Raster data handling in spatial databases: the case for images

Lúbia Vinhas , Ricardo Cartaxo , Gilberto Camara , Karine Ferreira,
Antonio Miguel Vieira Monteiro, DPI/INPE

INPE's motivation

- Satellite acquired data is everywhere!
- Satellite derived observational data
- – Large mass of highly dimensional spatio-temporal data
- 30 years of lessons learned from dealing with *high dimensional spatio-temporal* **image data** from earth remote sensing satellites and airborne sensors.
- INPE's image data centre project

The rationality for having images stored in DBMS

- A new generation of spatially enabled DBMS;
- Huge amount of data that must be dealt with, coming from a variety of sensors over a variety of platforms;
- Make data recovery and integration a more easy task;

The challenges

Technological

- Efficient spatially enabled DBMS
- Provide spatial operations on spatial data types stored in different DBMS

Scientific & technological

- Methods and techniques for Parameter/pattern/information-content extraction from high dimensional integrated spatio-temporal datasets

The applications needs driving the technology needs

- Run in a corporative environment
- Access data by internet and intranet
- Typical use of image data is visualization
- Integrates descriptive data stored in a conventional object-relational DBMS
- Integrates vector data

The basic requirements

- The image data should be stored in the existing *object-relational database management system*
- Data integrity and consistency
- Independent and effective access by users of multiple applications

The research needs driving the scientific needs

- Parameter/pattern/ information-content extraction:
- Another typical use of image data is getting *information* out of it. – Needs new methods and algorithms

Our aim

To provide a research testbed for dealing with large raster datasets that can help in:

- Enabling data integration. Grid data, image data, observations data and other geographic data types could be used together;
- Enabling easy new algorithms development for parameter extraction from satellite image datasets;
- Enabling the test of new spatial-temporal statistics methods for 'mining' high-dimensional datasets

And where we are at this stage

- Advances in database technology provide support for major advances in non-conventional database applications
- Spatial data in relational databases
- Integration of spatial data types in object-relational database management systems
- Efficient handling of spatial data types
- Vector: polygons, lines and points
- Raster data structures: images or any other gridded data
- Tools for query and manipulation of spatial data

It is time for images . . .

- A special interest in the spatial databases community is the efficient handling of *raster* data
- An approach is to develop *specialized* image data servers
- Main advantage – the capacity of performance improvements

Our approach

- Include building raster data management capabilities into object-relational database management systems

Main advantages

- easy interface with existing user environments
- To accommodate not only typical image data, but also raster data in general

Our technological solution – TerraLib (http://www.terralib.org)

This work is part of the development of TerralLib

- TerraLib is an Open Source Licenced (LGPL) Geographic Library for providing support for the development of Geographic Applications powered by Spatially enabled DBMS

Main features

- Geometry stored and managed in the DBMS
- Facilities supported in different DBMS as ORACLE, PostgreSQL, MySQL, ORACLE Spatial, PostgreSQL/PostGIS, MS Databases through ADO

Interface with DBMS

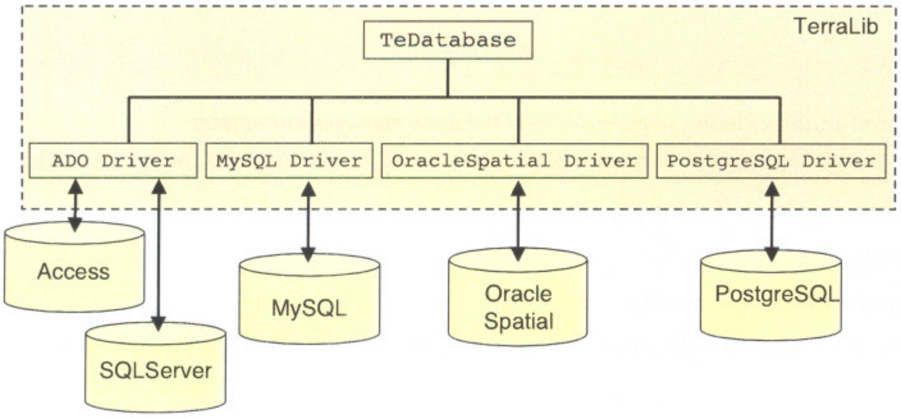


Image (raster) data needs

- Efficient storage and indexing mechanisms
- Decoding of the different image data formats
- Basic data manipulation functions
- Convenient ways of accessing the image data by algorithms

Two main aspects

1 *A DBMS data model*

- Tables schema
- Spatial indexing
- Support to compression

2 *A set of C^{++} classes to allow applications to deal with raster data*

- Efficiency and flexibility to access the data

DBMS data model

Defines, at a physical level, how to store raster data in an object-relational database

An ineffective approach

- Store each point of the image in a row of a table [x,y,z]

Another approach

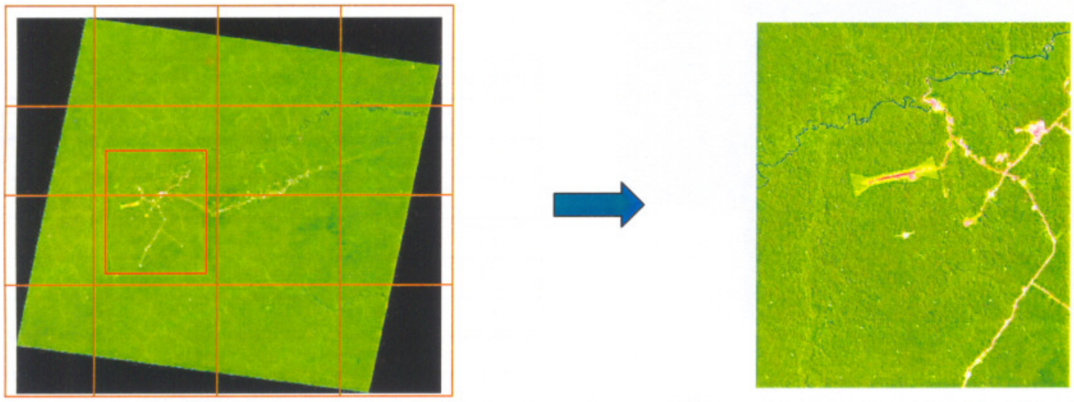
- The entire image is written to a BLOb and stored in a field of a table

A variation of the second approach was adopted:

- Tiles of image are writtento a BLOband stored in a field of a table

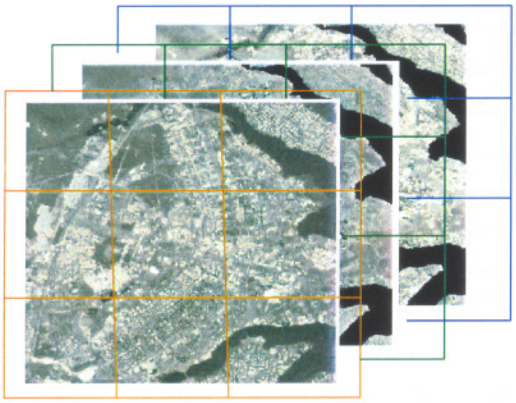
Tiling

- Specific parts of the image can be retrieved and processed independently
- User control over the size of the tiles
- Example: zooming operation



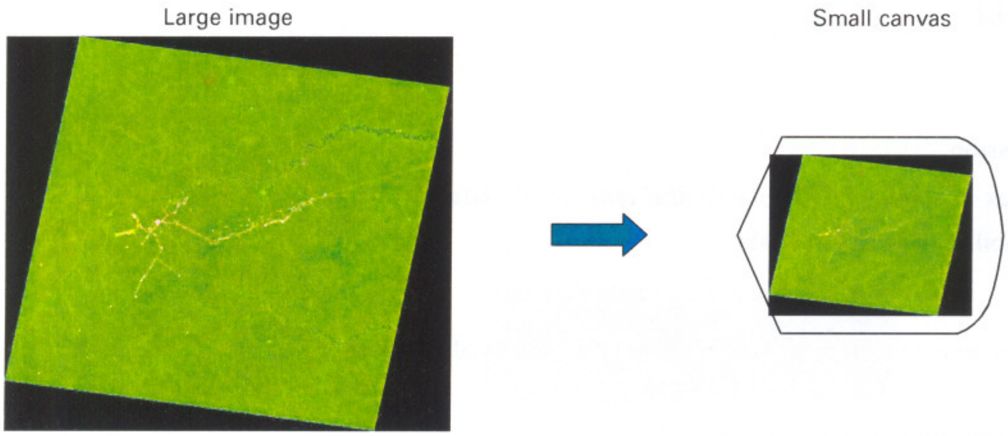
Tiling – DBMS data model

- Each raster data is stored in a table
- Each row stores a tile of a particular band



tile_id	band	BLOb
T1	1	...
T1	2	...
T1	3	...

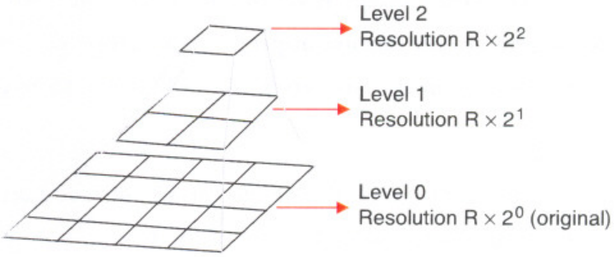
Multi-resolution

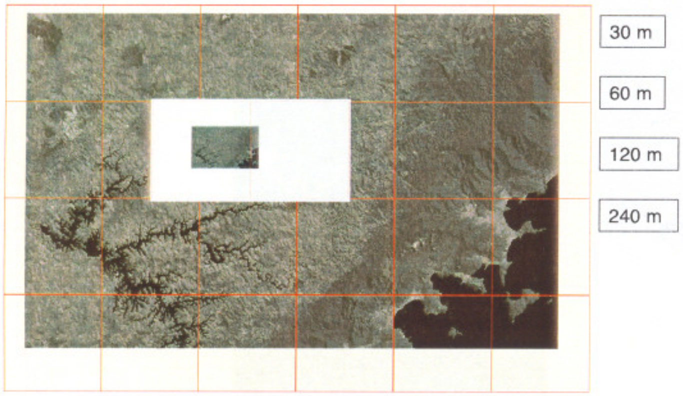


- Image is shown with a degraded resolution (Much of the information retrieved is not used)

Multi-resolution

- Lower ower resolution versions of the image are also stored in the database
- Application decides the best resolution level to be retrieved
- User control of the number of resolution levels





To store an image in a lower resolution less tiles are needed

tile_id	band	resolution_factor	blob
T1	1	0	–
T1	1	1	–
T1	2	0	–
T1	2	1	–
T1	3	0	–
T1	3	1	–

Each row of a Raster table contains information about the level of resolution of the tile.

Spatial indexing

- For each tile the coordinates of its bounding box are stored
- Using an SQL statement an application can select the tiles that intercept a given area in a given resolution level

tile_id	band	resolution_factor	Lower_x	Lower_y	upper_x	upper_y	blob
T1	1	0					–
T1	1	1					–
T1	2	0					–
T1	2	1					–
T1	3	0					–
T1	3	1					–

```
SELECT * FROM raster_table
WHERE NOT (lower_x > 10 OR upper_x < 20 OR lower_y > 10 OR upperY < 20 )
AND resolution_factor = 0
```

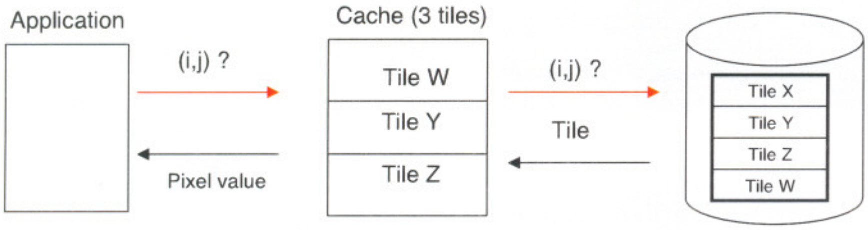
Accessing pixels individually

- Typical image processing algorithm
- To query the databasefor each pixel of an image can be costly– solution: keep a cache of tiles in memory

```
for i=0 to num rows
  for j=0 to num cols
  process Image(i,j)
```

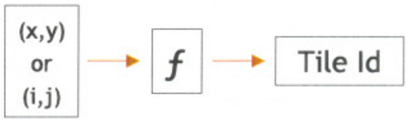
Virtual memory

- Optimize the access of pixels of an image
- Tiles in memoryhave the same identification of the database

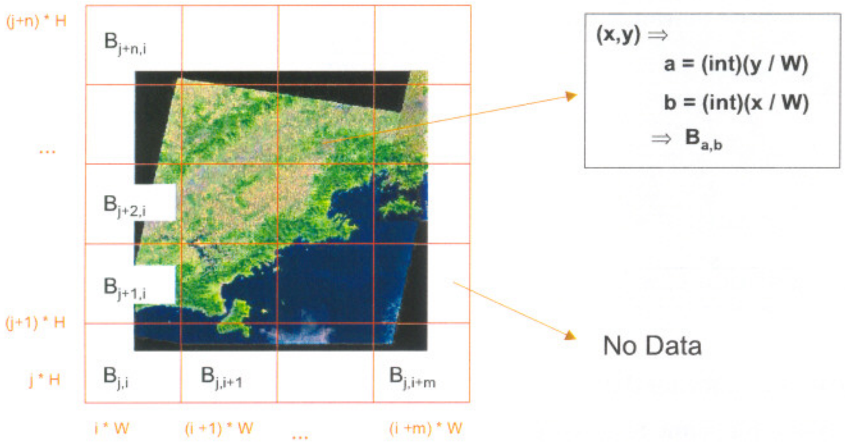


Tiles identification

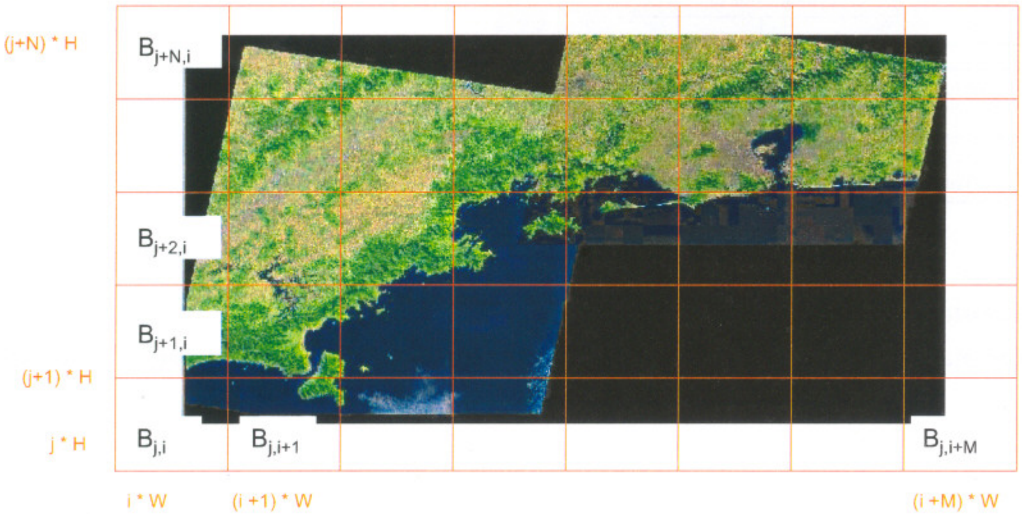
- A unique identification for each tile
- The function should return the same identification for every pixel that belongs to a tile
- The identification of tiles should remain consistent over mosaic operations



Tile size: **W** × **H** (in geographical units. i.e.: 1536m × 1536m)



Images can 'grow' and identification of the tiles remains consistent



Compression

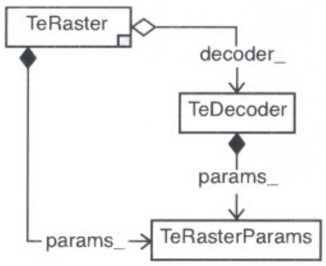
- Tiles can be compressed before being stored in the database
- Compression techniques: Zlib, JPEG or wavelets
- An image of de 1778 x 2804 pixels (4,985,512 pixels), 1 band, X and Y resolution of 25m, stored in tiles of 512 x 512 pixels:
- No compression -6,291,456 bytes
- ZLIB -3,746,080 bytes (~59.0%)
- JPEG 75% -814,694 bytes(~12.5%)

Metadata

- Database should also store metadata of the images in auxiliary tables

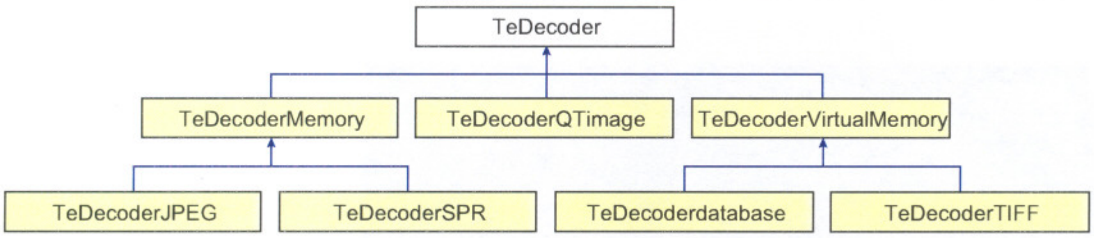
API Raster TerraLib

- TerraLib provides a set of C^{++} classes do deal with Raster Data
- Class TeRaster
  - Grid values are double
  - Methods getElement and setElement access elements of a Raster
- Class TeRasterParams
  - Information about a Raster representation
- Class TeDecoder
  - Strategy Pattern: allows access to different formats and storage aspects



Decoders

- Encapsulates the access to the elements of a Raster data
- Explicitly instantiated or defined from a file name for example
- Extensible



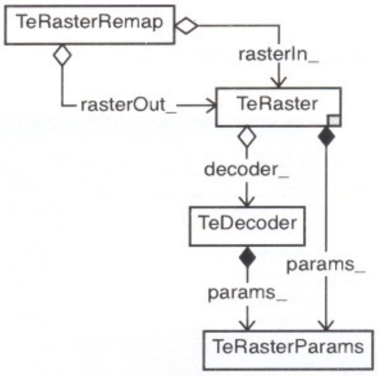
Manipulation

- Functions to import raster data into the database
- Class TeRasterRemap makes a copy of a raster data solving differences in projections, bounding boxes and resolutions

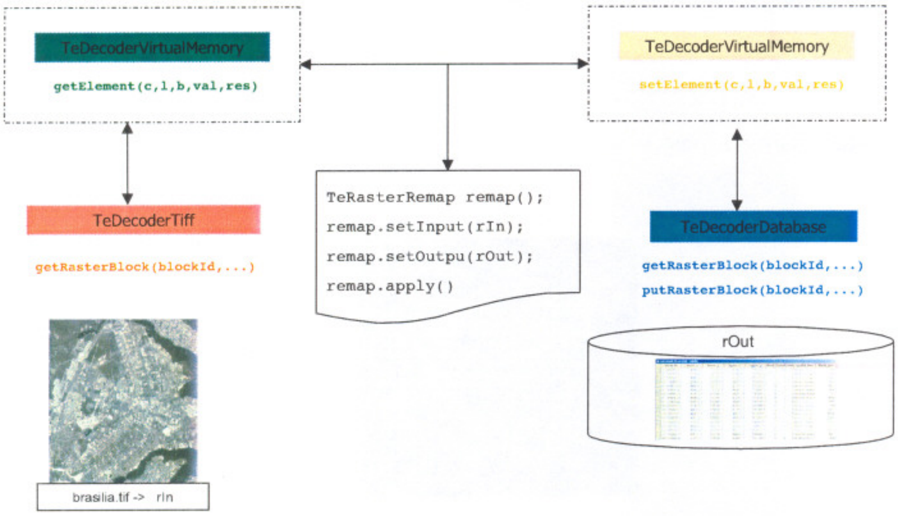
Manipulation

TeRasterRemap :

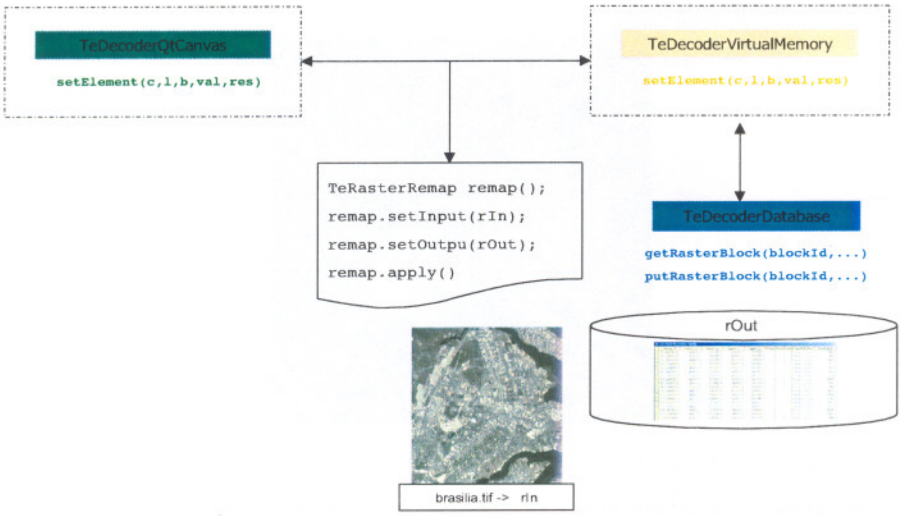
- Import from file to database
- Clipping
- Mosaic
- Visualization
- Reprojection



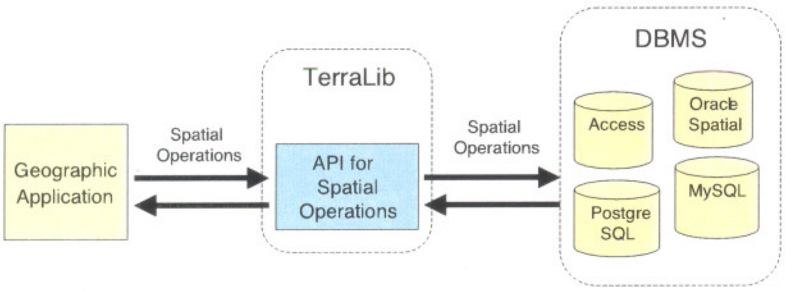
Importing



Visualisation

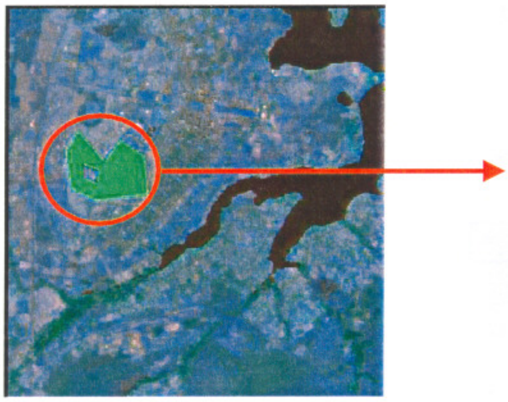


API for spatial operations on images



API – zonal operation

- Calculates statistics over a region or a zone of raster data

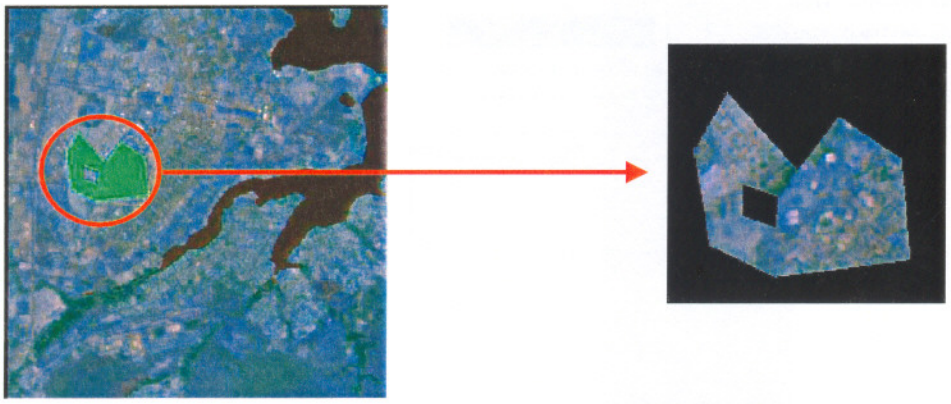


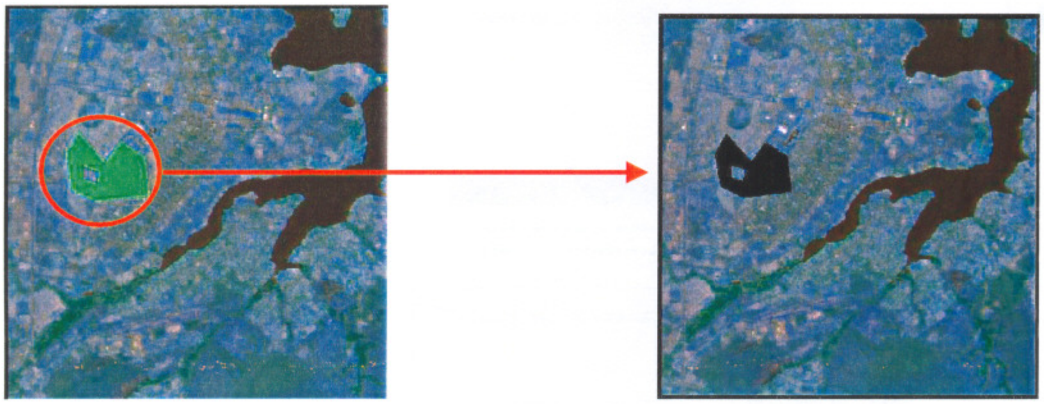
Estatísticas do raster

Estatísticas	Banda 0	Banda 1	Banda 2
soma	851164.000000	862173.000000	1091580.000000
valor máximo	205.000000	165.000000	206.000000
valor mínimo	30.000000	29.000000	28.000000
contagem	11365.000000	11365.000000	11365.000000
desvio padrão	18.811450	12.338327	24.338319
média	74.893445	75.862121	96.047514
variância	353.870652	152.234311	592.353748
assimetria	1.116281	1.030130	0.300146
curtose	5.929326	6.152706	3.588302
amplitude	175.000000	136.000000	178.000000
mediana	72.000000	74.000000	96.000000
coeficiennte de variação	25.117619	16.264147	25.339873
moda	70.000000	74.000000	97.000000

API – raster data

- Mask operation – clips a raster data using a mask

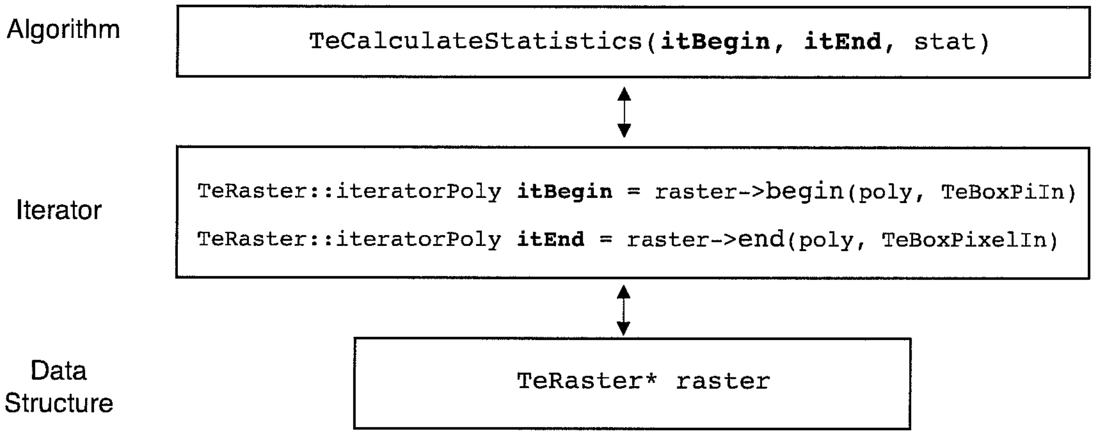




The use of iterators

- Mechanism to traverse a raster data only in a region inside or outsidea specific polygon
- Developed:
  - Iteratorconcept on TeRaster structure
  - IteratorPoly
  - Route strategies

Algorithm development made easy



Conclusions

- Tiling + Multi-resolution:
  - Efficient to visualization applications
- TeRaster provides an easy interface to algorithms
- TeDecoder provides flexibility to deal with different types of Raster data

The developed API:

- Provides spatial operations on a high level of abstraction for the developers of geographical applications
- Explores a new generation of object-relational DBMS that manage geographical data

Future work

- Implement other operations on raster data:
  - Mathematical operations
  - Reclassify
  - Slice
  - Weight
- Extend the API to support new spatial extensions
  - Spatial Extension in MySQL (release 4.1)
- Use future resources of spatial extensions to treat raster data (ex. Oracle Spacial)