GOCAD ENSG Conference
3D Modeling of Natural Objects : A Challenge for the 2000's
4 - 5 June 1998

OMEGA: a CAD-based approach to Geological Modelling [1]

Michael Floater [(1)], Yvon Halbwachs [(1)], Øyvind Hjelle [(1)] and Martin Reimers [(1)]

[(1)] SINTEF Applied Mathematics, P.O.Box 124 Blindern, N-0314 Oslo, Norway.
Phone: 47 22 06 73 00 - Fax: 47 22 06 73 50
E.mail: Michael.Floater@math.sintef.no, Yvon.Halbwachs@math.sintef.no,
Oyvind.Hjelle@math.sintef.no, Martin.Reimers@math.sintef.no

1 Introduction

The object of this paper is to summarize a new approach to generating geological models from measured scattered data. Our approach is to base the models on CAD (Computer Aided Design) technology and to add mathematical tools for creating explicit and freeform surfaces from the scattered data. Moreover we build the 3D topological model using G-map technology and generic surface/surface intersection algorithms.

Our approach was implemented as a part of the ESPRIT project OMEGA (Object Oriented Methods Environment for Geoscience Applications), and so this paper is to some extent a review of the relevant parts of the OMEGA project. The resulting software, which we call the *OMEGA platform*, builds on the CAD system CAS.CADE from Matra Datavision, SISCAT, a scattered data library from SINTEF, and the topological G-map concept.

2 The OMEGA Project

CAD models are usually constructed from the designer's mental image of the object he wishes to create. The construction process is thus basically divided into two steps: the first is to create geometries, and the second is to glue them together through the topology. All the tools needed for these operations are *constructive*: complex objects are created by applying Boolean operations to basic objects (Figure 1).

Geoscience models are usually constructed in a different way, since the user wants to reconstruct real geological entities from incomplete data. To do so, he has to *interpret* data obtained by physical measurements, such as seismic surveys. As this interpretation is usually not unique (mainly due to lack of data), several *construction tools* are required in order to make the modelling almost automatic (Figure. 1).

The goal of the geological modelling part of the *OMEGA project* is to develop such a set

[1] This work was supported by the European ESPRIT project OMEGA, No. 23245.

of construction tools. The approach is to base the modelling on the CAD system CAS.CADE and to add functionality for automatically creating surface approximations to scattered data and to intersect them to build up a topological model based on G-maps. In addition a user interface allows the user to create and interpret surfaces and G-maps from a geologist's point of view (see [21] for more information about the Omega Project).

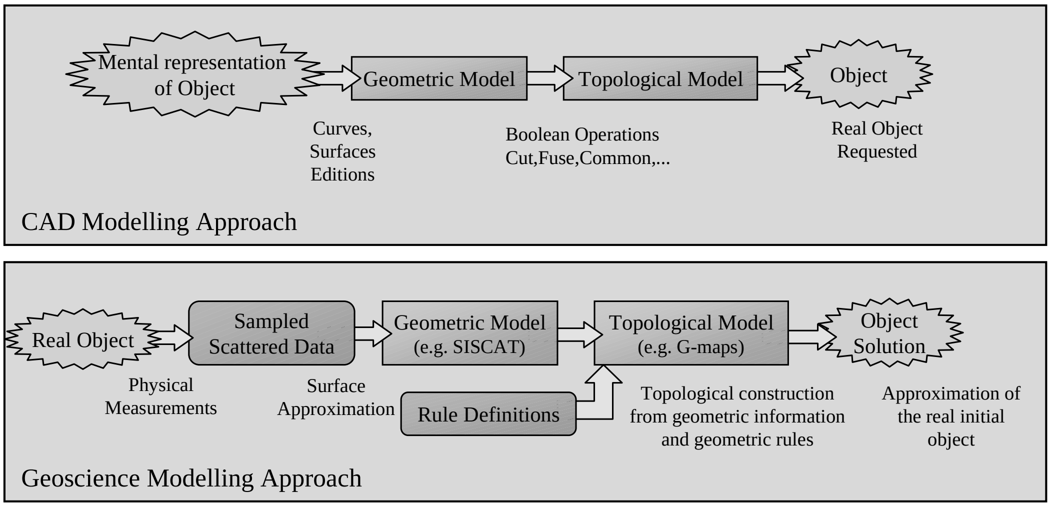

Figure 1: Comparison between CAD modelling and Geoscience modelling.

3 Surface approximation from scattered data

A crucial step in our approach to geological modelling is the reconstruction of geological surfaces from measured data. Generally speaking there are two kinds of surface occurring in geology: explicit and non-explicit.

A surface is *explicit* if it can be represented in the form

$$z = f(x, y).$$

For example z can represent the height over sea level of terrain at a location (x, y).

Otherwise a surface is said to be *non-explicit*. For example, a region of terrain in which there is an overhanging cliff is a non-explicit surface. Typically non-explicit surfaces are either represented *implicitly*, in other words as the set of three-dimensional points (x, y, z) satisfying an equation of the form

$$f(x, y, z) = 0,$$

or *parametrically*, in other words in the form

$$\mathbf{f}(u, v) = (x(u, v), y(u, v), z(u, v)).$$

In a typical geological application we will be given a set of three-dimensional measured points (x_i, y_i, z_i), called *scattered data*, sampled from some unknown geological surface. Thus the scattered data will be explicit or non-explicit depending on the corresponding type of underlying surface. The method (and difficulty) of making a surface approximation to the scattered data depends crucially on whether it is explicit or non-explicit.

3.1 Approximating explicit data

The basic scattered data problem when we have explicit data (x_i, y_i, z_i), $i = 1, \ldots, N$, is to find a scalar valued function $f(x, y)$ that approximates or interpolates the data. For interpolation we have $f(x_i, y_i) = z_i$ whereas for approximation we settle for $f(x_i, y_i) \approx z_i$.

There are several known methods for approximating explicit scattered data, for example piecewise polynomials on triangulations, radial basis functions, and least squares approximation relative to some basis functions such as tensor-product B-splines. Generally there is no single method which always yields the best result and the quality of the resulting surface depends heavily on the distribution of the points (x_i, y_i). Figure 2 illustrates some kinds of distributions which occur in practice and shows from top left to bottom right: dense; sparse; track; contour; cluster; and mixed data. In particular, interpreted seismic data occurs frequently as so-called track data.

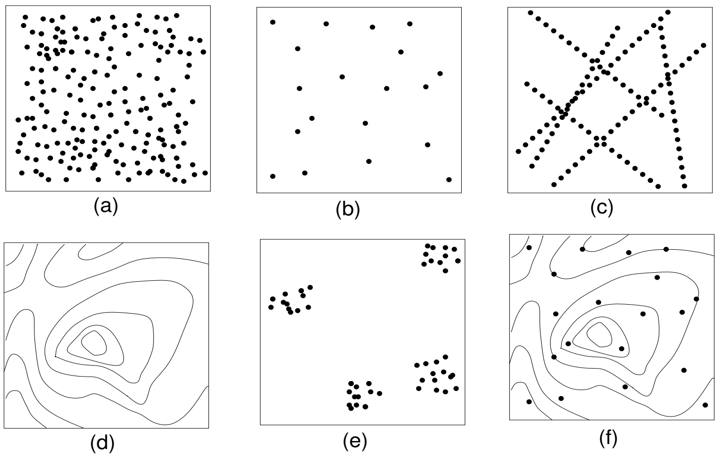

Figure 2: Various scattered data sets in the plane.

A method which we have used in this project is the so-called *mask method* [1], [2] which finds a rectangular grid of values which simultaneously approximate the data and are also smoothed via the biharmonic equation. The grid of values can subsequently be interpolated by a tensor-product spline, constructed for example from Coon's patches, see Farin [7].

A simpler method which we also use in the project is to triangulate the two-dimensional points (x_i, y_i) and make the piecewise linear interpolant to the heights z_i. However the resulting function is not smooth, merely continuous.

3.2 Approximating non-explicit data

When the data is non-explicit, the task of approximating with a surface is much more difficult and a lot of research has been carried out on this over the last few years. Here we outline two approaches which we have applied in our project.

3.2.1 Offset Surfaces

One method of approximating non-explicit scattered data is to use a parametric *base surface* $\mathbf{b}$ with respect to which the scattered data is 'explicit', that is for each point on the (unknown) surface the data was sampled from, there should be a unique nearest point to it on the base surface. We then approximate the data with a surface $\mathbf{f}$ which is an *offset surface* to the base surface. The approximation $\mathbf{f}$ will have the form

$$\mathbf{f}(u, v) = \mathbf{b}(u, v) + e(u, v)\mathbf{n_b}(u, v) \tag{1}$$

where $\mathbf{b} : D \to R^3$ is the base surface, $\mathbf{n_b} : D \to R^3$ is the unit normal vector to the base surface, $e : D \to R$ is the offset function and D is the parameter domain in R^2. This approach was described in [8].

If the scattered data is projectable onto a plane, the straight forward approach of calculating the least square fitting plane as the base surface will be adequate in many cases and we will be able to generate a parametric surface more or less automatically. If the scattered data is not projectable onto a plane, finding a base surface requires more user-interaction. Also if the data is 'close' to being non-projectable the least squares plane may not be a suitable base surface. Often the end-user must design the base surface interactively.

3.2.2 Surface Approximation of a Triangulation

An alternative method of approximating non-explicit scattered data is to

(1) build a triangulation from the data,

(2) parametrize the triangulation,

(3) approximate the parametrized triangulation by a parametric surface.

The weak point in this method is step (1) for all known methods of constructing triangulations are somewhat heuristic and can fail if the data is badly distributed. We note however that if the data is explicit, a *Delaunay* triangulation can always be constructed (efficiently) from the data. When the data is genuinely non-explicit, most methods for step (1) construct a triangulation which only *approximates* the data, that is the scattered data points need not be the vertices of the triangular facets of the triangulation. An algorithm for general scattered data is proposed by Hoppe and DeRose et al. [18]. In the special case when the data is in the form of contours in parallel cross-sections, there are several algorithms, see e.g. Schumaker [22] and Floater and Westgaard [11]. The basic approach in all these triangulation algorithms is to first construct a function $f(x, y, z)$ whose zero set approximates the data and from it generate a triangulation using a technique like 'marching cubes' (see [11]).

We assume next that we have a triangulation from the scattered data. In the case of explicit data, the triangulation will be topologically equivalent to a disc (provided the domain is simply connected, i.e. without holes). In the non-explicit case we have to make this an assumption. If it has, for example, the topology of a sphere or torus it should be decomposed into sub-triangulations which are topologically equivalent to a disc. Let us assume that the vertices of the triangulation are labelled (x_i, y_i, z_i).

Our method of solving step (2) is to *parametrize* the triangulation. A *parametrization* of a triangulation is any planar triangulation with the same topology. However in order that the parametrization will be useful for approximating the triangulation with a visually smooth surface, it is important that the parametrization mimics the geometry of the surface triangulation.

For the purpose of computing a suitable parametrization we have used the method described in [9]. Roughly speaking, the method consists of first mapping the boundary of the triangulation into some convex polygon in the plane. If the parametrization will subsequently be used to generate tensor-product spline surfaces then a suitable choice of polygon is a square. Then the interior vertices of the parametrization are found as the solution to a linear system defined by demanding that every vertex be a convex combination of its neighbouring vertices. By choosing the convex combinations appropriately the geometry of the triangulation will tend to be preserved. Figure 3 shows a triangulation and its parametrization in the unit square.

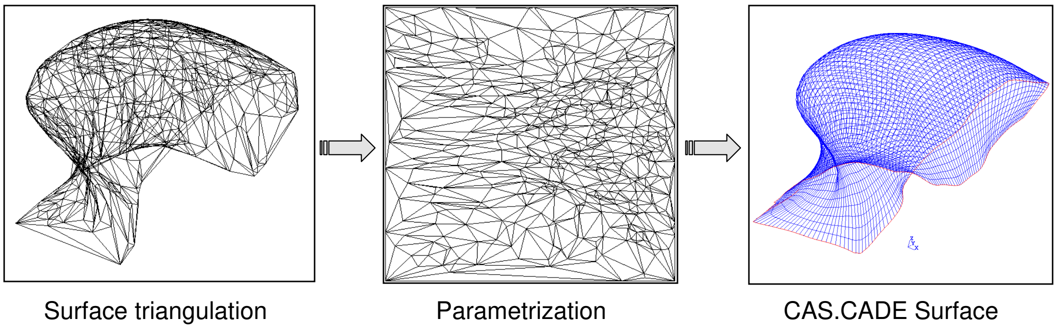

Figure 3: Triangulation, parametrization and surface approximation.

At this stage we have a parametrization whose vertices are (u_i, v_i) corresponding to the vertices (x_i, y_i, z_i) of the 3D triangulation. Using any chosen scattered data method, three smooth functions $x(u, v)$, $y(u, v)$, $z(u, v)$, can be constructed independently to satisfy

$$x(u_i, v_i) \approx x_i, \quad y(u_i, v_i) \approx y_i, \quad z(u_i, v_i) \approx z_i. \tag{2}$$

A parametric surface $\mathbf{f}(u, v)$ satisfying $\mathbf{f}(u_i, v_i) \approx (x_i, y_i, z_i)$ finally results from setting

$$\mathbf{f}(u, v) = (x(u, v), y(u, v), z(u, v)).$$

If interpolation is required then we must demand strict equality in equation (2). Figure 3 shows a surface approximation generated by the mask method (in each of the three components).

By further sampling the surface $\mathbf{f}$ on a rectangular grid in the parameter domain and performing a B-spline interpolation we can convert $\mathbf{f}$ to a tensor-product spline surface $\mathbf{f}'$, which is a common type of surface representation in computer aided design. An alternative method is to generate a B-spline surface $\mathbf{f}'$ directly from the parametrized data using a least squares method with a smoothing term [10].

4 The SISCAT library

SISCAT is an object-oriented C++ library developed at SINTEF Applied Mathematics for scattered data problems. It provides a variety of methods and utilities for surface approximation and interpolation from scattered data, such as geological measurement data. The kernel of SISCAT is used in several software systems e.g. in geological modelling [4] and modelling of medical data [12], see [23].

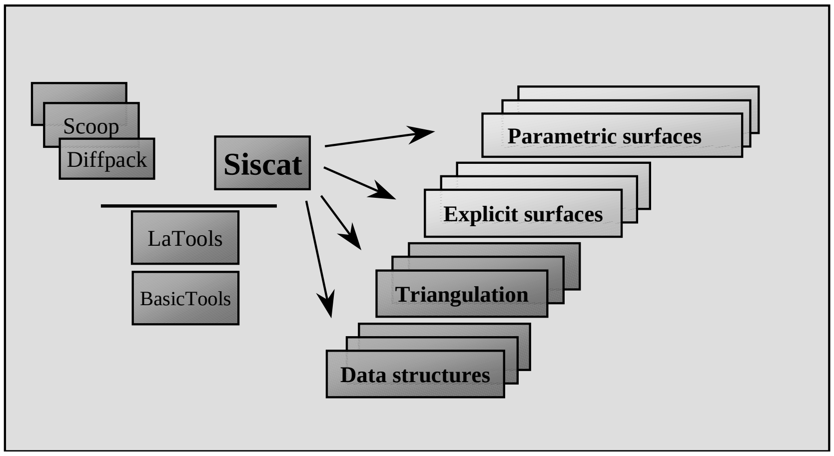

Figure 4: The SISCAT library.

4.1 Basic data structures

There are basic data structures and utilities for handling scattered data, such as points, point-sets, spatial query structures, domain descriptions, IO-routines, visualization etc.

The most interesting feature of the basic data structures from a users point of view is the interface to scattered data, encapsulated in the abstract base class `SsDataItem`. Through this class the user specifies the interface to the user data, often stored in specialized data formats that are suitable for the specific application, e.g in matrices, lists or special array structures. In practice a user will either use one of the standard data formats available in SISCAT or implement a few simple access routines in a class derived from `SsDataItem`. In this way all access to data is performed in a uniform way throughout the library.

There is also a mechanism for assigning properties, called item types, to the data-items. The user can associate data with tags that the approximation methods can recognize, such as the type of data. Examples of tags currently recognized are: scattered data, contour data, track data, grid data, data describing discontinuities and boundary data. The user can also define new item types suitable for the specific application.

A set of `SsDataItem`s constitutes a `SsDataCollection`. This class administrates a set of `SsDataItem`s and communicates with the approximation methods. The user can for instance subtract subsets of the data through the item type system.

4.2 Explicit surfaces

This module consists of a class hierarchy of methods for approximating or interpolating *explicit* scattered data, see section 3.1. The abstract base class `SsExplicit` contains a generic interface to the users data, encapsulated in a `SsDataCollection`, and methods for creating approximations, evaluation, derivatives etc. The methods currently implemented include: Shepard's method, radial basis functions, polynomial least squares methods, triangulations, cosine expansion and the mask method. See the report [17] and the references therein for a description of these methods. Each specific approximation method is implemented in a class derived from `SsExplicit`, such as `SsMaskMethod`; this is illustrated in Figure 5 where solid arrows indicate class inheritance and the dotted lines indicate containment. The 'n' indicates a list. The classes depicted as boxes with shadows are abstract.

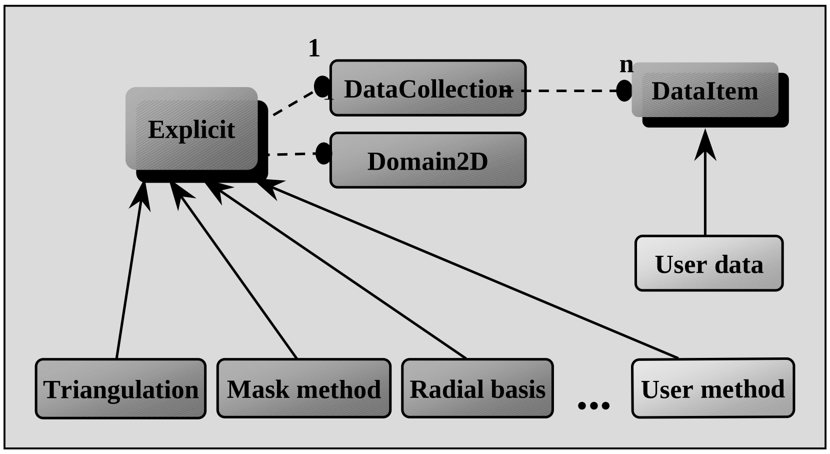

Figure 5: `SsExplicit`: C++ class hierarchy for explicit surfaces.

As pointed out earlier, the methods have different characteristics, strengths and weaknesses and no single method suits all kinds of problems. In the framework of SISCAT a user can choose between a set of methods and use the one or the ones that are best suited for the application. Figure 6 depicts contour plots of approximations using various methods in SISCAT. We find that the mask method often performs well on data sets from geological measurements since the data sets are often very large and contain faults (discontinuities).

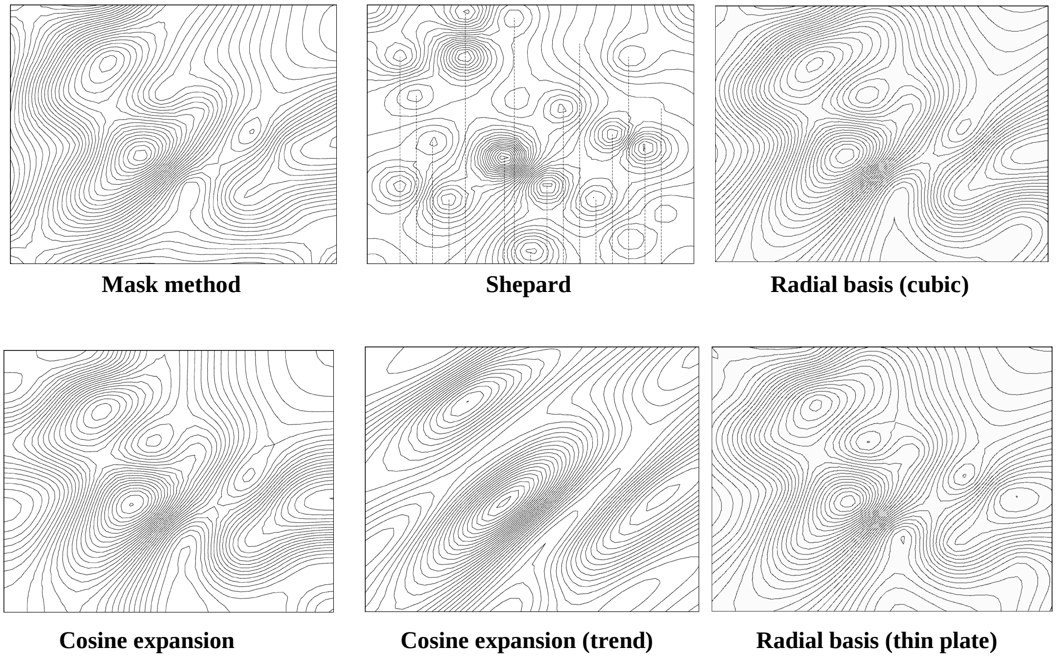

Figure 6: Contour plots of several SISCAT surfaces generated from the same data set.

4.3 Parametric surfaces

This module relies on explicit scattered data methods to approximate data that are not necessarily explicit with respect to the xy plane. Given a data set (x_i, y_i, z_i), $i = 1, \ldots, N$, we wish to find a vector valued function $\mathbf{f}(u, v)$ that approximates or interpolates the data. Due to the difficult nature of this reconstruction problem, some degree of user intervention is often necessary.

We currently have two methods for non-explicit data available in SISCAT: offset surfaces and component-wise approximation.

4.3.1 Offset surface approximation

When the data is explicit with respect to a parametric base surface, we can approximate by an offset surface $\mathbf{f}$, defined in equation (1). The class `SsPrmOffset` contains both the base surface $\mathbf{b}(u,v)$ and the explicit surface $e(u,v)$ which are enough to represent the offset surface. The base surfaces are organized in a class hierarchy with an abstract base class `SsPrmBaseSurface` and subclasses that implement generic methods for finding unique parameter points for a spatial point, surface normals, differentials, initialization etc. The currently implemented base surfaces are: plane, sphere and cylinder. Our experience is that the plane base surface can be used in most situations occurring in data from geological measurements, and it is also the simplest to initialize automatically. An application programmer might implement new base surfaces specialized for the application.

The class `SsPrmOffset` can also be used for creating triangulations over point sets that are explicit w.r.t. a plane, thereby creating the basis for the method of parametrizing the data, see section 3.2.2.

4.3.2 Component-wise approximation

This method relies on parametrized data, i.e. each point (x_i, y_i, z_i) is associated with a point (u_i, v_i) in the parameter domain. The process of assigning parameter values to scattered data is a nontrivial task. The method described in section 3.2.2 is available in SISCAT through an interface to a stand-alone library developed at SINTEF.

Regarding each component of $\mathbf{f}(u,v)$ as explicit functions

$$x(u,v), y(u,v), z(u,v)$$

of the parameters u and v, we compute explicit approximations over each of the data sets

$$(u_i, v_i, x_i), (u_i, v_i, y_i), (u_i, v_i, z_i)$$

utilizing methods in the `SsExplicit` hierarchy.

5 CAS.CADE

The CAS.CADE Object Libraries are reusable C++ components and modules which help to produce domain expertise or scientific applications. A typical application developed with CAS.CADE would be 2D or 3D geometric modelling, for instance, in general purpose or specialized CAD systems, manufacturing or analysis applications, simulation applications as well as illustration tools.

The CAS.CADE Object Libraries were carefully designed in order to be truly modular and extendible, see Figure 7. As such, they provide data structures for geometric and topological modelling, graphical display and graphical selection. Complex algorithms and high level programming interfaces have been implemented as separate classes, see [5] or [20] for more information about CAS.CADE.

5.1 Geometric modelling with CAS.CADE

The geometric modelling classes in CAS.CADE provide resources to create and modify geometric models based on boundary representation (B-rep). Separate APIs to 2D and 3D

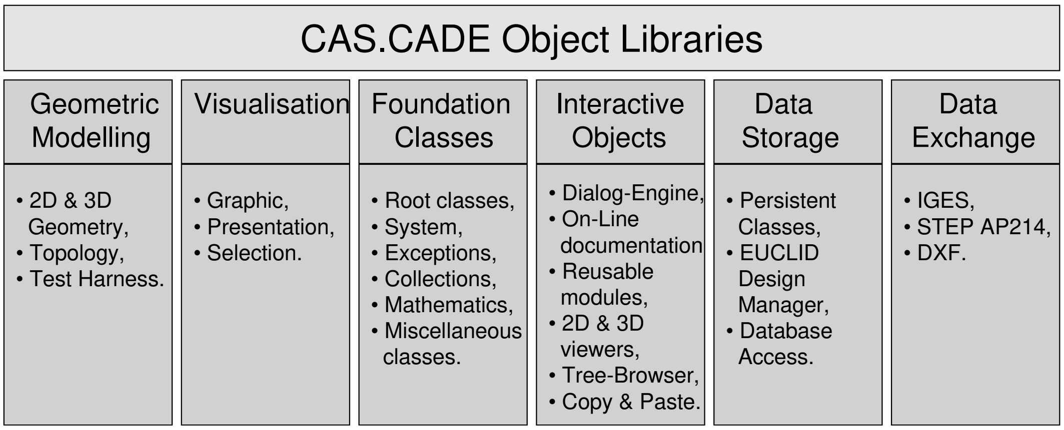

Figure 7: Domains of the CAS.CADE Object Libraries.

geometry, abstract topology and B-reps are provided. More precisely, the geometric classes consist of a set of general purpose services used to model a hierarchy of 2D and 3D STEP conforming geometric objects supporting analytic (conic and quadric) or rational and polynomial (NURBS and Bézier) curves and surfaces. Numerous complex algorithms operating on these objects are provided including:

- intersection between two 2D curves, a curve and a surface and two surfaces, either analytic or polynomial and rational,
- projection of curves on surfaces and calculation of a curve in the parametric space of a surface from a 3D curve,
- computation of a surface from a closed set of curves.

In the context of geological modelling in OMEGA, the most interesting benefit of these classes is their generic definition. For example, CAS.CADE provides generic interfaces to parametric curves and surfaces. An application programmer can use the concept of inheritance in the C++ programming language to include his own representation of geometric objects, and use them with all the existing algorithms of CAS.CADE as any other predefined CAS.CADE object. The only requirement is to redefine the virtual methods of the base classes in CAS.CADE in the new derived classes. In case of an alien parametric surface representation, such as a SISCAT surface, the only methods that need to be implemented are evaluation of position and first partial derivatives.

5.2 Interface between SISCAT and CAS.CADE

As explained in the previous paragraphs, the CAS.CADE libraries are originally dedicated to developing CAD applications. Adding the scattered data modelling functionality of SIS-

CAT to CAS.CADE is a way of combining two complementary object libraries. On the one hand, CAS.CADE provides the development environment and complex algorithms necessary for building geoscience applications, such as surface-surface intersection algorithms, topological operations, etc. On the other hand, SISCAT enriches CAS.CADE with various surface reconstruction algorithms. Linking together these two object libraries is possible in several ways. Globally our objective is to:

- **Create a new component in CAS.CADE for constructing surfaces from scattered data.**
  The concept of *reusable components* in CAS.CADE may be used for integrating functionality from SISCAT, see Figure 8.

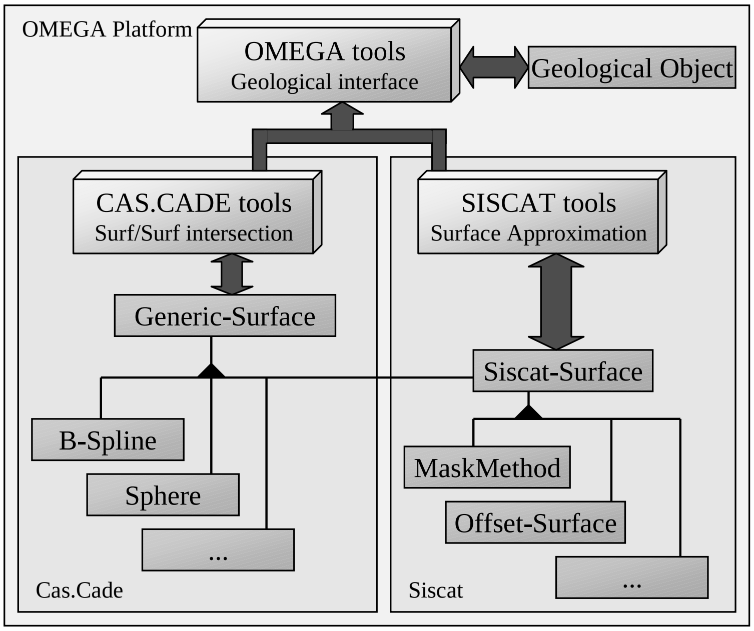

Figure 8: Integration of SISCAT and CAS.CADE in the OMEGA platform.

- **Create a high level application interface (API) for the user.**
  The user of this component should not be involved in the technical mathematical concepts used by SISCAT. Only geological terms should be dealt with, such as point sets, wells, layers and faults.

- **Create a new component in CAS.CADE for manipulating triangulations.**
  CAS.CADE has no suitable triangulation module but this functionality can be provided by SISCAT. The triangulation library in SISCAT provides several algorithms for building and modifying triangulations, e.g., optimisation, data reduction, refinement, and parametrization for converting surface triangulations to parametric surfaces.

- **Be able to use SISCAT surfaces as CAS.CADE surfaces.**
  Basically there are two different approaches for operating on surfaces created with SISCAT in CAS.CADE:

  1. By converting a SISCAT surface to a *parametric spline surface* as already defined in CAS.CADE. The CAS.CADE surface is created by sampling the SISCAT surface on a regular grid in each of the components x, y, and z, and then interpolating or approximating the samples to obtain a parametric CAS.CADE surface.

     In order to get a good approximation of the original SISCAT surface, the grid must be relatively dense, e.g. 300 grid lines in each direction and often more. Thus the computational cost of carrying out the conversion step, and at a later stage of performing intersection operations, might be beyond the limits required in interactive geoscience systems.

  2. By defining new CAS.CADE surface classes using inheritance in C++ as explained in section . Thus the generic surface hierarchy in CAS.CADE is enriched with new surface classes from SISCAT. In this way we can utilize functionality in CAS.CADE working on surfaces as defined in the generic CAS.CADE classes, for example generic surface-surface intersection and miscellaneous topological operations.

     The generic surface-surface intersection algorithms in CAS.CADE depend on evaluation of position and partial derivatives only. In addition it is required that the surfaces are C^1 (continuous first partial derivatives). Thus, surface-surface intersection can be performed on alien surfaces, e.g. SISCAT surfaces, if the virtual evaluation functions are available in the derived classes.

  In the OMEGA project, SISCAT has been extended to fulfil the requirements needed to implement both solutions. The results regarding the second approach above are promising. It is now possible to intersect two SISCAT surfaces, or a SISCAT surface and a CAS.CADE surface. This is also a classical example of object oriented programming; SISCAT alone cannot perform surface-surface intersection, and CAS.CADE performs intersection operations on SISCAT objects without knowing anything about the internal representation in SISCAT.

6 Generalized maps

In the previous sections, we focused on the geometric aspect of the modelling process of geoscience objects. We will now concentrate on the next step of the modelling process which concerns the topological aspect.

Since geological surfaces can be very discontinuous, not all the cases can be taken into account by the geometry alone, especially when dealing with holes. As these problems are often crucial when analyzing geoscience models, they cannot be neglected. This is one of the reasons why a consistent topological model has to be used.

6.1 The standard CAS.CADE topology

The standard CAS.CADE topological library is based on a *boundary representation model* (B-rep) which means that, for example, in the 3-dimensional case, a volume is described by its boundary faces. This model can handle non-manifold (or mixed-dimensional) topologies. A non-manifold model is a single model which mixes 0D (points), 1D (curves), 2D (surfaces) and 3D (volumes) objects. For example, a single object may be represented with several distinct bodies containing embedded curves and surfaces connected or not to the outer boundary. *Topological* objects are based upon *Geometric* object restrictions or sequences.

This library provides an abstract data structure, the *shape*, which defines the relations between geometric entities. It supports:

- **Vertex**, an extremity of an edge,
- **Edge**, a topological edge bounded by two vertices,
- **Wire**, a sequence of edges connected by their vertices,
- **Face**, a portion of a surface bounded by an external closed wire, or internal closed wires,
- **Shell**, a collection of faces connected by some of the edges of their wire,
- **Solid**, a portion of the 3D space bounded by a shell, and
- **Compound Solid**, a collection of solids.

The wire and the solid may be infinite or closed. The vertex, the edge and the face of a shape respectively refer to a point, a curve and a surface of the geometry, in the object space. The face may also refer to a triangulation that approximates the parametric surface.

Based on these shapes, the library provides a large set of tools:

- **construction tools**, basic construction of shape from geometric objects or more complex primitives such as sweeping, pipe, prism, or revolution of a body around an axis,
- **boolean operations**, fuse (union), common (intersection), cut (removal) and section by a surface.
- **miscellaneous computations**, triangulation in each face of the boundary of a solid while maintaining edge consistency with adjacent faces, local or global properties of objects such as edge length, area of a face, volume of a solid, etc.

6.2 Extending CAS.CADE with G-maps

The various problems encountered in geology require that the modelling process should be able to handle 2D information (such as connectivity between different faults or their geometry) as well as 3D information (such as connectivity between geological blocks or the nature of these blocks). It must also be easy to access 0D or 1D information, i.e. the edges and vertices of blocks which represent the intersections of faults and their extremities. Lastly, the large amount of data involved requires that the access to this information is efficient.

The basic tools in CAS.CADE were not designed with these issues in mind, although the topological model used is sufficient. Implementing the required functionalities with the basic CAS.CADE topological model is indeed not a good solution because of a lack of information in the data structure of the topology. This implies that the tools provided to access this missing information are either non-existent, or are very time consuming, and so impossible to use in a geoscience context.

Our solution was to enhance CAS.CADE with more advanced topological modelling, making sure that the enhancement is compatible with the original CAS.CADE. Recent work by BRGM in [14] and [15] has shown that the concept of 3-dimensional generalized maps (*3G-maps*) can be successfully applied to describe fault networks. This work has been implemented in a CAS.CADE environment, and this CAS.CADE component has been reused within the OMEGA project. More information about the G-map concept can be found in [3], [6] and [19].

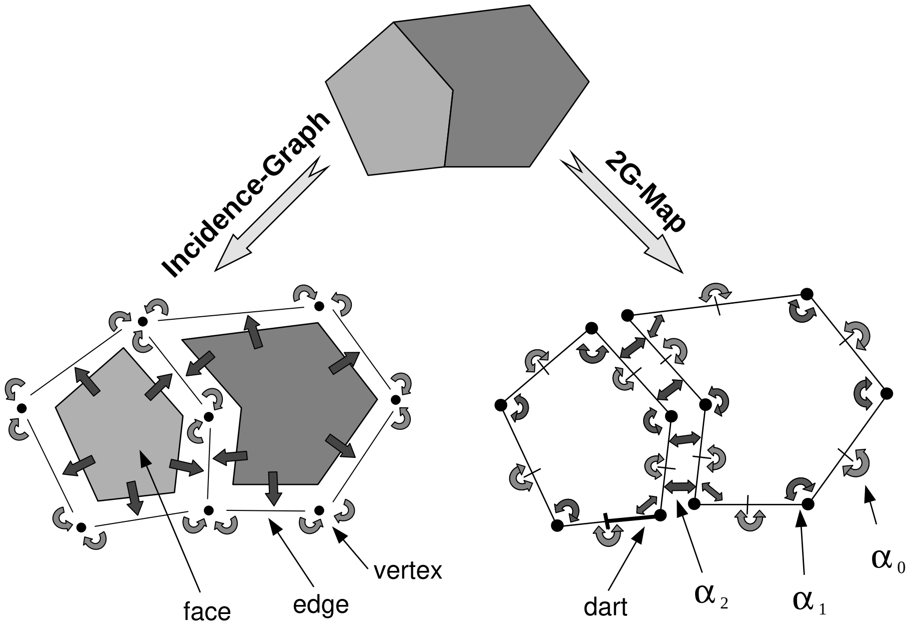

Figure 9: Comparison between Incidence Graph and G-maps.

Figure 9 shows briefly, by way of a 2-dimensional example, what the main differences are between the standard CAS.CADE topological model (Incidence Graph representation) and its equivalent based on a 2G-map (Generalized Map of dimension 2). At the top of the figure is a 2D object composed of two faces glued along two common edges and sharing three vertices. At the bottom left its topology is shown using incidence graph topology. The cells are decomposed in each dimension (vertex, edge and face), and the different topological links are displayed using arrows. At the bottom right, the same object is shown using a 2G-map. The cells are built from a single topological element called a *dart*, which is similar to a *half-edge*. All the local topological information is stored in darts. From one dart one has *direct access* to the other extremity of the edge it belongs to (0-dimensional link, or α_0), its adjacent edge (1-dimensional link, or α_1), or its adjacent face (2-dimensional link, or α_2). This example shows clearly the missing links of an incidence graph representation. It is not directly possible to get access to an n-dimensional cell from an (n-1)-dimensional cell (e.g. direct link from a face to its boundary edges, but no direct link from a vertex to its various incident faces).

6.3 Topological specifications

The general objective was to add a new geoscience topological module to CAS.CADE with the following features:

- Built on a **low-level implementation** of a 3G-map library,
- Contains a **GeoScience extension** to the basic concept of 3G-map,
- **Hides the concept of G-maps** from the geoscientist user,
- **Fully compatible** with the standard CAS.CADE topology.

These features lead to (1) specifications concerning the basic implementation of the 3G-map concepts, and (2) specifications concerning the definition of geoscience objects and tools.

- **(1) Basic 3G-map concepts**
  When designing the low level 3G-map library, we had the following objectives in mind. First, we wanted to implement an object oriented 3G-map library, separating rigorously topology and geometry. Second, we wanted to provide tools to convert (in both ways) a 3G-map into a CAS.CADE topological shape. Third, we wanted to base the library on the generic geometric classes and algorithms provided by CAS.CADE such as the parametric surface classes and the surface-surface intersection algorithm. In this way, any of the SISCAT surfaces described previously (explicit, parametric or triangulation) can be used.

- **(2) GeoScience extension**
  We extended the basic 3G-map concepts to geological entities such as fault, layer or discontinuous blocks.

  The main tool required concerns the *solid modelling of fault networks.* This can be the starting point of the construction of many other geological models because it involves very often the same methodology.

  The general approach consists in describing a fault network by a set of faults, a working volume (to be subdivided) and a set of relations between faults. The construction process inserts each fault of the network individually into the initial non-subdivided working volume. As each fault is inserted, it creates a new subdivision of space which is used for the next fault insertion. In addition, the final 3G-map model must take into account the fact that some of the faults may stop on others, i.e. some faults must not geometrically cross over other faults. Geometric information, such as the intersection between two faces or the relation between two faults, specifies which topological operations are needed to process a subdivision. This semi-automatic process helps to reduce the number of tedious user interactions during the modelling of a fault network.

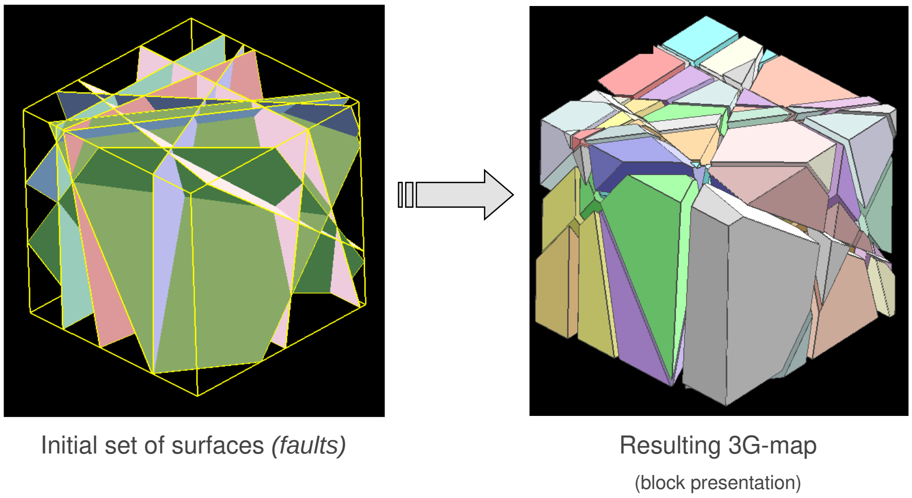

Figure 10: Solid modelling from a set of surfaces.

These specifications require several algorithms such as:

- creation of volumes delimited by a set of faces (3D operation, see Figure 10).
- creation of faces delimited by a set of edges (2D operation, see Figure 11).
- merging of volumes, faces or any other geological entities according to their own properties (for example, relations between faults).

Once a model is built, a large set of *exploration tools* are required to interpret efficiently and quickly a model, e.g. faults adjacent to a block or shortest path between two vertices. The 3G-map representation will help in describing easily these tools.

6.4 Putting things together with G-maps and CAS.CADE

Figure 12 shows the organization of the topological module within the complete OMEGA framework which includes geometry, topology and end-user interface. It rests on the generic

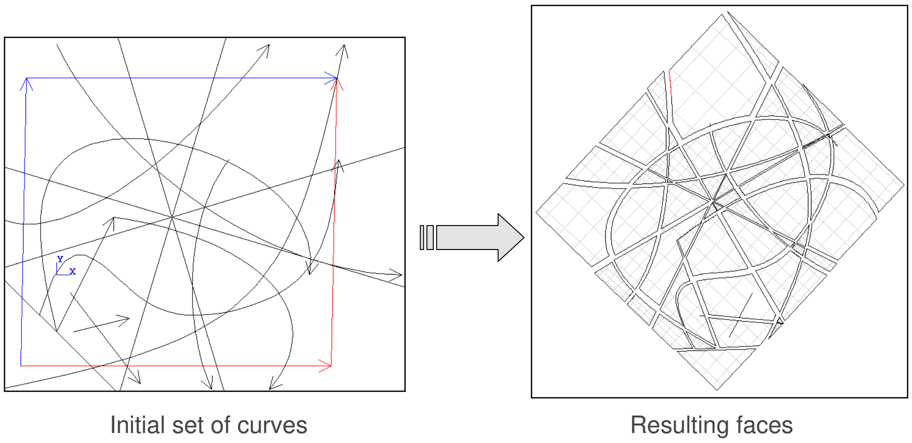

Figure 11: Face modelling from a set of edges.

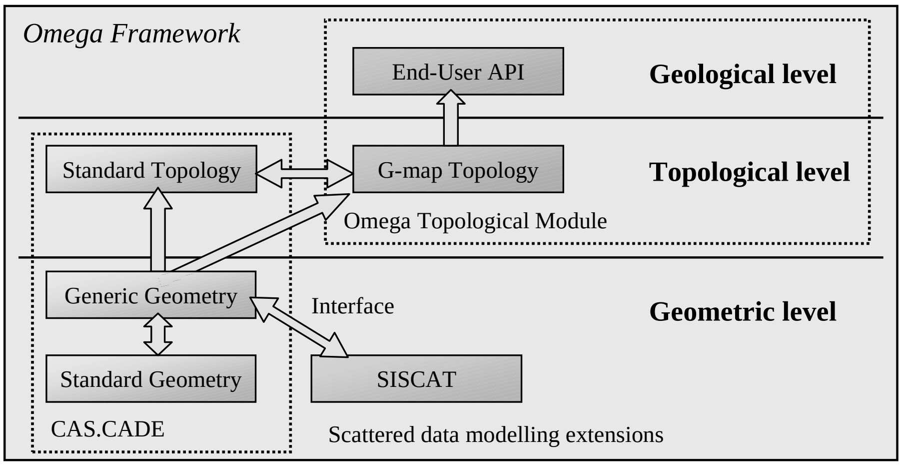

Figure 12: Extension of the standard CAS.CADE topological model with G-maps.

geometric library of CAS.CADE and is subdivided into two levels. A *kernel* based on G-maps and a *high level interface*, or end-user API, designed for end-users such as geoscientists. The specifications described in the previous section were used to design the library, see [16] for more informations about these specifications.

The *kernel* upgrades the standard CAS.CADE topology with the G-map concepts. This module could be used in any other domain different from geoscience since no end-user concepts will be implemented here. All the construction algorithms are based on the generic geometric classes, and intensively use the built-in intersection algorithms of CAS.CADE. It is thus crucial that the geometric module is efficient since all topological constructions depend on the interpretation of geometry.

We plan to introduce *topology levelling* in this module later to distinguish between *micro* and *macro topology*. This concept will help to model topology at different levels of detail; full geometric and topological detail in micro topology, and coarse outline of the model in macro topology, see [13] for more details.

The *end-user API* will use the G-map library to implement the end-user needs. This interface hides as much as possible the technical detail involved in the topology and geometry. This is possible thanks to the use of a collection of *geoscience oriented objects*.

The actual implementation allows the user to build solids delimited by a set of intersected surfaces (cf. Figure 10), to create surfaces delimited by a set of curves (cf. Figure 11), and to attach any information (e.g. geological or physical information such as the density of a block, or the roughness of a face) to any topological cell.

Further work still has to be done to enlarge the number of geometric cases treated when building a model from various geometric information. In particular, we do not currently take into account all possible non-manifold cases.

7 Conclusions and Further Work

In this paper we have discussed a methodology for bridging the gap between CAD and geological modelling. The essential features of our approach are the creation of smooth surfaces from scattered data and the modelling of complex topology with G-maps.

One of the strengths of this approach is that it allows smoother surfaces than triangulations which are traditionally used in geology. The great advantage of the generalized map concept over most existing topological models is the ease and efficiency with which one can navigate through the model. This makes it much easier for the geoscientist user to analyze the geological model. Our project required merging the two distinct software libraries SISCAT and CAS.CADE. The complexity of this task was greatly reduced by the fact that both are object oriented.

So far, the CAD approach to geological modelling used in OMEGA yields very promising results, but there are also some open questions and issues for further research. For example, the crucial step in the modelling process, namely building topology as generalized maps, relies heavily on the generic surface-surface intersection algorithms in CAS.CADE. Currently these algorithms only require evaluation of position and first derivatives. Although these algorithms show promising results, it still remains an open question whether all intersections will be guaranteed to be found, for example, in complex geological fault systems. Also, the computational speed of building very complex topological models might be unacceptable.

Although the process of modelling the topology of geological bodies is successful in simple cases, further work remains to be done concerning complex non-manifold geometric intersections. An extension of the G-map concept may be required to model such cases. Indeed, one can foresee such complex situations that, due to the computer's discrete representation, intersection algorithms alone are unlikely to find exactly the right topology and some user interaction would therefore be necessary.

References

[1] Arge E., Dæhlen M., and Tveito A. Approximation of scattered data using smooth grid functions. *J. Comput. Appl. Math.*, 59:191–205, 1995.

[2] Arge E. and Floater M. Approximating scattered data with discontinuities. *Numerical Algorithms*, 8:149–166, 1994.

[3] Bertrand Y., Lienhardt P., Francon J. and Dufourd J. F. *3-dimensional manifold modeling using generalized maps.* Report 93-03, Louis Pasteur University, Strasbourg, France, 1993.

[4] Dæhlen M., Bjaalie J.G. and Stensby T.V. Surface modelling of biomedical data. In: *Numerical methods and software tools in industrial mathematics* (eds. A. Tveito and M. Dæhlen), pp. 9-26. Birkhauser Publishing Cooperation, Boston, 1995.

[5] The CAS.CADE Home Page. World Wide Web document at `http://www.matra-datavision.fr/Products/Family/Cas.Cade`.

[6] Elter H. *Etude de structures combinatoires pour la représentation de complexes cellulaires.* PhD Thesis, Computer Science Departement, Louis Pasteur University, Strasbourg, France, 1994.

[7] Farin G. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, Boston, 1988.

[8] Floater M. S. Mathematical techniques for reverse engineering, SINTEF Report No. STF42 A96019, Oslo, 1996.

[9] Floater M. S. Parametrization and smooth approximation of surface triangulations, Comp. Aided Geom. Design **14**, 231–250, 1997.

[10] Floater M. S. How to Approximate Scattered Data by Least Squares, SINTEF Report No. STF42 A98013, Oslo, 1998.

[11] Floater M. S., Westgaard G.. Smooth surface reconstruction from cross-sections using implicit methods, SINTEF Report No. STF42 A96023, Oslo, 1996.

[12] Fremming N. P., Hjelle Ø., and Tarrou C. Surface modelling from scattered geological data. In M. Dæhlen and A. Tveito, editors, *Numerical Methods and Software Tools in Industrial Mathematics*, pages 305–320. Birkhäuser, 1997.

[13] Gerritsen B. H. M., Halbwachs Y. and Amand P. *Macro-Micro topology*, Internal Omega technical document, 1997.

[14] Halbwachs Y., Courrioux G., Renaud X. and Repusseau P. *Topological and Geometric Characterization of Fault Networks Using 3-Dimensional Generalized Maps*, Mathematical Geology, Vol. 28, No. 5, P. 625-656, 1996.

[15] Halbwachs Y. *Modélisation tri-dimensionelle d'un réseau de fractures*, Internal report, BRGM, Orléans, France, 1996.

[16] Halbwachs Y. *3D Model Consistency Topology Library*, Internal Omega technical document, 1997.

[17] Hjelle Ø. Explicit surfaces in Siscat, SINTEF Report No. STF42 A96008, Oslo, 1996.

[18] Hoppe H., DeRose T., DuChamp T., McDonald J. and Stuetzle W. Surface reconstruction from unorganized points, Computer Graphics **26**, 71–78, 1992.

[19] Lienhardt P. *N-dimensional generalized maps and cellular quasi-manifolds*, Intern. Jour. Computational Geometry and Applications, Vol. 4, No. 3, P. 275-324, 1994.

[20] Matra-Datavision. *Technical Overview, CAS.CADE 1.5B*, Matra-Datavision, Les Ulis, France, 1997.

[21] The OMEGA Project Home Page. World Wide Web document at `http://www.omega.beicip.fr`

[22] Schumaker L. L.. Reconstructing 3D objects from cross-sections, in *Computation of Curves and Surfaces, W. Dahmen, M. Gasca, and C. Micchelli eds., Kluwer, Dordrecht* , 275–309, 1990.

[23] The SISCAT Home Page. World Wide Web document at `http://www.oslo.sintef.no/siscat`.

Section : Geometry and Topology Issues
Area of applications : Structural Geology, Reservoir Forecasting
Names of participants : Floater, Halbwachs, Hjelle, Reimers