Revista Brasileira de Cartografia (2012) N^0 64/6: 797-806
Sociedade Brasileira de Cartografia, Geodésia, Fotogrametria e Sensoriamento Remoto
ISSN: 1808-0936

MOVING OBJECTS AND SPATIAL DATA SOURCES

Objetos Móveis e Fonte de Dados Espaciais

Karine Reis Ferreira[1]; Lúbia Vinhas[1];
Antônio Miguel Vieira Monteiro[1] & Gilberto Câmara[1]

[1]Instituto Nacional de Pesquisas Espaciais – INPE
Divisão de Processamento de Imagens - DPI
Av. do Astronautas, 1758 Jardim da Granja São José dos Campos-SP 12227-010
{karine, lubia, miguel, gilberto}@dpi.inpe.br

Recebido em 20 de julho, 2012/ Aceito em 09 de setembro, 2012
Received on july 20, 2012/ Accepted on september 09, 2012

ABSTRACT

Moving object is a well-established concept in geographic information system (GIS) science. It is an entity whose spatial position or extent changes continuously over time. Some examples are cars, animals and deforested regions. Nowadays, there is a growing demand for GIS tools that are able to handle and analyze moving objects. Most existing spatial file formats (e.g. KML and GML) and database systems (e.g. PostGIS) represent spatial and temporal information using structures and types predefined in specifications written by the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC). However, in these specifications, there is nothing about moving object representation in data files or databases. Each data producer adopts its own format to do it. Therefore, this work proposes an interoperable strategy to translate spatial and temporal information stored in different data sources into moving object trajectories for further analyses. The proposed approach is based on the processing of an additional metadata file that describes *how* moving objects are stored in a particular data source. Grounded on this strategy, we have built a new software module for moving object analysis in a geographical library called TerraLib. This module architecture is also described in this paper.

Keywords: Geographical Information Systems (GIS), Moving Objects, Spatial Data Sources, KML, PostGIS.

RESUMO

Objeto móvel é um conceito bem estabelecido na ciência de Sistemas de Informações Geográficas (SIG). Um objeto móvel é uma entidade cuja localização ou extensão espacial muda de maneira contínua ao longo do tempo. Alguns exemplos são carros, animais e áreas de desmatamento. Atualmente, existe uma crescente demanda por ferramentas de SIG capazes de manipular e analisar objetos móveis. A maioria dos formatos de arquivos de dados espaciais (por exemplo, KML e GML) e dos sistemas de bancos de dados espaciais (por exemplo, PostGIS) representam informações espaciais e temporais utilizando estruturas e tipos pré-definidos em especificações escritas pela *International Organization for Standardization* (ISO) e pelo *Open Geospatial Consortium* (OGC). Porém, nessas especificações, não existe nada sobre a representação de objetos móveis em arquivos de dados ou bancos de dados. Cada produtor desse tipo de dado adota seu próprio formato para representá-lo. Portanto, este trabalho propõe uma estratégia interoperável para traduzir informações espaciais e temporais armazenadas em diferentes fontes de dados em trajetórias de objetos móveis para análises posteriores. Nossa abordagem consiste no processamento de um arquivo adicional de metadados que descreve *como* os objetos móveis são armazenados em cada fonte de dados. Baseado nessa estratégia, foi construído um novo módulo de software para análise de objetos móveis em uma biblioteca geográfica chamada TerraLib. A arquitetura desse módulo é descrita também ao longo desse artigo.

Palavras chaves: Sistemas de Informações Geográficas (GIS), Objetos Móveis, Fontes de dados Espaciais, KML, PostGIS.

1. INTRODUCTION

This work extends (FERREIRA *et al.*, 2012), in two ways. First, the strategy designed only for KML files is generalized to deal with different kinds of spatial data sources, including database systems such as PostGIS. Second, we propose a new software architecture based on this extended strategy. To prove such strategy and architecture, we build a prototype and try out it with animal tracking and car movement data from different sources.

The recent technological advances in geospatial data collection, such as Earth observation and GPS satellites, mobile computing, and sensor networks, have motivated new applications that handle spatiotemporal information. Some examples are location-based systems, natural disaster and environmental change monitoring. To support these applications, there is a growing demand for geographical information systems (GIS) that deal with such information.

Since the beginning of the 2000s, the GIS community has made a serious effort towards spatial data interoperability. The International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC) have proposed standards to represent and store spatial information in data files and database systems. Geography Markup Language (GML) (OGC, 2007) and Keyhole Markup Language (KML) (OGC, 2008) are examples of file formats proposed by OGC for spatial data interchange. Many agencies and institutions throughout the word have distributed their spatial data using these formats. Spatial extensions of traditional DataBase Management Systems (DBMS), such as PostGIS and Oracle Spatial, deal with spatial information in compliance with the OGC Simple Feature Access (SFA) specification (OGC, 2006a) (OGC, 2006b).

The compliance with ISO and OGC standards has assured a high degree of spatial data interoperability. Many GIS tools and libraries are able to access spatial data files and databases that follow these standards. Standards are useful to promote spatial data interoperability. However, few results have been achieved regarding spatiotemporal data interoperability.

Moving object is a well-known category of spatiotemporal data. They are objects whose spatial positions or extents change continuously over time (ERWIG *et al.*, 1999). Examples of moving objects are cars, aircraft, ships, mobile phone users, polar bears, hurricanes, forest fires, and oil spills on the sea. Although the concept of a moving object is well-established in GIS science, there is not a standard way to represent it in data files or database systems. Each data producer adopts its own format to store moving objects. A particular format specifies the way to encode information and how it is organized.

This work focuses on this class of spatiotemporal data. It proposes an interoperable strategy to translate spatial and temporal information stored in different data sources into moving object trajectories for further analyses. The proposed approach is based on the processing of an additional metadata file that describes *how* moving objects are stored in a particular data source. It is an XML file that must be compliant with a schema proposed in this paper. Grounded on this strategy, we have built a new software module to deal with and analyze moving objects in a geographical library called TerraLib (CÂMARA *et al.*, 2008).

This work meets the GIS and Cartographic community needs on new technologies for dealing with a myriad of space-time point-source data, considering the diversity of storage means and of semantic interpretation given a certain domain application.

2. RELATED WORK

Erwig et al. (1999) propose a model, called Moving Object Model, which defines an algebra to deal with moving objects. This algebra specifies three main data types, *moving points*, *moving lines* and *moving regions*, and a set of operations over them, such as *trajectory* and *distance*. This work is based on this algebra.

Fig. 1 (a) and (b) shows the tracking of an animal and the evolution of a deforested region. The former is an example of a moving point because the animal position changes over time. The latter is a moving region, since the object extent evolves over time.

Although moving object spatial positions or extents change continuously over time, they are often represented by discrete observations. For instance, Fig. 1 (a) shows an animal tracking through an observation set. Each observation records a spatial position, represented by a point, and a time instant when the animal was at that position. Fig. 1 (b) presents the evolution of a deforested region through three observations. Each one contains the spatial extent of the deforested region, represented by a polygon, and the year when it was detected.

Trajectories are countable journeys associated to objects that are moving over time (SPACCAPIETRA *et al.*, 2008). Different kinds of trajectories can be extracted from a moving object. For example, if an application is interested in studying the daily behavior of an animal, it can extract its trajectories by grouping its daily observations. In another case, the application might extract trajectories that group the animal observations by its intersection with some regions of interest.

Based on the algebra proposed by Erwig et al. (ERWIG *et al.*, 1999), there are two main initiatives of Moving Object Database (MOD) systems, SECONDO (GUTING and SCHNEIDER, 2005) and Hermes (PELEKIS *et al.*, 2008). Both extend the SQL type system with data types to represent moving objects, such as *moving point* and *moving region*, and a set of functions to deal with them. SECONDO is an extensible database system prototype designed at the FernUniversität in Hagen. Hermes is a MOD engine that has been implemented as an Oracle data cartridge.

ISO defines a conceptual model called Moving Feature Model for moving features, that is, features whose geometries move over time as a rigid body (ISO, 2008). It supports changes of location, translation and rotation, but not deformation of a feature. The Moving Object Model is broader than the Moving Feature Model because it supports geometry deformation over time. By dealing with geometry deformations, the model can cope with a class of environmental problems, like deforested region evolution show in Fig. 1 (b), where entity geometries move and deform over time.

3. THE PROBLEM

Most existing spatial file formats (e.g. KML and GML) and database systems (e.g. PostGIS) do not provide data types or structures to represent moving objects. They represent spatial and temporal information using structures and types predefined in ISO and OGC specifications. However, in these specifications, there is nothing about moving object representation in data files or database systems. Each data producer adopts its own format to do it. Therefore, this work addresses the problem: *how to translate spatial and temporal information stored in different data sources into moving object trajectories for further analyses*?

To illustrate this problem, let us consider two real examples of data sources that contain spatial and temporal information related to moving objects: a KML file and a PostGIS database.

3.1 Moving Objects in KML files

KML stands for Keyhole Markup Language and is an OGC standard for encoding and transporting representations of geographic data, mainly for data display in an Earth browser. It is an XML file that follows a predefined XML schema. Such schema describes the grammar which KML file instances must be compliant with. All components

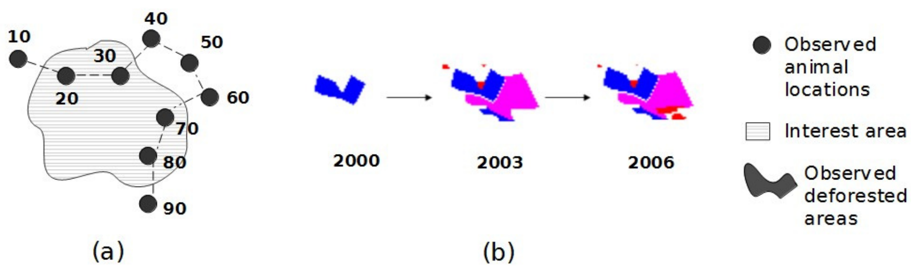

Fig. 1 - Examples of moving objects: (a) an animal tracking and (b) the evolution of a deforested region.

of the KML schema are defined in the namespace with the identifier "http://www.opengis.net/kml/2.2".

The KML Schema defines an element called kml::PlacemarkType to represent spatial objects and time stamps associated to them. Spatial objects are represented by five types: kml:MultiGeometryType, kml:PointType, kml:LineStringType, kml:LinearRingType and kml:PolygonType. It defines two types for time information: kml:TimeStampType and kml:TimeSpanType.

The first example is a KML file generated by a project that monitors sea elephants in the Antarctica (INPE, 2012). This file contains observations of eight animals during three years. Each observation has an animal location at a specific time and is represented by a kml::PlacemarkType element. The animal location is represented by kml::PointType and its associated time by kml::TimeStampType.

Although KML is used to describe journeys, there is not a predefined type in its schema that associates spatial and temporal elements to a same trajectory. There is nothing to indicate what kml::PlacemarkType elements must be grouped as the same moving object trajectory. In this example, the KML file uses a kml::FolderType element to group all observations of the same animal. However, KML files generated by other producer can use different elements to do it.

This file also contains visual style elements to describe how the data should be visualized. Fig. 2 shows the display of this KML file in the Google Earth, where the red lines represent the sea elephant trajectories.

3.2 Moving Objects in a PostGIS database

PostGIS extends the PostgreSQL, an open source object-relational database system, to deal with geographic objects. It is compliant with the OGC Simple Feature Access (SFA) specification (OGC, 2006a) (OGC, 2006b). It provides a set of data types to represent geometries, such as st_point and st_polygon, and of functions to handle these types, such as st_distance and st_intersection. These types and functions come from the OGC geometry model. For temporal information, PostgreSQL supports the full set of Structured Query Language (SQL) date and time types, such as timestamp, interval, date and time.

The SFA specification uses the term *feature tables* to refer to tables that have at least a spatial attribute, stored in a column whose domain is a geometry type. It proposes two metadata tables: geometry_columns and spatial_ref_sys. The spatial_ref_sys table holds the numeric identifications and textual descriptions of coordinate systems used in the spatial database. The geometry_columns table registers the available feature tables in the database and metadata about their geometry columns, such as their types and associated spatial reference systems identifications (srid).

The second example is a PostGIS database that has observations of moving cars in a city. Fig. 3 shows the trajectories of three cars during a day, where each point represents a car location at a specific time. All observations of all cars are stored

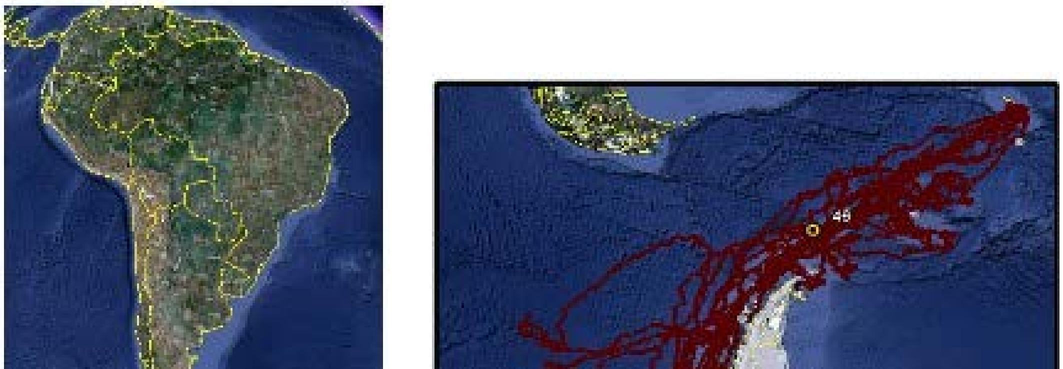

Fig. 2 - Trajectories of sea elephants: display of the KML in the Google Earth software.

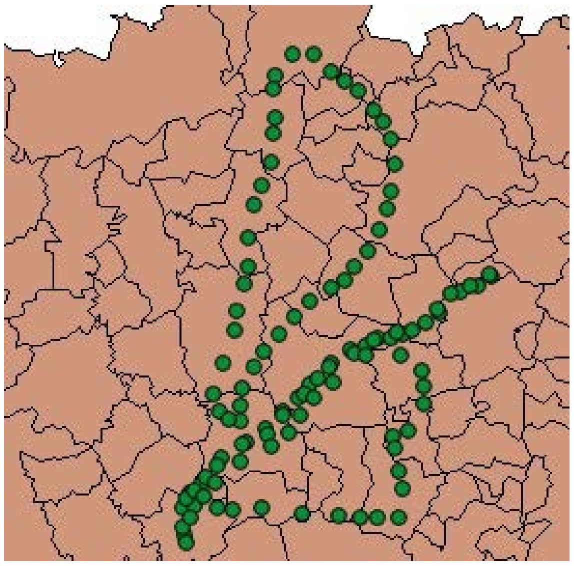

Fig. 3 - Trajectories of three cars in a city during a day.

in a feature table, called car_trajectories, that has three columns: (1) car_id: to store the car identities; (2) location: to store the car spatial locations (st_point type); and (3) date_time: to store the temporal information (timestamp type).

The car_trajectories is a feature table and so the metadata about its geometry column is registered in the geometry_columns table. However, there is not a metadata in this database that indicates how to translate the spatial and temporal information of the car_trajectories table into moving object trajectories.

3.3 Analyzing Moving Objects

Most GIS tools can access and display geometries and their associated times from PostGIS databases and KML files. Some of them, such as Google Earth, can automatically configure timelines and generates animations over time. However, they are not able to analyze them as moving object trajectories. They cannot answer questions like: (1) *Where was object o_1 at time t_5?* (2) *When did object o_1 enter a specific region r_{10} and how long did it stay in this region?* (3) *When and where did objects o_1 and o_2 meet each other (considering a meeting when the distance between two objects is less than 2 meters)?* (4) *Where and when was there a spatiotemporal cluster of objects?*

This requires a more specialized tool that is able to: (1) translate geometry objects associated to time stamps stored in data sources into data structures that represent moving object trajectories, and (2) analyze trajectories, by providing functions over its data structures that can answer questions like the ones presented above. To meet these requirements, we are developing a new software module in a geographical library called TerraLib (CÂMARA *et al.*, 2008). Its architecture is described at follow.

4. SOFTWARE ARCHITECTURE

This section describes the architecture of a new software module for moving object analysis, built in a geographical library called TerraLib. TerraLib is a C++ software library base to build geographical information systems. It is open source and is developed by the National Institute for Space Research (INPE) (CÂMARA *et al.*, 2008).

This new module is composed of three other ones, **ST** (SpatioTemporal), **STLoader** and **DataAccess**, as shown in Fig. 4. The **ST** module contains data structures and functions to represent and analyze moving objects. It provides functions to calculate the distance between two moving objects and the intersection between a moving object and a region of interest. The distance operation results in a time series that maps each time to the distance between the objects at that time. The intersection operation results in patches or trajectories of a moving object that intersect a region of interest, as shown in Fig. 5. In this figure, each

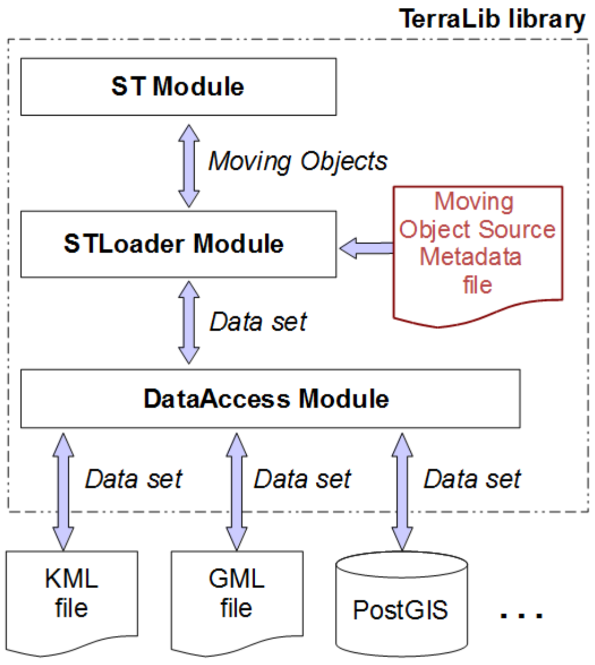

Fig. 4 - Software architecture.

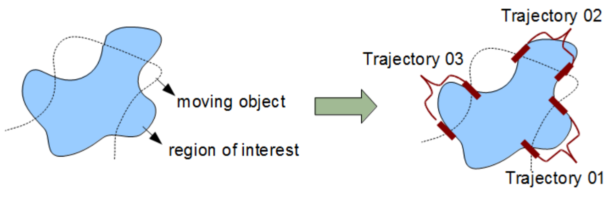

Fig. 5 - Intersection between a moving object and a region of interest.

trajectory represents a patch when the object was inside the region of interest. Using the ST module functions, a user can answer questions like the four ones presented in Section 3.3.

The **DataAccess** module is in charge of accessing data sets from different sources, such as KML and GML files as well as PostGIS databases. Each source stores spatial and temporal information using particular predefined structures. A point is stored in a kml::PointType element in KML files and in a st_point type in PostGIS databases. So, this module has to know the particularities of each source to load its data sets.

The **STLoader** module is responsible for translating the data sets loaded by the DataAccess into moving object structures of the ST module. To do this, it needs extra information about *how* the sources represent moving objects. Let us consider the PostGIS database presented in Section 3.2. To load its moving cars, this module has to know that the car_trajectories table contains moving objects. Besides that, it needs to know that its column car_id stores the car identities, location stores the car locations and date_time contains the temporal information. To load moving animals from the KML file described in Section 3.1, this module has to know that all observations of each animal are grouped in a kml::FolderType element.

Therefore, this module requires an additional metadata file, called *moving object source metadata*, which contains this necessary extra information.

5. MOVING OBJECT SOURCE METADATA

The *moving object source metadata* is an XML file. XML stands for eXtensible Markup Language and is a markup language designed to transport and store structured data. It is a World Wide Web Consortium (W3C) recommendation and has been widely used to carry and share data mainly in the Web environment (BRAY *et al.*, 2008). An XML file is structured through user-defined tags and can be described by a XML Schema. The purpose of an XML Schema is to define the legal building blocks of an XML document in terms of elements and attributes that can appear in an XML file. The XML Schema language is called XML Schema Definition (XSD).

Moving object source metadata files must be compliant with the XML Schema proposed in this section. The schema is:

```
<xs:complexType  name=”MovingObjectSourceType”>
   <xs:sequence>
      <xs:element  name=”DataSourceInfo”
                           type=”DataSourceInfoType”
                       use=”required”/>
      <xs:element  name=”MovingObjectInfo”
                       type=”MovingObjectInfoType”
                       use=”required”
                       maxOccurs=”unbounded”/>
   </xs:sequence>
</xs:complexType>
<xs:complexType  name=”DataSourceInfoType”>
   <xs:sequence>
      <xs:element  name=”name”
                   type=”xs:string”
                   use=”required”/>
      <xs:element  name=”type”  use=”required”>
         <xs:simpleType>
         <xs:restriction  base=”xs:string”>
            <xs:enumeration  value=”KML”/>
            <xs:enumeration  value=”POSTGIS”/>
         </xs:restriction>
         </xs:simpleType>
      </xs:element>
      <xs:element  name=”Params”
                   type=”DataSourceParamsType”
                   use=”required”/>
      </xs:sequence>
</xs:complexType>
<xs:complexType  name=”DataSourceParamsType”>
   <xs:sequence>
      <xs:element  name=”keyValuePair”
                   type=”xs:string”
                   use=”required”
                   maxOccurs=”unbounded”/>
   </xs:sequence>
</xs:complexType>
<xs:complexType  name=”MovingObjectInfoType”>
   <xs:sequence>
      <xs:element  name=”containerType”
                   type=”xs:string”
                   use=”required”/>
      <xs:element  name=”containerName”
                   type=”xs:string”
                  use=”required”/>
      <xs:element  name=”IdInfo”
                   type=”IdInfoType”
                  use="optional"/>
      <xs:element name="SpatialInfo"
                  type="SpatialInfoType"
                  use="required"/>
      <xs:element name="TemporalInfo"
                  type="TemporalInfoType"
                  use="required"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="IdInfoType">
   <xs:sequence>
      <xs:element name="name"
                  type="xs:string"
                  use="required"/>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="SpatialInfoType">
   <xs:sequence>
      <xs:element name="name"
                  type="xs:string"
                  use="required"/>
      <xs:element name="srid"
                  type="xs:integer"
                  use="optional"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="TemporalInfoType">
   <xs:sequence>
      <xs:element name="name"
                  type="xs:string"
                  use="required"/>
      <xs:element name="pattern"
                  type="xs:string"
                  use="optional"/>
     <xs:element name="resolution" use="optional">
        <xs:simpleType>
        <xs:restriction base="xs:string">
           <xs:enumeration value="SECOND"/>
           <xs:enumeration value="MINUTE"/>
           <xs:enumeration value="HOUR"/>
           <xs:enumeration value="DAY"/>
           <xs:enumeration value="MONTH"/>
           <xs:enumeration value="YEAR"/>
        </xs:restriction>
        </xs:simpleType>
      </xs:element>
   </xs:sequence>
</xs:complexType>
</xs:schema>
```

The schema above defines seven complex elements: MovingObjectSourceType, DataSourceInfoType, DataSourceParamsType, MovingObjectInfoType, IdInfoType, SpatialInfoType and TemporalInfoType.

MovingObjectSourceType is the root element. It encloses all the other elements that contain metadata about data sources and theirs moving objects. Information about each data source is described by the DataSourceInfoType element. It holds the data source name, its type and its access parameters. In this first version, the metadata file supports two types of data sources, KML and POSTGIS.

The access parameters are described by the DataSourceParamsType element. Since each type of data source requires a specific set of access parameters, this element is composed of key-value pairs instead of predefined elements. To access a PostGIS database, a user opens a connection that requires, at least, the database name, the server host name and its available port, the user name and its password. Otherwise, to access and open a KML file, a system only needs its path and name. A list of possible access parameters for each data source type is available in the TerraLib documentation available at www.terralib.org.

MovingObjectInfoType element carries information about the containers in the data sources that hold moving object observations. It includes: (1) the container type and name (containerType and containerName elements); (2) where the object identities are stored (IdInfoType type); (3) where the spatial and temporal information is stored (SpatialInfoType and TemporalInfoType types). The container that holds the moving car observations (Section 3.2) is the table car_trajectories. The identity of each car is stored in the column car_id. The columns location and date_time store the spatial and temporal information of each observation.

The IdInfoType element describes where the object identities are stored. The SpatialInfoType element describes where the spatial information is stored and its Spatial Reference System Identification (SRID). SRID is a unique number used to identify projected and local spatial coordinate system definitions. In the metadata file, the srid is optional since it can be already registered in the data source. A PostGIS database holds the srid of its feature tables in the geometry_columns table.

The TemporalInfoType element indicates where the temporal information is stored as well as its pattern and temporal resolution. Temporal pattern refers to the format of a textual representation of a date and time. For example, the text “01-03-2008” is ambiguous; it can represent the first day of March in 2008 or the third day of January in 2008. So, we have to inform what pattern it follows in order to understand its right meaning. ISO 8601:2004 (ISO, 2004) proposes some date and time format representations, such as DD-MM-YYYY or MM-DD-YYYY, and this work adopts them. The temporal pattern information is optional. It is only necessary when the temporal data is of a textual type.

Temporal resolution refers to the time granularity which must be considered to deal with temporal information. Each deforested region observation (shown in Fig. 1 (b)) can have a complete date associated to it (a day, a month and a year), such as '01-01-2003'. Nevertheless, the measurement of deforested regions is done yearly and so only the year must be considered in this date. In other words, the time resolution associated to it is YEAR. The possible time resolutions are: YEAR, MONTH, WEEK, DAY, HOUR, MINUTE and SECOND.

6. EXAMPLES

This section presents the moving object source metadata files related to the two examples presented in Section 3. These files contain necessary information to translate the spatial and temporal information in the KML file and in the PostGIS database into moving animals and moving cars, respectively. They follow the schema described in the previous section.

6.1 Moving Animals in the KML file

The moving object metadata file related to the KML file described in Section 3.1 is:

```
<?xml  version=”1.0"  encoding=”UTF-8"
      standalone=”no”?>
<MovingObjectSource
     xmlns:xlink=”http://www.w3.org/1999/xlink”
    xmlns:xsd=”http://www.w3.org/2001/XMLSchema
               instance”
    xmlns=”http://www.terralib.org/schemas/
           movingobjectsource”
   xsd:schemaLocation=”http://www.terralib.org/
          schemas/movingobjectsource
          movingobjectsource.xsd”>
<DataSourceInfo>
    <name>sea_elephants</name>
   <type>KML</type>
   <Params>
       <keyValuePair>NAME=C://sea_elephants.kml
      </keyValuePair>
   </Params>
</DataSourceInfo>
<MovingObjectInfo>
   <containerType>kml::FolderType</containerType>
   <containerName>40:  locations</containerName>
   <IdInfo>
        <name>kml::Placemark::Name</name>
   </IdInfo>
   <SpatialInfo>
        <name>kml::Placemark::Point</name>
        <srid>4326</srid>
   </SpatialInfo>
   <TemporalInfo>
        <name>kml::Placemark::TimeStamp</name>
        <resolution>SECOND</resolution>
   </TemporalInfo>
</MovingObjectInfo>
<MovingObjectInfo>
   <containerType>kml::FolderType</containerType>
   <containerName>41:  locations</containerName>
   <IdInfo>
        <name>kml::Placemark::Name</name>
   </IdInfo>
   <SpatialInfo>
        <name>kml::Placemark::Point</name>
        <srid>4326</srid>
   </SpatialInfo>
   <TemporalInfo>
        <name>kml::Placemark::TimeStamp</name>
        <resolution>SECOND</resolution>
   </TemporalInfo>
</MovingObjectInfo>
</MovingObjectSource>
```

The DataSourceInfo element contains the data source name (sea_elephants); its type (KML) and its access parameters (NAME=c://sea_elephants.kml). To access and open a KML file, a system needs only its name and path. So, we inform only the access parameter NAME.

In this file, there are two MovingObjectInfo elements that describe information about two containers of moving animal observations. The first container is a folder (containerType is kml::FolderType) called 40: locations (containerName). The second one is also a folder (containerType is kml::FolderType) called 41: locations (containerName). Both folders hold moving animals observations using the following KML elements: (1) kml::Placemark::Name to store the animal identities (IdInfo); (2) kml::Placemark::Point to store the animal locations (SpatialInfo); and (3) kml::Placemark::TimeStamp to store the temporal information associated to each location (TemporalInfo). The srid of the locations is 4326 that refers to the reference coordinate system WGS84. And, its temporal resolution is SECOND.

6.2 Moving Cars in the PostGIS database

The moving object metadata file related to the PostGIS database described in Section 3.2:

The DataSourceInfo element contains the data source name (cars), its type (POSTGIS) and its access parameters. These parameters contain necessary information to open a connection to a PostGIS database: the database name (NAME=stdatabase), the server host name (HOST=localhost), its available port (PORT=5432) and the user name (USER=postgres).

Information about how the moving cars are stored in this database is in the MovingObjectInfo element. The moving car observations are stored in a table (containerType is Table) called car_trajectories (containerName). The car identities are stored in a column called car_id (IdInfo). The car spatial locations are stored in a column called location (SpatialInfo) and their associated times in a column called date_time (TemporalInfo). We do not need to inform the srid in this file. It comes from the geometry_columns where the table car_trajectories is registered.

7. PROTOTYPE

We have built a new software module to deal with and analyze moving objects in the geographical library TerraLib. This module is based on the software architecture and strategy proposed in this paper. Besides that, we have adapted the open GIS TerraView to display and analyze moving objects, using this new module. TerraView is a geographical application built utilizing the TerraLib library (INPE, 2012). It is open source and developed by INPE.

Fig. 6 shows TerraView displaying the trajectories of two sea elephants (blue and yellow lines at the bottom) and the distance between both. The distance operation results in a time series (right side of the figure) that maps each time to the distance between both at that time. TerraView has loaded these two trajectories from the KML shown in Fig. 2, using the moving object source metadata file presented in Section 6.1. It can also display them through an animation over time.

We have built this module using three open source C++ software libraries: Xerces-C++, OGR and libpq. Xerces-C++ (http://xerces.apache.org/xerces-c/) is able to read and write XML data, checking its compliance with predefined schemas. It is used to read the moving object source metadata

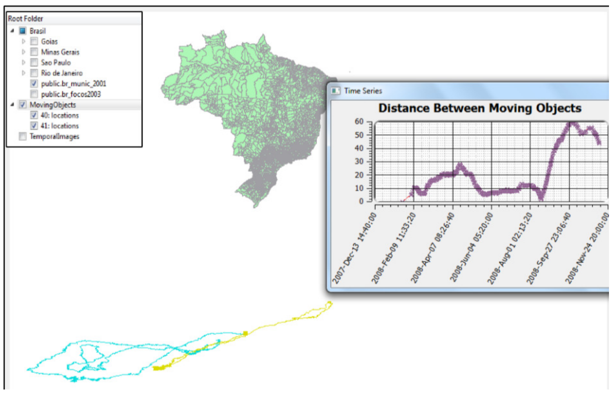

Fig. 6 - TerraView - displaying and analyzing sea elephant trajectories.

files that are XML files. OGR is provides read (and sometimes write) access to a variety of geographical vector file formats, including KML files (http://www.gdal.org/ogr/). We use the OGR LIBKML Driver to read KML files (http://www.gdal.org/ogr/drv_libkml.html). To access PostGIS databases, we use the libpq library (http://www.postgresql.org/docs/8.2/static/libpq.html).

8. FINAL REMARKS

The proposed approach consists in loading spatial and temporal information from data sources as it is and, afterwards, translating it into moving objects trajectories. To do this, it uses an additional metadata file that describes *how* moving objects are stored in a particular data source. This translation is essential to analyze the original information as moving object trajectories. To answer the question "*when and where did objects o_1 and o_2 meet each other (considering a meeting when the distance between two objects is less than 2 meters)?*", we need to structure the original data as moving object trajectories.

This strategy has two main advantages. The first one is that no change in the original data sources is required. It loads the original data as it is and uses the metadata file to know how to translate it into moving objects. This feature is particularly interesting when dealing with database servers and the final application do not have permission to change them.

The second advantage is that it can be easily extended to other data sources. In this paper, we show a prototype working with KML files and PostGIS databases. However, we can easily extend it to other kinds of data sources, such as GML or Oracle Spatial. To do it, we have to: (1) add the new types of data sources in the moving object source metadata file schema, including them in the element type of the DataSourceInfoType type (Section 5); and (2) build a new software piece in the DataAccess module that is able to load spatial and temporal information from these new data sources.

This proposal allows for dealing with moving objects data using common GIS spatial files and DBMS spatial extensions. In this perspective, this work advances towards a new generation of GIS that deals with spatiotemporal data.

REFERÊNCIAS BIBLIOGRÁFICAS

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E.; YERGEAU, F. Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C recommendation. **Report**. W3C, 2008.

CÂMARA, G.; VINHAS, L.; FERREIRA, K.; QUEIROZ, G.; SOUZA, R. C.; MONTEIRO, A. M. V.; CARVALHO, M. T.; CASANOVA, M. A.; FREITAS, U. M. TerraLib: An Open Source GIS Library for Large-scale Environmental and Socio-economic Applications. **Open Source Approaches to Spatial Data Handling**, Berlin, Springer-Verlag, 2008.

ERWIG, M.; GUTING, R. H.; SCHNEIDER, M.; VAZIRGIANNIS, M. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. **GeoInformatica**. v. 3, p. 265-291, 1999.

FERREIRA, K. R.; VINHAS, L.; MONTEIRO, A. M. V.; CÂMARA, G. **Moving Objects and KML Files**. *In*: Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE 2012) Workshop on Spatio Temporal data Integration and Retrieval. Washington D.C., USA, 2012.

GUTING, R. H.; SCHNEIDER, M. **Moving Objects Databases**. San Francisco, Morgan Kaufmann, 2005.

INPE. Projeto MEOP: INPE, 2012. Available at: <http://www.inpe.br/crs/pan/pesquisas/telemetria.php>. Access data: 12/07/2012.

INPE. TerraView software. São José dos Campos, SP: INPE, 2012. Available at: <http://www.dpi.inpe.br/terraview_eng/index.php>. Access data: 12/07/2012.

INTERNATIONAL STANDARD ORGANIZATION (ISO). ISO 8601:2004: Data elements and interchange formats - Representation of dates and times. **Report**. Geneva, Switzerland, 2004.

INTERNATIONAL STANDARD ORGANIZATION (ISO). ISO 19141:2008: Geographic information - Schema for moving features. **Report**. Geneva, Switzerland, 2008.

OPEN GEOSPATIAL CONSORTIUM (OGC): OpenGIS Implementation Specification for Geographic Information – Simple Feature Access - Part 1: Common architecture. Reference number: OGC 06-103r3. Version: 1.2.0. **Report**. Available at <http://www.opengeospatial.org>. 2006a.

OPEN GEOSPATIAL CONSORTIUM (OGC): OpenGIS Implementation Specification for Geographic Information – Simple Feature Access - Part 2: SQL option. Reference number: OGC 06-104r3. Version: 1.2.0. **Report**. Available at <http://www.opengeospatial.org>. 2006b.

OPEN GEOSPATIAL CONSORTIUM (OGC). OpenGIS Geography Markup Language (GML) Encoding Standard. Reference number: OGC 07-036. Version: 3.2.1. **Report**. Available at: <http://www.opengeospatial.org>. 2007.

OPEN GEOSPATIAL CONSORTIUM (OGC). OGC KML. Reference number: OGC 07-147r2. Version: 2.2.0. **Report**. Available at: <http://www.opengeospatial.org>. 2008.

PELEKIS, N.; FRENTZOS, E.; GIATRAKOS, N.; THEODORIDIS, Y. **HERMES: Aggregative LBS via a Trajectory DB Engine**. *In:* Proceedings of the ACM SIGMOD' 08 Conference. Vancouver, BC, Canada. 2008.

SPACCAPIETRA, S.; PARENT, C.; DAMIANI, M.; MACEDO, J. A. F.; PORTO, F.; VANGENOT, C. A conceptual view on trajectories. **Data & Knowledge Engineering**. v. 65, p. 126-146, 2008.