A Topological Method of Surface Representation

Vladimir Kovalevsky

Technische Fachhochschule Berlin, Luxemburger Str. 10, 13353 Berlin, Germany
kovalev@tfh-berlin.de

Abstract. A new method of representing a surface in the 3D space as a single digitally continuous sequence of faces is described. The method is based on topological properties of quasi-manifolds. It is realized as tracing the boundary of a growing set of labeled faces. As the result the surface is encoded as a single sequence of mutually adjacent faces. Each face is encoded by one byte. The code of the surface of a three-dimensional object takes much less memory space then the raster representation of the object. The object may be exactly reconstructed from the code. Surfaces of a genus greater that zero (e.g. that of a torus) may also be encoded by a single continuous sequence. The traversal algorithm recognizes the genus of the surface.

1 Introduction

We consider here the possibility to represent the boundary surface of an arbitrary connected subset of voxels in a 3D space as a single continuous sequence of faces. Such a sequence may be encoded by means of some kind of „generalized chain code“ in a rather economical way. The described method is also applicable for *scanning a connected subset* of a surface by stepping from one face to another which is adjacent with it, in such a way that the set of already scanned faces is „compact“, i.e. it is connected and its boundary is not too long. This is important for dissolving a surface into patches each of which is a subset of a digital plane similarly to dissolving curves into digital straight segments [Kov97]. The well-known methods of encoding surfaces are not applicable for this purpose: the method by Gordon and Udupa [GorUd89] dissolves the surface into closed loops isolated from each other; the method of representing the surface as an Euler circuit [RosKW91] is only applicable to a whole closed surface.

We consider here the three-dimensional space not as Z^3 but rather as a Cartesian abstract cell complex (ACC) as defined in earlier publications of the author, e.g. in [Kov93]. To make the reading easier we have gathered the most important definitions in Appendix 1 (Definitions 1 to 12). An ACC has all traditional properties of a topological space, i.e. it possesses a system of open subsets (Definition 3 in the Appendix) which satisfies the classical topological axioms [Kov89]. Therefore we often speak of a space, while implying an ACC.

We consider the coordinate axes of the space as one-dimensional ACC's (Definition 1) similar to the Khalimsky line [KhalKM90]. We assign subsequent integer numbers to the mutually incident cells of the axes: 0-cells of the axes get even numbers, 1-cells get odd numbers. These numbers are declared to be the *topological coordinates* of the cells. There are four different kinds of cells in the three-dimensional space: three-dimensional cells (3-cells) are the *voxels*, 2-cells are the *pixels* (the same as *surfels* or *faces*), 1-cells are *cracks* (the same as *linels* or *edges*) and 0-cells are *points* (the same as *pointels*). We shall use the term „pixel" (as in [RosKW91]) instead of „face" in order to retain the same term for a 2-cell both in a 3D and in a 2D space.

A cell of the space is considered as a Cartesian product of three cells of each of the axes: a voxel is a product (i.e. a triple) of a 1-cell of the X-axis, a 1-cell of the Y-axis and a 1-cell of the Z-axis; a pixel is a product of two 1-cells and one 0-cell etc. The bounding relation in the Cartesian complex is defined as follows: a cell with topological coordinates (X_1, Y_1, Z_1) bounds a distinct cell (X_2, Y_2, Z_2) iff all three absolute differences $|X_2-X_1|$, $|Y_2-Y_1|$ and $|Z_2-Z_1|$ are less or equal to 1, the dimension of the cell X_1 in the complex of the X-axis is not greater than that of X_2 and the same condition is fulfilled for the other two coordinates.

2 Boundaries of Sets of Voxels

We consider here surfaces which are boundaries of sets of voxels. We use the traditional topological definition of the boundary (frontier, Definition 4) by means of open sets, the latter being defined by means of the bounding relation of the cell complex (Definition 3).

Our aim is to detect and to encode surfaces, i. e. connected components of boundaries of objects in the 3D space. Topological properties of boundaries are of great importance for our method. We have found, that the boundary of a connected set of voxels may have a rather complicated topological structure even in the case of strongly connected sets of voxels (Definition 8): *boundaries are not always two-dimensional manifolds* (2-manifolds, Definition 10). Thus e. g. a crack incident with two voxels of a set V, which have no common pixel, is incident with four pixels of the boundary of V, which is prohibited in a 2-manifold. It is easy to find examples of strongly connected sets, in which a boundary point is incident with two umbrellas (Definition 12) which is also prohibited in a 2-manifold.

An umbrella in the boundary of a strongly connected set of voxels may have a self-intersection along a crack as demonstrated in Fig. 1. The umbrella about the point P consists of six cracks and seven pixels: four of them are shown as shaded areas, the remaining three are not visible; they are shown by dashed lines. The pixels F_1, F_2, F_3 and F_4 (the latter two are not indexed in Fig. 1) are all incident with the crack C_z.

It is easily seen that imposing additional restrictions, such as e.g. *locally strong* or *regular* connectedness, onto the voxel sets under consideration does not essentially change the situation: boundaries of voxel sets encountered in applications do rarely possess the properties of manifolds.

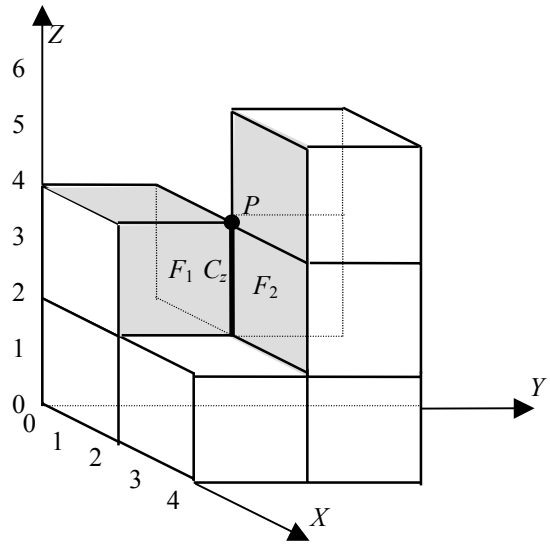

Fig. 1. Example of an umbrella with a self-intersection along a crack

However, the boundaries still have a rigorous topological structure which may be used to organize the tracing of the boundary. We suggest to denote this structure as a *two-dimensional quasi-manifold*.

Definition Q1: A *one-dimensional quasi-manifold* is a connected one-dimensional ACC in which every 1-cell is bounded by exactly two 0-cells and every 0-cell bounds an even number, at least two, of the 1-cells.

Definition Q2: A *two-dimensional quasi-manifold* is a connected two-dimensional ACC in which every 2-cell is bounded by 0- and 1-cells composing a one-dimensional manifold, i.e. a cycle; every 1-cell is bounded by exactly two 0-cells and bounds an even number, at least two, of the 2-cells. The 1- and 2-cells bounded by a 0-cell compose one or more subcomplexes each of which is *B*-isomorphic (Definition 9) to a one-dimensional quasi-manifold.

An example of a one-dimensional quasi-manifold is shown in Fig. 2. As an example of a two-dimensional quasi-manifold (2-quasi-manifold) consider the boundary (i. e. the surface) of the voxel set shown in Fig. 1.

A notion important for handling *n*-quasi-manifolds is that of adjacency of *n*-cells as suggested in [RosKW91]. The adjacency defined there depends upon the voxel connectedness: the given set of voxels may be declared either as a 6- or 18-connected one. There is no such notion in the topology of cell complexes; the membership of

cells of lower dimensions in the set under consideration must be specified instead: e.g. if the crack common to two voxels having no common pixel belongs to the same set as the voxels then they are connected, otherwise they are not.

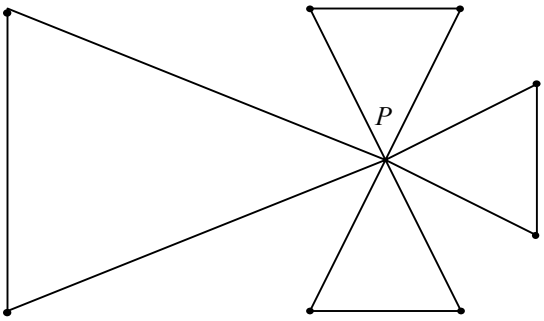

Fig. 2. An example of a one-dimensional quasi-manifold:
the point *P* bounds 8 1-cells; each of the other points bounds two 1-cells

The adjacency of the *n*-cells of an *n*-quasi-manifold may be defined here by *locally* dissolving the quasi-manifold into manifolds. "Locally" means that only *subsets* of the quasi-manifold and of the desired manifolds in the neighborhood of the cell which bounds all the *n*-cells under consideration are considered. These subsets intersect neither each other nor the connected set of voxels. This topological foundation and generalization of the adjacency of *n*-cells in an *n*-quasi-manifold will be published in a separate paper. The adjacency used here is equivalent to that defined in [RosKW91] for 6-connected objects and an 18-connected background.

In the simplest case of a Cartesian 3D space the difference between a 2-manifold and a 2-quasi-manifold consists in the following:

1) A crack of a 2-quasi-manifold may bound four pixels, while a crack of a 2-manifold may bound only two. These four pixels compose two pairs of adjacent pixels. A path through the pixels of a 2-quasi-manifold, when traversing such a crack, must go from one of the adjacent pixels to the other.
2) A point of a 2-quasi-manifold may belong to as many as four umbrellas (Definition 12), while a point of a 2-manifold may belong to only one umbrella. If there are more than one umbrella incident with a point *P*, then they may be either separate (except the common point *P*) or two umbrellas may share a crack. In the latter case pairs of adjacent pixels must be specified to define the order of traversing the pixels in the neighborhood SON(*P*, *B*) of a point *P* in the 2-quasi-manifold *B*. (The SON of a point is its smallest open neighborhood according to Definition 3a in the Appendix).

Let us firstly consider sets of voxels which are strongly connected (Definitions 6 to 8) and do not touch the border of the 3D image. Another important requirement is that both the set and its complement are homogeneously tree-dimensional complexes (Definition 11). We shall call such sets *solid* ones. This property is important when

regarding a set of voxels as an ACC, since the concept of an ACC allows that two voxels are in the set under consideration, however their common side (e.g. the common pixel or a common crack) is not. In such a case this side belongs to the boundary and makes the structure of the boundary rather complicated, not suitable for our purpose.

Proposition 1: The boundary B of a simply and strongly connected solid subset V of the Cartesian 3D space (ACC) is a two-dimensional quasi-manifold.

The proof may be found in Appendix 2.

3 The Ideas of the Traversal Algorithm

Our algorithm is based on the properties of a 2-quasi-manifold. Consider a surface which is the boundary B of a given set V of voxels. If V satisfies the conditions of Proposition 1 then B is a 2-quasi-manifold. Let us label the closure of a single pixel of B and consider it as the seed of the set L of labeled cells of B. The boundary of L relative to B is a 1-manifold, i.e. a closed sequence of alternating points and cracks. It may be traced by a sequence of steps from one point to the next one until the starting point is reached again. During this tracing the closures of pixels incident with the cracks of the boundary of L are joined with L and labeled as those belonging to L. In this way L becomes greater while remaining strongly connected. We shall show below, that if only pixels *simple with respect to* L are joined with L, then L remains simply connected and its boundary remains a 1-manifod. The boundary can be traced in the same manner as before. During the tracing of the current boundary of L the relative positions of joined pixels are stored as some kind of a three-dimensional chain code. This procedure must be repeated as long as there are unlabeled simple pixels incident with the cracks of the boundary of L. If the next pixel encountered during the tracing is a non simple one then it is not joined with L, i.e. it is not labeled. However its position is nevertheless included into the code to make the sequence of the encoded pixels continuous. At the beginning of the process the trajectory of the tracing is evolving like a spiral. The sequence of grasped pixels looks like the peeling of a potato.

The tracing stops, when there are no more simple pixels incident with the boundary of L. At this stage all pixels of B are grasped in the code (some of them more then once). This code may be used as the code of the surface B. The code specifies the surface B and therefore also the set V uniquely: the set V may be exactly reconstructed from the code.

Let us consider the theoretical background of these ideas.

Definition S: When given a 2-quasi-manifold B and a subset L of B, a pixel F of the set $B–L$ is called *simple with respect to* L if the intersection of its boundary $\mathrm{Fr}(F,B)$ with the boundary $\mathrm{Fr}(L,B)$ of L is connected.

We shall show next that if a simple pixel is united with L, then the number of components both of L and of $B–L$ remains unchanged. Thus when only simple pixels are labeled, the spiral tracing runs continuously along the boundary of L until all simple pixels are labeled. After that the non simple pixels at the end of the sequence are labeled. This method (with some improvements) works well on all possible kinds of surfaces.

Proposition 2: Let L be a subset of a 2-quasi-manifold B while the boundary $\mathrm{Fr}(L,B)$ of L relative to B is a 1-manifold. Then the boundary $\mathrm{Fr}(L\cup\mathrm{Cl}(F),B)$ of the union $L\cup\mathrm{Cl}(F)$ of L with the closure $\mathrm{Cl}(F)$ of a pixel F of $B–L$ having a common crack with L and being simple with respect to L is either still a 1-manifold or it is empty. In the latter case L becomes identical with B.

The proof may be found in Appendix 2.

To describe the algorithm we need the following notions.

The orientation of a movement along the boundary $\mathrm{Fr}(F,B)$ of a pixel F is called *clockwise* if the following three vectors compose a right system, similar to that of the unit coordinate vectors of the coordinate axes X, Y and Z:

- vector 1 is pointing from F to a boundary cell;
- vector 2 is that of the movement from that cell to the next one;
- vector 3 is the inner normal to F, pointing to the inside of the object's voxel.

Such a movement is seen as a clockwise one around the center point of F when looking from outside of the object along the inner normal.

Labeling a pixel F means setting a memory variable corresponding to this pixel to one. Simultaneously the variables corresponding to all cells of the boundary $\mathrm{Fr}(F,B)$ are set to one. At the start of the algorithm all variables are initialized by zero. In the computer realization each variable is a bit in the raster representation of the given 3D image containing one byte per voxel.

As explained in Section 1, we use here the Khalimsky-coordinates of the cells. Therefore it is easy to calculate the movement from one cell to another which is incident with it: the movement is specified by the vector whose components are the differences of the corresponding coordinates. It is possible to interpret the coordinates of cells of dimensions greater then zero as those of their middle points, which makes the representation descriptive and comprehensible.

To encode the movements from one cell to another during the tracing we use the notation represented in Fig. 3. The numbers from 0 to 5 denote the movements from one cell to another incident with it since these movements are always parallel to one of the coordinate axes.

Hypothesis: Let In be the set of all pixels of $B–L$, which are incident with $\mathrm{Fr}(L,B)$. If In contains no simple pixels then it is identical with $B–L$. If it is empty, then the genus of B is zero.

The hypothesis was confirmed by numerous computer experiments with surfaces of genus from 0 to 9.

We denote the code of the set In which contains no simple pixels as the *rest sequence*. It contains all codes obtained during *a full tracing* of the boundary $\mathrm{Fr}(L,B)$ after the last simple pixel was found.

The Algorithm:

1. Take any pixel of B as the starting pixel F_0, label its closure and save its coordinates as the starting coordinates of the code. This is the seed of L. Denote any one crack of the boundary of Fr(F_0,B) as C_{old} and find the pixel F of B which is incident with C_{old} and adjacent with F_0. Set F_{old} equal to F_0 and the logical variable *REST* to FALSE. *REST* indicates that the tracing of the rest sequence is running.
2. (Start of the main loop:) Find the crack C_{new} as the first unlabeled crack of Fr(F,B) encountered during the scanning of Fr(F,B) clockwise while starting with the end point of C_{old} which is in Fr(L,B). If there is no such crack and F is labeled stop the Algorithm: the encoding of B is finished.
3. If F is simple label its closure.
4. Put the direction of the movement from F_{old} to C_{old} and that of the movement from C_{old} to F into the next byte of the code. If the pixel F is non simple set the corresponding bit in the code (to recognize codes of non simple pixels in the ultimate sequence).

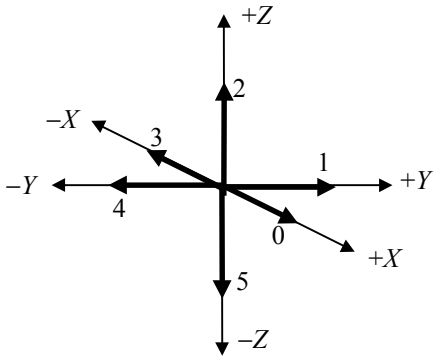

Fig. 3. Encoding of the directions of the coordinate axes and cracks

5. If *REST* is TRUE check, whether F is equal to F_{stop} and C_{new} is equal to C_{stop}. (These variables may be defined in item 6 of the previous loop). If it is the case stop the Algorithm and analyze the rest sequence to specify the genus of B as explained below. Delete multiple occurrences of pixels from the rest sequence.
6. If F is simple set *REST* equal to FALSE. If F is non simple set F_{stop} equal to F, C_{stop} equal to C_{new} and *REST* equal to TRUE.
7. Set F_{old} equal to F. Find the pixel F_{new} of B incident with C_{new} and adjacent to F. Set F equal to F_{new} and C_{old} equal to C_{new}. Go to item 2.

End of the Algorithm.

As it may be seen from the Algorithm, after the last simple pixel has been found, the tracing of Fr(L,B) is continued and the non simple pixel are recorded (in the rest sequence) until the path along the boundary Fr(L,B) becomes closed. The genus of the surface is computed by means of the Euler number of the rest sequence, which in turn

may be deduced from the number of non simple pixels which were visited three or four times.

Proposition 3: The genus G of B may be deduced from the properties of the rest sequence. It is equal to:

$$G=(1+N_3/2+N_4)/2; \quad (1)$$

where N_3 is the number of pixels which occur in the rest sequence three times and N_4 is the number of pixels which occur in the rest sequence four times. The proof may be found in Appendix 2.

The run time of the Algorithm is obviously proportional to the length of the code, since the time necessary to produce a code element is constant. The length of the code may be essentially greater then the number of pixels in B since the non simple pixels may arrive in the code many times. The worst case take place for sets of voxels looking like a dumbbell: two cylinders of a constant diameter connected at their flat ends by a long thin rod. After the pixels in the surface of the rod are labeled except of a sequence one pixel wide, the tracking may run along the rod back and forth as many times as the number of pixels in the height of the cylinders. Thus the worst case complexity is $O(N^2)$, N being the number of pixels in B.

There is a possibility, to make the complexity linear in N. This may be reached if the Algorithm changes the tracing mode from the clockwise to the counterclockwise one each time when a non simple pixel occurs. This version of the Algorithm is described below.

4 The Reversible Tracing

The only drawback of the method of Section 3 is that the part of the boundary of L which is incident with non simple pixels must be sometimes traced again and again in order to reach all simple pixels. This is due to the fact that in spite of the connectedness of $B-L$ the *subset of simple pixels* in $B-L$ may be disconnected. Thus the sequence may contain many non labeled pixels and the sequence becomes unnecessary long.

This difficulty may be overcome by means of *reversing the direction of the tracing* each time when a non simple pixel is encountered. This means that the tracing is being continued in such a direction that the labeled pixels lie no more to the left hand side of the direction (left tracing) but rather to the right hand side (right tracing).

Fig. 4a shows an example of the left tracing: the pixels labeled before are shown as a shaded region. They lie to the left hand side of the direction of the tracing. The pixel „N“ is a non simple one: its boundary crosses the boundaries of the pixels which were labeled before at two disconnected locations, namely at the pixels 3 and 14. Fig. 4b shows how the tracing has been continued as the right tracing starting with pixel 14 and running through the pixels 15 to 22.

When realizing the reversible tracing, it is more expedient to trace not the usual boundary of the labeled region but rather the so-called „open boundary“ (Definition 4a, [Kov92]) of this region. Sequences of traced pixels as shown in Fig. 4a and Fig. 4b are examples of subsets of the open boundary of the labeled region. The reason of changing to open boundaries is that the sequence of cracks with incident pixels does not contain the pixels incident with the „corners“, such as e.g. the pixel 3 in Fig. 4a. This was admissible for the algorithm of Section 3: a missing pixel was registered during the next turn of the spiral. However it is not admissible for the reversible tracing, as we shall see in what follows.

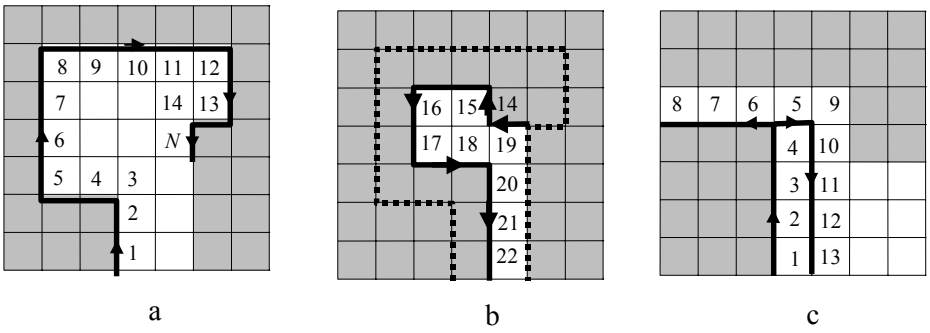

Fig. 4. Examples of reversible tracing: a) left tracing trough the pixels 1 to 14; b) continued as right tracing through pixels 15 to 22; c) simultaneous left and right tracing

In the most cases, when a non simple pixel is met in the open boundary, the tracing may be continued in the reversed direction, as e.g. in Fig. 4a, continued in Fig. 4b. However there are cases, when directly after such a reversing no simple pixels may be found (Fig. 4c). Then the tracing must be continued *simultaneously in both directions*. Two sequences of pixels are then recorded: one for the left tracing and one for the right one. As soon as in one of these sequences a simple pixel is encountered, the other sequence is discarded. In this way the number of non simple pixels in the resulting sequence is minimized.

In the example of Fig. 4c the left tracing goes on until the non simple pixel 5 is reached. Since that location two sequences are recorded: 5, 6, 7, 8, etc. for the left tracing and 5, 9, 10, 11 for the right tracing. At the location 12 a simple pixel is found. Thus the codes for the pixels from 6, 7, 8 etc. are discarded. The sequence 5, 9, 10, 11 is transferred into the ultimate sequence and the tracing is continued from pixel 12.

The question may arise: when will be the pixels 6, 7, 8 etc. picked up again and included into the code? According to Proposition 2, the boundary of L remains always a 1-manifold, i.e. a closed sequence of points and cracks without branching. Therefore one may be sure that at some time later on, the boundaries of these pixels will be reached by the tracing procedure.

To make this assertion illustrative we can continue the image of Fig. 4c in two different ways. If B is of genus 0, then the situation may look as shown in Fig. 5: pixels from 12 to 18 will be labeled since they are simple. After the pixel 18 has been labeled, the pixel X becomes simple an so do the pixels 11, 10, etc. until Y.

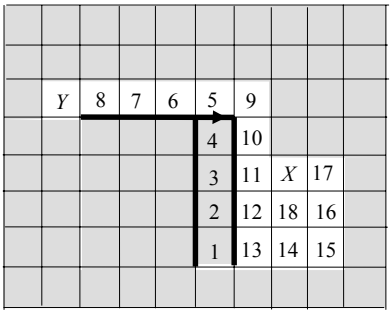

Fig. 5. A possible extension of the fragment of Fig. 4c; explanation in text

The ultimate sequence may contain (besides the rest sequence) some non simple pixels which are necessary to encode the path to further simple pixels. However the non simple pixels remain unlabeled. If the surface B is of genus 0 then the set of not labeled pixels *incident with the current boundary of L* becomes smaller and smaller until it is empty. Then the encoding of such a surface is finished.

The other possible way to extend the image of Fig. 4c is to considered it as a fragment of a surface of genus greater then 0. In this case the boundaries of the pixels 6, 7, 8 etc. will belong to the rest sequence as explained above and illustrated by Fig. 6. The pixels will be included into the code as non simple ones.

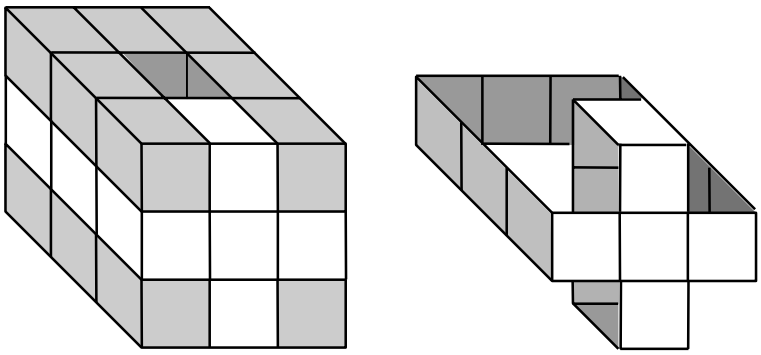

Fig. 6. An example of the pixels of the rest sequence:
a) the simple pixels of a torus are shaded, the non simple ones are left white;
b) the set of non simple pixels of the same torus without the voxels

In the case of a surface of a genus greater than 0 (e.g. a torus) a closed sequence of non simple pixels arises at the end of the tracing procedure: the left and the right tracings are continued without finding a simple pixel until they meet each other while having opposite directions. The algorithm must be organized in such a way that the sequences contain *all non simple pixels*: if some pixels at the starting point of the simultaneous tracing are missing, then they are lost for ever. This is the reason of using the open boundary rather then the closed one: the open boundary is a *continuous* sequence of adjacent pixels.

As in the method of Section 3, the non simple pixels contained in the rest sequence are labeled with „brute force“. The record of a pixel in the rest sequence is changed as for a simple pixel when the pixel arrives in the rest sequence for the first time. After that some records of non simple pixels still remain at the end of the sequence. These are the multiple records of pixels which were visited more than once. These records are used for the calculation of the genus and then deleted.

5 Computer Experiments

The method of Section 4 was successfully tested on many different 3D images. The first experiments were made with simple artificial volumes as a hemisphere or a hemisphere with a narrow vertically stretched parallelepiped attached at the upper side of the hemisphere like a chimney. The surface of the latter object has a folder in the umbrella where a vertical crack of the hemisphere touches a vertical crack of the chimney. More recent experiments were conducted with tori of different size and with objects having up to 9 tunnels, whose surfaces are of genus from 2 to 9. In all cases the surface was scanned and encoded completely. The 3D objects were exactly reconstructed from the code.

5.1 Efficiency of encoding

The pixels of the sequence are encoded by 7 bits each as explained below. Thus the code contains one byte per pixel. The encoding may be made more economical by the factor of 2 by means of the difference crack-code as suggested in [Kong92]. We have not realized this improvement in order to make the encoding and decoding as simple as possible.

The efficiency of encoding, measured as the ratio of the number of the pixels in the surface to the number of elements in the ultimate sequence (plus one), is essentially better as that of the method of Section 3. It depends on the genus of the surface, as shown in Table 1 for the “worst case”: small objects whose walls are only one voxel thick.

For bigger objects the efficiency is much better, as shown by the upper curve in Fig. 7. Thus, for example, the code of an parallelepiped of 38×38×19 voxels with 9 tunnels parallel to the Z-axis is 8952 bytes long. The surface contains 7704 pixels. The efficiency is equal to 0.861.

Table 1. Efficiency of the Code as a Function of the Number of Tunnels

Number of tunnels	0	1	2	3	4	5	6
Method of Sect. 3	1.0	0.485	0.336	0.231	0.179	0.147	0.125
Method of Sect. 4	1.0	0.842	0.685	0.624	0.619	0.608	0.557

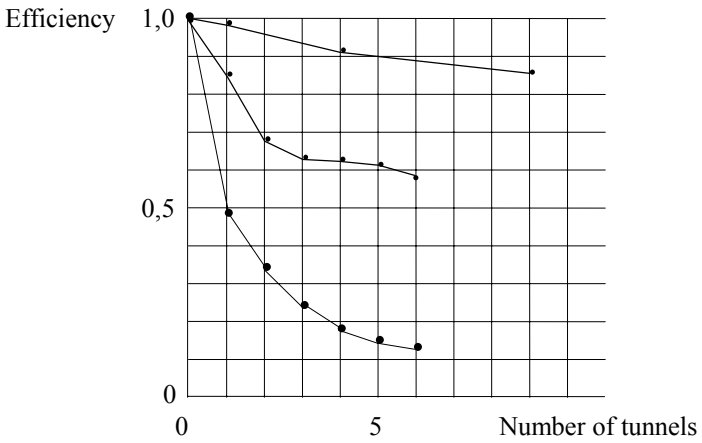

Fig. 7. Efficiency of encoding small objects with the method of Section 4 (middle curve) as compared to that of Section 3 (lowest curve);
the upper curve represents objects of 38×38×19 voxels

These results were compared with that of encoding a surface as an Euler circuit as suggested in [RosKW91]. In all our experiments the efficiency of the Euler circuit was about 0.67. It should be mentioned here, that the method of Euler circuit is only applicable to a whole closed surface, while our method may also be used to encode any connected part of the surface, which possibility is important e.g. for dissolving the surface into digital plane patches.

5.2 Examples of Codes.

The codes of the pixels are represented as octal numbers of three digits. The first digit shows, whether the pixel is simple (digit=0) or not (digit=1). The second digit is the direction of the movement from the a pixel to the following crack; the third digit is the direction of the movement from the crack to the next pixel. The directions are encoded as shown in Fig. 3.

1. A bar consisting of three voxels along the Y axis. 14 pixels, 13 code elements.
   Coordinates of the first pixel =(3, 3, 4);
   The code:
   035, 040, 001, 011, 023, 035, 011, 020, 005, 013, 054, 044, 044;
2. A torus of 8 voxels; the axis of the tunnel is parallel to the Z axis. 32 pixels, 37 code elements.
   Coordinates of the first pixel =(3, 3, 4);
   The code:
   035, 040, 000, 021, 015, 031, 023, 035, 011, 020, 000, 045, 051, 000, 144,
   002, 011, 113, 124, 144, 144, 005, 053, 011, 032, 020, 011, 015, 033, 033,
   054, 044, 044, 000, 100, 042, 021;
3. A *B*-shaped body with two tunnels parallel to the Z axis; 50 pixels, 72 code elements:
   Coordinates of the first pixel =(3, 3, 4);
   The code:
   035, 040, 000, 021, 015, 031, 023, 035, 011, 020, 000, 045, 051, 000, 144,
   002, 010, 051, 012, 000, 054, 044, 032, 020, 044, 133, 045, 000, 001, 011,
   011, 023, 033, 033, 044, 044, 000, 015, 054, 033, 011, 032, 120, 111, 015,
   033, 033, 054, 044, 044, 000, 100, 042, 121, 111, 135, 150, 144, 133, 133,
   111, 111, 112, 100, 100, 124, 100, 100, 105, 144, 144, 053,

The length of the code is always of the order of the number of the pixels in the surface: in the case of genus zero the number of bytes in the code is equal or a little greater then the number of pixels depending on the complexity of the shape of the surface. For surfaces with genus greater then zero the number of bytes is slightly greater than the number of pixels, depending on the genus and on the size of the surface (compare Fig. 7).

Conclusion

A new method of representing a surface in the 3D space as a single digitally continuous sequence of pixels is suggested. According to the method the surface is encoded as a single sequence of mutually adjacent pixels. Each pixel is encoded by one byte. The code of the surface of a three-dimensional object takes much less memory space then the raster representation of the object. The object may be exactly reconstructed from the code. Surfaces of a genus greater that zero (e.g. that of a torus) may also be encoded by a single continuous sequence. The algorithm recognizes the genus of the surface. The method is well suited for dissolving a given surface into patches of digital planes, which is its advantage over known methods.

References

[GorUd89] Gordon, D., Udupa, J.K.**,** *Fast Surface Tracking in Three-Dimensional Binary Images***,** CVGIP**, v. 45**, pp. 196-214, 1989.

[KhalKM90] Khalimsky, E., Kopperman, R. and Meyer, P.R., *Computer Graphics and Connected Topologies on Finite Ordered Sets*, Topology and Applications, **v. 36**, pp. 1-17, 1990.

[Kong92] T. Young Kong, „*On Boundaries and Boundary Crack-Codes of Multidimensional Digital Images*“, in „Shape in Picture“, Ying-Lie O et. all (Eds.), Springer-Verlag, 1992.

[Kov89] Kovalevsky, V.A., "*Finite Topology as Applied to Image Analysis*", Computer Vision, Graphics and Image Processing, **v. 46**, pp. 141-161, 1989.

[Kov92] Kovalevsky, V.A., "*Finite Topology and Image Analysis*", In "Image Mathematics and Image Processing", P. Hawkes (Ed.), „Advances in Electronics and Electron Physics“, **v. 84**, pp. 197-259, Academic Press 1992.

[Kov93] Kovalevsky, V.A., "*Digital Geometry Based on the Topology of Abstract Cell Complexes*", Proceedings of the Third International Colloquium "Discrete Geometry for Computer Imagery", pp. 259-284, University of Strasbourg 1993.

[Kov97] Kovalevsky, V.A., „*Applications of Digital Straight Segments to Economical Image Encoding*“, In: Ahronovitz, E. and Fiorio, C: (eds.), "Discrete Geometry for Computer Imagery", Proceedings of the 7th International Workshop, DGCI'97, pp. 51-62, Springer 1997.

[RosKW91] A. Rosenfeld, T. Young Kong and A.Y. Wu, „*Digital Surfaces“*, CVGIP GMIP, **v. 53**, pp. 305-312, 1991.

Appendix 1: Topology of Cell Complexes

In this section we shall remind the reader some topological notions necessary to read the presentation. In this way we hope to make the paper self-contained.

Definition 1: An *abstract cell complex* $C=(E, B, dim)$ (ACC) is a set E of abstract elements provided with an antisymmetric, irreflexive, and transitive binary relation $B \subset E \times E$ called the *bounding relation*, and with a dimension function $dim: E \rightarrow I$ from E into the set I of non-negative integers such that $dim(e') < dim(e'')$ for all pairs $(e',e'') \in B$.

Elements of E are called *abstract cells*. It is important to stress that abstract cells should *not* be regarded as point sets in a Euclidean space. That is why ACC's and their cells are called abstract. If a cell e' bounds another cell e'' then e' is called a *side* of e''. The sides of an abstract cell e'' are not parts of e'': the intersection of two distinct abstract cells, differently from that of Euclidean cells, is always empty.

The maximum dimension of the cells of an ACC is called its dimension. We shall consider ACC's of dimensions 2 and 3. Their cells with dimension 0 (0-cells) are called *points*, cells of dimension 1 (1-cells) are called *cracks* (edges), cells of dimension 2 (2-cells) are called *pixels* and that of dimension 3 are the *voxels*.

Definition 2: A *subcomplex S = (E', B', dim')* of a given ACC *C = (E, B, dim)* is an ACC whose set E' is a subset of E and the relation B' is an intersection of B with $E' \times E'$. The dimension *dim'* is equal to *dim* for all cells of E'.

Definition 3: A subset *OS* of cells of a subcomplex *S* of an ACC *C* is called *open in S* if it contains all cells of *S* bounded by some cells of *OS*.

We need this Definition to consider subsets which are open in the surface but not in the three-dimensional space. Note that in the topology of cell complexes there exists the notion of the *smallest open neighborhood* of a cell. This is the smallest open subset containing the cell.

Definition 3a: The smallest subset of a set *S* which contains a given cell *c* of *S* and is open in *S* is called the *smallest open neighborhood* of *c* relative to *S* and is denoted by SON(*c*, *S*).

Definition 4: The *boundary* Fr(*S*, *C*) of a subcomplex *S* of an ACC *C relative to C* is the subcomplex of *C* containing all cells *c* of *C* whose SON(*c*, *C*) contains both cells of *S* as well as cells of the complement *C*–*S*.

Definition 4a: The *open boundary* Ob(*S*, *C*) of a subcomplex *S* of an ACC *C relative to C* is the subcomplex of *C* containing all cells *c* of *C* whose closure $c \cup$ Fr(*c*, *C*) contains both cells of *S* as well as cells of the complement *C*–*S*.

Definition 5: Two cells e' and e'' of an ACC *C* are called *incident with each other in C* iff either $e'=e''$, or e' bounds e'', or e'' bounds e'.

Definition 6: Two cells e' and e'' of an ACC *C* are called *connected to each other in C* iff either e' is incident with e'', or there exists in *C* a cell *c* which is connected to both e' and e''. According to this recursive definition the relation of connectedness is a transitive closure of the incidence relation.

It may be easily shown that the connectedness relation according to Definition 6 is an equivalence relation (reflexive, symmetric and transitive). Thus it defines a partition of an ACC *C* into equivalence classes called the *components of C*.

Definition 7: An ACC *C* consisting of a single component is called *connected.*

Definition 8: A path in an ACC C of the form

$$x_0^n x_1^{n-1} x_2^n \ldots x_{l-1}^{n-1} x_l^n$$

where x_i^n is an n-dimensional and x_i^{n-1} is an $(n-1)$-dimensional cell of C, is called an *n-dimensional path in C*. An n-dimensional ACC C is called *strongly connected* if any two n-dimensional cells of C may be connected by an n- dimensional path in C.

Definition 9: Two ACC's are called B-isomorphic to each other if there exists a one-to-one correspondence between their cells which retains the bounding relation.

Definition 10: An *n-dimensional finite manifold* M_n is an n-dimensional ACC satisfying the following conditions:

a) a 0-dimensional manifold M_0 consists of two cells with no bounding relation between them;
b) an n-dimensional manifold M_n with $n > 0$ is connected;
c) for any cell c of M_n the subcomplex of all cells different from c and incident with c is B-isomorphic to an $(n-1)$-dimensional manifold.

Definition 11: An n-dimensional ACC C is called *homogeneously n-dimensional* if every k-dimensional cell of C with $k<n$ is incident with an n-cell of C.

Definition 12: A subcomplex of a two-dimensional ACC C, consisting of all cells of C incident with a 0-cell P of C is called an *umbrella* of P.

Appendix 2: Proofs of Propositions

Proof of Proposition 1: To prove the proposition we must demonstrate that:

1) The boundary of V is connected. This follows from the simply-connectedness of V.
2) The boundary cells incident with a boundary pixel compose a one-dimensional manifold (1-manifold);
3) There are two or four boundary pixels incident with each boundary crack. The pairs of adjacent pixels may be specified depending on the membership [Kov93] of the crack in the set V as explained below.
4) The boundary cells incident with a boundary point compose one or more subcomplexes (umbrellas) each of which is B-isomorphic to a 1-quasi-manifold.

Consider first a boundary pixel F. In a Cartesian 3D space it is bounded by four cracks and their four end points. These eight cells compose a 1-manifold. It is necessary to demonstrate, that all of them belong to the boundary of V. According to Definition 3, their SON's contain the SON of the pixel F as a subset. Since V is solid, it is impossible that the voxels bounded by F belong both to V or both to its complement. Thus the SON of F (consisting of F and two voxels) contains a voxel of

V and a voxel of the complement of V. So do the SON's of all the eight cells mentioned above. Thus they all belong to the boundary B and compose a 1-manifold.

A boundary crack C is incident with four voxels. They have four pixels incident with C. If only one of the four voxels is in V or the voxels being in V compose a strongly connected path, then exactly two of the four pixels are boundary pixels. They must be considered as adjacent. In the case when two diagonally adjacent voxels belong to V and the two other to the complement of V, all four pixels are boundary pixels. If the crack is considered as belonging to V, then two pixels bounding one and the same voxel of the complement must be considered as adjacent. Otherwise adjacent are pixels bounding one and the same voxel of V.

The most complicated situation we have with a boundary point P. In the Cartesian 3D space it is incident with 6 cracks and 12 pixels. Some of them belong to the boundary of V. Their number depends upon the location of the voxels of V incident with P. Let us take a boundary crack C_1 and a boundary pixel F_1 both incident with P. There exists exactly one crack C_2 different from C_1 which is incident with F_1 and P. Thus each pixel in the neighborhood SON(P, B) in the boundary B is incident with exactly two distinct cracks. According to the analysis of the preceding paragraph a boundary crack is incident with two or four boundary pixels. The cracks incident with P bound the pixels incident with P. This is the same situation as in a 1-quasi-manifold, where any crack is incident with two points and any point is incident with an even number of cracks. Thus the subcomplex consisting of cracks and pixels in the SON(P, B) is B-isomorphic to a 1-quasi-manifold.

Lemma 1: If a 1-manifold M_1 is a subset of another 1-manifold M_2 then the manifolds are identical.

Proof: Suppose the contrary. Then there must be in M_2 a pair of adjacent cracks the first one C_1 being in M_1 and the second C_2 not in M_1. The common end point P of C_1 and C_2 is both in M_1 and in M_2. Let C_3 be the second of two cracks of M_1 incident with P. According to the supposition, C_3 is not identical with C_2. Then C_3 is not in M_2, since according to the definition of a 1-masnifold the are exactly two cracks incident with P in M_2. A contradiction.

Proof of Proposition 2: Let L be a subset of a 2-quasi-manifold B while the boundary Fr(L,B) of L relative to B is a 1-manifold. Let us consider the union of L with the closure Cl(F) of a pixel F of B–L having a common crack with L and being simple with respect to L. The simplicity of F means that the intersection I of the boundary Fr(F,B) with L is connected. The intersection I contains at least one crack, namely the common one. The common cracks contained in I are no more boundary cracks of the union $L\cup$Cl(F). Only the cracks of the boundary Fr(F,B) which do not belong to the intersection belong to the boundary of the union. If there are no such cracks then the boundary Fr(F,B) lies completely in Fr(L,B). Both boundaries are 1-manifolds. According to Lemma 1 they are identical. Thus the boundary Fr($L\cup$Cl(F),B) becomes empty.

If there are cracks in Fr(F,B) not belonging to I then they replace the cracks of I in the new boundary of L. Since F is simple, this new part of the boundary of L is also connected and has no common cells with the old boundary of L besides two points of I. Thus the new boundary is a 1-manifold.

Proof of Proposition 3: To specify the genus G of B we first calculate its Euler number. B is a union of L and $B-L$. The Euler number of L is equal to 1 since L was developed from a single pixel by means of an operation which does not change the Euler number. According to Hypothesis of Section 3, the set $B-L$ consists of pixels incident with Fr(L,B) and cracks lying between these pixels. It is a stripe one pixel wide. It may be considered as a union of loops, like those shown in Fig. 6. The Euler number of a loop is zero since it consists of an equal number of pixels and cracks. If two loops intersect (compare Fig. 6) then they have P pixels and $P-1$ cracks in common (the cracks of the boundary Fr(L,B) do not belong to the loops). The count $P-(P-1)=1$ of these cells must be subtracted form the sum of the Euler numbers of the loop being zero. Thus each location containing an intersection of two loops reduces the Euler number by 1. On the other hand, at each intersection there is either one pixel with 4 adjacent pixels in the loops or two pixels with 3 adjacent pixels. Thus the number N_4 or half the number N_3, i.e. $N_3/2$, must be subtracted from the Euler number of the loops. Therefore the Euler number E of B is equal to $E=1-N_3/2-N_4$ and the genus G of B is equal to $G=(2-E)/2=(1+N_3/2+N_4)/2$.