajectory but completely ignores an intermediate keyframe.

H TRACKING ASSIST OVER TIME

We also track the usage of assistive forces within a single roll-out episode for a long jump motion, shown in Figure 17. The time period with the highest assistive forces corresponds to the motion segment that demands the most unphysical accelerations. This observation is also aligned with our intuition.

(a) Input keyframes for an impossible long jump



(b) Progression of optimized β_t values over time and snapshots of optimized motion corresponding to each value.

Fig. 17. This example shows the amount of assistive forces used to perform an impossible long jump by Folk. In the final optimized motion, only a significant amount of assist is used right before the flight phase.