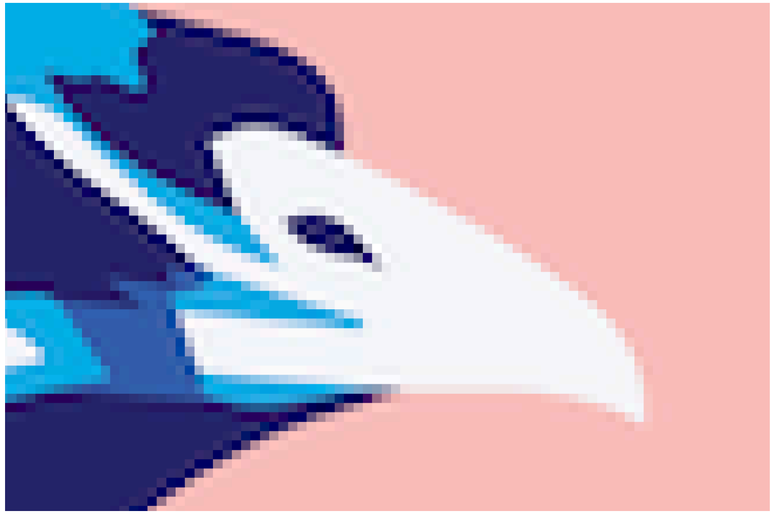
16 - Drawing on meshes

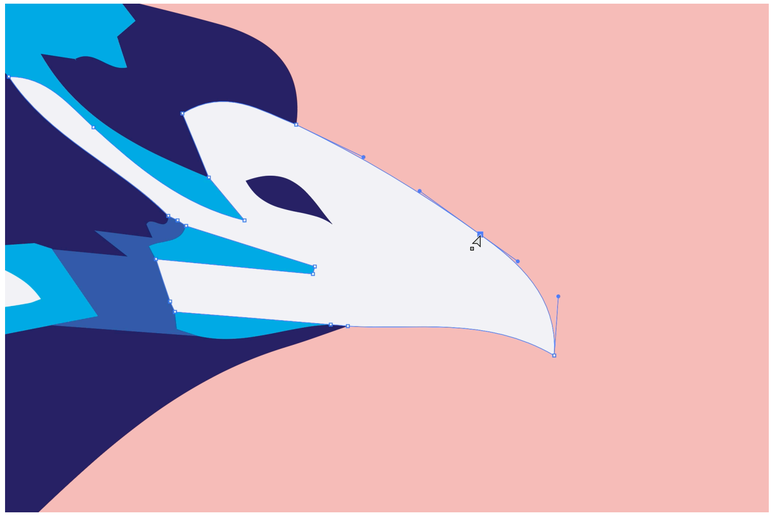
In this lecture

- Vector graphics on surfaces:
  - basic primitives (segment, circle, ellipse, …)
  - Bézier curves

Vector graphics vs raster graphics

Vector graphics vs raster graphics



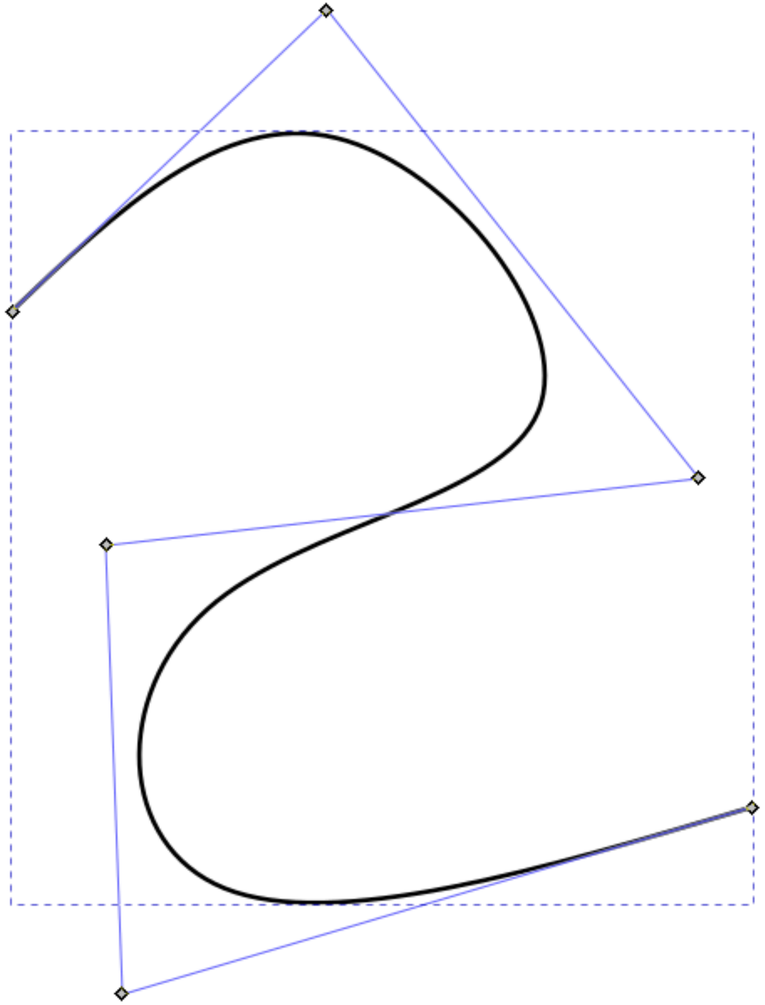


Raster graphics

Vector graphics

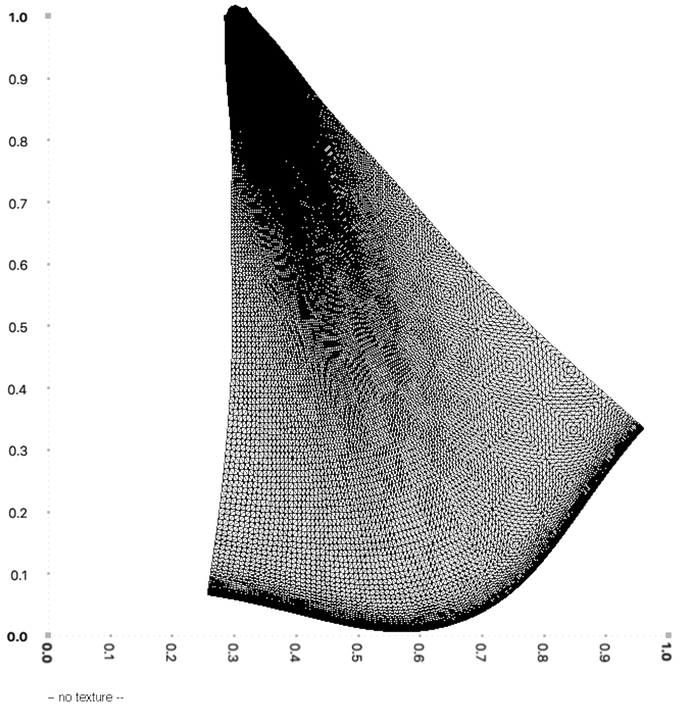
Vector Graphics

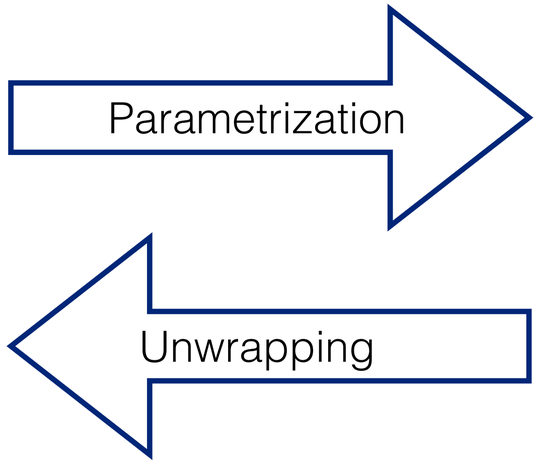
- Specify graphics drawings as composed of *vector primitives*
- A vector primitive is defined with:
  - A set of *control points* in a *reference system*
  - Possible additional *parameters*
  - *Math* to uniquely identify the primitive from the control points and the parameters

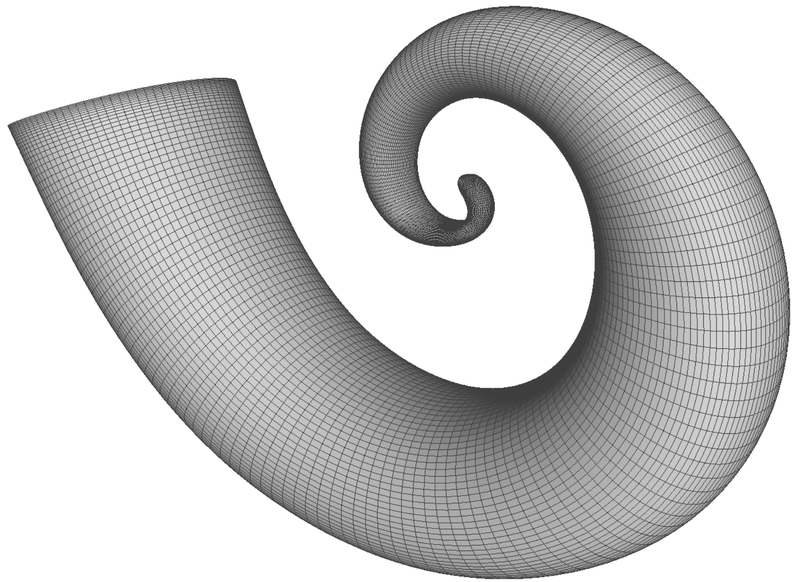


Vector Graphics

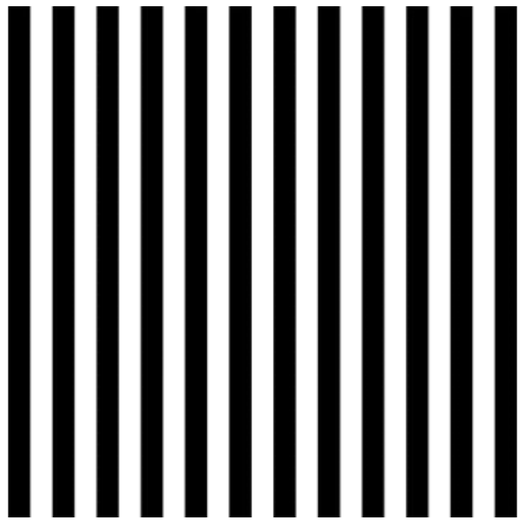
The traditional way: parametrization





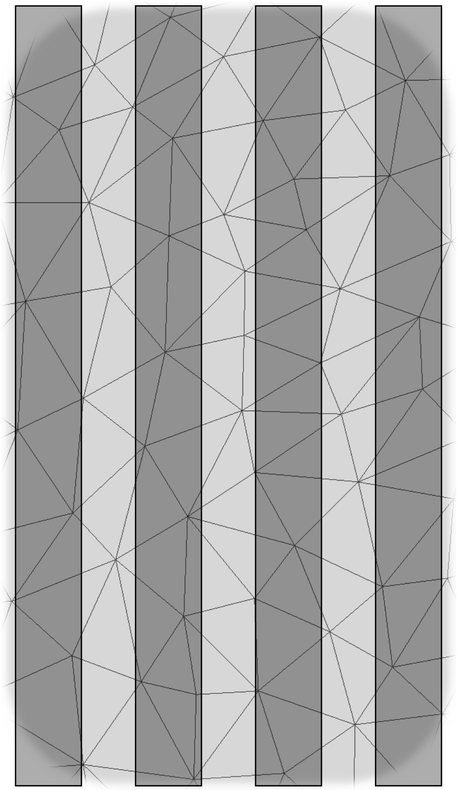


Vector Graphics

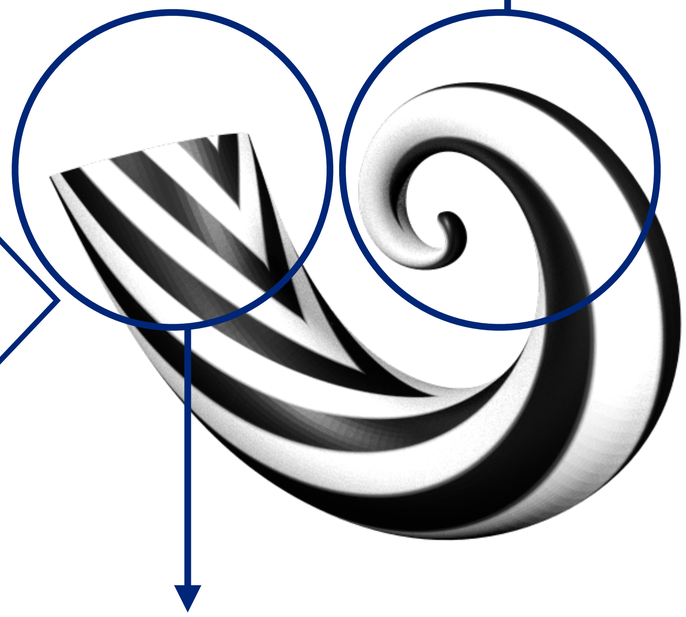


Drawing in 2D

parameter space



map to the mesh



Vector Graphics

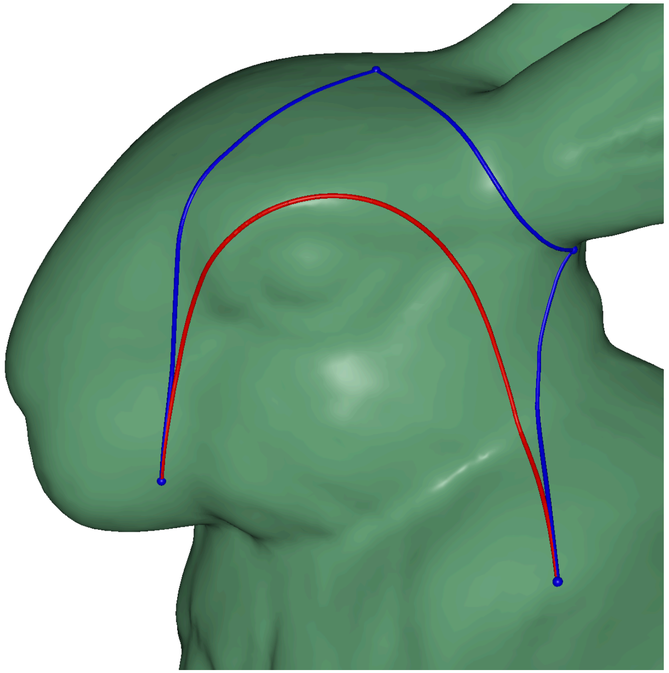
- Examples (in Euclidean space):
  - A straight line segment is defined by two control points p and q and parametric equation $p + \alpha(q - p) \quad \alpha \in [0, 1]$
  - A circle is defined by point c radius r and equation $|p - c|^2 = r^2$
  - A cubic Bézier curve is defined by four control points p_0, p_1, p_2, p_3 and parametric equation

$$\mathbf{b}(t) = \sum_{i=0}^{3} B_i^3(t) p_i$$

where the $B_i^3(t)$ are the Bernstein polynomials of degree 3

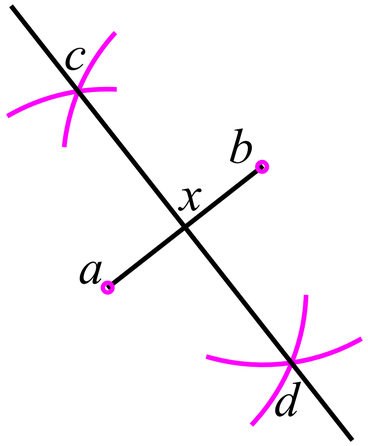
Vector graphics on a mesh

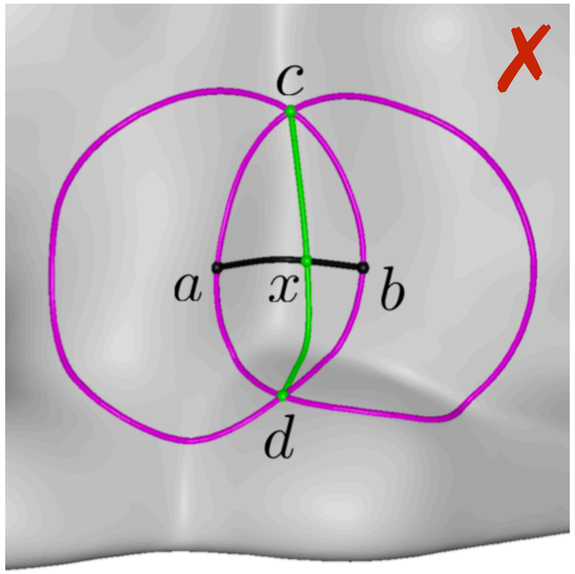
- Can we extend vector graphics to a manifold domain, e.g., a surface mesh?
- We should redefine everything under the *geodesic metric*:
  - Straight line segments can be replaced with geodesic lines
  - Geodesic circles? Ellipses?
  - What about angles? Regular shapes? Parallel lines?
  - Bézier curves and other splines?
  - Filling polygons? Gradients? Patterns?



Vector graphics on a mesh

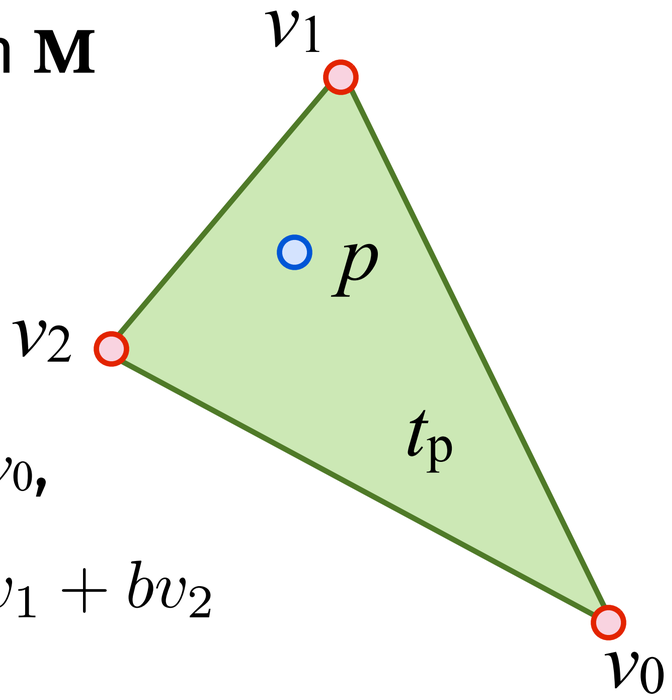
- Challenges:
  - Most of the theorems of Euclidean geometry no longer hold under the geodesic metric (intrinsic problem)
  - A unique reference system cannot be defined in general (intrinsic problem)
  - Equations in closed form are no longer available (computational problem)
- This is an active area of research
  - In particular: our current research





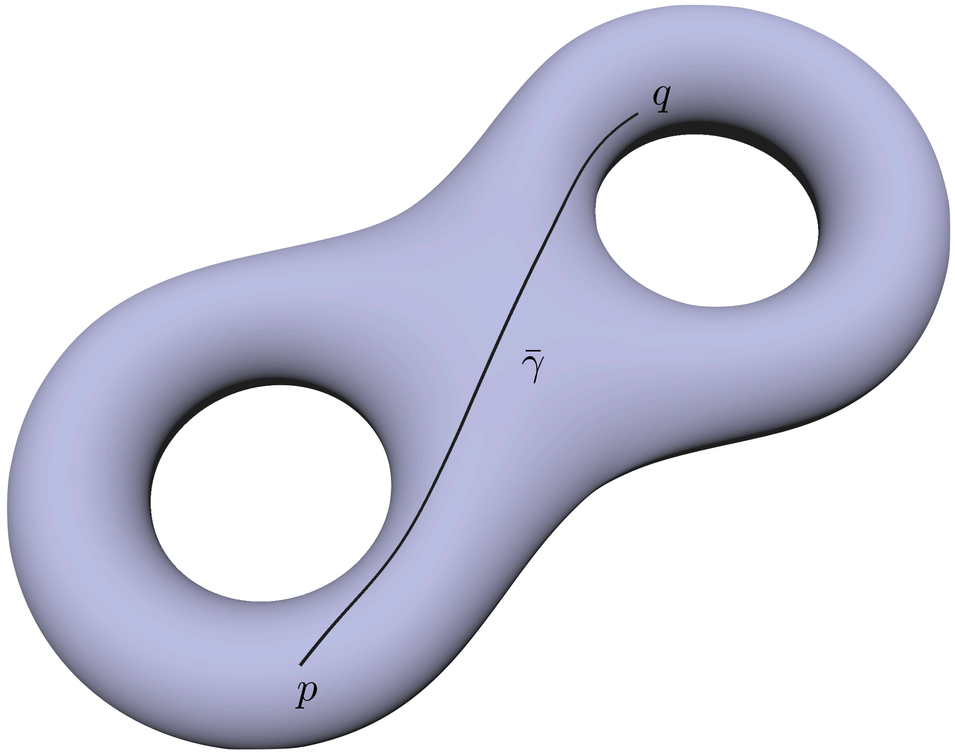
Referencing points on a mesh

- We can assign *mesh coordinates* to all points of a mesh $\mathbf{M}$
- Point p has coordinates (t_p, a_p, b_p) where:
  - t_p is the triangle of $\mathbf{M}$ containing p
  - a_p, b_p are the barycentric coordinates of p in t, i.e., if v_0, v_1, v_2 are the vertices of t then $p = (1-a-b)v_0 + av_1 + bv_2$
- A tuple of coordinates uniquely identifies a point of $\mathbf{M}$
- Coordinates are unique except at edges and vertices of $\mathbf{M}$



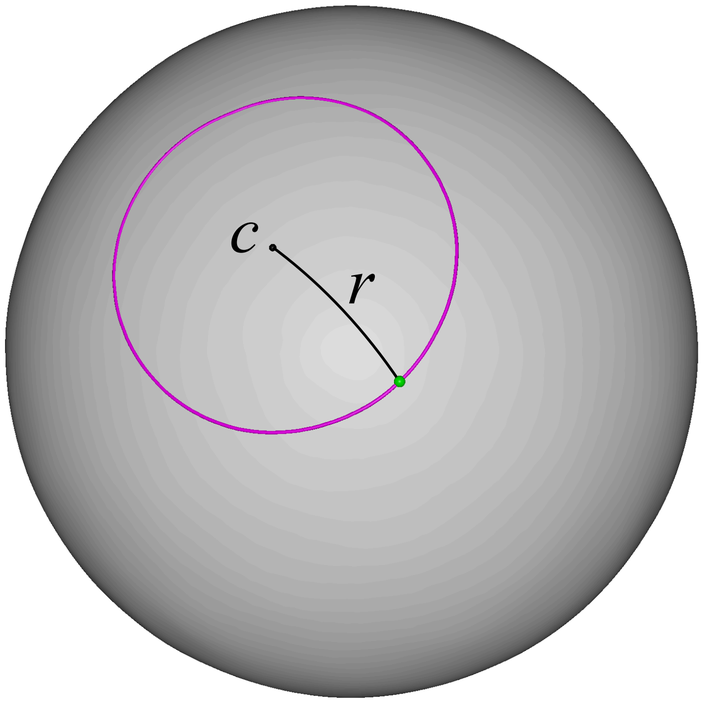
Simple vector primitives

- Geodesic line segment:
  - defined by a pair of points p, q
  - we assume this is the shortest geodesic joining them and it is unique: it can be computed efficiently with PPGP
  - ambiguous if p and q lie on each other's cut locus
  - what if we wish to connect them with a geodesic that is not a shortest path?



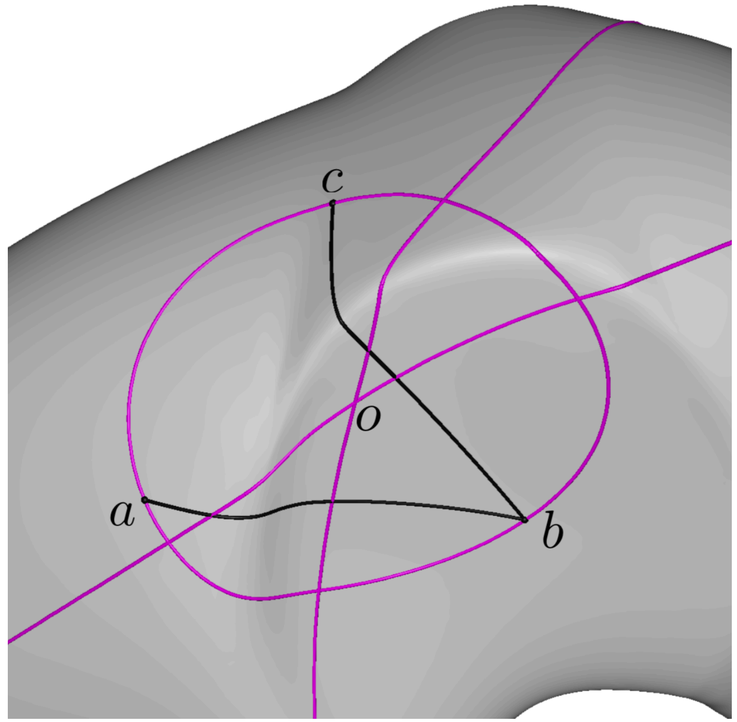
Simple vector primitives

- Geodesic circle:
  - defined by center c and radius r:
    1. SSGD from c gives distance field d_c at all vertices of $\mathbf{M}$
    2. extract isoline at value r by linearly interpolating d_c inside triangles
       - each triangle having vertices both above and below the given isovalue intersects the isoline in a straight-line segment



Simple vector primitives

- Geodesic circle:
  - defined by three points p_1, p_2, p_3:
    1. compute the distance fields from p_1, p_2, p_3 with SSGD
    2. find the point c where the three distance fields have the same value and the corresponding value r
    3. find circle centered at c of radius r with previous method
  - pitfalls: step 2 may be tricky and have more than one solution



Simple vector primitives

- Ellipse:
  - defined by foci p_1, p_2 and “radius” r, where r is the distance between a focus and the intersection between the ellipse and its transverse axis:
    - the ellipse is the locus of points x s.t. $d_{p_1}(x) + d_{p_2}(x) = 2r$
      1. compute distance fields d_{p_1} and d_{p_2} and their sum
      2. find isoline at value $2r$ of the sum field
    - not the most friendly definition of an ellipse…
    - … but the only one easily controllable with distances

Simple vector primitives

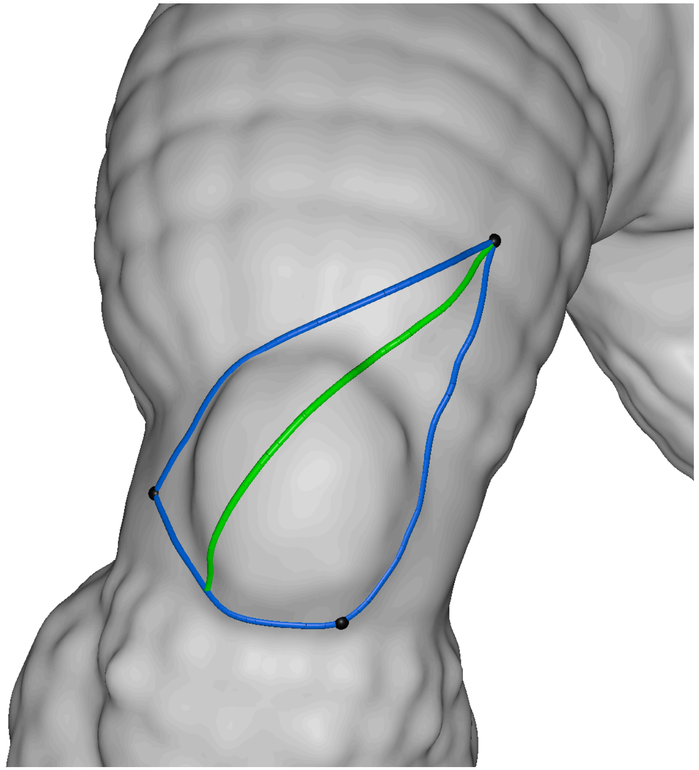
- Ellipse:
  - defined by major and minor axes l_1, l_2: lots of troubles
    - l_2 must be orthogonal to l_1 and cross it at its midpoint. How to compute it?
    - once we have l_1, l_2, we may apply Euclidean computations to obtain the distance of foci on l_1 and their distances from an endpoint of l_2 to compute the “radius”
    - applying the previous approach to the resulting data does not guarantee that the ellipse goes through the endpoints of l_1, l_2
  - defined by enclosing rectangle: similar, but even more troubles (see next)…

Simple vector primitives

- Triangle:
  - defined by its vertices p_1, p_2, p_3:
    1. connect the vertices with shortest geodesics with PPGP
    2. find the interior of triangle by flooding the surface from such geodesics
  - step 2 is tricky and has pitfalls:
    - orientation defines on which side of geodesics flood must occur
    - the cycle formed by the three geodesics might not disconnect the surface (only on objects of genus >0)

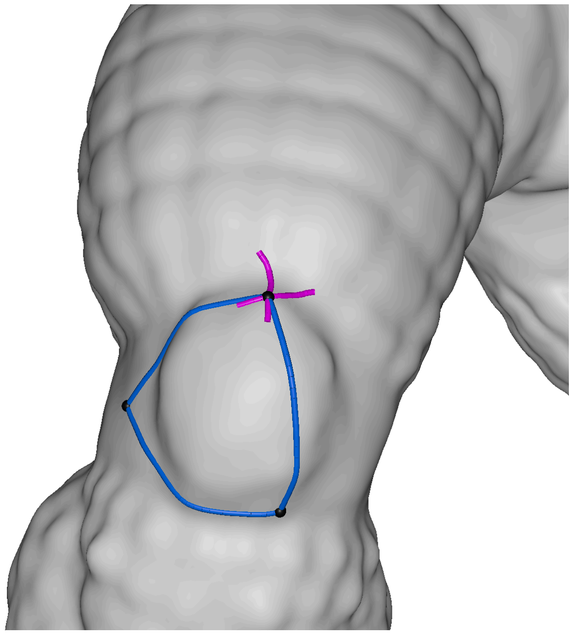
Simple vector primitives

- Isosceles triangle:
  - defined by basis and height
  - defined by basis and angle at basis
  - defined by basis and length of diagonal edges
- Different constructions are possible, but only one of the following properties can be guaranteed:
  A. Diagonal edges have equal length
  B. Angles at the basis are equal
  C. Projection of the apex to the basis crosses it at its midpoint



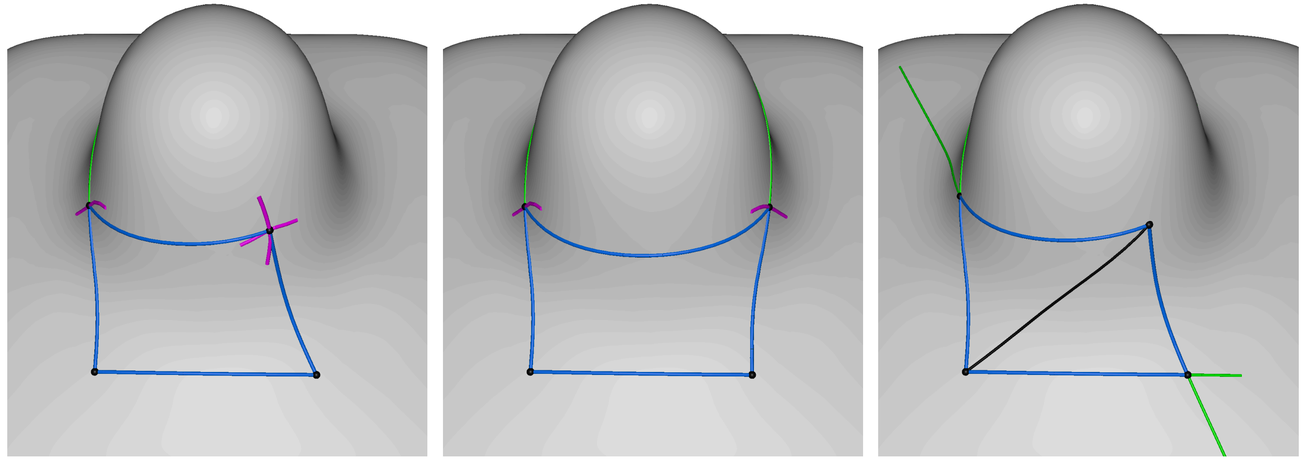
Simple vector primitives

- Equilateral/equiangular triangle: defined by its basis
- Different constructions are possible, but only one of the following properties can be guaranteed:
  A. All edges have equal length
  B. All angles are equal (very difficult!)
- In neither case, it is guaranteed that the projection of the apex to the basis crosses it at its midpoint



Simple vector primitives

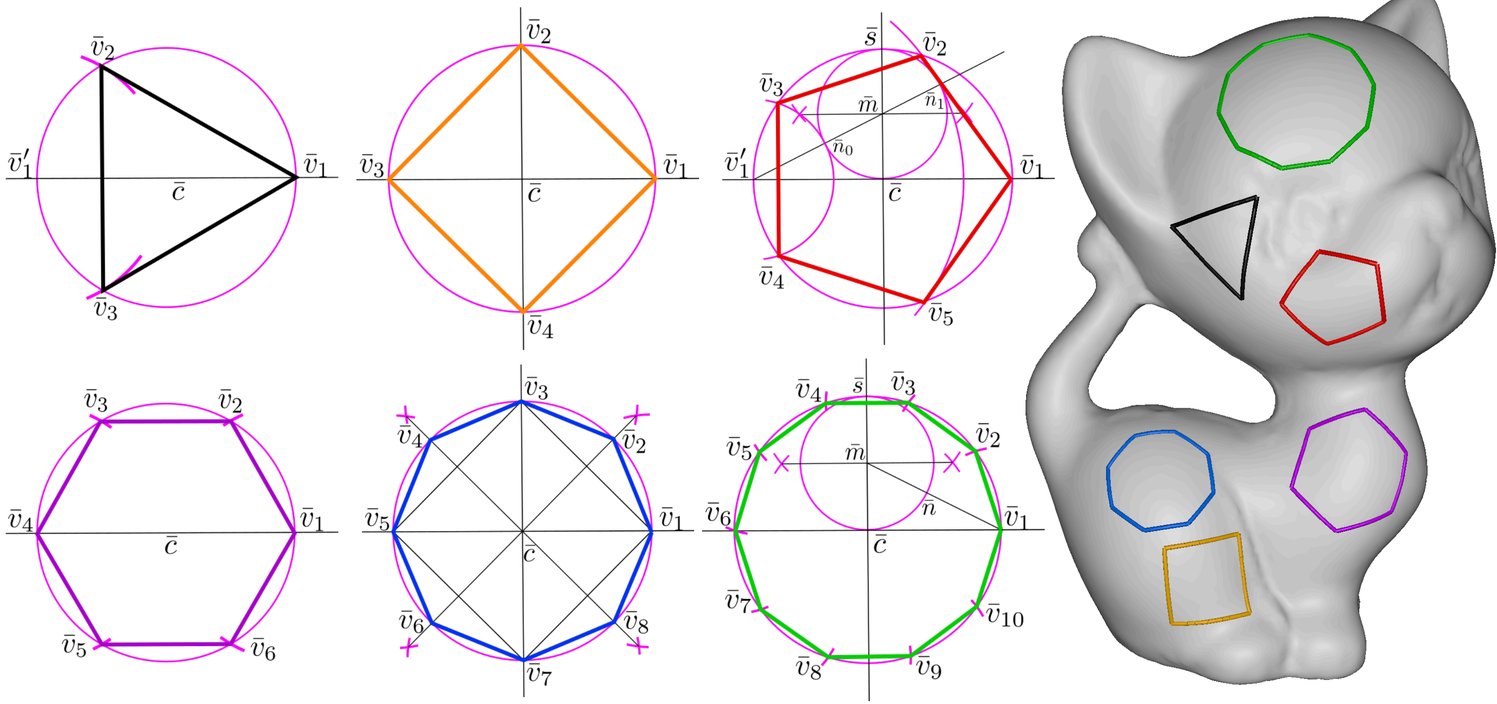
- Rectangle:
  - several possible definitions; more troubles than with triangles
  - if edges are geodesics, the basic properties of Euclidean triangles *cannot* be guaranteed:
    A. opposite edges have equal length
    B. all consecutive edges are mutually orthogonal
    - one may try to define rectangles in which all four angles are equal (very difficult!)



Simple vector primitives

Constructions in tangent space

- Use the Euclidean construction in the tangent space
- map the vertices onto the surface (mesh) using the exp map (straightest geodesic)
- connect the obtained points with shortest paths

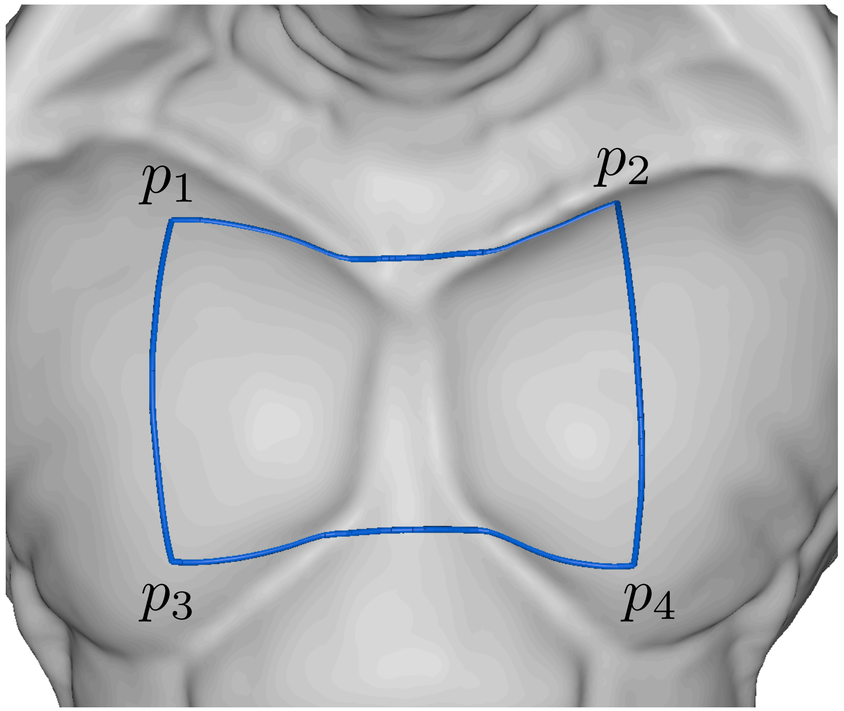


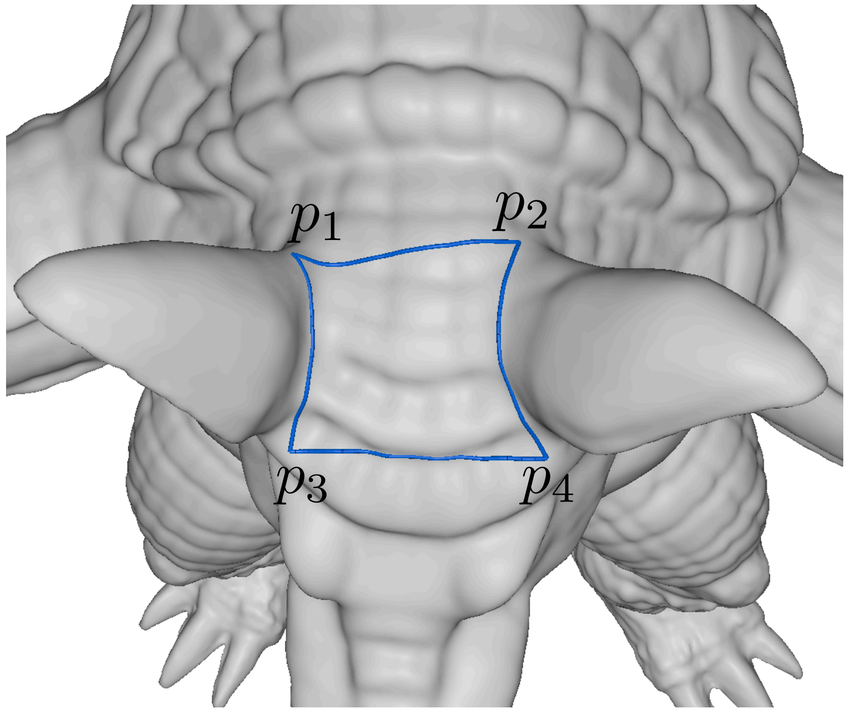
Note: sides and angles are not guaranteed to be equal!

Simple vector primitives

(Local) Gauss-Bonnet theorem

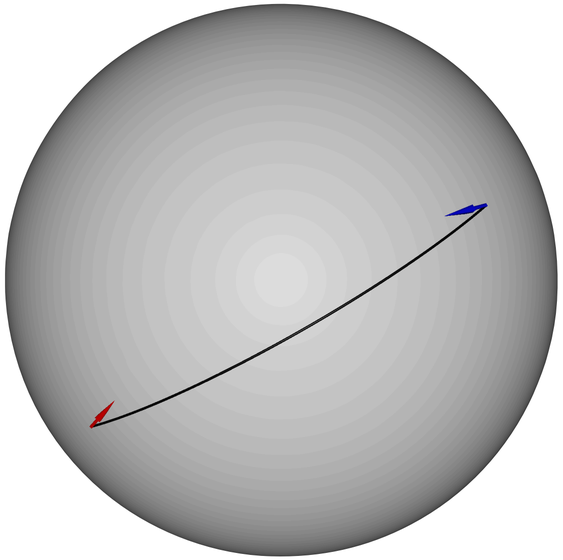
$$\int_{\partial R} \kappa_g ds + \iint_R K d\sigma + \sum_{i=0}^{k} \theta_i = 2\pi,$$

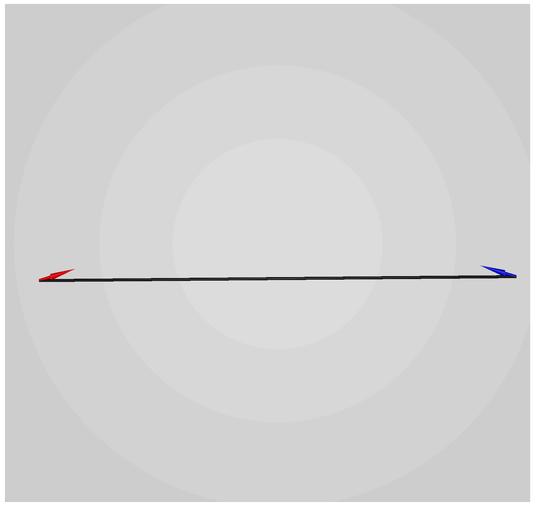




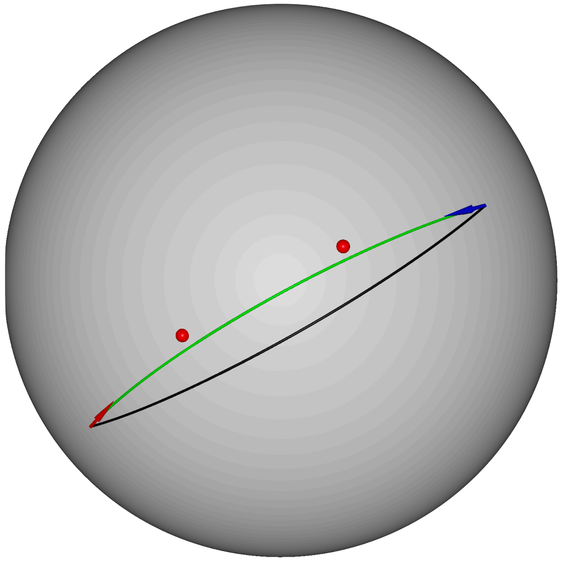
Simple vector primitives

Shortest path and prescribed tangents

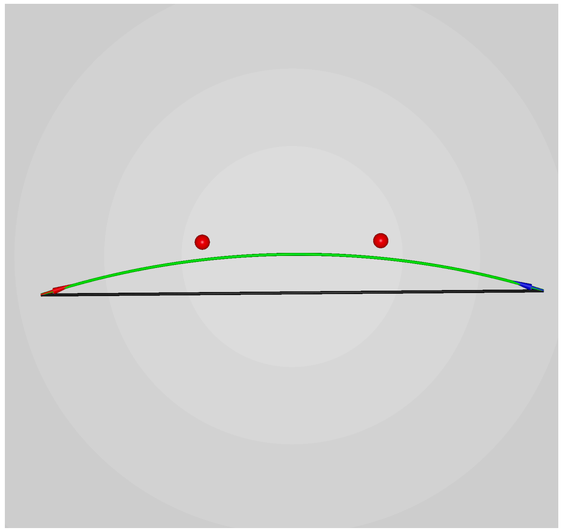




Same length, same angles



Handle points at same distance define manifold Bézier curve

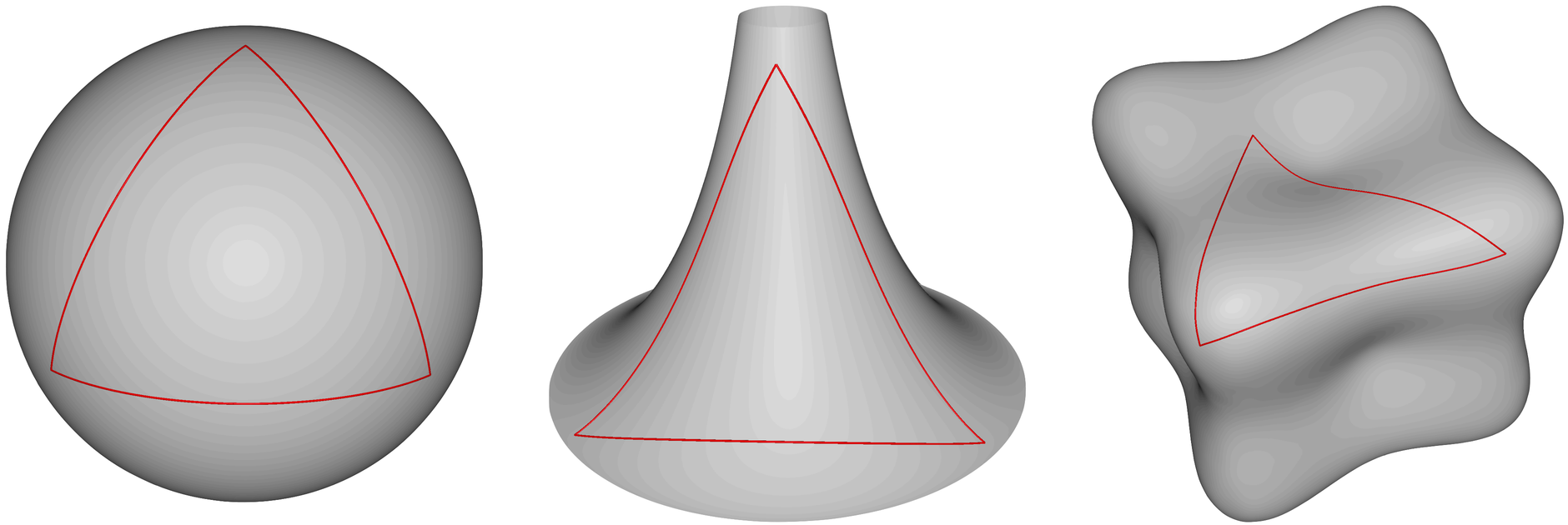


Handle points (red) of optimal curve (green)

Simple vector primitives

(Local) Gauss-Bonnet theorem

$$\int_{\partial R} \kappa_g ds + \iint_R K d\sigma + \sum_{i=0}^{k} \theta_i = 2\pi,$$



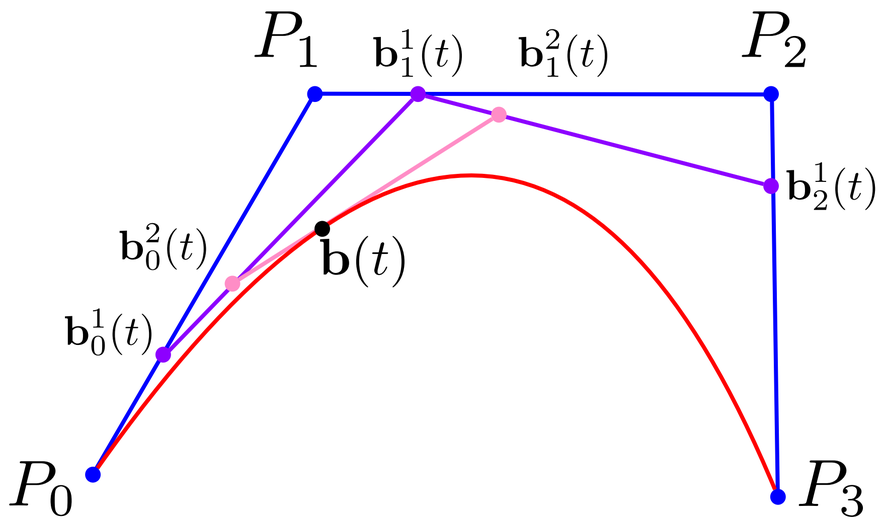
Bézier curves

Bernstein polynomials

$$\mathbf{b}^k(t) = \sum_{i=0}^{k} B_i^k(t) P_i \qquad t \in [0,1]$$

de Cateljeau's algorithm

$$\mathbf{b}_i^0(t) = P_i$$
$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t)$$



Bézier curves

Euclidean setting

$$\mathbf{b}^k(t) = \sum_{i=0}^{k} B_i^k(t) P_i \qquad t \in [0,1]$$

Manifold setting

$$\mathbf{b}^k(t) = \arg\min_{P \in M} \sum_{i=0}^{k} B_i^k(t) d^2(P, P_i)$$

Bézier curves

$$\mathbf{b}^k(t) = \arg\min_{P \in M} \sum_{i=0}^{k} B_i^k(t) d^2(P, P_i)$$

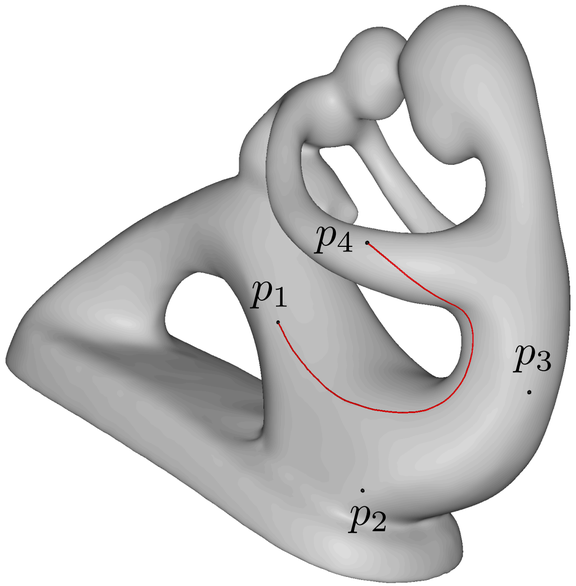
Pros

- wide variety of splines
- can be used to addressed other problems

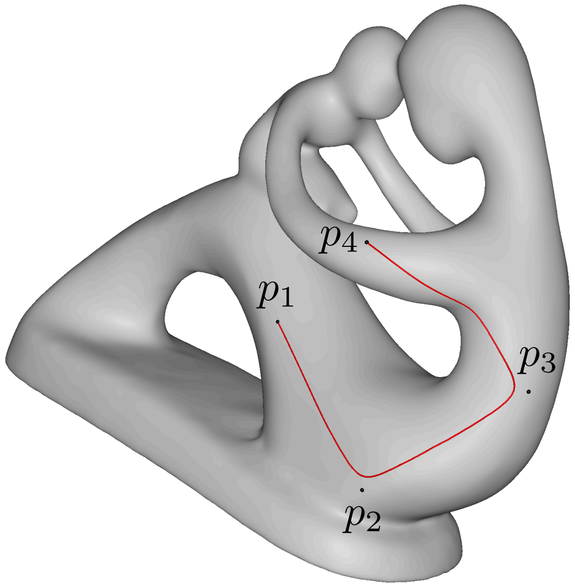
Cons

- minimization problem
- hard constraints on the position of the control points
- quite expensive

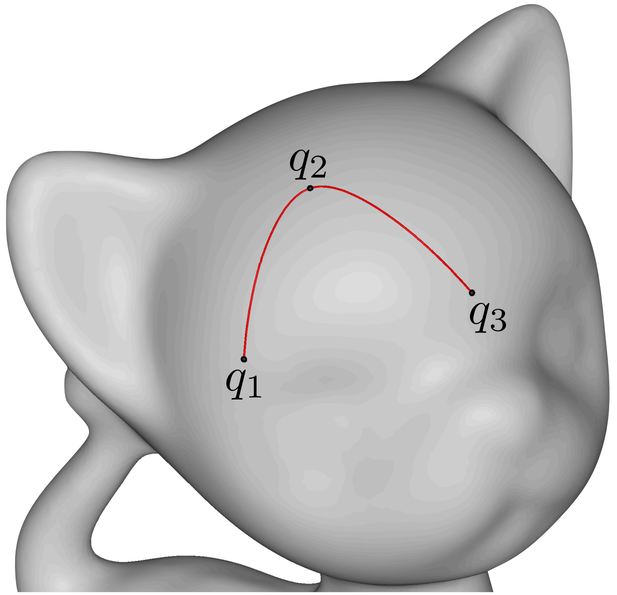
Bézier curves

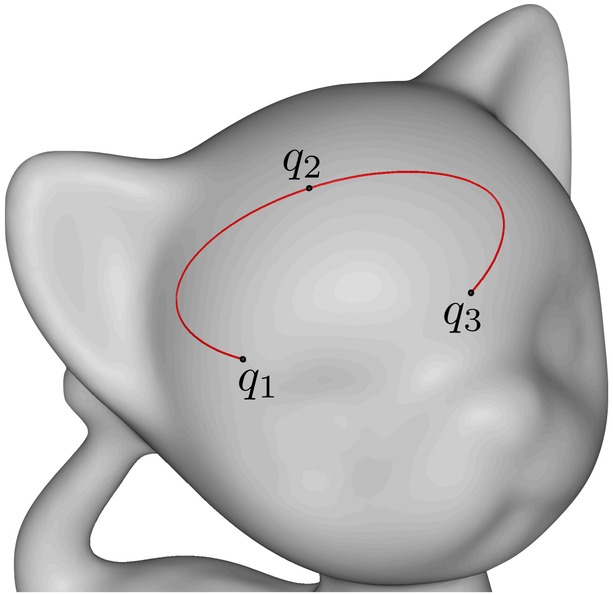


Bézier curve



rational Bézier curve





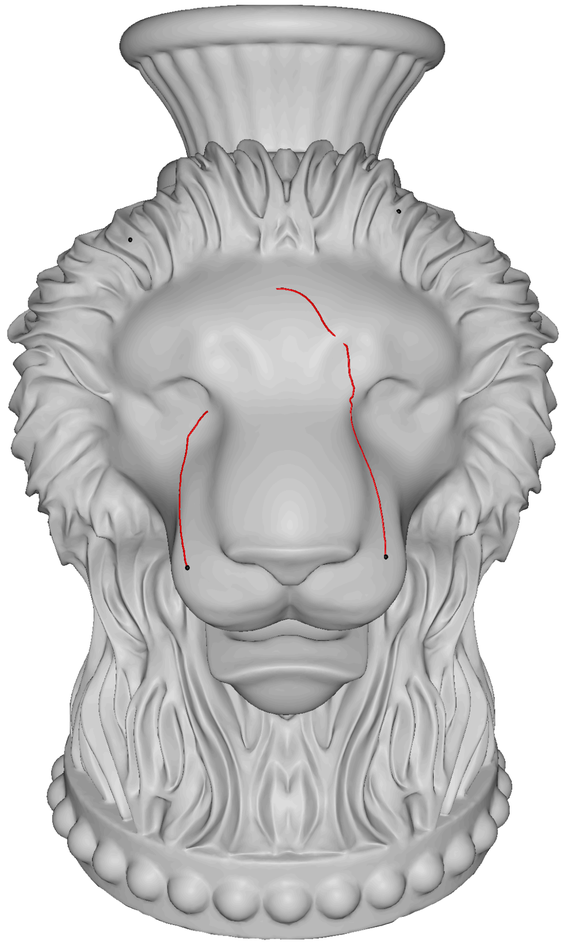
Bézier interpolant

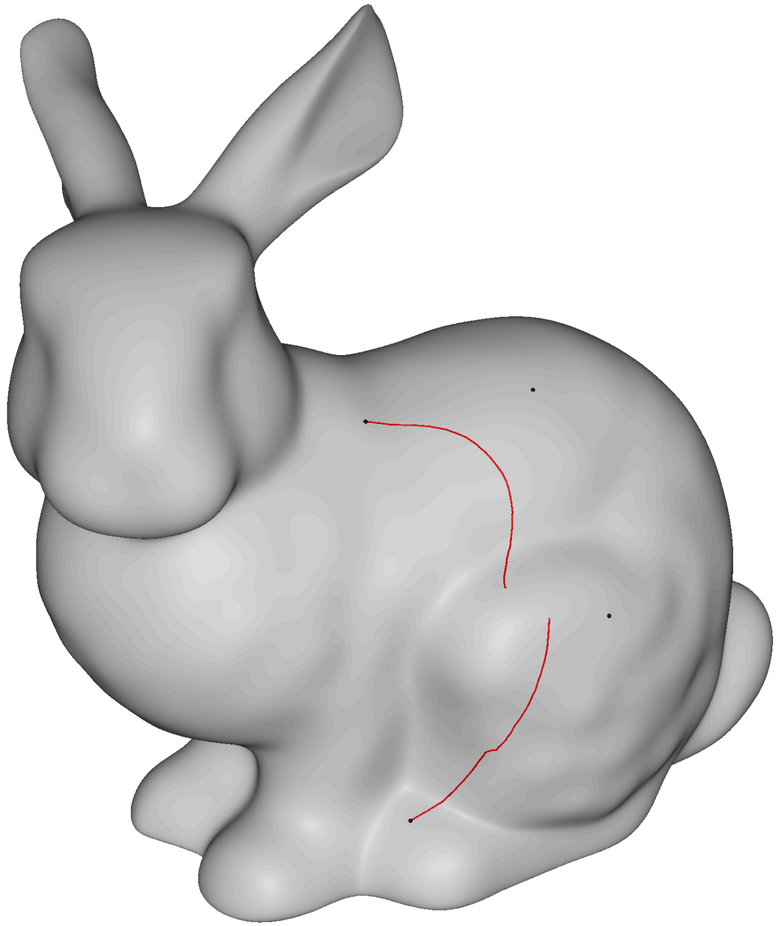
[Ramantantoanina and Hormann, 2021]

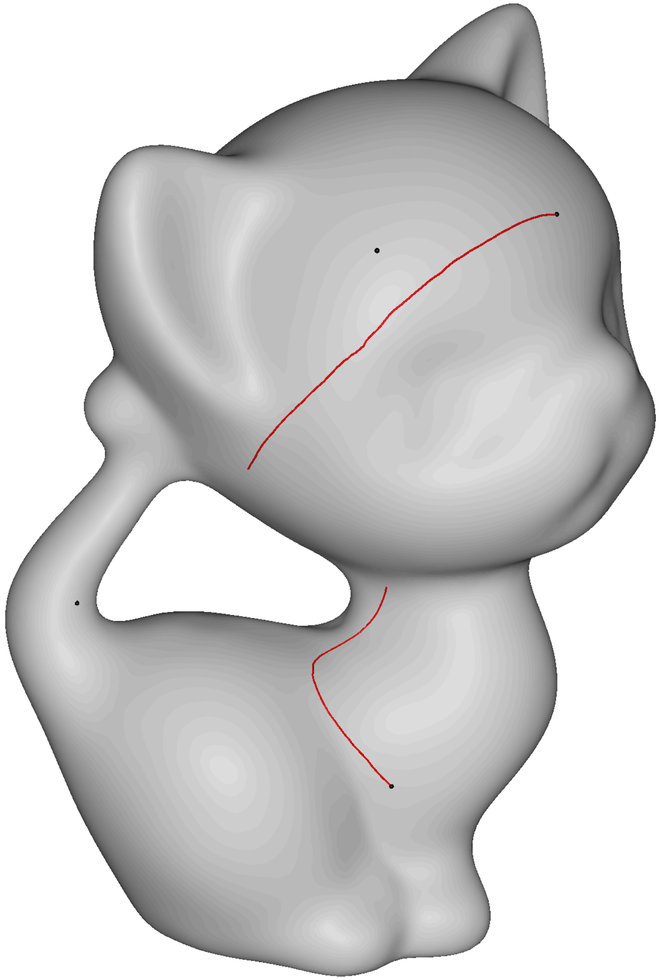
[Mancinelli and Puppo, 2023]

Bézier curves

If the control points are “too far” from each other, the minimum of the energy is no longer guaranteed to be unique. This causes “broken” curve.







Bézier curves

We define the *manifold average operator* between two points as

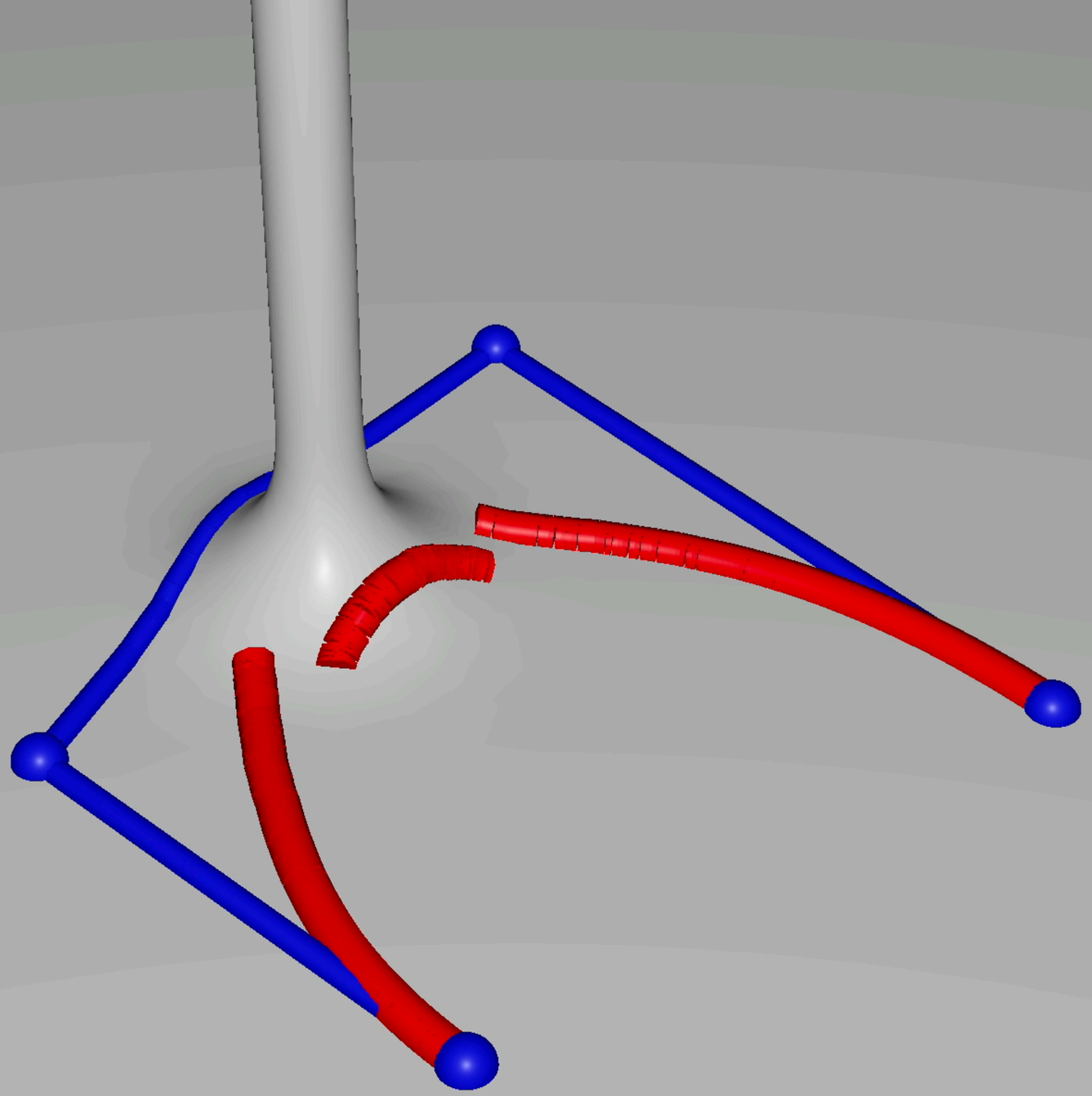
$$\mathcal{A} : M \times M \times [0,1] \longrightarrow M; \quad (P, Q; w) \mapsto \gamma_{P,Q}(w)$$

Always well defined except at the cut locus

$$\mathbf{b}_i^0(t) = P_i$$

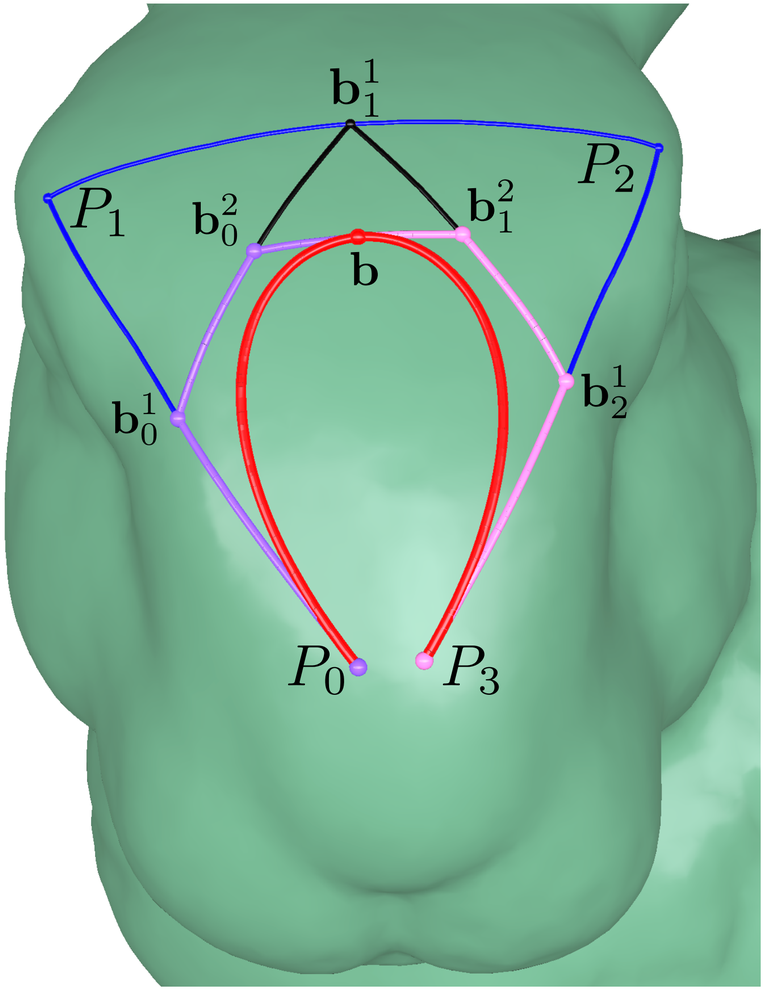
$$\mathbf{b}_i^r(t) = (1-t)\,\mathbf{b}_i^{r-1}(t) + t\,\mathbf{b}_{i+1}^{r-1}(t)$$

[Park and Ravani, 1995]



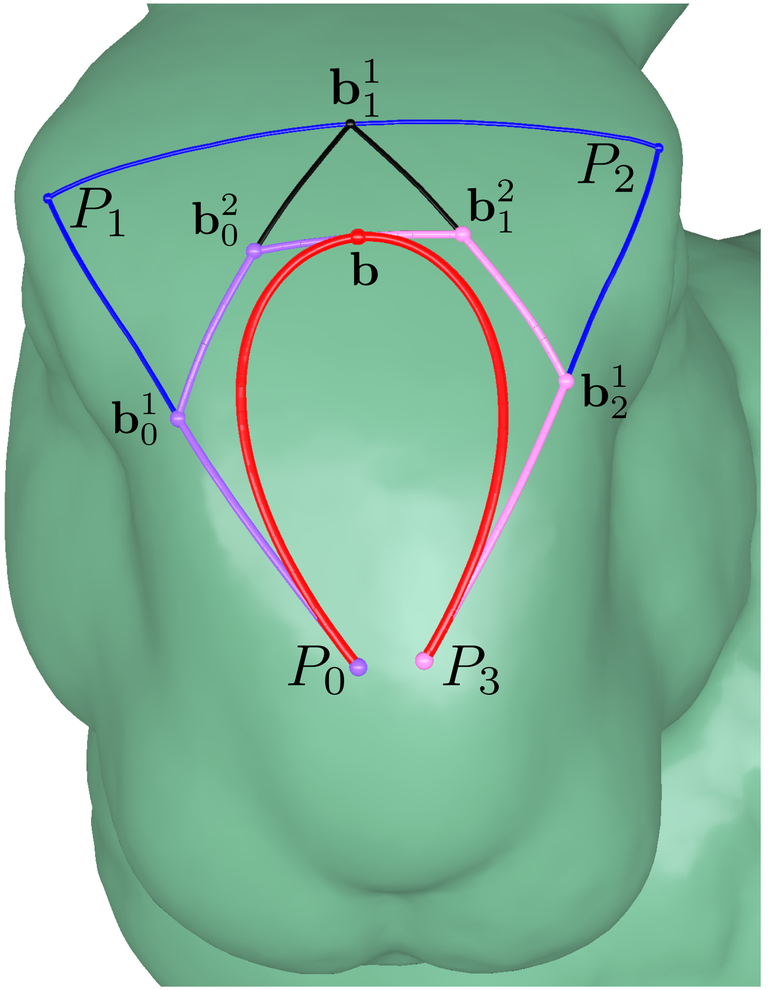
Bézier curves

- Construction with recursive De Casteljau [Mancinelli et al. 2021]
- Example for a cubic curve
- Straight-line segments are replaced with shortest geodesics computed with PPSP



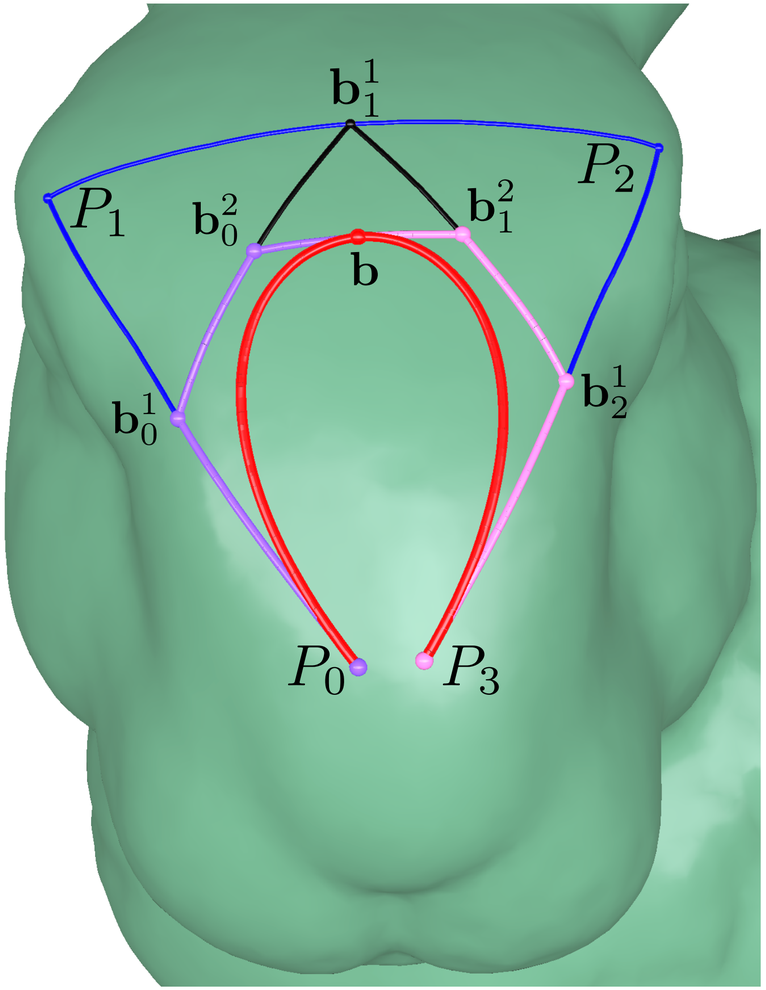
Bézier curves

- Given the initial control polygon (blue):
  1. build the De Casteljau construction for evaluating the midpoint of the curve:
     - the control polygon is substituted with a chain of two control polygons (violet and pink) joining at the midpoint
  2. apply Step 1 recursively to each sub-polygon
  3. stop when the control chain is “straight” enough

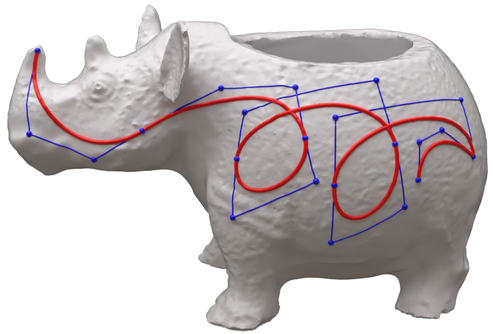


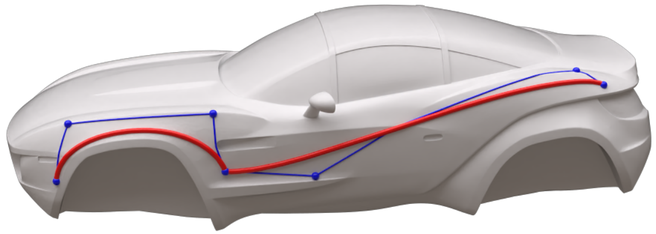
Bézier curves

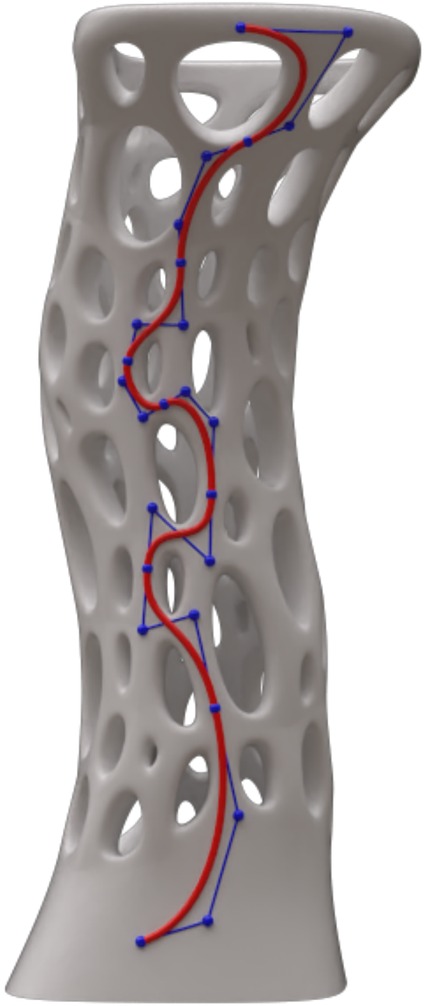
- Each recursion step requires:
  1. finding the midpoint of three existing geodesics $\mathbf{b}_0^1, \mathbf{b}_2^1, \mathbf{b}_1^1$
  2. computing geodesics $\mathbf{b}_0^1\mathbf{b}_1^1$ and $\mathbf{b}_2^1\mathbf{b}_1^1$ and their midpoints $\mathbf{b}_0^2, \mathbf{b}_1^2$
  3. computing geodesic $\mathbf{b}_0^2\mathbf{b}_1^2$ and its midpoint $\mathbf{b}$
  4. halving five existing geodesics



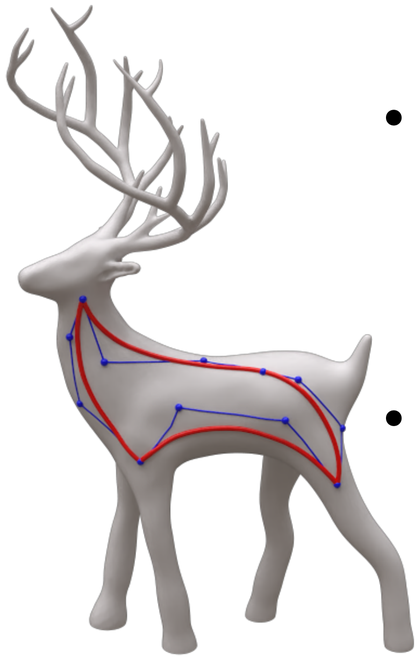
Bézier curves







- Fast and robust:
  - Procedure is always guaranteed to converge
  - Drawing and editing work in interactive time on meshes with millions of triangles and arbitrarily complex shape and genus
  - Support to splines of Bézier segments with smooth or sharp continuity at junction points



Bézier curves

- Construction with recursive De Casteljau [Morera et al., 2008; Mancinelli et al. 2021]
  - Example for a cubic curve
  - Straight-line segments are replaced with shortest geodesics computed with PPGP
  - Given the initial control polygon:
    1. build the De Casteljau construction for evaluating the midpoint of the curve:
       - this construction splits the control polygon into a chain of two control polygons joining at the midpoint
    2. apply Step 1 recursively to each sub-polygon

b/Surf: Interactive Bézier Splines on Surfaces

Thank you