Hierarchical topological structure for the design of a discrete modeling tool

Dexet M., Andres E.
SIC, Université de Poitiers
bât. SP2MI, Bvd Marie et Pierre Curie, BP 30179
86962 Futuroscope Chasseneuil Cedex, France
{dexet,andres}@sic.univ-poitiers.fr

ABSTRACT

In this paper, we present the design of a topological based geometrical modeler kernel. With this modeler, our goal is to create, import and handle geometrical objects represented both in a rasterized and a polygonal form. The aim is to provide a tool that mixes in a unified framework acquired discrete information (photos, MRI, ...) and synthetic continuous information (modeled objects). We propose the use of a multi-level hierarchical structure in which consecutive levels are linked. Each level of this structure corresponds to a particular representation of a same object. The lowest level is the raster representation and the highest level is the polygonal one. The way consecutive levels are computed, as well as the links between them are presented. Finally, we discuss the way the structure is updated using existing links in case of a modification applied on one level.

Keywords

Discrete geometry, geometrical and topological modeling, hierarchical structure, rasterization, polygonalisation.

1. INTRODUCTION

2D and 3D digital images are composed of discrete elements called *pixels* and *voxels*. Digital images can be generated, acquired (MRI, photos, ...) or can be the result of a rasterization (or *discretization*) process applied on a polygonal object (according to a discretization model). This data can be treated and manipulated using tools provided by *Discrete geometry*. One of the main problems we face when handling discrete information is that geometrical laws can be different in the discrete and in the continuous (Euclidean) world. For instance, some operations like rotations are simple in the Euclidean world while they present real difficulties in the discrete one. On the contrary, intersection of two objects is a simple operation in the discrete world and more difficult to perform in the Euclidean one. Having a discrete and a continuous representation of a same object can therefore have some interest. So far most existing modeling systems handle either vectorial based or raster information. Digital images can then be vectorized (using for instance 'Marching cubes' [LC87]) and polygonal objects can be rasterized, but no modeler proposes, to the best authors knowledge, to maintain both a polygonal and a raster representation of an object inside a same data structure. With this in mind, we present here a topology based modeling software that allows you to create, import or manipulate geometric objects in a continuous as well as in a discrete representation form. The particularity of this software is that discrete and continuous representations of a same object coexist inside a single hierarchical structure. It is thus possible, according to a given operation, to choose the most adapted representation. In a first approach [ABL01], objects where represented inside the modeling software with four different representations: continuous, discrete analytical, discrete contours and discrete. The discrete representations were all built based on the continuous one. Only the continuous representation could be modified and in this case, the other ones had to be entirely reevaluated. We now propose a new four level hierarchical structure. Each level corresponds to one of the previous representation forms. With this structure, each representation can be modified and all the others can be updated accordingly using reconstruction or discretization methods. The different consecutive levels in the hierarchical structure are

FULL Papers conference proceedings ISBN 80-86943-03-8
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

linked. These links allow to manipulate and propagate modifications locally. Updating the whole structure is thus not systematically needed. Of course such an architecture comes with a prize. The complexity of the hierarchical structure is much more complex than classical topology based modeling or imaging softwares. Indeed, we introduce an architecture that aims at keeping a topological and a geometrical coherence between multiple representation forms. Finally, our structure is designed in any dimension and our modeler is based on a 3D topological structure (it is constructed as an extension of a 3D topology based Euclidean modeler). In the following, we briefly recall the definitions and properties of the topological and discrete analytical model that forms the basis of our structure. We also present the discrete analytical recognition and reconstruction processes.

Recalls on generalized maps

A topological model is used to explicitly specify adjacency and inclusion relationships between the different cells (vertices, edges, faces and volumes for dimensions 0, 1, 2 and 3) of a geometrical object. Information, called *embeddings*, such as geometrical ones (for instance vertex coordinates) can be added to the model. The kernel of our modeler is based on *generalized maps* [Lie94] which is a combinatorial model that allows the representation of cellular quasi-manifolds, orientable and non-orientable, with or without boundaries. A generalized map is a set of abstract elements, called *darts*, and applications on these darts. The definition of an n-dimensional generalized map, also called *n-G-maps*, is the following one:

DEFINITION 1 (n-G-MAP). *Let $n \geq 0$. An n-dimensional generalized map is $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$ where :*

- *D is a finite set of darts;*
- *$\forall i,\ 0 \leq i \leq n$, α_i is an involution[1] on D;*
- *$\forall i, j,\ 0 \leq i < i+2 \leq j \leq n$, $\alpha_i \circ \alpha_j$ is an involution.*

In an *n*-G-map, each topological cell of dimension *i* corresponds to a subset of darts called *orbit*. Such an orbit is denoted $<\alpha_0, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n>$ and corresponds to the set of darts that can be reached starting from a dart, using any combination of involutions except α_i. A cell of dimension i (or *i-cell*) is defined as follows:

DEFINITION 2 (*i*-CELL). *Let $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$ be an n-G-map, d a dart and $i \in N = \{1, \ldots, n\}$. The i-cell incident to d is the orbit $<>_{N-\{i\}}(d) = <\alpha_0, \ldots, \alpha_i - 1, \alpha_i + 1, \ldots, \alpha_n>(d)$.*

[1] An involution f on S is a one to one mapping from S onto S such that $f = f^{-1}$.

For example, in a 2-G-map, the orbits $<\alpha_1, \alpha_2>$, $<\alpha_0, \alpha_2>$ and $<\alpha_0, \alpha_1>$ correspond respectively to topological vertices, edges and faces. We can see in Figure 1 the decomposition of a topological object in cells. The object in Figure 1a can be decomposed into topological faces linked by an α_2 involution (see Figure 1b). In the same way, a face can be decomposed into topological edges linked by an α_1 involution (see Figure 1c). To finish, an edge can be decomposed into two darts which are linked by an α_0 involution (see Figure 1d).

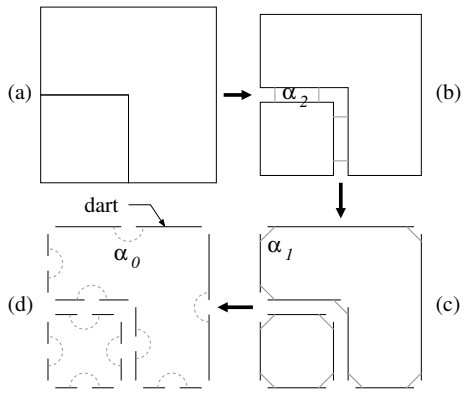


Figure 1: Topological decomposition of an object. (a) A 2D Euclidean object. (b) Its decomposition into topological faces. (c) Faces decomposition into topological edges. (d) Edges decomposition into darts.

In the following, in order to simplify figures, we will represent 2-G-maps as shown in Figure 2.

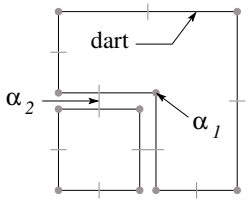


Figure 2: Representation of a 2-G-map.

The discrete standard analytical model

Our modeler allows to compute a discrete object from a given continuous object. In 2D (resp. 3D), the resulting discrete object must define 4-connected (resp. 6-connected) contours, in order to obtain a space subdivided in 4-connected (resp. 6-connected) regions. The discretization model we use is the *standard model* [And03]. We chose this model because it's the only model that allows the 4-connected analytical discretization of 2D objects (vertices and edges of polygons) and the 6-connected discretization of 3D objects (vertices, edges and faces of polyhedra). Moreover, this model is defined for any dimension.

In what follows, we focus on the 2D case. We first give the definitions of a discrete standard line and a discrete standard point. Then we explain how to compute the discrete analytical description of a segment. In [And03] the reader will find details about standard objects in 3D and higher dimensions.

DEFINITION 3 (2D STANDARD LINE). *A discrete standard line of parameters (a, b, c) is the set of integer points (x, y) verifying $-\omega \leq ax + by + c < \omega$, with $\omega = \frac{|a|+|b|}{2}$ and $a > 0$ or $a = 0$ and $b \geq 0$.*

DEFINITION 4 (2D STANDARD POINT). *Let P be a real point of coordinates (i, j). The standard discretization of P is the unique integer point (x, y) verifying $i - \frac{1}{2} \leq x < i + \frac{1}{2}$ and $j - \frac{1}{2} \leq y < j + \frac{1}{2}$.*

Figure 3 shows the standard discretization of a Euclidean segment (S in the figure).

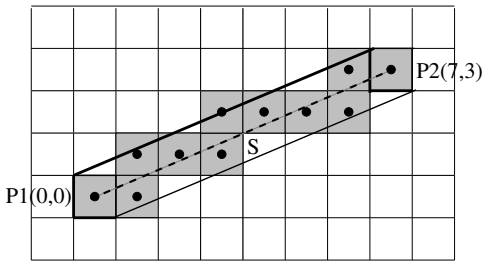


Figure 3: A standard segment and its analytical description. Pixels in gray correspond to the standard discretization of S (bold dotted line), i.e. pixels crossed by S. Bold (resp. thin) lines correspond to large (resp. strict) inequalities of the discrete analytical description.

The analytical description of S is deduced from the analytical descriptions of its end points $P1$ and $P2$, in addition to the analytical description of the line including S. These inequalities are:

- For the line: $-5 \leq 3x - 7y < 5$
- For P1: $-\frac{1}{2} \leq x < \frac{1}{2}$ and $-\frac{1}{2} \leq y < \frac{1}{2}$
- For P2: $7 - \frac{1}{2} \leq x < 7 + \frac{1}{2}$ and $3 - \frac{1}{2} \leq y < 3 + \frac{1}{2}$

The discrete analytical description of S is then composed of the following 6 inequalities (due to the removal of 4 inequalities):

$$\begin{cases} -5 \leq 3x - 7y < 5 \\ -\frac{1}{2} \leq x \\ -\frac{1}{2} \leq y \\ x < 7 + \frac{1}{2} \\ y < 3 + \frac{1}{2} \end{cases}$$

Note that two discrete spaces can be used to obtain discrete objects: the *classical discrete space* in which *discrete points* (i.e. integer coordinates points) are pixel centers (see Figure 4a), and the *dual discrete space* in which discrete points are pixel vertices (see Figure 4b). In our modeler, the discretization process is done in dual space, because we want to discretize the Euclidean region contours in order to obtain discrete region contours. The discretization of a polygon in the dual space is shown in figure 4b. Figure 4c shows the discretisation of two adjacent polygons.

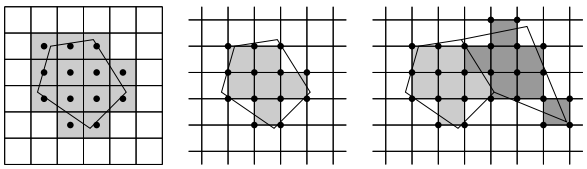


Figure 4: Standard discretization in the classical and dual discrete spaces. Black points are discrete ones. (a) Classical discrete space. (b) and (c) Dual discrete space.

Recognition and reconstruction processes

The discrete *recognition* process determines if a set of discrete points belongs or not to a given discrete object. The 2D recognition determines if a set of pixels belongs to a 2D discrete line. Figure 5a shows an example of discrete segment recognition. Adjacent pixels with the same color in the figure belong to the same discrete standard segment.

We call discrete *reconstruction* the operation that consists in obtaining a continuous object from a discrete one. This operation corresponds to the reverse of the discretization one, i.e. the discretization of the obtained continuous object corresponds to the original discrete object. Thus, the reconstruction of a 2D discrete segment is a Euclidean segment so that its discretization corresponds to the discrete segment. We see in Figure 5b an example of reconstructed segments.

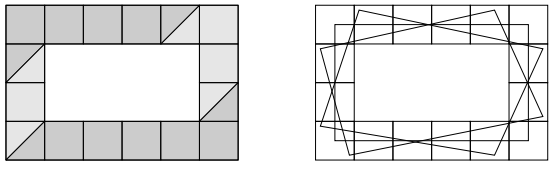


Figure 5: Discrete segment recognition and reconstruction. (a) Recognized discrete segments. Bicolor pixels belong to two discrete segments. (b) Three possible reconstructions for a same discrete curve.

Note that the reconstruction of an object is not unique (see Figure 5b). Indeed, an infinity of Euclidean objects have the same discretization. However, a reconstruction choice can be imposed, given a specific reconstruction method, choosing a starting point position (we choose the point with lowest abscissa and among

them the one with lowest ordinate) and a direction in which doing it (in our application, we choose the clockwise direction).
In the following, we describe our structure in dimension 2. In Sections 2 and 3, the different levels of the hierarchical structure are described, and the way each level is computed from a neighbor level is detailed. In particular, we focus on the computation of the links between two consecutive levels. Section 4 explains how those links are used to update the structure when a modification of a representation is performed. A short conclusion and perspectives are presented in Section 5.

2. MULTI-LEVEL STRUCTURE DESCRIPTION

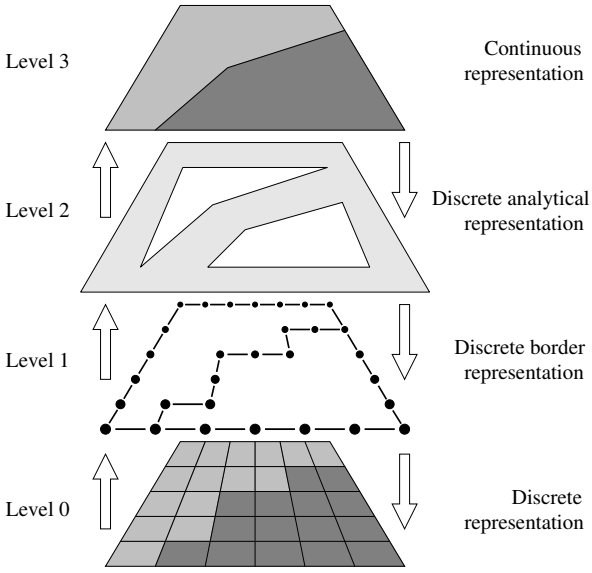


Figure 6: The hierarchical structure. Each level corresponds to a specific representation and is linked to its neighbors.

Our structure consists of four inter-dependent levels (see Figure 6): the discrete level, the discrete border level, the discrete analytical level and the continuous level. Each level corresponds to a particular representation of a same object. More precisely, a same primitive is represented either in a continuous form, and in three different discrete forms. Discrete and continuous levels are the end levels of the structure. The addition of two intermediate levels allows a progressive evolution from the discrete object to the Euclidean one, and vice versa. Each level of the structure is a generalized map. Information are associated with specific orbits of some levels in order to define the geometrical shape of the primitive.
In the following, we detail each level of this structure in dimension 2, i.e. its corresponding 2-G-map and embeddings. We illustrate our purpose by a simple example.

Level 0: the discrete level

This level corresponds to the classical discrete representation which is composed of pixels. The discrete space considered here is the dual one (see Section 1.2). The 2-G-map of this level corresponds to a space subdivision into pixels (see Figure 7a). Each pixel is represented by a square topological face associated with a color[2] embedding (see Figure 7b). Moreover, integer coordinates are attached to each topological vertex. The object in this level may be an imported image or the result of the discretization process applied on level 3.
However, the G-map of this level requires a lot of memory space, and particularly in 3D. For example, the memory space required for a 2D digital image of 512^2 pixels is at least 72 MB and for a 3D digital image of 512^3 voxels is 216 GB. In order to minimize the required memory space, the G-map of this level is coded in a more compact way as follows: embeddings are stored inside a matrix, and α_i involutions are not explicitly stored but recomputed from the neighborhood relations between the pixels. This is possible because pixels have a very simple topological form and adjacency relations are implicite. In the same way, connections between darts of level 0 and level 1 are simulated. This optimization brings the size of a 2D digital image of 512^2 pixels to less than 1 MB and the size of a 3D digital image of 512^3 voxels to 384 MB.

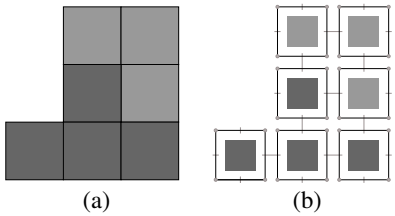
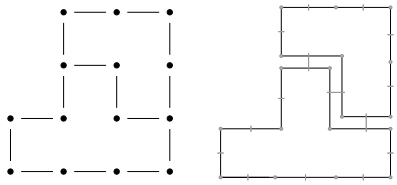


Figure 7: The discrete level. (a) Discrete view: here, a grayscale digital image. (b) Discrete 2-G-map: each topological face and each topological vertex are respectively associated with grayscale and integer coordinates embeddings.

Level 1: the discrete border level

This level corresponds to the contours obtained for each uniform colored 4-connected region. The representation of these contours is based on the *interpixel* model [KKM90, Kov89] (see Figure 8a). Each discrete point is represented by a point and two successive points are linked by a segment. This representation simplifies the coverage of level 0 regions boundaries. The 2-G-map of this level corresponds to a space subdivision into regions (see Figure 8b). There is no embedding at this level and geometrical information needed for the visualization of the level are located in level 0 and can be accessed from level 1 (see Section 4.1).

[2]This embedding can also be a label.

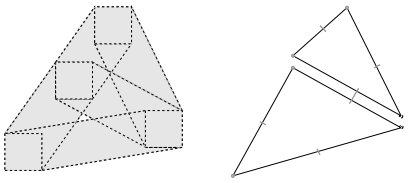


(a) (b)

Figure 8: The discrete border level. (a) Discrete border view. (b) Discrete border 2-G-map.

Level 2: the discrete analytical level

This level is an implicit representation of the discrete border primitive. It corresponds to the discrete analytical description of the level 1 region contours (see Figure 9a). More precisely, each contour is described as a discrete analytical polygon computed according to the discrete analytical standard model (see Section 1.2). The main interest of this representation is that it doesn't depend on the number of discrete points of the object. The 2-G-map of this level corresponds to a space subdivision into regions (see Figure 9b), in which each topological edge (resp. vertex) corresponds to a discrete analytical segment (resp. point). Each discrete segment (resp. vertex) is described by two (resp. four) opposite inequalities. These inequalities are associated to the topological edges and vertices of the level.



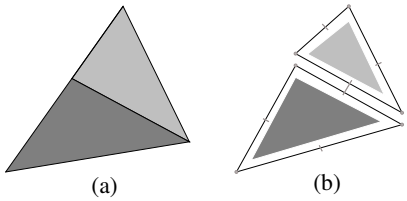
(a) (b)

Figure 9: The discrete analytical level. (a) Discrete analytical view. Gray areas are delimited by the analytical inequalities. (b) Discrete analytical 2-G-map. Inequalities are associated to each topological edge and each topological vertex.

Level 3: the continuous level

This level is an explicit representation. In this level, each region is described as a colored Euclidean polygon in the classical boundary representation form (see Figure 10a). The primitive of this level may be created using the tools available inside the modeler, or may be the result of the reconstruction process applied on level 0 (see Section 1.3). 2D Euclidean vertex coordinates and face colors are associated to the 2-G-map of this level (see Figure 10b).

For this level to be coherent with the others, we must ensure that the discretization (according to the standard model) of the Euclidean polygons is equal to the regions in level 1.



(a) (b)

Figure 10: The continuous level. (a) Continuous view. (b) Continuous 2-G-map with color and real coordinates embeddings.

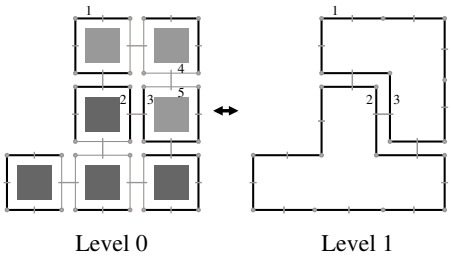
3. CONSTRUCTION AND LINKS BETWEEN THE LEVELS

Each level of the hierarchical structure corresponds to a step of the reconstruction or discretization process. In this section, we describe the way each level is built. To build the structure starting from level 0 or from level 3, topological operations are required. We won't describe them is this paper (see [DDAL05] for more details). Since our goal is to quickly reflect a modification of a level of the structure on the others, we set up bidirectional connections between the different levels, and more precisely between the darts of each level. The advantage of these connections is that all structure's embeddings become accessible for all levels. The information redundancy inside the structure is thus limited. In the next sections we explain how the different levels are built and connected.

Links between level 0 and level 1

- Level 1 is obtained from the discrete primitive of level 0 by merging pixels of same color.
- To build level 0 from level 1, we need to reconstruct the pixels, for example by using a flood-fill algorithm on a matrix corresponding to Level 1.

As shown in Figure 11, links are established between the darts of level 1 and the corresponding darts in level 0.



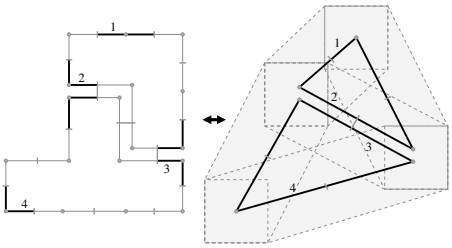
Level 0 Level 1

Figure 11: Links between level 0 and level 1. Black darts are connected ones. For example, darts numbered 1 (resp. 2 and 3) in the two representations are connected. Darts numbered 4 and 5 are not connected with darts of level 1. Here, there are exactly 36 links between the two levels.

Links between level 1 and level 2

- To obtain level 2 from level 1, we use an *analytical recognition* operation (see Section 1.3). The recognition algorithm we use is described in [BSDA03]. During the analytical recognition step, all vertices that belong to the same discrete segment are removed in order to keep only the two edge extremities. Removed darts are level 1 darts not connected with level 2 darts in Figure 12.
- To build level 1 from level 2, we need to discretize the contours of level 2 according to the standard model (see Section 1.2). Each edge of level 2 is incrementally discretized from one end point to the other, and a new vertex is inserted into the edge while the other end point is not reached.

For each edge of level 2, links are established between the darts of the edge and the extremity darts that belong to the corresponding discrete segment in level 1 (see Figure 12).



Level 1 Level 2

Figure 12: Links between level 1 and level 2. Black darts are connected ones. For example, darts numbered 1 (resp. 2, 3 and 4) in the two representations are together connected. Here, there are exactly 12 links between the two levels.

However, some problems can arise such as degenerated vertices (see Figure 13b). These vertices disappear in the discrete border level due to the loss of information inherent to the discretization process. We choose to delete these vertices (see Figure 13c), and move the links existing with these darts (numbered darts in Figure 13d) to keep the levels coherent.

Links between level 2 and level 3

- To build level 3 from level 2, we make a copy of level 2, and compute the Euclidean coordinates embeddings during the *analytical reconstruction* step (see Section 1.3). Reconstruction algorithms and details on particular cases that can occur during the analytical reconstruction are presented in [BSDA03] and [SBDA05].
- To build level 2 from level 3, we also make a copy of level 3, but we can eventually make some simplifications. For example, the gray square in Figure 14a contains vertices having the same standard discretization. All corresponding topological vertices, except the two extremities, can thus be removed (see Figure 14b).

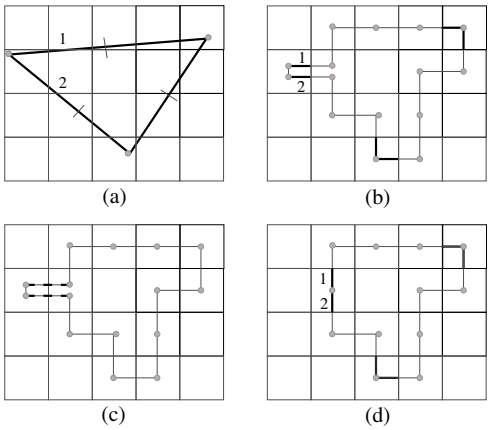


Figure 13: Example of degenerated vertex. (a) Level 2 primitive. (b) Primitive computed according to the standard discretization process. (c) Dashed darts are removed. (d) Level 1 primitive obtained. Note that black darts are connected ones.

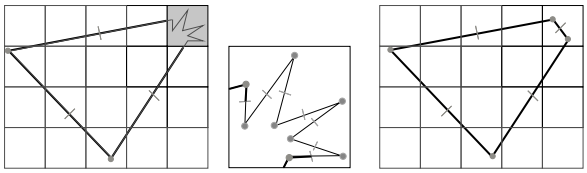


Figure 14: Simplification of the discrete analytical level. (a) Euclidean object. (b) The darts in gray are removed. (c) Discrete analytical level. Note that all black darts are connected ones.

Each dart of level 2 is connected with a dart of level 3 (see Figure 15). However, as in the example in Figure 14, some darts of level 3 may not be connected.

4. HIERARCHICAL STRUCTURE UPDATE

Information access

Stored information can be accessed using the links between consecutive levels. For example, integer coordinates stored at level 0 can be accessed from level 3 using successively the links between level 3 and level 2, the links between level 2 and level 1 and the links between level 1 and level 0. The main difficulty is to find linked darts between the current level and the following one. Indeed, all darts are not connected. For instance, some darts in level 1 are not connected with darts of level 2 (see Figure 12). Lets consider a face F of level 1. If we want to know the color associated to the face of level 3 corresponding to the reconstruction of F, we first must search for a connection between a dart of F and the

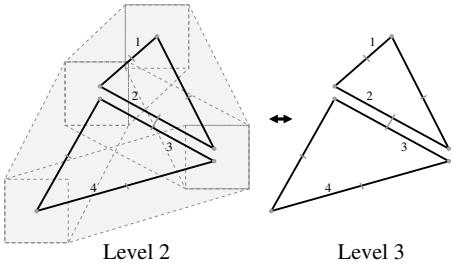


Figure 15: Links between level 2 and level 3. Black darts are connected ones. For example, darts numbered 1 (resp. 2, 3 and 4) in the two representations are together connected. On this example, there are exactly 12 links between the two levels.

corresponding face of level 2. The color embedding can then be reached using links between level 2 and level 3 since each dart of level 2 is connected with a dart of level 3.

Operations and updates on the structure

With our modeler, we can apply operations as well on the continous primitives as on the discrete ones. The most classical operation that can be used in level 0 is changing the color of one or several pixels. Many discrete operations can be based on it. Level 3 is based on a 3D Euclidean geometrical modeler called *Moka*[3] in which a lot of geometric operations have been already developed. Classical operations like translations, rotations, scales, ... or more elaborated operations like corefinement can be performed. In case of complex primitives (i.e. primitives composed of a great number of elements), updating all the structure consumes too much time. However, a level does not need to be totally recomputed when only a local modification is done. Moreover, it is often advisable to keep some details like those shown in figures 14a and 14b. In this case, a whole update of the structure will involve the loss of these informations. It is preferable to use the links between levels in order to reflect local modifications.

In our structure, when a level is modified by an operation, other levels may be updated. In this section, we show two different examples of operations on the structure: a discrete one and a continous one, and the way updates are computed.

4.2.1 Discrete operation: segmentation

The structure we consider here is based on a grayscale digital image (see Figure 16). We locally modify the segmentation of the digital image by filling the two middle regions in order to obtain only two regions. Thus, we can determine which darts must be removed in the other levels (darts connected – directly or not – with black ones in Figure 17). Indeed, in the original digital image, these darts where surrounded by two faces associated with different colors. After the edits, these darts are surrounded by two faces with the same color. Using the links between level 0 and level 1, we can easily find the darts of level 1 to be removed. And so on for level 2 and level 3. Here, the main interest of using links of the structure is that we can make local modification without recomputing the entire structure.

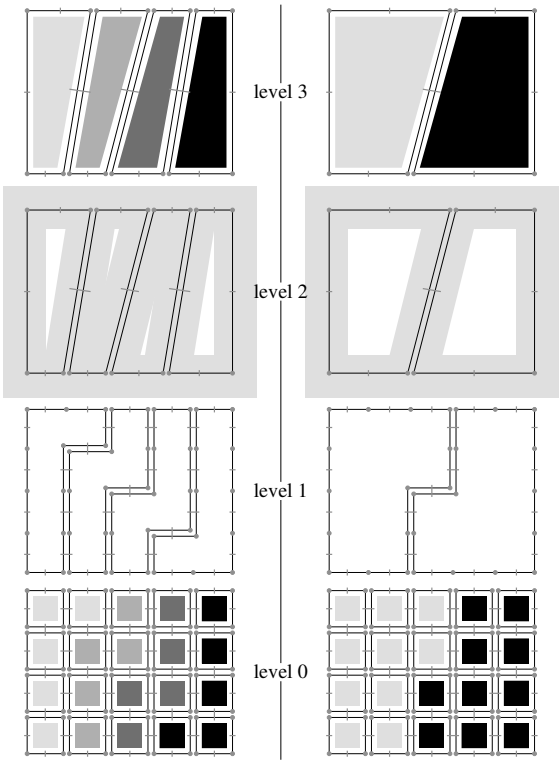


Figure 16: Our structure before and after color changes. On the left, the original digital image and the corresponding struture. On the right, the same structure after updates due to color changes.

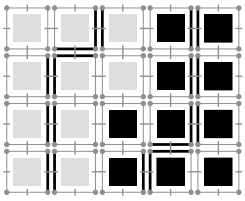


Figure 17: Darts concerned by the modification (in black).

4.2.2 Continuous operation: translation

The structure we consider here is based on a continous primitive (see Figure 18). We apply a non integer translation on level 3 (see Figure 19). We can see that the topology of the level 2 and 3 are not touched. Existing links between these two levels thus don't need to be changed. Nevertheless, we must modify the topology of the level 1 according to the discretization of level 2.

[3]Moka web site: http://www.sic.sp2mi.univ-poitiers.fr/moka

Some discrete segments can be kept if their discretization is still the same. Finally, level 0 has to be entirely recomputed (as well as links with level 1) because of the complexity of necessary local updates. We have thus seen that a global modification on level 3 may not require the structure to be entirely recomputed.

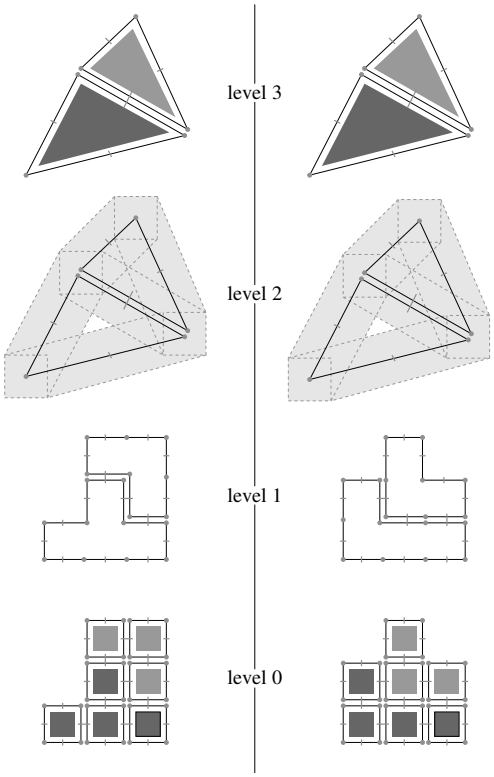


Figure 18: Our structure before and after level 3 primitive translation. On the left, the original Euclidean object and the corresponding structure. On the right, the same structure after updates due to the translation.

5. CONCLUSION

In this paper we have presented a modeler kernel based on a multi-level hierarchical structure. Each level corresponds to a particular representation of a same object: continuous, discrete analytical, discrete border of regions and discrete representations (see attached colored images). Each level is linked with the level above and below it. This ensures the coherence between all the representations. The way each level is computed from a neighbor one, as well as the way links are established between consecutive levels are presented. Two examples of operations on discrete and continuous levels are also detailed. As a short term goal, we plan to develop several operations and study more in details the propagation of local modifications inside the structure in order to optimize the updates even farther. Furthermore, a 3D extension is planned. Indeed, for the moment, our modeler deals only with 2D primitives because the 3D discrete analytical reconstruction operation, which aim is to provide a more compact reconstruction than the Marching cubes method, is still under developpement. A 3D version of our software should be available soon.

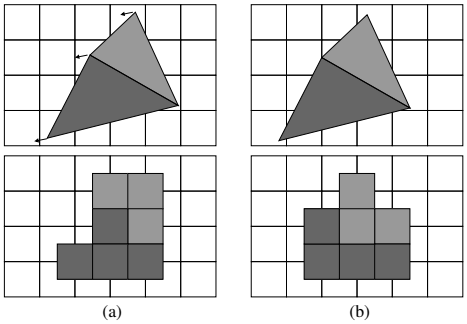


Figure 19: Discrete level before and after level 3 primitive translation. (a) Original continuous and discrete levels. (b) Continuous and discrete levels after translation.

6. REFERENCES

[ABL01] E. Andres, R. Breton, and P. Lienhardt. Spamod: design of a spatial modeling tool. *Digital and Image Geometry*, LNCS 2243:91–107, 2001.

[And03] E. Andres. Discrete linear objects in dimension n: the standard model. *Graphical Models*, 65:92–111, 2003.

[BSDA03] R. Breton, I. Sivignon, F. Dupond, and E. Andres. Towards an invertible euclidean reconstuction of a discrete object. In *Discrete Geometry for Computer Imagery*, volume LNCS 2886, pages 246–256, Naples, Italy, 2003.

[DDAL05] G. Damiand, M. Dexet, E. Andres, and P. Lienhardt. Removal and contraction operations to define combinatorial pyramids: application to the design of a spatial modeler. *Image and Vision Computing*, 23(2):259–269, 2005.

[KKM90] E. Khalimsky, R. Kopperman, and P.R. Meyer. Boundaries in digital planes. *Journal of Applied Mathematics and Stochastic Analysis*, 3:27–55, 1990.

[Kov89] V.A. Kovalevsky. Finite topology as applied to image analysis. *CVGIP*, 46:141–161, 1989.

[LC87] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. In *SIGGRAPH*, volume 21 of *Computer Graphics J.*, pages 163–169, Anaheim, USA, 1987.

[Lie94] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3), 1994.

[SBDA05] I. Sivignon, R. Breton, F. Dupond, and E. Andres. Discrete analytical curve reconstruction without patches. *Image and Vision Computing*, 23(2):191–202, 2005.