Complementary Dynamics

JIAYI ERIS ZHANG, SEUNGBAE BANG, DAVID I.W. LEVIN, ALEC JACOBSON, University of Toronto

Input skinning animation with two "bones"  
Output animation with secondary effects

Fig. 1. Animation artists use low-dimensional rigs to control the primary motion of their characters. Our displacement filtering optimization adds elastodynamic secondary effects orthogonal to the rig’s subspace, ensuring that interesting effects emerge (right) but do not compete with or undo the artist’s original animation (left). Throughout the figures in our paper, the symbol indicates a corresponding clip in the supplemental video.

We present a novel approach to enrich arbitrary rig animations with elastodynamic secondary effects. Unlike previous methods which pit rig displacements and physical forces as adversaries against each other, we advocate that physics should complement artists’ intentions. We propose optimizing for elastodynamic displacements in the subspace orthogonal to displacements that can be created by the rig. This ensures that the additional dynamic motions do not undo the rig animation. The complementary space is high-dimensional, algebraically constructed without manual oversight, and capable of rich high-frequency dynamics. Unlike prior tracking methods, we do not require extra painted weights, segmentation into fixed and free regions or tracking clusters. Our method is agnostic to the physical model and plugs into non-linear FEM simulations, geometric as-rigid-as-possible energies, or mass-spring models. Our method does not require a particular type of rig and adds secondary effects to skeletal animations, cage-based deformations, wire deformers, motion capture data, and rigid-body simulations.

CCS Concepts: • Physical simulation;

Additional Key Words and Phrases: physically based animation, constrained simulation, character rigging, secondary motion, orthogonality

ACM Reference Format:
Jiayi Eris Zhang, Seungbae Bang, David I.W. Levin, Alec Jacobson. 2020. Complementary Dynamics. ACM Trans. Graph., 39, 6, Article 179 (December 2020), 11 pages. https://doi.org/10.1145/3414685.3417819

1 INTRODUCTION

The performance of elastodynamics simulation algorithms for character animation has improved by leaps and bounds in the past few years. More than ever, the major bottleneck in realistic animation creation is artistic control.

Controlling physical effects by meticulously setting up initial conditions and material parameters alone is a non-starter. Ideally, an artist could use friendly interfaces to animate dominant effects (e.g., keyframing, mocap, skinning) and rely on physical simulation to add secondary effects. Space-time constraint systems use physics to interpolate sparse key poses, but this can be awkward, inflexible, and computationally intractable. Existing tracking methods can also be frustrating: either physics does not have enough freedom to add interesting effects or physics has too much freedom and undoes the artist’s intentions.

A paradox seems to appear. The artist’s rig displacements cannot be treated as hard constraints: otherwise physics has no room for secondary effects. Meanwhile, physics cannot have too much freedom, so as to undo the artist’s work.

We observe that these goals — creative primary effects and physical secondary effects — are not contradictory, but rather they are complementary. In this paper, we show that this is true not just philosophically, but also algebraically. We propose a displacement filtering approach that enables physically based secondary dynamics lying solely in the orthogonal complement of the artist’s animation. Our physical simulation can add wiggles and jiggles that respond to external forces by superimposing displacements that the artist strictly could not have made herself with the given rig. That is, the physical simulation cannot add a displacement that could have otherwise been added by the artist. This implements a contract with the artist ensuring that their creative intentions are not undone by the physical dynamics, while ensuring that the simulation has enough degrees of freedom to add interesting effects.

In contrast to previous tracking or constraint-based methods, our method does not require a careful segmentation, remeshing, or masking of the input geometry. Our method is plug-and-play with existing elastodynamics methods, and as a result inherits their real material parameters and non-linear effects. Our method is agnostic to the elastic model, applying both to continuum models and simple mass-spring systems. We require only first-order differentiability of the input animation with respect to the artist’s parameters allowing us to add secondary effects to animations created with...
We do not present a new model of elasticity nor a performance with recent advances in elasticity simulation is interesting, it is out-impossible with rigid internal bones alone.

soft surface features such as facial expressions can be difficult or a desired silhouette [Thomas and Johnston 1981] or controlling the visual exterior) delegated to the physical simulation. Achieving rigid embedded skeleton. This leaves the control of the "skin" (i.e., the physical simulation match its positions in a weak sense (integrated under pre-defined patches). Again, the user must find a good segmentation task, perhaps more difficult than modeling the input’s surface geometry. Fixing too much of the interior leads to diminished secondary effects; fixing too little or too deep will not allow the artistic intention to diffuse to the surface. Li et al. [2013] treat the entire geometry of the input animation as fixed except for secondary skin sliding effects tangential to the surface, however this requires a high quality UV parameterization with seam bookkeeping. Malgat et al. [2015] use a kinetic filter at the velocity level to add local, physics-driven motion to underlying coarse discretizations. In contrast to our method which allows global interactions, their method only allows adding localized motion, making it unsuitable for many tasks in animation. It also requires additional positional constraints to avoid constraint-drift caused by linearizing constraints at the velocity level.

Instead of geometric bones, the user could be asked to specify which regions of an input animation should be fixed and which should be free [Kim et al. 2017; Kozlov et al. 2017; Li et al. 2016]. In many cases, this is a non-trivial segmentation task, perhaps more difficult than modeling the input’s surface geometry. Fixing too much of the interior leads to diminished secondary effects; fixing too little or too deep will not allow the artistic intention to diffuse to the surface. Li et al. [2013] treat the entire geometry of the input animation as fixed except for secondary skin sliding effects tangential to the surface, however this requires a high quality UV parameterization with seam bookkeeping. Malgat et al. [2015] use a kinetic filter at the velocity level to add local, physics-driven motion to underlying coarse discretizations. In contrast to our method which allows global interactions, their method only allows adding localized motion, making it unsuitable for many tasks in animation. It also requires additional positional constraints to avoid constraint-drift caused by linearizing constraints at the velocity level.

This literal interpretation of the skeletal control rig requires meaningful “geometric bones” for each rig parameter, thus does not directly apply to: cage-based deformation (Fig. 2), blendshapes, abstract rigs, or rigid-body controllers. Many hierarchical rigs use the skeleton metaphor, but are not intended to be geometrically interpreted as "trees of rigid bars" inside the shape. In Fig. 4, two "bones" with smooth skinning weights control the soft trunk of an elephant. Interpreting the rig as a geometric skeleton so as to impose fixed value constraints results in awkward kinks. In contrast, our approach interprets the rig algebraically and augments the original smooth skinning deformation with smooth secondary dynamics.

In lieu of geometric bones, the user could be asked to specify which regions of an input animation should be fixed and which should be free [Kim et al. 2017; Kozlov et al. 2017; Li et al. 2016]. In many cases, this is a non-trivial segmentation task, perhaps more difficult than modeling the input’s surface geometry. Fixing too much of the interior leads to diminished secondary effects; fixing too little or too deep will not allow the artistic intention to diffuse to the surface. Li et al. [2013] treat the entire geometry of the input animation as fixed except for secondary skin sliding effects tangential to the surface, however this requires a high quality UV parameterization with seam bookkeeping. Malgat et al. [2015] use a kinetic filter at the velocity level to add local, physics-driven motion to underlying coarse discretizations. In contrast to our method which allows global interactions, their method only allows adding localized motion, making it unsuitable for many tasks in animation. It also requires additional positional constraints to avoid constraint-drift caused by linearizing constraints at the velocity level.

Instead of a geometric rig or fixed inner core, Bergou et al. [2007] input an existing coarse mesh animation and then require a high-resolution simulation match its positions in a weak sense (integrated over pre-defined patches). Again, the user must find a good segmentation. Fig. 5 shows that too few patches allow the tracked simulation to deviate from the input and too many prohibit secondary effects.

2.2 Spacetime Constraints

Keyframing is another popular animation metaphor, traditionally lacking secondary effects. Instead of interpolating rig parameters between key poses, the key poses can be interpreted as sparse constraints on a spacetime optimization to find the most physically plausible animation [Witkin and Kass 1988]. This changes the simulation from an instantaneous integration problem into a coupled problem over all timesteps; many of the recent advances have sought
Fig. 3. Simulation in the subspace of the rig (e.g., [Hahn et al. 2012]) is limited by the expressivity of the rig, which is typically designed for primary motions. If the artist plans to control the entire rig, then extra parameters need to be added for rig-space physics to have an effect. Our secondary effects lie in the space orthogonal to the rig and require no extra rigging.

Fig. 4. Previous methods interpret input skeletal rigs as "trees of rigid bars" embedded in the elastic shape (e.g., [Capell et al. 2002; Li et al. 2019]). But often the skeleton is just a hierarchy metaphor for controlling a smooth subspace. Our interpretation of the rig is through its action on the shape, not its geometric handle locations.

Fig. 5. Bergou et al. [2007] track input animations by weak constraints defined over patches. The output is sensitive to the distribution and number of patches. Problems occur at either extreme. Finding a good balance is an additional burden for the user. Our method does not require clustering.

2.3 Rig-Space Physics

Rig space physics constrains the displacements of secondary effects to lie in the subspace spanned by the artists’ rig [Hahn et al. 2012]. This has the immediate performance advantages of a reduced deformable model [Barbic and James 2005; Der et al. 2006; Gilles et al. 2011; Xu and Barbic 2014] and has proven useful as a form of regularization during performance capture [Liu et al. 2020; Ribera et al. 2017]. For character animation, the contract with the artist is explicit: the artist segments the rig parameters into free and fixed. The failure modes are two sides of the same coin. Since physics can only make motions spanned by the rig, interesting secondary effects may require augmenting the rig with new auxiliary degrees of freedom [Jacobson et al. 2012; Wang et al. 2015]. Alternatively, the artist loses control over parameters delegated to physics, which may prevent realization of the artist’s intent. When acting in the same subspace, the artist and physical simulation are in adversarial roles. Instead, they could be working in harmony to control primary and secondary effects respectively in complementary subspaces.

2.4 Modifying the Reference Configuration

A core common problem is that giving too much freedom to the physics simulation will undo the artists’ intended primary animation (see Fig. 6). Minimizing an elastic energy exerts forces pushing the shape back to its reference configuration away from the artist’s pose. Treating the current rig pose at any moment in the animation as

animators to providing sparse poses giving physics full power over motion, even primary motions, not just secondary effects.
3.1 Dynamic Simulation

Following many previous works (e.g., [Gast et al. 2015; Liu et al. 2013; Martin et al. 2011]), we employ an implicit Euler integration of Newton’s Second Law of Motion in terms of displacements. For a given moment in time, \( t \), the current displacements \( u_t \) are the result of a possibly non-linear optimization problem:

\[
\mathbf{u}_t = \underset{\mathbf{u}_t}{\operatorname{argmin}} \ E_t(\mathbf{u}_t),
\]

where \( E_t \) changes over time due to momentum and external forces and is defined as a sum of potentials:

\[
E_t(\mathbf{u}_t) = \frac{1}{2} \mathbf{M} \dot{\mathbf{u}}_t^2 + \frac{h^2}{2} \mathbf{M} \dot{\mathbf{u}}_t^2 + \mathbf{M} \mathbf{f}(\mathbf{u}_t),
\]

where \( h > 0 \) is the time step value, \( \mathbf{M} \in \mathbb{R}^{dn \times dn} \) is the mass matrix, \( \mathbf{f} : \mathbb{R}^{dn} \rightarrow \mathbb{R}^{dn} \) defines external forces, and we use dot notation to denote temporal finite differentiating:

\[
\dot{\mathbf{u}}_t := \frac{\mathbf{u}_t - \mathbf{u}_{t-h}}{h} \quad \text{and} \quad \ddot{\mathbf{u}}_t := \frac{\mathbf{u}_t - \mathbf{u}_{t-h}}{h}.
\]

Substituting Equation (1) into Equation (2) results in a generic optimization problem over the complementary displacements \( \mathbf{u}_c^c \):

\[
\mathbf{u}_c^c = \underset{\mathbf{u}_c^c}{\operatorname{argmin}} \ E_t(\mathbf{u}_t^r + \mathbf{u}_c^c).
\]

Unconstrained, this optimization will undo the rig displacements \( \mathbf{u}_t^r \) (see Fig. 6). Requiring \( \mathbf{u} \) to track \( \mathbf{u}^c \) in a least-squares sense (e.g., adding a \( \| \mathbf{u} - \mathbf{u}^c \|^2 \) or equivalently \( \| \mathbf{u}^c \|^2 \) term) will prevent undoing the rig displacements but also damp out high-frequency motions. These are motions that the rig cannot create, so they would otherwise be welcome secondary effects (see Fig. 5). Instead, we now introduce constraints to prevent exactly the dynamics lying in the space spanned by the rig.

3.2 Rig Orthogonality Constraints

Our goal is to constrain the complementary displacements \( \mathbf{u}_c^c \) to only those displacements that could not be created by the rig \( \mathbf{u}_c^r \). That is, physics should not presume to take over any controls from the artist using the rig. Given current rig parameters \( p_t \), we can verify whether any candidate displacements \( \mathbf{u}_c^c \) satisfies this criteria by ensuring that projecting to the closest rig displacement simply rediscovers \( p_t \):

\[
\underset{\mathbf{p}_t}{\argmin} \frac{1}{2} \| \mathbf{u}^r(p_t) + \mathbf{u}_c^c - \mathbf{u}^r(\mathbf{p}) \|^2_M = \mathbf{p}_t,
\]

where \( \| \mathbf{x} \|^2_M = \mathbf{x}^\top \mathbf{M} \mathbf{x} \). This should not be confused for a definition of \( \mathbf{p}_t \), rather it is a (currently non-linear) equality constraint on \( \mathbf{u}_c^c \). This constraint appears daunting as it involves argmin and the possibly non-linear rig function \( \mathbf{u}^r \).
Let us consider the first-order necessary conditions of the argmin operation with respect to \( p \) evaluated at the right-hand side \( p_t \):

\[
\frac{\partial}{\partial p} \left( \| u'(p_t) + u_c - u'(p) \|_H^2 \right)_{p_t} = 0, \tag{7}
\]

\[
\frac{\partial}{\partial p} \left( u'(p_t) + u_c - u'(p) \right)^\top M \left( u'(p_t) + u_c - u'(p) \right) = 0, \tag{8}
\]

\[
\frac{\partial}{\partial p} \left( u'(p_t) + u_c - u'(p) \right)^\top J^t_c = 0. \tag{9}
\]

The final expression is linear in the unknown complementary displacements \( u_c \). The rig Jacobian matrix \( J_t \in \mathbb{R}^{dn \times m} \) in general changes with each time step, but does not depend on the unknown complementary displacements \( u_c \).

We assume that the input rig function \( u' \) is differentiable. This is reasonable as most rigs are intended to create smooth animations. This is also milder and less computationally intensive than previous methods: Hahn et al. [2012] require second derivatives.

This constraint can be read as requiring that \( u_c \) lies in the orthogonal complement of the rig, to first-order.

The mass matrix \( M \) appears in Equation (9) due to the integrated measure in Equation (6). Omitting this mass matrix results in secondary effects that are biased by the mesh discretization (see Fig. 7).

### 3.3 Constrained Simulation

Our general algorithm involves differentiating the rig at the current pose, performing one constrained optimization integration step, and then summing the displacement contributions (Algorithm 1).

**Algorithm 1: Complementary Dynamics Simulation:** given \( p_t \)

\[
J_t \leftarrow \frac{\partial u_c}{\partial p} |_{p_t}
\]

\[
J_t^c \leftarrow \text{argmin}_{u_c} E_t(u_c + u_t^c) \text{ subject to } J_t^c M u_c^c = 0
\]

\[
u_t^c \leftarrow u_t^c + u_c^c
\]

To demonstrate the effectiveness and generality of this method, let us consider various choices of rig functions (\( u' \)) and various choices of elastic potentials (\( \Phi \) within \( E_t \) in Equation (3)).

**Linear Elasticity.** As a warm-up, let us start with linear elasticity which is captured by a quadratic potential:

\[
\Phi_{\text{linear}}(u) = \frac{1}{2} u^\top K u, \tag{10}
\]

where \( K \in \mathbb{R}^{dn \times dn} \) is the constant stiffness matrix.

---

Fig. 7. Complementary vibrations jiggle through a cartoon worm (top) with a coarse mesh (top-middle). Including the mass matrix \( M \) in the constraint ensures discretization independence (bottom-middle). Omitting it causes densely meshed regions to be overly stiff.

Using the Lagrange multiplier method, the optimal complementary displacements \( u_c^* \) can be found by solving the linear system:

\[
\begin{bmatrix}
K + \frac{M}{h} & M^\top J_t^c \\
J_t^c M & 0
\end{bmatrix}
\begin{bmatrix}
\lambda \\
u_c^*
\end{bmatrix}
= \begin{bmatrix}
-K u_t^c - \frac{M}{h} \left( \frac{u_t^c}{h^2} - \frac{u_{t+h} - u_{t-h}}{h} \right) + f_t \\
0
\end{bmatrix},
\tag{11}
\]

where the Lagrange multiplier values \( \lambda \in \mathbb{R}^m \) may be discarded.

As a quadratic function, the Hessian of \( \Phi_{\text{linear}} \) is constant and Equation (11) can be interpreted as a single (perfect) Newton–Raphson method iteration where \( K = \partial^2 \Phi / \partial u^2 \) and \( K(u_t^c + u_c^*) = \partial \Phi / \partial u^c \).

**Non-Linear Elasticity.** Linear elasticity is a poor model for large displacements. A popular non-linear replacement is neo-Hookean elasticity, a non-linear model, \( \Phi_{\text{neo}} \) (see, e.g., [Sifakis and Barbic 2012]), though the derivations below generalize to many non-linear models. We optimize the complementary displacements \( u_t^c \) again by using the Lagrange multiplier method, but now applied repeatedly during each iteration of a Newton-Raphson’s algorithm. Each iteration the potential \( E \) is approximated to second-order using a Taylor expansion, requiring the first and second derivatives of the neo-Hookean potential (Algorithm 2).

**Local-Global Energies.** While Newton’s method can be applied to a large class of energies, a popular style of optimization for elastic simulations are so-called local-global solvers (e.g., [Bouaziz et al. 2014; Jacobson et al. 2012; Kovalsky et al. 2016; Liu et al. 2013; Sorkine and Alexa 2007; Wang et al. 2015]). Let us consider the mass-spring solver of Liu.
Algorithm 2: Modified Newton’s Method: given $u^t, J_t$

\[
\begin{align*}
& u_{t}^c \leftarrow u_{t-1}^c \\
& \text{repeat} \\
& \quad g, K \leftarrow \partial \Phi / \partial u^c, \partial^2 \Phi / \partial u^c^2 \\
& \quad Q \leftarrow K + h^2 M \\
& \quad f \leftarrow -g + \frac{M}{h} (u_{t}^c - u_{t-1}^c - \hat{u}_{t-1}^c) + f_t \\
& \quad C \leftarrow J_t^H M \\
& \quad \text{Solve } Q C^T x = f \\
& \quad s \leftarrow \text{line search from } u_{t}^c \text{ toward } x \text{ according to } E_t \\
& \quad u_{t}^c \leftarrow u_{t}^c + s(x - u_{t}^c) \\
& \text{until } s < \epsilon;
\end{align*}
\]

et al. [2013]. This method introduces auxiliary “local” variables representing the fixed-length direction of each spring so that the elastic potential becomes quadratic with respect to the “global” displacement variables. The optimization alternates between updating the local variables via vector normalization and solving a linear system to update the global variables. We can immediately bootstrap this optimization by enforcing $J_t^H M u_{t}^c = 0$ as a linear equality constraint during the global step, leaving the local step intact. The inset figure shows mass-spring secondary-effects enriching a keyframe animation of a roller coaster.

3.4 Rig Derivatives

Our method requires differentiating the rig displacements with respect to the rig pose to build the Jacobian matrix of partial derivatives, $J = \frac{\partial u^c}{\partial p} \in \mathbb{R}^{d \times n \times m}$. Many popular rigging methods are linear in their rig parameters, so their Jacobian matrix is constant.

**Linear Blend Skinning** computes displacements at each vertex with rest position $v_i \in \mathbb{R}^d$ as a weighted average of $k$ affine “bone” transformations ($T_j \in \mathbb{R}^{d \times (d+1)}$ for each bone $j$):

\[
\begin{align*}
& u_{j}^{\text{lbs}} = \left( \sum_{j=1}^{k} w_{ij} T_j \right) v_i - v_i, \\
& \text{where the scalar per-vertex-per-bone skinning weights } w_{ij} \text{ are constants at pose time.}
\end{align*}
\]

If we vectorize each matrix $T_j$ into $p_j = \text{vec}(T_j) \in \mathbb{R}^{d(d+1)}$, then we can write linear blend skinning displacements in Equation (12) as a sum of matrix multiplications:

\[
\begin{align*}
& u_{j}^{\text{lbs}} = \left( \sum_{j=1}^{k} w_{ij} (I_d \otimes v_j^T) 1 \right) - v_i \\
& \text{where } I_d \in \mathbb{R}^{d \times d} \text{ is the identity matrix and } \otimes \text{ indicates the Kronecker product of } A \text{ and } B. \text{ Concatenating all of the } p_j \text{ vectors into one tall vector of rig parameters } p \in \mathbb{R}^{k(d+1)}, \text{ we can write the linear blend skinning displacement function as a matrix multiplication and vector subtraction:}
\end{align*}
\]

\[
\begin{align*}
& u^t = \begin{bmatrix} J_{11} & \cdots & J_{1k} \\ \vdots & \ddots & \vdots \\ J_{nk} & \cdots & J_{nk} \end{bmatrix} \begin{bmatrix} p - v \end{bmatrix} \\
& \text{As such, the matrix } J \text{ is immediately revealed to be the rig’s constant Jacobian.}
\end{align*}
\]

**Affine Body** A very useful special case of linear blend skinning is to consider that the entire object is controlled by a single affine transformation (and all weights are one $w_{ij} = 1$). This is a very common control metaphor in keyframing software such as Maya, Blender, and After Effects. In this case the matrix expression in Equation (14) reduces to:

\[
\begin{align*}
& u^{\text{affine}} = \begin{bmatrix} v_{1}^T \vdots v_{n}^T \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} p - v \\
& \text{where } p \text{ is the rig’s rest position, and } v_i \text{ is the } i\text{th vertex.}
\end{align*}
\]

**Cage-based Deformation** using generalized barycentric coordinates (e.g., [Joshi et al. 2007]) is another popular special case of linear blend skinning [Jacobson et al. 2014]. Generalized barycentric coordinates represent the new position of each vertex as a weighted average of the $k$ posed cage vertex positions $c_i \in \mathbb{R}^d$:

\[
\begin{align*}
& u_{j}^{\text{cage}} = \left( \sum_{j=1}^{k} w_{ij} c_i \right) - v_i. \\
& \text{Concatenating all cage vertex positions into } p \in \mathbb{R}^{kd}, \text{ then } u^{\text{cage}} = J p - v \text{ where } J \text{ simply contains the cage coordinates } (J_{ij} = w_{ij}).
\end{align*}
\]

**Blendshapes** displace vertices by taking weighted averages of $m$ sculpted poses of the entire model:

\[
\begin{align*}
& u^{\text{blend}} = \sum_{j=1}^{m} w_{j} b_j - v, \\
& \text{where } b_j \in \mathbb{R}^{d \times n} \text{ contains the vertex positions of the } j\text{th sculpted pose and } w_j \text{ is the corresponding weight. In this case, the weights } w_j \text{ are the rig parameters that vary over time. Written in matrix form, blendshape deformation reveals its constant Jacobian:}
\end{align*}
\]

\[
\begin{align*}
& u^{\text{blend}} = \begin{bmatrix} b_1 \cdots b_m \end{bmatrix} \begin{bmatrix} w_{1} \\ \vdots \\ w_{m} \end{bmatrix} - v. \\
& \text{For any of the previous linear rig functions, the Jacobian matrix } J \text{ can be pre-computed once at the beginning of the animation as it does not depend on the changing parameters } p. \text{ When used in combination with Linear Elasticity or Local-Global Energies, the}
\end{align*}
\]
resulting large sparse system matrix also remains constant and its \( LDL^T \) factorization can be precomputed.

**Non-Linear Rigs.** Other popular rigging systems are non-linear in their rig parameters \( p \), and therefore the rig Jacobian \( J \) changes over time. Non-linear modifications of linear-blend skinning such as Wires [Singh and Fiume 1998], Dual-Quaternion Skinning [Kavan et al. 2008], Optimized Centers of Rotation [Le and Hodgins 2016], Direct Delta Mush [Le and Lewis 2019] admit analytic — albeit cumbersome — derivatives (see, e.g., [Gilles et al. 2011]). Automatic differentiation provides a simpler more general solution at approximately the same cost. Most modern autodiff libraries provide APIs for computing Jacobians directly giving a multi-variable function.

In a pinch, or if the rig function’s implementation is not available, finite differencing (cf. [Hahn et al. 2012]) can gather approximate derivatives with simple forward evaluation of the rig function. Looping over all rig parameters, we can approximate the \( j \)th column of the the Jacobian with:

\[
J_j \approx \frac{u'(p_t + \epsilon \delta_j) - u'(p_t - \epsilon \delta_j)}{2 \epsilon},
\]

(19)

where \( \delta_j \in \mathbb{R}^m \) is a vector of zeros except the \( j \)th element is a one.

We reiterate that the rig Jacobian is computed once per time-step, so when optimizing for Non-Linear Elasticity the rig Jacobian does not need to be recomputed during each Newton iteration. Hence, Jacobian computation is rarely the computational bottleneck. The inset shows a non-linear wire deformer controlling the stem of a daisy, while the pedals wiggle with secondary effects.

The inset shows dynamics added in the space orthogonal to the classic dual quaternion skinning twisting bar setup.

With various energy models and rig derivatives in hand, we have enough information to bring rig animations to life with secondary effects (see Fig. 8). In Fig. 9, a 2D cartoon keyframed with affine motion collides with the ground and impulses cause ripples through the shape without disrupting the input path.

**3.5 Rig Momentum**

In the absence of external forces (e.g., winds or collisions), there will be no secondary effect even if the rig moves. This is a direct result of our separation of tasks: the artist controls primary effects and physics in its own subspace controls separate secondary effects.

Mathematically, we can see why momentum from the rig does not induce momentum via the secondary displacements \( u' \). The object’s momentum appears interacting with the rig displacements in the term \( u'^\top Mu' \) in Equation (3). Without loss of generality consider a linear rig, then the rig’s momentum \( Mu' \) can be written as \( M|p| \), so the rig’s contribution to the momentum term is \( p'^\top J^\top Mu' \). However, the orthogonality constraint \( J^\top Mu' = 0 \) causes this term to vanishes and momentum does not pass between the rig and the simulation. This strictly implements our contract with the artist. On the other hand, in computer animation we often prefer overly floppy and energetic characters. Exaggeration and follow through principles [Thomas and Johnston 1981] are easily accomplished with secondary effects coming from perceived internal momentum.

Fortunately, we can clearly identify where to inject momentum from the rig into the dynamics. By inserting a diagonal matrix \( D \in \mathbb{R}^{n \times n} \) into the constraints (replace \( J^\top Mu' \) with \( J^\top MDu' \) in Equation (9)), we can break the annihilation in the \( u'^\top Mu' \) term and allow rig momentum to travel through the shape into the secondary displacements. This matrix acts as a map that transfers rig momentum to complementary dynamics momentum and any non-constant diagonal matrix, and most reasonable choices just work.

If and only if \( D = \epsilon I \) where \( \epsilon \) is a non-zero constant and \( I \) is the identity matrix, there will be no momentum interaction, as in all our examples with external forces (Figs. 8, 9, 10, and inset in Sec. 4). For all other examples, we construct a choice of \( D \) as a smooth transition via Poisson diffusion (i.e., \( d = 1 \) on the surface and \( \Delta d = 0 \) on the interior). For these, where \( D_{jj} = 0 \), we can expect rig momentum to “leak” into complementary dynamics momentum. This choice mimics cartoon-ish latency and follow-through [Thomas
and Johnston 1981]. Because it is a dynamics simulation, this causes an elastic wave over the shape and quickly becomes high frequency “wiggles” everywhere. So, $D_{ii} = 0$ can be seen roughly as the start locations of complementary dynamics waves. Please note that, this is in stark contrast to painting softer/stiffer parts of the mesh, since rig momentum will respect the underlying material of wherever their induced wiggles travel.

4 RESULTS

We have implemented our method in MATLAB using gptoolbox [Jacobson et al. 2018] and Bartels [Levin 2020] for geometry processing routines and elastic energy derivatives, respectively. We did not optimize our code for performance and read in animations created in Maya from file. Our method has a similar overhead as past tracking methods [Bergou et al. 2007] which also augment the system matrix with linear equality constraints. Our prototype implementation uses MATLAB’s sparse $LDL^T$ factorization to solve the KKT system in Algorithm 2. We experimented with the null-space implementation uses MATLAB’s sparse $LDL^T$ factorization to solve the KKT system in Algorithm 2. We experimented with the null-space constraint (see Section 2.3) whose search spaces are much larger than the number of rig parameters ($m$). This is a very different situation compared to subspace/rig-space physics (see Section 2.3) whose search spaces are significantly reduced ($m$). In Fig. 10, two rigged cartoons punch each other and the resolved contacts cause secondary effects without disrupting the prescribed rig motion. Our method enforces rig orthogonality with hard constraints. This might raise fears of instability due to contradictory constraints or infeasibility, for example when handling collisions. We have not observed such instabilities or infeasibilities. We attribute this to the very large search space left to the simulation, assuming the number of mesh vertices ($n$) is much larger than the number of rig parameters ($m$). The very fact that the rig alone cannot resolve contacts locally is an indication that the constrained simulation can. While it is theoretically possible for an artist to create a large, highly constraining rig that could impede collision resolution, the premise of complementary

| Models                                      | n  | m  | Avg. Frame |
|---------------------------------------------|----|----|------------|
| Roller Coaster (Inset of Sec. 3.3)          | 39 | 6  | 0.003s     |
| Non-angry bird (Fig. 12)                    | 639| 18 | 0.38s      |
| Carpet (Fig. 14)                            | 239| 18 | 0.12s      |
| Worm (Fig. 11)                              | 438| 6  | 0.05s      |
| Daisy (Inset of Sec. 3.4)                   | 702| 6  | 0.42s      |
| Monkey head (Fig. 5)                        | 1244| 12 | 2.86s      |
| Plumber (Fig. 9)                            | 1502| 6  | 0.07s      |
| Bar (Inset of Sec. 3.4)                     | 1519| 16 | 4.61s      |
| Walrus (Fig. 2)                             | 1559| 18 | 0.14s      |
| Hedgehog (Fig. 3)                           | 1528| 24 | 0.89s      |
| Staypuft (Inset of Sec. 4)                  | 3258| 72 | 359s       |
| Cow (Bowl dropping) (Fig. 15)               | 3922| 12 | 9.21s      |
| Amoeba (Fig. 8)                             | 4918| 6  | 0.06s      |
| Fish (Fig. 1)                               | 6142| 24 | 14.8s      |
| Elephant (Fig. 16)                          | 8919| 288| 38.1s      |
| TRex (Fig. 13)                              | 15623| 600| 49.2s      |

Table 1. For each example we report the size of the simulation mesh $n$, the number of rig parameters $m$ and the average time spent computing complementary dynamics for an animation frame.

![Fig. 11. Our method inherits advantages of whichever elasticity model it is plugged into (e.g., ARAP here [Chao et al. 2010]). Material parameters can be controlled just like a full-space simulation.](image-url)
Complementary Dynamics

Fig. 12. An animator can focus on the primary rig motion (skeletal skinning, here) and our optimization adds secondary effects to create detailed motions.

Fig. 13. This complex 50-bone linear blend skinning rig was downloaded from the internet https://www.cgtrader.com/3d-models/character/other/toon-dinosaurs. Despite the large rig space, our method still finds room for interesting secondary dynamics.

dynamics is that the artist controls the high-level (and naturally low-dimensional) motions and physical simulation synthesizes complex motions independent of the rig complexity.

Since our method uses standard physics solvers to create secondary motion, any contact handling scheme should be applicable. Another advantage of our physically-based approach is that heterogeneous object material parameters can be tuned to produce desired results, such as stiffening the metal propeller on the worm in Fig. 11.

One of the highlights of our method is its generality. It can bring lively secondary motion to most common rig types. As well as rigs involving multiple bones (Fig. 12) and point handles (Fig. 3), our method works with cage-based deformers like this walrus (Fig. 2) and even nonlinear rigs like this flower — animated via Catmull-Rom wire deformer (daisy inset in Section 3.4).

Naturally, these advantages extend to 3D. Our teaser fish (Fig. 1) shows an example of a two bone rig. Such a simple rig is able to instantly bring this fish to life with secondary dynamics. The animated T-Rex in Fig. 13 demonstrates our ability to add complementary dynamics to complex rigs found in the wild.

Our method exists symbiotically with standard physics solvers. This means, for instance, that we can use fast alternating solvers to create complex cloth motions, as demonstrated by this magic ride (Fig. 14). Our dancing elephant (Fig. 16) illustrates how material parameter tuning can be used to achieve the desired dynamic effect.

Rigid body simulations are easy for animators to set up and use to create complex interactions with many objects. However, the objects will looks stiff and lack elastic wiggling and jiggling. Our method can treat each object of a rigid body simulation as a keyframed animation and enrich the animation with exaggerated secondary effects (see Fig. 15). Unlike previous tracking methods (e.g., [Bergou et al. 2007]), we do not require a tuning a segmentation (see Fig. 5).

We treat a rigidly keyframed object as an Affine Body. Technically this increases the span of the rig-space (from 6 to 12 in \( \mathbb{R}^3 \)) to include scaling and shearing modes. Our treatment is tantamount to assuming that the artist has intentionally held those modes fixed to the identity. We also experimented with true 6-degree-of-freedom rigid keyframe rig function (constructing \( J \) through an instantaneous exponential map), but the results were similar with perhaps a bit more shearing in the dynamics.

5 CONCLUSIONS AND FUTURE WORK

Our complementary dynamics bring a theoretically motivated, algebraic approach to combining physics simulation with rigged animations created by an artist. By construction, our complementary dynamics cannot create motion inside of the “rig-space”, and is thus prevented from interfering with artistic intent. The algebraic nature of the method means it can be applied to a wide variety of rigs and make use of a wide variety of physics simulation algorithms — as evidenced by the many results shown above.

Alas, complementary dynamics are not complimentary: they incur a computational cost due to the additional constraints that must be applied during simulation. This paper focused on demonstrating generality with respect to the rig and elastic model, rather than the well charted territory of performance optimization. Nonetheless, it would be interesting in future work to see complementary
dynamics in a real-time setting and integrated with advanced interfaces or live performance environment [Willett et al. 2017], perhaps by leveraging model reduction [cf., [Xu and Barbic 2016]]. Adding elastodynamics to an animation pipeline brings with it a host of constraints on input data. The required mapping between well-behaved undeformed and deformed states means that our method (like all physics-based methods) is limited to well behaved geometry. In general, making physics algorithms robust to pathological geometry (e.g. non-manifold or overlapping inputs [Li and Barbic 2018; Xu and Barbic 2014]) is one of the most important pursuits in the field of physics-based animation.

Currently our method does not support topological changes or artist prescribed transitions between artworks [Bai et al. 2016], but extending our secondary effects in this way is especially interesting for 2D animation. Despite these limitations, our complementary dynamics is already a useful tool for infusing rigged animations with life. Complementary dynamics turn physics simulation into the artist’s respectful partner, rather than an unruly party crasher.

Acknowledgements

This work is funded in part by NSERC Discovery (RGPIN–2017–05524, RGPIN–2017–05235, RGPAS–2017–057938, RGPAS–2017–057909), Connaught Fund (503114), CFI-JELF Fund, New Frontiers of Research Network (NFRN–201), the Ontario Early Research Award program, the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe Systems, Autodesk and MESH Inc. We especially thank Paul Kry for hosting the 2018 Bellairs workshop on Computer Animation and the attendees for inspiring initial project ideas. We thank Otman Bencheikhroun, Rinat Abdurashitov and Josh Holinfant for proofreading; John Hancock for the IT support; anonymous reviewers for their helpful comments and suggestions.

REFERENCES

Alexis Angelides and Karan Singh. 2007. Kinodynamic skinning using volume-preserving deformations. In Proc. SCA.

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

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In Proc. SIGGRAPH.

Jernej Barbic, Marco da Silva, and Jovan Popovic. 2009. Deformable object animation using reduced optimal control. ACM Trans. Graph. 28, 3 (2009), 53.

Jernej Barbic and Doug L. James. 2005. Real-Time subspace integral for St. Venant-Kirchhoff deformable models. ACM Trans. Graph. 24, 3 (2005), 982–990.

Jernej Barbic and Yili Zhao. 2011. Real-time large-deformation substructuring. ACM Trans. Graph. 30, 4 (2011), 91.

Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: toward directable thin shells. ACM Trans. Graph. 26, 3 (2007).

Soﬁen Bouaziz, Sebastian Martin, Tianqian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. ACM Trans. Graph. 33, 4 (2014).

Steve Capell, Seth Green, Brian Curless, Tom Danchum, and Zoran Popovic. 2002. Interactive skeleton-driven dynamic deformations. ACM Trans. Graph. 21, 3 (2002).

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. ACM Trans. Graph. 29, 4 (2010), 38:1–38:6.

Jiong Chen, Hujun Bao, Tianyu Wang, Mathieu Desbrun, and Jin Huang. 2018. Numerical coarsening using discontinuous shape functions. ACM Trans. Graph. (2018).

Jiong Chen, Max Budninskiy, Houman Owhadi, Hujun Bao, Jin Huang, and Mathieu Desbrun. 2019. Material-adapted refinable basis functions for elasticity simulation. ACM Trans. Graph. 38, 6 (2019), 161:1–161:15.

Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert W. Sumner, and Markus H. Gross. 2012. Deformable objects alive! ACM Trans. Graph. 31, 4 (2012), 69:1–69:9.

Kevin G. Der, Robert W. Sumner, and Jovan Popovic. 2006. Inverse kinematics for reduced deformable models. ACM Trans. Graph. 25, 3 (2006), 1174–1179.

Dumitar Dinzev, Tianqian Liu, Jing Li, Bernhard Thomaszewski, and Ladislav Kavan. 2018. FEPFR: fast energy projection for real-time simulation of deformable objects. ACM Trans. Graph. 37, 4 (2018), 79:1–79:12.

Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfaniang Jiang, and Joseph M. Teran. 2015. Optimization Integrator for Large Time Steps. IEEE TVCG (2015).

Benjamin Gülden, Guillaume Bousquet, François Faure, and Dinesh K. Pai. 2011. Frame-based elastic models. ACM Trans. Graph. 30, 2 (2011), 15:1–15:12.

Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space physics. ACM Trans. Graph. 31, 4 (2012), 72.

Bruno Hendelberger, Matthias Teschner, Richard Keiser, Matthias Müller, and Markus H. Gross. 2004. Consistent penetration depth estimation for deformable collision response. In Proc. WSCG.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2012. Interactive spacetime control of deformable objects. ACM Trans. Graph. (2012).

Alec Jacobson et al. 2018. gpgo Toolbox: Geometry Processing Toolbox. http://github.com/alecjakobson/gpgo Toolbox.

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popovic, and Olga Sorkine. 2012. Fast automatic skinning transformations. ACM Trans. Graph. 31, 4 (2012), 77:1–77:10.

Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. 2014. Skinning: Real-time Shape Deformation. In ACM SIGGRAPH 2014 Courses.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic coordinates for character articulation. ACM Trans. Graph. 26, 3 (2007), 71.

Ladislav Kavan, Steven Collins, Jiri Zara, and Carol O’Sullivan. 2008. Geometric skinning with approximate dual quaternions blending. ACM Trans. Graph. (2008).

Ladislav Kavan and Olga Sorkine. 2012. Elasticity-inspired deformers for character articulation. ACM Trans. Graph. 31, 6 (2012), 196:1–196:8.

Meekyoun Kim, Gerard Pons-Moll, Sergi Pujades, Seungbae Bang, Jinwook Kim, Michael J Black, and Sung-Hee Lee. 2017. Data-driven physics for human soft tissue animation. ACM Trans. Graph. 36, 4 (2017), 1–12.

Theodore Kim, Fernando de Goes, and Hayley N. Iben. 2019. Inverse kinematics for inversion-safety and element rehabilitation. ACM Trans. Graph. (2019).

Theodore Kim and Doug L. James. 2011. Physics-based Character Skinning using Multi-Domain Subspace Deformations. In Proc. SCA. 63–72.

Martin Komaritzan and Mario Botsch. 2018. Projective Skinning. ACM Proc. Comput. Graph. Interact. Tech. 1, 1 (2018), 12:1–12:19.

Martin Komaritzan and Mario Botsch. 2019. Fast Projective Skinning. In Proc. SIGGRAPH. 63–72.

Shazar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated quadratic approximations for surface fitting and optimization. ACM Trans. Graph. 35, 4 (2016), 134:1–134:11.

Years Kozlov, Derek Bradley, Moritz Bächer, Bernhard Thomaszewski, Thabo Beeler, and Markus H. Gross. 2017. Enriching Facial Blendshape Rigs with Physical Simulation. Comput. Graph. Forum 36, 2 (2017), 75–84.

Caroline Labrousset, Marie-Paule Cani, and Bruno Arnaldi. 2005. Dynamic skinning: adding real-time dynamic effects to an existing character animation. In Proc. SGGC.
