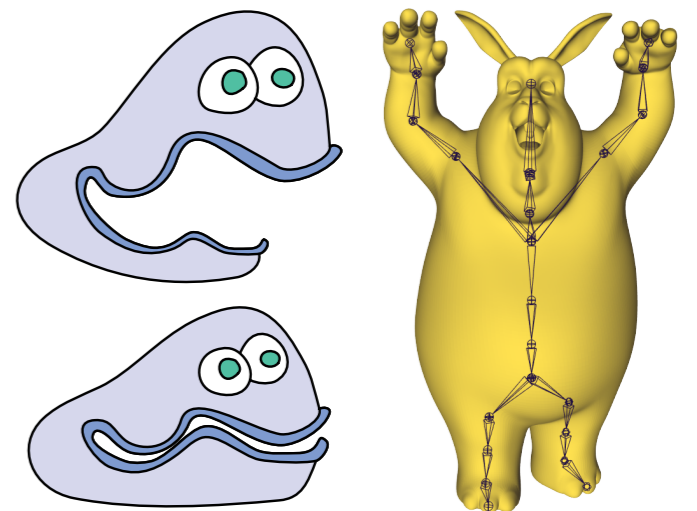


# Skinning Cubic Bézier Splines and Catmull-Clark Subdivision Surfaces

Songrun Liu    George Mason University  
Alec Jacobson    Columbia University  
Yotam Gingold    George Mason University



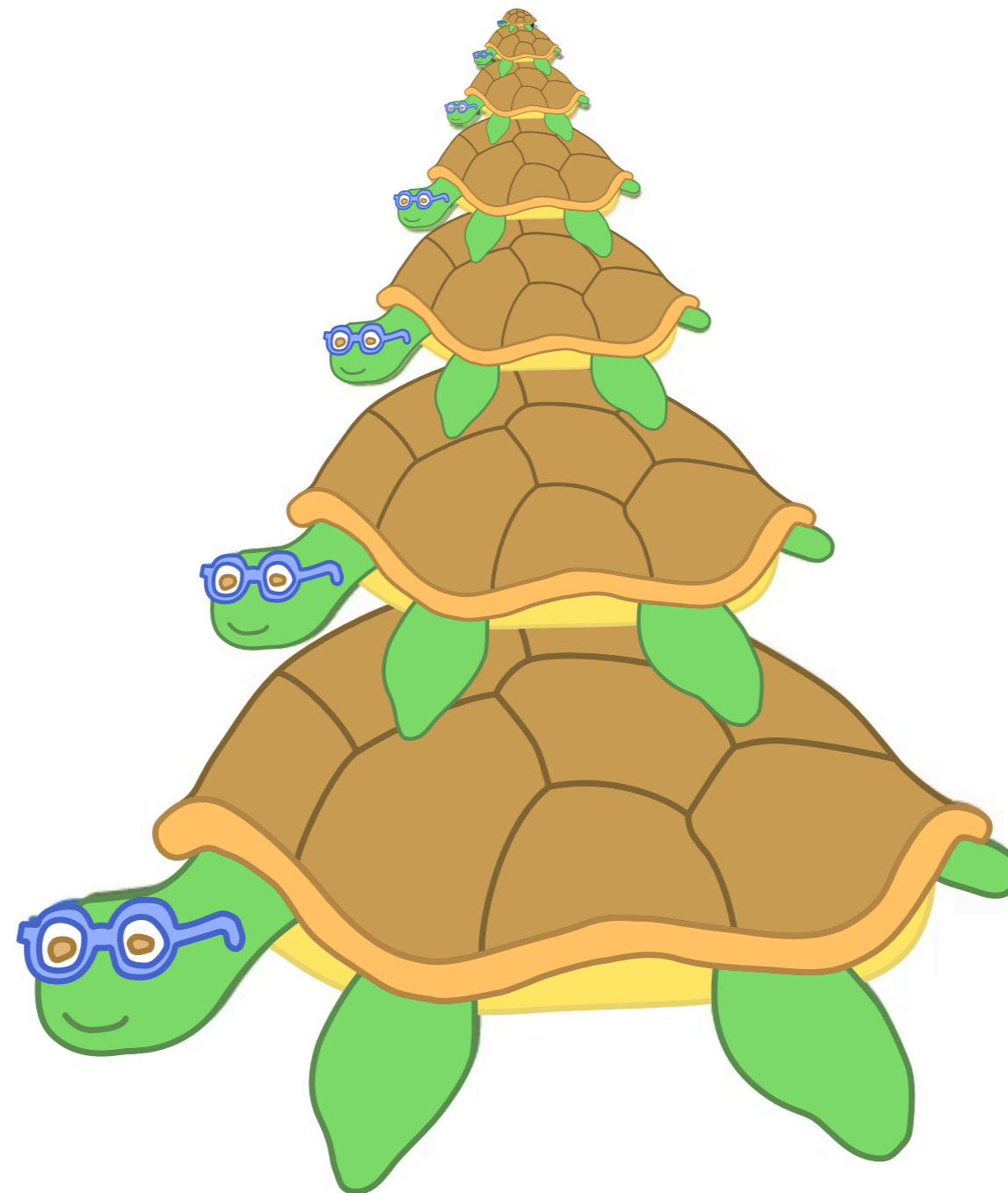
# Raster Deformation



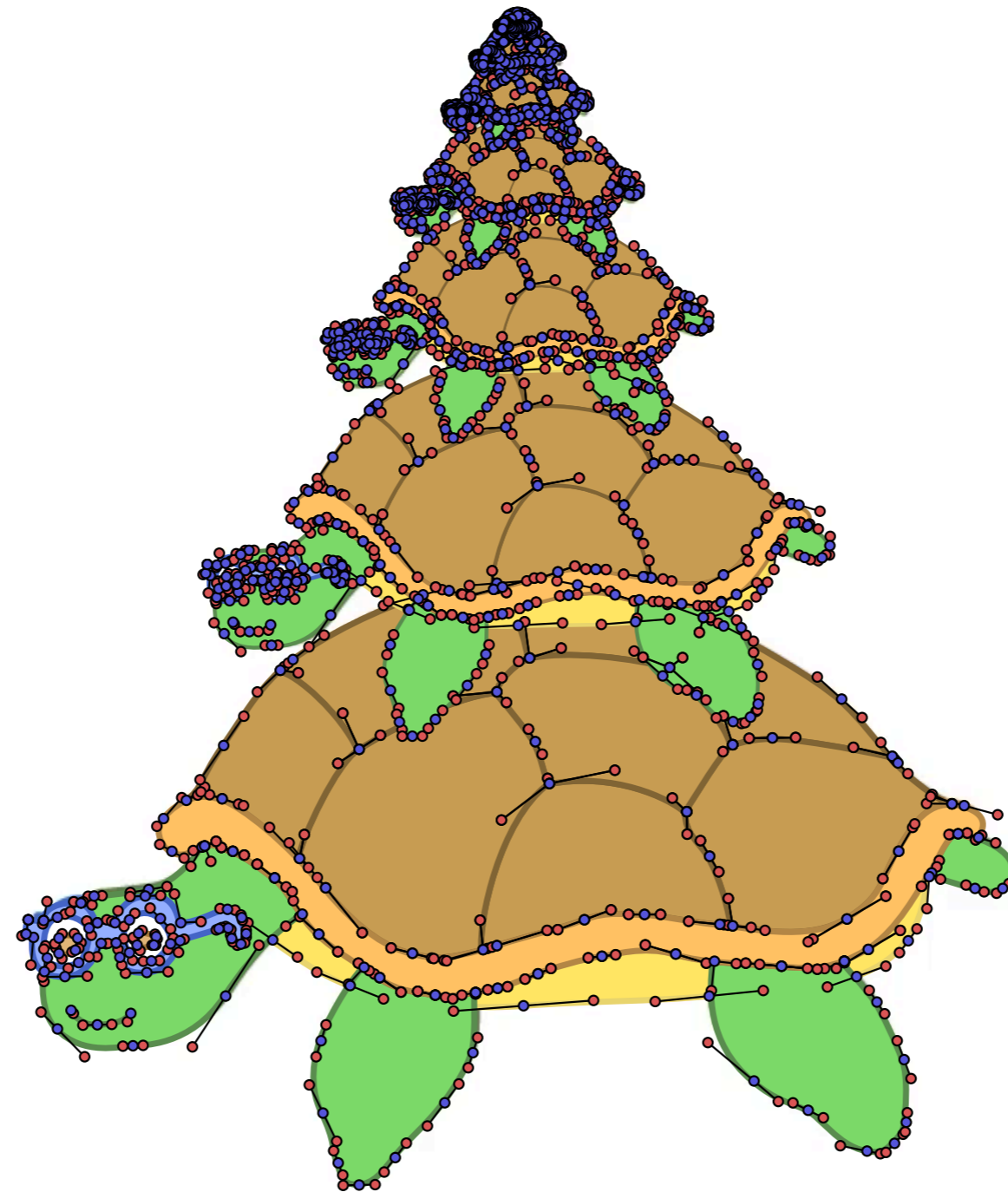
# Raster Deformation



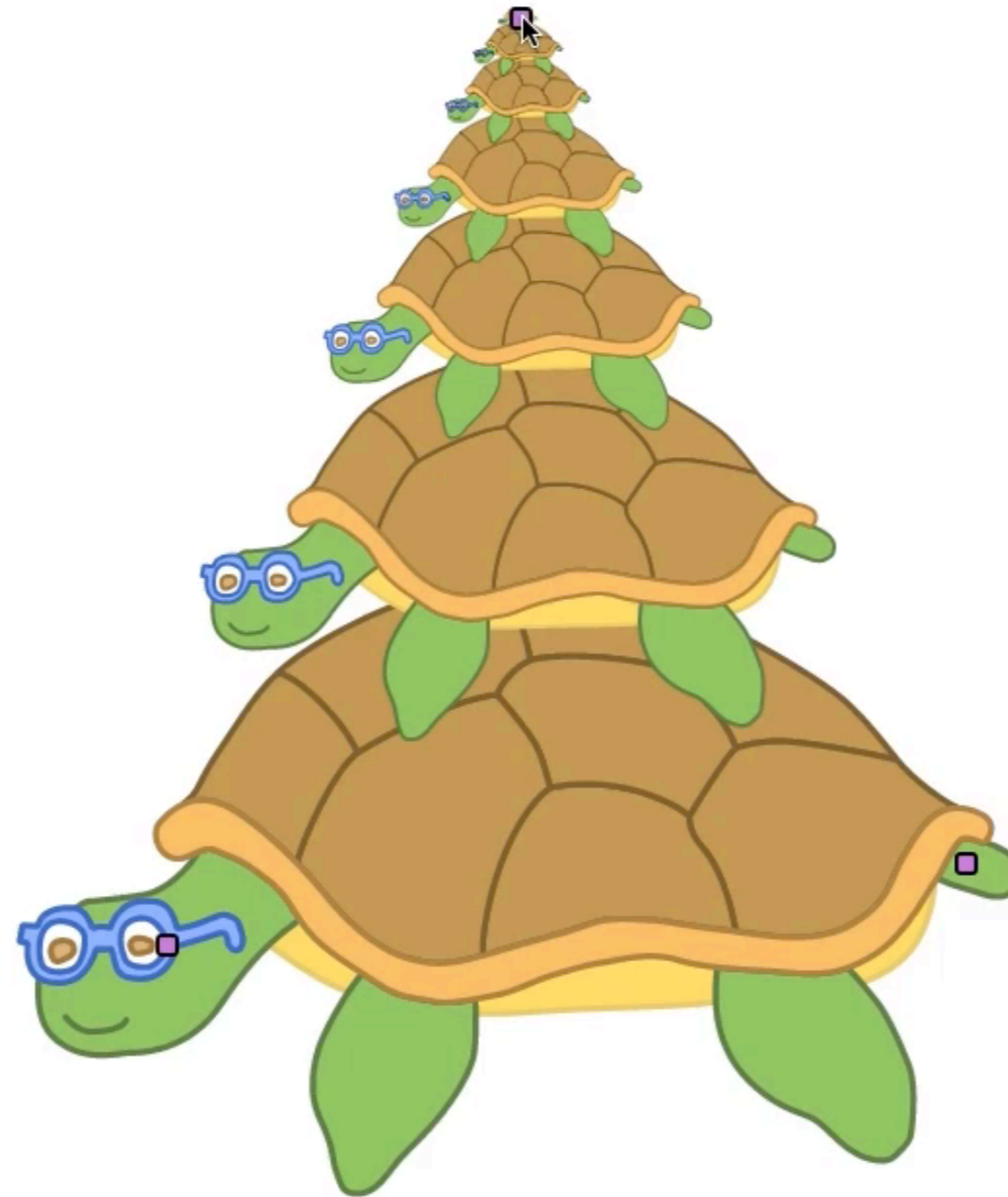
# Vector Graphics Deformation



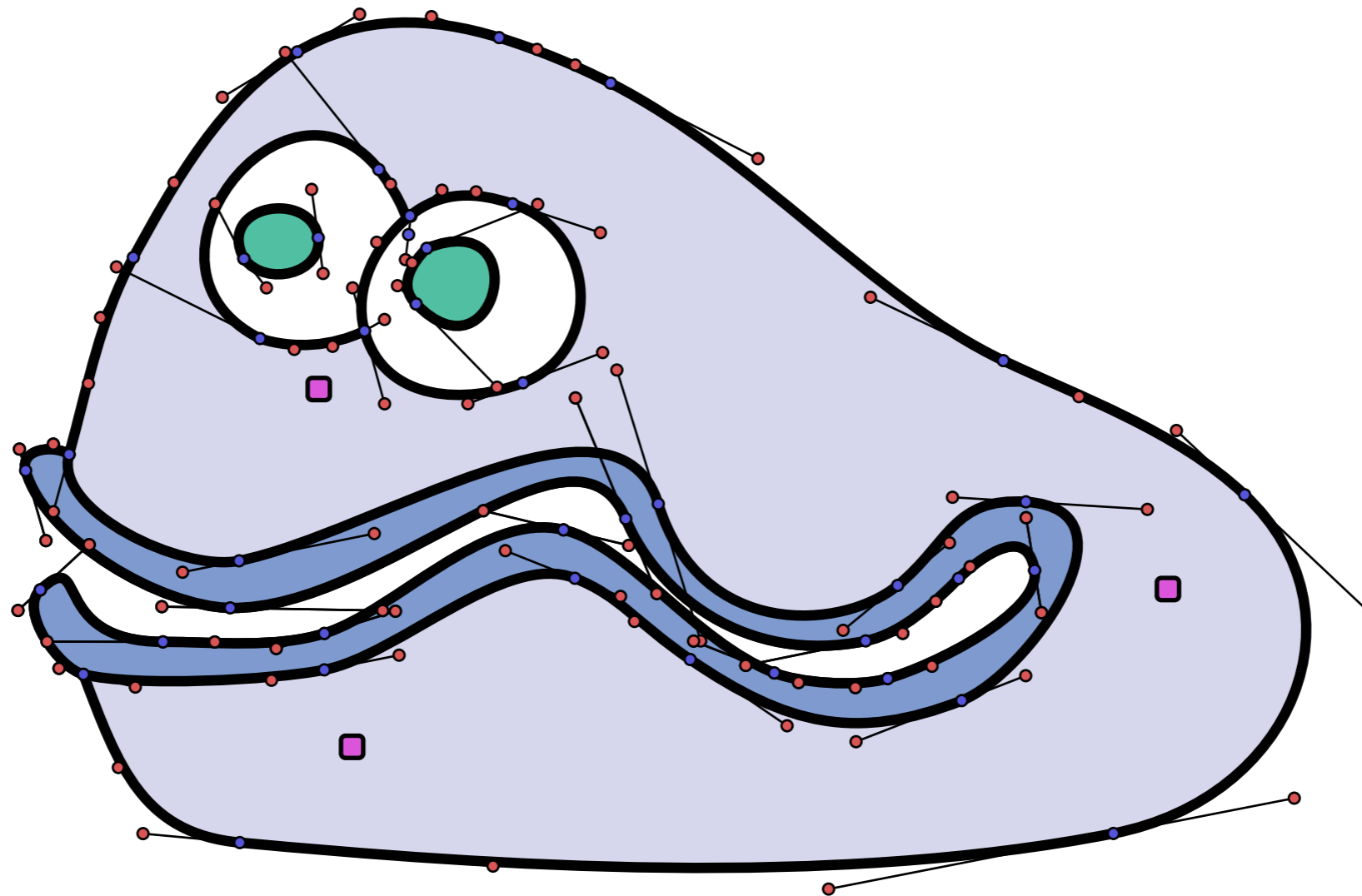
# Vector Graphics Deformation



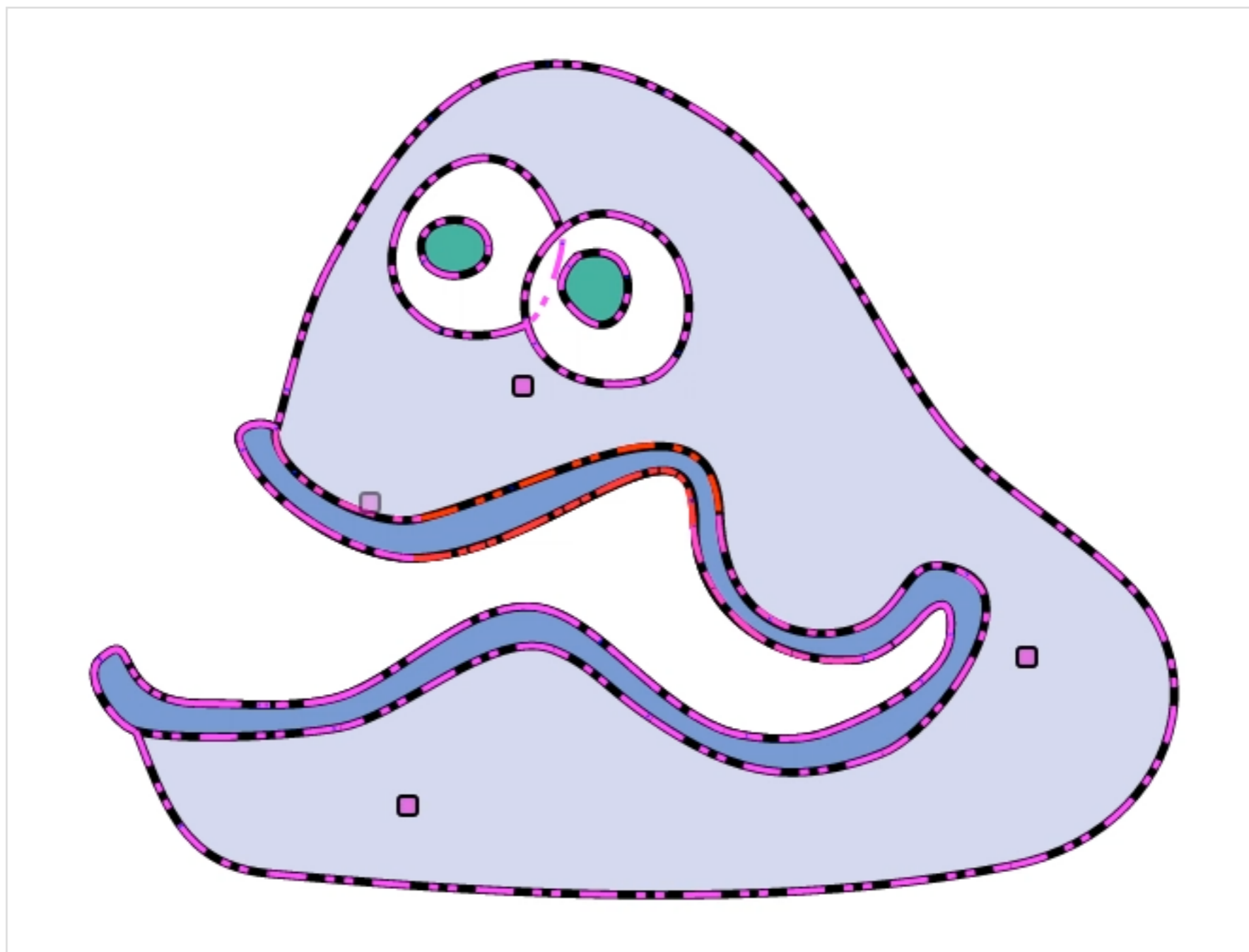
# Vector Graphics Deformation



# Our Approach

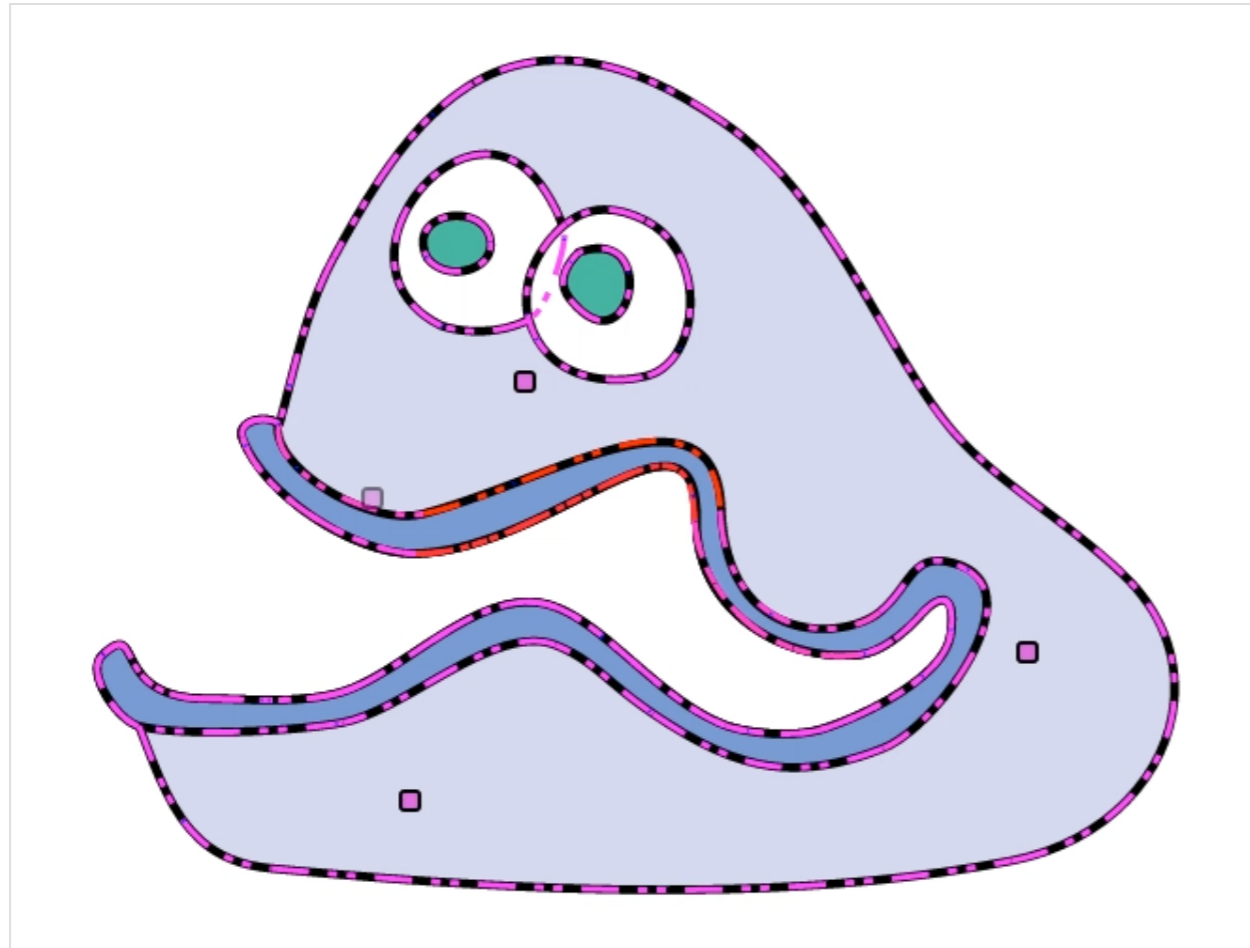


# Our Approach

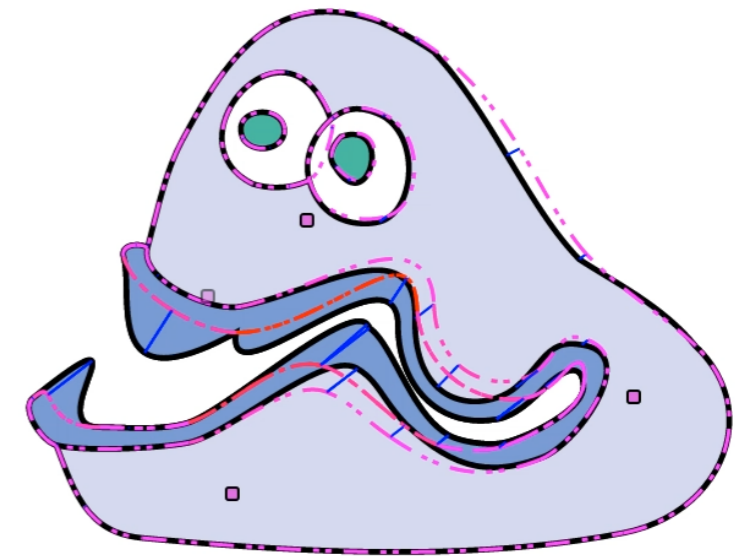
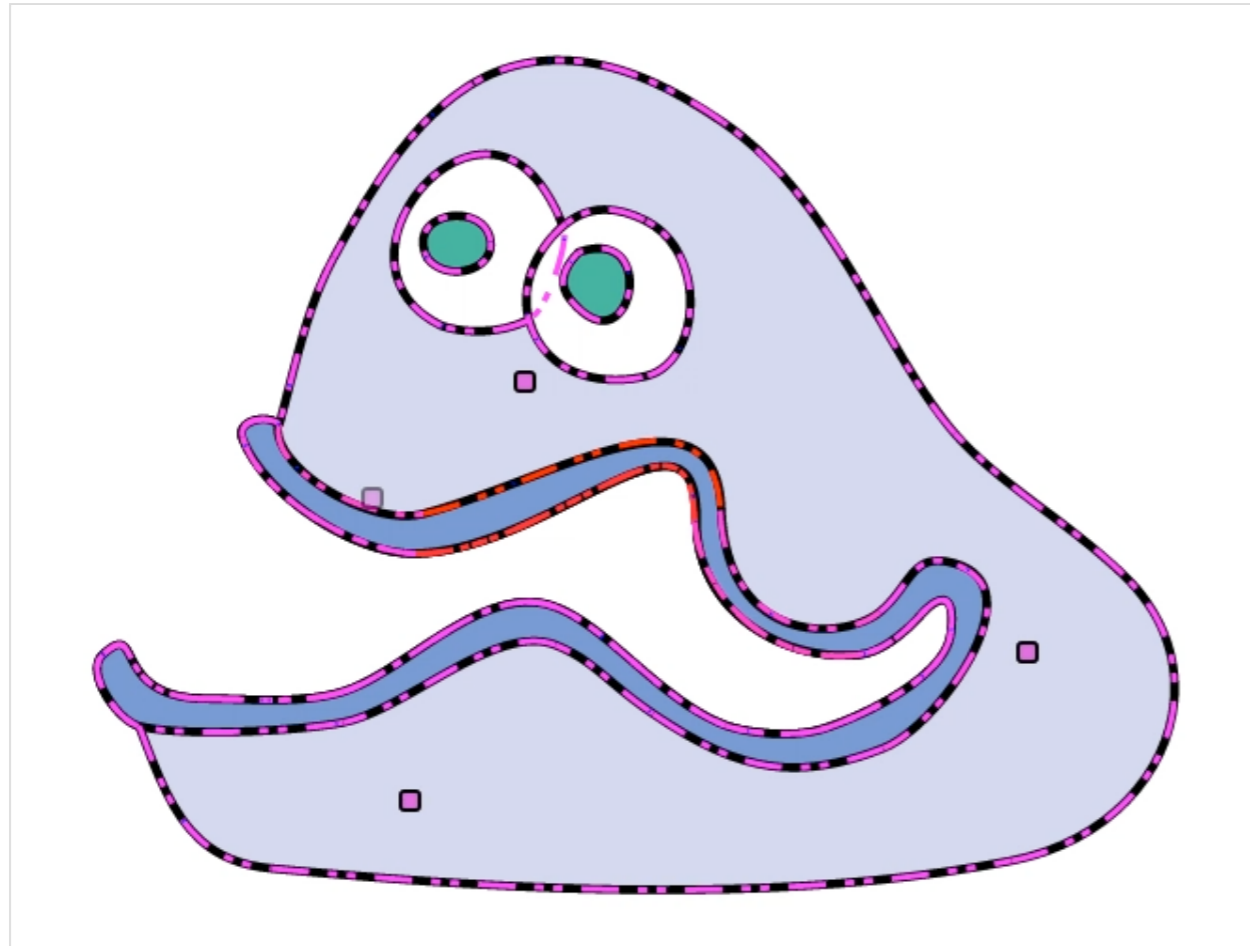




# Our Approach

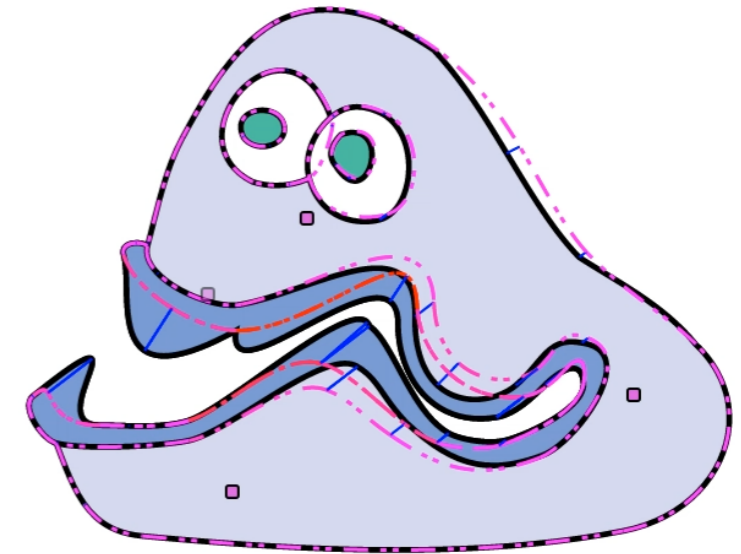
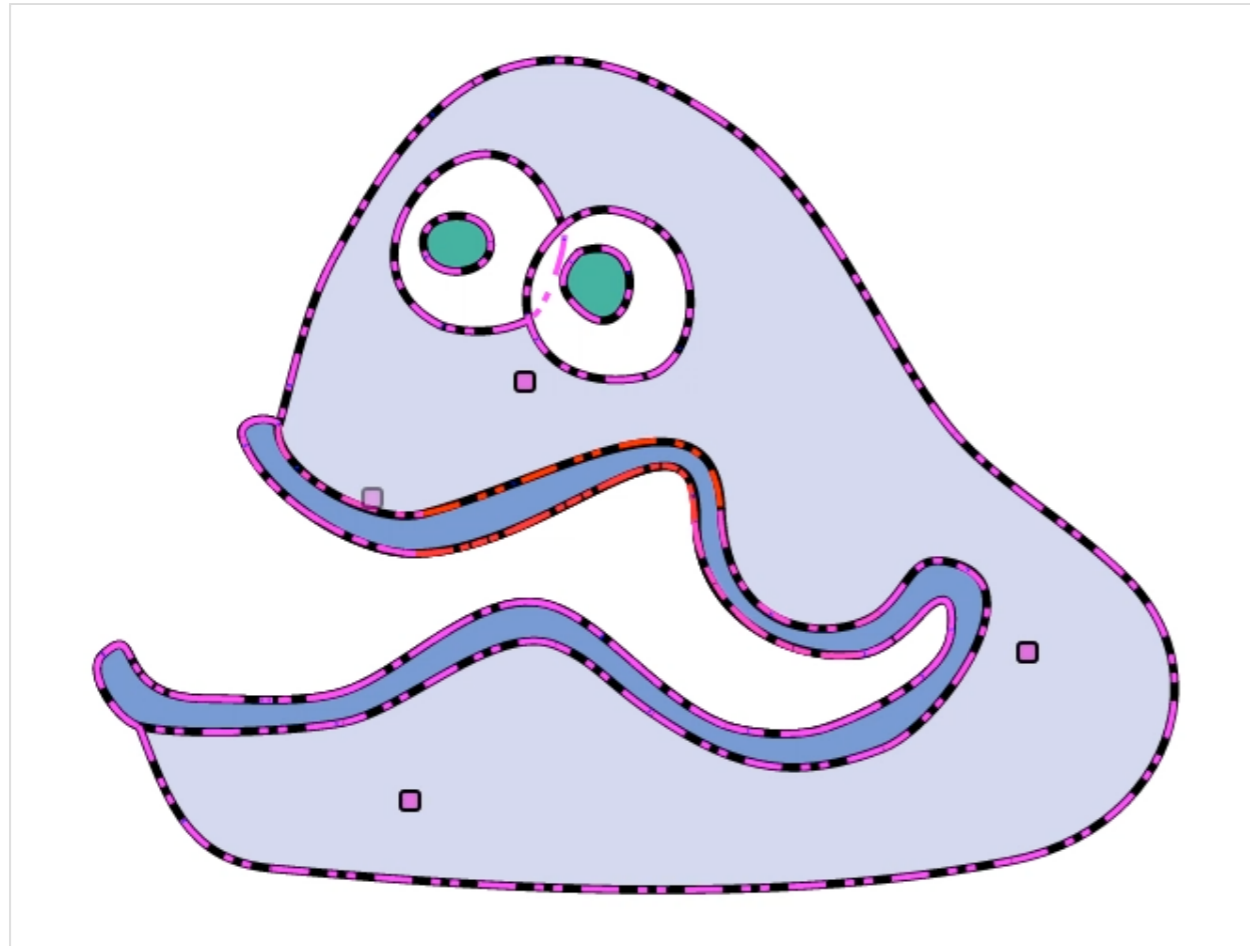


# Our Approach

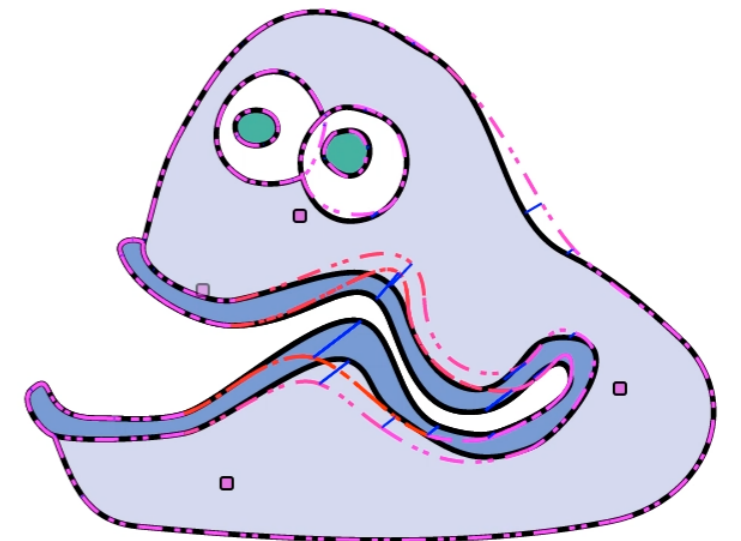


Deforming All  
Control Points

# Our Approach



Deforming All  
Control Points



Deforming Joint  
Control Points

LAPPHICS 2009 / P. Dutré and M. Stamminger  
Volume 28 (2009), Number 2

## Complex Barycentric Coordinates with Applications to Planar Shape Deformation

Ofir Weber, Mirela Ben-Chen and Craig Gotsman  
Technion - Israel Institute of Technology

To appear in the ACM SIGGRAPH conference proceedings

## Automatic Rigging and Animation of 3D Characters

Ilya Baran\* Jovan Popović<sup>†</sup>  
Computer Science and Artificial Intelligence Laboratory  
Massachusetts Institute of Technology

**Abstract**  
Animating an articulated 3D character currently requires manual rigging to specify its internal skeletal structure and to define how the input motion deforms its surface. We present a method for animating characters automatically. Given a static character mesh and a generic skeleton, our method adapts the skeleton to the character and attaches it to the surface, allowing skeletal motion data to animate the character. Because a single skeleton can be used with a wide range of characters, our method, in conjunction with a library of motions for a few skeletons, enables a user-friendly animation system for novices and children. Our prototype implementation, called Pinocchio, typically takes under a minute to rig a character on a modern midrange PC.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

**Keywords:** Animation, Deformations, Geometric Modeling

## Image Deformation Using Moving Least Squares

Scott Schaefer\* Travis McPhail<sup>†</sup> Joe Warren<sup>‡</sup>  
Texas A&M University Rice University Rice University

## As-Rigid-As-Possible Shape Manipulation

Takeo Igarashi<sup>1,3</sup> Tomer Moscovich<sup>2</sup> John F. Hughes<sup>2</sup>  
<sup>1</sup>The University of Tokyo <sup>2</sup>Brown University <sup>3</sup>PRESTO, JST

This paper presents an interactive system that allows the user to manipulate a shape without using a skeleton or FFD. The user chooses several points inside the shape as handles and moves each handle to a desired position. The system then moves, rotates, and deforms the overall shape to match the given handle positions while minimizing distortion. By taking the interior of the shape into account, our approach can model its rigidity (i.e., internal resistance to deformation), making the result much closer to the behavior of real-world objects than in space-warp approaches as in [Barrett and Cheney 2002; Llamas et al. 2003].

## Controllable Conformal Maps for Shape Deformation and Interpolation

Ofir Weber Craig Gotsman  
Technion - Israel Institute of Technology  
weber@cs.technion.ac.il gotsman@cs.technion.ac.il

## Green Coordinates

Yaron Lipman David Levin Daniel Cohen-Or  
Tel-Aviv University

To appear in SIGGRAPH 2006.

## Inverse Kinematics for Reduced Deformable Models

Kevin G. Der Robert W. Sumner<sup>†</sup> Jovan Popović<sup>‡</sup>  
Computer Science and Artificial Intelligence Laboratory Massachusetts Institute of Technology <sup>†</sup>ETH Zürich

We use a two-step closed-form algorithm for finding the shape configuration that minimizes distortion. The typical approach is to use a physically based simulation or nonlinear optimizations [Sheffer and Kraevoy 2004], but these techniques are too slow for interactive manipulation. A key aspect of our approach is the design of a quadratic error metric so that the minimization problem is formulated as a set of simultaneous linear equations. Our system solves the simultaneous equations at the beginning, and can therefore quickly find a solution during interaction. Ideally we would like a single quadratic error function that handles all properties of a shape, but no such function exists (see Appendix A). We therefore split the problem into a rotation part and a scale part. This divides the problem into two least-squares minimization problems that we can solve sequentially. This method can be seen as a variant of the method proposed by Sorkine et al. [2004].

Our technique can be useful in standard dragging operations with a mouse, but it is particularly interesting when using a multiple-point input device such as a SmartSkin touchpad [Rekimoto 2002] (Figure 1). With such a device, one can interactively move, rotate, and deform an entire shape as if manipulating a real object using both hands. This is difficult with existing shape deformation tools because most allow only local modification while the overall position and orientation of the shape remain fixed.

**Abstract**  
Conformal maps are considered very desirable for planar deformation applications, since they allow only local rotations and

**Abstract**  
Our method is extremely efficient: it requires only the solution of a small dense linear system at preprocessing time and a matrix-vector multiplication during runtime (which can be implemented on modern GPUs), thus the deformations, even on extremely

**Abstract**  
Articulated shapes are aptly described by reduced deformable models that express required shape deformations using a compact set of control parameters. Although sufficient to describe most shape deformations, these control parameters can be ill-suited for animation tasks, particularly when reduced deformable models are inferred automatically from example shapes. Our algorithm provides intuitive and direct control of reduced deformable models similar to a conventional inverse-kinematics algorithm for jointed rigid structures. We present a fully automated pipeline that transforms a set of unarticulated example shapes into a controllable, articulated model. With only a few manipulations, an animator can automatically and interactively pose detailed shapes at rates independent of their geometric complexity.

**Abstract**  
Animators often build a reduced deformable model that represents meaningful deformations by instrumenting a static mesh with control parameters that modify posture, bulge muscles, change facial expressions, and generate other necessary deformations. These controls provide a compact representation of the mesh deformation and allow the animator to generate movement efficiently. However, many animation tasks are more easily accomplished through direct manipulation. In particular, reaching for or interacting with surrounding objects is most effectively expressed through direct control of control vertices.

**Figure 1:** Conformal deformation of a giraffe with sharp bends at neck and legs. Original model (left) and three deformed versions.

**Figure 1:** (a-c) Cage-based 2D deformation of a Gecko. (b) Using Green Coordinates induces a pure conformal mapping. Harmonic Coordinates. Note the preservation of shape in the marked square. (d-f) Cage-based 3D articulation of an Ogre. Coordinates in 3D admits a quasi-conformal deformation. In (f) the result using Mean Value Coordinates is presented.

**Figure 1:** Our method uses example shapes to build a reduced deformable model, visualized in (A) by coloring portions of the mesh that move in a coordinated fashion. (B) A small number of proxy vertices are found that summarize the movement of the example meshes independent of their geometric complexity, providing a resolution-independent metric for mesh posing. The user can pose even highly detailed meshes interactively with just a few vertex constraints.

**Figure 1:** Shape manipulation using a SmartSkin touchpad. The user can interactively move, rotate, and deform the shape using both hands as if manipulating a real object.

## Bounded Biharmonic Weights for Real-Time Deformation

Alec Jacobson<sup>1</sup> Ilya Baran<sup>2</sup> Jovan Popović<sup>3</sup> Olga Sorkine<sup>1,4</sup>  
<sup>1</sup>New York University <sup>2</sup>Disney Research, Zurich <sup>3</sup>Adobe Systems, Inc. <sup>4</sup>ETH Zürich

## Harmonic Coordinates for Character Articulation

Pushkar Joshi Mark Meyer Tony DeRose Brian Green Tom Sanocki  
Pixar Animation Studios

**CR Categories:** I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

**Keywords:** Animation with Constraints, Deformations

**Contact:** {kevinder|jovan}@csail.mit.edu  
<sup>†</sup>sumnerb@inf.ethz.ch

## 1 Introduction

Efficient and intuitive manipulation of detailed triangle meshes is challenging because they have thousands of degrees of freedom. Modeling objects must cope with this geometric complexity to provide effective tools for sculpting broad changes as well as fine-scale details. However, in animation, the complexity of a character's movement is far less than its geometric complexity since vertices move in a coordinated fashion. An articulated character

Reduced deformable models can also be inferred automatically from a set of example deformations. Although this approach eases the laborious task of designing controls by hand, applications are limited because the inferred control parameters are often ill-suited for animation tasks. Our work hides these unintuitive parameters with a procedure for direct manipulation of reduced deformable models, allowing the animator to generate meaningful mesh deformations without learning the mapping between the controls and their effects.

Our method identifies control parameters of a reduced deformable model with a set of transformation matrices that control shape deformations. The animator can then select and move any subset of mesh vertices to pose the entire shape (Figure 1). In general, direct manipulation is an ill-posed problem whether using skeletons or inferred models, since many pose configurations can satisfy a given set of user constraints. We use a nonlinear optimization to find the pose whose transformations best meet the animator's constraints with a resolution-independent objective function, while favoring poses close to the space of examples. Lastly, a linear reconstruction computes the new deformed vertex positions. This final step is linear in the number of vertices but is computationally negligible when implemented efficiently in hardware. In total, the

**Abstract**  
Object deformation with linear blending dominates practical use as the fastest approach for transforming raster images, vector graphics, geometric models and animated characters. Unfortunately, linear blending schemes for skeletons or cages are not always easy to use because they may require manual weight painting or modeling closed polyhedral envelopes around objects. Our goal is to make the design and control of deformations simpler by allowing the user to work freely with the most convenient combination of handle types. We develop linear blending weights that produce smooth and intuitive deformations for points, bones and cages of arbitrary topology. Our weights, called bounded biharmonic weights, minimize the Laplacian energy subject to bound constraints. Doing so spreads the influences of the controls in a shape-aware and localized manner, even for objects with complex and concave boundaries. The variational weight optimization also makes it possible to customize the weights so that they preserve the shape of specified essential object features. We demonstrate successful use of our blending weights for real-time deformation of 2D and 3D shapes.

**Abstract**  
In this paper we consider the problem of creating and controlling volume deformations used to articulate characters for use in high-end applications such as computer generated feature films. We introduce a method we call harmonic coordinates that significantly improves upon existing volume deformation techniques. Our deformations are controlled using a topologically flexible structure, called a cage, that consists of a closed three dimensional mesh. The cage can optionally be augmented with additional interior vertices, edges, and faces to more precisely control the interior behavior of the deformation. We show that harmonic coordinates are generalized barycentric coordinates that can be extended to any dimension.

However free-form deformation has some drawbacks:

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

**Keywords:** shape deformation, articulated character animation, generalized barycentric coordinates, linear blend skinning

**Links:** DL PDF WEB VIDEO

**CR Categories:** I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

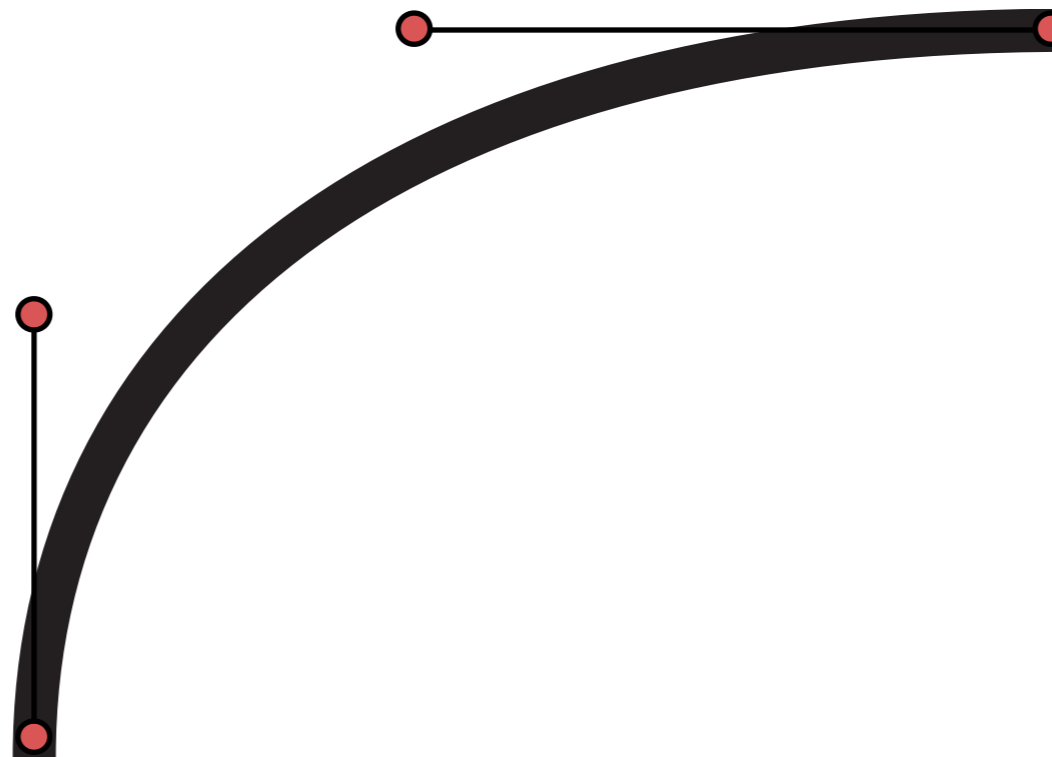
**Keywords:** Animation with Constraints, Deformations

**CR Categories:** I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

**Keywords:** Animation with Constraints, Deformations

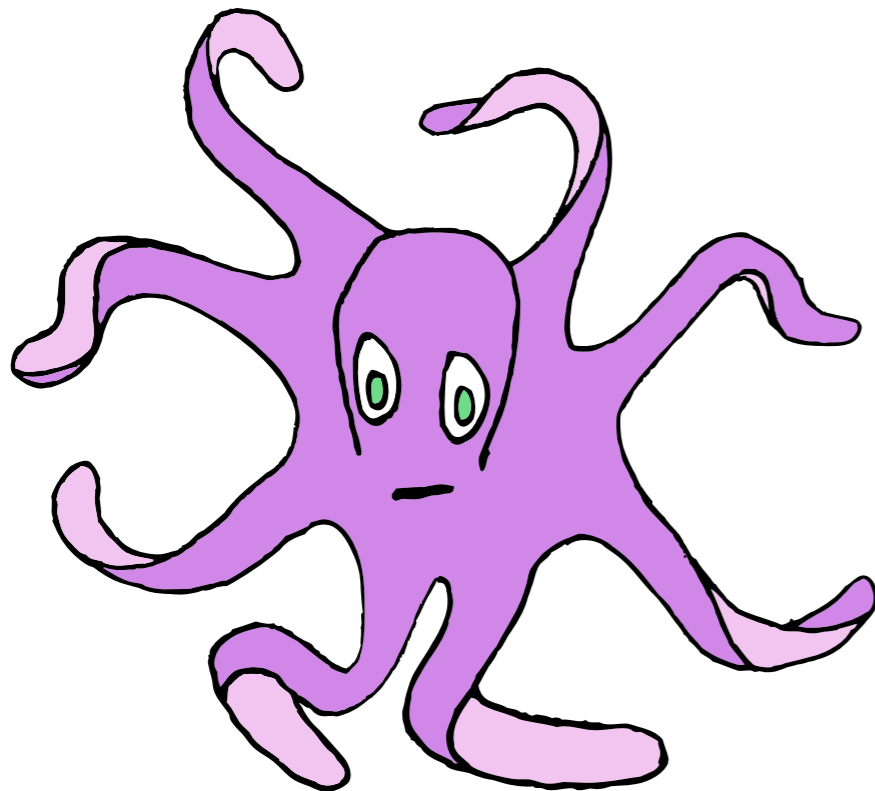
- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = \sum_{i=0}^n b_{i,n}(t) C_i, t \in [0, 1]$$



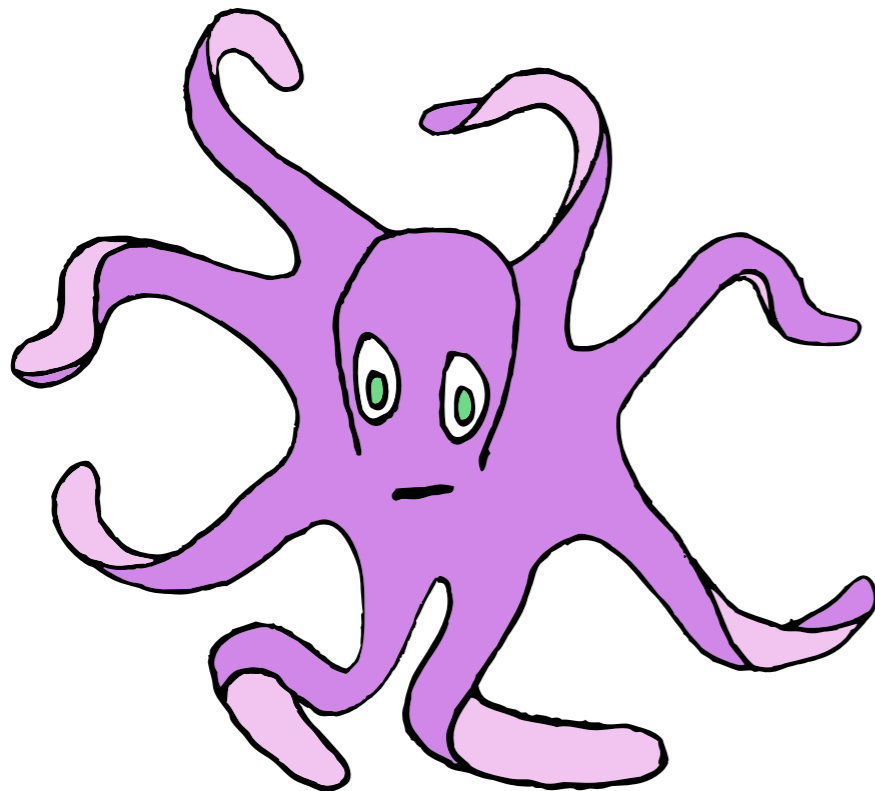
- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = \sum_{i=0}^n b_{i,n}(t) C_i, t \in [0, 1]$$



- Splines in 2D — Cubic Bézier splines.

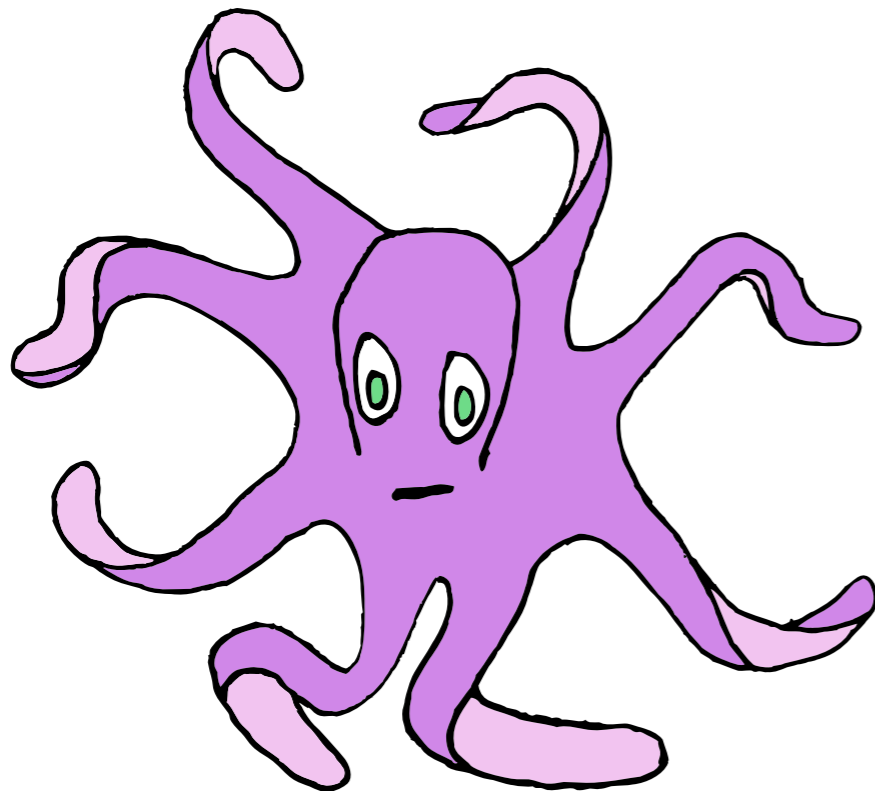
$$B_C(t) = \sum_{i=0}^n b_{i,n}(t) C_i, t \in [0, 1]$$



*Siggraph*

- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = \sum_{i=0}^n b_{i,n}(t)C_i, t \in [0, 1]$$

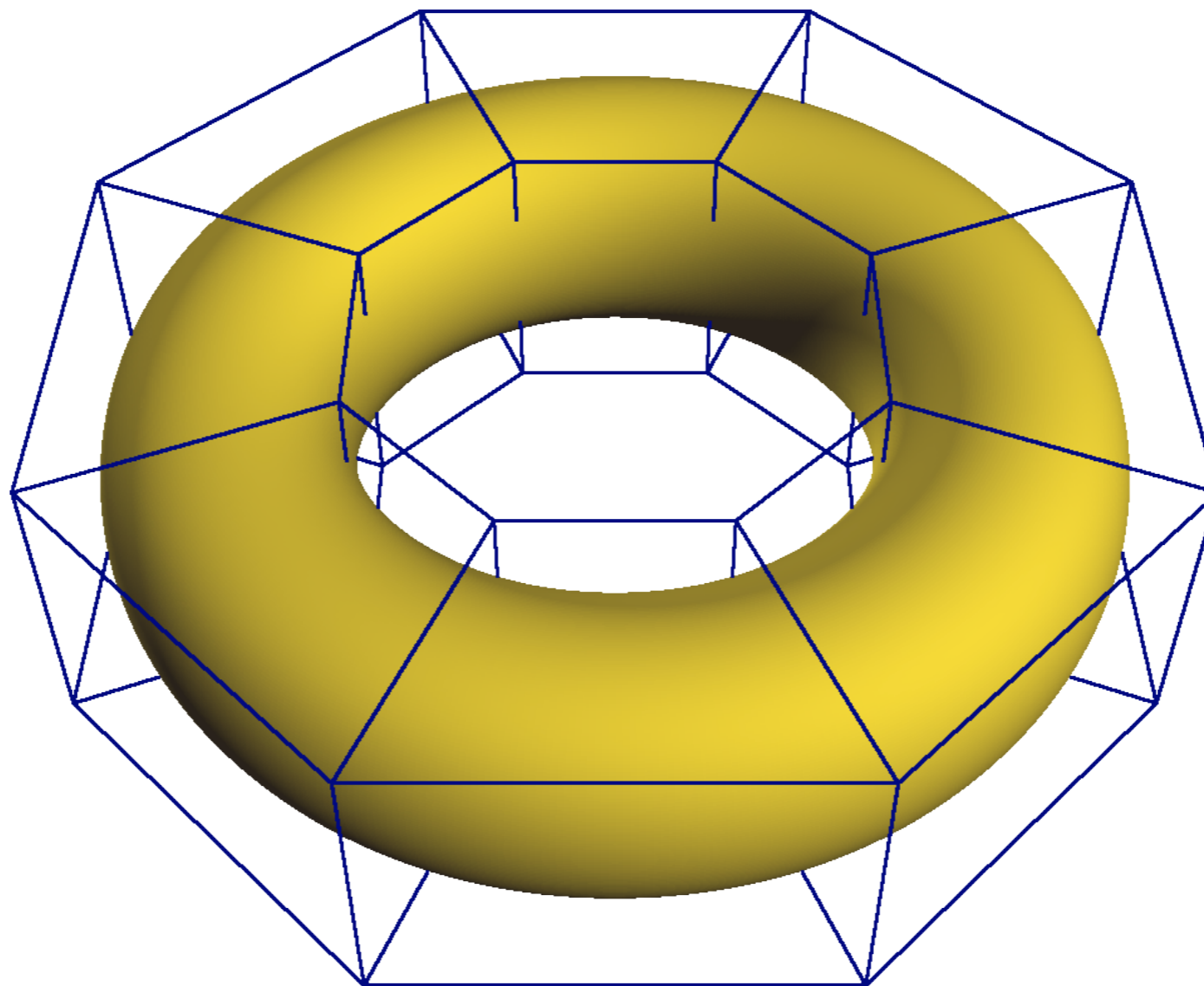


*Siggraph*

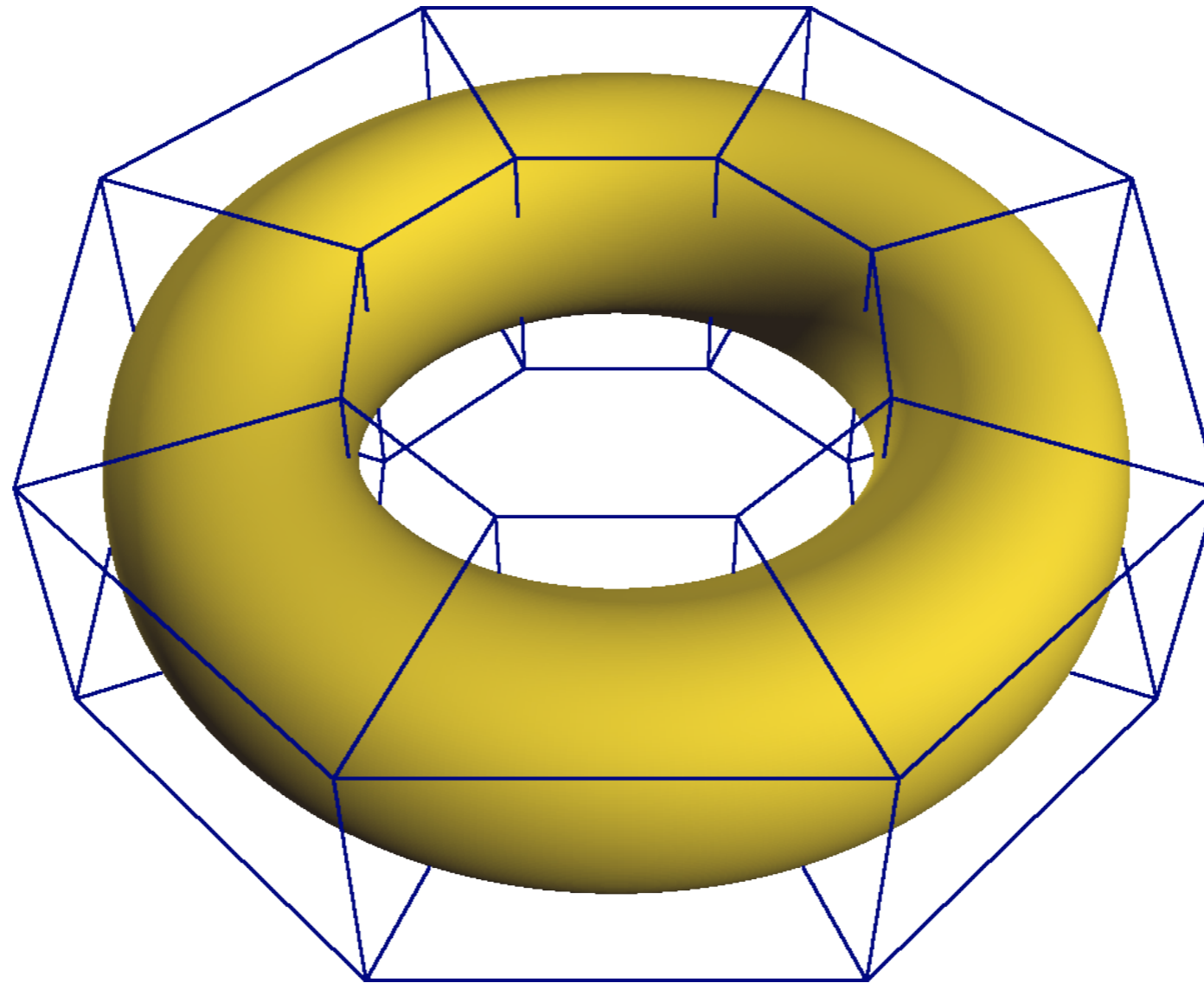




- Subdivision Surfaces in 3D — Catmull-Clark Subdivision Surfaces.

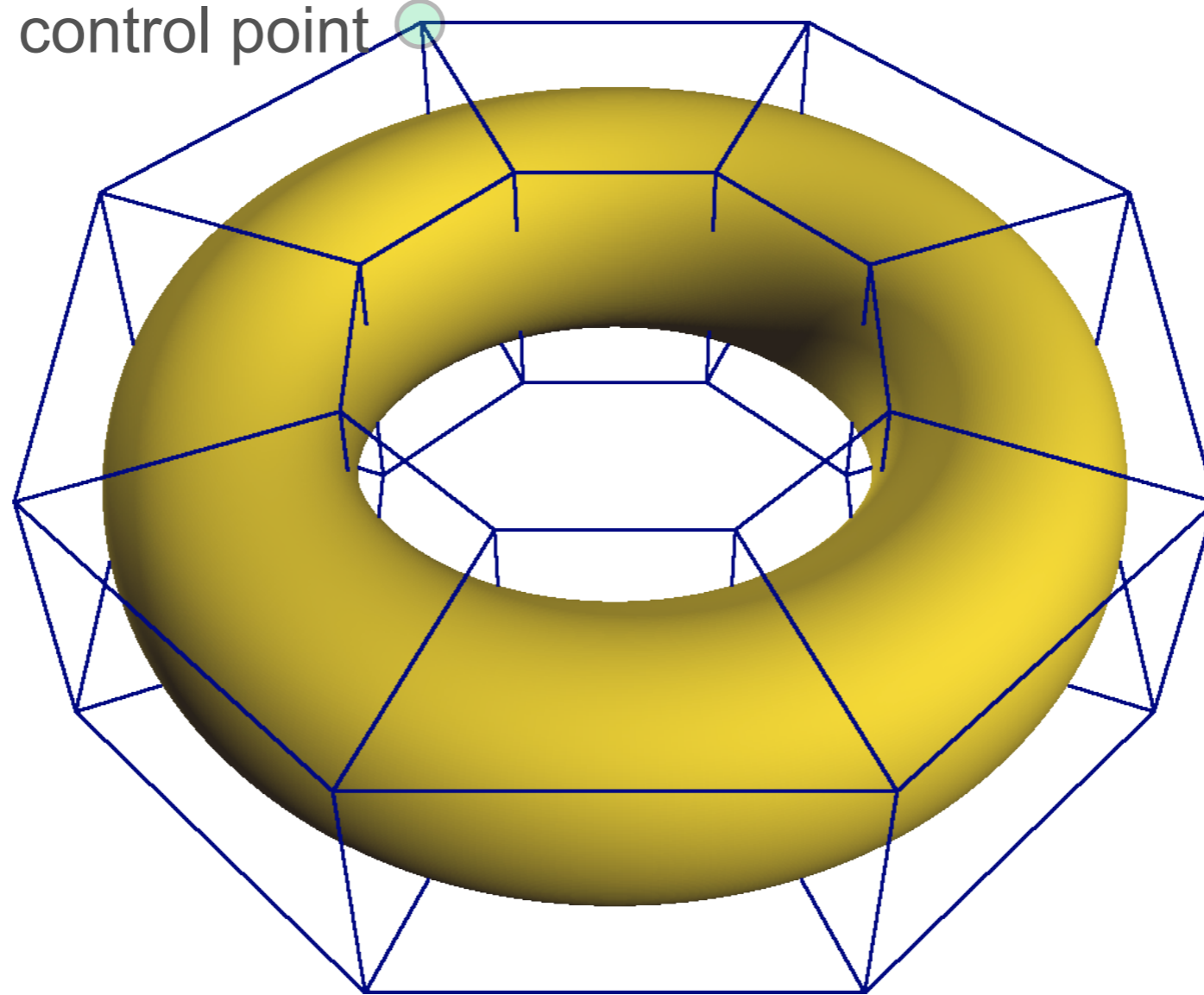


- Subdivision Surfaces in 3D — Catmull-Clark Subdivision Surfaces.



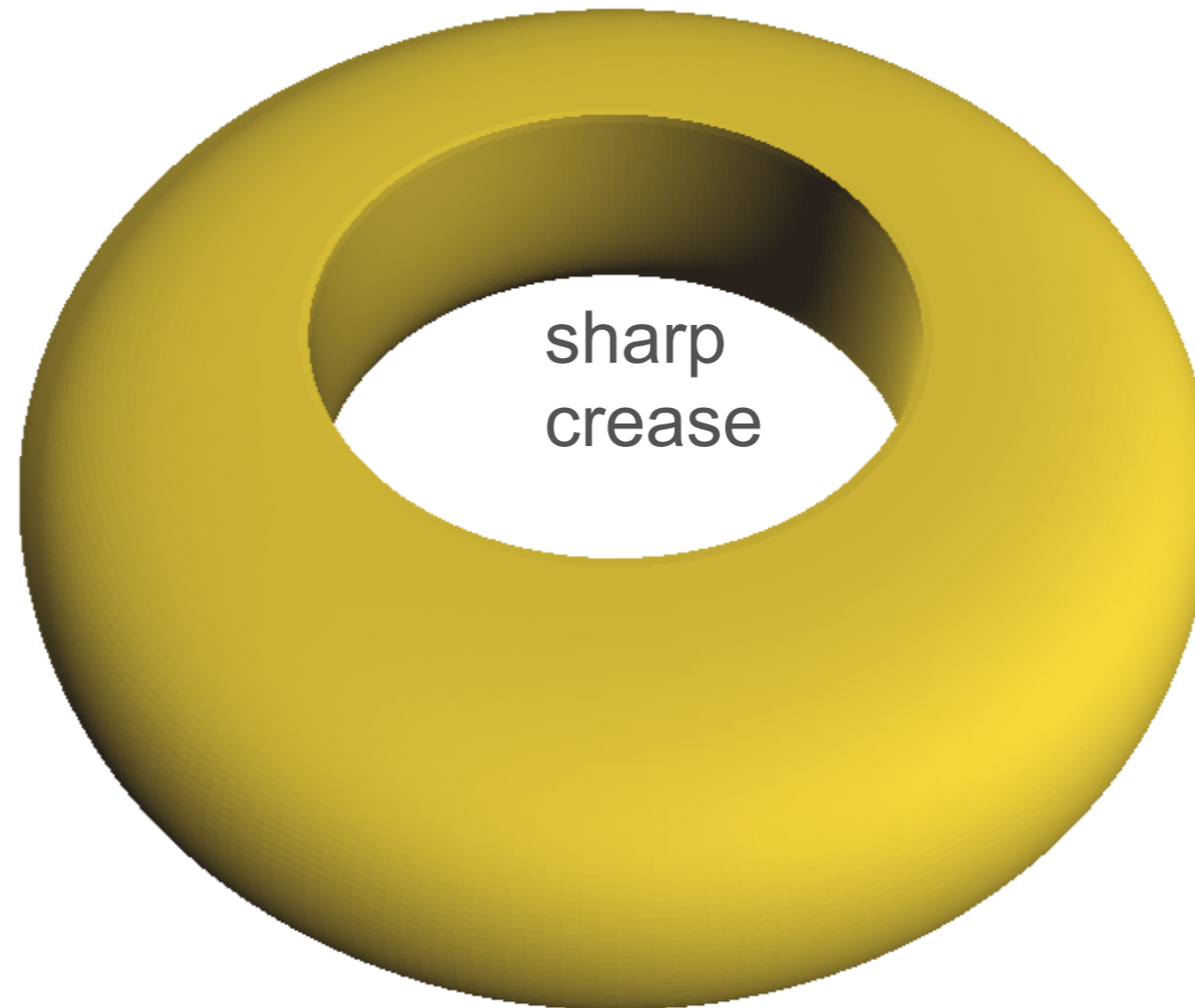
- Subdivision Surfaces in 3D — Catmull-Clark Subdivision Surfaces.

normal control point 



- Inherently C2 everywhere except extraordinary vertices (C1)

- Subdivision Surfaces in 3D — Catmull-Clark Subdivision Surfaces.

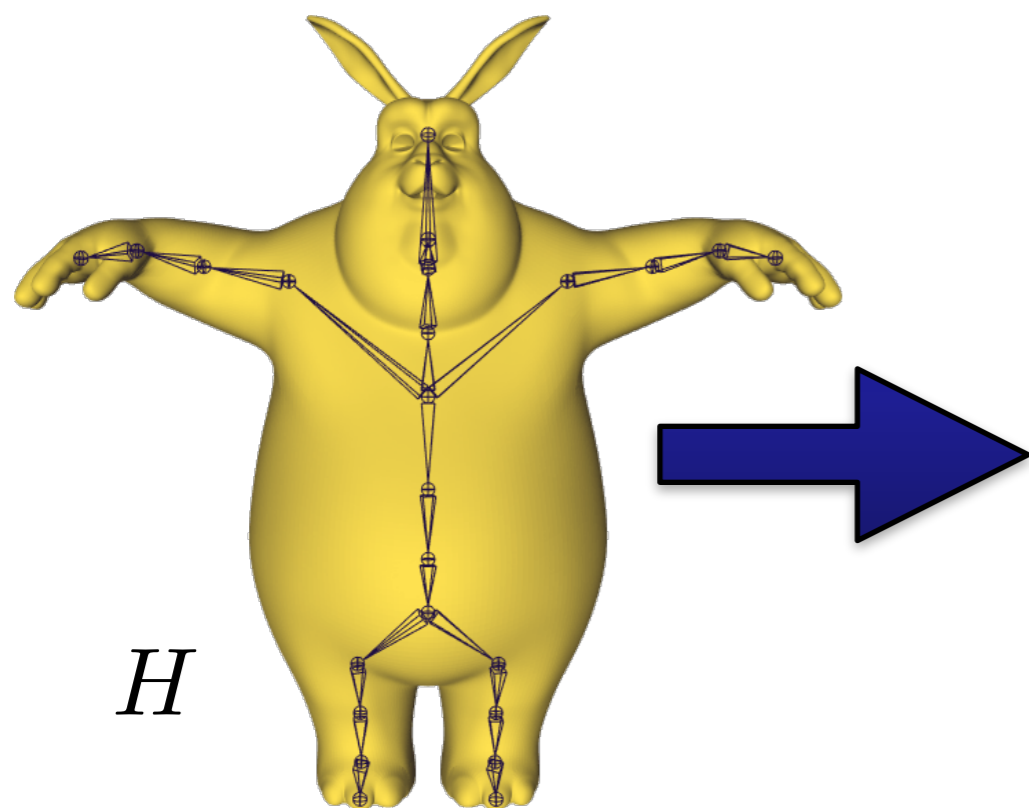


- Inherently C2 everywhere except extraordinary vertices (C1)
- Sharp creases can also be specified

# Linear Blend Skinning(LBS)

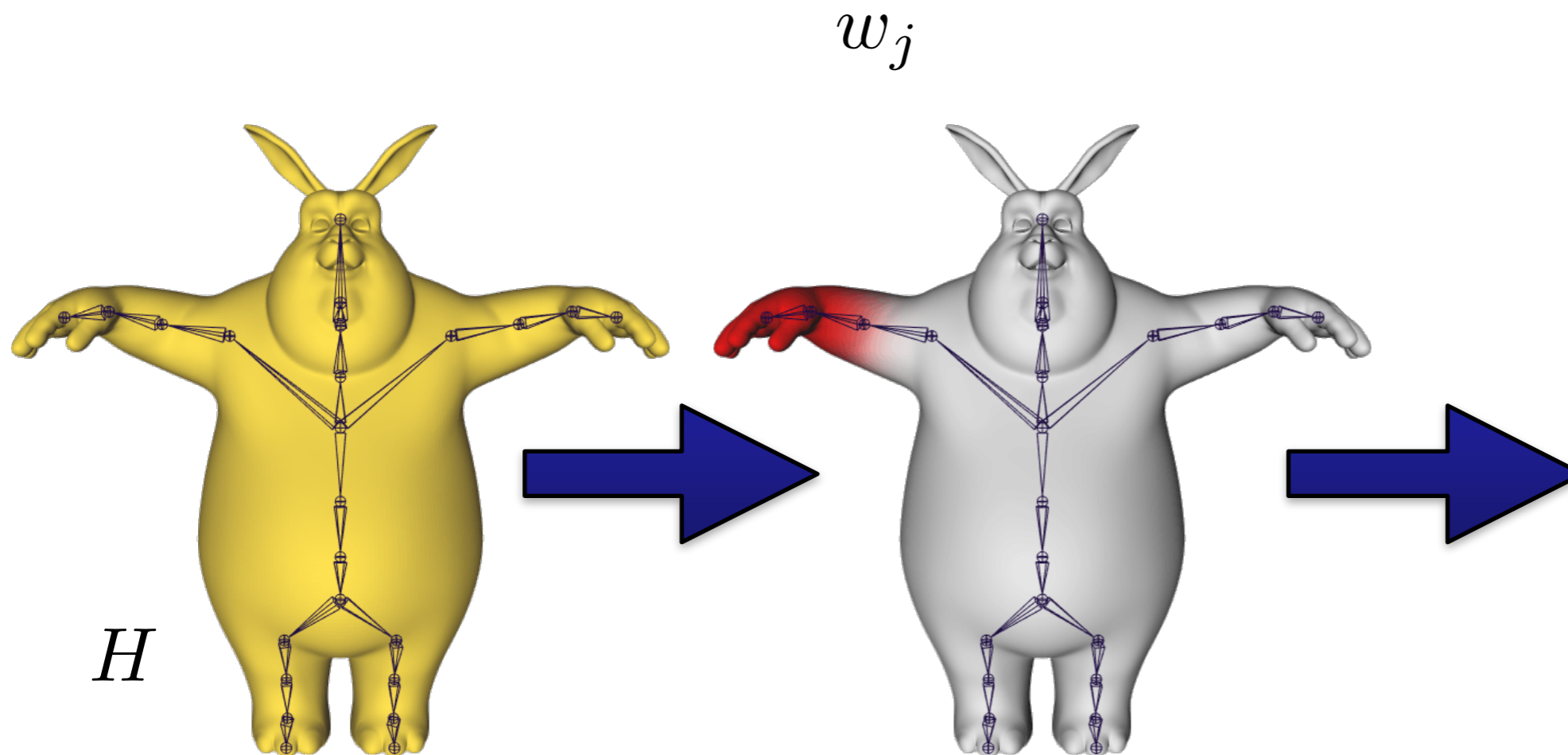
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

# Linear Blend Skinning(LBS)



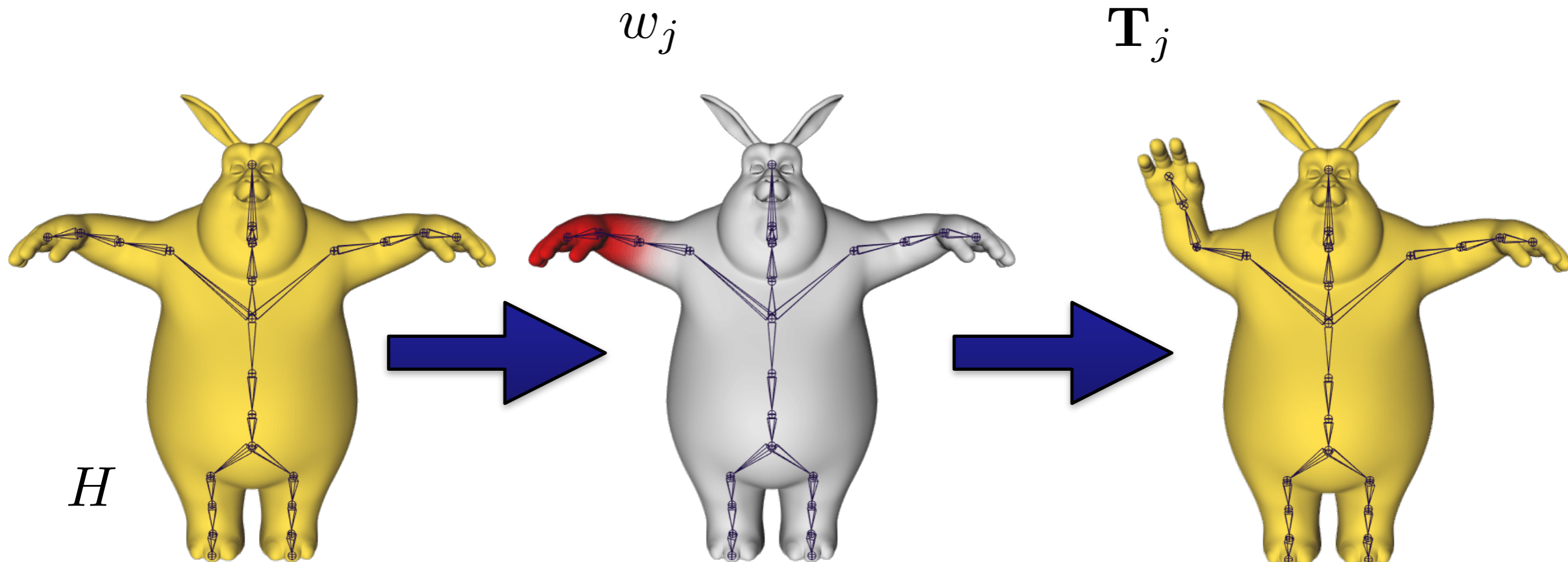
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

# Linear Blend Skinning(LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

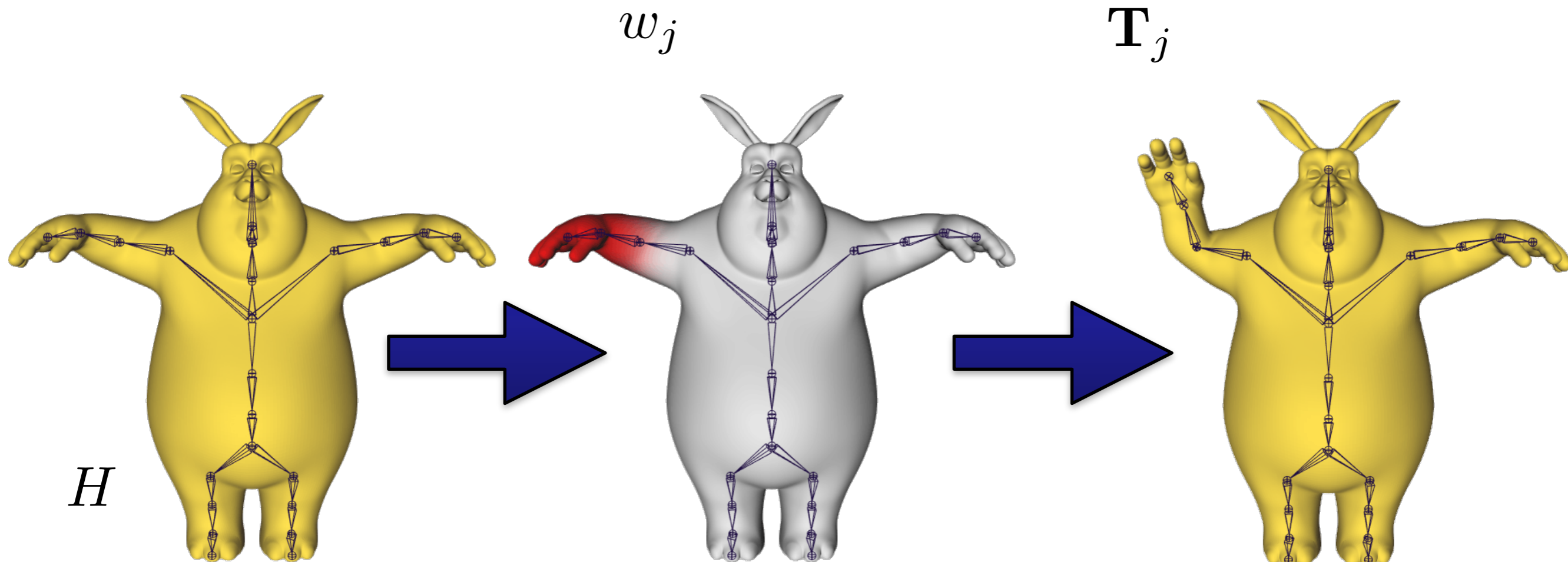
# Linear Blend Skinning(LBS)



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$



# Linear Blend Skinning(LBS)

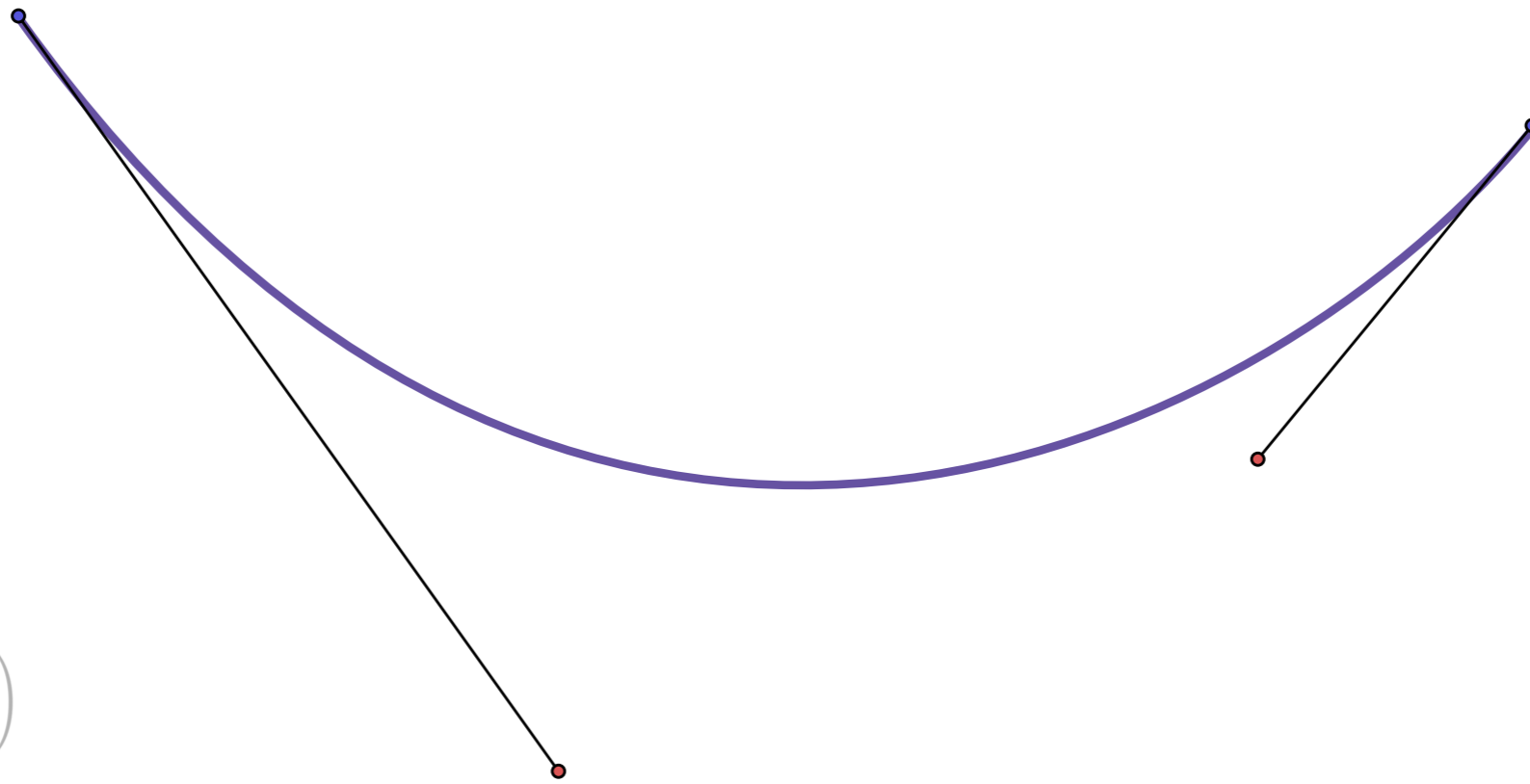


$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

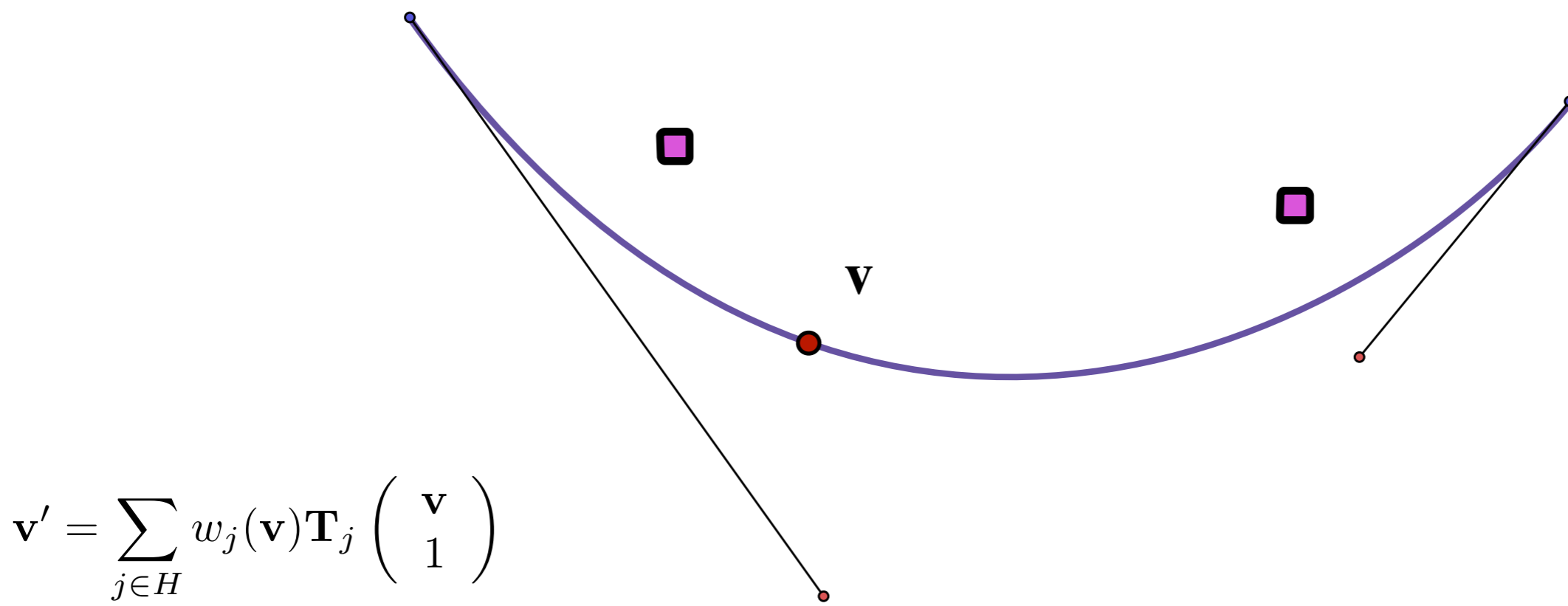
# Difficulty Skinning Cubic Bézier Curve

Apply skinning to  $v = B_C(t)$



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

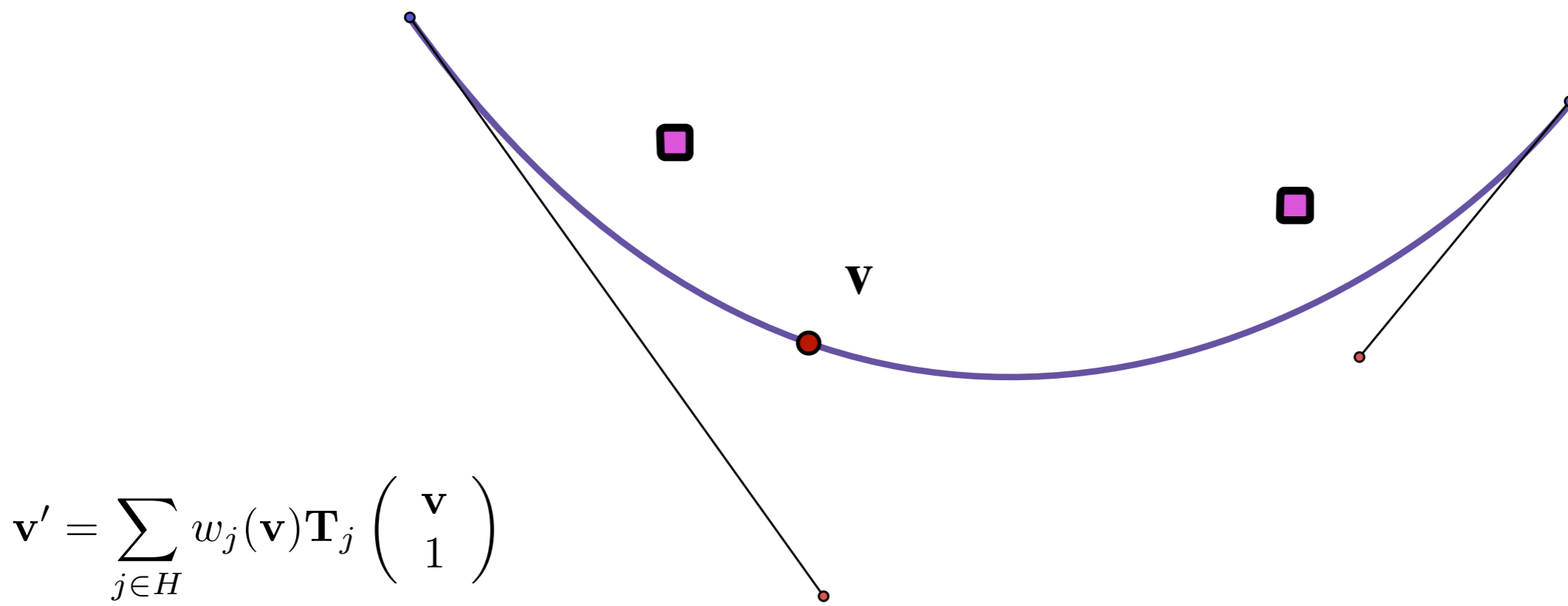
Apply skinning to  $v = B_C(t)$



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

# Difficulty in Deforming A Cubic Bézier Curve

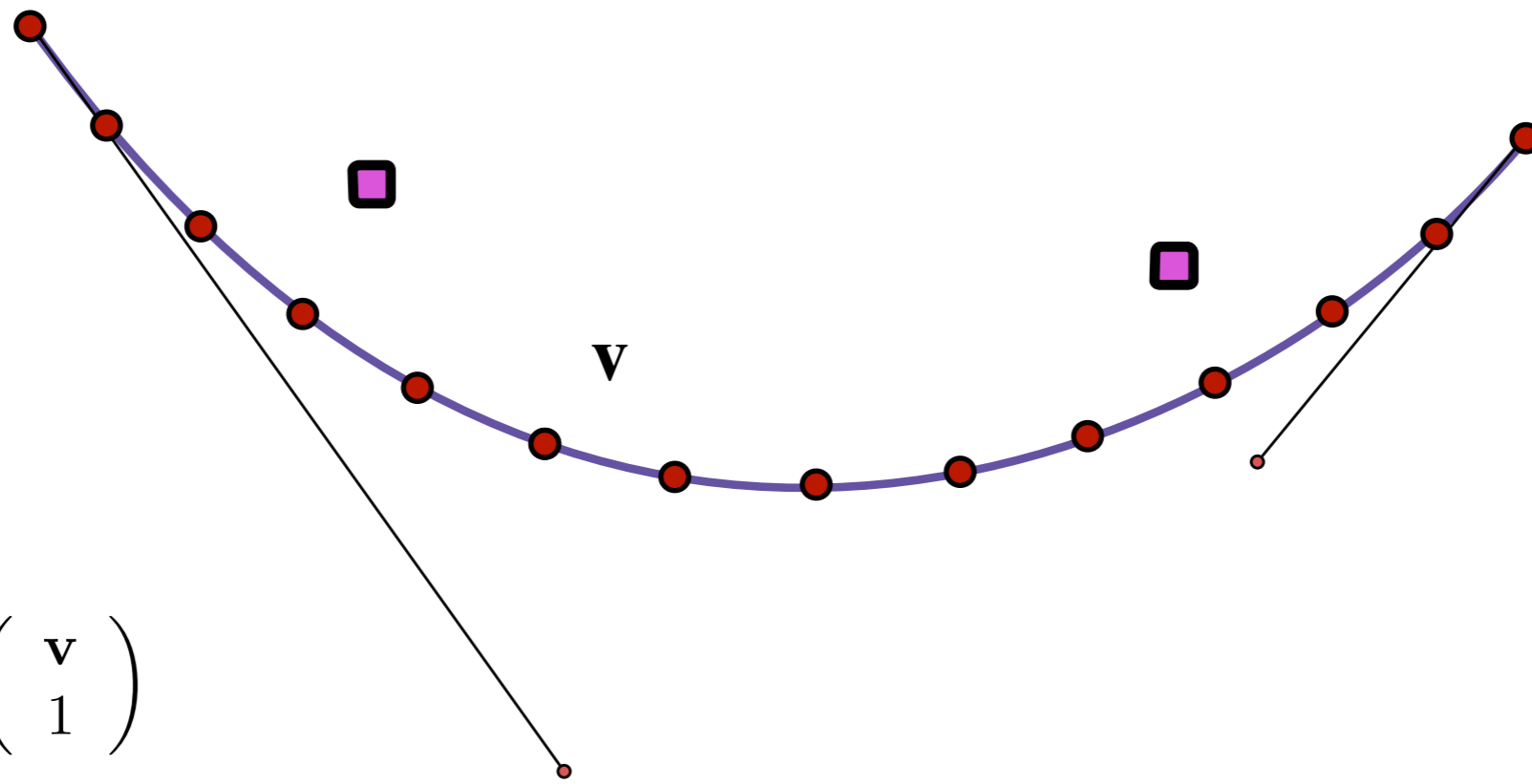
Apply skinning to  $v = B_C(t)$



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

# Difficulty in Deforming A Cubic Bézier Curve

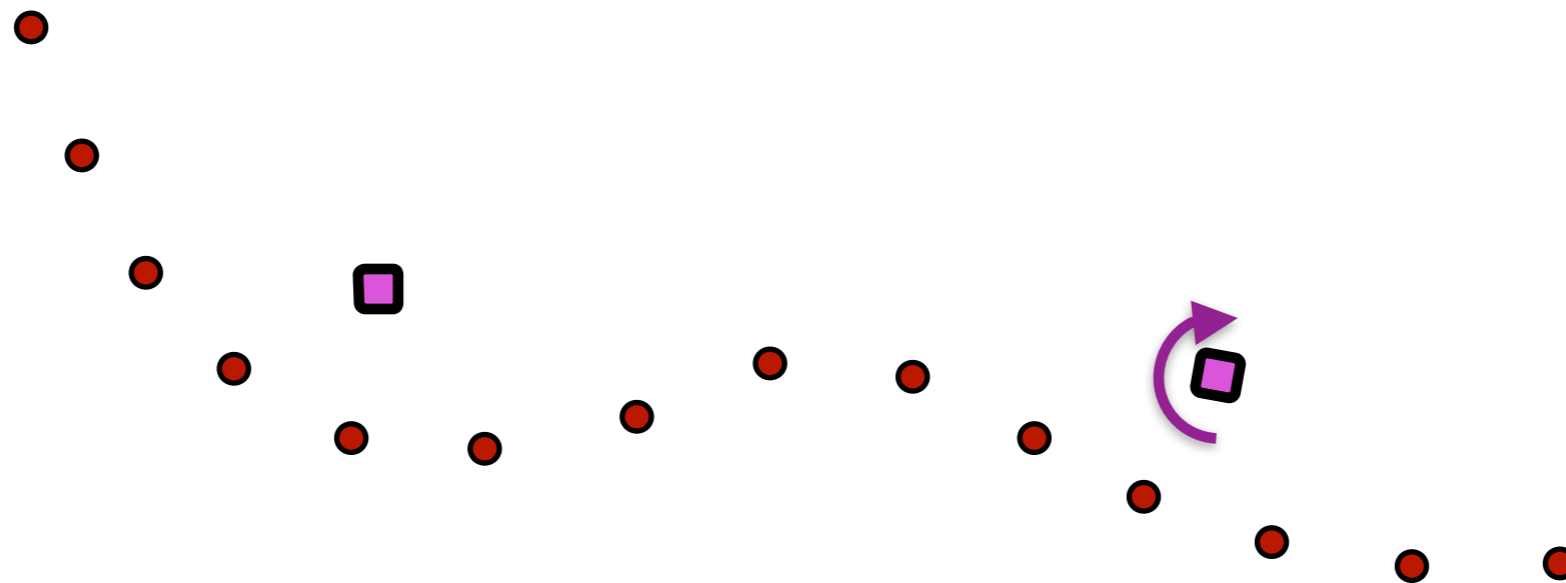
Apply skinning to  $v = B_C(t)$



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

# Difficulty in Deforming A Cubic Bézier Curve

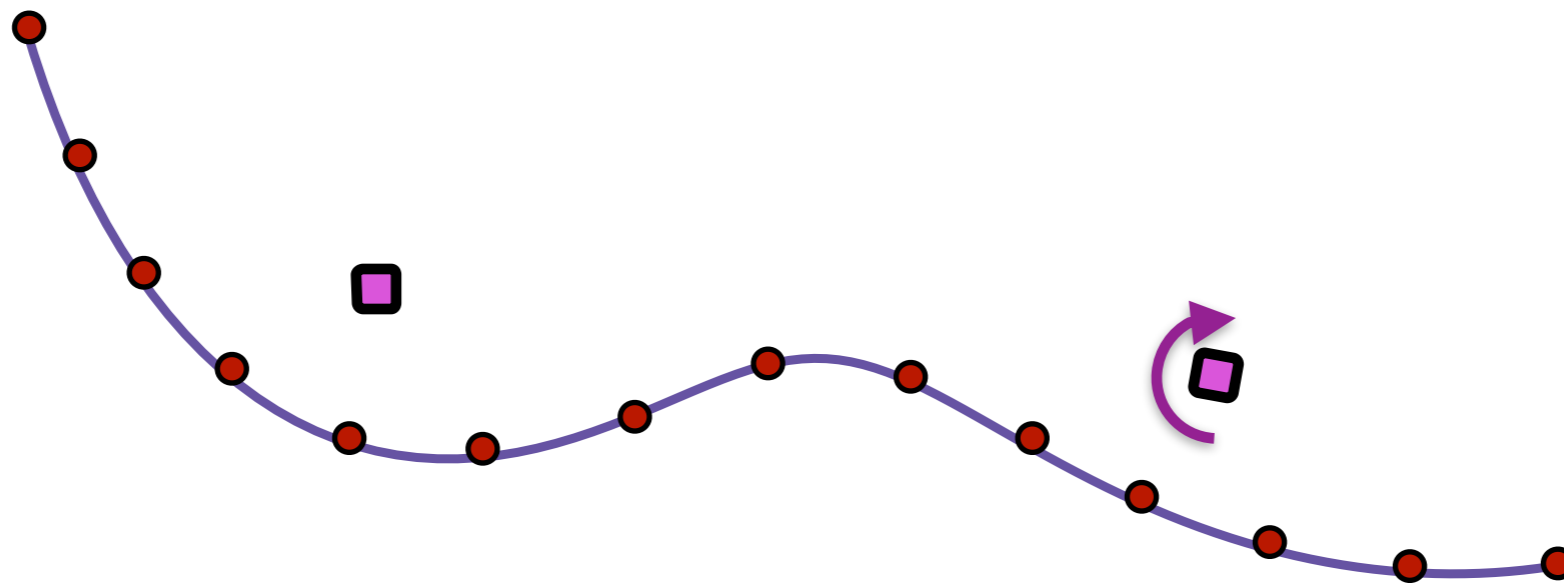
Apply skinning to  $v = B_C(t)$



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

# Difficulty in Deforming A Cubic Bézier Curve

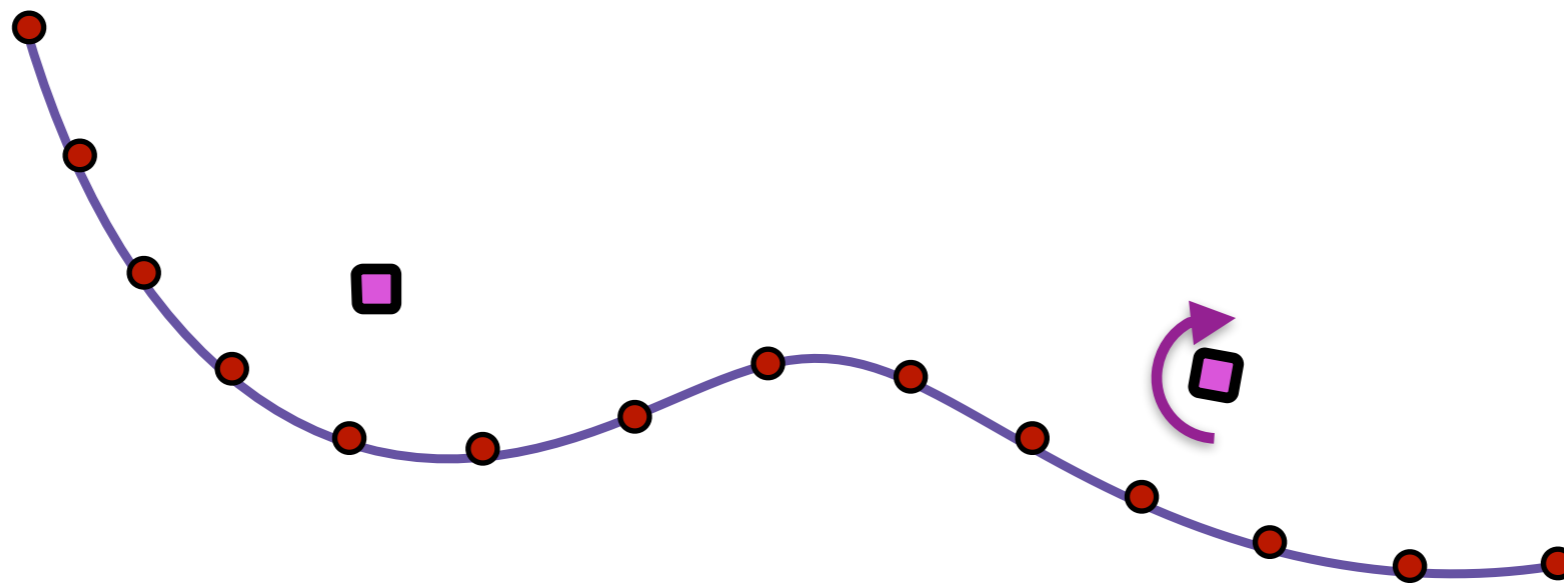
Apply skinning to  $v = B_C(t)$



$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$



Apply skinning to  $v = B_C(t)$

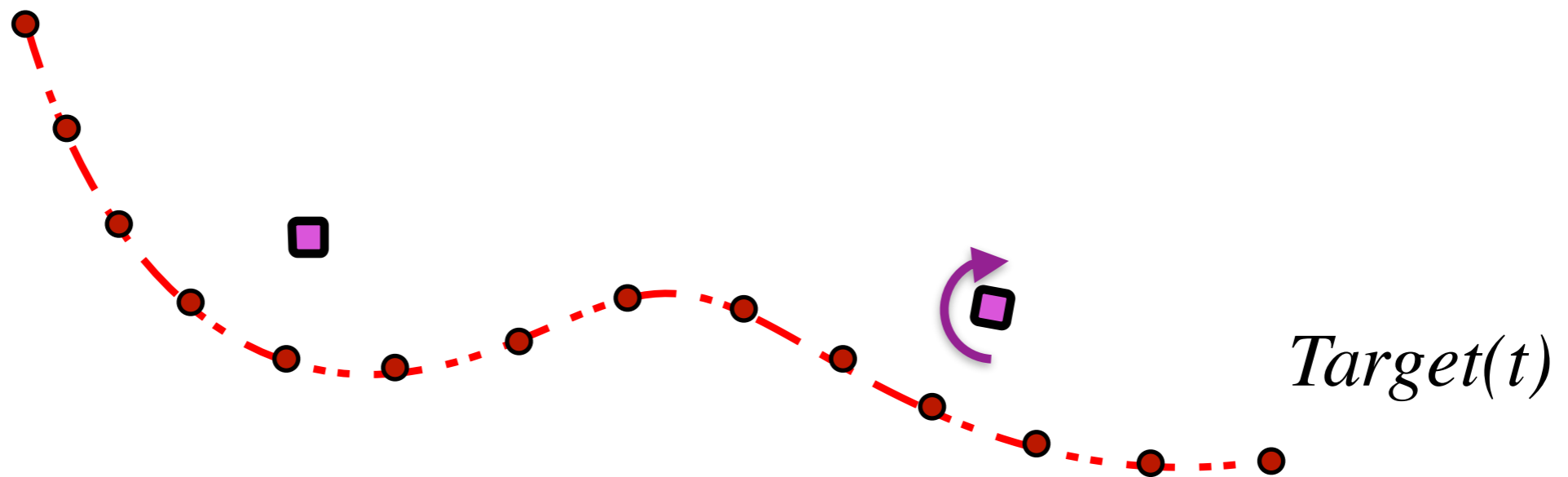


$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

No longer a cubic Bézier curve!

# Difficulty in Deforming A Cubic Bézier Curve

Apply skinning to  $v = B_C(t)$

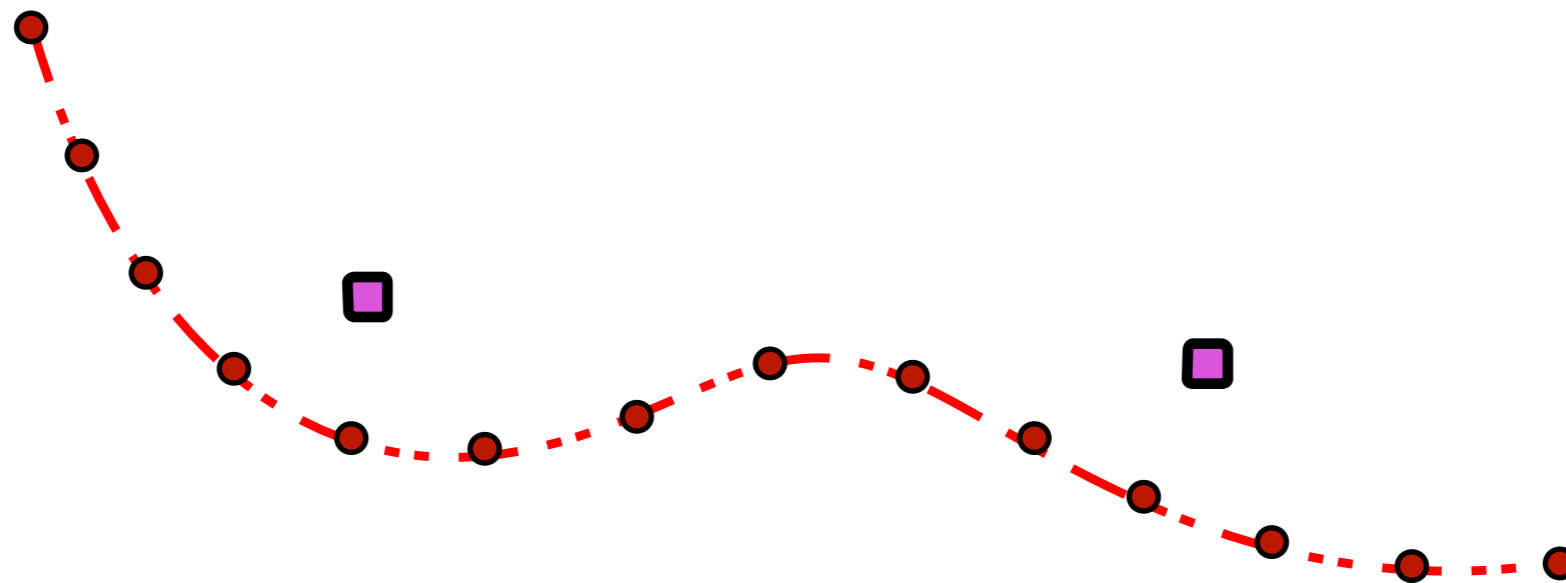


$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

No longer a cubic Bézier curve!

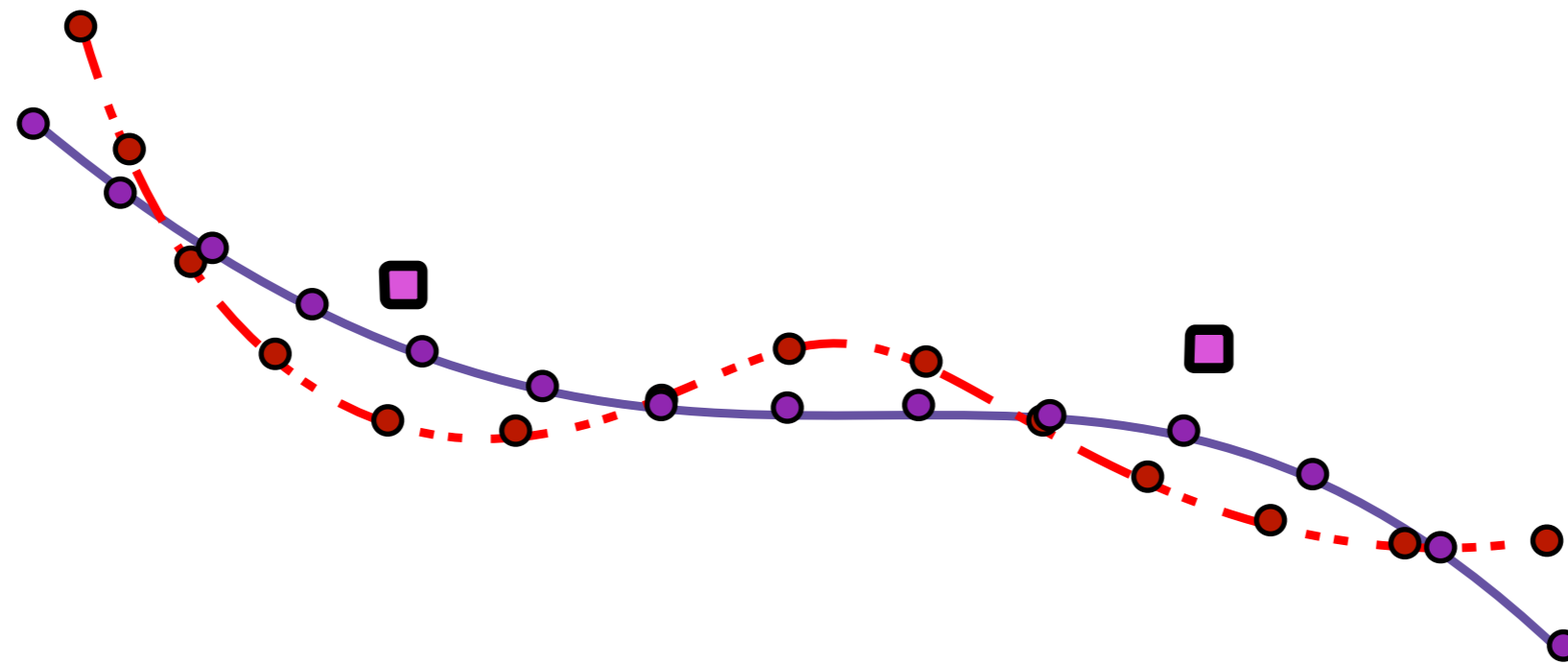
- Minimizing the L2 norm

*Target(t)*



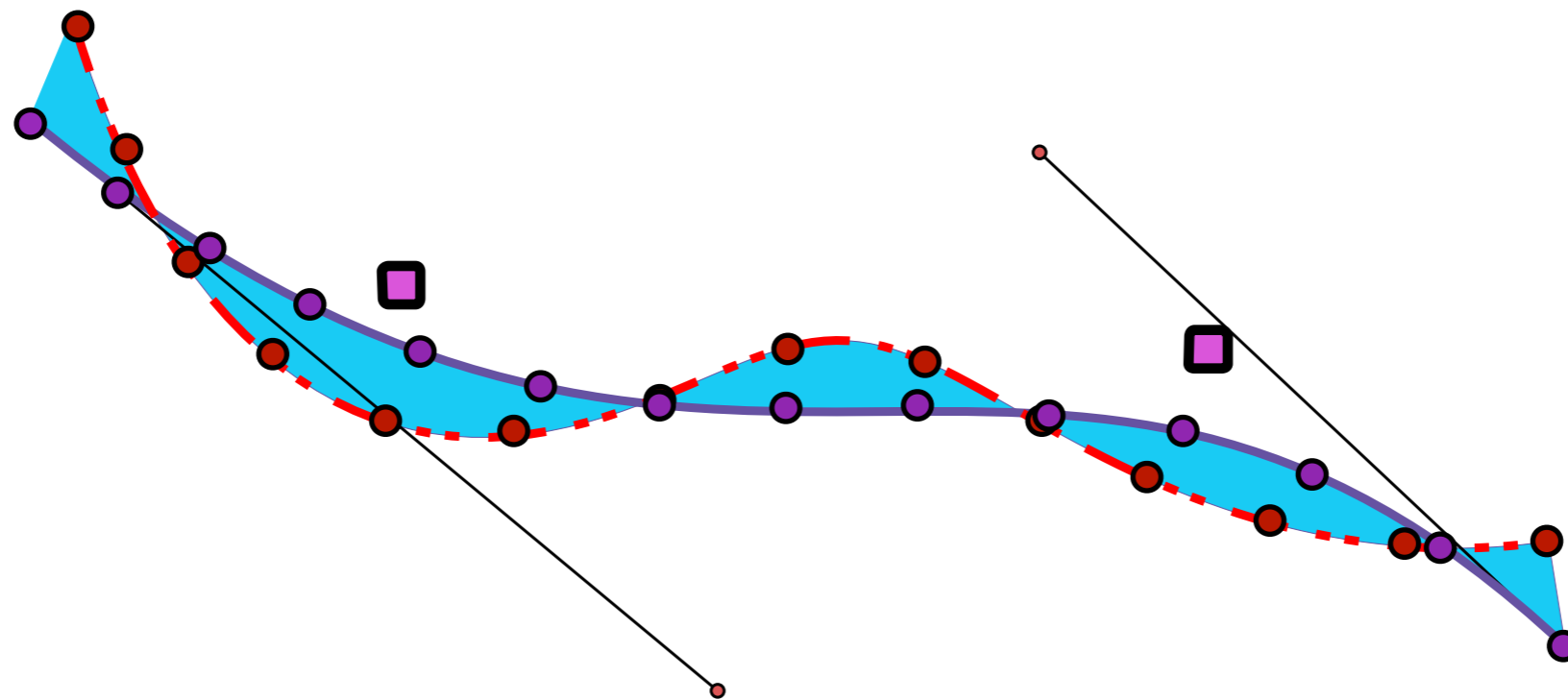
- Minimizing the L2 norm

$$B_{C'}(t) - Target(t)$$



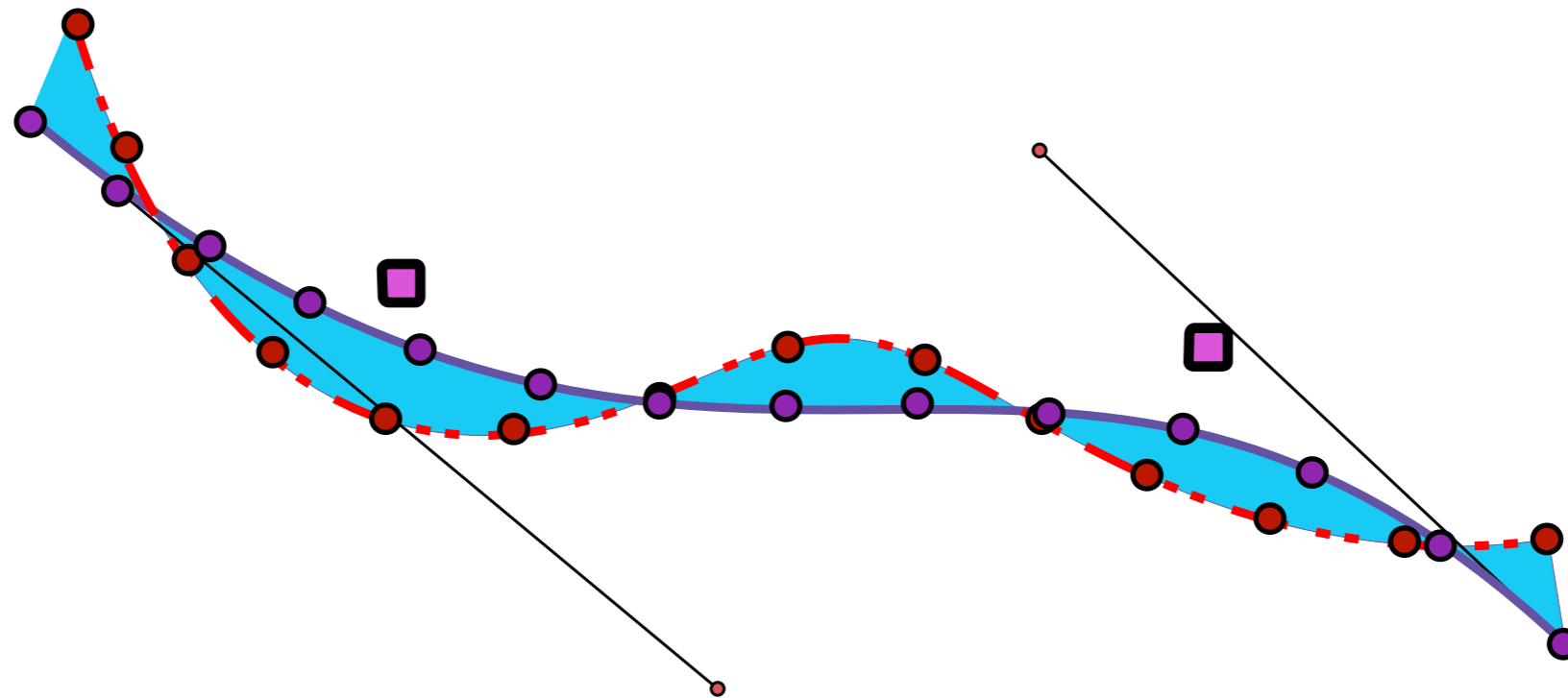
- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$



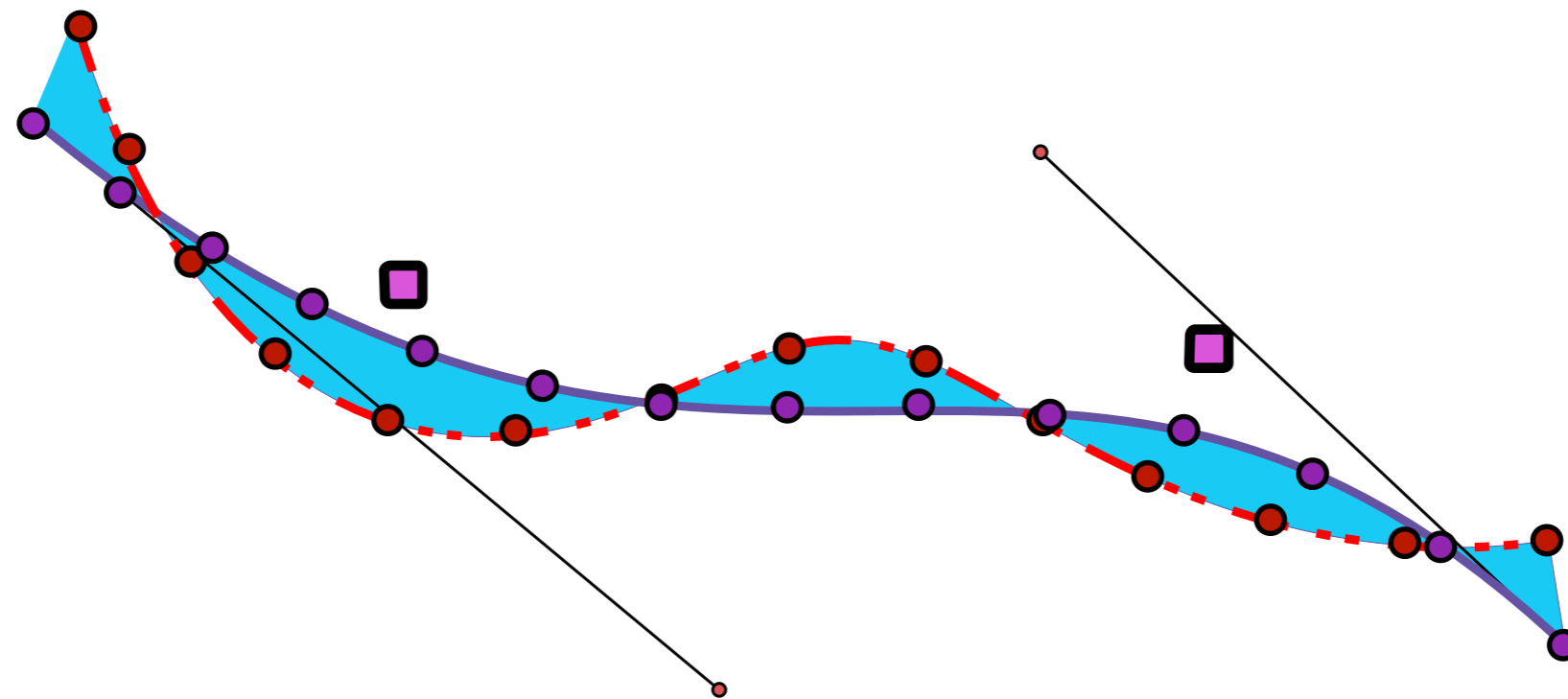
- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$



- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$



- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$



- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$

$$C' = \sum_{j \in H} T_j \hat{W}_j \hat{A}^{-1}$$

- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$

$$C' = \sum_{j \in H} T_j \hat{W}_j \hat{A}^{-1} \leftarrow \text{Pre-computed}$$

- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$

$$C' = \sum_{j \in H} T_j \hat{W}_j \hat{A}^{-1} \leftarrow \text{Pre-computed}$$

$$\text{LBS: } v' = \sum_{j \in H} w_j(v) T_j \begin{pmatrix} v \\ 1 \end{pmatrix}$$

- Minimizing the L2 norm

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$

$$C' = \sum_{j \in H} T_j \hat{W}_j \hat{A}^{-1} \leftarrow \text{Pre-computed}$$

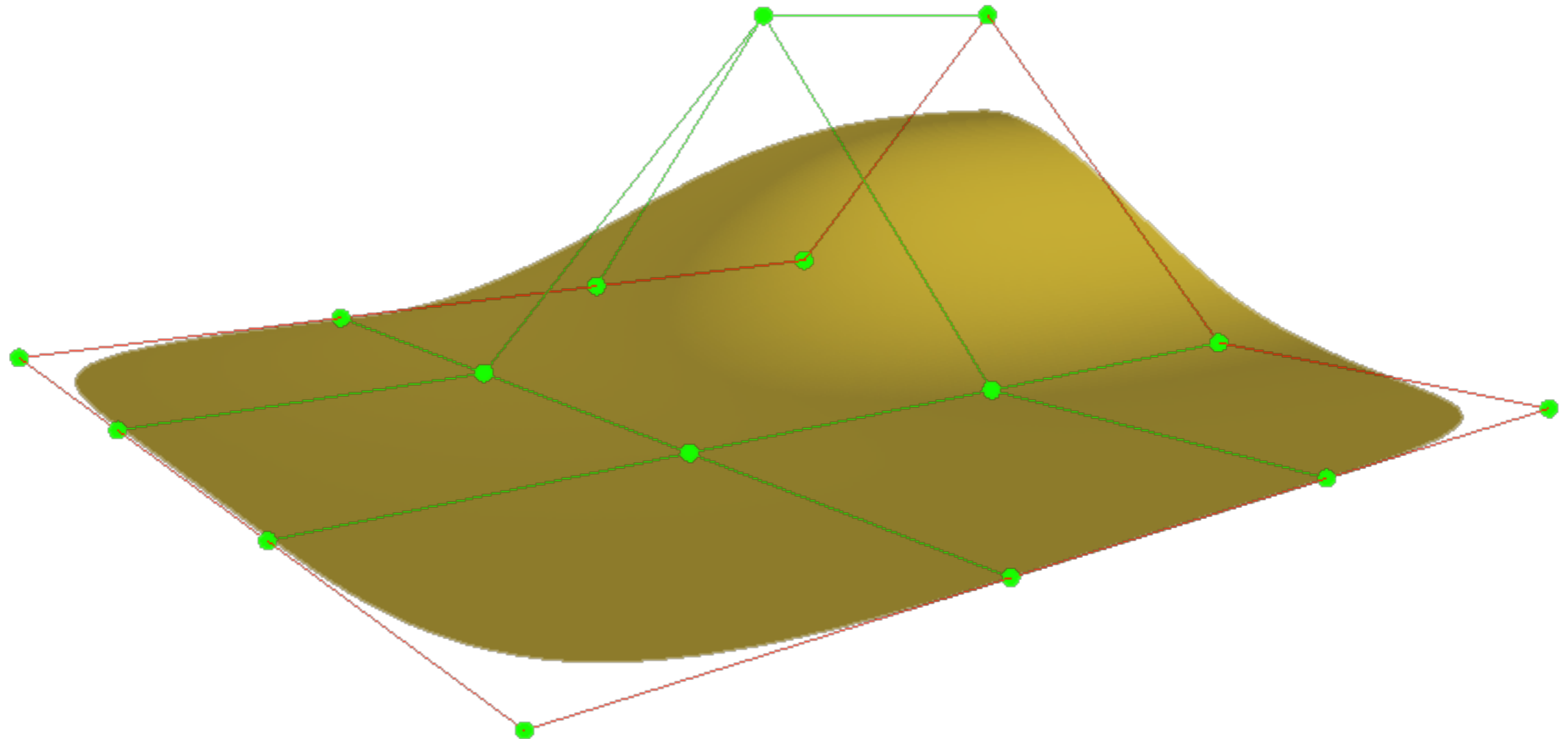
$$\text{LBS: } v' = \sum_{j \in H} T_j w_j(v) \begin{pmatrix} v \\ 1 \end{pmatrix}$$

# Our approach extends to 3D

$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 dt$$

# Our approach extends to 3D

$$E(C') = \int_D \|G_{C'}(u, v) - Target(u, v)\|^2 dudv$$



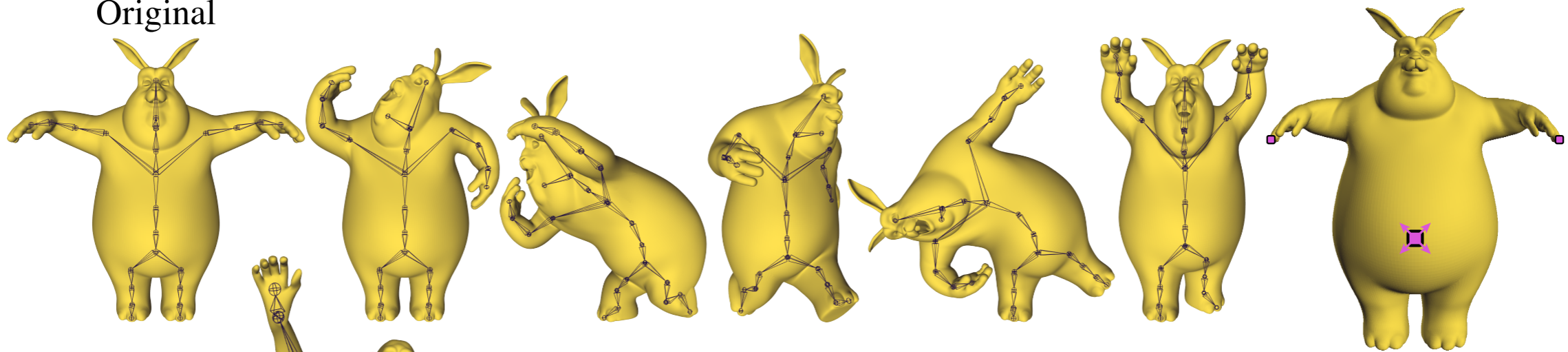
# 3D Live Demo

---

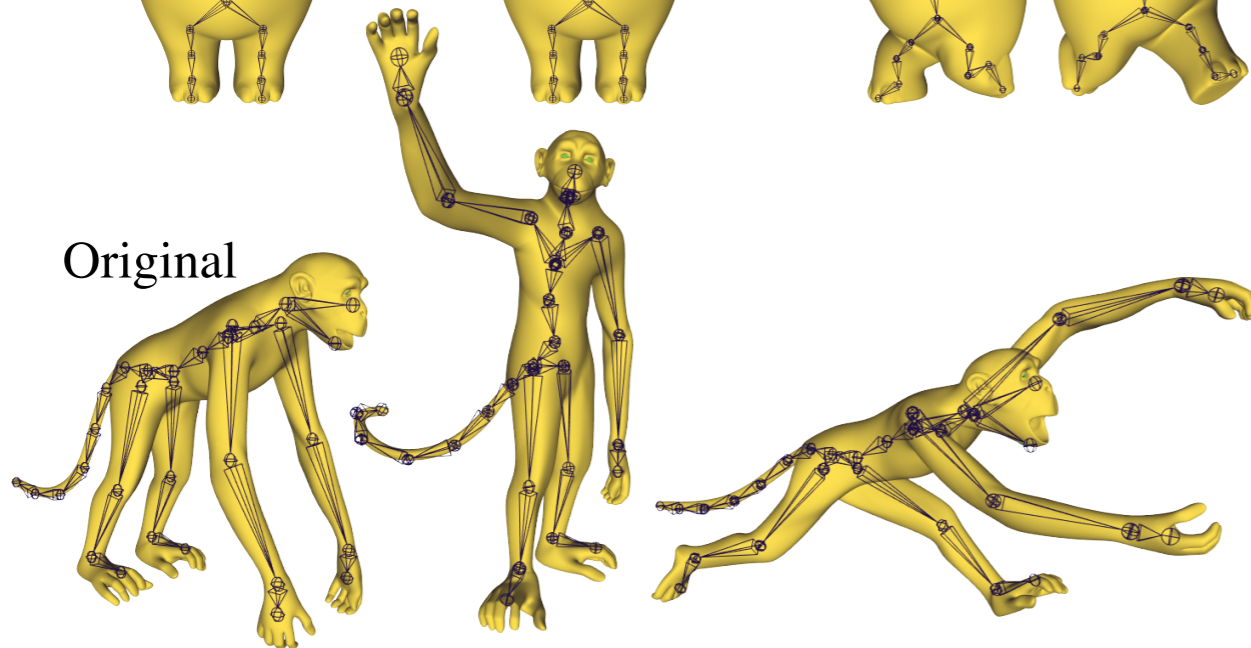


# 3D Results

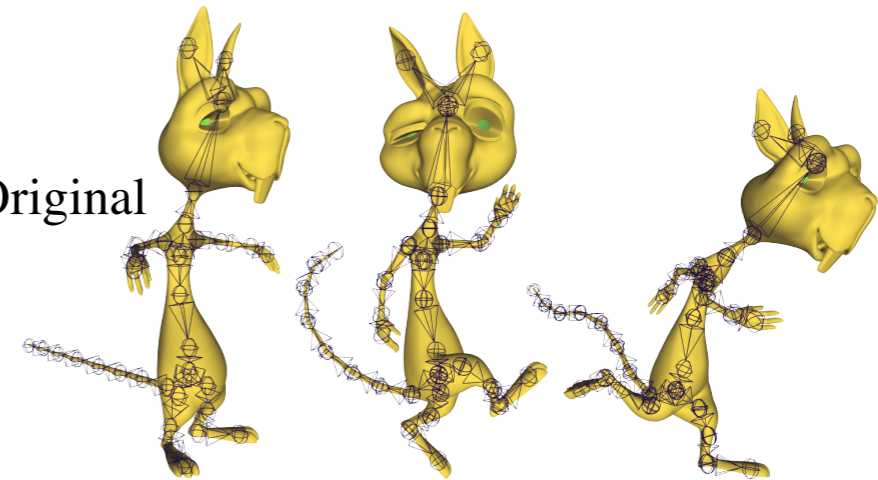
Original



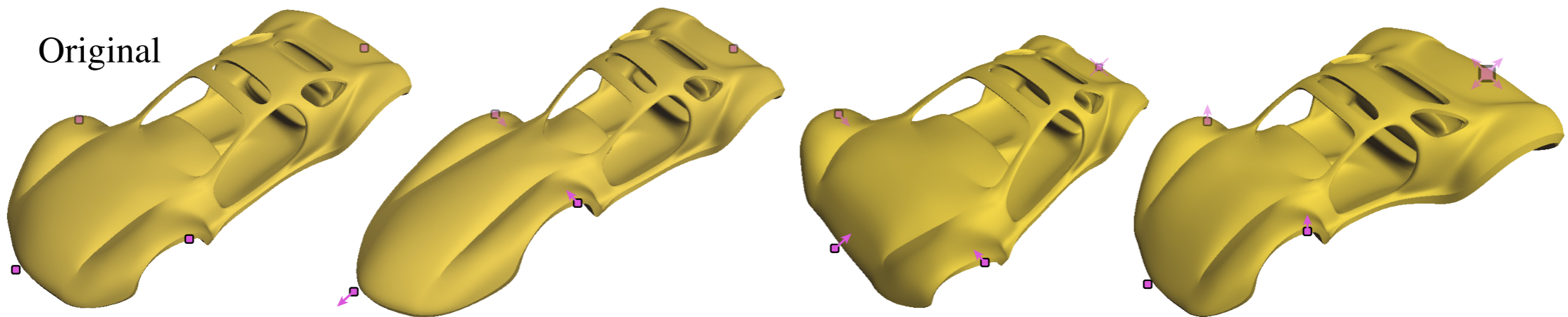
Original



Original

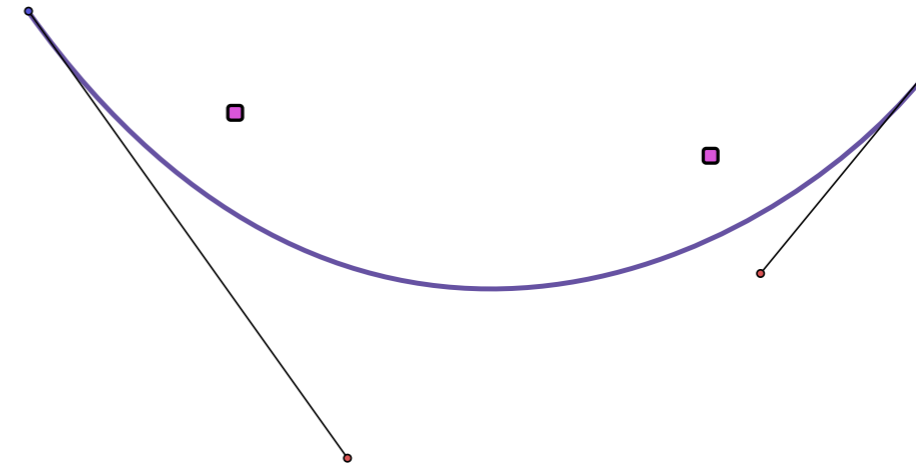
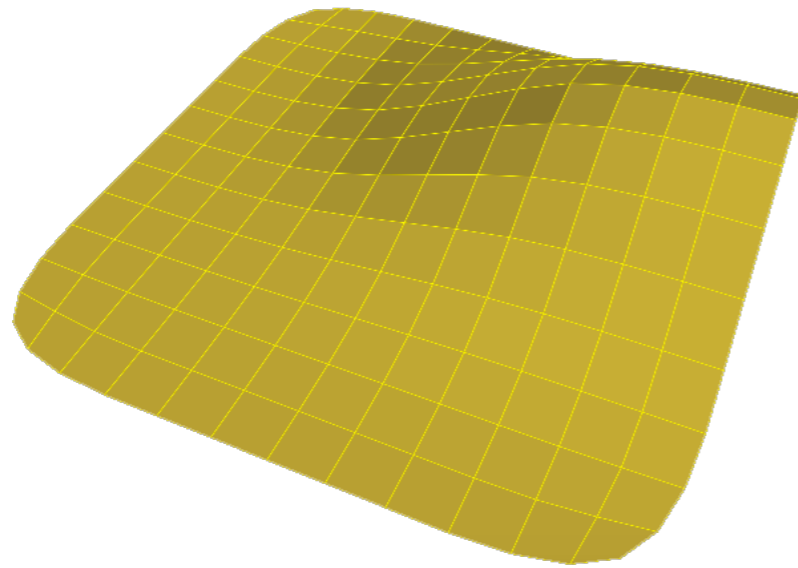


Original

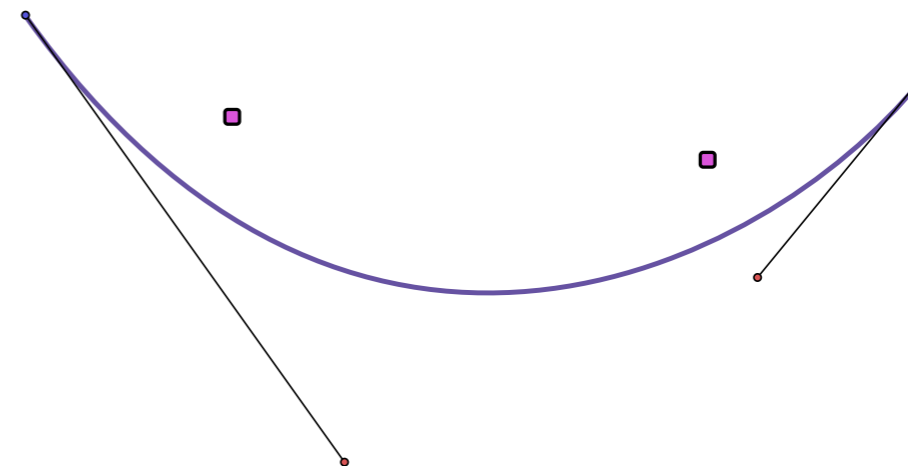
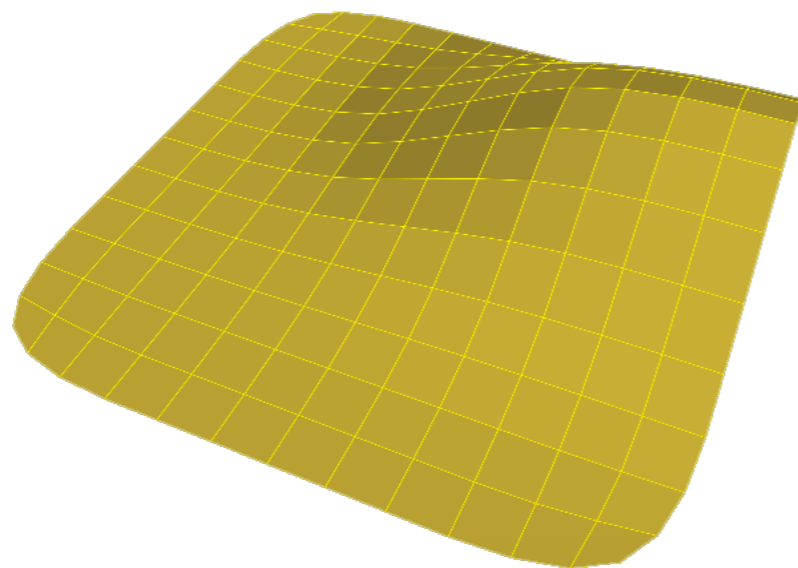




# Why 2D is more complicated?

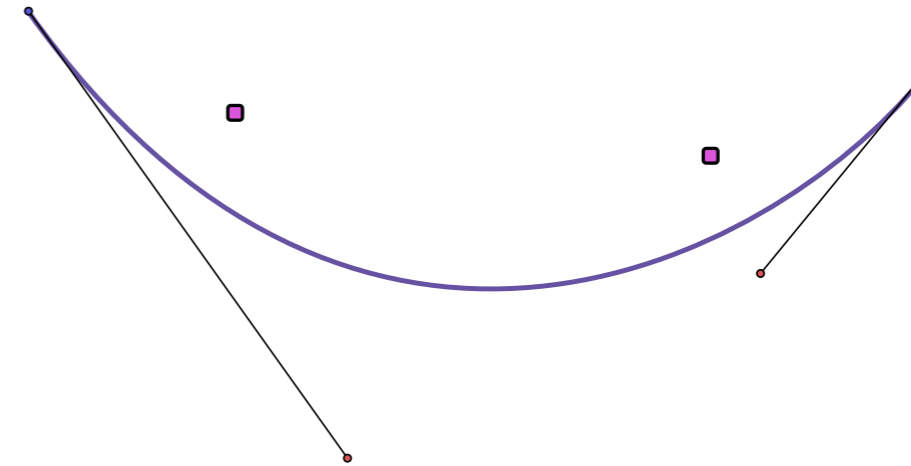
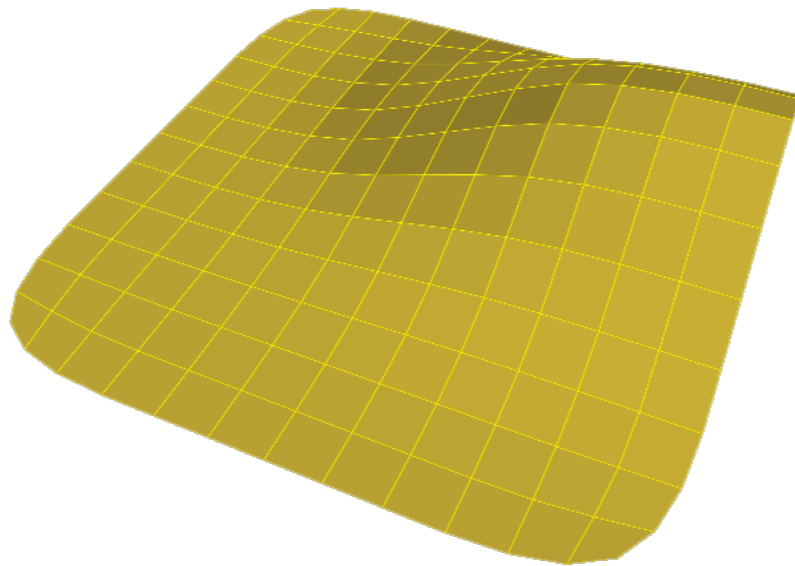


# Why 2D is more complicated?

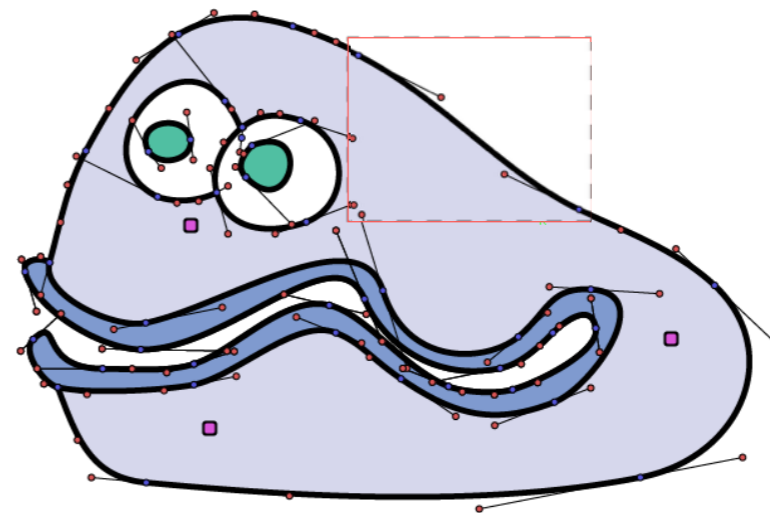
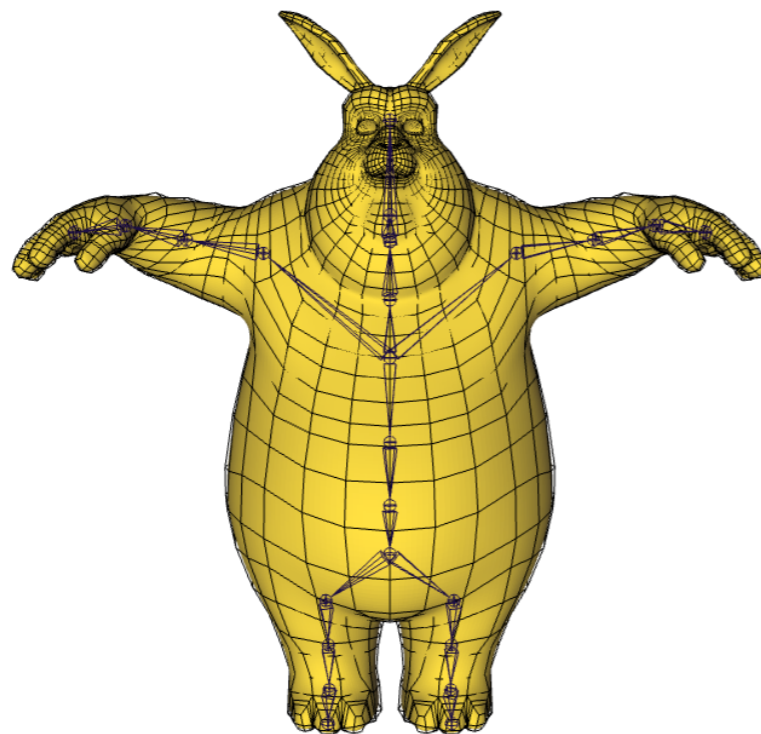


In practice:

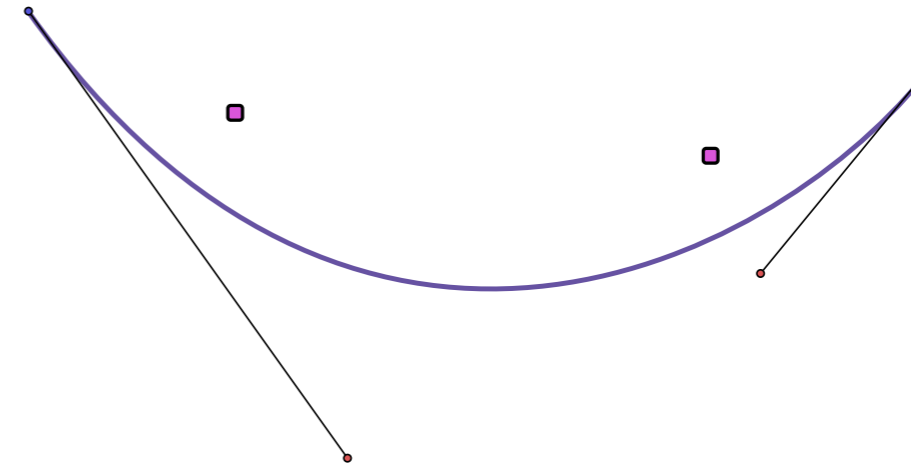
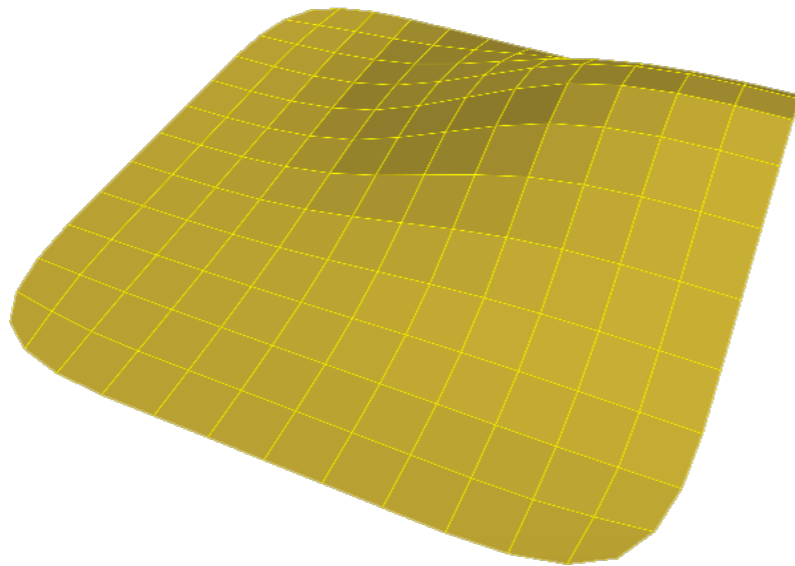
# Why 2D is more complicated?



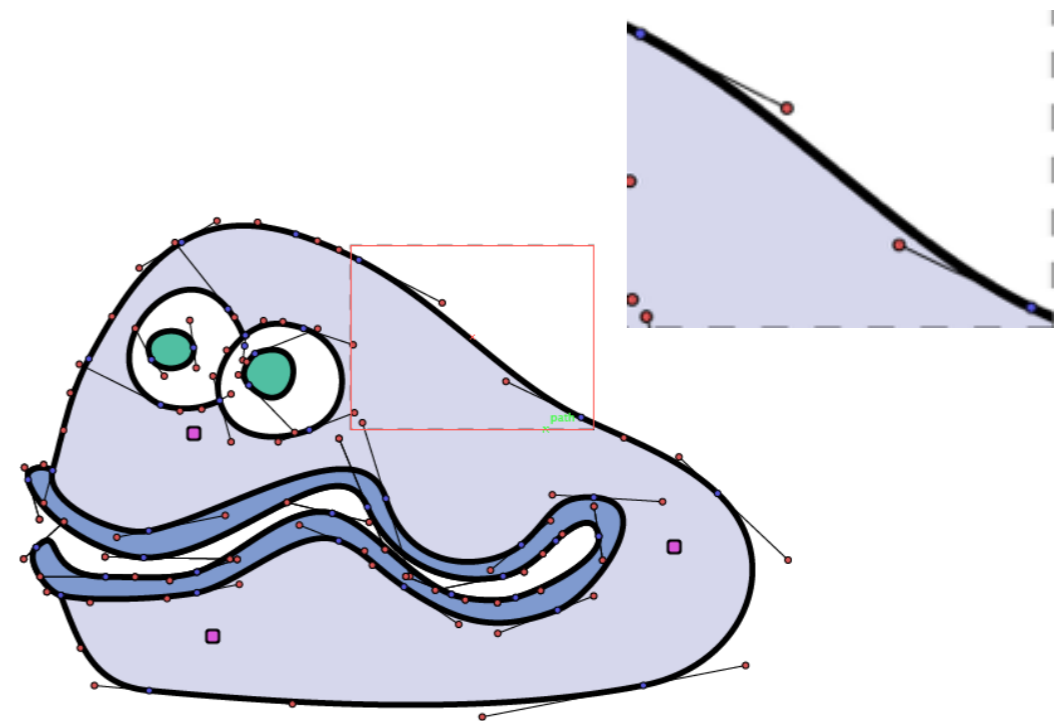
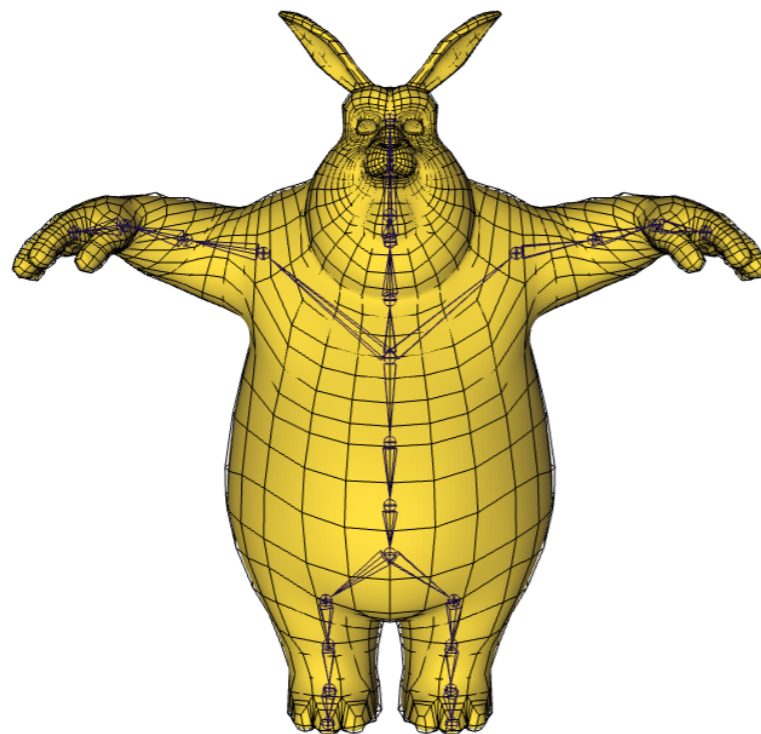
In practice:



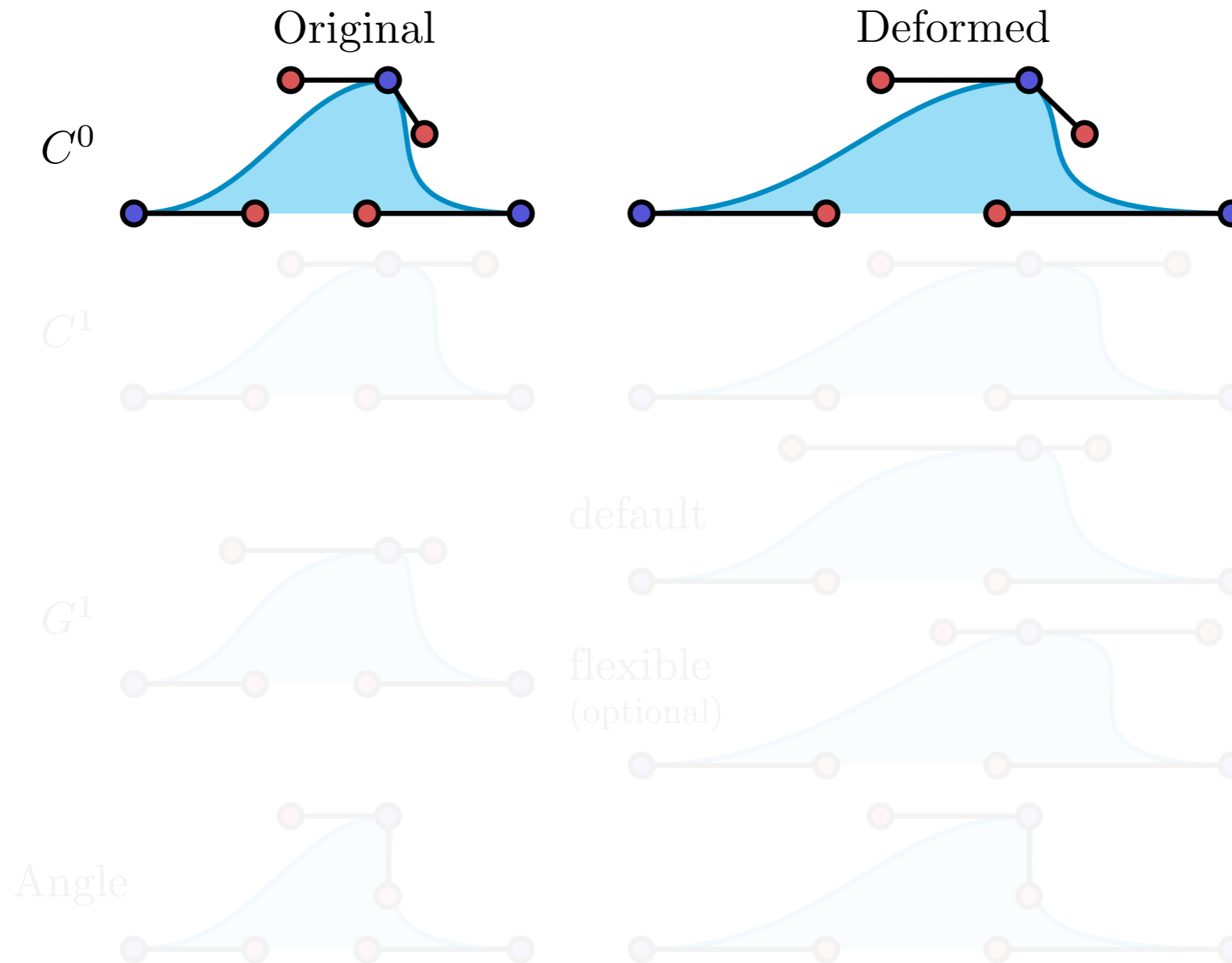
# Why 2D is more complicated?



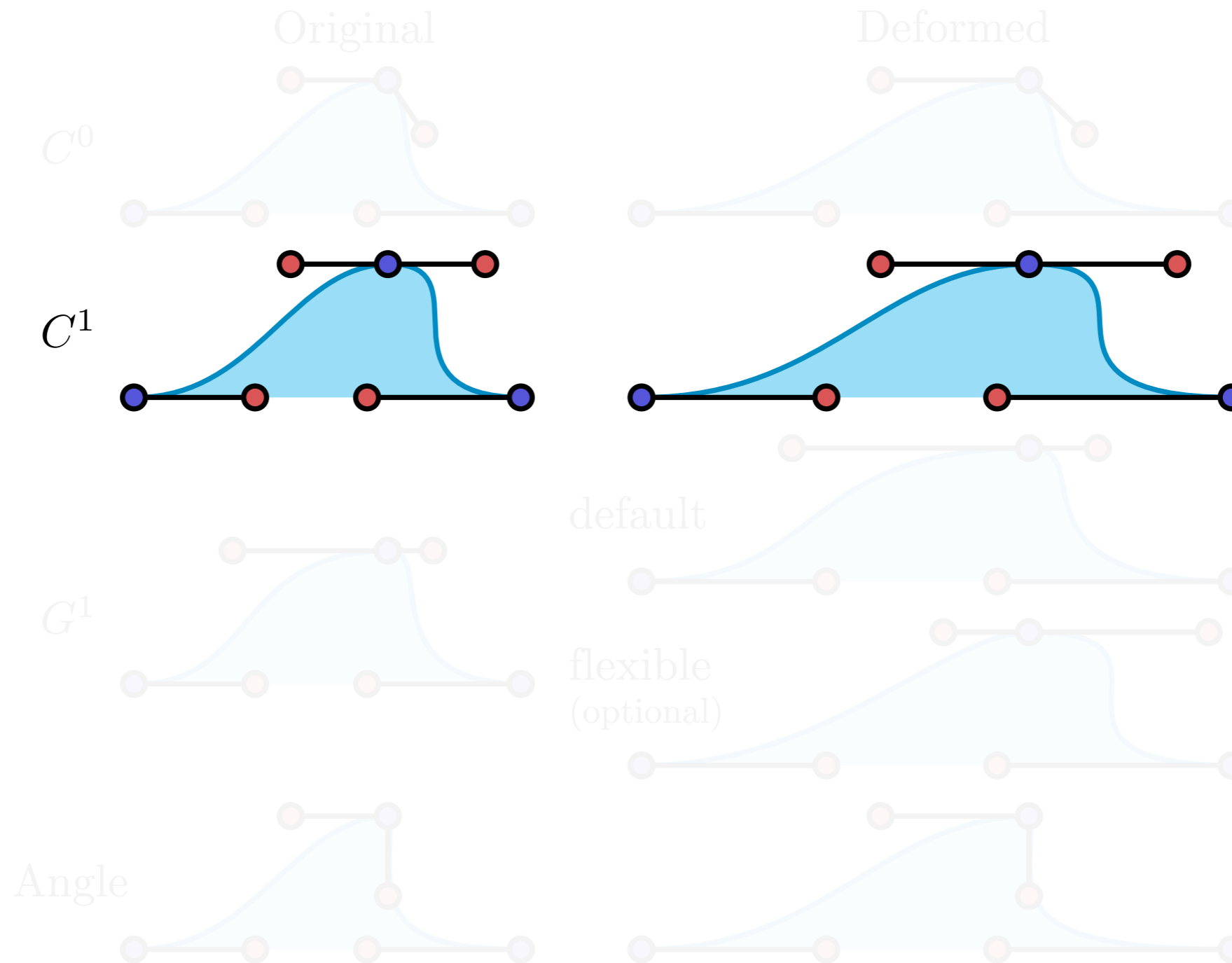
In practice:



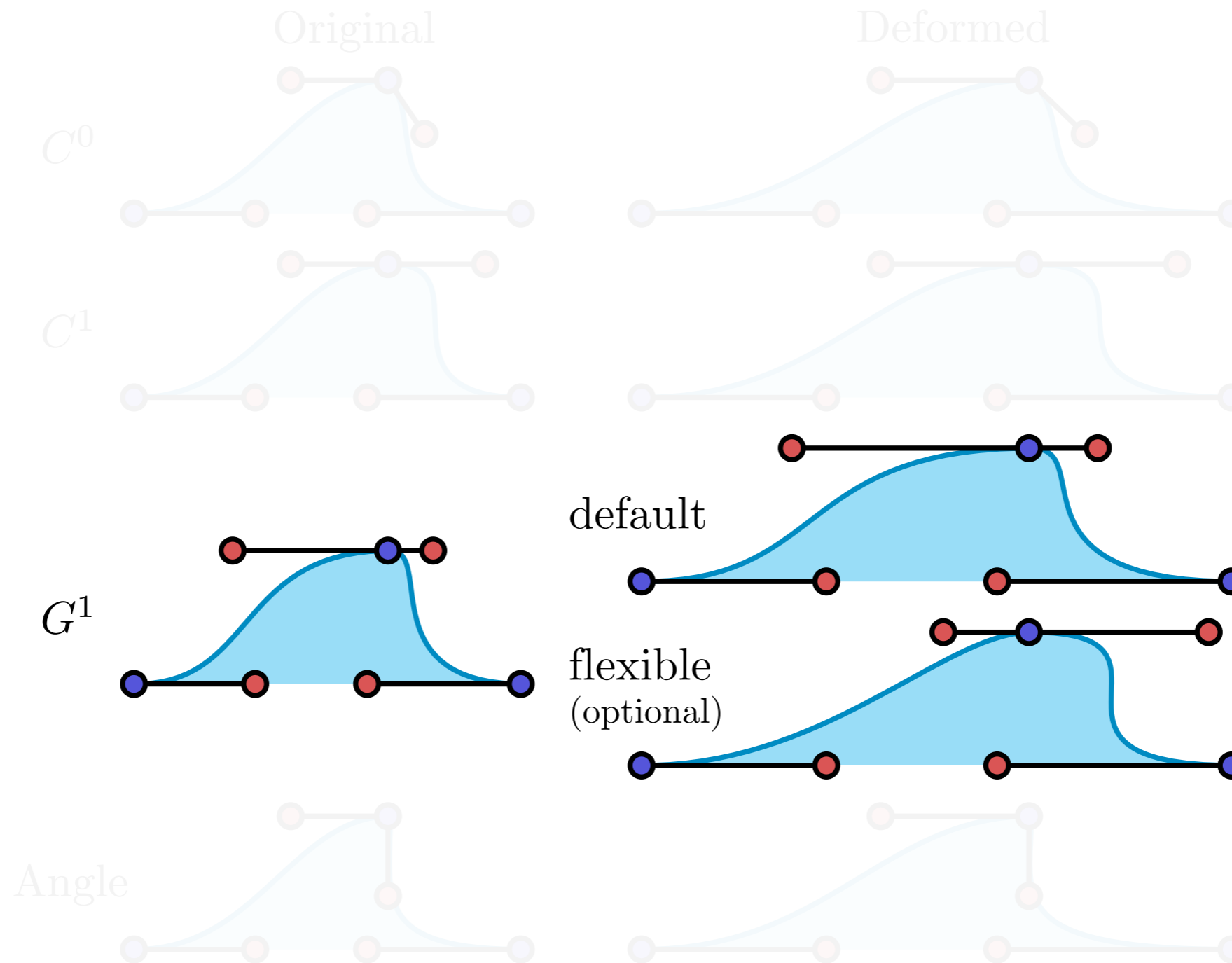
# Bézier Spline Junctions



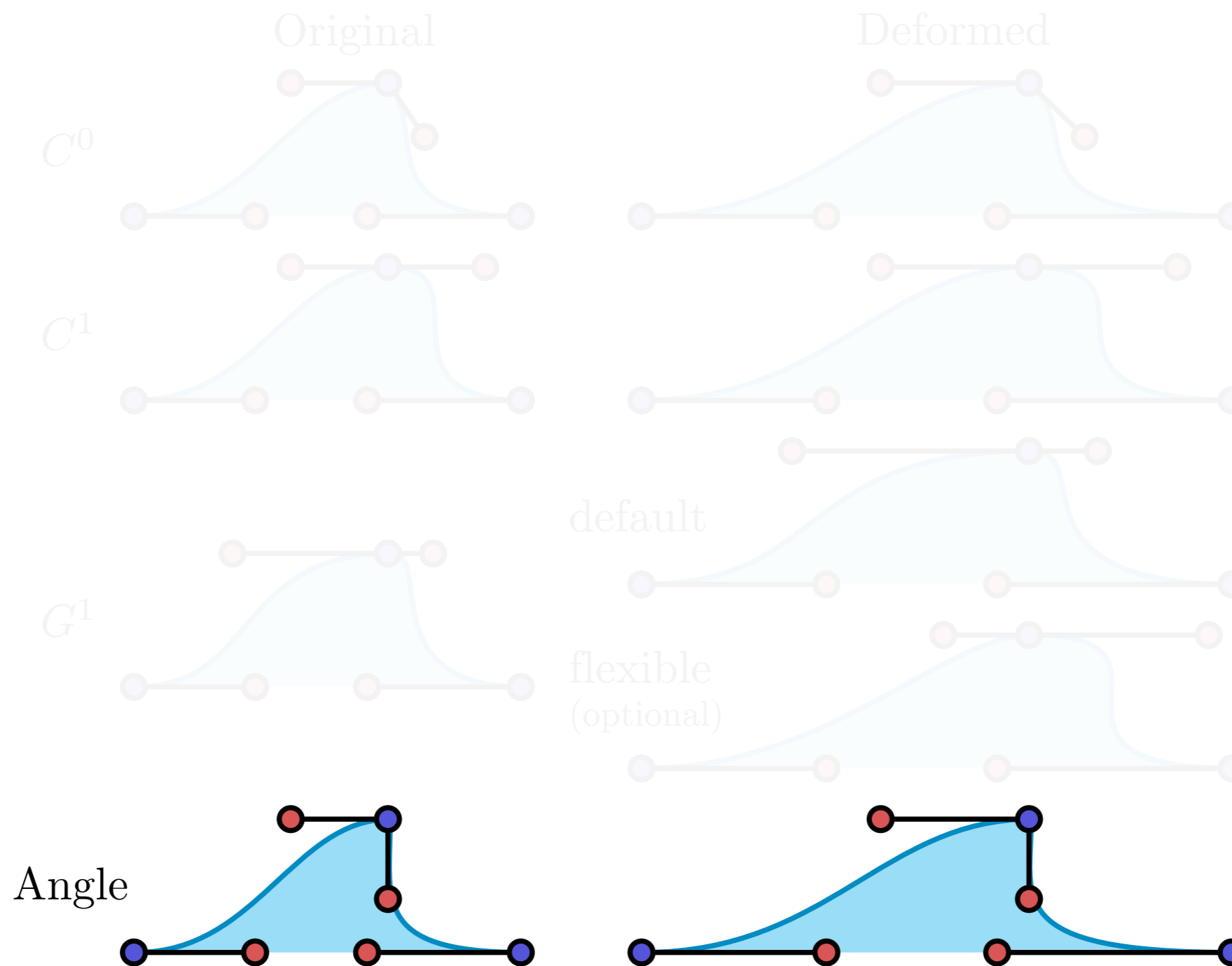
# Bézier Spline Junctions



# Bézier Spline Junctions



# Bézier Spline Junctions





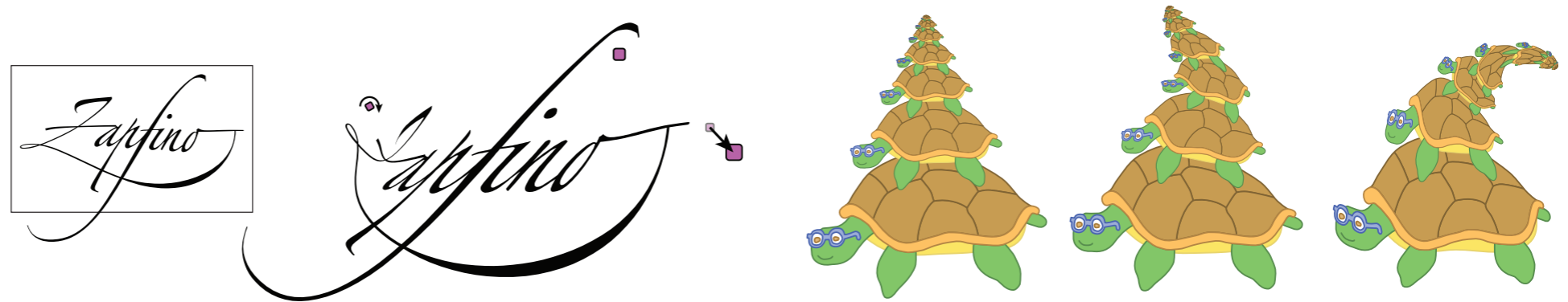
# 2D Live Demo

---

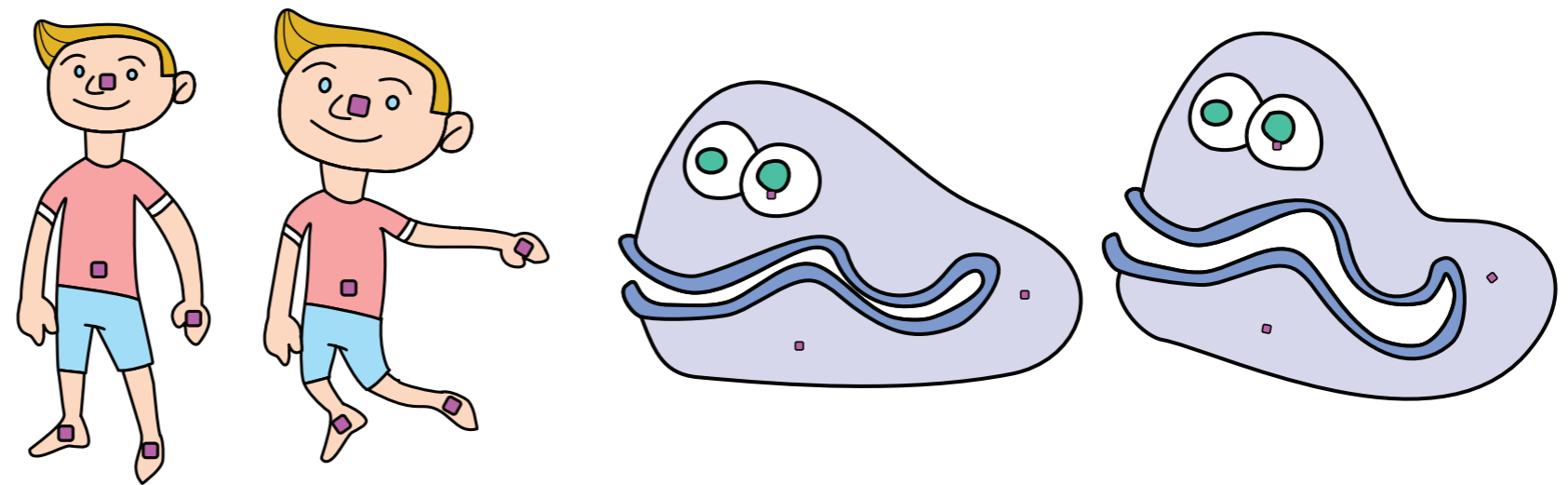


# Applied on different weights

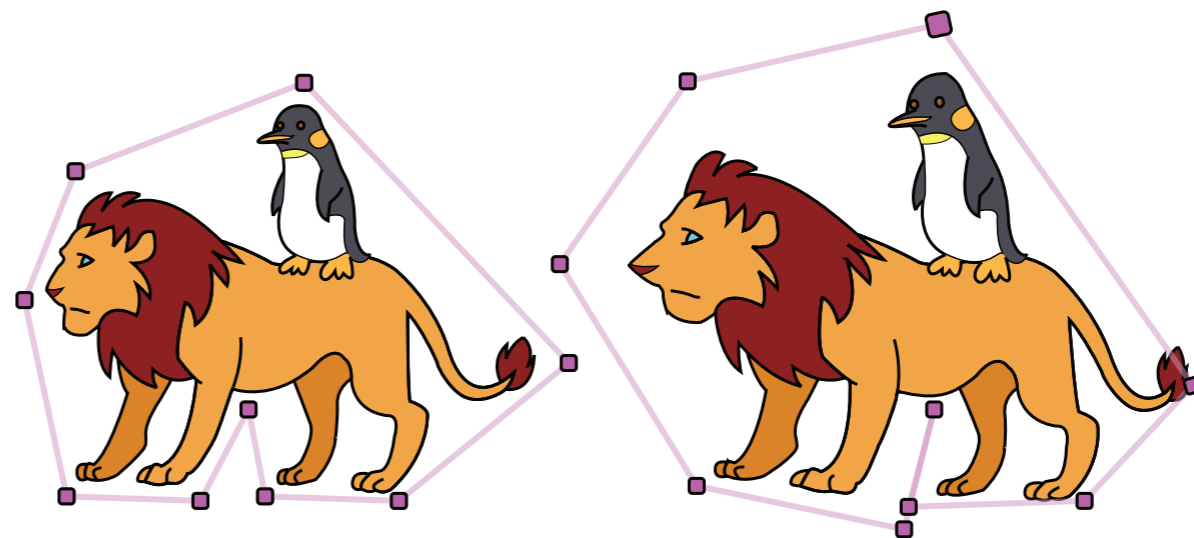
Shepard's Weights  
[Shepard 1968]



Bounded Biharmonic Weights  
[Jacobson et al. 2011]



Harmonic Coordinates  
[Joshi et al. 2007]

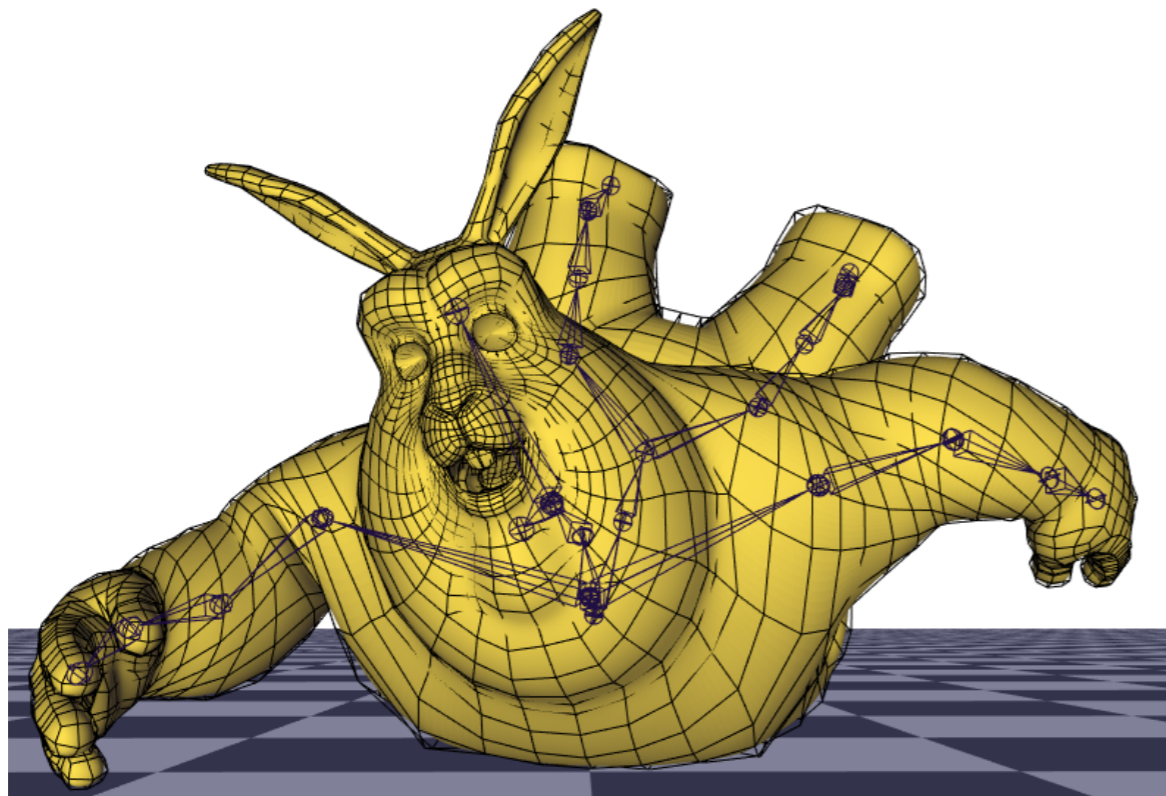


# Performance

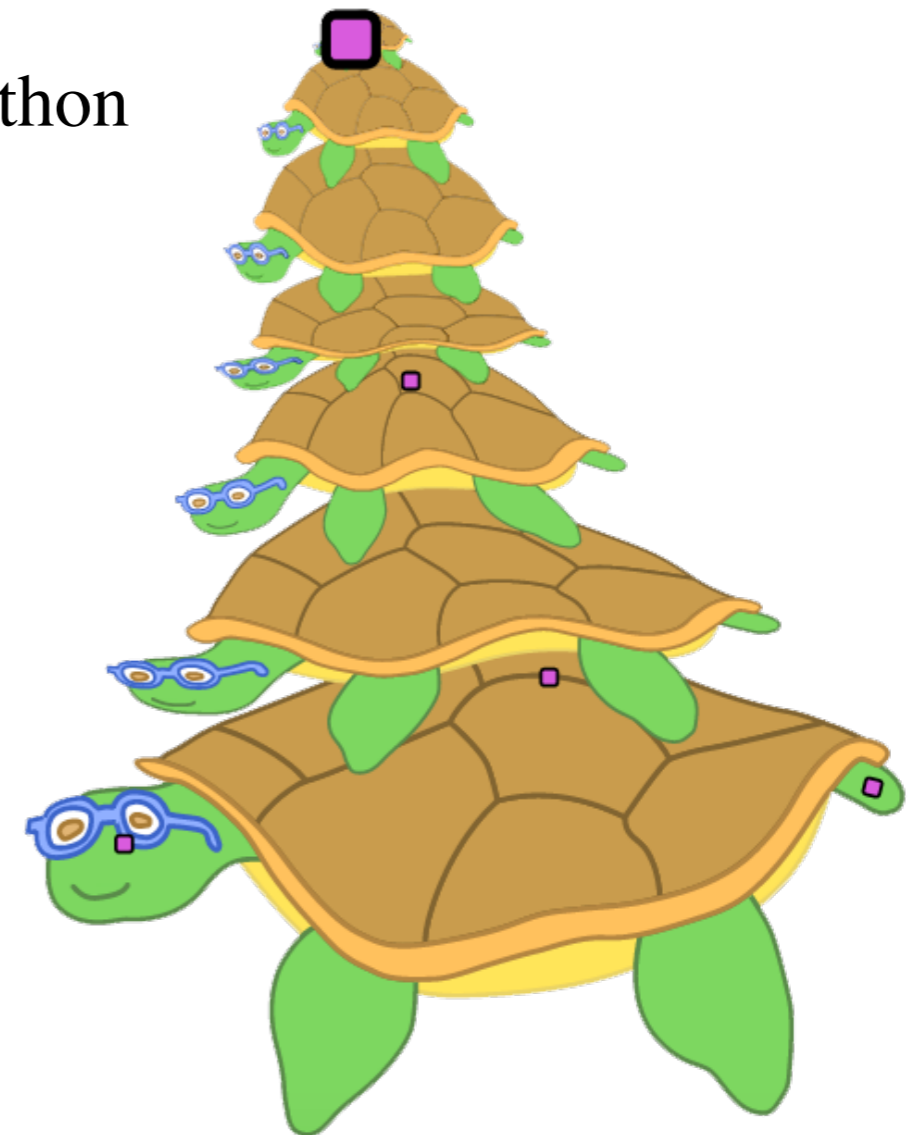
# faces	pre-compute time(secs)	seconds per update
4150	9	0.0002

# curves	pre-compute time(secs)	seconds per update
1324	3.2	0.03

in C++



in Python



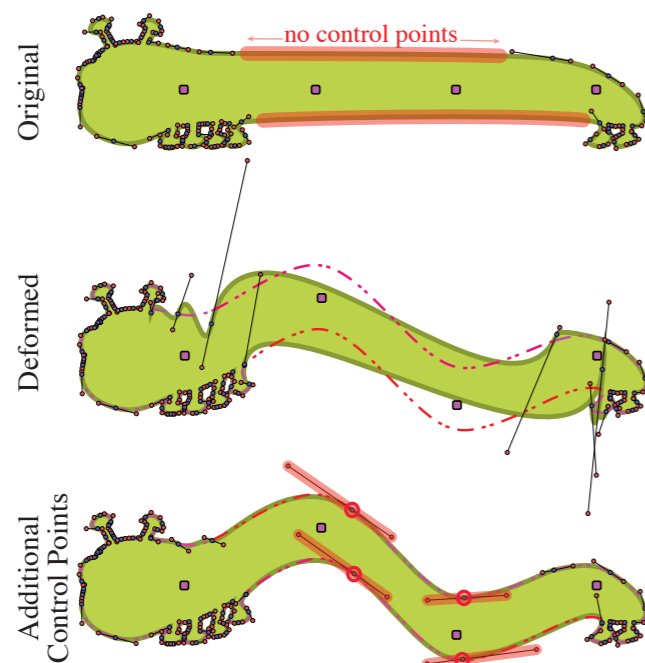
# Limitation and Future Work

---

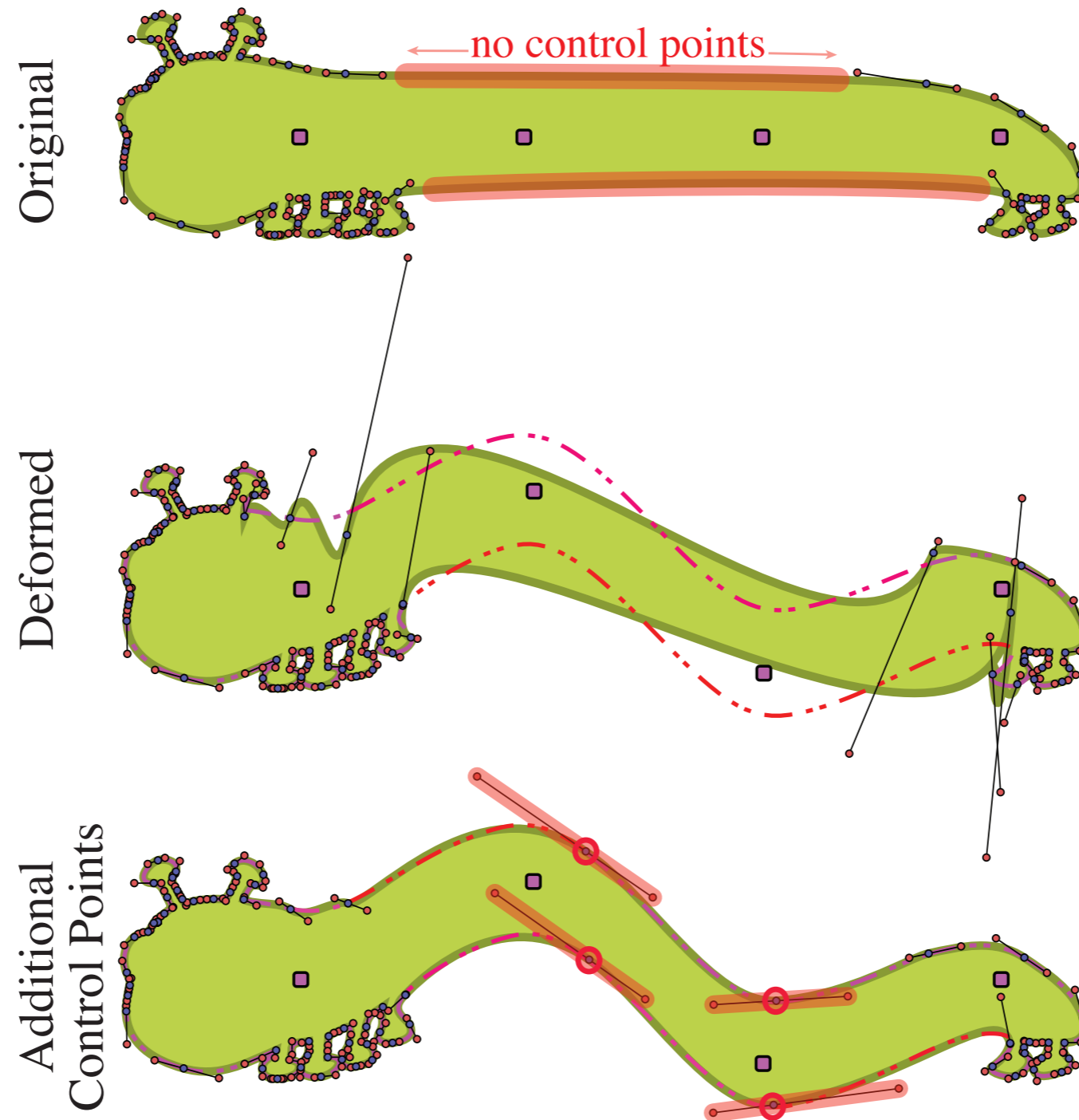


- **Other primitives in vector graphics**
  - **circular arcs, NURBS, clothoid splines.**
  - **linear or radial gradients, diffusion curves, texture.**

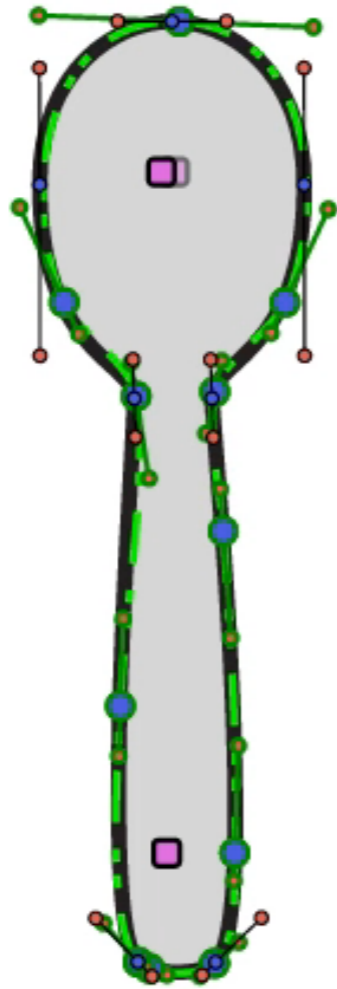
- Other primitives in vector graphics
  - circular arcs, NURBS, clothoid splines.
  - linear or radial gradients, diffusion curves, texture.
- Adaptivity



# Limitation and Future Work



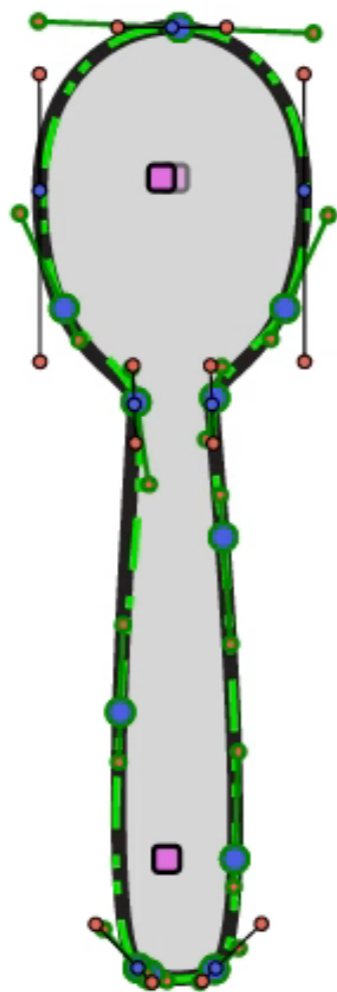
# Limitation and Future Work



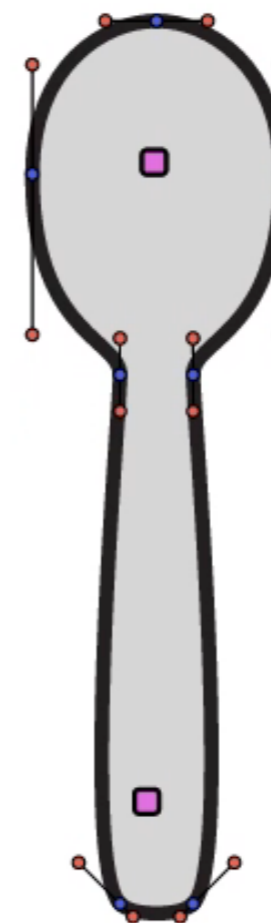
[Schneider 1990]



# Limitation and Future Work



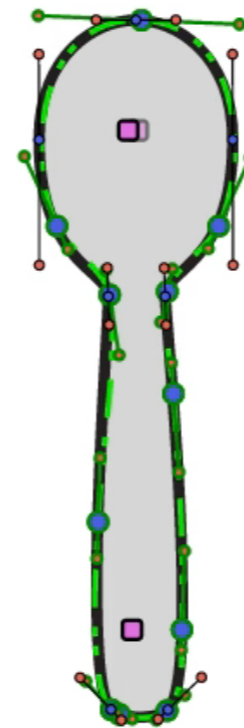
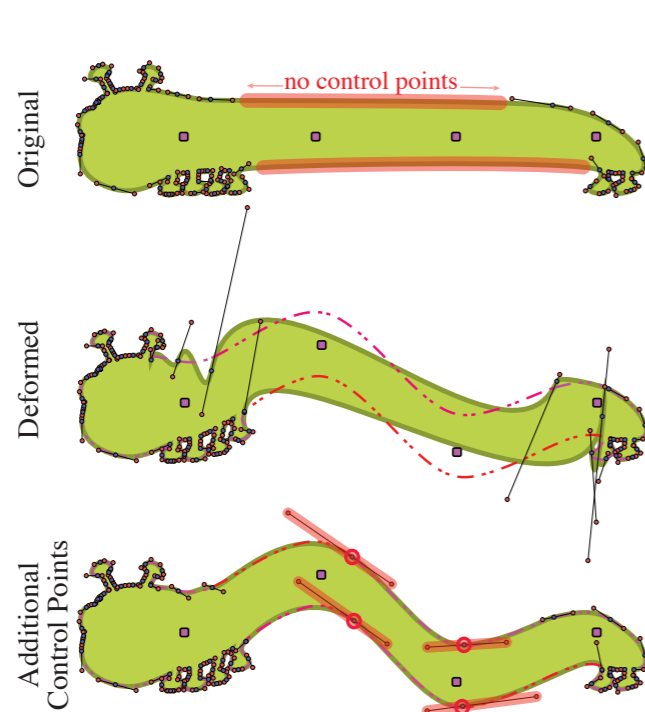
[Schneider 1990]



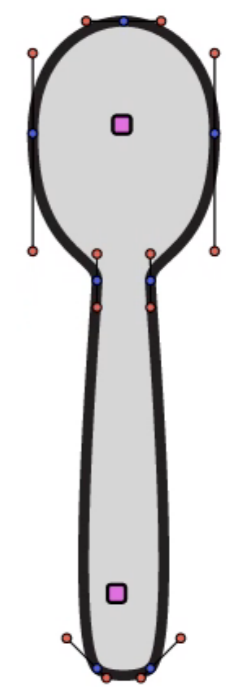
[Our approach]

# Limitation and Future Work

- Other primitives in vector graphics
  - circular arcs, NURBS, clothoid splines.
  - linear or radial gradients, diffusion curves, texture.
- Adaptivity



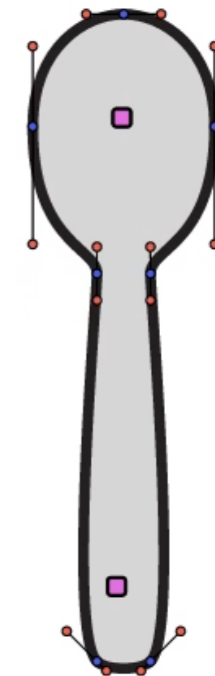
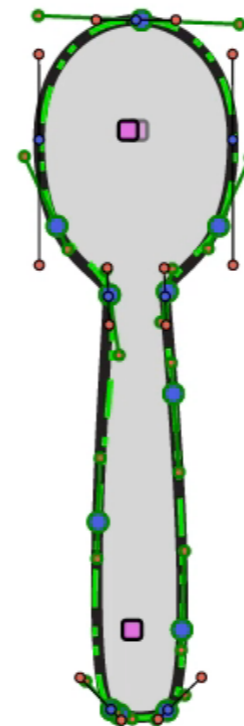
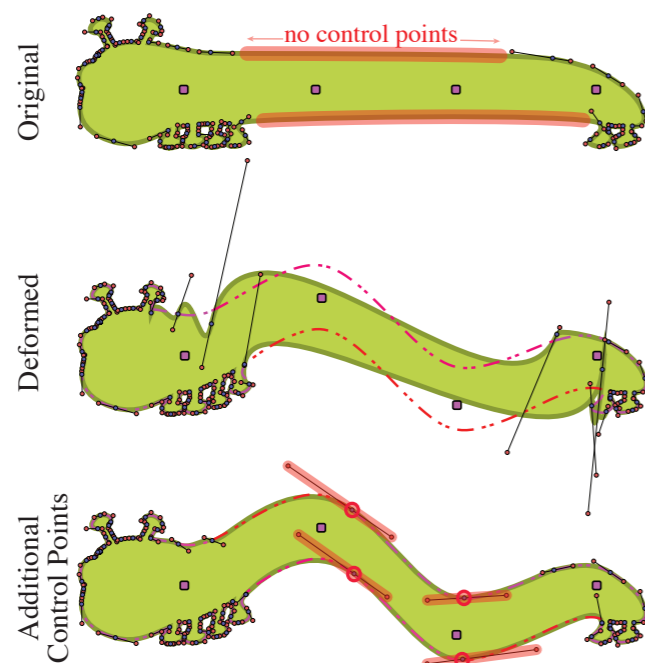
[Schneider 1990]



[Our approach]

# Limitation and Future Work

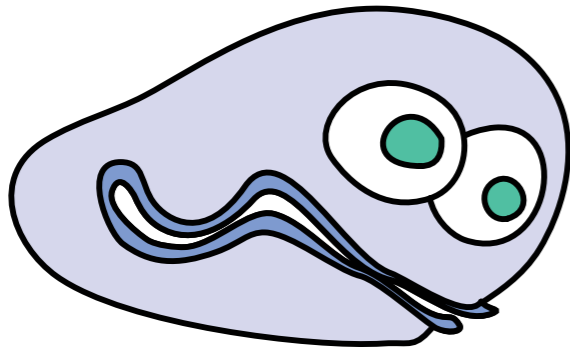
- **Other primitives in vector graphics**
  - circular arcs, NURBS, clothoid splines.
  - linear or radial gradients, diffusion curves, texture.
- **Adaptivity**



- **Other applications**
  - motion path planning

[Schneider 1990]

[Our approach]



# Thank You.

---

Songrun Liu, [sliu11@gmu.edu](mailto:sliu11@gmu.edu)

Alec Jacobson, [jacobson@cs.columbia.edu](mailto:jacobson@cs.columbia.edu)

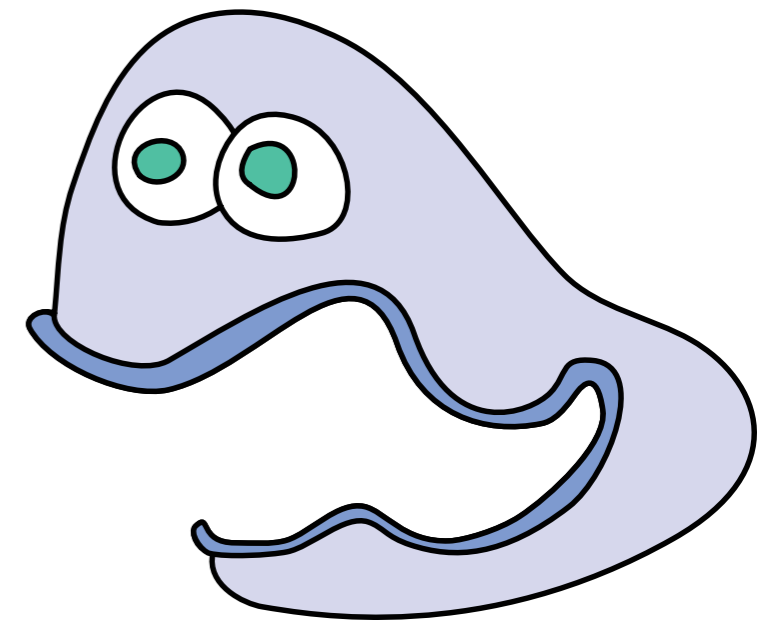
Yotam Gingold, [ygingold@gmu.edu](mailto:ygingold@gmu.edu)

project webpage: <http://cs.gmu.edu/~ygingold/splineskin/>, code coming soon.

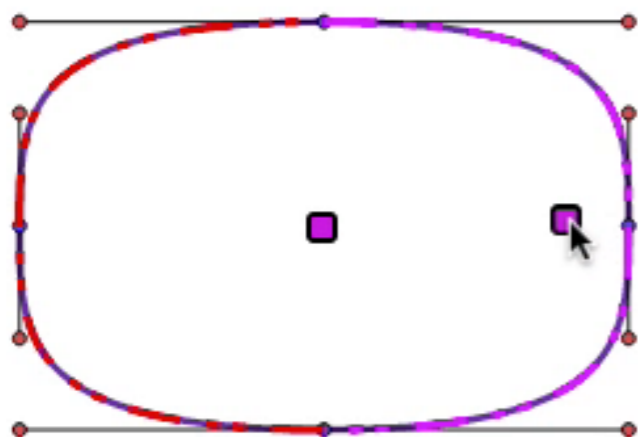
## Acknowledgements:

We thank support from NSF, Google, Intel, The Walt Disney Company, and Autodesk.

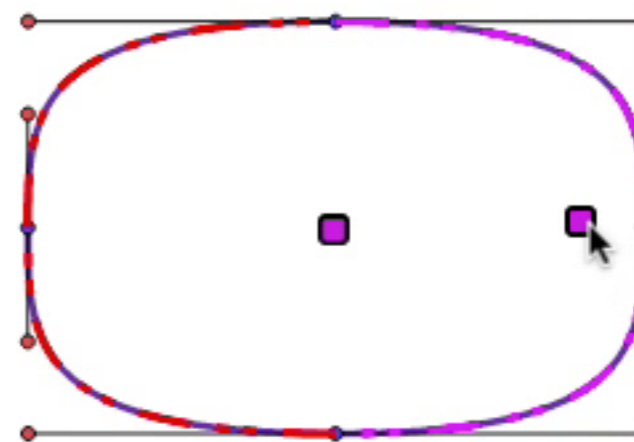
Special thanks to Michelle Lee for her 2D artwork and Blender Foundation for 3D models from “Big Buck Bunny”.



# Junction between Curves (smooth)

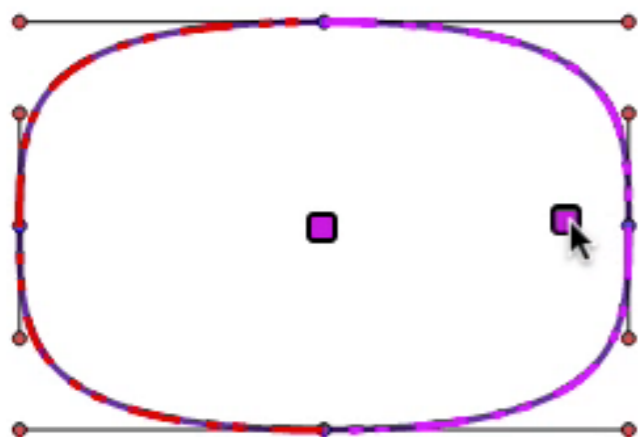


$C^1$  continuity

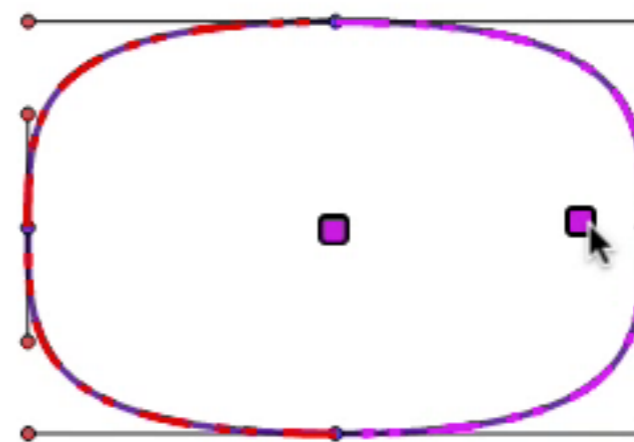


$G^1$  flexibility

# Junction between Curves (smooth)

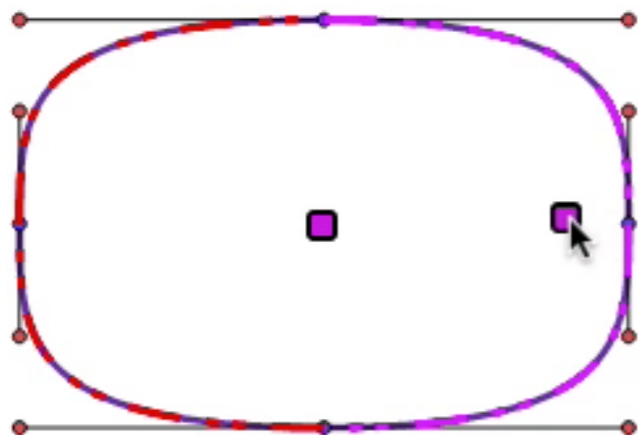


$C^1$  continuity

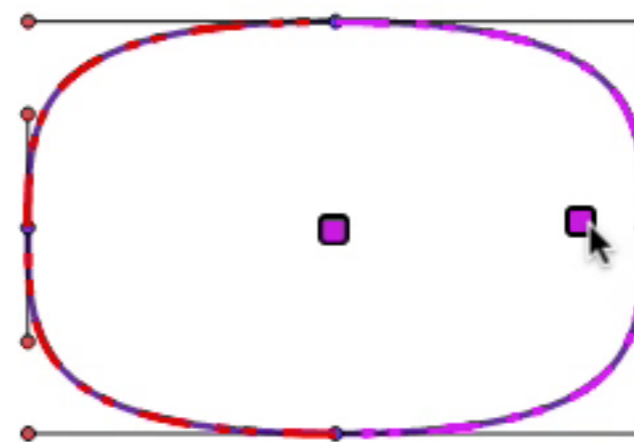


$G^1$  flexibility

# Junction between Curves (smooth)

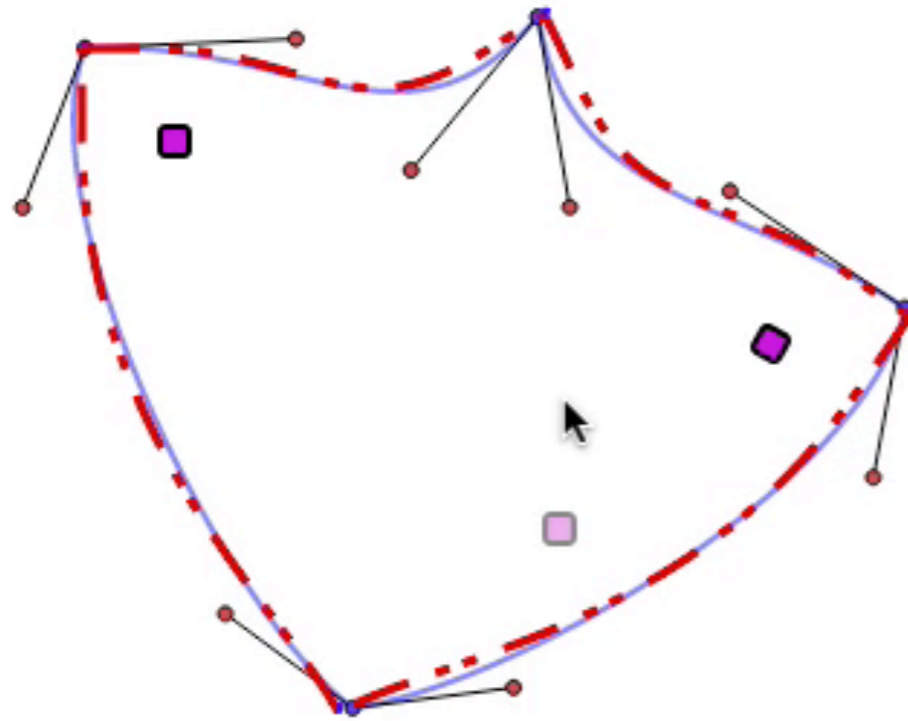


$C^1$  continuity

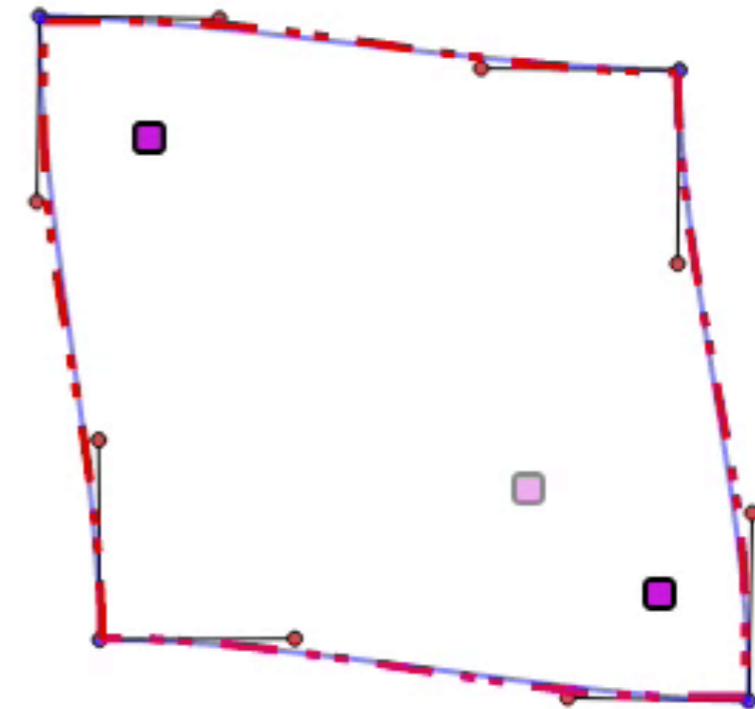


$G^1$  flexibility

# Junction between Curves (sharp)



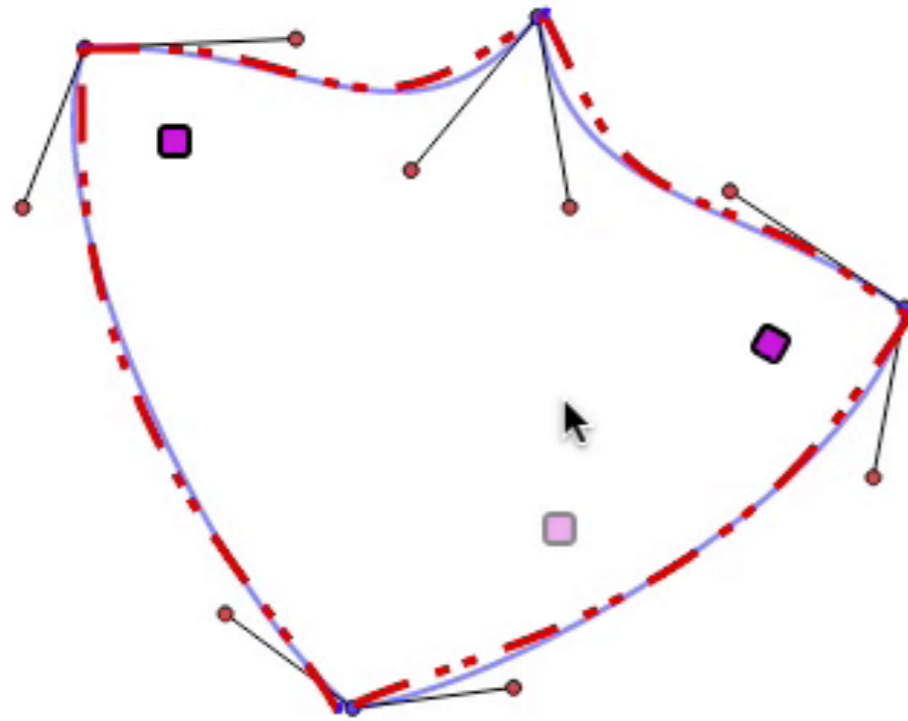
No Angle Preservation



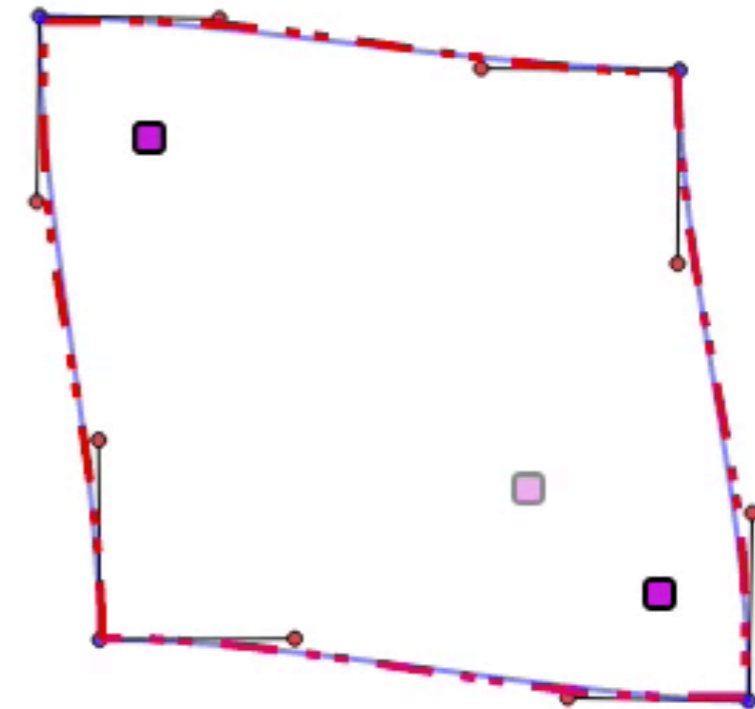
Angle Preservation



# Junction between Curves (sharp)



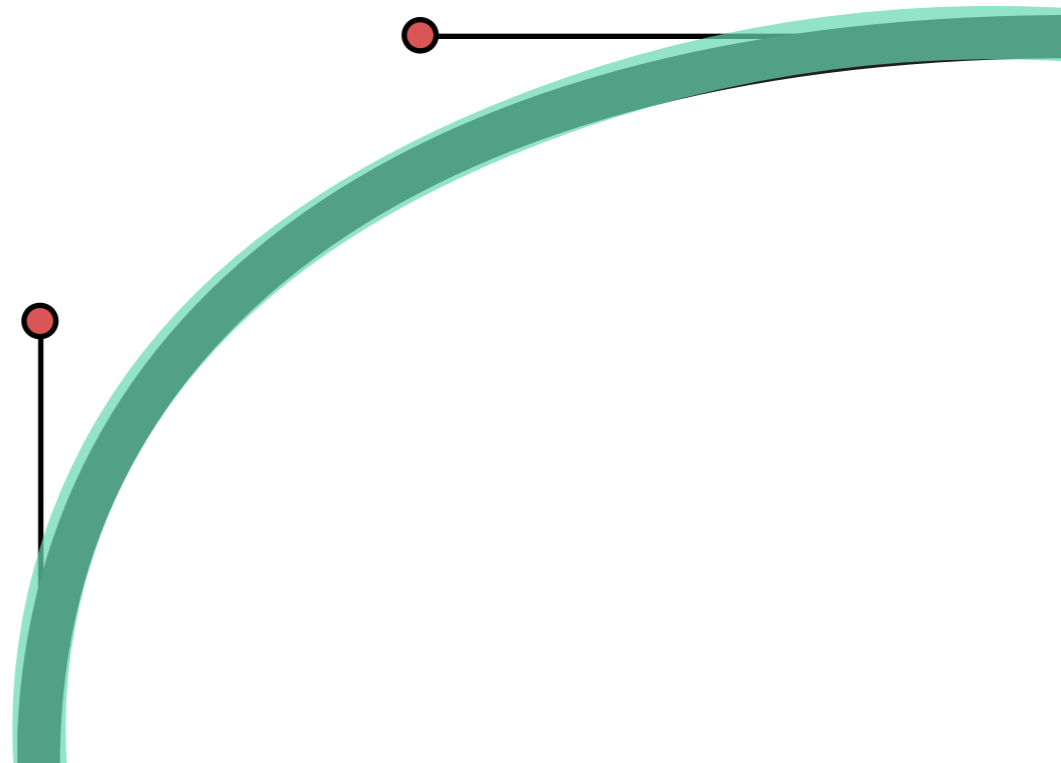
No Angle Preservation



Angle Preservation

- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = CM\bar{t} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = Cm_t,$$



- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = CM\bar{t} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = Cm_t,$$



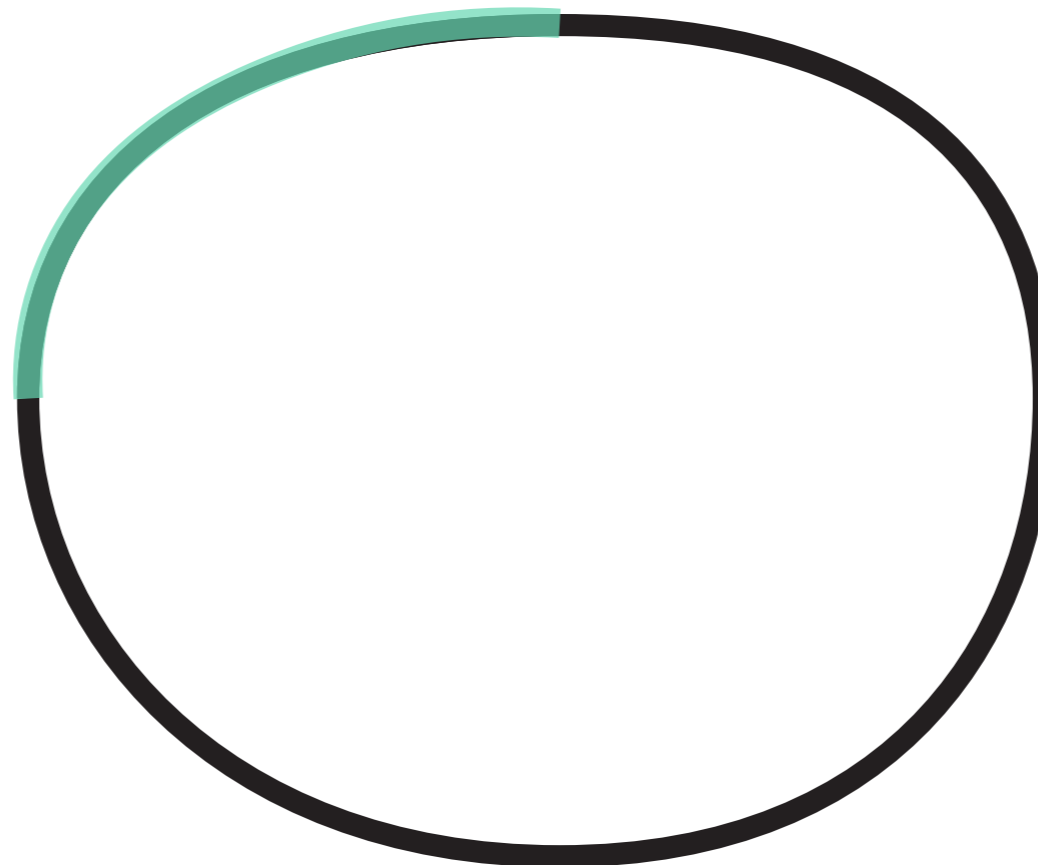
- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = CM\bar{t} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = Cm_t,$$



- Splines in 2D — Cubic Bézier splines.

$$B_C(t) = CM\bar{t} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = Cm_t,$$



- Minimizing the L2 norm

$$E(C') = \int_D \left\| G_{C'}(p) - \sum_{i=1}^h w_i(G_C(p)) T_i G_C(p) \right\|^2 dp$$

$$C' = \sum_{i=1}^h T_i \hat{W}_i \hat{A}^{-1} \quad \leftarrow \quad \text{Pre-computed}$$

$$A(p) = m_p m_p^T \quad \hat{A} = \int_D A(p) dp \quad \hat{W}_i = C \int_D w_i(C m_p) A(p) dp$$

- Minimizing the L2 norm

$$E(C') = \int_D \left\| G_{C'}(p) - \sum_{i=1}^h w_i(G_C(p)) T_i G_C(p) \right\|^2 dp$$

$$C' = \sum_{i=1}^h T_i \hat{W}_i \hat{A}^{-1} \quad \leftarrow \text{Pre-computed}$$

$$A(p) = m_p m_p^T \quad \hat{A} = \int_D A(p) dp \quad \hat{W}_i = C \int_D w_i(C m_p) A(p) dp$$

$$\text{LBS: } \mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

- Play the video of minimizing the L2 norm

$$E(C') = \int_D \left\| G_{C'}(p) - \sum_{i=1}^h w_i(G_C(p)) T_i G_C(p) \right\|^2 dp$$

$$C' = \sum_{i=1}^h T_i \hat{W}_i \hat{A}^{-1} \quad \leftarrow \text{Pre-computed}$$

↓

$$\hat{A}^{-1} = \begin{bmatrix} 16 & -24 & 16 & -4 \\ -24 & 69\frac{1}{3} & -57\frac{1}{3} & 16 \\ 16 & -57\frac{1}{3} & 69\frac{1}{3} & -24 \\ -4 & 16 & -24 & 16 \end{bmatrix}$$

$$\hat{W}_i = C \int_D w_i(Cm_p) A(p) dp$$



# 3D Performance

example	# control vertices	# control faces	precompute time (secs)	seconds per update
pyramid	5	5	1e-3	6e-6
torus	32	32	8e-3	6e-6
butterfly	1216	1222	0.3	5e-5
squirrel	4307	4278	4	2e-4
rabbit	4139	4150	9	2e-4

**Dual-core, 2.4 GHz Intel Core i5, in C++.**

# 3D Performance

example	# control vertices	# control faces	precompute time (secs)	seconds per update
pyramid	5	5	1e-3	6e-6
torus	32	32	8e-3	6e-6
butterfly	1216	1222	0.3	5e-5
squirrel	4307	4278	4	2e-4
rabbit	4139	4150	9	2e-4

**Dual-core, 2.4 GHz Intel Core i5, in C++.**

# 2D Performance

example	# curves	# handles	# $G^1$ & angle constraints	precomp. secs	seconds per update	
					regular	flexible
Boxes (Fig. 6)	4	2	4	0.03	2e-4	0.008
Spoon (Fig. 3)	7	2	7	0.08	2e-4	0.02
Clam (Fig. 1)	56	3	50	0.1	8e-4	0.1
Boy	226	5	118	0.7	5e-3	0.4
Zapfino	379	3	316	1.4	4e-3	0.65
Worm	395	2	261	8.1	3e-2	2.75
Penguin on Lion	400	9	168	2.5	1e-2	0.64
Man	516	4	240	1.3	1e-2	0.78
Seven Turtles	1324	5	914	3.2	3e-2	2.5
Octopus	1706	8	1181	11	2e-2	4.2
Coat of Arms	9496	4	8159	42	1e-1	16.1

**Dual-core, 2 GHz Intel Core i7, in Python.**

# 2D Performance

example	# curves	# handles	# $G^1$ & angle constraints	precomp. secs	seconds per update	
					regular	flexible
Boxes (Fig. 6)	4	2	4	0.03	2e-4	0.008
Spoon (Fig. 3)	7	2	7	0.08	2e-4	0.02
Clam (Fig. 1)	56	3	50	0.1	8e-4	0.1
Boy	226	5	118	0.7	5e-3	0.4
Zapfino	379	3	316	1.4	4e-3	0.65
Worm	395	2	261	8.1	3e-2	2.75
Penguin on Lion	400	9	168	2.5	1e-2	0.64
Man	516	4	240	1.3	1e-2	0.78
Seven Turtles	1324	5	914	3.2	3e-2	2.5
Octopus	1706	8	1181	11	2e-2	4.2
Coat of Arms	9496	4	8159	42	1e-1	16.1

**Dual-core, 2 GHz Intel Core i7, in Python.**

# 2D Performance

example	# curves	# handles	# $G^1$ & angle constraints	precomp. secs	seconds per update	
					regular	flexible
Boxes (Fig. 6)	4	2	4	0.03	2e-4	0.008
Spoon (Fig. 3)	7	2	7	0.08	2e-4	0.02
Clam (Fig. 1)	56	3	50	0.1	8e-4	0.1
Boy	226	5	118	0.7	5e-3	0.4
Zapfino	379	3	316	1.4	4e-3	0.65
Worm	395	2	261	8.1	3e-2	2.75
Penguin on Lion	400	9	168	2.5	1e-2	0.64
Man	516	4	240	1.3	1e-2	0.78
Seven Turtles	1324	5	914	3.2	3e-2	2.5
Octopus	1706	8	1181	11	2e-2	4.2
Coat of Arms	9496	4	8159	42	1e-1	16.1

**Dual-core, 2 GHz Intel Core i7, in Python.**