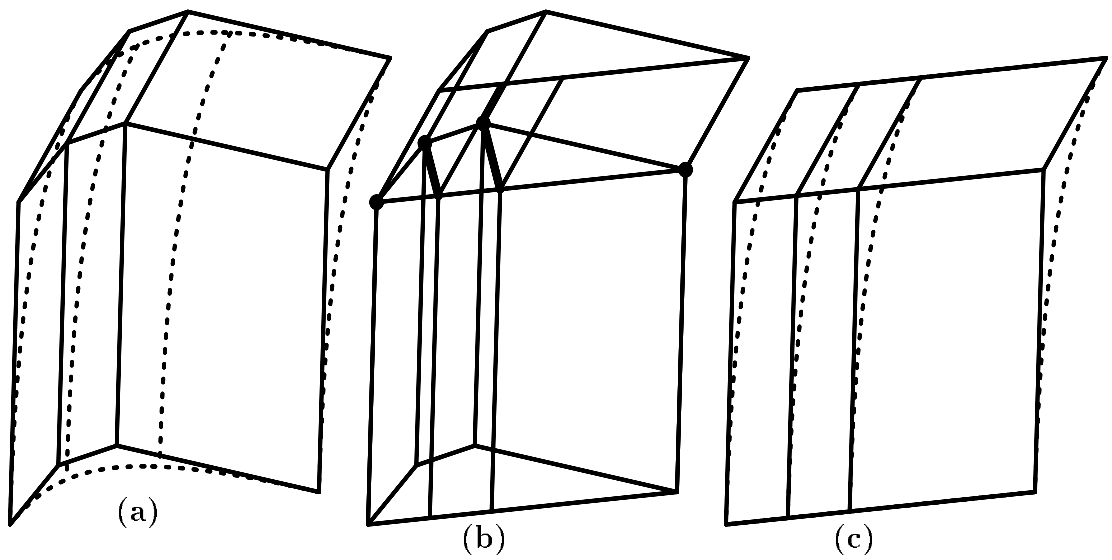


Figure 5.9. The speed of S's isocurve in the ruled direction is emulated by the ruled surface $\hat{R}$ approximating it. In (a), the jth column of S mesh, $P_{\bullet j}$, is projected in (b) onto the line connecting P_{0j} and P_{mj}. The spacing of the projected points is used to construct the mesh of $\hat{R}$'s in (c).

face as a set of ruled surfaces is derived in algorithm 5.2 based on this process.

Algorithm 5.2 returns a set of ruled surfaces that approximates the original surface S to within the required tolerance τ. Figure 5.10 shows an example of three consecutive stages of algorithm 5.2.

Assuming S satisfies a Lipschitz condition, which automatically holds for B-spline surfaces, let $(\frac{\Delta X}{\Delta v}, \frac{\Delta Y}{\Delta v}, \frac{\Delta Z}{\Delta v})$ be an upper bound on the first partial derivatives of S in the v direction. Given a finite range in the v parametric direction, $\mathcal{V}$, a bound on the Euclidean size is readily available as $(\mathcal{V}\frac{\Delta X}{\Delta v}, \mathcal{V}\frac{\Delta Y}{\Delta v}, \mathcal{V}\frac{\Delta Z}{\Delta v})$. Because algorithm 5.2 halves the parametric domain in each iteration, it halves the Euclidean bound in each iteration as well, so convergence in algorithm 5.2 is guaranteed. Note that we are concerned only with the v (ruled) direction because the representation is exact in the u direction.

Therefore, the less complex parametric direction, by some norm, may be a better candidate to select for the ruling direction approximation. Another measure for the selection of the subdivision direction may be the feasibility of the surface assembly.

Algorithm 5.2

Input:

```
S(u,v), surface to be divided in the v parametric direction.
τ, tolerance of approximation to be used.
```

Output:

```
𝒮, Set of ruled surfaces, approximating S(u,v) to within τ.
```

Algorithm:

```
RuledSrfApproximation( S, τ )
begin
  C1(t), C2(t) ⇐ Vmin and Vmax boundary of S.
  R ⇐ ruled surface between C1(t) and C2(t).
  R̂ ⇐ R refined and degree raised in v.
  If ( maxDistance( S, R̂ ) < τ )
    return { R }.
  else
  begin
    Subdivide S into two subsrfs S^1, S^2 along v.
    return
      RuledSrfApproximation( S^1, τ ) ∪
      RuledSrfApproximation( S^2, τ ).
  end
end
```

If S is an elongated tube, it may be easier to select and assemble the surfaces as sequence of rings than as a sequence of elongated strips. A third consideration may be whether the surface is closed in one direction or not. Such closed surfaces are very common, and it is very natural to approximate such surfaces as a set of rings (see Figures 5.10 and 5.11).

Once the set of ruled surfaces is determined, the surfaces must be laid flat on a plane, so they can be cut out. Lemma 5.1 can be used to verify whether the piecewise ruled surfaces are also developable. Because the isometry mapping is nonlinear, in general, an approximation must be used. We start the process by

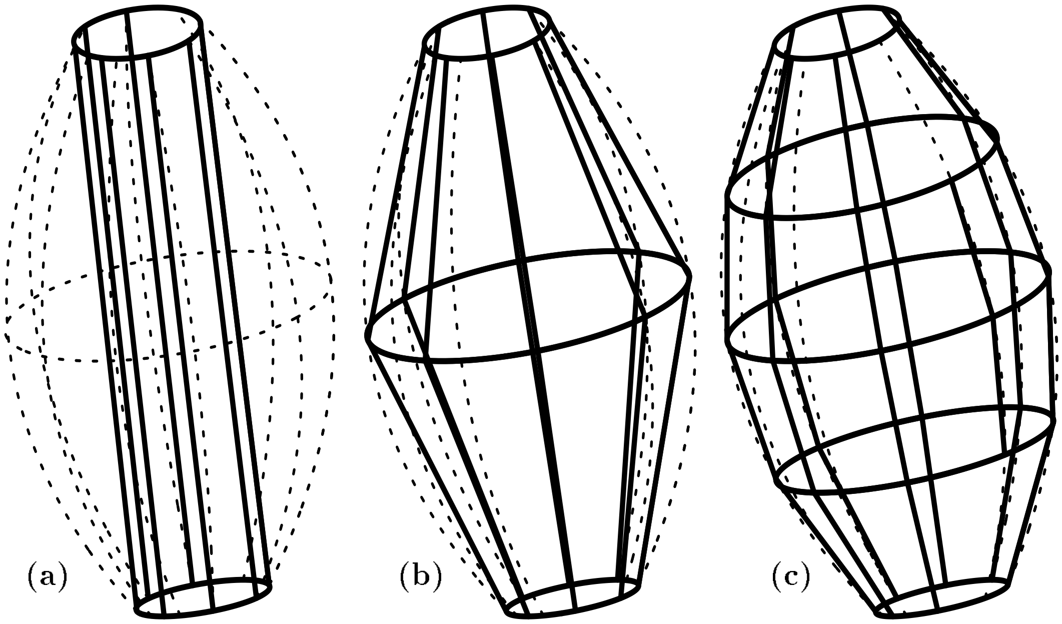


Figure 5.10. Three stages in approximating a surface with piecewise ruled surfaces.

approximating the two boundary curves of R that originated on S, $C_1(u)$ and $C_2(u)$, as piecewise linear curves $\hat{C}_1(u)$ and $\hat{C}_2(u)$, using refinement. An identical refinement should be computed and applied to both curves to insure they have the same number of linear segments, n. A one-to-one correspondence between the piecewise linear approximation of each curve is therefore established. Then, from each pair of corresponding linear segments, one from $\hat{C}_1(u)$ and one from $\hat{C}_2(u)$, a bilinear surface is created. Each bilinear is further approximated as two triangles along one of the bilinear diagonals. Finally, the $2n$ triangles are incrementally laid out and linearly transformed onto a plane (see Figure 5.12).

As stated above, the laying down of the surface is a nonlinear mapping and is only approximated. During the piecewise ruled surface approximation stage, it would be required to increase the number of ruled surfaces if a better approximation is necessary, complicating the assembly process. However, the penalty for a better layout approximation is reduced to only an enlargement of the data set.

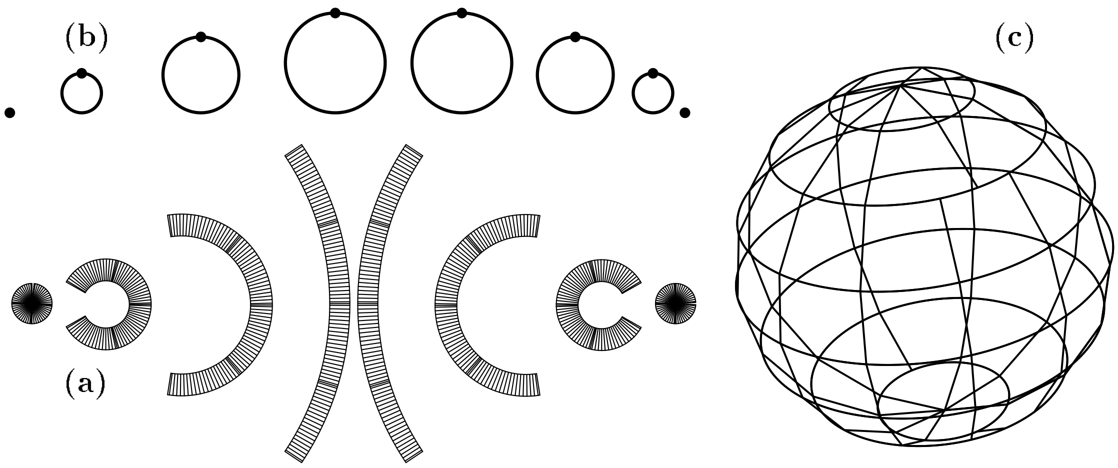


Figure 5.11. Piecewise ruled surface approximation layout of a sphere (a), its piecewise ruled surface cross sections (b), and assembled (c).

5.3.2 Extensions

It is a logical next step to improve the efficiency of algorithm 5.2 by subdividing S at v values that will minimize the number of ruled surfaces required to approximate S to within a given tolerance τ. Automatically determining candidate locations is difficult. However, a greedy approach can be adopted to determine a local minimum even though it does not guarantee global minimum in the number of ruled surfaces. The normal curvature in the v direction (the direction in which the approximating surfaces are ruled), $\kappa_n^v(u,v)$ can be computed symbolically. The maximum values of $\kappa_n^v(u,v)$ can then be used as subdivision locations. See Figure 5.13 for one such example. The normal curvature of the surface in tangent direction $\frac{\partial S}{\partial u}\frac{\partial u}{\partial t} + \frac{\partial S}{\partial v}\frac{\partial v}{\partial t}$ is

$$\kappa_n = \frac{II(a,b)}{I(a,b)} = \frac{II(\delta)}{I(\delta)} = \frac{\delta L \delta^T}{\delta G \delta^T}, \tag{5.6}$$

where $\delta = (\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t}) = (a,b)$ and G and L are the matrices of the first and second fundamental forms [21, 32], respectively.

From equation (5.6), when $\delta = (0,d)$, that is, the tangent vector direction is $d\frac{\partial S}{\partial v}$,

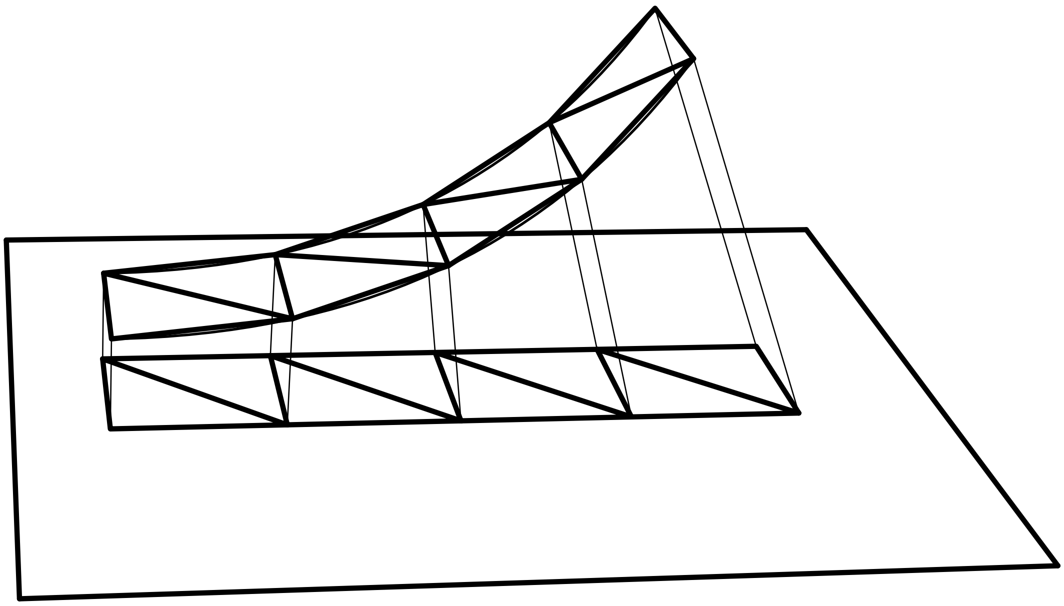


Figure 5.12. A ruled surface is approximated by triangles and unrolled onto a plane.

$$\begin{aligned}
\kappa_n^v &= \frac{(0,d)L(0,d)^T}{(0,d)G(0,d)^T} \\
&= \frac{d^2 \left\langle n, \frac{\partial^2 F}{\partial v^2} \right\rangle}{d^2 \left(\frac{\partial F}{\partial v} \right)^2} \\
&= \frac{\left\langle n, \frac{\partial^2 F}{\partial v^2} \right\rangle}{\left(\frac{\partial F}{\partial v} \right)^2}. \qquad (5.7)
\end{aligned}$$

Equation (5.7) is the normal curvature of the surface in the v direction. Equation (5.7) is also geometrically the curvature vector of the v iso-curve projected in the surface normal direction. For a nonarclength parameterized regular curve $C(t)$ (see [48]),

$$\kappa N = \kappa B \times T = \frac{\left(\frac{dC}{dt} \times \frac{d^2 C}{dt^2} \right) \times \frac{dC}{dt}}{\left(\frac{ds}{dt} \right)^4},$$

where s is the arc length parameterization of C. Because $\langle u, (v \times w) \rangle = \langle (u \times v), w \rangle$

$$\kappa \langle N, n \rangle = \frac{\left\langle \left(\frac{dC}{dt} \times \frac{d^2 C}{dt^2} \right) \times \frac{dC}{dt}, n \right\rangle}{\left(\frac{ds}{dt} \right)^4}$$

$$
\begin{aligned}
&= \frac{\left\langle \left(\frac{dC}{dt} \times \frac{d^2C}{dt^2}\right), \left(\frac{dC}{dt} \times n\right) \right\rangle}{\left(\frac{ds}{dt}\right)^4} \\
&= \frac{\left\langle \left(\frac{dC}{dt} \times n\right), \left(\frac{dC}{dt} \times \frac{d^2C}{dt^2}\right) \right\rangle}{\left(\frac{ds}{dt}\right)^4} \\
&= \frac{\left\langle \left(\frac{dC}{dt} \times n\right) \times \frac{dC}{dt}, \frac{d^2C}{dt^2} \right\rangle}{\left(\frac{ds}{dt}\right)^4} \\
&= \frac{\left\langle n, \frac{d^2C}{dt^2} \right\rangle}{\left(\frac{ds}{dt}\right)^2}, \\
&= \kappa_n^v
\end{aligned}
\tag{5.8}
$$

because $\|\frac{dC}{dt}\| = \frac{ds}{dt}$ and n is orthogonal to $\frac{dC}{dt}$.

κ_n^v can be symbolically represented as a scalar NURBs surface. Its isolated local maxima are the suggested preferred locations for the piecewise ruled surface approximation subdivision. For obvious reasons, a maximum occurring on the boundary is of no interest, but C^1 discontinuities in the v parametric direction are likely candidates for subdivision locations. Therefore, a surface should first be preprocessed and subdivided at all locations where it is not C^1 continuous. $\kappa_n^v(u, v)$ should then be computed for the resulting C^1 continuous subsurfaces. If the original surface is not C^2, κ_n^v will not even be C^0. Special care should be taken in evaluating κ_n^v along those discontinuous edges, because limits from both sides along the C^0 discontinuities would converge to different values.

An example is provided in Figure 5.13, which shows a surface with two very highly curved regions in the v direction (Figure 5.13a). Those regions are very noticeable in the $\kappa_n^v(u, v)$ (Figure 5.13b) computed for this surface. Therefore, $\kappa_n^v(u, v)$ can be used to automate the scheme to make more optimal ruled surface approximation.

In some cases, the laid out ruled surfaces can be insufficient to assemble the model, depending upon the assembly method. Some extra material might need to be

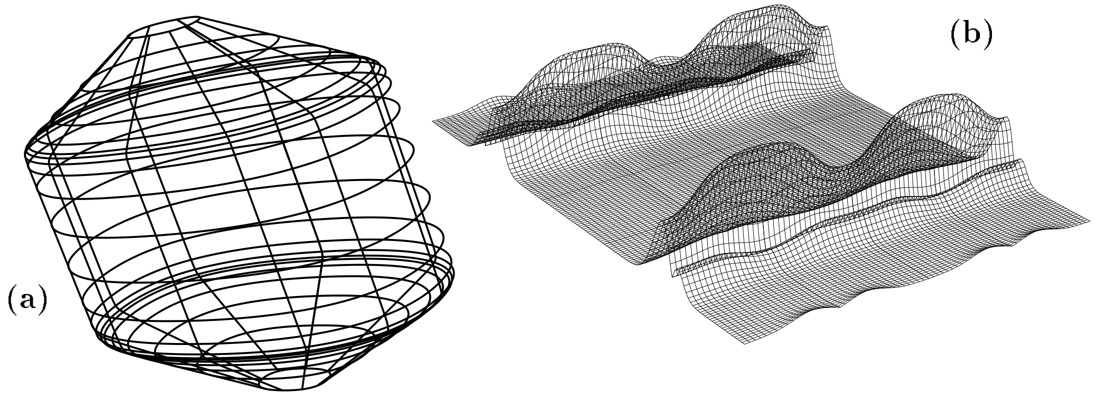


Figure 5.13. $\kappa_n^v(u,v)$ (b) is used to determine where to subdivide the surface (a).

included as stubs so the pieces may be stitched or welded together. Such stubs can be constructed by offsetting [25] the boundary curves of the planar representation of the approximating ruled surfaces. Figure 5.14 shows an examples of stubs generated using this approach that can be used for the layout of Figure 5.10 (c). However, such stubs can cause a C^0 seam between two folded developed surfaces resulting in a little stair with height equal to the material thickness. An alternative approach would be to connect two adjacent ruled surfaces using a separate stub made to span the two surfaces, from underneath, eliminating the stair.

When Boolean operators are applied to freeform models, trimmed surfaces result [47], and only part of each tensor product surface is used in the final model. In order to approximate freeform trimmed surfaces with piecewise ruled surfaces, it is necessary to position the trimming curves in the plane with the ruled surfaces. The problem is equivalent to finding the corresponding location of a specific surface Euclidean point in the planar representation of a ruled surface, given the (trimming curve) point in the surface parametric space. With the added constraint that the surface speed in the v direction must be constant to within a prespecified tolerance, locating the given (u,v) point in the planar ruled surface becomes a simplified

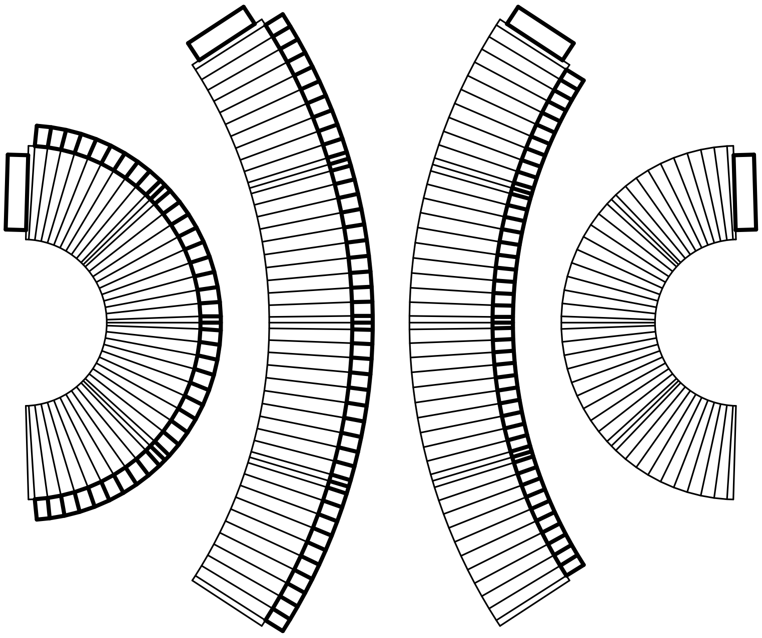


Figure 5.14. Stubs can be created by offsetting the planar boundary curves.

problem. Because the u direction is approximated as piecewise linear in the laying out stage, a binary search in u can efficiently reveal the bilinear segment containing the point. Within the bilinear surface the u direction is also assumed to be of constant speed and the exact location is then interpolated from the flat bilinear four corner points. Finally, because the ruled surface representation is only an approximation, it may be desired to re-execute the Boolean operations on the ruled surface approximations and create the appropriate trimming curves for the fabrication surfaces instead of the original surfaces because the intersections curves are not identical. Figure 5.15 shows a simple layout with trimming curves. Section 5.3.3 provides several examples of more complex models composed of trimmed surfaces as well.

5.3.3 Examples

The algorithm developed was used to generate layouts for several computer models, automatically. Figure 5.11 shows the sphere layout on a plane with its 3

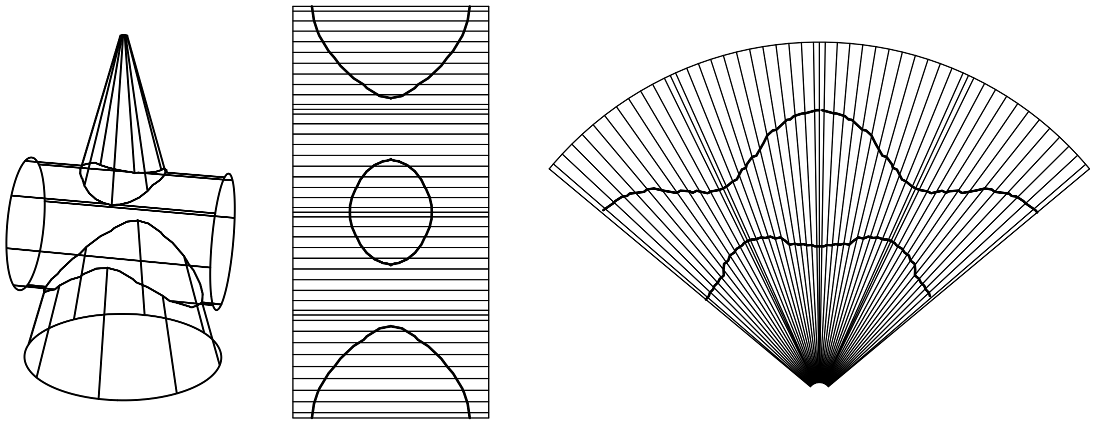


Figure 5.15. Trimming curves should be laid out with the ruled surfaces.

dimensional piecewise ruled surface approximation. Figure 5.14 shows the layout of the model in Figure 5.10 with an example of stubs. Figure 5.15 shows the layout of a cone and a cylinder intersecting each other with their trimming curves. Figure 5.16 shows a helicopter model [14], its layout projection with the ruled surface cross sections, and the assembled piece.

Figure 5.17 shows several models layed out using these techniques and then assembled from heavy paper. Each developable surface was cut from paper and folded into its 3-space shape. Paper connecting stubs were used to hold and keep the pieces together.

More complex models can be created using Boolean operations when the model is a union or intersection of several freeform surfaces. The layouts of the trimming curves of these surfaces are also computed, in a way similar to the ruled surface layouts. Figures 5.18 and 5.19 show more complex models having several trimmed surfaces.

Table 5.2 provides some timing results for the model decomposition and layout computation. Tests were run on a SGI4D 240 GTX (R3000 25MHz Risc machine). All tests are measured in seconds.

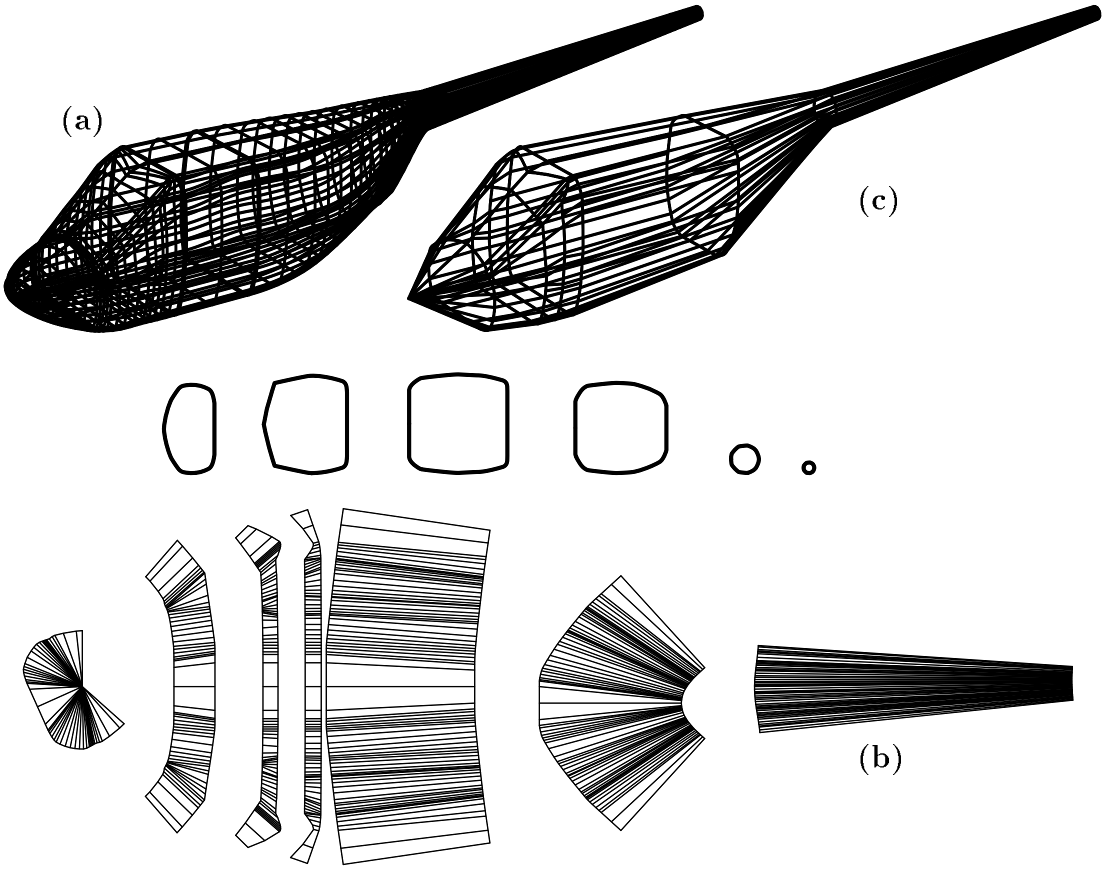
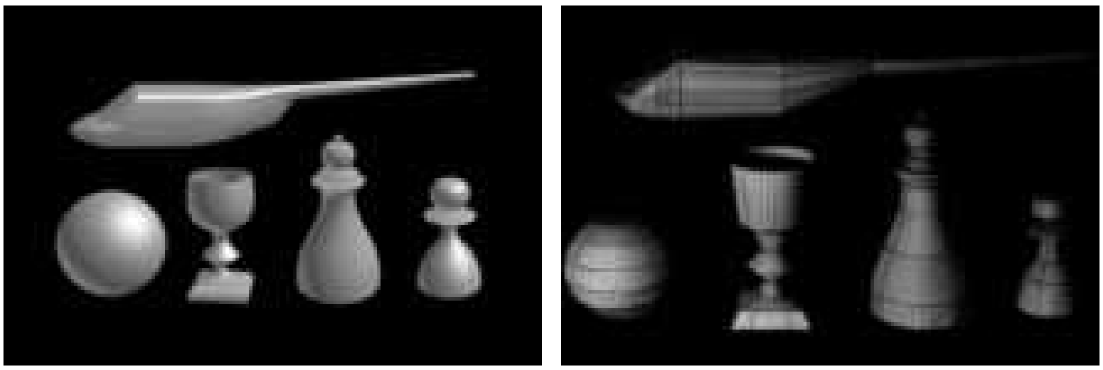


Figure 5.16. A helicopter model (a) laid out (b) and assembled (c).

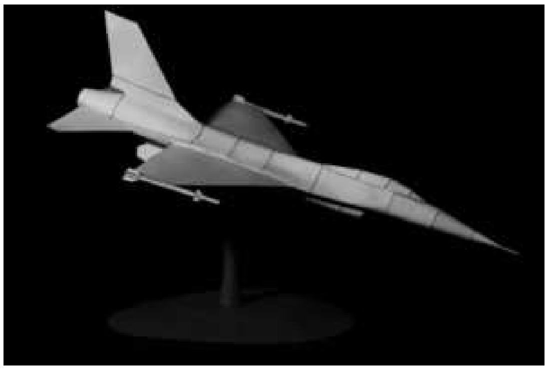


(a) (b)

Figure 5.17. Computer models (a) and assembled out of heavy paper (b).

(a) (b)

Figure 5.18. Teapot computer model (a) and assembled out of heavy paper (b).



(a) (b)

Figure 5.19. Computer model of an f16 (a) and assembled out of heavy paper (b).

Table 5.2. Different models layout construction times.

Model	Time (Sec.)
Tube (Figure 5.14)	1.4
Helicopter (Figure 5.16)	7
Pawn (Figure 5.17)	3
Teapot (Figure 5.18)	5
f16 (Figure 5.19)	150

CHAPTER 6

OTHER APPLICATIONS

Nothing is particularly hard if you divide it into small jobs.

Henry Ford

This Chapter will present several other applications that can benefit from combined symbolic and numeric computation. Some of these problems have undergone extensive research and are provided here to reflect on the power of this combination. In section 6.1 we develop an adaptive technique to approximate higher order Bézier curves using cubic Bézier curves. In section 6.2 we develop the tools so the composition operation may be added to the set of operation defined in Chapter 2. The composition tool will open the way for solving a whole set of problems. In section 6.3, we develop techniques for visualizing surface slopes and steepnesses. The steepness of a model may be of interest when only a limited set of slopes is allowed. This is important for road design or even for slides. Section 6.4 discusses more surface properties. The speed of the surface has a direct affect on the way the surface is milled. It also provides a bound on the amount of Euclidean movement while moving a fixed distance in parametric space. The twist is another measure for a surface shape and is not as intuitive as one would like, as will be shown. Like curvature estimation methods, analysis techniques of twist have been previously based on a presampled grid from the surface parametric space [3]. As in Chapter 4, we will demonstrate in section 6.5 the use of property surfaces to globally bound the twist properties.

6.1 Bézier Curve Approximation

Cubic polynomial curves are frequently used in graphics and CAGD. The fact that piecewise cubic polynomial curves are the curves with the lowest order that can provide C^2 continuous interpolation or approximation is one of the main reasons.

Because any polynomial basis may be used, we select the Bernstein polynomial which is numerically stable [31].

One can find a growing number of hardware implementations for evaluating cubic polynomials in modern workstations and devices, for mainly display purposes. Taking advantage of these implementations speeds up algorithms. Converting other types of curves into this simple form is not always obvious. Higher order curves cannot, in general, be represented as cubic polynomials. This is also the case for rational curves. Even rational quadratic curves are not representable as cubic polynomials. In other words, approximation techniques must be used. The Postscript [53] language is an example in which only cubic polynomial are supported. Therefore NURBs curves or even rational Bézier curves must be approximated to display them on Postscript devices.

Currently, the most common technique is to refine the curves and approximate them as piecewise linear curves which are then displayed. Because linear segments are displayable by almost every device, portability is gained. However, this method suffers from two major drawbacks. First, the size of the data is huge - several magnitudes larger than the original curves. Furthermore, the data are not exact any more and are not even C^1 continuous. Approximating higher order or rational curves as cubic polynomials is probably a better approach. In [38, 39], a subdivision based approach is used to create such an approximation. A cubic polynomial is compared to the curve being approximated. If the cubic polynomial is not accurate enough, the curve is subdivided and the two new cubic approximations are compared to the two parts of the original curve.

In this section, we enhance this technique so that the approximating cubic piecewise polynomials join with C^1 continuity are everywhere within a prescribed tolerance to the original curve everywhere.

Because we choose to derive this approximation only for with higher order or rational Bézier curves, NURBs curves will be preprocessed and converted to an ordered set of rational Bézier curves of the same order.

In Chapter 3, we introduced a global method to compute the distance between a curve and its offset approximation. The same technique may be used here to compute the distance between a curve and its approximation (superscript denotes order, subscript a denotes approximation):

Algorithm 6.1 provides a piecewise cubic approximation to a given curve in line (1). A cubic polynomial curve has twelve degrees of freedom (four E^3 points). Six of them are used to interpolate the original curve end points. Because we preserve end points tangents (to easily preserve C^1 continuity), two degrees of freedoms are left - the speeds of the tangents. One can use the original curve speed to provide the information to determine the last two degrees of freedom, a necessary condition from the way $\delta(t)$ is computed (see below). This approach was used in Figures 6.1 and 6.2.

Raising the order of Bézier curves as done in line (2) in algorithm 6.1 is a fairly simple task (see Chapter 2, equation (2.9)).

The subdivision (line (3), algorithm 6.1) exploits the distance function, $\delta(t)$, and instead of subdividing in the middle of the parametric domain, a common technique, the curve is subdivided at the location of the maximum distance (error). Because the end points of the piecewise cubics interpolate the original curve, this error is automatically reduced to zero.

A different approach is to adjust the tangent speeds to minimize the distance between the original curve and it approximation, in a similar way to that of offsets in 3.2. This approach needs further investigation.

Algorithm 6.1

Input:

```
  τ, required approximation curve tolerance.
  Cⁿ(t), input curve of order n.
```

Output:

```
  𝒞, set of piecewise cubic Bezier curves, C³ₐ(t) approx.  Cⁿ(t).
```

Algorithm:

```
  CubicBezierApprox(Cⁿ(t), τ)
  begin
(1) C³ₐ(t) ⇐ cubic Bezier approximation to Cⁿ(t).
(2) Raise C³ₐ(t) order to order n:  Cⁿₐ(t).
    Compute distance δ(t) between Cⁿₐ(t) and Cⁿ(t).
    if ( δ(t) maximum distance > τ ) Do
      begin
(3)     Subdivide Cⁿ(t) at δ(t) maximum into Cⁿ₁(t), Cⁿ₂(t).
        return CubicBezierApprox(Cⁿ₁(t), τ) ∪
             CubicBezierApprox(Cⁿ₂(t), τ).
      end
    else
      return C³ₐ(t).
  end
```

In some cases, when approximating shape of a higher order curve using a lower one, the speeds of the curve is irrelevant. One such case is display on Postscript devices, in which the only requirement is to preserve the curve shape. For each cubic Bézier and its corresponding higher order subcurve that it approximates, algorithm 6.1 guarantees

1. End points interpolate the original higher order subcurve,

2. End point tangents are in the same direction as the original higher order subcurve tangents (G^1), and possibly with the same length (C^1),

3. The distance between the two curves is within the specified tolerance, and

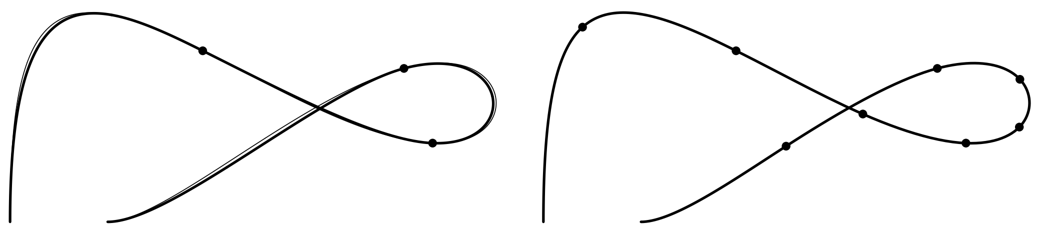


Figure 6.1. Cubic Bézier approximation to higher order curves (two tolerance).

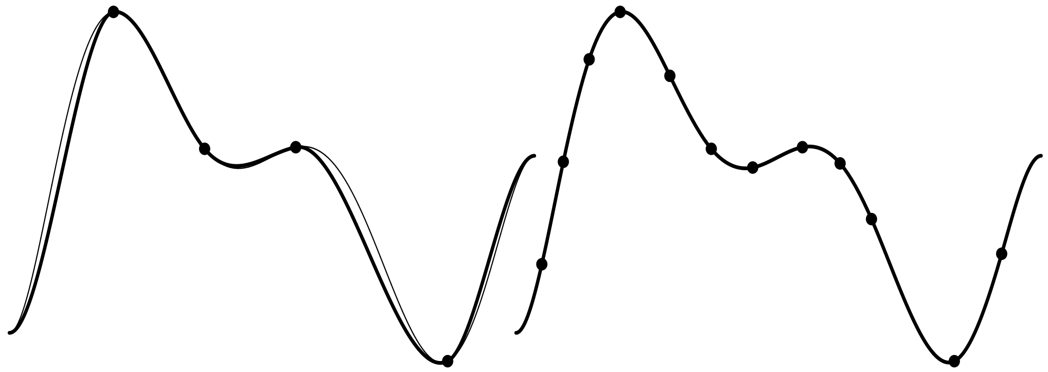


Figure 6.2. Cubic Bézier approximation to higher order curves (two tolerance).

4. The speed of both curves is the same to within the specified tolerance.

The first three properties are required in order to mimic the curve shape. However, property 4 is only a result of the way the distance, $\delta(t)$, between the two curves is computed. If one could efficiently answer whether the two curves are within the desired tolerance, this constraint could be omitted. Unfortunately, no such global and efficient algorithm exists and, as a result, the distances are computed with the respective parameter values and curve speed is preserved as well. Modifying the tangent lengths directly affects the speed of the curve so a different distance measure is needed.

6.2 Composition

Composition, $f \circ g$, is a powerful operation that has not found much use in graphics and CAGD yet. Some work can be found on the implicit use of composition in deformations [19, 61]. The bivariate surface g is warped in a field defined as a trivariate volume f, resulting in a bivariate composed and deformed surface. This deformation, implicitly using composition, can serve as a modeling tool as well and can provide exact and well behaved results. In [11], a parametric curve $g = (u(t), v(t))$ is composed with the surface $f(u, v)$ to find the exact Euclidean representation of the curve $f(u(t), v(t))$, to be used as a fillet boundary curve in fillet construction. $f(u(t), v(t))$ can be used anywhere the exact representation of the Euclidean curve is needed, given the parametric curves. Mapping trimming curves from parametric space to the Euclidean space is another example.

In this section we will explore the composition $f(u(t), v(t))$. Extending this to a trivariate deformation volume $f(u(r, s), v(r, s), w(r, s))$ is simple.

Let $C(t) = (u(t), v(t))$ be a Bézier curve such that $u(t) \in (0, \ldots, 1), \forall t$ and $v(t) \in (0, \ldots, 1), \forall t$. Let S be a Bézier surface.

$$\begin{aligned} S(u(t), v(t)) &= \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} \mathbf{B}_j^m(v(t)) \mathbf{B}_i^n(u(t)) \\ &= \sum_{i=0}^{n} \left(\sum_{j=0}^{m} P_{ij} \mathbf{B}_j^m(v(t)) \right) \mathbf{B}_i^n(u(t)). \end{aligned} \tag{6.1}$$

The curve-surface composition is now narrowed to the problem of computing the composition of $\mathbf{B}_i^n(c(t))$, where $c(t)$ is a scalar curve. Assuming one can compute and represent the composition $\mathbf{B}_i^n(c(t))$, $c(t) \in [u_{min}, u_{max}]$, as a curve, the curve $S(u(t), v(t))$ is also representable because it involves in scaling, addition and multiplication of $\mathbf{B}_i^n(c(t))$ terms only. These operations were explored in Chapter 2.

$$\mathbf{B}_i^n(c(t)) = \binom{n}{i} (1.0 - c(t))^{n-i} (c(t))^i. \tag{6.2}$$

Interestingly enough, equation (6.2) contains only tools developed in Chapter 2 namely, curve addition and curve multiplication (power).

What if either the curve or the surface is rational? For a rational surface, nothing changes. The P_{ij} in equation (6.1) should simply be treated as in projective space. If the curve is rational, equation (6.2) now becomes

$$\begin{aligned}\mathbf{B}_i^n(c(t)) &= \binom{n}{i}\left(1.0 - \frac{c(t)}{w(t)}\right)^{n-i}\left(\frac{c(t)}{w(t)}\right)^i \\ &= \binom{n}{i}\frac{(w(t)-c(t))^{n-i}(c(t))^i}{(w(t))^n}. \end{aligned} \tag{6.3}$$

Equation (6.3) should then be substituted into equation (6.1) in a similar way to equation (6.2). If surface $S(u, v)$ is rational as well the denominator term in equation (6.3), $(w(t))^n$, is canceled because it appears in both the surface numerator and denominator. If however, the surface was a polynomial, the resulting composed curve becomes rational.

Figures 6.3 and 6.4 show some examples for Bézier curves and surfaces. Figure 6.3 has a polynomial surface and several parametric curves mapped onto the surface. Figure 6.4 has a surface which is an extrusion of an arc and, as such, is rational. Both Figures have the parametric space on the left and the Euclidean mapping on the right.

Unfortunately, the order of the resulting composed curves is quite high. Let d be the curve degree while the surface degrees are m and n as can be seen from equation (6.1). It immediately follows from equations (6.1) and (6.2) that the degree of the composed curves is equal to $dn + dm$. Table 6.1 provides these orders for common cases. Note that even when either the surface or the curve is rational (or both), the order of the resulting curve does not change.

In some cases, it can be important to reparametrize a curve. The composition tool allows exactly that. By substituting s in $C(s)$ by $s = c(t)$, we reparametrize a curve to:

$$C(c(t)) = \sum_{i=0}^{n} P_i \mathbf{B}_i^n(c(t)). \tag{6.4}$$

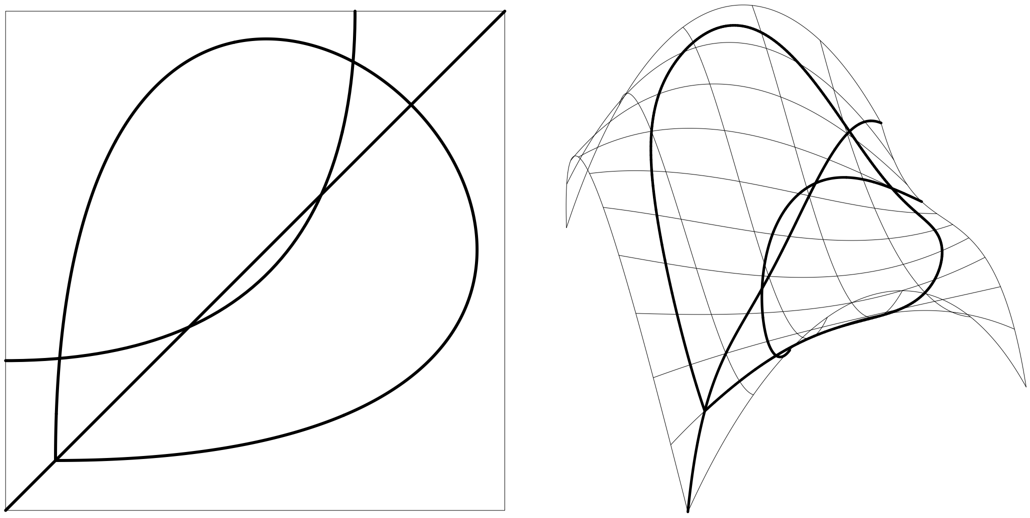


Figure 6.3. Bézier curve (polynomial) surface composition.

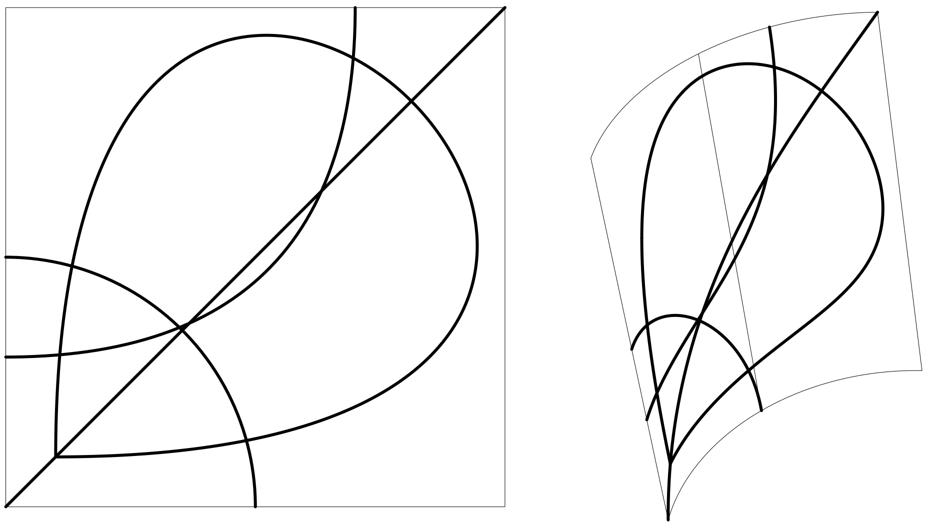


Figure 6.4. Bézier curve (rational) surface composition.

Computing equation (6.4) involves scaling (with P_i) and addition of curves resulting from the composition of $\mathbf{B}_i^n(c(t)$, which we dealt with in equations (6.2) and (6.3).

Figure 6.5 demonstrates the speed of three arcs after reparametrizing with $c(t) = t^2$ in the middle and reparametrizing with $c(t) = t^4$ on the right. The original arc is on the left.

Table 6.1. Curve on surface composition - orders.

Surface orders	Curve order	Composed curve order
3×3	2	5
3×3	3	9
3×3	4	13
4×4	2	7
4×4	3	13
4×4	4	19
3×4	2	6
3×4	3	11
3×4	4	16

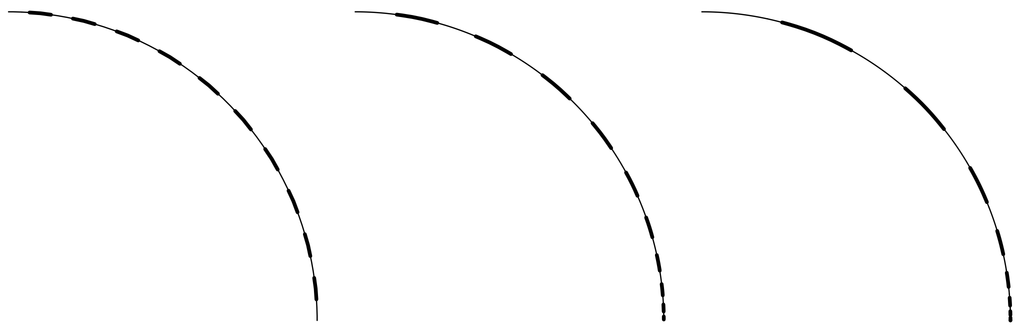


Figure 6.5. Rational Bézier curve reparametrizing using composition.

6.3 Surface Steepness

The slope of a planar curve at a given point is equal to the angle between the tangent to the curve and a reference line, usually the horizontal axis. In an analogous way we define the *surface slope* at a given point, p, as the angle between the plane tangent to the surface at p and a reference plane. Without loss of generality, in the discussion below we assume that the reference plane is the xy plane.

Because the angle between two planes is equal to the angle between their two normals, to compute surface slope, one need only compute the angle between the surface normal and the z axis. Let n be the surface unit normal and let n_z be its