Operations on parametric curves and surfaces: Intersection and Trimming

Lecture #6: Oct 14, 2002
Lecturer: Prof. Zorin
Scribe: Feng Ning
Reviewer: Prof. Zorin

In this lecture, we will discuss how to locate intersections of curves and surfaces and extract parts of surfaces bounded by curves (trimming).

1 Intersection of Curves

1.1 Introduction

Suppose we are given two parametric curves $f(u)$, $g(v)$, and we want to find the intersection points, that is, pairs of parameter values u_0, v_0, such that $f(u_0) = g(v_0)$. We consider 2D curves, so we can write $f(u) = [f_x(u)\ f_y(u)]^T$ and $g(v) = [g_x(v)\ g_y(v)]^T$. Then u_0, v_0 satisfy

$$f_x(u_0) - g_x(v_0) = 0$$
$$f_y(u_0) - g_y(v_0) = 0$$

If the curves are cubic Bezier curves, then this is a system of two cubic equations in two variables. Generally, there are two approaches to solve such polynomial equations.

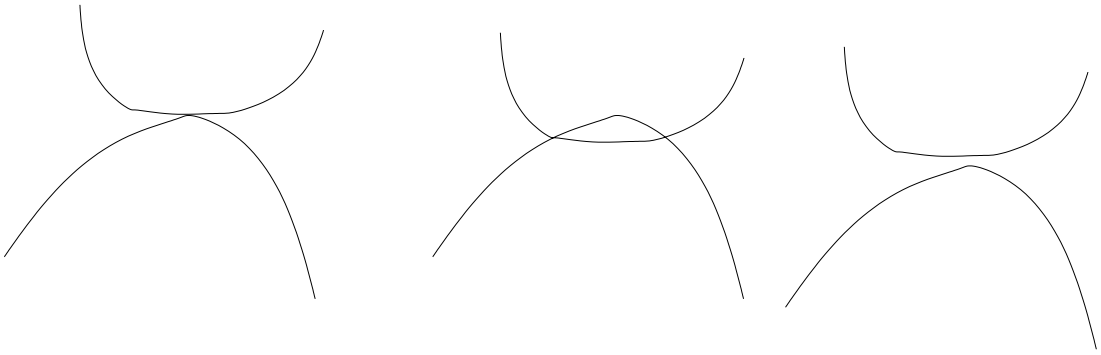
1. Algebraic approach using exact arithmetic. This approach is robust, but slow.

2. Numerical approach, using approximate calculations with machine accuracy. This approach is not robust. An example is given fig 1:

1.2 Intersection of a Bezier curve with a line

We start with a simple case to illustrate some general paradigms in the intersection algorithms. We assume that the Bezier curve is cubic throughout this section, though the method can be generalized to any degree. Since the curve is cubic, the number of intersection points can be from 0 to 3. Let the parametric form of the line be

$$\beta(t) = q_0 + rt \tag{1}$$



Left: two curves with 1 intersection point.

Mid, Right: after small perturbation, 2 or 0 intersection points can occur.

Figure 1: Numerical approach is not robust.

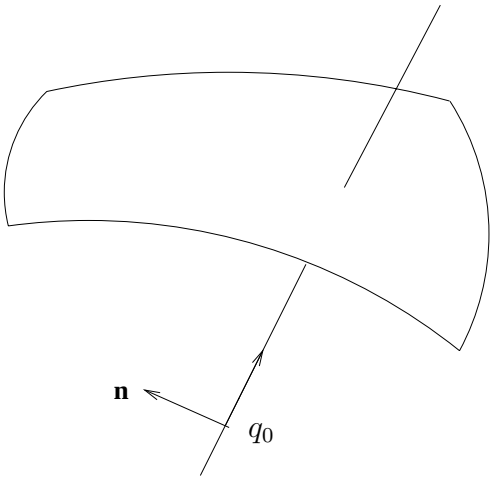


Figure 2: Intersection of a line and a surface.

Here q_0 is fixed point on the line, r is the direction vector.

The normal of the line is $n = [-r_y\ r_x]$ (the direction vector rotated by 90 degrees). The implicit equation for the line can be written as $n \cdot (p - q_0) = 0$, or

$$-r_y \cdot x + r_x \cdot y - (r_x q_0^y - r_y q_0^x) = 0$$

we use the notation $L(x, y) = ax + by + c$ for the right-hand side of the implicit line equation. The coordinate (x, y) of the intersection point of a cubic Bezier curve and the line will be the solution of the system

$$f(u) = [x\ y]^T \tag{2}$$
$$L(x, y) = 0 \tag{3}$$

where $f(u)$ is a cubic equation. Combining the two equations we obtain a cubic equation for u:

$$L(f_x(u), f_y(u)) = 0 \tag{4}$$

There is a formula to express explicitly the solutions of cubic equation, but it is relatively complex and not that easy to evaluate numerically in a stable way; furthermore there are no explicit formulas for degrees higher than five. In practice, iterative methods are typically used to solve equations of degree as low as as three. For example, *Newton's method* is a well-known iterative method used for solving nonlinear equations (see fig 3).

1. Pick up initial value x_0.

2. Apply the iterative formula $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ until required precision is achieved.

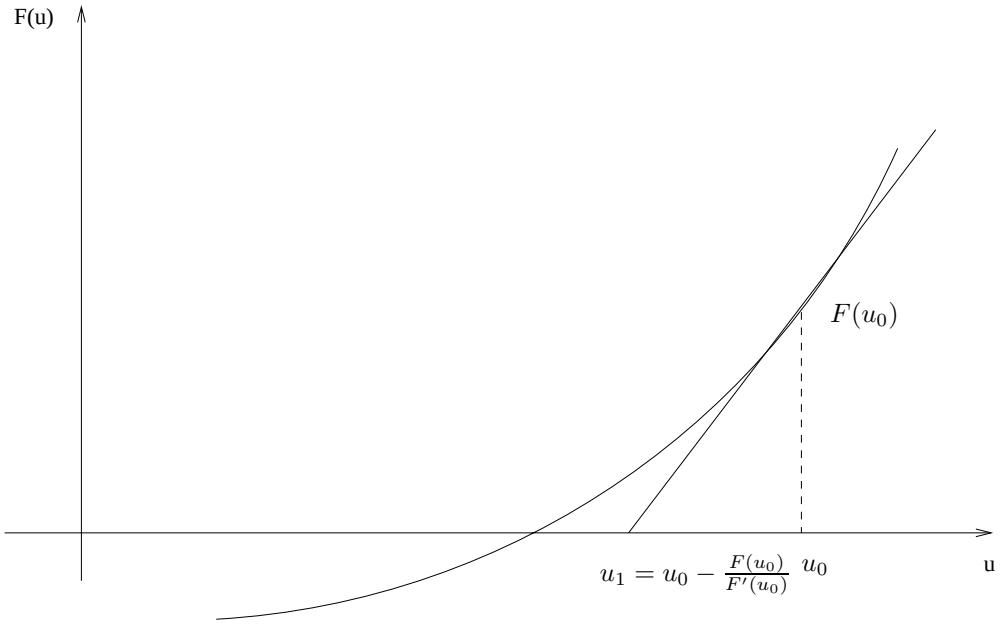


Figure 3: Newton Method.

Newton's method is easy to implement, it has potential problems:

1. It is not guaranteed to converge, if f is not twice differentiable, or the initial value is too far from the root, or $f'(0) = 0$ at the root.

2. It finds one solution, not all possible solutions.

1.3 Bezier clipping

We introduce a method called *Bezier clipping* which guarantees convergence and finds all solutions under certain conditions.

Let $B_i(u)$ be the cubic Bernstein polynomials. Since they form a basis for cubic polynomials, we can always write the equation $L(f_x(u), f_y(u)) = 0$ as as

$$\Sigma_{i=0}^{3} P_i B_i(u) = 0 \tag{5}$$

We will add one dimension, to make the above equation into equations:

$$\Sigma[\frac{i}{3}\ P_i]^T B_i(u) = [u\ 0]^T. \tag{6}$$

Recall the definition $B_i(u) = C_i^3(1-u)^{3-i}u^i$, we can directly verify the first identity.

So, instead of solving equation $L(f_x(u), f_y(u)) = 0$, we look for the intersecting points of curve C with u-axis. Here C is the 2D Bezier curve with control points $[0, P_0]^T$, $[\frac{1}{3}, P_1]^T$, $[\frac{2}{3}, P_2]^T$, $[1, P_3]^T$.

Initially, we know that the u value of intersection points is inside the interval $[0, 1]$ if it exists. Using the convex hull property of the Bezier curve, we observe that the intersection points have to be in the intersection of the convex hull of the control points and the u-axis. Suppose the convex hull intersects the u-axis at two points u_{min}, u_{max}, such that $0 \leq u_{min} \leq u_{max} \leq 1$. Then we know the intersection point(s) of the Bezier curve and u-axis will be inside $[u_{min}, u_{max}]$. Now we subdivide C into three segments, with u ranging in $[0, u_{min}]$, $[u_{min}, u_{max}]$, and $[u_{max}, 1]$ respectively. We know that $[0, u_{min}]$, $[u_{max}, 1]$ can be safely discarded. Now we can iterate by replacing $[0, 1]$ by $[u_{min}, u_{max}]$.

1.4 Intersection of Curves

Suppose we want to intersect two Bezier curves $f(u)$ and $g(v)$. By convex hull property, if $f(u)$ intersects with $g(v)$, then $f(u)$ has to intersect with convex hull of control points of $g(v)$. Since the convex hull is bounded by four line segments, we can use Bezier clipping to compute intersections how to compute the intersection with them. Once the two intersection points u_1, u_2 (with the convex hull) are determined, we can use them to clip $f(u)$ to a shorter segment $f(u) : u \in [u_1, u_2]$. (see fig 5) Now we reverse the role of f and g: we clip $g^1 = g$ against the convex hull of $f^2(u)$ to get $g^2(v)$.

Repeat the above procedure, we get two sequences of curve segments: $f^1(u)$, $f^2(u)$, $f^3(u)$... and $g^1(v)$, $g^2(v)$, $g^3(v)$ ... which will converge to the intersection points.

2 Intersection of Two Surfaces

Things become more complicated since the set of intersection points of two surfaces can have complex topology. For curves, the topology is very simple: the intersection is a finite set of

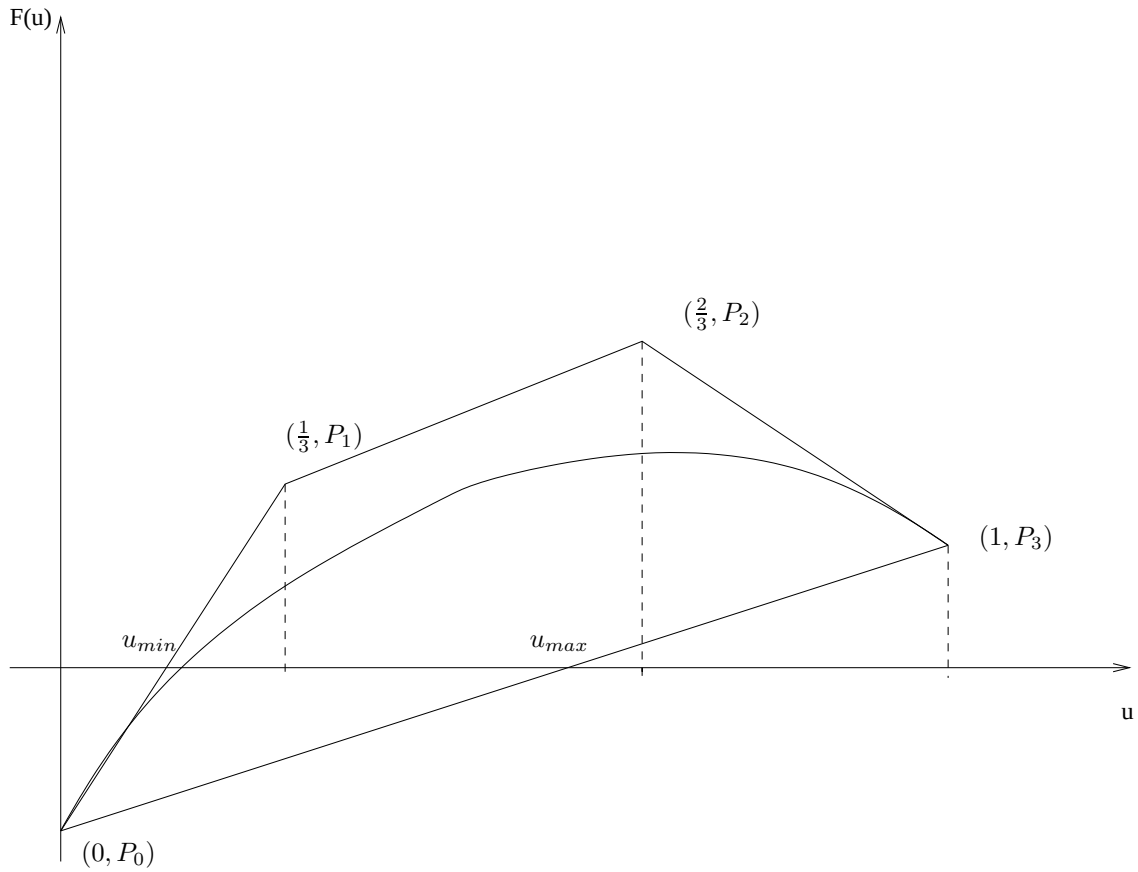


Figure 4: Bezier Clipping.

isolated points, unless parts of the curves coincide.

Let $f(u, v)$ and $g(s, t)$ be two surface patches, we make some assumptions:

1. A boundary curve E of $f(u, v)$ intersects $g(s, t)$.

2. The intersection of $f(u, v)$ and $g(s, t)$ is a simple curve, γ.

Then we can find the intersection curve by

1. Intersect E with $g(s, t)$ to get a point P_0 on γ.

2. From P_0 we step in the direction of the tangent to the intersection curve.

3. Then we locate the next point on the intersection curve by solving equations $f(u, v) - g(s, t) = 0$.

Comments:

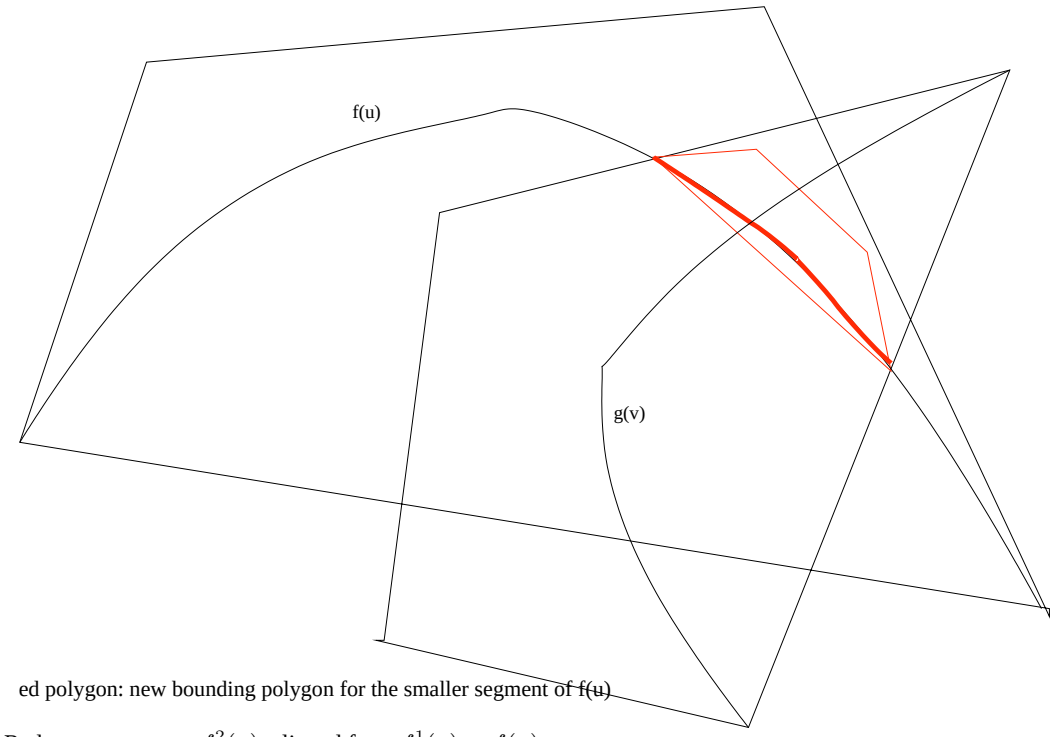


Figure 5: Intersecting two curves.

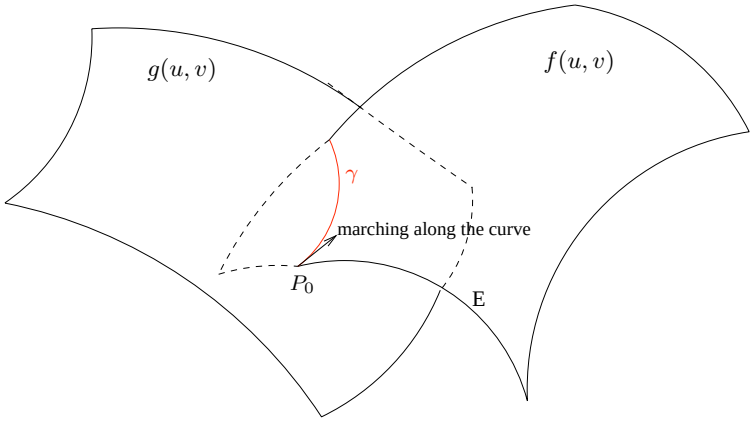


Figure 6: Intersecting surfaces

1. We can use Bezier clipping when we try to locate P_0.

2. The intersection set may or may not be a simple curve. For example, it can be a loop as shown in the figure below. This problem can be resolved by loop detection and subdivision, as it is discussed in the paper presented today. Once a loop is detected, we

subdivide the surface patch, until all the loops are separated into simple curves.

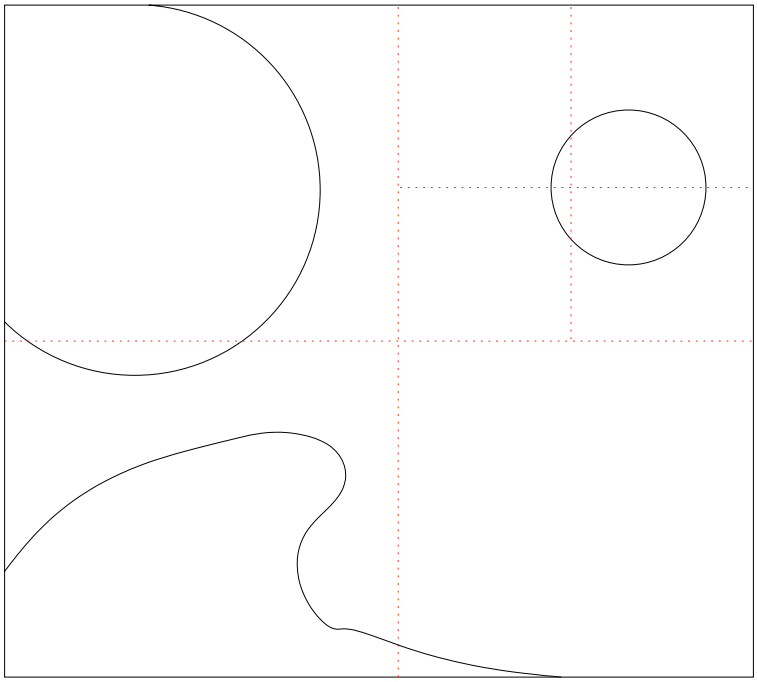


Figure 7: Subdividing surfaces to remove intersection loops.

3 Surface Trimming

Trimming can be used to create holes in patches. This operation is very important: once a hole is made, a handle can be attached along the boundary of the hole. In this way we can change the topology of the surface to have arbitrary genus.

Consider Bezier surface shown in the figure below. The traditional approach is to define a closed Bezier curve in the parameter domain of the surface: $f : R \rightarrow R^2, s \mapsto (u, v)$. Let us examine what happens when we do this in more detail. The boundary curve of the hole on the surface is the composite mapping $g(f(s))$. If the f is cubic Bezier curve and g is bi-cubic patch, then $g \circ f$ is of degree 9 in s. This leads to problems when we try to attach another patch along the curve:

1. If we continue using bi-cubic patches, then we have to match bi-degree 3 patch and the degree 9 curve along the hole boundary, which is not always possible. As the result, cracks can occur.

2. If we allow use higher order Bezier patches for handles, then we need degree 9 for the first handle attached. But if we want to attach a second handle to the first one, we will have to use a degree 27 patch. Required patch degree will grow exponentially.

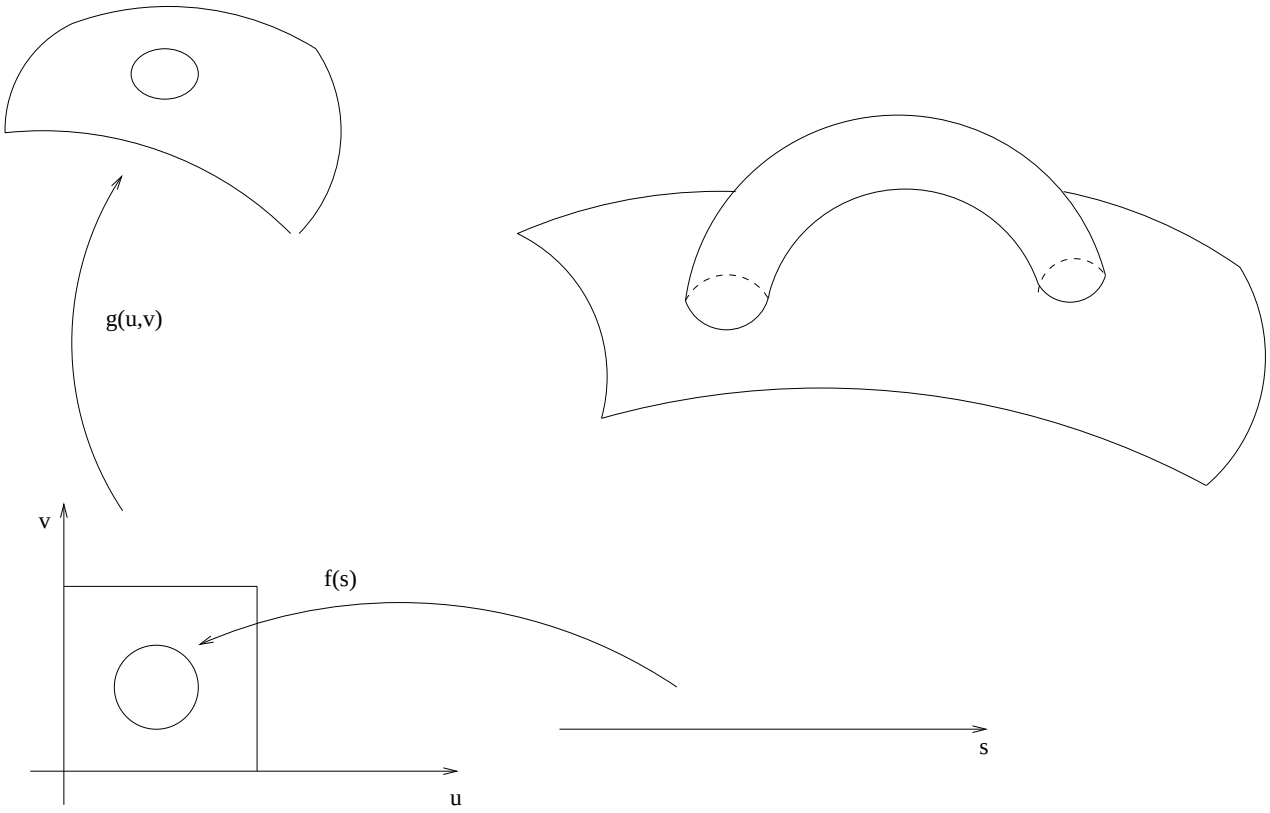


Figure 8: Trimming, attaching handles.

In practice, approximation is used instead of high order patch which leads to cracks in this type of surface representations.

Using subdivision surfaces provides an alternative solution. Instead of introducing a curve in the parameter domain to make a hole, we make a new mesh for the patch with a hole as shown in the figure below. In greater detail this problem is considered in a paper by Litke and others.

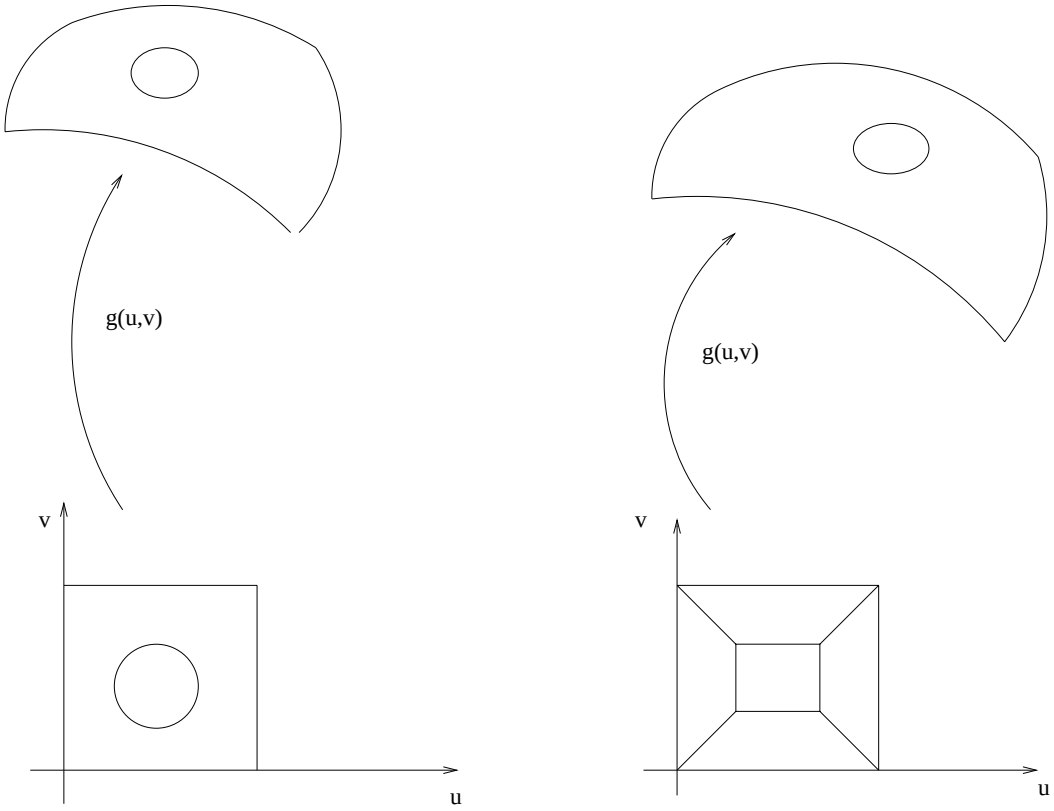


Figure 9: Subdivision approach to trimming.