



# Coverage Representation in TerraLib

Vitor Dantas

Marcelo G. Metello

Melissa Lemos

Marco A. Casanova

Tecgraf – Computer Graphic Technology Group  
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)



- Framework for representing coverages in TerraLib
- Compatible with OGC abstract specifications
- Addressing performance issues
- Fitting the current TerraLib coding conventions
  - Generic programming
  - Iterators

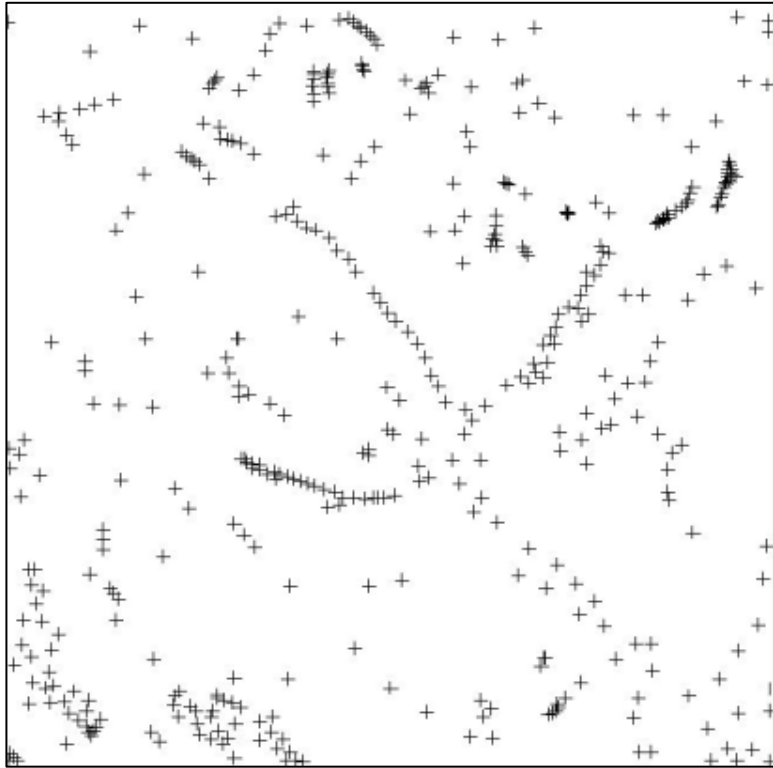


- Two basic ways of modeling geospatial features
  - Choice depends on the application
- Features with geometry
  - A.k.a. geo-objects, discrete objects
  - Focus on spots
  - For cities, parcels, mountains...
- Coverages
  - A.k.a. geo-fields, continuous fields
  - Focus on the whole picture
  - For altimetry, vegetation, soil properties...

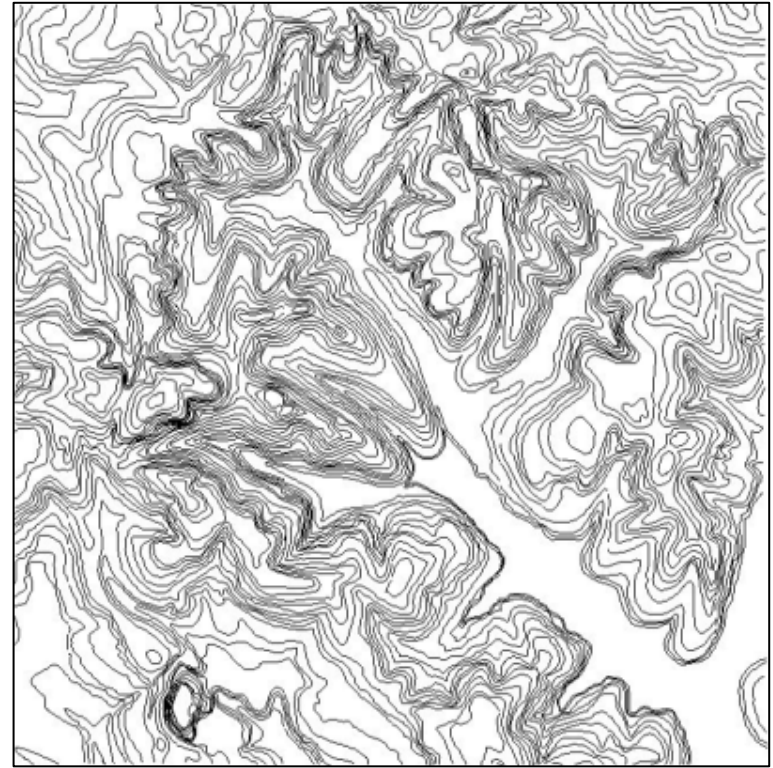


- What is needed to define a Coverage:
  - Underlying representation
  - Interpolation method

## Coverage Representations (1/2)

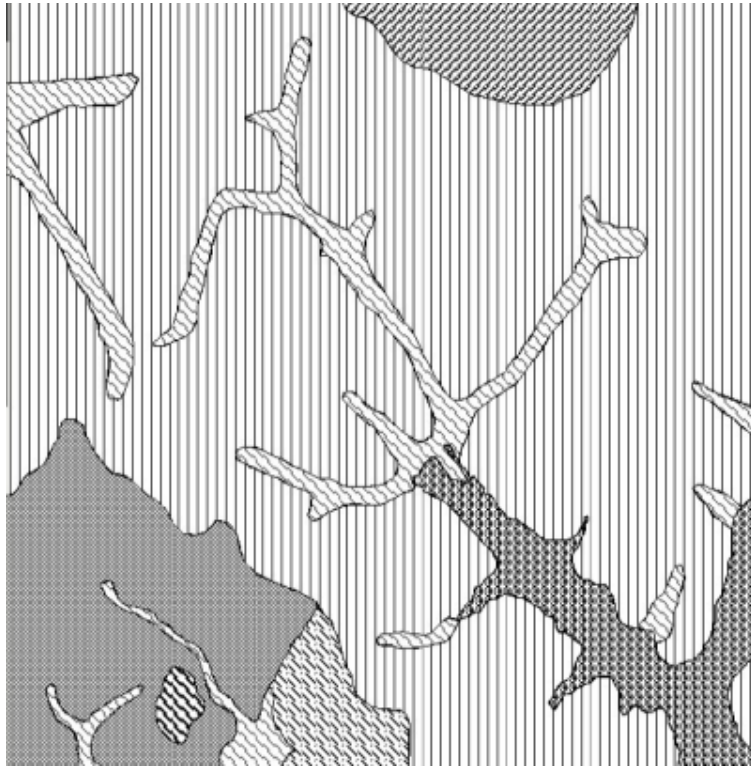


- Sample points

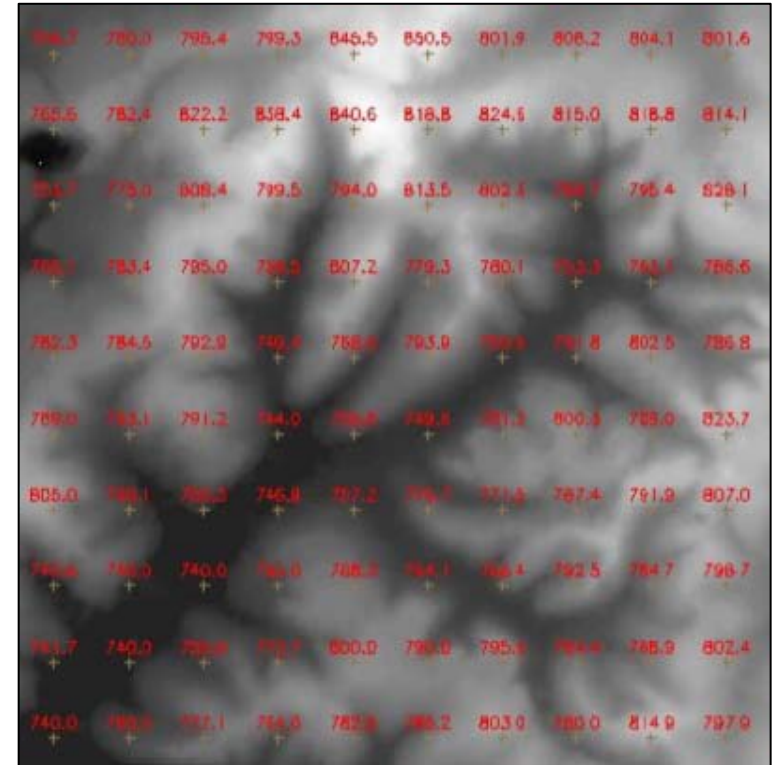


- Contour Lines

# Coverage Representations (2/2)



- Planar Subdivision



- Raster



- TerraLib TeCoverage interface

## TeCoverage<T>

```
+ begin (poly : TePolygon, relation : TeSpatialRelation) : TeCoverage<T>::iterator  
+ end (poly : TePolygon, relation : TeSpatialRelation) : TeCoverage<T>::iterator  
+ evaluate (position : TeCoord2D, value : vector<double>) : void
```

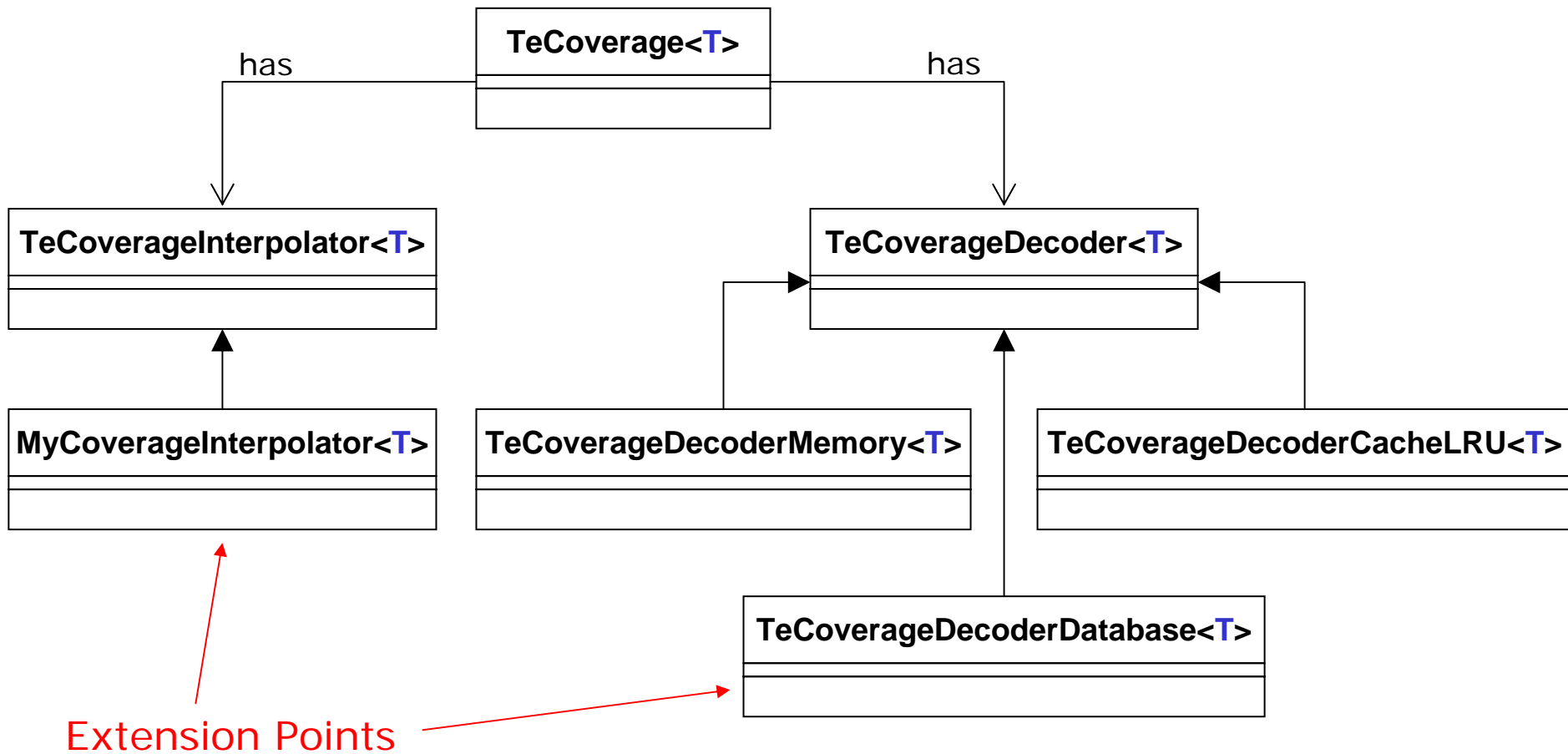
## TeCoverage<T>::iterator

```
+ operator* () : T  
+ operator[] (dimension : int) : double  
+ operator++ () : void
```



- Clustering
  - Few large blocks instead of many small objects
  - Improved load times
- Memory caching
  - Data access pattern by spatial proximity:  
Cluster by spatial proximity
  - Less frequent access to the database

- Overall architecture of Coverage framework





- Creating and initializing a TeCoverage instance

```
// Setup coverage parameters
TeCoverageParams params = TeCoverageParams(myDB, 1);
params.setPersistenceType(TePERSISTENCE_DATABASE_CACHELRU);

// Create and initialize coverage
TeCoverage<TePoint> coverage = TeCoverage<TePoint>(params);
coverage.init();
```



- Make a spatial query on the coverage instance

```
// Make spatial query
TeCoverage<TePoint>::iterator it = coverage.begin(poly);
TeCoverage<TePoint>::iterator end = coverage.end(poly);

// Iterate over selected points
while (it != end)
{
    TePoint& point = *it;
    double temperature = it[0];
    double salinity = it[1];
    it++;
}
```



- Get interpolated value at an arbitrary location

```
// Specify arbitrary location
TeCoord2D location = TeCoord2D(129.0, 104.0);
vector<double> values;

// Get interpolated values
coverage.evaluate(location, values);
double temperature = values[0];
double salinity = values[1];
```



- Framework implementation completed
- Sample Points Coverage implemented
  - TePoint as underlying representation
  - Nearest Neighbour as interpolation method
- To appear in the next version of TerraLib
- Future Work
  - Extensive performance testing and optimization
  - Implementation of other Coverages

- OGC (2000) "The OpenGIS Abstract Specification - Topic 6: The Coverage Type and its Subtypes", version 6.0, Open Geospatial Consortium Inc, USA.
- Vinhas, L. (2006) "Um subsistema extensível para o armazenamento de geo-campos em bancos de dados geográficos", PhD Thesis, INPE, São José dos Campos, Brazil (in Portuguese).
- Vinhas, L. and Souza, R. C. M. (2005) "Tratamento de dados matriciais na TerraLib", In "Bancos de Dados Geográficos", Edited by M. Casanova et al, Editora MundoGEO, Brazil (in Portuguese).
- The TerraLib Project. <http://www.terralib.org>