PostGIS-T: towards a spatiotemporal *PostgreSQL* database extension

Rolf E. O. Simoes[1], Gilberto Ribeiro de Queiroz[1], Karine Reis Ferreira[1], Lubia Vinhas[1], Gilberto Camara[1]

[1]Instituto Nacional de Pesquisas Espaciais (INPE)
Caixa Postal 15.064 – 91.501-970 – São José dos Campos – SP – Brazil

{rolf.simoes, gilberto.queiroz, karine.ferreira,

lubia.vinhas}@inpe.br

Abstract. *The temporal dimension of spatial data has been the subject of discussion in the literature for a long time. While there are numerous Database Management System (DBMS) solutions only for spatial dimension, we did not observe the same situation for* spatiotemporal *data. Considering this gap, our purpose is to design and implement an extension to the DBMS PostgreSQL that is based on a formal spatiotemporal algebra in order to incorporate representations of spatiotemporal data within the DBMS. The proposed extension can be used in a large range of applications. We intend that this extension be a reasonable framework to store and handling observational remote sensing data usually present in applications like animal migration researches, wildfires monitoring, vessel tracking for monitoring fishing, and the like. In this work, we show how to apply it in a case study based on spatiotemporal data collected from drifting buoys belonging to the NOOA's Global Drifter Program.*

1. Introduction

Earth Observation data generation has been increased since the last decades. This phenomenon occurs, considering that a great amount of data are daily collected by different missions such as *CBERS*[1] in Brazil/China, *Landsat*[2] in the USA and *Sentinel*[3] in Europe. The development of mobile positioning technologies and its low costs are also factors that enable spatial data gathering through time.

These different data sources are associated with temporal dimension, mainly by allowing the monitoring of spatially located objects in time, either by allowing the time analysis by increasing the temporal resolution of the observations. The collection, representation and processing of this data have been largely facilitated by database management systems (DBMS) and their spatial extensions, which are based on international standards such as the OGC Simple Feature Specification [Herring 2011] and ISO geographic information standards [Kresse and Fadaie 2004]. Furthermore, while there are numerous DBMS solutions supporting the spatial dimension, we do not observe the same situation for spatiotemporal data.

[1]http://www.cbers.inpe.br/
[2]http://landsat.usgs.gov/
[3]https://sentinel.esa.int/

The temporal dimension of spatial data has been the subject of discussion in the literature for a long time. Additionally, some conceptual systems regarding to representation of spatiotemporal data have been proposed [Camara et al. 2014, Ferreira et al. 2014, Erwig et al. 1999]. One of these systems, particularly discussed in [Ferreira et al. 2014], is structured around the concept of observations, the basic unit of data acquisition of a spatial temporal phenomenon. From observations, it is possible to generate three types of spatiotemporal data: *time series*, *trajectories* and *coverages*. With these three types, it is possible to represent the geo-ontological concepts of object and field and to define a space-time algebra. The authors have implemented their work as a C++ library, that works as middleware between the actual sources of data (databases of flat files, for example) and their conceptual model.

We have observed a number of solutions for processing spatiotemporal data as client side solutions. This approach may presents as disadvantage the introduction of an overhead by transferring data between processes or even between machines when those data are managed over a network. Differently from that, we propose a model that works inside the database system.

Besides that, this approach can avoid memory issues for huge volume of data since this is a solved subject in the context of relational DBMS such as PostgreSQL. Moreover, spatiotemporal data usually consumes a higher volume of memory space. Considering this drawback, our goal is to implement an extension to the DBMS PostgreSQL to provide support for spatiotemporal types within the DBMS. The PostGIS *geometry* type is used as a basic spatial representation for the extension. Furthermore, temporal dimension was integrated to get our spatiotemporal type.

Here, we propose a spatialtemporal database extension. Based on the work of [Ferreira et al. 2014], we introduce three new spatiotemporal data types for the DBMS PostreSQL. Moreover, we implemented some algebra functions of those data. In order to demonstrate how our extension can manipulate real data observations, we conducted a case study based on the Global Drifter Program (GDP) database.

2. Background

According to [Sinton 1978], space, time and theme (or a quantity measure) are the three dimensions of geographical structure and observation. In such a way, it is possible to observe by fixing one dimension, controlling another and measuring the other. Hence, six types or structure of observation can be produced. Proceeding in this manner [Ferreira et al. 2014] claim that we can capture all kind of spatiotemporal phenomena exhaustively with three of them:

1. **time series**: fix time, control space and measuring theme;
2. **trajectory**: fix time, control theme and measuring space;
3. **coverage**: fix space, control time and measuring theme;

Time series and trajectory play an important role considering the necessity to analyze data along time. The main difference between them remains in space dimension: in case of trajectory we are interested in measuring the space locations of an given observed phenomena, whereas in time series the measured data is gathered from a fixed space.

It is easy to see that the fix operation defines a domain from which the measured data must be filtered in. In this manner, a geo-located time series are sequences of ob-

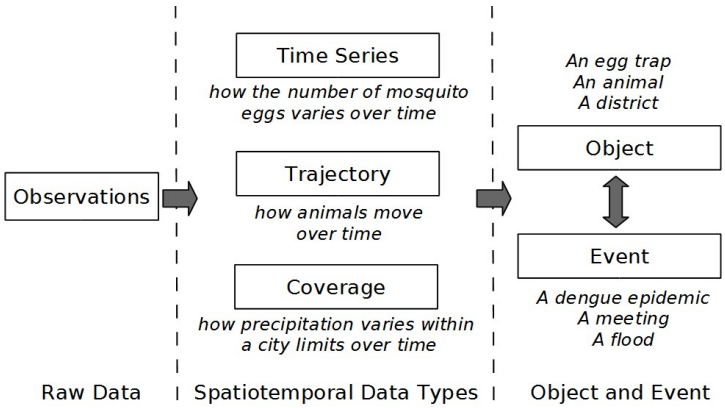

Figure 1. The proposed model. Source: [Ferreira et al. 2014, p. 258]

servations over time of a measured phenomena that takes place in a given space domain. For instance, Landsat series spans over 40 years and is a very informative temporal record of radiance of geo-located sites represented by pixels [Roy et al. 2014]. This database can provide significant data about each pixel spectral response time series that can be further analysed to give us information about land use and land cover change over time [Maus 2016].

On the other hand, trajectories can be seen as a sequence in time dimension of geo-located observed geometries (points, lines, polygons or volumes) that are associated to a given theme. For instance, database monitoring oceanic buoys give us a rich data about its positions, water temperature and salinity. One can be interested on tracking buoys positions over time to capture surface oceanic chains [Lumpkin and Pazos 2007].

Here, the time dimension organizes different space locations of a fixed theme (the buoy). Another example, the policy for preventing Dengue epidemic may depend on monitoring of egg traps over time [Regis et al. 2009]. In this case, one may be interested to know all locations over time that presents a given higher level of captured mosquito eggs (threshold theme). The main difference between these two examples of trajectories is the kind of fixed theme. In the first one, the fixed theme was an identifiable object entity, a buoy. In the second one, we used a measured data not related to a specific object entity but a condition that may involve a set of objects. This condition refers to object's properties and can denote events. For example, if we are interested in flood monitoring in some urban area, we may relate the event *flood* to a condition of water precipitation metered by a set of pluviometers.

Objects and events play a key role on the interpretation of a spatiotemporal data [Ferreira et al. 2014, Worboys 2005]. In this approach, an object is any identifiable entity over time and an event is an episode on the time that may relates to one or more objects. Episodes have a definite begin and an end. Events may be distinguished by punctual occurrence, if it occurs instantaneously, or durative, if it takes some time [Galton 2004]. An event does not change over time and we can derive them from conditions of spatial and non-spatial properties of objects, as we can see from the example above. Figure 1 summarizes this model.

Spatiotemporal structures can be implemented on computer systems as data types. A data type is a set of values over which we can define some operations. To formalize these ideas, [Ferreira et al. 2014] follow [Guttag and Horning 1978] and propose an algebraic specification that consists of (1) definitions of type names, their domains, ranges, and operations; (2) a set of axioms that expresses truth relations among those operations. PostgreSQL works similarly: all data has a type with an explicit or implicit ranges; all functions operates over some types and returning new data. To comply with [Ferreira et al. 2014] abstract model, we just need to guarantee that PostGIS-T functions be implemented in such a way that its behavior be consistent with axioms specification. In what follows, we describe how our extension implements and represents spatiotemporal data.

3. PostGIS-T model

PostGIS-T introduces the `SPATIOTEMPORAL` type, a composite PostgreSQL type that plays the same role as `SpatioTemporal` type in [Ferreira et al. 2014]. The main difference between them is that unlike `SpatioTemporal`, `SPATIOTEMPORAL` type can represent both `TimeSeries`, `Trajectory` and, `Coverage` and does not work with the idea of abstract type that is specialized later.

In order to access and manipulate the data, we must define functions. To achieve the same level of specification, we have implemented each declared operator in [Ferreira et al. 2014] model definition. We list all function signature in Table 1.

PostGIS-T stores observations as tuples in one or more relations that can be further queried to instantiate `SPATIOTEMPORAL` data. To get a new `SPATIOTEMPORAL` data, we use `TST_SPATIOTEMPORAL()` aggregate function.

As we can see, PostGIS-T was built over PostgreSQL and PostGIS. For example, spatial representations are `GEOMETRY` values, a type introduced by PostGIS. As a first prototype we have limited measure values as `NUMERIC` type, so that an observation must be a triple (`TIMESTAMP`, `GEOMETRY`, `NUMERIC`). In order to instantiate `SPATIOTEMPORAL` data we must call `TST_SPATIOTEMPORAL()` inside a query informing where to find these values.

Spatiotemporal data may be interpreted differently depending on the underlying phenomena it represents. As previously discussed, we could conceive an observation as taking place continuously or instantaneously. Our data type was defined to accommodate all spatiotemporal data phenomena without a specific semantic. For example, we does not know in advance if we should conceive an observation as an occurrent or a durative one, or if the sample observation is a time series or a trajectory. To overcome this limitation, we have implemented some functions that can be combined in order to get the right phenomena interpretation. These functions are `TST_ESTIMATE_MEASURE()`, `TST_ESTIMATE_LOCATION()`, `TST_RESAMPLE_TIME()`, and `TST_COVERAGE()`.

If we are interested in how to get an approximate measure between two empirical observed phenomenon we must take into account its underlying nature, that is, if it refers to an object or to an event. For example, we took observations of a drifter floating on the ocean at times `10:00AM` and `11:00AM`, and we are interested to know its location at `10:30AM`. We may assume that a linear interpolation would give us a good

approximation and so we call TST_ESTIMATE_LOCATION() function informing the SPATIOTEMPORAL data, the time of interest to calculate the interpolation (in this case 10:30AM), and the interpolation method name 'LINEAR' as parameters. For now, we have implemented a small set of interpolation methods that we can use, 'LINEAR', 'LAST' or 'NEAREST', meaning, respectively, simple linear interpolation, last registered location or measure before or equal a given time, and the closest time registered location or measure of a given time. The process to estimate a measure from a SPATIOTEMPORAL is analogous.

Other useful application of interpolators are re-sampling data. PostGIS-T is able to re-sampling time observations between a time range at regular time resolution with the function TST_RESAMPLE_TIME(). This function returns a regular time spaced SPATIOTEMPORAL data whose locations and measures were estimated according to a given interpolation method (see more details in section 4).

Furthermore, if we are interested to re-sampling our observations through space in order to produce what [Ferreira et al. 2014] calls *coverage*, we use the function TST_COVERAGE(). This function returns a SPATIOTEMPORAL data whose observations refer to a regular extent over which measures are aggregated into an unique value according to a given aggregate strategy (e.g. 'COUNT', 'AVG', 'MIN', 'MAX' and 'AREA'). The result is a time flattened spatiotemporal data where no gap and no overlapping area exists between two adjacent extents. In [Ferreira et al. 2014] model, this represents a Coverage.

Other functions are related to operations that retrieve SPATIOTEMPORAL properties or subset of the spatiotemporal data. For instance, the functions TST_BEGINS() and TST_ENDS() indicates the start and the end times for the sampling, whereas TST_HULL() gives its convex hull polygon where observations took place. Finally, to get the max and the minimum values of the measured data, we must use the functions TST_MIN() and TST_MAX(), respectivelly. On other hand, the TST_DURING() function returns all observations that have been made in a given time range. Likewise, to get only a given location samples, we use the function ST_INTERSECTION() passing to it the area or point of interest.

A complete and comprehensive documentation can be found in the extension's webpage https://gitlab.dpi.inpe.br/postgis-t. In the following section, we demonstrate an application of how to use PostGIS-T.

4. The Global Drifter Program: a case study with real spatiotemporal data

Regarding to spatiotemporal data and its complexity, we chose the satellite-tracked surface drifting buoy (drifter) data to evaluate and get experienced with the extension implementation details in PostgreSQL environment. The Global Drifter Program (GDP) is a branch of the National Oceanic and Atmospheric Administration (NOAA). It aims to maintain a global satellite-tracked surface drifting buoys and to provide the data set for scientific purposes, as climate predictions and climate research and monitoring. In this manner, GDP produces observations from most areas of the world's oceans at sufficient density to map the mean currents at one degree resolution [Lumpkin and Pazos 2007].

Drifter is a surface buoy connected with a subsurface drogue. Its observations have been largely used in oceanographic and climate researches. The main use of this data is to

Function signature	Return type
TST_SPATIOTEMPORAL(TIMESTAMP, GEOMETRY, NUMERIC)	SPATIOTEMPORAL
TST_ESTIMATE_MEASURE(SPATIOTEMPORAL, TIMESTAMP, TEXT)	NUMERIC
TST_ESTIMATE_LOCATION(SPATIOTEMPORAL, TIMESTAMP, TEXT)	GEOMETRY
TST_RESAMPLE_TIME(SPATIOTEMPORAL, TSRANGE, INTEGER, TEXT)	SPATIOTEMPORAL
TST_COVERAGE(SPATIOTEMPORAL, INTEGER, INTEGER, TEXT)	SPATIOTEMPORAL
TST_BEGINS(SPATIOTEMPORAL), TST_ENDS(SPATIOTEMPORAL)	TIMESTAMP
TST_HULL(SPATIOTEMPORAL)	GEOMETRY
TST_AFTER(SPATIOTEMPORAL, TIMESTAMP)	SPATIOTEMPORAL
TST_BEFORE(SPATIOTEMPORAL, TIMESTAMP)	SPATIOTEMPORAL
TST_DURING(SPATIOTEMPORAL, TSRANGE)	SPATIOTEMPORAL
TST_INTERSECTION(SPATIOTEMPORAL, GEOMETRY)	SPATIOTEMPORAL
TST_DIFFERENCE(SPATIOTEMPORAL, GEOMETRY)	SPATIOTEMPORAL
TST_MEASURE(SPATIOTEMPORAL, TIMESTAMP)	NUMERIC
TST_MEASURE(SPATIOTEMPORAL, GEOMETRY, TEXT)	NUMERIC
TST_MIN(SPATIOTEMPORAL), TST_MAX(SPATIOTEMPORAL)	NUMERIC
TST_LESS(SPATIOTEMPORAL, NUMERIC)	SPATIOTEMPORAL
TST_GREATER(SPATIOTEMPORAL, NUMERIC)	SPATIOTEMPORAL
TST_BETWEEN(SPATIOTEMPORAL, NUMRANGE)	SPATIOTEMPORAL
TST_LOCATION(SPATIOTEMPORAL, TIMESTAMP)	GEOMETRY
TST_EQUALS(SPATIOTEMPORAL, SPATIOTEMPORAL)	BOOLEAN
TST_INTERPOLATOR(SPATIOTEMPORAL)	TEXT
TST_SETINTERPOLATOR(SPATIOTEMPORAL, TEXT)	SPATIOTEMPORAL
TST_OBSERVATIONS(SPATIOTEMPORAL)	INTEGER

Table 1. List of all defined PostGIS-T functions

map oceanic surface currents of different seas and oceanic regions of the planet. Also, the data can be used to calibrate satellite sensors. Each drifter has an unique identifier code and is equipped with sensors that periodically measure properties such as salinity and surface temperature of the water. All these data are subsequently transmitted to satellites. The drifter's position and velocity are usually inferred by *Doppler shift*, which occurs during the transmission step. The positioning system, also known as *Argos*, provides drifter locations with $O\left(100m\right)$ errors. The raw data are then assembled and normalized by the Drifter Data Assembly Center of the Atlantic Oceanographic and Meteorological Laboratory (DAC/AOML).

The GDP drifter database used contains $2,263,842$ collected locations with respective zonal and meridional velocities observations for 408 drifters worldwide. The time resolution of the observation is one hour. More information about how this database was collected can be see in [Elipot et al. 2016].

All data sample were loaded in a table of observations as defined in the Listing 1. Subsequently, we proceeded with data instantiation by calling the aggregate function `TST_SPATIOTEMPORAL(TIMESTAMP, GEOMETRY, NUMERIC)`. All the following queries uses this table.

Sometimes it is useful to change the amount of observations of a given

```
CREATE TABLE buoy_obs_st (
  buoy_id    INTEGER PRIMARY KEY,
  spatiotemp SPATIOTEMPORAL
);
```

Listing 1: Definition of the spatiotemporal table `buoy_obs_st`.

data. In PostGIS-T, we can obtain a new sample of a trajectory by the function `TST_RESAMPLE_TIME()`. This function receives as parameter the spatiotemporal data, the time interval that we are want re-sampling, the number of observations to be re-sampled, and the interpolation method. Note that the extension makes no assumption about the observation continuity or duration. In this regard, an appropriate interpolation method must be informed by the user. The queries in Listing 2 shows how do we re-sample observations. A graphical result is presented in Figure 2.

```
SELECT buoy_id,
       TST_RESAMPLE_TIME(
           spatiotemp,
           TST_OBSERVATIONS(spatiotemp) / 10,
           'LINEAR')
  FROM buoy_obs_st;
```

```
SELECT buoy_id,
       TST_RESAMPLE_TIME(
           spatiotemp,
           TST_OBSERVATIONS(spatiotemp) * 2,
           'LINEAR')
  FROM buoy_obs_st;
```

Listing 2: Re-sampling on time. Query on the top (bottom) reduces (increase) the time resolution.

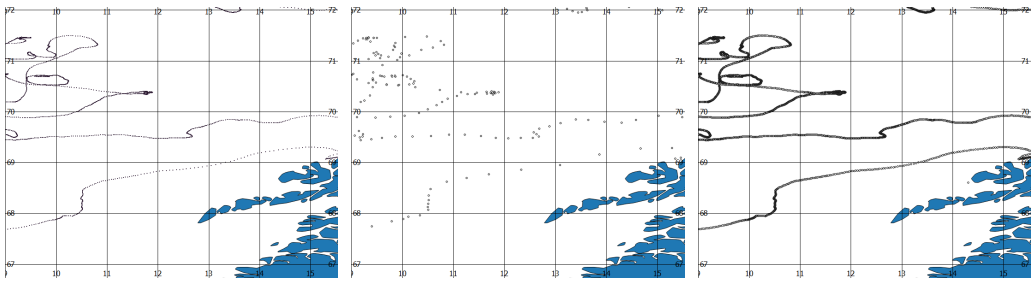

Figure 2. Trajectory re-sampling. From left to right: original data, re-sampling for every 10 hours, re-sampling for every 30 minutes.

Re-sampling technique may be employed to synchronize data observations, in order to speed-up queries that performs sequential time calculations. Comparing trajectories (a) and (b) from the Figure 2, we can note that some trajectory sections can be well ap-

proximated by few interpolated observations. These sections mainly resembles straight segments.

Another useful application of these data is to estimates the mean velocity of ocean currents over different regions and a given period of year. This is a very important question in related climate researches. Velocity may be represented by a vector that informs us about the direction and speed magnitude of the moving entity. Here, re-sampling technique may be useful if we would like to measure the mean direction of currents.

Suppose now we are interested to measure the mean velocity magnitude of a drifter between a small time range (for the sake of simplicity). How can we proceed in PostGIS-T? First we need a function that returns the observations of a given time interval. This function is `TST_DURING()` which returns a `SPATIOTEMPORAL` data. From this result, `TST_COVERAGE()` creates a regular grid whose extent is the same as observations location bounding box.

The dimensions of that grid is given as parameters to the function. A new measure is then calculated for each cell grid according to aggregate strategy. Thereafter, a new `SPATIOTEMPORAL` data is instantiated and returned. The corresponding query is presented in Listing 3 and a graphical representation is showed in Figure 3.

```
SELECT buoy_id,
       TST_COVERAGE(
           TST_DURING(
               spatiotemp,
               TSRANGE(
                   '2015-05-18 14:00:00',
                   '2015-05-20 14:00:00', '[]')
           ), 7, 13, 'AVG')
  FROM buoy_obs_st
 WHERE buoy_id = 132470;
```

Listing 3: Coverage.

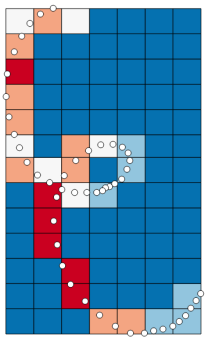

Figure 3. Coverage calculated from drifter mean velocity. Blue-Red colors denotes lower-higher velocities.

From Figure 3 we can see that not all cells grid contains a drifter observation. Only those regions that have at least one register has a measure value equals to that of

velocity average. In order to fill those cells grid we would need a spatial interpolator. The process to get a coverage of velocity magnitudes is similar.

5. Conclusions

Here, we have proposed a spatiotemporal database extension based on conceptual model of [Ferreira et al. 2014]. We have implemented this model with some adaptations for the relational database environment. However this adaptation was not conceptual but operational. For example, in this preliminary version we did not provide a way to extend the base of interpolation methods used as a parameter of function like `TSTRESAMPLETIME`. In spite of that, the implementation shows us that the spatiotemporal model proposed in [Ferreira et al. 2014] is feasible in an relational DBMS context.

Our first approach suggest that this model may be more indicated to applications of sparse geo-referenced data like movable objects (e.g. drifters, ship trajectories) and observed events (e.g. wildfires, disease occurrences). This applications should work better over snapshots of the original data as the task of packing a huge spatiotemporal tuples is expensive. However, it is too early to note some processing improvement from our data type columnar design.

Further research and development consist of: (a) designing a compact and efficient disk storage layout for values of `SPATIOTEMPORAL` type; (b) introduce the notion of subtypes of `SPATIOTEMPORAL` as type modifiers (`TIMESERIES`, `TRAJECTORY` and `COVERAGE`), which will give more constraint about the data and the result of operations; (c) how to include spatiotemporal indexes that could take advantage of approximations of spatiotemporal data; (d) explore the extension with big spatiotemporal data applications in a database cluster environment.

As a first approach, we have prototyped the PostGIS-T extension with the high level SQL and PL/pgSQL languages. The next version of this extension will be developed in the C programming language and it will include a *view* named `spatiotemporal_columns` with the same purpose of PostGIS `geometry_columns`. Moreover, we will provide a larger number of functions to deal with different spatiotemporal data manipulation demands.

References

Camara, G., Egenhofer, M. J., Ferreira, K., Andrade, P., Queiroz, G., Sanchez, A., Jones, J., and Vinhas, L. (2014). Fields as a generic data type for big spatial data. In *International Conference on Geographic Information Science*, pages 159–172. Springer.

Campbell, J. B. and Wynne, R. H. (2011). *Introduction to remote sensing*. Guilford Press.

Elipot, S., Lumpkin, R., Perez, R. C., Lilly, J. M., Early, J. J., and Sykulski, A. M. (2016). A global surface drifter data set at hourly resolution. *Journal of Geophysical Research: Oceans*.

Erwig, M., Gu, R. H., Schneider, M., Vazirgiannis, M., et al. (1999). Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296.

Ferreira, K. R., Camara, G., and Monteiro, A. M. V. (2014). An algebra for spatiotemporal data: From observations to events. *Transactions in GIS*, 18(2):253–269.

Galton, A. (2004). Fields and objects in space, time, and space-time. *Spatial cognition and computation*, 4(1):39–68.

Goncalves, M., Netto, M., Costa, J., and Zullo Junior, J. (2008). An unsupervised method of classifying remotely sensed images using kohonen self-organizing maps and agglomerative hierarchical clustering methods. *International Journal of Remote Sensing*, 29(11):3171–3207.

Guttag, J. V. and Horning, J. J. (1978). The algebraic specification of abstract data types. *Acta informatica*, 10(1):27–52.

Herring, J. (2011). Opengis implementation standard for geographic information-simple feature access-part 1: Common architecture. *OGC Document*, 4(21):122–127.

Kemp, K. K. (1996). Fields as a framework for integrating gis and environmental process models. part 1: Representing spatial continuity. *Transactions in GIS*, 1(3):219–234.

Kresse, W. and Fadaie, K. (2004). *ISO standards for geographic information*. Springer Science & Business Media.

Lillesand, T., Kiefer, R. W., and Chipman, J. (2014). *Remote sensing and image interpretation*. John Wiley & Sons.

Lumpkin, R. and Pazos, M. (2007). Measuring surface currents with surface velocity program drifters: the instrument, its data, and some recent results. In A. Griffa, A. D. Kirwan, A. J. M. T. O. and Rossby, T., editors, *Lagrangian analysis and prediction of coastal and ocean dynamics*, chapter 2, pages 39–67. Cambridge University Press New York, NY.

Maus, V. W. (2016). *Land Use and Land Cover Monitoring Using Remote Sensing Image Time Series*. PhD thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

Melton, J. and Eisenberg, A. (2001). Sql multimedia and application packages (sql/mm). *ACM SIGMOD Record*, 30(4):97–102.

OGC (2010). OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option. Technical report, Open Geospatial Consortium.

OGC (2011). OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. Technical report, Open Geospatial Consortium.

Regis, L., Souza, W. V., Furtado, A. F., Fonseca, C. D., Silveira Jr, J. C., Ribeiro Jr, P. J., Melo-Santos, M. A. V., Carvalho, M. S., and Monteiro, A. (2009). An entomological surveillance system based on open spatial information for participative dengue control. *Anais da Academia Brasileira de Ciências*, 81(4):655–662.

Richards, J. A. and Richards, J. (1999). *Remote sensing digital image analysis*, volume 3. Springer.

Roy, D. P., Wulder, M., Loveland, T., Woodcock, C., Allen, R., Anderson, M., Helder, D., Irons, J., Johnson, D., Kennedy, R., et al. (2014). Landsat-8: Science and product vision for terrestrial global change research. *Remote Sensing of Environment*, 145:154–172.

Sinton, D. (1978). The inherent structure of information as a constraint to analysis: mapped thematic data as a case study. In Dutton, G., editor, *Harvard Papers on Geographic Information Systems*, volume 6, pages 1–17. Harvard University Cambridge, MA.

Worboys, M. (2005). Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science*, 19(1):1–28.