Figure 3.6. shows a piecewise triangular contour approximation for the function $f(\mathbf{x}) = 1.2\left((x-10)^2 - (y-10)^2 + (z-10)^2\right)$ and the contour level $f(\mathbf{x}) = 60$. The trivariate function is evaluated on an equidistantly spaced rectilinear data grid of $21 \cdot 21 \cdot 21$ points in $[0, 20]^3$. All contour triangles are rendered using flat shading.

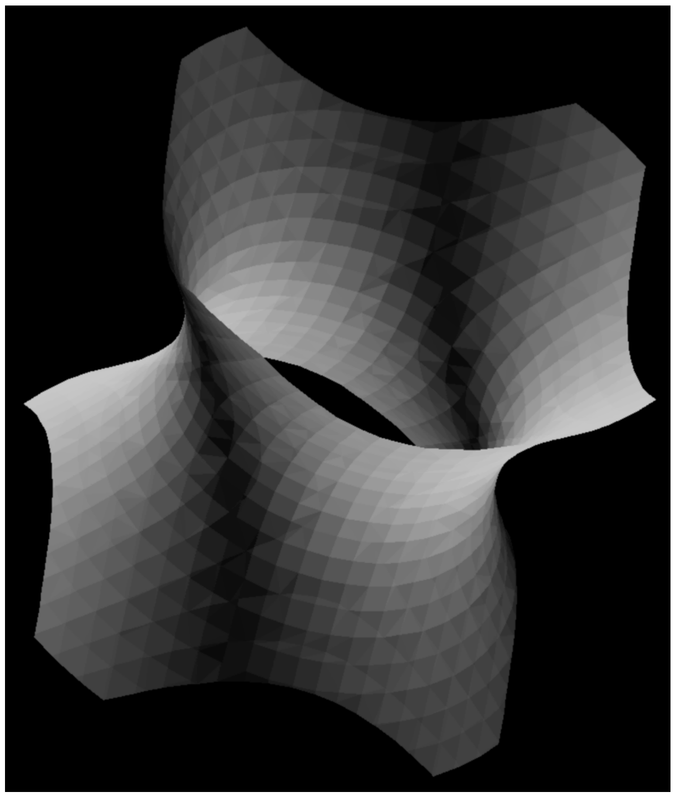

Fig. 3.6. Triangular approximation of contour level $f(x, y, z) = 60$ for $f(x, y, z) = 1.2\left((x-10)^2 - (y-10)^2 + (z-10)^2\right), \; x, y, z \in [0, 20]$.

Remark 3.13. At this stage of the triangulation process the quality of the triangulation within a single cell is not taken into account. As soon as one has obtained the whole set of triangles approximating the contour throughout all cells, *smoothness* criteria can be used to improve the triangulation.

3.3. Computing topological information for a piecewise triangular trivariate contour approximation

From a computational point of view, a rectilinear trivariate data set is investigated iteratively, cell by cell, to generate a contour approximation. Considering a single cell three tables are established, the first table storing all the contour points with their three-dimensional coordinates (ordered),

$$Y \;=\; \{\; (i, \mathbf{x}_i{}^T) = (i, x_i, y_i, z_i) \mid i = 0 ... n_C - 1 \;\},$$

a second table defining which vertices in the first table constitute line segments of closed polygons on the cell's faces (ordered),

$$E \;=\; \{\; (j, v_j^1, v_j^2) \mid j = 0 ... n_C - 1 \;\},$$

v_j^1 and v_j^2 being indices referring to Y, and a third table (derived from E) specifying which line segments (ordered) form the edges of closed polygons on the cell's faces,

$$P \;=\; \{\; (p, e_p^0, e_p^1, ..., e_p^{n_p - 1}) \mid p = 0 ... m - 1 \;\},$$

e_p^k, $k = 0 ... n_p - 1$, being indices referring to E.

The three tables Y, E, and P are for temporary use only. As soon as a triangulation for all the closed contour polygons in the set P has been computed, the contour points from the temporary table Y are copied into a permanent, global vertex table V (ordered),

$$V \;=\; \{\; (i, \mathbf{x}_i{}^T) = (i, x_i, y_i, z_i) \mid i = 0 ... n_v - 1 \;\}, \qquad (3.12.)$$

containing all contour points found throughout the whole data set. The triangles constructed in a single cell's interior are also added to a permanent, global table $\mathcal{T}$

defining the overall triangular contour approximation,

$$\mathcal{T} = \left\{ T_t = (t, v_t^1, v_t^2, v_t^3) \mid t = 0 ... n_t - 1 \right\}. \tag{3.13.}$$

Having computed the complete set $\mathcal{T}$ of all triangles approximating a certain contour, it is essential to derive topological information still "hidden" in the triangulation. Data reduction algorithms and surface generation schemes commonly require neighborhood information. An algorithm is given that computes the neighbors of each triangle considering the table $\mathcal{T}$ only. The contour approximation for $f(\mathbf{u}) = \alpha$ might be split into several non-connected parts, as mentioned before. Therefore, it is also necessary to know to which part a particular triangle belongs to if one wants to model the different contour parts separately.

Algorithm 3.1. determines the neighbors for each triangle, i.e., for a given triangle T_t the algorithm generates the indices of its maximal three neighbors in $\mathcal{T}$.

Algorithm 3.1. *Neighborhood*

Input: table $\mathcal{T}$ of triangles T_t, each given by its own index (referring to $\mathcal{T}$) and its three vertex indices (referring to V).
Output: number of neighbor triangles and their indices (referring to $\mathcal{T}$) for each triangle T_t in $\mathcal{T}$.

```
for i = 0 to n_t − 1
    (   cnt := 0;    /* number of neighbors */
        j := 0;
        while j < n_t and cnt < 3
            (   if  i ≠ j and T_i and T_j are neighbors
                    then
                    (   cnt := cnt + 1;
                        cnt − th neighbor of T_i := j;
                    )
                j := j + 1;
            )
        number of neighbors of T_i := cnt;
    )
```

This algorithm is of order $O({n_t}^2)$ with respect to the total number of triangles. Its performance can be improved by storing the index-triple (i, j, k) of the left-front-lower corner of the cell $C_{\mathbf{i}}$ for each triangle T_t when T_t lies in $C_{\mathbf{i}}$'s interior. Thus, the search for the neighbors of a particular triangle inside a cell $C_{\mathbf{i}}$ can be restricted to the cell $C_{\mathbf{i}}$ itself and its six neighbor cells, $C_{i-1,j,k}$, $C_{i+1,j,k}$, $C_{i,j-1,k}$, $C_{i,j+1,k}$, $C_{i,j,k-1}$, and $C_{i,j,k+1}$. Hence, order $O(n_t)$ can be achieved.

Two triangles in $\mathcal{T}$ belong to the same part of the contour approximation if there is a path from one triangle to the other one such that all triangles constituting this path determine pairs of neighbor triangles (Definition 3.9.). To effectively compute the part (index) a particular triangle belongs to, the following algorithm is used:

At the beginning all triangles share the not-yet-assigned part index -1. The first triangle T_0 in $\mathcal{T}$ is assigned to the first part with index 0. For all the other triangles T_t, one checks among its neighbors, whether at least one of them has already been assigned to some part. If none of the neighbors has been assigned yet, a new part (index) is introduced for triangle T_t; if at least one among its neighbors already belongs to a certain part, the minimal part (index) among all T_t's assigned neighbors is selected as T_t's part (index). If the neighbors of T_t which are assigned to a part do not all agree with this minimal part (index), all triangles in $\mathcal{T}$ assigned to such a different part (index) must now also be assigned to the selected minimal part (index).

Algorithm 3.2. *Part of contour*

Input: table $\mathcal{T}$ of triangles T_t (including the neighborhood information).
Output: part index for each triangle which determines the part of the contour approximation it belongs to.

```
/* initially all triangles have a not-yet-assigned part index −1 */
p := 0;    /* first valid part index */
for t = 0 to n_t − 1
    (   determine minimal part index min among
        all the part indices of T_t's neighbors;
        if min = −1
            then
            (   part index for triangle T_t := p;
                p := p + 1;    /* another part of contour introduced */
            )
            else
            (   part index for triangle T_t := min;
                if there is 1 [are 2] valid part index p̄ [indices p̄, p̿] ≠ min
                   among T_t's neighbors
                    then
                    (   part index for all triangles assigned to p̄ [p̄, p̿] := min;
                                                /* connecting triangle has been found */
                        p := p − 1 [2];        /* reduce number of parts appropriately */
                    )
            )
    )
```

The case in brackets ("[ ]") in algorithm 3.2. indicates the situation when a triangle has three neighbors with valid part indices ($\neq -1$) which are all different from each other. At the end, each triangle is assigned to a certain part of the triangular contour approximation. Obviously, algorithm 3.2. is of order $O(n_t)$.

The principal of selecting a part (index) for a triangle is shown in Figure 3.7. The minimal part (index) among T_5's neighbors is 1. Therefore, T_5 as well as all triangles belonging to part 2 are assigned to part 1.

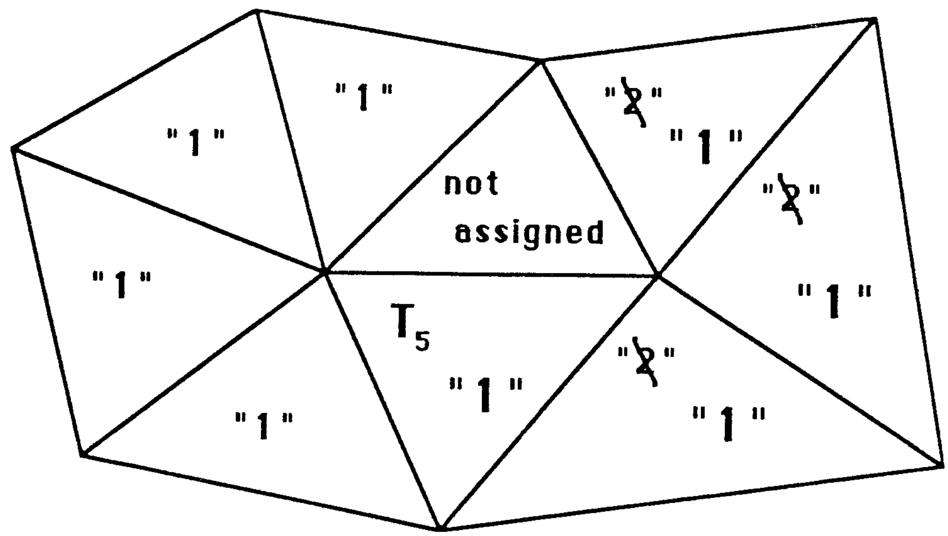

Fig. 3.7. Assigning the part index to a triangle in a contour triangulation.

Remark 3.14. It might be worth considering some methods for improving the triangulation of the contour approximation at this point. Knowing the neighbors for each triangle in $\mathcal{T}$, the "$max - min$" or "$min - max$" angle criteria could be used to iteratively eliminate triangles with small angles ([Cline & Renka '84], [Lawson '77]). Common algorithms swap diagonals of quadrilaterals (given by two neighbor triangles) in order to enhance the angle configuration.

Considering the fact that the triangulation to be improved is not a planar triangulation, different optimization criteria might be appropriate. An algorithm to increase the smoothness of a two-dimensional triangulation in three-dimensional space is described in [Choi et al. '88]. There, the (local) quality measure of a triangulation is the angle between normal vectors of neighbor triangles. The objective is to minimize these angles by swapping diagonals of quadrilaterals.

3.4. Gradient approximation for rectilinear data

It is not the purpose of this chapter to discuss or derive new ways for gradient/normal approximation in full detail. Rather, the principal problem is stated, and general solutions are reviewed in a more survey fashion. Gradient/normal information is necessary for curvature approximation, data reduction, and surface generation, discussed in the following chapters. The quality of these subsequent modeling steps is very much dependent on the quality of the gradient/normal estimation.

In order to construct trivariate functions approximating trivariate data sets given in either rectilinear or scattered form or to generate smooth surfaces fitting the different parts of a given two-dimensional contour triangulation of some trivariate function, gradients and normals must be estimated if positional information is available only. In the case of normal vector approximation, outward unit normal vectors are estimated defining oriented tangent planes for all contour points in the contour triangulation.

General information about the construction of bivariate/trivariate functions approximating scattered data can be found in [Alfeld '89], [Barnhill '85], [Foley '87], [Franke & Nielson '91], [Hoschek & Lasser '89], [Nielson & Franke '83], and [Worsey & Farin '87]. In [Stead '84] different schemes are compared for estimating gradients. A local operator generating normal vector estimates for rectilinear a data set in an “optimal” sense is described in [Zucker & Hummel ’81].

With respect to the strategy pursued here, it is of greater interest to approximate normal vectors since it is a two-dimensional contour triangulation which will be modelled later on. One can choose among two possibilities how to obtain normal vector estimates. The first possibility consists of constructing a trivariate function locally approximating the rectilinear/scattered data, hence also defining gradients at the points in a contour triangulation. The second possibility rather derives normal vector estimates from a contour triangulation directly.

If the first possibility is chosen, it is important to realize that the gradients at contour points also determine normal vectors:

Theorem 3.7. *Let $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ be a C^1 function and $\nabla f(\mathbf{x}_0)$ be non-vanishing. Let $\mathbf{x}_0$ be a point in $C_f(\alpha)$ and $\mathbf{v}$ any tangent vector to $C_f(\alpha)$ at $\mathbf{x}_0$; then $\nabla f(\mathbf{x}_0)$ is normal to the contour $C_f(\alpha)$ at $\mathbf{x}_0$,*

$$\nabla f(\mathbf{x}_0)\ \mathbf{v} \ = \ 0. \tag{3.14.}$$

Proof. Let $\mathbf{c}(t) \subset C_f(\alpha)$ be a curve on the contour such that $\mathbf{c}(0) = \mathbf{x}_0$ and $\dot{\mathbf{c}}(0) = \mathbf{v}$; considering the fact that $f(\mathbf{c}(t)) = \alpha$, and using the chain rule yields

$$0 = \tfrac{d}{dt} f(\mathbf{c}(t))\big|_{t=0} = \nabla f(\mathbf{x}_0)\ \dot{\mathbf{c}}(0) = \nabla f(\mathbf{x}_0)\ \mathbf{v}.$$

q.e.d.

Definition 3.14. Let $f : I\!R^3 \to I\!R$ be a C^1 function and $\bigtriangledown f(\mathbf{x}_0)$ be non-vanishing. The **outward unit normal vector** to $C_f(\alpha)$ at $\mathbf{x}_0$ is the vector

$$\mathbf{n}_0 = \left(\bigtriangledown f(\mathbf{x}_0) \; / \; || \bigtriangledown f(\mathbf{x}_0)|| \; \right)^T, \tag{3.15.}$$

where $||(x, y, z)|| = \sqrt{(x^2 + y^2 + z^2)}$. The **oriented tangent plane** at $\mathbf{x}_0$ is given by the set of all points $\mathbf{x} = (x, y, z)^T \in I\!R^3$ satisfying the equation

$$\bigtriangledown f(\mathbf{x}_0) \; (x - x_0, y - y_0, z - z_0)^T \;\; = \;\; 0. \tag{3.16.}$$

Choosing the alternative of constructing an approximating function in a neighborhood around a contour point, one must be aware to keep the original rectilinear/scattered trivariate data set.

A method for normal estimation which has proven to yield rather good results is discussed in [Zucker & Hummel '81]. Summing up the approach, Zucker reduces normal estimation to a minimization problem. The expression minimized is

$$|| \; f(\mathbf{x}) - E_{\{a,b,c\}}(\mathbf{x}) \; ||.$$

Here, $f(\mathbf{x})$ is a known trivariate function defined over the unit ball B ($B = \left\{ \mathbf{x} | x^2 + y^2 + z^2 \le 1 \right\}$) and $E_{\{a,b,c\}}(\mathbf{x})$ is the function

$$E_{\{a,b,c\}}(\mathbf{x}) \;\; = \;\; \begin{cases} +1, & \text{if } ax + by + cz \ge 0; \\ -1, & \text{otherwise.} \end{cases}$$

The coefficients a, b, and c are the unknowns defining an oriented plane through the origin with normal vector $\mathbf{n} = (a, b, c)^T$ used as the normal estimate. The norm is the L_2-norm,

$$||f(\mathbf{x})|| \ = \ \sqrt{\int\int\int_B \ f^2(\mathbf{x}) \ dxdydz}.$$

The result is then applied to the discrete case, given a rectilinear data set with equal, equidistant spacing in all three spatial directions, $\Delta = \triangle x_i = \triangle y_j = \triangle z_k$, all i, j, k. A simple, local operator is derived in order to optimally approximate the outward normal vector at a grid point $\mathbf{x_i}$ fixed as the origin of a local coordinate system for the discrete minimization problem.

Theorem 3.8. *Considering solely the* 27 *neighbor data* $(\mathbf{x}_{i+r,j+s,k+t}, f_{i+r,j+s,k+t})$, $r, s, t \in \{-1, 0, 1\}$, *nearest to* $(\mathbf{x_i}^T, f_\mathbf{i})$ *in an equally, equidistantly spaced rectilinear data set, a normal vector* $\mathbf{n_i} = (nx_\mathbf{i}, ny_\mathbf{i}, nz_\mathbf{i})^T$ *at* $\mathbf{x_i}$ *is optimally approximated by*

$$nx_\mathbf{i} = \sum_{r\in\{-1,1\},\ s,t\in\{-1,0,1\}} r \ c_{i+r,j+s,k+t} \ f_{i+r,j+s,k+t},$$

$$ny_\mathbf{i} = \sum_{s\in\{-1,1\},\ r,t\in\{-1,0,1\}} s \ c_{i+r,j+s,k+t} \ f_{i+r,j+s,k+t},$$

$$nz_\mathbf{i} = \sum_{t\in\{-1,1\},\ r,s\in\{-1,0,1\}} t \ c_{i+r,j+s,k+t} \ f_{i+r,j+s,k+t}, \qquad (3.17.)$$

where $c_{i+r,j+s,k+t} = \frac{\sqrt{|r|+|s|+|t|}}{|r|+|s|+|t|}$, *subject to minimizing* $||\ f(\mathbf{x}) - E_{\{a,b,c\}}(\mathbf{x})\ ||$ *in this particular discrete case.*

Proof. See [Zucker & Hummel '81], pages 326 and 329-331.

Normalizing $\mathbf{n_i}$ yields the desired outward unit normal vector at the point $\mathbf{x_i}$. The principle for computing the x-coordinate of $\mathbf{n_i}$ is sketched in Figure 3.8. The involved function values and their weights for the normal vector approximation at a grid point $\mathbf{x_i}$ are shown using Zucker's "$3 \cdot 3 \cdot 3$" operator.

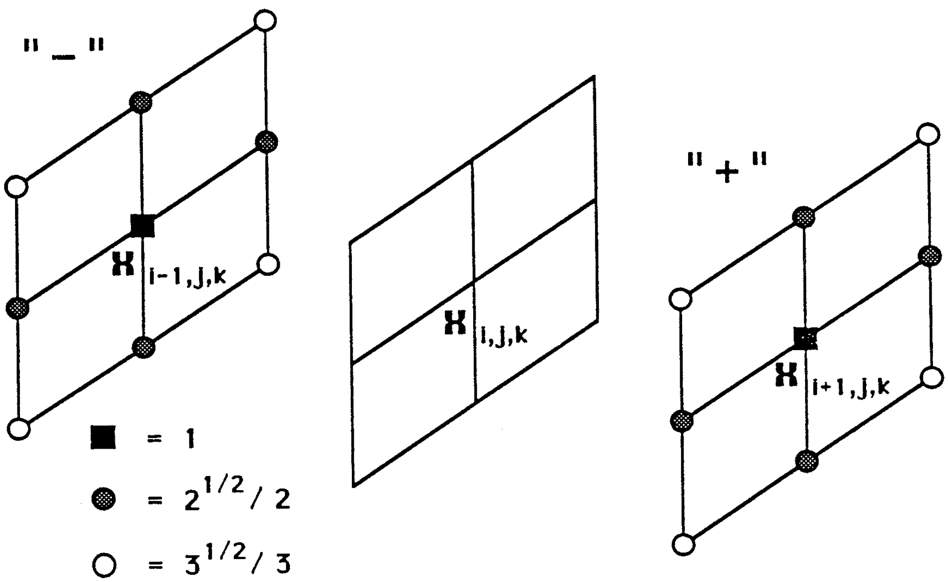

Fig. 3.8. The 18 function values and their weights needed for approximating the $x-$coordinate for a normal using Zucker's operator.

The normal vector for a contour point $\mathbf{x}_i \in V$ (formula (3.12.)) on an edge of a cell $C_{\mathbf{i}}$ is approximated by linear interpolation of the estimated normal vectors at the rectilinear grid points defining that edge, a normal vector in $C_{\mathbf{i}}$'s interior is approximated by trilinear interpolation of the eight normal vectors estimated for the cell's corner points. For the case that a contour point $\mathbf{x}_i$ is along the cell edge $\overline{\mathbf{ab}}$ one chooses the outward normal vector $\mathbf{n}_i$ at this point to be

$$\mathbf{n}_i = \mathbf{n}_i\big|_{\mathbf{x}=\mathbf{x}_i} = (1-t)\mathbf{n}\big|_{\mathbf{x}=\mathbf{a}} + t\mathbf{n}\big|_{\mathbf{x}=\mathbf{b}}, \quad t = \frac{||\mathbf{x}_i - \mathbf{a}||}{||\mathbf{b} - \mathbf{a}||}, \tag{3.18.}$$

where $t \in [0,1]$ and $||(x,y,z)^T||$ is the Euclidean norm. In the case that $\mathbf{x}_i$ is in the interior of the cell $C_{\mathbf{i}}$ one chooses

$$\mathbf{n}_i = \mathbf{n}_i\Big|_{\mathbf{x}=\mathbf{x}_i} = \sum_{t=0}^{1}\sum_{s=0}^{1}\sum_{r=0}^{1} \mathbf{n}_{\mathbf{r}}\ B_r^1(u)B_s^1(v)B_t^1(w),$$

$$u = \frac{|x_i - \mathbf{x}_{\mathbf{i}}^x|}{\Delta}, \qquad v = \frac{|y_i - \mathbf{x}_{\mathbf{i}}^y|}{\Delta}, \qquad w = \frac{|z_i - \mathbf{x}_{\mathbf{i}}^z|}{\Delta}, \tag{3.19.}$$

where $\mathbf{n}_{\mathbf{r}} = \mathbf{n}_{i+r,j+s,k+t}$, $r, s, t \in \{0, 1\}$, are the eight outward normal vector estimates at $C_{\mathbf{i}}$'s corner points, where an equally, equidistantly spaced rectilinear point set is assumed, $\Delta = \triangle x_i = \triangle y_j = \triangle z_k$, all i, j, k, $\mathbf{x}_i = (x_i, y_i, z_i)^T$ is the contour point, $\mathbf{x}_{\mathbf{i}} = (\mathbf{x}_{\mathbf{i}}^x, \mathbf{x}_{\mathbf{i}}^y, \mathbf{x}_{\mathbf{i}}^z)^T$ is the left-front-lower corner point of the cell $C_{\mathbf{i}}$, $B_l^1(t) = (1-t)^{1-l}t^l$, $t \in [0, 1]$, $l = 0, 1$, are the Bernstein polynomials of degree one and $u, v, w \in [0, 1]$.

Normalizing the estimates $\mathbf{n}_i$ finally determines the set of (ordered) outward unit normal vectors at each contour point,

$$N \ = \ \left\{ \ (i, {\mathbf{n}_i}^T) = (i, nx_i, ny_i, nz_i) \ | \ ||\mathbf{n}_i|| = 1, \ i = 0 \ldots n_v - 1 \ \right\}. \tag{3.20.}$$

Figure 3.9. is obtained from the same data set as the one used for Figure 2.1. CAT scan density measurements are given as an equally, equidistantly spaced rectilinear data set of $68 \cdot 64 \cdot 64$ points with associated density values $f_{\mathbf{i}} \in [0, 255]$. The contour level approximated is $f(\mathbf{x}) = 12.5$. The triangular approximation consists of almost $30,000$ contour points and $60,000$ triangles. Outward unit normal vectors for each contour point are estimated using Zucker's approach and required for this Gouraud-shaded rendering of the contour approximation.

Fig. 3.9. Human skull obtained from a rectilinear CAT scan data set, $68 \cdot 64 \cdot 64$ points, $f_{i,j,k} \in [0, 255]$, approximation for $f(x, y, z) = 12.5$.

Remark 3.15. The problem of estimating normal vectors for points in a two-dimensional triangulation in three-dimensional space using the triangulation alone has hardly been investigated. A possible good solution to the problem might be the following approach.

It is well known in differential geometry that a surface in three-dimensional space can locally be approximated by the graph of a differentiable bivariate function. In the case of a two-dimensional triangulation in three-dimensional space, one usually considers the points $\mathbf{y}_j \in V$, $j = 1...m$, (equation (3.12.)) determining an edge with a particular point $\mathbf{x}_i = \mathbf{y}_0 \in V$ in the triangulation as a localization of the triangulation,

$$Y \;=\; \{\ \mathbf{x}_i = \mathbf{y}_0\ \} \;\cup\; \{\ \mathbf{y}_j \mid \overline{\mathbf{x}_i\mathbf{y}_j} \text{ is an edge in the triangulation, } j = 1...m\ \}.$$

By introducing a local, right-handed coordinate system S, defined by $\mathbf{x}_i$ as origin and three mutually perpendicular unit vectors $\mathbf{d}_1$, $\mathbf{d}_2$, and $\mathbf{n}$, a plane P is defined by the origin and the first two unit vectors. The points in Y can now be projected into the plane P and their distances d_j, $j = 0...m$, from P be calculated. Assuming that all projected points in P are different, a bivariate function, e.g., a second degree polynomial, can be constructed using the least squares method to approximate the $m+1$ function values $f_j = d_j$, $j = 0...m$, considering the constraints

$$f(\overline{x}_j, \overline{y}_j) \;=\; \sum_{r+s+t\leq 2,\ r,s,t\geq 0} c_{r,s,t}\ \overline{x}_j{}^r \overline{y}_j{}^s \overline{z}_j{}^t \;=\; d_j, \qquad j = 0...m,$$

where $\overline{x}$ and $\overline{y}$ are the coordinates of a projected point in P with respect to the two-dimensional coordinates system defined by $\mathbf{x}_i$ (origin) and the unit vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, and d_j is interpreted as the function value of f at the corresponding projected point in the direction of $\mathbf{n}$.

Assuming that the linear system of equations for the least squares solution does not imply a vanishing determinant, the unknown coefficients $c_{r,s,t}$ determine a residual vector $\mathbf{r}$ which can be measured using the L_2-norm,

$$||\ \mathbf{r}\ || \;=\; \sqrt{\sum_{j=0}^{m} \left(\ f(\overline{x}_j, \overline{y}_j) - d_j\ \right)^2}.$$

This expression really depends on the choice of the orientation of the system S. Changing the orientation of $\mathbf{n}$ (determining the f-axis) appropriately, might lead to a minimization of $\|\, \mathbf{r} \,\|$. Choosing the normal of the graph of f at $\mathbf{x}_i$ based on such an "optimally oriented" coordinate system S presumably is a good estimation for a normal vector $\mathbf{n}_i$. The direction for the normal vector is still ambiguous ($\mathbf{n}_i$ or $-\mathbf{n}_i$).

Remark 3.16. It is also worth considering a contour approximation for a finite data set G in either scattered or rectilinear form obtained by computing the length of the gradient estimates for each point in an original trivariate data set,

$$G \;=\; \left\{\; (\mathbf{x}_i{}^T, g_i) = (\mathbf{x}_i{}^T, \|\bigtriangledown f(\mathbf{x}_i)\|) \mid \mathbf{x}_i \in \mathbb{R}^3,\;\, g_i \in \mathbb{R},\;\, i = 0...n \;\right\}.$$

This is a common approach in computer vision for edge detection. Boundaries of objects in an image (brightness or density functions) are characterized by large gradients.