30.4. RAY-TRACING

Methods to ray-trace freeform geometry directly by the numerical evaluation of roots or zero-sets are already known. The demand for extreme robustness and stability (for example, the failure of one pixel in a million would be unacceptable) puts severe restrictions on the possible methods one can use. Some approaches support only special classes of surfaces such as surfaces of revolution, extrusion surfaces and sweep surfaces, or combinations of these. In one such development [4], sweep surfaces are rendered directly, utilizing the generating curves of the sweep surface as the basis for the coverage.

Here, we will examine two approaches that support or emulate direct ray-tracing methods for freeform geometry. In Section 30.4.1, we will consider a method that is known as *Bézier clipping*, which is a robust yet efficient numerical method to derive *Ray-Surface Intersections* (RSI). In Section 30.4.2, we examine an extension to the AIC-based coverage that supports ray-tracing and is called *Ruled Tracing*.

30.4.1. Bezier clipping

In general, finding the roots of polynomial functions of arbitrary degree is a difficult problem. However, where polynomials represent geometry, the Bézier form may yield some benefits. It should be noted that a B-spline surface can easily be converted to a piecewise Bézier form and hence the method presented here will be equally valid in the domain of B-spline surfaces.

Consider the parametric Bézier curve $C(t) = (x(t), y(t))$ of degree d. We seek the intersection points of $C(t)$ with the line $L : Ax + By + C = 0$ (see Figure 30.7 (a)). Substituting $C(t)$ into L, we are left with an implicit equation in one variable whose zero-set corresponds to the desired intersection points, $f(t) = Ax(t) + By(t) + C = 0$. Clearly, $f(t)$ is a scalar Bézier function of the same degree as $C(t)$,

$$f(t) = \sum_{i=0}^{d} c_i B_{i,d}(t),$$

where $B_{i,d}(t)$ are Bézier basis functions of degree d. Since the function is scalar, one could replace the coefficients c_i by $\left(c_i, \frac{i}{d}\right)$, in the xy-plane, because $\sum_{i=0}^{d} \frac{i}{d} B_{i,d}(t) = t$. Let us denote the Bézier curve with coefficients $P_i = \left(c_i, \frac{i}{d}\right)$ as $\hat{f}_0(t)$.

The curve $\hat{f}_0(t)$ is contained in the convex hull $\mathcal{CH}(\hat{f}_0)$ formed by its control points, $\{P_i\}$. Moreover, the zeros of $f(u)$ correspond to the intersection points of $\hat{f}_0(t)$ with the x-axis. Consider the first control point, P_0 of $\hat{f}_0(t)$ (see Figure 30.7 (b)). If P_0 is below the x-axis, as seen in Figure 30.7 (b), then the curve $\hat{f}_0(t)$ is guaranteed to be below the x-axis, as long as $\mathcal{CH}(\hat{f}_0)$ is also below the x-axis. Similar arguments hold if P_0 is above the x-axis. Moreover, the same line of reasoning holds for the last control point, P_d. Hence, one can clip $\hat{f}_0(t)$ from $t = 0$ up to t_1 (see Figure 30.7 (c)), where $\mathcal{CH}(\hat{f}_0)$ intersects the x-axis for the first time, and from $t = 1$ back down to t_2, where $\mathcal{CH}(\hat{f}_0)$ intersects the x-axis for the last time, creating $\hat{f}_1(t)$. The clipped curve $\hat{f}_1(t)$ contains exactly the same zeros as $\hat{f}_0(t)$. This numerical process can be allowed to iterate until the zero-set solution is located with sufficient precision.

The extension of this Bézier clipping process to surfaces requires several intermediate representations, although it follows the general direction of the approach we have taken for curves. Consider the tensor product Bézier surface:

$$S(u, v) = \sum_i \sum_j P_{ij} B_{i,d_u}(u) B_{j,d_v}(v).$$

The ray intersecting $S(u, v)$ is defined as the intersection of two orthogonal planes, $A_k x + B_k y + C_k z + D_k = 0,\ k = 1, 2$. Nishita et al. [20] employ the scan plane and the plane

containing the ray and the y-axis are for primary rays. Substitute the control points $P_{ij} = (x_{ij}, y_{ij}, z_{ij})$ of $S(u, v)$ into the planes, and let $d_{ij}^k = A_k x_{ij} + B_k y_{ij} + C_k z_{ij} + D_k$. Assuming $A_k^2 + B_k^2 + C_k^2 = 1$, d_{ij}^k equals the distance from P_{ij} to the kth plane. We now define a new planar surface as:

$$D(u, v) = \sum_i \sum_j (d_{ij}^1, d_{ij}^2) B_{i,d_u}(u) B_{j,d_v}(v).$$

Surface clipping will be conducted over $D(u, v)$, which is merely an orthographic projection of $S(u, v)$ along the ray, if $S(u, v)$ is a polynomial surface. Even if $S(u, v)$ is rational, $D(u, v)$ remains a polynomial.

The solution of $D(u, v) = \mathbf{0}$ corresponding to the intersection between the ray and the original surface $S(u, v)$ is obtained by performing Bézier clipping steps over $D(u, v)$ with respect to some line in the plane. Nishita et al. [20] give more details and also discuss efficiency and timing considerations. They describe how trimmed surfaces are supported by the manipulation of Bézier trimming curves. Computations of the inclusion/exclusion decisions in both the untrimmed and trimmed domains of the surface are also derived through Bézier clipping. These authors discuss efficiency and timing considerations.

30.4.2. Ruled tracing

At the core of every ray-tracing technique is the need to compute the intersection between a ray R and some surface S, the Ray-Surface Intersection (RSI) problem.

Primary rays are rays from the viewer or the eye through each pixel into the scene; they are typically evaluated in scan-line order, exploiting Z-buffer coherence to solve the RSI problem efficiently. Now consider the problem of casting rays from the points found in primary RSI towards the light sources in the scene, in order to detect regions that are in shadow.

Suppose we have in our scene a horizontal triangular polygon, $\mathcal{T}$, at depth $z = 1$, positioned above a horizontal rectangular polygon $\mathcal{P}$ at depth $z = 0$ (see Figure 30.8 (a)).

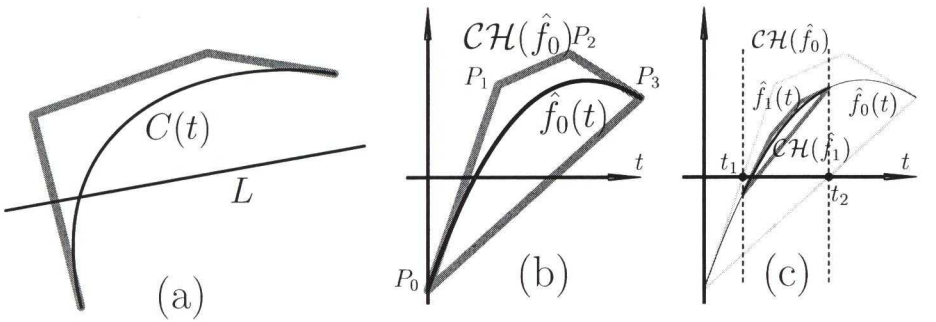

Figure 30.7. In (a), a cubic parametric Bézier curve is to be intersected with line L. The substitution of $C(t)$ into L is shown in (b) as an explicit function $\hat{f}_0(t)$. Clipping the convex hull domain of $\hat{f}_0(t)$ to be between t_1 and t_2 results in $\hat{f}_1(t)$, which is shown in (c).

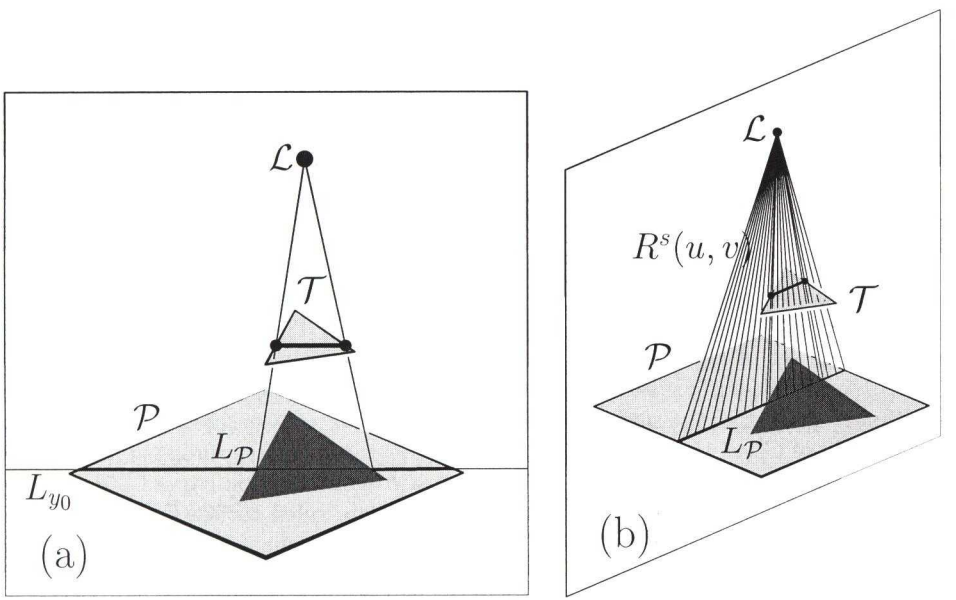

Figure 30.8. In (a), the shadowed regions of $\mathcal{P}$ due to $\mathcal{T}$ are sought. These regions are then separated from the illuminated domains along the current scan-line, L_{y_0}. In (b), a ruled surface $R^s(u,v)$ is constructed between $\mathcal{L}$ and $L_{\mathcal{P}}$, so that $R^s(u,v) \cap \mathcal{T}$ represents the portion of $\mathcal{U}$ that is in shadow.