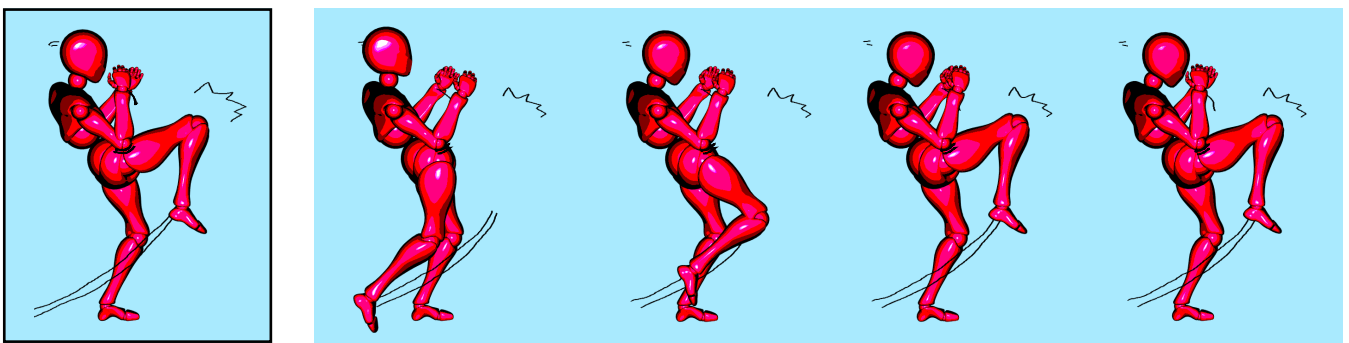


# SMACC: Sketching Motion for Articulated Characters with Comics-based annotations

Amandine Legrand<sup>1,2</sup>, Amal Dev Parakkat<sup>1</sup>, Damien Rohmer<sup>2</sup>

<sup>1</sup> LTCI - Télécom Paris, Institut Polytechnique de Paris

<sup>2</sup> LIX - École polytechnique, CNRS, Institut Polytechnique de Paris



**Figure 1:** A fight animation generated with SMACC. **Left:** input sketches, with trajectory, circumfixing and impact lines. **Right:** the resulting animation.

## Abstract

We introduce SMACC, a sketch-based system for animating short sequences of 3D articulated characters inspired by 2D comic motion line annotations. SMACC relies on classical rules of motion depiction used in comic books, allowing the depiction of dynamism in static images while being universally understood. Building on this, SMACC introduces an algorithmic interpretation of these principles in the context of a 3D character animation, guided by three fundamental types of motion lines: trajectory, circumfixing and impact. The adaptation to rigged 3D characters relies on the automatic computation of how these motion cues spatially influence the character's skeleton, achieved through a global analysis of sketch annotations relative to the character's pose. The resulting animation is generated by encoding the kinematic clues and constraints into joint angular velocities. Finally, the proof-of-concept demonstrated by SMACC is validated through a user study, which evaluates the effectiveness and accuracy of this sketch-based approach applied to 3D character animation.

## CCS Concepts

• *Human-centered computing* → *Interactive systems and tools*; • *Computing methodologies* → *Animation*;

## 1. Introduction

Comic books rely on long-employed techniques, such as motion lines, to very effectively convey the notion of dynamics on top of statically depicted shapes using only a few sparse visual clues. These are typically depicted in the form of sketched curves and lines and have be-



come an integral part of comic storytelling, enhancing the reader's perception and interpretation of action sequences and dynamic events within the narrative. By strategically placing these motion lines, artists can communicate the trajectory, speed, and intensity of movement, thereby enriching the overall storytelling experience. The inset figure (Courtesy, the cover image of CACM 1993, from Hsu et al. [HLW93]) illustrates such an example where a dynamic scene is represented as a static illustration.

In recent years, research on expressive 3D animations has been developing various techniques, including sketch-based approaches

[GRGC15b, DSC\*20, ABB\*24], to provide an intuitive way to create and edit arbitrary animated virtual shapes. These animation models typically consider that the controllers of the models are following the sketched space-time trajectory, thus requiring several sketch inputs when the number of degrees of freedom increases. Interestingly, the use of sparse inputs following the 2D comic book rules to control a complete animated character has not yet been fully studied. Indeed, generating a coherent motion of a full 3D rigged character using only sparse 2D-like sketch inputs is difficult, as the degree of freedom of the character is much larger than the number of inputs.

The main goal of this paper is to explore and study the effectiveness of using comic-inspired sketching techniques via motion lines to animate a complete 3D rigged character. We typically aim at animating a short sequence of the character for a time window equivalent to a comic frame, using only a few sparse sketched curves. The sketches are then able to depict the character's motion in a static pose while being able to guide the generation of a full 3D animated sequence.

To this end, the three following sub-objectives have to be tackled. First, the 2D motion lines should be accurately interpreted to understand the desired motion and its associated kinematic parameters. Second, the motion lines should establish appropriate spatial correspondence between motion lines and parts of the rigged character. Finally, while using 2D motion in a complete 3D space, visual coherence across arbitrary viewpoints should be maintained.

The key idea of our approach is to couple a global analysis of the user-sketched motion lines with the character rig projected in the view space, to interpret the motion lines as a set of motion primitives inspired by common practices in comic drawing based on perceptual studies [CM15]. In particular, we first propose to cluster the user-sketched motion lines into groups of lines acting as a single motion primitive to the same body part. Second, we compute the influence of each motion primitive on the character rig, with spatially limited influence depending on the character's skeleton and the distribution of other motion primitives. The actual animation is concretely defined by encoding angular velocity profiles at the joint level across the timeline with parameterised dynamics defining temporal properties such as duration and acceleration. Additionally, our system supports both local and global motion definitions, allowing users to specify movement at different levels of the character structure. A timing hierarchy further refines the animation by defining the order and dependencies among multiple motions.

In summary, we make the following contributions:

- Algorithmic interpretation of motion lines: A method to algorithmically interpret user sketches to extract kinematic parameters, facilitating the conversion of visual cues to motion dynamics
- Encoding motion primitives into joint animations: Motion primitives are translated to joint animations by encoding angular velocity profiles across the timeline with parameterised dynamics defining temporal properties
- Framework for animation from expressive inputs: A simple-to-use framework that facilitates the creation of animations using 2D comic-like expressive motion lines

Our approach preserves the expressive power of 2D motion lines

while helping extend its capabilities to 3D-rigged character animation. We evaluate the effectiveness of our methodology through a user study, which shows that the generated animation aligns with viewer-perceived motion and demands minimal technical expertise. Furthermore, we expand the framework to support sketching 2D motion lines from multiple 3D views, allowing interested users to create a variety of synchronised complex animations. These results demonstrate that 2D comic sketch rules can be used to animate short sequences of 3D rigged character models while preserving simple inputs accessible to novice users without prior expertise.

## 2. Related Work

We describe in the following the related research works developing sketch-based interfaces for animations.

**Sketch-based 2D animation:** Sketch-based 2D animation has evolved over time, transitioning from a traditional paper-based production pipeline to digital tools. Early research in this area primarily focused on using sketches as a direct medium for creating animations and transitioning them to digital media [HLW93, FBC\*95], allowing the artists to manipulate sketches digitally and paving the way for modern digital animation. However, as digital tools became more popular, subsequent works shifted their focus to developing tools to simplify and enhance the animation creation process. The main goal of many of these works was to make animation more accessible and effortless by automating many repetitive tasks. The works in this direction can be classified into two categories: methods that are used to directly manipulate the animation strokes and those where the sketches serve as hints to guide the animation of 2D imagery.

*Stroke manipulation techniques:* This category of methods focuses on directly manipulating animation strokes to simplify the animation process. This includes automatic computation of in-between frames where the user sketches only a few keyframes, and the correspondences between the keyframe strokes are computed automatically using stroke analysis [Kor02], user-assisted matching [WNS\*10], or prior knowledge [MGW24]. One of the primary challenges in this process is handling occlusions to ensure accurate and seamless animations. To address this, many studies proposed automatic completion of occluded contours by detecting occluded regions/strokes [CMB17, JSL22] and applying stroke completion techniques [EPB\*19]. Another line of work keeps track of the visibility of strokes [DRVDP15] during the motion by defining region contours [ZLWH16] or enabling interactive occlusion handling [EBB24]. Complementary work focuses on directly deforming the strokes or shapes to create the desired animations [MJ96], ARAP [IMH05], and its variants [SMW06, JBPS11] became foundational for such deformations as they preserve local geometric features during the process. Recent methods in this direction integrate learned priors to create animations by taking hints from physics [XZJJ25], videos [ZCXP24], texts [GVA\*24] and interaction [RKW\*24].

*Sketch-guided animation:* Here, the sketches act as guiding inputs to influence the animation of 2D imagery. In such motion sketching systems, the primary goal is to help users create interactive illustrations by specifying the desired motion behaviour. For in-

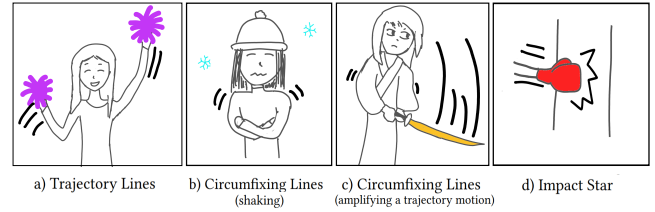
stance, K-Sketch [DCL08] allows users to sketch the global motion path that defines the trajectory an object must follow. Similarly, works like Guay et al. [GCR13] focus on local pose manipulation of the character. Instead of working directly on the character itself, as an additional layer, motion control over sketched similar objects is used in Draco [KCG\*14]. This work is then further improved by adding functional relationships between the group of animations [KCGF14] and by relying on physics-inspired flow particles [XKG\*16]. Additionally, some sketch-based systems aim to replicate traditional animation principles to create expressive 2D animations [KGUF16].

However, unlike the existing sketch-based animation approaches that primarily focus on refining and interpolating strokes in 2D space, we focus on utilising the 2D motion lines as a direct input for generating 3D animation. By treating motion lines as spatiotemporal movement cues (motion trajectories), we extend beyond the traditional 2D sketch-based animation and enable the conversion of hand-drawn motion cues to plausible 3D motion.

**Sketch-based 3D animation:** The works in this direction can be primarily classified into methods aimed at designing poses or animation.

*3D character posing:* Considering that common techniques involving the direct manipulation of joint angles or positions are tedious, existing techniques explored different ways of intuitively manipulating the character poses. Early works in this direction use 2D sketches to automatically pose stick figures, providing a simple interface for character posing [DAC\*06]. This approach is then extended to generic rigged 3D characters where the rigging parameters that align the character’s pose with user-defined strokes are computed [HMC\*15]. Techniques inspired by the line of actions [GCR13] further improved the expressiveness of these methods, facilitating intuitive control over 3D poses. Beyond skeletal representations and manipulations, researchers also explored the possibility of using silhouettes [KSVDP09], vector sketches [BVS16], and bitmaps [BB22] for inferring poses. While 3D posing focuses on intuitively manipulating poses, it mainly deals with static configurations of articulated characters. In contrast, we aim to extract and use motion cues from stylistic and expressive comic-style 2D motion lines to three-dimensional space and time.

*Motion editing:* Instead of explicitly defining styles, sketch-based motion editing techniques aim at providing intuitive tools that are capable of creating expressive animations. Early works in this direction used direct tools to manipulate the motion properties [NF03] and encode coordinated timing among movement features to enable motion editing through splines and displacement maps [CBSG08]. Another work worth noting is Dynamic Comics [CNKI13] that summarizes and structures 3D animation data in a comic style. To enhance expressiveness, the concept of line of action is extended by adding dynamics to it [GRGC15a] and is further improved by spacetime sketching [GRGC15b] that bridges the gap between pose and time editing of a character motion from a single user sketch. This approach is then extended to articulated characters by leveraging projective geometry to compute motion parameters from sketches treated as constraints [CiRL\*16]. This sketching of motion paths is further made easy by associating it with capture devices [CGNS17]. Recent advancements have incor-



**Figure 2:** Examples of Motion Cues

porated prior knowledge to create more natural and realistic motions [ABB\*24] and advanced systems that use a multi-conditional motion generator using 2D keypose, 2D trajectory and an action word [ZGX\*25]. Additionally, VR [GRC19, ZLFA24] and AR-based [YKSF20] techniques allow characters to perform predefined movements while following hand controllers or mobile devices, respectively.

### 3. User inputs

Our system starts with the user sketching a set of motion lines over a 2D view of a rigged articulated character composed of a hierarchical set of joints.

Following the standard cartoon motion depiction, we assume the posed character depicting the desired action is viewed from an angle where the view plane captures most of the motion. Based on the user-given motion cues, the character transitions to the given pose, from which subsequent movements emerge and unfold.

The proposed system considers three main sub-types of motion lines that we call *motion cues*, defined as follows:

- **Trajectory lines:** Strokes that are used to depict motion by drawing lines trailing behind an object to denote its movement. In other words, these lines visually encode the memory of the location of an object in the direction of its implied motion (See Fig. 2-a).
- **Circumfixing lines:** Strokes that are used to indicate small localised movements around a specific area. Interestingly, these lines can indicate two types of animation effects. Firstly, to indicate a local movement of a body part, typically corresponding to a local shaking effect as illustrated in Fig. 2-b, where they depict shivering shoulders. Secondly, they can emphasise the effect of a primary movement associated with trajectory lines acting on a different body part. For instance, in Fig. 2-c, the strokes suggest that the motion of the upper body is influenced by the sword, but it does not follow the same trajectory.
- **Impact stars:** Strokes that are used to amplify the effect of motion rather than explicitly depicting it. When placed at the endpoints of a trajectory, impact stars represent a sudden and forceful impact, reinforcing the sense of acceleration.

The type of motion cues associated with sketched lines is explicitly and manually selected by the user during the drawing session.

While additional types of motion cues can be found in cartoons - like future lines, backfixing lines, partial polymorphism or suppletion lines - they are either not commonly used for expressive

storytelling or can also be depicted as the same as trajectory lines, as indicated by Cohn and Maher [CM15].

#### 4. Global Strokes Analysis

Interpreting sketched motion lines in relation to an articulated character presents several challenges. First, the system must accurately associate 2D strokes with correct 3D joints while maintaining spatial consistency. Then, strokes describing similar motions should be grouped into clusters, ensuring a coherent propagation of movement. Finally, the conflicts between the overlapping motions must be resolved to preserve the naturalness.

Once the user sketches the desired motion cues on a 2D view of the model, these sketch strokes are analysed with respect to both the articulated character and the sketched view. Initially, our system performs a global analysis to cluster strokes and infer their associated articulated joints and kinematic parameters. Subsequently, the system computes the corresponding motion and identifies the joints involved along with their timings, all while satisfying the constraints imposed by other motion lines.

##### 4.1. Sketched Lines to Joint Association

Associating a sketched line with a joint presents the challenge of resolving 2D-to-3D ambiguity while maintaining spatial consistency. To achieve this, we first identify the closest joint in screen space by projecting all joint positions and computing the Euclidean distance between each projection and the sketch. The depth of this closest joint in the view space is then used to define a 3D plane, orthogonal to the view direction and passing by this point, where the sketched line is lifted, therefore setting its 3D embedding.

To be precise, we follow the observation that “*The visual representation of routes in motion events is more important than their start or endpoint, while the results also correspond with the idea that the starting point of paths is less important than the endpoint*” [DvE23], and compute the distances from each joint to the endpoint (for trajectory lines) or midpoint (for circumfixing lines and impact stars) of the strokes.

While most of the motion lines have a local influence on the body parts of the character, we also consider the special case of motion lines associated with global motion. This special case is explicitly set by the user and leads to a motion trajectory directly attached to the root joint. The motion line is then either interpreted as a continuous translation along the direction indicated by the sketch or as a spinning motion when the line forms a closed loop.

##### 4.2. Clustering Sketched Lines into coherent Motion Cues

As illustrated in Fig. 2, several sketched lines can actually be related to the same motion cue; as repeating lines is a common way to reinforce movement. Treating each sketched line individually would lead to sub-optimal results where some nearby lines could be associated with different joints and have possibly counter-acting actions. To avoid this, we propose a global analysis of the sketched lines to cluster them into coherent groups representing the same motion cue acting on the same character’s articulated structure.

To this end, we consider a set of lines to belong to the same cluster if they are of the same type and either share the same associated joint or have associated joints such that one is an ancestor of the other (up to the root) in the joint hierarchy.

Then, each cluster, defined by the sketch strokes it contains, is associated with the following parameters:

- A type (Trajectory or Circumfixing).
- A *StartJoint* and *EndJoint*, marking the start and the end, respectively, of a kinematic chain of joints which is influenced by this cluster.

Since an impact represents a deformation of a trajectory motion, such as a sudden stop, it is always linked to a trajectory cluster. Thus, any impact placed after a trajectory cluster, meaning its associated joint is a descendant of the joint in the trajectory cluster, will be connected to that cluster.

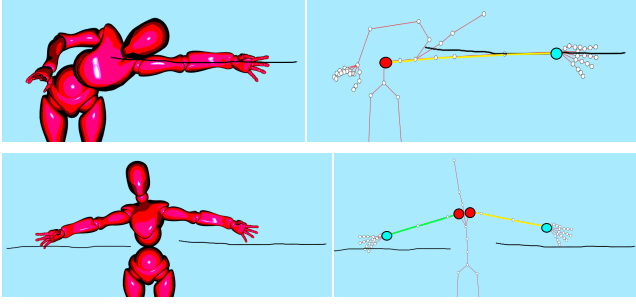
##### 4.3. Finding the Kinematic Chain

Defining the region of influence for a motion in an articulated character is challenging because it requires maximising movement for expressive exaggeration while avoiding conflicts between overlapping motions.

By default, as stated in THE ANIMATOR’S SURVIVAL KIT: “*In most big actions of the body, the start of the action is in the hips*” [Ric02], we consider the hip as the root for the humanoid-like character used throughout this paper. For that, we assume that the cluster’s kinematic chain begins at the direct child of the root joint that lies on the path to the associated joint and extends to it, and is considered as the *EndJoint*. This ensures that the motion propagates effectively along the hierarchy, preventing inconsistent inputs and enhancing expressiveness. However, when multiple motion lines associated with different clusters are applied, propagating all their influences up to the root joint may lead to conflicting motions.

To avoid this, we ensure that no joint belongs to more than one kinematic chain, by choosing an appropriate *StartJoint* that maintains consistency across all motions. The process of adjusting the *StartJoint* to an unambiguous one is outlined in Algorithm 1, where *clusters* is the set of trajectory and circumfixing clusters,  $N_{clusters}$  is the size of *clusters*, and  $IsJointAncestor(j1, j2, skeleton)$  checks if  $j2$  is an ancestor of  $j1$  in the skeleton hierarchy. An example of the result is shown in Fig. 3, where applying a motion line to a single hand causes a global bending of the character extending up to the root hips, whereas applying motion lines to both the right and left hands results in a more localised influence.

Additionally, since a circumfixing line can act as a constraint for trajectory-based motions, as it helps to amplify key movements while preserving coherence (as discussed in Section 3), it can be used to refine the region of influence. We perform this check as a first step before adjusting the *StartJoints*. To integrate this constraint, we first check if a circumfixing cluster’s *EndJoint* belongs to the kinematic chain of a trajectory cluster. If so, we set the *StartJoint* of the trajectory cluster to be equal to the *EndJoint* of the circumfixing cluster. This ensures that circumfixing lines naturally influence the motion, reinforcing important movements while preventing conflicts.



**Figure 3:** Regions of influence on an articulated character. The kinematic chains are represented, with their *StartJoint* (red dot) and *EndJoint* (blue dot). **Top:** The character tries to reach the line as far as possible with its whole top body. The kinematic chain of the cluster is highlighted in yellow. **Bottom:** to make the two motions possible, only the arms are reaching their lines. Two different kinematic chains, in yellow and green, start from the shoulders.

#### Algorithm 1 Adjusting *StartJoint*

```

1: for  $i$  from 0 to  $N_{clusters} - 1$  do
2:   for  $j$  from 0 to  $N_{clusters} - 1$  do
3:     if  $i \neq j$  then
4:        $c_1 \leftarrow clusters[i]$ 
5:        $c_2 \leftarrow clusters[j]$ 
6:       while  $c_2.StartJoint < c_2.EndJoint$  and
         [IsJointAncestor( $c_1.EndJoint$ ,  $c_2.StartJoint$ , skeleton) or
         not IsJointAncestor( $c_2.EndJoint$ ,  $c_2.StartJoint$ , skeleton)] do
7:          $c_2.StartJoint \leftarrow c_2.StartJoint + 1$ 
    
```

## 5. Inferring Motion

Another challenge is to infer the motion from the user's sketch while ensuring the movement remains natural and coherent with the character's articulated structure. Once the motion types and affected body parts have been identified, we must compute the 3D key-frame positions for the joints to animate the character.

Our approach goes beyond simply interpolating between key positions; it adapts motion generation based on the type of sketch input and the current state of the character. This requires a combination of Inverse Kinematics (IK) and Forward Kinematics (FK), as well as motion profiles that account for acceleration and impact effects.

To determine the motion's speed, we rely on visual cues provided by the sketch lines. Indeed, in his studies, Neil Cohn concluded that "the number and length of lines used in a representation may influence the perceived speed that they convey: more lines and longer lines lead to participants interpreting faster movement" [CM15]. Based on this principle, we propose to set the motion speed  $v$  to be proportional to the length of the sketched lines and their number :

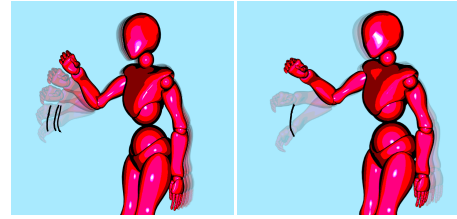
$$v = \sum_{i=1}^N \text{clamp}(k_s L_i, v_{\min}, v_{\max}) \quad (1)$$

where  $L_i$  is the length of the  $i^{th}$  motion line,  $N$  is the total number of lines corresponding to the same motion,  $k_s$  is a scaling coefficient



**Figure 4:** Effect of circumfixing lines on motion propagation. **Left:** Without circumfixing lines, the arm motion propagates to the entire upper body. **Middle:** Circumfixing lines over the shoulders constrain the motion, limiting it to the arm only. **Right:** With only circumfixing lines and no trajectory lines, the upper body exhibits a "shaking" motion.

(0.8 in our case), and  $v_{\min}$  and  $v_{\max}$  are user-defined minimum and maximum speed values for a single line. An example illustrating the influence of the number of lines on speed is shown in Fig. 5.



**Figure 5:** The number of lines controls the speed of a motion. The arm rises faster on the left than on the right, as three lines are drawn on the left compared to only one on the right.

To ensure a natural motion, each joint follows a trajectory with built-in smooth-in/out timing computed as follows. The base speed  $v$  is weighted using a time-dependent factor  $f(r)$ , where  $r \in [0, 1]$  represents the normalised progress of the motion. The modulation is defined as a piecewise sigmoid-based function:

$$f(r) = \begin{cases} \sigma\left(k_t \left(\frac{r}{r_a} - 0.5\right)\right) & \text{if } r < r_a \\ 1 & \text{if } r_a \leq r \leq r_d \\ 1 - \sigma\left(k_t \left(\frac{r-r_d}{1-r_d} - 0.5\right)\right) & \text{if } r > r_d \end{cases} \quad (2)$$

where  $\sigma(x) = \frac{1}{1+e^{-x}}$  is the sigmoid function,  $k_t$  controls the steepness of the transition, and  $r_a$ ,  $r_d$  denote the thresholds for the acceleration and deceleration phases, respectively (in our case, we fixed  $k_t = 2.5$ ,  $r_a = 0.1$  and  $r_d = 0.9$  as they gave perceptually smooth transitions). The final speed  $s$  is then obtained as  $s = v f(r)$ .

For impacts, to add expressiveness by emphasising the sudden burst of force before stabilisation, we apply an exponential increase to the base speed once the covered distance exceeds a threshold. The speed is then defined by:

$$s = \begin{cases} v & \text{if } r \leq 0.25 \\ v + a e^r & \text{if } r > 0.25 \end{cases} \quad (3)$$

where  $a$  controls the amplitude of the exponential acceleration (in our experiments, we used  $a = 0.9$  for local motion and  $a = 1.5$  for

global motion). Please refer to the supplementary video to see how modifying these parameters affects the outcome.

Next, we detail how we tailor the different sketch-based interactions to create plausible animations from minimal expressive inputs.

- Following a line: Directly constraining a joint’s motion to match a sketched trajectory.
- Continuing a trajectory: Extending motion beyond the sketch by propagating previous joint velocities.
- Follow-through on impact: Enhancing expressivity by allowing certain joints to overshoot their motion while keeping others constrained.

### 5.1. Following the Line

When a sketched line explicitly dictates motion, we must seamlessly integrate it into the articulated structure.

For each case, we extract key positions from the relevant stroke of the cluster (the one that is the closest to the *EndJoint*) and apply **Inverse Kinematics (IK)** to propagate motion along the kinematic chain. We use FABRIK [AL11], which is both computationally efficient compared to alternative methods and straightforward to implement [ALCS18].

A key aspect of this process is ensuring consistency across local and global coordinate spaces since joints are represented as  $4 \times 4$  transformation matrices encoding both translation and rotation. After computing the new positions and rotations of the joints, we update the skeleton in two steps by applying transformations in local space to maintain hierarchical dependencies and converting all transformations to global space, ensuring coherence across the entire structure.

### 5.2. Continuing a Trajectory

Once a motion has been defined, the next problem is in maintaining the natural continuation of joints without having an explicitly defined path. This is especially important, as the user-drawn sketches do not always show the entire path but the intended motion. To address this, we extend the motion using **Forward Kinematics (FK)**.

We propose to extrapolate the motion while preserving the averaged angular velocity for each joint moving. We consider  $N_L$  as the number of key positions that have been extracted from the sketch,  $\Delta t$  as the time step, and  $\mathbf{a}_i \in \mathbb{R}^3$  and  $\theta_i \in \mathbb{R}$  as the respective unit axis and angle of the rotation acting on the current joint at step  $i$ . The averaged angular velocity  $\hat{\omega}$  is then computed as:

$$\hat{\omega} = \frac{1}{N_L} \sum_{i=0}^{N_L} \frac{2\theta_i \mathbf{a}_i}{\Delta t} \quad (4)$$

The extrapolated orientation of the joint can then be computed via its quaternion  $q$ , considering that

$$q_{i+1} = \text{normalize}(q_i + \frac{1}{2}q_i\hat{\omega}^*, \Delta t), \quad (5)$$

where  $\hat{\omega}^*$  is the pure quaternion associated to the averaged angular velocity  $\hat{\omega}^* = (\hat{\omega}, 0)$ .

### 5.3. Follow-Through Effect on Impact Motion

Impact motions require careful handling to maintain realism. In traditional animation, follow-through effects help convey momentum: affected parts continue moving slightly before settling, while others remain constrained.

In our case, when a trajectory involves an impact, the affected joint (e.g., a foot hitting the ground as shown in the inset) must stop, while other joints (e.g., the knee) continue their motion briefly before stabilising.



To implement this, we invert the IK problem:

- *ImpactJoint* (the blocked joint) becomes the start.
- *EndJoint* (the free-moving joint) becomes the end effector.
- The target is *EndJointBasePose*, its original position as if there were no impact.

For each key position after the input pose, we perform the following steps:

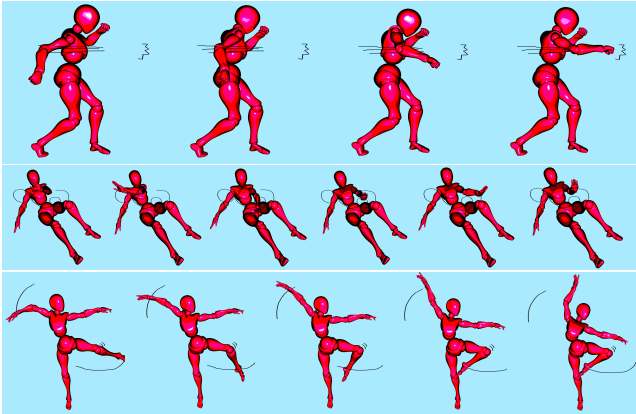
1. Initialise the skeleton: Place all joints in their previously computed positions using Inverse Kinematics instead of Forward Kinematics, as described in Section 5.1.
2. Store *EndJoint*’s position: Save the current position of *EndJoint* before modification, called *EndJointBasePose*.
3. Translate the kinematic chain: Shift all joints in the kinematic chain such that *ImpactJoint* returns to its original position, i.e. the position it had when the user initially sketched the impact.
4. Solve the IK problem:
  - If the target (*EndJointBasePose*) is reachable, compute and apply the new joint positions in the kinematic chain using the newly defined IK setup.
  - If the target is not reachable, stop the motion at the last successfully computed joint position. Then, add the most recent joint positions in reverse order. Therefore, the character will follow a slight backward motion before settling.

To prevent excessive continuation, we impose a maximum number of forward steps, ensuring a controlled decay of motion.

This method successfully balances momentum conservation with physical constraints, making impacts appear natural and responsive.

## 6. Results & Discussion

Our interactive interface is implemented in C++ using OpenGL and is run on a DELL Precision 5490. The interface allows us to load any articulated character as input, draw sketches and visualise the resulting animation within the application window. Fig. 6 shows various sample animations created using the SMACC interface (*the code will be made public upon acceptance*). As can be seen, we could create short animations with just a few sketch strokes, requiring a maximum of two minutes to create. Note that during the design phase, our method does not enforce strict geometric constraints at the joint level. This aligns with the observation that in



**Figure 6:** Animations drawn in a single 2D view, generated with SMACC. **Top:** the arm of the robot gives a punch, accelerates and is suddenly stopped by the impact. **Middle:** a trajectory motion has been drawn, causing the hand to follow the sketched path. **Bottom:** two motion clusters have been set - on the leg and the arm. Circumfixing lines are blocking the knee.

cartoon animation, unrealistic joint angles are common: for example, the technique of "breaking joints to give flexibility" [Ric02].

As illustrated in Fig. 7, it is also worth noting that our method is not restricted to humanoid characters and can be applied to any arbitrary skeletal structures, assuming a clear tree-like hierarchy.



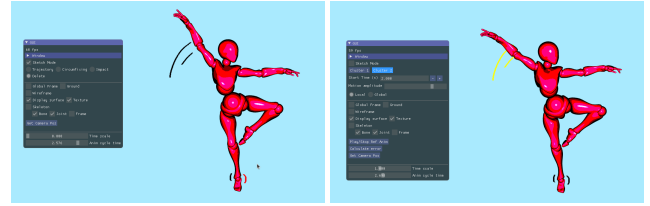
**Figure 7:** Example of a cow model animated using SMACC.

## 6.1. User Interface

As shown in Fig. 8, the SMACC interface provides two modes: sketch mode and view mode. In sketch mode, the users can sketch the desired motion lines (trajectory, circumfixing or impact) by selecting one of these three options, followed by a mouse click and drag. They can also delete a sketch by selecting the delete option and clicking on the desired stroke to be removed. In view mode, the resulting animation is shown based on the sketch strokes. In addition to viewing the animation, users can also select motion clusters and edit their parameters, such as start time (when the cluster's animation begins), motion amplitude (extent of the cluster's motion), and toggle between local and global motion.

## 6.2. User Study

To evaluate our methodology and framework, we conducted two user studies: one assessing whether users perceive the generated animation as consistent with typical cartoon expectations, and another measuring the efficiency of our interface for creating animation. To our knowledge, no prior work tackles animation generation from



**Figure 8:** Sketch and view modes of our interface.

comic annotations. As such, we compare our results to a default baseline rather than a dedicated interface.

### 6.2.1. User Evaluation 1: Accessing Perceptual Alignment of SMACC Animations with Comic Motion Lines

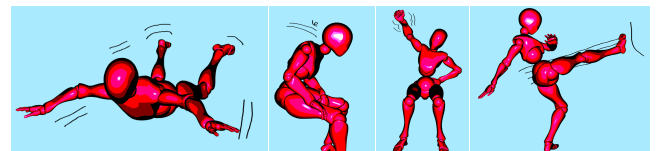
Our first objective was to assess whether the animations generated by SMACC align with what people naturally imagine from a comic strip panel.

To investigate this, we conducted a study with 24 anonymous participants of different ages, nationalities and familiarity with comics. More precisely, participants were between 22 to 30 years old, with a median of 24.5. The participants were from a range of nationalities: French, Indian, Italian, Chinese, Lebanese, Guinean, Chilean, Greek and German. And regarding the comic-reading frequency, 17% of them read comics daily, 33% weekly, 8% monthly, 21% yearly and 21% never.

Each participant first observed a comic strip panel alongside its reproduction in the SMACC interface (a few examples are shown in Fig. 9) and tried to imagine how the character might move. They then watched the animation generated by SMACC and rated how well this animation matched their expectations on a scale of 1 ("completely different from what I imagined") to 4 ("almost exactly as I imagined"). We chose a four-point scale to avoid neutral answers. A rating of 4 indicates a close, but not perfect, match with what the user might imagine – acknowledging that an exact match is unrealistic in our context.

Additionally, to evaluate the influence of our algorithmic choices and to have a more objective comparison, comic strips were presented with two animation variants: one generated using the full SMACC framework and another with **one** deliberately missing a key feature. This allowed us to evaluate how different animation properties influenced perception, including versions that:

- relied only on global character motion,
- relied only on inverse kinematics (IK),
- or lacked the "Impact Motion effect".

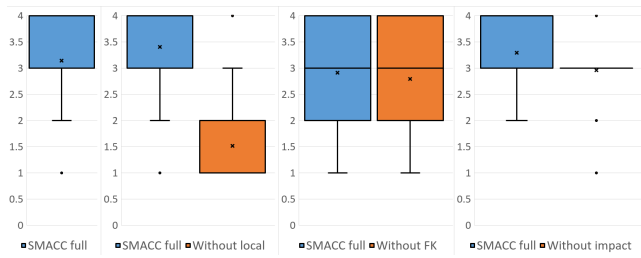


**Figure 9:** Examples of comic strip panels reproductions in SMACC.

On average, the 10 SMACC-generated animations, incorporating various comic cues, received a mean rating of 3.141 out of 4.

This shows that the animation generally aligned with what the users imagined, irrespective of their nationality or background. The mean rating for SMACC (3.40) was much higher than for global motion alone (1.52), highlighting the importance of animating articulated characters with joint-based motion.

Additionally, as seen in Fig. 10, SMACC demonstrates an advantage over animations lacking impact effects or using only Inverse Kinematics (IK) for motion continuation, though the difference was smaller. This is likely because these variations introduce more subtle differences that are harder to perceive. However, continuing the motion with Forward Kinematics (FK) offers an advantage by allowing for easier enforcement of rotational constraints on joints, which is more challenging with IK alone.



**Figure 10:** Results of Perceptual Alignment Validation. The mean is represented by a cross, the median by a black line and the interquartile range by a colored area. The dots represent the outliers. Animations missing key features are compared with their corresponding full SMACC animations only, which is why the four "SMACC full" plots differ.

### 6.2.2. User Evaluation 2: Reproducing animations using SMACC Interface

In addition to the perceptual study, we conducted a second user evaluation to assess whether novice users could create desired animations using comic cue annotations with our SMACC interface. We recruited 20 participants from diverse backgrounds, including those with no animation experience. The evaluation is divided into four consecutive phases without any interruptions:

1. **Demonstration Phase** (approx. 3 minutes): Groups of 1 to 4 participants observed a demonstration of the SMACC interface. This included explanations on the different types of lines, how to draw and delete them, and how to control their amplitude.

2. **Training Phase** (approx. 5 minutes): Each participant individually tested the interface using a swimming pose. They were encouraged to explore freely, and this training phase ended only when the participant explicitly stated they were ready to proceed.

3. **Test Phase:** The participant was shown an animation. When they indicated they were ready, the recording began. They then drew and adjusted motion lines in an attempt to reproduce the animation without providing any guidance or predefined solutions. They were allowed to rewatch the animation as many times as needed. To simplify the evaluation, users were limited to adjusting joints locally and working with a single provided view, allowing them to focus on fine-tuning their animation. Once satisfied

with their result, they informed the experimenter, and the recording stopped. The average angular error between their result and the reference animation was then computed and saved. This process was repeated four times for the following animation types:

- One simple, custom-made animation
- One complex, custom-made animation
- One simple animation created by professional animators
- One complex animation created by professional animators

As shown in Fig. 11, users drew different motion lines based on their own perception and could still create the desired animation using our interface, showing the flexibility given to the users to express their thoughts in their own way.

4. **Final Phase:** After completing the test phase, participants were asked to fill out a questionnaire to evaluate various aspects of the system.

Tables 1 and 2 summarise the results. Participants completed each animation in a matter of minutes, even without prior experience in creating animations. While task complexity affected the timing, all participants succeeded in producing animations resembling the references. The overall mean angular error was 0.0174, indicating that sketches yielded accurate motion representations. As expected, perfect reproduction was not achieved, but minor differences in stroke placement and scale did not impede the interpretability of the animation.

**Table 1:** Animation Completion Times in SMACC

	Anim 1	Anim 2	Anim 3	Anim 4
Mean	1min44s	4min08s	2min36s	4min19s
Min	33s	1min22s	23s	1min41s
Max	3min40s	9min16s	6min53s	8min28s

**Table 2:** Mean Angular Error in SMACC

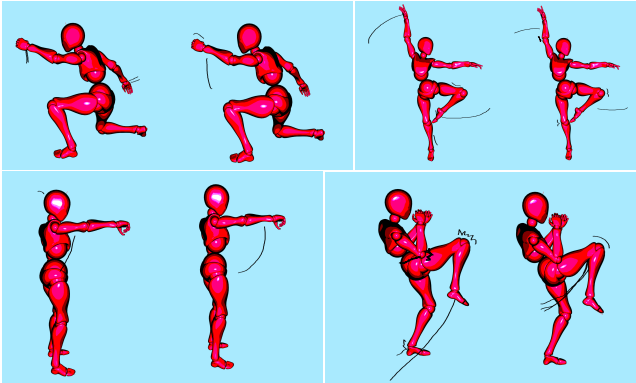
	Anim 1	Anim 2	Anim 3	Anim 4
Mean	0.00514	0.02811	0.01207	0.02356
Min	0.00107	0.01019	0.00919	0.01966
Max	0.02174	0.04988	0.01818	0.03834

For the animations originally created in SMACC, some participants misinterpreted specific cues, for example, confusing the use of impact strokes with amplitude control. For animations created by professional animators, certain fine details were difficult to reproduce, but participants still succeeded in creating the animation that satisfied their perception, as reflected by an average rating of 3.7 out of 5. These outcomes suggest that users were capable of capturing essential motion dynamics using only the comic cue vocabulary, even for unfamiliar or complex movements.

Participants rated the clarity of the interface and sketching process positively (3.9/5), and most reported that the concept of using comic cues was easy to grasp (3.95/5). Even among participants without prior experience in digital content creation (40% of the group), they were able to complete all tasks, confirming the accessibility of the method.

Several participants commented on the expressiveness of the cue

system, citing their ability to convey specific effects like the "possibility to introduce complex movements like shaking" or "the twisting of limbs on impacts". Suggestions for improvement included allowing users to modify the initial pose within the system by offering "more steps" and providing better control over post-sketch motion adjustments. A notable limitation raised by participants was the fixed joint hierarchy, making it impossible to move the character's root independently of the external joints.



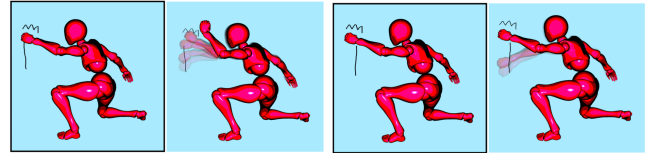
**Figure 11:** A few sample sketches the participants used to represent similar motion.

### 6.3. Inconsistent inputs

Despite the simple interface, results obtained with SMACC can be affected by inconsistent inputs. The first such scenario is the contradicting strokes - where the user draws two sketches on a single joint, each instructing the joint to move in opposite directions as shown in Fig. 12(a). To handle such cases, we prioritise the strokes based on the distance to the joint and ties broken based on the drawing order (result shown in Fig. 12(b)). If the computed motion is unsatisfactory, further iterations can be applied by deleting the contradictory strokes using the interface as in Fig. 12(c). Another scenario is when the user sketches the stroke in the wrong direction, resulting in the selection of the incorrect joint, as SMACC selects the joint that is closest to the last drawn point of the stroke. For example, Fig. 12(e) shows a stroke drawn in the wrong direction, leading to choosing the knee instead of the intended ankle. In such cases, the user has to delete and redraw the stroke.

### 6.4. Complex Animations using SMACC

While the user study demonstrates that novices can create desired animations using the SMACC interface, it is also possible to generate complex animations with comic-cue annotations through advanced features. These include the ability to adjust the global motion, fine-tune the start time of individual motion clusters, or sketch from different viewpoints. A few sample complex animations using these advanced features are illustrated in Figures 13a, 13b and 13c (see supplementary video to see the animations).

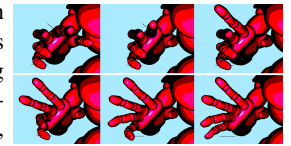


**Figure 14:** Unexpected result due to obscured views.

### 6.5. Limitations

While the user study demonstrated that users could create the desired animation irrespective of their expertise, we found a few areas for improvement:

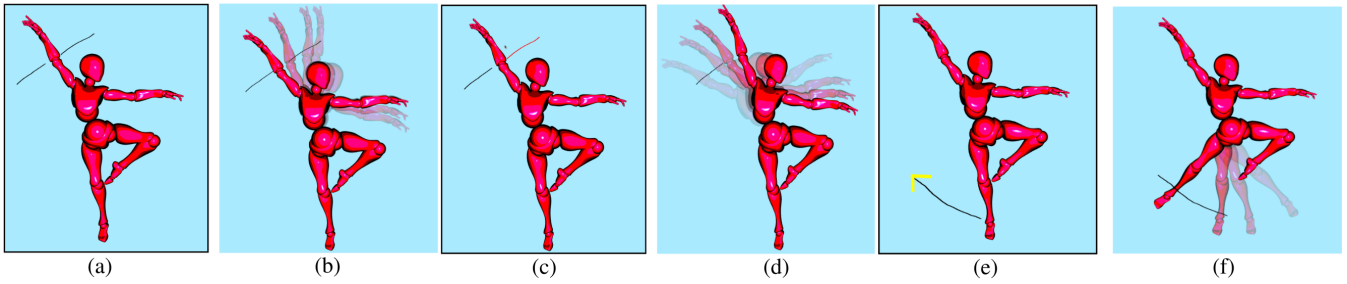
- Users found it difficult to animate inner joints independently of the outer ones. This limitation could be alleviated by allowing users to specify external joints to act as constraints and to develop a possibility for the root joint to infer the motion. In addition, although faithful to comic conventions, users expressed a desire to adjust initial poses.
- Since sketches are drawn from a particular view, they can sometimes get attached to unintended joints as the intended joints might be obscured. For instance, in the example shown in Fig. 14, the user intends to move the character's hand upwards and stop it with an impact. However, the trajectory line finds the closest joint to be a finger, while the impact line finds another finger. As these two joints are not part of the same chain, the impact is not linked to the trajectory line, causing the hand to continue moving upwards. This requires users to delete and carefully redraw strokes from the best view.
- Our primary focus is on motion lines representing the character's large movements. Incorporating fine details would require zooming in or out on the relevant parts, as illustrated in the inset example. However, expressive motions such as facial expressions fall outside the scope of our work.



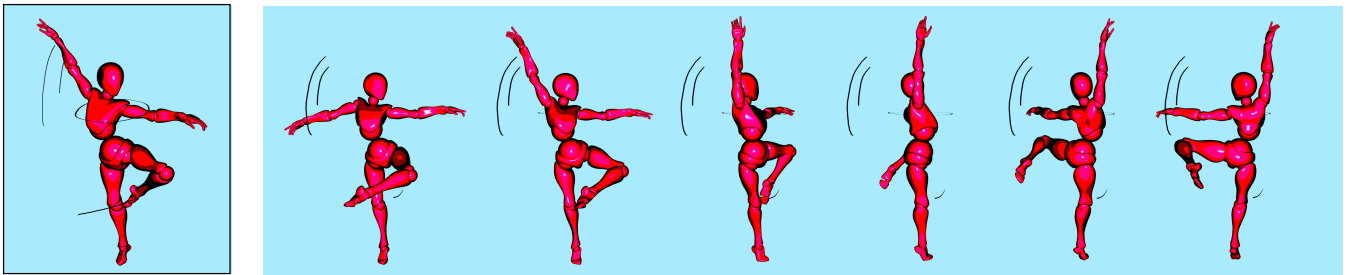
### 7. Conclusion

We introduced a sketch-based system, SMACC, for the quick creation of short animations. Taking inspiration from traditional comic books, SMACC allows users to sketch motion lines, interpret them, and use them to infer kinematic parameters. The user-drawn sketches are first analysed and clustered, and they are then associated with joints and a region of influence is defined to promote expressive exaggeration. These joints are then made to follow sketched trajectories while ensuring expected natural motion. With two user studies, we demonstrated that users were able to express the motion in their minds using our interface and could create the desired animation without the need for animation expertise.

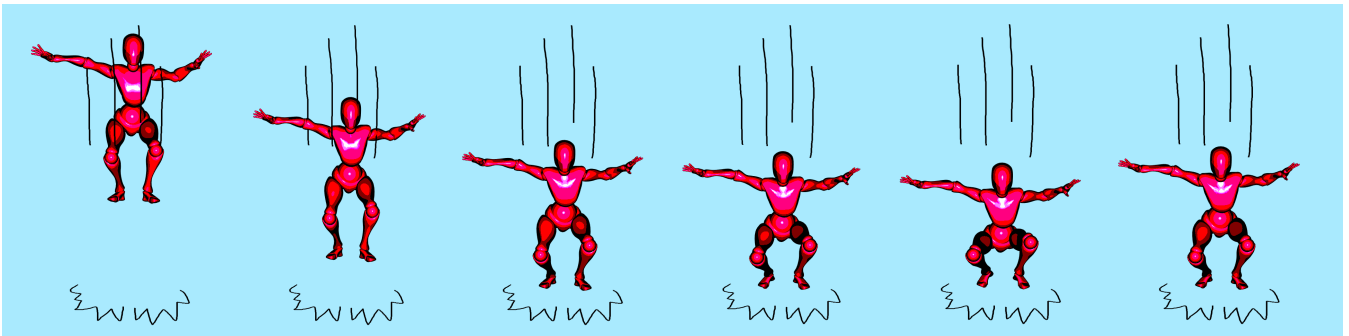
*Future Work:* In the future, we want to animate comic panels directly by reconstructing 3D characters from 2D drawings, which respond directly to the motion lines. In addition to simple animations, we also want to improve our interface to handle shape deformations, artistic animations, and secondary motion. Additionally, the SMACC interface could be incorporated with well-established



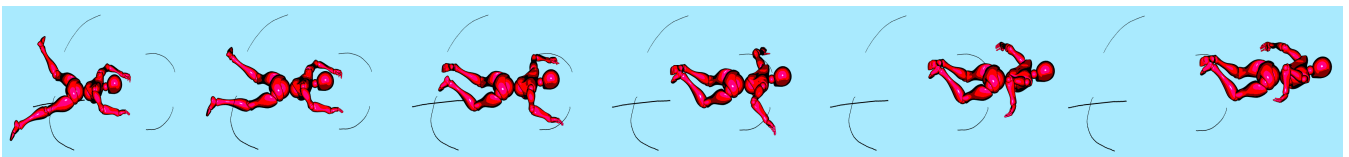
**Figure 12:** Inconsistent inputs: (a) Two contradicting strokes on a joint, (b) Resulting animation by prioritising the stroke order, (c) User deleting one stroke (in red), (d) Resulting animation, (e) Stroke drawn in opposite direction, (f) Resulting animation.



(a) A complex dance animation. **Left:** input sketches drawn on three separate planes (arm, foot, and body). **Right:** the resulting animation, with the dancer spinning while moving the arm and leg.



(b) A complex falling animation. Sketch lines are applied to the entire body, causing the character to fall and come to a stop through foot impacts.



(c) A complex swimming animation. The timing hierarchy is adjusted so that the legs initiate the motion before the arms, while the entire character moves forward.

**Figure 13:** Complex animations generated using advanced features in SMACC.

animation software like Blender, to mix it with traditional key-frame animations. Another extension would be to enable additional functionalities like sequential motions (i.e. allowing a single joint to perform multiple movements over time). Finally, the interface could be refined to integrate more HCI concepts and animator feedback.

### Acknowledgments

The authors thank all anonymous reviewers for their constructive feedback. The work is funded by the ANR JCJC project SketchMAD (ANR-23-CE33-0009) and generous support from Adobe.

## References

- [ABB\*24] AGRAWAL D., BUHMANN J., BORER D., SUMNER R. W., GUAY M.: Skel-betweener: a neural motion rig for interactive motion authoring. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–11. 2, 3
- [AL11] ARISTIDOU A., LASENBY J.: Fabrik: A fast, iterative solver for the inverse kinematics problem. *Graphical Models* 73, 5 (2011), 243–260. 6
- [ALCS18] ARISTIDOU A., LASENBY J., CHRYSANTHOU Y., SHAMIR A.: Inverse kinematics techniques in computer graphics: A survey. In *Computer graphics forum* (2018), vol. 37, Wiley Online Library, pp. 35–58. 6
- [BB22] BRODT K., BESSMELTSEV M.: Sketch2pose: Estimating a 3d character pose from a bitmap sketch. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–15. 3
- [BVS16] BESSMELTSEV M., VINING N., SHEFFER A.: Gesture3d: posing 3d characters via gesture drawings. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–13. 3
- [CBSG08] COLEMAN P., BIBLIOWICZ J., SINGH K., GLEICHER M.: Staggered poses: A character motion representation for detail-preserving editing of pose and coordinated timing. In *Symposium on Computer Animation* (2008), pp. 137–146. 3
- [CGNS17] CICCONE L., GUAY M., NITTI M., SUMNER R. W.: Authoring motion cycles. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2017), pp. 1–9. 3
- [CiRL\*16] CHOI B., RIBERA R. B., LEWIS J. P., SEOL Y., HONG S., EOM H., JUNG S., NOH J.: Sketchimo: sketch-based motion editing for articulated characters. *ACM Transactions on Graphics (ToG)* 35, 4 (2016), 1–12. 3
- [CM15] COHN N., MAHER S.: The notion of the motion: The neurocognition of motion lines in visual narratives. *Brain research* 1601 (2015), 73–84. 2, 4, 5
- [CMB17] CARVALHO L., MARROQUIM R., BRAZIL E. V.: Dilight: Digital light table–inbetweening for 2d animations using guidelines. *Computers & Graphics* 65 (2017), 31–44. 2
- [CNKI13] CHOI M. G., NOH S.-T., KOMURA T., IGARASHI T.: Dynamic comics for hierarchical abstraction of 3d animation data. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 1–9. 3
- [DAC\*06] DAVIS J., AGRAWALA M., CHUANG E., POPOVIĆ Z., SALESIN D.: A sketching interface for articulated figure animation. In *Acm siggraph 2006 courses*. 2006, pp. 15–es. 3
- [DCL08] DAVIS R. C., COLWELL B., LANDAY J. A.: K-sketch: a ‘kinetic’ sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008), pp. 413–422. 3
- [DRVDP15] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector graphics animation with time-varying topology. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12. 2
- [DSC\*20] DVOROŽNÁK M., SÝKORA D., CURTIS C., CULLESS B., SORKINE-HORNUNG O., SALESIN D.: Monster mash: a single-view approach to casual 3d modeling and animation. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–12. 2
- [DVE23] DEGEN I., VAN ENSCHOT R.: Depicting motion in comics: A cross-cultural analysis of motion cues and path structures. 4
- [EBB24] EVEN M., BÉNARD P., BARLA P.: *Inbetweening with Occlusions for Non-Linear Rough 2D Animation*. PhD thesis, Inria; Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, 2024. 2
- [EPB\*19] ENTEM E., PARAKKAT A. D., BARTHE L., MUTHUGANAPATHY R., CANI M.-P.: Automatic structuring of organic shapes from a single drawing. *Computers & Graphics* 81 (2019), 125–139. 2
- [FBC\*95] FEKETE J.-D., BIZOUARN É., COURNARIE É., GALAS T., TAILLEFER F.: Tictactoon: A paperless system for professional 2d animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 79–90. 2
- [GCR13] GUAY M., CANI M.-P., RONFARD R.: The line of action: an intuitive interface for expressive character posing. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–8. 3
- [GRC19] GARCIA M., RONFARD R., CANI M.-P.: Spatial motion doodles: Sketching animation in vr using hand gestures and laban motion analysis. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games* (2019), pp. 1–10. 3
- [GRGC15a] GUAY M., RONFARD R., GLEICHER M., CANI M.-P.: Adding dynamics to sketch-based character animations. In *Sketch-Based Interfaces and Modeling (SBIM) 2015* (2015), Eurographics Association, pp. 27–34. 3
- [GRGC15b] GUAY M., RONFARD R., GLEICHER M., CANI M.-P.: Space-time sketching of character animation. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–10. 2, 3
- [GVA\*24] GAL R., VINKER Y., ALALUF Y., BERMANO A., COHEN-OR D., SHAMIR A., CHECHIK G.: Breathing life into sketches using text-to-video priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 4325–4336. 2
- [HLW93] HSU S. C., LEE I. H., WISEMAN N. E.: Skeletal strokes. In *Proceedings of the 6th annual ACM symposium on User interface software and technology* (1993), pp. 197–206. 1, 2
- [HMC\*15] HAHN F., MUTZEL F., COROS S., THOMASZEWSKI B., NITTI M., GROSS M., SUMNER R. W.: Sketch abstractions for character posing. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), pp. 185–191. 3
- [IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005), 1134–1141. 2
- [JBPS11] JACOBSON A., BARAN I., POPOVIC J., SORKINE O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (2011), 78. 2
- [JSL22] JIANG J., SEAH H. S., LIEW H. Z.: Stroke-based drawing and inbetweening with boundary strokes. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 257–269. 2
- [KCG\*14] KAZI R. H., CHEVALIER F., GROSSMAN T., ZHAO S., FITZMAURICE G.: Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), pp. 351–360. 3
- [KCGF14] KAZI R. H., CHEVALIER F., GROSSMAN T., FITZMAURICE G.: Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (2014), pp. 395–405. 3
- [KGUF16] KAZI R. H., GROSSMAN T., UMETANI N., FITZMAURICE G.: Motion amplifiers: sketching dynamic illustrations using the principles of 2d animation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), pp. 4599–4609. 3
- [Kor02] KORT A.: Computer aided inbetweening. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002), pp. 125–132. 2
- [KSVDP09] KRAEVOY V., SHEFFER A., VAN DE PANNE M.: Modeling from contour drawings. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based interfaces and Modeling* (2009), pp. 37–44. 3
- [MGW24] MO H., GAO C., WANG R.: Joint stroke tracing and correspondence for 2d animation. *ACM Transactions on Graphics* 43, 3 (2024), 1–17. 2
- [MJ96] MACCRACKEN R., JOY K. I.: Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 181–188. 2

- [NF03] NEFF M., FIUME E.: Aesthetic edits for character animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 239–244. [3](#)
- [Ric02] RICHARD W.: The animator’s survival kit, 2002. [4](#), [7](#)
- [RKW\*24] ROSENBERG K. T., KAZI R. H., WEI L.-Y., XIA H., PERLIN K.: Drawtalking: Building interactive worlds by sketching and speaking. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (2024), pp. 1–25. [2](#)
- [SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. In *ACM SIGGRAPH 2006 Papers*. 2006, pp. 533–540. [2](#)
- [WNS\*10] WHITED B., NORIS G., SIMMONS M., SUMNER R. W., GROSS M., ROSSIGNAC J.: Betweenit: An interactive tool for tight in-betweening. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 605–614. [2](#)
- [XKG\*16] XING J., KAZI R. H., GROSSMAN T., WEI L.-Y., STAM J., FITZMAURICE G.: Energy-brushes: Interactive tools for illustrating stylized elemental dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), pp. 755–766. [3](#)
- [XZJJ25] XIE T., ZHAO Y., JIANG Y., JIANG C.: Physanimator: Physics-guided generative cartoon animation. *arXiv preprint arXiv:2501.16550* (2025). [2](#)
- [YKSF20] YE H., KWAN K. C., SU W., FU H.: Aranimator: In-situ character animation in mobile ar with user-defined motion gestures. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 83–1. [3](#)
- [ZCXP24] ZHENG Y., CUN X., XIA M., PUN C.-M.: Sketch video synthesis. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15044. [2](#)
- [ZGX\*25] ZHONG L., GUO C., XIE Y., WANG J., LI C.: Sketch2anim: Towards transferring sketch storyboards into 3d animation. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–15. [3](#)
- [ZLFA24] ZHOU Q., LEDO D., FITZMAURICE G., ANDERSON F.: Timetunnel: Integrating spatial and temporal motion editing for character animation in virtual reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (2024), pp. 1–17. [3](#)
- [ZLWH16] ZHU H., LIU X., WONG T.-T., HENG P.-A.: Globally optimal toon tracking. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10. [2](#)