Adding dynamics to sketch-based character animations

Martin Guay, Rémi Ronfard, Michael Gleicher, Marie-Paule Cani

Adding dynamics to sketch-based character animations

Martin Guay[†1] Rémi Ronfard[1] Michael Gleicher[2] Marie-Paule Cani[1]

[1]Université de Grenoble
INRIA, LJK

[2]University of Wisconsin
Madison

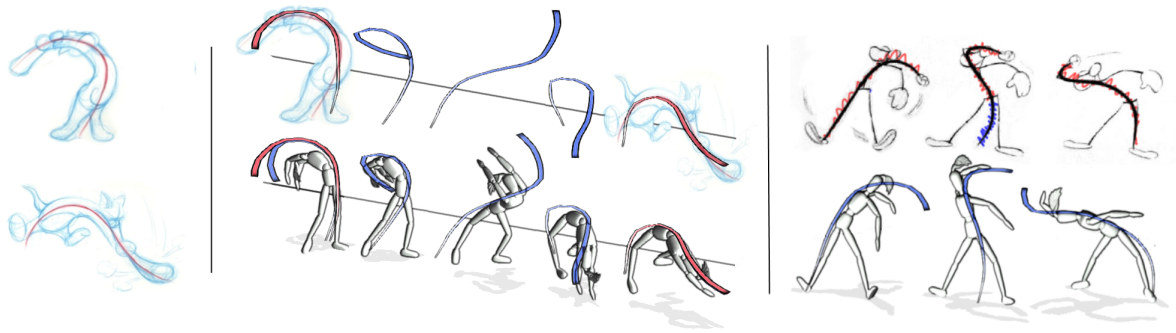

Figure 1: *Artists typically sketch lines of action to convey motion. In the artist sketch on the left, we can almost "feel" the dynamics of the line, how it seems to be storing potential energy. Perhaps we are perceiving, or inferring the character's main muscle contractions. Inspired by these drawings, we devised a physics-based model for the line's motion which allows the animator to unleash the line physically while interpolating between sketched lines of action, while exhibiting anticipation and follow through (middle sequence). The moving line may require changing body parts dynamically, which our 3D character motion synthesis can take into account (right figure). See also Fig.3 and Fig.5. Left cartoon illustration ©The Estate of Preston Blair.*

Abstract

Cartoonists and animators often use lines of action to emphasize dynamics in character poses. In this paper, we propose a physically-based model to simulate the line of action's motion, leading to rich motion from simple drawings. Our proposed method is decomposed into three steps. Based on user-provided strokes, we forward simulate 2D elastic motion. To ensure continuity across keyframes, we re-target the forward simulations to the drawn strokes. Finally, we synthesize a 3D character motion matching the dynamic line. The fact that the line can move freely like an elastic band raises new questions about its relationship to the body over time. The line may move faster and leave body parts behind, or the line may slide slowly towards other body parts for support. We conjecture that the artist seeks to maximize the filling of the line (with the character's body)—while respecting basic realism constraints such as balance. Based on these insights, we provide a method that synthesizes 3D character motion, given discontinuously constrained body parts that are specified by the user at key moments.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

† lemailamartin@gmail.com

1. Introduction

While free-form animation tools have evolved greatly over the last decades—especially to create poses—it remains a challenge to create expressive character animations. Lines of action are often used by artists to create more "dynamic" poses. If we look at the cartoons on the left of Fig.1, we can almost see and feel the motion from the drawing alone—the same way we can visualize motion between panels in comics. Our goal in this work is to extract dynamics and motion from simple abstract line drawings, similar to those on the left of Fig.1. One way to approach this would be to utilize data and build prior motions. It is true that our perception of motion is largely due to the strong motion priors we develop over the course of our lives, which allows us to infer motion from simple abstract information such as lines. However, in the case of expressive (e.g. cartoon) motions that may be be exaggerated or stylistic, the example-based solution may not be sufficient.

In this paper, we take a simple view of the line's dynamics that will allow physically interpolating between different drawn strokes, while exhibiting desirable motion features such as anticipation and follow-through [Las87]. By viewing the line's shape as storing potential energy with respect to a target shape, we can derive simple elastic models for the line's motion. Our approach allows the user to specify keystrokes as the target shapes over time, and we provide a re-targeting method that ensures a matching of the strokes over time.

A physically simulated line leads to new questions about the nature of the line's connection to the body over time. When the line can move freely, like an elastic band, how should the body be driven by the line? Even before computers were widely available, animators designed continuous line motions as to provide rhythm and directness in the motion. If we pay close attention to the body in Fig.4, we can see different parts joining and departing from the line.

Understanding how these should be triggered is a challenge. It could be a purely artistic choice. In this work, we take a first step into automating these dynamic shifts in constrained bones, and provide a method that can solve for character motion matching the line, while smoothly taking into account discontinuous bone constraints. With our method, the user sets the constrained bones at key moments (red squares in Fig.4), and the system automatically solves for the common subset of bones, while smoothly interpolating the parts that are not being constrained.

In short, this paper contributes two elements:

- A physically-based line of action interpolation method.
- A dynamic skeletal line matching algorithm that transitions between discontinuously constrained bones.

2. Related work

Many recent works on sketch-based character animation have focused on devising more natural interfaces for *posing*. Stick figures—an almost universally accepted abstraction of humanoids—have been proposed as a more natural way of specifying poses [DIC*03, LIMS10, WC11], and retrieving motion clips [CYI*12]. However, stick-figures do not help ensure a smooth reading of the body (*flow*) and can hold stiff corners. Smooth curved strokes can increase the aesthetics of limb deformations [KG05], and [OBP*13], in which the line of action is used to specify partial body parts. To provide the pose with a coherent and aesthetically pleasing overall flow in a given view, a single line of action is used to pose a character in [GCR13]. However, their method does not provide dynamics for the line and cannot dynamically transition between constrained body parts.

In this paper we synthesize 3D motion from 2D dynamic sketches. The problem of recovering 3D curves from their projections in a single 2D view is an ill-posed problem in general. Even when the 2D structure of the character is provided, e.g. with a stick figure, recovering a 3D pose without data is challenging. Taylor [Tay00] has looked at the problem of computing 3D joint positions of real human actors from their position in one camera, making use of higher-level knowledge about the human shape. In [GCR13], depth ambiguities are resolved, when posing a character from a 2D line of action, by constraining the transformations to the viewing plane. We relax this constraint to an "action plane" which can be rotated by the user—allowing for better staging.

Creating motion based on physics is a powerful tool in computer animation, as it allows realistic interactions with the environment. Early works provided forward simulations of elastic materials [TPBF87]. To create articulated motion of characters, the forward simulation rapidly falls short, as coordinated character motion requires control. Early works [HWBO95] propose state machines and simple elastic models for the bones based on pre-defined keyframe targets. Instead of synthesizing the motion via simulations, Neff et al. [NF03] warps within the parametric space of Hermite splines (interpolating the keyframes) to generate animation principles such as overlap and anticipation. While this approach is robust, it does not allow interactions with the environment. Instead, we take a similar route to Hodgins et al. [HWBO95], where in our case, the user specifies the target line shapes via sketches. In contrast to previous works, we focus on the motion of a sketched 2D abstraction of the character and provide a way to ensure continuity between keystrokes, without being restricted to periodic movements or predefined motions.

3. Physically-based line of action

To realize our approach, we consider two separable components. The first is the creation of the 2D line movement

by sketching and physically-interpolating strokes. The second is a 3D motion synthesis procedure, that solves for the character's skeletal motion as to match the 2D line's motion, while taking into account discontinuous bone constraints.

3.1. Stroke representation

In this section, we describe how we represent the line, derive an elastic model for its motion, and describe how to ensure continuity across keystrokes specified by the user. When the user draws a stroke, we record a set of 2D samples in screen space and then build a spatially smooth stroke denoted $\mathbf{x}_{stroke}(s)$. Discretizing elastic behaviors on the cartesian coordinates of the strokes often leads to spurious motions. Instead, we fit a set of piecewise rigid elements to the strokes, as to represent the line with a set of angles and lengths, similarly to [SG92]. The root's position is fixed at the center of *two chains*.

To transfer the recorded stroke $x_{stroke}(s)$ into the coordinates of the piecewise rigid chain, we first perform a constant length parametrization of the curve. Then we estimate the length of the chain's nodes $l = \int_s \frac{1}{N} \frac{\partial x_{stroke}(s)}{\partial s}\, ds$, which is the same for each rigid element j, given a number of elements N (10 in our case). Then we place the chain's root at the center of the curve $\mathbf{x}_{root} = x_{stroke}(0.5)$, to finally fit the angles γ_j to match the stroke $x_{stroke}(s)$ in screen space:

$$\min_{\gamma_j} \sum_{j}^{N} \left\| \mathbf{x}_{stroke}(l*j) - \mathbf{x}_{loa}(s_j) \right\|^2 , \qquad (1)$$

where $\mathbf{x}_{loa}(s_j)$ is computed with the linked hierarchy of angles γ_j, and lengths $l_j = l \; \forall j$.

Note that to simply interpolate the 2D strokes, we interpolate the angles and lengths separately using Hermite cubic splines, and then reconstruct the curve via forward kinematics, resulting in a dynamic line of action $\mathbf{x}_{dloa}(s,t)$.

3.2. Physically-based stroke interpolation

In this section, we introduce a physically-based stroke interpolation method that will generate richer motion that linear interpolation of strokes. The key to our method is the idea that some lines of action would be storing potential energy (e.g. Fig. 2) that could be unleashed into momentum, like an elastic. This is perhaps caused by our perception of the muscle contractions along the line. To realize this concept, we model the LOA as an elastic and the propose a method to ensure the dynamic line matches (interpolates) a series of keystrokes drawn by the artist—over time.

To obtain a certain behavior for the elastic line, we must know what forces to add. This necessarily involves the character's intention, which is hard to infer from a single stroke. As in early physics-based controllers (e.g. [HWBO95]) we

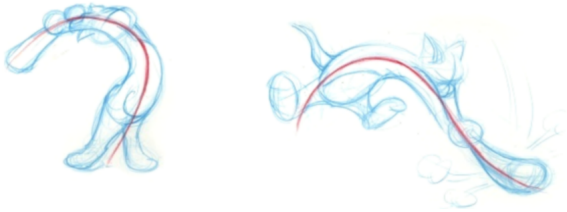

Figure 2: *In this example, we can almost "feel" the dynamics of the line, how it seems to be storing potential energy. Perhaps such shapes trigger our perception of the character's main muscle contractions. Inspired by this artist drawing, we devised a physics-based model for the line's motion which allows to unleash the line as an elastic, as well as to physically-interpolate between consecutive keystrokes. See Fig. 3 for more details. Cartoon image from Preston Blair's book Cartoon Animation [Bla94], ©The Estate of Preston Blair.*

can define forces based on a target (or reference) shape by defining an elastic potential energy as the difference in angles between two strokes. In other words, the animator could draw the target (reference) shape which will provide the goal of the elastic, and thereby the forces to add at any given time.

With this tool the animator can animate the line by interacting with its target shape, i.e. by sketching it. However, this form of interaction will only allow the elastic line to wobble around the reference stroke. In practice it can be useful to control the animation by having the elastic line pass through each stroke—while anticipating, overlapping and following through. To realize this, we use the forward simulation of the elastic line starting at the first keyframe and "shooting" for the next target keystroke—while a re-targeting method ensures the simulated frames land on the intended keyframe.

The simulation starts at a keyframe $t_{i=0} = t_k$, and moves forward over a set of frames $t_i = \{t_0 = t_k, t_1, t_2, ..., t_{M-1} = t_{k+1}\}$. To have the simulated frames hit the next keyframe at time t_{k+1}, we start by finding the closest *simulated* frame:

$$t_i^* = \underset{t_i}{\operatorname{argmin}} \sum_{j}^{N} \|\gamma(t_i) - \gamma(t_{k+1})\|^2. \qquad (2)$$

Once we have the closest matching frame t_i^*, we compute a correction for the frames before t_i^*. This is done by measuring the offset to the next keyframe (t_{k+1}), and interpolating between 0 and the offset for all the frames between, resulting in the correction: $\Delta\gamma(t_i) = (1-t)\,(\gamma(t_{k+1} - \gamma(t_i^*))$, which is applied to all the frames before $\gamma(t_i^*)$.

In practice, we use $M = 40$ frames for the simulation between the keyframes. Remembering that the lines are connected chains, we must determine the motion of not only

the angles γ, but also the root $\mathbf{x}_{root}$. Adding elastic forces based on the difference in position between two strokes does not behave realistically in the case of the root. Hence for the root position, we simply interpolate the position between keyframes.

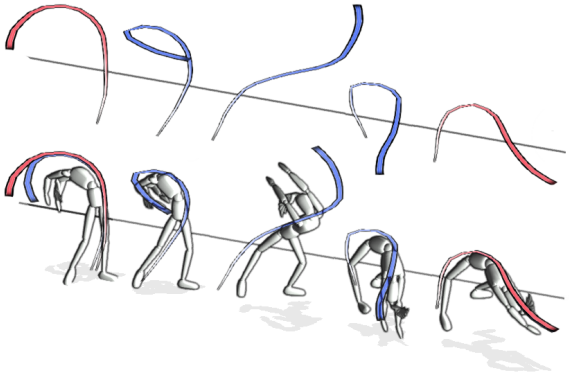

Figure 3: *The lines in blue are frames of the physically interpolated line of action, generated between two drawn line of action strokes in red. Note how the motion exhibits anticipation and follow through, also shown in the video. Our dynamic posing in the second row, ensures the character matches the line's shape in between the keyframes of the line. Note that our character motion synthesis (described in Section 4) solves for motion in the action plane which is at an angle from the viewing plane.*

t_i	time of frame i
t_k	time of keyframe k
q_j	local orientation of bone j
Q_j	global orientation of bone j
x_{root}	root of the kinematic tree
$x(s)$	interpolated position of a kinematic chain
x_j	position of bone j
$*(t)$	interpolated trajectory of d.o.f. *
$\Omega(t_k)$	set of bones constrained at keyframe t_k
$\Omega(t)$	$\Omega(t) = \Omega(t_k) \cap \Omega(t_{k+1})$ for a time $t \in]t_k, t_{k+1}[$

Table 1: *Notation.*

4. Synthesizing 3D character motion with dynamic bone shifts

In this section, we describe how to match a 3D character to the moving line. Following the definition of the line of action in [GCR13], the following should hold: at any instant in time t, a set of bones in the character's body should match the shape of the line—the shape being first derivatives or tangents. To have the character be close to the moving LOA at time t, we proceed as in [GCR13] and minimize the average distance between both lines:

$$\min_{\mathbf{x}_{root}(t),\mathbf{Q}(t)} \quad \int_{s\in\Omega(t)} \|\mathbf{P}\mathbf{x}(s,t) - \mathbf{x}_{loa}(s,t)\|^2 \, ds,$$

$$\text{subject to} \quad \frac{\partial \mathbf{P}\mathbf{x}}{\partial s}(s,t) = \frac{\partial \mathbf{x}_{loa}}{\partial s}(s,t), \; \forall s \in \Omega(t).$$

where $\mathbf{x}(s,t)$ is the character's skeleton position computed with forward kinematics of the $\mathbf{x}_{root}(t)$ root trajectory and the set of orientations trajectories $\mathbf{Q}(t) = \mathbf{q}_0(t), ...\mathbf{q}_{N-1}(t)$ of the constrained bones $\Omega(t)$. The matrix $\mathbf{P}$ is a view and perspective projection matrix in homogenous coordinates.

In [GCR13], the tangents-matching constraint is turned into a soft constraint. Because this problem is under-constrained, they introduce a viewing plane constraint which is met by construction: the bones' rotations are parameterized with a single angle $\theta(s)$ around the viewing direction (detailed bellow). Finally, they solve the problem using local nonlinear optimization.

Unfortunately, applying this method sequentially to each frame of the animation is problematic. We found that convergence between consecutive frames can be inconsistent, leading to visual jumps in the animation. Secondly, it does not take into account dynamic constraints for the set of bones being attached to the line, as in Fig. 5.

To obtain a smooth motion, we formulate the dynamic matching problem as a space-time optimization problem minimizing a temporal smoothness objective E_s with a (soft) dynamic shape matching constraint E_{dloa} (for dynamic line of action), assuming a dynamic set of bones $\Omega(t)$ to be described bellow:

$$\min_{\mathbf{x}_{root}(t),\mathbf{Q}(t)} \int_t E_{\dot{x}}(t) + E_{dloa}(t)\, dt.$$

Dynamic bone sets. We will denote the dynamic set of constrained bones as $\Omega(t)$. We observed (see Figure 5) that changes in the set of bones $\Omega(t)$ happen at key moments, and that between these key moments, what is constrained is the common subset of bones. That is, assuming these changes occur at the keyframes, we have between keyframes $\Omega(t) = \Omega(t_k) \cap \Omega(t_{k+1})$ for a time $t \in]t_k, t_{k+1}[$.

Smoothness. The smoothness term penalizes the difference between consecutive frames t_i. However, we do not wish to smooth out the motion of all the bones, but only those that are constrained by the line:

$$E_{\dot{x}}(t) = \sum_{j\in\Omega(t)} \|log(\dot{\mathbf{q}}_j(t))\|^2 + \|\dot{\mathbf{x}}_{root}(t)\|^2, \qquad (3)$$

where $\dot{\mathbf{q}}_j(t) = \mathbf{q}_j^{-1}(t_{i+1})\mathbf{q}_j(t_i)$, log converts an orienta-

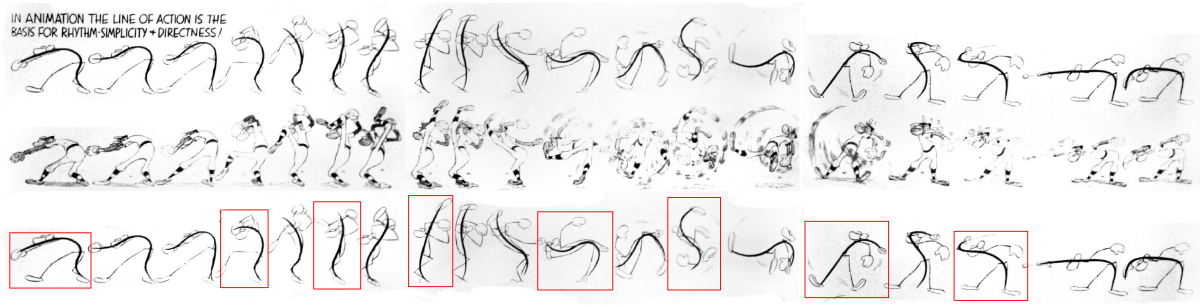

Figure 4: *The animator designs a fluent line motion, and different parts of the body are attached to the line as it moves. In terms of body attachments, we hypothesize that artist seeks to maximize line fulfillment, while respecting basic principles of articulated motion such as balance. In this paper, we take a first step and observe that transitions happen at key moments (red squares), and that between the key moments, it is the common subset of bones from the key moments that remain constrained. Illustration from the book Cartoon Animation [Bla94],* ©*The Estate of Preston Blair.*

tion into a 3D angle vector, and $\dot{\mathbf{x}}_{root}(t)$ is the time derivative of the root trajectory.

Shape-matching constraint is turned into a soft constraint, resulting in the energy function:

$$E_{loa}(t) = \int_{s\in\Omega(t)} w_{\mathbf{x}} E_x(s,t) + w_\partial E_\partial(s,t)$$

$$E_x(s,t) = \|\mathbf{Px}(s,t) - \mathbf{x}_{dloa}(s,t)\|^2$$

$$E_\partial(s,t) = \left\| \frac{\partial \mathbf{Px}}{\partial s}(s,t) - \frac{\partial \mathbf{x}_{dloa}}{\partial s}(s,t) \right\|^2$$

where s is the space that covers each constrained bones in $\Omega(t)$.

Staging and solving for motion. In [GCR13], the problem is reduced by constraining the bone rotations to a single angle, where the angle rotates the bone in the viewing direction of the camera. It is true that many motions meant to be watched do happen in a plane nearly parallel to the viewing plane. However, the staging of the animation, can be at an angle from the viewing plane. More generally, we will refer to the plane in which the animation lies as the *action plane*, and the user is allowed to rotate it. For example, consider Fig. 3 where the character's motion lies on plane at angle from the viewing plane. This does not mean the skeleton is planar, only the transformations of the joints are 1 dimensional and rotating around the action plane's normal.

We solve for the angles controlling an axis-angle rotation in a direction $\vec{\mathbf{n}}$ (normal to action plane), and angle θj, for every bone j. The bone orientation of the character is thus defined as: $q_j^{'} = q_j exp(\theta_j\, \vec{\mathbf{n}})$. When the problem is solved, we set the new bone orientation to the previous one $q_j \leftarrow q_j^{'}$, and the angle θ_j to 0. The root position is parametrized as $\mathbf{x}_{root}^{'} = \mathbf{x}_{root} + w_u\vec{u} + w_v\vec{v}$ where $\vec{u}$ and $\vec{v}$ are orthogonal vectors in the plane. We optimize with respect to w_u and w_v, and when finished set the new value of $\mathbf{x}_{root}$ to $\mathbf{x}_{root}^{'}$.

The physically-based line of action will hold rich motion between keyframes. Solving for the character's pose only at the keyframes and interpolating the configuration of the character (the orientations and root) does not make the character match the elastic line between keyframes. The problem is that solving for several frames between the keyframes leads to inconsistencies between frames—even when using smoothing.

To obtain a closer match that remains smooth, we use a simple form of motion reduction: we use Hermite spline interpolation for the trajectories and optimize over this reduced set of degrees of freedom. We use one node for each keyframe stroke, and subdivide once between keyframes resulting in $2N-1$ nodes, where N is the number of stroke keyframes. This scheme provides the extra liberty necessary to match the physically-based line of action, while remaining smooth.

Each frame is initialized with an initial pose, we solve (4) over the whole set of frames using local non-linear optimization. In the next section, we detail how to add 3D twist and contact constraints.

4.1. Additional 3D constraints

In the previous section, we solved for a planar 3D character motion. With this approach it is quite difficult to represent and control aspects of the 3D motion such as 3D twist and contacts with the environment. In this section we extend our space-time optimization framework to consider 3D constraints for twist and contacts. This is done by adding new terms to the optimization problem (4) reflecting twist along

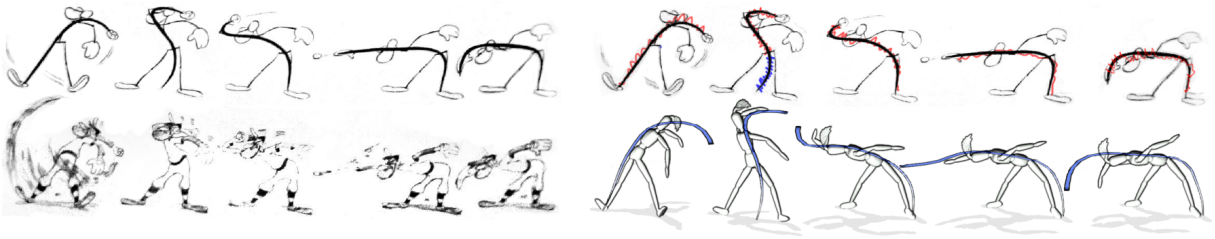

Figure 5: *We scribbled in red the parts of the line that are constraining bones, and in blue the parts that are not. Our motion synthesis allows the line to transition between the right leg and the left leg. Cartoon image from Preston Blair's book Cartoon Animation [Bla94], ©The Estate of Preston Blair.*

the line, which we denote $E_{twist}(t)$, and $E_{\perp}$ for contacts. But first, let us describe the workflow from the animator's perspective.

Work flow. In our system the character has a rig that includes twist and contact controllers. The rigs are similar to those setup by animators in Maya and Blender: they are 3D objects that visualize and interface the constraints (twist and contacts in our case). The twist rig—represented by can-shaped primitives along the line—controls a single angle around the line. The animator rotates the cans by clicking and dragging. The contacts are represented by rectangles the animator can activate and displace by clicking and dragging, while the orientation is manipulated using a rotation widget. The twist angles, as well as the contacts' positions and orientations are interpolated to define trajectories and constrain the character's motion via the terms $E_{twist}(t)$ and $E_{\perp}$.

The twist term. The user controls twist widgets along the line which control a single twist angle that does not affect the shape of the line—only the orientation of the character. Each *twist can* constrains the orientation of a single bone (e.g. upper spine and pelvis), and all constraints χ are added into a penalty:

$$E_{twist}(t) = \sum_{j \in \chi} \|\mathbf{Q}_j^{-1}(t)\mathbf{q}_j^{can}(t)\|^2,$$

where $\mathbf{Q}_j(t)$ is the orientation of the bone j, which is not to be confused with the local bone orientation $q_j(t)$.

The contacts terms. The contacts are 3D rigid constraints (position and orientation) for a single bone (e.g. foot or hand). The rigid transformations of the contacts are interpolated to produce trajectories the character's pose seeks to satisfy while preserving the moving line's shape. All the contacts are summed into a penalty:

$$E_{\perp}(t) = \sum_{j \in \Upsilon} w_x \|\mathbf{x}_j(t) - \mathbf{x}_j^{\perp}(t)\|^2 + w_q \|\mathbf{Q}_j^{-1}(t)\mathbf{q}_j^{\perp}(t)\|^2,$$

where Υ is the set of contacts, and $\mathbf{x}_j^{\perp}(t), \mathbf{q}_j^{\perp}(t)$ are the position and orientation trajectories of the contacts.

Putting it all together. Because the additional constraints are 3D, they may not lie on the *action plane* described in the previous Section (4). Hence, our final matching procedure is broken down into two steps: we first initialize the motion and solve for planar motion (Section 4), then we add the 3D constraints—the terms (4.1) and (4.1)—to solve for full 3D bone orientations.

5. Results and discussion

We implemented the methods presented in this paper in a self-contained program. The sketch-based interface allows quickly drawing strokes and creating dynamic lines of action (Section 3). The user specifies the bones that are to be constrained at key moments by selecting from a set of predefined bone sets. Our system then automatically creates a smooth 3D motion that satisfies the discontinuous constraints (Section 4). The user can also edit secondary parts of the character that are not concerned by the main line by drawing strokes (as in [GCR13]), and can edit contact and twist widgets over time (Section 4.1) by clicking and dragging on different controllers.

We used the interface ourselves to generate the figures in the paper, and the animations in the accompanying video. We used the physics interpolation method (Section 3) to create Fig. 3. Although it is possible to create interesting motions, it requires fine tuning the stiffness parameters of the angles over time as to obtain the right behavior. Once the parameters are set, it is only a matter of sketching a few strokes to create different versions of the motion.

We asked a few users to test this method in order to obtain feedback. We observed that many users found the idea of sketching keystrokes to specify the dynamics not necessarily intuitive. Some would have preferred drawing strokes as forces, or to grab the line in order to wobble it as an elastic. These are interesting insights into how physics simulations could be used interactively to create movements.

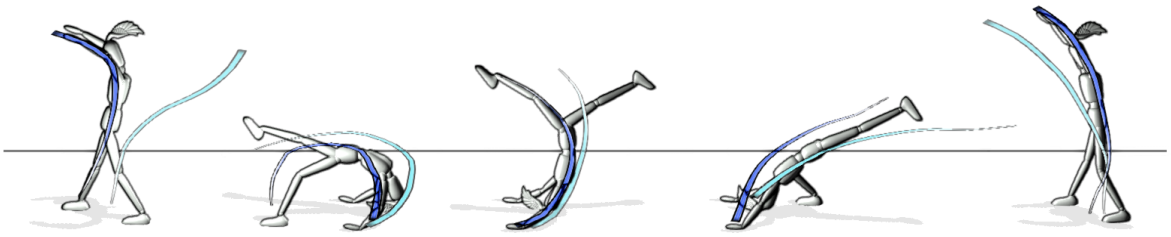

Figure 6: *The backwalk in gymnastics is an example of a motion that holds a continuous line of action throughout the animation. The first pose in this figure has the right leg constrained, then the left leg for the 3 subsequent poses, and finally the right leg again at the last pose.*

We then tested our method to deal with dynamic bone constraints (Section 4). Note that in these results, the line's motion was produced with basic geometric interpolation of the strokes. The animation in Fig.5 requires setting a set of bones at a first key moment for the right leg, and setting a second set of bones—holding the right leg—at an other key moment. In our implementation, the user selects the body parts from a set of pre-defined bone sets by scribbling a stroke over a set of bones. For the moment, it does not allow re-producing the long sequence in Fig. 4, where individual bones lie in the middle of the stroke. We used the same procedure to create Fig. 6—which transitions from the right leg, to the left leg, and back to the left leg. Note that a few additional results in the video show these capacities, while satisfying 3D contact constraints (Section 4.1).

Limitations and future work

We explored the idea of using a physically-simulated line of action that would mimic the action of the character's internal forces. Our experiments showed it could produce desirable features such as anticipation and follow-through. However, we found that different users could expect different "behaviors" for the line, and we observed that fine tuning the stiffness parameters over time to obtain a given behavior was quite long. We think this area could be improved with some training of the parameters, perhaps from cartoon videos.

While our method allows squashing and stretching the strokes, our 3D motion synthesis algorithm based on [GCR13] solves only for piecewise rigid motion, and therefore, does not allow squashing and stretching the character. Additionally, the solver requires a fair amount of smoothing, realized by adding a smoothness term, and representing the motion with a low-dimensional spline interpolation scheme—which in consequence, prevents from an exact match between the character and the line. Recently an exact algorithm that support squash and stretch is introduced in [GRGC15].

We made a first step towards allowing dynamic transitions between body parts fulfilling the line over time. For the moment, the user specifies the constraints at key moments, but we foresee a future where these transitions could be inferred automatically. For instance, we could have approached the problem from a different direction: by using a controller for a physically-simulated 3D character, and adding the line's shape matching as an additional objective as to provide the character with an aesthetically pleasing line whenever possible. Information from the character's 3D dynamics, such as balance and foot placements could be used to determine when the legs should join or depart from the line.

We demonstrated our concept only with articulated humanoid characters. Our methods should apply to other morphologies such as quadrupeds. In the future, it would be intersecting to investigate ways of fulfilling the motion with more abstract matters—such as fluids—that dynamically change topology.

Conclusion

By viewing the line of action as an elastic, the animator can create energetic movements for 3D character by sketching strokes. To realize this, we provide a 3D motion synthesis procedure that supports smoothly changing body parts that are being driven by the elastic line over time. While the physically-based interpolation method requires fine-tuning stiffness parameters, we were able to reproduce cartoon motions. In particular, we demonstrated its effectiveness for producing anticipation and follow-through. Future work will further investigate other animation principles, and interactions with the environment.

Ackowledgements

We thank Estelle Charleroy for help with video editing and Deidre Stuffer with text editing. We also thank the anonymous reviewers for their useful comments and suggestions. This work was partially funded by the ERC advanced grant no. 291184 *EXPRESSIVE* from the European Research Council (ERC-2011-ADG 20110209) and NSF award NRI-1208632.

References

[Bla94] BLAIR P.: *Cartoon Animation*. Walter Foster, 1994. 3, 5, 6

[CYI*12] CHOI M. G., YANG K., IGARASHI T., MITANI J., LEE J.: Retrieval and visualization of human motion data via stick figures. *Computer Graphics Forum 31*, 7 (2012), 2057–2065. 2

[DIC*03] DAVIS J., IGARASHI M., CHUANG E., POPOVIC' Z., SALESIN D.: A sketching interface for articulated figure animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), SCA '03, pp. 320–328. 2

[GCR13] GUAY M., CANI M.-P., RONFARD R.: The line of action: an intuitive interface for expressive character posing. *ACM Transactions on Graphics 32*, 6 (2013). 2, 4, 5, 6, 7

[GRGC15] GUAY M., RONFARD R., GLEICHER M., CANI M.-P.: Space-time sketching of character animation. *ACM Transactions on Graphics* (2015). 7

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), SIGGRAPH '95, pp. 71–78. 2, 3

[KG05] KHO Y., GARLAND M.: Sketching mesh deformations. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (2005), I3D '05, pp. 147–154. 2

[Las87] LASSETER J.: Principals of traditional animation applied to 3d computer animation. *ACM SIGGRAPH Computer Graphics 21*, 4 (1987), 35–44. 2

[LIMS10] LIN J., IGARASHI T., MITANI J., SAUL G.: A sketching interface for sitting-pose design. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium* (2010), SBIM '10, pp. 111–118. 2

[NF03] NEFF M., FIUME E.: Aesthetic edits for character animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), SCA '03, pp. 239–244. 2

[OBP*13] ÖZTIRELI A. C., BARAN I., POPA T., DALSTEIN B., SUMNER R. W., GROSS M.: Differential blending for expressive sketch-based posing. In *Proceedings of the 2013 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2013), SCA '13, pp. 155–164. 2

[SG92] SEDERBERG T. W., GREENWOOD E.: A physically based approach to 2–d shape blending. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 25–34. 3

[Tay00] TAYLOR C. J.: Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Comput. Vis. Image Underst. 80*, 3 (2000), 349–363. 2

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 205–214. 2

[WC11] WEI X., CHAI J.: Intuitive interactive human-character posing with millions of example poses. *IEEE Comput. Graph. Appl. 31*, 4 (2011), 78–88. 2