Linear Interpolation

Given 4 neighbors with coordinates $f(n_{10}, n_{20}), f(n_{11}, n_{21}), f(n_{12}, n_{22}), f(n_{13}, n_{23})$
.

Define new gray level

$$g(n_1, n_2) = A_0 + A_1 n_1 + A_2 n_2 + A_3 n_1 n_2$$

Then

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 1 & n_{10} & n_{20} & n_{10}n_{20} \\ 1 & n_{11} & n_{21} & n_{11}n_{21} \\ 1 & n_{12} & n_{22} & n_{12}n_{22} \\ 1 & n_{13} & n_{23} & n_{13}n_{23} \end{bmatrix}^{-1} \begin{bmatrix} f(n_{10}, n_{20}) \\ f(n_{11}, n_{21}) \\ f(n_{12}, n_{22}) \\ f(n_{13}, n_{23}) \end{bmatrix}$$

Note - the mapping coordinates need not fit within the pixel ranges

Advanced topics:

- cubic interpolation
- splines
- WENO

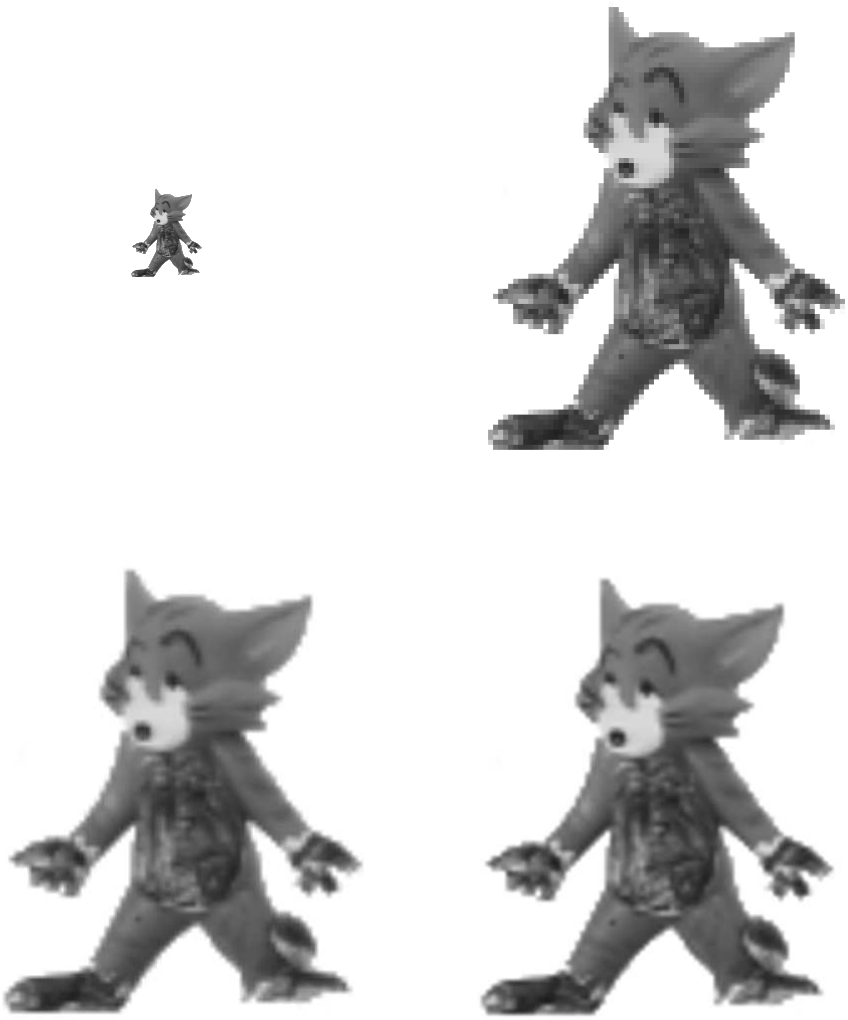
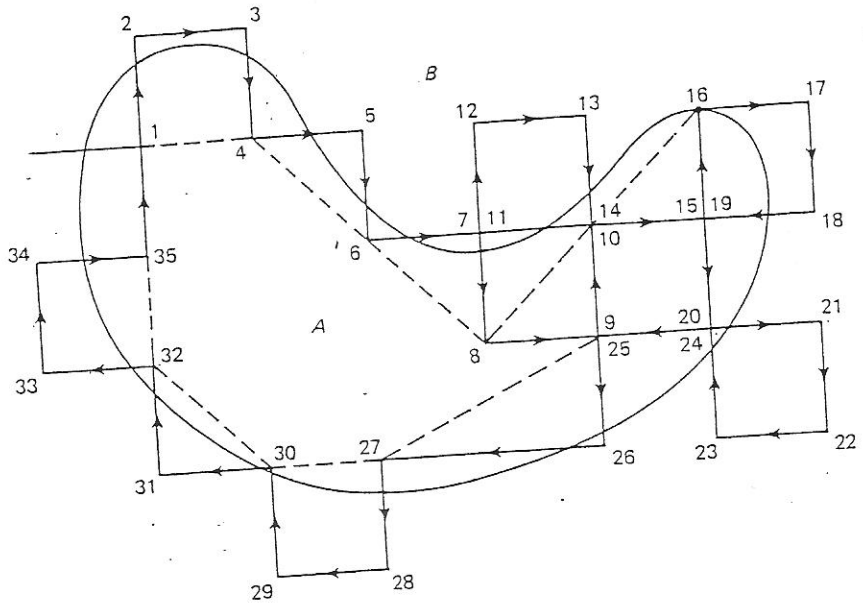


Figure 6: **Image interpolation methods:** the original image (top left) was enlarged five times using three different interpolation techniques - nearest neighbor (top right), bilinear (bottom left), and bicubic (bottom right).

septic). However, their experiments showed only marginal improvement in comparison with cubic interpolation at an highly increased computational cost.

Several survey papers on resampling techniques have been published in the last years. A detailed investigation and comparison of methods was carried out in [146] for 2-D images and in [79] for 3-D data. Thevenaz et al. [191] paid attention to the elimination of undesired interpolation artifacts. Lehman et al. [114] published a survey article covering main interpolation methods (various versions of *sinc* functions, nearest neighbor, linear, quadratic, cubic, cubic B-spline, Lagrange and Gaussian kernels) with the emphasis on medical imaging applications. They compared them using the spatial and Fourier analysis and tested the computational complexity as well as interpolation errors. Most recently, Thevenaz et

A boundary can also be viewed as a path through a graph formed by linking the edge elements together. Linkage rules give the procedure for connecting the edge elements. Suppose a graph with node locations x_k, $k = 1, 2, \ldots$ is formed from node A to node B. Also, suppose we are given an *evaluation function* $\phi(x_k)$, which gives the value of the path from A to B constrained to go through the node x_k. In *heuristic search algorithms,* we examine the successors of the start node and select the node that maximizes $\phi(\cdot)$. The selected node now becomes the new start node and the process is repeated until we reach B. The sequence of selected nodes then constitutes the boundary path. The speed of the algorithm depends on the chosen $\phi(\cdot)$ [20, 21]. Note that such an algorithm need not give the globally optimum path.



Algorithm
1. Start inside A (e.g., 1)
2. Turn left and step to next pixel if in region A, (e.g., 1 to 2), otherwise turn right and step (e.g., 2 to 3)
3. Continue until arrive at starting point 1

Figure 9.15 Contour following in a binary image.

Example 9.2 Heuristic search algorithms [19]

Consider a 3×5 array of edges whose gradient magnitudes $|g|$ and tangential contour directions θ are shown in Fig. 9.16a. The contour directions are at 90° to the gradient directions. A pixel X is considered to be linked to Y if the latter is one of the three eight-connected neighbors (Y_1, Y_2, or Y_3 in Fig. 9.16b) *in front* of the contour direction and if $|\theta(x) - \theta(y)| < 90°$. This yields the graph of Fig. 9.16c.

As an example, suppose $\phi(x_k)$ is the sum of edge gradient magnitudes along the path from A to x_k. At A, the successor nodes are D, C, and G, with $\phi(D) = 12$, $\phi(C) = 6$, and $\phi(G) = 8$. Therefore, node D is selected, and C and G are discarded. From here on nodes E, F, and B provide the remaining path. Therefore, the boundary path is $ADEFB$. On the other hand, note that path $ACDEFB$ is the path of maximum cumulative gradient.

with the knots are *nodes* s_i, which are defined as the mean locations of successive $k-1$ knots, that is,

$$s_i \triangleq \frac{1}{k-1}(t_{i+1} + t_{i+2} + \cdots + t_{i+k-1}), \qquad 0 \le i \le n, \qquad k \ge 2 \tag{9.36}$$

The variable t is also called the node parameter, of which t_i and s_i are special values. Figure 9.23 shows some of the B-spline functions. These functions are nonnegative and have finite support. In fact, for the normalized B-splines, $0 \le B_{i,k}(t) \le 1$ and the region of support of $B_{i,k}(t)$ is $[t_i, t_{i+k})$. The functions $B_{i,k}(t)$ form a basis in the space of *piecewise-polynomial functions*. These functions are called *open B-splines* or *closed (or periodic) B-splines*, depending on whether the boundary being represented is open or closed. The parameter k controls the order of continuity of the curve. For example, for $k = 3$ the splines are piecewise quadratic polynomials. For $k = 4$, these are cubic polynomials. In computer graphics $k = 3$ or 4 is generally found to be sufficient.

When the knots are uniformly spaced, that is,

$$t_{i+1} - t_i = \Delta t, \qquad \forall i \tag{9.37a}$$

the $B_{i,k}(t)$ are called uniform splines and they become translates of $B_{0,k}(t)$, that is,

$$B_{i,k}(t) = B_{0,k}(t - i), \qquad i = k-1, k, \ldots, n-k+1 \tag{9.37b}$$

Near the boundaries $B_{i,k}(t)$ is obtained from (9.35). For uniform open B-splines with $\Delta t = 1$, the knot values can be chosen as

$$t_i = \begin{cases} 0 & i < k \\ i - k + 1 & k \le i \le n \\ n - k + 2 & i > n \end{cases} \tag{9.38}$$

and for uniform periodic (or closed) B-splines, the knots can be chosen as

$$t_i = i \bmod (n+1) \tag{9.39}$$

$$B_{i,k}(t) = B_{0,k}[(t - i) \bmod (n+1)] \tag{9.40}$$

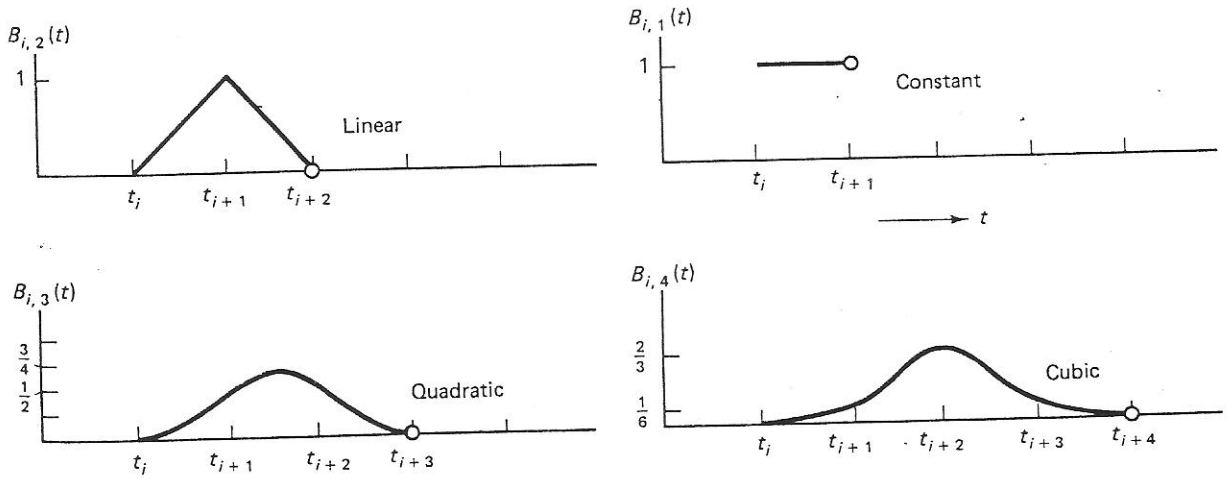


Figure 9.23 Normalized B-splines of order $K = 1, 2, 3, 4$.

Fitting Line Segments [1]

Straight-line segments give simple approximation of curve boundaries. An interesting sequential algorithm for fitting a curve by line segments is as follows (Fig. 9.22).

Algorithm. Approximate the curve by the line segment joining its end points (A, B). If the distance from the farthest curve point (C) to the segment is greater than a predetermined quantity, join AC and BC. Repeat the procedure for new segments AC and BC, and continue until the desired accuracy is reached.

B-Spline Representation [27–29]

B-splines are piecewise polynomial functions that can provide local approximations of contours of shapes using a small number of parameters. This is useful because human perception of shapes is deemed to be based on curvatures of parts of contours (or object surfaces) [30]. This results in compression of boundary data as well as smoothing of coarsely digitized contours. B-splines have been used in shape synthesis and analysis, computer graphics, and recognition of parts from boundaries.

Let t be a boundary curve parameter and let $x(t)$ and $y(t)$ denote the given boundary addresses. The B-spline representation is written as

$$\begin{aligned} \mathbf{x}(t) &= \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(t) \\ \mathbf{x}(t) &\triangleq [x(t), y(t)]^T, \qquad \mathbf{p}_i \triangleq [p_{1i}, p_{2i}]^T \end{aligned} \tag{9.34}$$

where $\mathbf{p}_i$ are called the *control points* and the $B_{i,k}(t), i = 0, 1, \ldots, n, k = 1, 2 \ldots$ are called the *normalized B-splines of order k*. In computer graphics these functions are also called *basis splines* or *blending functions* and can be generated via the recursion

$$B_{i,k}(t) \triangleq \frac{(t - t_i)B_{i,k-1}(t)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - t)B_{i+1,k-1}(t)}{(t_{i+k} - t_{i+1})}, \qquad k = 2, 3, \ldots \tag{9.35a}$$

$$B_{i,1}(t) \triangleq \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{9.35b}$$

where we adopt the convention $0/0 \triangleq 0$. The parameters $t_i, i = 0, 1, \ldots$ are called the *knots*. These are locations where the spline functions are tied together. Associated

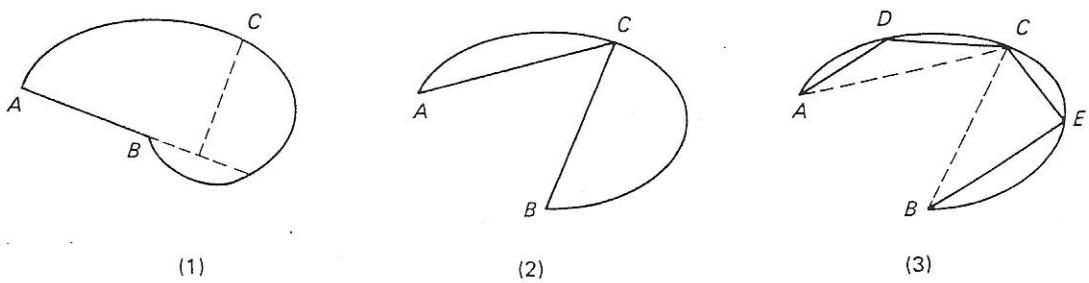


Figure 9.22 Successive approximation by line segments.

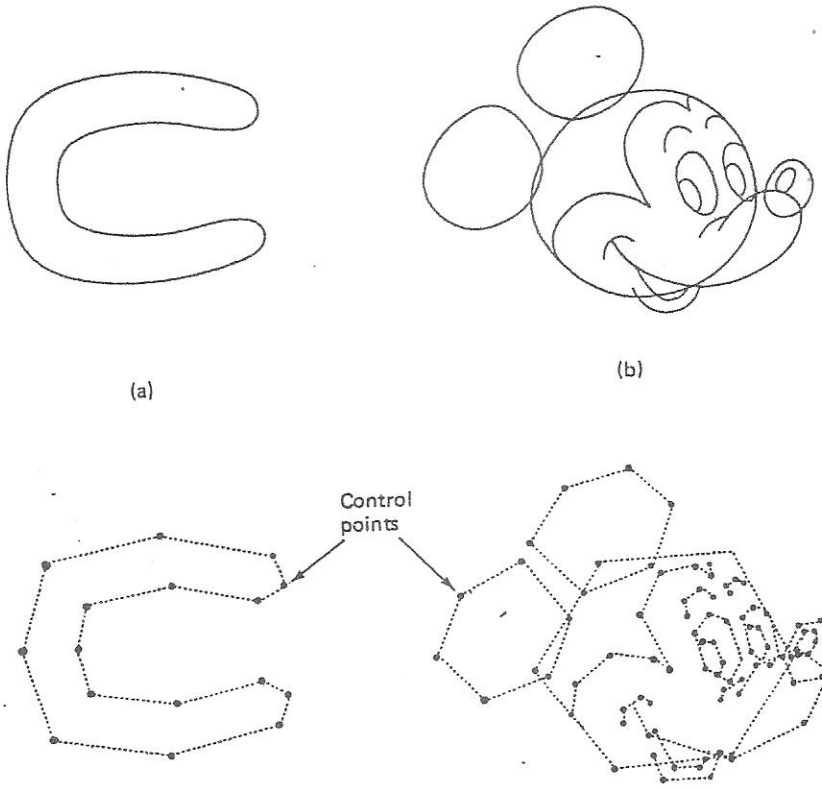


Figure 9.24 (a), (b) B-spline curves fitted through 128 and 1144 points of the original boundaries containing 1038 and 10536 points respectively to yield indistinguishable reproduction: (c), (d) corresponding 16 and 99 control points respectively. Since (a), (b) can be reproduced from (c), (d), compression ratios of greater than 100 : 1 are achieved.

where $\mathbf{B}_k$, $\mathbf{P}$, and $\mathbf{x}$ are $(n+1)\times(n+1)$, $(n+1)\times 2$, and $(n+1)\times 2$ matrices of elements $B_{i,k}(s_j)$, $\mathbf{p}_i$, $\mathbf{x}(s_j)$ respectively. When s_j are the node locations the matrix $\mathbf{B}_k$ is guaranteed to be nonsingular and the control-point array is obtained as

$$\mathbf{P} = \mathbf{B}_k^{-1}\mathbf{x} \tag{9.46}$$

For uniformly sampled closed splines, $\mathbf{B}_k$ becomes a circulant matrix, whose first row is given by

$$\mathbf{b}_0 \triangleq [b_0 b_1 \ldots b_q, 0 \ldots 0, b_q \ldots b_1] \tag{9.47}$$

where

$$b_j \triangleq B_{0,k}(s_j), \qquad j = 0, \ldots, q, \quad \text{and} \quad q = \text{Integer}\left[\frac{(k-1)}{2}\right]$$

$$s_{j+1} - s_j = t_{j+1} - t_j = \text{constant}, \qquad \forall j$$

In the case of open B-splines, $\mathbf{B}_k$ is nearly Toeplitz when $s_j = t_j$, for every j.

From these we obtain

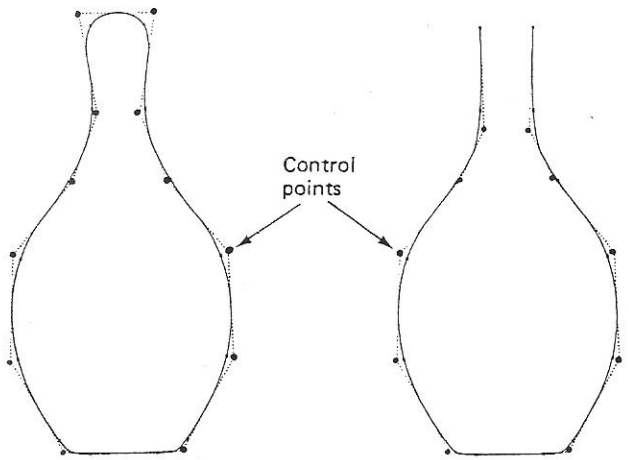
$$\mathbf{B}_3 = \tfrac{1}{8}\begin{bmatrix} 8 & & & & & & & \\ 2 & 5 & 1 & & & & \mathbf{0} & \\ & 1 & 6 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & 1 & 6 & 1 & & \\ & \mathbf{0} & & & 1 & 5 & 2 & \\ & & & & & & & 8 \end{bmatrix}_{n+1}^{n+1}$$

Figure 9.25 shows a set of uniformly sampled spline boundary points $x(s_j)$, $j = 0, \ldots, 13$. Observe that the s_j, and not $x(s_i), y(s_i)$, are required to be uniformly spaced. Periodic and open quadratic B-spline interpolations are shown in Fig. 9.25b and c. Note that in the case of open B-splines, the endpoints of the curve are also control points.

The foregoing method of extracting control points from uniformly sampled boundaries has one remaining difficulty—it requires the number of control points be equal to the number of sampled points. In practice, we often have a large number of finely sampled points on the contour and, as is evident from Fig. 9.24, the number of control points necessary to represent the contour accurately may be much smaller. Therefore, we are given $\mathbf{x}(t)$ for $t = \xi_0, \xi_1, \ldots, \xi_m$, where $m \gg n$. Then we have an overdetermined $(m + 1) \times (n + 1)$ system of equations

$$\mathbf{x}(\xi_j) = \sum_{i=0}^{n} B_{i,k}(\xi_j)\,\mathbf{p}_i, \qquad j = 0, \ldots, m \tag{9.48}$$

Least squares techniques can now be applied to estimate the control points $\mathbf{p}_i$. With proper indexing of the sampling points ξ_j and letting the ratio $(m + 1)/(n + 1)$



(a) (b) (c)

Figure 9.25 (a) Given points; (b) quadratic periodic B-spline interpolation; (c) quadratic nonperiodic B-spline interpolation.