STAG: Smart Tools and Applications in Graphics (2018)
M. Livesu, G. Pintore and A. Signoroni (Editors)

CageLab: an Interactive Tool for Cage-Based Deformations

Sara Casti[1] Fabrizio Corda[1] Marco Livesu[2] Riccardo Scateni[1]

[1]University of Cagliari, Italy
[2]CNR IMATI, Genoa, Italy

Abstract

Posing a digital character by acting on the vertices of a coarse control cage is, after skeleton-based, probably the most widely used technique for digital animation. While skeleton-based techniques have been deeply researched and a variety of industrial and academic tools are available for it, cage-based techniques have historically received less attention. In recent years we observed an increasing interest in the field, which results in a growing number of publications both on algorithms for automatic or semi-automatic cage generation, and for smooth barycentric coordinates for general polyhedral meshes. We introduce CageLab: a novel research-oriented software tool that allows scholars and practitioners in general to get acquainted with cage-based animation in a lightweight and easy to use environment. Users can: (i) load digital characters and their associated cages, applying character deformations with a selection of the most widely used barycentric coordinates available in literature; (ii) compare alternative cages for a given digital character; (iii) compare alternative barycentric coordinates w.r.t their smoothness and locality within the cage; (iv) use CageLab for educational purposes, or to produce images and videos for scientific articles. We publicly release the tool to the community, with the hope to support this growth, and possibly foster even more research in the field.

CCS Concepts
•***Computing methodologies*** → ***Animation;*** *Shape modeling;*

1. Introduction

Animating a digital character is a fundamental task in computer graphics, with huge impact in the film and game industries. Animations are typically realized displaying a sequence of pictures (or *frames*) at certain interval rate. A small subset of these frames are generated by posing the digital character in a number of relevant stances, generating the so-called *key-frames*. The rest of the frames are computer generated, interpolating or extrapolating the key-frames [IMH05]. The generation of key-frames is often time consuming, as it requires a substantial amount of manual work. Professionals have at their disposal mainly two tools to pose a character: skeletons and cages.

In this paper we focus on cages, which have historically received less attention from the scientific community, but have recently gained more popularity, as witnessed by a number of new techniques for automatic or semi-automatic cage generation [LD17, SVJ15, XLX15, GPCP13, TTB12, XLG12], and new studies on smooth barycentric coordinates for general polytopes [BLTD16, ZDL*14, WPG12, JBPS11].

A cage is a low resolution control mesh which tightly envelopes a digital character without intersecting it. Each point on the surface of the character is defined as a linear combination of the cage vertices, creating a connection between the two. As a result, when a cage vertex is moved in space, the digital character follows it. The locality and the smoothness of the resulting deformation are the two key properties that determine the quality of an animation. These properties map one to one with the locality and smoothness of the underlying barycentric coordinates that were used to connect the character with the cage. Ideally, barycentric coordinates should be as smooth as possible inside (and nearby) the cage, and should also be very local (i.e. the deformation obtained by moving a node of the cage should not affect parts of the character that are distant from such node, otherwise the deformation will not be intuitive). Cage-wise, the quality of the animation mostly depends on the positioning of the nodes, which may favor or prevent certain movements. A cage should be *shape-aware*, meaning that its nodes should be located nearby the bending points of the character, and should also adhere to its skin, to avoid the global propagation of local deformations [NS13].

DOI: 10.2312/stag.20181299

We introduce CageLab, a software tool for the visualization, editing and assessment of animation cages. The main purpose of CageLab is to support this growing interest for cage-based animation, providing an open source and easy to install shared platform were researchers and practitioners can interact with a system that allows them to:

- **Animate a digital character**, setting a number of key-frames and interpolating between them. CageLab accepts data in the most common file formats used in our community (e.g. OFF, OBJ), and can internally compute barycentric coordinates of various types. Resulting deformations can be closely inspected in a 3D canvas, controlling that the impact of any action on the nodes of the cage is sufficiently smooth and local;
- **Evaluate a cage**, visualizing it on top of the digital character, and checking whether its nodes are well-positioned with respect to it (and the poses one wants to realize). CageLab can also be used to directly compare two given cages, posing the character with both of them and comparing the obtained deformations. To this end, users can exploit a convenient copy paste system for camera parameters, which allows to observe different versions of the same pose always from a fixed point of view, with same perspective and amount of zoom;
- **Evaluate barycentric coordinates**, plotting the relation between each cage vertex and the underlying digital character. CageLab uses the widespread jet map (from blue to red, spanning the HSV spectrum) to visualize the influence of a cage node directly on the surface of the character (Figure 3). As for cages, direct comparisons between alternative barycentric coordinates can be created by fixing a point of view and plotting cage-character attraction with respect to a specific node. These visuals are very popular in literature [ZDL*14], and allow to easily compare locality and smoothness of each tested coordinate. New barycentric coordinates can be loaded into the system in the form of ASCII files;
- **Take snapshots or videos** of an animation, obtained interpolating a sequence of key-frames. This is a useful feature for researchers, to create images for their papers and content for the accompanying videos;
- Last but not least, CageLab can be used for **educational** purposes, both for preparing educational material, but also as support tool to teach animation at universities and high schools.

In the remainder of the paper we briefly discuss the state of the art of cage-based animation, and present the details of the main features of the software.

2. Related work

In the key-framing approach the animators manually specify the pose of the character at finite set of frames, and the remaining poses are automatically computed via interpolation [IMH05]. Each key-frame is manually created by the animator, deforming the character by acting on *handles* [WJBK15]. In [Jac15] Jacobson identifies three types of handles (skeletons, cages and points), each one with its own strengths and limitations. Generally speaking, skeletons are used to pose the limbs of the character, while cages and points are better suited to add secondary motions, such as the swish of a cloak, or face gestures and other small scale skin deformations. CageLab is completely dedicated to cage type handles, we therefore review only the literature related to them.

2.1. Cages

First researches in this field were done at the beginning of digital film animation. The first method developed by Sederberg and Parry [SP86] is the Free Form Deformation (FFD), which uses three dimensional control lattices. This technique allows to smoothly deform the character through the lattice, but it is not flexible enough to realize complicated deformations like legs or arms movements. An extension of this work was proposed in [MJ96], and consists in recursive subdivision of the control lattice to obtain the topological flexibility needed for deformation.

Control lattices can hardly fit an articulated model. Cage-based animation can be seen as an evolution of FFD methods, where the lattice is substituted by a polyhedral mesh. Cages allow for a better approximation of the digital character at a far lower complexity than lattices. A number of desirable properties for cages are listed in [JDKL14] and [NS13]. Summarizing, the authors state that cages: (i) must completely envelope the character without intersecting it; (ii) must be a faithful representation of the character, meaning that they should resemble it and should not deviate too much from its skin; (iii) must be coarse, meaning that (i) and (ii) should be obtained with the least number of vertices possible.

Recent literature in the field focuses mostly on methods to generate a coarse animation cage for a given digital character. These works can be roughly classified into automatic and user assisted techniques. Automatic techniques are typically based on mesh simplification [CVM*96, BCWG09, SVJ15, DLM11] or voxelization [XLG09, XLG12, XLX15]. These methods are purely geometric, and tend to produce sub-optimal cages not always suitable for animation. In fact, control points may be badly positioned with respect to the bending points of the character, and may prevent to realize certain poses. User assisted methods allow the user to plug semantic information into the cage, typically using some simplified paradigm such as cutting planes [LD17] to roughly control where to place cage vertices. These methods tend to produce better cages, but are more time consuming and require experienced users. The analysis of pre-existing animation sequences to generate the cage that better fits them was also studied [TTB12], and can be used either for compression or to produce new animations with the resulting cage. Garcia and colleagues [GPCP13] presented various ways to combine multiple cages (possibly using different barycentric coordinates) in a semlessy yet hierarchical way, offering a powerful system for cage-based animation. Finally, cages have been also used by [JZvdP*08] to reduce skeleton-based skinning artifacts. CageLab does not include any cage generation facility, but can import and use cages produced with any of the aforementioned techniques (as well as manually crafted).

2.2. Barycentric Coordinates

Barycentric coordinates realize the connection between a character and its cage. They were first introduced by Möbius in 1827, and have been subsequentially generalized to 3D in many ways [Flo03, ZDL*14, HS08]. Mean Value Coordinates [FKR05, JSW05, FHK06] were the firsts to be introduced. They are generally smooth and well defined, but suffer from two drawbacks: they are not very local, and they may be negative for concave cages [JMD*07], leading to non intuitive deformations. Harmonic coordinates [DM06,JMD*07] were proposed in alternative, and effectively address both issues, being more local and guaranteed positive inside any cage. For completeness, in [LKCOL07] Lipman et. al proposed an evolution the of Mean Value Coordinates method which solves the negativity issue using a GPU visibility test. A step forward in cage based animation was achieved one year later with Green coordinates [LLCO08], which offer better preservation of surface details under deformations, producing more realistic animations. They use not only vertices but also cage normals, and relax the constraint of having the character completely inside the cage. In recent years, various biharmonic coordinates [JBPS11, WPG12] and local barycentric coordinates [ZDL*14] were introduced, offering better locality than previous methods. In particular, bounded biharmonic coordinates [JBPS11] received a lot of attention, because they offer for the first time a unified framework were cages, skeletons and point handles co-exist in the same deformation space. In its current version, CageLab can internally compute a selection of the aforementioned coordinates, namely the Mean Value Coordinates and the Green Coordinates. We plan to add more options in future versions. Alternative coordinates can be pre-computed outside CageLab for a specific cage and then imported into our tool with a text file.

3. Basics of cage-based deformation

We provide here some basic information on how cage-based deformation works. The same paradigm applies to any type of barycentric coordinates, with the only exception of Green Coordinates, which are treated separately.

In cage-based deformations, the position of the model vertices are expressed as affine sum of the cage vertices. Let us denote as $\mathcal{M}$ the model to be deformed, and $\mathcal{C}$ the cage. $\mathcal{M}$ is a polyhedral mesh and $\mathcal{C}$ is a coarse triangle mesh enveloping $\mathcal{M}$. The deformation formula is the following:

$$p_i = \sum_l \omega_l(p_i)c_l$$

where p_i is a point belonging to the $\mathcal{M}$, c_l is a point belonging to $\mathcal{C}$, and $\omega_l(p_i)$ is the weight function applied to the model vertex p_i and the cage vertex c_l.

The weights are calculated once in pre-processing, thus the model deformation can be performed in realtime. The weight values differ for the barycentric coordinates definition used to calculate them. As we have seen in section 2.2 each barycentric coordinates definition has different properties. We have implemented two of the most valuable definitions: *Mean Value* and the *Green Coordinates*. Their main differences are the deformation domain, the coordinates negativity, and the shape and details preservation. The Green Coordinates are characterized by a local domain, their coordinates values are non-negative and they can preserve the model shape and its surface details. In fact, the Green Coordinates formula is slightly different from the general one. They add a weight function defined over cage faces which allow to preserve the surface details. The Green's formula is the following:

$$p_i = \sum_l \omega_l(p_i)c_l + \phi_k(p_i)s_k t_k$$

where $\phi_k(p_i)$ is the weight computed for the cage faces, s_k is the scaling factor representing the stretch of the face t_k during the deformation, and t_k is a face of the cage.

4. CageLab

We present here the main features of our tool. Cage-based deformations have received growing attention in recent years. Similarly to tools that were released by the community and sustained the growth of skeleton-based techniques [BP07, JP*17], we believe that CageLab can sustain researchers and practitioners who want to improve the cage-based animation pipeline, as well as compare their ideas with the state of the art in the field.

CageLab is essentially a key-framing system to create a complete animation pipeline. In this way, the animation can be exported and easily imported in another external software for different final purposes. Note that professional software such as Maya, Blender or 3D Studio Max in general offer a wider set of functionalities, such as enveloping [LCF00], blend shapes [JTDP06] and a variety of other deformation methods. These tools are intended for professionals (i.e. animators), therefore they are difficult to master and may be overly complex or intimidating for a young researcher.

4.1. User-Interface

During the development of CageLab (fig. 1), we have been extremely motivated to develop a user-friendly as well as lightweight User Interface.

The main window of CageLab is composed of three sections:

- The central section includes the Canvas, where the three-dimensional character mesh and the cage are represented, and where the user can interact with them.
- On the right side, the Tools sidebar enables the user to select the canvas interaction modes, and activate other features described in the next sections.
- On the left side, there are four panels:
  - The first one, on the top, is related to the **FBX Importer**. It is useful to open a compatible fbx file (containing the character mesh and its cage).
  - The second one, the **Character Manager panel**, enables the user to configure the settings related to the character mesh rendered in the canvas.

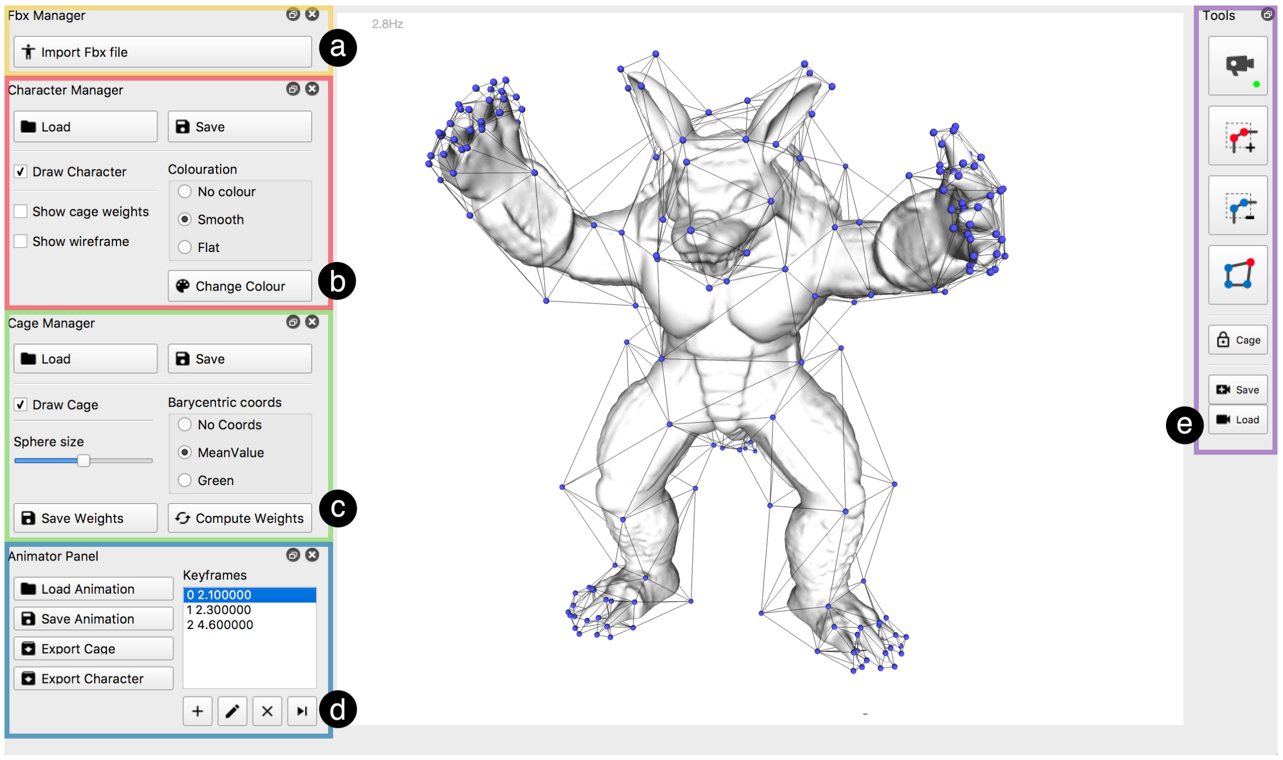


Figure 1: *The CageLab User Interface. On the left side the FBX Importer is highlighted in yellow (a), the Character Manager panel in red (b), the Cage Manager panel in green (c), and the Animator Panel in blue (d). On the right side there is the Tools sidebar (e). The central part of the UI includes the canvas.*

– In the **Cage Manager panel**, the user can configure the settings related to the cage rendered in the canvas and to the cage weights.
– The last one, the **Animator Panel**, enables the user to import and export the cage animation, and manage all the keyframes that compose the animation.

In the next paragraphs we will discuss in details every single function provided by the User Interface (UI).

4.2. The Canvas

The Canvas is the UI element used to render the cage and the relative character mesh, and to directly interact with the user. After the interaction mode is selected through the sidebar or by using the keyboard shortcuts, the user may use the mouse click or the mouse wheel to perform different tasks, such as camera movements, cage vertex selection or deselection and cage deformation. These actions are described in section 4.3.

The three-dimensional character mesh will be rendered with the graphical settings specified by the user in the *Character Manager* panel. The cage will be rendered as a wireframe mesh, with each vertex (called also handle) rendered as coloured sphere, red if selected, blue otherwise. Selected vertices are the ones involved in the deformation process.

In the lower side of the canvas small snippets of text with graphical hints and feedback are shown. They help the final user to understand which action is being performed.

4.3. Tools sidebar

The Tools sidebar on the right side of the User Interface allows to perform several actions, like the activation of different interaction modes.

The first four buttons from the top represent the available interaction modes (Camera Mode, Selection Mode, Deselect Mode, Deformation Mode). The active Interaction mode will be characterized by a green dot. The fifth button allows to lock the cage, not allowing the user to perform deformation on it. The sixth button (Camera Save) allows to Save the current camera point of view, allowing to restore it later after a modification, using the last button (Camera Load).

In order to make it easier for the user to understand which action is performed by each command button, we designed every action button trying to use clear and intuitive icons.

4.3.1. Camera Mode

Through the activation of the Camera Mode (see icon aside), the scene camera and the point of view can be manipulated by the user. When this mode is active, the interaction with the camera is made possible by moving the mouse over the canvas while the following buttons are pressed (as a typical 3D modeling software):

- The **left mouse button** rotates the camera
- The **right mouse button** translates the camera
- The **mouse wheel** scrolling enables the user to zoom in/zoom out the scene

This Interaction Mode can also be activated using the **C** keyboard key and is the default interaction mode.

4.3.2. Select/Deselect Cage Vertex Mode

These interaction modes (see icons aside), allows the user to select or deselect one or multiple handles of a cage. The selected handles will, then, be involved in the deformation process. To select/deselect a single handle, the user must simply click on it. To choose multiple handles, the user needs to press the left mouse button on the canvas, move the cursor over the desired handles, and then release the left button.

This interaction mode can be activated by pressing the **S** keyboard button for selection or **R** for deselection. Besides, if another interaction mode has been already activated, it is also possible to select the cage vertices preserving the active interaction mode, by pressing the **SHIFT** keyboard button for selection or **ALT** for deselection while clicking and dragging the mouse on the desired area. Once the SHIFT or ALT key are released, the previous interaction mode is restored.

4.3.3. Cage Deformation Mode

The Cage Deformation Mode (see icon aside), enables the user to deform the selected cage handles by moving them in space and consequently deforming the associated mesh.

It is possible to rotate the handles along their barycenter by clicking the left mouse button and dragging the cursor on the canvas. By clicking the right mouse button and dragging the cursor, it is possible to translate the selected cage vertices. The user can also scroll the mouse wheel to expand or contract the handles around their centroid, in order to inflate or deflate the mesh.

Every time a cage vertex deformation is performed, this deformation will be propagated to its mesh accordingly to the selected barycentric coordinate.

This interaction mode can be also activated by pressing the **D** button, or temporarily pressing the **CTRL** key during the mouse manipulation. Using the **x**, **y** or **z** key it is possible to constrain the cage vertices rotation and translation along the x, y and z axis of the camera point of view.

4.4. Character Manager panel

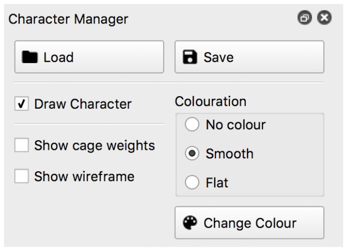

Character Manager
Load | Save
[✓] Draw Character
[ ] Show cage weights
[ ] Show wireframe
Colouration: ( ) No colour [✓ Smooth] ( ) Flat
Change Colour

Figure 2: *A screenshot of the Character Manager panel.*

The Character Manager panel (fig. 2) provides all the functionalities and personalization settings related to the character mesh that is rendered into the canvas and is deformable using the cage.

Load and **Save** buttons allow the user to import (or export) a triangle mesh file. The format used for these operations is the *.obj*, *.off* or *.ply* format.

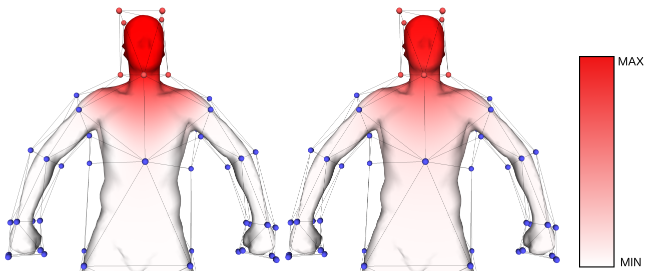
The **Colouration** radio buttons allow the user to choose the rendering options of the character mesh, using a smooth or a flat triangle shading.

The **Draw Character** checkbox can activate or deactivate the rendering of the character mesh on the canvas.

The **Show Wireframe** checkbox enables or disables the rendering of the character mesh wireframe. It is possible to render only the wireframe, without showing the mesh surface (flat or smooth), activating the wireframe checkbox and using the *No Colour* colouration setting.

The **Change Colour** button allows the user to choose the character mesh colour.

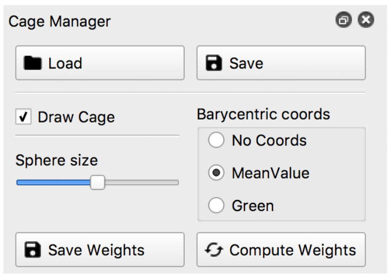
The **Show cage weights** checkbox (fig. 3) allows the user to observe the influence of the selected cage vertices over the character mesh, based on the current cage weights. The red parts of the character are the areas more involved by the selected cage vertices (or handles) during the cage deformation process. The blue parts, instead, are not influenced by those handles.



(a) *MVC* **(b)** *GC*

Figure 3: *In order to compare the smoothness and locality of alternative barycentric coordinates, CageLab allows to plot them with respect to a selection of cage node (see red spheres). This selection can be composed by a single node or by a set of the cages handles. In this example Mean Value (left) and Green (right) coordinates are shown. As can be noticed, Green are a bit less local.*

4.5. Cage Manager panel



Draw Cage [✓]

Barycentric coords: No Coords [✓ MeanValue] Green

Figure 4: *A screenshot of the Cage Manager panel.*

The Cage Manager panel (fig. 4) provides all the functionalities and settings relative to the cage that is rendered into the canvas.

Load and **Save** buttons allow to import (or export) the cage mesh from (or in) a file on the hard drive. The file will be saved in *.obj*, *.off* or *.ply* format, which represents the cage as a triangle mesh.

The **Draw** checkbox allows the user to activate/deactivate the cage rendering on the canvas.

The size of the cage spheres is set through the **Sphere size** slider. By default, this value is set to 0.5% of the diagonal of the cage bounding box. By moving the slider to the left or to the right, the sphere size may be decreased or increased.

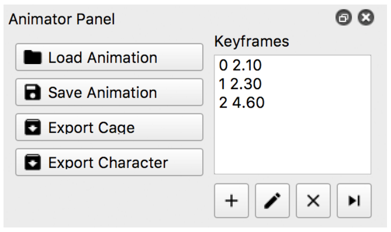
The **Compute Weights** button allows the computation of the *Mean Value Coordinates (MVC)* and *Green Coordinates*, which, in this way, can be used in the deformation process. Subsequently, this button activates the *Barycentric Coords* selection radio-buttons.

The **Barycentric Coords** radio-buttons allow the user to choose what kind of barycentric coordinates must be used in order to generate the deformation of the character mesh using the cage (MVC or Green).

With the **No Coords** setting, the deformation of the character will be disabled. This is particularly useful if we want to edit the cage easily, moving its vertices to better envelop the mesh but without generating a character deformation.

The **Save Weights** button, allows the user to save on a text file the active barycentric coordinates of the current character.

4.6. Animator panel



0 2.10
1 2.30
2 4.60

Figure 5: *A screenshot of the Animator panel.*

The Animator panel (fig. 5) provides all the functionalities needed to define the keyframe for the character animation.

On the right side of the panel, a list of all the animation keyframes is available. Each keyframe is defined with its sequence number and its timing (in seconds). Clicking on a keyframe on this list, it will be shown in the canvas.

The user can add, edit and erase a keyframe. Each operation can be performed through the dedicated buttons placed below the keyframes list.

When all the keyframes are defined, the user is able to save the animation sequence on a txt file using the **Save Animation** button. The saved animation can be reloaded in another session using the **Load Animation** button.

Using the **Export Cage** or **Export Character** buttons the user is able to export the sequence of all the deformed cage keyframes or character keyframes. Every keyframe is saved as a single obj or ply. The name of each file starts with a user defined string and the timing of the keyframe.

5. Implementation details

We have implemented our tool as a single threaded C++ application on a MacBook Pro equipped with a 2,7GHz Intel Core i5 and 8 GB of RAM. Our tool relies on the **Qt Framework** and it makes use of **Eigen** [GJ*10] to perform mathematical operations and the library **libQGLViewer** for the creation of the user interface. The UI icons come from the Material Design icon library, but some graphical elements are designed by us. We use FBX SDK to import the fbx files. The developed tool has been successfully tested under both MacOs (Yosemite and Sierra) and Linux (Ubuntu and Elementary) platform.

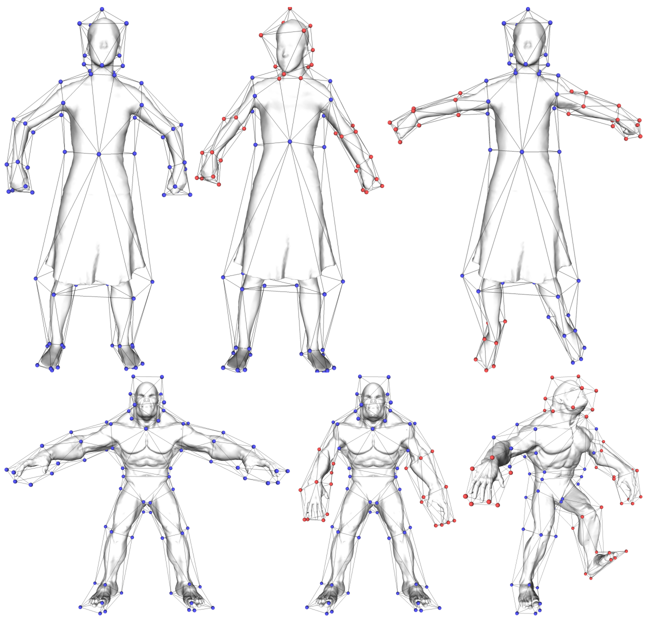


Figure 6: *Some deformations are shown.*

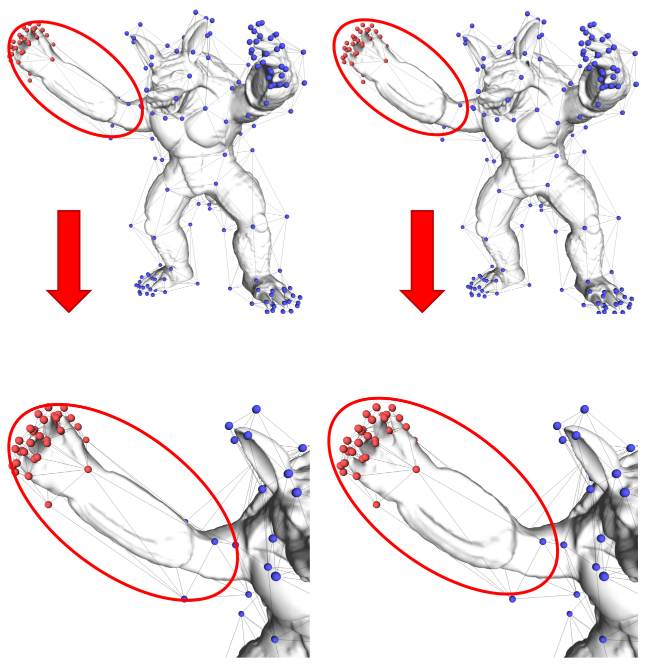


Figure 7: *Stretching Armadillo's arm with Mean Value (left) and Green (right) coordinates. Green coordinates better preserve surface details (see closeup). CageLab allows to switch between them in real time, so that the use can spot the differences and change barycentric coordinates depending on the intended deformation.*

6. Conclusions and future works

We presented CageLab, a novel tool for interactive cage-based deformations of digital characters. CageLab is intended for researchers and practitioners who want to get acquainted with (and improve on) the digital animation pipeline. It allows users to perform cage-based deformations using two of the most popular barycentric coordinates (Mean Value and Green coordinates); it allows to compare alternative cages for the same character; and to compare different differential coordinates (and the deformations they produce). It is also possible to define, export and import animation key-frames. We publicly release the tool to the community, with the hope to support the recent cage-based animation growth that we observed in our community, and possibly foster even more research in the field. We plan to extend CageLab with additional barycentric coordinates, as well as new features for deformations and animation.

Similarly to other research oriented tools [BTP*18], alongside the source code we will release a database of publicly available characters and cages produced with state of the art methods. With this, we hope to create the basis for a benchmark on cage generation, where new methods can be applied to a set of known digital characters for which cages produced with alternative methods are known and comparisons can be made.

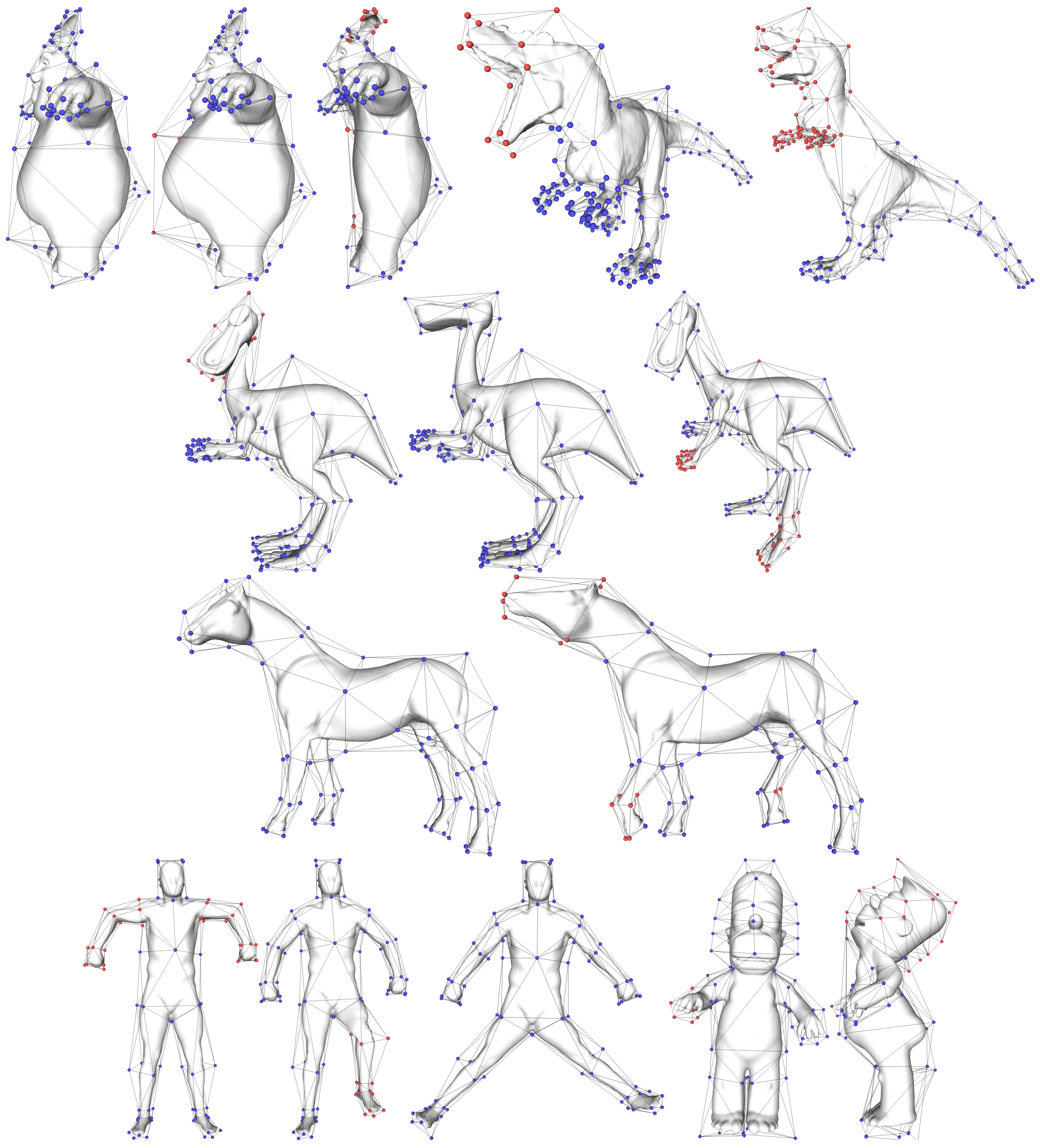


Figure 8: *An overview of cage-based deformations performed through CageLab.*

References

[BCWG09] Ben-Chen M., Weber O., Gotsman C.: Spatial deformation transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 67–74. URL: http://doi.acm.org/10.1145/1599470.1599479, doi:10.1145/1599470.1599479. 2

[BLTD16] Budninskiy M., Liu B., Tong Y., Desbrun M.: Power coordinates: a geometric construction of barycentric coordinates on convex polytopes. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 241. 1

[BP07] Baran I., Popović J.: Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG) 26*, 3 (2007), 72. 3

[BTP*18] Bracci M., Tarini M., Pietroni N., Livesu M., Cignoni P.: Hexalab. net: an online viewer for hexahedral meshes. *arXiv preprint arXiv:1806.06639* (2018). 7

[CVM*96] Cohen J., Varshney A., Manocha D., Turk G., Weber H., Agarwal P., Brooks F., Wright W.: Simplification envelopes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 119–128. URL: http://doi.acm.org/10.1145/237170.237220, doi:10.1145/237170.237220. 2

[DLM11] Deng Z.-J., Luo X.-N., Miao X.-P.: Automatic cage building with quadric error metrics. *Journal of Computer Science and Technology 26*, 3 (May 2011), 538. URL: https://doi.org/10.1007/s11390-011-1153-4, doi:10.1007/s11390-011-1153-4. 2

[DM06] DeRose T., Meyer M.: Harmonic coordinates. In *Pixar Technical Memo 06-02, Pixar Animation Studios* (2006), Citeseer. 3

[FHK06] Floater M. S., Hormann K., Kós G.: A general construction of barycentric coordinates over convex polygons. *advances in computational mathematics 24*, 1-4 (2006), 311–331. 3

[FKR05] Floater M. S., KÃşs G., Reimers M.: Mean value coordinates in 3d. *Computer Aided Geometric Design 22*, 7 (2005), 623 – 631. Geometric Modelling and Differential Geometry. URL: http://www.sciencedirect.com/science/article/pii/S0167839605000725, doi:https://doi.org/10.1016/j.cagd.2005.06.004. 3

[Flo03] Floater M. S.: Mean value coordinates. *Computer aided geometric design 20*, 1 (2003), 19–27. 3

[GJ*10] Guennebaud G., Jacob B., et al.: Eigen v3. http://eigen.tuxfamily.org, 2010. 6

[GPCP13] García F. G., Paradinas T., Coll N., Patow G.: *cages:: A multilevel, multi-cage-based system for mesh deformation. *ACM Trans. Graph. 32*, 3 (July 2013), 24:1–24:13. URL: http://doi.acm.org/10.1145/2487228.2487232, doi:10.1145/2487228.2487232. 1, 2

[HS08] Hormann K., Sukumar N.: Maximum entropy coordinates for arbitrary polytopes. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 1513–1520. 3

[IMH05] Igarashi T., Moscovich T., Hughes J. F.: Spatial keyframing for performance-driven animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 107–115. URL: http://doi.acm.org/10.1145/1073368.1073383, doi:10.1145/1073368.1073383. 1, 2

[Jac15] Jacobson A.: Breathing life into shapes. *IEEE Computer Graphics and Applications 35*, 5 (2015), 92–100. doi:doi.ieeecomputersociety.org/10.1109/MCG.2015.110. 2

[JBPS11] Jacobson A., Baran I., Popović J., Sorkine O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (July 2011), 78:1–78:8. URL: http://doi.acm.org/10.1145/2010324.1964973, doi:10.1145/2010324.1964973. 1, 3

[JDKL14] Jacobson A., Deng Z., Kavan L., Lewis J.: Skinning: Real-time shape deformation. In *ACM SIGGRAPH* (2014), vol. 22. 2

[JMD*07] Joshi P., Meyer M., DeRose T., Green B., Sanocki T.: Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3 (July 2007). URL: http://doi.acm.org/10.1145/1276377.1276466, doi:10.1145/1276377.1276466. 3

[JP*17] Jacobson A., Panozzo D., et al.: libigl: A simple C++ geometry processing library, 2017. http://libigl.github.io/libigl/. 3

[JSW05] Ju T., Schaefer S., Warren J.: Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 561–566. 3

[JTDP06] Joshi P., Tien W. C., Desbrun M., Pighin F.: Learning controls for blend shape based realistic facial animation. In *ACM Siggraph 2006 Courses* (2006), ACM, p. 17. 3

[JZvdP*08] Ju T., Zhou Q.-Y., van de Panne M., Cohen-Or D., Neumann U.: Reusable skinning templates using cage-based deformations. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 122:1–122:10. URL: http://doi.acm.org/10.1145/1409060.1409075, doi:10.1145/1409060.1409075. 2

[LCF00] Lewis J. P., Cordner M., Fong N.: Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 165–172. URL: http://dx.doi.org/10.1145/344779.344862, doi:10.1145/344779.344862. 3

[LD17] Le B. H., Deng Z.: Interactive cage generation for mesh deformation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2017), I3D '17, ACM, pp. 3:1–3:9. URL: http://doi.acm.org/10.1145/3023368.3023369, doi:10.1145/3023368.3023369. 1, 2

[LKCOL07] Lipman Y., Kopf J., Cohen-Or D., Levin D.: Gpu-assisted positive mean value coordinates for mesh deformations. In *Symposium on geometry processing* (2007). 3

[LLCO08] Lipman Y., Levin D., Cohen-Or D.: Green coordinates. *ACM Transactions on Graphics (TOG) 27*, 3 (2008), 78. 3

[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* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 181–188. URL: http://doi.acm.org/10.1145/237170.237247, doi:10.1145/237170.237247. 2

[NS13] Nieto J. R., Susín A.: Cage based deformations: a survey. In *Deformation models*. Springer, 2013, pp. 75–99. 1, 2

[SP86] Sederberg T. W., Parry S. R.: Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 151–160. URL: http://doi.acm.org/10.1145/15886.15903, doi:10.1145/15886.15903. 2

[SVJ15] Sacht L., Vouga E., Jacobson A.: Nested cages. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 170:1–170:14. URL: http://doi.acm.org/10.1145/2816795.2818093, doi:10.1145/2816795.2818093. 1, 2

[TTB12] Thiery J.-M., Tierny J., Boubekeur T.: Cager: Cage-based reverse engineering of animated 3d shapes. *Comput. Graph. Forum 31*, 8 (Dec. 2012), 2303–2316. URL: http://dx.doi.org/10.1111/j.1467-8659.2012.03159.x, doi:10.1111/j.1467-8659.2012.03159.x. 1, 2

[WJBK15] Wang Y., Jacobson A., Barbič J., Kavan L.: Linear subspace design for real-time shape deformation. *ACM Trans. Graph. 34*, 4 (July 2015), 57:1–57:11. URL: http://doi.acm.org/10.1145/2766952, doi:10.1145/2766952. 2

[WPG12] Weber O., Poranne R., Gotsman C.: Biharmonic coordinates. *Comput. Graph. Forum 31*, 8 (Dec. 2012), 2409–2422. URL: http://dx.doi.org/10.1111/j.1467-8659.2012.03130.x, doi:10.1111/j.1467-8659.2012.03130.x. 1, 3

[XLG09] Xian C., Lin H., Gao S.: Automatic generation of coarse bounding cages from dense meshes. In *Shape Modeling and Applications, 2009. SMI 2009. IEEE International Conference on* (2009), IEEE, pp. 21–27. 2

[XLG12] Xian C., Lin H., Gao S.: Automatic cage generation by improved obbs for mesh deformation. *The Visual Computer 28*, 1 (Jan 2012), 21–33. URL: https://doi.org/10.1007/s00371-011-0595-6, doi:10.1007/s00371-011-0595-6. 1, 2

[XLX15] Xian C., Li G., Xiong Y.: Efficient and effective cage generation by region decomposition. *Computer Animation and Virtual Worlds 26*, 2 (2015), 173–184. 1, 2

[ZDL*14] Zhang J., Deng B., Liu Z., Patanè G., Bouaziz S., Hormann K., Liu L.: Local barycentric coordinates. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 188:1–188:12. URL: http://doi.acm.org/10.1145/2661229.2661255, doi:10.1145/2661229.2661255. 1, 2, 3