FieldGML: An Alternative Representation For Fields*

Hugo Ledoux
TUDelft
h.ledoux@tudelft.nl

March 25, 2008

While we can affirm that the representation, storage and exchange of two-dimensional objects (vector data) in GIS is solved (at least if we consider the *de facto* standards *shapefile* and GML), the same cannot be said for fields. Among the GIS community, most people assume that fields are synonymous with raster structures, and thus only representations for these are being used in practice (many formats exist) and have been standardised. In this paper, I present a new GML-based representation for fields in 2D and 3D, one that permits us to represent not only rasters, but also fields in any other forms. This is achieved by storing the original samples of the field, alongside the interpolation method used to reconstruct the field. The solution, called *FieldGML*, is based on current standards, is flexible, extensible and is also more appropriate than raster structures to model the kind of datasets found in GIS-related applications.

1 Introduction

There exist two contrasting conceptualisations of space: the *object* and the *field* views (Couclelis, 1992; Goodchild, 1992; Peuquet, 1984). In a nutshell, the former view considers space as being 'empty' and populated with discrete entities embedded in space, while the latter considers the space as being continuous, and every location in space has a certain property (there is *something* at every location). In the former model, entities can be for example roads, cups of tea, churches, etc., and they have certain properties; in the latter, they are formed by clusters of properties. When one wants to represent and store a certain piece of space in a computer, the field-view approach is much more problematic than its counterpart. The problems are most likely caused by the fact that the definition of a field itself changes from discipline to discipline, and that the issues can be seen from a philosophic, conceptual or implementation point of view (Peuquet et al., 1999). There is also much confusion among users between spatial models, data structures, and spatial concepts (Frank, 1992). While in the GIS jargon object- and field-views of space are often synonymous with respectively vector and raster models, Goodchild (1992), among

*This article will be published in the Proceedings of SDH 2008—The 13th International Symposium on Spatial Data Handling, 23rd to 25th June, 2008, in Montpellier, France.

others, explains that this is simply false as both views can be stored with either model. Put on top of that that fields are by definition something continuous—and that computers are discrete machines—and one can start understanding the confusion among users. (More information about fields and their representations is available in Section 3.)

In recent years, with the multitude of formats available, practitioners have turned to defining and using standards (e.g. those of the Open Geospatial Consortium (OGC), and of the International Organization for Standardization (ISO)) to facilitate the storage and exchange of GIS data. While the current standards are somewhat successful and promising for objects (with the Geography Markup Language (GML) leading the way), their use for fields are very scarce, and are mostly limited to raster solutions. However, as argued in Section 4, rasters are technically and theoretically restrictive and therefore alternative solutions should be sought.

In this paper, instead of using solely raster formats to represent fields, I propose an alternative representation called *FieldGML*. As described in Section 5, this generic solution is based on current standards (i.e. GML), and permits us to efficiently store and exchange field-based geographic information. The main idea behind this representation is that instead of storing explicitly grids or tessellations, we store the data that were collected to study the field (the samples), and we also store the interpolation method that will permit us to reconstruct the field in a computer. I also argue in the following that FieldGML offers a better representation than current ones because: (i) it takes into account the nature of datasets as found in GIS-related applications; (ii) it is valid for fields in 2D and 3D, but can be readily extended to higher-dimensions; and (iii) it is flexible in the sense that different types of fields can be stored (scattered points, tessellations, tetrahedralizations, voxels, etc.). I also present in Section 5 a prototype that was developed to create FieldGML files from already existing fields, and also to transform FieldGML files into different formats and representations that are being used by commercial GISs.

2 Related Work

From a "standards" point of view, different XML-based languages (eXtensible Markup Language) have been proposed. First of all, there is the more general-purpose GML that implements many of the ISO/OGC standards for fields, but not all of them. Note that the definitions of these standards can be found in Section 4, and their implementation with GML is discussed in Section 5. Based on GML/XML, there are different languages to model fields. For instance, Nativi et al. (2005) propose the *NcML-GML*, which permits us to store with GML the metadata associated with netCDF files (this is a multi-dimensional raster format described in the next section). Also, Woolf and Lowe (2007) propose the *Climate Science Modelling Language* (CSML), which is used to represent all the different kinds of climate data (often fields) and their relevant information. The particularity of CSML is that, for the sake of simplicity and performance, the authors chose to use only parts of the standards: they offer a GML-based 'wrapper' around legacy formats to simplify exchange, but they are still using the legacy file for applications (these legacy files are all raster-based). Furthermore, the *Geoscience Markup Language* (GeoSciML) can be used to store any kind of information related to geology (Sen and Duffy, 2005). When fields are involved, they are usually stored in raster formats, but GeoSciML also allows the storage of the observations that were collected (interpolation methods are however not discussed).

From a GIScience point of view, different alternatives to the ubiquitous rasters have been proposed over the years, starting with tessellations into triangles (Mark, 1975; Peucker, 1978).

Kemp (1993) proposes different alternatives to store 2D fields, and shows how to convert them from one representation to another when needed (for analysis). Gold and Edwards (1992), and Ledoux and Gold (2006), among others, have also discussed the use of the Voronoi diagram (in 2D and 3D) as an interesting alternative to raster-based approaches. In a proposition that is similar to the one in this paper (at least conceptually), Haklay (2004) proposes, in an attempt to model and manipulate 2D fields, to store only the samples collected, and the parameters of interpolation functions.

FieldGML is also conceptually very similar to the concept of *virtual data set* (VDS) (Stephan et al., 1993; Včkovski and Bucher, 1996; Včkovski, 1998). A VDS is a dataset "enhanced" with a set of methods that are used to access, manipulate or transform the data—it is an object in the object-oriented sense of the term. The term "virtual" means that different representations of a dataset can be generated for different users/applications. In the context of fields, that means that the samples of a field are stored, and also that interpolation methods to generate different representations of that field are available (pixel size, format, data model, etc.). It is implemented as a Java class where an interface is defined.

VDSs were introduced around 15 years ago as a solution to the interoperability of GISs and to improve the quality of datasets used in GIS. The whole concept of interoperability through VDS was based on the idea that "data exchange is not specified by a standardized data structure (e.g. a physical file format) but a set of interfaces" (Včkovski, 1998, p.54). If we fast-forward to 2008, we now have widely-accepted GIS-related standards (see Section 4) and even a *de facto* language (GML). These standards have taken a different approach to interoperability since all datasets are coded with the same language, which clearly contrasts with VDS where one could store the datasets in his own format as long as he/she implemented the interface. FieldGML can thus be seen as implementation of the conceptual ideas of VDS in a 2008 context where GML is synonymous with interoperability in the GIS world.

3 Fields and Their Representations

This section gives a brief overview of what fields are, from the point of view of GISscience.

3.1 Definition of a Field

A field is a concept rather difficult to define because it is not tangible and not part of our intuitive knowledge. It is easy for us to see and describe entities such as houses or chairs, but, although we can imagine fields, they are somewhat an abstract concept. The consequences of that are firstly that formalising a field is difficult, and secondly that many definitions exist in different disciplines (Peuquet et al., 1999). The definition usually used in a GIScience context is borrowed and adapted from physics. Physicists in the 19th century developed the concept of a *force field* to model the magnetic or the gravitational force, where a force (a vector with an orientation and a length) has a value everywhere in space, and changes from location to location. For most GIS applications, the vector assigned to each point of the Euclidean space is replaced by a scalar value, and we obtain *scalar fields* (it is assumed in the following that all fields are of that type).

Because each location in space possesses a value, a field must be represented mathematically. It is a model of the spatial variation of a given attribute a over a spatial domain, and it is modelled by a function, from $\mathbb{R}^d$ to $\mathbb{R}$ in a d-dimensional Euclidean space, mapping the location to the value of a, thus

$$a = f(location).$$

The function can theoretically have any number of independent variables (i.e. the spatial domain can have any dimensions), but in the context of geographical phenomena the function is usually bivariate (x, y) or trivariate (x, y, z). Notice that the domain can also incorporate time as an extra dimension, and thus we have $a = f(location, time)$.

3.2 Representation in Computers

The representation of a field in a computer faces many problems. Firstly, fields are continuous functions, and, by contrast, computers are discrete machines. Secondly, it should be stressed out that we never have access to a 'complete representation' of a geographical phenomenon. Indeed, to obtain information about a given phenomenon, one must sample it, and reconstruct the field from these samples[1]. In the context of GIS-related applications (e.g. modelling of elevation, geosciences, geology, hydrology, bathymetry, etc.), this collection of samples is hindered by the fact that unlike disciplines like medicine or engineering, we seldom have direct access to the whole object (think of collecting data underground, or at sea for instance). And even if we have complete access to the object, it is often too expensive to sample the object everywhere.

In short, to represent a field in a computer (i.e. to be able to model a continuous phenomenon), we need to:

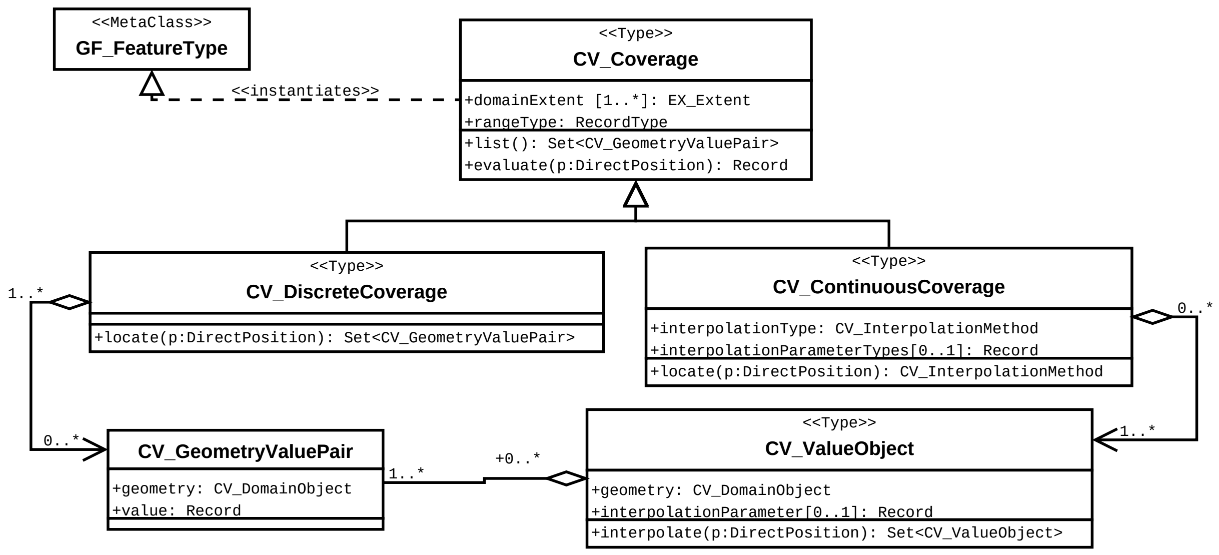
1. have a set of samples for the given fields—they are the "ground truth" of a field. The samples are usually point-based, but other forms can also exist (for instance an image obtained with remote sensing).
2. define a set of rules to obtain the values of the attribute studied, at any location. This operation is referred to as spatial interpolation.

4 Standards and Formats to Represent Fields

4.1 GIS Standards

There are two "levels" of geographic information standards: abstract and implementation specifications. The former defines a conceptual architecture (or reference model) for different aspects related to the storage and exchange of information; and the latter are at a lower-level, i.e. they define an interface to access the properties and methods of classes defined in the abstract specifications.

[1]Even if a sensor is used to collect samples, the result (e.g. an image with pixels) is not a complete representation since each pixel usually averages the value of the studied phenomenon over the pixel area, or each pixel represents the value located in the middle of the pixel.



<<MetaClass>>
GF_FeatureType
<<instantiates>>
<<Type>>
CV_Coverage
+domainExtent [1..*]: EX_Extent
+rangeType: RecordType
+list(): Set<CV_GeometryValuePair>
+evaluate(p:DirectPosition): Record
<<Type>>
CV_DiscreteCoverage
+locate(p:DirectPosition): Set<CV_GeometryValuePair>
<<Type>>
CV_ContinuousCoverage
+interpolationType: CV_InterpolationMethod
+interpolationParameterTypes[0..1]: Record
+locate(p:DirectPosition): CV_InterpolationMethod
1..*
0..*
CV_GeometryValuePair
+geometry: CV_DomainObject
+value: Record
1..*
+0..*
<<Type>>
CV_ValueObject
+geometry: CV_DomainObject
+interpolationParameter[0..1]: Record
+interpolate(p:DirectPosition): Set<CV_ValueObject>
0..*
1..*

Figure 1: UML diagram for the main classes of an ISO/OGC coverage. (Figure after ISO (2005))

Abstract specifications

In the case of fields, two documents exist: the 'Schema for coverage geometry and functions' (ISO, 2005), and the OGC document with the same title (OGC, 2007b). Notice here that fields are referred to as 'coverages' in these documents; both terms are synonymous and used interchangeably in the following. Both documents have the same content. A coverage is considered a *feature*[2], like is any geographic object in the ISO/OGC documents. So while each geographic object in a representation of a field is a feature, the field as a whole is a feature too.

The formal definition of "coverage" is the following (and its principal classes are shown in Figure 1):

> A coverage is a feature that acts as a function to return values from its range for any direct position within its spatial, temporal or spatiotemporal domain. [...] [it] has multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.

Notice that a ISO/OGC coverage can have many different attribute types, but that this is not relevant here, and we simply assume that one coverage is for one attribute type (let that be the temperature of the air, the elevation of a terrain, the density of the population, etc.)

The coverage type is divided into two distinct but closely related subtypes:

Continuous Coverage: coverage that returns different values for the same feature attribute at different direct positions within a single spatial object, temporal object or spatiotemporal object in its domain.

Discrete Coverage: coverage that returns the same feature attribute values for every direct position within any single spatial object, temporal or spatiotemporal object in its domain.

[2] A feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth (ISO, 2003).

The definition of a continuous coverage is more or less equivalent to that of the general coverage type. The definition refers to the fact that interpolation is used to obtain the attribute value at a given location x. The terms "within a single object" can be misleading, but means that interpolation is always performed with a function defined over one geometric object (e.g. a polygon in 2D); if no object is present at a location x (possible according to the definition of a coverage) then no value at x is returned. The latter type, the discrete coverage, seems to exist only because "a coverage can be derived from a collection of discrete features with common attributes" (ISO, 2005). As explained in Section 3.2, this is true (the samples), provided that we have a set of rules to reconstruct the coverage at every location, but this is not the case in the ISO/OGC documents. It is also stated that "a discrete coverage has a domain that consists of a finite collection of geometric objects and the direct positions contained in those geometric objects". The problem here is that these geometric objects do not have to fully partition the domain, i.e. according to that definition a set of unconnected lines and/or polygons (in which each object has a value attached to it) is considered a coverage. Even worse, the objects are permitted to overlap, which means not only do we have locations without any answer, but that there can be more than one answer at one given location! This might be useful for some applications—I am however not aware of any—but none are mentioned in the documents.

Another shortcoming is the list of interpolation methods discussed in the ISO/OGC documents is very restricted. Many interpolation methods are simply ignored, and if one wanted to use them it would be very difficult to integrate them in the coverage framework. Inverse-distance to a power (IDW) and Kriging are for instance not listed, and for many subtypes (such as *CV_TIN-Coverage*, to store triangulated irregular networks (TINs)) only one type of interpolation is possible within each piece (which is restrictive in practice).

Also, the ISO/OGC documents state that the concepts are valid not only for the 2D case, but also for three and higher dimensions. The problem is that it is only a statement weakly backed up by a few types in 3D and no explanations of interpolation methods in 3D are given.

In brief, the abstract standards for coverages do avoid the distinction between raster and vector, but by creating two types for which the differences are rather blurred and subtle, they probably also contribute to the confusion that already exists about fields.

Implementation Specifications

To my knowledge, the only implementation of the ISO/OGC abstract specifications is that of GML. It is an XML-based modelling language developed to facilitate the exchange of geographic data, and has been fairly successful in recent years. While a GML file is verbose (and thus files can become enormous), there are many advantages to using it. Lake (2000) mentions, among others: (i) it is self-descriptive, (ii) it can be processed with already existing XML software, (iii) there are mechanisms to store metadata, and (iv) data integrity can be verified with the help of *schemas*. The reader is referred to Lu et al. (2007) and OGC (2007a) to learn more.

As of GML version 3.2, only the *CV_DiscreteCoverage* types have been implemented: there are GML schemas for all subtypes of *CV_DiscreteCoverage*, and also for grids (*CV_Grid*). That results in a representation that does not necessarily cover the whole spatial domain, and no mechanisms are present to estimate the value of an attribute where there are no spatial objects, or a default and simplistic method is assumed. Using simplistic interpolation methods, or the wrong parameters for a method, is dangerous as many researchers have highlighted (see Watson (1992) for instance).

Not all abstract classes were implemented in GML, *CV_GeometryValuePair* is for instance not present, and was replaced by an implementation that follows closely the conceptual distinction between the spatial domain and the range (the attribute modelled). The resulting XML file has to have three separate types: the domain, the range and another one for mapping these two correctly. As Cox (2007) explains, although this is conceptually valid, it also hinders the use of these standards in practice because of the difficulties of processing large files, of updating files, etc.

4.2 Formats Used in Practice

Among GIS practitioners, fields are being used almost exclusively in 2D, while in the geoscience community 3D and higher-dimensional fields are extensively used. Note that the dimensions in oceanographic/atmospheric coverages are not necessarily spatial dimensions, as any parameters (e.g. temperature of the air, or density of water) can be considered a dimension.

As mentioned before, within the GIS community, coverages are more or less synonymous with grids, although it must be said that TINs are also widely used for modelling terrain elevation. There exist many different formats for 2D grids, but they can be easily all converted to one another.

In geoscience, netCDF[3] seems to be the *de facto* standard to exchange datasets, although other similar formats, such as HDF5[4], are also popular. These formats are raster-based, and permit users to use n-dimensional grids, with different spacing for different dimensions. They are binary and spatially structured, which means that parts of a dataset can be efficiently retrieved and processed. The use of other representations (e.g. tetrahedralizations or arbitrary polyhedra) is very rare and mostly limited to the academic community.

4.3 The Dangers of Using Raster Formats

As argued by many over the years, using raster structures has many drawbacks (Gold and Edwards, 1992; Kemp, 1993; Haklay, 2004; Ledoux and Gold, 2006). Firstly, as Fisher (1997) points out, the use of pixels as the main element for storing and analysing geographical data is not optimal. The problems most often cited are: (i) the meaning of a grid is unclear (are the values at the centre of each pixel, or at the intersections of grid lines?), (ii) the size of a grid (for fine resolutions, grids can become huge), (iii) the fact that the space is arbitrarily tessellated without taking into consideration the objects embedded in that space. Secondly, the use of grids in GIS/geoscience applications has wider repercussions since we can assume in most cases that a given grid was constructed from a set of point samples. Converting samples to grids is dangerous because the original samples, which could be meaningful points such as the summits, valleys or ridges or a terrain, are not present in the resulting grid. The importance of the original samples for a field is such that they have even been dubbed the *meta-field* by Kemp and Včkovski (1998). It should also be said that when a user only has access to a grid, he often does not know how it was constructed and what interpolation method was used, unless metadata are available. Notice that all the previous statements are also valid in 3D (a pixel becomes a voxel).

[3] `http://www.unidata.ucar.edu/software/netcdf/`
[4] http://www.hdfgroup.com

5 FieldGML: The Field Geography Markup Language

Because of the current standards' shortcomings and weaknesses, as highlighted in the previous section, I propose an alternative to represent fields: FieldGML. It is an XML-based language based on GML, and it permits us to represent fields in 2D and 3D, although conceptually it can be easily extended to higher dimensions. Unlike current standards where there is a distinction between discrete and continuous fields/coverages, I argue in this paper that a field should always have one—and only one!—value for a given attribute at every location in the spatial domain (be this domain the surface of the Earth, a 3D volume, or even a 4D spatio-temporal hypercube). The concept of discrete coverage can be then removed, as it is misleading and it creates confusion among users.

5.1 A Field = Samples + Interpolation Rules

The principal idea behind FieldGML is that two things are needed to have a coverage: (i) a set of samples of the phenomenon, and (ii) an interpolation function to reconstruct the continuity of the phenomenon studied.

Samples. By that it is meant what is referred to as 'discrete coverage' in ISO/OGC terms. It is any data that were collected to study the phenomenon:

1. a set of scattered points in 2D or 3D.
2. a set of lines, e.g. contour lines coming from a topographic map.
3. a set of scattered polygons to which one value is attached. Although this case is possible, I am not aware of any interpolation method that would take a set of polygons as input. It is nevertheless always possible to discretise each polygon into a set of points. Polyhedra in 3D are also considered samples.
4. a raster image coming from remote sensing or photogrammetry where the value of each pixel represents the temperature of the sea for instance.

Observe that a set of samples is simply a "normal" vector file or a grid (as defined in other ISO/OGC standards, e.g. in ISO (2003)), where each object is assigned a value for a common attribute.

Interpolation Method. The set of rules used to reconstruct the field from samples can take many forms. Interpolation methods are rather difficult to categorise because they are based on different paradigms, and some methods fall into more than one category. No attempts will be made here to introduce categories (see Mitas and Mitasova (1999) and Watson (1992) for that), but what should be kept in mind is that although several different interpolation methods are used in GIS and that several publications advocate the use of "better" methods, the current standards, while discussing a few methods, give no importance to interpolation and do not permit the use of many of the known methods.

Storing explicitly the interpolation method, as FieldGML is doing, is efficient in practice as only a few parameters have to be stored. Finding the appropriate values for interpolation parameters is a difficult and time-consuming task, as the user must have a good understanding of the spatial distribution of the objects in the set of samples, and of the details of the method. A vivid example is Kriging (Oliver and Webster, 1990), with which experienced users can obtain very good results, but which also leaves newcomers clueless with its many parameters and options. Using Kriging with the appropriate parameters leads to a result that has statistically minimum variance, however, simply using the default values for the parameters will most likely lead to unreliable results. Thus, if we leave the job of modelling the datasets and deriving the interpolation parameters to specialists, the users would not have to worry about these anymore. This is one of FieldGML's main benefits.

5.2 Abstract and Implementation Specifications

Figure 2 shows the class diagram of FieldGML (for the main classes). To ensure that a FieldGML file respects the rules in the model (and that it is therefore 'valid'), a GML application schema has also been developed. What follows is an overview of the engineering decisions that were taken in order to develop FieldGML and its schema. I tried to use GML types as much as possible, but for practical reasons (e.g. simplicity of implementation and performances for processing files) new types also had to be defined. The full application schema is not described here (for lack of space), but it can be obtained on the website of FieldGML[5].

The first thing to notice is that a FieldGML type *Field* inherits directly from a GML feature, which means that it can use all the mechanisms already defined by the OGC to deal with metadata.

An important decision that was taken was not to use directly the GML implementation of *CV_DiscreteCoverage* for the set of samples, for the reasons described in Section 4.1. Instead, four new types were created: (i) scattered points (notice here that even if points are regularly spaced, this type can still be used); (ii) scattered lines; (iii) full tessellations; and (iv) arrays, which includes all the raster-based types. All these types inherit from `gml:AbstractGeometryType` (which means that mechanisms defined by GML for reference systems can be used), and GML types were used where possible (e.g. `gml:MultiPointType` for the scattered points). In addition, these types were extended so that an attribute (a scalar value) is attached to each object; a version of the *CV_GeometryPairValue* was implemented, as in Cox (2007). Also, it should be noticed here that if a tessellation is needed for the interpolation (e.g. Delaunay triangulation for a piecewise interpolation) this structure need not be persistent: only the samples can be stored, and it is calculated on the fly. Constraints that a triangulation must follow can also be stored, but if it is impossible to define rules to automatically construct a tessellation (there has been human intervention in the construction) then the full tessellation must be stored. The type *FullTess* represents this full description (each cell is described); the GML types `gml:MultiSurface` and `gml:MultiSolid` can be used for that, albeit they are rather non-efficient in practice.

As is the case for GeoSciML (Woolf and Lowe, 2007), it was decided that 'legacy files' (i.e. raster formats used in commercial GISs) could be used directly without having to convert them to GML types (which are non-efficient and cumbersome to use in practice). It is however still possible to use *CV_Grid* as defined in ISO/OGC standards and implemented in GML (as a discrete coverage). Legacy files are simply referenced to by a pointer; the metadata about the file (georeferencing, pixel size, etc.) have however to be stored in the FieldGML file with GML types and/or attributes.

Interpolation methods play an important role in the FieldGML model, and several methods used in the GIS world have been listed. The list of methods in Figure 2 is by no means exhaustive as

[5] `www.gdmc.nl/ledoux/fieldgml.html`

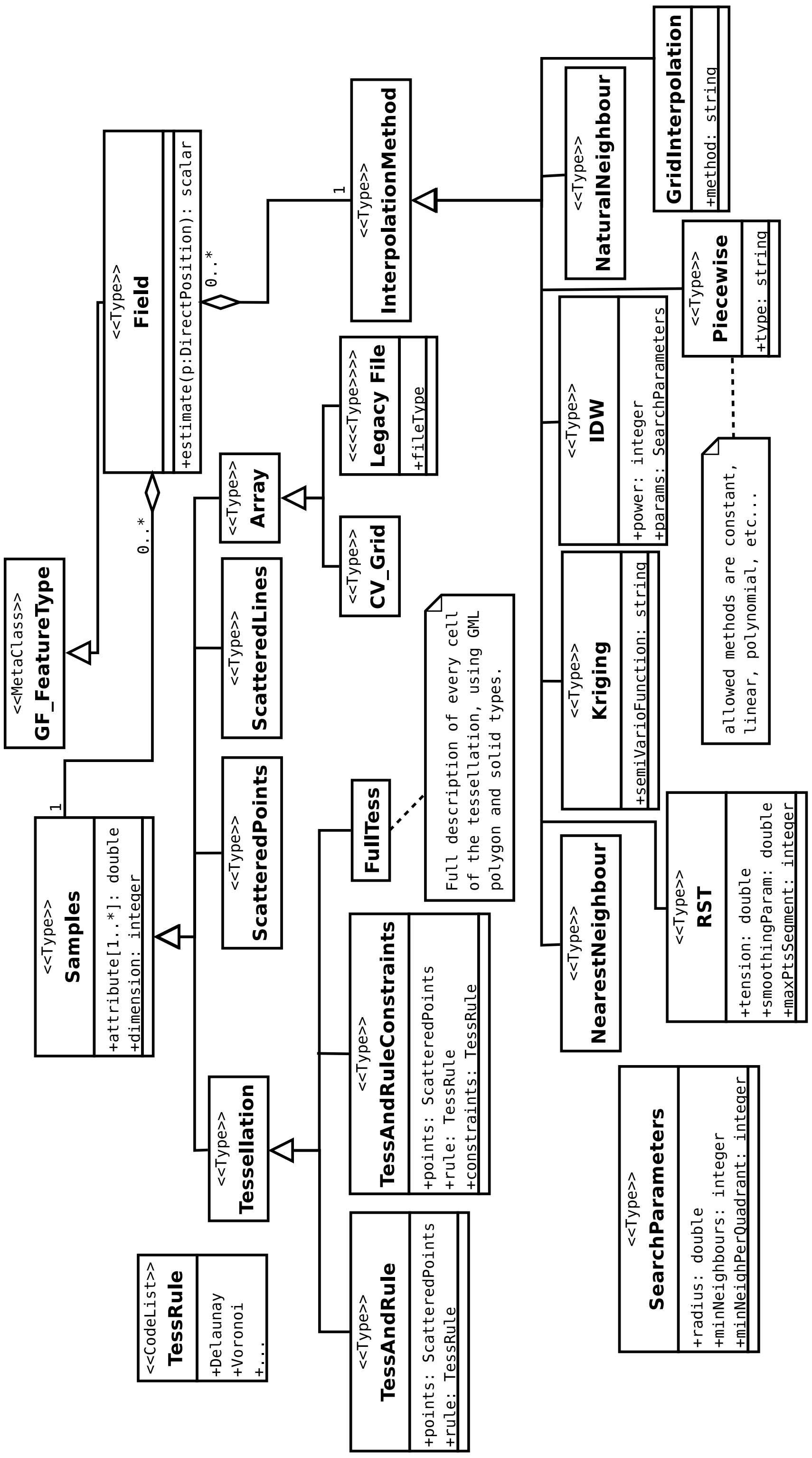


Figure 2: UML diagram for the main classes of FieldGML.

other ones can be easily be added if needed (which is a big advantage over current standards). It should also be noticed that all the methods listed are perfectly valid in 2D and 3D.

While space constraints does not permit to discuss the details of these methods, the manual of FieldGML will describe them and discuss what the parameters imply, and that for interpolation in 2D and 3D.

A few examples of the methods available in FieldGML:

Piecewise: a function is defined over each cell of a tessellation (usually constant or linear).

IDW: as described in Shepard (1968), it requires different parameters to define which points are involved in the interpolation at a given location (different criteria can be used), and also the power must be defined.

Kriging: while the modelling of a dataset is a difficult and time-consuming task, the output of the modelling (a function characterising the dependence between the attributes of any two samples that are at a given distance from each other) can be simply stored as a string. The parameters and the functions as defined in the program *gstat* (Pebesma and Wesseling, 1998) are used.

Natural neighbour: the basic method (Sibson, 1981) does not need any user-defined parameters, but it is possible to obtain a smoother interpolation if certain parameters are used (see Watson (1992)).

RST—regularized spline with tension: this method is available in the open-source GIS GRASS, and by storing a few parameters a field can be reconstructed from a set of samples (Mitasova and Mitas, 1993).

Grid interpolation: while a grid can be seen as a special case of scattered points, different methods optimised for grids have been developed. FieldGML implements a few of them, for instance bilinear and biquadratic. See Kidner et al. (1999) for a discussion in 2D, but these methods trivially generalise to higher dimensions.

In brief, when fields are represented with FieldGML, any kinds of fields can be defined. Observe also that all the ISO/OGC (sub)types can be mapped to a samples/interpolation in FieldGML (so there are no needs to define explicitly subtypes). For example, the *CV_TINCoverage* is based on a set of points, and the interpolation is a piecewise function (linear function inside each Delaunay triangle). Also, notice that even if a grid is the set of samples for a field, an interpolation method must be also defined (it can be for instance constant or bilinear inside each cell).

5.3 Prototype

To convert back and forth between FieldGML representations and the formats used in GIS and geoscience applications, a prototype was built. Currently it permits users to read a FieldGML file and output to different formats, and it is also possible to create a FieldGML file when a set of samples is already available. To output to a format used by commercial GISs, the user has to choose the spatial extent, the resolution of the grids (only grids are possible right now, although triangulation could be implemented in the future), and the format. The possible grid formats currently supported are the ones in the GDAL library[6] (in 2D), and netCDF (in 3D).

[6]The Geospatial Data Abstraction Library: `www.gdal.org`

The prototype was developed with the Python programming language, and uses only open-source software. The interpolation methods described in the precedent section were implemented or their libraries were linked to the prototype. For instance, the program *gstat* (Pebesma and Wesseling, 1998) was used for Kriging, GRASS for RST, and CGAL[7] to create triangulations in 2D and 3D.

5.4 Discussion Over the Implementation

At this moment, the interpolation methods have been implemented in the prototype, but to favour interoperability, I plan to make use of the newly adopted OGC standards about web processing service (WPS) (OGC, 2007c), which defines how GIS operations can be performed over the Internet. The methods used by FieldGML would simply be available on a server, and a user would upload his FieldGML file, specify what representation is needed, and then he/she would get the file.

Also, it is interesting to observe that while FieldGML and VDS have very similar conceptual ideas, the implementations are totally different because of the way interoperability is tackled. The VDS approach was about having proprietary formats not directly accessible to users, who had to access data through common interfaces. While theoretically very sound, this was not the choice the GIS community picked, and now instead we have one language (GML) that can be used to represent any geographical dataset. While probably less efficient (GML is very verbose and complex), it offers more flexibility as anyone can read a FieldGML file and extract the original samples, while in the case of VDS you would have to have a piece of software implementing the interface. With the original dataset, the user can then choose another interpolation method, if needed.

6 Conclusions

The ultimate goal of a digital field representation is to reconstruct in a computer the continuity of a studied phenomenon, i.e. to be able to accurately estimate, or calculate, the value of the phenomenon at any location in a spatial domain. As an alternative to current standards for fields (i.e. the discrete coverage as implemented in GML), what has been proposed in this paper, a GML-based representation, is admittedly rather simple from a theoretical point of view, but yet it permits us to model every situation (and that in two and three dimensions), and it uses the types already defined in current standards (thus it is a step in the direction of interoperability). It is also more adapted than raster structures to the kind of datasets found in GIS-related applications, because it permits us to always keep the original data that were collected to study a phenomenon, and simply generate new representations that are adapted to a particular application. FieldGML was also designed with flexibility in mind, so that other interpolation methods and sample forms can be added.

While the use of FieldGML requires a rethinking from people who produce fields, the users need not be affected. Indeed, a potential user of FieldGML would simply obtain a field in the form of a FieldGML file, select the format and resolution of the output file, and carry on with his/her work as before. But when he/she would need to exchange the field with someone else, the shortcomings of raster structures would not arise.

[7]The computational geometry algorithms library: `www.cgal.org`

Future works include the implementation of a WPS and the processing of very large datasets (which are common in GIS-related applications, especially in three dimensions). I also plan to extent FieldGML so that dynamic fields, and fields having a nominal scale of measurement, are handled.

Acknowledgments

This research was financially supported by the project "Internationale geostandaarden" (RGI-116) in the Netherlands, and by the EU project HUMBOLDT. I would also like to thank my colleagues Marian de Vries and Wilko Quak, and the excellent comments from the three anonymous reviewers, especially Mr or Mrs "Reviewer 3" who pointed out Andrej Včkovski's work on VDS.

References

Couclelis H (1992) People manipulate objects (but cultivate fields): Beyond the raster-vector debate in GIS. In AU Frank, I Campari, and U Formentini, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *Lecture Notes in Computer Science*, pages 65–77. Springer-Verlag.

Cox S (2007) GML encoding of discrete coverages (interleaved pattern). Open Geospatial Consortium inc. Document 06-188r1, version 0.2.0.

Fisher PF (1997) The pixel: A snare and a delusion. International Journal of Remote Sensing, 18(3):679–685.

Frank AU (1992) Spatial concepts, geometric data models, and geometric data structures. Computers & Geosciences, 18(4):409–417.

Gold CM and Edwards G (1992) The Voronoi spatial model: Two- and three-dimensional applications in image analysis. ITC Journal, 1:11–19.

Goodchild MF (1992) Geographical data modeling. Computers & Geosciences, 18(4):401–408.

Haklay M (2004) Map Calculus in GIS: A proposal and demonstration. International Journal of Geographical Information Science, 18(2):107–125.

ISO (2003) ISO 19107: Geographic information—Spatial schema. International Organization for Standardization.

ISO (2005) ISO 19123: Geographic information—Schema for coverage geometry and functions. International Organization for Standardization.

Kemp KK (1993) Environmental modeling with GIS: A strategy for dealing with spatial continuity. Technical Report 93-3, National Center for Geographic Information and Analysis, University of California, Santa Barbara, USA.

Kemp KK and Včkovski A (1998) Towards an ontology of fields. In *Proceedings 3rd International Conference on GeoComputation*. Bristol, UK.

Kidner D, Dorey M, and Smith D (1999) What's the point? Interpolation and extrapolation with a regular grid DEM. In *Proceedings 4th International Conference on GeoComputation*. Mary Washington College Fredericksburg, Virginia, USA.

Lake R (2000) Introduction to GML: Geography Markup Language. In *Proceedings W3C Workshop on Position Dependent Information Services*. Sophia Antipolis, France. Available at `http://www.w3.org/Mobile/posdep/GMLIntroduction.html`.

Ledoux H and Gold CM (2006) A Voronoi-based map algebra. In A Reidl, W Kainz, and G Elmes, editors, *Progress in Spatial Data Handling—12th International Symposium on Spatial Data Handling*, pages 117–131. Springer.

Lu CT, Dos Santos RF, Sripada LN, and Kou Y (2007) Advances in GML for Geospatial Applications. GeoInformatica, 11:131–157.

Mark DM (1975) Computer analysis of topography: A comparison of terrain storage methods. Geografiska Annaler, 57A(3–4):179–188.

Mitas L and Mitasova H (1999) Spatial interpolation. In PA Longley, MF Goodchild, DJ Maguire, and DW Rhind, editors, *Geographical Information Systems*, pages 481–492. John Wiley & Sons, second edition.

Mitasova H and Mitas L (1993) Interpolation by regularized spline with tension: I. Theory and implementation. Mathematical Geology, 25:641–655.

Nativi S, Caron J, Davies E, and Domenico B (2005) Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-GML). Computers & Geosciences, 31(9):1104–1118.

OGC (2007a) Geography Markup Language (GML) Encoding Standard. Open Geospatial Consortium inc. Document 07-036, version 3.2.1.

OGC (2007b) Topic 6: Schema for coverage geometry and functions. Open Geospatial Consortium inc. Document 07-011, version 7.0.

OGC (2007c) Web Processing Service. Open Geospatial Consortium inc. Document 05-007r7, version 1.0.0.

Oliver MA and Webster R (1990) Kriging: A method of interpolation for geographical information systems. International Journal of Geographical Information Systems, 4:313–332.

Pebesma EJ and Wesseling CG (1998) Gstat: a program for geostatistical modelling, prediction and simulation. Computers & Geosciences, 24(1):17–31.

Peucker TK (1978) Data structures for digital terrain models: Discussion and comparison. In *Harvard Papers on Geographic Information Systems*. Harvard University Press.

Peuquet DJ (1984) A conceptual framework and comparison of spatial data models. Cartographica, 21(4):66–113.

Peuquet DJ, Smith B, and Brogaard B (1999) The ontology of fields: Report of a specialist meeting held under the auspices of the VARENIUS project. Technical report, National Center for Geographic Information and Analysis, Santa Barbara, USA.

Sen M and Duffy T (2005) GeoSciML: Development of a generic geoscience markup language. Computers & Geosciences, 31(9):1095–1103.

Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings 23rd ACM National Conference*, pages 517—524.

Sibson R (1981) A brief description of natural neighbour interpolation. In V Barnett, editor, *Interpreting Multivariate Data*, pages 21–36. Wiley, New York, USA.

Stephan EM, Včkovski A, and Bucher F (1993) Virtual Data Set: An Approach for the Integration of Incompatible Data. In *Proceedings AutoCarto 11 Conference*, pages 93–102. Minneapolis, USA.

Včkovski A (1998) *Interoperable and Distributed Processing in GIS*. Taylor & Francis.

Včkovski A and Bucher F (1996) Virtual Data Sets—Smart Data for Environmental Applications. In *Proceedings 3rd International Conference/Workshop on Integrating GIS and Environmental Modeling*. Santa Fe, USA.

Watson DF (1992) *Contouring: A guide to the analysis and display of spatial data*. Pergamon Press, Oxford, UK.

Woolf A and Lowe D (2007) Climate Science Modelling Language Version 2—User's Manual. `http://ndg.badc.rl.ac.uk/csml/`.