Lecture 9: Introduction to Surface Construction

Non-ParametricSurfaces

We now turn to the question of how to represent surfaces, and how to draw them. 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:

$$[x\ y\ z\ 1]\begin{bmatrix} a & b & c & d \\ b & e & f & g \\ c & f & h & i \\ d & g & i & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0$$

which multiplies out to:

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

and the nine scalar unknowns $(a, b, ..i)$ can be found by specifying nine points $\mathbf{P_i} = [x_i, y_i, z_i]$ through which the surface must pass. This creates a system of nine linear equations to solve. Notice that because of the inherent symmetry of the formulation 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 a simple generalisation to go 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 $\mathbf{P}(\mu, \nu)$. We could define a parametric surface using a matrix formulation:

$$\mathbf{P}(\mu,\nu) = [\mu, \nu, 1]\begin{bmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \\ \mathbf{b} & \mathbf{d} & \mathbf{e} \\ \mathbf{c} & \mathbf{e} & \mathbf{f} \end{bmatrix}\begin{bmatrix} \mu \\ \nu \\ 1 \end{bmatrix} = 0$$

Multiplying out we get:

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

The values of the constant vectors (**a**,**b**, . . **f**) determine the shape of the surface. The surface has edges given by the four curves for which one of the parameters is either 0 or 1. These are the quadratics:

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

The unknown values in the matrix (**a**, **b**, **c** etc) 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:

	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

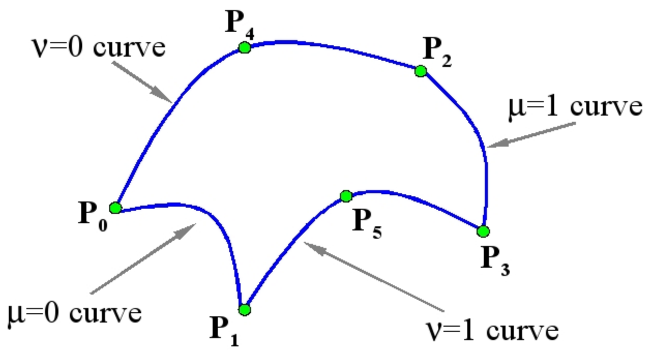


Figure 1: A simple Quadratic Surface

These values will create a surface that is similar to figure 1. Substituting them into equation 1 we obtain a set of linear equations to find the values of the constants (**a,b,c,d,e,f**) that define the patch.

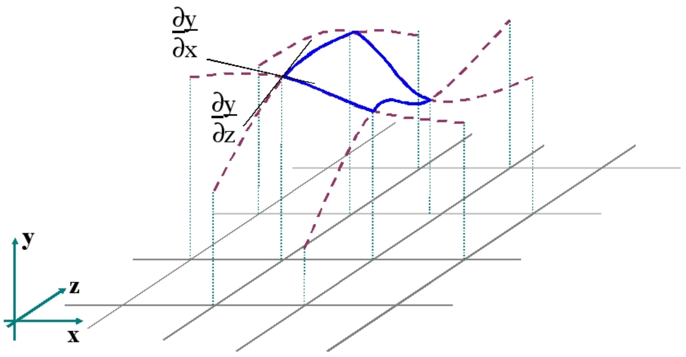
P0 = **f**
P1 = **d** + 2**e** + **f**
P2 = **a** + 2**c** + **f**
P3 = **a** + 2**b** + 2**c** + **d** + 2**e** + **f**
P4 = **a**/4 + **c** + **f**
P5 = **a**/4 + **b** + **c** + **d** + 2**e** + **f**

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

The patch method is available, and represents a good general method, but it is more complex 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 instance can be writen in a non-parametric formulation in which a function defines the height at each grid point:

$$y = f(x, z)$$



$\frac{\partial y}{\partial x}$
$\frac{\partial y}{\partial z}$
y
z
x

Figure 2: A “Terrain” Surface Patch

A typical patch is shown in figure 2. At every grid point there are two gradients which we can write as $\frac{\partial y}{\partial x}$ and $\frac{\partial y}{\partial z}$. Thus, if we are going to fit a patch to the four points, we need twelve parameters in the equation of the patch, so that we can match the positions and both gradients at the four corners. Furthermore, we need to ensure continuity along the boundaries of the patches. Rather than try to develop one equation which will do this for us, we normally use bi-cubic interpolation. It will be seen from figure 2 that we can easily design spline curves for the four edges. These curves will be continuous with the neighbouring patches. Thus we adopt a solution where we use these curves at the edges of the patch, and we interpolate them in the middle of the patch.

Parametric Surface patches - the Coon’s 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 gerneral though, the points which are to be interpolated need not be on a regular grid in the $z - x$ plane though we still need them to form a rectangular array. A typical patch in parametric space is shown in figure 3. We

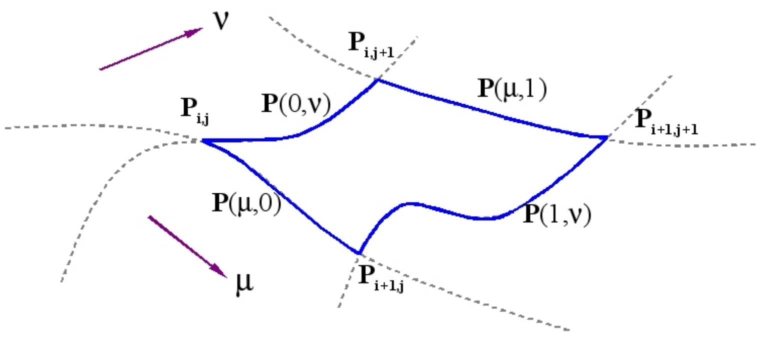


Figure 3: A Parametric Surface Patch

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 $\mathbf{P}_{i,j}$ to $\mathbf{P}_{i,j+1}$ and $\mathbf{P}_{i+1,j}$ to $\mathbf{P}_{i+1,j+1}$, are both functions of the ν parameter as it varies from

0 to 1, with constant μ. Similarly the contours joining $\mathbf{P}_{i,j}$ to $\mathbf{P}_{i+1,j}$ and $\mathbf{P}_{i,j+1}$ to $\mathbf{P}_{i+1,j+1}$ 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:

$\mathbf{P}(0,\nu)$ joining $\mathbf{P}_{i,j}$ to $\mathbf{P}_{i,j+1}$
$\mathbf{P}(1,\nu)$ joining $\mathbf{P}_{i+1,j}$ to $\mathbf{P}_{i+1,j+1}$
$\mathbf{P}(\mu,0)$ joining $\mathbf{P}_{i,j}$ to $\mathbf{P}_{i+1,j}$
$\mathbf{P}(\mu,1)$ joining $\mathbf{P}_{i,j+1}$ to $\mathbf{P}_{i+1,j+1}$

We need now to define the locus of $\mathbf{P}(\mu,\nu)$ 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, the equation being:

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

This is a tricky equation to understand, but you can verify that you get the four edge contours by substituting μ=0, μ=1, ν=0 and ν=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 Coon's Patch, and is probably the easiest to use surface construction method.

Rendering Surface Patches

A simple way to draw a patch of this kind is called polygonisation. The method is illustrated in figure 4. Using equation 2 we can calculate a value of $\mathbf{P}(\mu,\nu)$ for any given ν and ν. If they are in in the range $[0..1]$ the value is a point on the patch. Thus we can calculate a regular grid of points over the patch and join them into triangles. These are then treated as polygons and fed to a polygon rendrer. 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.

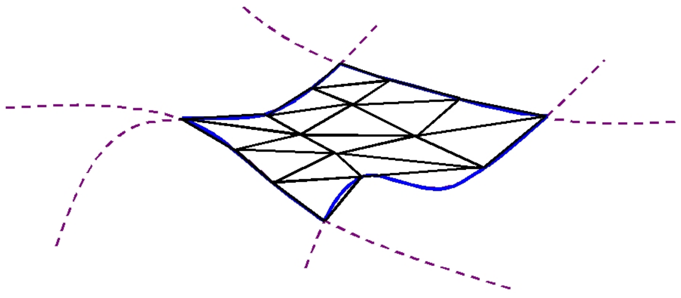


Figure 4: Triangulating a Surface Patch

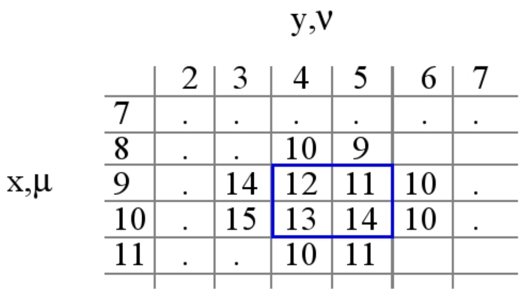
Rendering the patch using the method of ray tracing is also possible, but it is made difficult by the high order of the patch equation. As we will see in the lecture on ray tracing we need to solve for the intersection of a ray with the patch. The ray in can be written as a stright line $\mathbf{P}(\gamma) = \mathbf{S} + \gamma\mathbf{d}$ then we can equate $\mathbf{P}(\gamma)$ with $\mathbf{P}(\mu,\nu)$) and then attempt to solve for the three parameters, γ, μ and ν. However, the equation of the patch is fourth order (it has terms of $\mu^3\nu$ and $\mu\nu^3$) and so only an iterative solution is possible. There will also 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 polygonistion, 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 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 only 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 eliminate the hidden lines, and 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.

Example of Using a Coon's 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 x,y grid is shown in figure 5. We will construct a parametric Coon's spline patch on the four centre points. The corners are defined directly in figure 5 so we can write:

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



y,ν

x,μ

	2	3	4	5	6	7
7	.	.	.	.	.	.
8	.	.	10	9		
9	.	14	12	11	10	.
10	.	15	13	14	10	.
11	.	.	10	11		

Figure 5: An example of the Coons Patch

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:

$\partial\mathbf{P}(0,0)/\partial\mu = ((10,4,13) - (8,4,10))/2 = (1,0,1.5)$
$\partial\mathbf{P}(1,0)/\partial\mu = ((11,4,10) - (9,4,12))/2 = (1,0,-1)$
$\partial\mathbf{P}(0,1)/\partial\mu = ((10,5,14) - (8,5,9))/2 = (1,0,2.5)$
$\partial\mathbf{P}(1,1)/\partial\mu = ((11,5,11) - (9,5,11))/2 = (1,0,0)$

and for the gradients in the ν or y direction:

$\partial\mathbf{P}(0,0)/\partial\nu = ((9,5,11) - (9,3,14))/2 = (0,1,-1.5)$
$\partial\mathbf{P}(1,0)/\partial\nu = ((10,5,14) - (10,3,15))/2 = (0,1,-0.5)$
$\partial\mathbf{P}(0,1)/\partial\nu = ((9,6,10) - (9,4,12))/2 = (0,1,-1)$
$\partial\mathbf{P}(1,1)/\partial\nu = ((10,6,10) - (10,4,13))/2 = (0,1,-1.5)$

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 which yields:

$\mathbf{a}_0 = \mathbf{P}_0 = (9,4,12)$
$\mathbf{a}_1 = \mathbf{P}'_0 = (1,0,1.5)$
$\mathbf{a}_2 = -3\mathbf{P}_0 - 2\mathbf{P}'_0 - 3\mathbf{P}_1 - \mathbf{P}'_1 = -3*(9,4,12) - 2*(1,0,1.5) + 3*(10,4,13) - (1,0,1) = (0,0,1)$
$\mathbf{a}_3 = 2\mathbf{P}_0 + \mathbf{P}'_0 - 2\mathbf{P}_1 + \mathbf{P}'_1 = 2*(9,4,12) + (1,0,1.5) - 2*(10,4,13) + (1,0,1) = (0,0,0.5)$

Similarly we can solve for the other bounding curves $\mathbf{P}(\mu,1)$, $\mathbf{P}(0,\nu)$ and $\mathbf{P}(1,\nu)$.

We now have equations for all the individual terms in the Coon's patch:

$\mathbf{P}(\mu,0)$: 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 given μ and ν we can evaluate each of these eight terms and so find the coordinate on the Coon's patch using equation 2.