Chapter 13

Curves and Surfaces

There are two fundamental problems with surfaces in machine vision: reconstruction and segmentation. Surfaces must be reconstructed from sparse depth measurements that may contain outliers. Once the surfaces are reconstructed onto a uniform grid, the surfaces must be segmented into different surface types for object recognition and refinement of the surface estimates.

This chapter begins with a discussion of the geometry of surfaces and includes sections on surface reconstruction and segmentation. The chapter will cover the following topics on surfaces:

Representations for surfaces such as polynomial surface patches and tensor-product cubic splines

Interpolation methods such as bilinear interpolation

Approximation of surfaces using variational methods and regression splines

Segmentation of point measurements into surface patches

Registration of surfaces with point measurements

Surface approximation is also called surface fitting, since it is like a regression problem where the model is the surface representation and the data are points sampled on the surface. The term *surface reconstruction* means estimating the continuous function for the surface from point samples, which can be implemented by interpolation or approximation.

There are many machine vision algorithms for working with curves and surfaces. This is a large area and cannot be covered completely in an introductory text. This chapter will cover the basic methods for converting point measurements from binocular stereo, active triangulation, and range cameras into simple surface representations. The basic methods include converting point measurements into a mesh of triangular facets, segmenting range measurements into simple surface patches, fitting a smooth surface to the point measurements, and matching a surface model to the point measurements. After studying the material in this chapter, the reader should have a good introduction to the terminology and notation of surface modeling and be prepared to continue the topic in other sources.

13.1 Fields

This chapter covers the problems of reconstructing surfaces from point samples and matching surface models to point measurements. Before a discussion of curves and surfaces, the terminology of fields of coordinates and measurements must be presented.

Measurements are a mapping from the coordinate space to the data space. The coordinate space specifies the locations at which measurements were made, and the data space specifies the measurement values. If the data space has only one dimension, then the data values are scalar measurements. If the data space has more than one dimension, then the data values are vector measurements. For example, weather data may include temperature and pressure (two-dimensional vector measurements) in the three-dimensional coordinate space of longitude, latitude, and elevation. An image is scalar measurements (image intensity) located on a two-dimensional grid of image plane positions. In Chapter 14 on motion, we will discuss the image flow velocity field, which is a two-dimensional space of measurements (velocity vectors) defined in the two-dimensonal space of image plane coordinates.

There are three types of fields: uniform, rectilinear, and irregular (scattered). In uniform fields, measurements are located on a rectangular grid with equal spacing between the rows and columns. Images are examples of uniform fields. As explained in Chapter 12 on calibration, the location of any grid point is determined by the position of the grid origin, the orientation of the grid, and the spacing between the rows and columns.

Rectilinear fields have orthogonal coordinate axes, like uniform fields, but the data samples are not equally spaced along the coordinate axes. The data samples are organized on a rectangular grid with various distances between the rows and columns. For example, in two dimensions a rectilinear field partitions a rectangular region of the plane into a set of rectangles of various sizes, but rectangles in the same row have the same height, and rectangles in the same column have the same width. Lists of coordinates, one list for each dimension, are needed to determine the position of the data samples in the coordinate space. For example, a two-dimensional rectilinear grid with coordinate axes labeled x and y will have a list of x coordinates $\{x_j\}, j = 1, 2, \ldots, m$ for the m grid columns and a list of y coordinates $\{y_i\}, i = 1, 2, \ldots, n$ for the n grid rows. The location of grid point $[i, j]$ is (x_j, y_i).

Irregular fields are used for scattered (randomly located) measurements or any pattern of measurements that do not correspond to a rectilinear structure. The coordinates (x_k, y_k) of each measurement must be provided explicitly in a list for $k = 1, \ldots, n$.

These concepts are important for understanding how to represent depth measurements from binocular stereo and active sensing. Depth measurements from binocular stereo can be represented as an irregular, scalar field of depth measurements z_k located at scattered locations (x_k, y_k) in the image plane or as an irregular field of point measurements located at scattered positions (x_k, y_k, z_k) in the coordinate system of the stereo camera with a null data part. Likewise, depth measurements from a range camera can be represented as distance measurements $z_{i,j}$ on a uniform grid of image plane locations (x_j, y_i) or as an irregular field of point measurements with a null data part. In other words, point samples of a graph surface $z = f(x, y)$ can be treated as displacement measurements from positions in the domain or as points in three-dimensional space.

13.2 Geometry of Curves

Before a discussion of surfaces, curves in three dimensions will be covered for two reasons: surfaces are described by using certain special curves, and representations for curves generalize to representations for surfaces. Curves can be represented in three forms: implicit, explicit, and parametric.

The parametric form for curves in space is

$$\mathbf{p} = (x, y, z) = (x(t), y(t), z(t)), \tag{13.1}$$

for $t_0 \leq t \leq t_1$, where a point along the curve is specified by three functions that describe the curve in terms of the parameter t. The curve starts at the point $(x(t_0), y(t_0), z(t_0))$ for the initial parameter value t_0 and ends at $(x(t_1), y(t_1), z(t_1))$ for the final parameter value t_1. The points corresponding to the initial and final parameter values are the start and end points of the curve, respectively. For example, Chapter 12 makes frequent use of the parametric form for a ray in space:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = t \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}, \tag{13.2}$$

for $0 \leq t < \infty$, where the (unit) vector (u_x, u_y, u_z) represents the direction of the ray and (x_0, y_0, z_0) is the starting point of the ray.

The parametric equation for the line segment from point $\mathbf{p}_1 = (x_1, y_1, z_1)$ to point $\mathbf{p}_2 = (x_2, y_2, z_2)$ is

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = t \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}. \tag{13.3}$$

Curves can also be represented implicitly as the set of points (x, y, z) that satisfy some equation

$$f(x, y, z) = 0 \tag{13.4}$$

or set of equations.

13.3 Geometry of Surfaces

Like curves, surfaces can be represented in implicit, explicit, or parametric form. The parametric form for a surface in space is

$$(x, y, z) = (x(u, v), y(u, v), z(u, v)), \tag{13.5}$$

for $u_0 \leq u \leq u_1$ and $v_0 \leq v \leq v_1$. The domain can be defined more generally as $(u, v) \in D$.

The implicit form for a surface is the set of points (x, y, z) that satisfy some equation

$$f(x, y, z) = 0. \tag{13.6}$$

For example, a sphere with radius r centered at (x_0, y_0, z_0) is

$$f(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0. \tag{13.7}$$

The explicit (functional) form

$$z = f(x, y) \tag{13.8}$$

is used in machine vision for range images. It is not as general and widely used as the parametric and implicit forms, because the explicit form is only useful for graph surfaces, which are surfaces that can be represented as displacements from some coordinate plane.

If a surface is a graph surface, then it can be represented as displacements normal to a plane in space. For example, a range image is a rectangular grid of samples of a surface

$$z = f(x, y), \tag{13.9}$$

where (x, y) are image plane coordinates and z is the distance parallel to the z axis in camera coordinates.

13.3.1 Planes

Three points define a plane in space and also define a triangular patch in space with corners corresponding to the points. Let $\mathbf{p}_0$, $\mathbf{p}_1$, and $\mathbf{p}_2$ be three points in space. Define the vectors $\mathbf{e}_1 = \mathbf{p}_1 - \mathbf{p}_0$ and $\mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_0$. The normal to the plane is $\mathbf{n} = \mathbf{e}_1 \times \mathbf{e}_2$. A point $\mathbf{p}$ lies in the plane if

$$(\mathbf{p} - \mathbf{p}_0) \cdot \mathbf{n} = 0. \tag{13.10}$$

Equation 13.10 is one of the implicit forms for a plane and can be written in the form

$$ax + by + cz + d = 0, \tag{13.11}$$

where the coefficients a, b, and c are the elements of the normal to the plane and d is obtained by plugging the coordinates of any point on the plane into Equation 13.11 and solving for d.

The parametric form of a plane is created by mapping (u, v) coordinates into the plane using two vectors in the plane as the basis vectors for the coordinate system in the plane, such as vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ above. Suppose that point $\mathbf{p}_0$ in the plane corresponds to the origin in u and v coordinates. Then the parametric equation for a point in the plane is

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = u\mathbf{e}_1 + v\mathbf{e}_2 + \mathbf{p}_0. \tag{13.12}$$

If the coordinate system in the plane must be orthogonal, then force $\mathbf{e}_1$ and $\mathbf{e}_2$ to be orthogonal by computing $\mathbf{e}_2$ from the cross product between $\mathbf{e}_1$ and the normal vector $\mathbf{n}$,

$$\mathbf{e}_2 = \mathbf{n} \times \mathbf{e}_1. \tag{13.13}$$

The explicit form for the plane is obtained from Equation 13.11 by solving for z. Note that if coefficient c is close to zero, then the plane is close to being parallel to the z axis and the explicit form should not be used.

13.3.2 Differential Geometry

Differential geometry is the local analysis of how small changes in position (u, v) in the domain affect the position on the surface, $\mathbf{p}(u, v)$, the first derivatives, $\mathbf{p}_u(u, v)$ and $\mathbf{p}_v(u, v)$, and the surface normal, $\mathbf{n}(u, v)$. The geometry is illustrated in Figure 13.1.

The parameterization of a surface maps points (u, v) in the domain to points $\mathbf{p}$ in space:

$$\mathbf{p}(u, v) = (x(u, v), y(u, v), z(u, v)). \tag{13.14}$$

The first derivatives, $\mathbf{p}_u(u, v)$ and $\mathbf{p}_v(u, v)$, are vectors that span the tangent plane to the surface at point $(x, y, z) = \mathbf{p}(u, v)$. The surface normal n at point $\mathbf{p}$ is defined as the unit vector normal to the tangent plane at point $\mathbf{p}$ and is computed using the cross product of the partial derivatives of the surface parameterization,

$$\mathbf{n}(\mathbf{p}) = \frac{\mathbf{p}_u \times \mathbf{p}_v}{\|\mathbf{p}_u \times \mathbf{p}_v\|}. \tag{13.15}$$

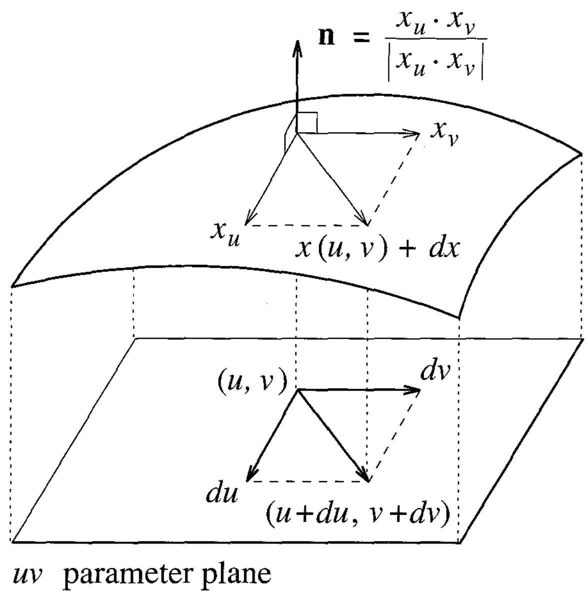

Figure 13.1: A parametrically defined surface showing the tangent plane and surface normal.

The tangent vectors and the surface normal $\mathbf{n}(u, v)$ define an orthogonal coordinate system at point $\mathbf{p}(u, v)$ on the surface, which is the framework for describing the local shape of the surface.

The curvature of a surface is defined using the concept of the curvature of a planar curve. Suppose that we wish to know the curvature of the surface at some point $\mathbf{p}$ on the surface. Consider a plane that slices the surface at point $\mathbf{p}$ and is normal to the surface at point $\mathbf{p}$. In other words, the surface normal at point $\mathbf{p}$ lies in the slicing plane. The intersection of the plane with the surface is a planar curve that includes point $\mathbf{p}$. The normal curvature at point $\mathbf{p}$ is the curvature of this planar curve of intersection. Refer to Figure 13.2. The plane is spanned by the surface normal and a vector in the plane tangent to the surface at $\mathbf{p}$. The normal curvature is not unique, since it depends on the orientation of the plane about the surface normal. The minimum and maximum curvatures, κ_1 and κ_2, are called the

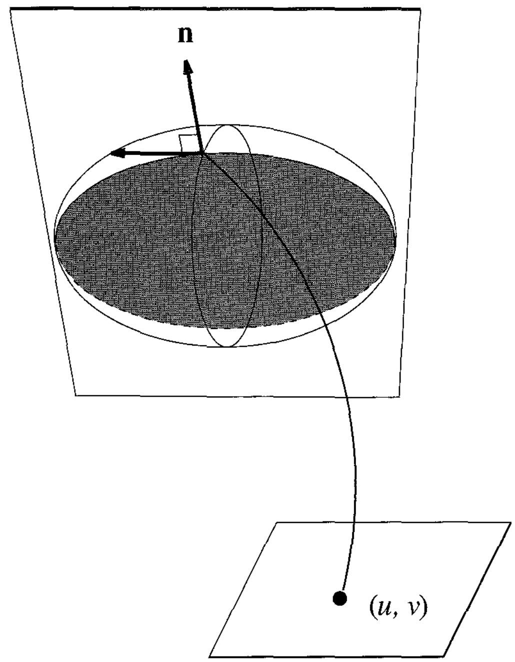

Figure 13.2: Diagram illustrating the geometry of the normal curvature at a point on a surface. The intersecting plane contains the surface normal.

principal curvatures, and the directions in the tangent plane corresponding to the minimum and maximum curvatures are the principal directions.

The Gaussian curvature is the product of the principal curvatures,

$$K = \kappa_1 \kappa_2, \tag{13.16}$$

and the mean curvature is the average of the principal curvatures,

$$H = \frac{\kappa_1 + \kappa_2}{2}. \tag{13.17}$$

The lines of curvature are curves in the surface obtained by following the principal directions. Umbilic points are locations on the surface, such as the end of an egg or any point on a sphere, where all normal curvatures (and hence the two principal curvatures) are equal.

13.4 Curve Representations

Curves can be represented in parametric form $x(t)$, $y(t)$, $z(t)$ with the position of each point on the curve specified by a function of the parameter t. Many functions can be used to specify curves in parametric form. For example, we can specify a line segment starting at (x_1, y_1, z_1) and ending at (x_2, y_2, z_2) as

$$x(t) = tx_2 + (1-t)x_1 \tag{13.18}$$
$$y(t) = ty_2 + (1-t)y_1 \tag{13.19}$$
$$z(t) = tz_2 + (1-t)z_1, \tag{13.20}$$

for $0 \le t \le 1$. Cubic polynomials are used in a general representation of space curves described in the next section.

13.4.1 Cubic Spline Curves

A cubic spline is a sequence of cubic polynomial curves joined end to end to represent a complex curve. Each segment of the cubic spline is a parametric curve:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \tag{13.21}$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \tag{13.22}$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z, \tag{13.23}$$

for $0 \le t \le 1$. Cubic polynomials allow curves to pass through points with a specified tangent and are the lowest-order polynomials that allow nonplanar curves.

These equations can be represented more compactly by organizing the coefficients into vectors. If the coefficient vectors are written as

$$\mathbf{a} = (a_x, a_y, a_z) \tag{13.24}$$
$$\mathbf{b} = (b_x, b_y, b_z) \tag{13.25}$$
$$\mathbf{c} = (c_x, c_y, c_z) \tag{13.26}$$
$$\mathbf{d} = (d_x, d_y, d_z) \tag{13.27}$$

and $\mathbf{p}(t) = (x(t), y(t), z(t))$, then a cubic polynomial curve can be written as

$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}. \tag{13.28}$$

More complex curves are represented as sequences of cubic polynomials joined at their end points:

$$\begin{aligned}
\mathbf{p}_1(t) &= \mathbf{a}_1 t^3 + \mathbf{b}_1 t^2 + \mathbf{c}_1 t + \mathbf{d}_1 \\
\mathbf{p}_2(t) &= \mathbf{a}_2 t^3 + \mathbf{b}_2 t^2 + \mathbf{c}_2 t + \mathbf{d}_2 \\
&\vdots \\
\mathbf{p}_n(t) &= \mathbf{a}_n t^3 + \mathbf{b}_n t^2 + \mathbf{c}_n t + \mathbf{d}_n
\end{aligned} \tag{13.29}$$

for $0 \leq t \leq 1$. Note that $\mathbf{a}_i$, $\mathbf{b}_i$, $\mathbf{c}_i$, and $\mathbf{d}_i$ are the vectors of coefficients for curve segment i. If we define cubic polynomial segment i on the unit interval $i-1 \leq t \leq i$, then the entire sequence is defined on the interval $0 \leq t \leq n$ and the sequence of cubic polynomial segments can be treated as a single parametric curve starting at point $\mathbf{p}_1(0)$ and ending at point $\mathbf{p}_n(n)$. This sequence of cubic polynomials is called a cubic spline and is a common way of representing arbitrary curves in machine vision and computer graphics.

13.5 Surface Representations

This section will cover some simple surface representations that are used in machine vision.

13.5.1 Polygonal Meshes

Planar polygons, also called planar facets or faces, can used to model complex objects. Planes were covered in Section 13.3.1. In this section, we will show how many planar facets can be combined to model the surface of an object using a polygonal mesh.

In Chapter 6, we showed how a polyline can be represented by the list of coordinates for the vertices that connect the line segments. Likewise, a polygonal mesh is represented by the list of vertex coordinates for the vertices that define the planar polygons in the mesh. Since many polygons may share each vertex, we will use an indirect representation that allows each vertex to be listed once. Number the vertices from 1 to n and store the coordinates for each vertex once:

$$
\begin{aligned}
\mathbf{v}_1 &= (x_1, y_1, z_1) \\
\mathbf{v}_2 &= (x_2, y_2, z_2) \\
&\vdots \\
\mathbf{v}_n &= (x_n, y_n, z_n).
\end{aligned}
\tag{13.30}
$$

Represent each face by a list of the vertices in the polygon for the face. To ensure the consistency needed for using the vertex information, follow the convention of listing the vertices in the order in which they are encountered, moving clockwise around the face as seen from the outside of the object or from the same side of the surface (for example, above the surface). For outward-facing normals, this means that the vertex order follows the left-hand rule: the order of the vertex around the polygon is the same as the direction of the fingers of the left hand with the thumb pointing in the direction of the surface normal. This representation makes it very easy to find all of the vertices for a given face, and any change in the coordinates of a vertex automatically (indirectly) changes all faces that use the vertex. The list of vertices representation does not explicitly represent the edges between adjacent faces and does not provide an efficient way to find all faces that include a given vertex. These problems are resolved by using the winged edge data structure.

The winged edge data structure is a network with three types of records: vertices, edges, and faces. The data structure includes pointers that can be followed to find all neighboring elements without searching the entire mesh or storing a list of neighbors in the record for each element. There is one vertex record for every vertex in the polygonal mesh, one edge record for every edge in the polygonal mesh, and one face record for every face in the polygonal mesh. The size of the face, edge, or vertex record does not change with the size of the mesh or the number of neighboring elements. Both vertices at the ends of an edge and both faces on either side of an edge can be found directly; all of the faces (edges) that use a vertex can be found in time proportional to the number of faces (edges) that include the vertex; all of the vertices (edges) around a face can be found in time proportional to the number of vertices (edges) around the face.

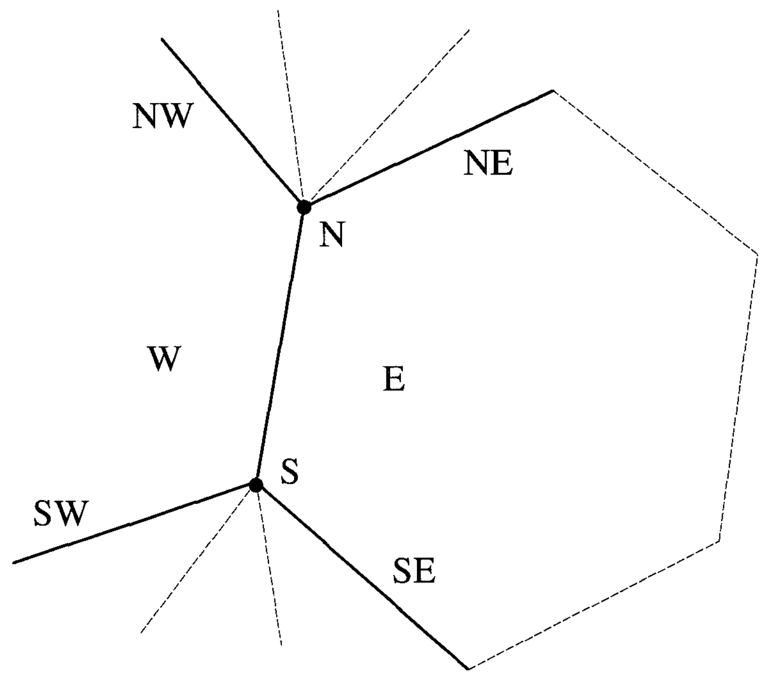

Figure 13.3: Each edge record contains pointers to the two vertices at the ends of the edge, the two faces on either side of the edge, and the four wing edges that allow the polygonal mesh to be traversed.

The winged edge data structure can handle polygons with many sides (the representation is not limited to polygonal meshes with triangular facets), and it is not necessary for all polygons in the mesh to have the same number of sides. The coordinates of vertices are only stored in the vertex records; the position of a face or edge is computed from the vertex coordinates.

Each face record points to the record for one of its edges, and each vertex record points to the record for one of the edges that end at that vertex. The edge records contain the pointers that connect the faces and vertices into a polygonal mesh and allow the polygonal mesh to be traversed efficiently. Each edge record contains a pointer to each of the vertices at its ends, a pointer to each of the faces on either side of the edge, and pointers to the four wing edges that are neighbors in the polygonal mesh. Figure 13.3 illustrates the information contained in an edge record. The faces, vertices, and wing edges are denoted by compass directions. This notation is just a convenience; in a polygonal mesh, there is no global sense of direction. Each wing allows the faces on either side of the edge to be traversed in either direction. For example, follow the northeast wing to continue traversing clockwise around the east face.

Whether a given face is east or west of an edge depends on the order in which faces were entered in the winged edge data structure. As a face is traversed, it is necessary to check if the face is east or west of each edge that is encountered. If the face is east of the edge, follow the northeast wing clockwise around the face or the southeast wing counterclockwise around the face. Otherwise, if the face is west of the edge, follow the southwest wing clockwise around the face or the northwest wing counterclockwise around the face.

Directions like clockwise and counterclockwise are viewer-centered; in other words, they assume that the polygonal mesh is seen from a particular position. When creating and using the polygonal mesh, it is necessary to adhere to some convention that provides an orientation for the viewer-centered directions relative to the polygonal mesh. Directions around the face are given from the side of the face with the surface normal pointing toward the viewer. Clockwise means the direction around the face indicated by how the fingers of the left hand point when the thumb points in the direction of the surface normal; counterclockwise means the direction around the face indicated by how the fingers of the right hand point when the thumb points in the direction of the surface normal. All face normals point to the outside if the polygonal mesh represents the boundary of a volume. If the polygonal mesh represents a surface, then the normals all point to the same side of the surface. If the surface is a graph surface, then the normals point in the direction of positive offsets from the domain. For example, if the polygonal mesh represents the graph surface $z = f(x, y)$, then the projection of the face normals onto the z axis is positive.

The algorithm for adding a face to a polygonal mesh is stated in Algorithm 13.1 and the algorithm for traversing the edges (or vertices) around a face is stated in Algorithm 13.2. Algorithm 13.1 assumes that the vertices are listed in clockwise order around the face, assuming the conventions described above. Algorithm 13.2 can be modified to find all edges (or faces) around a given vertex.

Algorithm 13.1 Adding a Face to a Winged Edge Data Structure
The input is a list of successive vertices for the face, including vertex numbers which are used to uniquely identify each vertex and the vertex coordinates, listed in clockwise order around the face.

1. *For each vertex in the list of vertices, add a record for the vertex to the winged edge data structure if the vertex is not already in the data structure.*

2. *For each pair of successive vertices (including the last and first vertices), add a record for the edge if an edge with those vertices is not currently in the data structure.*

3. *For each of the records for edges around the face, add the wings for clockwise and counterclockwise traversal around the face. The wing fields in each record that are affected depend on whether the new face being inserted in the data structure is the east or west face of each edge record.*

4. *Create a record for the face and add a pointer to one of the edges.*

13.5.2 Surface Patches

Portions of curved graph surfaces can be represented as surface patches modeled using bivariate (two-variable) polynomials. For example, a plane can be represented as

$$z = a_0 + a_1 x + a_2 y, \tag{13.31}$$

and curved surface patches can be modeled using higher-order polynomials.

Bilinear patches, so named because any cross section parallel to a coordinate axis is a line,

$$z = a_0 + a_1 x + a_2 y + a_3 xy, \tag{13.32}$$

biquadratic patches,

$$z = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2, \tag{13.33}$$

bicubic patches,

$$\begin{aligned} z = \; & a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 \\ & + a_6 x^3 + a_7 x^2 y + a_8 x y^2 + a_9 x^3, \end{aligned} \tag{13.34}$$

and biquartic patches,

$$\begin{aligned} z = \; & a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2 \\ & + a_6 x^3 + a_7 x^2 y + a_8 x y^2 + a_9 y^3 \\ & + a_{10} x^4 + a_{11} x^3 y + a_{12} x^2 y^2 + a_{13} x y^3 + a_{14} y^4, \end{aligned} \tag{13.35}$$

are bivariate polynomials that are frequently used in machine vision to represent surface patches.

Polynomial surface patches are good for modeling portions of a surface, such as the neighborhood around a point, but surface patches are not convenient for modeling an entire surface and cannot model surfaces that are not graph surfaces. More complex surfaces can be modeled using cubic spline surfaces, presented in the next section.

Algorithm 13.2 Follow the Edges Clockwise Around a Face
The inputs are a pointer to the record for the face to traverse and a procedure to invoke for each edge that is visited.

1. *Get the first edge from the face record and make it the current edge.*
2. *Process the current edge: perform whatever operations must be done as each edge is visited. For example, to compile a list of vertices clockwise around the face, record the vertex at the end of the edge in the direction of traversal.*
3. *If the west face of the current edge is being circumnavigated, then the next edge is the southwest wing.*
4. *If the east face of the current edge is being circumnavigated, then the next edge is the northeast wing.*
5. *If the current edge is the first edge, then the traversal is finished.*
6. *Otherwise, go to step 2 to process the new edge.*

13.5.3 Tensor-Product Surfaces

In Section 13.4.1, we showed how a complex curve can be represented parametrically as a sequence of cubic polynomials. This representation can be extended to get a parametric representation for complex surfaces.

Write the equation for a parametric cubic polynomial curve in matrix notation:

$$\mathbf{p}(u) = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_4 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}. \tag{13.36}$$

Note that each coefficient is actually a three-element column vector.

A tensor-product surface representation is formed from the combination of two curve representations, one in each parametric coordinate,

$$\mathbf{p}(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \mathbf{b}_4 \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}, \tag{13.37}$$

where each $\mathbf{a}_i$ is a three-element row vector, each $\mathbf{b}_j$ is a three-element column vector, and each product $\mathbf{a}_i\mathbf{b}_j$ is the pairwise product of the coefficients for each coordinate. The parametric surface can be written as

$$\mathbf{p}(u,v) = U^T M V, \tag{13.38}$$

with

$$U = \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix} \tag{13.39}$$

and

$$V = \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}. \tag{13.40}$$

The elements of the 4×4 matrix M are the vectors of coefficients for each coordinate in the parametric surface. In this notation, we can see that the tensor-product surface is literally the product of two curves: one curve in the u coordinate and another curve in the v coordinate. Any cross section through the tensor-product cubic polynomial surface, parallel to a coordinate axis in the parametric domain, is a parametric cubic polynomial curve. In other words, if one coordinate is fixed, then the result is a parametric cubic polynomial curve with the other coordinate as the curve parameter.

13.6 Surface Interpolation

This section shows how some of the surface representations presented above can be used to interpolate samples of a graph surface, such as depth measurements obtained with binocular stereo or active triangulation. Surface interpolation may be necessary when depth measurements do not conform to the uniform grid format required for image processing. It may be necessary to interpolate depth measurements onto a uniform grid before using image processing algorithms such as edge detection and segmentation.

13.6.1 Triangular Mesh Interpolation

Suppose that we have obtained samples z_k of a graph surface, $z = f(x, y)$, at scattered points (x_k, y_k) for $k = 1, \ldots, n$ using binocular stereo or active triangulation and we need to interpolate the depth measurements at grid locations $[i, j]$ in the image plane. The coordinates (x_j, y_i) at each location in the $n \times m$ grid (image array) are given by

$$x_j = j - \frac{m-1}{2} \tag{13.41}$$

$$y_i = -i + \frac{n-1}{2}. \tag{13.42}$$

We need to interpolate the z value at each point (x_j, y_i).

Use the scattered point coordinates and depth values (x_k, y_k, z_k) to create a triangular mesh. Since the depth measurements are from a graph surface, each triangle has an explicit planar equation

$$z = a_0 + a_1 x + a_2 y \tag{13.43}$$

with coefficients calculated from the formulas presented in Section 13.3.1. For each grid location, find the triangle that encloses point (x_j, y_i) and use the equation corresponding to this triangle to calculate the depth value at the grid location:

$$z_{ij} = a_0 + a_1 x_j + a_2 y_i. \tag{13.44}$$

13.6.2 Bilinear Interpolation

Sometimes it is necessary to interpolate values that already lie on a rectilinear grid. For example, rectification, discussed in Section 12.5, requires that the image intensity at a location between grid coordinates be interpolated from the pixel values at the four nearest grid locations. Bilinear interpolation is an easy method for interpolating values on a rectilinear grid.

The function

$$f(x, y) = a_1 + a_2 x + a_3 y + a_4 xy \tag{13.45}$$

is called bilinear because if one variable is set to a constant, then the function is linear in the other variable. In other words, each cross section of a bilinear surface patch taken parallel to a coordinate axis is a line segment. For any rectangle in the plane with sides parallel to the coordinate axes, there is a unique bilinear polynomial that interpolates the corner values.

Suppose that we need to interpolate the value at point (x, y) between four values on a rectilinear grid. The point is enclosed by a rectangle with sides parallel to the coordinate axes; the corners of the rectangle are the closest grid coordinates to the point at which the value is to be interpolated. The corner coordinates are (x_1, y_1), (x_1, y_2), (x_2, y_1), and (x_2, y_2) with values z_{11}, z_{12}, z_{21}, and z_{22}, respectively, as shown in Figure 13.4.

The coefficients of the bilinear interpolant are determined by the values at the four corners of the grid rectangle and are computed by plugging each grid coordinate and value into Equation 13.45:

$$\begin{aligned}
z_{11} &= a_1 + a_2 x_1 + a_3 y_1 + a_4 x_1 y_1 && (13.46)\\
z_{12} &= a_1 + a_2 x_1 + a_3 y_2 + a_4 x_1 y_2 && (13.47)\\
z_{21} &= a_1 + a_2 x_2 + a_3 y_1 + a_4 x_2 y_1 && (13.48)\\
z_{22} &= a_1 + a_2 x_2 + a_3 y_2 + a_4 x_2 y_2 && (13.49)
\end{aligned}$$

and solving the four simultaneous equations for the coefficients of the interpolant:

$$a_1 = \frac{x_2 y_2 z_{11} - x_2 y_1 z_{12} - x_1 y_2 z_{21} + x_1 y_1 z_{22}}{(x_2 - x_1)(y_2 - y_1)} \tag{13.50}$$

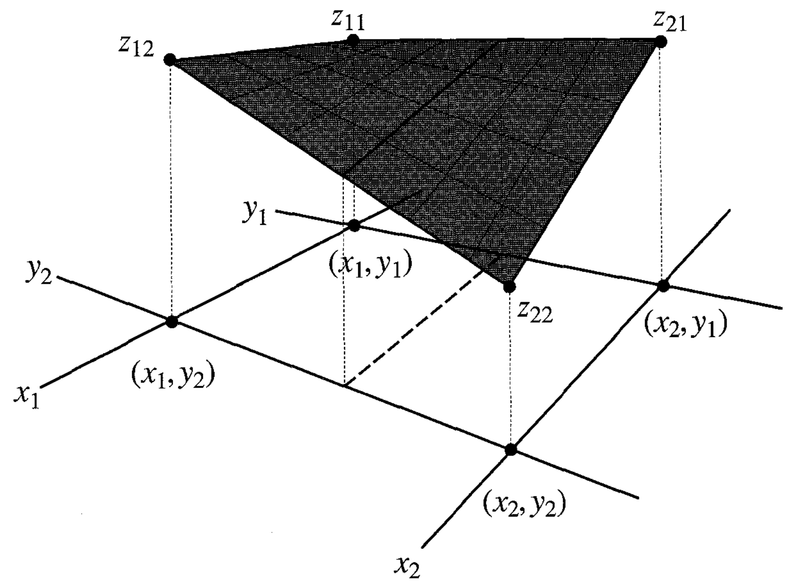

Figure 13.4: The bilinear interpolant is used to estimate the value between samples on a rectilinear grid. The values at the corners are known and uniquely determine the coefficients of the bilinear interpolant.

$$a_2 = \frac{-y_2 z_{11} + y_1 z_{12} + y_2 z_{21} - y_1 z_{22}}{(x_2 - x_1)(y_2 - y_1)} \tag{13.51}$$

$$a_3 = \frac{-x_2 z_{11} + x_2 z_{12} + x_1 z_{21} - x_1 z_{22}}{(x_2 - x_1)(y_2 - y_1)} \tag{13.52}$$

$$a_4 = \frac{z_{11} - z_{12} - z_{21} + z_{22}}{(x_2 - x_1)(y_2 - y_1)}. \tag{13.53}$$

The bilinear interpolant has a very simple form for the special case where the rectilinear grid is a square with unit spacing between the rows and columns. Let the point (x, y) at which interpolation is to be performed be given as offsets $(\delta x, \delta y)$ from the upper left corner of a grid square. The bilinear interpolant is

$$f(\delta x, \delta y) = z_{11} + \delta x(z_{21} - z_{11}) + \delta y(z_{12} - z_{11}) + \delta x\, \delta y(z_{11} - z_{12} - z_{21} + z_{22}). \tag{13.54}$$

This formula can be used to interpolate the pixel value at an image plane point that is between the pixel locations.

13.6.3 Robust Interpolation

In Section 6.8.3, we presented the least-median-squares algorithm for fitting lines to edge points with outliers. Least-median-squares regression can be used to fit surface patches to depth measurements with outliers.

Least-median-squares is a robust regression algorithm with a breakdown point of 50% like the median on which it is based. The local least-median-squares estimator uses least-median-squares regression to fit a parametric model over local neighborhoods in the data. The algorithm finds the parameters that minimize the median of the squared residuals:

$$\min_a \left\{ \operatorname*{med}_{(x_i,y_i)\in N} \left[(z_i - f(x_i,y_i;a))^2 \right] \right\} \tag{13.55}$$

where a is an estimated parameter vector and $f(x_i, y_i; a)$ is an estimate of the actual value of the graph surface at point measurement (x_i, y_i, z_i).

Least-median-squares can be used to grid surface samples that contain outliers. For example, binocular stereo depth measurements contain outliers due to mismatches, and range cameras may generate outliers at surface discontinuities. If least-median-squares is used to fit surface patches to depth measurements in local neighborhoods, then the surface fits will be immune to outliers. When the surface patches are used to grid the data, the outliers do not influence the grid values. Compare this algorithm with interpolation using a triangular mesh, presented in Section 13.6.1, which is very sensitive to outliers. Fitting surface patches with least-median-squares is very useful for processing sparse depth measurements that may be contaminated by outliers. We call this process cleaning and gridding the depth measurements.

A straightforward expression for least-median-squares is difficult to write, but the algorithm for implementing it is easy to explain. Assume that the surface patches are fit to the depth measurements in a local neighborhood about each grid point. The algorithm can be easily extended to fit higher-order surface patches. For each grid point, select the n depth measurements that are closest to the grid point. From this set, try all possible combinations of m data points, where m is the number of data points used to fit the surface patch. For each of the k subsets of the data points in the local neighborhood,

$$k = \binom{n}{m}, \tag{13.56}$$

fit a surface patch to the points in the subset and denote the corresponding parameter vector by a_k. Compare all data points in the local neighborhood with the surface patch by computing the median of the squared residuals:

$$\chi_k^2 = \operatorname*{med}_i \left[(z_i - f(x_i, y_i, a_k))^2 \right]. \tag{13.57}$$

After surface patches have been fit to all possible subsets, pick the parameter vector a_k corresponding to the surface patch with the smallest median of squared residuals.

This procedure is computationally expensive since the model fit is repeated $\binom{n}{m}$ times for each local neighborhood; however, each surface fit is independent, and the procedure is highly parallelizable. Adjacent neighborhoods share data points and could share intermediate results. In practice, it may be necessary to try only a few of the possible combinations so that the probability of one of the subsets being free of outliers is close to 1.

13.7 Surface Approximation

Depth measurements are not free of errors, and it may be desirable to find a surface that approximates the data rather than requiring the surface to interpolate the data points.

If depth measurements are samples of a graph surface,

$$z = f(x, y), \tag{13.58}$$

then it may be desirable to reconstruct this surface from the samples. If we have a model for the surface,

$$z = f(x, y; a_1, a_2, \ldots, a_m), \tag{13.59}$$

with m parameters, then the surface reconstruction problem reduces to the regression problem of determining the parameters of the surface model that best fit the data:

$$\chi^2 = \sum_{i=1}^{n} (z_i - f(x_i, y_i; a_1, a_2, \ldots, a_m))^2. \tag{13.60}$$

If we do not have a parametric model for the surface and still need to reconstruct the surface from which the depth samples were obtained, then we must fit a generic (nonparametric) surface model to the data. The task is to find the graph surface $z = f(x, y)$ that best fits the data:

$$\chi^2 = \sum_{i=1}^{n} (z_i - f(x_i, y_i))^2. \tag{13.61}$$

This is an ill-posed problem since there are many functions that can fit the data equally well; indeed, there are an infinite number of functions that interpolate the data. The term *ill-posed* means that the problem formulation does not lead to a clear choice, whereas a well-posed problem leads to a solution that is clearly the best from the set of candidates.

We need to augment the approximation norm in Equation 13.61 to constrain the selection of the approximating surface function to a single choice. There are many criteria for choosing a function to approximate the data. A popular criterion is to choose the function that both approximates the data and is a smooth surface. There are many measures of smoothness, but we will choose one measure and leave the rest for further reading. For the purposes of this discussion, the best approximation to the data points $\{(x_i, y_i, z_i)\}$ for $i = 1, \ldots, n$ is the function $z = f(x, y)$ that minimizes

$$\chi^2 = \sum_{i=1}^{n} (z_i - f(x_i, y_i))^2 + \alpha^2 \int\int \frac{\partial^2 f}{\partial x^2} + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial y^2}\, dx\, dy \tag{13.62}$$

with $\alpha > 0$. This norm is the same as Equation 13.61 except for the addition of a smoothness term, weighted by α. The smoothness term is called the regularizing term or stabilizing functional, or stabilizer for short. The weight α is called the regularizing parameter and specifies the trade-off between achieving a close approximation to the data (small α) and forcing a smooth solution (large α). The first term in Equation 13.62 is called the problem constraint. The process of augmenting a problem constraint by adding a stabilizing functional to change an ill-posed problem into a well-posed problem is called regularization.

The notion of a well-posed problem is stronger than just requiring a unique solution. An approximation problem such as Equation 13.61 may have a unique solution, but the shape of the solution space is such that many other, very different solutions may be almost as good. A well-posed problem has a solution space where the solution corresponding to the minimum of the norm is not only unique, but is definitely better than the other solutions.

The solution to the regularized surface approximation problem in Equation 13.62 is obtained by using variational calculus to transform Equation 13.62 into a partial differential equation. Numerical methods are used to change the partial differential equation into a numerical algorithm for computing samples of the approximating surface from the depth measurements.

The partial differential equation for this problem in the variational calculus is

$$\alpha^2 \nabla^4 f(x,y) + f(x,y) - z(x,y) = 0, \tag{13.63}$$

which may be solved using finite difference methods as explained in Section A.4.

Another approach is to replace the partial derivatives in Equation 13.62 with finite difference approximations and differentiate with respect to the solution f to get a system of linear equations that can be solved using numerical methods.

13.7.1 Regression Splines

Another approach to surface approximation is to change Equation 13.61 into a regression problem by substituting a surface model for the approximating function. Of course, if we knew the surface model, then we would have formulated the regression problem rather than starting from Equation 13.61. If we do not know the surface model, we can still start with Equation 13.61 and substitute a generic surface representation, such as tensor-product splines, for the approximating function and solve the regression problem for the parameters of the generic surface representation. This technique is called regression splines.

Many surface representations, including tensor-product splines, can be represented as a linear combination of basis functions:

$$f(x, y; a_0, a_1, a_2, \ldots, a_m) = \sum_{i=0}^{m} a_i B_i(x, y) \tag{13.64}$$

where a_i are the scalar coefficients and B_i are the basis functions. With tensor-product splines, the basis functions (and their coefficients) are organized into a grid:

$$f(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} B_{ij}(x, y). \tag{13.65}$$

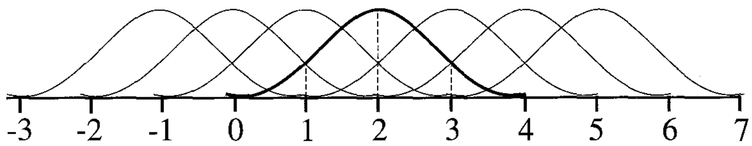

Figure 13.5: The B-spline curve in one dimension is a linear combination of basis functions located at integer positions in the interval $[0, m]$.

With either of these two equations, if we substitute the generic surface model into Equation 13.61, the result is a system of linear equations that can be solved for the regression parameters. Since the system of equations is sparse, it is better to use sparse matrix techniques, rather than singular value decomposition, for solving this regression problem.

The methods for calculating regression splines will be presented in detail, starting with the one-dimensional case. The B-spline in one dimension is a linear combination of basis functions,

$$\sum_{i=0}^{m} a_i B_i(x). \tag{13.66}$$

Assume that the B-spline basis functions are spaced uniformly at integer locations. The B-spline basis function $B_i(x)$ is nonzero over the interval $[i, i+4)$. The basis functions are located at positions 0 through m, so there are $m+1$ coefficients that define the shape of the B-spline basis curve. (Refer to Figure 13.5.)

The B-spline curve is defined on the interval from $x = 3$ to $x = m + 1$. The basis function at $x = 0$ extends three intervals to the left of the curve and the basis function at $x = m + 1$ extends three intervals to the right of the curve. The extra intervals beyond the ends of the curve establish the boundary conditions for the ends of the curve and must be included so that the first and last curve segments on the intervals $[0, 1)$ and $[m, m + 1)$ are defined correctly.

Each B-spline basis function is nonzero over four integer intervals (segments). Since the basis functions overlap, each interval is covered by four basis functions. Each segment $b_i(x)$ of the basis function $B_i(x)$ is a cubic polynomial that is defined only on its integer interval, as illustrated in Figure 13.6. The cubic polynomials for the individual segments of the B-spline

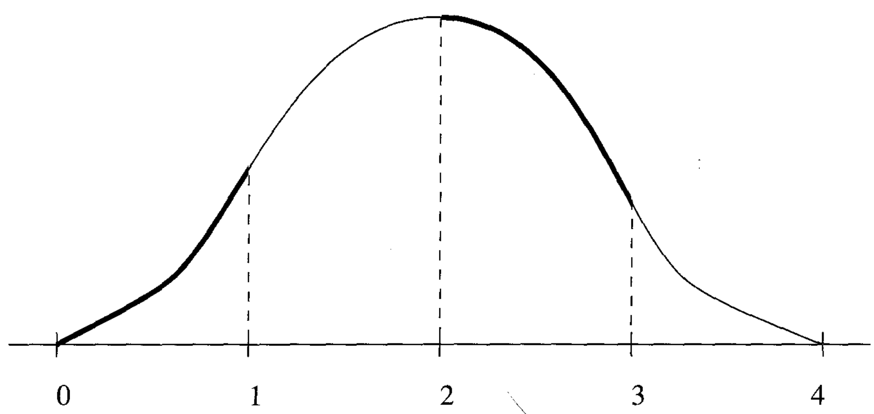

Figure 13.6: Each B-spline basis function consists of four cubic polynomials that are nonzero on adjacent integer intervals.

are

$$b_0(x) = \frac{x^3}{6} \tag{13.67}$$

$$b_1(x) = \frac{1 + 3x + 3x^2 - 3x^3}{6} \tag{13.68}$$

$$b_2(x) = \frac{4 - 6x^2 + 3x^3}{6} \tag{13.69}$$

$$b_3(x) = \frac{1 - 3x + 3x^2 - x^3}{6}. \tag{13.70}$$

It is not correct to simply add these cubic polynomials together to get a B-spline basis function since each segment is restricted to its interval; that is, each segment must be treated as nonzero outside of the interval over which it is intended to be used. To evaluate a particular B-spline basis function at a particular location x, it is necessary to determine in which segment x lies (in which piece of the B-spline basis function x is located) and evaluate the corresponding cubic polynomial.

Suppose that x is located in the interval $[i, i+1)$. The B-spline curve is the linear combination of $m+1$ B-spline basis functions:

$$\sum_{i=0}^{m} a_i B_i(x). \tag{13.71}$$

Since each B-spline basis function covers four intervals, the interval $[i, i+1)$ is covered by B-spline basis function $B_i(x)$ and the three basis functions to the left: $B_{i-1}(x)$, $B_{i-2}(x)$, and $B_{i-3}(x)$. So to evaluate the B-spline curve at x in the interval $[i, i+1)$, it is only necessary to evaluate the portion of the linear combination that corresponds to the four B-spline basis functions that cover the interval:

$$a_{i-3}B_{i-3}(x) + a_{i-2}B_{i-2}(x) + a_{i-1}B_{i-1}(x) + a_iB_i(x). \tag{13.72}$$

Since each B-spline basis function consists of four cubic polynomial segments, with each segment defined over its own interval, it is only necessary to evaluate the cubic polynomial segments from each basis function that correspond to the interval containing the location at which the cubic spline curve is being evaluated. The value of the B-spline curve at location x in interval $[i, i+1)$ is

$$\begin{aligned}\frac{(1-3x+3x^2-x^3)a_{i-3}}{6} &+ \frac{(4-6x^2+3x^3)a_{i-2}}{6} \\ &+ \frac{(1+3x+3x^2-3x^3)a_{i-1}}{6} + \frac{x^3a_i}{6}.\end{aligned} \tag{13.73}$$

To evaluate a B-spline curve at any location, it is necessary to determine the interval in which the location x lies and apply the formula in Equation 13.73, with the index i for the starting point of the interval, so that the correct coefficients for the four B-spline basis functions that cover the interval are used.

The formula for evaluating a B-spline curve in Equation 13.73 can be used for linear regression since the model is linear in the coefficients. It does not matter that the equation contains terms with powers of x. This model can be used to develop a regression algorithm for determining the coefficients of the B-spline from a set of data points (x_i, z_i). Each data point yields one equation that constrains four of the B-spline coefficients:

$$\begin{aligned}\frac{(1-3x_i+3x_i^2-x_i^3)a_{i-3}}{6} &+ \frac{(4-6x_i^2+3x_i^3)a_{i-2}}{6} \\ &+ \frac{(1+3x_i+3x_i^2-3x_i^3)a_{i-1}}{6} + \frac{x_i^3a_i}{6} = z_i.\end{aligned} \tag{13.74}$$

These equations are used to form a system of linear equations,

$$M\mathbf{a} = \mathbf{b}, \tag{13.75}$$

where the solution **a** is the column vector of B-spline coefficients a_i, the column vector on the right-hand side is the vector of data values z_i, and each row of the matrix M is zero except at the four elements corresponding to the coefficients of the B-spline basis functions that cover the interval that contains x_i. The system of linear equations can be solved using singular value decomposition, but since the system of equations is sparse, it is better to use sparse matrix techniques [197].

For surface approximation in three dimensions, the model is a tensor-product B-spline surface:

$$f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{i,j} B_{i,j}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{i,j} B_j(x) B_i(y). \tag{13.76}$$

The surface approximation problem is solved by minimizing the norm

$$\chi^2 = \sum_{k=1}^{N} (z(k) - f(x,y))^2. \tag{13.77}$$

Replace the general surface model with the expression for the tensor-product B-spline surface:

$$\chi^2 = \sum_{k=1}^{N} \left[z(k) - \left(\sum_{i=0}^{n}\sum_{j=0}^{m} a_{i,j} B_{i,j}(x,y) \right) \right]^2. \tag{13.78}$$

The data values range over a single index since the data can occur at scattered points in the plane. Each data value z_k is located at point (x_k, y_k) in the x–y plane.

The formulation of the regression problem for determining the coefficients of the tensor-product B-spline surface is similar to the one-dimensional case presented above. Each basis function $B_{ij}(x,y)$ covers 16 grid rectangles in $[i, i+4) \times [j, j+4)$. Since the basis functions overlap, each grid rectangle is covered by 16 basis functions. Each basis function consists of 16 bicubic polynomial patches. Each patch is defined on a grid rectangle and is nonzero outside of its grid rectangle.

Assume that the grid is uniform, so all grid rectangles are squares of the same size. The formulas for the bicubic patches in the B-spline basis functions are the same for each grid square. Each tensor-product basis function can be separated into the product of one-dimensional basis functions

$$B_{ij}(x,y) = B_j(x) B_i(y), \tag{13.79}$$

and each of the one-dimensional basis functions consists of four cubic polynomials. Each of the 16 polynomial patches defined on a grid rectangle is the product of two cubic polynomials, one in x and the other in y. The four cubic polynomials in x are

$$\begin{aligned} b_0(x) &= \frac{x^3}{6} && (13.80)\\ b_1(x) &= \frac{1+3x+3x^2-3x^3}{6} && (13.81)\\ b_2(x) &= \frac{4-6x^2+3x^3}{6} && (13.82)\\ b_3(x) &= \frac{1-3x+3x^2-x^3}{6}. && (13.83) \end{aligned}$$

The four cubic polynomials in y are identical except for the change of variable:

$$\begin{aligned} b_0(y) &= \frac{y^3}{6} && (13.84)\\ b_1(y) &= \frac{1+3y+3y^2-3y^3}{6} && (13.85)\\ b_2(y) &= \frac{4-6y^2+3y^3}{6} && (13.86)\\ b_3(y) &= \frac{1-3y+3y^2-y^3}{6}. && (13.87) \end{aligned}$$

The 16 polynomial patches in a B-spline basis function are formed from the pairwise products of the polynomials given above. The formula for evaluating the B-spline surface at point (x, y) in the grid is

$$\sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} b_j(x) b_i(y). \qquad (13.88)$$

Substituting the formulas for the one-dimensional cubic polynomials yields the formula for evaluating the B-spline surface. The terms for each of the 16 coefficients are listed in the following table.

$a_{i,j}$	$x^3y^3/18$
$a_{i,j-1}$	$x^3/18 + (x^3y + x^3y^2 - x^3y^3)/6$

$$
\begin{array}{ll}
a_{i,j-2} & 2x^3/9 - x^3y^2/3 + x^3y^3/6 \\
a_{i,j-3} & (x^3 - 3x^3y + 3x^3y^2 - x^3y^3)/18 \\
a_{i-1,j} & (1 + 3x + 3x^2 - 3x^3)y^3/18 \\
a_{i-1,j-1} & (1 + 3x + 3x^2 - 3x^3)/18 + (1 + 3x + 3x^2 - 3x^3)y/6 \\
 & + (1 + 3x + 3x^2 - 3x^3)y^2/6 - (1 + 3x + 3x^2 - 3x^3)y^3/6 \\
a_{i-1,j-2} & 2(1 + 3x + 3x^2 - 3x^3)/9 - (1 + 3x + 3x^2 - 3x^3)y^2/3 \\
 & + (1 + 3x + 3x^2 - 3x^3)y^3/6 \\
a_{i-1,j-3} & (1 + 3x + 3x^2 - 3x^3)/18 - (1 + 3x + 3x^2 - 3x^3)y/6 \\
 & + (1 + 3x + 3x^2 - 3x^3)y^2/6 - (1 + 3x + 3x^2 - 3x^3)y^3/18 \\
a_{i-2,j} & (4 - 6x^2 + 3x^3)y^3/18 \\
a_{i-2,j-1} & (4 - 6x^2 + 3x^3)/18 + (4 - 6x^2 + 3x^3)y/6 \\
 & + (4 - 6x^2 + 3x^3)y^2/6 - (4 - 6x^2 + 3x^3)y^3/6 \\
a_{i-2,j-2} & 2(4 - 6x^2 + 3x^3)/9 - (4 - 6x^2 + 3x^3)y^2/3 \\
 & + (4 - 6x^2 + 3x^3)y^3/6 \\
a_{i-2,j-3} & (4 - 6x^2 + 3x^3)/18 - (4 - 6x^2 + 3x^3)y/6 \\
 & + (4 - 6x^2 + 3x^3)y^2/6 - (4 - 6x^2 + 3x^3)y^3/18 \\
a_{i-3,j} & (1 - 3x + 3x^2 - x^3)y^3/18 \\
a_{i-3,j-1} & (1 - 3x + 3x^2 - x^3)/18 + (1 - 3x + 3x^2 - x^3)y/6 \\
 & + (1 - 3x + 3x^2 - x^3)y^2/6 - (1 - 3x + 3x^2 - x^3)y^3/6 \\
a_{i-3,j-2} & 2(1 - 3x + 3x^2 - x^3)/9 - (1 - 3x + 3x^2 - x^3)y^2/3 \\
 & + (1 - 3x + 3x^2 - x^3)y^3/6 \\
a_{i-3,j-3} & (1 - 3x + 3x^2 - x^3)/18 - (1 - 3x + 3x^2 - x^3)y/6 \\
 & + (1 - 3x + 3x^2 - x^3)y^2/6 - (1 - 3x + 3x^2 - x^3)y^3/18
\end{array}
$$

These expressions are evaluated at the desired location (x, y), multiplied by the corresponding coefficient $a_{i,j}$, and summed together to get the value of the B-spline surface at (x, y).

The formula for evaluating a B-spline surface is also the model used in the regression problem for determining the coefficients of the B-spline surface. There are $(n + 1) \times (m + 1)$ coefficients in the B-spline surface. Each data point (x_k, y_k, z_k) yields one equation that constrains 16 of the coefficients. As in the one-dimensional case, the regression problem leads to a system of linear equations,

$$M\mathbf{a} = \mathbf{b}, \tag{13.89}$$

where the solution $\mathbf{a}$ is the vector of B-spline coefficients (the two-dimensional grid of coefficients is unfolded into a column vector), the column vector $\mathbf{b}$ on the right-hand side is the vector of data values x_k for $k = 1, \ldots, N$, and each row of the matrix is zero except for the 16 elements corresponding to the coefficients of the basis functions that cover the grid square containing (x_k, y_k).

This method can be used to smooth images by fitting a B-spline surface to the uniform grid of pixels and sampling the spline surface. It is not necessary for the B-spline grid to correspond to the image grid; in fact, more smoothing is achieved if the B-spline grid has greater spacing. The B-spline basis function is like the Gaussian smoothing filter, covered in Chapter 4, and the widths of the B-spline basis functions are determined by the spacing of the B-spline grid. When using the formulas presented above for the B-spline regression problem, the image locations (x_j, y_i) for the pixel values z_{ij} in the image must be mapped into the grid coordinate system of the B-spline surface. After the B-spline coefficients have been determined, the B-spline surface can be sampled as needed, taking samples at the grid coordinates of the original image or at other coordinates, to calculate the pixels of the smoothed image.

The development of the equation for the B-spline surface shows that a B-spline is just a set of bicubic surface patches, which were presented in Section 13.5.2. The difference between a set of bicubic patches and the B-spline surface is that the bicubic patches in the B-spline surface are continuous up to second order, where the patches join, whereas an arbitrary collection of bicubic patches need not be continuous at all. The tensor-product B-spline surface is smooth, in the sense of having C^2 continuity, and can be used to model objects such as human bodies and organs, automobile body panels, and the skin of aircraft.

In many practical applications, the tensor-product B-spline surface must be in parametric form: $(x, y, z) = (x(u, v), y(u, v), z(u, v))$. As with the notation presented in Section 13.5.3, the coefficients for the spline surface in the regression problem will be three-element vectors, one element for each of the x, y, and z coordinates. The u–v domain is rectangular and is subdivided into a uniform grid (equal spacing between the basis functions). The most difficult part is constructing a formula for mapping (x, y, z) point measurements into the u–v domain so that the point measurements are correctly associated with the bicubic patches that make up the B-spline surface.

13.7.2 Variational Methods

The surface approximation problem can also be cast as a problem in variational calculus. The problem is to determine the surface $z = f(x, y)$ that approximates the set of depth measurements z_k at locations (x_k, y_k) in the image plane.

There are an infinite number of surfaces that will closely fit any set of depth measurements, but the best approximation can be defined as the surface that comes close to the depth measurements and is a smooth surface. The best approximation is obtained by minimizing the functional

$$\chi^2 = \sum_{k=1}^{n} (z_k - f(x_k, y_k))^2 + \int\int \left[\frac{\partial^2 f}{\partial x^2} + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial y^2} \right] dx\, dy. \qquad (13.90)$$

This minimization problem is solved using variational calculus to obtain the partial differential equation

$$\nabla^4 f(x, y) = \sum_{k=1}^{n} (z_k - f(x_k, y_k)). \qquad (13.91)$$

Appropriate boundary conditions must be chosen for the edges of the graph surface.

13.7.3 Weighted Spline Approximation

The problem with the surface approximation methods presented so far is that the solutions are smooth surfaces, even across discontinuities in the data that most likely correspond to surface boundaries in the scene. Smoothness is explicit in the smoothness functional used for the variational methods in Section 13.7.2 and is implicit in the choice of a smooth generic function for the regression splines presented in Section 13.7.1. If the approximating surface could be formulated as a set of piecewise smooth functions,

$$f(x, y) = \bigcup_{l=1}^{M} f_l(x, y), \qquad (13.92)$$

and the depth measurements could be segmented into regions with one region for each piecewise smooth function, then the solution would more accurately reflect the structure of the surfaces in the scene. However, this leads to algorithms for manipulating surfaces that are beyond the scope of this book. Some aspects of surface segmentation will be discussed in Section 13.8. We can avoid some difficult topics in surface representation and still achieve a good surface approximation that conforms to discontinuities by changing the smoothness functional in Equation 13.62 to reduce smoothing across discontinuities in the depth data. This approach leads to weighted regularization.

In one dimension, the weighted regularization norm is

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \int w(x)[f_{xx}(x)]^2\, dx. \tag{13.93}$$

If the weight function $w(x)$ is choosen to be small at discontinuities in the data and large elsewhere, then the weight function cancels the smoothness criterion at discontinuities.

In two dimensions, the norm for weighted spline surface approximation is

$$\chi^2 = \sum_{i=1}^{n}(z_i - f(x_i, y_i))^2 + \int\int w(x,y)\left[\frac{\partial^2 f}{\partial x^2} + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial y^2}\right] dx\, dy \tag{13.94}$$

with the weight function given by

$$w(x,y) = \frac{\alpha^2}{1 + \|\rho(x,y)\|^2}, \tag{13.95}$$

where $\rho(x,y)$ is the gradient of the surface from which the samples were taken. Note that the regularizing parameter α from Equation 13.62 has been absorbed into the weight function.

Variational methods can be used to transform Equation 13.94 into a partial differential equation, as shown in Section 13.7.2 and Appendix A.3, or the method of regression splines can be used to replace Equation 13.94 with a linear regression problem. The weight function in Equation 13.94 is computed from an approximation to the gradient by assuming that the weights are constant over the span of the grid on which the tensor-product basis functions are defined.

13.8 Surface Segmentation

A range image is a uniform grid of samples of a piecewise smooth graph surface

$$z = f(x, y). \tag{13.96}$$

This section will describe how a set of range samples, defined on a uniform grid, can be segmented into regions that have similar curvature and how to approximate each region with low-order bivariate polynomials. The variable-order surface segmentation algorithm, which is like the region-growing techniques described in Section 3.5, will be presented. Surface curvature properties are used to estimate core regions, which are grown to cover additional range samples. The regions are modeled by bivariate surface patches (see Section 13.5.2). The segmentation algorithm grows the regions by extending the surface patches to cover range samples that are neighbors of the region if the residual between the new range samples and the surface patch is low.

The surface segmentation problem is formally stated as follows. A piecewise smooth graph surface $z = f(x, y)$ can be partitioned into smooth surface primitives

$$z = f(x, y) = \sum_{l=1}^{n} f_l(x, y)\xi(x, y, l) \tag{13.97}$$

with one surface primitive for each region R_l. The characteristic function

$$\xi(x, y, l) = \begin{cases} 1 & \text{if } (x, y) \in R_l \\ 0 & \text{otherwise} \end{cases} \tag{13.98}$$

represents the segmentation of the range data into surface patches (regions). The surface for each region is approximated by a polynomial patch

$$f(x, y) = \sum_{i+j \leq m} a_{ij} x^i y^j \tag{13.99}$$

for points $(x, y) \in R_l$. The surface model includes planar, bilinear biquadratic, bicubic, and biquartic polynomial patches, which provide a nested hierarchy of surface representations.

The characteristic function $\xi(x, y, l)$ can be implemented as a list of the locations $[i, j]$ of the pixels that are in each region. Other representations such as bit masks or quad trees, described in Section 3.3, can be used.

		$K[i,j]$ +	$K[i,j]$ 0	$K[i,j]$ −
$H[i,j]$	−	Peak	Ridge	Saddle ridge
$H[i,j]$	0		Flat	Minimal surface
$H[i,j]$	+	Pit	Valley	Saddle valley

Table 13.1: The eight surface types defined by the signs of the mean and Gaussian curvatures.

13.8.1 Initial Segmentation

The core regions for the initial segmentation are estimated by computing the mean and Gaussian curvatures and using the signs of the mean and Gaussian curvatures to form initial region hypotheses. The portion of the regions near the region boundaries can include samples that do not belong in the regions since it can be difficult to accurately estimate the curvature in the vicinity of the transitions between regions. The regions are shrunk to core regions (see Section 2.5.12) to ensure that the initial region hypotheses do not include any falsely labeled samples. There are eight surface types corresponding to the signs of the mean and Gaussian curvatures, listed in Table 13.1. These surface types are used to form core regions.

Assume that the range data are a square grid of range samples that may be processed like an image. The first and second partial derivatives of the range image, f_x, f_y, f_{xx}, f_{xy}, f_{yy}, are estimated by convolving the range image with separable filters to estimate the partial derivatives. Compute the mean curvature

$$H[i,j] = \frac{(1+f_y^2[i,j])f_{xx}[i,j] + (1+f_x^2[i,j])\,f_{yy} - 2f_x[i,j]f_y[i,j]f_{xy}[i,j]}{2\left(\sqrt{1+f_x^2[i,j]+f_y^2[i,j]}\right)^3} \tag{13.100}$$

and the Gaussian curvature

$$K[i,j] = \frac{f_{xx}[i,j]f_{yy}[i,j] - f_{xy}^2[i,j]}{\left(1+f_x^2[i,j]+f_y^2[i,j]\right)^2}$$

using the estimates of the first and second partial derivatives of the range image. Compute an integer label that encodes the signs of the mean and Gaussian curvatures using the formula

$$T[i,j] = 1 + 3(1 + \operatorname{sgn}(H[i,j])) + (1 + \operatorname{sgn}(K[i,j])). \tag{13.101}$$

Use the sequential connected components algorithm (Section 2.5.2) to group range samples with identical labels into connected regions and reduce the size of the regions to a conservative core of range samples using a shrink or erosion operator. A size filter can be used to discard regions that are too small to correspond to scene surfaces. The regions that remain form the seeds that initialize the region-growing process.

13.8.2 Extending Surface Patches

Bivariate polynomials are fit to the range samples in the core regions. The surface fit starts with a planar patch, $m = 1$, and the order of the surface patch is increased until a good fit (root-mean-square error below a threshold) is achieved. If a good fit is not achieved, then the region is discarded.

The rest of the segmentation algorithm tries to extend the regions to cover neighboring range samples. Unassigned range samples are added to regions using a region-growing process. The similarity predicate for deciding if a range sample should be added to a region is based on comparing the value of the range sample to the bivariate surface patch evaluated at the same location as the range sample. If the residual between the surface patch and range sample is below a threshold, then the range sample is included in the set of candidates for inclusion in the region; otherwise, the range sample is not added to the region. The bivariate surface patch is refit to all range samples in the region, including the samples that are currently in the region and the set of candidates, increasing the order of the surface patch if necessary, up to the maximum order. If the fitting error is below a threshold, then the candidates are permanently added to the region; otherwise, the entire set of candidates is discarded. Region growing continues until no regions are enlarged. The segmentation algorithm is summarized in Algorithm 13.3.

Algorithm 13.3 Surface Segmentation
Segment a range image into regions represented as a surface patches.

1. *Compute the first and second partial derivatives of the range image using separable filters.*
2. *Compute the mean and Gaussian curvature at each image location.*
3. *Label each range pixel with the surface type.*
4. *Shrink the regions to eliminate false labels near the region boundaries.*
5. *Identify core regions using the sequential connected components algorithm.*
6. *Remove core regions that are too small.*
7. *Fit a bivariate patch to each region.*
8. *For each region, determine the set of neighboring pixels with values that are close to the value of the surface patch evaluated at the location of the neighbor. This set of pixels are candidates for inclusion in the region.*
9. *Refit the surface patch to the union of the set of pixels in the region and the set of candidates. If the fit is acceptable, add the set of candidates to the region; otherwise, discard the set of candidates.*
10. *Repeat steps 8 through 9 until no region is changed.*

Note that it is possible for a set of candidates to be discarded because some of the candidates should be included in another region. Later, some of the candidates may be retested and included in the region. Once a pixel is assigned to a region, the assignment cannot be reversed.

13.9 Surface Registration

The transformation between corresponding points in a set of conjugate pairs can be determined by solving the absolute orientation problem (Section 12.3), but there are applications where it is necessary to align two shapes without point correspondences. For example, it may be necessary to align a set of range samples to an object model without advance knowledge of how the point samples correspond to points on the object. This section describes the iterative closest point algorithm for determining the rigid body transformation between two views of an object without using point correspondences.

The iterative closest point algorithm can be adapted to a wide variety of object models including sets of points, curves in two or three dimensions, and various surface representations. The curves may be represented as polylines, implicit curves, or parametric curves, and the surfaces may be represented as faceted surfaces (polygonal meshes), implicit or parametric surfaces, or tensor-product B-splines. It is not necessary for both views of an object to have the same representation. For example, one view could be a set of space curves obtained with active triangulation, and the other view could be a tensor-product cubic spline surface. The key idea is that points sampled from the representation for one view can be mapped to the closest points on the representation for the other view to approximate a set of conjugate pairs. Solving the absolute orientation problem with these correspondences will improve the alignment between views. The process of finding closest point correspondences and solving the absolute orientation problem with these approximate conjugate pairs can be repeated until the two views are aligned.

The iterative closest point algorithm will be presented for the case of registering a set of points with a surface model. Given a set of points P sampled over the surface of an object and a model M for the object, determine the set of points on the model that are the closest points to each sample point. The distance between a point p in the sample set and the model M is

$$d(p, M) = \min_{q \in M} \|q - p\|. \tag{13.102}$$

Compute $Q = \{q_1, q_2, \ldots, q_n\}$, the set of points $q \in M$ that are the closest points to each point in the sample set $P = \{p_1, p_2, \ldots, p_n\}$. Pair each sample point p with its closest point q in the model to form a set of conjugate pairs $\{(p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)\}$. Solve the absolute orientation problem using this set of conjugate pairs. Even though the point sets are not true conjugate pairs, an approximate solution to the transformation between the views will be obtained. Apply the transformation to the point set P and recompute the set of closest points Q. Now the point set is closer to the model and the pairs of points in P and Q will be closer to correct conjugate pairs. The absolute orientation problem can be solved again and the new transformation applied to point set P. This procedure is repeated until the sum of the squared distances between closest points (the root-mean-square closest point distance) is below a threshold.

In the early iterations, the closest point pairs may be very poor approximations to true conjugate pairs, but applying the rigid body transformation obtained by solving the absolute orientation problem with these approximate conjugate pairs brings the point set closer to the model. As the iterations continue, the closest point pairs become more like valid conjugate pairs. For example, if the point set is initially very far from the model, then all points may be mapped to the same point on the model. Clearly, such a many-to-one mapping cannot be a valid set of conjugate pairs, but the first iteration pulls the point set to the model so that the points are centered on the matching model point and subsequent iterations align the center of the point set with the center of the model. The final iterations rotate the point set and adjust its position so that the point set is aligned with the model. The steps of iterative closest point registration are listed in Algorithm 13.4.

Algorithm 13.4 Iterative Closest Point Registration

Register a set of points to a surface modeled as a polygonal mesh.

1. *Compute the set of closest points.*

2. *Compute the registration between the point sets.*

3. *Apply the transform to register the point sets.*

4. *Return to step 1 unless the registration error is within tolerance.*

Further Reading

An excellent introduction to the differential geometry of curves and surfaces is the book by do Carmo [67]. Bartels, Beatty, and Barsky [22] have written an excellent introduction to spline curves and surfaces. The series of books, called *Graphics Gems*, contains much useful information on geometry, curves, and surfaces [89]. The most recent volume includes a disk with code for all of the algorithms presented in the series.

Surface reconstruction is a common problem in computer vision [92, 232, 233, 35]. Surface reconstruction is required for fitting a surface to sparse depth values from binocular stereo and range sensors [28, 92, 232]. Sinha and Schunck [223] have proposed a two-stage algorithm for surface reconstruction. The first stage removes outliers from the depth measurements and interpolates the sparse depth values onto a uniform grid. The second stage fits weighted bicubic splines to the output from the second stage.

The section on surface segmentation is based on the work of Besl and Jain [28]. The iterative closest point algorithm was developed by Besl and McKay [29].

Exercises

13.1 Define implicit, explicit, and parametric representations of a surface. When we consider an image as a two-dimensional mathematical function, can we consider the image as a surface? If so, which of the three representations above corresponds to an image?

13.2 Given four points in space, how will you determine whether they are coplanar?

13.3 Define principal curvature, Gaussian curvature, and mean curvature for a surface. What information about the nature of a surface do you get by knowing these curvature values at a point? Do you get local information or global information about the surface from these curvature values?

13.4 What is a spline curve? Why are cubic splines frequently used to represent curves?

13.5 Polygonal meshes are commonly used to represent surfaces. What are polygonal meshes? Why are they used to represent arbitrary curved surfaces? How closely can you represent an arbitrary curved surface using a polygonal mesh?

13.6 What is a winged edge data structure? Where and why is one used?

13.7 What is a graph surface? Show that a graph surface can be modeled using a bivariate polynomial surface representation. Discuss the limitations of polynomial patch representations for surfaces.

13.8 Define tensor product surface. Compare it with the polynomial surface representation for applications in machine vision. You should consider three concrete applications in inspection, measurement, and object recognition and study the requirements of these applications. Then compare these representations for the specific task requirements.

13.9 What is the difference between surface approximation and surface interpolation? What is segmentation: approximation or interpolation? Explain your answer.

13.10 What is regularization? When should it be used in computer vision? Explain your answer using an example where regularization may be used to solve a problem effectively.

13.11 Give the mathematical formulation of a B-spline surface. How do you evaluate the quality of fit for these surfaces?

13.12 How can you use B-splines for smoothing imges? Which type of applications will be suitable for spline smoothing?

13.13 Using differential geometric characteristics, it is possible to classify the local nature of a surface as one of the eight surface types. Which characteristics are used and what are the classes?

Computer Projects

13.1 Write a program to segment images, including range images, starting with a seed that contains local regions of similar type based on the differential geometric characteristics. Use a region growing method to fit biquadratic and bicubic surfaces in an image. Select suitable region growing and region termination criterion. Apply this program to several images. Study the error characteristics of surfaces approximated by your program.

13.2 Implement an algorithm to take scattered three-dimensional points and convert them to a uniform grid representation using an interpolation algorithm. Test the algorithm using the output of a stereo algorithm.