Posing 3D characters in virtual reality through in-the-air sketches

Posing 3D Characters in Virtual Reality through In-the-Air Sketches

Alberto Cannavò
Politecnico di Torino
alberto.cannavo@polito.it

Congyi Zhang
The University of Hong Kong
cyzh@hku.hk

Wenping Wang
The University of Hong Kong
wenping@cs.hku.hk

Fabrizio Lamberti
Politecnico di Torino
fabrizio.lamberti@polito.it

Abstract

Generating computer animations is a very labor-intensive task, which requires animators to operate with sophisticated interfaces. Hence, researchers continuously experiment with alternative interaction paradigms that could possibly ease the above task. Among others, sketching represents a valid alternative to traditional interfaces since it can make interactions more expressive and intuitive; however, although the literature proposes several solutions leveraging sketch-based interfaces to solve different computer graphics challenges, generally they are not fully integrated in the computer animation pipeline. At the same time, Virtual Reality (VR), is becoming commonplace in many domains, and recently started to be recognized as capable to make it easier also the animators' job, by improving their spatial understanding of the animated scene and providing them with interfaces characterized by higher usability and effectiveness. Based on all of the above, this paper presents an add-on for a well-known animation suite that combines the advantages offered by a sketch-based interface and VR to let animators define poses and create virtual character animations in an immersive environment.

Keywords: 3D Animation Sketch-based interfaces Virtual Reality

1 Introduction

Nowadays, the generation of virtual character animations is becoming fundamental for a number of applications, from the production of movies and video-games to the creation of a variety of virtual environments (VEs) used, e.g., in education, cultural heritage, product design, and social networking scenarios, to name a few [1, 2]. However, animating 3D characters still represents a very labor-intensive task and is generally characterized by the involvement of expert users with significant skills in the field of computer animation [3, 4].

According to [5], the complexity underlying the creation of animated characters lays in the posing step. In this step, animators are often requested to select and manipulate a large number of on-screen 3D "handles" for articulating the character's virtual skeleton. This set of handles is often referred to as an "armature" or a "rig", constituted by rigid elements named "bones". Handles/Bones can be used by the animators to directly or indirectly manipulate the degrees of freedom (DOFs) of all the individual character's parts/joints [6]. Unfortunately, many animation systems still leverage traditional interfaces for system input and output, like mouse and keyboard or 2D displays, which represent sub-optimal means to handle interactions encompassing a larger number of DOFs [3, 7].

Considering the user input, among the ap-

proaches proposed in the literature to tackle the above issues a promising means is represented by sketch-based interfaces. It is worth mentioning that sketching is already exploited in others steps of the creative process, e.g., for building up shapes, exploring motion with rough key poses, drawing storyboards, etc. [8]. In fact, the literature shows that sketch-based interfaces have been successfully investigated by the research community to cope with limitations faced in various computer graphics applications, since they enable an expressive, simple and intuitive interaction that is close to the functioning of many cognitive processes [9]. Applications of sketch-based interfaces for character animation encompass modeling [10], rigging [11], posing [6], crowd simulation [12], etc. However, articulated characters may have a relatively high number of DOFs to control, and taking into account the complexity of operating with 3D elements through 2D devices or of interpreting 2D line drawings in 3D, it is not surprising that sketch-based animation of articulated characters remains an open problem [7, 13].

For what it concerns system output, it is possible to notice that, similarly to the the user input, it can be affected by the limited dimensionality of the visualization methods. With 2D displays, animators are requested to continuously change the position of the virtual camera or simultaneously look at multiple views of the scene been animated in order to visualize the contents of interest from the required viewpoint [14]. Both these solutions could led to an increased complexity in the usage of the animation tools, especially for novice users [3]. Given such limitations, an increasing number of users with different skills like, e.g., digital artists, filmmakers and storytellers, among others, recently started to pose their attention to the opportunities offered by Virtual Reality (VR) not only as a means to visualize character animations, but also to create them [15]. Although various commercial products and research prototypes are already available in the main VR stores, most of them provide a limited integration with common animation suites [3, 16]. More specifically, in order to apply changes or reuse the animations generated within immersive environments, animators are generally requested to continuously perform import/export operations. These additional operations may slow down the whole animation process, and could be perceived as highly distracting by the animators.

By moving from the above considerations, this paper presents a system for character posing able to combine the benefits offered by sketch-based interfaces and VR technology. With this system, animators can manipulate a rigged virtual character by sketching lines into an immersive VE. To this aim, relevant works in the literature have been considered in order to extend the capabilities of existing 2D solutions to 3D immersive VEs. Moreover, the sketch-based interface is integrated in the well-know Blender modeling and animation suite[1], with the aim to let animators reuse articulated characters without the need for import/export operations. Lastly, since in the devised system the traditional and the sketch-based VR interfaces can be used at the same time, multiple users can observe and possibly work on the scene in a collaborative way.

2 Related works

Researchers have long acknowledged the benefits brought by the use of sketch-based interfaces to execute a broad range of tasks in the computer animation field. For example, the authors of [17] presented a mathematical definition of the *Line of Action* (LOA), a conceptual tool used by cartoonists and illustrators to make the animated characters more consistent and dramatic. The system provides animators with an automatic procedure (based on an optimization problem) to align a 3D virtual character with a user-specified LOA. By considering this simple abstraction, the animator can easily adjust and refine the overall character's pose from a fixed viewpoint. The work proposes an automatic algorithm to define the correspondences between the LOA and a subset of the character's bones; the well-known *depth ambiguities* problem of 2D sketches [17] is addressed by constraining the transformations to the viewing plane.

In [6], a sketch-based posing system for 3D rigged characters was proposed letting animators create a custom *sketch abstraction*, i.e., a set of rigged curves that constitute an iconographic

[1] https://www.blender.org/

2D representation of the character from a particular viewpoint, on top of a character's shape. When the animator provides a new input sketch, the system automatically tries to determine the rigging parameters that best align the character's sketch abstraction to the input sketch by minimizing the nonlinear iterative closest point energy. The distinguishing feature of the method presented in [6] is that it does not require any specific sketch representation a priori, but rather allows the animator to encode the sketch abstractions that are the most appropriate for the character to be deformed.

The work in [13] introduced a sketch-based character posing system which is more flexible than those introduced above. In fact, the sketches provided as input for character deformation may depict the skeleton of the character, its outline, or even a combination of the two elements. An optimization problem is formulated to determine the match between the subset of vertices of the character's mesh and the points obtained by sampling the input sketch.

The method that was presented in [18] is able to reconstruct the character's pose by using 2D "gesture drawings" and a 3D character rigged model as input. The benefit coming from the use of gesture drawings over other 2D inputs is the lack of perceptual ambiguities. Unlike stick-figures, LOA, and outer silhouettes, gesture drawings allow animators to unambiguously convey poses to human observers. By recognizing and leveraging the perceptual pose cues provided when creating these drawings, the system is able to automatically reconstruct character's poses that are consistent with the animator's intent. The system is able to manage complex poses with varying and significant part foreshortening, occlusions, and drawing inaccuracies.

The work in [8] described a method to infer a 3D character's pose from a monocular 2D sketch. Since the devised method does not assume any specific characteristics of the model, it can be exploited with different types of characters. The 3D pose estimation is expressed as an optimization problem. In particular, a parallel variation of a Particle Swarm Optimization (PSO) algorithm [19] is used to manipulate the pose of a preexisting 3D model until a suitable pose is found. The pose is determined by automatically comparing the 3D rendering of the model and the input drawing. During the process, user's input is still needed to pinpoint the joints on the drawing.

3 Proposed system

From the analysis of the literature, it can be noticed that existing systems for posing 3D characters using sketches are still based on 2D devices. As a result, the viewpoint from which the sketch is drawn represents a key factor in the creation of the pose, since it influences the accuracy of the final 3D result.

The basic idea behind the system presented in this paper is to turn the existing methodology from 2D to 3D, letting animators draw sketches directly in an immersive VE. Fig. 1 shows the expected usage of the proposed system. Given a 3D rigged character and some sketches drawn by the animator into the VE, the system is able to automatically align them by minimizing their distance. The system assumes that a skeleton is already defined for the character: hence rigging and skinning are not considered (both immersive and non-immersive methods reported, e.g., in [4] could be used to those purposes).

Another disadvantage of the solutions proposed in the literature is the fact that they generally come as standalone applications. Hence, integration with common animation suites like Blender, Autodesk Maya, etc. can take place only in a separate step of the animation workflow, thus making the process more tricky. The design reported in this paper considered the aspect above, and devised the system as an add-on for Blender. The aim of the proposed sketch-based system is not to replace Blender, but rather to offer an alternative interaction means that

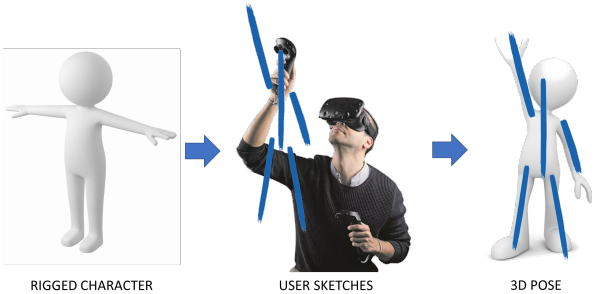

Figure 1: Expected use of the proposed system.

combines the affordances of posing characters via in-the-air sketches and the advanced functionalities targeted to character animation provided by traditional software, with the ultimate goal to speed up the overall process.

The steps followed in the development of the proposed system and the challenges to be solved can be summarized as follows:

- creating an immersive environment where the user can draw sketches;
- designing a methodology to find the best mapping between the sketches provided as input and the skeleton of the character to be deformed;
- developing a methodology to find the transformations to be applied to the bones in the character's skeleton in order to align them with the sketches;
- integrating the devised solution into a well-know animation suite.

These requirements were considered in the design and implementation of the two main components of the system, i.e., a VR-based environment integrated in Blender letting animators draw 3D sketches, and a *matching & aligning* algorithm in charge of articulating the character's skeleton to make it assume the poses represented by the given sketches. In the following, more details about the VR-based environment as well as the functioning of the algorithm will be given. Current development state of the proposed system will be provided too.

3.1 VR-based environment for 3D sketching

The devised system was developed as an add-on for Blender by leveraging two existing libraries: the Virtual Reality Viewport library[2] and the Pyopenvr SDK[3].

The Virtual Reality Viewport library lets Blender's users visualize a 3D scene (containing the characters to be animated and the sketches being created) into an immersive environment through a Head-Mounted Display (HMD). Pyopenvr is a binding for the Valve's OpenVR SDK designed to made the status of the HTC Vive's controllers (and HMD) available in Python; it can be exploited to implement specific functionalities when the user interacts with the controllers.

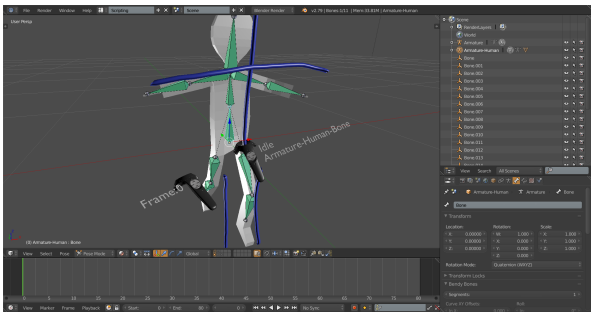

Figure 2: Blender's interface and the new add-on for sketching in VR.

Fig. 2 shows the main Blender's interface and the new add-on. On the left side, the window of *3D View* editor is shown, which contains the 3D objects, i.e., the virtual character (in gray), its skeleton (in green), the virtual representation of the two controllers (in black), and the sketches drawn by the user (in blue). The content of this window is visualized in VR through the HMD. The remaining panels are those of the traditional, mouse & keyboard-based, Blender's interface.

The developed tool's functionalities can be accessed in the immersive environment by acting on the controllers. Currently, the tool allows the user to:

- draw a stroke (right controller's Trigger);
- select the character to pose (right/left controller's Gripper);
- apply translation and rotation transformations to the selected character (left controller's Trigger)
- launch the algorithm (right controller's Trackpad Up);
- reset the transformations applied to the skeleton by setting the rest pose for it (right controller's Trackpad Right);

[2]VR Viewport: `https://github.com/dfelinto/virtual_reality_viewport`

[3]Pyopenvr: `https://github.com/cmbruns/pyopenvr`

- delete all the strokes drawn by the user (right controller's Trackpad Down);
- delete the last stroke drawn by the user (right controller's Trackpad Left);
- activate the playback of the animation (left controller's Trackpad Up);
- navigate the timeline by increasing/decreasing the current frame (left controller's Trackpad Right/Left);
- insert a keyframe to record the orientation of all the bones in the character's skeleton for the current frame (left controller's Trackpad Down).

Controllers' buttons functionalities are illustrated in Fig. 3. Visual feedback was introduced to simplify interaction with the system. In particular, on the right controller, a label indicates the current system's status, i.e., Idle (waiting for a new command) or Selection (functionalities for changing the skeleton to be manipulated available), and the currently skeleton selected. On the left controller, a label shows the current frame and the presence of a keyframe for the selected skeleton; if a keyframe has been set for the current frame, the text is colored yellow, otherwise it remains gray (convention used in Blender).

3.2 Matching & aligning algorithm

The devised algorithm represents a revised version of the method proposed in [13], where the problem of identifying the mapping between the pose of a virtual character and an input sketch was expressed as an optimization problem. The problem can be summarized as follows: given two sets of points, the sampled input sketch $Y = (y_1, y_2, ..., y_M)$ and a subset of points belonging to the character model $V = (v_1, v_2, ..., v_K)$, find the correspondence, or match matrix, ω, and the amount of deformations in p that minimize the energy $E_{3D}(\omega, p)$ expressed as:

$$min(\omega, p) \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_{ki} \left\| y_i - v_k(p) \right\|_2^2 + \\ + \Phi(p) - \zeta \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_{ki} \quad (1)$$

Figure 3: Functionalities available through the controllers.

subject to the following constraints:

$$\sum_{i=1}^{M+1} \omega_{ki} = 1; \sum_{k=1}^{K+1} \omega_{ki} = 1; \omega_{ki} \in \{0, 1\} \quad (2)$$

where $\omega = \{\omega_{k,i}\}_{(K+1)\times(M+1)}$ is the correspondence matrix consisting of two parts: the upper-left $K \times M$ part defines the correspondence, whereas the extra $K+1$-th row and $M+1$-th column are introduced to handle the outliers; the points in V and Y having no correspondences are considered as outliers; p is a vector containing the character posing parameters on the joints which deform points in V to Y in order to obtain a new pose $V(p)$ as close as possible to Y; $\Phi(p)$ is a regularization term, used to add further constraints for searching candidate solutions in a limited space; ζ is a scalar factor to weight the contribution of the last term of the equation, introduced to prevent treating too many points as outliers.

In order to solve equation (1), the methodology presented in [13] proposes an alternating strategy to find the correspondence parameter ω and the deformation parameters p. Going back and forth between the correspondence and pose in an iterative way can help to solve the problem, since the knowledge of one element relatively makes it easier the determination of the other one.

By fixing p, it is possible to determine the sub-optimal values for ω by using two techniques: *softassign* [20] and *deterministic annealing* [21]. Unfortunately, in [13], no technical details were provided on how to use these two methods for the given problem. Hence, to implement them, the original paper presenting the use of these techniques for 2D and 3D point matching was considered [21]. The idea of the

softassign algorithm is to relax the binary correspondence variable ω to be a continuous-valued matrix in the interval [0, 1]. The continuous nature of the matrix basically allows for fuzzy, partial matches between the two sets of points [21]. From an optimization point of view, this fuzziness makes the resulting energy function behave better, because the correspondences are able to improve gradually and continuously during the optimization, without jumping around in the space of binary permutation matrices (and outliers) [20]. The row and column constraints in equation (2) can be enforced via iterative row and column normalization of ω [22]. Deterministic annealing can be applied to directly control the fuzziness introduced with the softassign algorithm by adding an entropy term to the original assignment energy function in equation (1) [21]. The newly introduced parameter β is called the temperature parameter. The name comes from the fact that, as one gradually reduces β, the energy function is minimized by a process similar to physical annealing. At higher temperatures, the entropy term forces the correspondence to be fuzzier. The values achieved at each temperature are adopted as initial conditions for the next stage as the temperature is lowered. According to the above techniques, the correspondence matrix is updated at each iteration using the expression $\omega_{ki} = \exp(\beta Q_{ki})$, where $Q_{ki} = -\frac{\partial E_{3D}}{\partial \omega_{jk}}$. At each iteration, the value of β is incremented by a fixed amount which is defined at the beginning of the process.

Once the correspondence is found, it is possible to fix ω to obtain the parameters in p which minimize the energy function in equation (1). Blender's functionalities were used to get the transformation values that best aligns the character's skeleton to the corresponding points in the sketch.

The algorithm is executed until the correspondence matrix converges or the maximum number of iterations is reached.

3.3 Current development state

At present, the system supports multiple strokes and multiple skeletons; if the scene contains more than one character, all of them can be manipulated using the above approach (one at a time) by drawing a stroke for each of their bones. Fig. 4 shows several characters (whose geometry was kept intentionally simple) characterized by armatures with a different topology. Fig. 4a, 4e, and 4g shows the armatures in *rest pose* and the drawn sketches, whereas Fig. 4b, 4f, and 4h illustrate automatically computed poses. Fig. 4c and Fig. 4d show a set of alternative keyframes that have been obtained for the armature in rest pose of the character in Fig. 4a by drawing the sketches represented in the figures. In order to show the current development state and the effect that drawn sketches have on the characters' poses, a video was recorded: the video, which is available for download[4], shows a user animating two different characters. The source code of the project is available too[5].

Currently, the system presents the limitations reported below.

- The proposed algorithm assumes that strokes are provided for all the bones in the character's armature. If some strokes are not drawn, possible splits in the armature (e.g., the arms of the human character in Fig. 4a) will be mapped on the same stroke, by overlapping the two chains. A mechanism could be implemented to let the users specify the set of bones to be actually aligned, making the algorithm disregard some parts of the armature.

- Labels on the controllers representing available functionalities could be difficult to read in VR due to the limited resolution of the HMD. More meaningful graphics (e.g., 3D icons) could be used to improve the user experience.

- Users may find it difficult to accurately draw strokes in VR with the controllers, since visual feedback representing the actual position in which the strokes will be drawn is missing. Possible solutions could consider the integration of ad-hoc interaction devices (like, e.g., Logitech's VR Ink[6]) as well as methods to snap the strokes on given points.

[4]Video: `https://bit.ly/35GdKqO`
[5]Source code: `https://bit.ly/3eQPHK0`
[6]VR Ink: `https://bit.ly/3ghpfcI`

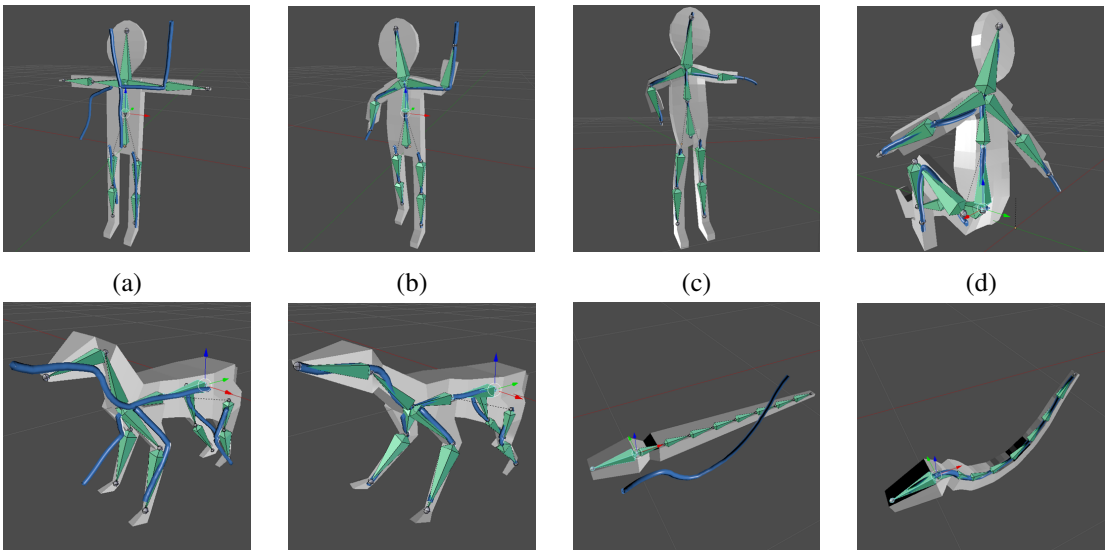

Figure 4: Examples of armatures articulated through 3D sketches.

- The high sampling rate of the controller movements could generate noisy sketches. Mechanisms could be introduced for sketch beautification.

4 Conclusions and future work

The proposal described above represents the baseline for a system able to perform a challenging computer graphics task, i.e., virtual character posing, by combining advantages brought by the use of sketch-based interfaces and VR technology. Besides tackling limitations mentioned above, experiments with end-users could be performed in the future to assess the effectiveness/intuitiveness of the proposed interaction method and to estimate the actual contribution of VR by collecting, e.g., the time required for posing different types of characters, data about users' tiredness, etc., possibly comparing it with traditional approaches. Moreover, effort could be devoted to integrating machine learning algorithms for making the system able to reconstruct the entire character pose or the overall animation by sketching only a few lines of the pose.

Acknowledgements

Work has been supported by VR@POLITO initiative.

References

[1] Paul C DiLorenzo. Premo: Dreamworks animation's new approach to animation. *IEEE Computer Graphics and Applications*, 35(4):14–21, 2015.

[2] Juncong Lin, Takeo Igarashi, Jun Mitani, Minghong Liao, and Ying He. A sketching interface for sitting pose design in the virtual environment. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1979–1991, 2012.

[3] Daniel Vogel, Paul Lubos, and Frank Steinicke. AnimationVR – Interactive controller-based animating in virtual reality. In *Proc. 1st Workshop on Animation in Virtual and Augmented Environments*, pages 1–6. IEEE, 2018.

[4] Alberto Cannavò, Claudio Demartini, Lia Morra, and Fabrizio Lamberti. Immersive virtual reality-based interfaces for character animation. *IEEE Access*, 7:125463–125480, 2019.

[5] Mikko Kytö, Krupakar Dhinakaran, Aki Martikainen, and Perttu Hämäläinen. Improving 3D character posing with a gestural interface. *IEEE Computer Graphics and Applications*, 37(1):70–78, 2015.

[6] Fabian Hahn, Frederik Mutzel, Stelian Coros, Bernhard Thomaszewski, Maurizio Nitti, Markus Gross, and Robert W. Sumner. Sketch abstractions for character posing. In *Proc. 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 185–191, 2015.

[7] Byungkuk Choi, Roger B Ribera, J. P. Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Junyong Noh. Sketchimo: Sketch-based motion editing for articulated characters. *ACM Transactions on Graphics*, 35(4):146:1–146:12, 2016.

[8] Alexandros Gouvatsos, Zhidong Xiao, Neil Marsden, and Jian J. Zhang. Posing 3D models from drawings. *Computer in Entertainment*, 15(2):2:1–2:14, 2017.

[9] Michelle Annett, Fraser Anderson, Walter F Bischof, and Anoop Gupta. The pen is mightier: Understanding stylus behaviour while inking on tablets. In *Proc. of Graphics Interface 2014*, pages 193–200. Canadian Information Processing Society, 2014.

[10] Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. Robust flow-guided neural prediction for sketch-based freeform surface modeling. In *Proc. SIGGRAPH Asia 2018*, page 238. ACM, 2018.

[11] Péter Borosán, Ming Jin, Doug DeCarlo, Yotam Gingold, and Andrew Nealen. Rigmesh: Automatic rigging for part-based shape modeling and deformation. *ACM Transactions on Graphics*, 31(6):1–9, 2012.

[12] Chen Mao, Sheng Feng Qin, and David Wright. Sketch-based virtual human modelling and animation. In *Proc. 8th International Symposium on Smart Graphics*, pages 220–223, 2007.

[13] Simone Barbieri, Nicola Garau, Wenyu Hu, Zhidong Xiao, and Xiaosong Yang. Enhancing character posing by a sketch-based interaction. In *Proc. SIGGRAPH 2016 Posters*, pages 56:1–56:2, 2016.

[14] Michael F Deering. Holosketch: A virtual reality sketching/animation tool. *ACM Transactions on Computer-Human Interaction*, 2(3):220–238, 1995.

[15] Jeff Gipson, Lauren Brown, Ed Robbins, Jose Gomez, Mike Anderson, Jose Velasquez, Jorge Ruiz, and Dan Cooper. VR Story production on Disney animation's cycles. In *ACM SIGGRAPH 2018 Talks*, pages 1–2, 2018.

[16] Natapon Pantuwong. A tangible interface for 3D character animation using augmented reality technology. In *Proc. 8th International Conference on Information Technology and Electrical Engineering*, pages 1–6. IEEE, 2016.

[17] Martin Guay, Marie-Paule Cani, and Rémi Ronfard. The line of action: An intuitive interface for expressive character posing. *ACM Transactions on Graphics*, 32:1–8, 2013.

[18] Mikhail Bessmeltsev, Nicholas Vining, and Alla Sheffer. Gesture3d: Posing 3D characters via gesture drawings. *ACM Transactions on Graphics*, 35:165:1–165:13, 2016.

[19] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proc. International Conference on Neural Networks*, pages 1942–1948. IEEE, 1995.

[20] Alan L Yuille and JJ Kosowsky. Statistical physics algorithms that converge. *Neural computation*, 6(3):341–356, 1994.

[21] Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. *Pattern Recognition*, 31(8):1019–1031, 1998.

[22] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.