# Iso-Surface Extraction in 4D with Applications Related to Scale Space

Márta Fidrich*

Project Epidaure, INRIA, B.P. 93, 06902 Sophia-Antipolis Cedex, FRANCE

** Research Group on Artificial Intelligence, Hungarian Academy of Sciences
H-6720 Szeged, Aradi vértanúk tere 1, HUNGARY

**Abstract.** We present a method for extracting iso-surfaces and their intersections in 4D. Our work is a significant extension of the 3D Marching Lines algorithm with new orientation and implementation considerations. As a practical tool, it can be applied to track efficiently space curves, defined by differential invariants, across increasing scale.

**keywords:** Iso-intensity-surface extraction, Marching Lines algorithm, Topology, Scale space, Medical image processing

## 1 Introduction

Due to the intrinsic nature of intensity in medical images, iso-surface extraction is a generally used method in 3D medical image processing [8]. As 4D images are becoming increasingly common, the challenge of the extraction of 4D hyper-iso-surfaces and their intersections has emerged. Practical applications consist of processing temporal 3D images like beating heart sequences or tracking topological changes of 3D characteristic lines in scale space. The latter is necessitated for high-level computerized tasks, such as inter-patient registration or building reliable anatomical atlases. In fact, our motivation for designing a 4D algorithm comes from multiscale analysis of 3D images, which is a natural continuation of our previous work [5]. We have recently found a paper which deals with a similar subject [6]; however it fails to consider the correctness of its algorithm.

This work is inspired by the Marching Lines algorithm [15], which is briefly summarized in the first section. In the next section we describe its extension to the 4D case; to do so, we have to revise our way of thinking about orientation. Then we show how this method can be applied to follow characteristic space curves at multiple scales. Finally, we discuss some implementation details.

## 2 3D overview

There are several techniques to extract iso-surfaces in 3D, classical examples are [17, 11]. To compute the intersection curves of two iso-surfaces, the Marching Lines algorithm (shortened as *MLA*) is an effective tool [15]. It uses a beveled-form [8] representation, i.e. voxels are viewed as 3D grid points, 8 adjacent voxels form a

* Supported by the grant MESR

** On leave during 1996

cubic *8-cell*. By means of interface and orientation conventions it resolves singular situations and ensures good topological properties. The reconstructed discrete surface is oriented, complete i.e. without holes (except at the boundary of the 3D grid) and contains no self-intersections. Also, each intersection curve is oriented (linked list of 3D points where the order of points reflects the orientation) and closed (if the curve is contained within the 3D grid). Since the MLA is a starting point of our 4D extension, we now summarize the conventions used and the main steps of the algorithm; for proof of correctness see [15].

### 2.1 Extraction of iso-surfaces

**Definition 1 (iso-surface as interface).** The iso-surface is the interface between the regions of the image $f$: $f \geq I$ (the inside) and $f < I$ (the outside), where $I$ is an intensity constant.

According to this convention, each voxel is labeled either to be positive ($f \geq I$) or negative ($f < I$); to follow the explication see Fig. 1. If at least one of the voxel label of the 8-cell differs from the others, the 8-cell will contain some iso-surface. The points at which the iso-surface intersects the edges of the cube, called *vertices of iso-patches* (2 or 4 par face), are obtained by linear interpolation along the cell edges. To define adjacency-relations between vertex pairs on a *4-side*, the next conventions are used, so that each face of a cube is oriented according to the left-hand convention if it is seen from outside.

**Definition 2 (left-hand orientation of a 4-side).** When following a closed planar curve its inside should always be on the left.

**Definition 3 (outside orientation of a 8-cell).** The surface of a cubic cell is oriented toward the outside of the cell.

On each 4-side the vertices of iso-patches are labeled with the label of the endpoint of the left-hand oriented edges. The vertices are formed to iso-segments from points labeled $-$ to points labeled $+$. The ambiguous case of alternately labeled voxels – giving 4 vertices – on a 4-side is resolved by the mean-value rule.

**Definition 4 (mean-value rule).** Assuming that 4 interpolated vertices are generated: If the mean-value of a 4-side is positive ($f \geq I$), two segments are formed to give one positive iso-patch component containing the center point; otherwise to give two components without the center point.

The last three conventions ensure that the vertices of a cubic 8-cell (giving directed iso-segments on each 4-side) can be organized into oriented iso-patch cycles.

### 2.2 Extraction of intersection curves

To get the intersection line segments, we consider a regular 3D grid whose voxels are labeled according to two images $f$ and $g$. We first compute iso-patch cycles with the intensity values of $f$, then we proceed to get *bi-iso-segments* (similarly as we did on a 4-side), this time using the iso-value $J$ corresponding to the second iso-surface $g$ at the iso-patch vertices. To avoid the ambiguity at tangent intersection of surfaces, the order of iso-surfaces is preset:

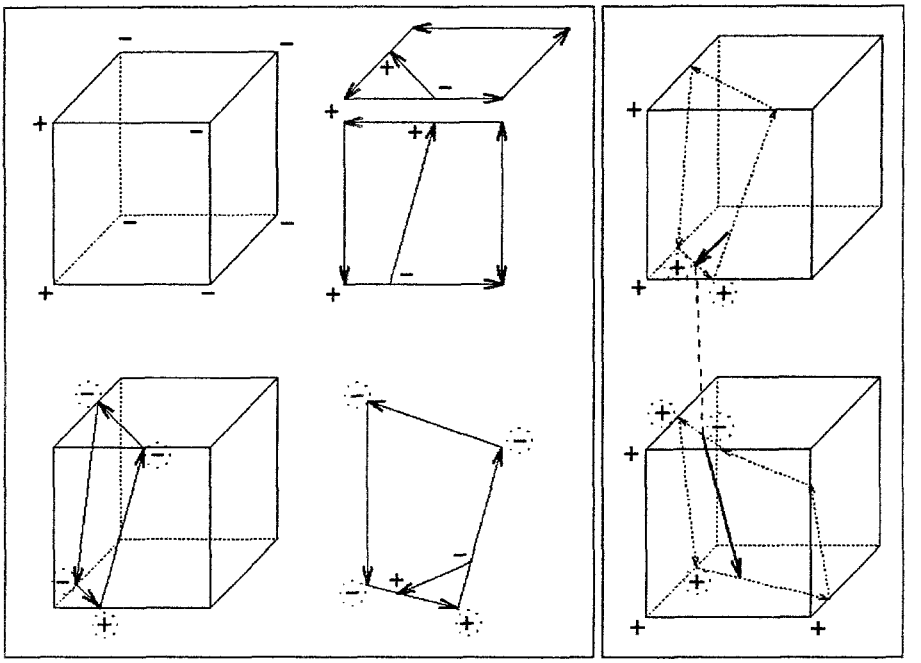

**Fig. 1.** Explication of the 3D Marching Lines algorithm.
*left: (1)* Voxels are labeled according to $(f \geq I)$ or $(f < I)$ given by the first image $f$.
*(2)* Faces are oriented using the left hand convention; vertices of the iso-patch are computed with linear interpolation and labeled with the label of the endpoint of the left-hand directed edges. *(3,4)* The directed bi-iso-segment on the iso-patch is similarly obtained with interpolation, this time using the intensity values of the second image $g$ at the iso-patch vertices. (These labels are in dotted circles.)
*right:* The iso-patches have opposite orientation in adjacent cubic cells, the bi-iso-segment can be easily followed using its direction.

**Definition 5 (intersection order of iso-surface).** Let $S$ $(f \geq I, f < I)$ and $T$ $(g \geq J, g < J)$ be two iso-surfaces. $C(f, g)$ is an iso-contour on $S$ defined as the interface between the regions of $T$: $g \geq J$ and $g < J$. $C(g, f)$ is similarly defined; the difference between the curves is how they bypass the tangency region.

An important application is to extract feature lines e.g. parabolic lines, lines of extrema of the principal curvatures... Here the values of $g$ are the corresponding differential expressions with iso-constant $J = 0$, while $(f, I)$ defines a usual iso-surface. We can also compute the intersection of three iso-surfaces as well [14].

Again, there is an ambiguity of forming bi-iso-segments in case of more than 2 bi-interpolated points. This has no matter for the 3D case; however in 4D it should be raised in a deterministic way, otherwise holes can be created. We propose to connect the consecutive points (this ensures that bi-iso-segments do not intersect) using the mean-value rule, which is to be applied on the bi-interpolated points.

## 3 4D extension

The good topological properties of the MLA are ensured with the help of interface *(1,4,5)* and orientation *(2,3)* conventions. If we want to extend it, we have to find a dimension-independent generalization of the orientation conventions, while the others can be used as they are.

### 3.1 Orientation by induction

A discrete nD image is considered as a regular nD grid composed of *hypercubes* i.e. $2^n$*-cells*, which is formed by $2^n$ adjacent voxels. Both the hypercube and its orientation is defined inductively (see Fig. 2):

**Definition 6 (hypercube).**

**0D** point
**1D** unit line-segment
**nD** (n-1)D hypercube is translated along the $n$th mutually perpendicular direction with a unit vector tracing out an nD hypercube.

**Definition 7 (orientation of a hypercube).**

**0D** without orientation
**1D** arbitrary vector direction (to be fixed)
**nD** The orientation of two (n-1)D hypercube, having a common (intersected) (n-2)D hypercube, should be opposite in the given nD hypercube.
Which is equivalent to:
The intersection of two adjacent nD hypercube, i.e. a (n-1)D hypercube should have opposite orientation in the two nD hypercube.

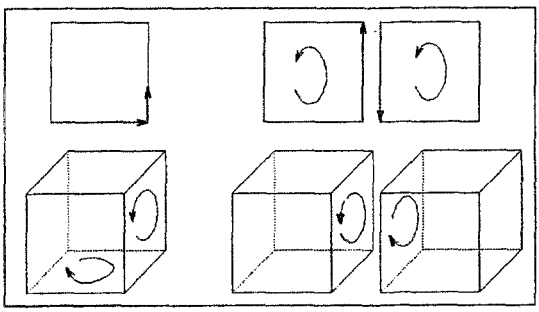

**Fig. 2.** *top:* Orientation in 2D: There is one incoming and one outgoing edge at each voxel *(left).* $\Longleftrightarrow$ The same edge has opposite directions in adjacent squares *(right).*
*bottom:* Orientation in 3D: The orientation of faces having a common edge is opposite *(left).* $\Longleftrightarrow$ The same face has opposite orientations in adjacent cubes *(right).*

Once the direction of any one line-segment is fixed, then the orientation of hypercubes in an nD grid is determined. To get a coherent generalization of the orientation

conventions used in the MLA, we complete this description with the following convention (see also Fig. 3):

**Definition 8 (fixing the orientation).** We direct the first line segment so as to regain the left-hand orientation for squares (and consequently the outside orientation for cubes).

Not only can these definitions be defined inductively, but also the structure of the nD generalized algorithm can be recursively given. In the sequel, we restrict ourselves to the 4D case, though the statements are valid for any dimension $n$.

### 3.2 Reducing the complexity

It is convenient to number the voxels of a hypercube in such a way that adjacent voxels have labels whose binary codes differ in 1 bit; see Fig. 3 for an example. The

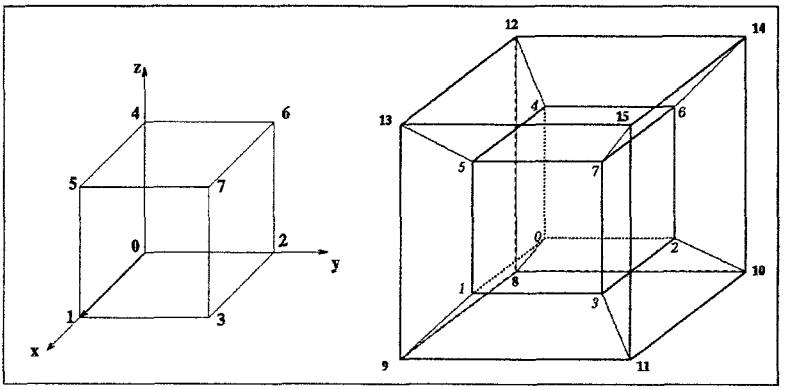


**Fig. 3.** Voxels of a 3D cube *(left)* and a 4D hypercube *(right)* are numbered according to 1-distance binary codes which are relative to a standard origin in the cell. Also, the direction of the **01** edge is fixed to define the orientation coherently with the left-hand / outside conventions.

hyper-faces (4-side for 3D, 8-cell for 4D case) can be obtained from the equation of the corresponding hyper-plane. The following notations are employed:

| 3D cube | 0 | 1 |
|---|---|---|
| z | 0 | 5 |
| y | 1 | 4 |
| x | 2 | 3 |

| 4D hypercube | 0 | 1 |
|---|---|---|
| t | 0 | 7 |
| z | 1 | 6 |
| y | 2 | 5 |
| x | 3 | 4 |

which is to be read as e.g.: face numbered by 0 (5) of a 3D cube is given by equation $z = 0\,(1)$. Naturally, hyper-faces of an nD hypercube are equivalent to (n-1)D hypercubes; we call the one coded by 0 as the *base hyper-face*. The order of vertices in a hyper-face specified by the equation of the corresponding hyper-plane does not

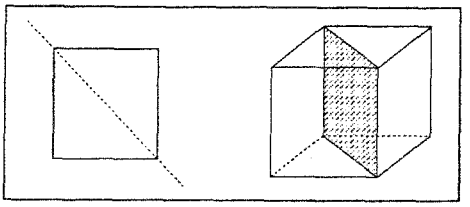

**Fig. 4.** Reflection axes of a 2D square *(left)* and a 3D cube *(right)*.

necessarily indicates the correct orientation. That is why the hyper-faces coded by odd numbers should be reflected as Fig. 4 shows, which establishes an *orientation-preserving bijection* between the base and the other hyper-faces. As a consequence, we can trace back any hyper-face of an nD hypercube to a base (n-1)D hypercube in an orientation-preserving way (for details see [3]).

Taking all these into account, *hyper-iso-patch* cycles and *hyper-bi-iso-segments* in a 16-cell can be computed as follows:

```
for all the 8 face-cubes do
  biject the voxels of the given face-cube into the voxels of the base cube;
  compute the iso-patch cycles in the base 8-cell;
  compute the bi-iso-segments in the base 8-cell;
endfor
organize the obtained cycles into hyper-cycles;
organize the obtained segments into hyper-segments;
```

Hyper-segments or cycles [2] are oriented, not necessarily planar polygons, while hyper-cycles are oriented, not necessarily 3D polyhedra whose faces are exactly the above mentioned cycles. We emphasize that not only the cycles, but also the reconstructed hyper-cycles are oriented, and this orientation is coherent with the orientation of the corresponding hypercubes. That is, adjacent cycles composing the hyper-faces of a hyper-cycle have opposite orientation, and likewise: hyper-cycles have opposite orientation in adjacent 16-cells. A simple example for hyper-bi-iso-segment calculation is demonstrated in Fig. 5.

The previous scheme can be used independently for all 16-cells leading to an exhaustive sequential scan. We can also start the extraction at some seeds, and if a hyper-iso-patch cycle has been found, we propagate the computation to the neighboring cells. In either case we state:

**Proposition 9.** *This algorithm produces an oriented, complete (except at the boundary) hyper-iso-surface, without self-intersections. The same statement applies to the intersection of hyper-iso-surfaces.*

The proof of correctness is based on the inductive definition of hypercube orientation and the proposed orientation-preserving reduction. It is the extension of the proof of the MLA, its description may be found in [3]. At last we note that the intersection of three (four) hyper-iso-surfaces can be similarly computed, giving lines (points).

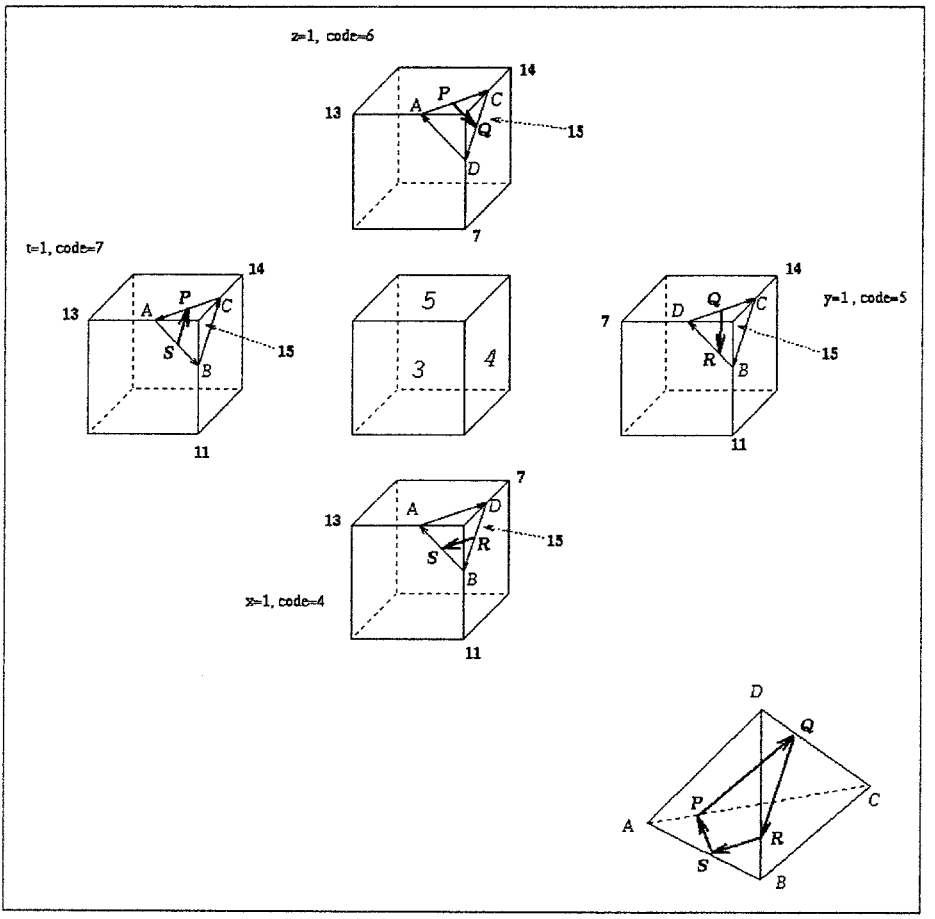


**Fig. 5.** Computing a hyper-bi-iso-segment in a simple case when only the 15th voxel is labeled to be positive.
*(left:)* The inner cube is the base cube; to the left, the cube numbered 7 is placed. The other three cubes are adjacent to the marked (i.e. front, right, top) faces of the base cube. The corresponding iso-patches and bi-iso-segments are drawn in each cube.
*(bottom right:)* The iso-patch cycles and also the bi-iso-segments are properly organized due to the correct orientation. The reconstructed hyper-cycle is oriented; its adjacent hyper-faces (here: triangles) have opposite orientation (e.g.: $A \to D \to B \to A$ and $B \to D \to C \to B$) due to the opposite direction of their common edge (e.g.: $D \to B$ in cube 4 and $B \to D$ in cube 5).

## 4 Some scale-space consideration

The main idea of scale-space theory [16, 9] is to consider an image and its details at different levels of scale, applying logarithmically increased smoothing. A well-known scale-space property (refered later as "simplification property") is that "no spurious detail is generated" when the scale is increased. At a coarse scale, only the most characteristic structures of the original image are conserved, however, they are delocalized and may have undergone important topological changes (splitting, merging, disappearing). To get precisely located and significant structures, they should be tracked from coarse to fine scale. In practice, sampling, delocalization and topological changes make this task difficult.

### 4.1 About topology

We cannot guarantee that the topology of a reconstructed discrete iso-surface is exactly the same as the one of the original continuous object: All the information represented at a scale smaller than the voxel size is lost after discretization; moreover, the connectivity varies with the extraction scale. Lots of techniques have been proposed (mean-value, gradient heuristics, ... see [8] for a survey) to correctly determine the topology, but non of them take into account that the border changes with scale and derivatives must be taken in some neighborhood whose size influences the result. To treat this problem we should extract and consider iso-surfaces at increasing levels of scale i.e. we should analyze a 4D hyper-iso-surface.

### 4.2 Following features across scale

Theoretical results about the behavior of *differential singularities* – features which can be defined as zero-crossings of differential expressions – have been obtained only in the continuous case [10, 7]. In practice [1, 12], these are difficult to employ, discretization effects the detection, so we need to exemplify the orbits of some features in scale space to get a statistical overview. We have already described a multiscale representation method [5, 4] based on iso-surface detection, that we have applied to follow 2D corner points (absolute maxima of the iso-phote curvature) at increasing scales and analyzed their evolution. Now we have a method to investigate the scale-space behavior also of 3D characteristic lines like parabolic [13] or crest lines [15] (absolute maxima of the largest principal curvature). It consists of the intersection of two hyper-iso-surfaces, where the 4th dimension is the scale. The first hyper-iso-surface $(f_{scale}, I)$ is a usual 3D iso-surface extracted at increasing scales, while the second $(g_{scale}, 0)$ is defined by the corresponding differential expression computed also at increasing scales.

Multiscale extraction, based on zero-crossings i.e. (hyper-)iso-surface detection, is preferable to local extrema search on each level of resolution followed by pairing between adjacent levels. Since simply matching features extracted at different levels can give false pairing due to possible strong delocalizations. However, in the case of (hyper-)iso-surface detection, the parent-child connection of singularities is established directly from the voxel structure, which ensures that the topological changes are automatically followed. Fig. 6 shows the change of parabolic lines, where the connectivity information is obtained during the 4D extraction.

Tracking crest lines across scale is difficult compared to other differential characteristic. Crest lines are not defined in case of zero gradient or at umbilic points, that

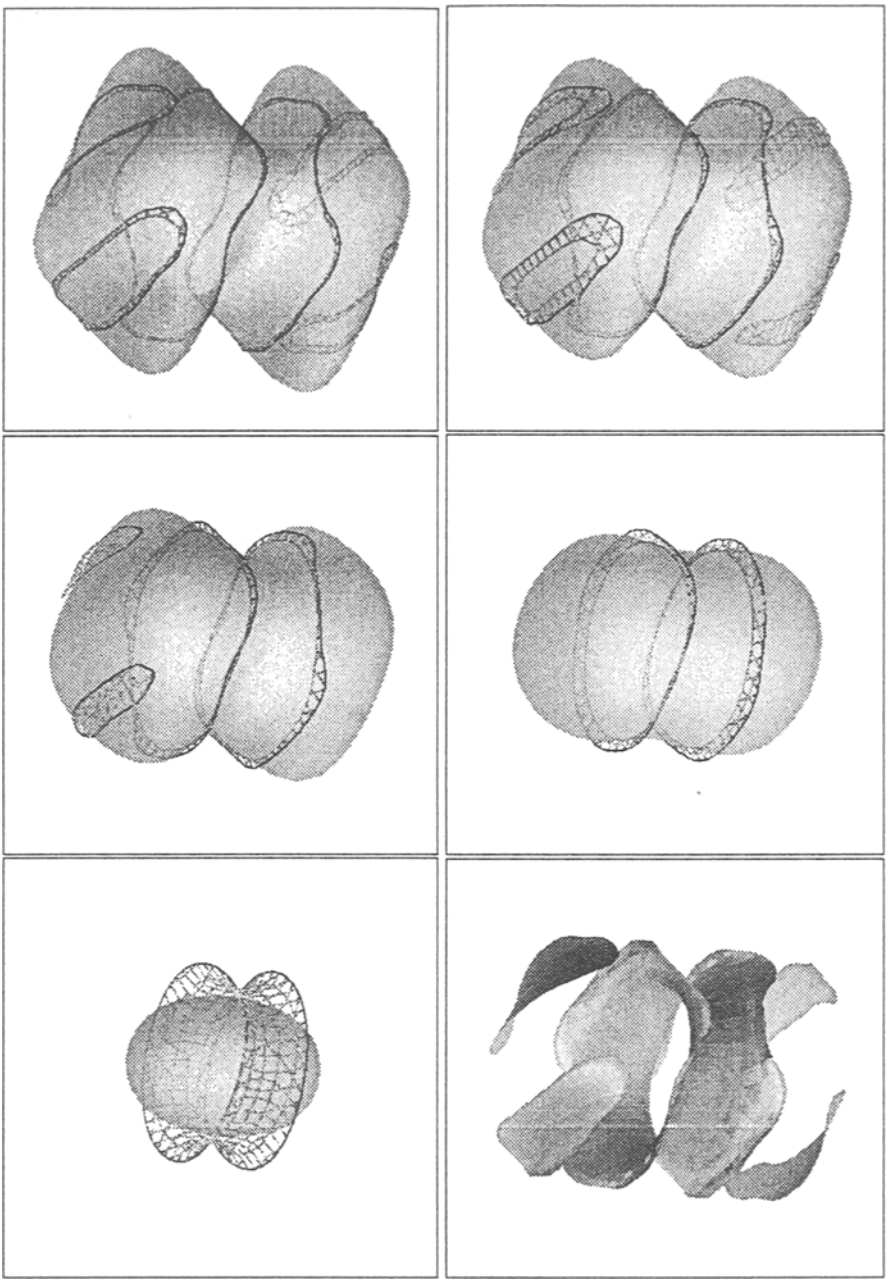

**Figure 6.** Change of parabolic lines with connectivity information.
*(1 - 5)* Iso-surfaces are extracted at scales $\sigma_i = \varepsilon\tau^i$ where $i = 1, \ldots 5$, $\varepsilon = 1.0$ and $\tau = \sqrt{2}$. Parabolic lines are green if obtained at that scale, they are red if obtained at the previous scale, while the connecting segments are yellow. *(6)* Change of lines via scales gives a surface in 4D, the scale dimension is represented by colors. Lines are red at the finest scale, as the scale increases the color changes according to a rainbow, so the lines at the coarsest scale are purple. Observe that the connectivity is correct even in case of high delocalization.

is why the MLA gives not necessarily closed lines. Practically it means that we have to stop the marching at one scale but we have to continue at an other scale, while processing the same position (since zero gradient or umbilicity are singularities not only in space but also in scale); as a consequence, the extracted bi-iso-segments are degenerated to a point. Work is still in progress to find a proper solution to this problem. First results about multiscale following are presented in Fig. 7.

## 5 Implementation details

To reduce the computation time and the use of memory, we structure the program in such a way that only two adjacent levels of a 4D image (i.e. two 3D images) are processed at the same time. In fact, in multiscale applications or when following time evolution, we can suppose that the adjacent levels are fairly related. Moreover, thanks to the simplification property of scale space, the levels at the two finest scale contain much more details than the rest. (We are not sure that similar statements exist in case of time evolution.) So intensive computation is only needed at two neighboring levels, corresponding to the highest scales, to start the algorithm, then the obtained (hyper-)cycles are to be propagated for the next levels, that is, we have to process the rest voxels only if they are on a reached 16-cell. However, this way new emerging events are not necessarily detected, contrary to an intensive sequential scan. A possible variant, which is more useful in practice, is the coarse-to-fine propagation: in this case we extract features at the finest scale only if they have a descendant at the coarsest scale. The propagation can be controlled with two temporary queues (one for the actual level, one for the next); see [3] for further details.

As we have seen, the 4D algorithm can be traced back to 3D calculations via orientation-preserving bijections. The transformation of hyper-faces can be obtained by a look-up table procedure, as opposed to solving equations. This greatly speeds up the 4D $\rightarrow$ 3D reduction; similarly, other topological information like edge and face connection can also be stored in look-up tables.

## 6 Conclusion

We have presented a method to extract 4D hyper-iso-surfaces and their intersection curves, which is a natural extension of the 3D Marching Lines algorithm with new orientation and implementation considerations. Though our main discussion and also the implementation have been done in 4D, all the statements are valid for any dimension. We have also shown some possible applications related to scale space. Currently, an important task is to follow 3D characteristic curves, such as crest lines, across scale. In the future we intend to investigate the multiscale behavior of these lines so as to get robust detection and good base for inter-patient registration. Also, we plan to apply this method to analyze gated SPECT images of the beating heart.

**Acknowledgment**
We wish to thank Alexis Gourdon and Jean-Philippe Thirion for stimulating discussions about the Marching Lines algorithm, also to Hervé Delingette and Stéphane Cotin who have drawn our attention to some results of topology. We thank Bruce Latimer, Director at the Cleveland Museum of Natural History, Court Cutting, David Dean and André Guéziec for the CT-scan data of the skull.

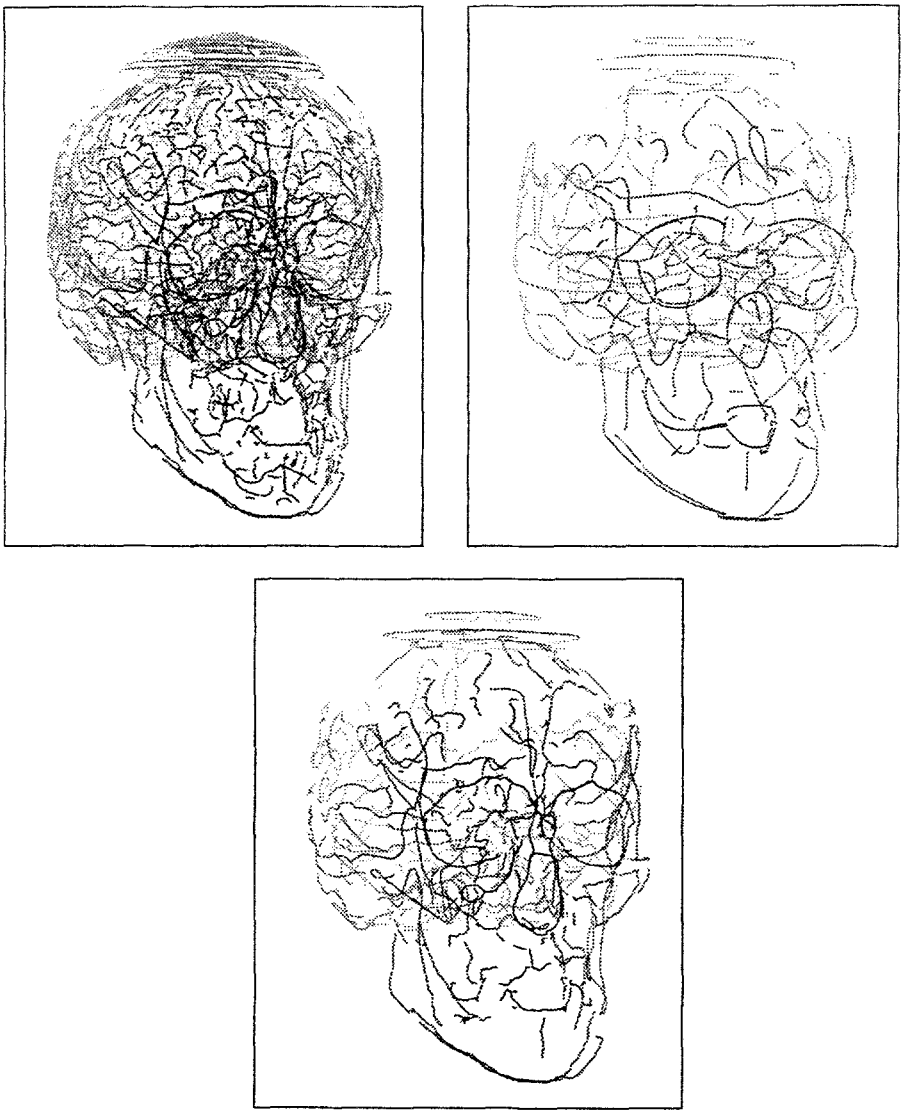

**Fig. 7.** Crest lines of a segmented, binary image of the skull.
*(top left:)* Extraction scale is $\sigma = 2$. We can observe that there are lot of artifactual curves due to discretization. *(top right:)* Extraction scale is $\sigma = 6$. Only the most significant curves are preserved, however they are not at all precisely located (e.g. the nose). Moreover, important topological changes have taken place (e.g. the orbit of the eye). *(bottom:)* Only the lines extracted at $\sigma = 2$ and having a descendant at $\sigma = 6$. These lines are more characteristic than the complete set at $\sigma = 2$. We emphasize that the connectivity information between lines obtained at different scales is established during the 4D extraction.

## References


1. Haruo Asada and Michael Brady. The curvature Primal Sketch. *IEEE PAMI*, 8, 1986.
2. H.S.M. Coxeter. *Regular Polytopes*. Dover Publications, 1973.
3. Márta Fidrich. Iso-surface Extraction in 4D. Technical Report 2833, INRIA, Feb 1996.
4. Márta Fidrich and Jean-Philippe Thirion. Multiscale Extraction of Features from Medical Images. In *Int. Conf. on Computer Analysis of Images and Patterns*, volume 970 of *LNCS*, pages 637–642, Prague, September 1995.
5. Márta Fidrich and Jean-Philippe Thirion. Multiscale Representation and Analysis of Features from Medical Images. In *Int. Conf. on Computer Vision, Virtual Reality and Robotics in Medicine*, volume 905 of *LNCS*, pages 358–364, Nice, April 1995.
6. Jacob D. Furst, Stephen M. Pizer, and David H. Eberly. Marching Cores: A Method for Extracting Cores from 3D Medical Images. In *SIAM Workshop on Mathematical methods in Biomedical Image Analysis*, San Francisco, California USA, June 1996.
7. John M. Gauch and Stephen M. Pizer. Multiresolution Analysis of Ridges and Valleys in Grey-Scale Images. *IEEE PAMI*, 15, 1993.
8. Alan D. Kalvin. A Survey of Algorithms for Constructing Surfaces from 3D Volume Data. Technical Report RC 17600, IBM Research Division, January 1992.
9. Jan J. Koenderink. The structure of Images. *Biological Cybernetics*, 50:363–370, 1984.
10. Tony Lindeberg. Scale-space behaviour of local extrema and blobs. *Journal of Mathematical Imaging and Vision*, 1:65–99, March 1992.
11. William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. *Computer Graphics*, 21(4), July 1987.
12. Farzin Mokhtarian and Alain K. Mackworth. Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes. *IEEE PAMI*, 8, Jan 1986.
13. A. Noble, D. Wilson, and J Ponce. Computing Aspect Graphs of Smooth Shapes from Volumetric Data. In *SIAM Workshop on Mathematical methods in Biomedical Image Analysis*, San Francisco, California USA, June 1996.
14. Jean-Philippe Thirion. New Feature Points based on Geometric Invariants for 3D Image Registration. Technical Report 1901, INRIA, April 1993. (Accepted for publication in IJCV).
15. Jean-Philippe Thirion and Alexis Gourdon. Computing the Differential Characteristics of Isointensity Surfaces. *CVGIP*, pages 190–202, March 1995. (also a Tech. Report $n°$ 1881).
16. Andrew P. Witkin. Scale Space Filtering. In *Int. Conf. on Artificial Intelligence*, volume 511, 1983. Karlsruhe.
17. Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2(4):227–234, 1986.