Lecture 18: Surfaces I -- The General Theory

Thou hast set all the borders of the earth: Psalm 74:17

1. Surface Representations

Recursive ray tracing is a technique for displaying realistic images of objects bordered by surfaces. To render surfaces via ray tracing, we require two essential procedures: we must be able to compute surface normals and we need to calculate ray-surface intersections.

There are four standard ways to represent surfaces in Computer Graphics: implicit equations, parametric equations, deformations of known surfaces, and specialized procedures. In this lecture we shall review each of these general surface types and in each case explain how to compute surface normals and how to calculate ray-surface intersections. To facilitate further analysis, at the end of this lecture we shall also provide explicit formulas for mean and Gaussian curvature.

1.1 Implicit Surfaces. Planes, spheres, cylinders, and tori are examples of implicit surfaces. An *implicit surface* is the collection of all points P satisfying an implicit equation of the form $F(P) = 0$. Typically, the function F is a polynomial in the coordinates x, y, z of the points P. For example, a sphere can be represented by the implicit equation

$$F9x, y, z) \equiv x^2 + y^2 + z^2 - 1 = 0.$$

Generating lots of points along an implicit surface may be difficult because for complicated expressions F it may be hard to find points P for which $F(P) = 0$. On the other hand, determining if a point P lies on an implicit surface is easy, since we need only check if $F(P) = 0$. Moreover points on different sides of an implicit surface are distinguished by the sign of F; for closed surfaces $F(P) < 0$ may indicate points on the inside, whereas $F(P) > 0$ may indicate points on the outside. This ability to distinguish inside from outside is important in solid modeling applications.

1.2 Parametric Surfaces. Planes, sphere, cylinders, and tori are also examples of parametric surfaces. A *parametric surface* is a surface represented by parametric equations -- that is, to each point P on the surface, we assign a pair of parameter values s,t so that there is a formula $P = P(s,t)$ for computing points along the surface. In terms of rectangular coordinates, the equation $P = P(s,t)$ is equivalent to three parametric equations for the coordinates: $x = p_1(s,t), y = p_2(s,t), z = p_3(s,t)$. In Computer Graphics, the functions $p_1(s,t), p_2(s,t), p_3(s,t)$ are typically either polynomials or rational functions -- ratios of polynomials -- in the parameters s,t. For example, a sphere can be represented by the parametric equations

$$x(s,t) = \frac{2s}{1+s^2+t^2} \qquad y(s,t) = \frac{2t}{1+s^2+t^2} \qquad z(s,t) = \frac{1-s^2-t^2}{1+s^2+t^2}$$

since it is easy to verify that

$$x^2(s,t) + y^2(s,t) + z^2(s,t) = 1.$$

Generating lots of points along a parametric surface is straightforward: simply substitute lots of different parameter values s,t into the expression $P = P(s,t)$. Thus parametric surfaces are easy to display, so parametric surfaces are a natural choice for Computer Graphics. However, determining if a point P is on a parametric surface is not so simple, since it may be difficult to determine whether or not there are parameters s,t for which $P = P(s,t)$.

1.3 Deformed Surfaces. An ellipsoid is a deformed sphere; an elliptical cone is a deformed circular cone. Any surface generated by deforming another surface is called a *deformed surface*. The advantage of deformed surfaces is that deformations often permit us to represent complicated surfaces in terms of simpler surfaces. This device can lead to easier analysis algorithms for complicated shapes. For example, we shall see in the next lecture that ray tracing an ellipsoid can be reduced to ray tracing a sphere.

If the original surface has an implicit representation $F_{old}(P) = 0$ or a parametric representation $S_{old}(u,v)$, then the deformed surface also has an implicit representation $F_{new}(P) = 0$ or a parametric representation $S_{new}(u,v)$; moreover, we can compute $F_{new}(P)$ and $S_{new}(u,v)$ from $F_{old}(P)$ and $S_{old}(u,v)$. In fact, if M is a nonsingular transformation matrix that maps the original surface into the deformed surface, then

$$F_{new}(P) = F_{old}(P * M^{-1}) \qquad (1.1)$$

$$S_{new}(u,v) = S_{old}(u,v) * M. \qquad (1.2)$$

Equation (1.2) is easy to understand because M maps points on S_{old} to points on S_{new}. Equation (1.1) follows because P is a point on the deformed surface if and only if $P * M^{-1}$ is a point on the original surface. Thus

$$F_{new}(P) = 0 \Leftrightarrow F_{old}(P * M^{-1}) = 0.$$

Equations (1.1) and (1.2) are valid for any affine transformation M of the original surface. Thus the map M can be a rigid motion -- a translation or a rotation -- so we can use Equations (1.1) and (1.2) to reposition as well as to deform a surface.

1.4 Procedural Surfaces. Fractal surfaces are examples of procedural surfaces. Typically fractals are not represented by explicit formulas; instead fractals are represented either by recursive procedures or by iterated function systems. Any surface represented by a procedure instead of a formula is called a *procedural surface*. Often surfaces that blend between other surfaces are

represented by procedures for generating the blend rather than by explicit formulas, so blends are another important class of procedural surfaces. Though procedural surfaces are important in Computer Graphics, ray tracing procedural surfaces requires specialized algorithms that depend on the particular procedure used to generate the surface. Since we are interested here in general ray tracing algorithms, we shall not have much to say about ray tracing for procedural surfaces.

2. Surface Normals

For implicit, parametric, and deformed surfaces there are straightforward, explicit formulas for calculating normal vectors. We present these general formulas below.

2.1 Implicit Surfaces. For an implicit surface $F(x,y,z)=0$, the normal vector N is given by the gradient -- that is,

$$N = \nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right). \tag{2.1}$$

The proof that the gradient is normal to the surface is just the chain rule. Suppose that $P(t) = \big(x(t), y(t), z(t)\big)$ is a parametric curve on the surface $F(x,y,z)=0$. Then

$$F\big(x(t), y(t), z(t)\big) = 0.$$

Hence by the chain rule:

$$\frac{\partial F}{\partial x}\frac{dx}{dt} + \frac{\partial F}{\partial y}\frac{dy}{dt} + \frac{\partial F}{\partial z}\frac{dz}{dt} = \frac{dF}{dt} = 0.$$

Therefore

$$\underbrace{\nabla F}_{Gradiant} \bullet \underbrace{P'(t)}_{Derivative} = \underbrace{\left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right)}_{Normal\ vector} \bullet \underbrace{\left(\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{\partial t}\right)}_{Tangent\ vector} = 0,$$

so

$$N = \nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right).$$

2.2 Parametric Surfaces. For a parametric surface $P(s,t) = \big(x(s,t), y(s,t), z(s,t)\big)$, the normal vector N is given by the cross product of the partial derivatives of $P(s,t)$ -- that is

$$N = \frac{\partial P}{\partial s} \times \frac{\partial P}{\partial t} = \left(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s}\right) \times \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t}\right). \tag{2.2}$$

To derive Equation (2.2), observe that if we fix $t = t_0$, then

$$P(s) = P(s,t_0) = \big(x(s,t_0), y(s,t_0), z(s,t_0)\big)$$

is a curve on the surface $P(s,t)$, and the tangent to this curve is given by

$$\frac{\partial P}{\partial s} = \left(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s} \right).$$

Similarly, if we fix $s = s_0$, then

$$P(t) = P(s_0, t) = \left(x(s_0, t), y(s_0, t), z(s_0, t) \right)$$

is another curve on the surface $P(s,t)$, and the tangent to this curve is given by

$$\frac{\partial P}{\partial t} = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t} \right).$$

Since the normal to the surface is orthogonal to the surface, the normal vector must be perpendicular to the tangent vector for every curve on the surface. Therefore, since the cross product of two vectors is orthogonal to the two vectors,

$$N = \frac{\partial P}{\partial s} \times \frac{\partial P}{\partial t} = \left(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s} \right) \times \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t} \right).$$

2.3. Deformed Surfaces. Consider a surface S_{new} that is the image of another surface S_{old} under a nonsingular 4×4 affine transformation matrix *M*. Recall that if

$$M = \begin{pmatrix} M_u & 0 \\ w & 1 \end{pmatrix},$$

then only the upper 3×3 linear transformation matrix M_u affects tangent vectors to the surface, since vectors an unaffected by translation. Thus if v_{old} is tangent to S_{old}, then the corresponding tangent v_{new} to the surface S_{new} is given by

$$v_{new} = v_{old} * M_u .$$

For rotations normal vectors transform in the same way as tangent vectors, since rigid motions preserve orthogonality. Similarly, uniform scaling preserves orthogonality, since uniform scaling preserves angles even though scaling does not preserve length. Thus for all conformal transformations -- that is, for all composites of rotation, translation, and uniform scaling -- the normal vector transforms in the same way as the tangent vector. Hence, for conformal maps

$$v_{new} = v_{old} * M_u$$
$$N_{new} = N_{old} * M_u .$$

Nevertheless, perhaps surprisingly, the general transformation rule for normal vectors is different from the general transformation rule for tangent vectors, since non-conformal maps such as non-uniform scaling do not preserve orthogonality (see Figure 1). For arbitrary transformations *M*, if we know the normal vector N_{old} to the surface S_{old}, then we can compute the corresponding normal vector N_{new} to the surface S_{new} by the formula

$$N_{new} = N_{old} * M_u^{-T} .$$

For rotations the transpose is equivalent to the inverse, so for rotations $M_u^{-T} = M_u$. Similarly, if M_u is uniform scaling by the scale factor s, then M_u^{-T} is uniform scaling by the scale factor $1/s$. However, for arbitrary transformations, M_u^{-T} can be very different from M_u.

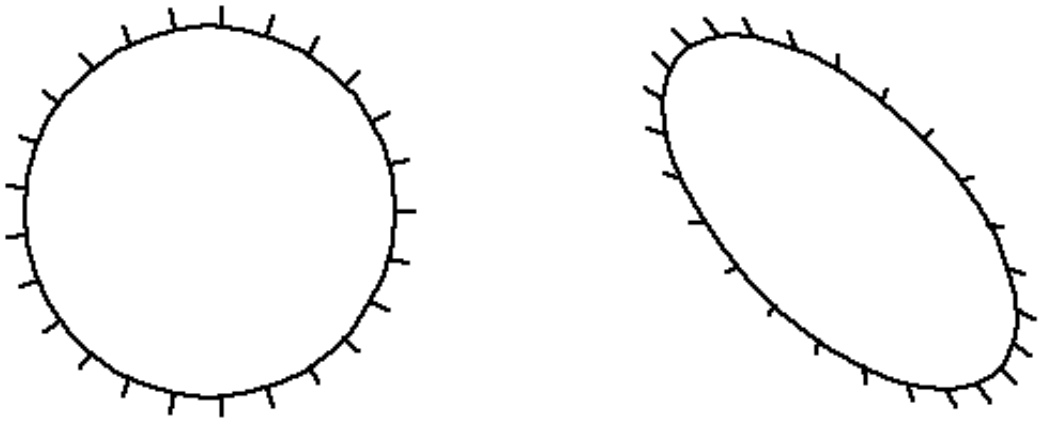

Figure 1: A circle with its normal vectors, and the circle transformed into an ellipse by scaling non-uniformly by the factor 1/2 along the minor axis. Notice that, in general, non-uniform scaling does not map normal vectors to the circle to normal vectors to the ellipse.

The general transformation formula for the normal vector is a consequence of the following observations. Since N_{old} is orthogonal to S_{old}, we know that the normal vector N_{old} is perpendicular to the tangent vector v_{old}. Therefore by the properties of the dot product,

$$v_{old} * N_{old}^T = v_{old} \bullet N_{old} = 0.$$

Now we seek a vector N_{new} such that

$$v_{new} * N_{new}^T = v_{new} \bullet N_{new} = 0.$$

But we can easily verify that if $N_{new} = N_{old} * M_u^{-T}$, then $v_{new} * N_{new}^T = 0$ because

$$v_{new} * N_{new}^T = (v_{old} * M_u) * \left(N_{old} * M_u^{-T}\right)^T = (v_{old} * M_u) * \left(M_u^{-1} * N_{old}^T\right) = v_{old} * N_{old}^T = 0.$$

To summarize: using affine coordinates, points and tangent vectors transform by the same rule:

$$(P_{new},1) = (P_{old},1) * M$$

$$(v_{new},0) = (v_{old},0) * M,$$

but tangent vectors and normal vectors transform by different rules:

$$v_{new} = v_{old} * M_u$$
$$N_{new} = N_{old} * M_u^{-T} .$$

Moreover, inserting a zero affine coordinate for normal vectors will not work because the fourth coordinate of $(N_{old},0) * M^{-T}$ is not zero. Thus normal vectors are qualitatively different from tangent vectors. The single term *vector* is not rich enough to capture the full diversity of different kinds of vectors. Both in differential geometry and in modern physics this distinction between tangent vectors and normal vectors is emphasized by calling tangent vectors *covariant vectors* and normal vectors *contravariant vectors*.

3. Ray-Surface Intersections

Ray tracing a surface requires computing the intersection points of an arbitrary line with the surface as well as calculating the corresponding parameter values along the line of these intersection points. For implicit, parametric, and deformed surfaces, there are straightforward algorithms for finding these intersection points along with their corresponding parameter values.

3.1 Implicit Surfaces. Many of the surfaces encountered in recursive ray tracing have implicit representations $F(P) = 0$. Let $L(t) = P + t\,v$ be the parametric equation of a line. Then we can compute the parameters along the line of the intersection points of the ray and the implicit surface by solving the equation $F(L(t)) = 0$. Substituting the roots of this equation into the parametric equation $L(t) = P + t\,v$ for the line yields the actual intersection points. Thus we have the following intersection algorithm.

Ray-Surface Intersection Algorithm -- Implicit Surfaces

1. Solve $F(L(t)) = 0$.
   The roots $t = t_1,\ldots,t_n$ are the parameter values along the line of the actual intersection points.
2. Compute $R_i = L(t_i) = P + t_i\,v$, $i = 1,\ldots,n$.
   The values $R_1,\ldots,R_n$ are the actual intersection points of the line and the surface.

The bottleneck in this algorithm is clearly step 1, since the equation $F(L(t)) = 0$ may be difficult to solve. When the function F is a polynomial in the coordinates x,y,z, then $F(L(t)) = 0$ is a univariate polynomial equation. For low degree polynomials (degree<5), there are explicit formulas for the roots; for high degree polynomials a numerical method can be used to solve this equation. Thus implicit polynomial surfaces are a natural choice for recursive ray tracing.

3.2 Parametric Surfaces. Ray tracing parametric surfaces is more difficult than ray tracing implicit surfaces. Let $S(u,v) = \big(x(u,v), y(u,v), z(u,v)\big)$ be a parametric surface, and let $L(t) = P + tv$ be the parametric equation of a line. The line and the surface intersect whenever there are parameter values t,u,v for which $S(u,v) = L(t)$. This equation really represents three equations -- one for each coordinate -- in three unknown -- t,u,v. Thus to find the intersection points of the line and the surface, we must solve three equations in three unknowns. Substituting the solutions in t into the parametric equation $L(t) = P + tv$ for the line or the solutions in u,v into the parametric equations $S(u,v) = \big(x(u,v), y(u,v), z(u,v)\big)$ of the surface yields the actual intersection points. Thus we have the following intersection algorithm.

Ray-Surface Intersection Algorithm -- Parametric Surfaces

1. Solve $S(u,v) = L(t)$ for the parameters t,u,v. That is, solve the three simultaneous equations:
   $x(u,v) = p_1 + tv_1 \qquad y(u,v) = p_2 + tv_2 \qquad z(u,v) = p_3 + tv_3.$
   The roots $t = t_1,\ldots,t_n$ are the parameter values along the line of the actual intersection points.
2. Compute $R_i = L(t_i) = P + t_i v$, $i = 1,\ldots,n$, or equivalently $R_i = S(u_i, v_i)$, $i = 1,\ldots,n$.
   The values $R_1,\ldots,R_n$ are the actual intersection points of the line and the surface.

The bottleneck in this algorithm is again clearly step 1, since the simultaneous equations $S(u,v) = L(t)$ may be difficult to solve. We can simplify from three equations in three unknown to two equations in two unknowns in the following fashion. Let N_1, N_2 be two linearly independent vectors perpendicular to v. Then $v \bullet N_1 = v \bullet N_2$. Therefore dotting the equation $S(u,v) = L(t)$, with N_1, N_2 eliminates t, the coefficient of v. We are left with two equations in two unknowns:

$$\big(S(u.v) - P\big) \bullet N_1 = 0$$
$$\big(S(u.v) - P\big) \bullet N_2 = 0$$

which we now must solve for u,v. Even with this simplification, however, we must still solve two non-linear equations in two unknowns. Thus, typically ray tracing for parametric surfaces, even for parametric polynomial surfaces, requires more sophisticated numerical root finding techniques than ray tracing for implicit surfaces.

3.3 Deformed Surfaces. Ray tracing deformed surfaces is often easy because the deformed surface is usually the image of a simple surface which we already know how to ray trace. Suppose that S_{new} is the image under a nonsingular transformation matrix M of the surface S_{old} -- that is, $S_{new} = S_{old} * M$. For ray tracing the key observation is that intersecting a line L with the surface S_{new} is equivalent to intersecting the line $L * M^{-1}$ with the surface $S_{new} * M^{-1} = S_{old}$. More precisely, a line L and the surface S_{new} intersect at a point P if and only if the line $L * M^{-1}$ and the

surface $S_{new} * M^{-1} = S_{old}$ intersect at the point $P * M^{-1}$ Thus for deformed surfaces, we have the following intersection algorithm.

Ray-Surface Intersection Algorithm -- Deformed Surfaces

1. Transform the line *L* by the matrix M^{-1}.
   If $L(t) = P + t\,v$, then $L(t) * M^{-1} = P * M^{-1} + t(v * M^{-1})$.
2. Find the intersection points $Q_1,\ldots,Q_n$ and the corresponding parameter values $t = t_1,\ldots,t_n$ of the intersection of the line $L * M^{-1}$ and the surface S_{old}.
3. Compute $R_i = L(t_i) = P + t_i\,v$, $i = 1,\ldots,n$ or equivalently $R_i = Q_i * M$, $i = 1,\ldots,n$.
   The values $R_1,\ldots,R_n$ are the actual intersection points of the line *L* and the surface S_{new}, and $t = t_1,\ldots,t_n$ are the corresponding parameter values along the line *L*.

The main difficulty in this algorithm is step 2, since we must know how to find the intersections of an arbitrary line with the surface S_{old}. This intersection algorithm works well whenever the surface S_{old} is easy to ray trace.

4. Mean and Gaussian Curvature

To fully analyze surfaces, in addition to expressions for the normal vectors, curvature formulas are often required. For easy reference we present here without proof formulas for the mean and Gaussian curvature of implicit and parametric surfaces. Rigorous definitions and derivations are beyond the scope of this lecture. Formal mathematical foundations for surface curvature can be found in most classical books on differential geometry.

4.1 Implicit Surfaces. For an implicit surface $F(x,y,z) = 0$, let ∇F denote the gradient and $hess(F)$ denote the hessian of *F* -- that is, let

$$\nabla F = \begin{pmatrix} \dfrac{\partial F}{\partial x} & \dfrac{\partial F}{\partial y} & \dfrac{\partial F}{\partial z} \end{pmatrix}$$

$$hess(F) = \begin{pmatrix} \dfrac{\partial^2 F}{\partial x^2} & \dfrac{\partial^2 F}{\partial x \partial y} & \dfrac{\partial^2 F}{\partial x \partial z} \\ \dfrac{\partial^2 F}{\partial x \partial y} & \dfrac{\partial^2 F}{\partial y^2} & \dfrac{\partial^2 F}{\partial y \partial z} \\ \dfrac{\partial^2 F}{\partial x \partial z} & \dfrac{\partial^2 F}{\partial y \partial z} & \dfrac{\partial^2 F}{\partial z^2} \end{pmatrix}.$$

In addition, let $hess^*(F)$ denote the adjoint of $hess(F)$ -- the matrix generated by replacing each entry of $hess(F)$ by its cofactor. Then the Gaussian curvature is given by

$$K = \frac{\nabla F * hess^*(F) * \nabla F^T}{|\nabla F|^3} = -\frac{Det\begin{pmatrix} hess(F) & \nabla F^T \\ \nabla F & 0 \end{pmatrix}}{|\nabla F|^4}, \tag{4.1}$$

and the mean curvature is given by

$$H = \frac{|\nabla F|^2 Trace(hess(F)) - \nabla F * hess(F) * \nabla F^T}{|\nabla F|^3} = \nabla \bullet \left(\frac{\nabla F}{|\nabla F|}\right), \tag{4.2}$$

where $\nabla \bullet G = \frac{\partial G}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial G}{\partial z}$ denotes the divergence of G.

For planar implicit curves $F(x,y) = 0$, all of these formulas for the mean and Gaussian curvature reduce to the standard curvature for a planar curve.

4.2 Parametric Surfaces. For a parametric surface $P(s,t) = (x(s,t), y(s,t), z(s,t))$, define the first and second fundamental forms by

$$I = \begin{pmatrix} \frac{\partial P}{\partial s} \bullet \frac{\partial P}{\partial s} & \frac{\partial P}{\partial s} \bullet \frac{\partial P}{\partial t} \\ \frac{\partial P}{\partial s} \bullet \frac{\partial P}{\partial t} & \frac{\partial P}{\partial t} \bullet \frac{\partial P}{\partial t} \end{pmatrix}$$

$$II = \begin{pmatrix} \frac{\partial^2 P}{\partial s^2} \bullet N & \frac{\partial^2 P}{\partial s \partial t} \bullet N \\ \frac{\partial^2 P}{\partial s \partial t} \bullet N & \frac{\partial^2 P}{\partial t^2} \bullet N \end{pmatrix}$$

where

$$N(s,t) = \frac{\frac{\partial P}{\partial s} \times \frac{\partial P}{\partial t}}{\left|\frac{\partial P}{\partial s} \times \frac{\partial P}{\partial t}\right|},$$

is a unit normal to the surface. Then the Gaussian curvature

$$K = \frac{Det(II)}{Det(I)} \tag{4.3}$$

and the mean curvature

$$H = \frac{Trace(I * II^*)}{2Det(I)}, \tag{4.4}$$

where II^* is the adjoint of II.

4.3 Deformed Surfaces. If M is a nonsingular transformation matrix that maps some original surface in implicit or parametric form into a deformed surface, then by Equations (1.1) and (1.2)

$$F_{new}(P) = F_{old}(P * M^{-1})$$

$$S_{new}(u,v) = S_{old}(u,v) * M .$$

Since we already have formulas for the mean and Gaussian curvature both for implicit and for parametric surfaces, we can calculate the mean and Gaussian curvature of deformed surfaces from their implicit or parametric equations -- that is, from F_{new} or S_{new}.

5. Summary

The three most common types for surfaces in Computer Graphics are implicit, parametric, and deformed surfaces. For each of these surface representations we have explicit formulas for the normal vectors and curvatures as well as straightforward algorithms for calculating ray-surface intersections. For easy reference we summarize these formulas and algorithms below.

5.1 Implicit Surfaces.

Surface Representation

$$F(x,y,z) = 0$$

Normal Vector

$$N = \nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right)$$

Hessian

$$hess(F) = \begin{pmatrix} \frac{\partial^2 F}{\partial x^2} & \frac{\partial^2 F}{\partial x \partial y} & \frac{\partial^2 F}{\partial x \partial z} \\ \frac{\partial^2 F}{\partial x \partial y} & \frac{\partial^2 F}{\partial y^2} & \frac{\partial^2 F}{\partial y \partial z} \\ \frac{\partial^2 F}{\partial x \partial z} & \frac{\partial^2 F}{\partial y \partial z} & \frac{\partial^2 F}{\partial z^2} \end{pmatrix}$$

Gaussian Curvature

$$K = \frac{\nabla F * hess^*(F) * \nabla F^T}{|\nabla F|^3} = -\frac{Det\begin{pmatrix} hess(F) & \nabla F^T \\ \nabla F & 0 \end{pmatrix}}{|\nabla F|^4},$$

Mean Curvature

$$H = \frac{Trace(I * II^*)}{2Det(I)}$$

Ray-Surface Intersection Algorithm

1. Solve $F(L(t)) = 0$.

The roots $t = t_1, \ldots, t_n$ are the parameter values along the line of the actual intersection points.

2. Compute $R_i = L(t_i) = P + t_i v$, $i = 1, \ldots, n$.

The values $R_1, \ldots, R_n$ are the actual intersection points of the line and the surface.

5.2 Parametric Surfaces.

Surface Representation

$$P(s,t) = \left(x(s,t), y(s,t), z(s,t)\right)$$

Normal Vector

$$N = \frac{\partial P}{\partial s} \times \frac{\partial P}{\partial t} = \left(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s}\right) \times \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t}\right)$$

First Fundamental Form

$$I = \begin{pmatrix} \frac{\partial P}{\partial s} \bullet \frac{\partial P}{\partial s} & \frac{\partial P}{\partial s} \bullet \frac{\partial P}{\partial t} \\ \frac{\partial P}{\partial s} \bullet \frac{\partial P}{\partial t} & \frac{\partial P}{\partial t} \bullet \frac{\partial P}{\partial t} \end{pmatrix}$$

Second Fundamental Form

$$II = \begin{pmatrix} \frac{\partial^2 P}{\partial s^2} \bullet N & \frac{\partial^2 P}{\partial s \partial t} \bullet N \\ \frac{\partial^2 P}{\partial s \partial t} \bullet N & \frac{\partial^2 P}{\partial t^2} \bullet N \end{pmatrix}$$

Gaussian Curvature

$$K = \frac{Det(II)}{Det(I)}$$

Mean Curvature

$$H = \frac{Trace(I * II^*)}{2Det(I)}$$

Ray-Surface Intersection Algorithm

1. Solve $S(u,v) = L(t)$ for the parameters t,u,v. That is, solve the three simultaneous equations:

   $x(u,v) = p_1 + t v_1 \qquad y(u,v) = p_2 + t v_2 \qquad z(u,v) = p_3 + t v_3$,

   or equivalently the two simultaneous equations

   $\big(S(u.v) - P\big) \bullet N_1 = 0$

   $\big(S(u.v) - P\big) \bullet N_2 = 0$

   where $N_1 . N_2$ are two linearly independent vectors perpendicular to *v*.

The roots $t = t_1,\ldots,t_n$ are the parameter values along the line of the actual intersection points.

2. Compute $R_i = L(t_i) = P + t_i v$, $i = 1,\ldots,n$, or equivalently $R_i = S(u_i, v_i)$, $i = 1,\ldots,n$.

The values $R_1,\ldots,R_n$ are the actual intersection points of the line and the surface.

5.3 Deformed Surfaces.

Surface Representation

An implicit surface $F_{old}(P) = 0$ or a parametric surface $S_{old}(u,v) = \big(x(u,v), y(u,v), z(u,v)\big)$ together with a deformation matrix $M = \begin{pmatrix} M_u & 0 \\ v & 1 \end{pmatrix}$.

Implicit Representation	*Parametric Representation*
$F_{new}(P) = F_{old}(P * M^{-1})$	$S_{new}(u,v) = S_{old}(u,v) * M$

Normal Vector

$$N_{new} = N_{old} * M_u^{-T}$$

Mean and Gaussian Curvature

From implicit or parametric representations.

Ray-Surface Intersection Algorithm

1. Transform the line *L* by the matrix M^{-1}.
   If $L(t) = P + t v$, then $L(t) * M^{-1} = P * M^{-1} + t(v * M^{-1})$.
2. Find the intersection points $Q_1,\ldots,Q_n$ and the corresponding parameter values $t = t_1,\ldots,t_n$ of the intersection of the line $L(t) * M^{-1}$ and the surface F_{old} or S_{old}.

3. Compute $R_i = L(t_i) = P + t_i v$, $i = 1,\ldots,n$ or equivalently $R_i = Q_i * M$, $i = 1,\ldots,n$. The values $R_1,\ldots,R_n$ are the actual intersection points of the line L and the surface F_{new} or S_{new}, and $t = t_1,\ldots,t_n$ are the corresponding parameter values along the line L.

Exercises:

1. Suppose that u,v are tangent vectors. Let M be a 3×3 matrix, and let M^* be the adjoint of M.
   a. Give an example to show that $(u \times v) * M \neq (u * M) \times (v * M)$.
   b. Show that for all u,v,M $(u \times v) * M = (u * M^*) \times (v * M^*)$.

2. Let $D(s) = (x(s), y(s), z(s))$ be a parametrized curve lying in a plane in 3-space, and let $v = (v_1, v_2, v_3)$ be a fixed vector in 3-space not in the plane of $D(s)$. The parametric surface:
$$C(s,t) = D(s) + t\,v,$$
is called the *generalized cylinder* over the curve $D(s)$. Compute the normal $N(s,t)$ at an arbitrary point on the cylinder $C(s,t)$.

3. Let $D(s) = (x(s), y(s), z(s))$ be a parametrized curve lying in a plane in 3-space, and let $Q = (q_1, q_2, q_3)$ be a fixed point in 3-space not in the plane of $D(s)$. The parametric surface:
$$C(s,t) = (1-t)\,D(s) + t\,Q,$$
is called the *generalized cone* over the curve $D(s)$. Compute the normal $N(s,t)$ at an arbitrary point on the cone $C(s,t)$.

4. Let $U_1(s), U_2(s)$ be two curves in 3-space. The parametric surface
$$R(s,t) = (1-t)U_1(s) + t\,U_2(s)$$
is called the *ruled surface* generated by the curves $U_1(s), U_2(s)$. Compute the normal $N(s,t)$ at an arbitrary point on the ruled surface $R(s,t)$.

5. Consider the surface defined by the implicit equation
$$xyz - 1 = 0.$$
   a. Find the normal vector to this surface at the point $P = (2, 1, 0.5)$.
   b. Find the implicit equation of this surface after rotating the surface by $\pi/6$ radians around the z-axis and then translating the surface by the vector $v = (1,3,2)$.

6. Let S be a sphere of radius R. The mean and Gaussian curvature of S are given by

$$H = \frac{1}{R} \quad \text{and} \quad K = \frac{1}{R^2}.$$

Derive these curvature formulas in two ways:

i. From the implicit equation

$$x^2 + y^2 + z^2 - R^2 = 0.$$

ii. From the parametric equations

$$x(s,t) = \frac{2Rs}{1+s^2+t^2} \qquad y(s,t) = \frac{2Rt}{1+s^2+t^2} \qquad z(s,t) = \frac{R(1-s^2-t^2)}{1+s^2+t^2}.$$