Lecture 14 Spline Surface Construction

Non-ParametricSurfaces

We now turn to the question of how to represent surfaces, and how to draw them. As before, one possibility is to adopt the simple solution of non parametric Cartesian equations. A quadratic surface would have an equation of the form:

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

which multiplies out to:

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

and the nine unknowns could be found by specifying nine points **Pi** = [xi,yi,zi], which yields a system of nine linear equations to solve. Notice that because of the inherent symmetry of the formulation there are only nine unknowns, not 16. The constant term can always be taken as 1. This method however suffers the same limitations as the analogous method for curves. It is difficult to control the 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, but we note that we now need two parameters to define the surface. We will call these μ and ν. We could define a parametric surface using a matrix formulation:

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

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

The surface has edges given by the following four curves, which are quadratics

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

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

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

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

The 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. 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

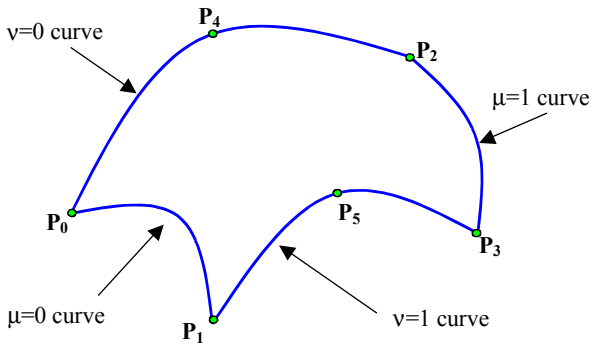

Diagram 1: A Parametric Spline Patch

Finally we solve a set of linear equations to find the actual patch, which will be like that shown in diagram 1.

$\mathbf{P}_0 = \mathbf{f}$

$\mathbf{P}_1 = \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

$\mathbf{P}_2 = \mathbf{a} + 2\mathbf{c} + \mathbf{f}$

$\mathbf{P}_3 = \mathbf{a} + 2\mathbf{b} + 2\mathbf{c} + \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

$\mathbf{P}_4 = \mathbf{a}/4 + \mathbf{c} + \mathbf{f}$

$\mathbf{P}_5 = \mathbf{a}/4 + \mathbf{b} + \mathbf{c} + \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

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 vectors. 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.

Bi-cubic surface patches

The patch method is available, and represents a good general method, but it is more complex. Consider four points belonging to a surface, $\mathbf{P}_0$,$\mathbf{P}_1$,$\mathbf{P}_2$, and $\mathbf{P}_3$. We will consider first the case where the points are part of a regular grid. This gives us a non-parametric formulation in which we can write, for each grid points:

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

A typical patch is shown in diagram 2. We can see that, at every grid point there are two gradients:

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

Thus, if we are going to fit a patch to the four points, we need twelve parameters in our equation, 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 diagram 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.

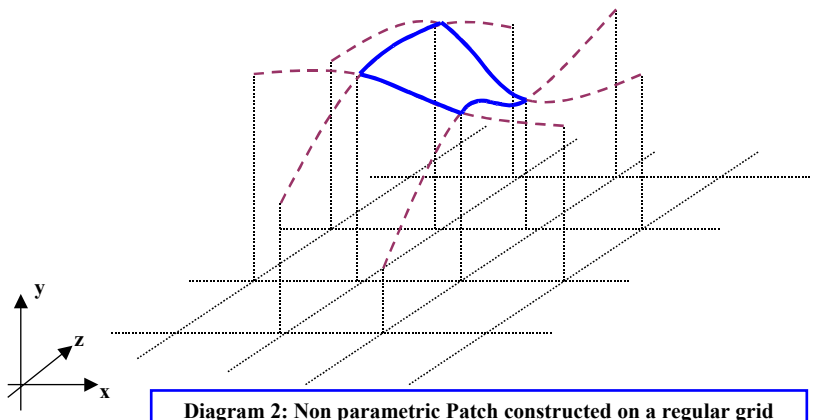

Diagram 2: Non parametric Patch constructed on a regular grid

As a refinement of this process, it is convenient to use a parametric formulation. This means that 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 diagram 3.

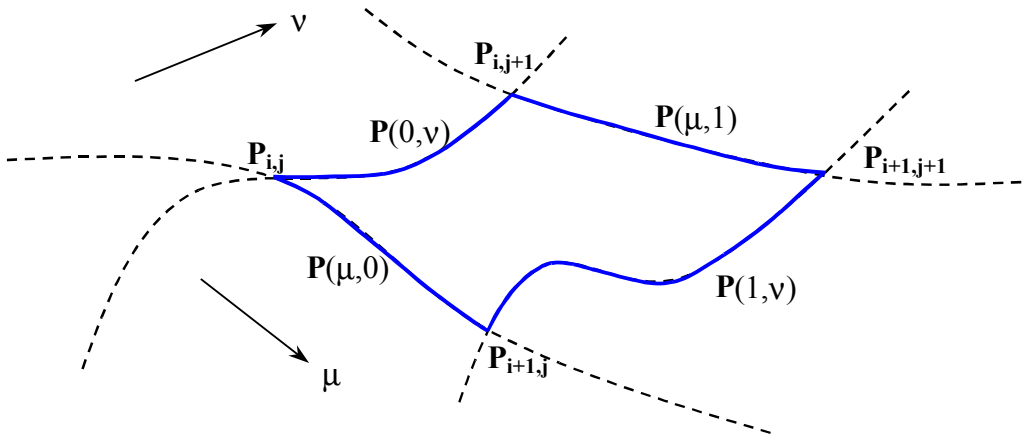

Diagram 3: A parametric spline patch

Referring to diagram 3, it is clear that we can specify four curves that bound each patch, using the method of cubic spline curve patches described previously. 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. It is convenient to introduce a new notation for these curves. In the previous

case we considered the locus of a point as a function of one parameter, we now wish to consider a point inside the patch as a function of two parameters, so we write this as $\mathbf{P}(\mu,\nu)$. We can describe the contours joining the knots simply by treating one of the parameters as fixed. Thus for the four contours we write:

$\mathbf{P}(0,\nu)$ 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 most simply done by linear interpolation, the equation being:

$$\mathbf{P}(\mu,\nu) = \mathbf{P}(\mu,0)(1-\nu) + \mathbf{P}(\mu,1)\nu + \mathbf{P}(0,\nu)(1-\mu) + \mathbf{P}(1,\nu)\mu$$
$$- \mathbf{P}(0,0)(1-\nu)(1-\mu) - \mathbf{P}(0,1)\nu(1-\mu) - \mathbf{P}(1,0)(1-\nu)\mu - \mathbf{P}(1,1)\nu\mu$$

This is a tricky equation to understand, but you can verify it for 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. Consequently we subtract the last four negative terms which correct the curve at the defining points without introducing any discontinuity. This formulation is called the Coon's Patch, and is probably the easiest to use surface construction method.

A simple way to draw a patch of this kind is called polygonisation. The method is illustrated in diagram 4.

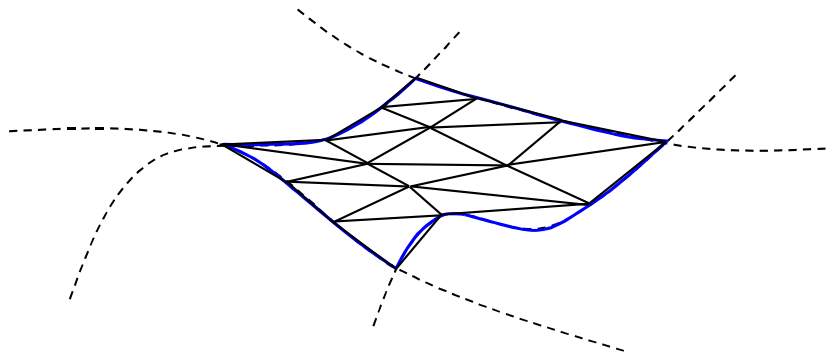

Diagram 4: Polygonisation of a spline patch

We can easily calculate a value of P given μ and ν, and so we can calculate a grid of points over the patch. 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.

Ray tracing is also possible, but it is made difficult by the high order of the patch equation. If we write the ray in the form $\mathbf{P} = \mathbf{S} + \gamma\mathbf{d}$ then we can equate $\mathbf{P}$ with $\mathbf{P}(\mu,\nu)$ and then attempt to solve for the three parameters, γ,μ,ν. However, the equation of the patch is fourth order (it has terms of $\nu\mu^3$ and $\mu\nu^3$) and so only numerical solution is possible. Unfortunately, there may be several valid intersections between the ray and the patch, and the problem is 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 Coon's patch. Part of a terrain map defined on a regular x,y grid is shown in diagram 5. We will construct a spline patch on the four centre points.

The corners are defined directly in the figure so we can write:
P(0,0) = (9,4,12)
P(0,1) = (9,5,11)
P(1,0) = (10,4,13)
P(1,1) = (10,5,14)

x,μ \ y,ν	3	4	5	6
8		10	9	
9	14	12	11	10
10	15	13	14	10
11		10	11	

Diagram 5

The next task is to find the gradients at the corners. For gradients in the x(μ) direction we can take the difference of the two adjacent points which gives a central difference approximation to the gradient.
∂P/dμ (at P(0,0)) = ((10,4,13) - (8,4,10))/2 = (1,0,1.5)
∂P/dμ (at P(1,0)) = ((11,4,10) - (9,4,12))/2 = (1,0,-1)
∂P/dμ (at P(0,1)) = ((10,5,14) - (8,5,9))/2 = (1,0,2.5)
∂P/dμ (at P(1,1)) = ((11,5,11) - (9,5,11))/2 = (1,0,0)

Similarly we can find the gradients in the y(ν) direction
∂P/dν (at P(0,0)) = ((9,5,11) - (9,3,14))/2 = (0,1,-1.5)
∂P/dν (at P(1,0)) = ((10,5,14) - (10,3,15))/2 = (0,1,-0.5)
∂P/dν (at P(0,1)) = ((9,6,10) - (9,4,12))/2 = (0,1,-1)
∂P/dν (at P(1,1)) = ((10,6,10) - (10,4,13))/2 = (0,1,-1.5)

Finding the boundary curves we use the equation for a simple cubic patch. For example:
$P(\mu,0) = \mathbf{a}_3\mu^3 + \mathbf{a}_2\mu^2 + \mathbf{a}_1\mu + \mathbf{a}_0$
Solving for the constants $\mathbf{a}_0$ - $\mathbf{a}_3$
$\mathbf{a}_0$ = (9,4,12)
$\mathbf{a}_1$ = (1,0,1.5)
$\mathbf{a}_2$ = -3*(9,4,12) - 2*(1,0,1.5) + 3*(10,4,13) - (1,0,1) = (0,0,1)
$\mathbf{a}_3$ = 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 curves P(μ,1), P(0, ν) and P(1, ν).

We now have all the individual bits:
P(μ,0) cubic polynomial in μ
P(μ,1) cubic polynomial in μ
P(0,ν) cubic polynomial in ν
P(1,ν) cubic polynomial in ν
P(0,0), P(0,1), P(1,0), P(1,1)
Given μ and ν we can evaluate each of these eight points
So, for any given value for μ and ν we can evaluate the coordinate on the Coon's patch using the formula:

$$\mathbf{P}(\mu,\nu) = \mathbf{P}(\mu,0)(1-\nu) + \mathbf{P}(\mu,1)\,\nu + \mathbf{P}(0,\nu)(1-\mu) + \mathbf{P}(1,\nu)\,\mu - \mathbf{P}(0,0)(1-\mu)(1-\nu) - \mathbf{P}(1,0)\,\mu\,(1-\nu) - \mathbf{P}(0,1)(1-\mu)\nu - \mathbf{P}(1,1)\,\mu\,\nu$$

$$\mathbf{P}(\mu,\nu) = \mathbf{P}(\mu,0)\,(1-\nu) + \mathbf{P}(\mu,1)\,\nu + \mathbf{P}(0,\nu)\,(1-\mu) + \mathbf{P}(1,\nu)\,\mu - \mathbf{P}(0,0)(1-\nu)(1-\mu) - \mathbf{P}(0,1)\nu(1-\mu) - \mathbf{P}(1,0)(1-\nu)\mu - \mathbf{P}(1,1)\,\nu\mu$$