Chapter 9

Three-Dimensional Lookup Table with Interpolation

Color space transformation using a 3D lookup table (LUT) with interpolation is used to correlate the source and destination color values in the lattice points of a 3D table, where nonlattice points are interpolated by using the nearest lattice points. This method has been used in many applications with quite satisfactory results, and incorporated into the ICC profile standard.[1]

In this chapter, the structure of the 3D-LUT approach is discussed and the mathematical formulations of interpolation methods are given. These methods are tested using several sets of data points. The similarities and differences of these interpolation methods are discussed.

9.1 Structure of 3D Lookup Table

The 3D lookup method consists of three parts—packing (or partition), extraction (or find), and interpolation (or computation).[2] Packing is a process that partitions the source space and selects sample points for the purpose of building a lookup table. The extraction step is aimed at finding the location of the input pixel and extracting color values of the nearest lattice points. The last step is interpolation where the input signals and the extracted lattice points are used to calculate the destination color specifications for the input point.

9.1.1 Packing

Packing is a process that divides the domain of the source space and populates it with sample points to build the lookup table. Generally, the table is built by an equal step sampling along each axis of the source space, as shown in Fig. 9.1, of a five-level LUT. This will give $(n-1)^3$ cubes and n^3 lattice points, where n is the number of levels. The advantage of this arrangement is that it implicitly supplies information about which cell is next to which. Thus, one needs only to store the starting point and the spacing for each axis. Generally, a matrix of n^3 color patches at the lattice points of the source space is made and the destination color specifications of these patches are measured. The corresponding values from the source and destination spaces are tabulated into a lookup table.

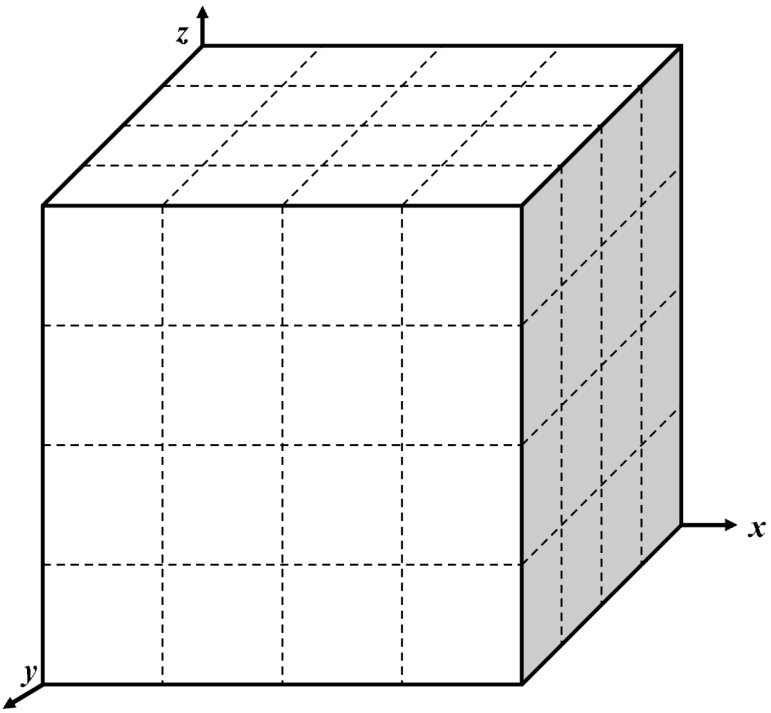

Figure 9.1 A uniformly spaced five-level 3D packing.

Nonuniformly spaced LUTs are also used extensively. They lose the simplicity of the implementation edges, but gain the versatility and conversion accuracy as discussed in Section 9.7.

9.1.2 Extraction

Nonlattice points are interpolated by using the nearest lattice points. This is the step where the extraction performs a search to select the lattice points necessary for computing the destination specification of the input point. A well-packed space can make the search simpler. In an 8-bit integer environment, for example, if the axis is divided into 2^j equal sections where j is an integer smaller than 8, then the nearest lattice points are given in the most significant j bits (MSB_j) of the input color signals. In other words, the input point is bounded between the lattice points $\mathrm{p}(\mathrm{MSB}_j)$ and $\mathrm{p}(\mathrm{MSB}_j + 1)$. To find the point of interest requires the computer operations of the masking byte and shifting bits that are significantly faster than the comparison operations. For unequally spaced packing, a series of comparisons are needed to locate the nearest lattice points.

Further search within the cube may be needed, depending on the interpolation technique employed to compute the color values of nonlattice points. There are two interpolation methods: the geometrical method and cellular regression. Cellular regression does not require a search within the cube. All geometric interpolations

except the trilinear approach require a search mechanism to find the subdivided structure where the point resides. These search mechanisms are inequality comparisons.

9.1.3 Interpolation

Interpolation uses the input signals and the extracted lattice points that contain the destination specifications to calculate the destination color specifications for the input point. Interpolation techniques are mathematical computations that employ geometrical relationships or cellular regression. Geometrical interpolations exploit the ways of subdividing a cube. There are four geometrical interpolations, trilinear, prism, pyramid, and tetrahedral. The first 3D interpolation that appeared in the literature is the trilinear interpolation, disclosed in a 1974 British patent by Pugsley.[3] The prism scheme was published in 1992 by Kanamori, Fumoto, and Kotera.[4] This prism architecture has been made into a single-chip color processor by Matsushita Research Institute Tokyo, Inc.[5,6] The pyramid interpolation was patented by Flanklin in 1982.[7] The idea of linear interpolation using tetrahedral segmentation of a cube was published as early as 1967 by Iri.[8] A similar concept of a linear interpolation by searching for the nearest four neighbors that enclose the point of interest and form a tetrahedron was applied to compute dot areas of color scanners by Korman and Yule in 1971.[9] The application of tetrahedral interpolation to color-space transformation was later patented by Sakamoto and Itooka in U.S. Patent No. 4,275,413 (1981) and related worldwide patents.[10–12] The extensive activities in developing and patenting interpolation techniques during the 1980s show the importance of the technique, the desire of dominating market share by the manufacturers, and the subsequent financial stakes.

9.2 Geometric Interpolations

Basically, 3D interpolation is the multiple application of the linear interpolation; therefore, we start with the linear interpolation, then extend to 2D (bilinear) and 3D (trilinear) interpolations. A linear interpolation is depicted in Fig. 9.2; a point p on the curve between the lattice points p_0 and p_1 is to be interpolated. The interpolated value $p_c(x)$ is linearly proportional to the ratio of $(x - x_0)/(x_1 - x_0)$, where $(x_1 - x_0)$ is the projected length of the line segment connecting points p_0 and p_1, and $(x - x_0)$ is the projected distance of the line connecting points p and p_0.

$$p_c(x) = p(x_0) + [(x - x_0)/(x_1 - x_0)][p(x_1) - p(x_0)]. \tag{9.1}$$

As shown in Eq. (9.1), the major computational operation in the interpolation is to calculate the projected distances. In view of the implementation, using a uniform 8-bit LUT, the projected distance at each axis of an input is simply $p(\mathrm{LSB}_{8-j})$,

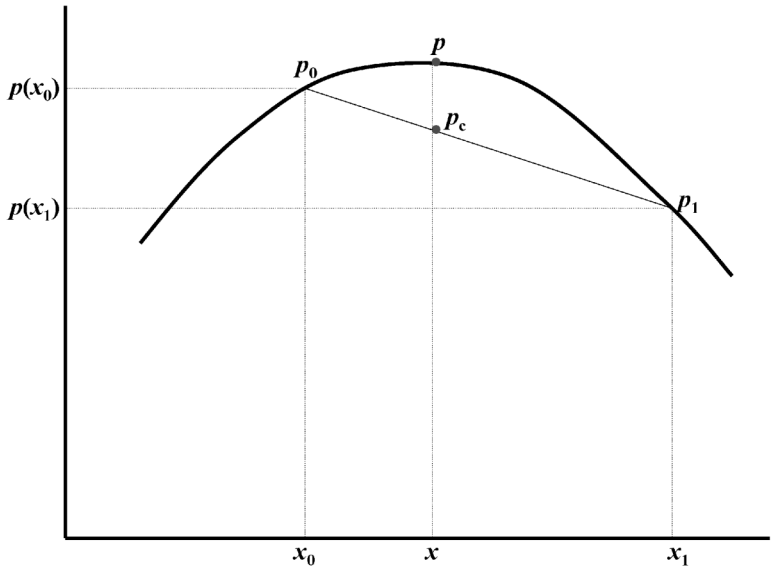

Figure 9.2 Linear interpolation.

where the LSBs are the least significant bits. This is a simple byte-masking operation that significantly lowers the computational cost and increases speed. The interpolation error is given as

$$\delta = p(x) - p_c(x). \tag{9.2}$$

9.2.1 Bilinear interpolation

In two dimensions, we have a function of two variables $p(x, y)$ and four lattice points $p_{00}(x_0, y_0)$, $p_{01}(x_0, y_1)$, $p_{10}(x_1, y_0)$, and $p_{11}(x_1, y_1)$ as shown in Fig. 9.3. To obtain the value for point p, we first hold y_0 constant and apply the linear interpolation on lattice points p_{00} and p_{10} to obtain p_0.

$$p_0 = p_{00} + (p_{10} - p_{00})[(x - x_0)/(x_1 - x_0)]. \tag{9.3}$$

Similarly, we calculate p_1 by keeping y_1 constant.

$$p_1 = p_{01} + (p_{11} - p_{01})[(x - x_0)/(x_1 - x_0)]. \tag{9.4}$$

After obtaining p_0 and p_1, we again apply the linear interpolation to them by keeping x constant.

$$p(x, y) = p_0 + (p_1 - p_0)[(y - y_0)/(y_1 - y_0)]. \tag{9.5}$$

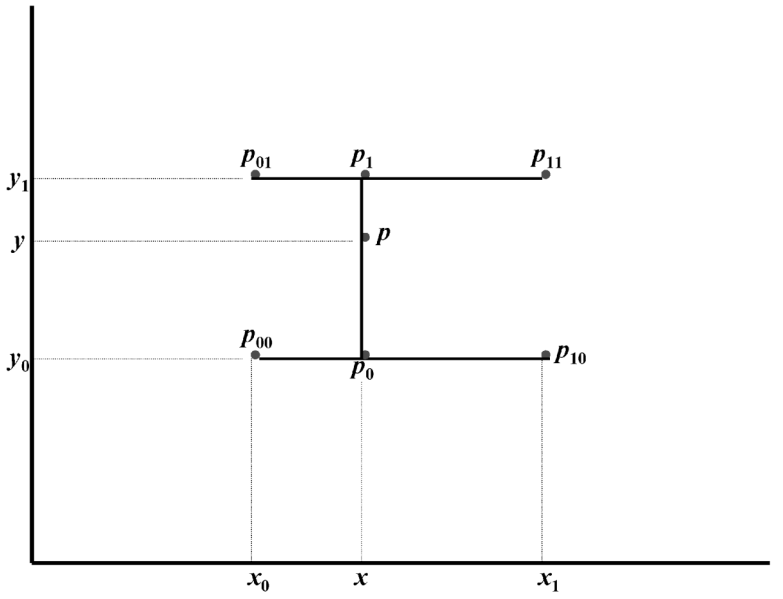

Figure 9.3 Bilinear interpolation.

Substituting Eqs. (9.3) and (9.4) into Eq. (9.5), we obtain

$$\begin{aligned} p(x, y) = {} & p_{00} + (p_{10} - p_{00})[(x - x_0)/(x_1 - x_0)] \\ & + (p_{01} - p_{00})[(y - y_0)/(y_1 - y_0)] \\ & + (p_{11} - p_{01} - p_{10} + p_{00})[(x - x_0)/(x_1 - x_0)] \\ & \times [(y - y_0)/(y_1 - y_0)]. \end{aligned} \tag{9.6}$$

9.2.2 Trilinear interpolation

The trilinear equation is derived by applying the linear interpolation seven times (see Fig. 9.4); three times each to determine the points p_1 and p_0 as illustrated in the 2D bilinear interpolation, then one more time to compute the point p. The general expression for the trilinear interpolation is given in Eq. (9.7).

$$\begin{aligned} p(x, y, z) = {} & c_0 + c_1\Delta x + c_2\Delta y + c_3\Delta z + c_4\Delta x\Delta y \\ & + c_5\Delta y\Delta z + c_6\Delta z\Delta x + c_7\Delta x\Delta y\Delta z, \end{aligned} \tag{9.7a}$$

where Δx, Δy, and Δz are the relative distances of the point with respect to the starting point p_{000} in the x, y, and z directions, respectively, as shown in Eq. (9.7b).

$$\Delta x = (x - x_0)/(x_1 - x_0); \quad \Delta y = (y - y_0)/(y_1 - y_0); \quad \Delta z = (z - z_0)/(z_1 - z_0). \tag{9.7b}$$

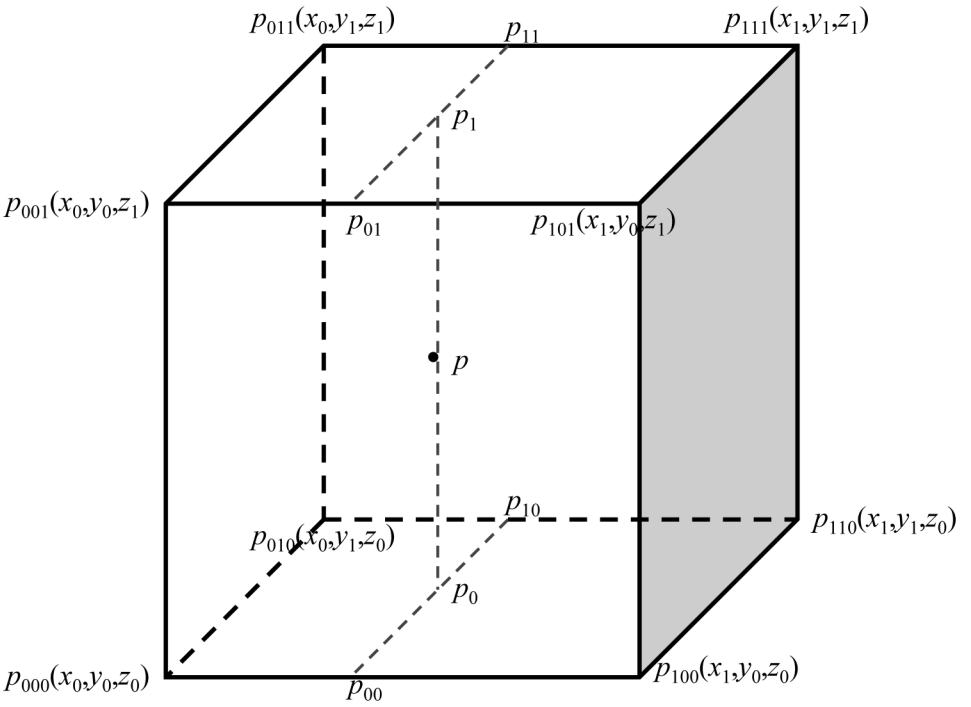

Figure 9.4 Trilinear interpolation.

Coefficients c_j are determined from the values of the vertices.

$$
\begin{aligned}
&c_0 = p_{000};\ c_1 = (p_{100} - p_{000});\ c_2 = (p_{010} - p_{000});\\
&c_3 = (p_{001} - p_{000});\ c_4 = (p_{110} - p_{010} - p_{100} + p_{000});\\
&c_5 = (p_{011} - p_{001} - p_{010} + p_{000});\ c_6 = (p_{101} - p_{001} - p_{100} + p_{000});\\
&c_7 = (p_{111} - p_{011} - p_{101} - p_{110} + p_{100} + p_{001} + p_{010} - p_{000}).
\end{aligned}
\tag{9.7c}
$$

Equation (9.7a) can be written in vector-matrix form as

$$
p = \boldsymbol{C}^{\mathrm{T}} Q_1 \quad \text{or} \quad p = Q_1^{\mathrm{T}} \boldsymbol{C}, \tag{9.7d}
$$

where $\boldsymbol{C}$ is the vector of coefficients,

$$
\boldsymbol{C} = [c_0 \quad c_1 \quad c_2 \quad c_3 \quad c_4 \quad c_5 \quad c_6 \quad c_7]^{\mathrm{T}}, \tag{9.8}
$$

and $\boldsymbol{Q}_1$ is the vector of distances related to Δx, Δy, and Δz.

$$
\boldsymbol{Q}_1 = [1 \quad \Delta x \quad \Delta y \quad \Delta z \quad \Delta x \Delta y \quad \Delta y \Delta z \quad \Delta z \Delta x \quad \Delta x \Delta y \Delta z]^{\mathrm{T}}. \tag{9.9}
$$

Note that the sizes of vectors $\boldsymbol{Q}_1$ and $\boldsymbol{C}$ must be the same. The coefficients $\boldsymbol{C}$ can be put into vector-matrix form as shown in Eq. (9.10a) by expanding Eq. (9.7c).

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ p_{011} \\ p_{100} \\ p_{101} \\ p_{110} \\ p_{111} \end{bmatrix}, \tag{9.10a}$$

or

$$\boldsymbol{C} = \boldsymbol{B}_1 \boldsymbol{P}, \tag{9.10b}$$

where vector $\boldsymbol{P}$ is a collection of vertices,

$$\boldsymbol{P} = [p_{000} \quad p_{001} \quad p_{010} \quad p_{011} \quad p_{100} \quad p_{101} \quad p_{110} \quad p_{111}]^{\mathrm{T}}, \tag{9.11}$$

and the matrix $\boldsymbol{B}_1$, given in Eq. (9.10a), represents a matrix of binary constants, having a size of 8×8.

Substituting Eq. (9.10b) into Eq. (9.7d), we obtain the vector-matrix expression for calculating the destination color value of point p.

$$p = \boldsymbol{C}^{\mathrm{T}} \boldsymbol{Q}_1 = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{B}_1^{\mathrm{T}} \boldsymbol{Q}_1, \tag{9.12a}$$

$$p = \boldsymbol{Q}_1^{\mathrm{T}} \boldsymbol{C} = \boldsymbol{Q}_1^{\mathrm{T}} \boldsymbol{B}_1 \boldsymbol{P}. \tag{9.12b}$$

Equation (9.12) is exactly the same as Eq. (9.7), only expressed differently. There is no need for a search mechanism to find the location of the point because the cube is used as a whole. But the computation cost, using all eight vertices and having eight terms in the equation, is higher than other 3D geometric interpolations.

9.2.3 Prism interpolation

If one cuts a cube diagonally into two halves as shown in Fig. 9.5, one gets two prism shapes. A search mechanism is needed to locate the point of interest. Because there are two symmetric structures in the cube, a simple inequality comparison is sufficient to determine the location: if $\Delta x > \Delta y$, then the point is in Prism 1, otherwise the point is in Prism 2. For $\Delta x = \Delta y$, the point is laid on the diagonal plane, and either prism can be used for interpolation.

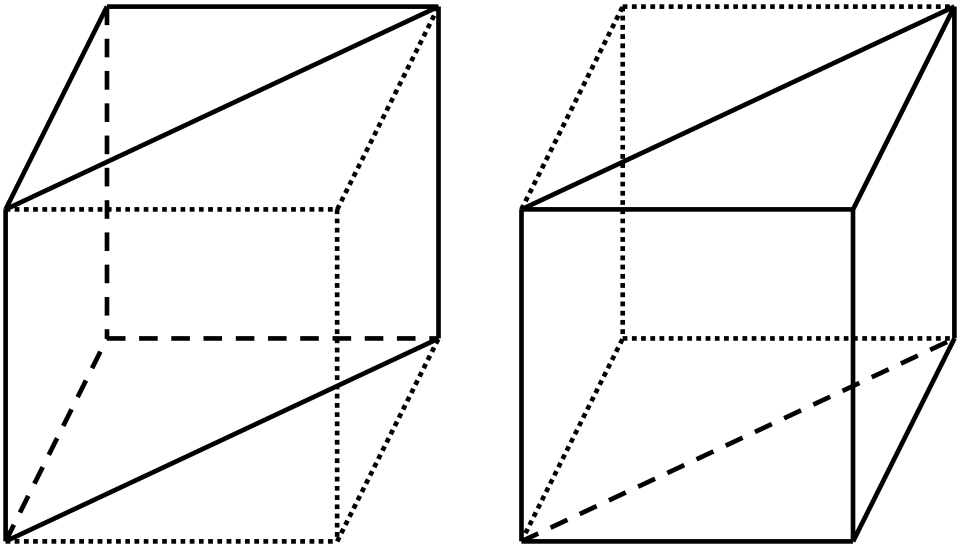

Figure 9.5 Prism interpolation.

The equation has six terms and uses six vertices of the given prism for computation. Equation (9.13) gives prism interpolation in vector-matrix form.

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} p_{000} & (p_{100}-p_{000}) & (p_{110}-p_{100}) & (p_{001}-p_{000}) \\ p_{000} & (p_{110}-p_{010}) & (p_{010}-p_{000}) & (p_{001}-p_{000}) \end{bmatrix} \begin{bmatrix} (p_{101}-p_{001}-p_{100}+p_{000}) & (p_{111}-p_{101}-p_{110}+p_{100}) \\ (p_{111}-p_{011}-p_{110}+p_{010}) & (p_{011}-p_{001}-p_{010}+p_{000}) \end{bmatrix} \times \begin{bmatrix} 1 \\ \Delta x \\ \Delta y \\ \Delta z \\ \Delta x \Delta z \\ \Delta y \Delta z \end{bmatrix}. \tag{9.13}$$

By setting

$$\boldsymbol{Q}_2 = [1 \quad \Delta x \quad \Delta y \quad \Delta z \quad \Delta x \Delta z \quad \Delta y \Delta z]^{\mathrm{T}}, \tag{9.14}$$

Eq. (9.13) can be expressed in vector-matrix form as given in Eq. (9.15).

$$p_1 = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{B}_{21}^{\mathrm{T}} \boldsymbol{Q}_2 = \boldsymbol{Q}_2^{\mathrm{T}} \boldsymbol{B}_{21} \boldsymbol{P}, \tag{9.15a}$$

$$p_2 = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{B}_{22}^{\mathrm{T}} \boldsymbol{Q}_2 = \boldsymbol{Q}_2^{\mathrm{T}} \boldsymbol{B}_{22} \boldsymbol{P}, \tag{9.15b}$$

where vector $\boldsymbol{P}$ is given in Eq. (9.11), and $\boldsymbol{B}_{21}$ and $\boldsymbol{B}_{22}$ are binary matrices, having a size of 6×8, given as follows:

$$\boldsymbol{B}_{21} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{bmatrix},$$

$$\boldsymbol{B}_{22} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The location of the data point is determined by the following IF-ELSE construct:

$$\begin{array}{ll} \text{If } \Delta x > \Delta y, & p = p_1, \\ \text{else} & p = p_2. \end{array}$$

9.2.4 Pyramid interpolation

For pyramid interpolation, the cube is sliced into three pieces; each one takes a face as the pyramid base, having its corners connected to a vertex in the opposite side as the apex (see Fig. 9.6). A search is required to locate the point of interest. The equation has five terms and uses five vertices of the given pyramid to compute the value.

Equation (9.16) gives the vector-matrix form of the pyramid interpolation.

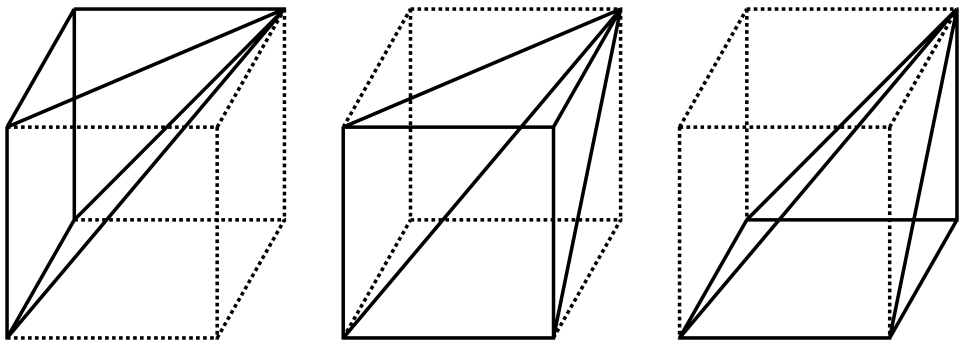

Figure 9.6 Pyramid interpolation.