DETERMINISTIC AND HEURISTIC METHODS IN 2-MANIFOLD RECONSTRUCTION

SEBASTIAN PEÑA SERNA

EAFIT UNIVERSITY
MECHANICAL ENGINEERING DEPARTMENT
DESIGN AREA
MEDELLÍN
2004

DETERMINISTIC AND HEURISTIC METHODS IN 2-MANIFOLD RECONSTRUCTION

SEBASTIAN PEÑA SERNA

Final Project presented to obtain the B.Sc. Diploma in Mechanical Engineering

Adviser
OSCAR EDUARDO RUIZ S.
Ph.D. Mechanical Engineering

EAFIT UNIVERSITY
MECHANICAL ENGINEERING DEPARTMENT
DESIGN AREA
MEDELLÍN
2004

Medellín, December 10, 2004

Acceptation Note

_____

_____

_____

_____

Jury: Dr. Fernando Pérez Fontán

_____

Jury: Dr. Manuel Julio García R.

_____

Adviser: Dr. Oscar Eduardo Ruiz S.

To my family

ACKNOWLEDGEMENTS

I would like to thank to my family Adriano, Yolanda und Carolina, for supporting and accompanying my development such a person and such a professional.

I want to thank for the people of the CAD / CAM / CAE Laboratory at EAFIT University, in special my tutor and adviser Prof. Dr. Eng. Oscar Ruiz, by its patience and lessons distributed during my formation. Like with the professors Manuel Julio García, Carlos Cadavid and Carlos Lopez, and my companions who always listened and understood to me.

Finally, I would like to give and a special acknowledgement to Prof. Dr. Eng. Fernando Pérez Fontán to believe and to trust in me, and to offer me the possibility of working with him and his group, in the Department of Signal Theory and Communications at University of Vigo.

CONTENT

LIST OF FIGURES

LIST OF ALGORITHMS

page

GLOSSARY

The source of some definitions included here are in [Ruiz, 2002], [Schlumberger, 2004], [Fact Index, 2004], [Farlex, 2004] and [Outfo, 2004].

Manifold: is a topological space that looks locally like the Euclidean space R^n and is a Hausdorff space. A connected manifold has a definite dimension, the number of coordinates needed in each local coordinate system. An example is the surface of a sphere, which is not a plane, but small patches of it are homeomorphic to (i.e., topologically equivalent to) patches of the Euclidean plane.

Geometry: is defined as the study of points, lines, angles, shapes, their relationships, and their properties. Geometry is closely tied with various coordinate systems, since geometric attributes can best be defined via their location on a coordinate system.

Topology: is the study of the so-called topological properties of figures, that is to say properties that do not change under bicontinuous one-to-one transformations (called homeomorphisms).

N-D: N dimensional Euclidean space.

B-REP: Boundary Representation of a Body. It requires a strict hierarchy of geometric and topologic entities (geometries like: surface, curve and points, and topologies like: body, lump, shell, face, loop, edge and vertex).

Π_i: is the i-th sampling plane with normal the axis Z = (0, 0, 1).

A, B, C, …: simple non-intersecting closed contour on plane Π_i.

1, 2, 3, …: simple non-intersecting closed contour on plane Π_{i+1}.

S_i: set of simple closed contours on plane Π_i.

$X \subset Y$: X is contained in Y, $X \not\subset Z$ if $Z \subset Y$.

R: solid region made up of contour A and it holes (B, C, D, …), (B, C $\wedge$ D) $\subset$ A.

T_i: is a hierarchy organization (tree)of S_i.

F_i: set of T_i on plane Π_i.

$G_{k,i}$: is the k-th set of planar regions on plane Π_i.

mg_k: is the k-th Mapping Group made up of the k-th set of planar regions on plane Π_i ($G_{k,i}$) and the k-th set of planar regions on plane Π_{i+1} ($G_{k,i+1}$).

Area( ): is the planar region bounded by the boundary of the solid region. The sign of the area is positive if the contour A is CCW-oriented with respect to the vector Z= (0, 0, 1), otherwise it is negative.

MG_i^{j+1}: set of all Mapping groups between the cross section i and i+1.

M_i^{j+1}: is a 2-manifold with borders (possible disconnected) between the cross-section on plane Π_i and Π_{i+1}. The borders are S_i and

S_{i+1} respectively.

n-handle: is a topological event which affect the surface between consecutive cross-sections. This event may be (i) 0-handle ($f(x, y) = x^2 + y^2$), (ii) 1-handle ($f(x, y) = x^2 - y^2$), or (iii) 2-handle ($f(x, y) = - x^2 - y^2$).

C^i*:* means continuity up to the i-th derivative.

$C_i(u)$*:* is a Piecewise Linear approximation of a planar curve, considered in a coordinate system in which the planar curve will have constant Z_i value.

1. INTRODUCTION

The topologic space 2-manifold may be considered a surface in R^3. It may have either border or no. If it does not have border, it is a closed 2-manifold and it bounded a solid. If it has border, it is an open 2-manifold that represents a surface or it could be joint with other 2-manifolds to form a solid.

The 2-manifold is used in many engineering and medical applications. Common engineering applications are: digitization reconstruction, engineering design, reverse engineering, numerical analysis, prototyping, manufacturing and robotics, among others. Medical applications are: human or animal organs reconstruction from X-Ray, virtual and robotic surgery, and virtual medical visualization, among others. These applications may be achieved with different techniques such as deterministic and heuristic methods for processing 2-manifolds for computer applications.

Deterministic methods are techniques, which use equations or algorithms that have been previously developed for similar situations. Deterministic methods are generally easier and faster to apply in computer applications. Related methods applied in this project are: Voronoi and Delone methods, 2-D Boolean operations and 2-manifold Boolean operations.

Heuristic methods are techniques, which provide a way to approach difficult combinatorial optimization problems. Combinatorial search gives a method to construct possible solutions and find the best one, given a

function that measures how good each candidate solution is. Heuristic methods such as genetic algorithms and neural networks provide general ways to search for good and optimal solutions but not the best. Related methods applied in this project are: 2-D shape similarity for creating mapping groups and hole classification and best plane of face similarity for merging faces and fitting satellite data.

These methods are implemented in many applications where the computational processing of data is necessary to obtain CAD models. These are some of the fields where it is used with more relevance: (a) reconstruction of medical images from planar sampling, (b) digitalized models for Finite Element Analysis (FEA), (c) Geographic Information Systems (GIS), (d) electromagnetism, (e) Ray-Tracing (RT) and (f) robotics simulation, among others.

This project presents two applications of deterministic and heuristic methods in 2-manifold reconstruction:
(i) 2-D Shape Similarity for Surface Reconstruction from Slice Sample data.
(ii) Coupling Terrain and Building data 2-Manifold for Ray-Tracing applications.

The present project is based on two papers:
(i) Title: “2D Shape Similarity as a Complement for Voronoi-Delone Methods in Shape Reconstruction”
Authors: Oscar Ruiz, Carlos Cadavid, Miguel Granados, Sebastian Peña and Eliana Vásquez, from the CAD / CAM / CAE Laboratory of the EAFIT University, Medellín, Colombia.
Submitted to: Elsevier Journal on Computer & Graphics.

Title: “Usage of 2D Region Similarity for Surface Reconstruction from Planar Samples”

Authors: Oscar Ruiz, Carlos Cadavid, Miguel Granados, Sebastian Peña and Eliana Vásquez, from the CAD / CAM / CAE Laboratory of the EAFIT University, Medellín, Colombia.

Submitted to: SIAM, Conference on Geometric Design and Computing. November 10-13, 2003. Seattle, Washington, USA.

(ii) Title: “Coupling Terrain and Building Database Information for Ray-Tracing Applications”

Authors: Fernando Pérez Fontán from the Department of Signal Theory and Communications ETSE of the University of Vigo, Spain, and Oscar Ruiz and Sebastian Peña, from the CAD / CAM / CAE Laboratory of the EAFIT University, Medellín, Colombia.

Submitted to: ClimDiff, Conference on Coupling Terrain and Building Database with Propagations Models for Loss Predictions. November 17-19, 2003. Fortaleza, Brazil.

2. HOW TO READ THIS DOUMENT

This document has two main topics in the context of Deterministic and Heuristic Methods in 2-Manifold Reconstruction.

The first main topic presented shows a process to handle parallel planar cross-sections with the purpose of pre-processing contours, creating and pre-processing mapping groups. The post-processing of mapping groups has the aim of recognizing possible topologic problems and solving them to obtain a correct set of 2-manifolds with border.

The second topic presented shows a development for achieving single data of building and terrain for Ray-Tracing applications. It presents a complete way to pre-process, process and post-process the terrain and building even obtaining a unique model available to run a Ray-Tracing application on it.

3. 2-D SHAPE SIMILARITY

3.1 CONTEXT

Handling surface reconstruction methods with a set of points collected in parallel cross sections of a solid could be geometrical or topological. The geometrical methods ([Barequet and Sharir, 1996], [Boissonat and Geiger, 1993]) use neighborhood information and metric considerations of points and regions to build the surface. The topological methods ([Fomenko and Kunii, 1997], [Shinagawa et al., 1991]) recognize topological events that influence the surface between consecutive cross sections.

The presented algorithm attacks the problem of surface reconstruction from cross section samples using the point of view of the evolution of the cross sections of the 2-manifold to be recovered. Using 2-D shape similarity, inferences on the topological events that take place between consecutive cross-sections. The match of 2-D similar composed shapes also helps to steer the application of well known Voronoi-Delone-based algorithms, which are effective in many cases, but have the disadvantage of building overstretched branches or bridges between 2-D regions of the plane Π_i and unrelated ones on the plane Π_{i+1}.

The algorithm presented here succeeds in avoiding such overstretched or overslanted surfaces, and therefore represents a step forward in ensuring both geometrical and topological faithfulness between the object and the reconstructed model, while the Voronoi-Delone-based methods only ensure geometrical similarity.

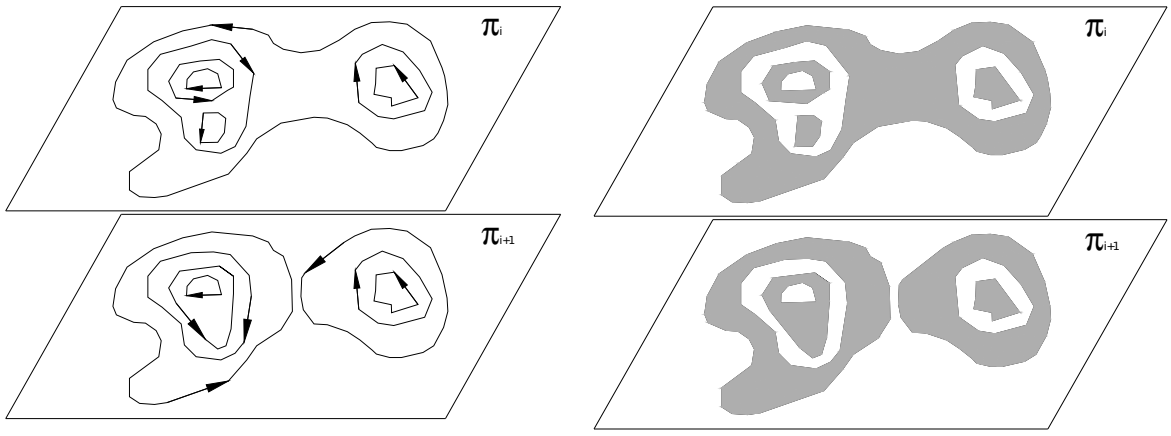
This project presents the 2-D shape similarity method for pre-processing and dividing S_i and S_{i+1} into subsets. Every subset represent regions in cross-sections i and i+1 which are globally similar and will be referred as mapping groups. With these mapping groups is possible to build the surface between the cross sections i and i+1. Each subset could be used as input with any algorithm from [Barequet and Sharir, 1996] or [Boissonat and Geiger, 1993]. A domestic implementation of the Boissonat & Geiger algorithm was developed in the CAD / CAM / CAE laboratory, which will be referred as the BG( ) algorithm. The M_i^{i+1} surface is the union of the results in sequential calls to the BG( ) algorithm.

3.2 PROCESSING OF 2-D SIMILARITY MAPPING GROUPS

3.2.1 Background

In surface reconstruction from planar cross sections it is necessary to build surfaces between 2-D contours in consecutive cross-sections (see Figure 1).

Figure 1. Consecutive 2-D planar cross section on planes Π_i and Π_{i+1}.



(a) Planar cross section contour orientation.

(b) Solid regions of the planar cross-section.

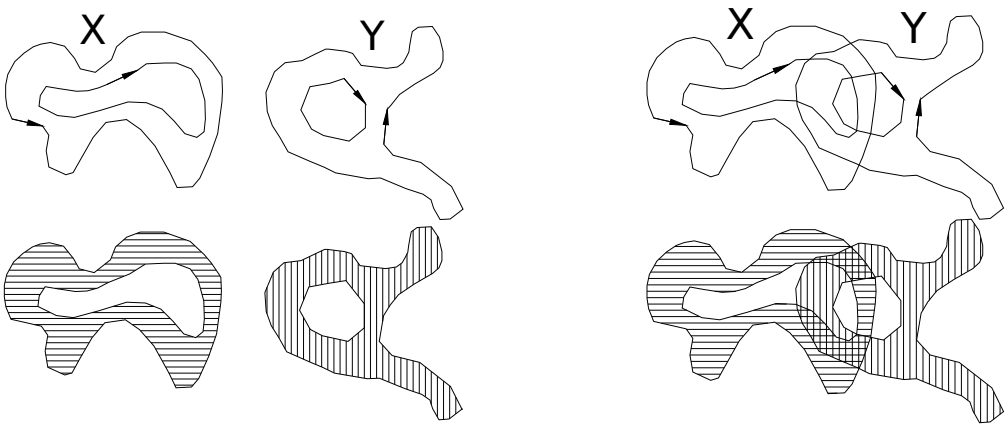
Π_i

The implemented algorithm pre-processes the S_i and S_{i+1} contour sets by identifying subsets $G_{k,i} \subseteq S_i$ and $G_{k,i+1} \subseteq S_{i+1}$ which represent 2-D similar regions, because of $G_{k,i+1}$ is considered as the evolution of the region $G_{k,i}$ along the sampling axis Z = (0,0,1). The sets $G_{k,i}$ and $G_{k,i+1}$ are in general disconnected, they must approximately face each other. To create the mapping groups and then to build the surface, it is need to explain 2-D Boolean operations, 2-D matching and surface building as fallow:

3.2.1.1 2-D Boolean Operations: The term Boolean operation comes from its inventor, George Boole (1815-1864), who came up with a way to combine logic elements using operators called AND, OR, NOT, IF, and THEN, among others.

Typical 2-D Boolean operations between solid regions (R) are: union ($\cup$) intersection ($\cap$) and subtraction (-) ([Murta, 1999]). Given X and Y (Figure 2), two set of planar regions with a correct orientation, it is possible to find a common set of solid regions, if they are overlapped (Figure 2).

Figure 2. Planar region X and Y in 2-D space.

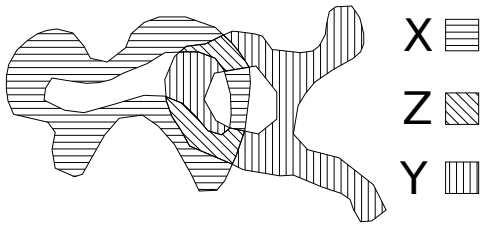


(a) Contour orientation and solid region X and Y.

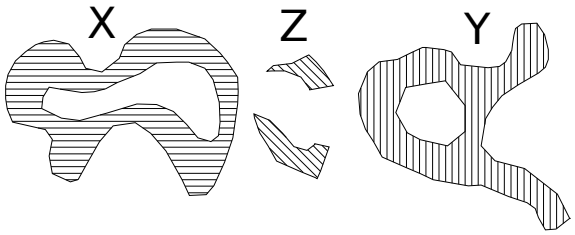
(b) Contour orientation and solid region X and Y overlapped.

The common set of solid regions (Z) determines the result of the Boolean operation (Figure 3).

Figure 3. Planar region X and Y, and the set of common regions Z.



(a) Common set of solid regions Z.



(b) Solid regions X, Y and Z separated.

The development of the Boolean operations depends on the characteristics of the set of solid regions. As it was mentioned before the common set of solid regions Z determines the Boolean operation, because of this set of solid regions contributes or takes away area to the new solid region generated depending on the type of the Boolean operation. The typical Boolean operations between solid regions will be presented as fallow:

Union ($\cup$): The result of the Boolean union between X and Y corresponds to all the area from X and Y. The Area( ) of the Boolean union fallows the equation 1.

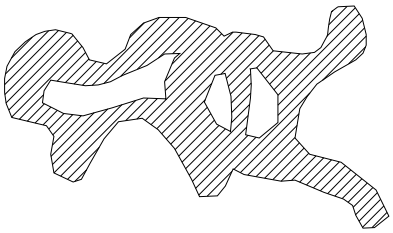
$$Area(X \cup Y) = Area(X) + Area(Y) - Area(Z) \quad (1)$$

Where:

X, Y : Solid regions to be operated
Z : Common set of solid regions between X and Y

In Figure 4 is displayed the result of the Boolean operation $X \cup Y$.

Figure 4. The Boolean operation $X \cup Y$.



Intersection ($\cap$): The Boolean intersection between two solid regions corresponds to the set of common regions (Z). The Area( ) of the Boolean Intersection fallows the below equation:

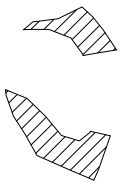
$$Area(X \cap Y) = Area(Z) \quad (2)$$

Where:

X, Y : Solid regions to be operated
Z : Common set of solid regions between X and Y

In Figure 5 is displayed the result of the Boolean operation $X \cap Y$.

Figure 5. The Boolean operation $X \cap Y$.



Subtraction (-): The Boolean subtraction takes into consideration the operation order of the solid regions (X - Y) or (Y - X). The resulting solid regions correspond to the area from one of the operand solid regions outside the other ones. The Area() of the Boolean Subtraction fallows the equations 3 and 4:

$$Area(X-Y) = Area(X) - Area(Z) \quad (3)$$

$$Area(Y-X) = Area(Y) - Area(Z) \quad (4)$$

Where:

X,Y : Solid regions to be operated
Z : Common set of solid regions between X and Y

In Figure 6 is displayed the result of the Boolean operation X - Y and Y - X.

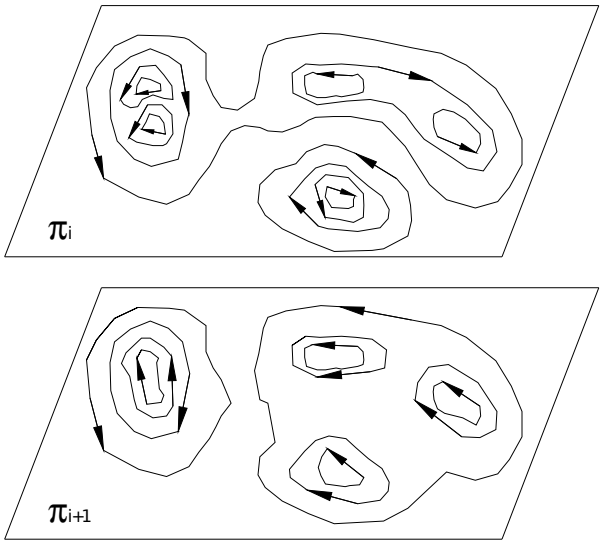
Figure 6. The subtraction (-) Boolean operation.

(a) The Boolean operation X - Y (b) The Boolean operation Y - X

3.2.1.2 2-D Shape Matching:

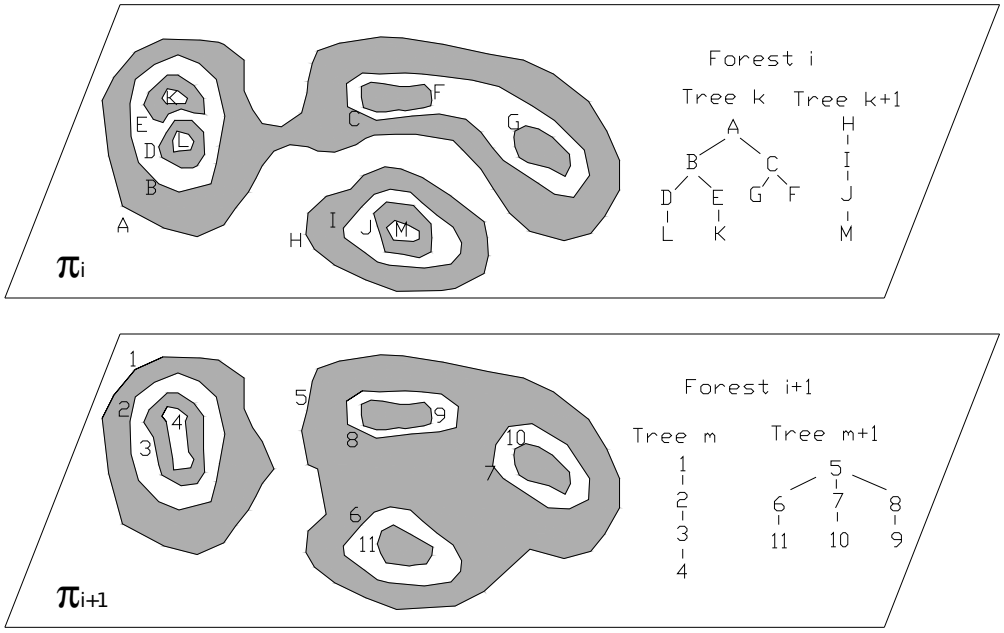
Contour Orientation and Inclusion Calculation: The sets of contours S_i and S_{i+1} are pre-processed to ensure correct orientation, area signs, and to identify the inclusion relations ($X \subset Y$) among contours (Figure 7).

Figure 7. Contour orientation of the sets of contours S_i and S_{i+1}.



With the correct contour orientation and inclusion calculation, it builds trees $T_{k,i}$ (k-th tree in i-th level) and trees $T_{m,i+1}$ (m-th tree in (i+1)-th level) and therefore the forests F_i and F_{i+1} (Figure 8).

Figure 8. Inclusion calculation of the solid regions on planes Π_i and Π_{i+1}.



Calculation of 2-D Similar Regions: Sets of regions in cross sections Π_i and Π_{i+1} are paired, forming a set $MG_i^{i+1} = \{mg_1, mg_2, ..., mg_N\}$, where mapping group $mg_k = [G_{k,i}, G_{k,i+1}]$, with $G_{k,i}$, $G_{k,i+1}$ being regions similar to each other in levels i and i+1, respectively. In every mapping group the $G_{k,i}$ and $G_{k,i+1}$ are planar regions either solid or void regions (for example in Figure 8 a solid region are contours 1(2) and a void region are contours 2(3)). Solid regions are matched to solid regions, and void regions are matched to void regions.

A mapping group is denoted by [{<planar_region>}, {<planar_region>}], where {<planar_region>} may be (i) the empty set: {Φ}, (ii) a connected region bounded by X with holes $x_1, x_2,..., x_n$: {X ($x_1, x_2,..., x_n$)}, (iii) a connected region bounded by X with no holes: {X}, or, (iv) a set of disjoint (connected) regions (possibly with holes): {X ($x_1, x_2,..., x_s$), Y ($y_1, y_2,..., y_t$)}. In Figure 9 is possible to see a result of the calculation of 2D-similar regions.

Figure 9. Calculation of the mapping groups of the Figure 8.

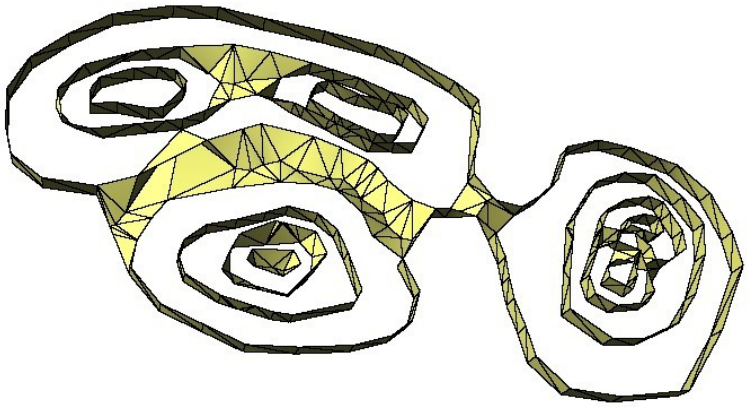
mg_1 = [{A(B(D(L), E(K)),C(F, G)), H(I(J(M)))}, {1(2(3(4))), 5(6(11), 7(10), 8(9))}]
mg_2 = [{B(D(L), E(K))}, {2(3(4))}] mg_3 = [{C(F, G)}, {8(9), 7(10)}]
mg_4 = [{I(J(M))}, {6(11)}] mg_5 = [{E(K), D(L)}, {3(4)}] mg_6 = [{F}, {9}]
mg_7 = [{G}, {10}] mg_8 = [{J(M)}, {11}] mg_9 = [{K, L}, {4}] mg_{10} = [{M}, {Φ}]
$MG_i^{i+1} = \{mg_1, mg_2, mg_3, mg_4, mg_5, mg_6, mg_7, mg_8, mg_9, mg_{10}\}$

Surface Building: The mapping groups resulting from the calculation of 2-D similar regions may be used for sequential calls to the Voronoi-Delone

based algorithm BG( ). The form of each call is BG($G_{k,i}$, $G_{k,i+1}$). However, the generated surface may have topologic problems that the geometrical algorithm (BG( )) could not identified and solved. Therefore, a post-processing of mapping groups is required to identify and solve topologic problems.

In the Figure 10 is displayed the result of the surface building of the Figure 8, without post-processing of mapping groups and therefore with topologic problems.

Figure 10. Surface building result with topologic problems.



3.2.2 Contribution of this project in 2-D Similarity Mapping Groups

3.2.2.1 Post-processing of Mapping Groups: The mapping groups performed in the calculation of 2-D similar regions may have topologic problems. These problems need to be identified and solved. Therefore a post-processing of the mapping groups is needed.

Depurating mapping groups: After performing the calculation of 2-D similar regions, a contour may appear in several mapping groups. Such participation is eliminated in all but one of the mapping groups in which it appears. Each contour will appear in exactly one of the new mapping

groups. To reach a depurated mapping group, the following definitions are necessary:

(i) Depth of a contour inside a forest: The depth of a contour X (depth(X)) in forest F is the number of levels of the tree T (T belongs to F) that separate X from the root of T, starting with 0 up to the total number of levels minus 1. For example in Figure 8, depth(H) = 0 and depth(F) = 2.

(ii) Depth of mg_i: The depth of a mapping group (depth(mg_i)), is the lowest depth(X) in mg_i. For example in Figure 9, depth(mg_1) = 0 and depth(mg_7) = 2.

(iii) Descendant sorting of MG_i^{i+1}: The set of mapping groups need to be sorted in descendant form, determined by depth(mg_i). For example in Figure 9, mg_7 will be processed first than mg_1 because depth(mg_1) is less than depth(mg_7).

The Algorithm 1 is presented for depurating mapping groups. First is necessary to sort the mapping groups in descendant form (line 2). After sorting mapping groups the common components of the first item of the whole list (line 4) are removed from every mapping groups of the list (line 7).

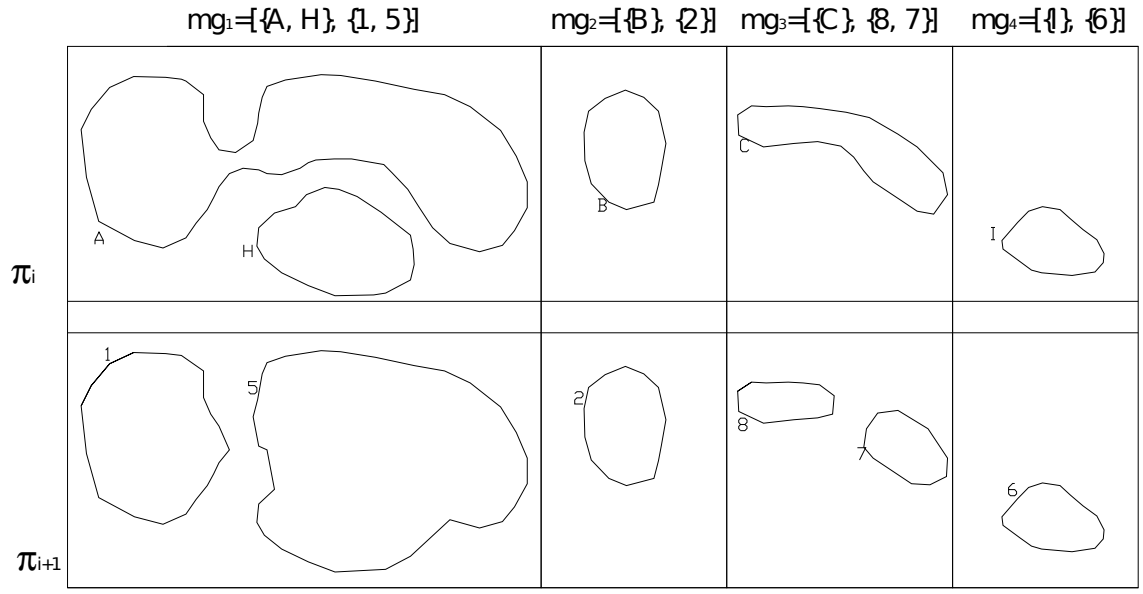
Algorithm 1. Mapping group depuration.

MG_{out} = depurateMapGroups(MG_{in})		
	Input:	MG_{in}: Mapping groups without depuration.
	Output:	MG_{out}: Depurated mapping groups.
	Precondition:	Mapping groups should be generated.
	Postcondition:	Mapping groups should be classified with a handling of hole.
1:	MG_{out} = {}	
2:	MG_s = descendantSortMG(MG_{in})	
3:	while MG_s is not empty do	

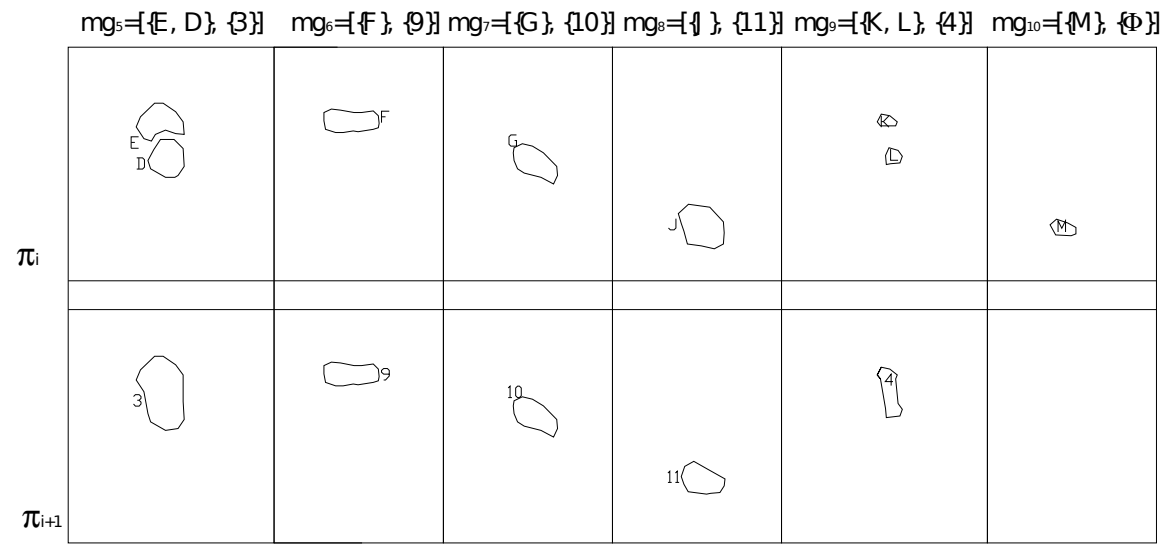
```
4:              mg = getFirstItem( MG_s )
5:              remove( MG_s, mg )
6:              for every mg_i in MG_s do
7:                      remove( mg_i, ( mg_i ∩ mg ) )
8:              end for
9:              append( MG_out, mg )
10:     end while
11:     return( MG_out )
```

Figure 11 presents the result of the mapping group depuration of the Figure 9.

Figure 11. Depuration of the mapping groups in the Figure 9.



(a) Mapping groups from mg_1 to mg_4.



(a) Mapping groups from mg_5 to mg_{10}.

Searching for incorrect mapping groups: After depurating mapping groups, mapping groups cannot be a base for surface reconstruction, because:

(a) Simultaneous death or birth of contours must be solved,

(b) Geometrically impossible surfaces must be eliminated,

(c) Mapping groups that represent impossible topologies must be eliminated.

Therefore a classification and handling of hole mappings is necessary:

(i) A mapping group mg_i hole to hole is discarded if their parents are not mapping. For some solid region R_i in the cross section opposite to the one of the hole X:

(ii) A mapping group mg_i, hole to void ([{X}, {Φ}]) is treated with 0- or 2-handles if

$$\left[\frac{Area(X \cap R_i)}{Area(X)}\right] \geq (1 - threshold) \quad (5)$$

Where:

X: Hole

R_i: Solid Region in the opposite cross section to the hole (X)

threshold: Accepted similarity relation (0 - 1)

In the Figure 11 mg_{10} is treated with a 0-handles.

(iii) a mapping group mg_i hole to void mappings is handled with 1-handles if

$$\left[\frac{Area(X \cap R_i)}{Area(X)} \right] \leq threshold \quad (6)$$

Where:

X: Hole

R_i: Solid Region in the opposite cross section to the hole (X)

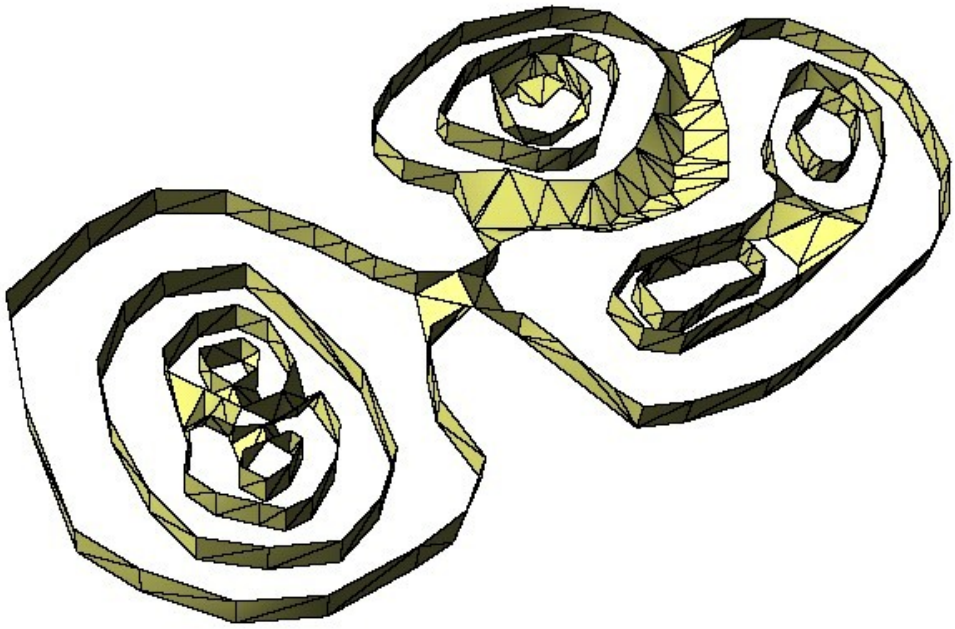
threshold: Accepted similarity relation (0 - 1)

(iv) otherwise BG(mg) directly applies.

In the Figure 11 mg_2, mg_3, mg_4 and mg_9 are treated with BG( ) directly applies.

In the Figure 12 is displayed the result of the surface building of the Figure 8, with post-processing of mapping groups and therefore without topologic problems.

Figure 12. Surface building result with post-processing of mapping groups.



The Algorithm 2 is presented for searching hole mapping-groups that could provoke invalid surface, because of non-handling of topologic events. The algorithm searches hole mapping-groups (line 5) and then verify the validity of it (line 6), with the hole mapping-groups classification and handling.

Algorithm 2. Hole mapping group validity.

MG_{out} = holeMappingGroupValidity (MG_{in})		
	Input:	MG_{in}: Mapping groups with possible topologic problems.
	Output:	MG_{out}: Mapping groups without topologic problems.
	Precondition:	Mapping groups should be depurated.
	Postcondition:	Mapping groups should not have topologic problems and these can be processed with the BG ( ) algorithm.
1:	MG_{out} = {}	
2:	while MG_{in} is not empty do	
3:	mg = getFirstItem(MG_{in})	
4:	remove(MG_{in}, mg)	
5:	if mg is HOLE_GROUP then	
6:	verifyHoleValidity(mg)	

```
7:              end if
8:              append( MG_out, mg )
9:      end while
10:     return( MG_out )
```

3.2.2.2 Joining and Testing of 2-Manifold: A PL Boundary Representation (B-REP) is used for the manifold $M = M_1^2 \cup M_2^3 \cup M_3^4 \cup ....$ with the M_i^{i+1} being the 2-manifolds (with border) produced in the surface building. The B-REP structure makes explicit the face neighborhood relations, the normal vector uniformity and the borders. In this manner, the quality of the manifold M may be evaluated.

Notice that the Algorithm 3 is capable of producing two-dimensional manifolds with and without border at will. First, it builds the B-REP structure of every disjoint shell (line 3) and then, if there is more than one disjoin shell, the algorithm trays to join them by the boundary (line11).

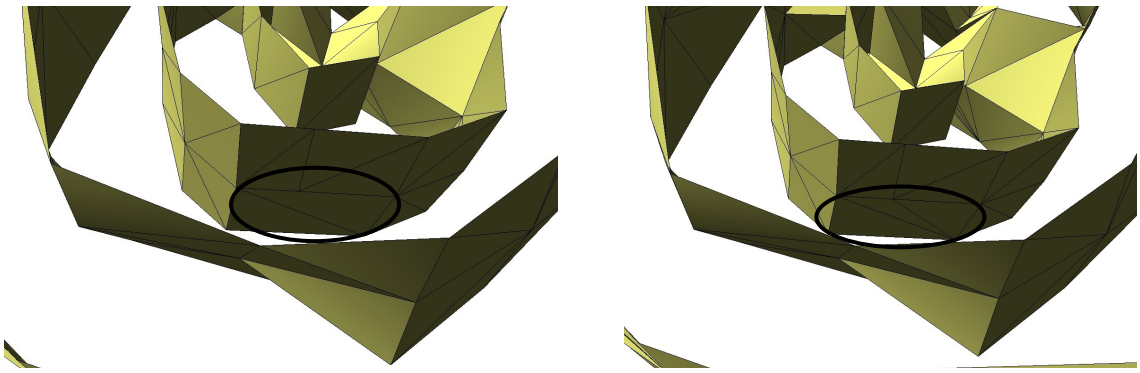
Algorithm 3. Joining shells.

```
M_out = joinShells( DSL )
        Input:          DSL: A list of disjoining shells.
        Output:         M_out: A list of joining shells.
        Precondition:   DSL should be made up of triangles.
        Postcondition:  M_out is a B-REP structure, with connectivity and neighborhood information.
1:      brep_ dsl = {}
2:      for every shell_i in DSL do
3:              brep_shell_i = buildingBrepShell( shell_i )
4:              append(brep_ dsl, brep_shell_i )
5:      end for
6:      M_out = {}
7:      brep_shell_i = getFirstItem( brep_dsl )
8:      remove( brep_dsl, brep_shell_i )
9:      while in brep_dsl is not empty do
10:             brep_shell_i+1 = getFirstItem ( brep_dsl )
11:             brep_shell_i = joinBrepShells( brep_shell_i, brep_shell_i+1 )
12:             append(M_out, brep_shell_i )
```

13: remove(brep_dsl, $brep\_shell_{i+1}$)
14: $brep\_shell_i = brep\_shell_{i+1}$
15: end while
16: return(M_{out})

The Figure 13 shows a comparison between the results of the 2-manifolds join without (discontinuity) and with (continuity) the post-processing of the Algorithm 3.

Figure 13. Comparison of the joining shells algorithm post-processing.



(a) Without post-processing. (b) With post-processing.

3.3 STUDY CASES AND EXAMPLES

The described method for 2-manifold surface reconstruction based on 2-D Shape Similarity was applied in many examples for improving the whole processes and algorithms. In special, two study cases were probed: (a) a brain and (b) a skull of a monkey.

3.3.1 Brain

This model comes from a CNT file, and was processed up to be able to make the surface reconstruction with the modules described in this project and the algorithms of the CAD / CAM / CAE laboratory. This model has many obstacles to get reconstruct, because of the bad sampling made

in the real object. Therefore, it is a good study case to improve and refine the algorithms.

Brain Information:

Number of levels:	15
Levels parallel to:	Π = [pv = (0, 0, 0),n = (0, 0, 1)]
Number of contours:	105
Produced triangles:	13777

In Figure 14 and 15 are displayed two sets of contours on two consecutive levels (8 - 9) of the brain and the set of levels (respectively).

Figure 14. Levels 8 - 9 of the brain.

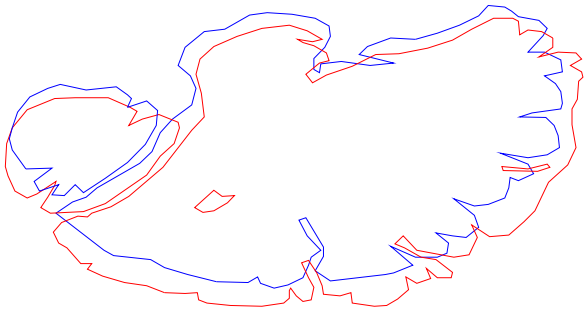
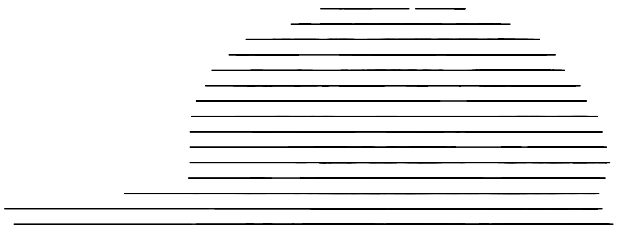
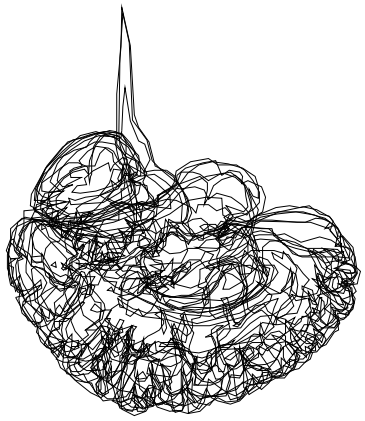


Figure 15. Levels of the brain.



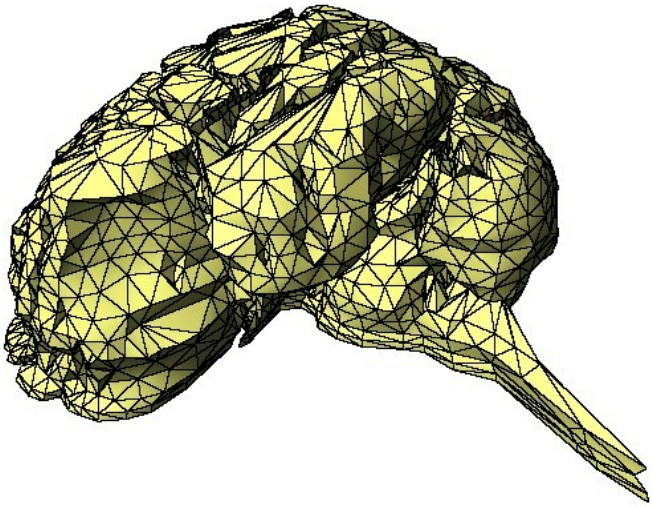
(a) Front view of the brain



(a) Top view of the brain

In Figure 16 were displayed the reconstructed surface of the brain.

Figure 16. Reconstructed surface of the brain.



3.3.2 Skull of a Monkey:

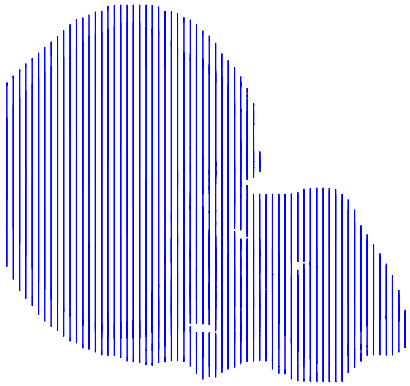
The skull was got in scalar field format in a TXT file, and was processed up to build the contours in every level. The surface reconstruction was made with the modules of this project and the algorithms of the CAD / CAM / CAE laboratory for achieving improvements in the algorithms.

Skull Information:

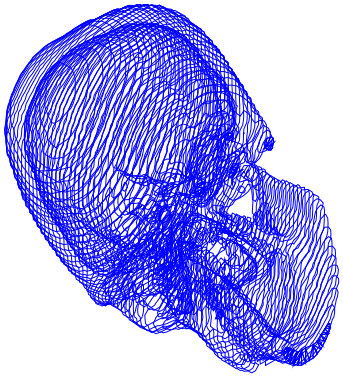
Number of levels:	64
Levels parallel to:	Π = [pv = (0, 0, 0),n = (0, 1, 0)]
Number of contours:	344
Produced triangles:	38656

In Figure 17 and 18 are displayed the set of levels and two sets of contours on two consecutive levels (30 - 31) of the skull (respectively).

Figure 17. Levels of the skull.

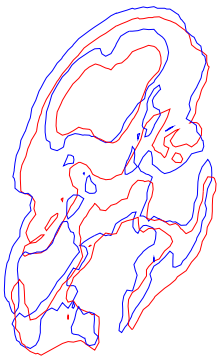


(a) Lateral view of the skull



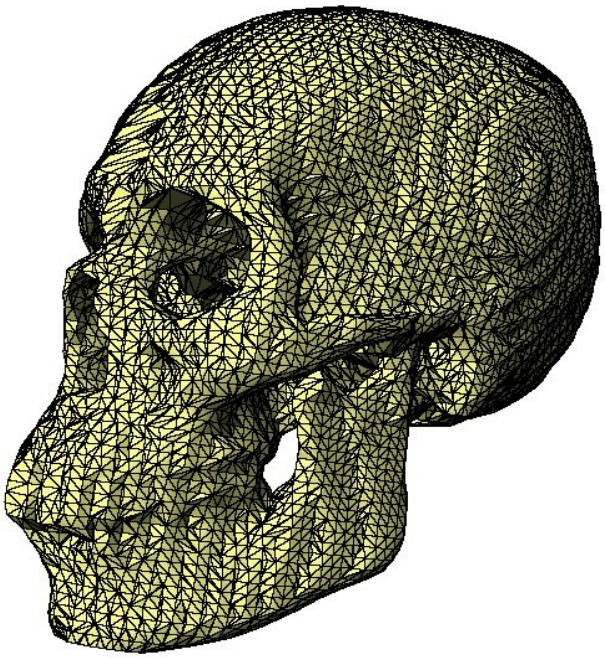
(b) Isometric View of the skull

Figure 18. Levels 30 - 31 of the skull.



In the Figure 19 was displayed the reconstructed surface of the skull.

Figure 19. Reconstructed surface of the skull.



3.4 CONCLUSIONS AND FUTURE WORK

3.4.1 Conclusions for 2-D Shape Similarity:

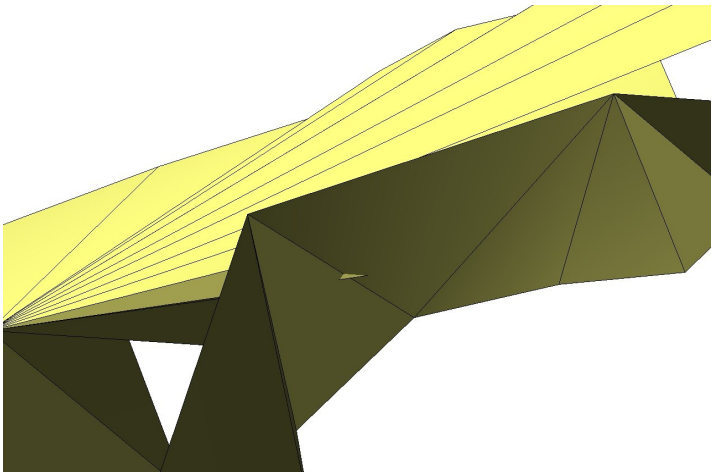
(i) The presented method is successful in avoiding overstretched surfaces, because of the simultaneous evaluation of geometrical and topological conditions for 2-manifold reconstruction based on 2-D shape similarity.

(ii) The algorithm for shell integration detects and corrects discontinuities between shells caused by the differences among the boundaries of each 2-manifold with border. In addition, this algorithm permits the uniformity among the normal of the faces and therefore improves the quality of the final 2-manifold.

3.4.2 Future work for 2-D Shape Similarity:

(i) The presented method may be improved, avoiding large surfaces between two consecutive cross-sections, which generate interference between surfaces, but non-topologic problems. It may be possible with more strict conditions on the geometrical calculation of the 2-D similar regions (Figure 20).

Figure 20. Interference between surfaces.



4. 2-MANIFOLD BUILD FOR TERRAIN AND BUILDING MODELING

4.1 CONTEXT

A method for the combination and integration into a single data of terrain and building database is presented. This study is justified if ray-tracing techniques are to be used in propagation and channel modeling studies. Usually terrain is available in grid or elevation form while building information is normally facet-oriented. Ray-tracing (RT) techniques deal with flat facets and straight edges, if possible in triangular format. To allow the use of RT on urban areas over irregular terrain a common format made up of facets and edges is therefore needed.

The combination and integration into a single database of terrain and building data ([De Floriani et al., 1999]) has two main matters: (i) the first concern is the conversion and completion of terrain data. Most geography institutions publish their elevation maps in either grid or contour formats.

Conversion from iso-altitude contour to grid format includes the application of stochastic prediction ([Sákösy, 1999], [Barbosa and Custódio, 1998]) such as “kriging” ([Van Beers and Kleijnen, 2001], [Gebhardt, 2003]) or Principal Component Analysis (PCA [Popovici and. Thiran, 2002]).

Conversion from grid to iso-altitude contour format includes the production of a triangle-based mesh as an intermediate step, followed by a parallel slicing of the triangle mesh. The production of a triangle-based

mesh is greatly facilitated by the neighborhood information implicit in the grid ([Ruiz, and Neugebauer, 2000]) and the assumption that no void spaces are present in the grid.

The maintenance of correct topological void information is a more complex task. In any case, the result of such technique applied to terrain modeling is a 2-manifold shell ([Ruiz, 2002]) with borders (which means incomplete), with C^0 continuity if it is tiled with triangles.

(ii) A second main task comes with the integration of buildings and terrain information. For many applications, buildings are represented as 2-D plant contours. This representation is not a 3-D one, and therefore has no topological or geometrical consistency.

In this project, 2-D plant contours are extruded along the vertical direction to generate solids. There is an incompatibility, because of to achieve an integrated geometric model of terrain and buildings, terrain is expressed as a shell with boundaries and the buildings are solids.

This incompatibility is presented by: (a) extracting from the building solid information its shell, (b) exploding the building complete shell into several incomplete sub-shells, and (c) performing under-determined Boolean operations of every shell against the incomplete shells representing the terrain. Given that Boolean operation, it has been proposed and implemented for 2-manifolds without border; operation (c) is not well defined.

4.2 MODELING TERRAIN AND BUILDING 2-MANIFOLD

4.2.1 Background

The modeling of terrain and building data has considerable computational complexity. Such computational complexity ([Ruiz, 2002]) comes from: (a) the translation among modeling schema such as complete / incomplete Boundary Representations (B-REPs), constructive solid geometry, simplified triangular B-REP, exhaustive enumeration, etc. (b) the difficult solution of geometrical-topological problems, (c) the maintenance of consistency in simplified models, and (d) the control of explosively large data sets.

The integration of Terrain and Building Data Bases (TDB and BDB respectively) implemented in this project included the following 3 steps: (i) Data pre-processing and schema conversion to ensure topological and geometrical compatibility, (ii) utilization of under-defined Boolean algorithms ([Krishnan and Manocha, 1996]) to joint building and terrain data, with correct disposal of “dangling” faces and edges, which are side-products of the operation. (iii) Decimation ([Ruiz, 2002], [Garcia, 2003]) of resulting shell to improve its geometrical and topological quality, to reduce its size and to enforce a selective level of data simplification. To achieve the integration of terrain and building data into a single date base, it is need to explain surface modeling and 2-manifold Boolean operation as fallow:

4.2.1.1 Surface Modeling: Surface modeling refers in this context to change the data format, to make it operable under certain algorithms and / or applications. Data conversion is different from schema conversion. The first implies parsing, scanning, translation from hard copy to raster data, etc. Schema conversion implies calculation of topological relations

and geometrical parameters, to ensure completeness and consistency of the particular sample at hand. A typical case of schema conversion is the translation from grid to implicit surface, iso-altitude into implicit surface, implicit surface into iso-altitude, etc. Notice that, because of insufficiency in data, schema conversion is not always possible, and not always renders unique results.

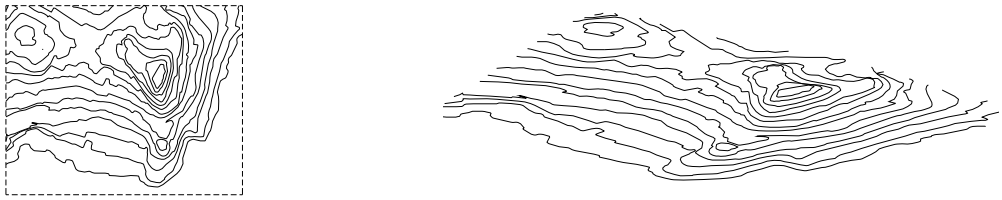
Architectural housing information is still commonly represented as planar 2-D contours showing the plant view of the construction. Notice that even this information may not be available in city councils; therefore, a digitization of hard copy drawings may be necessary to obtain it.

Terrain Representation: The treatment of terrain data aims to convert Geographic Information System (GIS) data into a Boundary-Representation structure. As mentioned before, conversion between schemas is not always possible, usually because there is insufficient information or ambiguity at either side of the conversion process. The B-REP structure prescribes that a “solid” is the “interior” of a closed “surface”. With no formal definition on those terms, it is noticeable here that the Boundary Representation also used to represent incomplete surfaces, with the understanding that when the surface is incomplete, no solid is being represented.

For the purpose of terrain and housing representation, a partial shell is adequate, if additional information is provided (for example, what is “inside” or “outside”). The surface to be created starting from GIS data may be either triangle (C^0) or NURBS (C^2) type. Two types of GIS data are processed in this project:

Iso-altitude Contours: An iso-altitude contour is a Piecewise Linear approximation of a planar curve, considered in a coordinate system in which the planar curve will have constant Z (usually called “altitude”) value. The GIS data is commonly a set of iso-altitude curves, confined to a rectangular region in the XY plane (see Figure 21), with the Z_i values of the iso-altitude $C_i(u)$ curves usually forming a increase uniformly spaced in Z axis. The $C_i(u)$ curves are supposed to be a cross sectional sample of a non - self intersecting surface (in this case, the terrain itself).

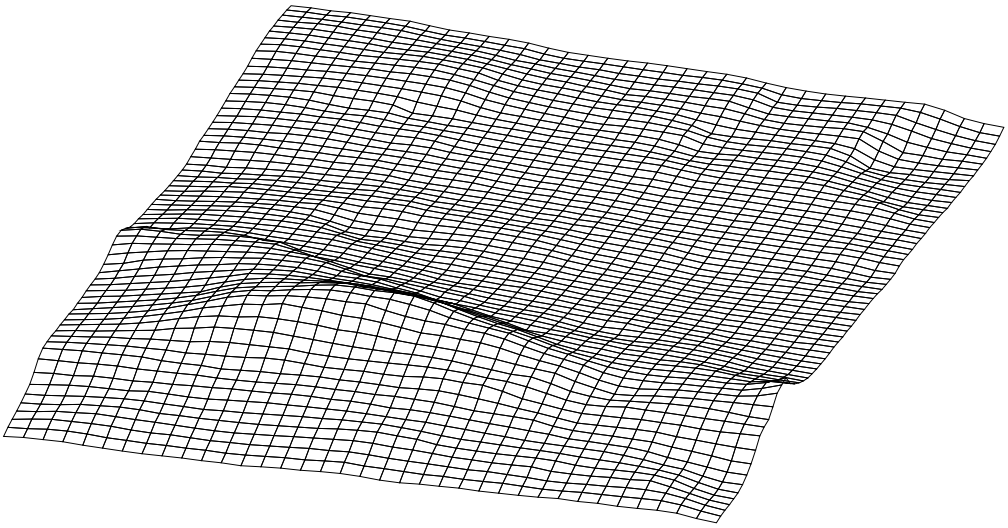
Figure 21. Iso-altitude contours.



(a) Iso-altitude (plant view) (b) Iso-altitude (isometric)

Elevation grid data: A grid elevation ([Felicísimo, 1994]) data set represents a function f: R x R -> R (Figure 22). It supposes that for a particular pair (x, y), the function has a unique value f(x, y). Therefore, it is adequate to represent most of terrain data. In this case, f(x, y) = z(x, y), an altitude value. It is usual that the (x, y) couples be sampled from a regular rectangular grid, therefore having a (N x M) number of samples. This formalism is widely used, even with the f( ) function being temperature, humidity, pressure, etc. Also, in range pictures for 3-D digital optical sampling the same information is stored.

Figure 22. Elevation regular grid.



NURBS Surface: At this point, a parallel shell representation is devised, depending on the algorithms that later on will be commissioned to calculate Boolean operations between shells. If the Boolean operators are not able to perform intersections or unions between triangular meshes, an alternative NURBS representation must be used, as in the present case.

A Non-Uniform Rational B-Spline (NURBS) is a mathematical representation of a 3-D object. Most CAD applications support NURBS, which can be used to represent analytic free-form shapes. A NURBS surface is defined ([Piegl and Tiller, 1997]) as:

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{ij} N_{i,p}(u) N_{j,q}(v) P_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{ij} N_{i,p}(u) N_{j,q}(v)} \quad (7)$$

Where:

u, v: parameter
$N_{i,p}, N_{j,q}$: base function
P_{ij}: control point
w_{ij}: weigth

And

$$U=\{\underbrace{u_0,\ldots u_0}_{p+1},u_1,\ldots u_{i-1},\underbrace{u_i,\ldots u_i}_{p+1}\}\quad V=\{\underbrace{v_0,\ldots v_0}_{q+1},v_1,\ldots v_{j-1},\underbrace{v_j,\ldots v_j}_{q+1}\} \tag{8}$$

The surface is defined from (n+1) x (m+1) control points and weights (n for u-direction, m for v-direction), two-knot vectors having n+p+2 knots or m+q+2 knots, when its degrees are p for u-direction and q for v-direction. The base functions are defined as below:

$$N_{i,k}(t)=\begin{cases}1 & \text{if } t_i\le t<t_{i+1}\\ 0 & \text{else}\end{cases} \tag{9}$$

And

$$N_{i,k}(t)=\frac{(t-t_i)N_{i,k-1}(t)}{t_{i+k}-t_i}+\frac{(t_{i+k+1}-t)N_{i+k+1,k-1}(t)}{t_{i+k+1}-t_{i+1}} \tag{10}$$

Where:

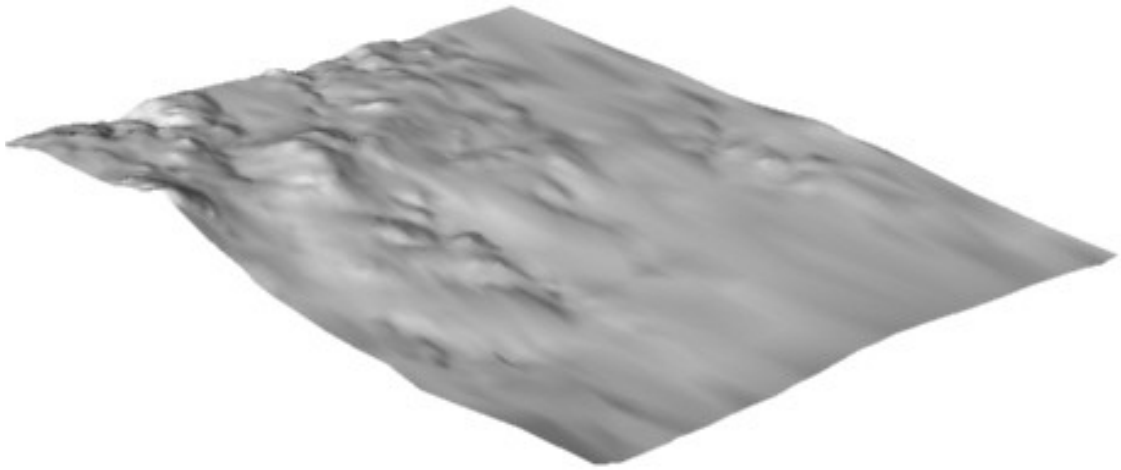
k: degree of curve
t: parameter
t_i: knot

The basis function is defined from knots value, and knot vector is a set of the knots as:

$$T=\{\underbrace{t_0,\ldots t_0}_{k+1},t_1,\ldots t_{i-1},\underbrace{t_i,\ldots t_i}_{k+1}\} \tag{11}$$

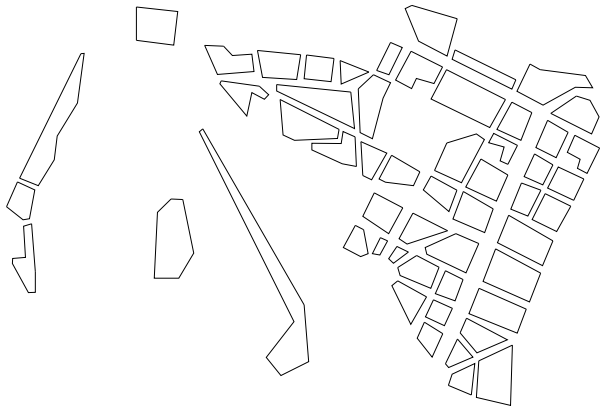
The conversion (Figure 23) from grid-elevation data to NURBS format is achieved by using CAD application like Rhino.

Figure 23. NURBS representation for grid elevation terrain data.



Building Representation: As it was mentioned before, architectural drawings are commonly 2-D plant views of buildings, in the form of closed, non-self-intersecting polygonal contours (Figure 24).

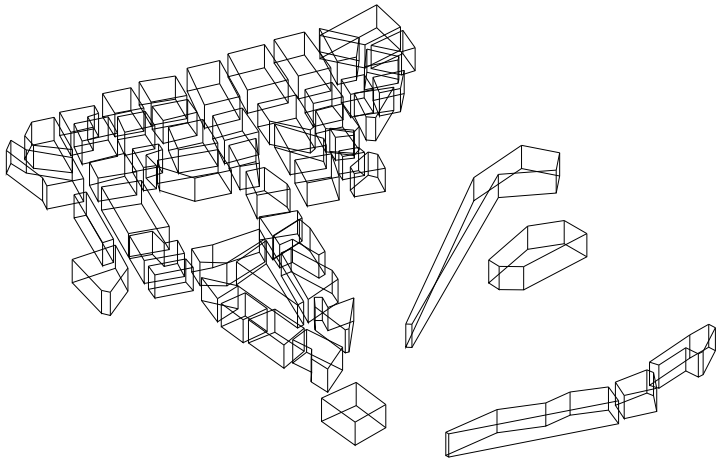
Figure 24. Building representation, planar contour (plant view).



This representation scheme is incomplete and ambiguous. In order to upgrade it and obtain 3-D geometric models of buildings, every section is extruded in the vertical (Z) direction, a distance dependent on the height of the building. The result of this operation is indeed 3-D solid models

(Figure 25), which includes the closed shell model of the boundary. The solid model is therefore decomposed and its shell information.

Figure 25. Solid representation of the buildings data.



4.2.1.2 2-Manifold Boolean Operations: A central goal in the geometric work in this project is to integrate building and terrain information. This integration means that buildings are modeled as protrusions in the original terrain shell. The shell does not need to be closed (in fact, it would be senseless to close it), but it needs to have a unique border (the external one). The condition of unique border excludes internal holes, as well as folds, T-joints, dangling edges and faces, etc.

Boolean union: The Union operation is not strictly a Boolean one since these operations (in the context of geometric solid modeling) require (a) closed shell, and (b) a convention that defines the interior and exterior of the closed shell. Also, Boolean operations ([Krishnan and Manocha, 1996]) unite, intersect or subtract the whole set of points in the interior of solid objects. In strict sense, for example, the intersection of two shells (see Figure 26) would render a set of curves in the space (Figure 27).

Figure 26. Two 2-manifolds in R^3 with its senses.

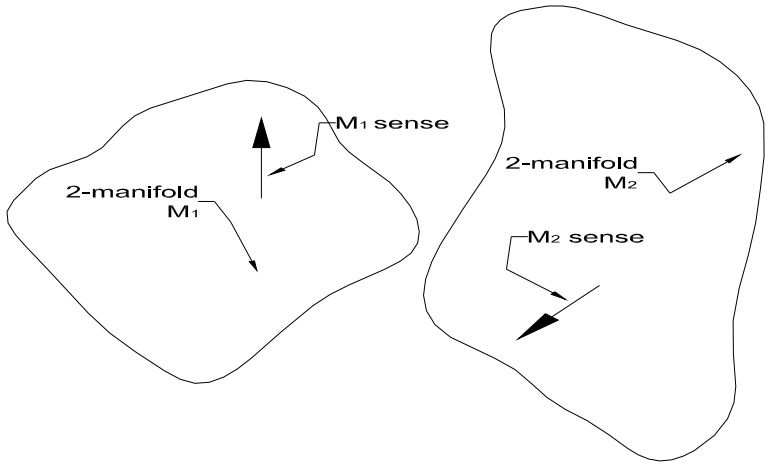
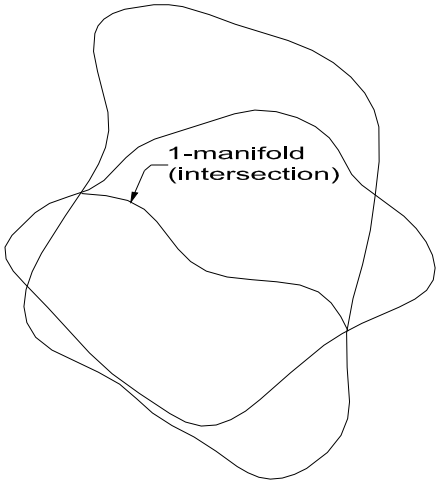


Figure 27. Intersection of two 2-manifolds.



This is a non-intuitive result, as is the union, subtraction, etc. of shells. Therefore the operation used for the present application, can be defined as follows:

Given M_1, M_2, 2-manifolds (shells) in R^3 with n_{M1} ( ): R^3 -> {-1,0,1 } and n_{M2}( ): R^3 -> {-1,0,1 } functions in R^3 which take value -1 at an arbitrary side of M_1, value 0 on M_1 and +1 at the remaining side. n_{M1} ( ) divides the space R^3 in inside/ on / outside M_1. Equivalent definition may be made for n_{M2} ( ). The n_{M1} ( ) and n_{M2} ( ) functions in general are consistent with the 2-

manifold sense. The operation $\cup^*$ (a very particular type of Boolean union) is:

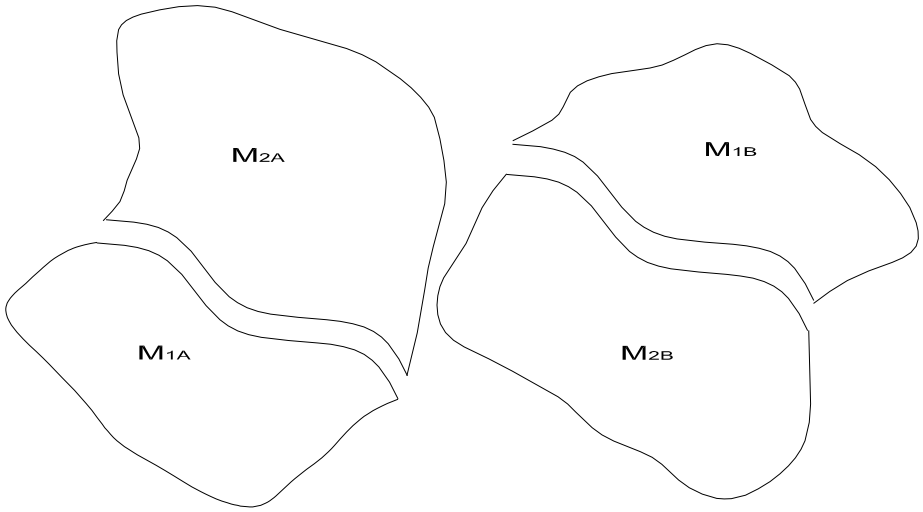
$$M_1 \cup^* M_2 = \{(p \in M_1) \vee (p \in M_2) \mid n_{M_1}(p) \geq 0 \wedge n_{M_2}(p) \geq 0\} \quad (12)$$

Where:

M_1, M_2 : Given 2 - manifolds
p: Point to be evaluated
n_{M_1}, n_{M_2} : Evaluating functions

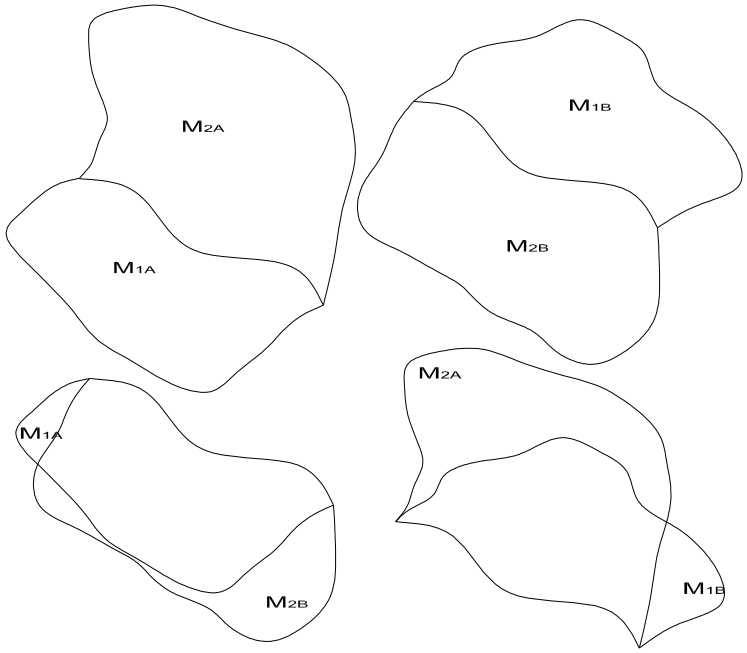
This operation basically keeps all portions of surface M_1 at one "side" of M_2 and neglects the points at the other side. It also keeps all points of M_2 at one side of M_1, and neglects the others. There is obviously a 4-choice combination of what to keep and what to neglect. Intuitively, it can be thought that M_2 divides M_1 into two parts (M_{1A} and M_{1B}) and vice versa in M_2 (M_{2A} and M_{2B}) (Figure 28).

Figure 28. Division combinations of the 2-manifold, M_1 and M_2.



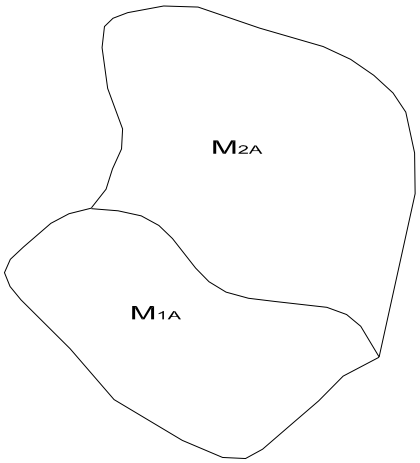
The new 2-manifold may be built in 4 different ways: {M_{1A}, M_{2A}}, {M_{1A}, M_{2B}}, {M_{1B}, M_{2A}}, {M_{1B}, M_{2B}} (Figure 29).

Figure 29. Built choices of the $M_1 \cup^* M_2$.



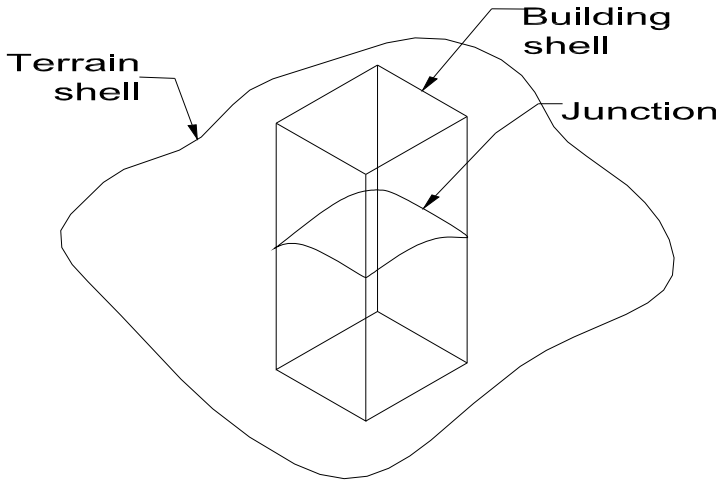
The junction presents C^0 continuity while other spots of the original terrain and building shells have C^2 continuity. In general, the result of the Boolean union corresponds to the positive sides of the two 2-manifolds. Therefore, to obtain a particular result is needed to combine the sense directions of the two 2-manifolds (Figure 30).

Figure 30. Result of the Boolean union between two 2-manifolds with border in R^3.



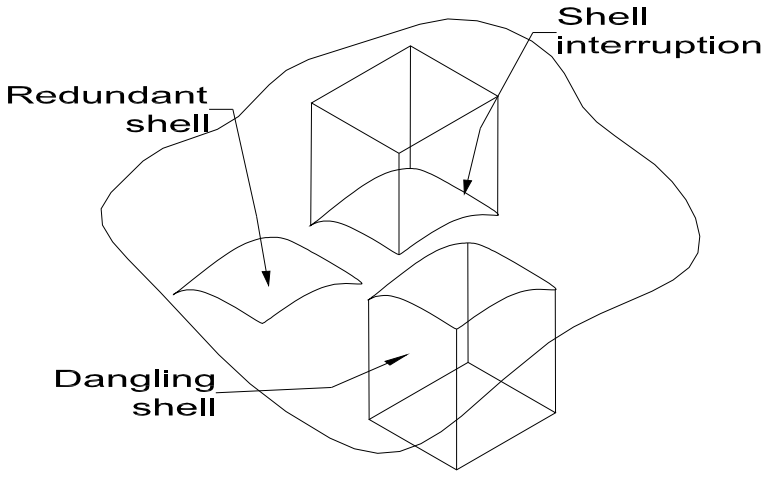
Joining Terrain and Building Shells: To reach this point, a set of context-defined Boolean operations is used. These are under defined Boolean operations among surfaces, which require: (i) every participant object should have border or should be an incomplete shells, and (ii) dangling faces or edges (by-product of surface intersection and splitting) should be eliminated. These two conditions determine the efforts made. In particular, condition (i) is satisfied by creating NURBS surfaces of C^2 or higher continuity, even for flat data. Housing data requires to be integrated within terrain data (Figure 31).

Figure 31. Joining terrain and building shells definition.



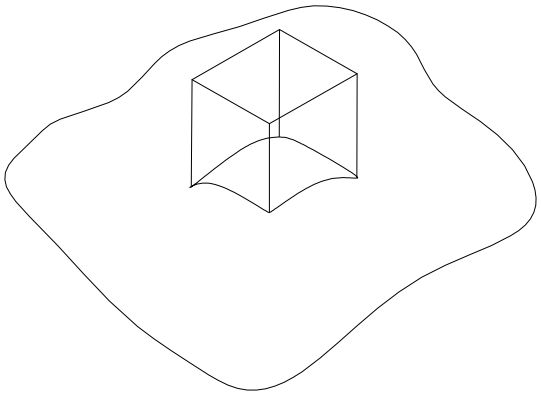
This integration must be complete, in such a way that buildings become protrusions on the terrain shell. The housing-terrain shell must be C^0-continuous (no holes) at the junctions of the building-terrain. There can be no creases, interruptions, folds or dangling edges, non-manifold conditions (redundant “T” surfaces) or faces in such junctures (Figure 32).

Figure 32. Participants of the shell union.



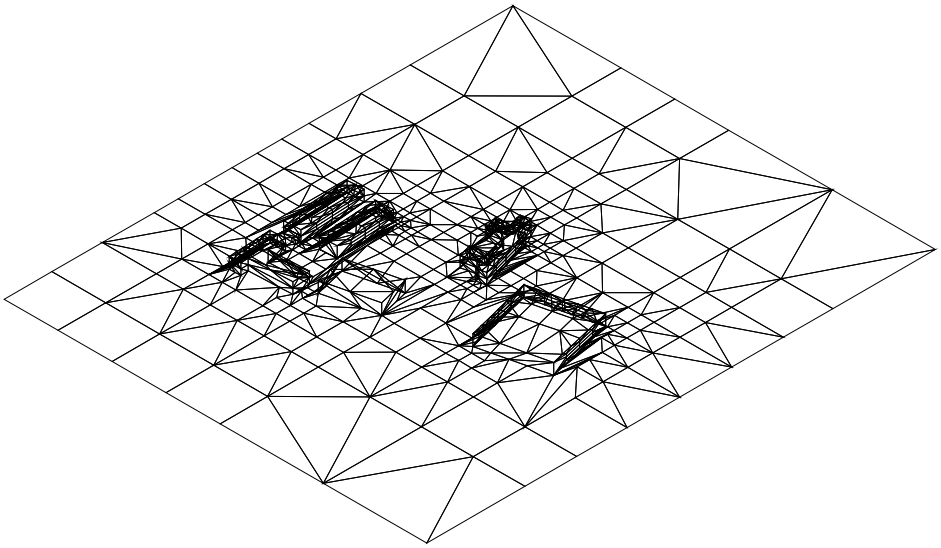
The surface must be perfect at that neighborhood, with the only concession being that the continuity accepted in such junctions is C^0, while all the other places on the building- and terrain-NURBS are C^2-continuous (Figure 33).

Figure 33. Result of the union between terrain and building shells.



For exporting and for the sake of calculations, the C^2 NURBS terrain-housing shell is converted to a C^0 triangle or quadrangle-based shell (Figure 34). Technically, this outcome is ready, since it is a 2-manifold, continuous, with only the outermost border (no internal holes).

Figure 34. Result of the Boolean operation.



4.2.2 Contribution of this project in Modeling Terrain and Building 2-Manifold

4.2.2.1 Surface Building:

Closing Contours: As it was mentioned before, the $C_i(u)$ curves are supposed to be a cross sectional sample of a non-self intersecting surface. This $C_i(u)$ curves are open contours $C_i(u)$, which represent portions of originally closed cross sections being interrupted by the artificial grid of the GIS. This is one of the insufficiencies of this scheme, and must to be overcome by using the rectangle of the grid itself to complete the missing part of the contours, therefore closing them.

The Algorithm 4 is presented for closing iso-altitude contours. First the algorithm identifies the contours in the same plane (line 5) and then, it processes the contours in the same plane with a human operator (lines 9, 11 and 16). When the algorithm and the human operator have processed the total levels in same planes, the algorithm close by itself the trivial contours (line 19). The remaining contours are closed with the

understanding of the human operator an the bounding box of the total contours (lines 25 and 27)

Algorithm 4. Closing contours.

```
CCL = contourClose( OCL )
        Input:          OCL: A list of opened contours.
        Output:         CCL: A list of closed contours.
        Precondition:   The set of opened contours should be in parallel planes.
        Postcondition:  Vertices are added to the contours for closing them.
1:      CCL = {}
2:      boundBox = getBoundingBox( OCL )
3:      while OCL is not empty do
4:              seed = getFirst( OCL )
5:              samePlaneCont = getContSameLevel( seed, OCL )
6:              append( samePlaneCont, seed )
7:              remove( OCL, samePlaneCont )
8:              displayCont( samePlaneCont )
9:              answer = askUser( “Is it need to join contours?” )
10:             while answer is true do
11:                     conList = selectCont( “What do contours join between it?” )
12:                     joinContThemselves ( contList )
13:                     append( CCL, contList )
14:                     remove( samePlaneCont, contList )
15:                     hideCont( contList )
16:                     answer = askUser( “Is it necessary to join more contours?” )
17:             end while
18:             hideCont( samePlaneCont )
19:             contList = closeTrivialCont( samePlaneCont, boundBox )
20:             append( CCL, contList )
21:             remove( samePlaneCont, cont_list )
22:             while samePlaneCont is not empty do
23:                     cont_i = getFirstItem( samePlaneCont )
24:                     displayCont( cont_i )
25:                     answer = askUser( “Does it close with 1, 2 or 3 corners?” )
26:                     hideCont( cont_i )
27:                     closeContour( cont_i, answer, boundBox )
28:                     append( CCL, cont_i )
29:                     remove( samePlaneCont, cont_i )
```

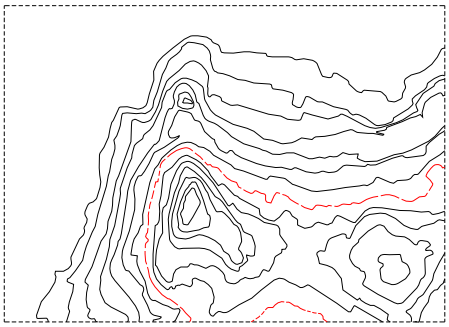
```
30:                 end while
31:         end while
32:         return( CCL )
```

This process is not a trivial one, since the operation of completing contours from pieces of the rectangular grid is not completely defined, and produces a number of possible outcomes. In this particular instance, a human operator, who uses an understanding of the terrain to complete the contours, assists this process.

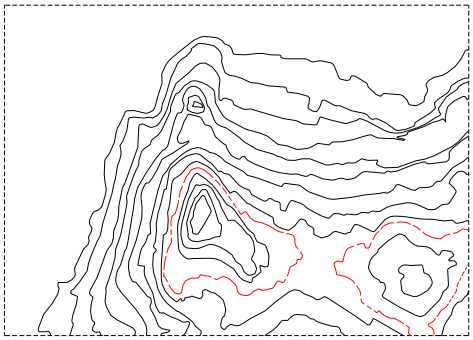
This step is required to be able to reconstruct a C^0 surface from the closed, oriented contours. Algorithms for this purpose are usually based in Delone Triangulations and Voronoi Diagrams ([Boissonat and Geiger, 1993]). A variation, equipped with 2-D shape similarity reasoning is described in the chapter 3 of this document ([Ruiz et al., 2003] and [Ruiz, Cadavid et al., 2002]).

The process and the result of the contourClose( ) algorithm are displayed in the Figures 35 and 36.

Figure 35. Process of the contourClose( ) algorithm.

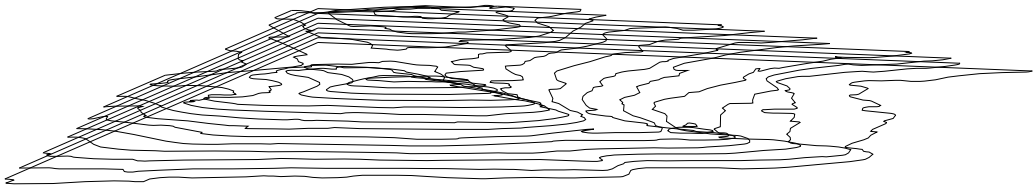


(a) Joining and Closing contours.

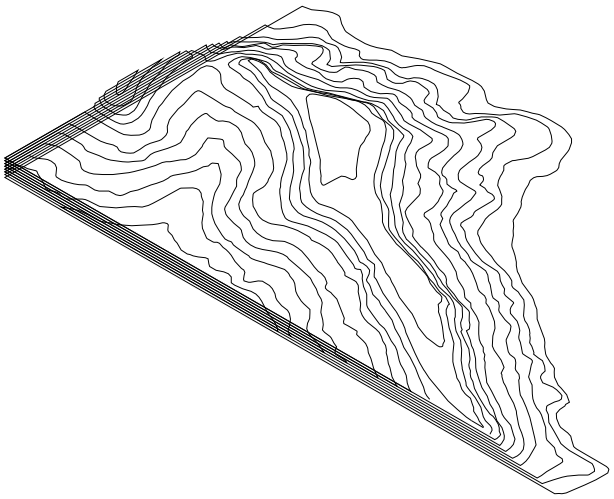


(b) Joining contours.

Figure 36. Result of the contourClose ( ) algorithm.



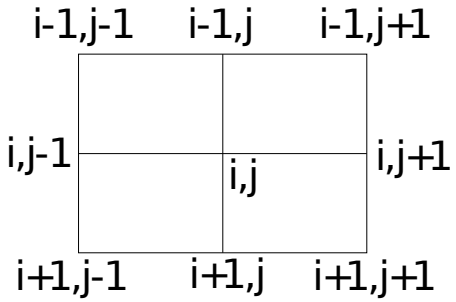
(a) First example of the contourClose ( ) algorithm.



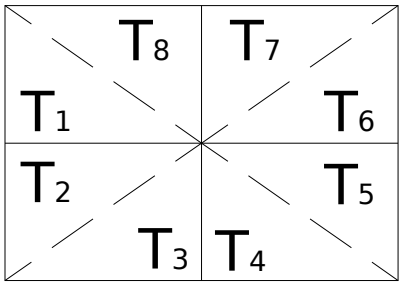
(b) Second example of the contourClose ( ) algorithm.

Triangle Mesh: A grid elevation data set represents a function f( ). For reconstructing the surface, reconstruction algorithm takes advantage of the neighboring information implicit in the grid itself, to build a C^0 set of triangles, interpolating the values of the function f( ) at the grid intersections. Given a grid of 3x3 (x, y) points (Figure 37a) and the function f( ), it is possible to build eight triangles (Figure 37b).

Figure 37. Advantage of the neighboring information.



(a) Grid of 3x3 (x, y) points.



(b) Triangles built.

When points of the (x, y) grid register non value of f( ), a void is produced in the triangle mesh. This situation is not uncommon since the optical device may not register the point due to optical or atmospheric conditions. In those cases explicit information on the external and internal borders of the shell is required. For example in the Figure 38, either the internal node (Figure 38a) or the external nodes (Figure 38b) can not have f( ) value. In those cases, less than eight triangles (Figure 37b) can be built.

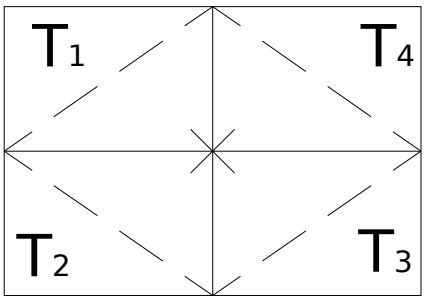
For reconstructing the surface of the grid elevation data, the Algorithm 5 was developed. First, it verifies the valid registration of the f( ) value in the portion grid (line 4) and then, it generates the correct index (line 5) to build the triangles (line 6).

Algorithm 5. Grid faceting.

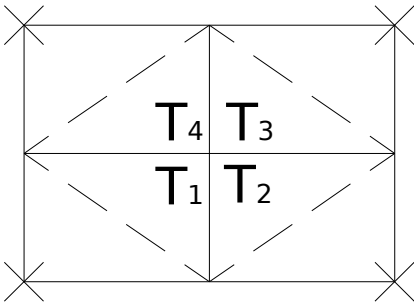
T = gridFaceting(G)	
Input:	G: Grid elevation data. Typically, three matrices of MxN values for X, Y and Z coordinates.
Output:	T: A list of triangles got from the grid data.
Precondition:	G should come from rectangular information, with explicit token for void information.
Postcondition:	T is a 2-manifold split in triangles (C^0-continuity).
1:	T = {}
2:	for every $node_i$ from 2 by 2 in G do

```
3:              for every node_j from 2 by 2 in G do
4:                      correctNodes = verifyingValues( G, node_i, node_j )
5:                      triangleIndices = getIndices( correctNodes )
6:                      trianglePatch = buildTriangles( triangleIndices )
7:                      append( T, trianglePatch )
8:              end for
9:      end for
10:     return( T )
```

Figure 38. Points (x, y) without f( ) value information registered.



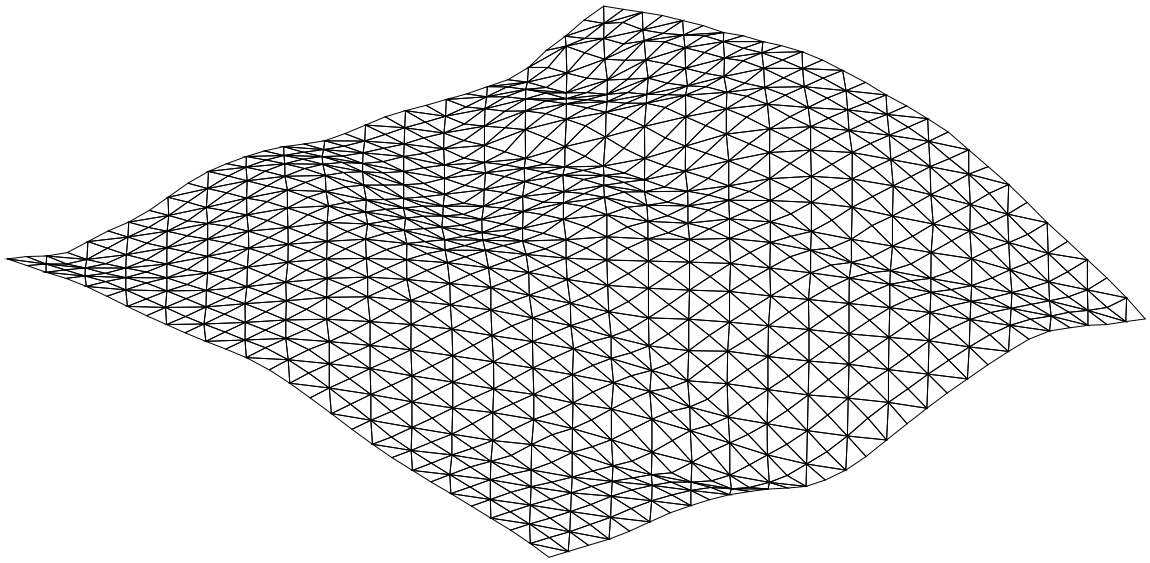
(a) Internal node without value of f( ).



(b) Corner nodes without value of f( ).

The result of the Algorithm 5 is displayed in Figure 39.

Figure 39. Triangles mesh of elevation grid data.



4.2.2.2 Modeling decimation: Although the structure from Figure 34 is mathematically adequate, its number of triangles, heterogeneity in triangle size and extreme triangle aspect ratios ($side_i$ / $side_j$) greatly impairs any calculation. Therefore a process of quality assurance, aspect ratio and size homogenization is required.

This objective was achieved by (i) forming n-sided flat polygons from triangular and quadrangular tiled regions and then (ii) breaking down the n-sided polygons into triangle sets with good geometric characteristics. The Algorithm 6 was developed for these objectives. First, it creates groups of faces with similar characteristics (line 5) and then, it creates the polygons of every generated group of faces (line 7) and then, it splits the polygon into triangles (line 8).

Algorithm 6. Merging faces.

T = mergeFaces(PF, mergeGrade)	
Input:	PF: A list of planar faces without good characteristics. mergeGrade: Value between 0 - 1. Where 0 means that the algorithm makes the minimum possible merge and 1 means that the algorithm

```
                    makes the maximum possible merge.
        Output:     T: A list of triangles with good characteristics.
        Precondition: PF should be a 2-manifold with border made up of planar faces.
        Postcondition: T is a 2-manifold split into triangles (C0-continuity).
1:      groupsList = {}
2:      for every pfi in PF do
3:          n = getNormal( pfi )
4:          pfiEd = extractEdges( pfi )
5:          groupsList = createGroups( pfiEd, n, mergeGrade, groupsList )
6:      end for
7:      polygons = createPolygons( mergeGrade, groupsList )
8:      T = triangulate( polygons )
9:      return( T )
```

Triangle to Polygon Clustering: To produce polygonal regions out of triangular and quadrangular ones the procedure is: (i) collecting all triangles or quadrangles $S_k = \{ T_1, T_2, T_3, ..., T_m \}$ whose areas are approximately coplanar with a statistically calculated best plane $\Pi_k=[pv, n] \in R^3$, (ii) building with S_k an n-side flat polygon P_k (possibly with holes), (iii) repeating (i) and (ii) for triangles or quadrangles contained in significantly different planes Π_w, until all triangles and quadrangles are exhausted.

The Algorithm 7 is presented for creating group faces with similar characteristics. First, it searches in the created group similar characteristic of the face to be evaluated (line 5), if it found one, it appends the edges of the face to the found group (line 8) else; it creates a new group with the characteristic of the face (10).

Algorithm 7. Creating groups.

```
groupsList = createGroups( pfiEd, n, mergeGrade, groupsList )
        Input:      pfiEd: Set of edges which belong to the planar face.
```

n: Normal plane where the planar face lives.

mergeGrade: Value between 0 - 1. Where 0 means that the algorithm makes the minimum possible merge and 1 means that the algorithm makes the maximum possible merge.

groupsList: Groups list of edges that lives on similar planes (with similar normal).

Output: groupsList: Groups list of edges with one set of edges more.

Precondition: The variable groupsList has groups with a representative normal. Every group has face edges with normal similar to the representative normal of the group.

Postcondition: If the variable groupsList has not a group with a representative normal similar to n, a new group will be created and added.

```
1:  if groupsList is empty then
2:      gp = createNewGroup( pfiEd, n )
3:      append( groupsList, gp )
4:  else
5:      gp = findGroup( groupsList, n, mergeGrade )
6:      if gp is not void then
7:          pfiEdDepurated = depurateEdges( pfiEd, gp )
8:          addEdges( gp, pfiEdDepurated )
9:      else
10:         gp = createNewGroup(pfiEd, n )
11:         append( groupsList, gp )
12: end if
13: return( groupsList )
```

For every S_k, there is an outermost straight segment contour (P_k), traversed in counterclockwise direction with respect to an outwards pointing vector (±n) normal to the plane Π_k=[pv, n]. All inner contours of P_k, built in clockwise direction with respect to the normal vector, represent the boundaries of this face with other faces contained in planes significantly different from Π_k. The best (in the statistical sense) plane Π_k is to be calculated from a cloud of quasi-planar points by using for example Principal Curves ([Hastie and Stuetzle, 1989], [Kegl and Krzyzak, 2002]).

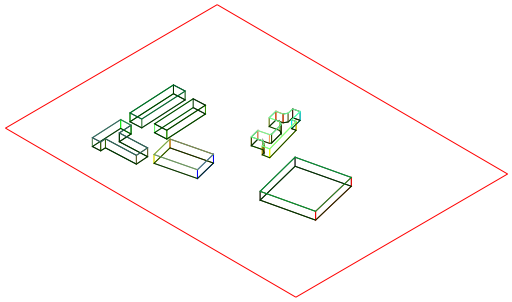
The Algorithm 8 is presented for getting the polygons of every group faces created in the Algorithm 7. First, it creates the contours of a group faces with a correct orientation (line 3) and then, it builds the polygon with the created contours, making the inclusion calculation with the correct orientation of each contours and the relative position between contours (line 4).

Algorithm 8. Getting polygons.

Polygons = getPolygons(mergeGrade, groupsList)		
	Input:	mergeGrade: Value between 0 - 1. Where 0 means that the algorithm makes the minimum possible merge and 1 means that the algorithm makes the maximum possible merge. groupsList: Group list of edges that lives on similar planes (with similar normal).
	Output:	Polygons: Polygon list where every polygon has identified its external contour and its internal contours (if it has).
	Precondition:	The variable groupsList should not have groups with repeated edges.
	Postcondition:	It is possible that each polygon does not live on a plane and it will be needed to map out the polygon to a plane.
1:	Polygons = {}	
2:	for gp_i in groupsList do	
3:	contours = createContours(gp_i, mergeGrade)	
4:	polyList = createPolygons(contours)	
5:	append(Polygons, polyList)	
6:	end for	
7:	return(Polygons)	

The result of this step applied to data from Figure 34 is shown in Figure 40.

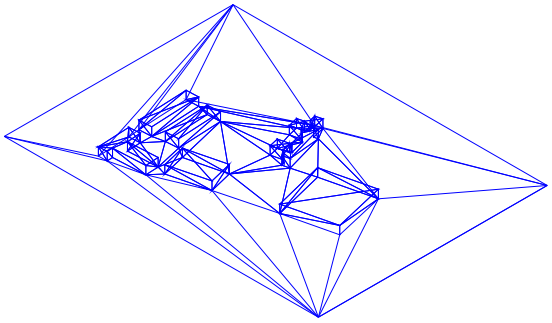
Figure 40. Obtained polygons from the algorithm 7 and 8.



Splitting polygon into triangles: Once n-sided polygons are obtained an application may proceed. However, the algorithmic part of them is more complicated than with triangles. A basic difficulty is that polygons may be non-convex regions, even if they have no holes. In the case in which they have holes, the situation is more complicated.

Although intersection, point inclusion, etc. calculations for general polygons with holes are not impossible, it is definitely easier to deal with triangles. Therefore, n-sided polygons are split into triangles, which have significant better aspect ratio, size, etc. than the original ones. This step was performed by using an external, Delone Triangulation-based ([Shewchuk, 1996]), which allows exactly this type of polygon partition by enforcing different geometric characteristics of the final triangle set. The Figure 41 shows the result of this step applied to data from Figure 40.

Figure 41. Result from breaking down polygon into triangles.



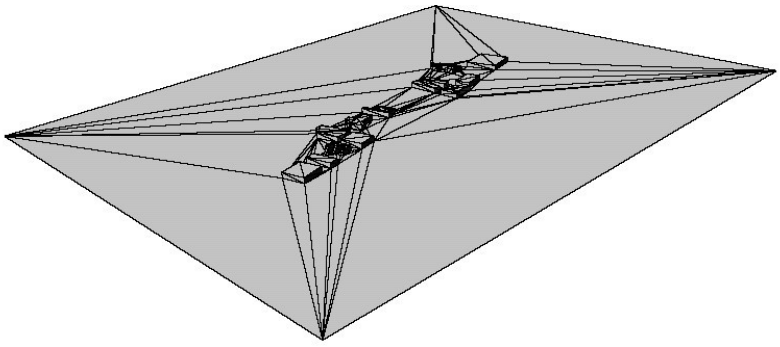
4.3 STUDY CASES AND EXAMPLES, AND APPLICATIONS

The method for 2-manifold build for terrain and building modeling described was applied in many examples for improving the whole processes and algorithms. In special, two study cases were studied and one application was made:

4.3.1 Study Case 1

A 16000 m^2 terrain and building area was modeled with the described method. The integration between terrain and building data generate 2983 triangles. The mergeFaces( ) algorithm produced 112 polygons with 125 contours (indicating the normal and legal presence of internal holes in the polygonal faces). Therefore, the algorithm proposed reduced the number of triangles to 394. The final result is displayed in Figure 42.

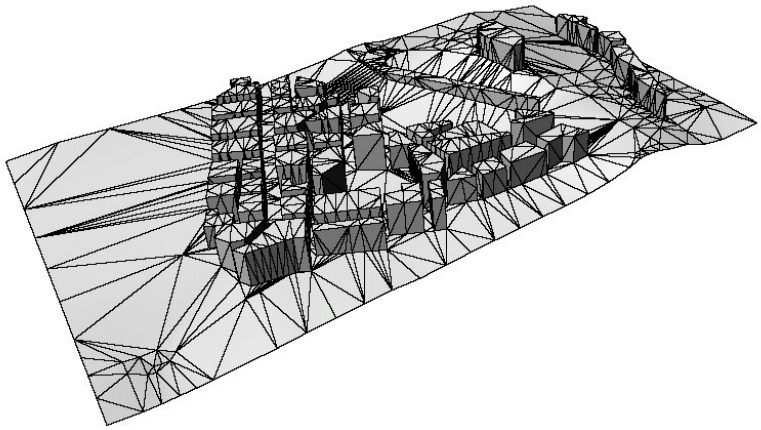
Figure 42. Study case 1.



4.3.2 Study Case 2

A center of a City was modeled with the described method. The integration between terrain and building data generate 6332 triangles. The mergeFaces( ) algorithm produced 328 polygons with 382 contours. In addition, the algorithm proposed reduced the number of faces to 2803 triangles. The final result is shown in Figure 43.

Figure 43. Study case 2.



4.3.3 Application

An application of Ray-Tracing (RT) was applied to the study case 1. The RT consists in calculate over the surface built with triangles reflections and diffractions. In the Figure 44 and 45 are displayed an example of the RT calculation, the result is a diffracted-reflected ray, which comes from a satellite in the space.

Figure 44. Diffracted-Reflected ray from the satellite over the integrated model.

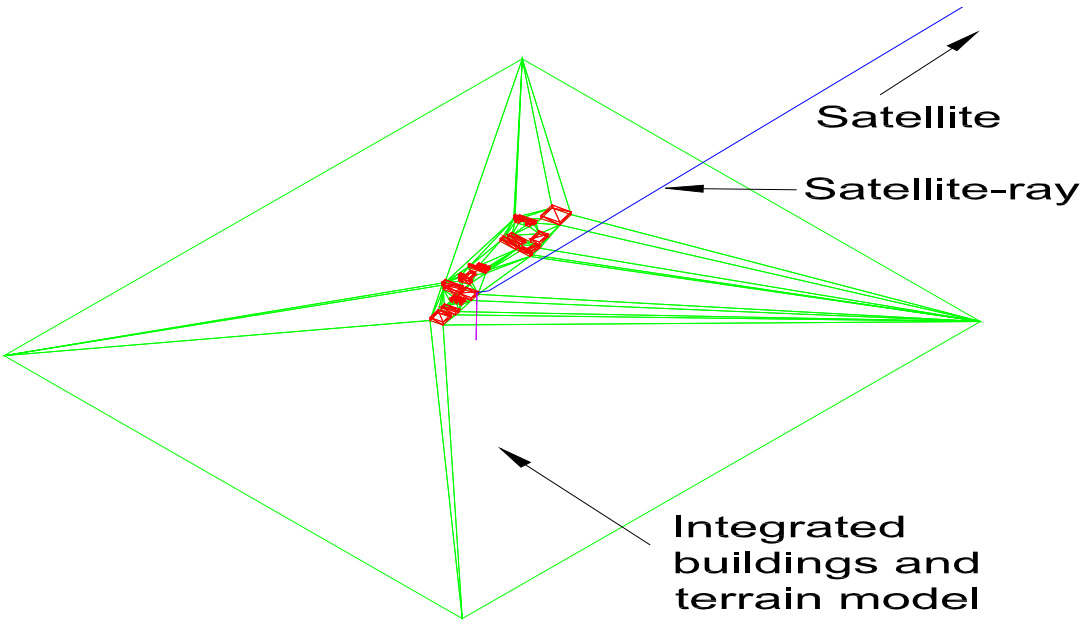
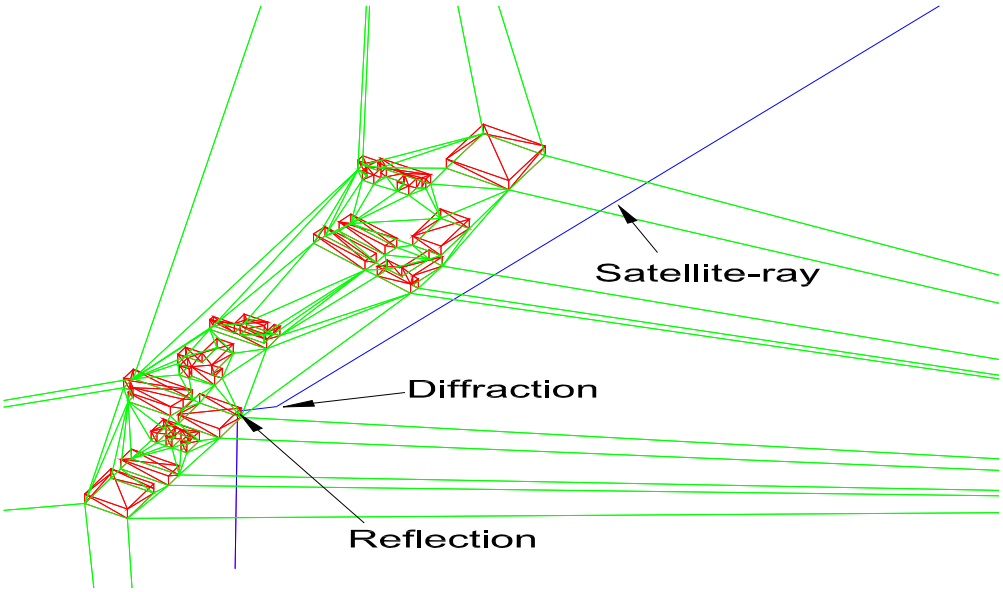


Figure 45. Detail of the Figure 44.



4.4 CONCLUSIONS AND FUTURE WORK

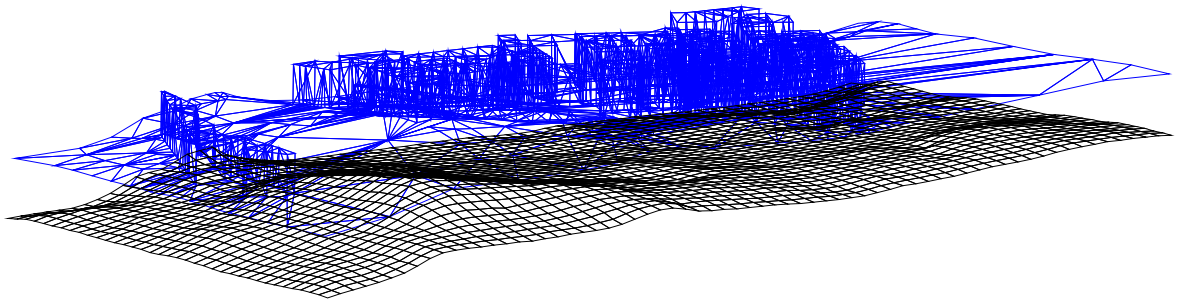
4.4.1 Conclusions for 2-Manifold Build for Terrain and Building Modeling

(i) The presented methodology shows a process, which allows the integration of terrain and housing database in four steps, with two different format of terrain. The obtained model can be used for different applications such as Ray-Tracing, Telecommunication, Virtual Reality and Agriculture, among others.
(ii) The modeling decimation step permits an improvement of the geometric characteristics of the resulting 2-manifold, which represents housing and terrain. This step permits a correct execution of downstream applications, as it controls the amount (and in future releases, the quality) of data.

4.4.2 Future work for 2-Manifold Build for Terrain and Building Modeling

(i) The presented method needs an evaluation, which may be made with a comparison (Hausdorff distance) between the obtained 2-manifold model and the GIS data (original data), to get the reliable and fidelity of the obtained model (Figure 46).

Figure 46. Building and terrain 2-manifold evaluation.



5. GENERAL CONCLUSIONS

((i) This project presents different methods and algorithms with deterministic and heuristic techniques, which can be applied in the administration, handling, transformation, operation and treatment of 2-manifolds.

(ii) In addition, this project shows different processes for modeling and reconstructing surfaces, starting or getting 2-manifolds for several engineering applications.

(iii) The algorithms presented in this project have geometric and topologic considerations, which achieve faithful models with good quality.

(iv) Finally, this project does not only apply mathematic fields such as geometry or topology, also conceptual reasoning is applied to improve time and avoid errors in engineering calculations.

BIBLIOGRAPHY

RUIZ, O. Understanding CAD / CAM / CG. American Society of Mechanical Engineers, ASME. Continuing Education Institute. Global Training. NY-USA. 2002. ASME Code GT-006.

RUIZ, O., CADAVID, C., GRANADOS, M., PEÑA, S. and VÁSQUEZ E. 2D Shape Similarity as a Complement for Voronoi-Delone Methods in Shape Reconstruction. United Kingdom. 2005. Submitted to Elsevier Journal on Computer & Graphics.

RUIZ, O., CADAVID, C., GRANADOS, M., PEÑA, S. and VÁSQUEZ E. Usage of 2D Region Similarity for Surface Reconstruction from Planar Samples. Seattle, Washington-USA. November 10-13, 2003. Submitted to SIAM, Conference on Geometric Design and Computing.

FOTÁN, F.P., RUIZ, O. and PEÑA, S. Coupling Terrain and Building Database Information for Ray-Tracing Applications. Fortaleza-Brazil. November 17-19, 2003. Submitted to ClimDiff, Conference on Coupling Terrain and Building Database with Propagations Models for Loss Predictions.

RUIZ, O., CADAVID, C., VÁSQUEZ E., GRANADOS, M. and PEÑA, S. Virtual Perturbations to Solve Degeneracies in Voronoi-Delone Methods. Plzen-Czech Republic. January 31-February 4, 2005. Submitted to WSCG 2005.

RUIZ, O. and KARANGELIS, G. Boundary Representation of Anatomical Features. Computer Graphics Topics ISSN:0936-2770 2002;13(6): 24-25.

BAREQUET, B. and SHARIR, M. Piecewise-linear interpolation between polygonal slices. United Kingdom. 1996. Elsevier: Computer Vision and Image Understanding. 63(2):251-272.

BOISSONAT, JD. and GEIGER, B. Three Dimensional Reconstruction of Complex Shapes based on Delaunay Triangulation. San Jose, USA, 31 Jan-05 Feb 1993. Bellingham-USA. In: Acharya RS, Goldgof DB, editors. SPIE Proceedings Vol. 1905: Biomedical Image Processing and Biomedical Visualization: SPIE Press, 1993. p. 964-975.

FOMENKO, A. and KUNII, T. Topological Modeling for Visualization. Tokyo. 1997. Springer Verlag.

SHINAGAWA Y., KERGOSIEN Y. and KUNII, T. Surface coding based on Morse Theory. Los Almitos, California-USA. 1991. IEEE Computer Graphics and Applications. 11(5):66-78.

DE FLORIANI, L., PUPPO, E. and MAGILLO P. Applications of Computational Geometry to Geographic Information Systems. Genova-Italy. 1999. CiteSeer. p 16-27.

SÁKÖSY, F. Gis Functions-Interpolation, Periodica Polytechnica Ser. Civ. Eng. 1999. Volume 43, p. 63 - 86.

BARBOSA, R.L. and CUSTÓDIO, J.F. Geração de um Modelo Digital de Terreno por Aproximações Sucesivas. São Paulo-Brazil. 1998. p. 5, Unesp.

VAN BEERS, W. and KLEIJNEN, J. Kriging For Interpolation in Random Simulation. Tilburg-Netherlands, Oct. 2001.Center-Discussion paper. p. 22

GEBHARDT, A. PVM Kriging with R. Klagenfurt-Austria. 2003. DSC 2003 Working Papers. p. 10.

POPOVICI, V and. THIRAN, J.P. PCA in Autocorrelation Space. Quebec-Canada. August 2002. ICPR 2002. p. 4.

RUIZ, O. and NEUGEBAUER, P. Topologically Consistent Partial Surface Reconstruction from Range Pictures. Las Vegas-USA. Nov. 19-23 2000. Computer Graphics and Imaging (CGIM) conference of IASTED (Intl. Assoc. of Science and Technology for Development).

MURTA, A. A General Polygon Clipping Library. Manchester-UK. 1999. Advanced Interfaces Group, Department of Computer Science. University of Manchester.

VÁSQUEZ, E. Surface Reconstruction. Medellín-Colombia. 2003. Final Project to obtain the B.Sc. Diploma in Computer Science. CAD / CAM / CAE Laboratory-EAFIT University.

GEIGER, B. Three-Dimensional Modeling of Human Organs and its Application to Diagnosis and Surgical Planning. Sophia Antipolis, France. 1993. Institut National de Recherche en Informatique et Automatique-INRIA. Technical Report RR-2105.

KRISHNAN S. and MANOCHA D. Computing Boolean Combinations of Solids Composed of Free-form Surfaces. North Carolina-USA. 1996. Department of Computer Science-University of North Carolina.

GARCÍA, M.J. Lecture Notes in Advanced Numerical Analysis. Medellín-Colombia, January 2003. EAFIT University.

RUIZ, O., CADAVID, C. and GRANADOS, M. Evaluation of 2D Shape Likeness for Surface Reconstruction. Journal Anales de Ingeniería Gráfica ISSN:1137-7704 2002;1(15):16-24.

FELICÍSIMO, A.M. Modelos Digitales del Terreno. Oviedo-Spain. 1994. University of Oviedo.

PIEGL L. and TILLER W. The NURBS Book. New York-USA. 1997. Springer-Verlag. Second Edition. ISBN 3-540-61545-8.

HASTIE, T. and STUETZLE, W. Principal curves. Alexandria, Virginia-USA. 1989. Journal of the American Statistical Association, Vol. 84, p. 502-516.

KEGL, B., and KRZYZAK, A. Piecewise linear skeletonization using principal curves. Los Almitos, California-USA. 2002. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, N. 1, p. 59 - 74.

SHEWCHUK, J.R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. Philadelphia, Pennsylvania-USA. May 1996. Applied Computational Geometry, ACM. p. 124-133.

INTERNET BIBLIOGRAPHY

Schlumberger Oilfield Glossary. [On line]. [Visited in September 06, 2004]. Available from: http://www.glossary.oilfield.slb.com/.

Fact Index, Mathematical and Natural Sciences. [On line]. [Visited in September 05, 2004]. Available from: http://fact-index.com/.

The Free Dictionary by Farlex. [On line]. [Visited in September 06, 2004]. Available from: http://encyclopedia.thefreedictionary.com/.

Outfo Open Information Project. [On line]. [Visited in September 10, 2004]. Available from: http://outfo.org/.