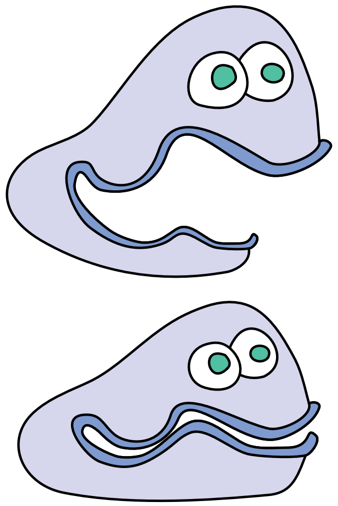
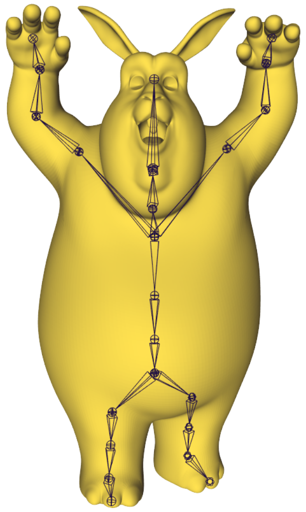
Skinning Cubic Bézier Splines and Catmull-Clark Subdivision Surfaces

SIGGRAPH ASIA 2014 SHENZHEN

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

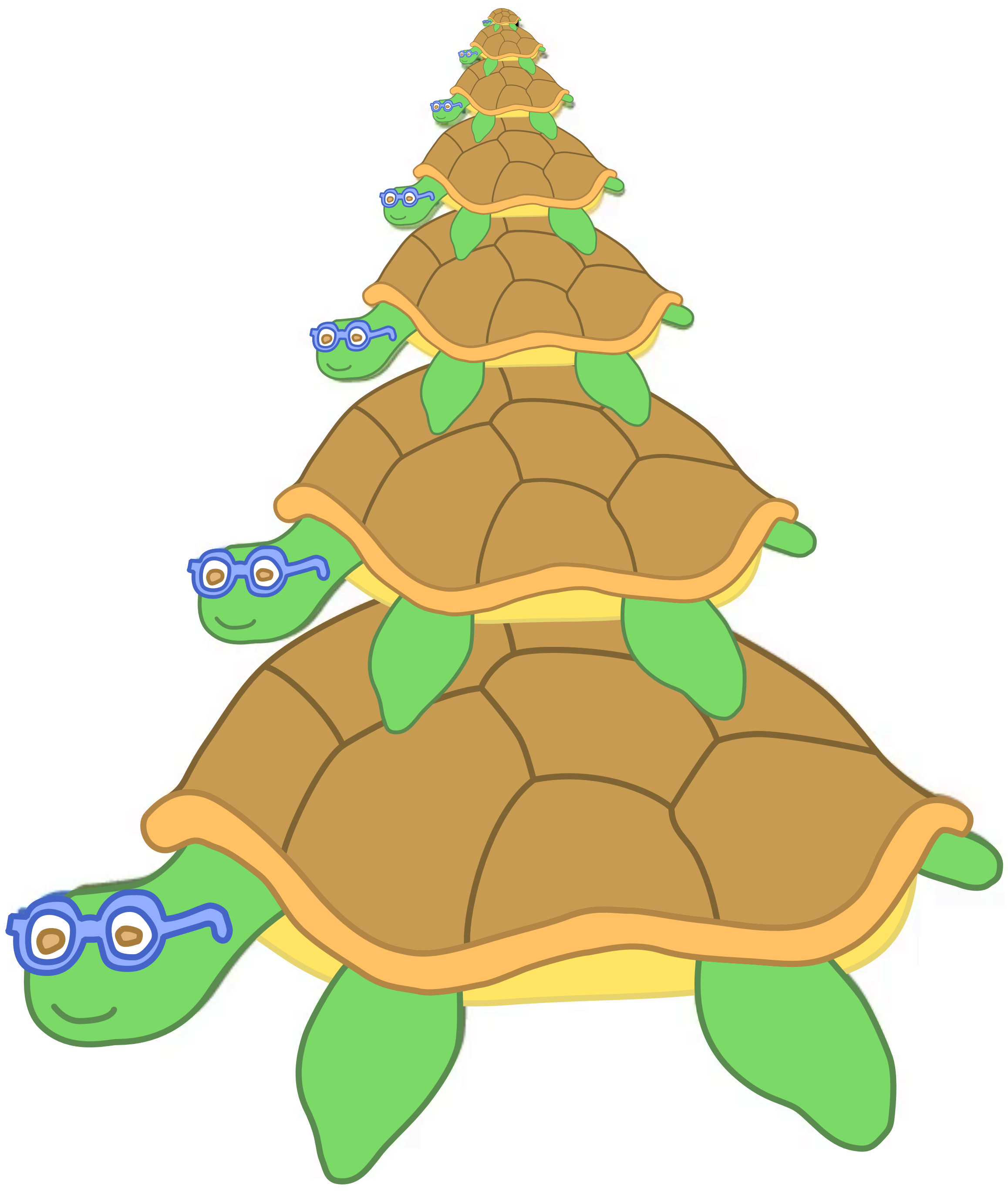




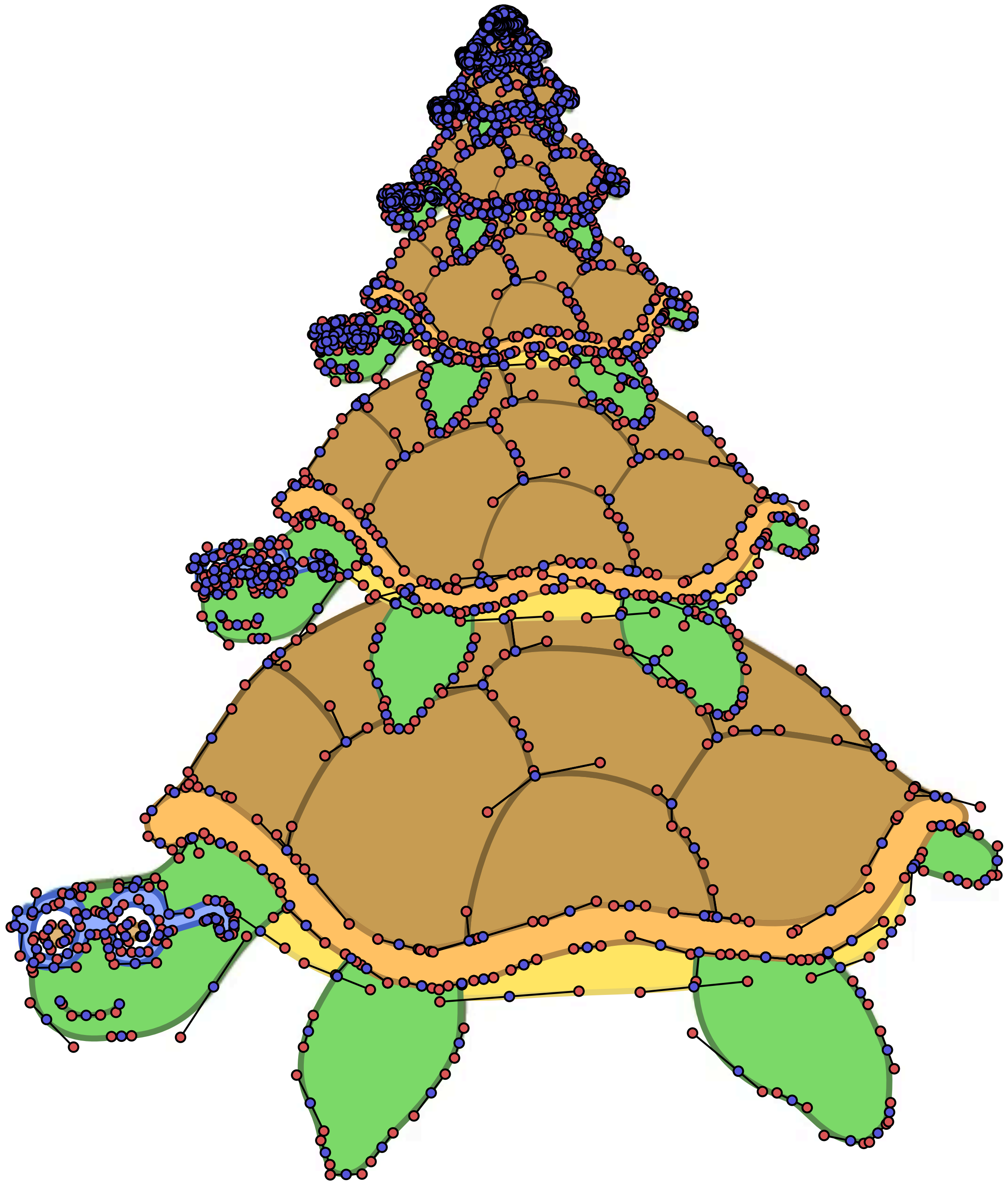
Raster Deformation

Raster Deformation

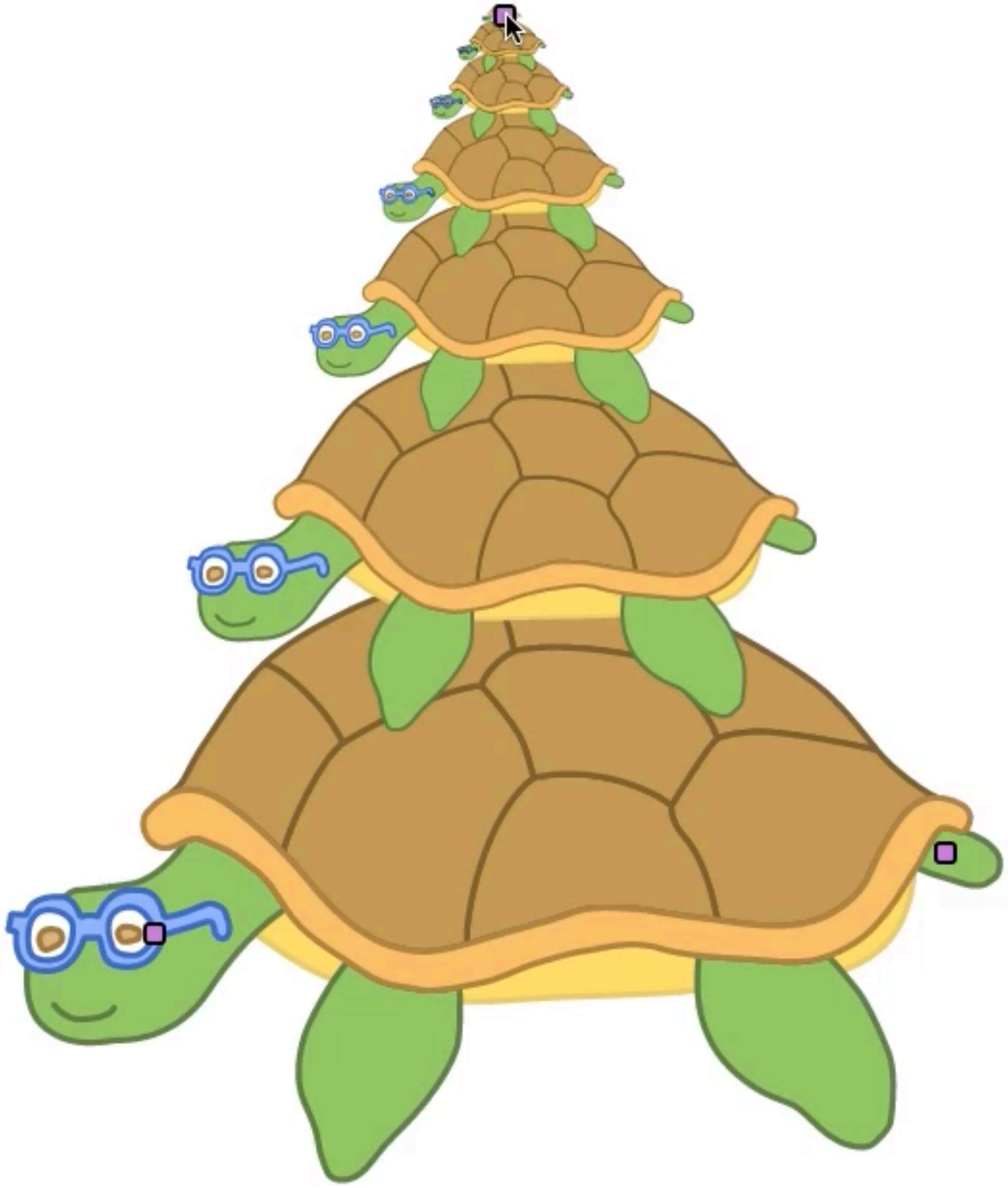
Vector Graphics Deformation



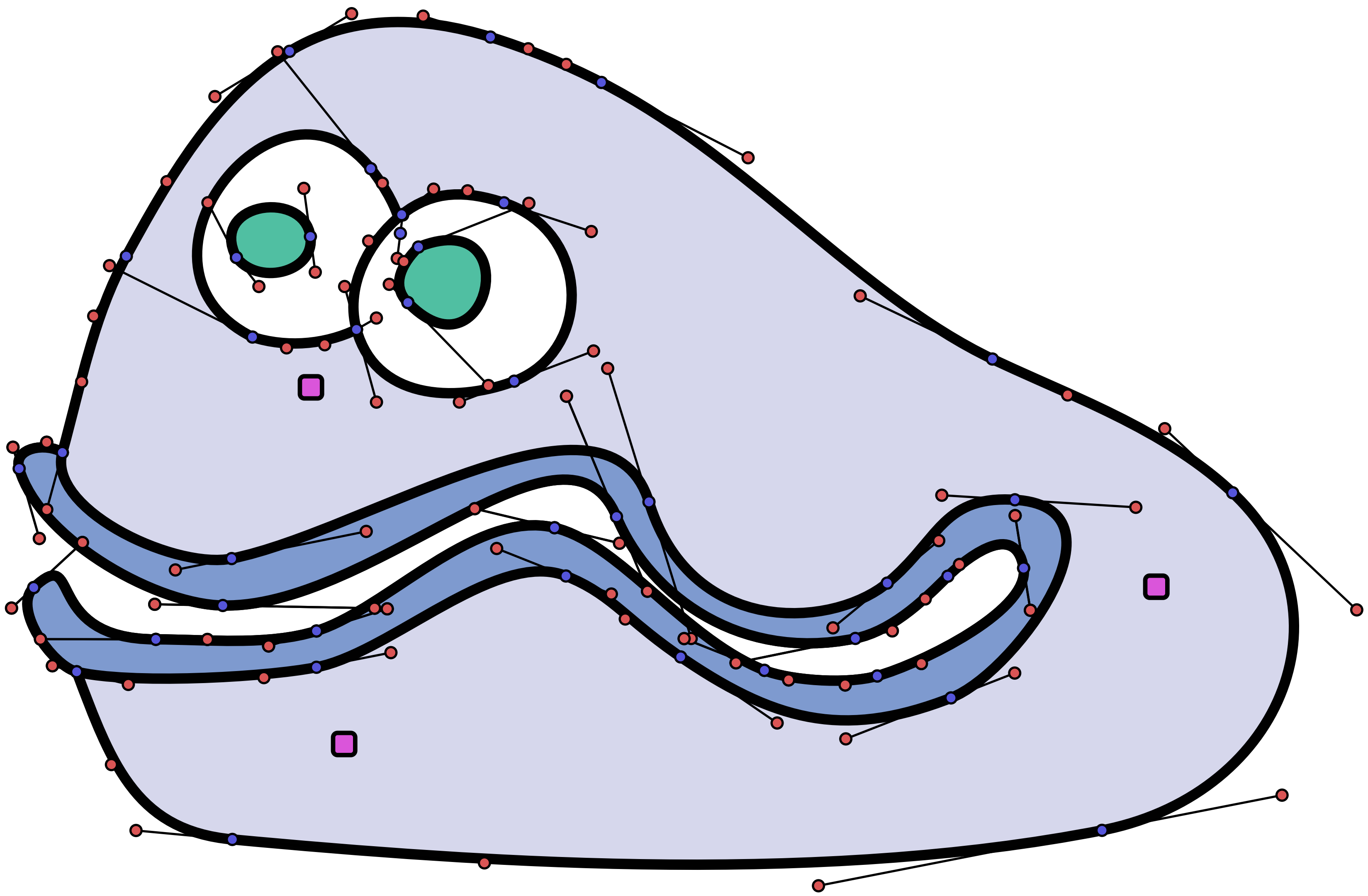
Vector Graphics Deformation



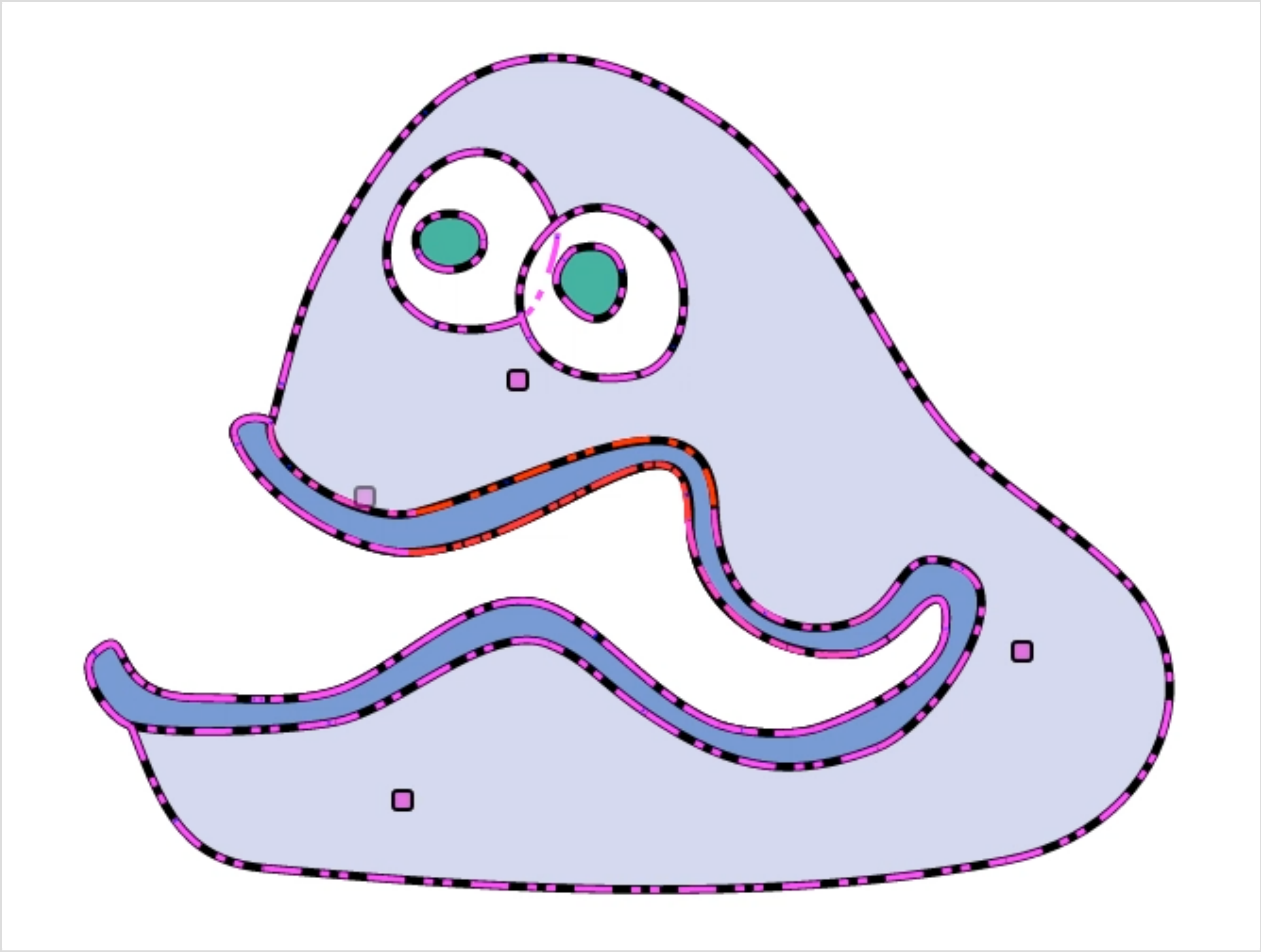
Vector Graphics Deformation



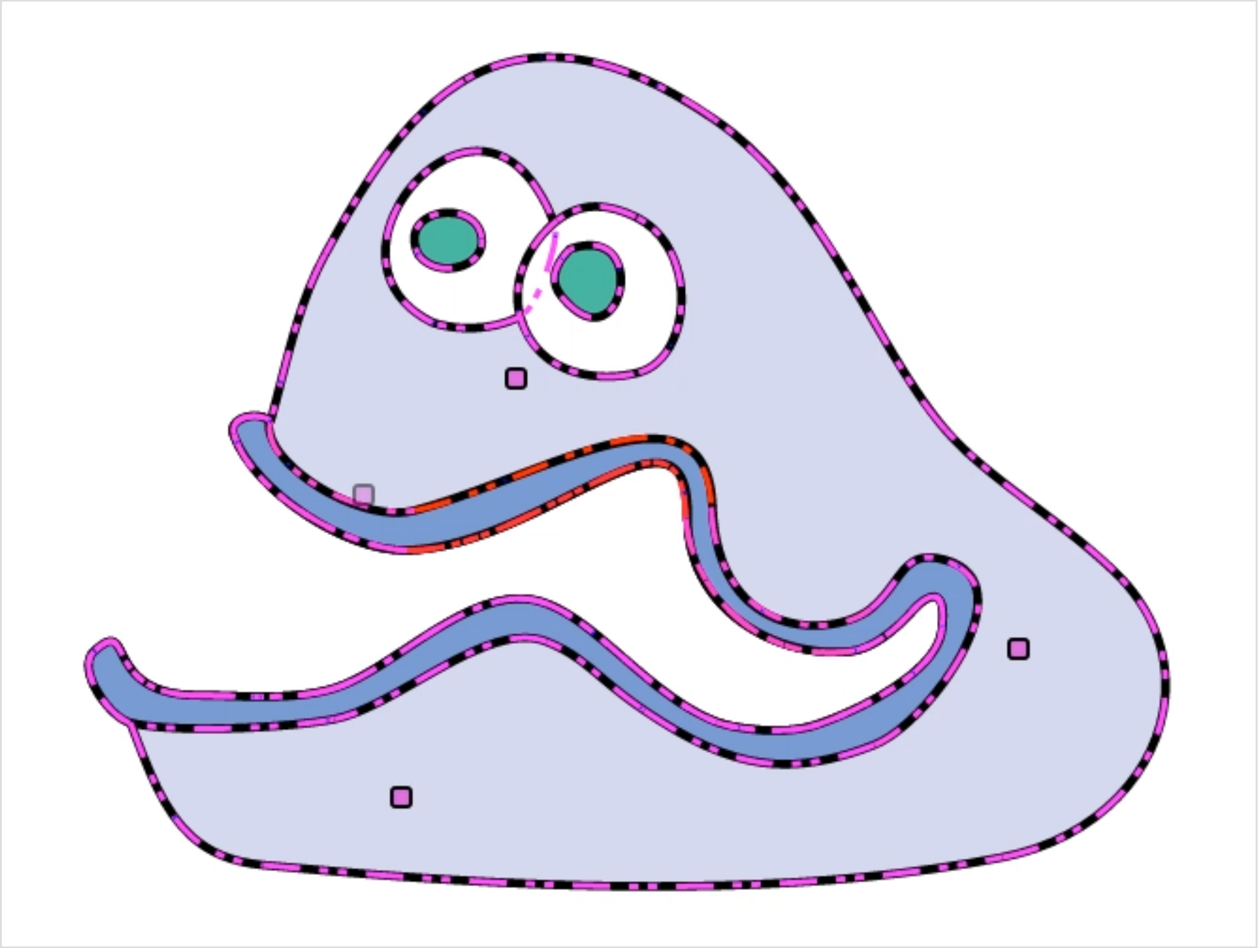
Our Approach



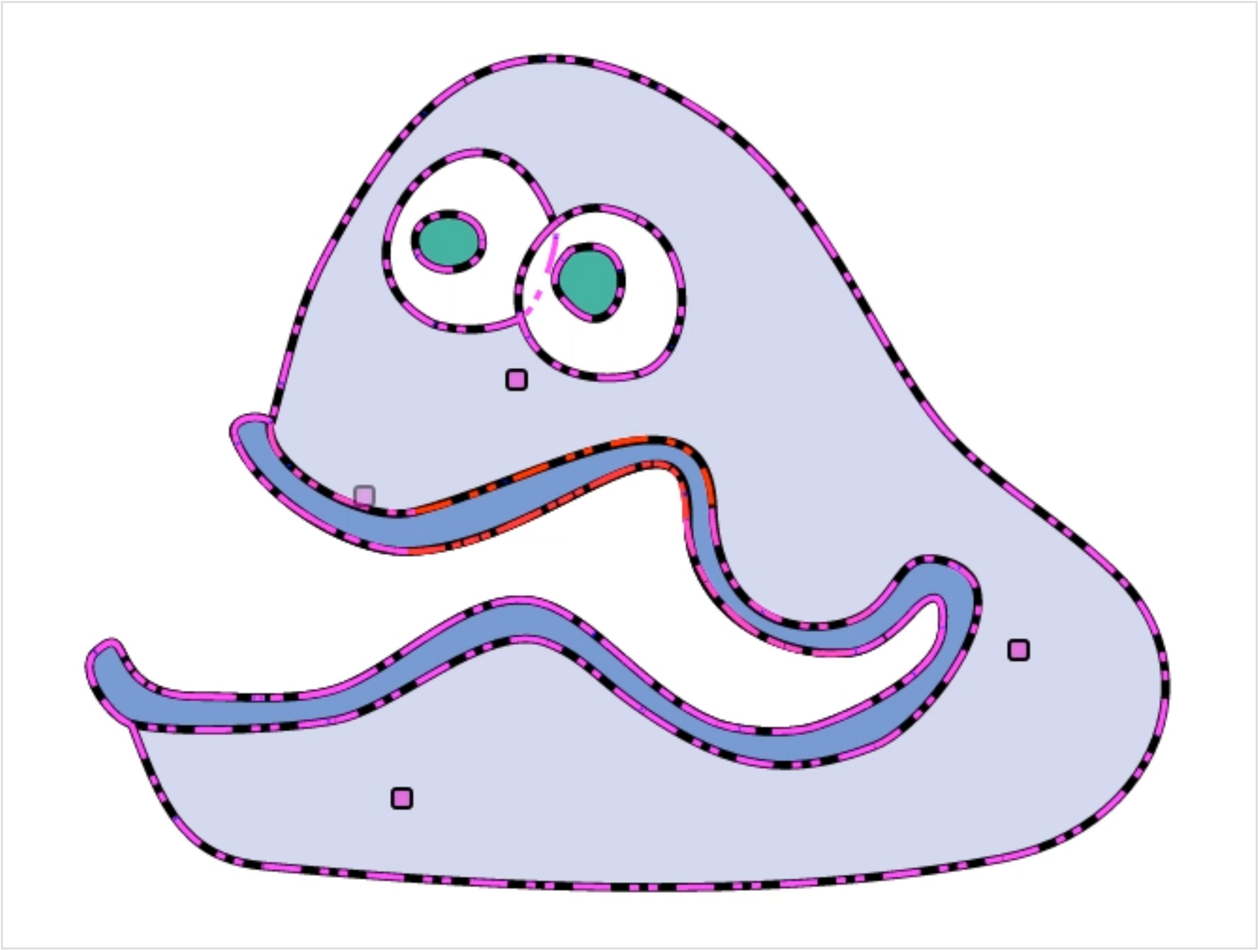
Our Approach

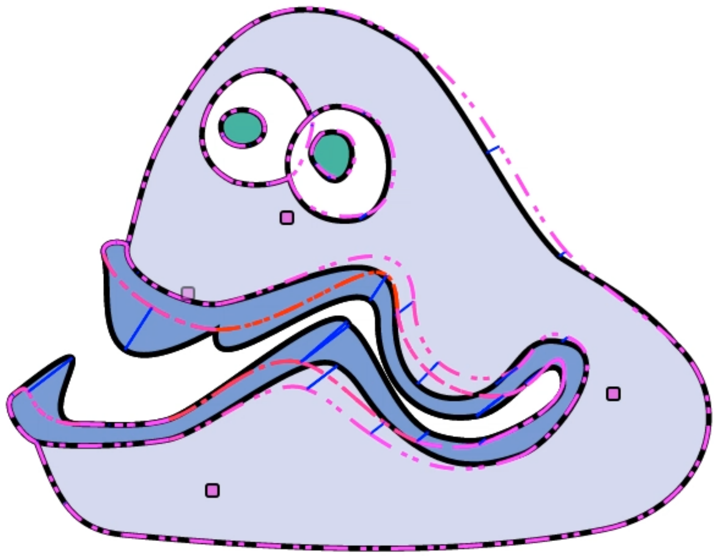


Our Approach



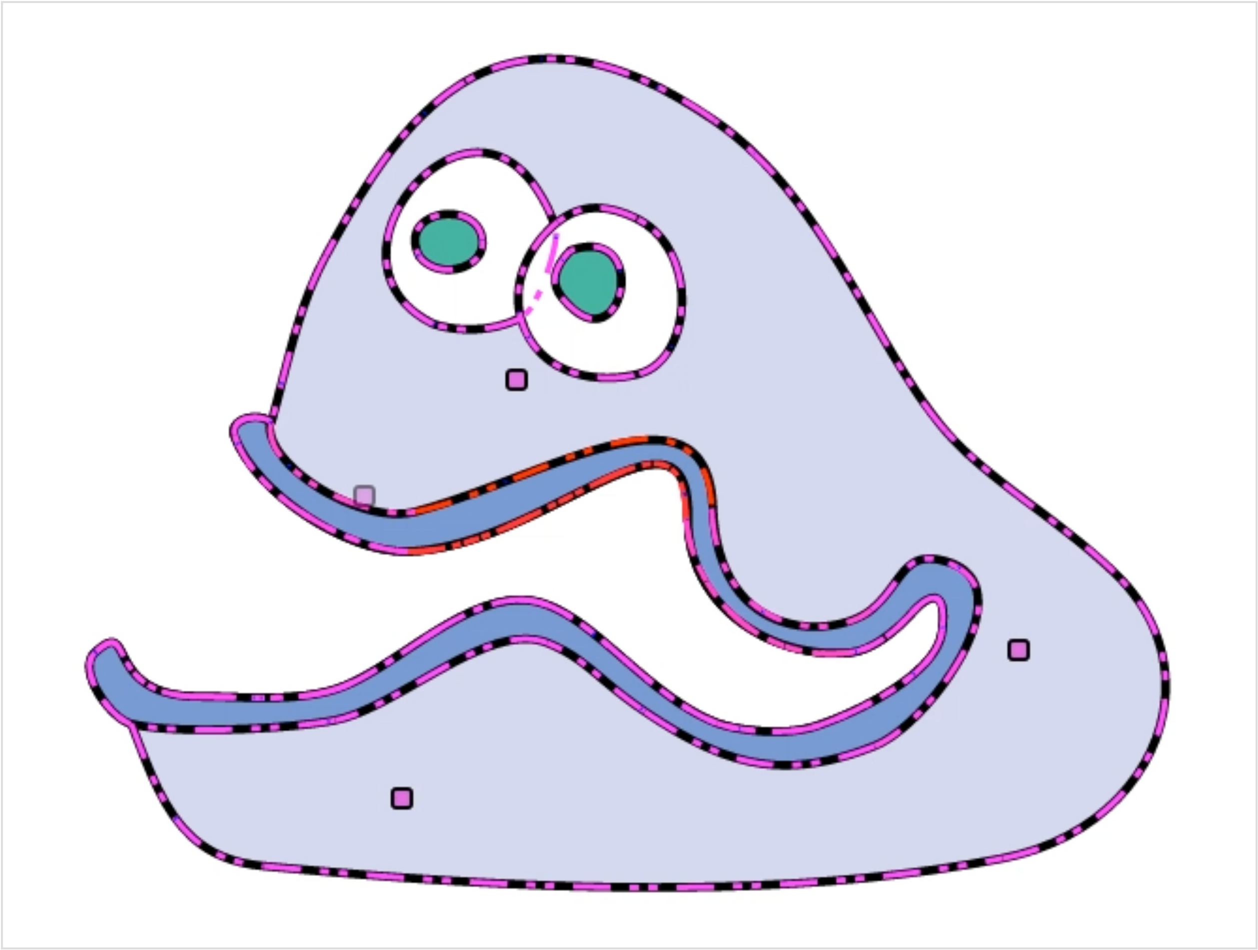
Our Approach

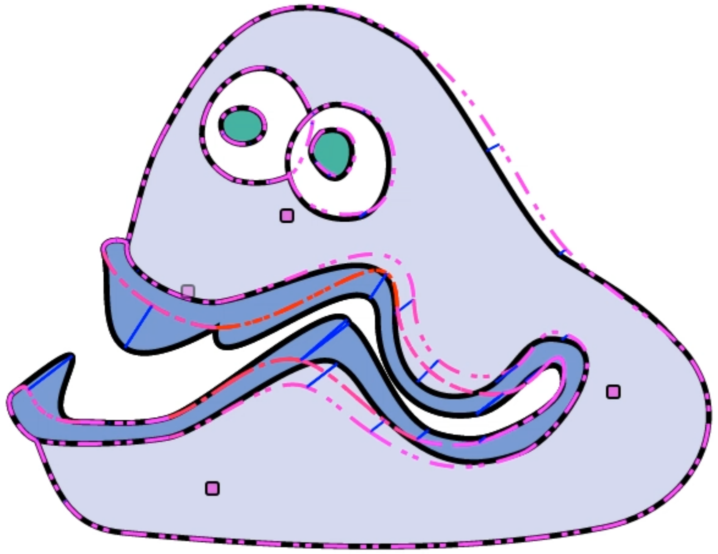




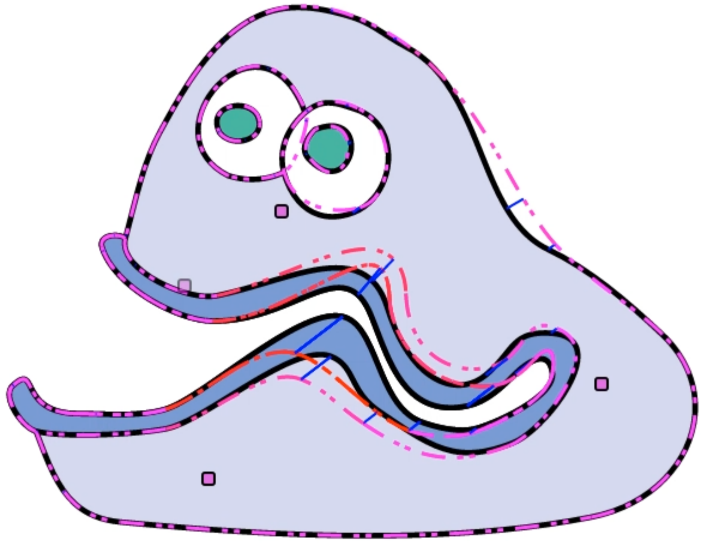
Deforming All
Control Points

Our Approach



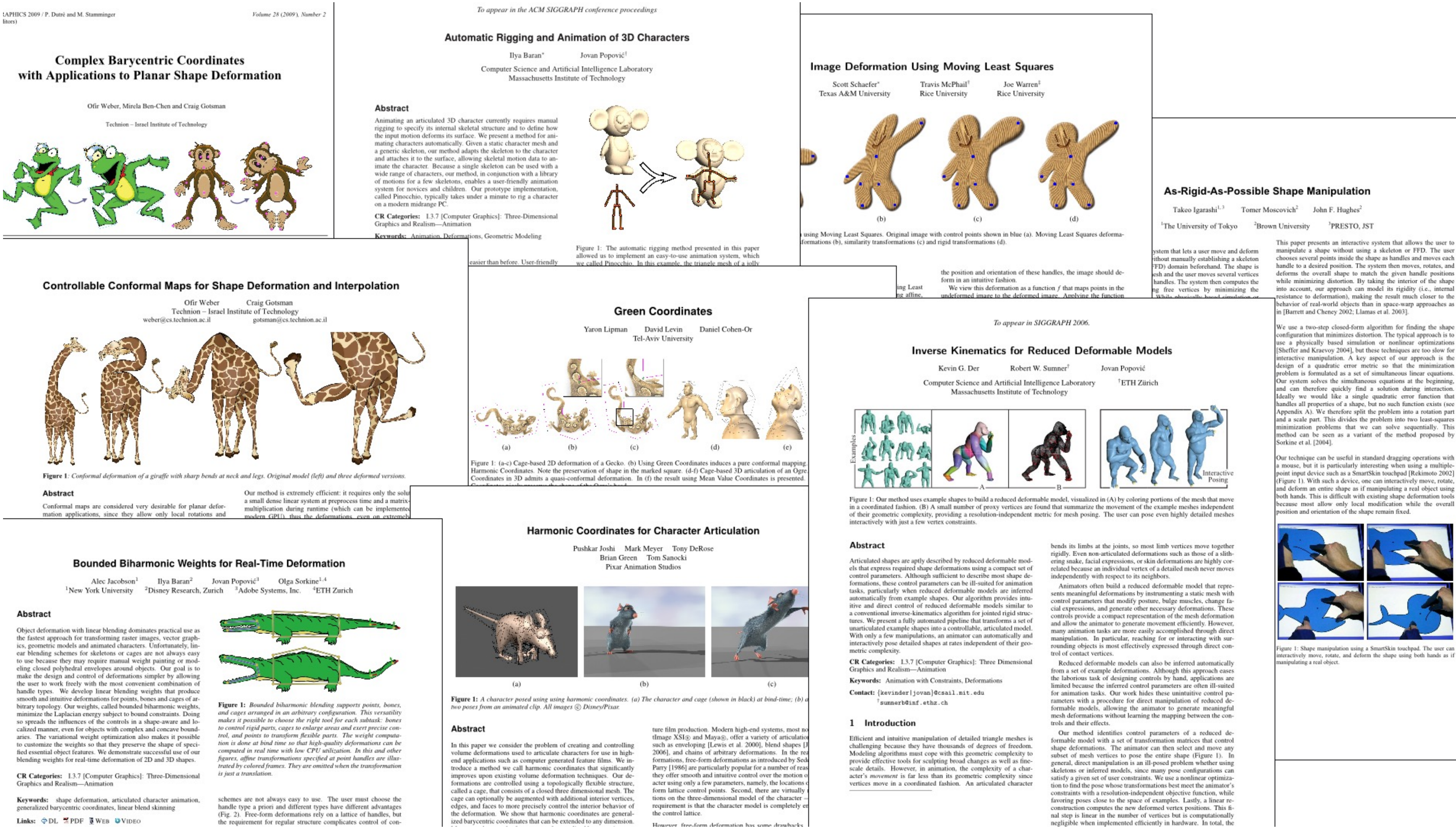


Deforming All
Control Points



Deforming Joint
Control Points

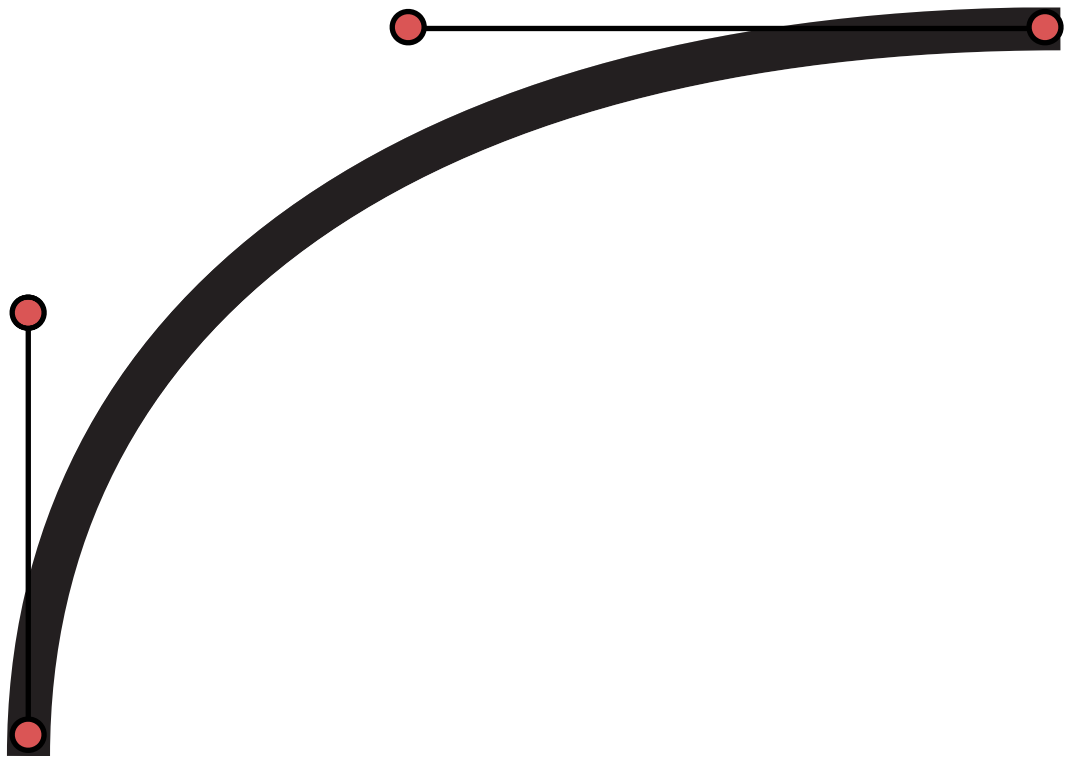
Related Work



2D Vector Graphics

- Splines in 2D — Cubic Bézier splines.

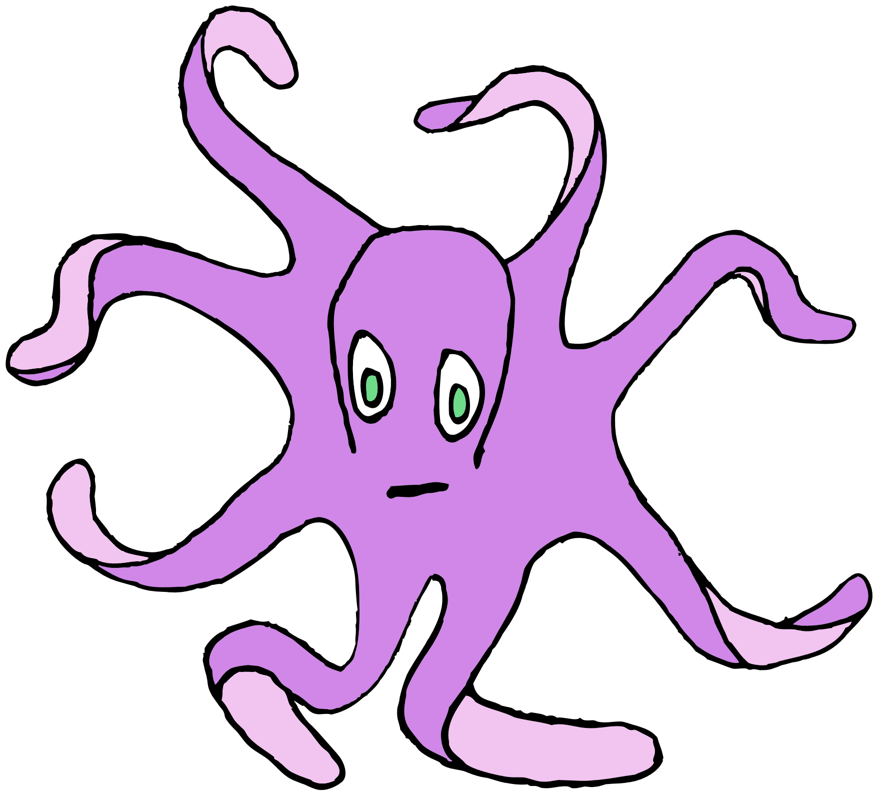
$$B_C(t) = \sum_{i=0}^{n} b_{i,n}(t) C_i, t \in [0,1]$$



2D Vector Graphics

- Splines in 2D — Cubic Bézier splines.

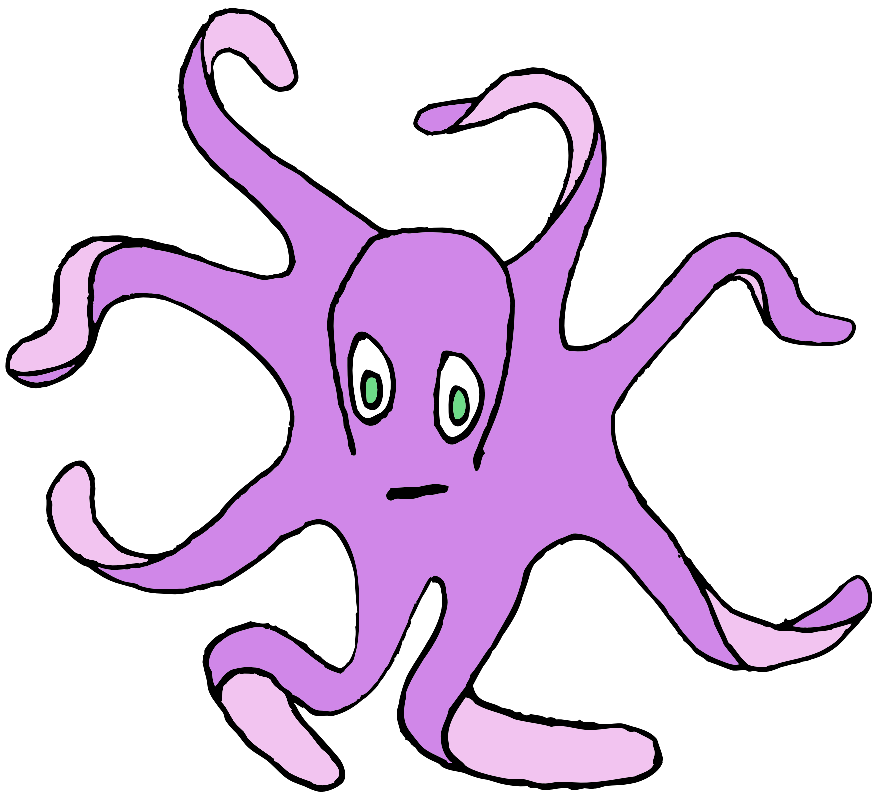
$$B_C(t) = \sum_{i=0}^{n} b_{i,n}(t) C_i, t \in [0, 1]$$



2D Vector Graphics

- Splines in 2D — Cubic Bézier splines.

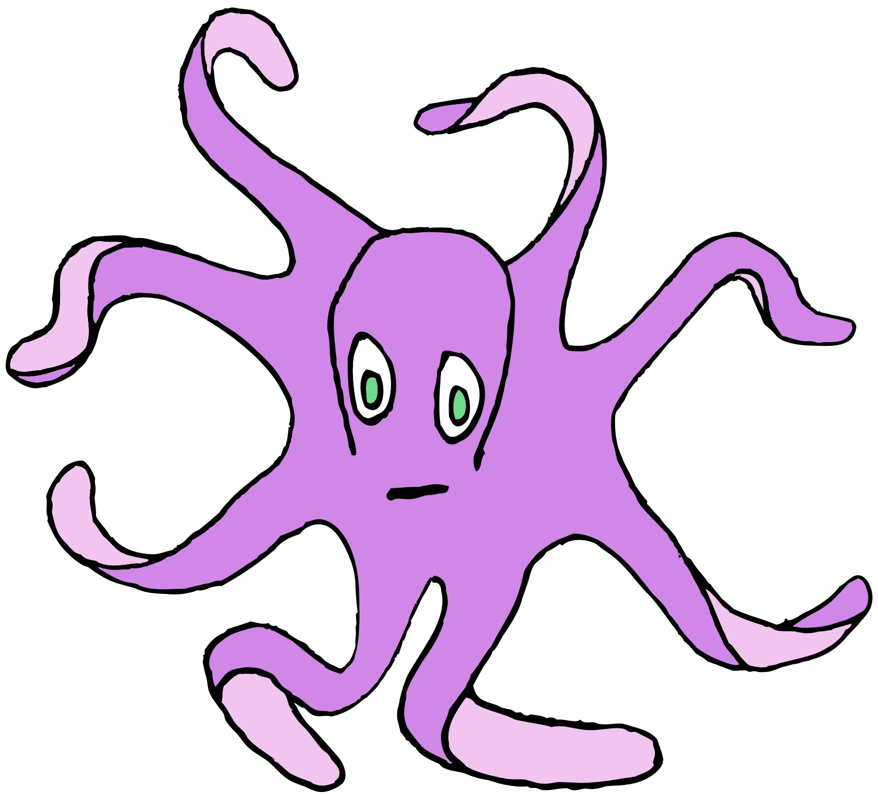
$$B_C(t) = \sum_{i=0}^{n} b_{i,n}(t) C_i, t \in [0,1]$$



2D Vector Graphics

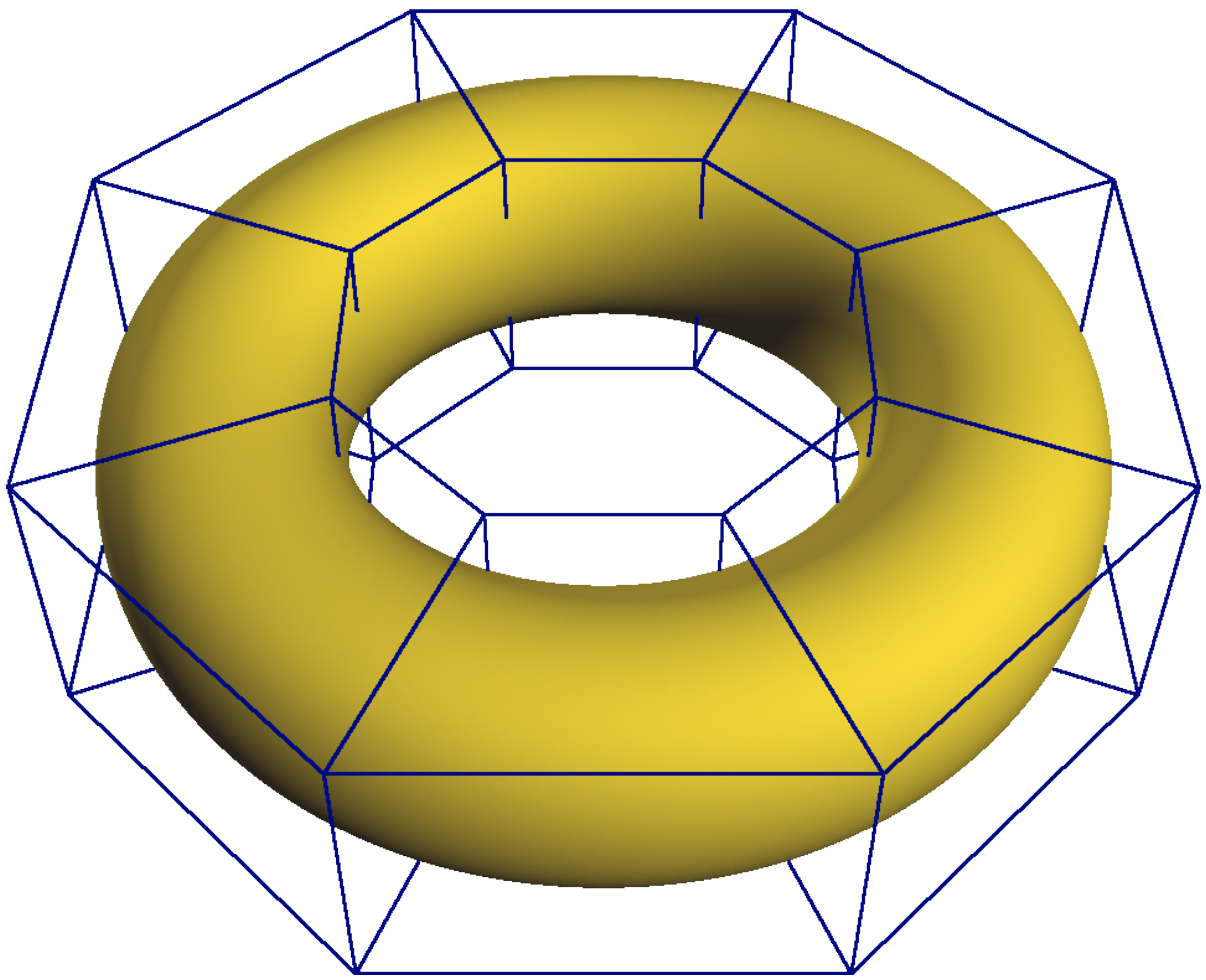
- Splines in 2D — Cubic Bézier splines.

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



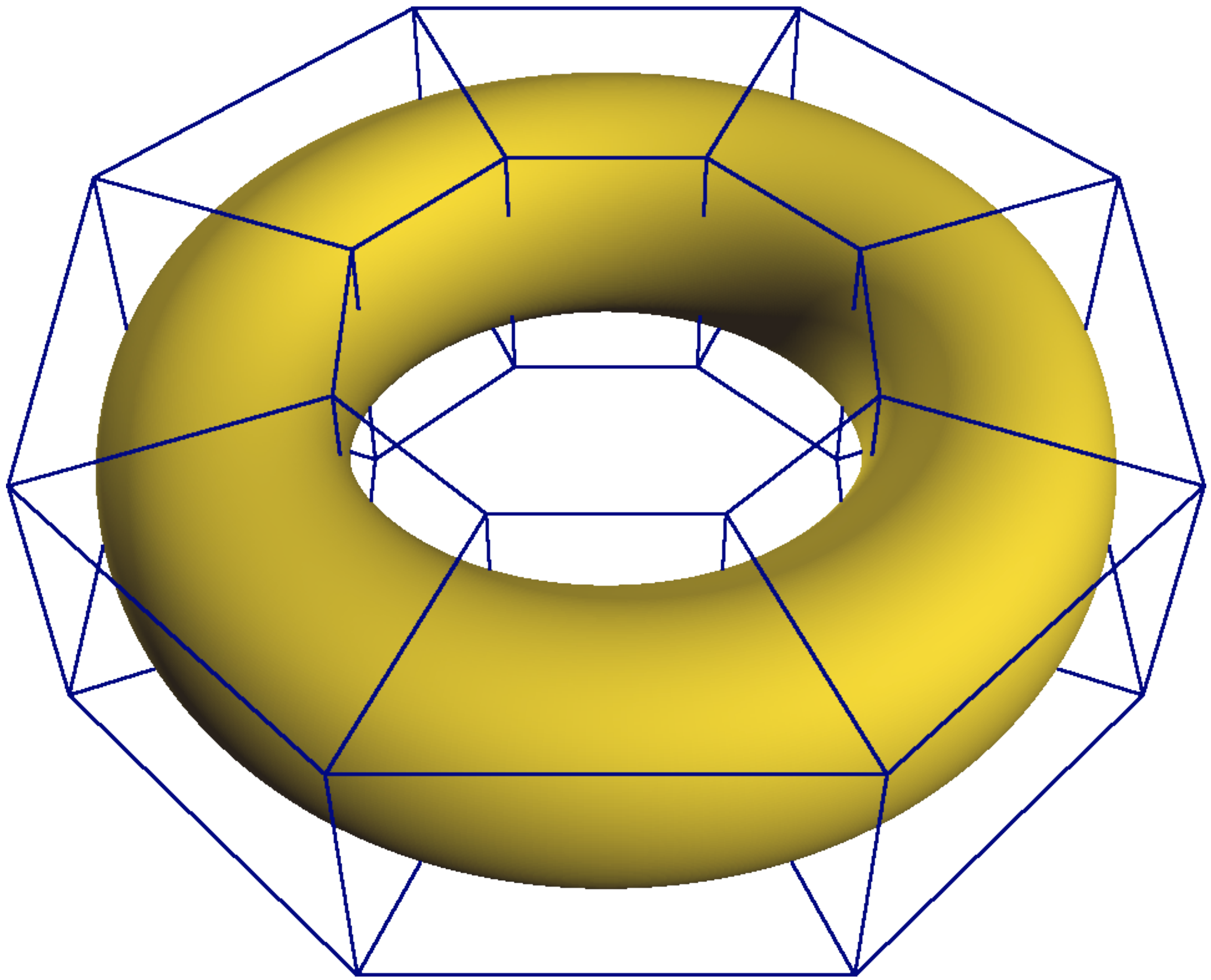
3D Vector Graphics

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



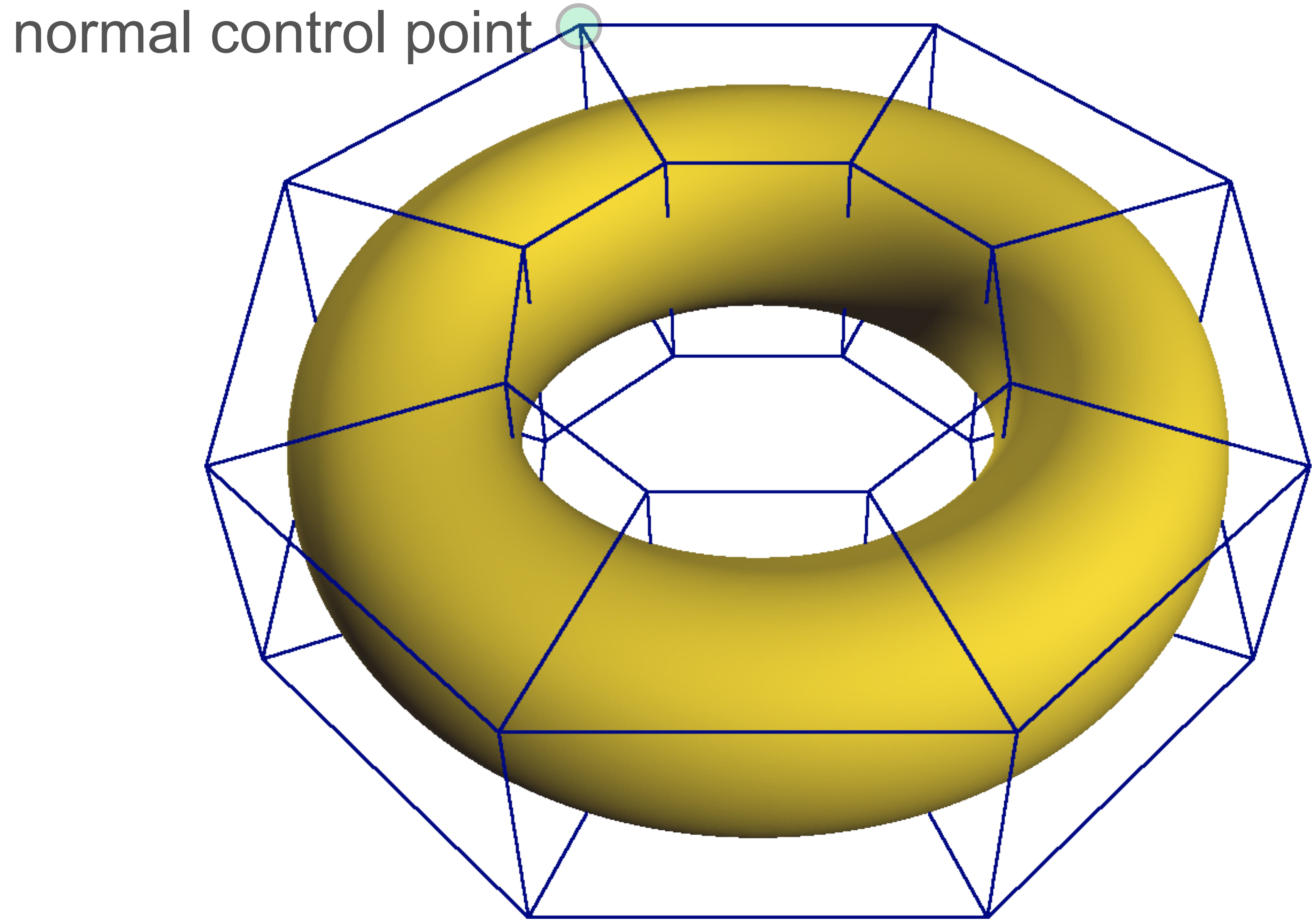
3D Vector Graphics

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



3D Vector Graphics

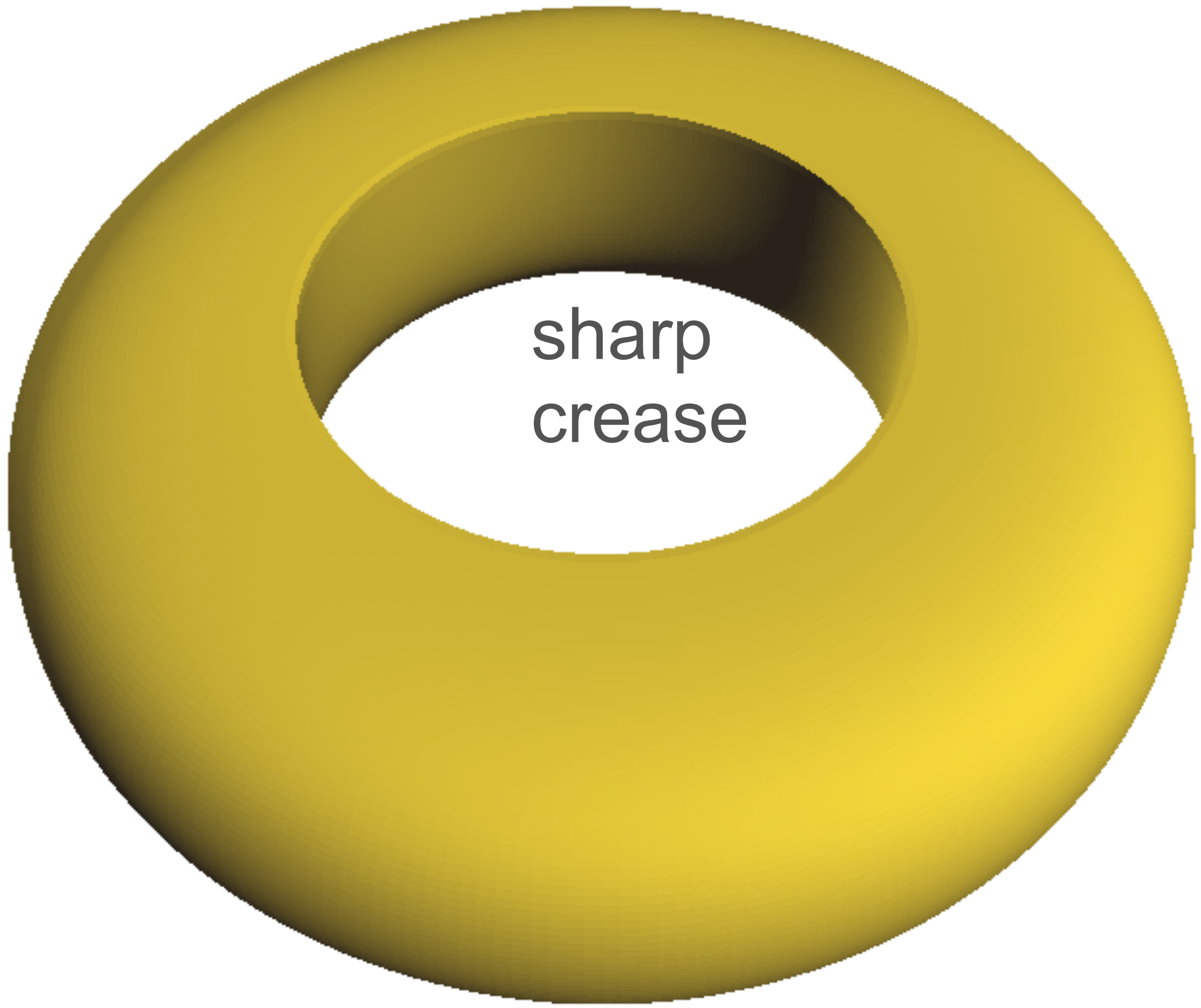
- Subdivision Surfaces in 3D — Catmull-Clark Subdivision Surfaces.



- Inherently C2 everywhere except extraordinary vertices (C1)

3D Vector Graphics

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

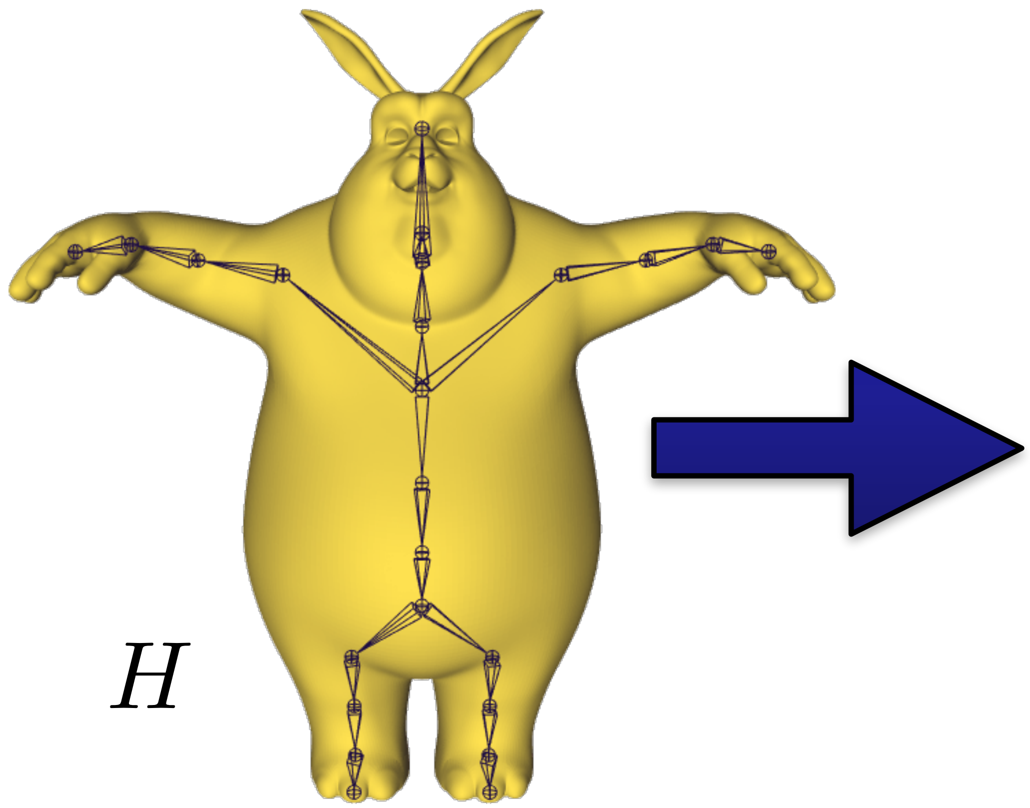


- Inherently C2 everywhere except extraordinary vertices (C1)
- Sharp creases can also be specified

Linear Blend Skinning(LBS)

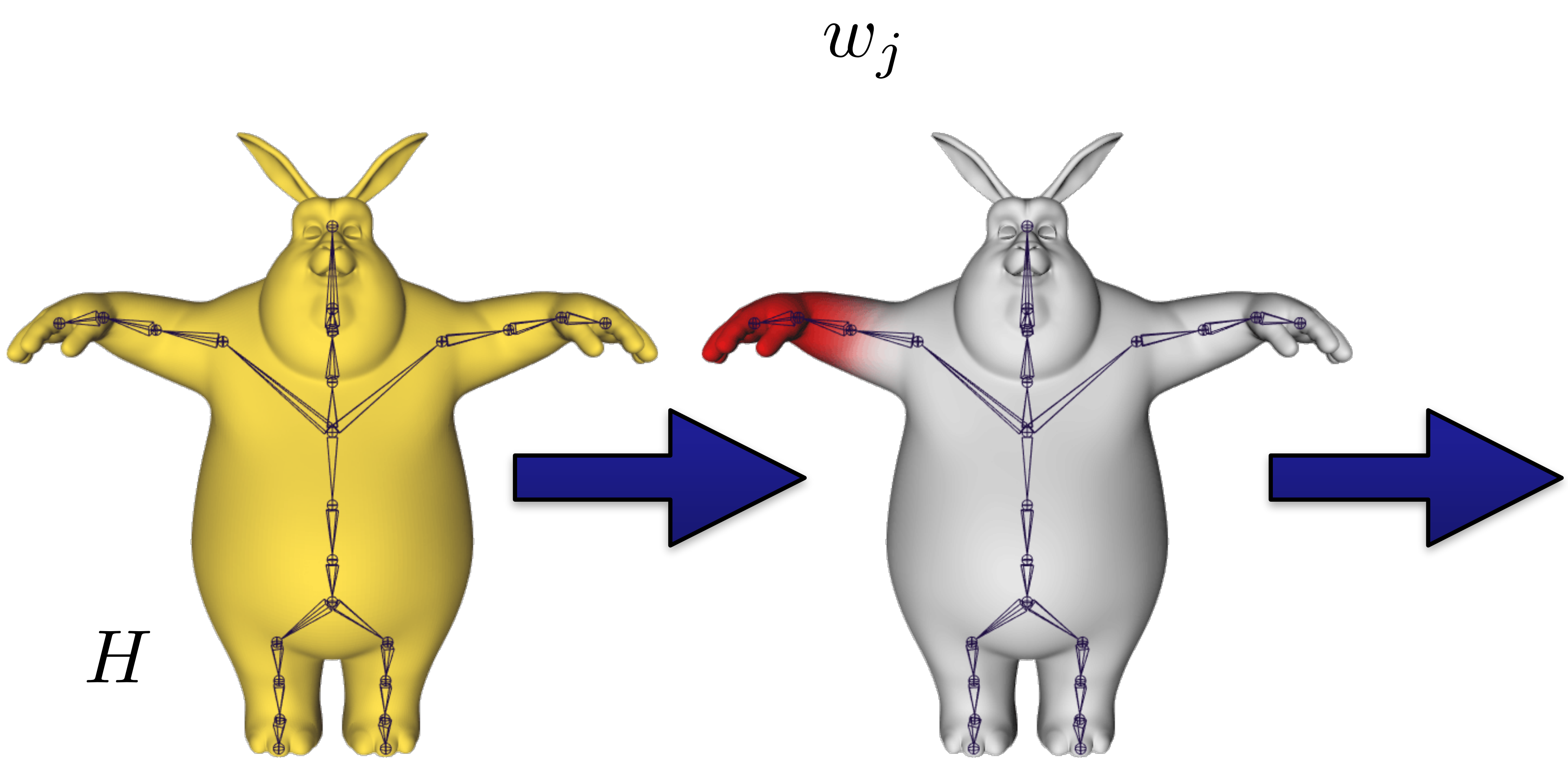
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

Linear Blend Skinning(LBS)



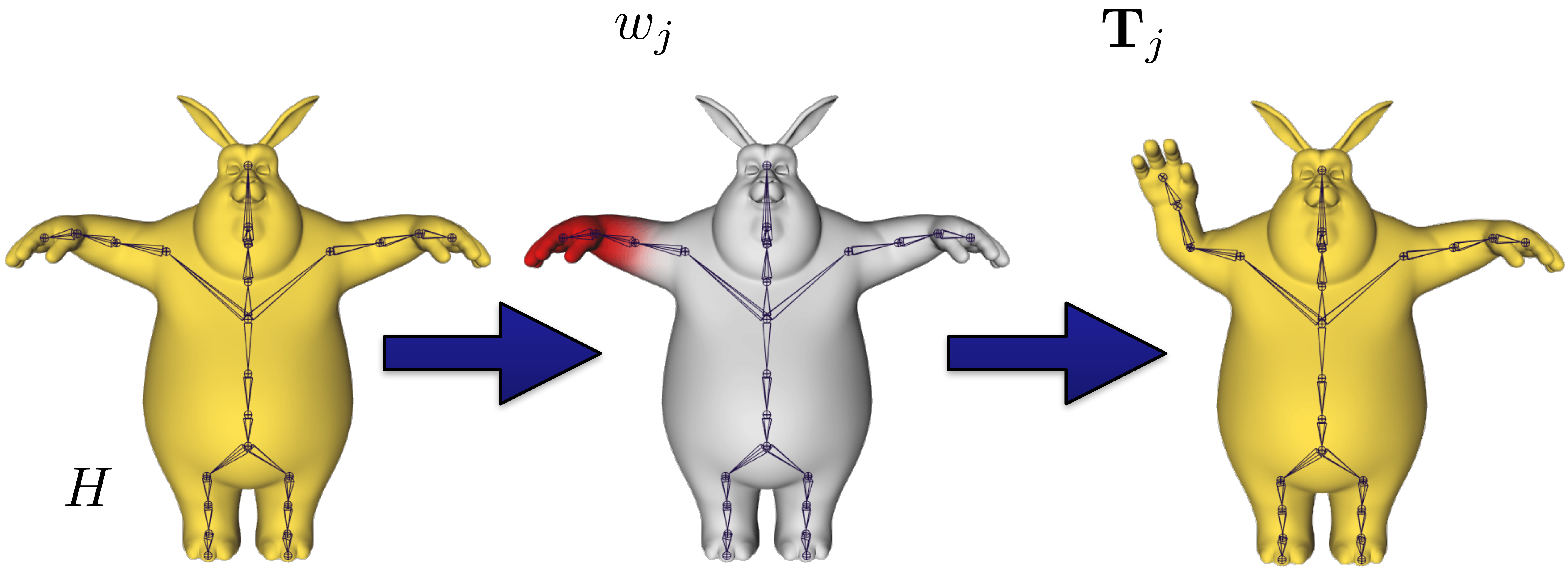
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

Linear Blend Skinning(LBS)



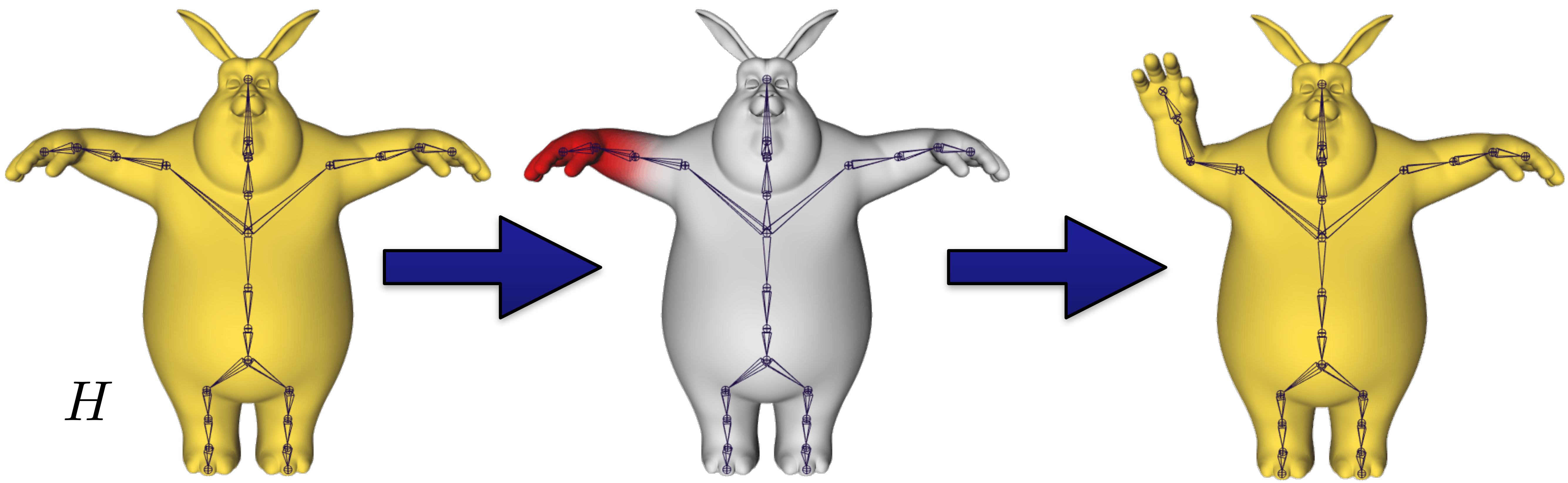
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

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)



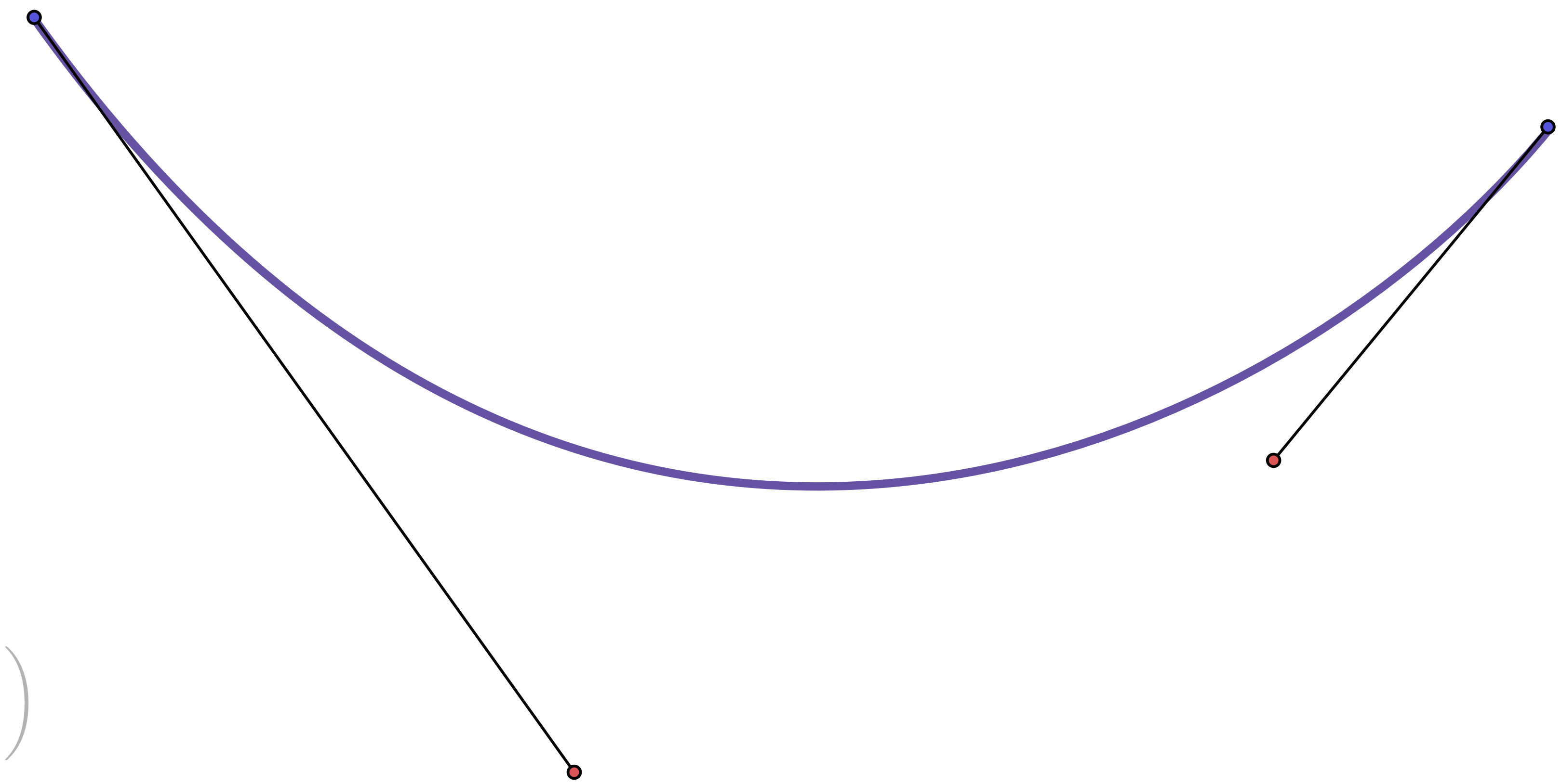
$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

Linear Blend Skinning(LBS)

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

Difficulty Skinning Cubic Bézier Curve

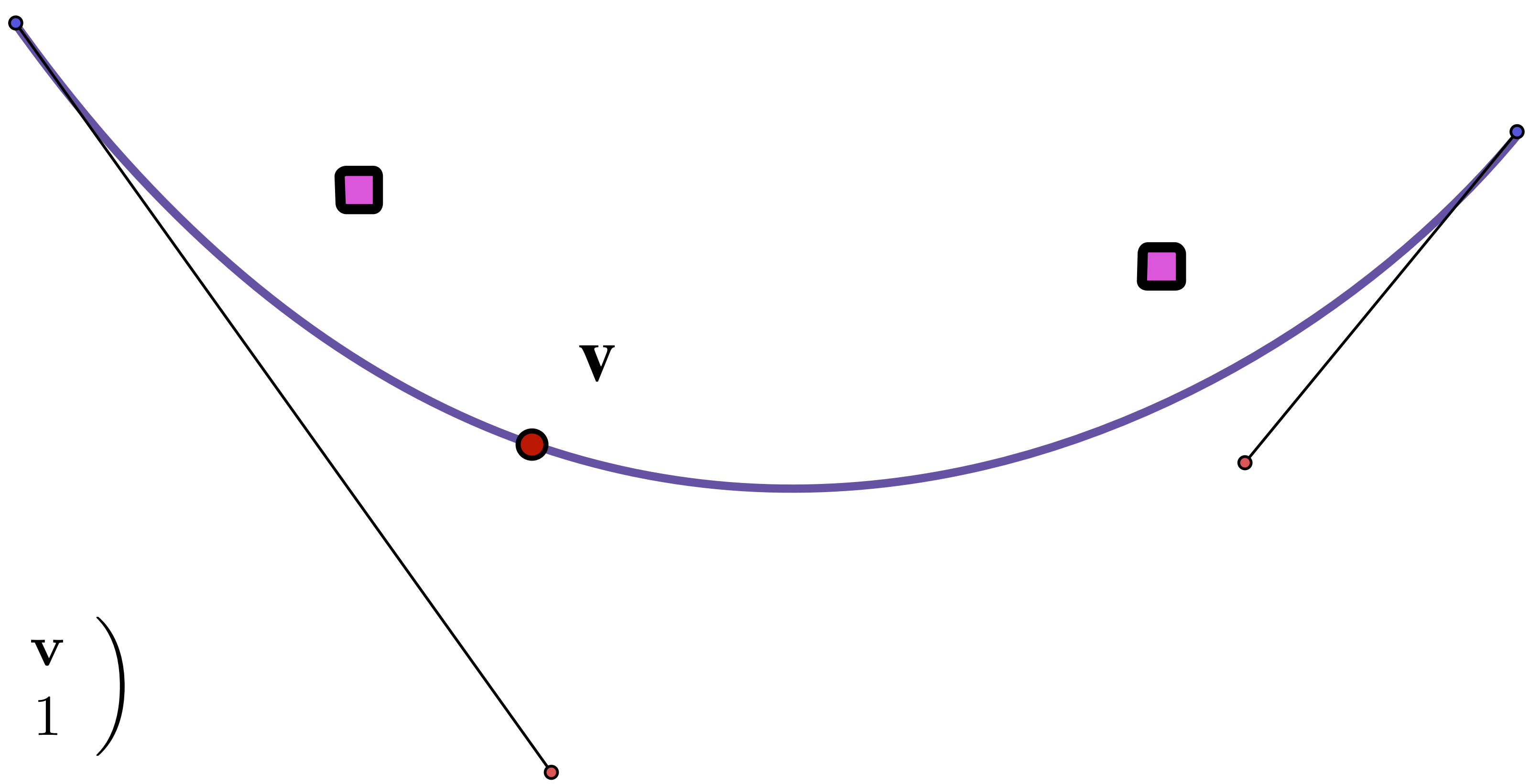
Apply skinning to $v = B_C(t)$



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

Difficulty in Deforming A Cubic Bézier Curve

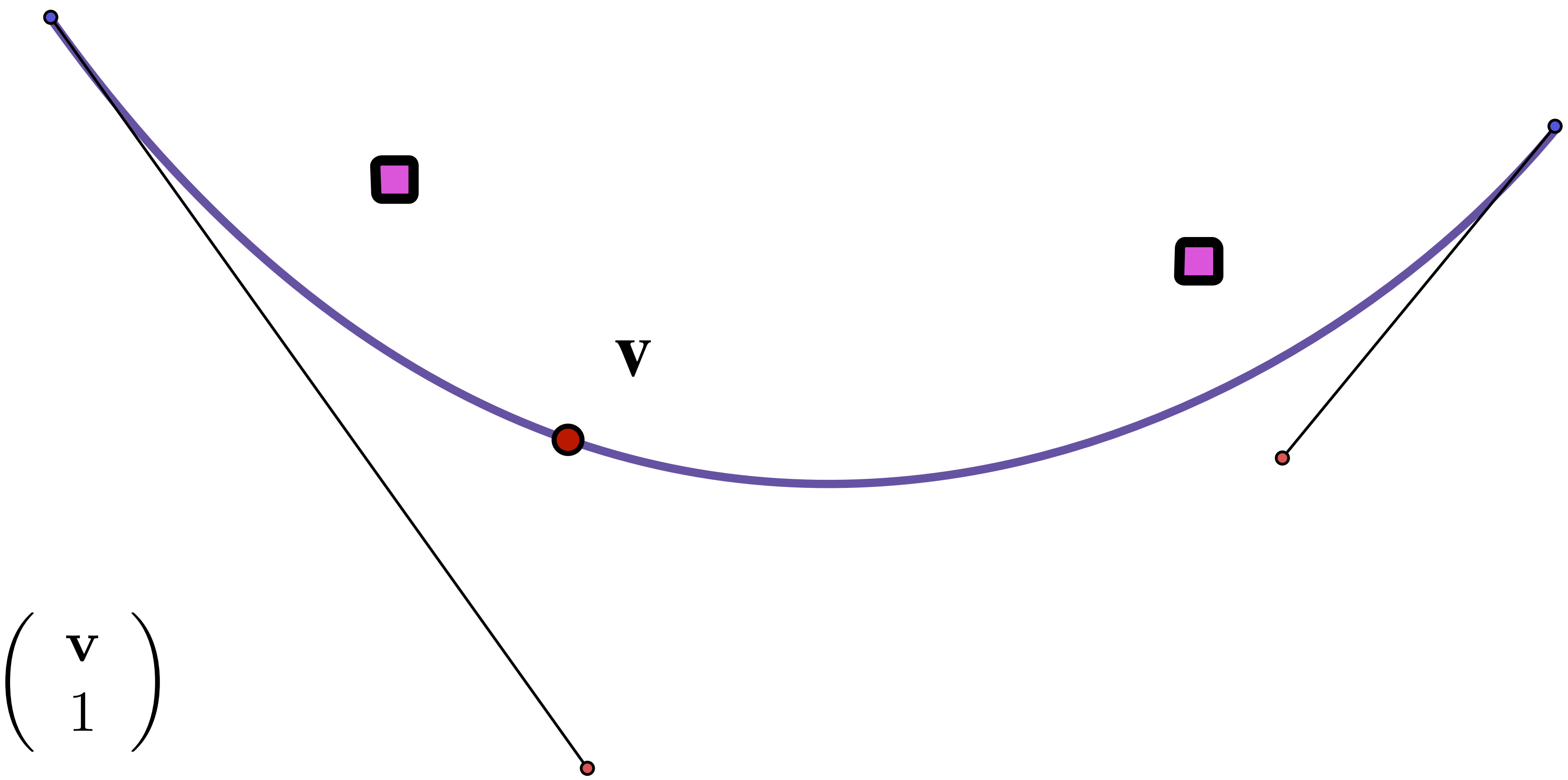
Apply skinning to $v = B_C(t)$



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

Difficulty in Deforming A Cubic Bézier Curve

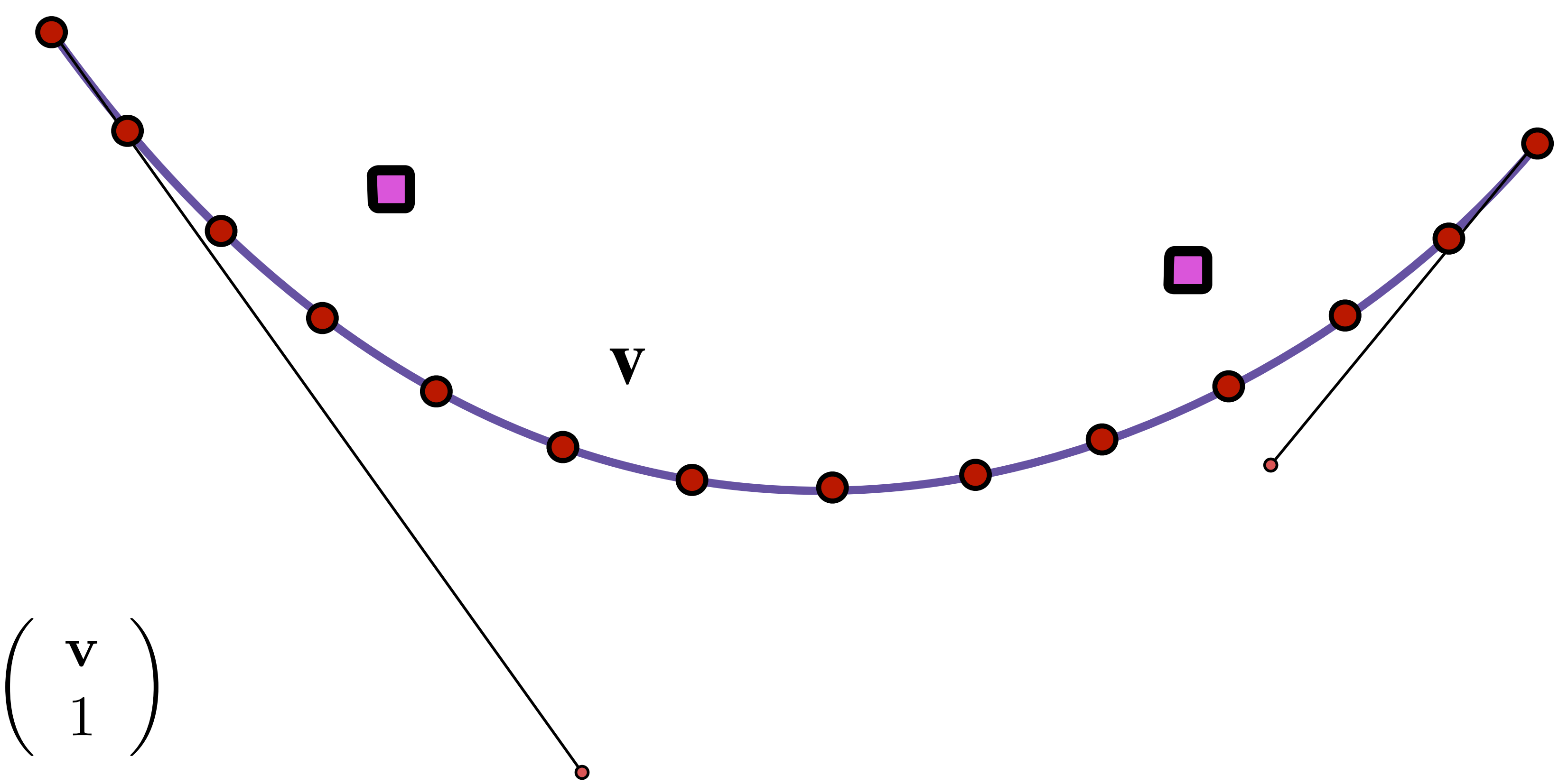
Apply skinning to $v = B_C(t)$



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

Difficulty in Deforming A Cubic Bézier Curve

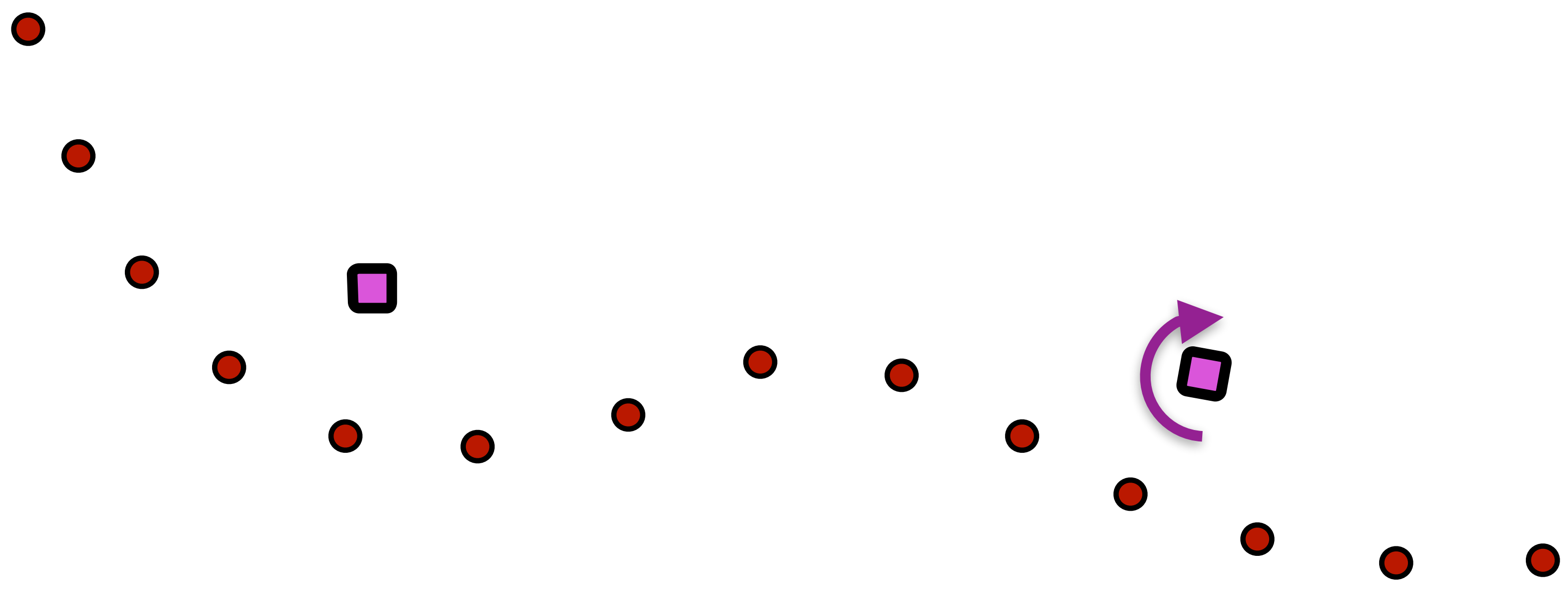
Apply skinning to $v = B_C(t)$



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

Difficulty in Deforming A Cubic Bézier Curve

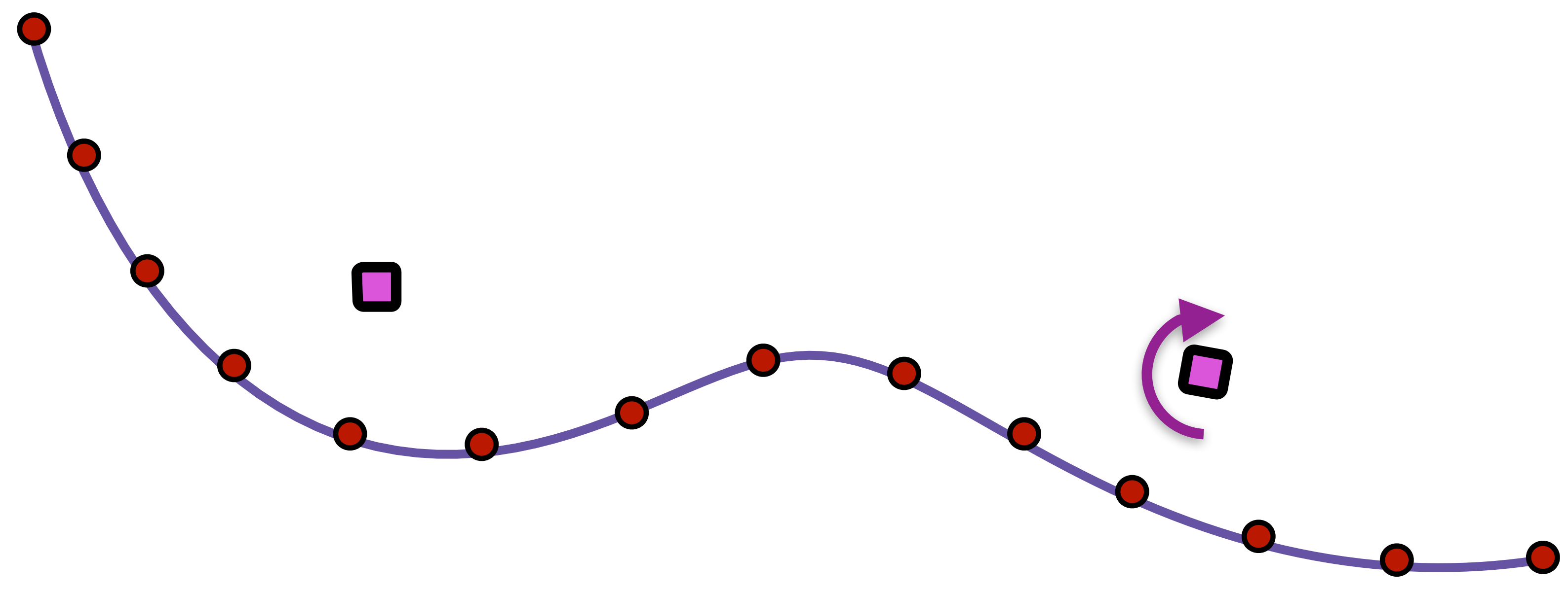
Apply skinning to $v = B_C(t)$



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

Difficulty in Deforming A Cubic Bézier Curve

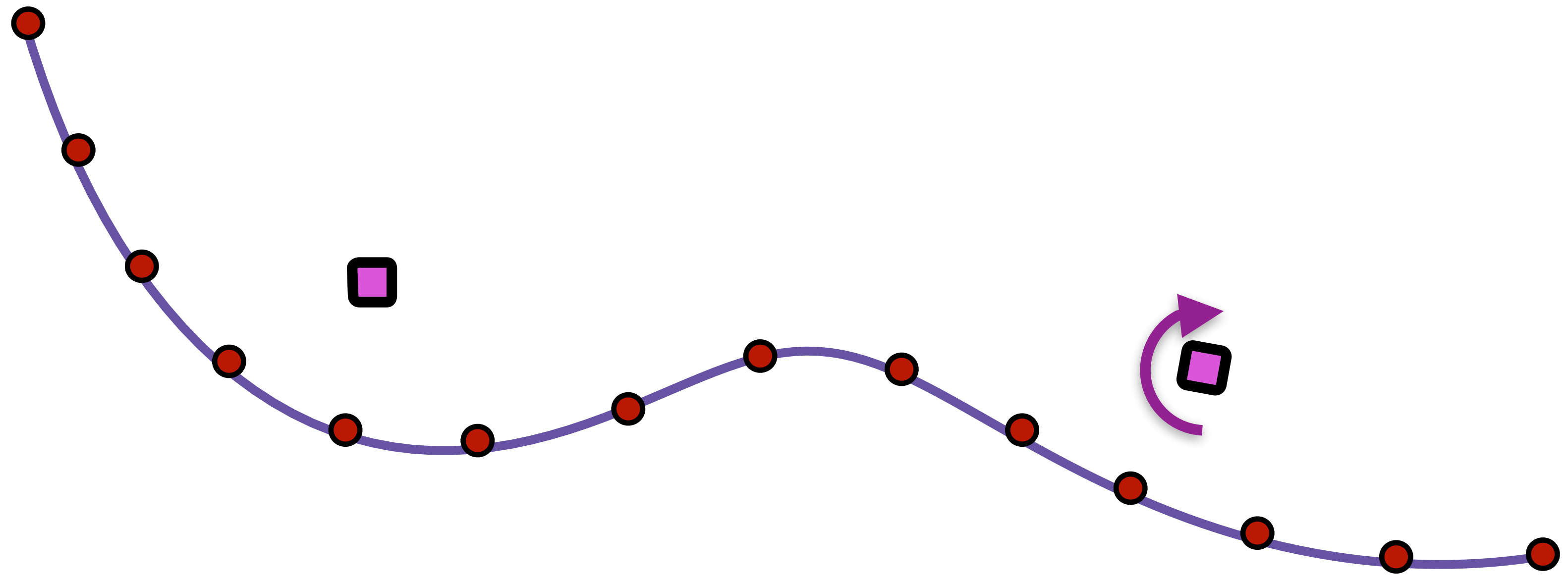
Apply skinning to $v = B_C(t)$



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

Difficulty in Deforming A Cubic Bézier Curve

Apply skinning to $v = B_C(t)$

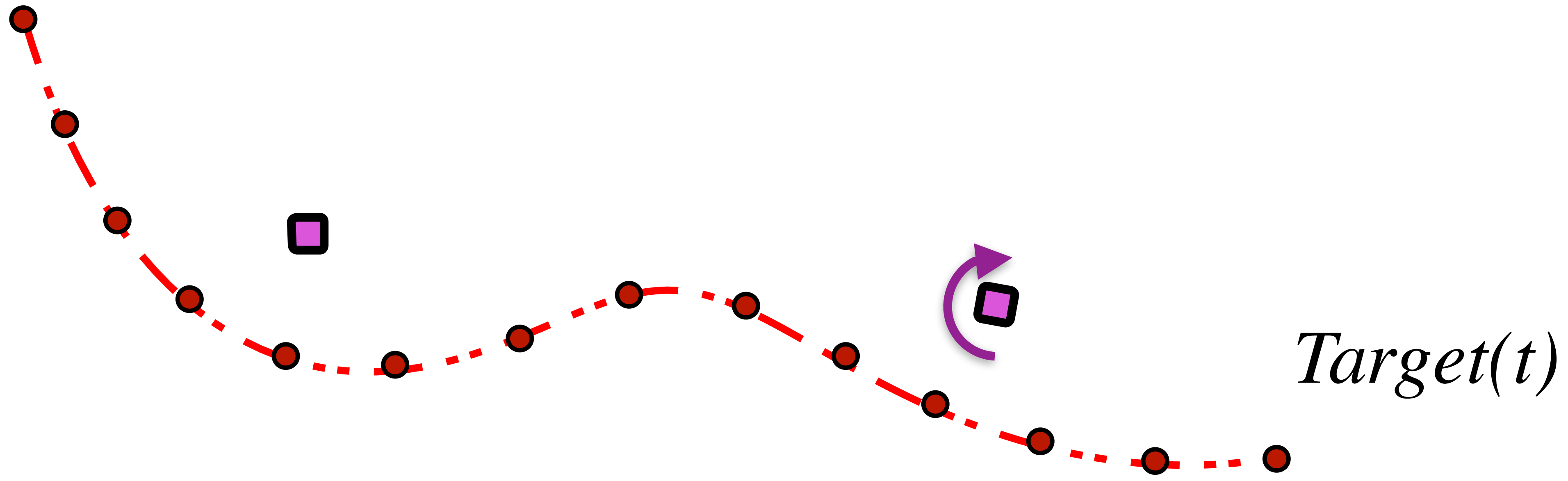


$$\mathbf{v}' = \sum_{j \in H} w_j(\mathbf{v}) \mathbf{T}_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}$$

No longer a cubic Bézier curve!

Difficulty in Deforming A Cubic Bézier Curve

Apply skinning to $v = B_C(t)$



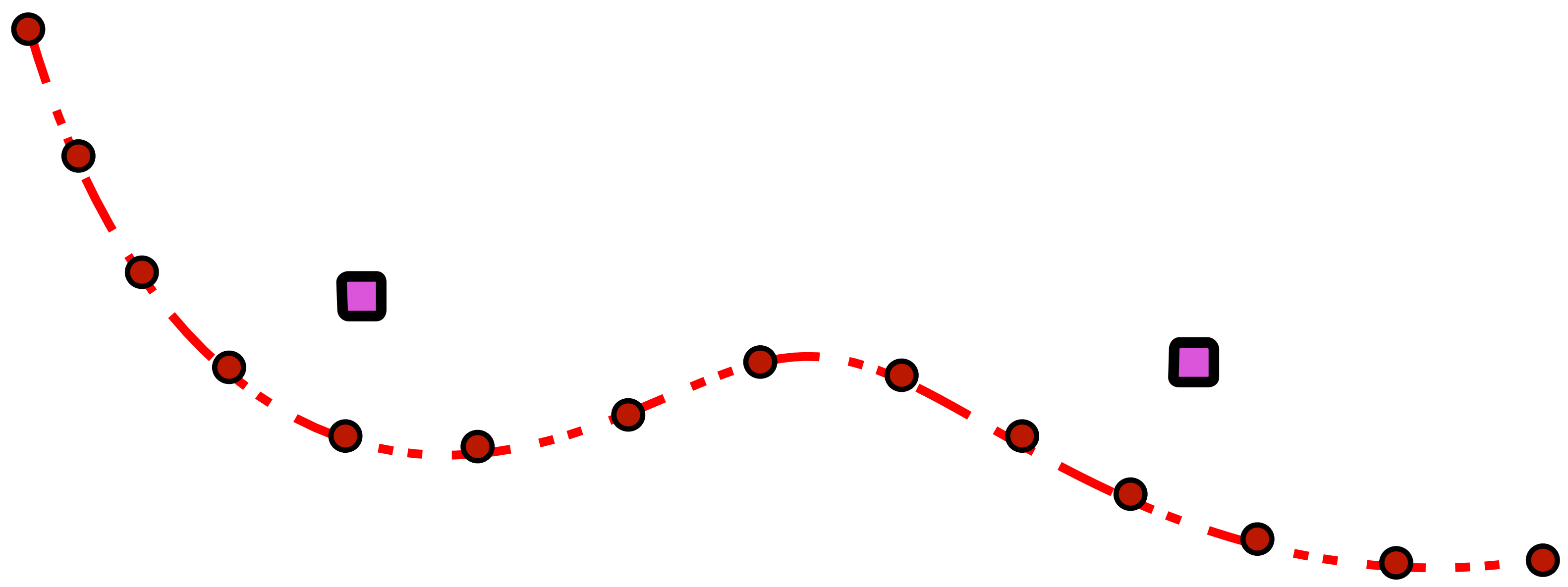
No longer a cubic Bézier curve!

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

How it works

- Minimizing the L2 norm

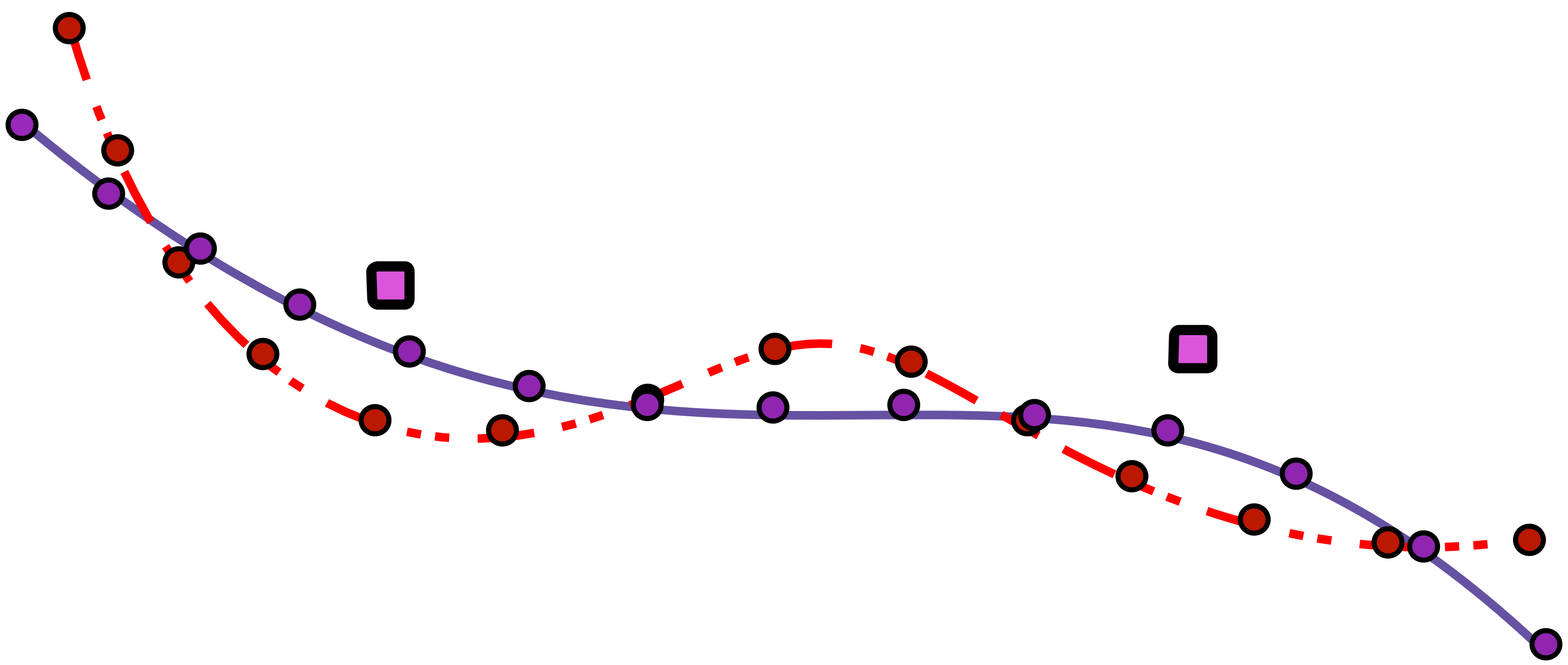
$$Target(t)$$



How it works

- Minimizing the L2 norm

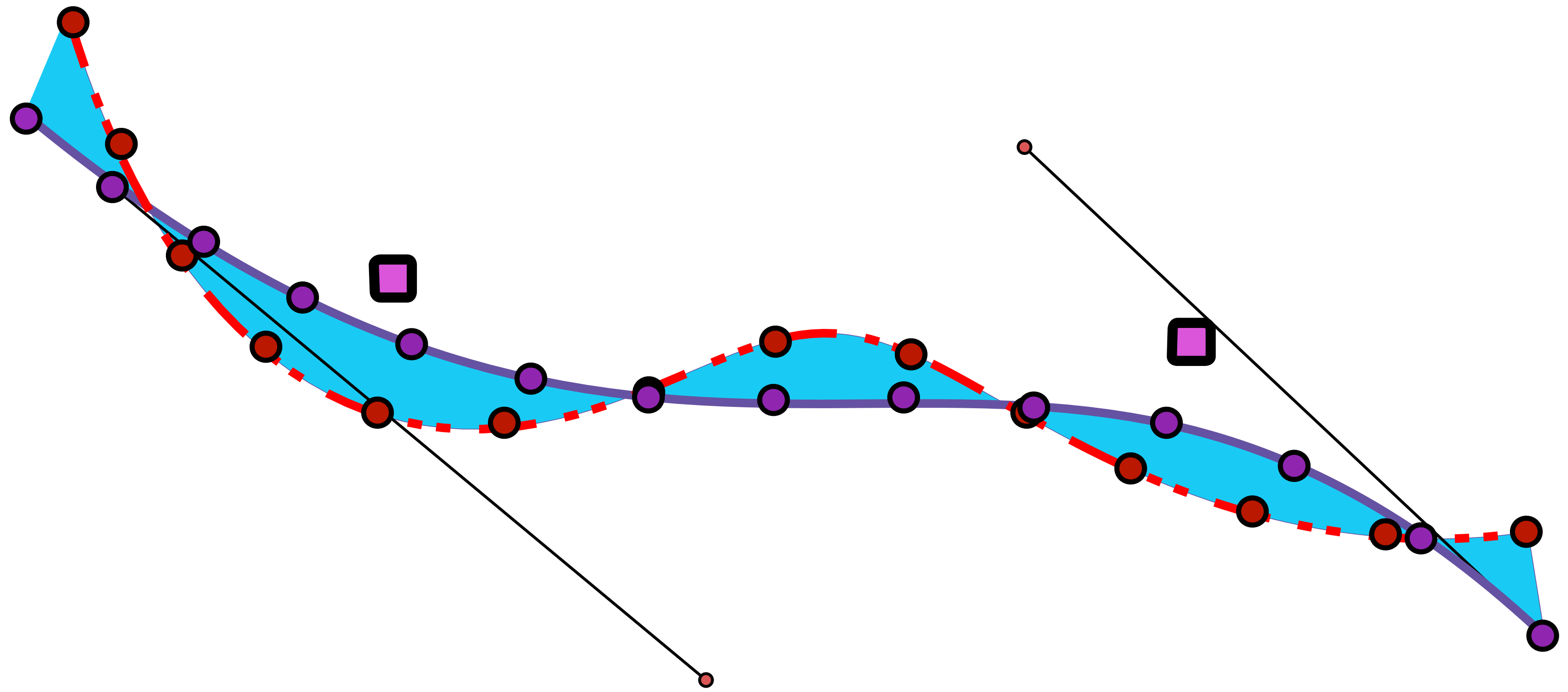
$$B_{C'}(t) - \; Target(t)$$



How it works

- Minimizing the L2 norm

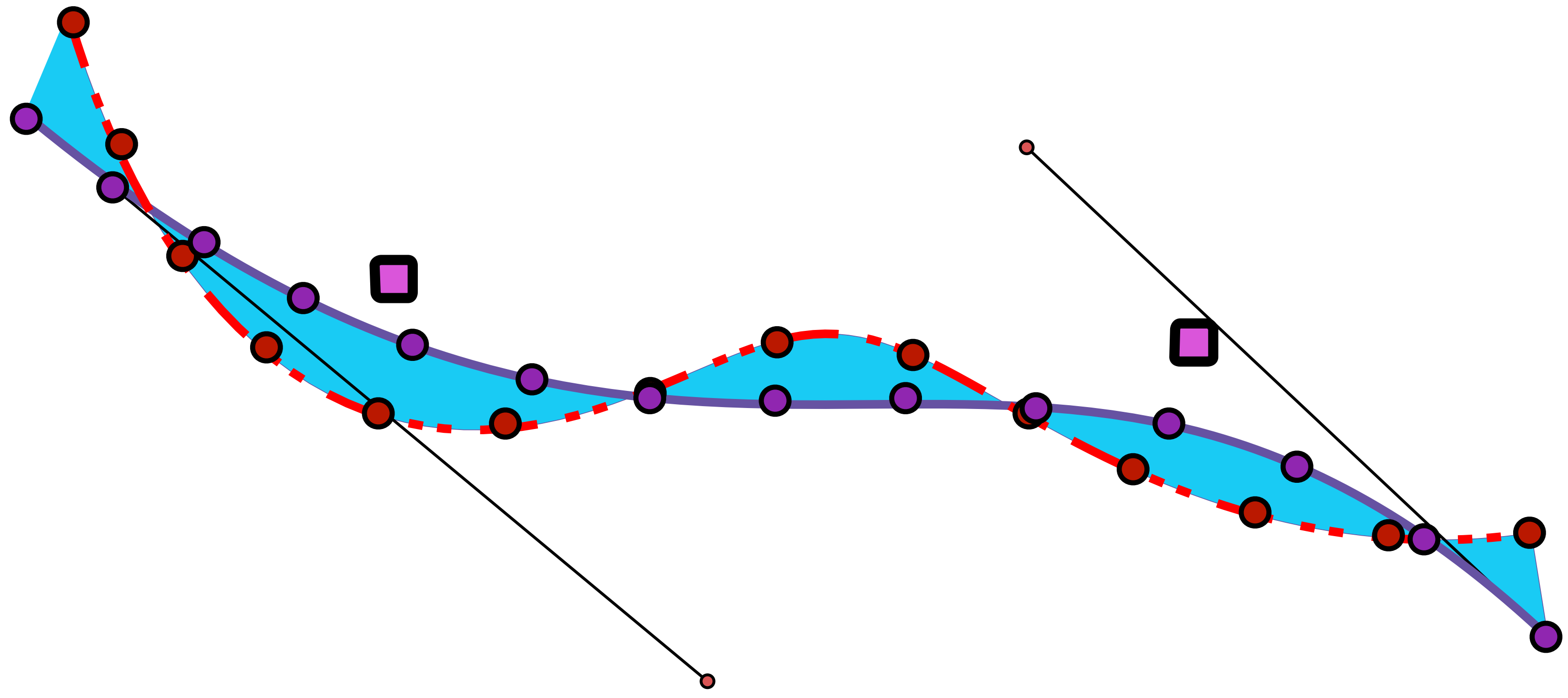
$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 \, \mathrm{d}t$$



How it works

- Minimizing the L2 norm

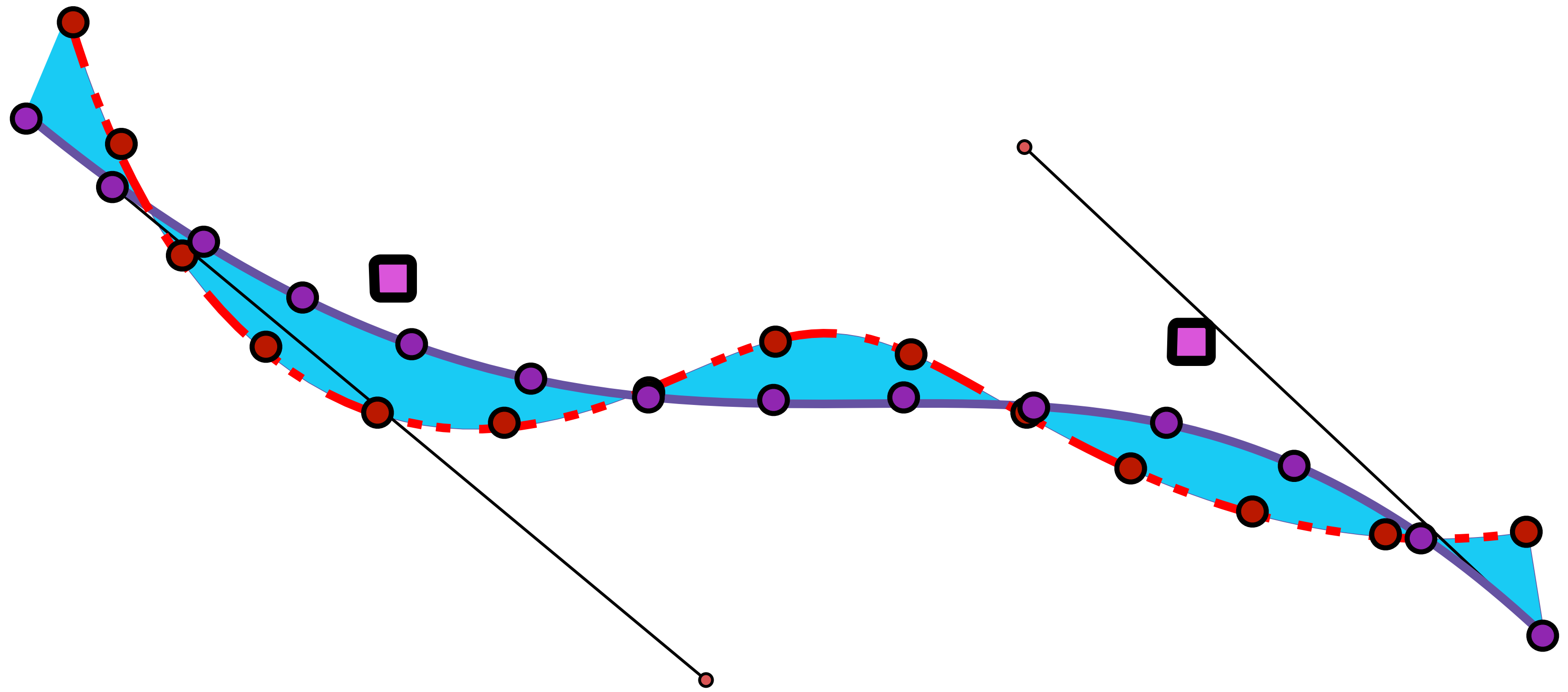
$$E(C') = \int_L \|B_{C'}(t) - Target(t)\|^2 \, \mathrm{d}t$$



How it works

- Minimizing the L2 norm

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



How it works

- Minimizing the L2 norm

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

How it works

- Minimizing the L2 norm

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

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

How it works

- Minimizing the L2 norm

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

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

How it works

- Minimizing the L2 norm

$$E(C') = \int_L \left\| B_{C'}(t) - Target(t) \right\|^2 \mathrm{d}t$$

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

Pre-computed

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

How it works

- Minimizing the L2 norm

$$E(C') = \int_L \left\| B_{C'}(t) - Target(t) \right\|^2 \mathrm{d}t$$

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

Pre-computed

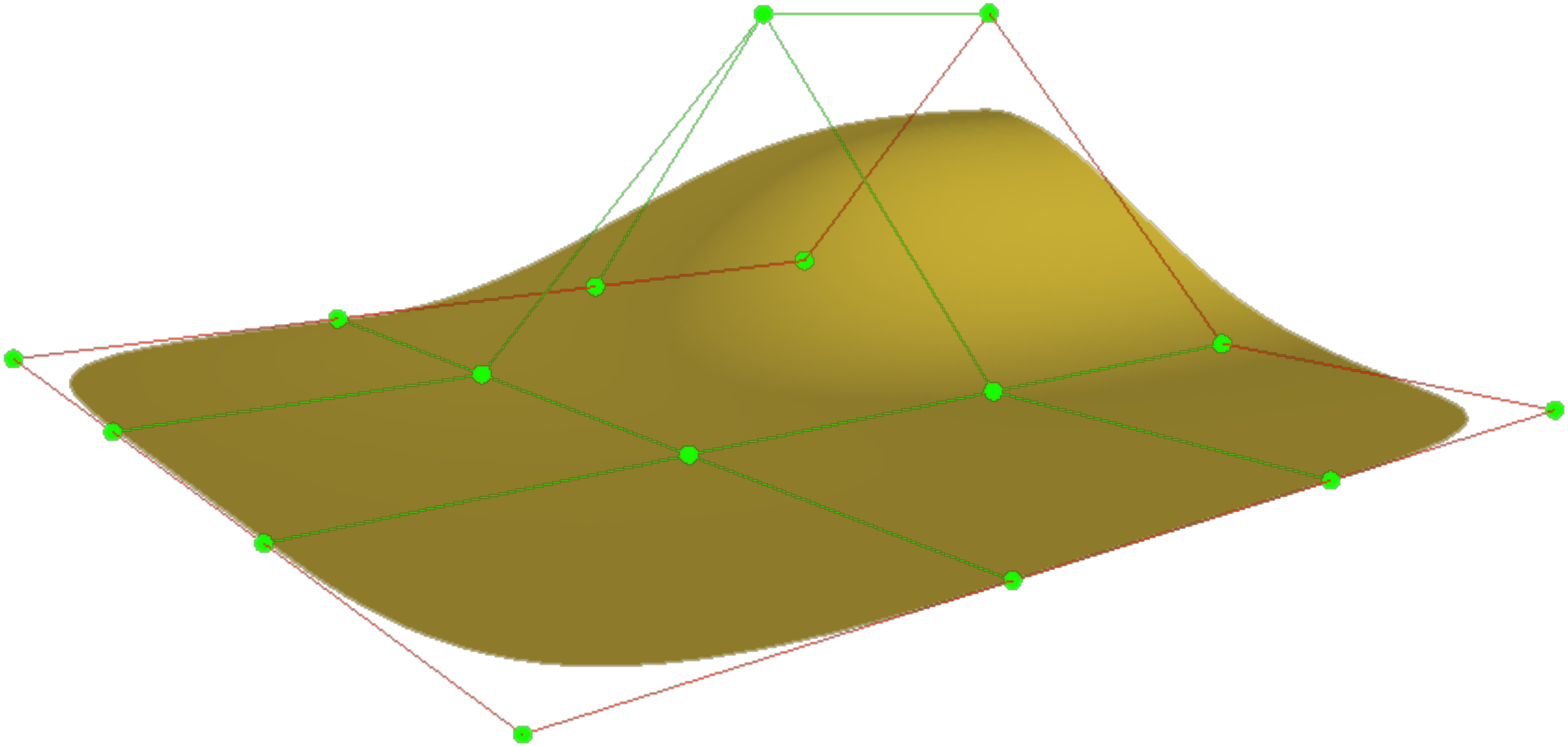
LBS: $$v' = \sum_{j \in H} T_j w_j(v) \begin{pmatrix} v \\ 1 \end{pmatrix}$$

Our approach extends to 3D

$$E(C') = \int_L \left\| B_{C'}(t) - Target(t) \right\|^2 \mathrm{d}t$$

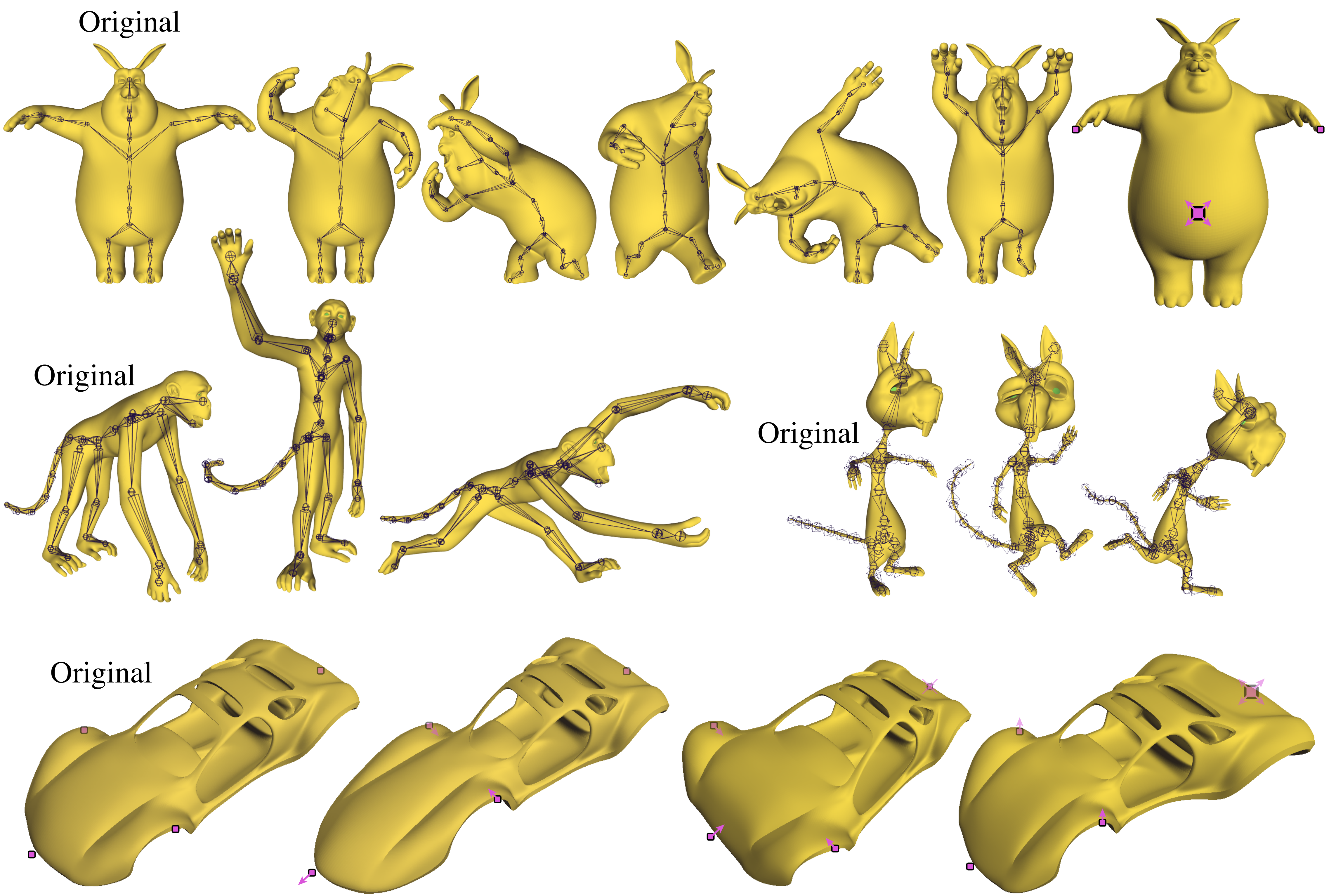
Our approach extends to 3D

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

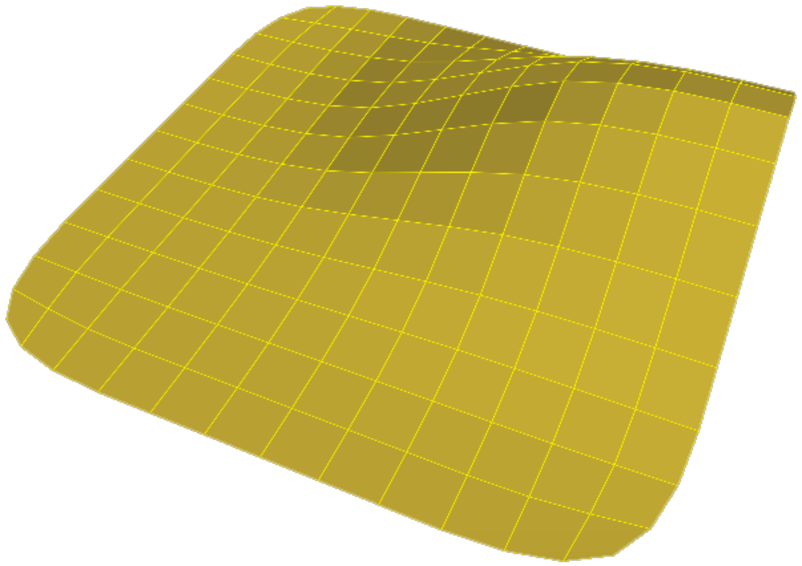


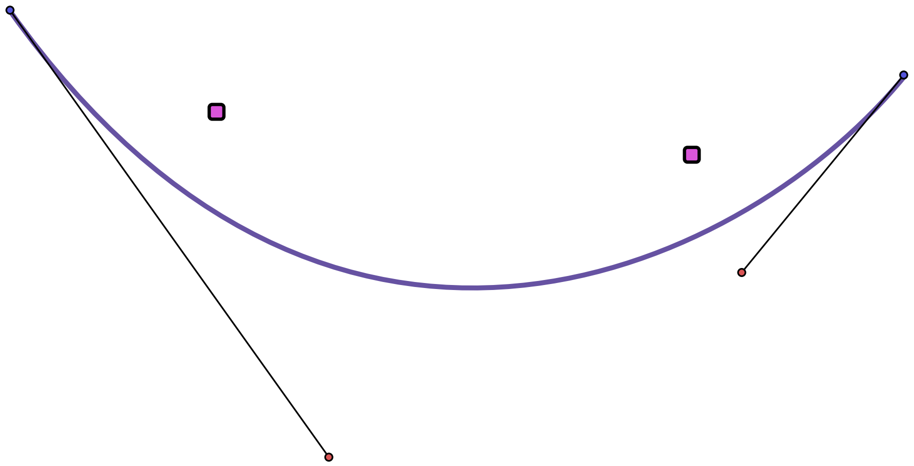
3D Live Demo

3D Results

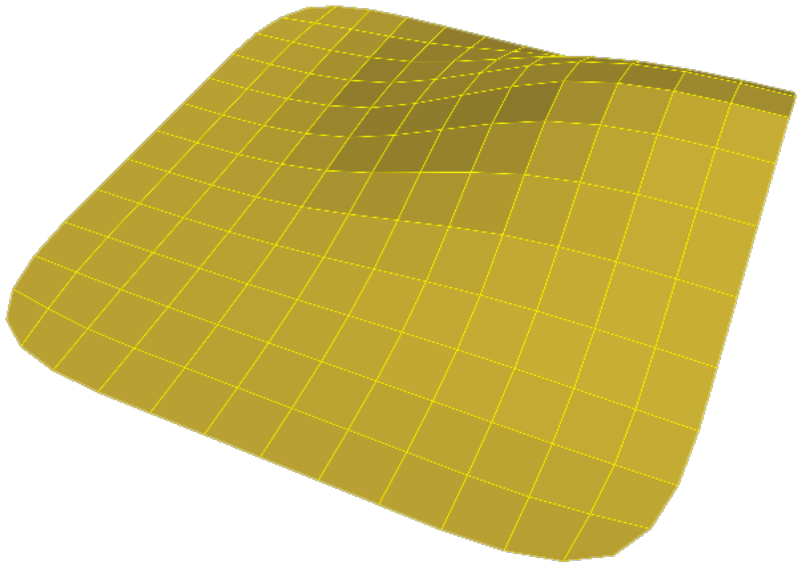


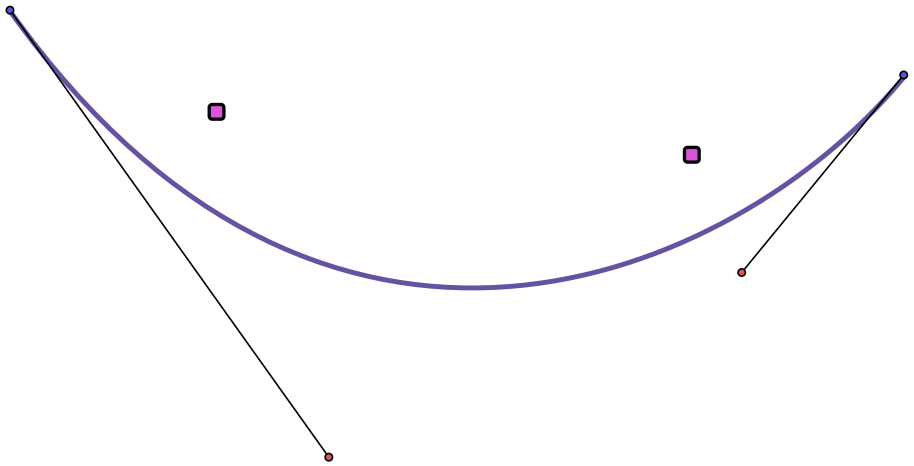
Why 2D is more complicated?





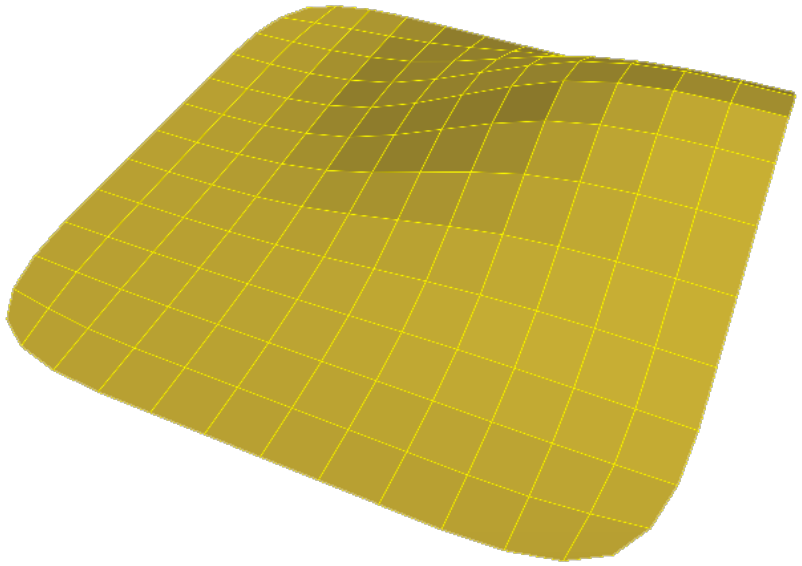
Why 2D is more complicated?

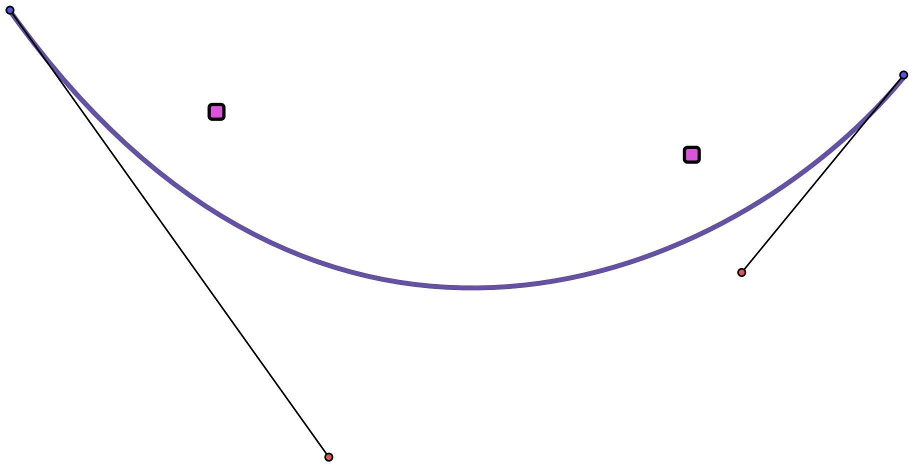




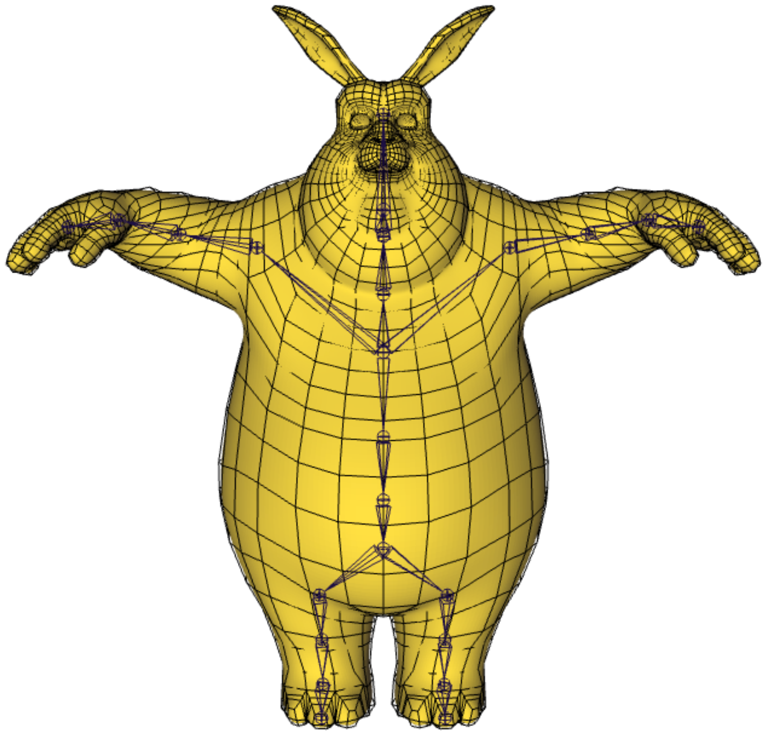
In practice:

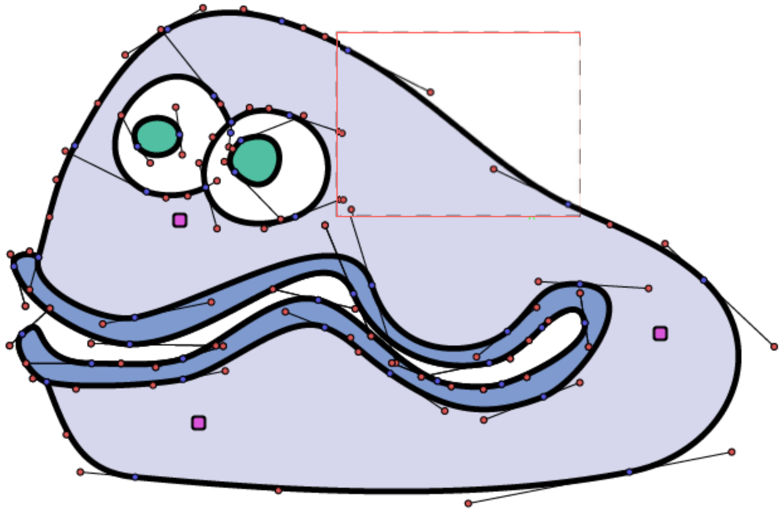
Why 2D is more complicated?



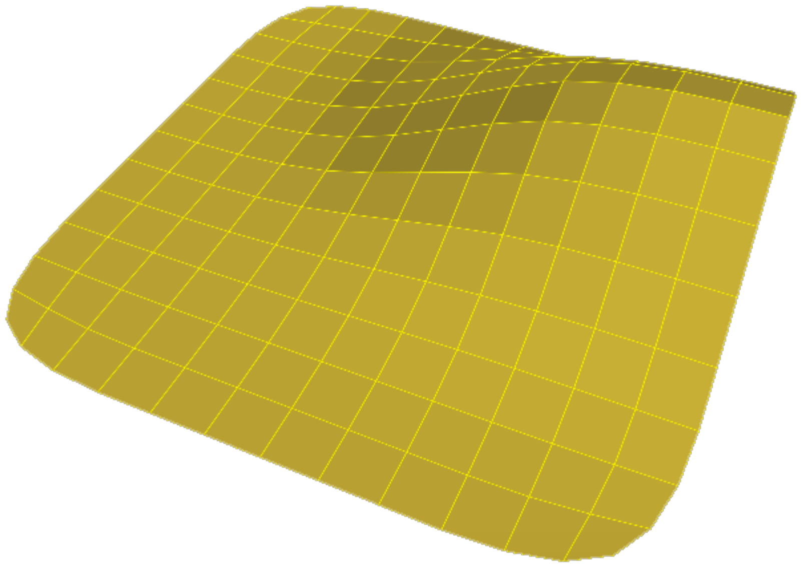


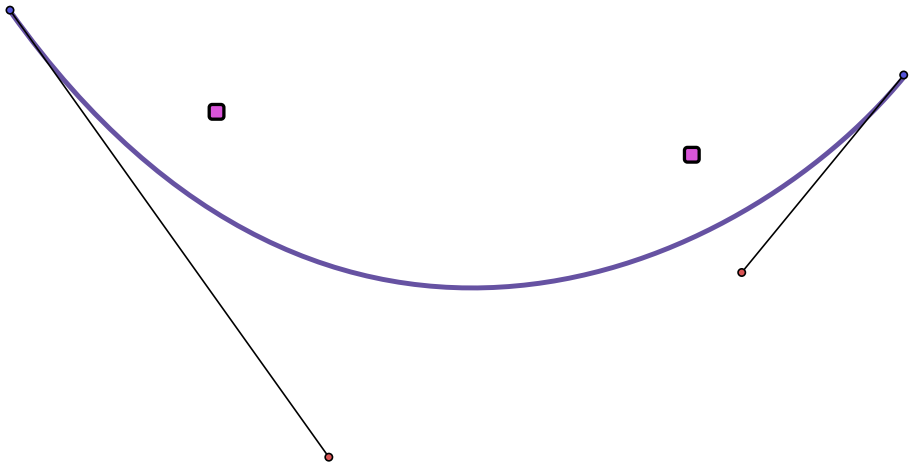
In practice:



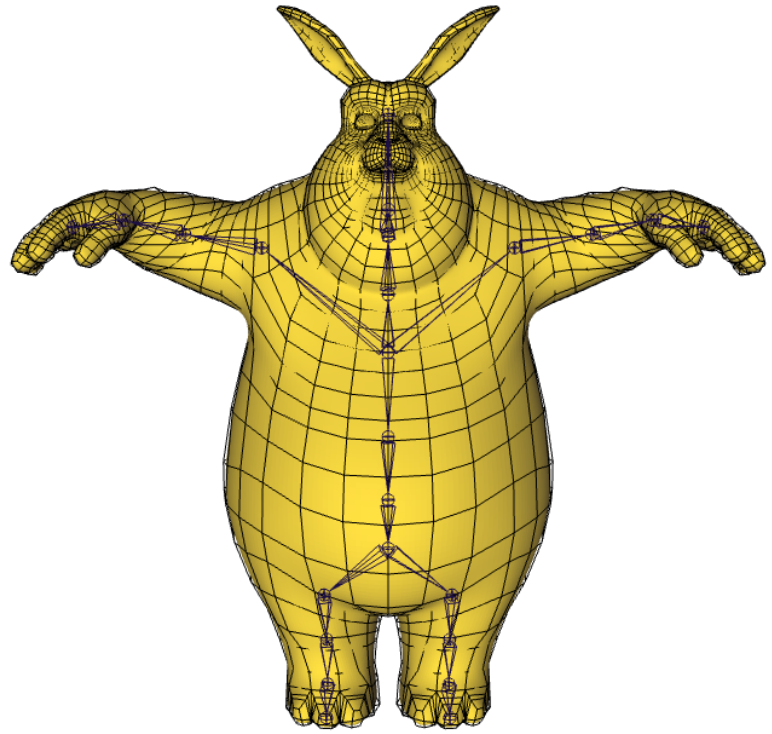


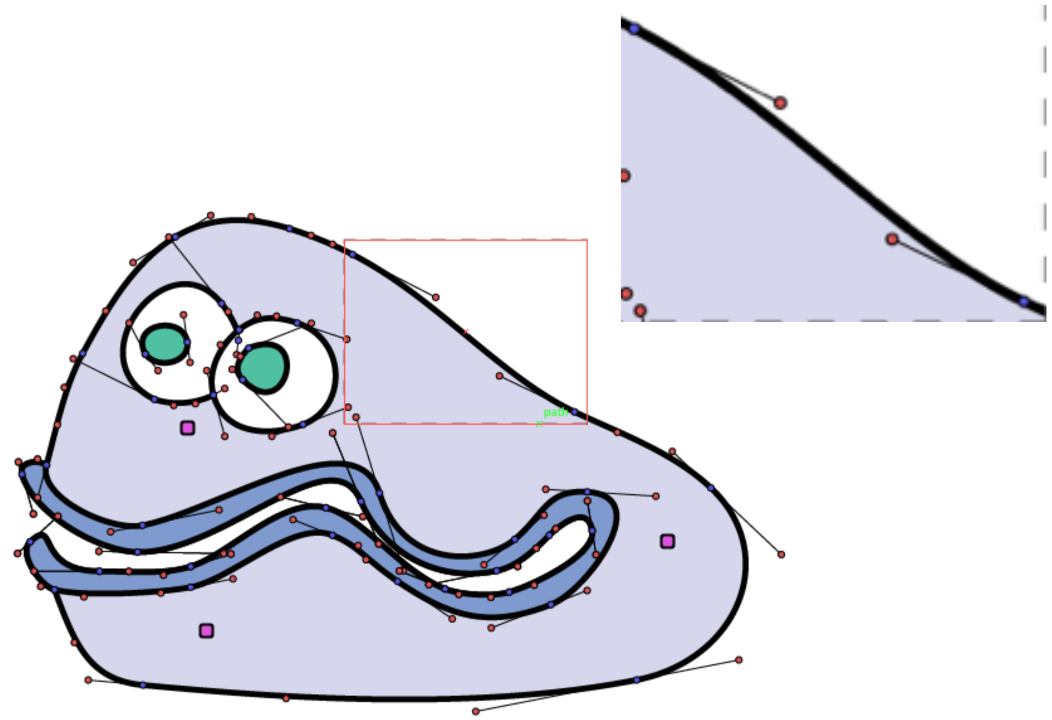
Why 2D is more complicated?



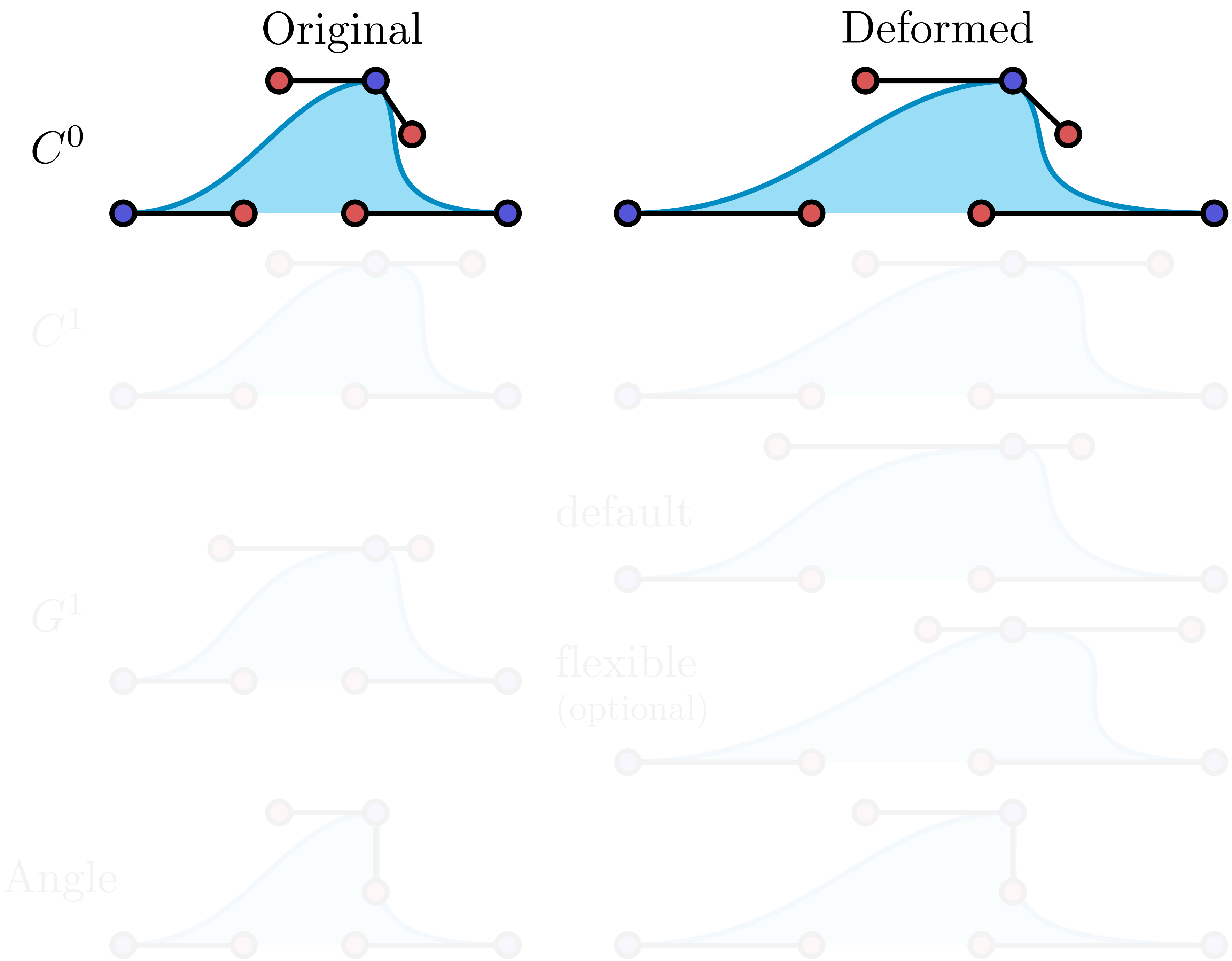


In practice:

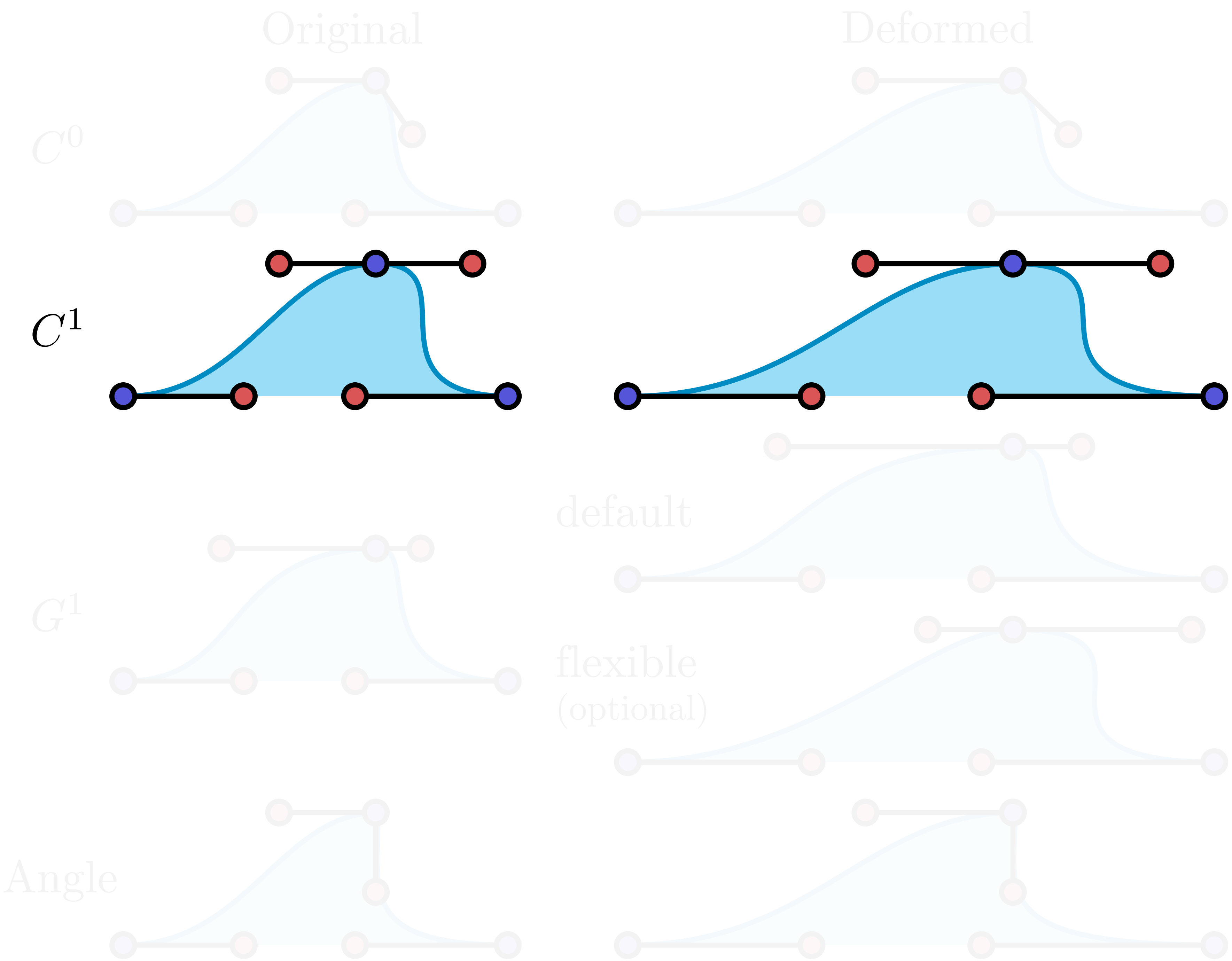




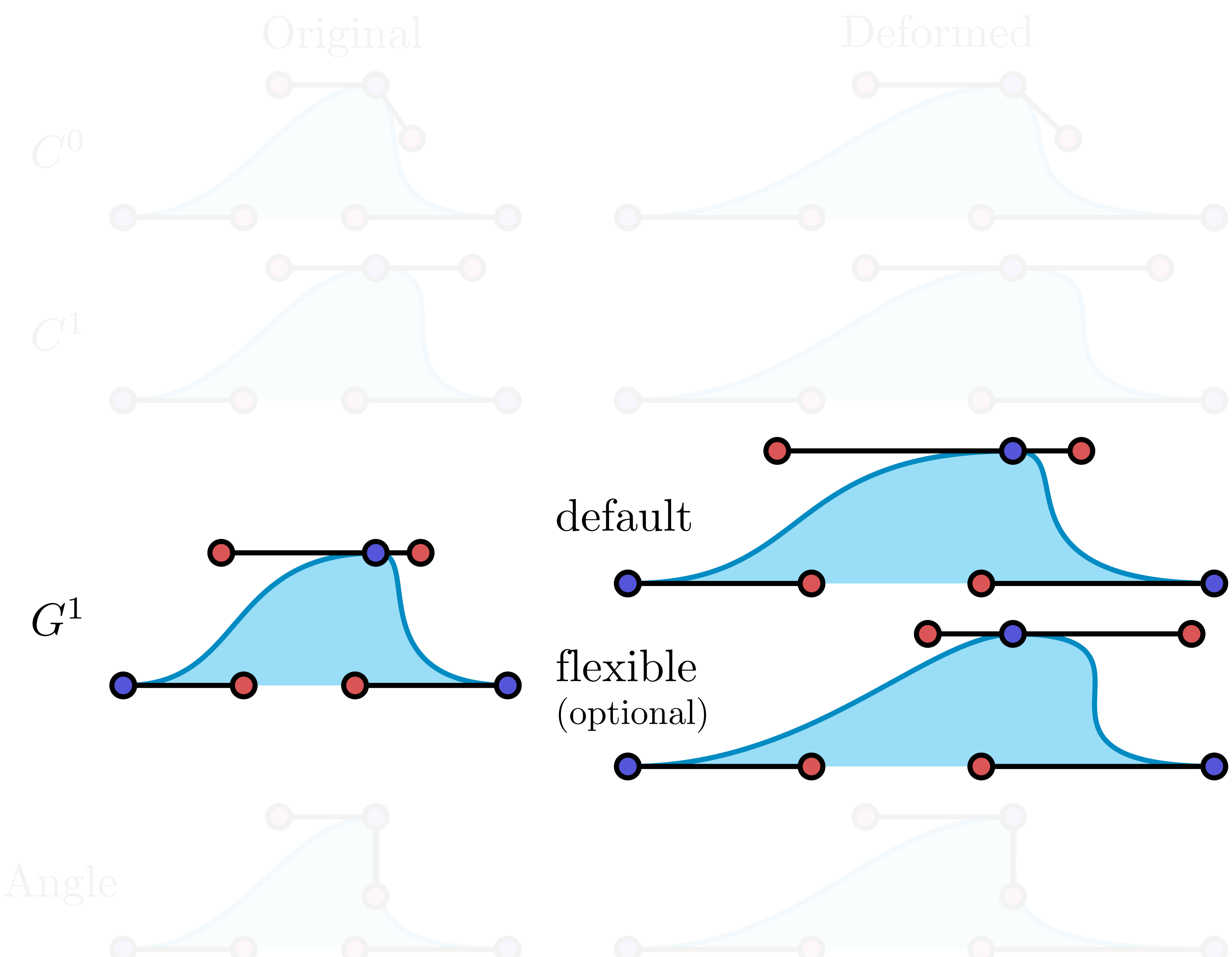
Bézier Spline Junctions



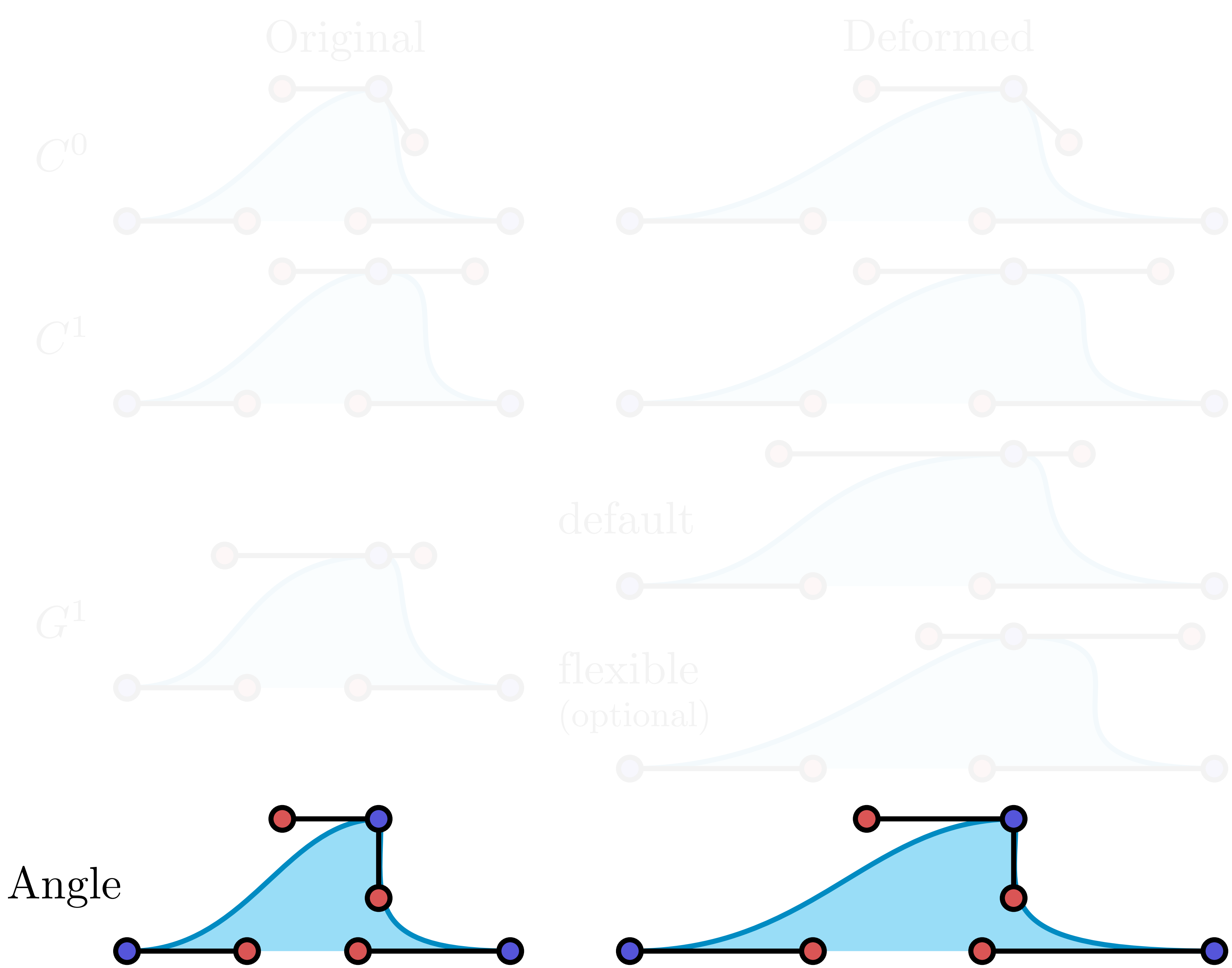
Bézier Spline Junctions



Bézier Spline Junctions



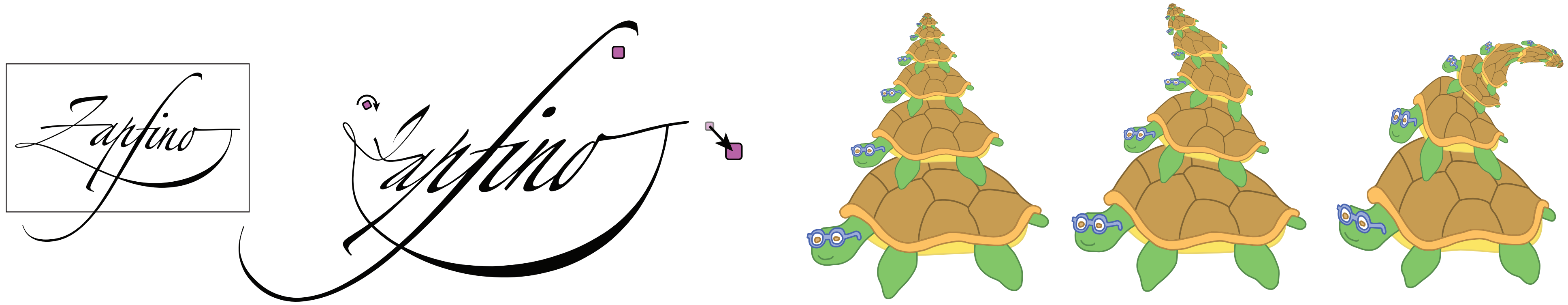
Bézier Spline Junctions



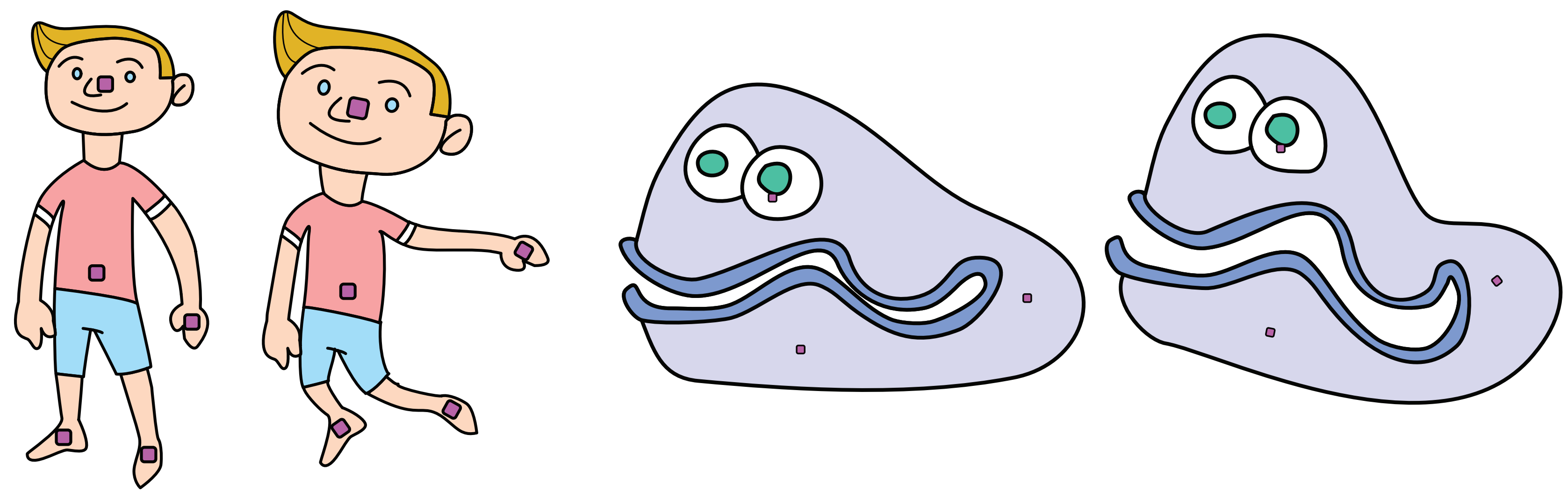
2D Live Demo

Applied on different weights

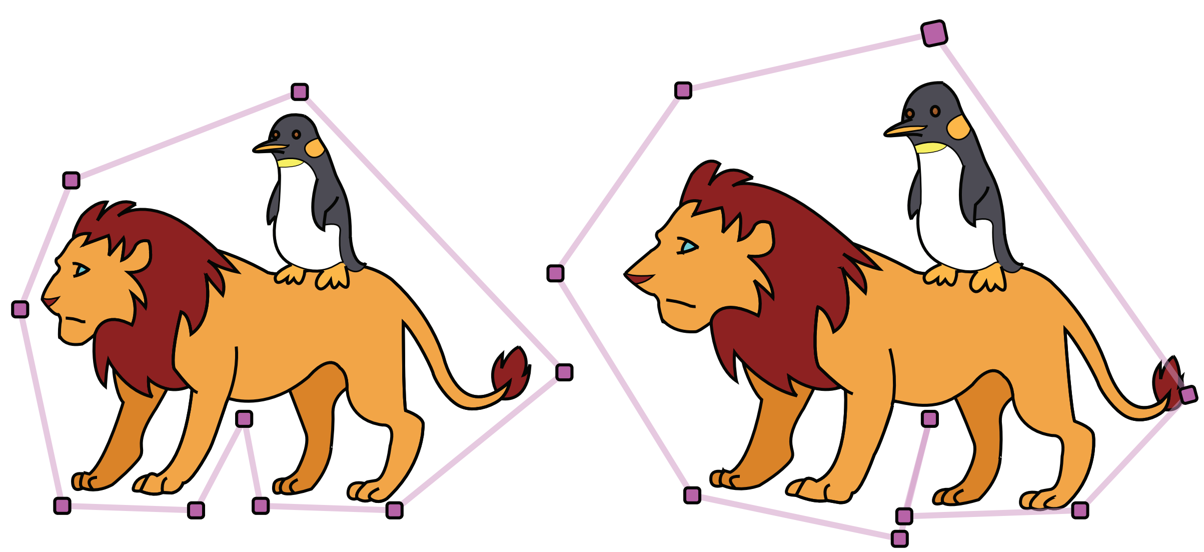
Shepard's Weights
[Shepard 1968]



Bounded Biharmonic Weights
[Jacobson et al. 2011]



Harmonic Coordinates
[Joshi et al. 2007]

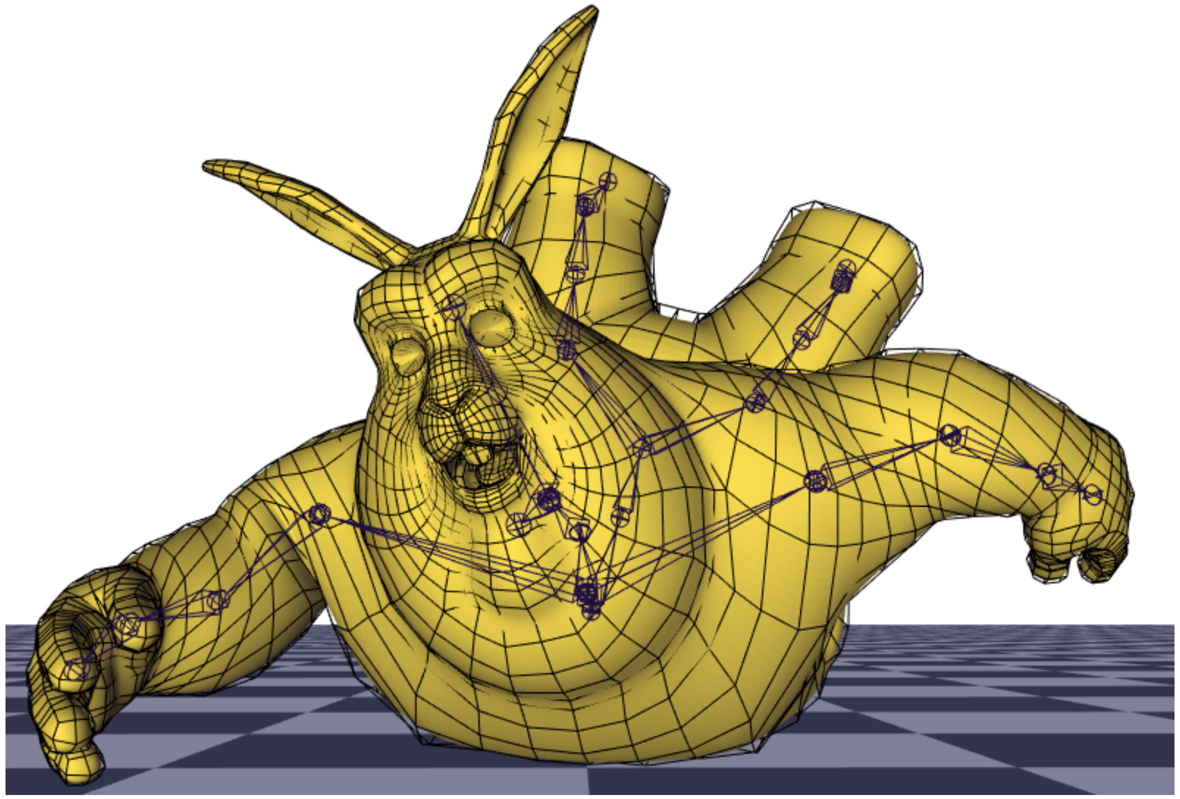


Performance

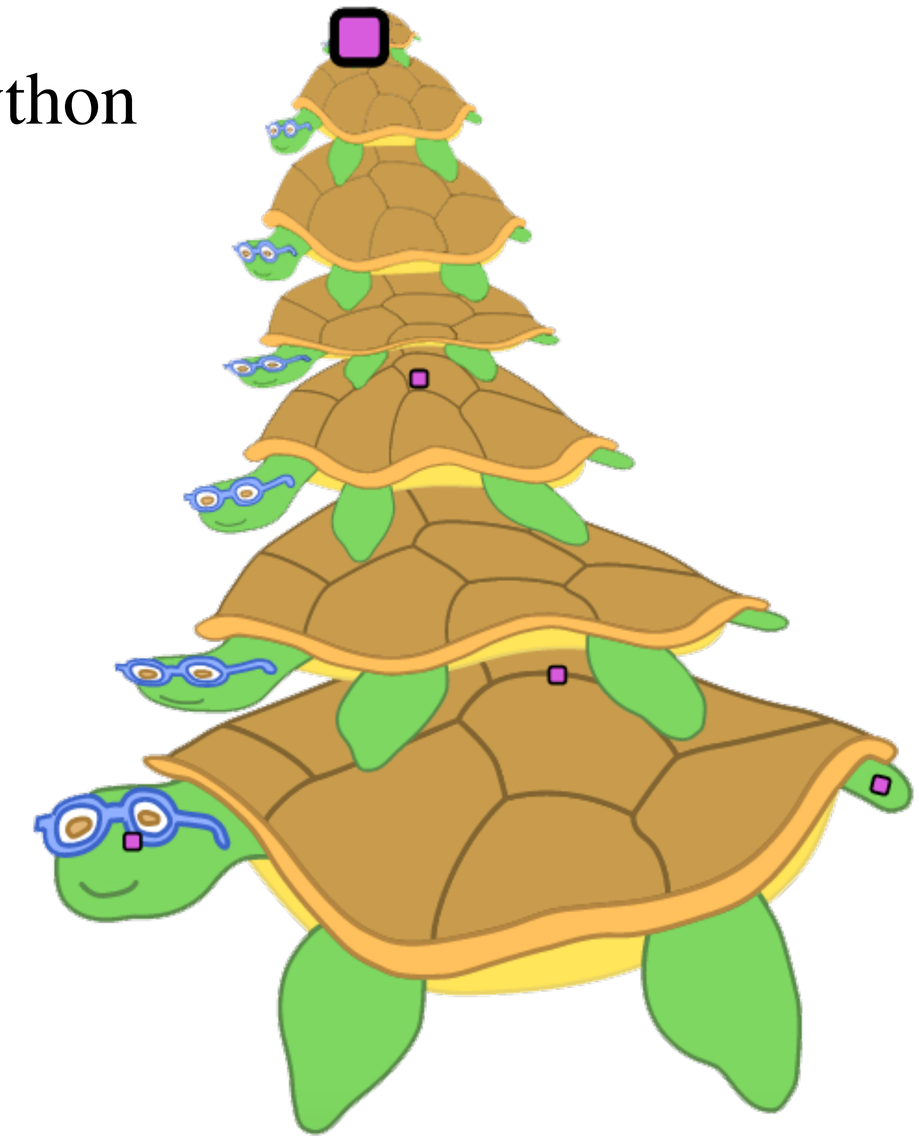
# faces	pre-compute time(secs)	seconds per update
4150	9	0.0002

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

in C++



in Python



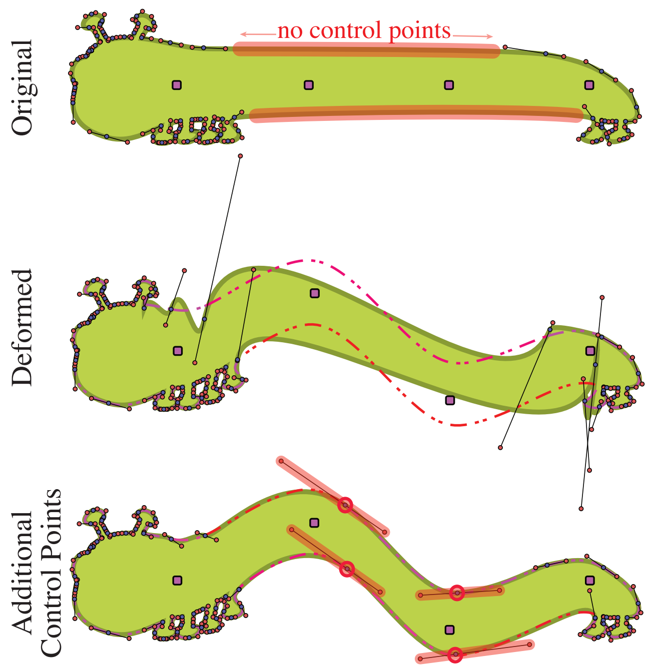
Limitation and Future Work

Limitation and Future Work

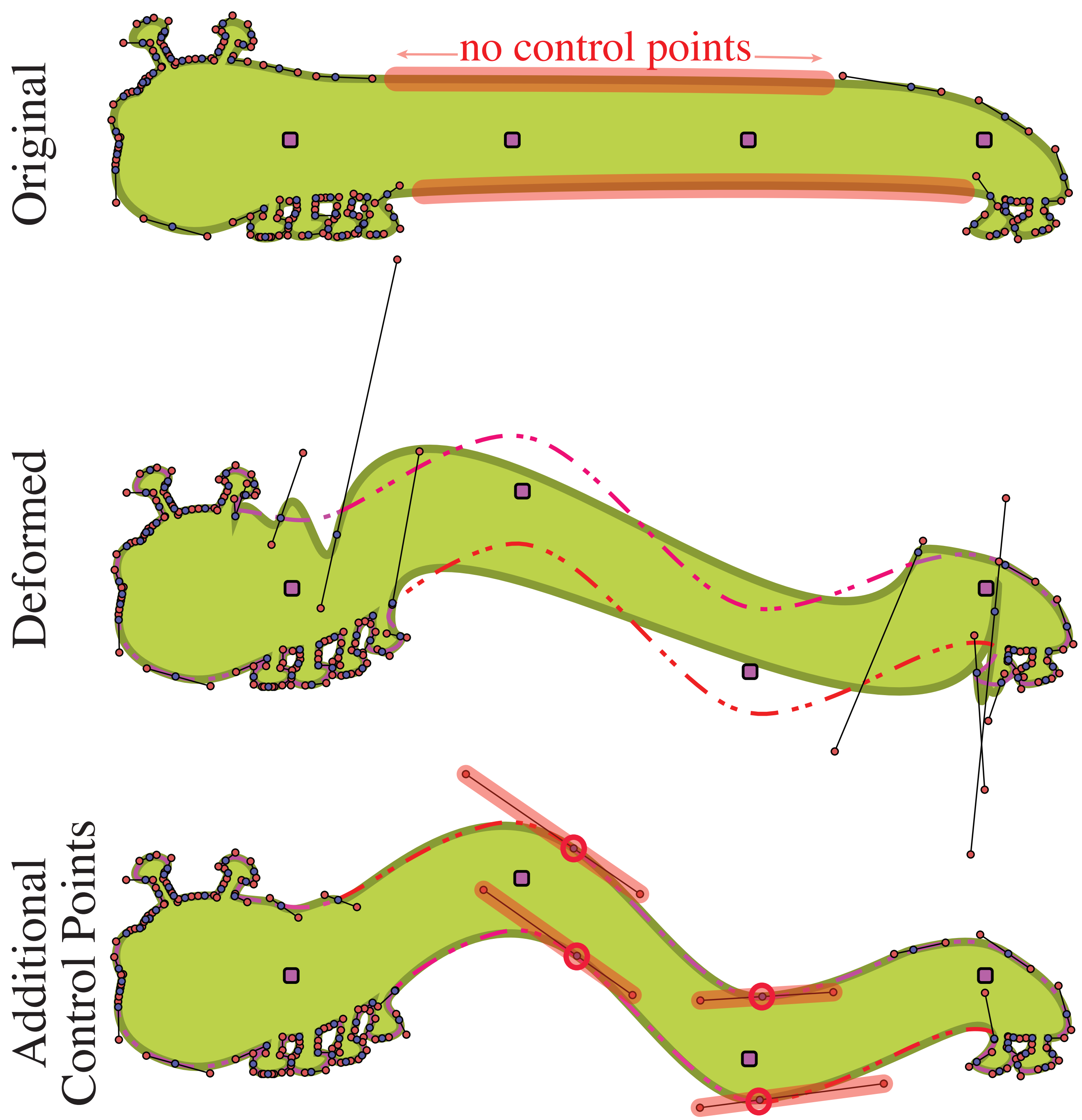
- **Other primitives in vector graphics**
  - **circular arcs, NURBS, clothoid splines.**
  - **linear or radial gradients, diffusion curves, texture.**

Limitation and Future Work

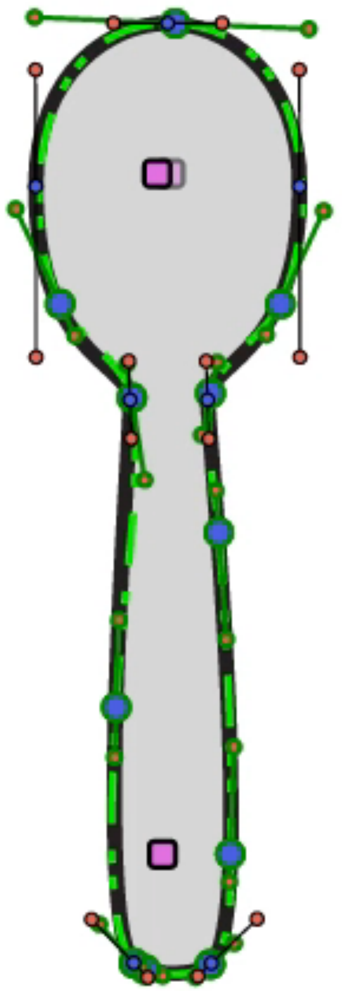
- **Other primitives in vector graphics**
  - **circular arcs, NURBS, clothoid splines.**
  - **linear or radial gradients, diffusion curves, texture.**
- **Adaptivity**



Limitation and Future Work

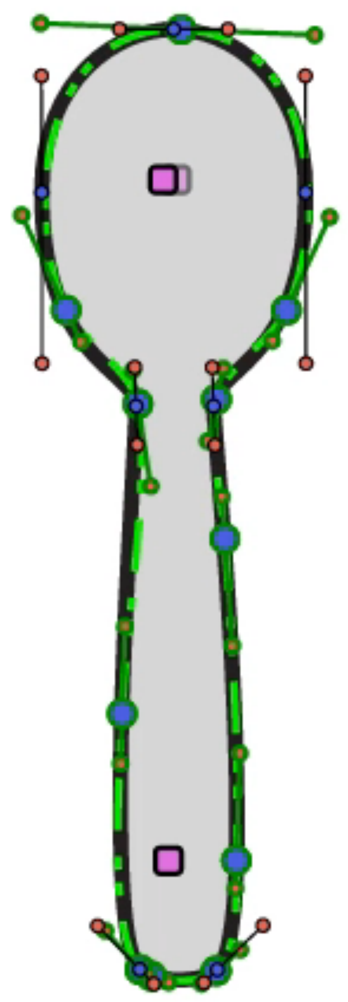


Limitation and Future Work

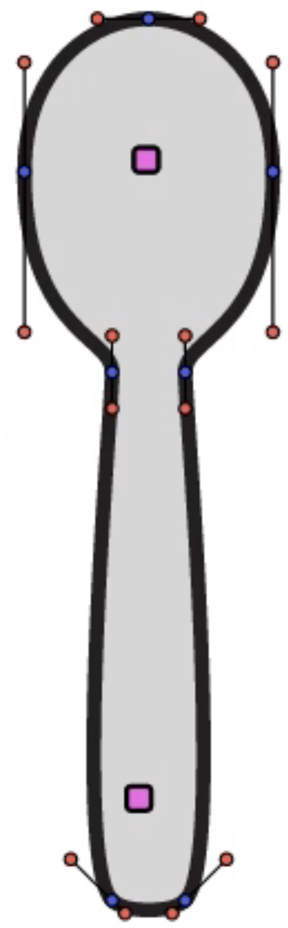


[Schneider 1990]

Limitation and Future Work



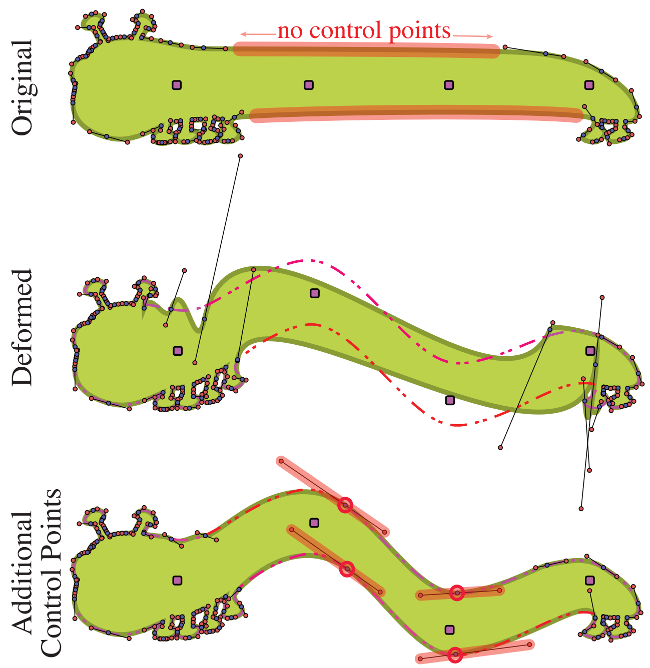
[Schneider 1990]

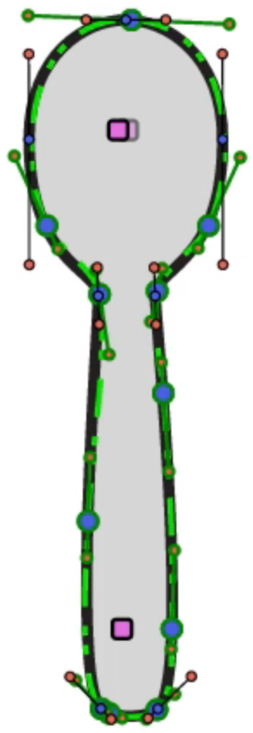


[Our approach]

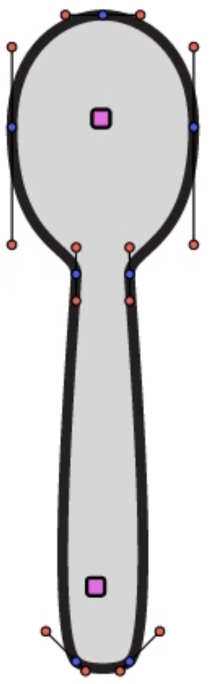
Limitation and Future Work

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





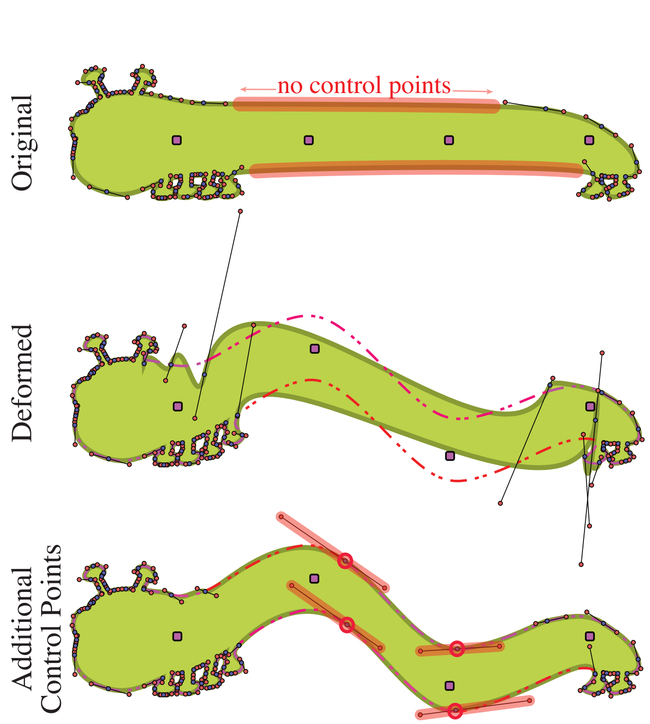
[Schneider 1990]

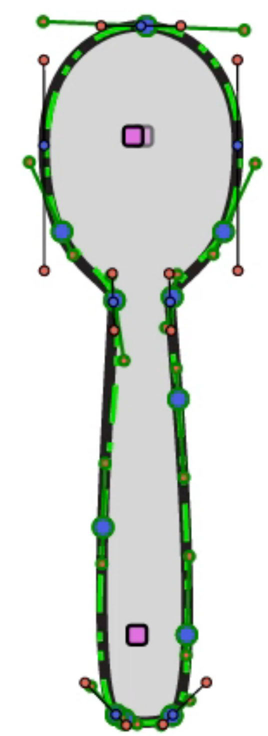


[Our approach]

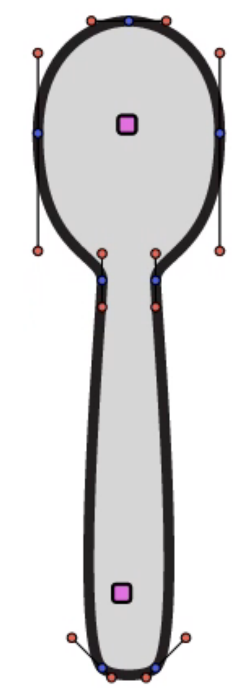
Limitation and Future Work

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



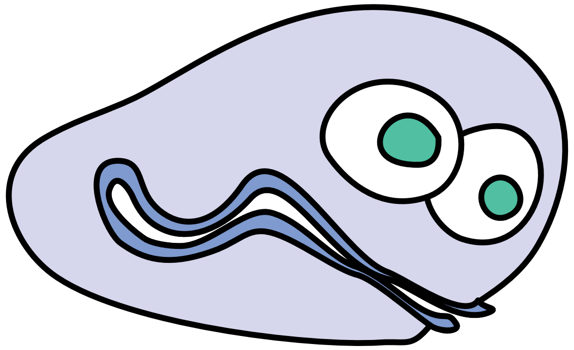


[Schneider 1990]



[Our approach]

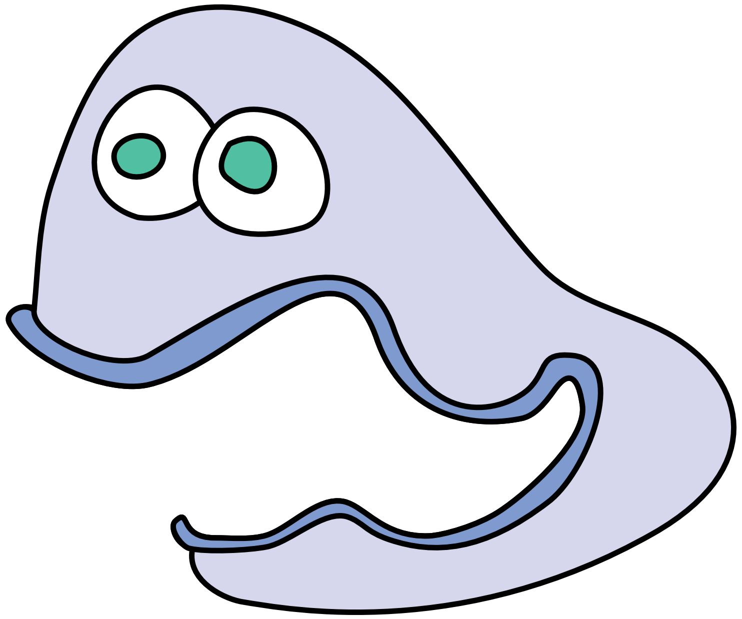
- **Other applications**
  - **motion path planning**



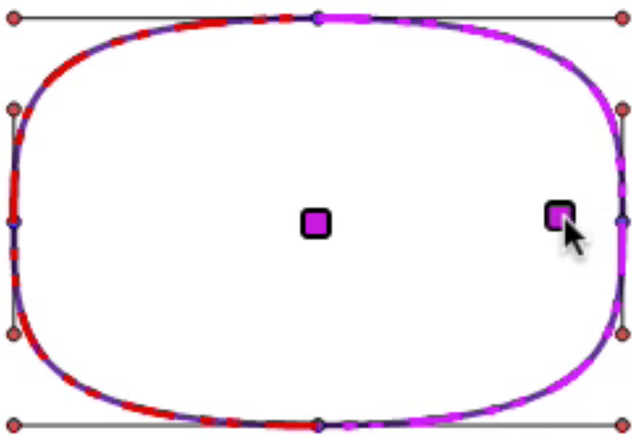
Thank You.

Songrun Liu, sliu11@gmu.edu
Alec Jacobson, jacobson@cs.columbia.edu
Yotam Gingold, ygingold@gmu.edu
project webpage: http://cs.gmu.edu/~ygingold/splineskin/, code coming soon.

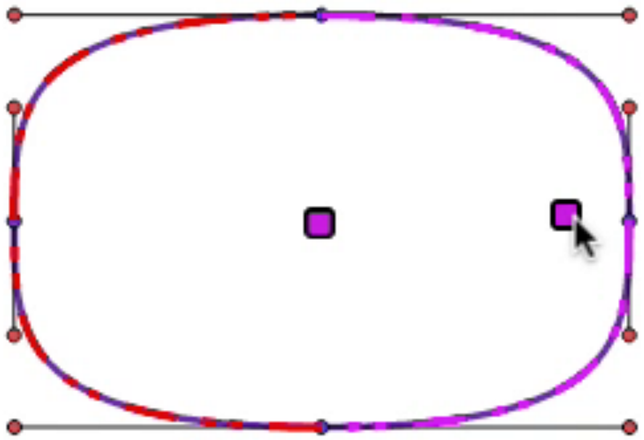
Acknowledgements:
We thank support from NSF, Google, Intel, The Walt Disney Company, and Autodesk.
Special thanks to Michelle Lee for her 2D artwork and Blender Foundation for 3D models from “Big Buck Bunny”.



Junction between Curves (smooth)

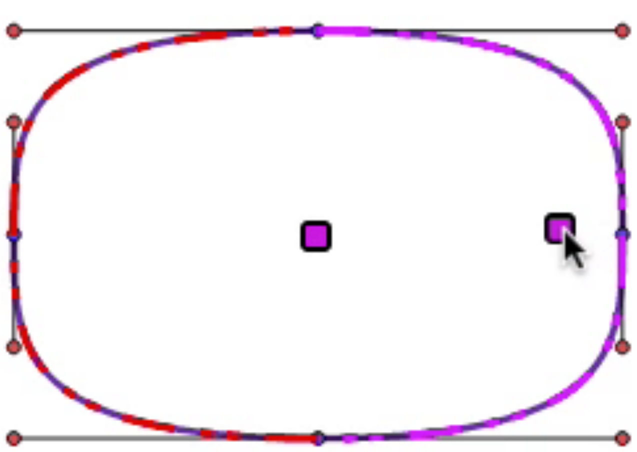


C^1 continuity

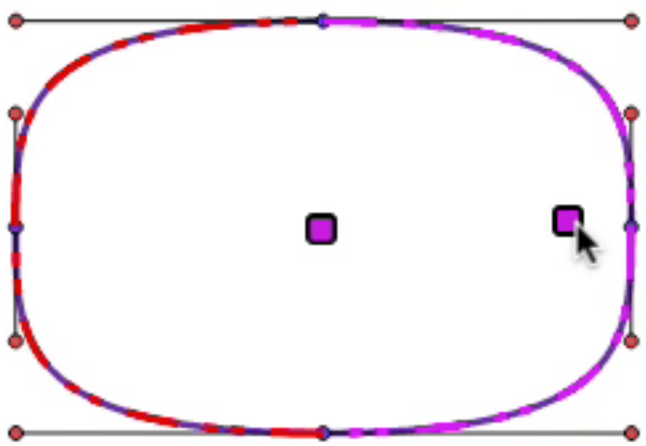


G^1 flexibility

Junction between Curves (smooth)

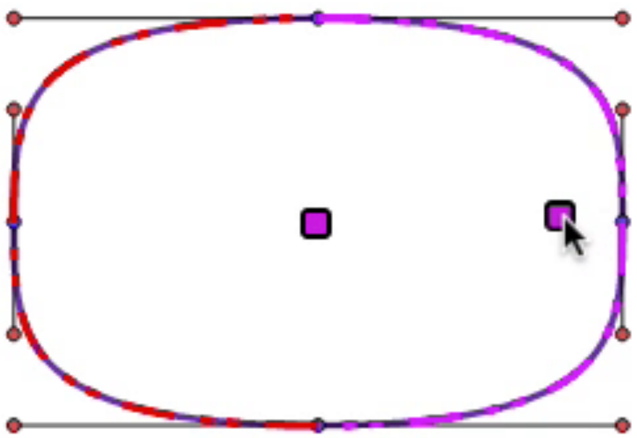


C^1 continuity

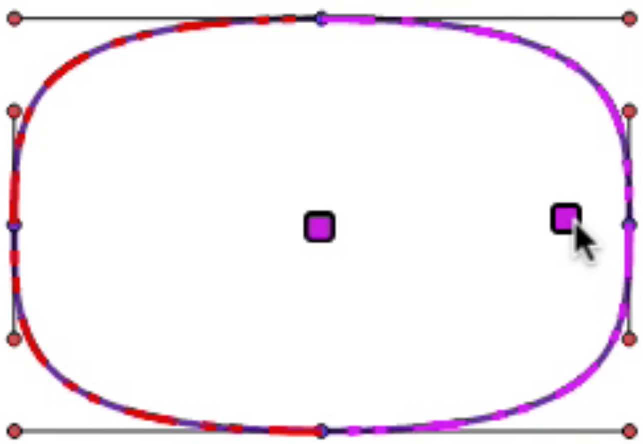


G^1 flexibility

Junction between Curves (smooth)

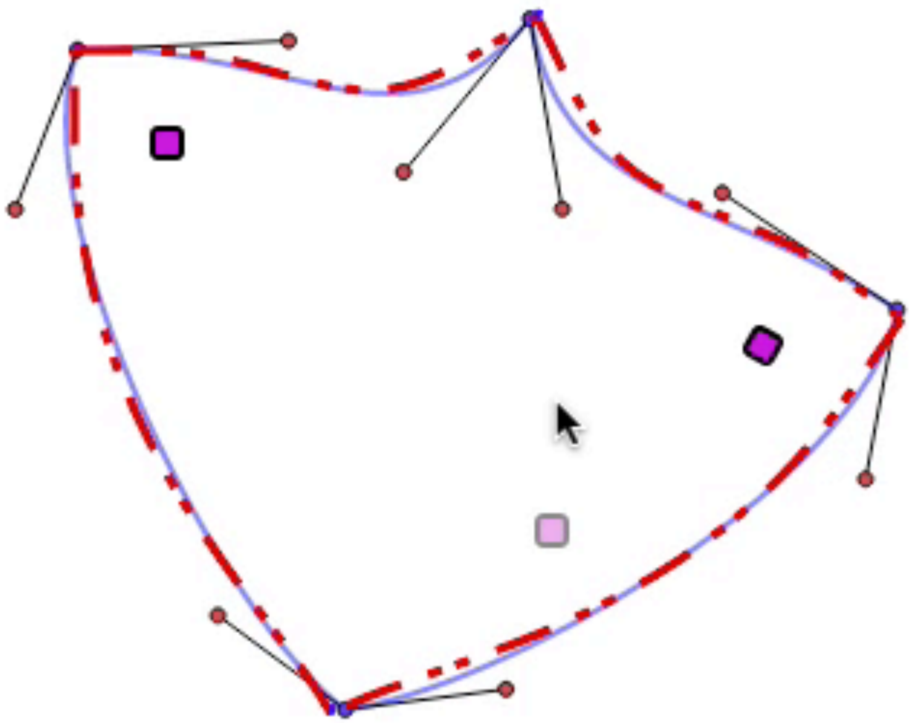


C^1 continuity

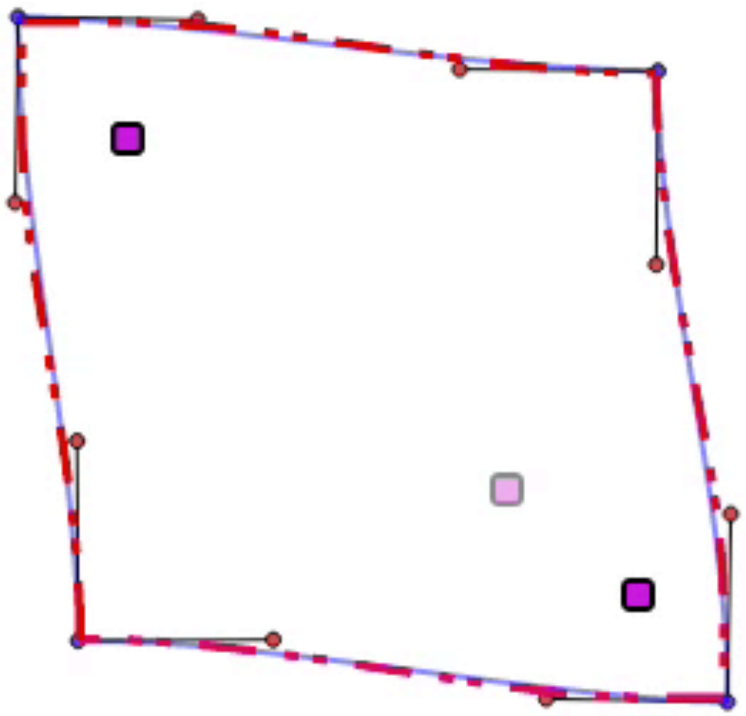


G^1 flexibility

Junction between Curves (sharp)

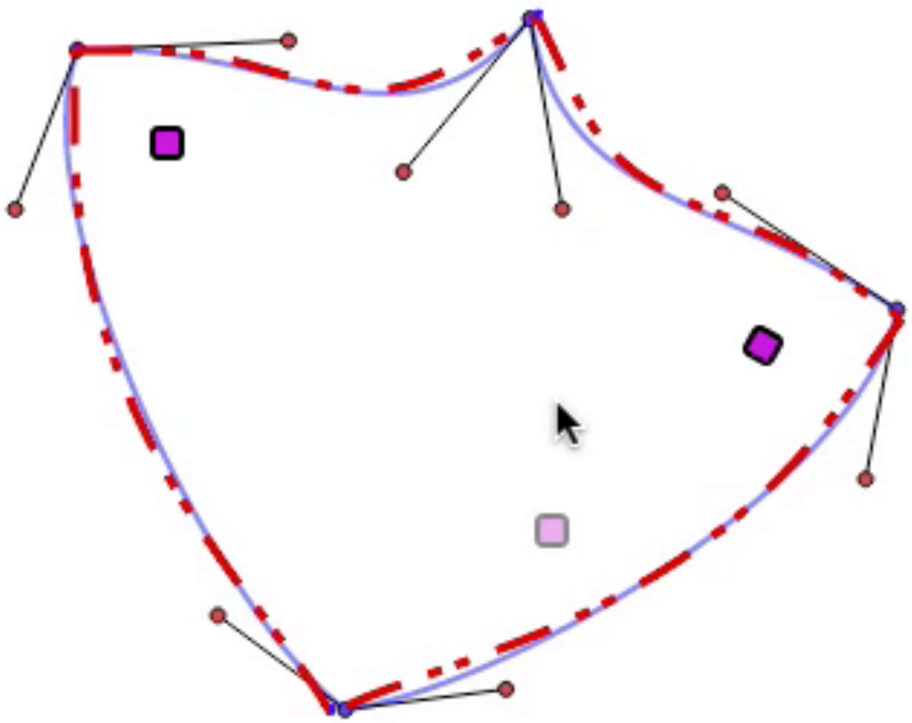


No Angle Preservation

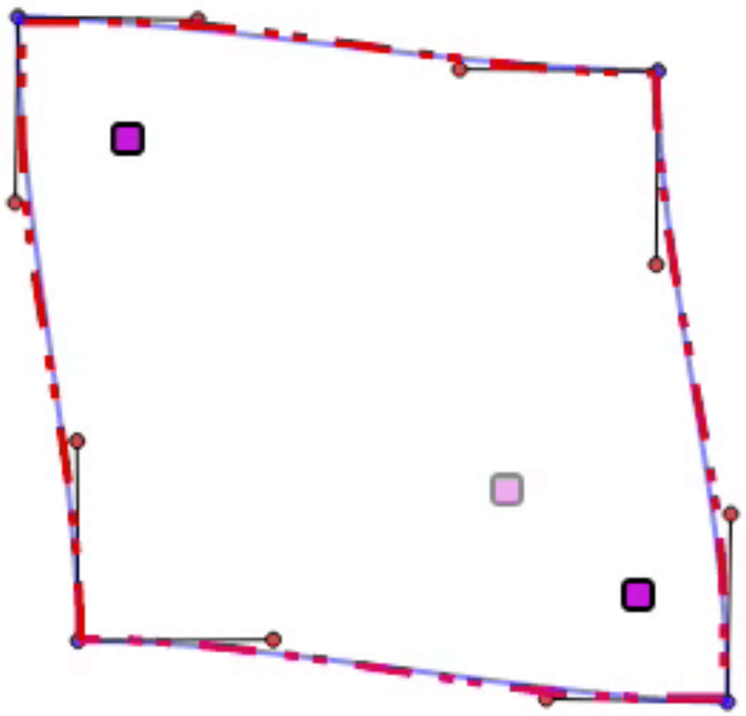


Angle Preservation

Junction between Curves (sharp)



No Angle Preservation

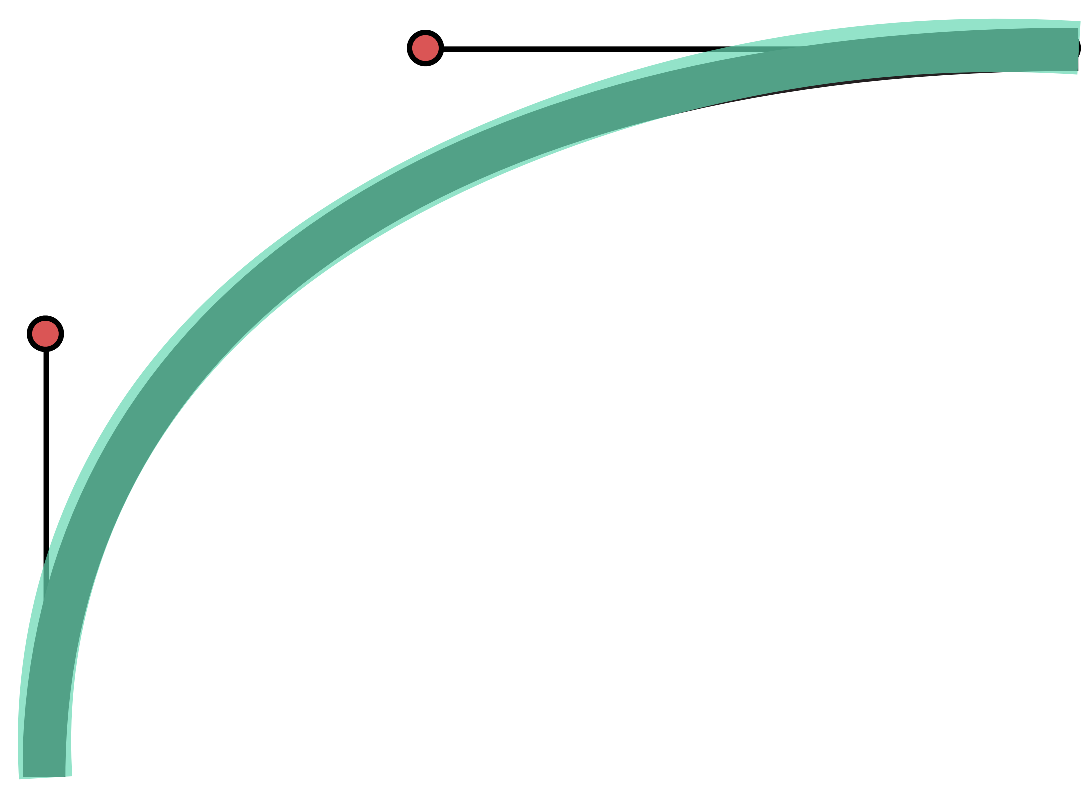


Angle Preservation

2D Vector Graphics

- Splines in 2D — Cubic Bézier splines.

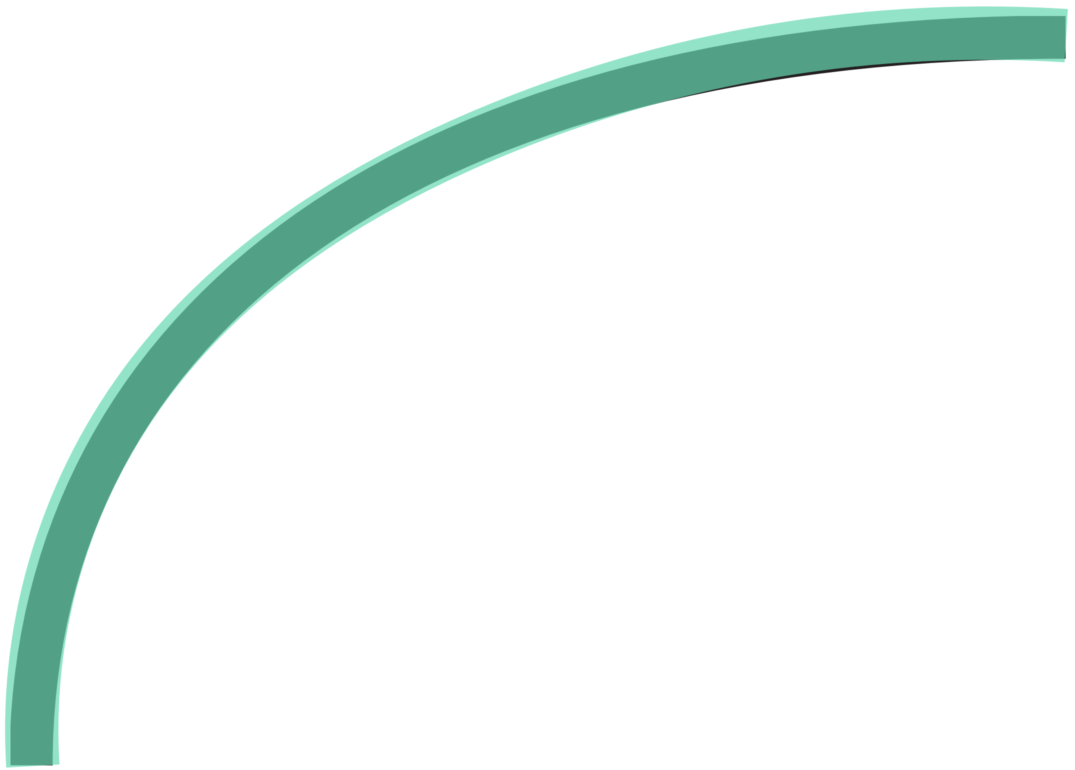
$$B_C(t) = CM\boxed{\bar{t}} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \boxed{\begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}} = Cm_t,$$



2D Vector Graphics

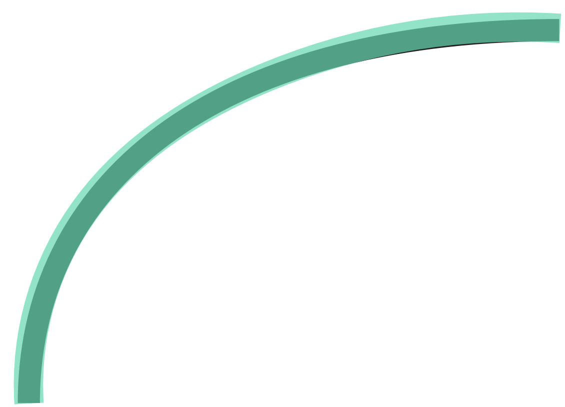
- Splines in 2D — Cubic Bézier splines.

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



- Splines in 2D — Cubic Bézier splines.

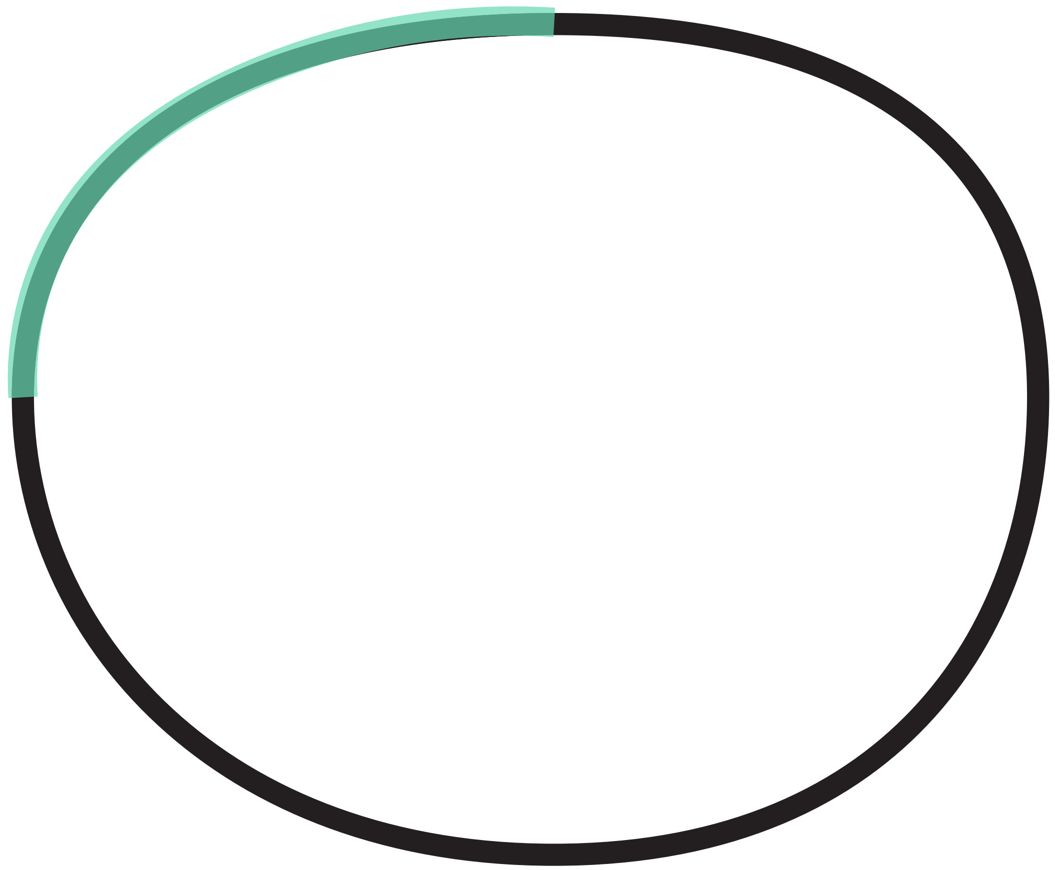
$$B_C(t) = CM\bar{t} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = Cm_t,$$



2D Vector Graphics

- Splines in 2D — Cubic Bézier splines.

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



How it works

- Minimizing the L2 norm

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

$$C' = \sum_{i=1}^{h} T_i \boxed{\hat{W}_i \hat{A}^{-1}}$$

Pre-computed

$$A(p) = m_p m_p^T \qquad \hat{A} = \int_D A(p)\,\mathrm{d}p \qquad \hat{W}_i = C \int_D w_i(C m_p) A(p)\,\mathrm{d}p$$

How it works

- Minimizing the L2 norm

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

$$C' = \sum_{i=1}^{h} T_i \hat{W}_i \hat{A}^{-1}$$

Pre-computed

$$A(p) = m_p m_p^T \qquad \hat{A} = \int_D A(p)\,\mathrm{d}p \qquad \hat{W}_i = C \int_D w_i(C m_p) A(p)\,\mathrm{d}p$$

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

How it works

- Play the video of minimizing the L2 norm

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

$$C' = \sum_{i=1}^{h} T_i \boxed{\hat{W}_i \hat{A}^{-1}}$$

Pre-computed

$$\hat{A}^{-1} = \begin{bmatrix} 16 & -24 & 16 & -4. \\ -24 & 69\frac{1}{3} & -57\frac{1}{3} & 16 \\ 16 & -57\frac{1}{3} & 69\frac{1}{3} & -24 \\ -4 & 16 & -24 & 16 \end{bmatrix} \qquad \hat{W}_i = C \int_D w_i(Cm_p) A(p)\, \mathrm{d}p$$

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

SIGGRAPH ASIA 2014 SHENZHEN

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	seconds per update: 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	seconds per update 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.