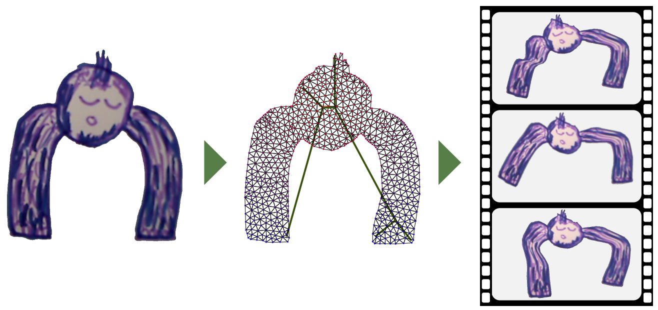
Towards Automatic Drawing Animation using Physics-based Evolution

Lasse Lingens
Robert W. Sumner
Stéphane Magnenat
ETH Zürich
Game Technology Center,
Universitätstrasse 25,
CH-8092 Zürich, Switzerland
llingens@student.ethz.ch
robert.sumner@inf.ethz.ch
stephane.magnenat@inf.ethz.ch



Interaction Design and Children (IDC '20 Extended Abstracts), June 21–24, 2020, London, United Kingdom
ACM 978-1-4503-8020-1/20/06.
https://doi.org/10.1145/3397617.3397842

Abstract

We demonstrate a system to automatically animate hand-drawn characters. Starting with skeleton extraction, meshing and vertex skinning, our system simulates characters using a neural network in a physics-based environment. Using an evolutionary algorithm, it searches for networks that move characters far while keeping a good posture. We validated the system through a user study with 26 participants. For most drawings (60 %), they felt satisfied with the generated animation, and in 76 % of cases, they wished to draw and animate additional characters. The participants reported mostly positive emotions after seeing the animations. Only a minority had feelings of strangeness or had negative emotions. This work demonstrates the possibility of creating an automated 2-D character animation system making little assumption on what is drawn. We believe that this work can enable more children to engage in creative play and explore their imagination.

Author Keywords

automatic animation; physics-based simulation; artificial evolution; skeleton-based animation; augmented creativity

CCS Concepts

•**Computing methodologies** → *Image and video acquisition;* **Physical simulation; •Human-centered computing** → *Human computer interaction (HCI);*

Introduction

Creative play is both engaging and educational. Finger painting, building with blocks, drawing, or dressing up – all such activities intimately link imagination and self expression with exploration and problem solving. With creative play, children have fun while learning new skills and enhancing their understanding of the world. However, with the popularity and ubiquity of digital devices, an increasing amount of play time happens in the digital world, such as with video games where the link to creativity is sometimes questionable. Instead of *creative play*, children simply *play*.

To address this situation, we recently developed an app that builds on the popularity of video games but lies squarely in the space of creative play [1]. This app allows children to draw game characters on paper with crayons or markers, and take a photo with their tablet's camera to import their drawings into the app as game sprites. Then, they can program these sprites through a visual language. While very successful with children, that app has a limitation: the sprites themselves are static and lack animation.

Given the success of animated feature films, it is clear that animation is very engaging for children. However, animation is notoriously difficult, and the film industry employs hundreds of highly skilled artists for each movie. Yet, recent progress in computer graphics and machine learning bring the vision of fully automated animation closer. In this work, we build on these ideas by exploring a pipeline for fully-automated animation of hand-drawn characters. Our **key contributions** include a **proof of concept** that automatic animation is feasible and a **roadmap** of the next steps.

Related work

Automatic animation in general is hard, especially for characters, because it is an ill-defined problem. Given a drawing, there are multiple ways to animate it, and prior knowledge must be provided to the system to choose between these different possibilities. Traditional character animation requires an artist to define a set of bones, and then to set weights associating each vertex with one or more bones and define animation curves. Semi-automatic sketching tools have followed this pipeline. In early works such as Yonemoto [9], the user explicitly defines the different object parts, the skeleton joining them, and then draws an overall animation. More recent works, for example Feng et al. [4] in the field of augmented reality, have analyzed the object shape through morphological operators [6] to define body parts, and let the user place end joints of a pre-defined skeleton structure. Until now, fully-automatic deformation has only been achieved in objects without bones. For example, computational geometry researchers such as Sorkine et al. [8] have used the shape of a character to deform it in an as-rigid-as-possible way. However, the resulting characters look as if they were made of a gummy material due to the lack of bones.

In this work, we aim at generating fully automatically characters drawn by humans, especially children. To solve the problem of the many possible animations for a given drawing, we take inspiration from evolutionary robotics [7], and use a neural network to control the skeleton animation within a 2-D physics-based simulator [3]. As in Feng et al. [4], we automatically extract the skeleton from the drawing using a morphological thinning operation [6], but where the latter asks the user to select parts that belong together, we use a set of heuristics to fully automatically generate a skeleton. Finally, we animate the original

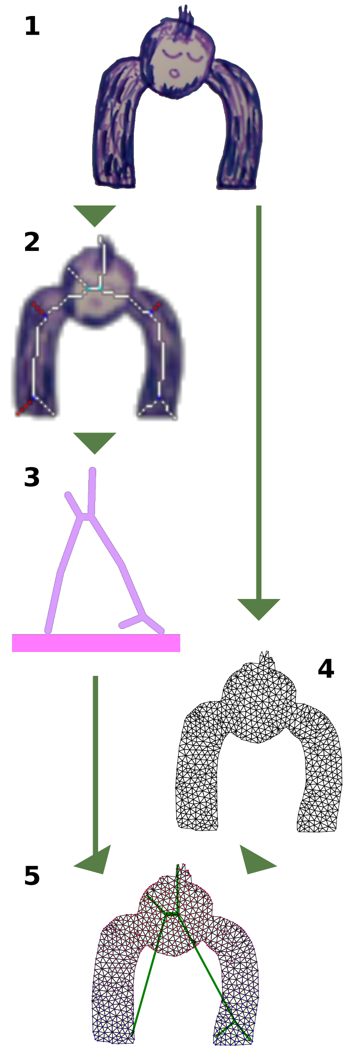


Figure 1: The animation generation pipeline.

drawing by meshing it and applying linear blend skinning as described in Kavan [5].

Generation method

Figure 1 shows the steps of the animation pipeline:

1. Extraction. The drawing with alpha channel is extracted from the background image by estimating a grayscale Gaussian distribution of the background pixels and considering pixels far from the mean to be foreground.

2. Skeletonization. The drawing is downsampled with Gaussian blur, and a skeleton is created using morphological thinning operations [6]. Junctions and line endings are detected using a Hit-And-Miss transform [2]. The resulting skeleton is cleaned-up by clustering small bones and splitting long ones.

3. Physical Simulation. The skeleton is converted into an articulated body with parts connected through revolute joints, which is run in a physics-based simulator[1]. Individual parts are constrained by an angle span, maximum torque and velocity. Besides the torque, which is proportional to the bone length, they are constant. This body is controlled by a feed-forward neural network with input being all joint angles and creature pose and output being all desired joint velocities. The network has one hidden layer of the same size as the output, and each layer is fully connected to the next. Network weights are optimized using an evolutionary algorithm[2]. The fitness function is the horizontal position of the animated body after a given time in the physical simulation, modified by a factor to favor keeping a pose close to the initial one. To avoid exploitation of numerical errors in the physics engine, a run is terminated early if the character flips or if a joint's rotation direction changes too often.

[1]Rust crate NPhysics, ver. 0.14, https://crates.io/crates/nphysics2d
[2]Rust crate revonet, ver. 0.2.1, https://crates.io/crates/revonet

4. Triangulation. To be animated, the drawing is transformed into a 2-D textured triangle mesh. The outline is generated with a marching squares algorithm, and then simplified. A first triangle mesh is generated with an ear clipping algorithm. The mesh is refined by repeatedly applying edge flipping, edge splitting, face splitting and vertex smoothing.

5. Vertex skinning. To animate the drawing given the skeleton movement, we perform vertex skinning [5]. Each vertex of the triangle mesh is associated with at most four bones from the skeleton. We compute the mesh walking distance to each bone, and we keep the four closest. We weight their contribution using a function of the distance through a Gaussian kernel. We export the result to the glTF binary format[3].

User study

To assess the effectiveness of our approach, we ran a user study with 26 participants who submitted 38 drawings. While our goal is to develop a system usable by children, in this pilot study our participants are mostly young adults (median approximately 26 years old). The reason is that the generated animations can be uncanny sometimes and we did not want to scare children. We interacted with the participants through emails, received their drawings, and sent them the resulting animations along with a link to a Google Forms questionnaire per drawing. The questionnaire had the following questions:

1. How do you feel about the generated animation?

[3]glTF is a standard 3D format, see https://www.khronos.org/gltf/

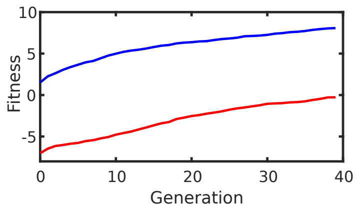


Figure 2: The best (top blue) and mean (bottom red) fitness averaged over the 38 drawings.

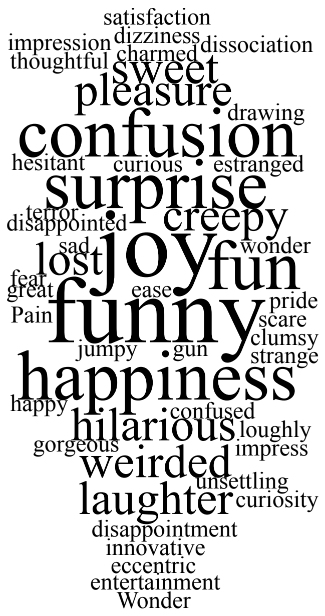


Figure 3: A word cloud of the emotions expressed by our user study participants.

2. I feel the desire to draw and view additional characters and the resulting animations.
3. The resulting animation gave me additional ideas for characters to draw.
4. Should I make my own video game, I would prefer the animated character over its static version.
5. Please write up to three emotions that you felt at the sight of the resulting animation or which you now associate with it.

Question 1 had possible answers from “Very Satisfied” to “Very Dissatisfied” (with corresponding smileys) and questions 2, 3, 4 from “Strongly Agree” to “Strongly Disagree”, all on a 5-point Likert scale. Question 5 was free text, and participants could optionally provide comments.

In this study, we also performed these manual steps:
Extraction. We extracted the drawing by hand using a photo editor.
Selection. For each drawing we ran 5 evolutions, and selected the best result in terms of motion distance, periodicity, and pose consistency of the movement.
Video preparation. Before sending the results to the users, we removed the initial, non-periodic part of the motion.

Results

We implemented our system using Rust[4]. For each drawing, we ran the evolutionary algorithm over 40 generations, with a population of 100 individuals, and 15 seconds of simulation per creature and generation. Figure 2 shows the best and mean fitness of each evolution averaged over the 38 drawings. In average, extraction and skeletonization takes 1.9 s, the physics-based simulation 64 s, and the triangulation and vertex skinning 1.4 s.

[4]https://www.rust-lang.org/

Most participants (60 %) felt satisfied with the generated animation (Figure 4). The method elicits curiosity, as 75 % of the participants wanted to draw additional characters and see them animated (Figure 5). Almost half (44 %) were inspired with new ideas of characters to draw after viewing the animation (Figure 6). A majority (60 %) preferred the animated version to a static version should they create a video game (Figure 7).

The emotions reported by the participants are summarized in Figure 3. We see that positive emotions such as “joy” and “fun” dominate. Then we can see a feeling of strangeness with words such as “surprise”, “confusion” and “laughter”. Finally, there is a wide span of low-frequency dissatisfaction.

Discussion

User perception. In Figure 8 we see that the satisfaction is highly correlated to the naturalness of the animation. In particular, when participants are dissatisfied, it is either because the result is uncanny, sometimes even spooky, or because the movement is not clear enough.

Limitations of the method. Our method explicitly assumes the character to have a skeleton. If the drawing consists primarily of a face or is a sponge-like creature, the result will not be ideal. We run the simulation in 2 dimensions, but 2-D characters are typically projections of 3-D characters, and solely animating them using a 2-D physics simulation is sub-optimal.

Limitations of the implementation. We do not yet enforce bones to be within the mesh. For example, if a leg is bent, the associated knee joint might lie outside. While we devised stopping conditions to avoid exploitation of the numerical instabilities of the physics engine, they still are present in some cases.

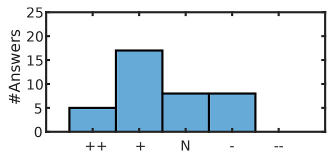


Figure 4: Q. 1, *How do you feel about the generated animation?*

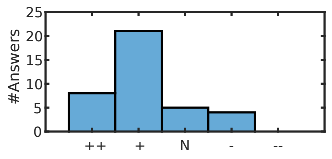


Figure 5: Q. 2, *I feel the desire to draw/animate more characters*.

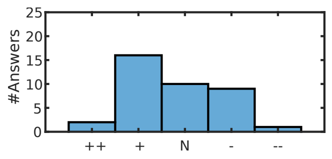


Figure 6: Q. 3, *Animation gave me new character ideas*.

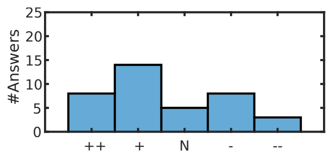


Figure 7: Q. 4, *I prefer animated vs static for a video game*.

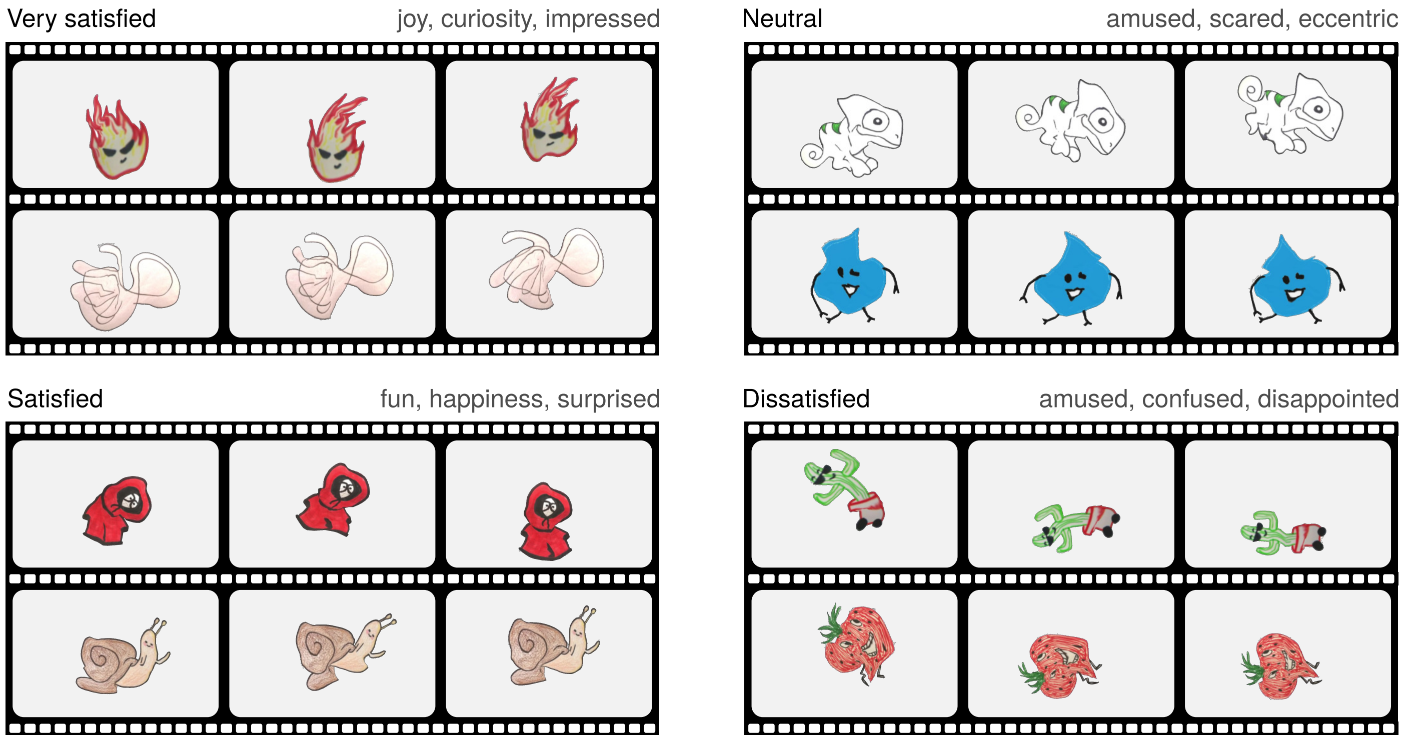

Very satisfied
joy, curiosity, impressed
Neutral
amused, scared, eccentric
Satisfied
fun, happiness, surprised
Dissatisfied
amused, confused, disappointed

Figure 8: Example animations for the different levels of satisfaction (no one was very dissatisfied), along with the top three associated emotions.

Future work. Besides improving the implementation, we foresee several changes that should improve the results. First, we could detect the kind of body parts bones represent, for example using machine learning, and use this information to simulate creatures in three dimensions. This approach should give more realistic gaits. Second, we could detect the class of creature drawn. If it is a marine creature, for example, we could use a different physics engine appropriate for underwater movement. Finally, if the creature is a face or sponge-like, we could use a deformable body simulation.

Conclusion

Building on a combination of techniques from computer graphics and evolutionary robotics, we showed that a fully-automated 2-D character animation system making few assumptions on what is drawn is possible. Our system delivers compelling animations, that, as shown by our user study, often elicit curiosity and creativity. We believe that this work can further encourage children to engage in creative play and exploration using their imagination.

Acknowledgments

The authors thank the user study participants for their time and Julia Chatain for her help in the preparation of the study.

REFERENCES

[1] Julia Chatain, Olivier Bitter, Violaine Fayolle, Robert W. Sumner, and Stéphane Magnenat. 2019. A Creative Game Design and Programming App. In *Motion, Interaction and Games (MIG '19)*. ACM. DOI: `http://dx.doi.org/10.1145/3359566.3360056`

[2] Edward R. Dougherty. 1992. *An Introduction to Morphological Image Processing*. Society of Photo Optical.

[3] Elahe Eskandari, Arash Ahmadi, Shaghayegh Gomar, Majid Ahmadi, and Mehrdad Saif. 2016. Evolving Spiking Neural Networks of artificial creatures using Genetic Algorithm. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 411–418. DOI: `http://dx.doi.org/10.1109/IJCNN.2016.7727228`

[4] Lele Feng, Xubo Yang, and Shuangjiu Xiao. 2017. MagicToon: A 2D-to-3D creative cartoon modeling system with mobile AR. In *Virtual Reality (VR)*. IEEE, 195–204. DOI: `http://dx.doi.org/10.1109/VR.2017.7892247`

[5] Ladislav Kavan. 2014. SIGGRAPH Course 2014 – Skinning: Real-time Shape Deformation. Part I: direct skinning methods and deformation primitives. (2014).

[6] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. 1992. Thinning Methodologies – A Comprehensive Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 9 (Sept. 1992), 869–885. DOI: `http://dx.doi.org/10.1109/34.161346`

[7] Jordan B. Pollack and Hod Lipson. 2000. The GOLEM project: evolving hardware bodies and brains. In *The Second NASA/DoD Workshop on Evolvable Hardware*. IEEE, 37–42. DOI: `http://dx.doi.org/10.1109/EH.2000.869340`

[8] Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Fifth Eurographics Symposium on Geometry Processing*. Eurographics Association, 109–116.

[9] Satoshi Yonemoto. 2012. A Sketch-based Skeletal Figure Animation Tool for Novice Users. In *Ninth International Conference on Computer Graphics, Imaging and Visualization*. IEEE, 37–42. DOI: `http://dx.doi.org/10.1109/CGIV.2012.18`