CSE 413
(Computer Graphics)
Curves and Surface Design

Iyolita Islam

Department of Computer Science and Engineering
Military Institute of Science and Technology

Last Updated: November 5, 2020

Unintentional Mistakes

Best efforts have been exercised in order to keep the slides error-free, the preparer does not assume any responsibility for any unintentional mistakes. The text books must be consulted by the user to check veracity of the information presented.

Outline

3D Objects Representation

- Solid Modeling attempts to develop methods and algorithms to model and represent real objects by the computer.
- There are three types of objects in 3D: 1D, 2D and 3D
- Objects in 3D can be modeled in two ways:
  - Modeling of the existing objects
    - mathematical description may be unavailable.
    - not precise
    - need to approximate the object with pieces of planes, spheres or any other known shapes to describe mathematically.
  - Modeling of a new object from the scratch
    - no preexisting object to model
    - user creates the object in modeling process
    - precise
    - can be easily described mathematically
    - need to generate physical realizations

Polygon Mesh

- A set of connected polygonally bounded planar surfaces.
- Open box, cabinet, building exterior can be naturally and easily represented by polygon meshes as the volume is bounded by planar surfaces.
- can be used to represent objects with curved surfaces but this is approximate.
- The obvious errors can be reduced by using more and more polygons:
    - it will increase the space and time consumption of the algorithm.
    - enlarging the image the straight lines will be visible.

3D Objects Representation

Fig. 11.1 A 3D object represented by polygons.

Fig. 11.2 A cross-section of a curved object and its polygonal representation.

Polygon Mesh

- Polygon: closed sequence of edges
- Polygon Mesh: A collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons.
- Polygon mesh can be represented in many ways.
- need to choose the appropriate one evaluating on: space and time.
- Typical operation of polygon mesh:
    - finding all the edges incident to the vertex
    - finding the polygons sharing an edge or vertex
    - finding the vertices connected by an edge
    - displaying the mesh
    - identifying the errors in representation (i.e: missing edge, vertex, polygon)
- The more explicitly the relations among the polygons, vertices and edges are represented, the faster the operations are and the more space and time requires.

Polygon Mesh Representation - Explicit

- each polygon is represented by a list of vertex coordinates: $P = ((x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n))$
- sorted in the order of being encountered travelling around the polygon.
- edges between successive vertices and between the last and the first one.

Explicit Representation

- For a single polygon, representation is space-efficient.
- But not for polygon mesh since the coordinates for the shared vertices are duplicated.
- Even worse, there is no explicit representation of the shared vertices and edges.
- For instance, to drag a vertex and all its incidents edges, we need to find all polygons that share this vertex.
- This require to compare the coordinate triples of one polygon with those of all other polygons.
- it can be performed efficiently by sorting all N coordinate triples, but in best case it is $Nlog_2N$ process.
- Even there is a chance that due to computational roundoff, same vertex might have different coordinate values in different polygons.

Explicit Representation

- displaying a mesh either as filled polygons or as polygon outlines requires to transform each vertex and to clip each edge of each polygon.
- To draw edges, each shared edge is drawn twice.
- This causes problems due to overwriting.
- It can intensify extra pixels also.

Polygon Mesh Representation - Pointer to a vertex list

- each vertex of the polygon is stored once in the vertex list:
  $V = ((x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n))$
- A polygon is defined by a list of indices or pointers into the vertex list.
- A polygon made up of vertices 3, 5, 7, and 10 in the vertex list is represented as:
  P = (3, 5, 7, 10)

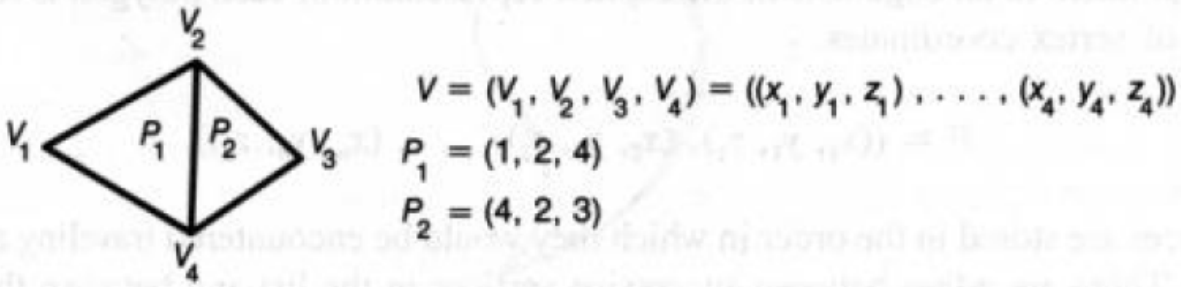


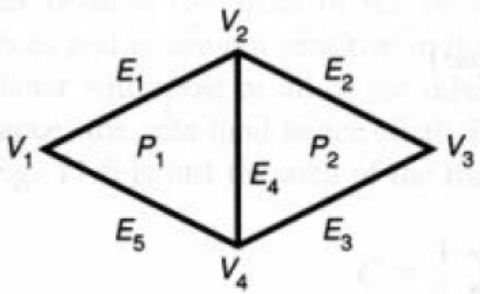
Fig. 11.3 Polygon mesh defined with indexes into a vertex list.

Pointer to a vertex list Representation

- Since each vertex is stored just once, considerable space is saved.
- The coordinates of a vertex can be changed easily.
- But it is difficult to find the polygons that share edge.
- And shared polygon edges are drawn twice while displaying all polygon outlines.

Polygon Mesh Representation - Pointer to an edge list

- each vertex of the polygon is stored once in the vertex list: $V = ((x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n))$
- A polygon is defined by a list of indices or pointers to the edge list.
- A polygon is represented as: $\mathrm{P} = (E_1, E_2, ..., E_n)$
- An edge is represented as: $\mathrm{E} = (V_1, V_2, P_1, P_2)$
- When an edge belongs to only one polygon, one of P_1 and P_2 is null.



$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4))$
$E_1 = (V_1, V_2, P_1, \lambda)$
$E_2 = (V_2, V_3, P_2, \lambda)$
$E_3 = (V_3, V_4, P_2, \lambda)$
$E_4 = (V_4, V_2, P_1, P_2)$
$E_5 = (V_4, V_1, P_1, \lambda)$
$P_1 = (E_1, E_4, E_5)$
$P_2 = (E_2, E_3, E_4)$

Fig. 11.4 Polygon mesh defined with edge lists for each polygon (λ represents null).

Polygon Mesh Representation - Pointer to an edge list

- Polygon outlines are shown by displaying all edges, rather than by displaying all polygons.
- So, redundant clipping, transformation and scan conversion are avoided.
- Filled polygons are displayed easily.
- In some cases like- 3D honecomblike sheet-metal structure, an edge can be shared by three polygons.
- Then, edge description can be:
  $E = (V_1, V_2, P_1, P_2, ..P_n)$

Polygon Mesh Representation

- Polygon meshes are generated interactively (operators digitizing drawings). So, errors are inevitable.
- So, it is appropriate to make sure
  - all polygons are closed, all edges are used at least once but not more than maximum.
  - each vertex is referenced by at least two edges.
- In some applications, the mesh is to be
  - completely connected: any vertex can be reached from any other vertex moving along the edges.
  - topologically planar: the binary relation on vertices defined by edges can be represented as planar graph
  - to have no holes: just one boundary - a connected sequence of edges each of which is used by one polygon.
- The explicit-edge scheme is the easiest to check for consistency.

Check

```
for (each edge E(j) in set of edges)
    use_count(j) = 0;
for (each polygon P(i) in set of polygons)
    for (each edge E(j) of polygon P)
        use_count(j)++;
for (each edge E(j) in set of edges)
{
    if (use_count(j) ==0)
        Error();
    if (use_count(j) > maximum)
        Error();
}
```

Polygon Mesh Representation

- Explicit-edge scheme can't ensure complete consistency check.
- An edge used twice in the same polygon remains still undetected.
- To ensure that each vertex is part of at least one polygon, we check whether at least two different edges of the same polygon refer to the vertex.
- It should be an error also for the two vertices of an edge to be the same (unless zero length of the edges are allowed).

Connected Components

- The relationship of sharing an edge between polygons is a binary equivalence relation and partitions a mesh into equivalence classes called *Connected Components*.
- Usually, a polygon mesh contains a single connected component.

Plane Equations

- While working with polygon meshes, the equation of the plane is important in which the polygon lies.
- In some cases, the equation is known implicitly through the interactive construction methods used to define the polygon.
- Otherwise, we can use three vertices on the plane and find: $Ax + By + Cz + D = 0$
- The coefficients define the normal to the plane [A, B, C].
- Given point P_1, P_2, P_3 on the plane. Normal to the plane can be computed as: $P_1P_2 \times P_1P_3$
- If the product is zero, then the points are co-linear and do not define any plane.
- In that case, other vertices are to use.
- Given a non-zero cross product, D can be found by substituting the normal [A B C] and any one of the three points into the equation of the plane.

Plane Equations: Another technique

- It can be shown that A, B and C are proportional to the signed areas of the projections of the polygon onto the YZ , XZ and XY planes respectively.
- For example, if the polygon is parallel to XY plane, $A = B = 0$.
- The projection of the polygon on YZ and XZ planes have zero area.
- This method is better because the areas of the projections are a function of the coordinates of all the vertices.
- And so are not sensitive to the choice of a few vertices that might not to be coplanar with most or all the vertices or might be colinear.

Plane Equations

- For example, the area (and coefficient) C of the polygon projected onto the XY plane is just the area of the trapezoid A_3, minus the areas of A_1 and A_2.

$$C = \frac{1}{2}\sum_{i=1}^{n}(y_i + y_{i\oplus 1})(x_{i\oplus 1} - x_i)$$

  where, $\oplus$ is normal addition except that $n \oplus 1 = 1$

- The area for A and B are calculated in same procedure, except the area for B is negated.

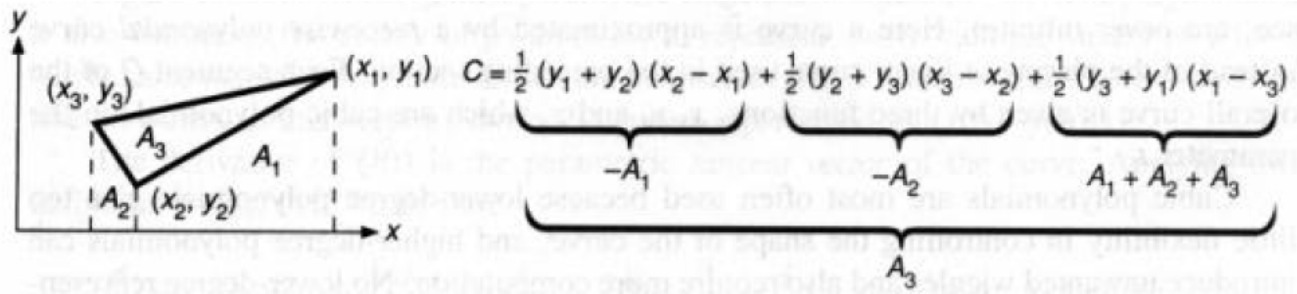


Fig. 11.6 Calculating the area *C* of a triangle using Eq. (11.2).

Plane Equations

- The equation for C gives the sum of the area of all trapezoids firmed by successive edges of the polygons.
- The sign of the sum is also useful.
- If the vertices have been enumerated in a clockwise direction (as projected onto the plane), the sign is positive, otherwise, it is negative.
- After determining the plane equation by using all vertices, it can be estimated that how nonplanar the polygon is by calculating the perpendicular distance from the plane to each vertex.
- The distance d for the vertex (x, y, z):

$$d = \frac{Ax+By+Cz+D}{\sqrt{A^2+B^2+C^2}}$$

Plane Equations

- The distance d for the vertex (x, y, z):

$$d = \frac{Ax+By+Cz+D}{\sqrt{A^2+B^2+C^2}}$$

- The distance is either positive or negative depending on which side of the plane the point is located.
- $d = 0$, if the vertex is on the plane.
- The equation of the plane is not unique.
- Any nonzero constant k can change the equation but not the plane.
- To store the plane coefficients with normalized normal:

$$k = \frac{1}{\sqrt{A^2+B^2+C^2}}$$

Parametric Cubic Curves

- Polylines and polygons are first degree, piecewise linear approximations to the curves and surfaces respectively.
- For this, a large number of coordinates need to be created and stored to achieve reasonable accuracy.
- The general approach to generate a curve is to use a higher degree of function than to use linear functions.
- It will still approximate the shape but will require less storage and offer easier manipulation.

Explicit equation

- Express y and z as the explicit function of x: $y = f(x)$ and $z = g(x)$
- Problems:
  - It is impossible to get multiple values of y for a single x. So,curves like circles and ellipses must be represented by multiple curve segments.
  - such a definition is not rationally invariant to describe a rotated version of the curves requires a great deal of work and may require a breaking a curve segment into many others.
  - describing curves using vertical tangent is difficult because a slope of infinity is difficult to represent.

Implicit equation

- To model curves as solutions to implicit equations of the form f(x, y, z) = 0
- Problems:
  - The equation may have more solution than needed. For example, to represent a circle we use $x^2 + y^2 = 1$ but to define a half circle there's no way. We need to add constraint like $x \geq 0$ which can't be contained in a implicit function.
  - if two implicitly defined curve segments are joined together, it may be difficult to determine whether their tangent directions agree at the joint point.

Parametric Representation

- $x = x(t)$, $y = y(t)$ and $z = z(t)$.
- A curve is approximated by a piecewise polynomial curve instead of piecewise linear curve.
- Each segment Q of the curve is given by three functions, x, y and z which are cubic polynomials in the parameter t.
- Parametric curves replace the use of geometric slope (can be infinite) with parametric tangent vector (are never infinite)
- Cubic polynomials are used over lower and higher degree polynomials because
    - lower one gives too little flexibility to control the shape of the curve. And higher degree polynomials can introduce unwanted wiggles and also require more computation.
    - lower degree representation allows a curve segment to interpolate (pass through) two specified endpoints with specified derivatives at each endpoint.

Parametric Representation

- Parametric cubics are the lowest degree of curves that are nonplanar in 3D.
- The cubic polynomial to define a curve segment $Q(t) = [x(t)y(t)z(t)]$ are of the form $(0 \leq t \leq 1)$:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

- To deal with finite segments of the curve without loss of generality, we restrict the parameter t to the [0, 1] interval.
- With $\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$ and defining the matrix of the coefficients of the three polynomials as

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

Parametric Representation

- So, we can rewrite as, $Q(t) = [x(t)y(t)z(t)] = T.C$ are of the form
- We consider, 2D curve represented by [x(t) y(t)]

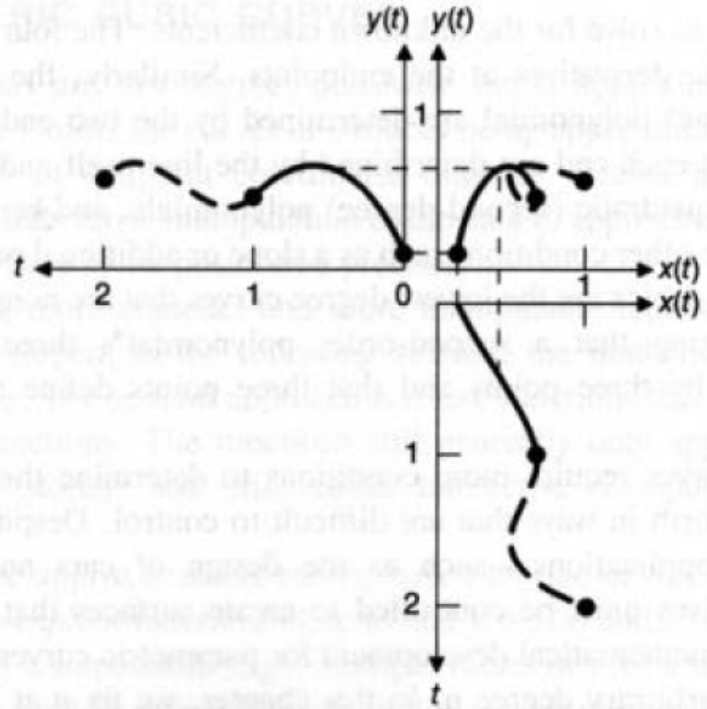
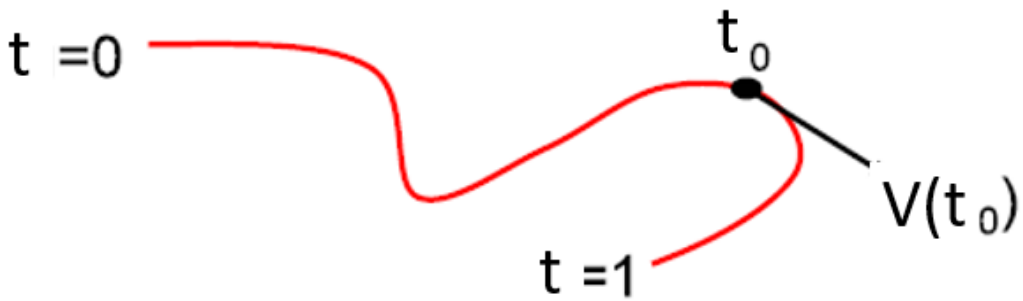


Fig. 11.7 Two joined 2D parametric curve segments and their defining polynomials. The dashed lines between the (x,y) plot and the $x(t)$ and $y(t)$ plots show the correspondence between the points on the (x,y) curve and the defining cubic polynomials. The $x(t)$ and $y(t)$ plots for the second segment have been translated to begin at $t = 1$, rather than at $t = 0$, to show the continuity of the curves at their join point.

Parametric Representation

- The figure also illustrates the ability of parametric to represent easily multiple values of y for a single value of x with single valued polynomials.
- The derivative of Q(t) is the parametric tangent vector of the curve.
- Let, the tangent vector at t_0, $T(t_0)$ is,
  $\frac{d}{dt}Q(t) = Q'(t) = [\frac{d}{dt}x(t)\frac{d}{dt}y(t)\frac{d}{dt}z(t)]$
  $\Rightarrow Q'(t) = \frac{d}{dt}T.C = \begin{bmatrix}3t^2 & 2t & 1 & 0\end{bmatrix}.C$
  $= \begin{bmatrix}3a_x t^2 + 2b_x t + c_x & 3a_y t^2 + 2b_y t + c_y & 3a_z t^2 + 2b_z t + c_z\end{bmatrix}$



Parametric Representation

- If two curve segments join together, the curve has G^0 *geometric continuity*.
- If the directions (but not necessarily the magnitudes) of the two segments tangent vectors are equal at join point, the curve has G^1 *geometric continuity*.
- G^1 **geometric continuity**: The geometry slopes of the segments are equal at the join point.
- For two tangent vectors TV_1 and TV_2 to have the same direction, it is necessary that one be a scalar multiple of the other: $TV_1 = k.TV_2$ with $k < 0$

Parametric Representation

- Let, $Q_1(t)$ and $Q_2(t), t \Rightarrow [0, 1]$ be two parametric curves.
- Level of parametric continuity of the curves at the joint between $Q_1(t)$ and $Q_2(t)$:
  - C^{-1}: The joint is discontinuous, $Q_1(1) \neq Q_2(0)$
  - C^0: Positional continuous, $Q_1(1) = Q_2(0)$
  - C^1: Tangent continuous, C^0 and $Q_1'(1) \neq Q_2'(0)$
  - C^n: Continuous upto n-th derivative, $Q_1^{(j)}(1) = Q_2^{(j)}(0), 0 \leq j \leq k$

Parametric Representation

- **Parametric continuity/ first degree continuity in the parameter t/** C^1*continuous*: If the tangent vectors of the two cubic curve segments are equal (i.e.: their directions and magnitude) at the segments' join point.
- C^n **continuous:** If the direction and magnitude of $\frac{d^n}{dt^n}[Q(t)]$ through the n-th derivative are equal at the join point.
- Figure 11.8 shows curves with three different degrees of continuity.
- Parametric curve segment is itself everywhere continuous; the continuity of concern here is at the join points.

Parametric Representation

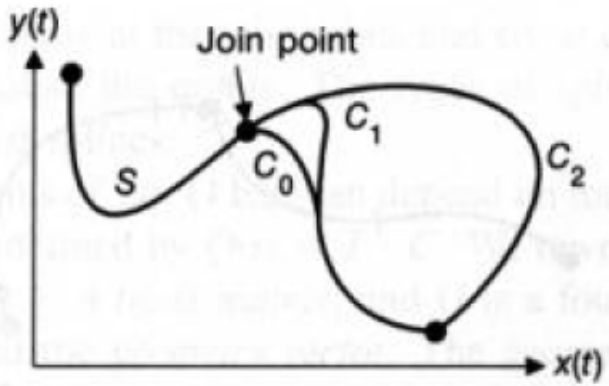


Fig. 11.8 Curve segment S joined to segments C_0, C_1, and C_2 with the 0, 1, and 2 degrees of parametric continuity, respectively. The visual distinction between C_1 and C_2 is slight at the join, but obvious away from the join.

Parametric Representation

- The tangent vector Q'(t) is the velocity of a point on the curve with respect to the parameter t.
- Similarly, the second derivative of Q(t) is the acceleration.
- If a camera is moving along a parametric cubic curve in equal time steps and records a picture after each step, the tangent vector gives the velocity of the camera along the curve.
- The camera velocity acceleration at joint points should be continuous to avoid jerky movements in the resulting animation sequence.
- It is this continuity of acceleration accross the join point in Figure 11.8 that makes the C^2 curve continue farther to the right than the C^1 curve, before bending around the endpoint.

Continuity

- Generally, C^1 continuity implies G^1 but the converse isn't generally true.
- G^1 continuity is generally less restrictive than is C^1, so curves can be G^1 but not necessarily C^1.
- Join points with G^1 continuity will appear just as smooth as those with C^1 continuity (Fig 11.9).
- There is a special case in which C^1 continuity doesn't imply G^1 continuity: when both segments' tangent vectors are [0 0 0] at the join point.
- In this case, the tangent vectors are indeed equal but their directions can be different (Fig 11.10).
- Fig 11.11 shows, the camera velocity slows down to zero at the join point, changes its direction when the velocity is zero, and the camera accelerates in the new direction.

Continuity

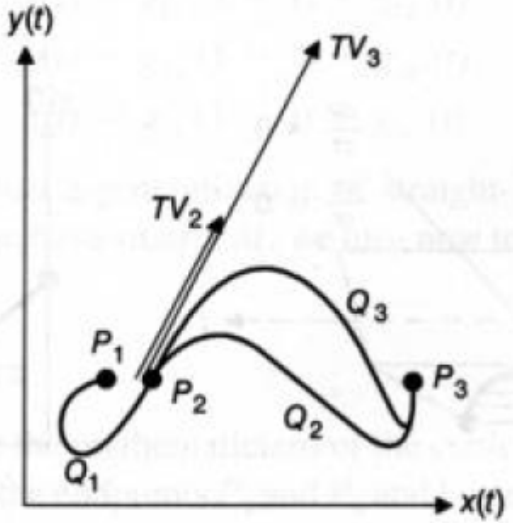


Fig. 11.9 Curve segments Q_1, Q_2, and Q_3 join at the point P_2 and are identical except for their tangent vectors at P_2. Q_1 and Q_2 have equal tangent vectors, and hence are both G^1 and C^1 continuous at P_2. Q_1 and Q_3 have tangent vectors in the same direction, but Q_3 has twice the magnitude, so they are only G^1 continuous at P_2. The larger tangent vector of Q_3 means that the curve is pulled more in the tangent-vector direction before heading toward P_3. Vector TV_2 is the tangent vector for Q_2, TV_3 is that for Q_3.

Continuity

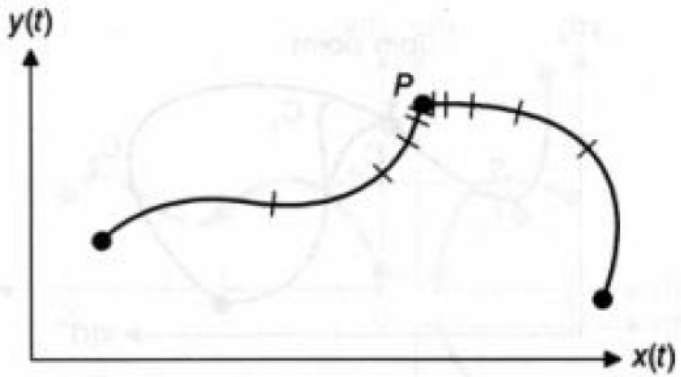


Fig. 11.10 The one case for which C^1 continuity does not imply G^1 continuity: the tangent vector (i.e., the parametric velocity along the curve) is zero at the join joint P. Each tick mark shows the distance moved along the curve in equal time intervals. As the curve approaches P, the velocity goes to zero, then increases past P.

Continuity

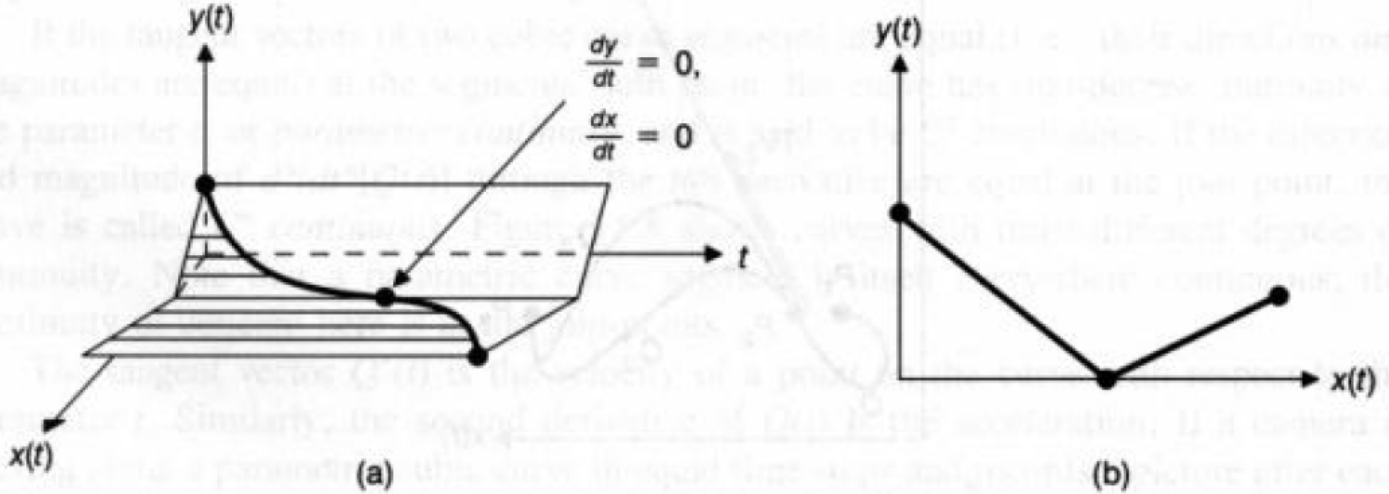


Fig. 11.11 (a) View of a 2D parametric cubic curve in 3D (x, y, t) space, and (b) the curve in 2D. At the join, the velocity of both parametrics is zero; that is, $dy/dt = 0$ and $dx/dt = 0$. You can see this by noting that, at the join, the curve is parallel to the t axis, so there is no change in either x or y. Yet at the join point, the parametrics are C^1 continuous, but are not G^1 continuous.

Continuity

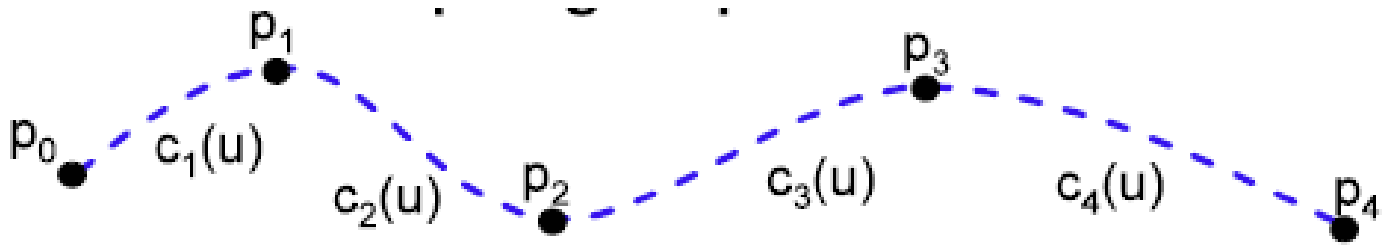
- The plot of a parametric curve is distinctly different from the plot of an ordinary function, in which the independent variable is plotted on the y axis.
- In parametric curve plots, the independent variable t is never plotted at all.
- This means that we can't determine, just by looking at a parametric curve plot, the tangent vector to the curve.
- It is possible to determine the direction of the vector, not the magnitude.

Continuity

- If $\gamma(t), 0 \leq t \leq 1$ is a parametric curve, its tangent vector at time 0 is $\gamma'(0)$.
- If we let $\eta(t) = \gamma(2t), 0 \leq t \leq \frac{1}{2}$, then the parametric plots of γ and η are identical.
- On the other hand, $\eta'(0) = 2\gamma'(0)$.
- Thus, two curves that have identical plots can have different tangent vectors.
- This is the motivation for the definition of geometric continuity: For two curves to join smoothly, we require only that their tangent-vector directions match, not that their magnitudes match.

Splines

- Splines have C^1 and C^2 continuity at the join points and come close to their control points, but generally do not interpolate the control points.
- So, splines are piecewise, low degree, polynomial curves, with continuous joints.
- Types of splines are:Uniform B-splines, Nonuniform B-splines, and β splines - Self Study
- Advantages of splines:
  - Rich representation
  - Geometrically meaning coefficients
  - Local effects
  - Interactive sculpting capabilities



Parametric Cubic Curves

- A curve segment Q(t) is defined by constraints on endpoints, tangent vectors, and continuity between curve segments.
- Each cubic polynomial has four coefficients, so four constraints will be needed, allowing us to formulate four equations for the four unknowns.
- We can define a curve Q(t) by T . C.
- We can rewrite the coefficient matrix as C = M . G, where M is a 4X4 basis matrix, and G is a four element column vector of geometric constraints, called *Geometric vector*.
- The geometric vectors are just the conditions such as endpoints or tangent vectors, that define the curve.
- We use G_x to refer to the column vector of just the x components of the geometry vector; G_y and G_z have the similar definitions.

Parametric Cubic Curves

- M or G, or both M and G , differ for each type of curve.
- The elements of M and G are constants, so the product T . M . G is just three cubic polynomials in t.
- Expanding the product Q(t) = T . M . G gives:

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} . \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} . \begin{bmatrix} G_1 \\ G_2 \\ G3 \\ G_4 \end{bmatrix}$$

- Multiplying out just x(t) = T . M . G_x gives

$$x(t) = (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_{1x} + (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) g_{2x} + (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) g_{3x} + (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) g_{4x}$$

Parametric Cubic Curves

- The equation emphasizes that the curve is a weighted sum of the elements of the geometry matrix.
- The weights are each cubic polynomials of t, and are called *Blending functions*.
- The blending functions B are given by B = T . M.
- Notice the similarity to a piecewise linear approximation, for which only two geometric constraints (the endpoints of the line) are needed, so each curve segment is a straight line defined by the endpoints G_1 and G_2: $x(t) = g_{1x}(1-t) + g_{2x}(t)$
  $y(t) = g_{1y}(1-t) + g_{2y}(t)$
  $z(t) = g_{1z}(1-t) + g_{2z}(t)$
- Parametric cubic curves are just a generalization of straight line approximations.

Hermite Curves

- The Hermite form of the cubic polynomial curve segment is determined by constraints on the endpoints P_1 and P_4 and tangent vectors at the endpoints R_1 and R_4.
- To find the Hermite basis matrix, M_H, which relates the Hermite geometry vector, G_H to the polynomial coefficients, we write four equations, one for each constraints, in four unknown polynomial coefficients, and then solve them.

Hermite Curves

- $Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.M_H.G_H$

  where, G_H is the column vector $\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$

- $x(t) = T.M_H.G_{H_x}$
  $y(t) = T.M_H.G_{H_y}$
  $z(t) = T.M_H.G_{H_z}$

  where, $G_{H_x} = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_x$ $G_{H_y} = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_y$ $G_{H_z} = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}_z$

- Derivation of Q(t) - self study

Bezier Curves

- The Bezier form of the cubic polynomial curve segment indirectly specifies the endpoint tangent vector by specifying two intermediate points that are not on the curve.
- The starting and the ending tangent vectors are determined by the vectors P_1P_2 and $P_3P_4$and are related to R_1 and R_4 by,
  $R_1 = Q'(0) = 3(P_2 - P_1)$
  $R_4 = Q'(1) = 3(P_4 - P_3)$

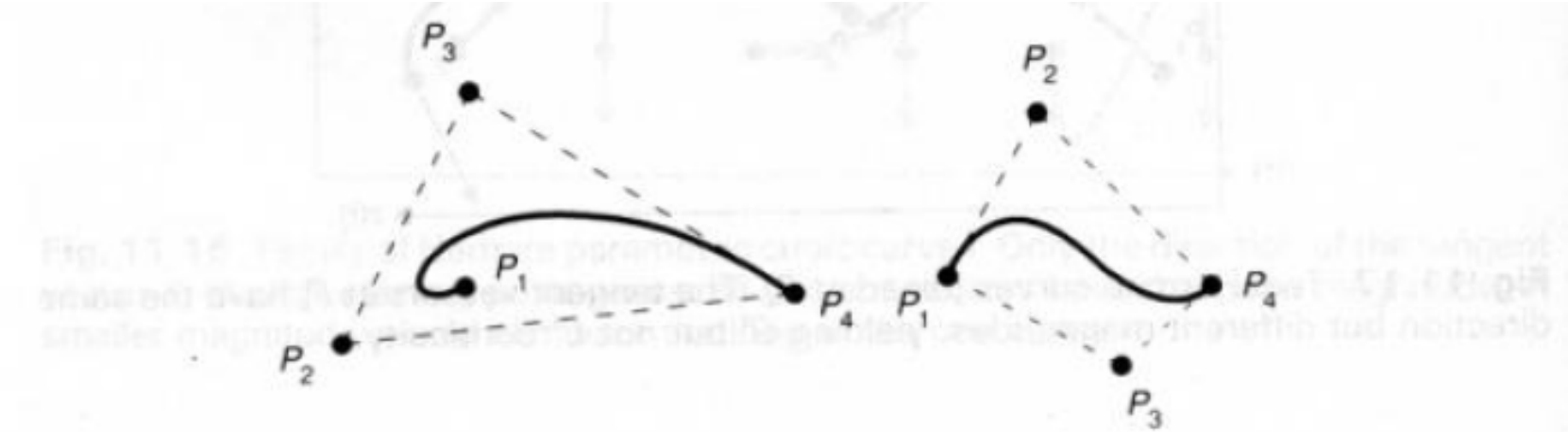


Fig. 11.19 Two Bézier curves and their control points. Notice that the convex hulls of the control points, shown as dashed lines, do not need to touch all four control points.

Bezier Curves

- The Bezier Geometry vector is G_B and the Bezier Basis Matrix is M_B
  $Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.M_B.G_B$

  where, G_B is the column vector $\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$
- Derivation of Q(t) - self study

End of Slides