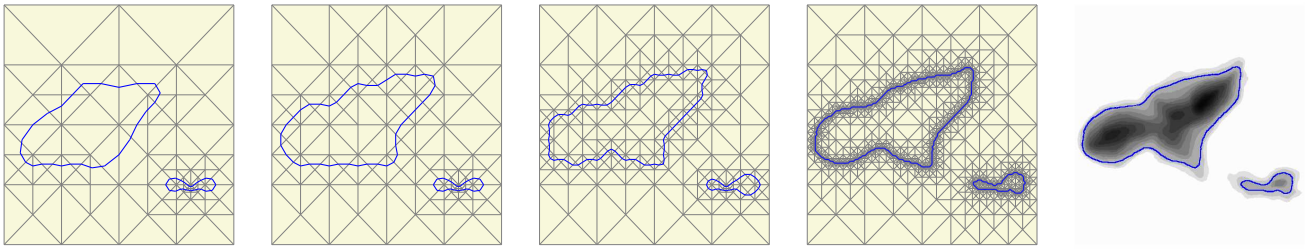
Hierarchical isocontours extraction and compression

THOMAS LEWINER[1,2], HÉLIO LOPES[1], LUIZ VELHO[3] AND VINÍCIUS MELLO[3]

[1] Department of Mathematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Géométrica Project — INRIA – Sophia Antipolis — France
[3] Visgraf Project — IMPA — Rio de Janeiro — Brazil
{tomlew, lopes}@mat.puc--rio.br. {lvelho, vinicius}@visgraf.impa.br.

Abstract. In this work, we introduce a new scheme to extract hierarchical isocontours from regular and irregular 2D sampled data and to encode it at single rate or progressively. A dynamic tessellation is used to represent and adapt the 2D data to the isocontour. This adaptation induces a controlled multi–resolution representation of the isocontour. We can then encode this representation and control the geometry and topology of the decoded isocontour. The resulting algorithms form an efficient and flexible isocontour extraction and compression scheme.
Keywords: ***Level sets. Data Compression. Simplicial Methods. Progressive Transmission. Geometry Processing.***



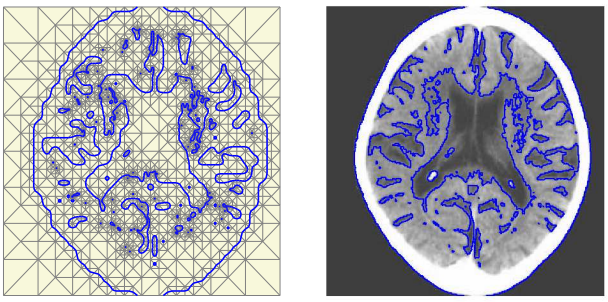
(a) 21 bytes (b) 31 bytes (c) 46 bytes (d) 156 bytes (e) 690 bytes

Figure 1: *Progressive compression of an elevation curve of the Sugar Loaf. The tessellation is adapted to the curve to minimize the distortion and to preserve the topology. (a) The shape of the tubular neighborhood of the curve is sent first, with the nodes sign. (b),(c),(d) Then the refinements of the tessellation are encoded with the signs of the new nodes. (e) Finally, the values of the nodes in the tubular neighborhood are refined.*

1 Introduction

Curves are one of the basic building blocks of geometry processing. They are used to represent shape in 2D images, terrain elevation on maps, and equations in mathematical visualization. In most of those applications, the curves can be interpreted as an isocontour of a 2D dataset, possibly mapped on a more complex space. Those isocontours are flexible objects that can be refined or reduced, that can deform with differential simulations or mathematical morphology, and that can be described for shape classification or automatic diagnostic in medicine or geosciences.

Problem statement. Given the sampling $\hat{f}$ of scalar function f defined over a domain D embedded in $\mathbb{R}^2$ (such as a 2D image or a discrete surface), the *isocontour* of an *iso*value α is the curve $f^{-1}(\alpha)$. Such an isocontour corresponds to only a small part of D, but usually covers a large area of the domain. For example, the cortex corresponds to only specific x–ray scintillation inside the scan of the whole head (see Figure 2), the elevation curve is only a small part inside a topographic map (see Figure 1). Therefore, specific compression techniques for isocontours should provide better compres-



(a) 1619 bytes (b) 4024 bytes

Figure 2: *Progressive compression of the cortex of a Computational Tomography image, topology controlled.*

Preprint MAT. 13/04, communicated on May 10^{th}, 2004 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 234–241 IEEE Press, 2004.

sion rate than the encoding of the entire 2D data.

Contributions. In this paper we introduce a new method for extracting and compressing isocontours based on a dynamic tessellation of the 2D data. This structure shows very nice adaptation properties, allowing extraction of the isocontour with different level of details. The main idea is to encode the tubular neighbourhood of the isocontour extracted from different levels of detail of the tessellation. Moreover, the adaptation the tessellation can depend on the isocontour, providing to our compression scheme a full control on the geometry and topology of the decoded isocontour. The resulting algorithms are flexible, can handle irregular 2D data, single–rate and progressive transmission together with uniform and adapted refinements.

2 Related work

In this work, we will use dynamic adaptive triangulations to represent and encode two–dimensional isocontours. This section describes some relevant works related to dynamic adaptive tessellations, and hierarchical isocontour extraction and isocontour compression.

Adaptive Tessellations. Hierarchical data structures are traditionally used for progressive compression and visualization of images. The usual representation for images relies on a rectangular grid that is subdivided uniformly or adaptively with a quad–tree. However, these structures are restricted to rectangular data sets. The size of those rectangles reduces twice as fast as the sizes of triangles in triangular tessellations, resulting in less adaptability. We will therefore focus on triangulations. [15] introduced multi–triangulations as a general concept for adapted variable resolution simplicial structures. [14] developed a binary multi–resolution structure based on stellar operators, which is a multi–triangulation with optimal properties. [18] proposed a very simple scheme for dynamically adapting triangulations while maintaining regularity conditions.

Hierarchical Isocontour Extraction. A hierarchical representation of an isocontour can be obtained by reduction of the polygonal curve of a single isocontour: [5] introduced the first algorithm for reduction of the polygonal approximation of a curve in the plane. Since then, this algorithm has been extended and improved in many aspects (see [19] for guarantees on the consistency of the reduced curve).

Nevertheless, this hierarchy can be obtained by extracting isocontours at each level of detail of a multi–resolution representation of the 2D data [16, 12]. The approximation of complex implicit curves usually requires robust computation, whose result can be seen as an isocontour. Hierarchical structures such as quad–trees usually provide simple and efficient solutions [11]. Similarly, the hierarchical representation of the image data can be adapted to the specified isocontour. For example [9] provides a hierarchy of rectangles to represent the isocontour, using genetic algorithm to optimize the dimensions of the rectangles. Those hierarchical representations are numerous when dedicated to a specific application, in particular for shape analysis [13, 3].

Isocontour compression. Isocontour compression is usually done as a compression of a non–self–intersecting curve, for example as two separate signals for each coordinate, or as a vector displacement [4, 17]. It can also be compressed by the popular chain code when the curve points are limited to pixel quantization [6]. In that case, it can be compressed as a 2D signal [8, 2]. [7] introduced another concept by encoding a hierarchical representation of the isocontour induced by a multi–resolution of the 2D data. The progression tries to maintain the chessboard distance from the original curve to the encoded one. The coarser resolution is encoded as two separate signals, and the position of the points introduced by the refinements are encoded as a difference with the near-by points.

3 Overview

In this work, we intend to encode a hierarchy of isocontours by their tubular neighbourhood. The tubular neighbourhoods are extracted from an adapted triangulation of the original 2D data. The data structure we will use to represent the 2D data is the one of [14, 18]. We adapt it to the neighbourhood of the isocontour, and reduce it according to the isocontour topology, geometry and position inside the triangulation. This structure allows to a very simple multi–resolution isocontour extraction, and enables us to compress the curve at single rate or progressively. Moreover, it is well suited for uniform and adapted progression on both regular and irregular 2D data. It can prevent topological changes or high geometric distortion during the progression. Finally, it works with sub–pixel interpolation for the curve, which enables smooth curve reconstruction at any level of detail.

Paper outline. We will introduce the adaptive triangulation we used in section 4 *Adaptive Tessellation*. This structure can represent regularly sampled data with the same amount of sample points as the classical quad-tree representation, but it can also adapt to irregular data. It also provides simple and effective controls on the topology and geometry of the isocontour as explained in section 5 *Isocontour Extraction*. Our compression scheme is introduced in section 6 *Isocontour Compression*, and the results are showed in section 7 *Results*.

Definitions. The *2D data* is given as a collection of samples v_i, each of which is associated with its *isovalue* $\hat{f}(v_i)$. Those samples are triangulated. In particular, when the samples are regularly spaced on a 2D grid (a grey–scale image for example), this triangulation can be automatically generated by the subdivision of a two–triangles square.

Assuming that we want to obtain the isocontour corresponding to the isovalue α, a sample of the 2D data is classified as *positive* or *negative* depending whether its isovalue is greater than α or not. An edge of the triangulation is *crossing* when its end–points have opposite signs. A triangle is crossing when it contains a crossing edge. The *tubular neighbour-*

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 234–241 IEEE Press, 2004.

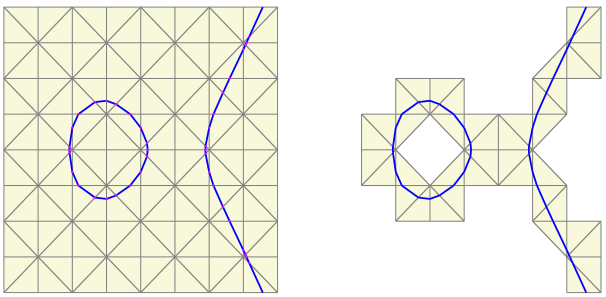


Figure 3: *A regular BMT adapted to an isocontour and the corresponding tubular neighbourhood.*

hood of the isocontour is the set of all the crossing triangles (see Figure 3).

From now on, the isocontour will be approximated by a polygonal line linking the *isocontour points* interpolated at each crossing edge of the tessellation. In the case of a grey–scale image, this corresponds to a linear sub–pixel interpolation (as the purple points of Figure 3). The samples of the 2D data will be referred as the *vertices* of the triangulation, as opposed to the *points* of the isocontour.

4 Adaptive Tessellation

The multi–resolution structure we will use here is the Regular Binary Multi–Triangulation (RBMT) [14, 18]. This structure is constructed using Stellar operators on edges and can decompose adaptively the 2D data. These decompositions can be regarded as hierarchies of conforming triangulations. In this section, we will give a brief description of the Stellar operators on edges and the binary multi–triangulations.

(a) Stellar Operators

Stellar theory [1] studies the equivalencies between simplicial complexes (i.e., a generalization of a triangulation) and defines topological operators that change structures while maintaining their integrity and coherence with the modelized object.

The *stellar subdivision* operation inserts a vertex into an r–simplex of an n–dimensional simplicial complex. The inverse of this operation is called *stellar simplification*. Stellar theory states that stellar operators *on edges* are sufficient to map any two equivalent manifolds [1]. Edge–subdivision and Vertex–simplification will be the basic operators to construct Binary Multi–Triangulations (see Figure 4).

(b) Binary Multi–Triangulation

The Binary Multi–Triangulation (BMT) is a multi–resolution structure based on stellar subdivision on edges. When subdividing an edge, its incident triangles are subdivided in two. Therefore, a sequence of subdivisions on edges can be represented as a binary tree structure, where each node represents a triangle and the two sons of a node t are the two triangles obtained by subdividing t. This binary tree

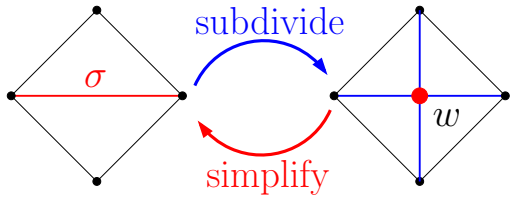
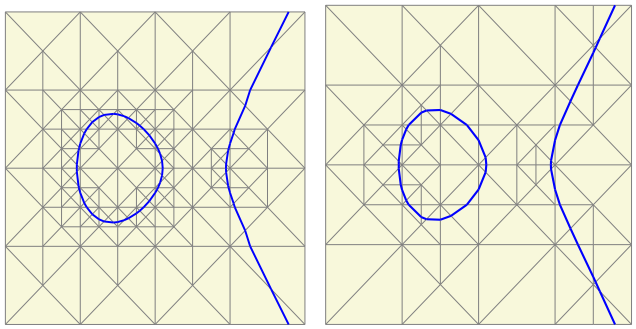


Figure 4: *Subdividing an edge σ $\leftrightarrows$ simplifying a vertex w.*



(a) curvature (b) distortion

Figure 5: *The isocontour of Figure 3 adapted according to (a) the curvature and (b) the distortion.*

(actually a forest) adapts more nicely than the classical quadtree for image decomposition. The *level* of a triangle is its depth in the binary tree.

The BMT is *reduced* or *refined* by walking up and down the binary tree, creating a hierarchy of triangulations at different *resolutions*. We perform those operations efficiently by maintaining for each triangle t, the vertex w that has been inserted during the subdivision that created t. The vertex w is called the *simplification vertex* of t, and the edge opposite to w is called the *subdivision edge* of t. For example Figure 6 shows the subdivision edges of the BMT as darker edges.

(c) Adaptation Properties and Regularity

The adaptability of the BMT comes from the possibility to refine and reduce the triangulation, while maintaining the dependencies between its triangles. In particular, we will maintain gradual transitions by preventing two adjacent triangles from differing by more than one level. To do so, it is necessary to propagate a subdivision or simplification to adjacent triangles. The propagation of a subdivision on an edge e is performed by checking that each triangle t adjacent to e has e as a subdivision edge. If it is not the case, a subdivision is performed on the subdivision edge of t. This subdivision can require other triangles to be subdivided if they do share their subdivision edges. The propagation of a simplification on a vertex is done in a similar way. With this restriction, the resulting structure is called a *regular binary multi–triangulation* (RBMT). In that structure, subdividing a triangle means subdividing its subdivision edge, while simplifying it means simplifying its simplification vertex.

For example, Figure 6 illustrates a sequence of refine-

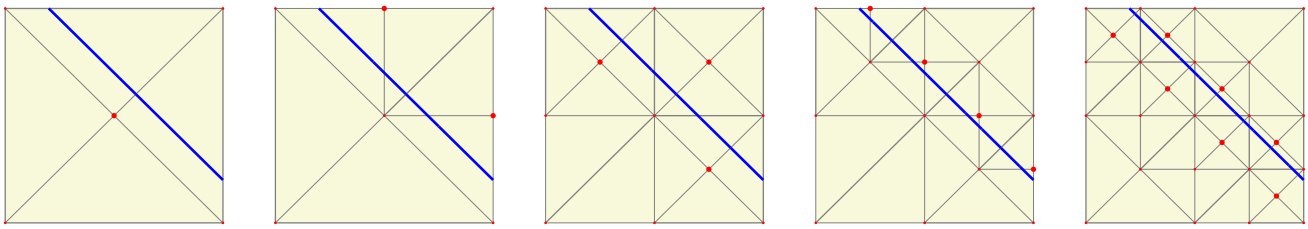


Figure 6: *RBMT adapted to the bold line: at each level, every triangle crossing the bold line is refined, but subdivisions in the upper–right part of the square propagate to the lower–left part.*

ments adapting the triangulation to the bold line. In order to preserve gradual transition between resolution levels, local refinements around the bold line propagates inside the triangulation (in this example, far away from the bold line), as what happens to the bottom left part.

5 Isocontour Extraction

The isocontour is computed as a polygonal curve. Each crossing triangle t of the RBMT contains a segment $[p_1, p_2]$ of the isocontour. The extremities p_1 and p_2 of the segment are linearly interpolated on the two crossing edges v_0v_1 and v_0v_2 of t: $p_i = \lambda \cdot pos(v_0) + (1-\lambda) \cdot pos(v_i)$ with $\lambda = \frac{isovalue(v_i)}{isovalue(v_i)-isovalue(v_0)}$. If the isocontour is extracted from an image, the subpixel interpolation can generate an ambiguity. This ambiguity is solved (somehow arbitrarily) by the tessellation, since only triangular elements are interpolated. The resulting isocontour is controlled redundantly by the isovalue of the vertices (grey level of the representing pixels) and by the size and position of the edge. This double control provides more flexibility on the isocontour progressive compression.

Multi–resolution Representation. The representation of the isocontour corresponds to the polygonal interpolation on the edges for different resolutions of the RBMT. These levels are obtained by successive reductions of the finest level. The reductions can be uniform, simplifying all triangles whose level is greater than a given threshold (see Figure 6). It can also be adapted to the topology of the curve. For example, it is possible to simplify all the triangles that are not crossing, preserving the isocontour and its tubular neighbourhood (see Figure 6. The refinement can also be done in order to preserve small triangles where the curve has a high curvature, and to reduce the triangulation where it is more flat (see Figure 5(a)). In a similar way, during compressing, we will be able to minimize the distortion of the curve and to preserve its topology (see Figure 5(b)).

Distortion control. The distortion induced by a simplification of a vertex w can be estimated by the encoder. To perform such a simplification, the star of w needs first to be composed of only four triangles (see Figure 7). The points of the reduced curve will be the interpolation of the isocontour on the remaining edges, with the isovalues of the vertices

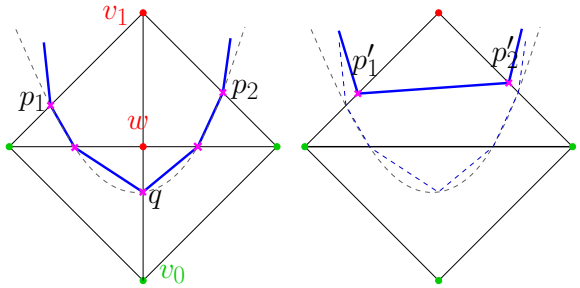


Figure 7: *The simplification of w induces a distortion that can be measured as* $\max\{d(q,p'_1); d(q,p'_2)\}$.

quantized according to the compression tuning. For example, in the case of Figure 7, the distortion can be estimated as the maximal distance between q and p'_i.

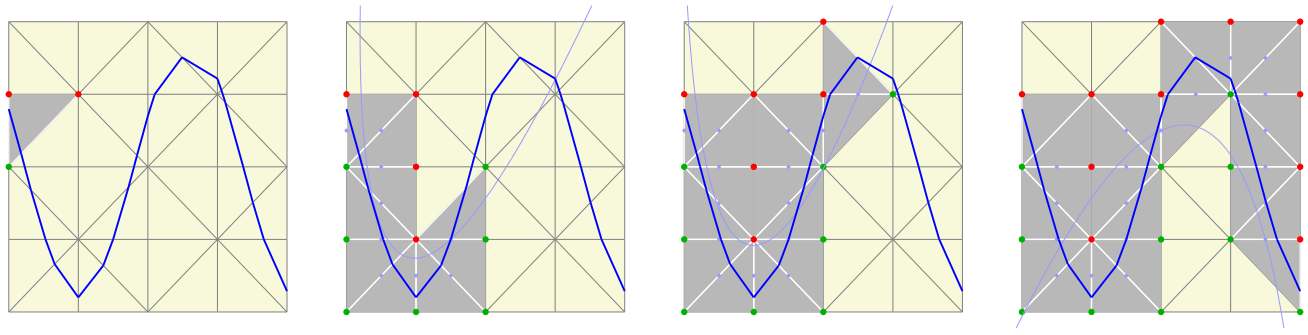
Topology control. Similarly, the topology of the isocontour can be easily controlled during the simplification of a vertex w. Actually, there are a few prohibited simplifications. The destruction of a connected component occurs when all the four edges incident to w are crossing. The separation of a connected component in two or the merge of two connected components happen when the subdivision edge v_0v_1 is not crossing, while wv_0 and wv_1 were crossing edges.

6 Isocontour Compression

The techniques we are now to introduce perform both direct and progressive encoding of the tubular neighbourhood of an isocontour. Moreover, the encoding of this neighbourhood can be done uniformly (i.e., the triangles of the tubular neighbourhood of the isocontour have constant size, see Figure 8) or adaptively (i.e. the size of the triangles varies according to the contour see Figure 10).

This section is organized as follows. We will introduce our method for encoding the tubular neighbourhood of the isocontour first uniformly and then adaptively. These methods can be used to encode the coarser resolution of the progression, or to encode the entire isocontour at single rate. Then, we will detail our techniques to encode the refinements, used for the progressive compression. Again, this process can be done uniformly or adaptively. Finally, we will explain the geometry encoding, and describe how to reuse the

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 234–241 IEEE Press, 2004.



(a) Location: level 4, 1, $L\ R\ L\ R$. $+\ -\ -$. (b) Vertex signs: $--+++++$. (c) 2 known vertices, then: $-\ +\ -$. (d) $----++-+-$*End.*

Figure 8: *Uniform encoding of the coarser resolution of a small sinusoid. The light curve is a second–order fitting of the decoder's points (in the middle of the crossing edges), and serves as geometrical predictor.*

compressed data of one isocontour to encode other near–by ones on the same scalar data.

(a) Coarser Resolution

Our main idea is to encode the tubular neighbourhood going along with the isocontour, from one crossing triangle to an adjacent one in the tubular neighbourhood. The algorithm encodes first the localization of an unvisited crossing triangle t_0, and the signs of its vertices. From this initial triangle, it follows the isocontour, encoding the sign of each unvisited vertex encountered. When the traversal is done, it continues on the next connected component. Notice that the vertices of the only tubular neighbourhood have their isovalue encoded, leaving our algorithm almost independent of the initial size of the 2D data size.

Localization. The location of a triangle t_0 can be encoded using the binary tree inherent to the RBMT. The root of the tree contains only a few triangles: 2 for a regular grid. The triangle containing t_0 is encoded by its index. Then, knowing the level l of the triangle to encode, it can be localized using a sequence of l symbols Left and Right (see Figure 8(a), Figure 10(a) and Figure 10(d)). The signs of its vertices are then encoded and all of them are marked as visited; the initial triangle t_0 is also marked as visited.

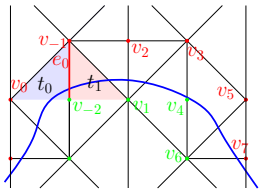


Figure 9: *Coarser resolution compression: the traversal goes from t_0 to t_1 through the gate e_0, and encodes the vertex v_1.*

Uniform encoding. Once the signs of the vertices of the initial triangle t_0 have been encoded, its crossing edges are known to the decoder. The first one is chosen to be the first gate e_0 (see Figure 9). The triangle t_1 ($\neq t_0$) incident to the gate e_0 is also crossing. The sign of the vertex v_1 opposite to e_0 is encoded, and both t_1 and v_1 are marked as visited. The algorithm then continues the same way starting as t_1. The traversal ends when both triangles incident to the gate are marked, or when the gate is a boundary edge of the RBMT.

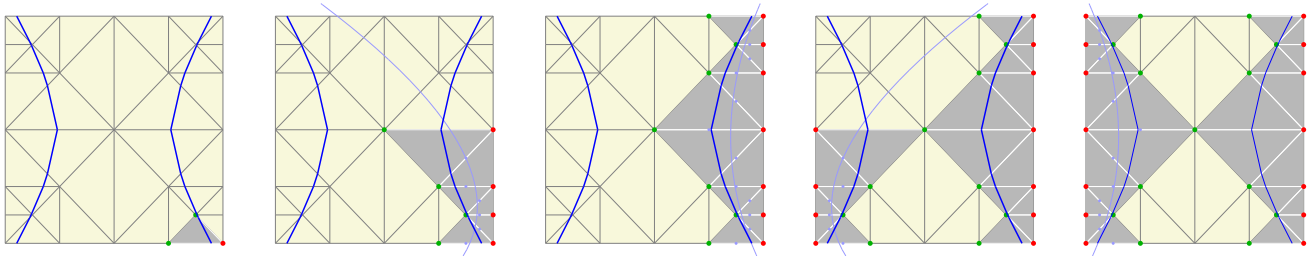
Actually, the sign of the vertex v_1 is encoded only when it has not been marked. This guarantees to encode exactly one sign bit per vertex, with an overhead of a few bits per connected component, used for the localization procedure (see Figure 8).

Adaptive encoding. The above algorithm can be easily modified to encode the tubular neighbourhood of the isocontour when it is composed of triangles of different levels. In that case, the algorithm also encodes the level of the current triangle (t_1) during the traversal. The decoder will read the required level for t_1 and subdivide or simplify t_1 if necessary. The RBMT we use maintain a difference of at most one level between adjacent triangles (see section 4 *Adaptive Tessellation*). Therefore, the decoder will have to subdivide or simplify an unvisited triangle at most once, and the encoder will manage only 3 symbols: $'=$' when the levels match, $'>'$ or $'<'$ when the levels differ. The encoding of the signs is done similarly to the uniform one.

This guarantees to encode exactly one bit per vertex of the tubular neighbourhood, plus one symbol of $\{=, <, >\}$ per triangle of the tubular neighbourhood, with an overhead of a few bits per connected components, used for the localization procedure (see Figure 10).

(b) Progressive refinement

The refinement methods allow a progressive adaptation of the tubular neighbourhood to the isocontour, using each time smaller triangles. The algorithm can simply encode the signs of the new vertices created by the subdivisions of all triangles on the tubular neighbourhood, or it can first specify which triangles to subdivide and then send the sign of the new vertices inserted.



(a) Location: level 5, 0, $L\ R\ L\ L\ L.\ -\ +\ +.$

(b) Vertex levels and signs: $>\ -\ =\ -\ <\ +\ <\ -\ <\ +.$

(c) $=\ +\ >\ -\ >\ +\ >\ -\ =\ -\ <\ +.$

(d) level 5, trig 0, $L\ L\ L\ L\ L.\ -\ +\ +.\ >\ -\ =\ -\ <\ +\ <\ -\ <.$

(e) $=\ +\ >\ -\ >\ +\ >\ -\ =\ -\ <\ +$*End.*

Figure 10: *Adaptive encoding of the coarser resolution of a small hyperbola.*

Uniform refinement. The uniform refinement simply subdivides first all the triangles of the tubular neighbourhood, and then en/decodes the signs of the vertices created by the subdivision inside the former tubular neighbourhood. The order of the new vertices is induced by the traversal used for the coarser resolution en/decoding.

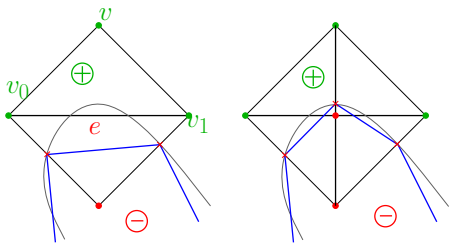


Figure 11: *A subdivision can extend the tubular neighbourhood of the isocontour if the sign of $vv_0 > 0$ and $vv_1 > 0$.*

When subdividing a triangle crossing the isocontour, the configuration of its sub–triangles is determined by the sign of the new vertex introduced on the subdivision edge. This sign is encoded during the compression. This subdivision can locally extend the tubular neighbourhood (see Figure 11). This case occurs when a non–crossing subdivision edge $e = v_1v_2$ gives rise to two crossing sub–edges. In that case, the vertex v opposite to e will be included in the tubular neighbourhood, but its sign need not to be encoded as it can be deduced by the decoder as the sign of v_1 (see Figure 11).

Adaptive refinement. Our method also allows an adaptive progression, creating smaller triangles where the isocontour is more complex, and leaving bigger triangles where it is simple. For each subdivision edge in the tubular neighbourhood, we encode the Refine and Keep code. When a Keep symbol is encoded, the subdivision edge will not be considered in future refinements, keeping it as a leaf on the hierarchy. The subdivision edges of the tubular neighbourhood are collected in the order of the coarser resolution traversal. The sign of the newly inserted vertices is encoded in the same way as for uniform refinement.

Actually two successive resolution of the RBMT can differ locally by more than one level in the binary tree. Therefore, the above sequence will be repeated as long as a Cont/Stop bit is read by the decoder after the vertex signs are received.

(c) Final geometry encoding

Once the tubular neighbourhood of a resolution level is encoded, only one bit for each vertex of the RMBT is known to the decoder. Therefore, the position of each point p of the isocontour is *a priori* the middle point of a crossing edge e. This can be improved by sending the isovalue of the endpoints of e. Moreover, this scalar value will be used to encode other isocontour generated with a different isovalue.

The input 2D data regularity can actually influence the quality of the compression (see Figure 12). For specific applications such as human face contour compression, the input data can be sampled according to the mean and variances of the original isocontour, resulting in an elliptic–radial initial distribution of the 2D data.

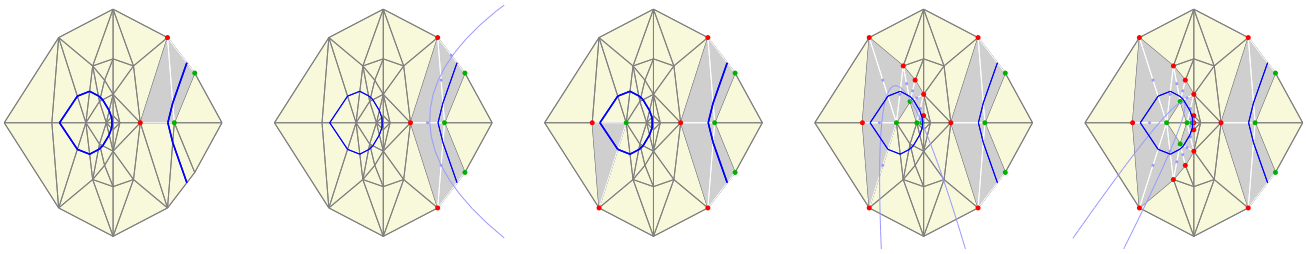
(d) Close–by isocontour encoding

Once an isocontour C corresponding to an isovalue α_C has been decoded, the decoder knows its entire tubular neighbourhood. It can be used to encode also isocontours C' corresponding to isovalues $\alpha_{C'}$ close to α_C, especially for medical applications where the contrast of the 2D data is not well defined. C' can be easily encoded as a new coarser resolution by sending first the isovalue $\alpha_{C'}$ and then the quantized isovalues of the vertices not precise enough or unknown to the decoder.

7 Results

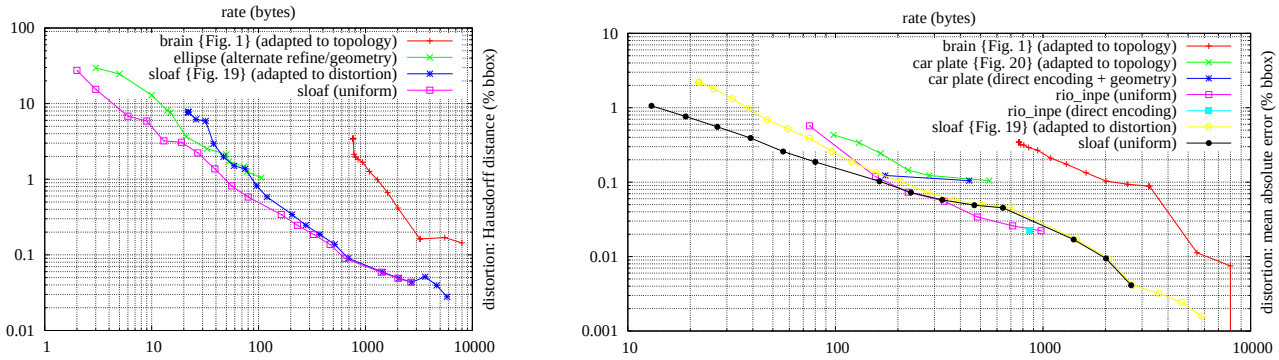
The methods we introduced here are flexible and can be used in many ways. For isocontours of images, we chose to reduce the complete triangulation of the pixels. We then encoded the isocontour at each step of this reduction, which can be uniform or adapted to the isocontour. For implicit curves, we sent in–between two levels of detail a part of the geometry (2 bits). We implemented our compression scheme with a context arithmetic coder of order 0 for the coarser level transmission, 2 for the refinements and 1 for the final geometry encoding. We used a simple predictor based on a second–order curve fitting [10] (see Figure 8 and Figure 10).

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 234–241 IEEE Press, 2004.



(a) Location: level 6, 1, $L\ L\ L\ L\ L\ R. - + +. < -.$ (b) Vertex levels and signs: $= - > +.$ (c) level 5, trig 0, $L\ L\ L\ L\ L.$ (d) $- + -. = - = - > + > - = - < + = -.$ (e) $> - = - = - =< + = - > - = + << End.$

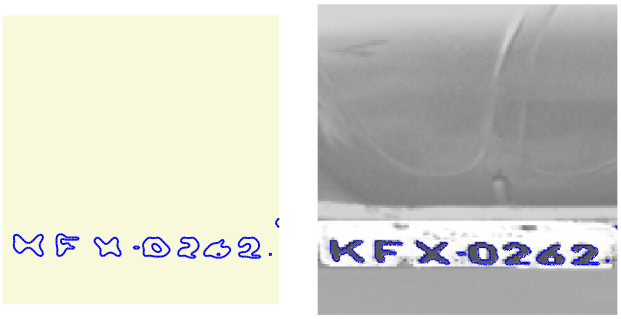
Figure 12: *Adaptive encoding of the coarser resolution of a cubic: irregular tessellation can reduce the distortion.*



(a) Hausdorff distance (b) Mean absolute error

Figure 13: *Compression results: on complex models, topology control is mode expensive with progressive encoding.*

Our method resulted in efficient rate/distortion curves (see Figure 13). The different controls on the adaptation of the multi–resolution can have a significant cost. For example, on regular models such as the elevation curve of the Sugar Loaf of Figure 1, the distortion and topology controls provide nicer results to the eyes at the beginning of the compression than uniform refinements, while finally resulting in similar performances. Topology control means an extra cost depending on the complexity of the model (compare the car number plate of Figure 14 with the brain model of Figure 2 on Figure 13), but the rate/distortion performance has a similar evolution for all methods and models (see Figure 13(a)). Geometry encoding seems a good alternative to refinements for regular models such as the ellipse of Figure 13(a). Single rate encoding is, as usual, a little more efficient than progressive encoding, but can be complemented by the final geometry compression of section 6(c) *Final geometry encoding*.



(a) progressive: 97 bytes (b) direct: 173 bytes, progressive: 283 bytes

Figure 14: *Progressive compression of a car number plate (with enhanced contrast), topology controlled: direct encoding and progressive encoding both provide efficient compression ratio.*

8 Next steps

We introduced a complete isocontour multi–resolution extraction and compression scheme. The dynamic triangulation we used provides a very simple way of creating a multi–resolution structure of the 2D data adapted to the isocontour. This structure allowed us to achieve both direct and progressive compression, and to encode the isocontour uniformly or adaptively. Moreover, it provides a full control on the progression, granting topological and geometrical control on the decoded isocontour.

We used binary multi–triangulations for the scalar data representation, implemented using dimension–independent generic programming [14]. This offers a very simple extension of our algorithm to compress level sets in any dimension with a similar efficiency. This will require first to optimize the arithmetic coding and prediction parameters we use for

the compression, and to quantify the advantages of geometry encoding upon refinement operations.

References

[1] J. W. Alexander. The combinatorial theory of complexes. *Annals of Mathematics*, 31:219–320, 1930.

[2] F. Bossen and T. Ebrahimi. A simple and efficient binary shape coding technique based on bitmap representation. In *Acoustics, Speech, and Signal Processing*, pages 3129–3132, 1997.

[3] L. da Fontoura Costa and R. M. Cesar Jr. *Shape analysis and classification*. CRC Press, 2000.

[4] M. Craizer, D. A. Fonini Jr and E. A. B. da Silva. Alpha–expansions: a class of frame decompositions. *Applied and Computational Harmonic Analysis*, 13:103–115, 2002.

[5] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required for represent digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, 1973.

[6] H. Freeman. Computer processing of line drawing images. *Computing Surveys*, 6(1):57–97, 1974.

[7] C. Le Buhan and T. Ebrahimi. Progressive polygon encoding of shape contours. In *Image Processing and its Applications*, pages 17–21, 1997.

[8] G. Langdon Jr and J. Rissanen. Compression of black–white images with arithmetic coding. *Transactions on Communications*, 29(6):858–867, 1981.

[9] P. Lee and T. Nagao. Hierarchical description of two dimensional shapes using a genetic algorithm. In *Evolutionary Computation*, pages 637–640. IEEE, 1995.

[10] T. Lewiner, J. Gomes Jr, H. Lopes and M. Craizer. Arc–length based curvature estimator. In *Sibgrapi*, pages 250–257, Curitiba, Oct. 2004. IEEE.

[11] H. Lopes, J. B. Oliveira and L. H. de Figueiredo. Robust adaptive polygonal approximation of implicit curves. *Computers & Graphics*, 26(6):841–852, 2002.

[12] S. Mallat. *A wavelet tour of signal processing*. Academic Press, 1999.

[13] S. Marshall. Review of shape coding techniques. *Image and Vision Computing*, 7(4):281–294, 1989.

[14] V. Mello, L. Velho, P. Roma Cavalcanti and C. Silva. A generic programming approach to multiresolution spatial decompositions. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 337–360. Springer, Heidelberg, 2003.

[15] E. Puppo. Variable resolution triangulations. *Computational Geometry: Theory and Applications*, 11(3–4):219–238, 1998.

[16] A. Rosenfeld, editor. *Multiresolution image processing and analysis*. Springer, Berlin, 1984.

[17] A. Safonova and J. Rossignac. Compressed piecewise–circular approximations of 3D curves. *Computer–Aided Design*, 35(6):533–547, 2003.

[18] L. Velho. A dynamic adaptive mesh library based on stellar operators. *Journal of Graphics Tools*, 9(2), 2004.

[19] S. T. Wu and M. R. G. Márquez. A non–self–intersection douglas–peucker algorithm. In *Sibgrapi*, pages 60–66. IEEE, 2003.

The corresponding work was published in the proceedings of the Sibgrapi 2004, pp. 234–241 IEEE Press, 2004.