

Lecture 12: Introduction to Surface Construction

Non-Parametric Surfaces

We now turn to the question of how to represent and draw surfaces. As was the case with constructing spline curves, one possibility is to adopt the simple solution of non-parametric Cartesian equations. A quadratic surface would have an equation of the form:

$$\begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} a & b & c & d \\ b & e & f & g \\ c & f & h & j \\ d & g & j & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = 0$$

which multiplies out to:

$$ax^2 + ey^2 + hz^2 + 2bxy + 2cxz + 2fyz + 2dx + 2gy + 2jz + 1 = 0$$

and the nine scalar unknowns $\{a, b, \dots, j\}$ can be found by specifying nine points $\mathbf{P}_i = (x_i, y_i, z_i)$, $i = 1, \dots, 9$, through which the surface must pass. This creates a system of nine linear equations to solve. Notice that because of the symmetry of the matrix there are only nine unknowns, not sixteen. The constant term can always be taken as 1 without loss of generality.

This method however suffers the same limitations as the analogous method for curves. It is difficult to control the surface shape since there is only one quadratic surface that will fit the points.

Parametric Surfaces

It is simple to generalise from parametric curves to parametric surfaces. In the case of spline curves, the locus of a point was a function of one parameter. However a point on a surface patch is a function of two parameters, which we write as $\mathbf{P}(\mu, \nu)$. One way to do this is to define a parametric surface using a matrix formulation:

$$\mathbf{P}(\mu, \nu) = \begin{pmatrix} \mu & \nu & 1 \end{pmatrix} \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \\ \mathbf{b} & \mathbf{d} & \mathbf{e} \\ \mathbf{c} & \mathbf{e} & \mathbf{f} \end{pmatrix} \begin{pmatrix} \mu \\ \nu \\ 1 \end{pmatrix}$$

Multiplying out we get:

$$\mathbf{P}(\mu, \nu) = \mathbf{a}\mu^2 + \mathbf{d}\nu^2 + 2\mathbf{b}\mu\nu + 2\mathbf{c}\mu + 2\mathbf{e}\nu + \mathbf{f} \quad (1)$$

The values of the constant vectors $\{\mathbf{a}, \mathbf{b}, \dots, \mathbf{f}\}$ determine the shape of the surface. The surface has edges given by the four curves for which one of the parameters ν and μ is constant at either 0 or 1. These are the quadratics:

$$\begin{aligned} \mathbf{P}(0, \nu) &= \mathbf{d}\nu^2 + 2\mathbf{e}\nu + \mathbf{f} \\ \mathbf{P}(1, \nu) &= \mathbf{a} + 2(\mathbf{b} + \mathbf{e})\nu + 2\mathbf{c} + \mathbf{d}\nu^2 + \mathbf{f} \\ \mathbf{P}(\mu, 0) &= \mathbf{a}\mu^2 + 2\mathbf{c}\mu + \mathbf{f} \\ \mathbf{P}(\mu, 1) &= \mathbf{a}\mu^2 + 2(\mathbf{b} + \mathbf{c})\mu + \mathbf{d} + 2\mathbf{e} + \mathbf{f} \end{aligned}$$

The unknown values in the matrix $(\mathbf{a}, \mathbf{b}, \mathbf{c} \dots)$ are all vectors whose values can be computed by substituting in six points to be interpolated for given values of μ and ν . Just as we did for the spline curves, we need to specify the values of μ and ν where the knots are located. For example, one possibility is shown on the right. These values will create a surface that is similar to the one in Figure 1.

	μ	ν
\mathbf{P}_0	0	0
\mathbf{P}_1	0	1
\mathbf{P}_2	1	0
\mathbf{P}_3	1	1
\mathbf{P}_4	1/2	0
\mathbf{P}_5	1/2	1

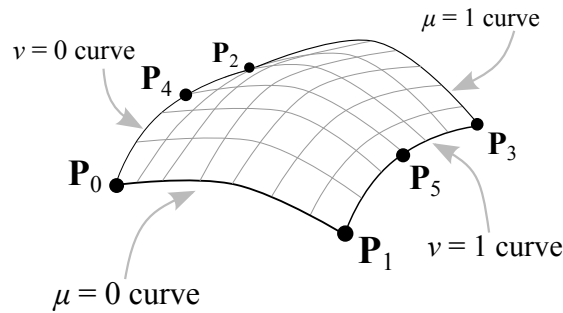


Figure 1: A quadratic surface determined by choosing 6 points

Substituting the various choices of μ, ν for these points into equation 1 we obtain a set of linear equations:

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{f} \\ \mathbf{P}_1 &= \mathbf{d} + 2\mathbf{e} + \mathbf{f} \\ \mathbf{P}_2 &= \mathbf{a} + 2\mathbf{c} + \mathbf{f} \\ \mathbf{P}_3 &= \mathbf{a} + 2\mathbf{b} + 2\mathbf{c} + \mathbf{d} + 2\mathbf{e} + \mathbf{f} \\ \mathbf{P}_4 &= \mathbf{a}/4 + \mathbf{c} + \mathbf{f} \\ \mathbf{P}_5 &= \mathbf{a}/4 + \mathbf{b} + \mathbf{c} + \mathbf{d} + 2\mathbf{e} + \mathbf{f} \end{aligned}$$

We can solve these equations to find the values of the constants $\{\mathbf{a}, \mathbf{b}, \dots, \mathbf{f}\}$ that define the patch. This is more flexible than the non-parametric formulation, but it still does not provide us with a very useful spline since there are no intuitive ways of using it to create a particular shape.

Higher orders can be designed by including μ^2 and ν^2 in the formulation. However little is gained by doing this. Like the equations we developed for the curves, this simple parametric form is only really applicable for solving specific interpolation problems. It is not suitable for use as a general method of surface construction.

Bi-cubic surface patches

As with curves, a good general method is to model surfaces with patches. But it is more complex for surfaces than for spline curves. It is quite common to adopt a formulation where the points are distributed on a regular grid, one example is the case of a *terrain map*. The data in this case can be used in a non-parametric formulation in which a function defines the height at each grid point, i.e. $y = f(x, z)$. A typical patch is shown in Figure 2.

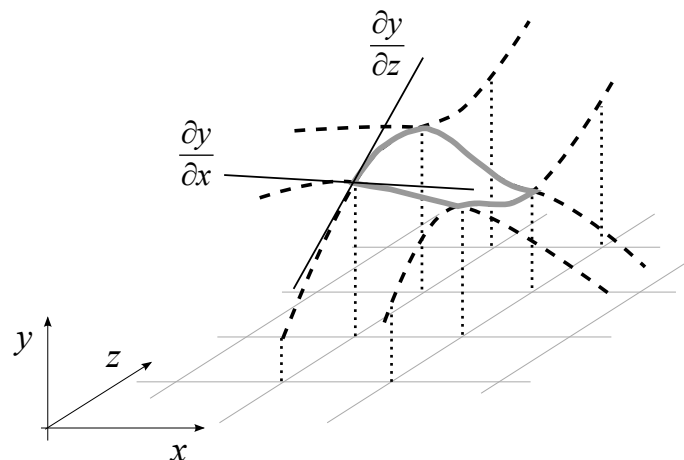


Figure 2: A "Terrain" Surface Patch

At every grid point there are two gradients which we can write as

$$\frac{\partial y}{\partial x} \quad \text{and} \quad \frac{\partial y}{\partial z}$$

Thus, if we are going to fit a patch to match the height and gradients at each of the four corner points, we need twelve (4×3) parameters in the equation of the patch. Furthermore, we need to ensure that the shared boundaries of adjacent patches match each other.

Rather than trying to develop a single equation to do all of this, we normally use bi-cubic interpolation. We can see from Figure 2 that we can easily design spline curves for the four edges of the patch. These curves will be continuous with the neighbouring patches. Thus, in the next section, we adopt a solution where we begin with these curves at the edges of the patch and we interpolate them in the middle of the patch.

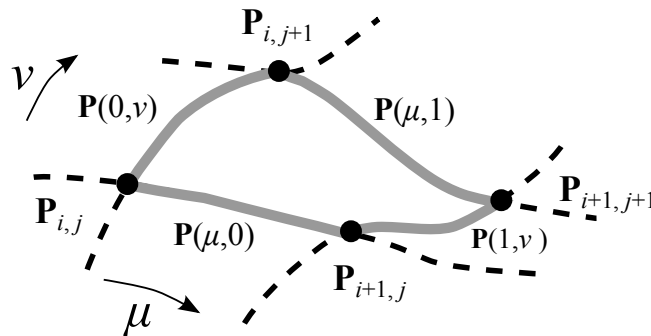


Figure 3: A Parametric Surface Patch

Parametric Surface patches - the Coons Patch

Following the methods we applied to the spline curves we adopt a parametric formulation for the surface patches which gives more flexibility in design. The parametric formulation can be used for simple terrain maps described above where the Cartesian and parametric axes are the same. In general though, the points to be interpolated need not be on a regular grid in the xz plane though we still need them to form a rectangular array.

A typical patch in parametric space is shown in Figure 3. We can specify the four curves that bound a patch, using the method of cubic patches described in the lecture on spline curves. In particular, we can derive the gradients from the central differences of the adjoining points such that the surface will fit a rectangular grid of points smoothly.

Notice that the contours are orthogonal in the parameter space, and we have two parameters. That is to say, contours joining $P_{i,j}$ to $P_{i,j+1}$ and $P_{i+1,j}$ to $P_{i+1,j+1}$, are both functions of the ν parameter as it varies from 0 to 1 while μ is held constant.

Similarly the contours joining $P_{i,j}$ to $P_{i+1,j}$ and $P_{i,j+1}$ to $P_{i+1,j+1}$ are both functions of μ in the range 0 to 1 with constant ν . We can describe the edge contours joining the knots simply by treating one of the parameters as fixed at 1 or 0. Thus we denote the four contours that bound the patch as follows:

Edge curve	Points joined	
$P(0, \nu)$	$P_{i,j}$	$P_{i,j+1}$
$P(1, \nu)$	$P_{i+1,j}$	$P_{i+1,j+1}$
$P(\mu, 0)$	$P_{i,j}$	$P_{i+1,j}$
$P(\mu, 1)$	$P_{i,j+1}$	$P_{i+1,j+1}$

We need now to define $P(\mu, \nu)$ within the patch, in such a way that at the edges it follows the contours, and

in the middle it is a reasonable blend of them. This is done by linear interpolation with the equation:

$$\begin{aligned} \mathbf{P}(\mu, \nu) = & \mathbf{P}(\mu, 0)(1 - \nu) + \mathbf{P}(\mu, 1)\nu + \\ & \mathbf{P}(0, \nu)(1 - \mu) + \mathbf{P}(1, \nu)\mu - \\ & \mathbf{P}(0, 1)(1 - \mu)\nu - \mathbf{P}(1, 0)\mu(1 - \nu) - \\ & \mathbf{P}(0, 0)(1 - \mu)(1 - \nu) - \mathbf{P}(1, 1)\mu\nu \end{aligned} \quad (2)$$

This is a tricky (and long) equation, but you can verify that you get the four edge contours by substituting $\mu = 0$, $\mu = 1$, $\nu = 0$ and $\nu = 1$. The first four terms are simply a linear interpolation of both the bounding curves. However it is clear that we cannot just add them together, as this would no longer go through the four points that define the patch. The last four negative terms correct the curve at the corner points without introducing any discontinuity. This formulation is called the *Coons Patch* after its inventor, the graphics pioneer Steven A. Coons, and is probably the easiest to use surface construction method.

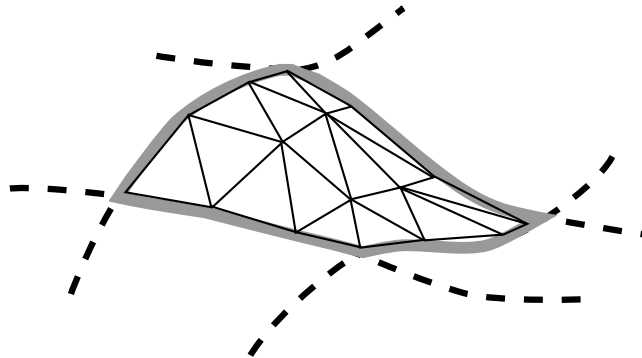


Figure 4: Triangulating a Surface Patch

Rendering Surface Patches

A simple way to draw a patch of this kind is by using *polygonisation*. The method is illustrated in Figure 4 where the polygons used are triangles (i.e. the patch is *triangulated*). Using Equation 2, we can calculate a value of $\mathbf{P}(\mu, \nu)$ for any values μ and ν . If the values are in the range $[0, 1]$ then $\mathbf{P}(\mu, \nu)$ is a point on the patch. Thus we can calculate a regular(ish) grid of points over the patch and join them into triangles. These are then treated as polygons and fed to a polygon renderer. Providing we make the grid fine enough we can get an exact representation down to pixel resolution. For faster results we can use coarser polygons, and apply Gouraud or Phong shading to smooth out the discontinuities.

It is also possible to render the patch using the ray tracing method but this is made difficult by the high order of the patch equation. As we will see in the lecture on ray tracing we need to solve for the intersection of a ray with the patch. The ray in can be written as a straight line

$$\mathbf{P}(\gamma) = \mathbf{S} + \gamma\mathbf{d}$$

We can then equate $\mathbf{P}(\gamma)$ with $\mathbf{P}(\mu, \nu)$ and attempt to solve for the three parameters, γ , μ and ν . However, the equation of the patch is fourth order (it contains terms like $\mu^3\nu$ and $\mu\nu^3$) and so only an iterative solution is possible. Also, there may be several valid intersections between the ray and the patch, and it is necessary to find the nearest. This can be done by numerical methods, but is computationally very expensive.

If the surface is smooth, and relatively well behaved, then a practical solution is to do a coarse polygonisation, using triangles, and intersect the ray with each triangle to find the closest intersection. This area of the patch can then be polygonised further, and the process repeated until sufficient accuracy is reached. Thus the calculation is simply to find the intersection of the ray and the triangle.

An older way of drawing surfaces is to represent them by contours. This is called *lofting* (a term originating from the aircraft industry). To draw a contour we need to fix one of the parameters and draw the curve as the other ranges over the interval 0 to 1. To produce any meaningful representation of a complex surface however,

it is necessary to prevent lines that should be hidden lines from being drawn. This is done by a method called the *floating horizon*. Basically we sort the contours into the order of the distance from us, which in the normal configuration is in order of z . Then, we set up a record, for each pixel in the x direction of the largest y (height) found so far: ie the horizon. Before each part of a contour is drawn it is checked against this horizon. If it is above it, it is drawn, and the horizon is updated, if not it is ignored.

Table 1: An example of terrain map values for a Coons Patch. The box indicates the corners of the patch to be constructed. (The dots indicate values that are not needed for the patch being constructed and they have been omitted for clarity.)

		$y, \nu \rightarrow$					
		2	3	4	5	6	7
x, μ ↓	7
	8	.	.	10	9	.	.
	9	.	14	12	11	10	.
	10	.	15	13	14	10	.
	11	.	.	10	11	.	.

Example: Constructing a Coons Patch

We will conclude with an example of the construction of a Coons patch. Part of a terrain map defined (for simplicity) on a regular xy grid is shown in Table 1 where the heights indicated by the values in the table represent z -values. We will construct a parametric Coons spline patch on the four centre points. The dots in the table stand for other values in the terrain map that are not needed for the patch we want to construct and they have been omitted for clarity.

The corners are defined directly in Table 1 so we can write:

$$\begin{aligned} \mathbf{P}(0, 0) &= (9, 4, 12) & \mathbf{P}(1, 0) &= (10, 4, 13) \\ \mathbf{P}(0, 1) &= (9, 5, 11) & \mathbf{P}(1, 1) &= (10, 5, 14) \end{aligned}$$

We need to define the gradients at the corners of the patch. We can use the central difference approximation using the two points adjacent to the corners. For gradients in the μ (or x) direction:

$$\begin{aligned} \left. \frac{\partial \mathbf{P}}{\partial \mu} \right|_{(0,0)} &= \frac{(10, 4, 13) - (8, 4, 10)}{2} = (1, 0, 1.5) & \left. \frac{\partial \mathbf{P}}{\partial \mu} \right|_{(1,0)} &= \frac{(11, 4, 10) - (9, 4, 12)}{2} = (1, 0, -1) \\ \left. \frac{\partial \mathbf{P}}{\partial \mu} \right|_{(0,1)} &= \frac{(10, 5, 14) - (8, 5, 9)}{2} = (1, 0, 2.5) & \left. \frac{\partial \mathbf{P}}{\partial \mu} \right|_{(1,1)} &= \frac{(11, 5, 11) - (9, 5, 11)}{2} = (1, 0, 0) \end{aligned}$$

and for the gradients in the ν (or y) direction:

$$\begin{aligned} \left. \frac{\partial \mathbf{P}}{\partial \nu} \right|_{(0,0)} &= \frac{(9, 5, 11) - (9, 3, 14)}{2} = (0, 1, -1.5) & \left. \frac{\partial \mathbf{P}}{\partial \nu} \right|_{(1,0)} &= \frac{(10, 5, 14) - (10, 3, 15)}{2} = (0, 1, -0.5) \\ \left. \frac{\partial \mathbf{P}}{\partial \nu} \right|_{(0,1)} &= \frac{(9, 6, 10) - (9, 4, 12)}{2} = (0, 1, -1) & \left. \frac{\partial \mathbf{P}}{\partial \nu} \right|_{(1,1)} &= \frac{(10, 6, 10) - (10, 4, 13)}{2} = (0, 1, -1.5) \end{aligned}$$

To find the bounding contours we use the cubic spline patch equation. Thus:

$$\mathbf{P}(\mu, 0) = \mathbf{a}_3\mu^3 + \mathbf{a}_2\mu^2 + \mathbf{a}_1\mu + \mathbf{a}_0$$

We need to solve for the four constant vectors \mathbf{a}_3 , \mathbf{a}_2 , \mathbf{a}_1 and \mathbf{a}_0 , and we do this using the matrix formulation introduced in the lecture on spline curves:

$$\begin{pmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}'_0 \\ \mathbf{P}_1 \\ \mathbf{P}'_1 \end{pmatrix}$$

For the boundary curve $\mathbf{P}(\mu, 0)$, we make the substitutions $\mathbf{P}_0 = \mathbf{P}(0, 0)$, $\mathbf{P}'_0 = \frac{\partial \mathbf{P}}{\partial \mu} \Big|_{(0,0)}$, $\mathbf{P}_1 = \mathbf{P}(1, 0)$ and $\mathbf{P}'_1 = \frac{\partial \mathbf{P}}{\partial \mu} \Big|_{(1,0)}$ to get

$$\begin{pmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 9 & 4 & 12 \\ 1 & 0 & 1.5 \\ 10 & 4 & 13 \\ 1 & 0 & -1 \end{pmatrix}$$

Now we can calculate the required parameter vectors for $\mathbf{P}(\mu, 0)$

$$\mathbf{a}_0 = \mathbf{P}_0 = (9, 4, 12)$$

$$\mathbf{a}_1 = \mathbf{P}'_0 = (1, 0, 1.5)$$

$$\begin{aligned} \mathbf{a}_2 &= -3\mathbf{P}_0 - 2\mathbf{P}'_0 - 3\mathbf{P}_1 - \mathbf{P}'_1 \\ &= -3 \times (9, 4, 12) - 2 \times (1, 0, 1.5) + 3 \times (10, 4, 13) - (1, 0, 1) = (0, 0, 1) \end{aligned}$$

$$\begin{aligned} \mathbf{a}_3 &= 2\mathbf{P}_0 + \mathbf{P}'_0 - 2\mathbf{P}_1 + \mathbf{P}'_1 \\ &= 2 \times (9, 4, 12) + (1, 0, 1.5) - 2 \times (10, 4, 13) + (1, 0, 1) = (0, 0, 0.5) \end{aligned}$$

Using the same method, we can find the parameter vectors for the other three bounding curves: $\mathbf{P}(\mu, 1)$, $\mathbf{P}(0, \nu)$ and $\mathbf{P}(1, \nu)$.

We now have equations and values for all the individual terms in the Coons patch Equation:

- $\mathbf{P}(\mu, 0)$: a cubic polynomial in μ
- $\mathbf{P}(\mu, 1)$: a cubic polynomial in μ
- $\mathbf{P}(0, \nu)$: a cubic polynomial in ν
- $\mathbf{P}(1, \nu)$: a cubic polynomial in ν
- $\mathbf{P}(0, 0)$, $\mathbf{P}(0, 1)$, $\mathbf{P}(1, 0)$ and $\mathbf{P}(1, 1)$: the corner points,

So, if we are given values of μ and ν we can evaluate each of these eight terms and so find the coordinate on the Coons patch using equation 2.