EXTENSION TO MULTI-RESOLUTION OF AN EMBEDDED SPATIAL INFORMATION VISUALIZATION SYSTEM

Jean-Michel Follin, Alain Bouju, Frédéric Bertrand and Patrice Boursier
University of La Rochelle
Laboratoire d’Informatique et d’Imagerie Industrielle (L3i)
Avenue Michel Crépeau, 17042 La Rochelle, France
{jean-michel.follin, …}@univ-lr.fr

INTRODUCTION

Advances in mobile technologies (i.e. communications and devices) have led to new applications in mobile computing, in particular web-based mapping applications. Through Internet, a vehicle can download applications, maps and data. Spatial location of the user must be integrated in the data management ([1]).

The amount of data, which has to be transferred and displayed, can be optimized for the user’s purpose according to theme and scale. The work presented in this paper should be considered as continuation of works from [2] which have focused on importance of the reuse of vector thematic data in a mobile environment characterized by limited data transfer rate. However this previous works has not taken into account the transmission of data at different Levels of Details (LoD, cf. [3]). Multi-resolution data may be needed for embedded navigation application “where only parts of the navigation process need detailed information (e.g., the departure and arrival areas), while for the rest of the navigation only coarse level information is needed (e.g., for travelling on a highway section)” ([4]).

During transition between different levels of representation, entities preserved throughout the LoD must be reused on the client side as much as possible in order to reduce data transfer between client and server. Furthermore, consistency must be preserved during decrease (or increase) of detail.

This paper proposes a framework allowing multi-resolution navigation in an embedded spatial information visualization system.

After a review of previous works in relation with the multi-scale aspect in section 1, we propose an extension of the system to multi-resolution in section 2. It includes matching of data, use of generalization and refinement operators, management of the differences between levels of representation and client-server model for vector data transfer.

1. AN EMBEDDED SPATIAL INFORMATION VISUALIZATION SYSTEM

Our general framework is similar to the one presented in [2]. We adopt a client-server model where data are stored and updated on the server side, and transferred as answers to user requests. Our goal is to minimize amount of data exchanged between client and server. We give more details on the aspects related to scale change mechanisms: general architecture, data model and transfer model.
1.1 General architecture
The system organization is divided in two main parts: a distant server and a client.
On the one hand, the distant server manages the access to different data sources and, on the other hand, it manages the data themselves. It chooses to answer the user queries by accessing local data sources or redirecting the request to other servers.

The data manager on the client “manages data visualization, user requests and communication with data servers” ([1]). A concrete prototype is implemented by a Java applet that can be executed by a Java-enabled Web Browser.

1.2 Data model
Data organization in the model is based on traditional definition of the geographical maps: objects are grouped into layers and the sequence of layers forms a map.
An object entity is formed by the quadruple \((o, v, g, t)\) with:
- \(o\): unique identifier,
- \(v\): value \(v = (v_1, v_2, ..., v_n)\), accessed through the set of object’s attributes \(a = (a_1, a_2, ..., a_n)\),
- \(g\): geographical position defined in domain \(G\) and modelled by one among six two-dimensional geographical objects: Point, Line or Region for simple (i.e. connected) objects, and MultiPoint, MultiLine and MultiRegion for complex (i.e. not connected) ones,
- \(t\): timestamp value (last modification time).
Such a model allows the manipulation of only the useful part of an object during data exchange.

A layer is a collection of objects associated with description of objects attributes (it defines properties shared by all objects).
In addition to different themes (e.g. transportation and buildings), a layer could consider various levels of detail.
A map is a succession of layers grouping objects according to their structure and their semantics.
An instance of a map is a concrete set of layers and objects at a given time.
The queries allow data transfer from the server to the client according to certain criteria.
The definition of query, noted by \(q\), follows conventional OQL (Object Query Language) notations ([5]):

SELECT \(o\) FROM \(l_1, ..., l_n\) WHERE \(C\)
where \(o\) is an object of the layer \(l\) (layer to which belongs the set of selected objects) and \(C\) is the selection condition defined over the objects from the layers in \(L\) such that:
\[L = l_1, ..., l_n (l \in L)\].

The result of query \(q\) executed on the instance \(l\) of a map is noted by \(q(l)\).
Because data are not supposed to be created or modified locally, queries are defined with a restriction: the set of selected objects always belongs to only one layer. In this way, different objects cannot be combined to create a new object or layer.
The principles of data management and transfer developed in this formal frame have been designed for data transmission between instances (on the server and client sides) of a same layer.

1.3 Transfer models
In [1], three cases of client-server transfer are distinguished:
1. all the packet \((o, v, g, t)\), noted by \(V\), is sent;
2. the set \(T\) of object identifier and modification time (i.e. the packet \((o, t)\)) is sent;
3. only object identifier \(o\), the set \(O\), is sent.
In order to reduce the volume of exchanged data, three schemas of data transfer between the client and the server have been defined.
In simple communication mode, the server sends directly the complete answer \( V \) to a query \( q \) transmitted by the client.

In two-step communication schema, the server sends the answer in two steps upon a query \( q \) by the client. First, it sends the set \( T \) of objects from the result \( q(I) \), then, after that the client validates them locally by choosing objects \( O \) that are missing or updated, it sends the missing part of the answer \( V \).

In communication with a pre-computed answer, the client sends a query \( q \) with the description \( T \) of objects belonging to the answer \( q(I) \) locally computed (i.e. objects available on the client). The server sends the answer \( V \) of objects to create or update with, if necessary, objects \( O \) to delete.

Extension of our system to multi-resolution involves a partial revision of the above defined data and transfer models in order to make possible:

- local creation of objects (to reuse parts of these objects already locally available),
- use of several layers in the selection of objects (for queried LoD and source LoD),
- use of identifiers to match objects representing the same real-world entities at different LoD (and not only to synchronize client-server transfer of object).

2. PROPOSALS FOR AN EXTENSION TO MULTI-RESOLUTION

We aim at extending our embedded system with a model providing links between map objects representing the same real-world entities at different LoD.

In the data structure, two approaches may be identified ([4]):

1. "one object = one multi-resolution instance" where "each object has a single representation (i.e. one database instance) including multiple geometries, and all object instances are stored in a single multi-resolution database";

2. "one object = many single-resolution instances" where "each object has multiple interconnected representations", one for each resolution where it exists.

By considering the above presented formal model, we place ourselves in the second approach. We suppose that we have several layers of data at various LoD: these layers can indifferently come from only one source (by generalization) or result from different sources. We also suppose that topological consistence of data is preserved through LoD layers.

We do not address the problems of matching pointed out by [6]. We rather deal with problems involved in management of geographical data structured on several LoD layers: in a mobile environment data transfer between client and server has to be optimized.

First, we see different possible matching cases between objects at different resolutions. Then, we mention the generalization and refinement operators required in the navigation across different LoD. Finally, we propose data and transfer models for the multi-scale data management in a client-server context.

2.1 Geometric matching of data

Different matching configurations between objects describing same real-world entities can be considered: 1:1, 1:n, and n:m. Figure 1 illustrates three possible matching cases between polyline objects in two LoD layers.

In the 1:1 matching case, identification of the same object in different LoD can be based on a single identifier (fig. 1.a).

In the 1:n and n:m matching cases, situations similar to fragmentation conflict ([6]) are found: we have different partitionings of same objects by layer.

In the 1:n matching case, the less detailed polyline could take one identifier among these of the more detailed polylines:

- either the one sharing the greatest number of point with it (polyline with identifier Id1 in fig. 1.b),
- or the one with greatest size (polyline with identifier Id3 in fig. 1.c).
A \( n:m \) matching can be split up in several \( 1:n \) matching (fig. 1.c).

![Fig. 1 Three LoD 2 : LoD 1 matching cases between representations of polylines](image)

Fig. 1 Three LoD 2 : LoD 1 matching cases between representations of polylines

Links that can join multi-scale representations are formalized by means of two transformation functions: generalization and refinement.

### 2.2 Generalization and refinement operators

Two changes of object representations are distinguished according to user operations on a map:

- **generalization** (decrease of detail) resulting from a **zoom out**, i.e. affecting the shape of objects.
- **refinement** (increase of detail) during a **zoom in**.

When one of the transformation functions is applied, object should be reconstructed in a topologically consistent format. Generalization and refinement operators can be classified according to their transformation on map, their effect on object geometries and the number of changes in their participating entities.

According to previous works [7] [8], we can consider three categories of operators:

- **metric operators** handling changes related to simplifications and decreases in size, i.e. affecting the shape of objects,
- **topological operators** handling changes in dimension and complexity of objects,
- **semantic operators** handling changes related to attributes (which we will not see).

Operators are applied to objects geometries and they formally take their attributes in domain \( G \). We can call intra-type (resp. inter-type) operator an operator keeping (resp. changing) object’s spatial type.

According to the number of changes in participating entities, operators can be seen as 1:1, 1:n or n:m spatial entity mapping procedure ([9]). For example, 1:1 mapping operators take one object as input and send back the same and modified object as output.

All the 1:1 mapping generalization and refinement operators could be written as follow:

\[
\text{generalize}(o_{id}^{\text{lod } n}, \Delta_g) = o_{id}^{\text{lod } n+1} \\
\text{refine}(o_{id}^{\text{lod } n}, \Delta_g) = o_{id}^{\text{lod } n-1}
\]

with \( o_{id}^{\text{lod } i} \) object with identifier \( id \) and level of detail \( lod i \),

and \( \Delta_g \) values corresponding to the geometrical difference between representations of the same objects at two LoD.

The below described operators of generalization and refinement are shown in fig. 2. They are only presented to give an idea of the variety of representations changes in a multi-resolution context and are not expected to be all implemented in our system.
2.2.1 Metric operators

Metric operators are 1:1 mapping intra-type operators. Simplification operators (A) eliminate details of a polyline or region by selecting ("filtering") a subset of its original points which is considered more representative of its essential shape. Point insertion and point removal operators take as parameters an object and a set \( \Delta \) of points to insert or remove.

Operators of enhancement (B) are used to enlarge objects of all geometries. They include enlargement operators that "enlarge object equally in each direction" and caricature (or exaggeration) operators that "enlarge only some parts of objects" ([10]).

Operators of aesthetic refinement (C) are applied on an object of any type to improve its visual impression by altering its geometry. They include smoothing operators that "reduce sharp angularity from objects having smooth shapes", and rectification operators that "rectify the geometry from objects, which are expected to have a rectangular shape" ([10]).

Operator of displacement can be applied to two or more distinct objects (external displacement) or to components (points or segments) of one object Polyline or Region (internal displacement). It resolves spatial conflict (objects too close or overlapping). "In this procedure the operated object is a spatial entity pair rather than an independent entity" ([9]).

2.2.2 Topological operators

Contraction (or collapse) operators (D) reduce dimensionality of objects. They perform change either from Region to Polyline or Point, or from Polyline to Point. Expansion operators perform inverse transformations. They are 1:1 mapping inter-type operators.

Operators of selection/ elimination and selection/ addition (E) can be applied to objects collections (e.g., a network of polylines) to add or remove objects according to theme, feature type, or conflict type (for example objects of less importance in fig. 2). They are intra-type n:m mapping operators.

Operators of aggregation include 1:n mapping operators of amalgamation (fusion and merge) and combining, and n:m mapping operator of typification ([10]).

Amalgamation intra-type operators (F) of fusion or merge correspond to fusion of at least two entities (connected in case of fusion, disjoint in case of merge) of same dimension (polylines or regions) in one entity. Object split operator corresponds to splitting of one entity into two entities of same dimension.

Combining inter-type operators (G) combine a set of objects of a same class (points or polylines) to one object of higher dimensionality (polyline or region).

Typification operators (H) change a set of discrete objects (points, polylines or regions) in a smaller set of objects with similar structural characteristics. These structuring processes are often the combination of several basic operators (selection, aggregation, displacement, simplification). They are n:m mapping intra-type operator (objects typified in fig.2 are bends and buildings).

Figures 3.a and 3.b illustrate two cases of representations of a polyline at two LoD. Each LoD representation of a polyline corresponds to a list of vertices with index and coordinates.

Three categories of vertices are noticed:
- those inserted (from LoD 2 to LoD 1) or removed (from LoD 1 to LoD 2),
- those kept from a LoD to another (cf. simplification operators in 2.2.1),
- those displaced from a LoD to another (cf. displacement in 2.2.1).

A more visual example is given in figures 4. We can see that extremities of polylines connected to a traffic circle (fig. 4.a) are displaced (displacement operator) when the circle is reduced to a point (contraction operator) in a more generalized dataset (fig. 4.b). Furthermore, some streets of less importance are omitted (selection/ elimination operator).

These operators formalize inter-level links between entities and must be encoded like instructions to communicate from server to client.
2.3 Management of the increments between different LoD

Increments are regarded as complex sets of entities corresponding to the difference between two consecutive LoD of a vector dataset.

A sequence of instructions (including resolution change operators) and data (including Δ values corresponding to the geometrical difference, for each object, between consecutive LoD representations) can be pre-computed and stored on the server side and transmitted in order to increase (or decrease) detail to the client upon request.

The notions of LoD layers and LoD data have to be defined before to see how increments can be encoded.

2.3.1 Different LoD layers

In addition to $l_1, \ldots, l_n$ corresponding to thematic layers, we can define representations of thematic layers at different LoD as:

$$l_1^{\text{lod} 1}, l_1^{\text{lod} 2}, l_1^{\text{lod} 3}, \ldots \text{ and } l_n^{\text{lod} 1}, l_n^{\text{lod} 2}, l_n^{\text{lod} 3}, \ldots$$

For a given theme, we have a sequence of LoD layers, each one including an identifier and a "validity" interval in term of resolution. Upon a zoom request by the user, client is able (according to zoom factor) to determine locally if a new LoD layer is required and, if necessary, which one.
2.3.2. Types of LoD data

Several types of data can be distinguished in an embedded navigation application extended to multi-resolution. Figure 5 illustrates real-time navigation across two LoD representations of a same thematic layer. Mobile user accesses to LoD $n$ data at time $t_0$, then zooms out at time $t_1$ by requesting LoD $n+1$ data, and finally makes a new LoD $n+1$ data request at time $t_2$. Datasets $D_{t_0}$ and $D_{t_1}$ (respectively copied from the server at $t_0$ and $t_1$) correspond to local instance available on client side and $D_{t_2}$ is dataset requested at $t_2$.

In $D_{t_2}$, three sub-sets of data can be identified in function of LoD data available:

1. dataset already available that can be reused from the same level of detail (LoD $n$) corresponding to the intersection between data $D_{t_0}$ downloaded at $t_0$ and $D_{t_2}$.
2. dataset that can only be reused from the previous level of detail (LoD $n+1$) corresponding to the intersection between data $D_{t_1}$ and the part of data $D_{t_2}$ which is not covered by $D_{t_0}$.
3. dataset that is unavailable on the client for all LoD layers and needs to be downloaded from the server.

In the second type of dataset, we can define three sub-sets of objects: LoD $n+1$ objects totally contained within LoD $n$ objects (e.g. polyline in fig. 3.a), LoD $n+1$ objects partially contained (e.g. polyline in fig. 3.b) and LoD $n+1$ objects totally absent from LoD $n$ objects (e.g. the circle in fig.4.a).

These new types of LoD data have to be integrated in our client-server transfer model. Objects contained in the second type of dataset must be used as much as possible by the operators with increments to reconstruct data required at LoD $n$. 

---

**Fig. 3** Transitions between two LoD representations of a polyline

**Fig. 4** Representations of a same dataset at two different LoD
2.3.3. Encodings for increments

Changes to perform on objects geometries during a transition from a layer $l_{lod1}$ to a layer $l_{lod2}$ can be encoded like an instructions set associated to a data set, noted by $Diff$.

These differences are linked to two LoD layers and encoded for each matching configuration. They include:

- an identifier for the linked objects that take into account their matching configuration (cf. 2.1),
- a set of operations and data which implements operators seen in section 2.2. For example, the operators:

\[
\text{insertPoints}(o_{lod1}, (i_0 \cdot (x_0, y_0)), \ldots, (i_n \cdot (x_n, y_n))) : \text{insert points of coordinates } (x_i, y_i) \text{ at index } i \text{ of the polyline or region object } o_{lod1},
\]

\[
\text{displacePo lylinePoin ts}((o_{lod1}^{i}, (i_0 \cdot \Delta x_0, \Delta y_0)), \ldots, (i_n \cdot \Delta x_n, \Delta y_n)) : \text{move points of indexes } i \text{ of the polyline or region object } o_{lod1}^{i} \text{ with a shift } (\Delta x_i, \Delta y_i)
\]

The transformation function from a LoD layer to another unnecessary consecutive one (e.g. from $l_{lod1}$ to $l_{lod3}$) may be obtained by composition of atomic operators allowing each step of the transition (i.e. from $l_{lod1}$ to $l_{lod2}$ and from $l_{lod2}$ to $l_{lod3}$).

Such encoded operations sequences should allow reducing amount of data which has to be transferred from server to client by reusing, when possible, data already locally available. Topological consistence of data should also be preserved.

The principle of integration of layers differences adds a vertical dimension to data transfer model exposed by [2]. We propose a new transfer schema adapted to multi-scale.
2.4 A client-server model for transfer of vector data at different LoD

The purpose of the below described transfer model is to maximize vertical transfers of data (between different LoD layers on the client side) while minimizing horizontal transfers (between the server and the client sides).

Only two consecutive LoD of a same thematic layer will be considered in the following description of a client-server transfer schema. \( l \) and \( l' \) are the two layers on the server side (with \( l' \) less detailed than \( l \)) and \( l_c \), \( l'_c \) are layers on the client side.

We propose a general transfer model of multi-scale data based on communication with a pre-computed answer (cf. 1.3). It is divided in three steps (fig. 6). The notion of working zone \( Wz \) is used for reducing the data set transferred and anticipating the movement of vehicle. It corresponds to “a buffer of data around the visible zone” on the screen ([11]). To simplify our presentation, we consider \( Wz \) as a fixed-size rectangular window.

Multi-scale data transfer is performed as answer to a client request of transition between a LoD \( m \) to a LoD \( n \) representation (zoom in or out). The source layer can indifferently be \( l \) or \( l' \) and is noted by \( l_m \) on the server side, and \( l_c m \) on the client side. The destination layer is noted by \( l_n \) on the server side, and \( l_c n \) on the client side.

1. The working zone \( Wz \) is compared on the client side with the available data at source level \( m \) and at queried level \( n \): identifiers of objects already present at level \( n \) and only present at level \( m \) (cf. 2.3.2) are computed (by partial inclusion of objects belonging to instance at layers \( l_c m \) and \( l_c n \) with \( Wz \));
2. A request \( q \) is sent to the server including identifiers of \( l_m \) and \( l_n \), completed by two sets: \( T_n \) for the already available objects at queried level \( n \) and \( T_m \) for objects exclusively available at source level \( m \);
3. An answer is sent by the server for missing objects at the queried level, with:
   • LoD \( n \) objects \( V_n \) to create or update, and \( O_n \) to delete (in answer to \( T_n \)),
   • LoD \( m \) objects \( V_m \) (to create or update) and \( O_m \) (to delete) needed for LoD \( n \) and the instructions and data set \( Diff_{mn} \) allowing reuse of objects only available in \( l_m \) and required for \( l_n \) (in answer to \( T_m \)),
   • and objects \( V \) unavailable from both \( l_c m \) and \( l_c n \).

Two options can be examined in the above presented model:
   • either the entire set of objects of \( l_m \) which intersects \( Wz \) and is absent in \( l_n \) is copied at the step 1 and operators of \( Diff_{mn} \) are then used at the step 3 to modify objects geometries (e.g. simplification operators delete points not required),
   • or these objects are not copied and the instance for missing objects in \( l_n \) is reconstructed by using \( Diff_{mn} \) at the step 3 (e.g. simplification operators select points required).

The instructions to send from the server to the client depend on scale change: figure 6 illustrates a zoom out case.

![Fig. 6 Different steps of client-server transfer for data at two levels of detail](image-url)
CONCLUSION

Our solution aims at managing multi-scale data. It combines:
• matching of data at different LoD,
• encoding of operations allowing navigation across levels of representation,
• use of a transfer schema suitable for a multi-scale structure of data.

This solution fulfills requirements of multi-resolution navigation in an embedded context by reducing the amount of data exchanged between client and server. One of the main purposes of such a data management is to improve autonomy of the client.

First encouraging experimentations have been made on 1:1 matching data. We will now extend that experimentations to more complex cases, i.e. to 1:n and n:m matching data.

Our study has been limited to LoD representations of one thematic layer and we plan to extend it to LoD representations of several layers. At a given scale, the preservation of consistency between LoD of different thematic layers is an issue of our future work.

Moreover, the LoD of data could be automatically adapted to vehicle speed: for example, low detailed information may be need for a high speed movement.

Others models of data transfer can be tried and compared by calculating statistics (size, time) on the answer which has to be transferred on client side upon a zoom request.

ACKNOWLEDGMENTS

This research was partially founded by the Communauté d’Agglomération of the city of La Rochelle, France.

REFERENCES: