the metric coefficients and the connectivity pattern of the grid is sufficient to compute a distance map on the parametric surface. This is a very handy property in some applications, for example, where surface acquisition techniques do not provide the surface itself, but rather its gradients [77].

As a final remark on this method, recall that the fast marching update scheme was valid for acute triangles only. On parametric surfaces, this condition can be stated as $e_{12} > 0$. In case where a triangle is obtuse ($e_{12} < 0$), Spira and R. K. proposed to split it into two acute triangles by adding a connection to another "virtual" non-adjacent neighbor on the grid [367]. The virtual connection is selected as the one producing two acute angles and having the shortest length in the parameterization domain. Adding such virtual connections to grid points can be done in $\mathcal{O}(N)$ at the grid initialization stage.

4.5 Marching even faster

One of the core components of all Dijkstra-type distance computation algorithms, including fast marching methods, is the heap capable of extracting the vertex with the smallest value of d in logarithmic time.[5] Selecting the next vertex to be updated according to minimum distance ensures that the grid points are visited in an order simulating the propagation of a wavefront. This fact conceals one of the major drawbacks of fast marching: the grid traversal order depends on the data and cannot be known *a priori*. Moreover, this order is not well-structured, making problematic an efficient use of memory systems. Such a strident lack of structure calls for searching alternative traversal orders.

In his classic paper [121], Per-Erik Danielsson studied the computation of distance maps from arbitrary sources in the plane. He observed that as the Euclidean geodesics are straight lines, the characteristics of the eikonal equation fall into one of the four plane quadrants and can be therefore covered by a sequence of four directed scans, where in each scan a grid point is updated from the previously updated "causal" neighbors in the scan order (Figure 4.8). Each point is updated four times, implying linear complexity in the grid size, $\mathcal{O}(N)$. Because the order in which the grid points are visited is known in advance and is independent of the data, Danielsson's raster scan algorithm is characterized by regular access to the memory and can benefit significantly from the *caching* mechanism supported by most modern processor architectures.

Equipping the Danielsson's raster scan algorithm with the fast marching update scheme gives rise to a family of distance computation algorithms, where the Dijkstra-type wavefront propagation is replaced by sweeping the grid in four alternating directions. One of the earliest mentioning of this technique dates back to Dupuis and Oliensis' studies on shape from shading from 1994 [142]. A raster-scan algorithm for solving the eikonal equation

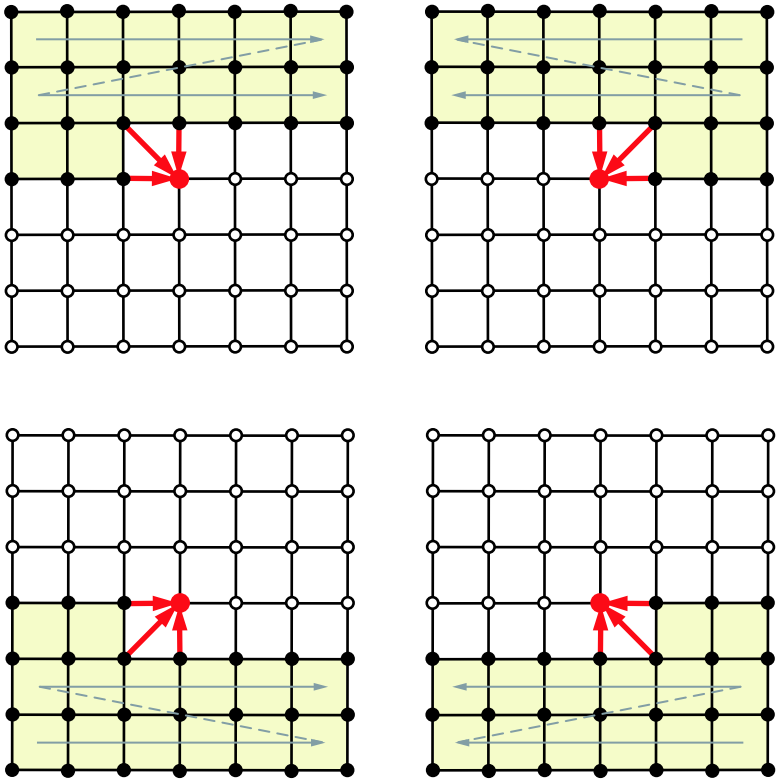

Figure 4.8. Raster scan grid traversal used in Danielsson's algorithm. Four directed raster scans are sufficient to cover all the geodesic directions in the plane.

on weighted Euclidean domains was also studied by Zhao, who introduced the name *fast sweeping* [411].

Because the raster scan algorithm operates on a regular Cartesian grid, it is an attractive alternative to fast marching for computation of distance maps on parametric surfaces [61, 60]. However, it is important to realize that unlike the Euclidean case where the geodesics are straight lines and thus can be covered by four directed raster scans, on a general surface geodesics are usually curved. This implies that four raster scans may cover only a part of an eikonal equation characteristic; in order to obtain a consistent distance map, the scans have to be repeated several times. This fact is visualized in Figure 4.9, where the raster scan algorithm is used to compute the distance map in a "maze" with complicated spiral-shaped geodesics. Six iterations of alternating raster scans are required in order to cover it completely. We can therefore conclude that the complexity of the raster scan algorithm is *data-dependent*. Nevertheless, it appears that the maximum number of iterations required to produce a consistent distance map on a parametric surface

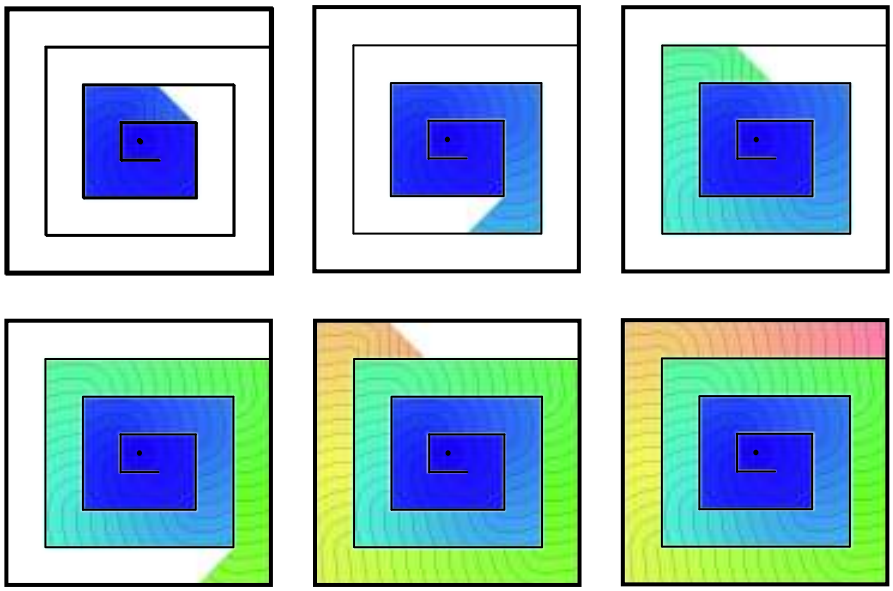

Figure 4.9. Distance map computation on the "maze" surface using the raster scan algorithm after an increasing number of iterations (left-to-right, top-down). Note that a single repetition of the four directed raster scans is insufficient to cover the complicated spiral-shaped characteristic.

can be bounded assuming some regularity of the metric coefficients and the second-order derivatives of the parameterization [61, 60]. The bound does not depend on the grid size, leaving the theoretical complexity of the algorithm $\mathcal{O}(N)$. Yet, the number of iterations depends on the properties of both the surface itself and its parameterization. This means that some parameterizations of the same surface may be less advantageous than others from the point of view of the raster scan algorithm. For example, with the trivial parameterization of the plane $(x^1, x^2, x^3) = (u^1, u^2, 0)$, the geodesics are straight lines requiring one iteration. Using a more bizarre parameterization, say, $(x^1, x^2, x^3) = (u^1 \cos u^2, u^1 \sin u^2, 0)$, several iterations are needed.

4.6 Parallel distance computation

In addition to the better structured access to memory, the raster scan algorithm has another important advantage over traditional fast marching: unlike fast marching methods, which are inherently *sequential*, raster scan can benefit from *parallelization*. To demonstrate the parallelism, let us consider for example the right-down scan (upper left in Figure 4.8) starting from the top leftmost grid point d_{11} (we denote by d_{ij} the value of the distance map d