AGILE 2018
21st Conference on Geo-Information Science

A Framework for the Management of Deformable Moving Objects

Authors

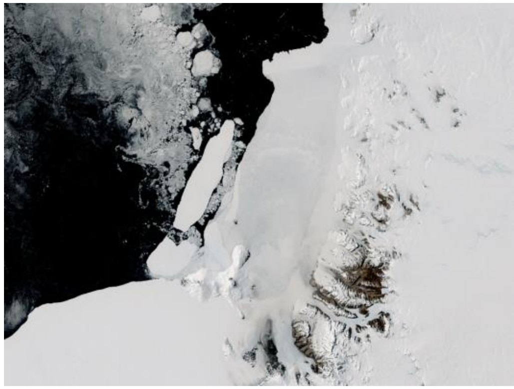
José Duarte, Paulo Dias, and José Moreira

Acknowledgments This work is partially funded by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UID/CEC/00127/2013.

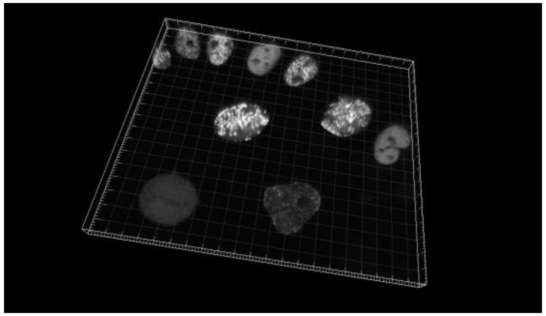
IEETA
Institute of Electronics and
Informatics Engineering of Aveiro

University of Aveiro

Context



Satellite images of the evolution of an iceberg (source (RossSea 2004))



Evolution of biological tissues (source (Codesolorzano Datasets))

Potential applications in several domains:

Environmental and climate sciences.

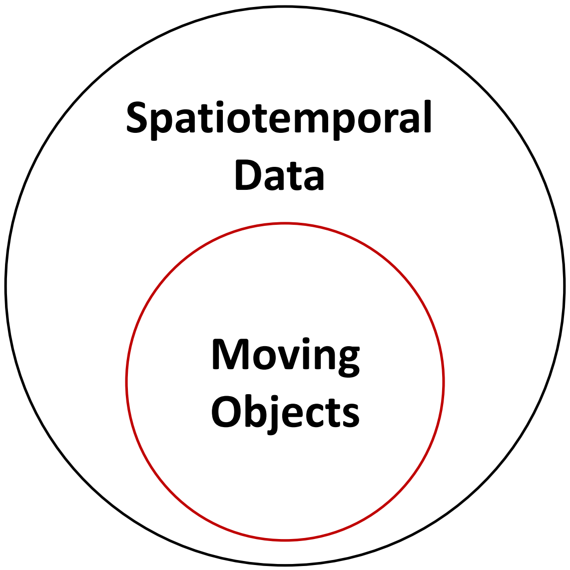
Agriculture.

Medical biology.

Spatiotemporal Data

Moving Objects:

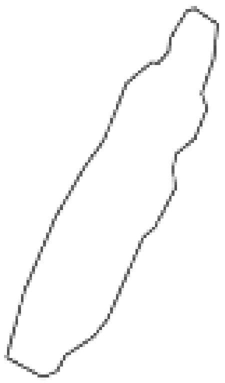
- Moving Points.
- Moving Regions.



Moving objects

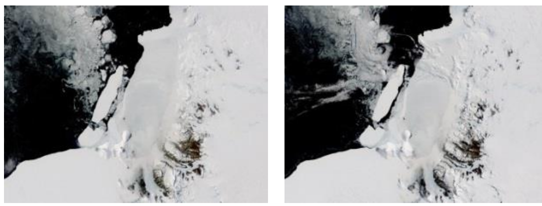
Efficient Management:

- Storage.
- Analysis.
- Manipulation.

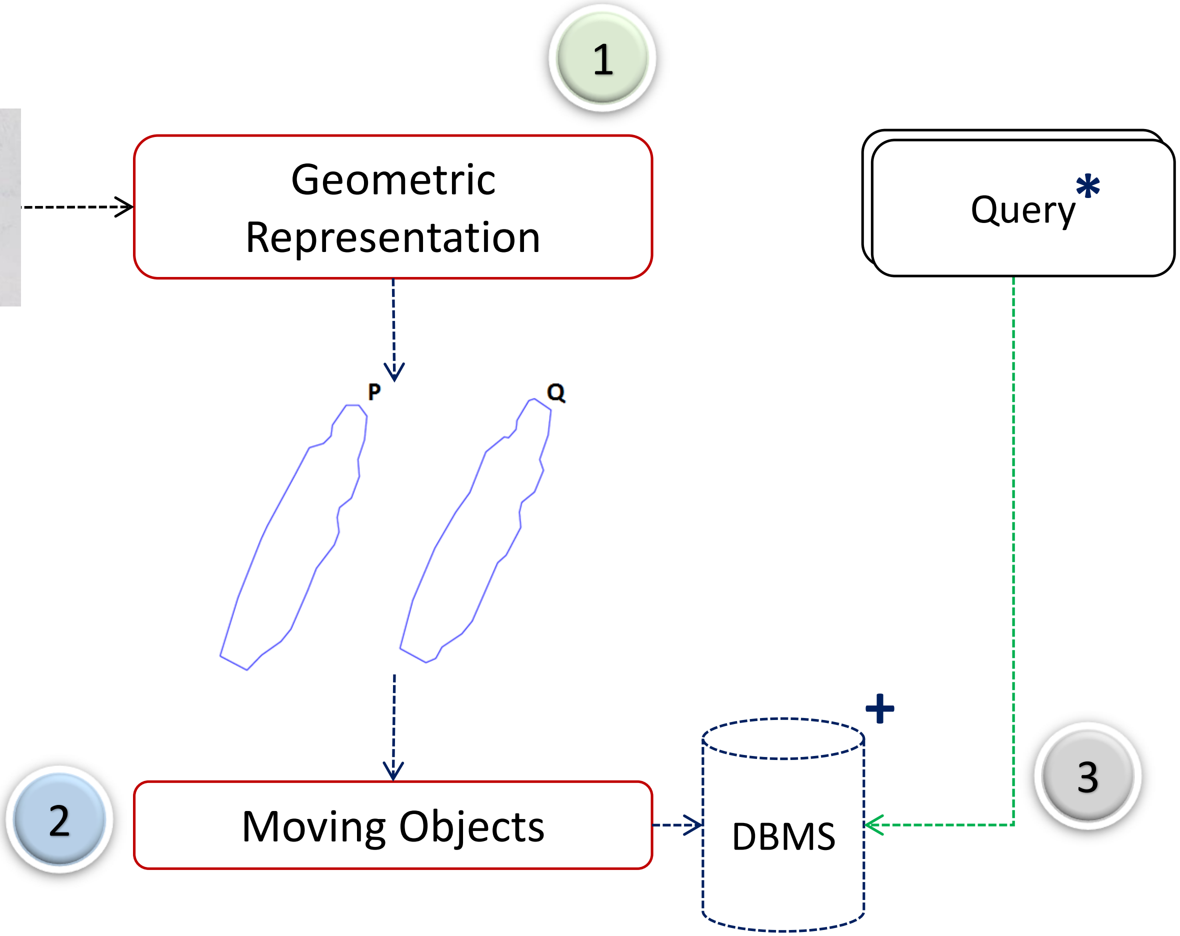


A deformable moving object

Spatiotemporal Databases



Two observations



Related Work I - IV

Most well-known data model and query language for representing and querying moving objects:

- Uses Abstract Data Types (ADTs).
- (Güting et al. 2000), (Forlizzi et al. 2000) and (Cotelo Lema et al. 2003).

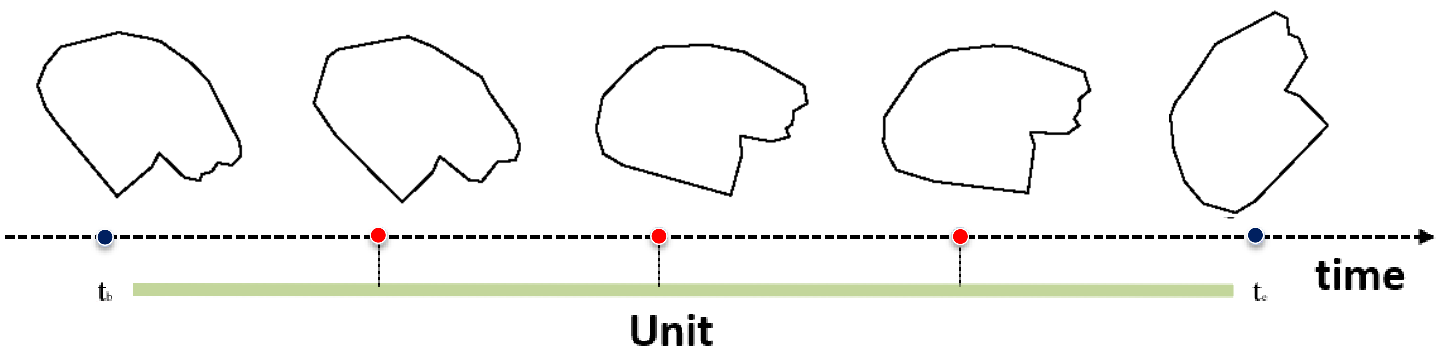
Several spatiotemporal data models were proposed in the literature:

- Secondo (Güting et al. 2010).
- Hermes (Pelekis et al. 2006).
- And others.

Representing Moving Objects II - IV

The *sliced representation* (Forlizzi et al. 2000):

- A moving object > an ordered collection of units.
- A unit > evolution of the moving object between two known consecutive observations.
- Evolution during a unit is given by a function ϕ.

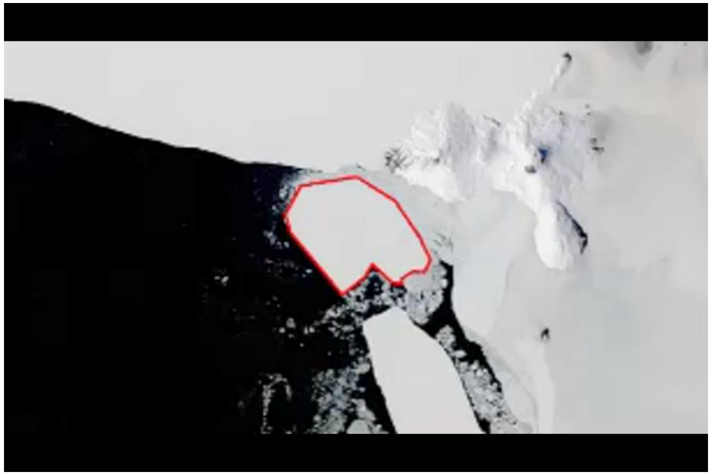


Observations of an iceberg evolving continuously over time

Representing Moving Objects III - IV

ϕ desirable properties:

- Provide a realistic approximation of the real evolution.
- Generate only valid geometries.
- Low storage and computation time.



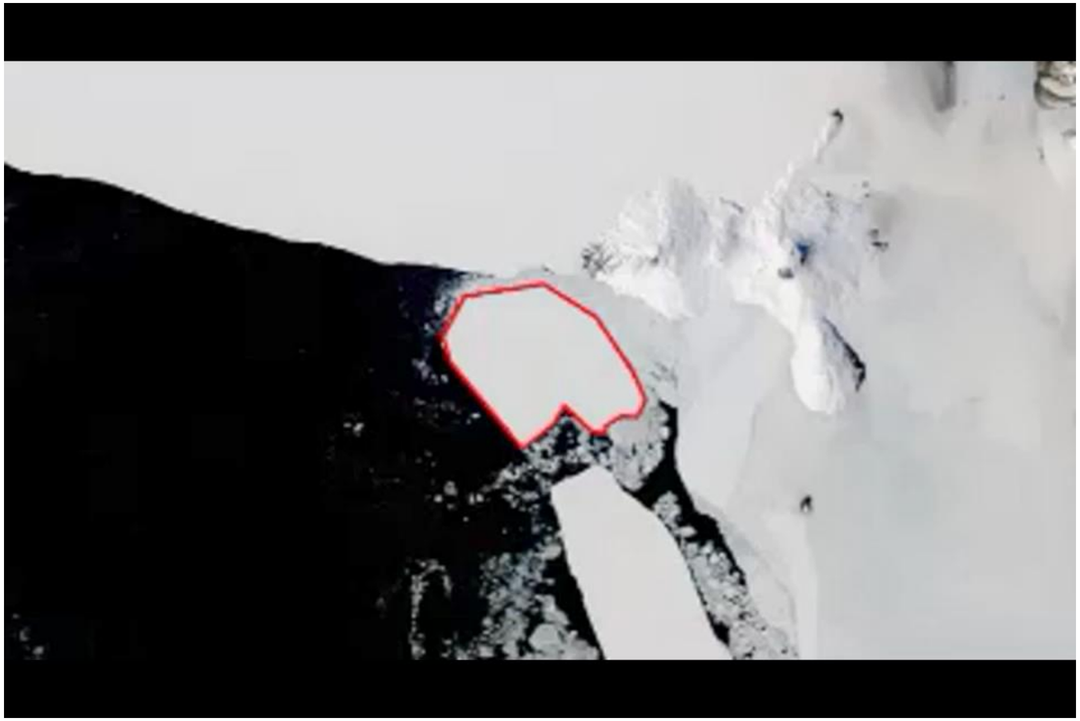
2D geometric representation of the evolution of an iceberg

University of Aveiro

Related Work IV - IV

Creating moving regions from observations, ϕ:

- (Tossebro & Güting 2001), (Mckenney & Webb 2010), (Mckennney & Frye 2015) and (Heinz & Güting 2016).

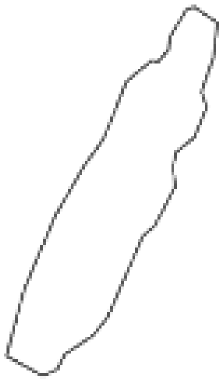


Representation of the evolution of an iceberg. Source: (Mesquita, 2013)

Morphing Techniques I-III

Using morphing techniques:

- Represent the continuous evolution between known observations.
- Compatible triangulation (Gotsman & Surazhsky 2004).
- Rigid interpolation (Alexa et al. 2000) and (Baxter et al. 2008).

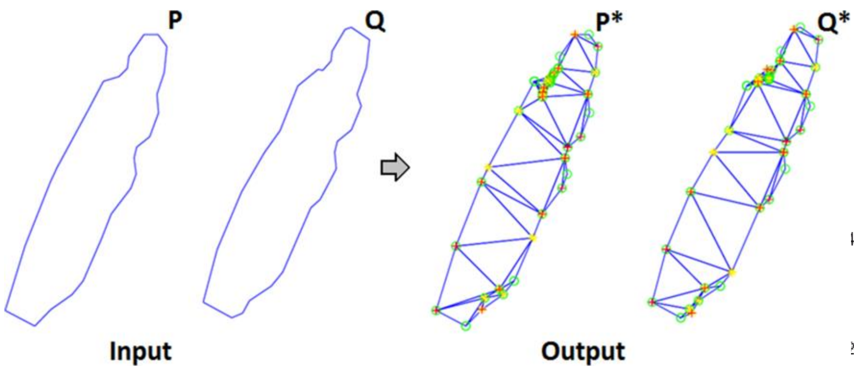


Continuous representation of the spatial transformation of a shape

Morphing Techniques II-III

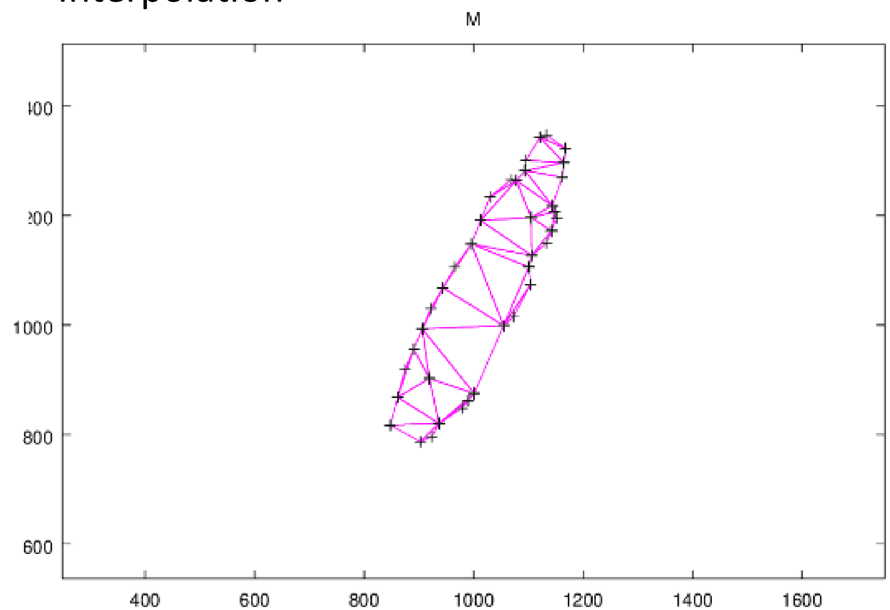
Two main steps:

1. Find a compatible triangulation between two known observations.
2. Compute the interpolation components.



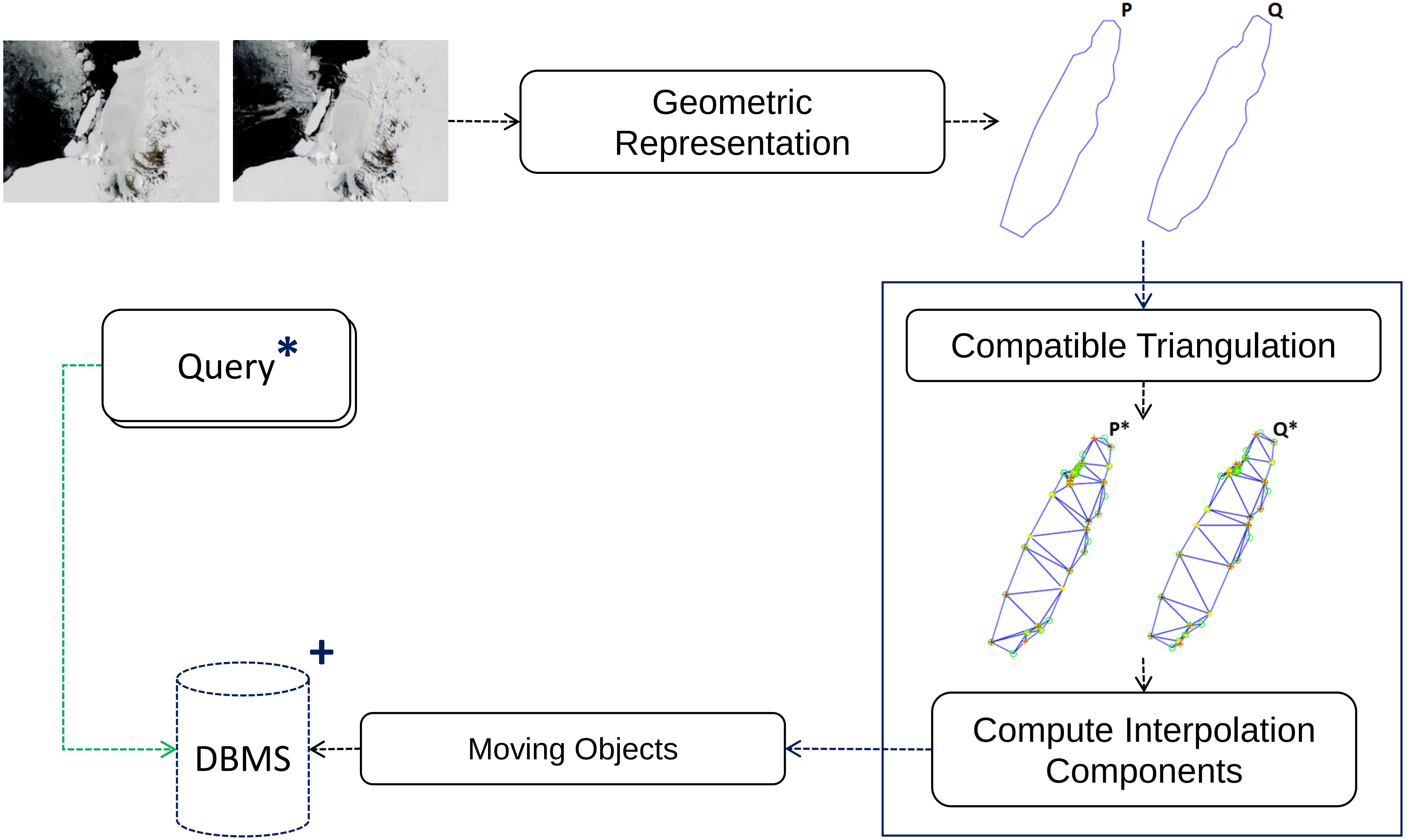
Compatible triangulation method
input (*left*) and output (*right*)

Interpolation



University of Aveiro

Morphing Techniques III-III



Using morphing techniques

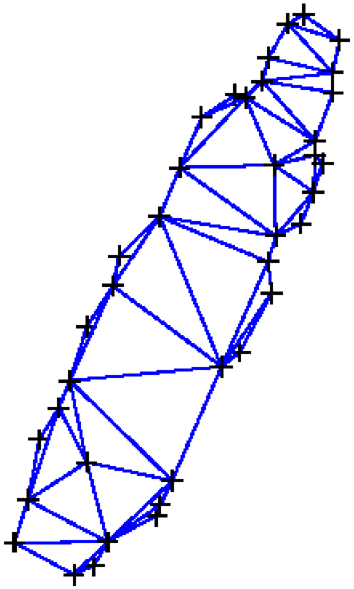
SPTMesh: A Framework for Moving Objects

C++ library for the analysis and manipulation of moving objects.

Independent of any client or application using it.

Uses morphing techniques (Gotsman & Surazhsky 2004), (Alexa et al. 2000) and (Baxter et al. 2008).

Introduces a new spatial type called mesh.



A mesh

SPTMesh: A Framework for Moving Objects

Has the architecture of the GEOS library as a reference.

Has the data types proposed in (Güting et al. 2000) and (Forlizzi et al. 2000) as a reference.

Does not consider lines, collections and regions with holes.

SPTMesh: Data Structures

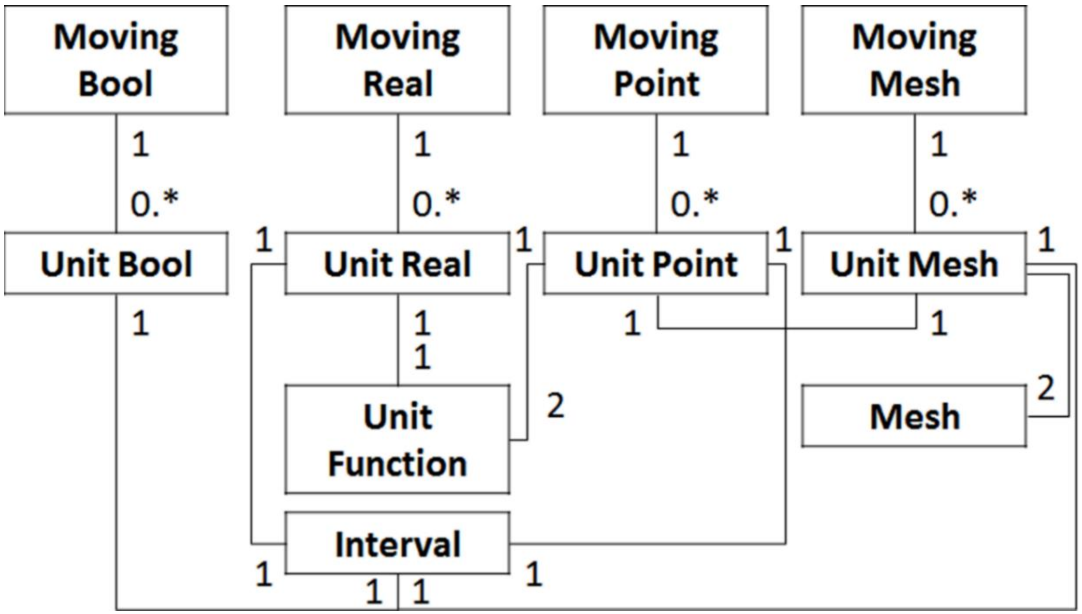
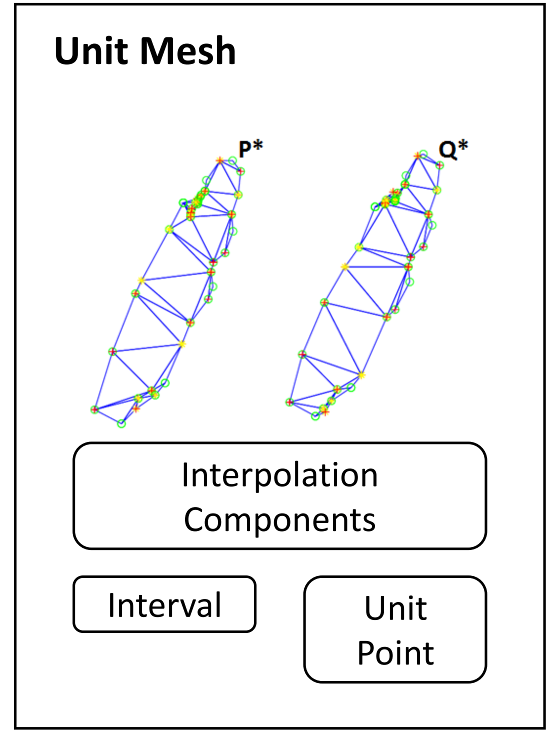
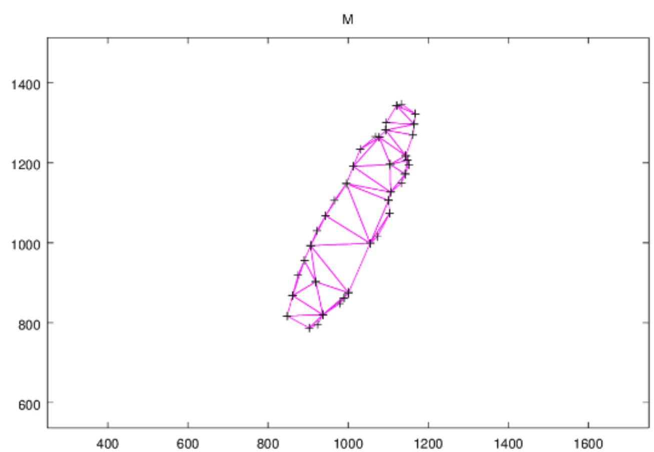
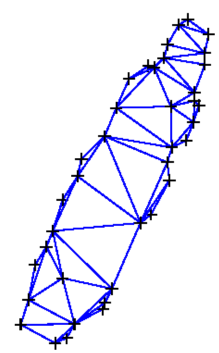


Fig. 11 The SPTMesh type system



A Unit Mesh





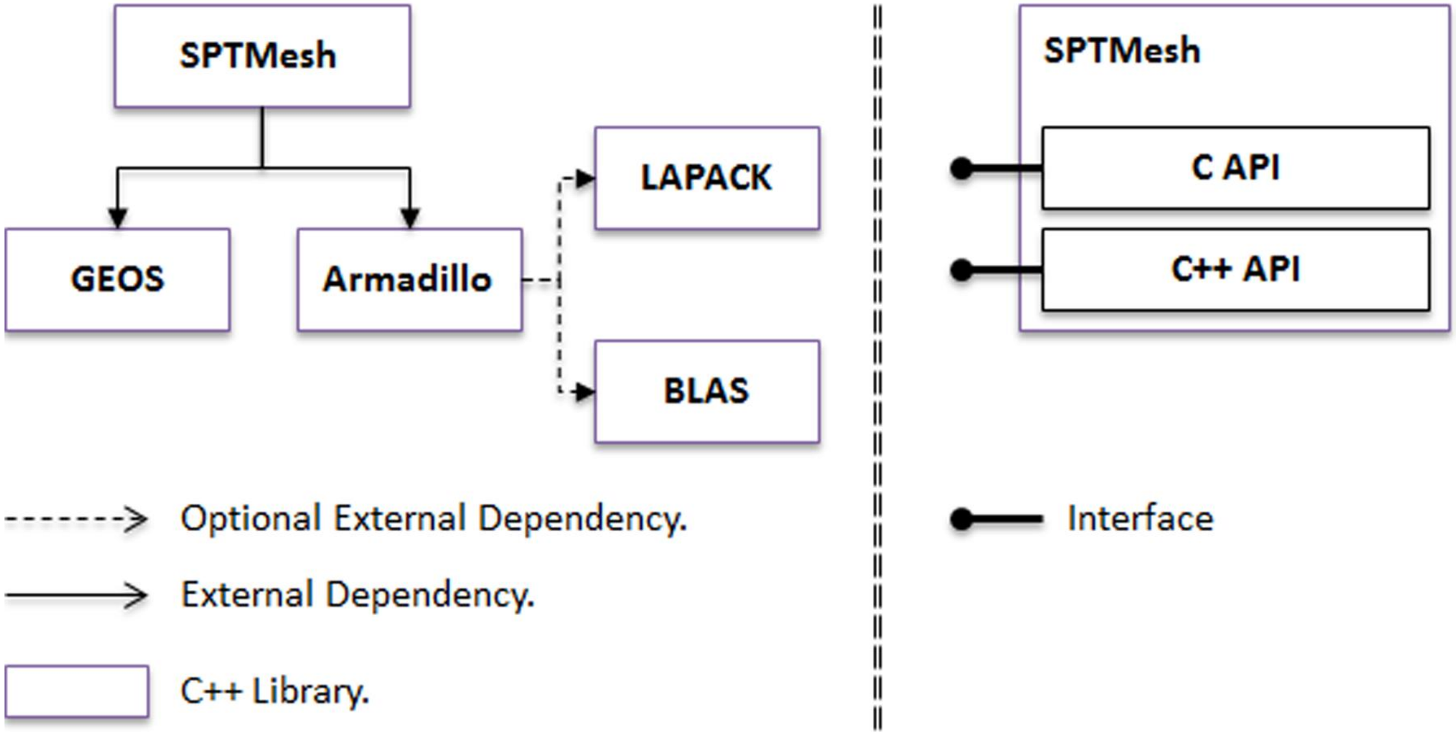
SPTMesh: Operations on Moving Types

Provides a subset of the spatiotemporal operations proposed in (Güting et al. 2000).

Class of Operation	**Operation**
Predicates	equals, intersects*
Set Operations	intersection*
Numeric	area
Projection to Domain and Range	deftime
Interaction with Domain and Range	atinstant, atperiod, present
Constructors	unit, moving

SPTMesh Operations on MOVING types

SPTMesh: Architecture



SPTMesh architecture (*left*) and APIs (*right*)

C library.

Uses SPTMesh to analyze and manipulate moving objects.

Has the PostGIS architecture as a reference.

Allows the moving objects provided by SPTMesh to be:

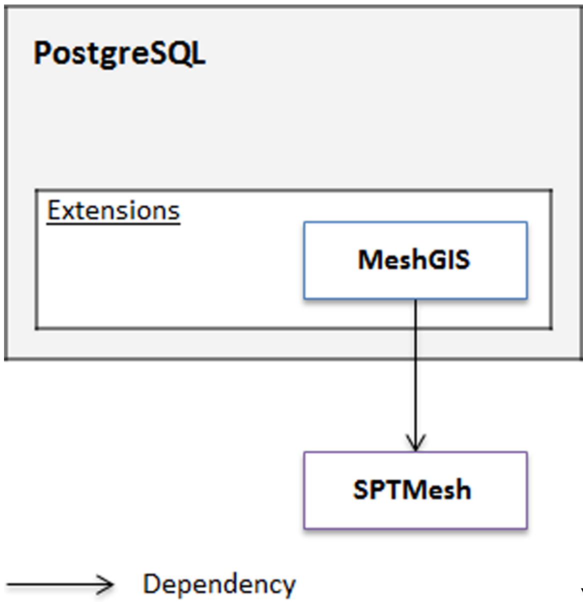
- Stored in a PostgreSQL database.
- And manipulated using SQL.

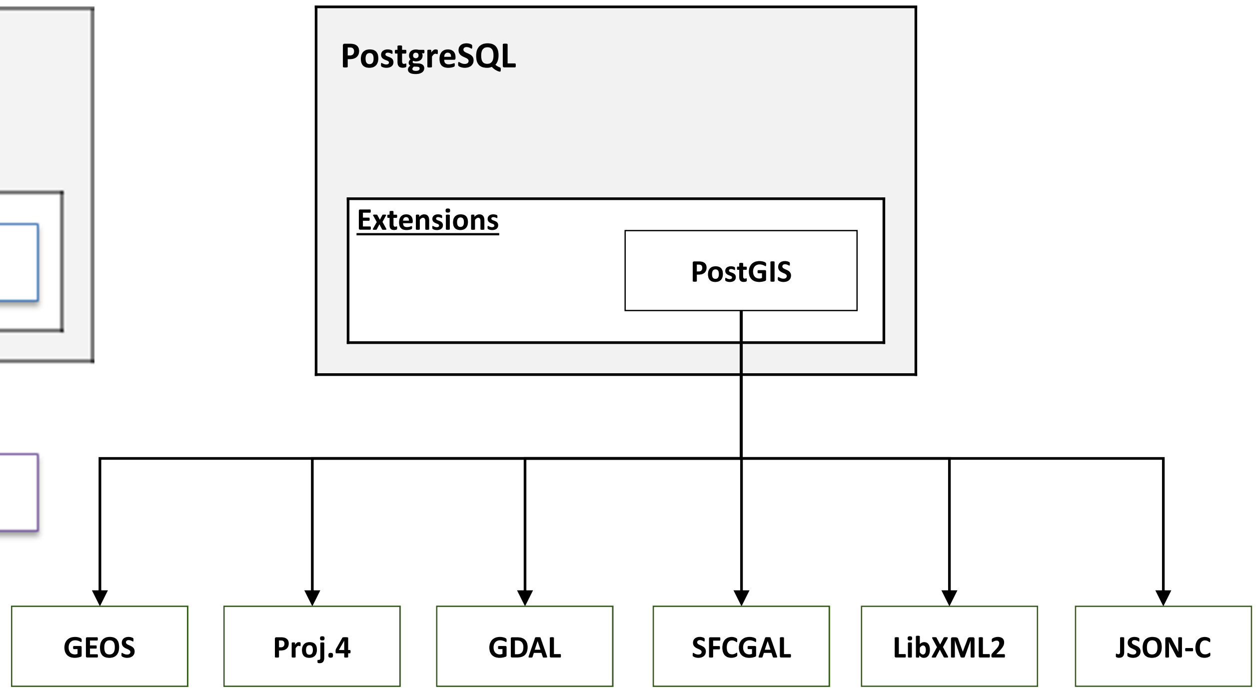
MeshGIS: Data Structures

Data Structure	Description
ArrayOfX	A generic array to hold units or other elements.
UnitFunction	A SPTMesh UnitFunction.
UnitInterval	A SPTMesh Interval.
UnitBool	A SPTMesh UnitBool.
UnitReal	A SPTMesh UnitReal.
UnitPoint	A SPTMesh UnitPoint.
UnitMesh	A SPTMesh UnitMesh.
SerializedPostgreSQLObject	A PostgreSQL data type for variable size user-defined data types.
SerializedMovingObject	Abstract data type that represents any type of moving object.
SerializedMovingX	Represents the SPTMesh MovingBool, MovingReal and MovingPoint types.
SerializedMovingMesh	A SPTMesh MovingMesh.

MeshGIS data structures used to represent SPTMesh types

MeshGIS: Architecture

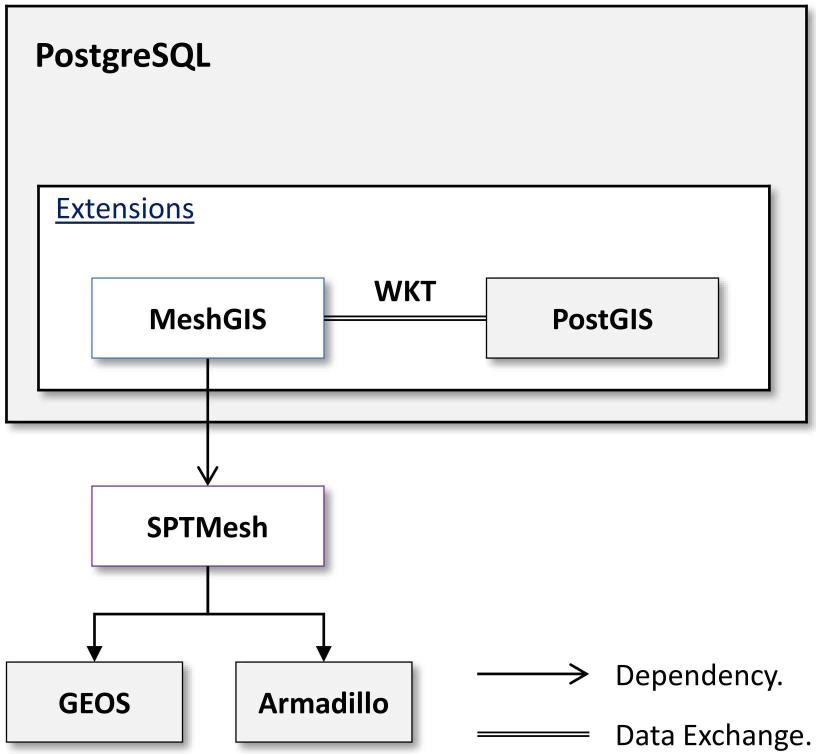




MeshGIS architecture

PostGIS architecture

Overall Architecture of the System

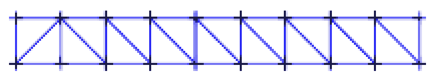


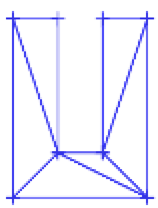
Overall architecture

Experimental Results I-V

Tests using synthetic data.

Coil interpolation test





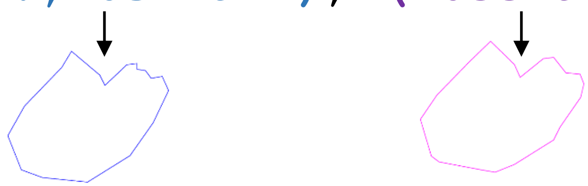
180º rotation test

Experimental Results II-V

Tests using real data obtained from (*RossSea Subsets*, 2004).

```
CREATE TABLE db.icebergs(
  id    integer,
  name  varchar(50),
  mobj  movingmesh
)
```

```
INSERT INTO db.icebergs(id, name, mobj) VALUES(2, ‘ice 2’, ST_MovingMesh_FromSTWKT('MOVINGMESH((1000 2000, (1052 987,...,1034 941), (1055 999,...,1001 875)))'));
```



```
UPDATE db.icebergs SET mobj = ST_Add_UnitMesh((SELECT mobj FROM db.icebergs WHERE id=2), ‘UNITMESH(2000 3000, (1055 999,..                30 942,…,996 896))',false)
WHERE id=2;
```

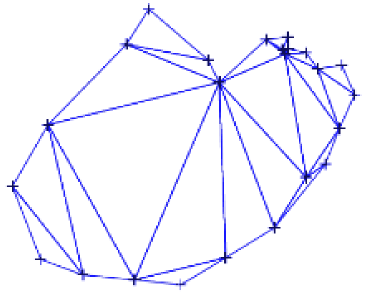
Experimental Results III-V

```
SELECT * FROM db.icebergs;
```

Id	Name	Mobj
2	ice 2	MOVINGMESH((1000 2000, (1052 987, 1090 1037, …,1034 941), (1055 999, …,1001 875)), …)
1	ice 1	MOVINGMESH EMPTY

Select results

```
SELECT ST_Get_AtPeriod(mobj,'PERIOD(1100 4500)') FROM
db.icebergs WHERE id=2;
```

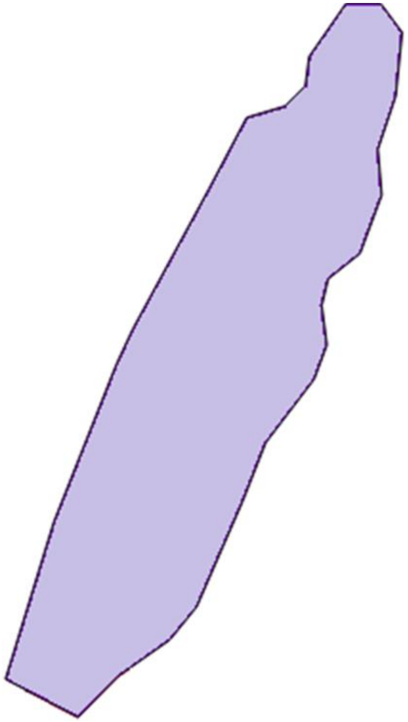


Iceberg 2 evolving during the period (1100 4500)

Experimental Results IV-V

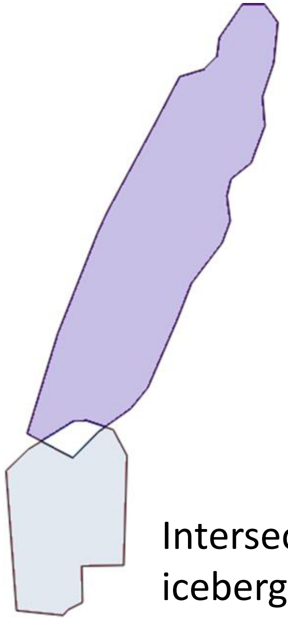
University of Aveiro

```
SELECT ST_Get_AtInstant(mobj, 1500) FROM db.icebergs WHERE id=1;
```



Iceberg 1 at instant 1500

```
SELECT ST_Intersection((SELECT mobj FROM db.icebergs WHERE id=3), (SELECT mobj FROM db.icebergs WHERE id=4), 1000);
```



Intersection of two icebergs at instant 1000

Experimental Results V-V

```
SELECT ST_Area(ST_GeomFromText(ST_Intersection((SELECT mobj
FROM db.icebergs WHERE id=3), (SELECT mobj FROM db.icebergs
WHERE id=4),1000)));
```

Intersection Area (abstract units)
1815.20

Using PostGIS to get the area of the intersection of two moving regions at instant 1000

Conclusions

Main goal:

- Propose and implement a framework (data model).
  - To manage moving objects.
  - Uses morphing techniques.
  - Is client-independent and can be used by a DBMS.

This is not a complete implementation.

Future Work

Implement a larger set of spatiotemporal operations.

Extend SPTMesh to work with regions with holes and collections.

Test SPTMesh using larger and more diverse datasets.

Evaluate and Compare the quality of the evolution obtained when using morphing techniques and when using the methods proposed in the literature.

References

Alexa, M., Cohen-or, D. & Levin, D., 2000. As-Rigid-As-Possible Shape Interpolation. In *SIGGRAPH '00 Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. pp. 157–164.

Baxter, W., Barla, P. & Anjyo, K., 2008. Rigid shape interpolation using normal equations. In *NPAR '08 Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*. pp. 59–64. Available at: http://dl.acm.org/citation.cfm?id=1377993.

Cotelo Lema, J. et al., 2003. Algorithms for Moving Objects Databases. *The Computer Journal*, 46(6), pp.680–712.

Forlizzi, L. *et al.* (2000) 'A Data Model and Data Structures for Moving Objects Databases', in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 319–330. doi: http://doi.acm.org/10.1145/342009.335426.

Gotsman, C. & Surazhsky, V., 2004. High quality compatible triangulations. *Engineering with Computers*, 20(2), pp.147–156.

Güting, R. H. *et al.* (2000) 'A Foundation for Representing and Querying Moving Objects', *ACM Trans. Database Systems*, 25(1), pp. 1–42. doi: 10.1145/352958.352963.

Güting, R.H., Behr, T. & Düntgen, C., 2010. SECONDO : A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 33(2), pp.56–63.

Heinz, F. & Güting, R.H., 2016. Robust high-quality interpolation of regions to moving regions. *GeoInformatica*, 20(3), pp.385–413. Available at: http://link.springer.com/10.1007/s10707-015-0240-z.

Mckenney, M. & Webb, J., 2010. Extracting Moving Regions from Spatial Data. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. San Jose, California, pp. 438–441. Available at: http://doi.acm.org/10.1145/1869790.1869856.

Mckennney, M. & Frye, R., 2015. Generating Moving Regions from Snapshots of Complex Regions. *ACM Trans. Spatial Algorithms Systems*, 1(1), pp.1–30. Available at: http://doi.acm.org/10.1145/2774220.

Pelekis, N. et al., 2006. Hermes - A Framework for Location-Based Data Managment. In *EDBT'06 Proceedings of the 10th international conference on Advances in Database Technology*. Munich, Germany, pp. 1130–1134. Available at: https://doi.org/10.1007/11687238_75.

RossSea Subsets (2004). Available at: http://rapidfire.sci.gsfc.nasa.gov/imagery/subsets/?project=antarctica&subset=RossSea&date=11/15/20

Sanderson, C. and Curtin, R. (2016) 'Armadillo: a template-based C ++ library for linear algebra', *The Journal of Open Source Software*, 1, p. 26.

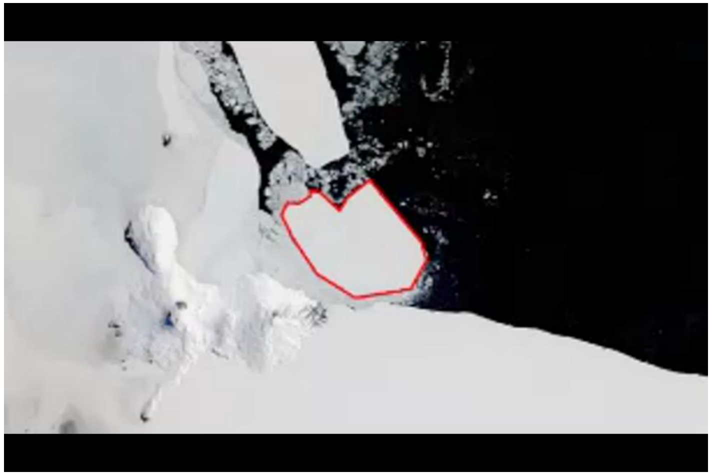
Mesquita, P. (2013) '*Morphing Techniques For Representation of Geographical Moving Objects',* University of Aveiro.

Tøssebro, E. & Güting, R., 2001. Creating Representations for Continuously Moving Regions from Observations. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*. Springer-Verlag Berlin Heidelberg, pp. 321–344. Available at: https://doi.org/10.1007/3-540-47724-1_17.

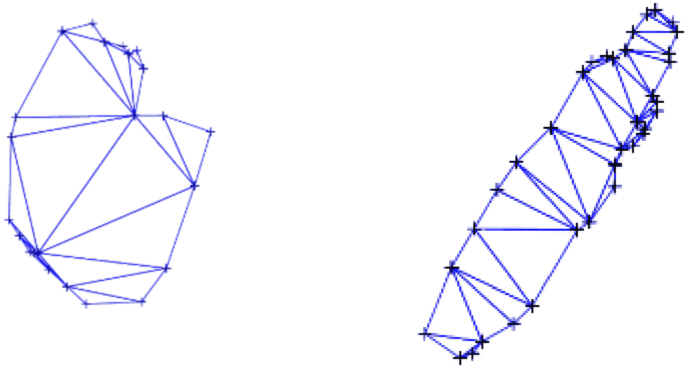
Codesolorzano Datasets. http://www.codesolorzano.com/Challenges/CTC/Datasets.html

RossSea Subsets. http://rapidfire.sci.gsfc.nasa.gov/imagery/subsets/?project=antarctica&subset=RossSea&date=11/15/2004 (Set 20, 2016).

University of Aveiro



2D geometric representation of the evolution of an iceberg



Two moving regions

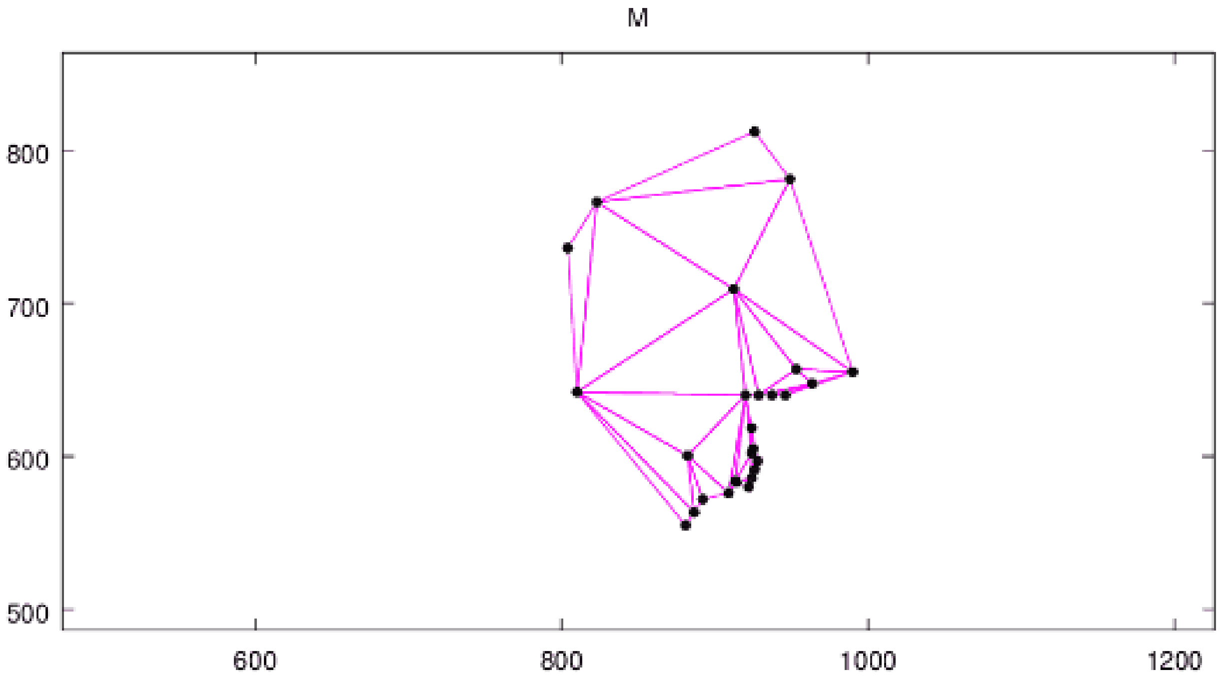
Project FCT-032636: Modeling, querying and interactive visualization of spatiotemporal data (MoST)

June 2018 – November 2020

- **New datasets (case studies) are welcome!**
- Fellowships to be opened in July, 2018: 1 candidate with PhD + 1 candidate with MSc (Field: Computer Science | Location: University of Aveiro, Portugal)

Presenter: José Duarte, hfduarte@ua.pt
Co-authors: paulo.dias@ua.pt, jose.moreira@ua.pt

Invalid Geometry



Invalid geometry during interpolation

SPTMesh: Unit Types and Continuity

Two *UnitMesh* objects are continuous if:

$$distance(m_i, m_j, t) \leq \xi_p \qquad (1)$$

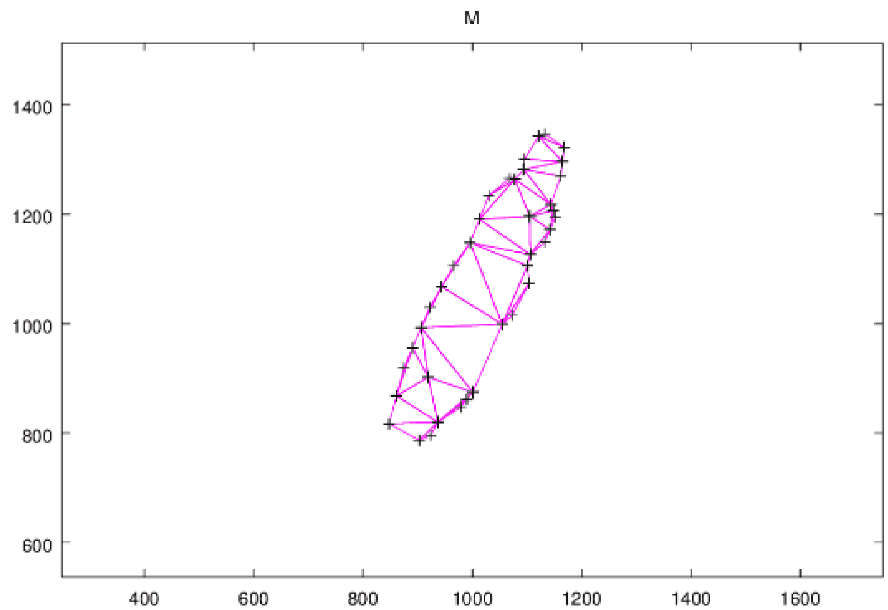
$$\frac{area(m_i, m_j, \cap, t)}{area(m_i, m_j, \cup, t)} \leq \delta_s \qquad (2)$$

where m_i, $m_j \in UnitMesh$, $t \in instant$.

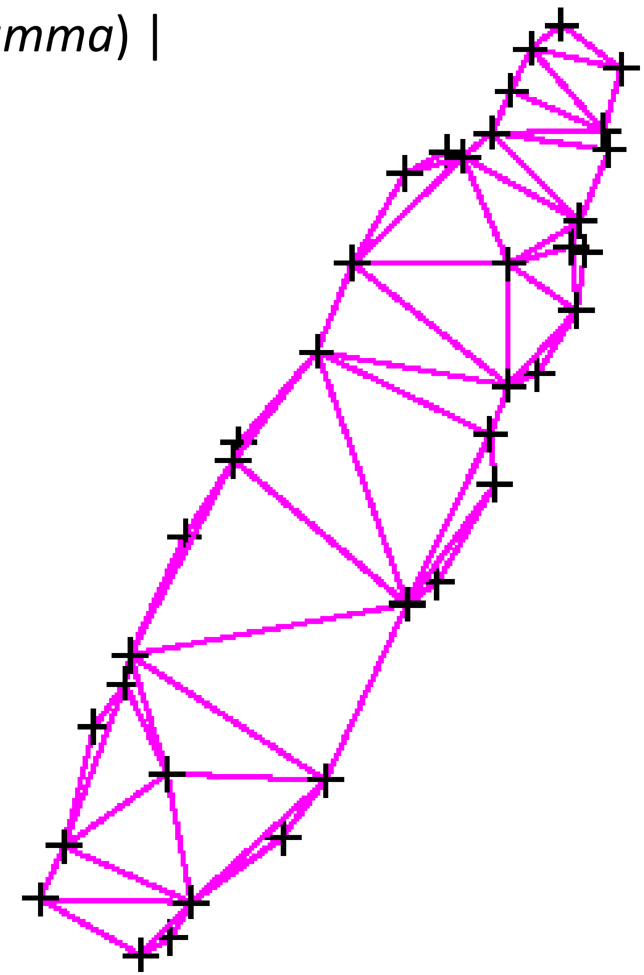
University of Aveiro

SPTMesh: Mesh and Unit Mesh Data Types

UnitMesh = {(*I*, t_b, t_e, *c*, *P**, *Q**, *pStar*, *rScale*, *rGamma*) |
I ∈ *Interval*,
t_b, t_e ∈ *long*,
c ∈ *UnitPoint*,
*P**, *Q** ∈ *Mesh*,
pStar, *rScale* ∈ *mat*,
rGamma ∈ *vector<double>*}



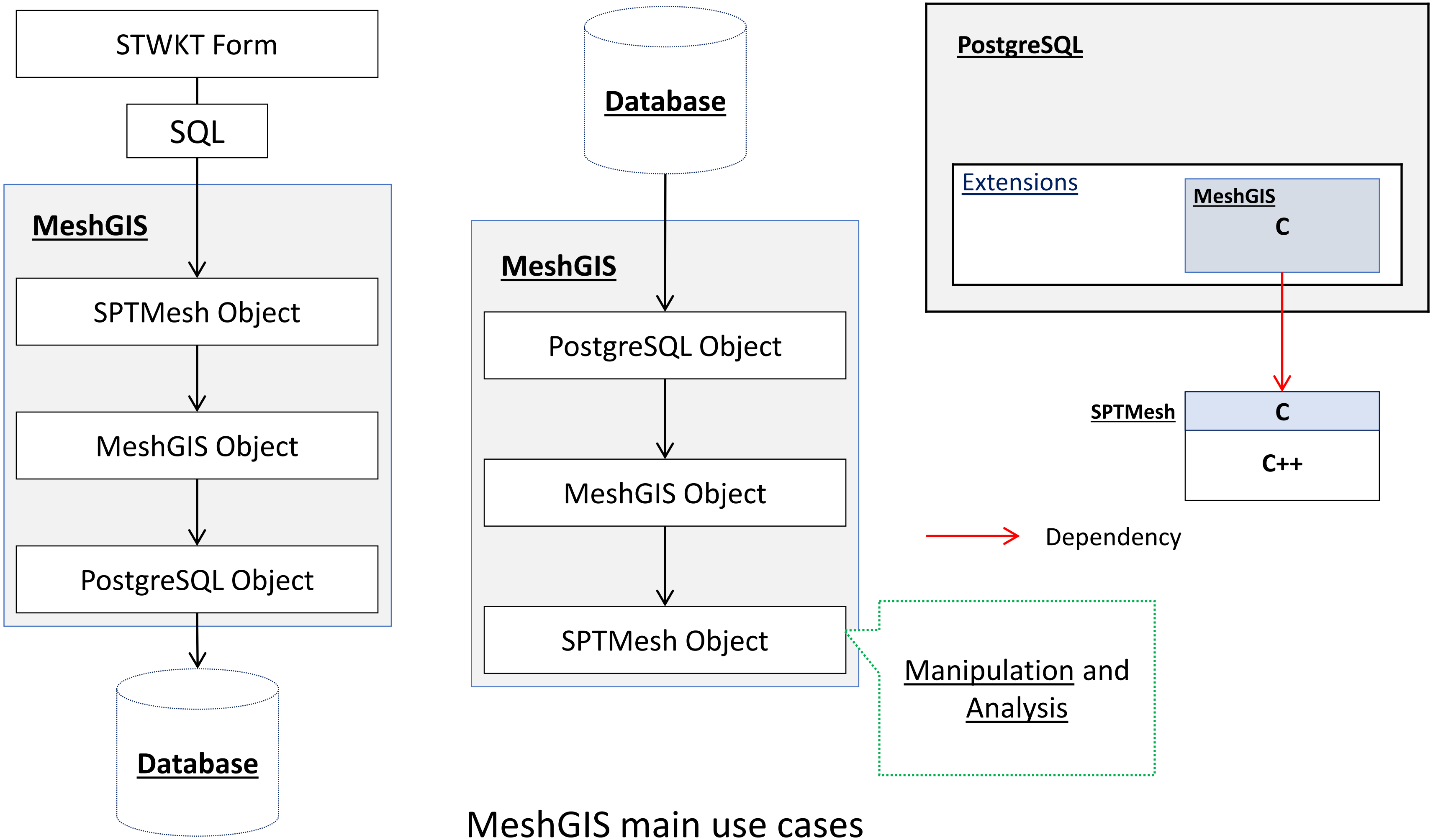
Evolution during a unit



A mesh

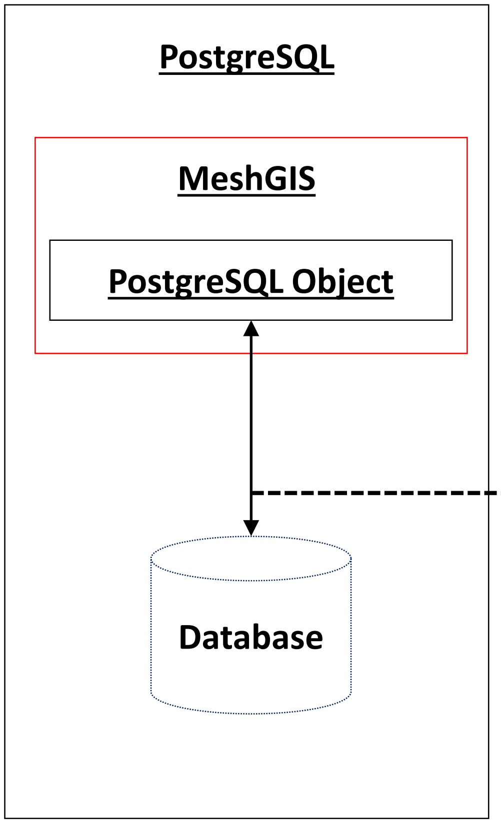
University of Aveiro

MeshGIS: Main Use Cases



MeshGIS main use cases

Persistence of Moving Objects in PostgreSQL




```
/**
 * GSERIALIZED
 * PostgreSQL data type for variable size user defined data types.
 */
typedef struct
{
    uint32_t size;      For PgSQL use only, use VAR* macros to manipulate.
    uint8_t srid[3];    24 bits of SRID.
    uint8_t flags;      HasZ, HasM, HasBBox, IsGeodetic, IsReadOnly.
    uint8_t data[1];    Data.
} SerializedPostgreSQLObject;
```

Persistence of moving objects in PostgreSQL