intecs

the Brainware company

SELEX

Sistemi Integrati

FOSS4G 2010
Barcelona
Free and Open Source Software
for Geospatial CONFERENCE
SEP 6th - 9th

pgRouting extension for courses calculation in a VTMS

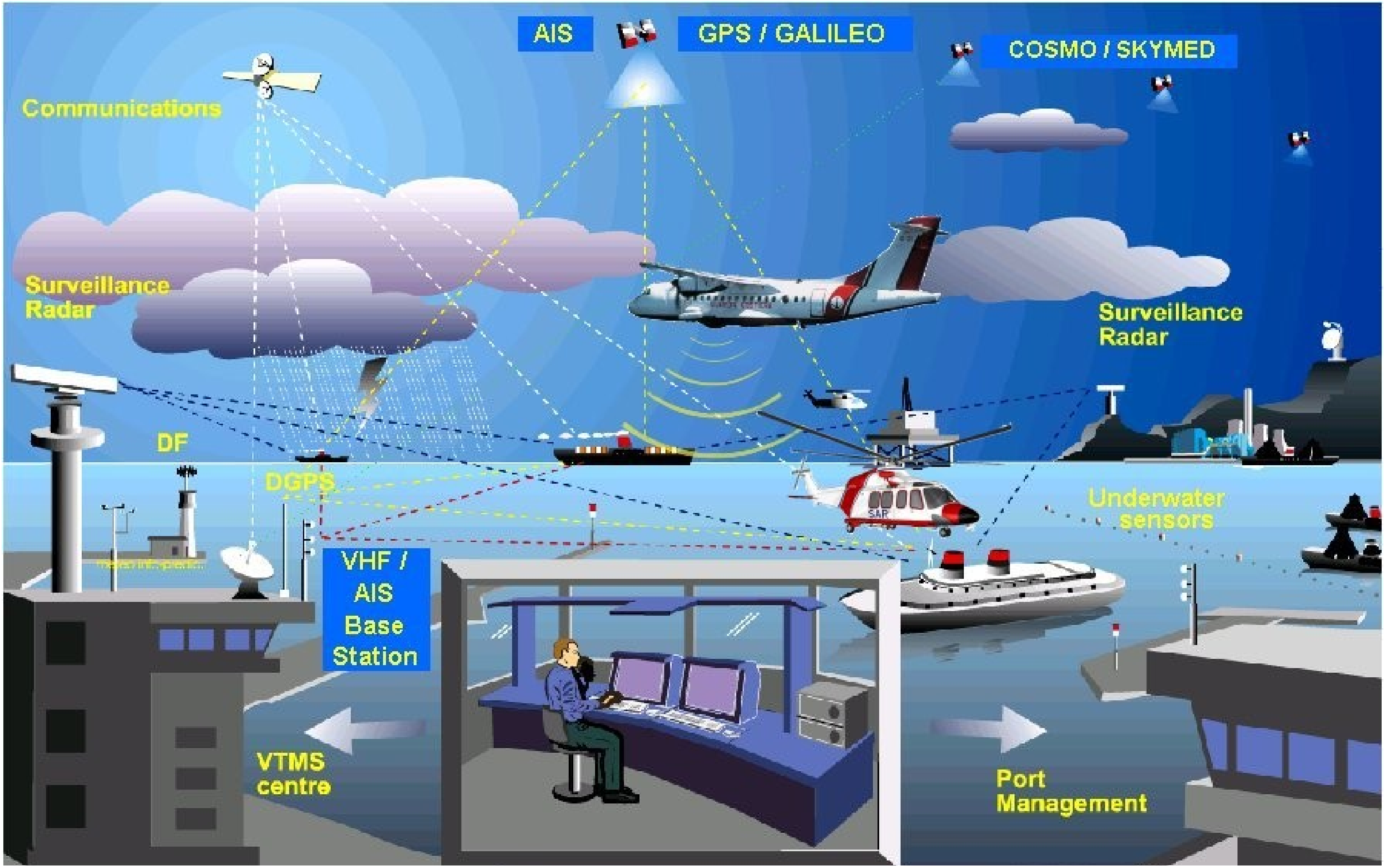
AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

VTMS
(Vessel Traffic Management System)



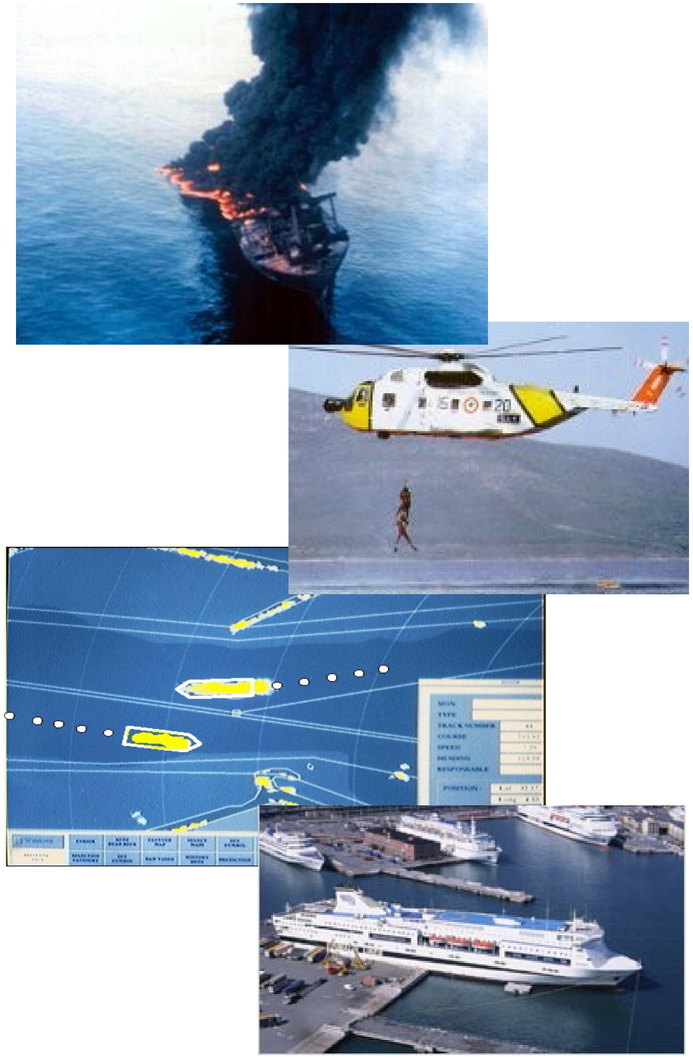
VTMS Services

Safe and laws enforcement

- Ships and ports safety monitoring
- Police actions support
- Fishing activities control
- Dangerous loads monitoring

Decision support to human life safeguard at sea

- Accidents prevention
- Environment protection
- Search and Rescue (SAR) support
- Emergency missions planning and monitoring



AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

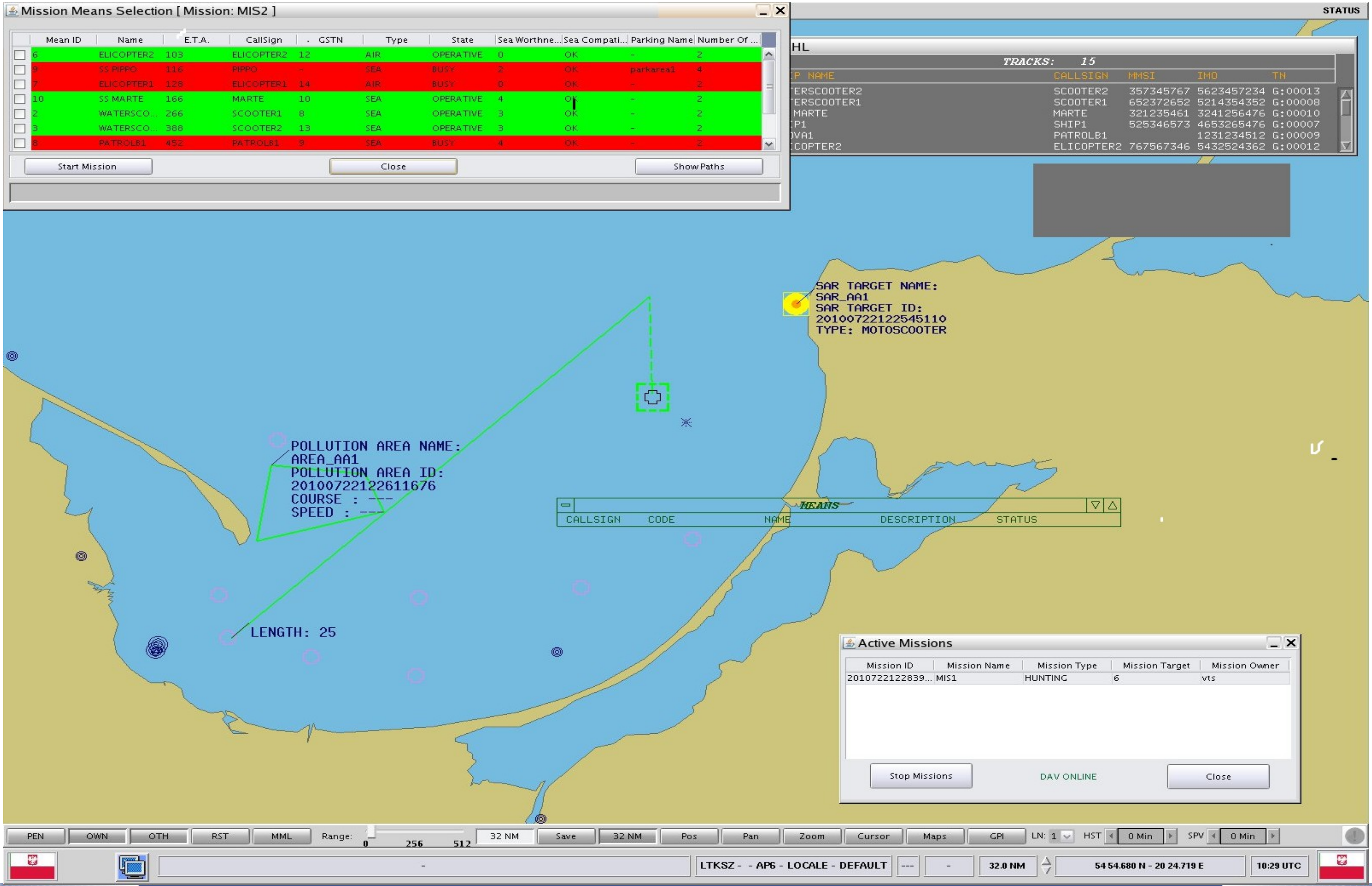
Emergency Missions

Hunting

Anti-pollution

SAR

Mission Planning



Mission Means Selection [Mission: MIS2]

☐	Mean ID	Name	E.T.A.	CallSign	GSTN	Type	State	Sea Worthne...	Sea Compati...	Parking Name	Number Of ...
☐	6	ELICOPTER2	103	ELICOPTER2	12	AIR	OPERATIVE	0	OK	-	2
☐	9	SS PIPPO	116	PIPPO	-	SEA	BUSY	2	OK	parkarea1	4
☐	7	ELICOPTER1	128	ELICOPTER1	14	AIR	BUSY	0	OK	-	2
☐	10	SS MARTE	166	MARTE	10	SEA	OPERATIVE	4	OK	-	2
☐	2	WATERSCO...	266	SCOOTER1	8	SEA	OPERATIVE	3	OK	-	2
☐	3	WATERSCO...	388	SCOOTER2	13	SEA	OPERATIVE	3	OK	-	2
☐	8	PATROLB1	452	PATROLB1	9	SEA	BUSY	4	OK	-	2

TRACKS: 15

IP NAME	CALLSIGN	MMSI	IMO	TN
TERSCOOTER2	SCOOTER2	357345767	5623457234	G:00013
TERSCOOTER1	SCOOTER1	652372652	5214354352	G:00008
MARTE	MARTE	321235461	3241256476	G:00010
IP1	SHIP1	525346573	4653265476	G:00007
OVA1	PATROLB1		1231234512	G:00009
ICOPTER2	ELICOPTER2	767567346	5432524362	G:00012

SAR TARGET NAME:
SAR_AA1
SAR TARGET ID:
20100722122545110
TYPE: MOTOSCOOTER

POLLUTION AREA NAME:
AREA_AA1
POLLUTION AREA ID:
20100722122611676
COURSE : ---
SPEED : ---

LENGTH: 25

MEANS

CALLSIGN	CODE	NAME	DESCRIPTION	STATUS

Active Missions

Mission ID	Mission Name	Mission Type	Mission Target	Mission Owner
2010722122839...	MIS1	HUNTING	6	vts

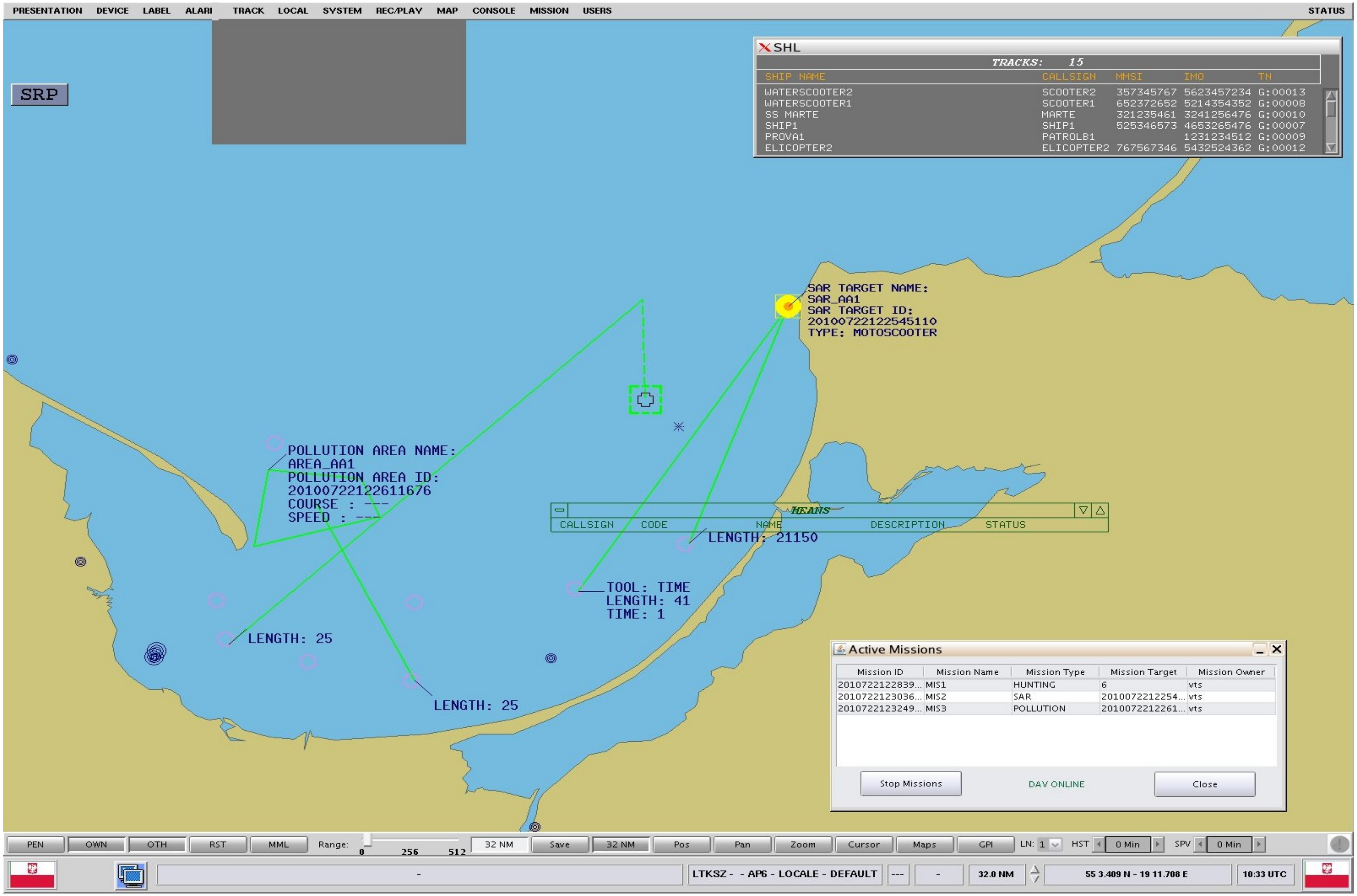
LTKSZ - - AP6 - LOCALE - DEFAULT

32.0 NM

54 54.680 N - 20 24.719 E

10:29 UTC

Mission Execution



SHL

TRACKS: 15

SHIP NAME	CALLSIGN	MMSI	IMO	TN
WATERSCOOTER2	SCOOTER2	357345767	5623457234	G:00013
WATERSCOOTER1	SCOOTER1	652372652	5214354352	G:00008
SS MARTE	MARTE	321235461	3241256476	G:00010
SHIP1	SHIP1	525346573	4653265476	G:00007
PROVA1	PATROLB1		1231234512	G:00009
ELICOPTER2	ELICOPTER2	767567346	5432524362	G:00012

SAR TARGET NAME:
SAR_AA1
SAR TARGET ID:
20100722122545110
TYPE: MOTOSCOOTER

POLLUTION AREA NAME:
AREA_AA1
POLLUTION AREA ID:
20100722122611676
COURSE : ---
SPEED : ---

MEANS

CALLSIGN	CODE	NAME	DESCRIPTION	STATUS

LENGTH: 21150

TOOL: TIME
LENGTH: 41
TIME: 1

LENGTH: 25

LENGTH: 25

Active Missions

Mission ID	Mission Name	Mission Type	Mission Target	Mission Owner
2010722122839...	MIS1	HUNTING	6	vts
2010722123036...	MIS2	SAR	2010072212254...	vts
2010722123249...	MIS3	POLLUTION	2010072212261...	vts

DAV ONLINE

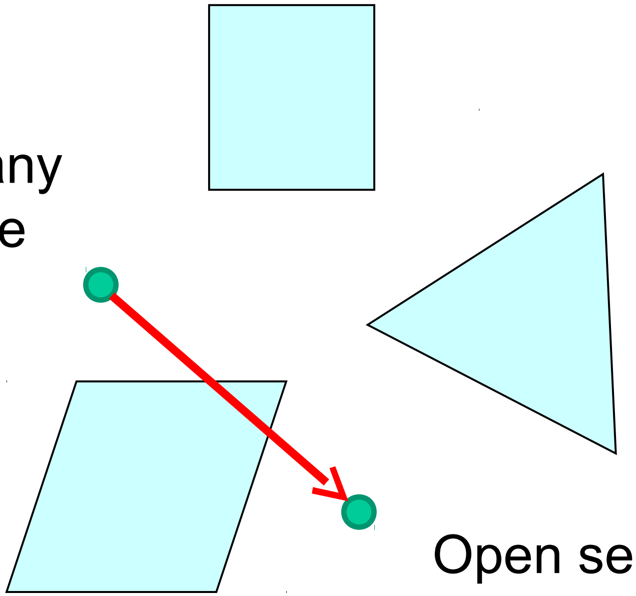
- | LTKSZ - - AP6 - LOCALE - DEFAULT | --- | - | 32.0 NM | 55 3.409 N - 19 11.708 E | 10:33 UTC

AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

Open sea scenery

Open sea scenery is characterized by islands, peninsulas, buoys and many other obstacles that can be modeled as POLYGONS



In this kind of scenery two objects cannot be connected by a single straight line if it intersects obstacles

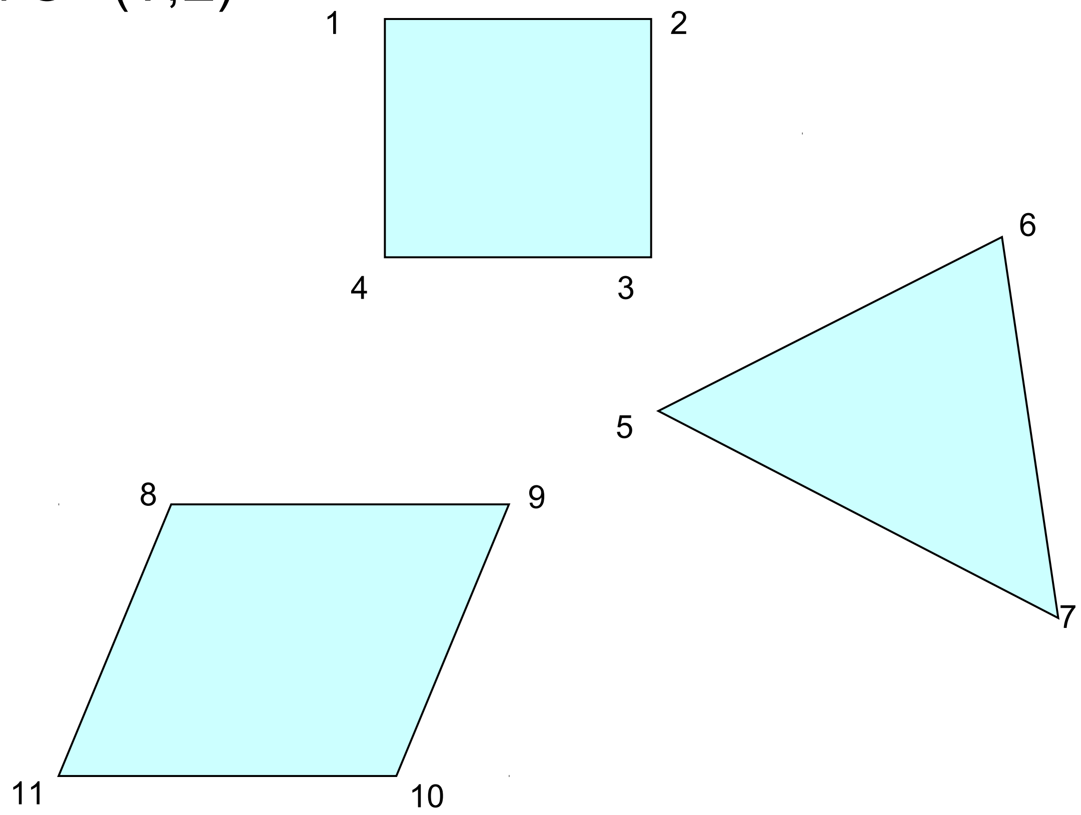
Open sea scenery can be reasonably modeled with a “Visibility Graph” where two objects are reachable one each other by a **shortest path**

A **shortest path** is like a **course** in open sea

Visibility Graph
Algorithm Input/Output

Input: A set of Polygons: P1,..,Pn

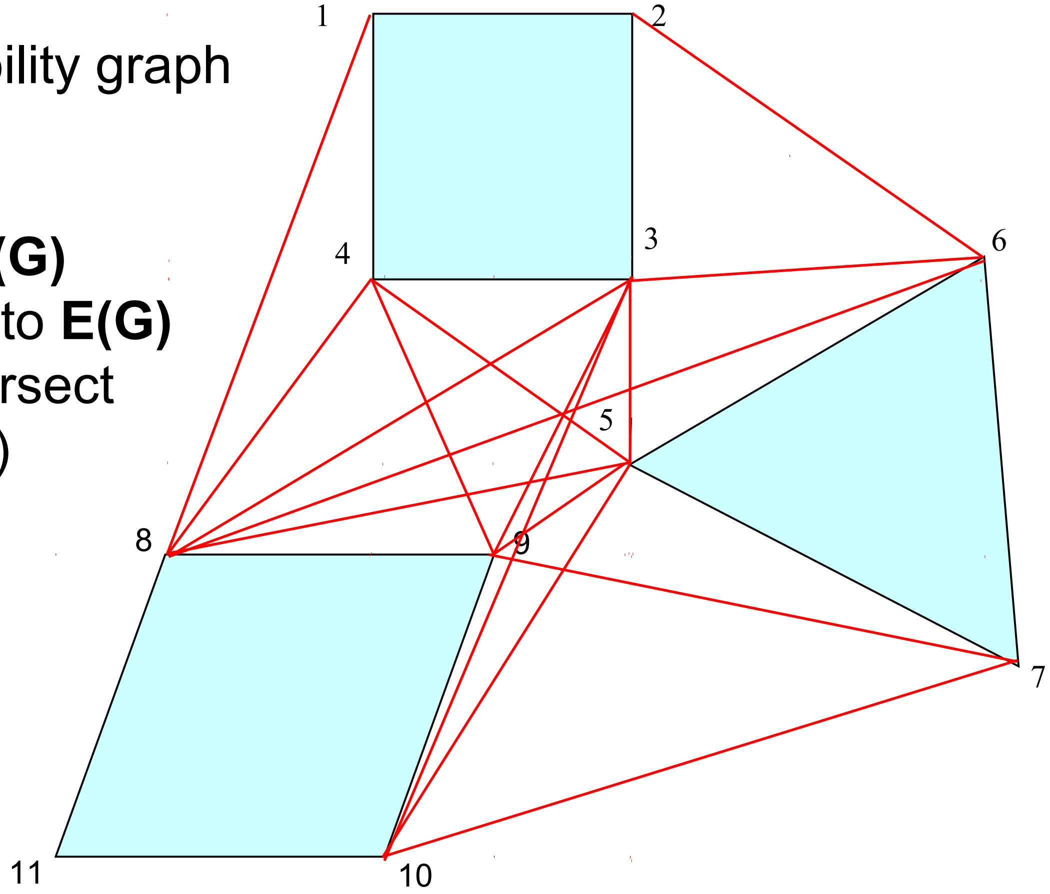
Output: A weighted graph G= (V,E)



Visibility Graph building Algorithm

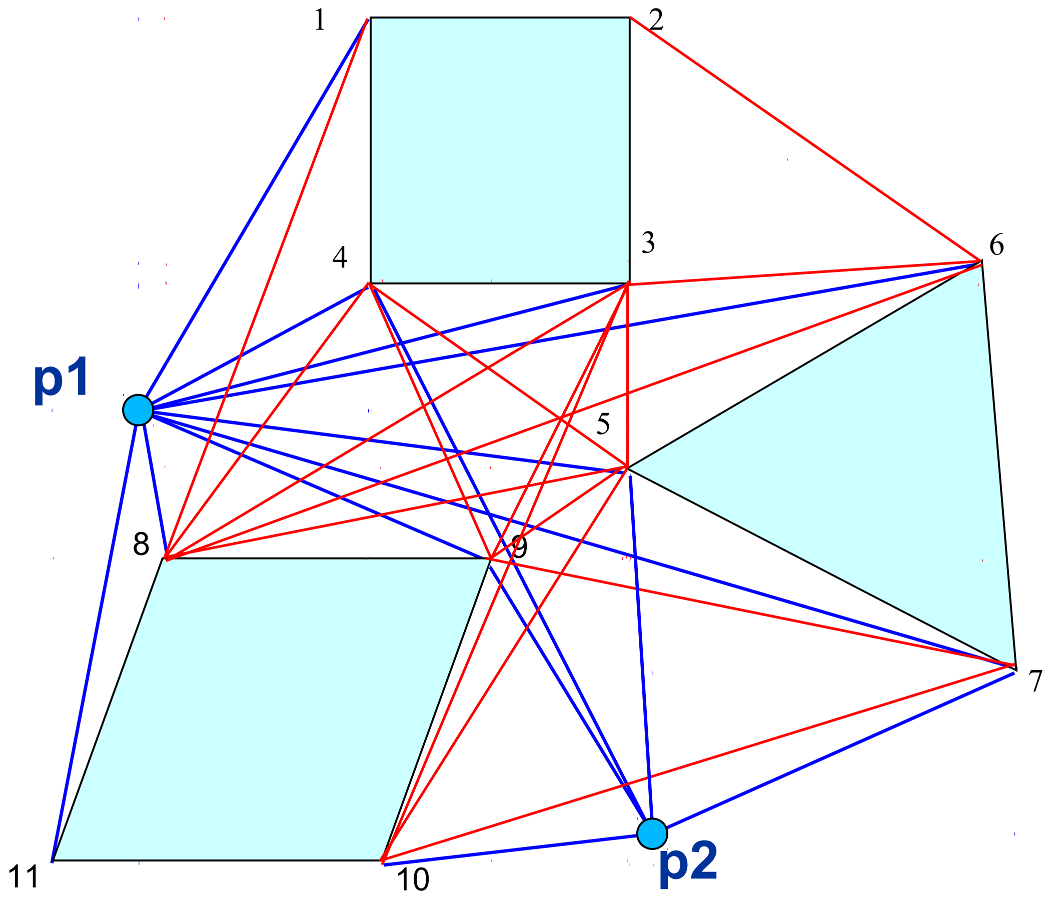
- Given **V(G)** = set of all input polygons vertexes
- Taken **E(G)** = set of visibility graph edges

For each ***u*** in **V(G)**, ***v*** in **V(G)**
An edge ***e=(u,v)*** is added to **E(G)**
if and only if ***e*** doesn't intersect
any of polygons P_i (i=1,..n)



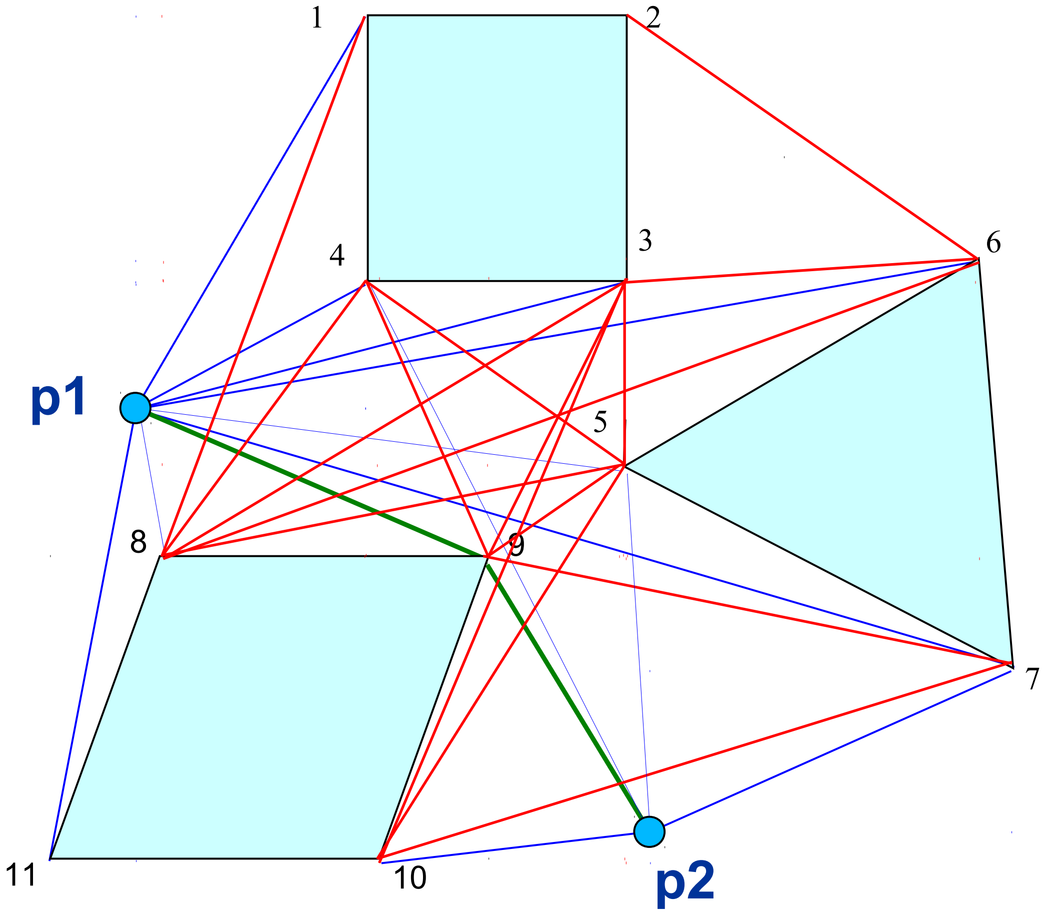
Shortest Path in a Visibility Graph 1/2

Step 1:
Insertion of two input points **p1** and **p2** into the visibility graph



Shortest Path in a Visibility Graph 2/2

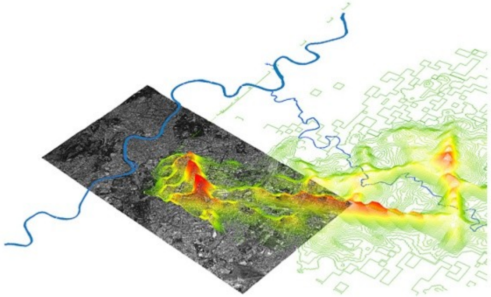
Step 2:
Shortest Path calculation between **p1** and **p2** by a routing algorithm as Dijkstra

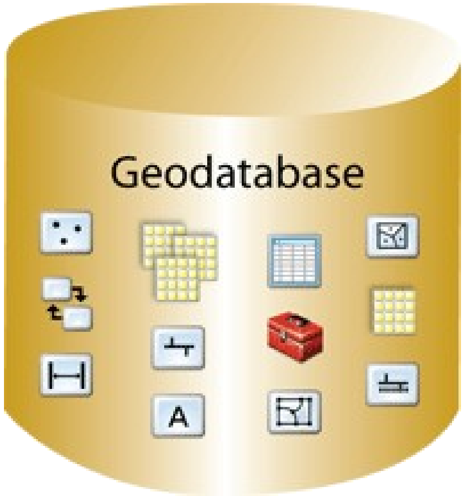


AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

Use of GIS





The need to analyze, elaborate and store geographic information and geometric data suggested to use **Geographic Information System**

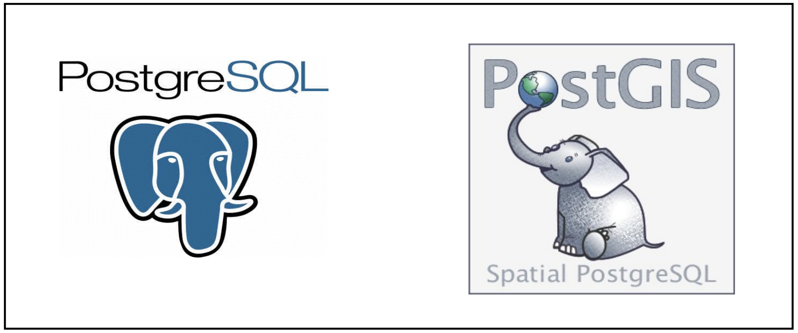
PostgreSQL+Postgis

- **PostgreSQL** has been selected as object-relational DBMS

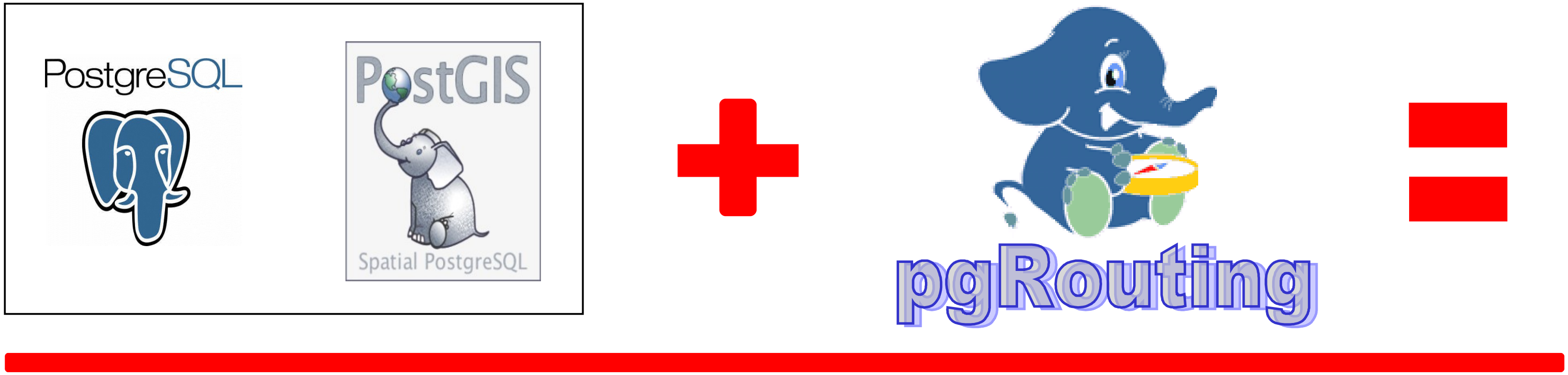
- **PostgreSQL** is supported from ***PostGIS*** for geographic data management

- ***PostGIS*** defines data types that allow to store spatial information as records of a database table

- ***PostGIS*** provides to DBMS functions to manage spatial data



Shortest Path
by GIS-based DB



SHORTEST PATH

•**pgRouting** is a C library that provides **routing** functionality to ***PostGIS/PostgreSQL***

• **pgRouting** already implements algorithms like the following:
- Shortest Path Dijkstra, shortest path algorithm with exact result
- Shortest Path A*, shortest path algorithm with heuristics
- Traveling Sales Person (TSP)

Extended pgRouting for Visibility Graph

PostgreSQL PostGIS Spatial PostgreSQL + pgRouting +

Extensions =

VISIBILITY GRAPH BUILDING

AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

Extended pgRouting

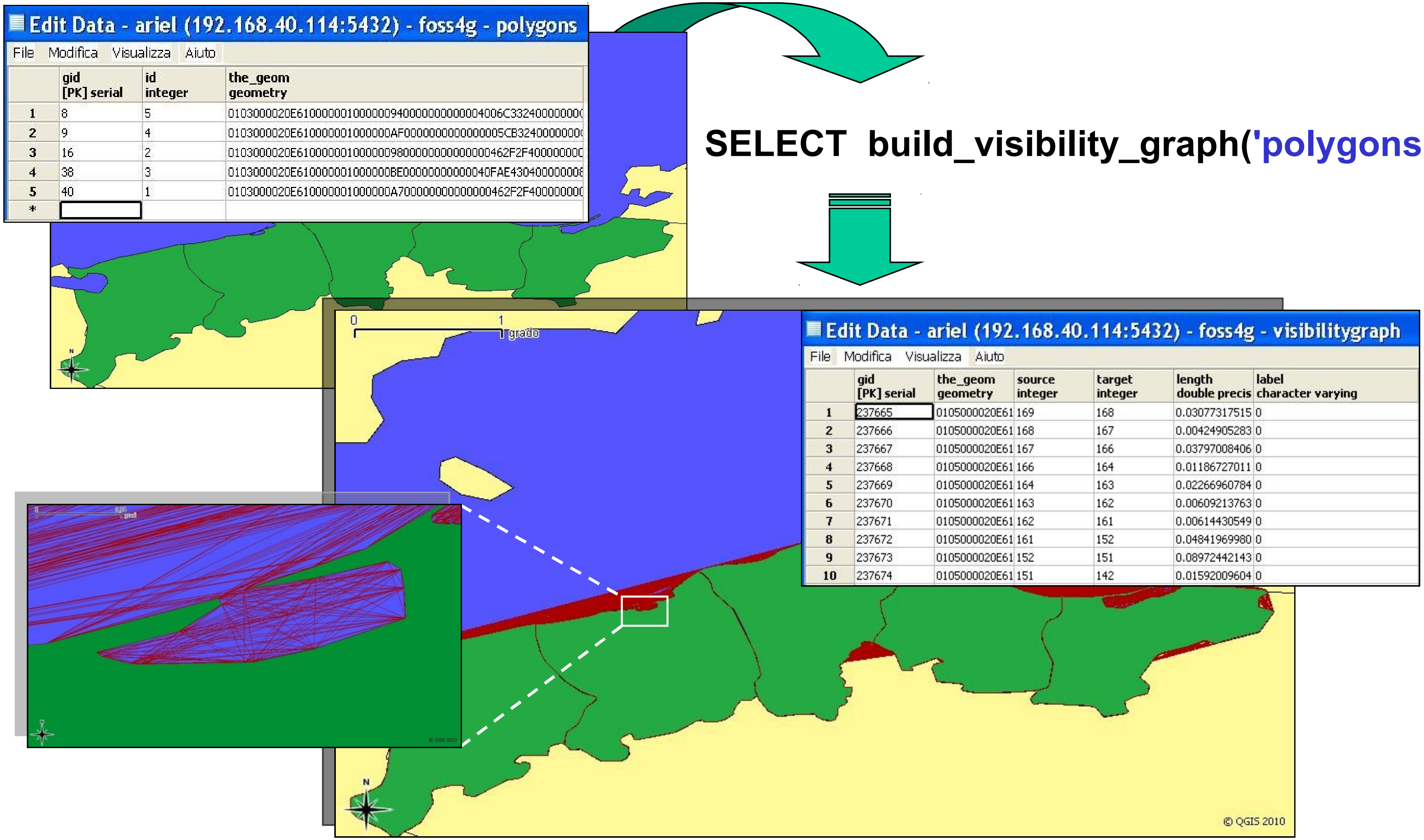
Extensions

1. Visibility graph building starting from POLYGON type objects

2. New points insertion into a previously built visibility graph

3. Shortest path calculation by Dijkstra algorithm between two points into a visibility graph

1.build_visibility_graph



Edit Data - ariel (192.168.40.114:5432) - foss4g - polygons

	gid [PK] serial	id integer	the_geom geometry
1	8	5	0103000020E610000001000000940000000000004006C33240000000
2	9	4	0103000020E610000001000000AF000000000000005CB32400000000
3	16	2	0103000020E610000001000000980000000000462F2F400000000000
4	38	3	0103000020E610000001000000BE00000000000040FAE43040000000
5	40	1	0103000020E610000001000000A70000000000462F2F400000000000
*			

SELECT build_visibility_graph('polygons')

Edit Data - ariel (192.168.40.114:5432) - foss4g - visibilitygraph

	gid [PK] serial	the_geom geometry	source integer	target integer	length double precis	label character varying
1	237665	0105000020E61	169	168	0.03077317515	0
2	237666	0105000020E61	168	167	0.00424905283	0
3	237667	0105000020E61	167	166	0.03797008406	0
4	237668	0105000020E61	166	164	0.01186727011	0
5	237669	0105000020E61	164	163	0.02266960784	0
6	237670	0105000020E61	163	162	0.00609213763	0
7	237671	0105000020E61	162	161	0.00614430549	0
8	237672	0105000020E61	161	152	0.04841969980	0
9	237673	0105000020E61	152	151	0.08972442143	0
10	237674	0105000020E61	151	142	0.01592009604	0

1.build_visibility_graph function

FUNCTION build_visibility_graph(tablename varchar)

Parameter:

- Name of DB table containing POLYGON type objects

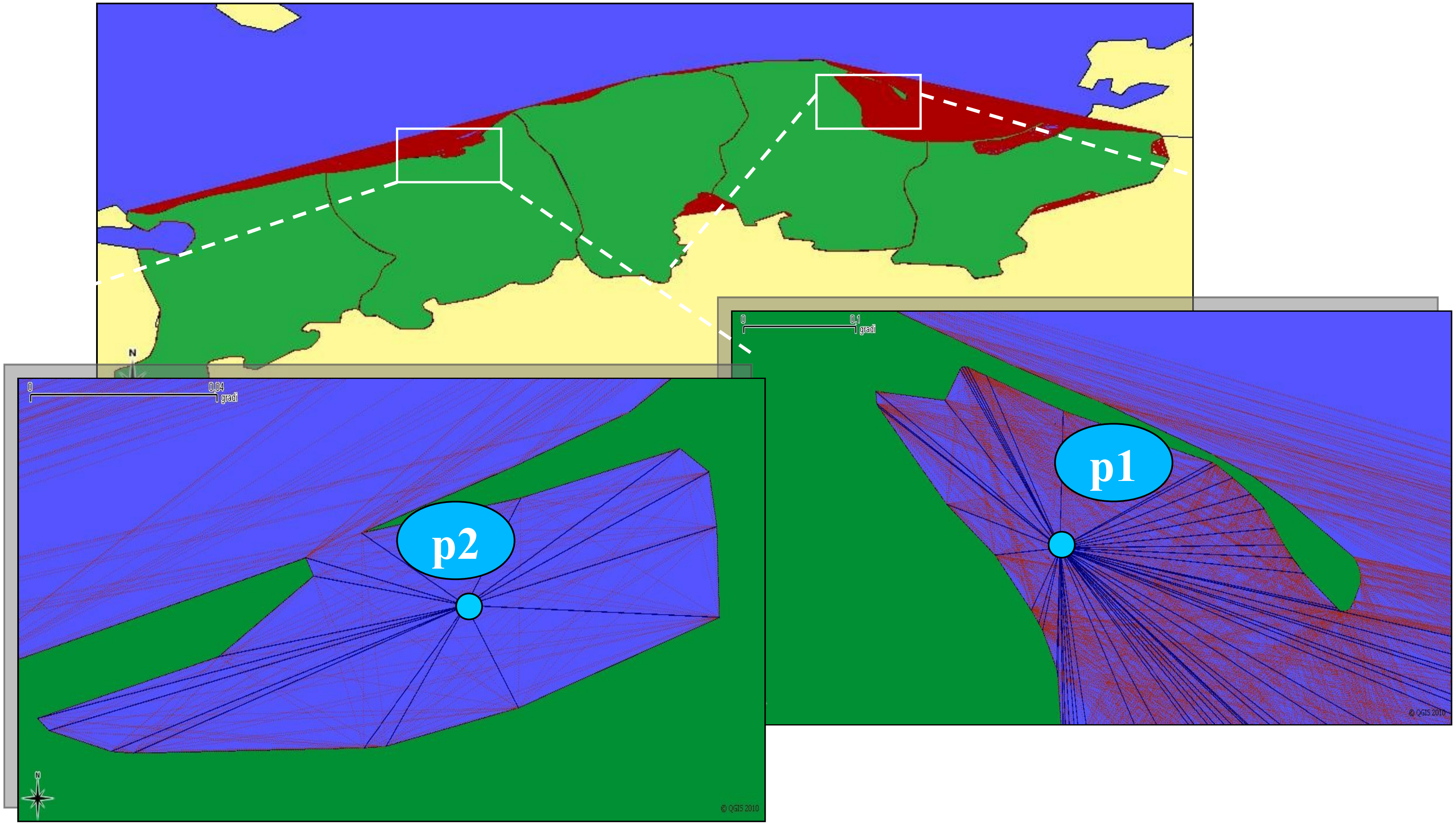
Constraint: Polygons have to be closed.

Description:

- Build a visibility graph by inserting a visibility edge between each couple of input polygons points
- Assign an index to each node of built visibility graph
- Calculate length of each inserted edge

2. insert_point_into_graph

```
SELECT insert_point_into_graph('polygons', ‘label’, point_x,point_y);
```



2. insert_point_into_graph function

```
Datum insert_point_into_graph(tablename varchar,
                              label varchar,
                              p_x float8,
                              p_y float8)
```

p (x,y)

Parameters:

a. name of table containing POLYGONS for which visibility graph has been built

b. label to identify edges to be included for input point **p**

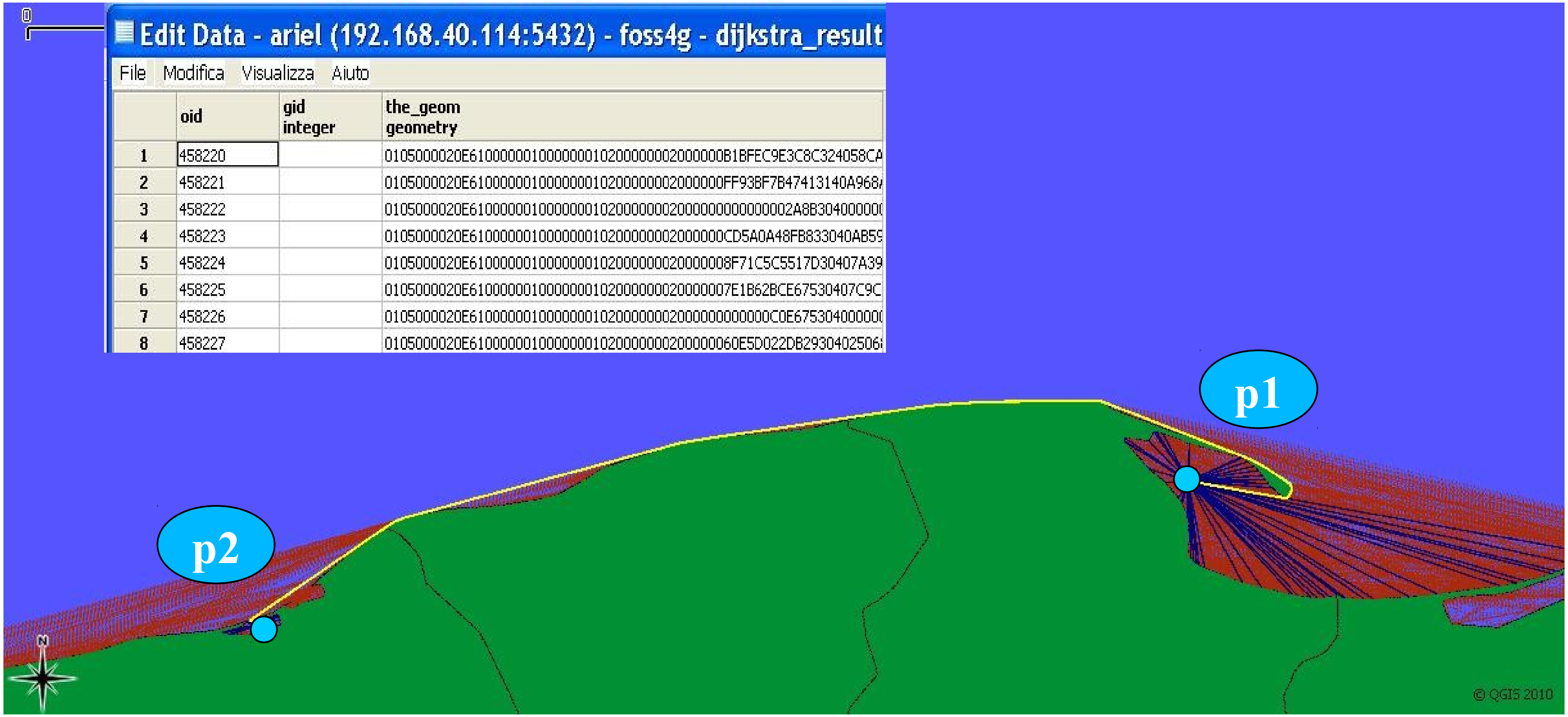
c. x and y coordinates of input point **p**

Description:

- Build visibility edges for input point **p**, of coordinates **(x,y)**, in the visibility graph built from polygons **table**
- Assign a **label** to identify new included edges for point **p**
- Assign a new index to input point **p** in the visibility graph

3. shortest_path_into_ visibilitygraph

SELECT shortest_path_into_visibilitygraph('polygons', *p1_x*, *p1_y*, *p2_x*, *p2_y*, 'label');



Edit Data - ariel (192.168.40.114:5432) - foss4g - dijkstra_result

	oid	gid integer	the_geom geometry
1	458220		0105000020E610000001000000010200000002000000B1BFEC9E3C8C324058CA
2	458221		0105000020E610000001000000010200000002000000FF93BF7B47413140A968
3	458222		0105000020E610000001000000010200000002000000000000002A8B30400000
4	458223		0105000020E610000001000000010200000002000000CD5A0A48FB833040AB59
5	458224		0105000020E6100000010000000102000000020000008F71C5C5517D30407A39
6	458225		0105000020E6100000010000000102000000020000007E1B62BCE67530407C9C
7	458226		0105000020E610000001000000010200000002000000000000C0E67530400000
8	458227		0105000020E61000000100000001020000000200000060E5D022DB2930402506

3. shortest_path_into_ visibilitygraph function

FUNCTION shortest_path_into_visibilitygraph(tablename varchar, p1_x float8, p1_y float8, p2_x float8, p2_y float8, label varchar)

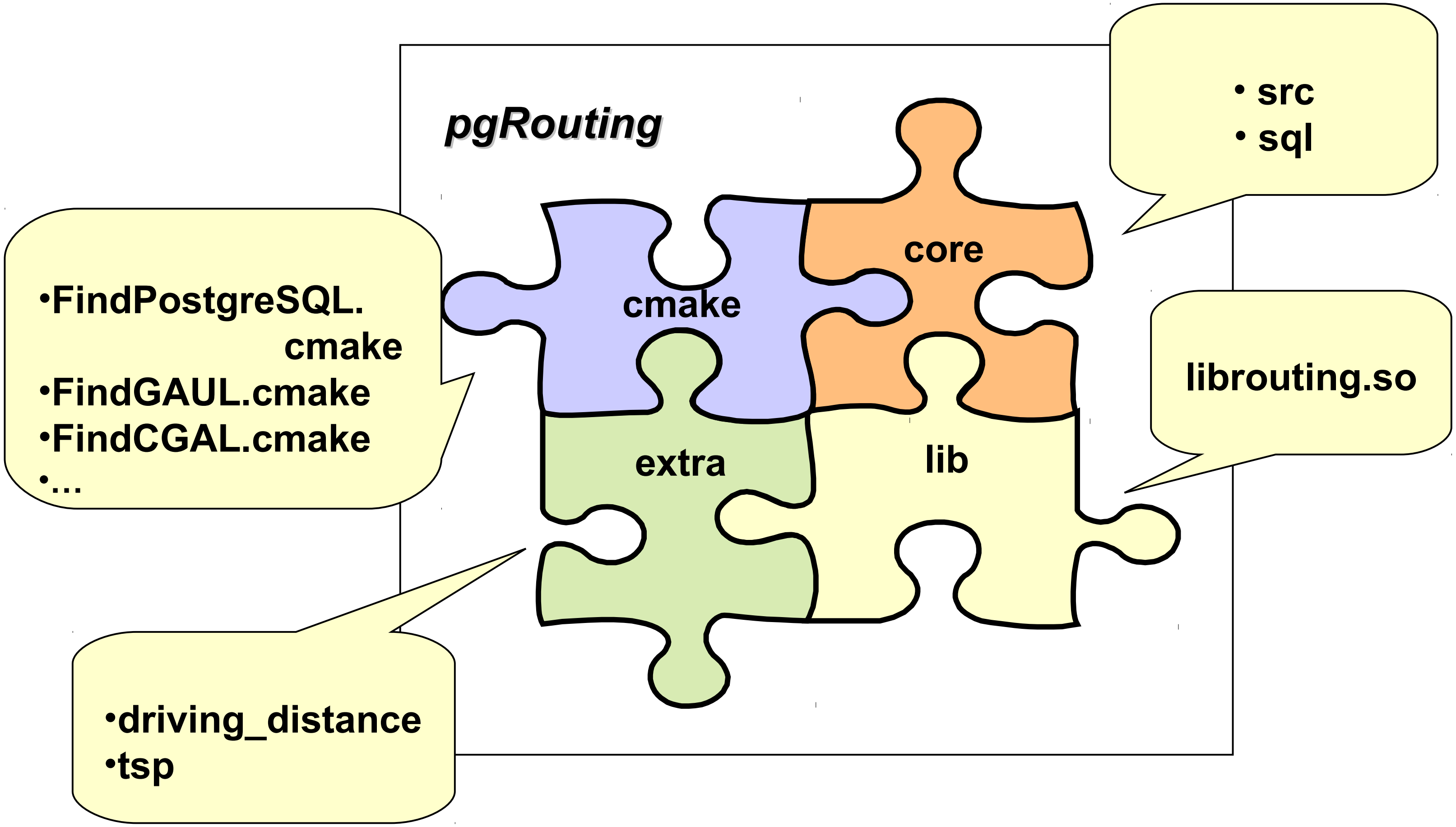
***Parameters*:**

a. **name** of POLYGONS table for which visibility graph has been built
b. **label** to identify edges to be included for input points
c. x and y coordinates of input points **p1**, **p2**

***Description*:**

- insert two points in the visibility graph, built from polygons **table**, by calling ***insert_point_into_graph*** function with '**label**' as argument
- calculate shortest path between **p1** and **p2** by pgRouting *Dijkstra SP* function

Current pgRouting deploy



pgRouting

- src
- sql

core

cmake

extra

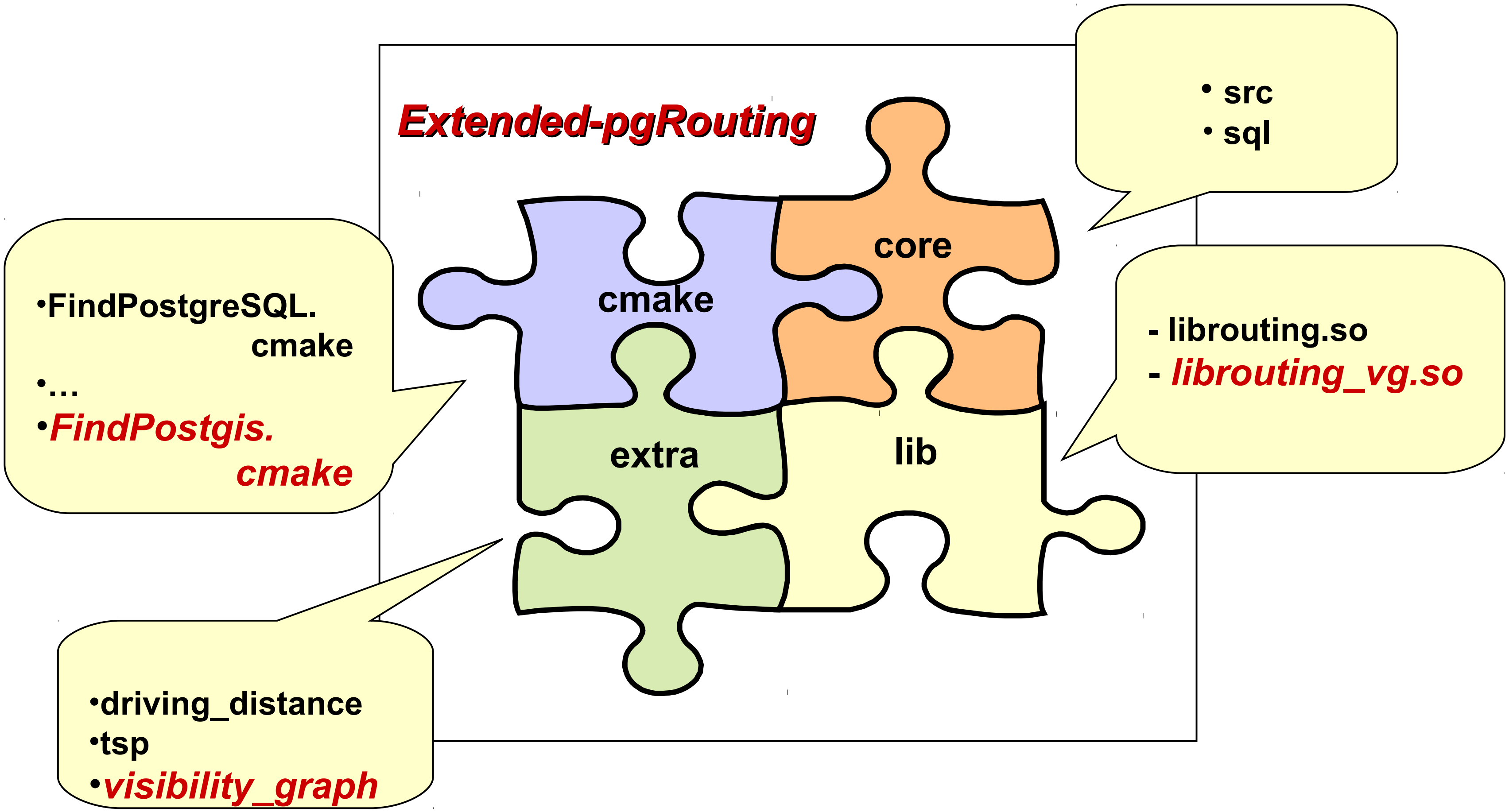
lib

librouting.so

•FindPostgreSQL.cmake
•FindGAUL.cmake
•FindCGAL.cmake
•...

•driving_distance
•tsp

Extended pgRouting deploy



Extended-pgRouting

cmake

core

extra

lib

- src
- sql

•FindPostgreSQL.
cmake
•…
•*FindPostgis.*
cmake

- librouting.so
- *librouting_vg.so*

•driving_distance
•tsp
•*visibility_graph*

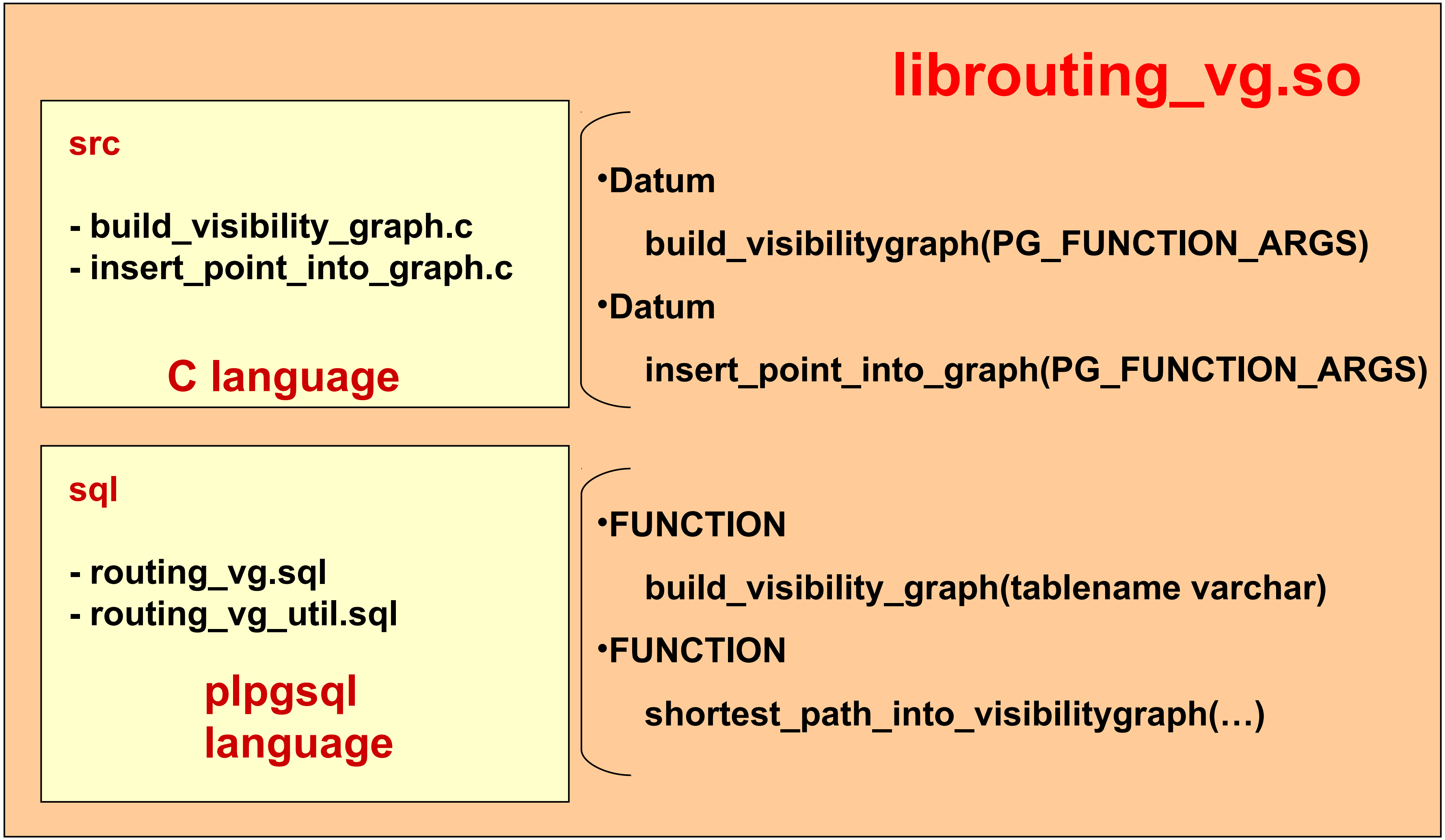
Extended pgRouting:
cmake

FindPostgis.cmake

```
if(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
  set(POSTGIS_FOUND TRUE)

else(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)

 FIND_PATH(POSTGIS_INCLUDE_DIR postgis_config.h
      /usr/local/pgsql/include/server
      /usr/local/include/pgsql/server
       ${PGROUTING_CORE_INCLUDE_DIR}
      $ENV{ProgramFiles}/PostgreSQL/*/include/server
      $ENV{SystemDrive}/PostgreSQL/*/include/server
  )

  FIND_PATH(POSTGIS_INCLUDE_DIR liblwgeom.h
      $POSTGIS_HOME/liblwgeom
      )

  find_library(POSTGIS_LIBRARIES NAMES libpostgis
postgis
     PATHS
     /usr/lib
     ….
     $ENV{ProgramFiles}/PostgreSQL/*/lib/ms
     $ENV{SystemDrive}/PostgreSQL/*/lib/ms
     )
```

```
 if(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
    set(POSTGIS_FOUND TRUE)
    message(STATUS "Found Postgis: $
{POSTGIS_INCLUDE_DIR}, ${POSTGIS_LIBRARIES}")
    INCLUDE_DIRECTORIES(${POSTGIS_INCLUDE_DIR})
  else(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
    set(POSTGIS_FOUND FALSE)
    message(STATUS "Postgis not found.")
  endif(POSTGIS_INCLUDE_DIR AND
POSTGIS_LIBRARIES)

  mark_as_advanced(POSTGIS_INCLUDE_DIR
POSTGIS_LIBRARIES)

endif(POSTGIS_INCLUDE_DIR AND POSTGIS_LIBRARIES)
```

Extended pgRouting: *extra/visibility_graph*



librouting_vg.so

src

- build_visibility_graph.c
- insert_point_into_graph.c

C language

- **Datum**

  build_visibilitygraph(PG_FUNCTION_ARGS)
- **Datum**

  insert_point_into_graph(PG_FUNCTION_ARGS)

sql

- routing_vg.sql
- routing_vg_util.sql

plpgsql language

- **FUNCTION**

  build_visibility_graph(tablename varchar)
- **FUNCTION**

  shortest_path_into_visibilitygraph(…)

AGENDA

- VTMS overview
- Decision Support in a VTMS
- Visibility Graph to model Open Sea
- Geospatial DB and GIS functions to solve SP
- Extended pgRouting for Visibility Graph
- Conclusions

Conclusions

UP:

- Optimized DBMS GIS libraries can be exploited to easily represent and manage geographic objects

- The complexity of calculations is totally demanded to the DBMS set of functions

DOWN:

- Extended pgRouting approach turned out to be not effective for near real time application involving lots of multi sensor tracks

e.g. In a scenery with more than 2000 tracks, the insertion of one node into a previously built visibility graph, on a PowerPC needs about 4 seconds. Moreover, time to calculate shortest path between two points is about 7,5 seconds

- Extended pgRouting is work in progress again and has been used only internally as object of study

Thank you!

Contacts:

Angela Pappagallo - angela.pappagallo@intecs.it

Domenico Balestrieri - domenico.balestrieri@intecs.it

Massimo Costantini - massimo.costantini@intecs.it

Roberto De Felici - rdefelici@selex-si.com (VTMS System)

Visit our web-sites

http://www.intecs.it

http://www.selex-si.com