Software Engineering Project

Cornellà, Garcia, Kozlovskis, Pak

University of Burgundy

7 of January of 2014

Outline

- Tools and resources
- Methodology
- C++ implementation
- Matlab implementation

Tools and resources

IDE

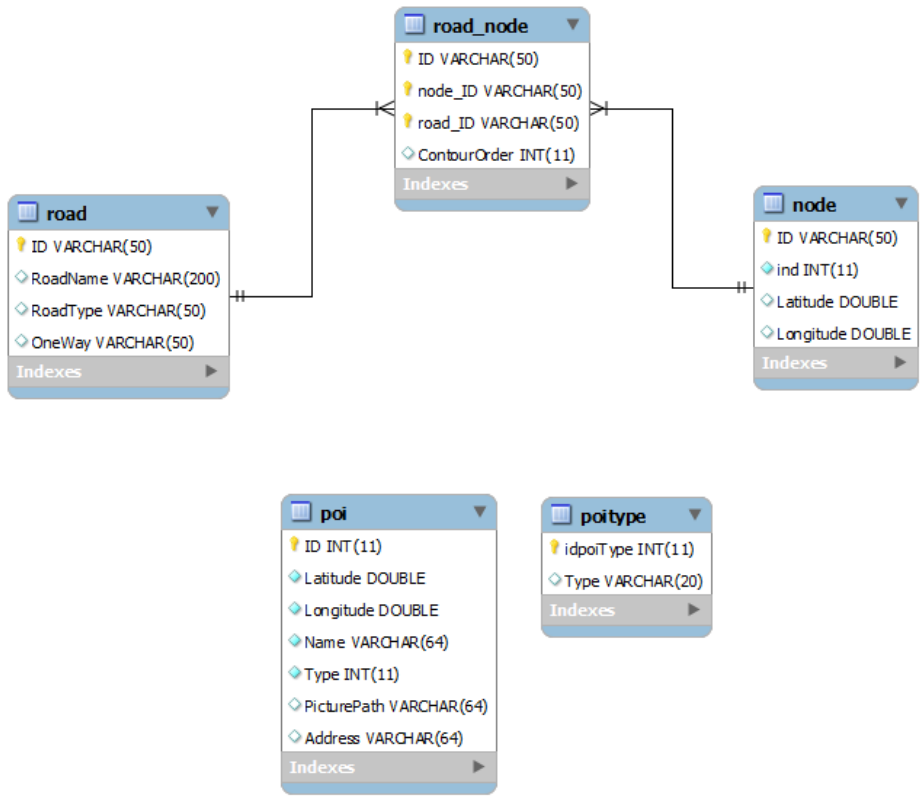
- Qt 5.1 with OpenGL
- Matlab 2013Ra

Group meetings and coordination

- Trello
- Git and Bitbucket

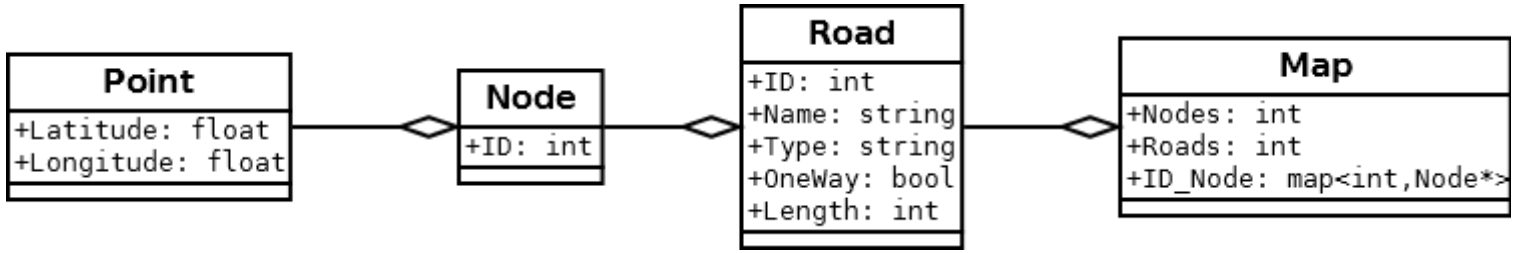
Database

- MySQL Server 5.6
- PostgreSQL



C++ Implementation

- Main Structure:



- Loaded in memory through QMYSQL Driver
- Displayed using GL_STRIPES of OpenGL
- Creation of a map of ID_Node* because of the complexity
  - Avoiding Queries
  - Avoiding double loops

C++: Dijkstra's algorithm

- Other kind of algorithms:
  - Bellman-Ford and SPFA
  - Jonhson's algorithm
  - Floyd-Warshall algorithm
  - Dijkstra's algorithm
- Why? Performs better with non negative values
- How it works?
  1. Initialize D to 0 in the diagonal and ∞ in non connected nodes
  2. Suppose that $a = x$ (current node)
  3. Check all adjacent nodes of a except the ones that are marked
  4. If $(D_i > D_a + d(a, v_i))$ then, $D_i = D_a + d(a, v_i)$
  5. Marked as completed the node a
  6. We take as next current node the smaller in D
  7. Go back to step 3 until there are nodes not marked

C++: Adjacency Matrix

- Sparse Matrix of the Euclidean distance between nodes (Eigen library)

0	3	0	0	0
22	0	0	0	17
7	5	0	1	0
0	0	0	0	0
0	0	14	0	8

Results measured under Debug mode and dependent on the performance of each processor

Values:	22	7	3	5	14	1	17	8
InnerIndices:	1	2	0	2	4	2	1	4

Table 1: Sum up of the benefits of Sparse matrices

	Matrix	Sparse
Num. nodes	$\simeq 84{\cdot}10^6$	$\simeq 16{\cdot}10^3$
Memory	$\simeq$336 MB	$\simeq$64 KB + Zero vector
Time	>30 sec	<1 sec
Dijkstra	>30 sec	<3 sec

Route path

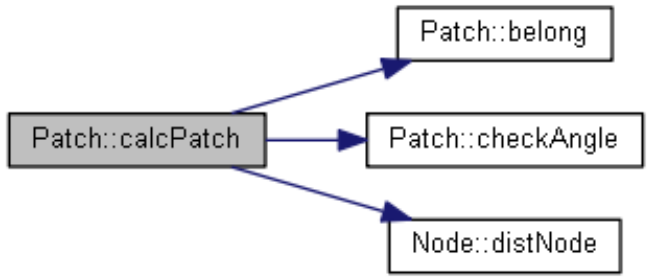


Figure : Functions involved in the route path calculation

- Functions:
  - Path calculation
  - Distance and time
  - Export
- General case:
  Same road $\rightarrow$ NO $\rightarrow$ *Angle* $< 180 \rightarrow$ Right
  Same road $\rightarrow$ NO $\rightarrow$ *Angle* $> 180 \rightarrow$ Left
  Same road $\rightarrow$ YES $\rightarrow$ Distance

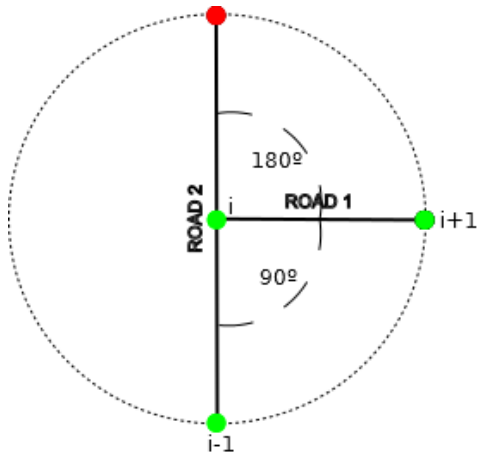


Figure : Example of road intersection

Heuristics

Normalization and displaying

Selecting a route

Matlab implementation

Getting the data

- PostgreSQL database
- Queries were made only once.
- Created objects can be saved in .MAT files
- No need to recreate object on each program start

Matlab implementation

Data Structure

- Objects of 4 classes were constructed.
- No pointers in Matlab.
- Solution: handle classes.

Figure : Simplified diagram of relationship of a classes

Matlab implementation

The shortest path

- Dijkstra's algorithm
- Determine the window on which the closest nodes and projections will be searched.
- Find the closest node on determined window.
- Find the closest projection (if possible) on the same window.
- Update the adjacency matrix according shortest distance.

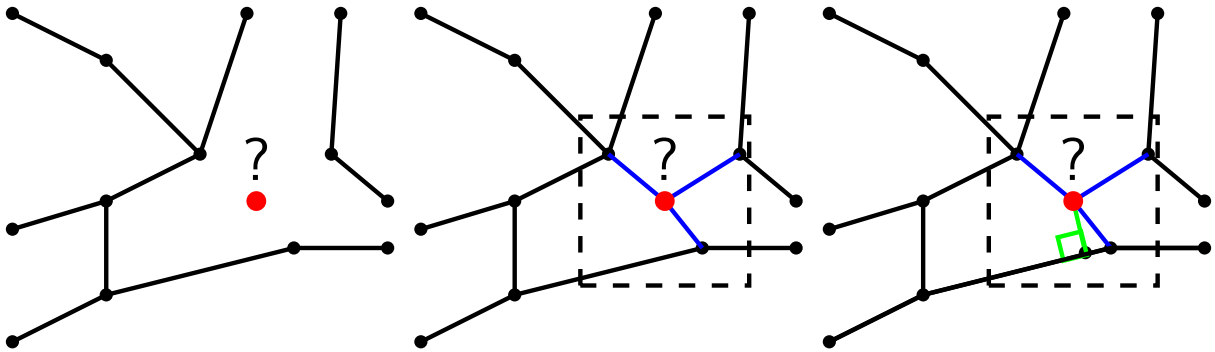
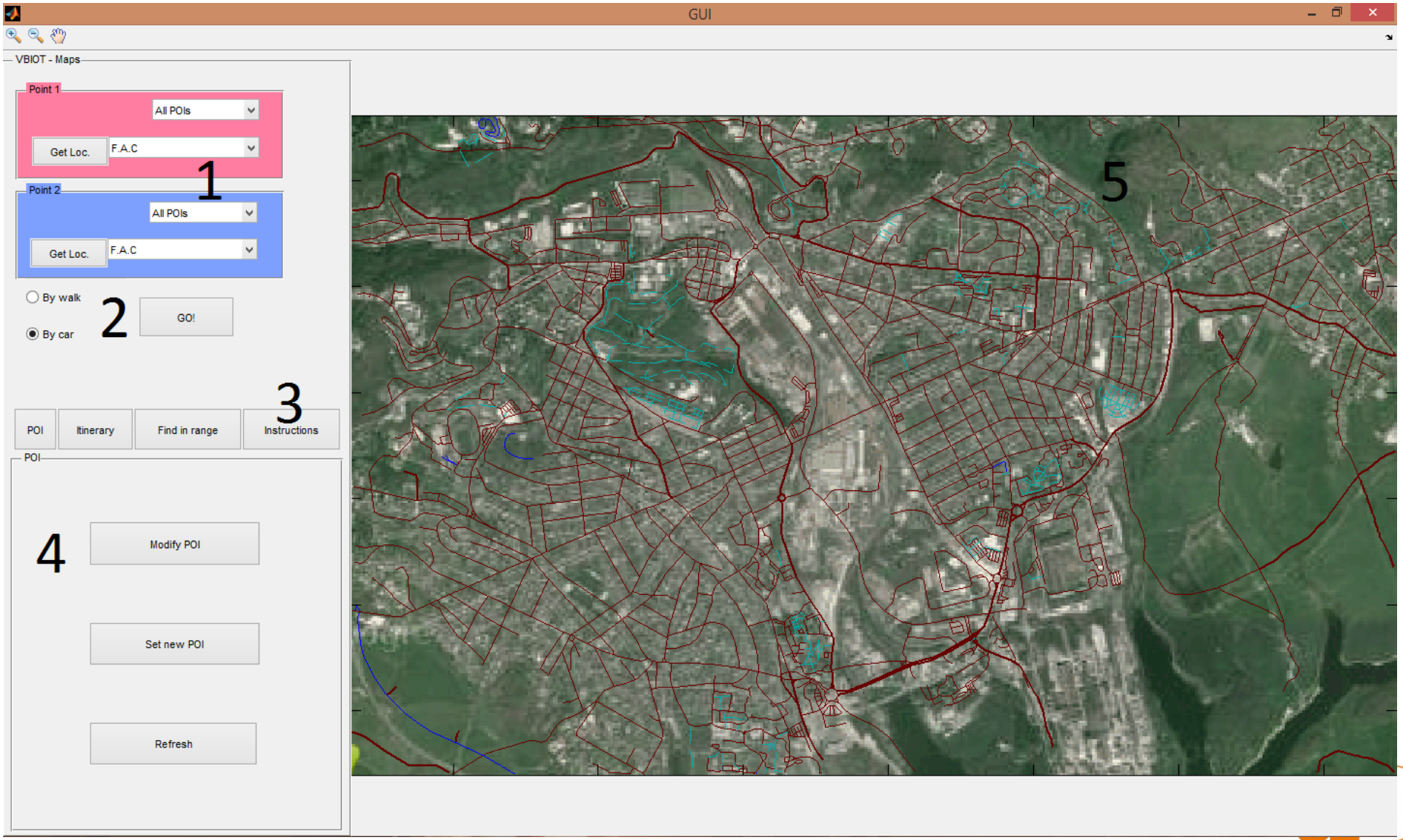


Figure : Finding the shortest point/projection for the random point on window domain

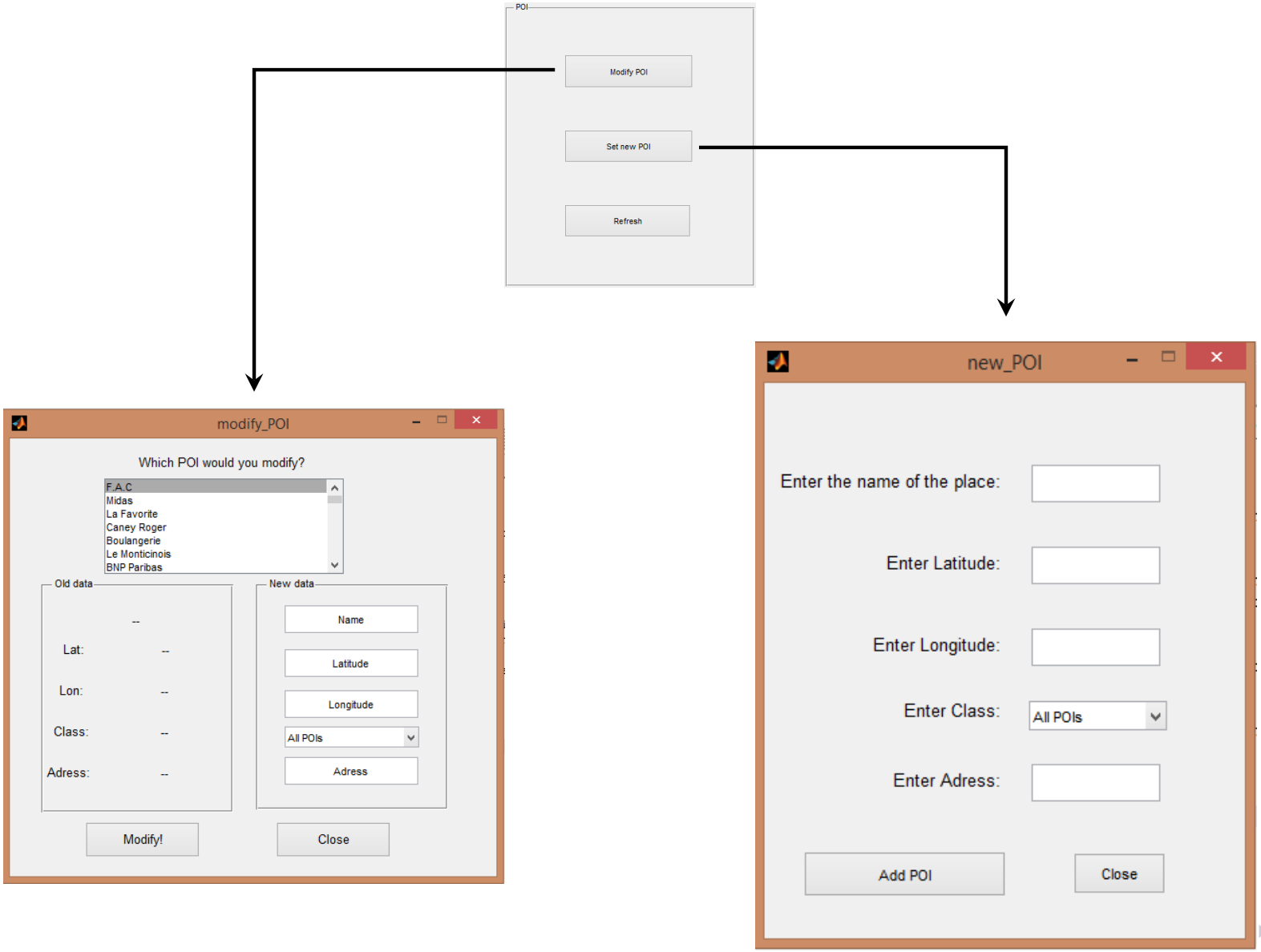
Graphical User Interface (GUI)



Insertion of points

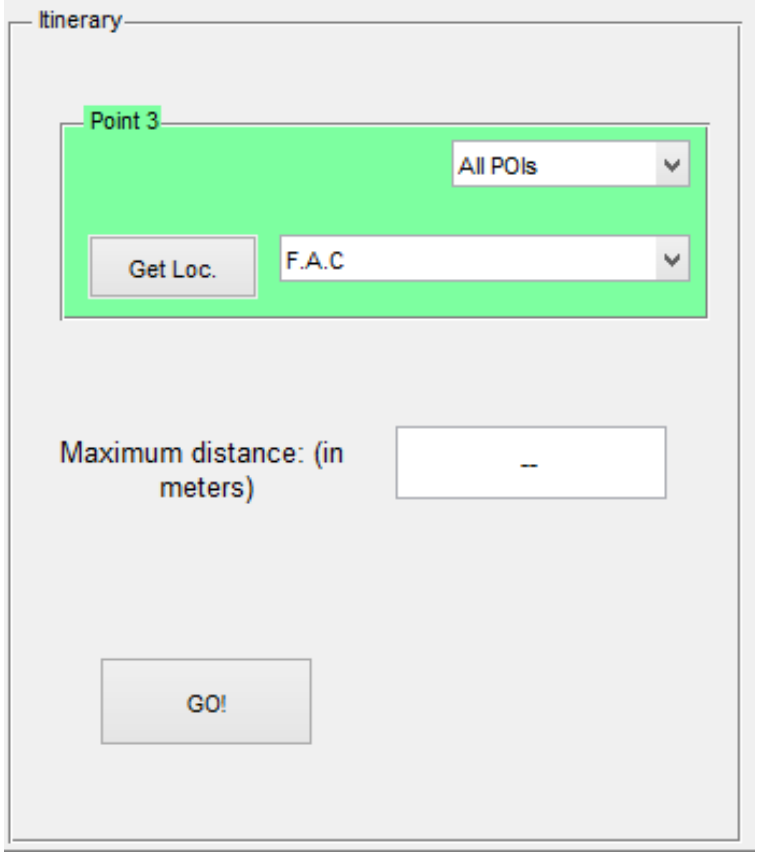
- Point from a list
- Filter by class
- Get Location from the map
- Walk / Car
- GO!

Manipulating Points of interest



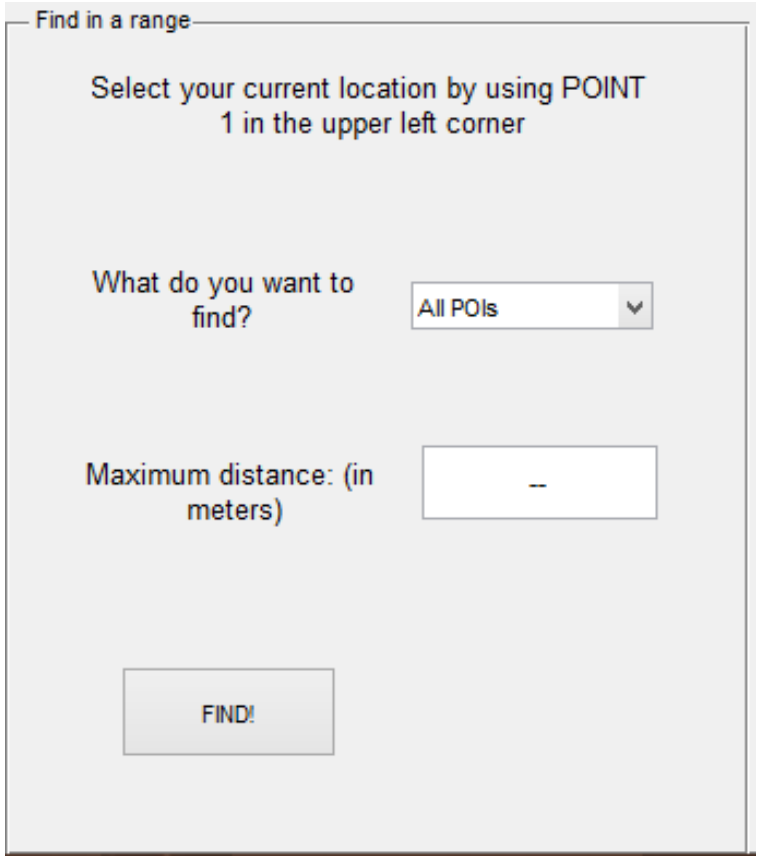
Itinerary

- Adding a third point
- Getting and storing the distance
- How it works
- Cost of time



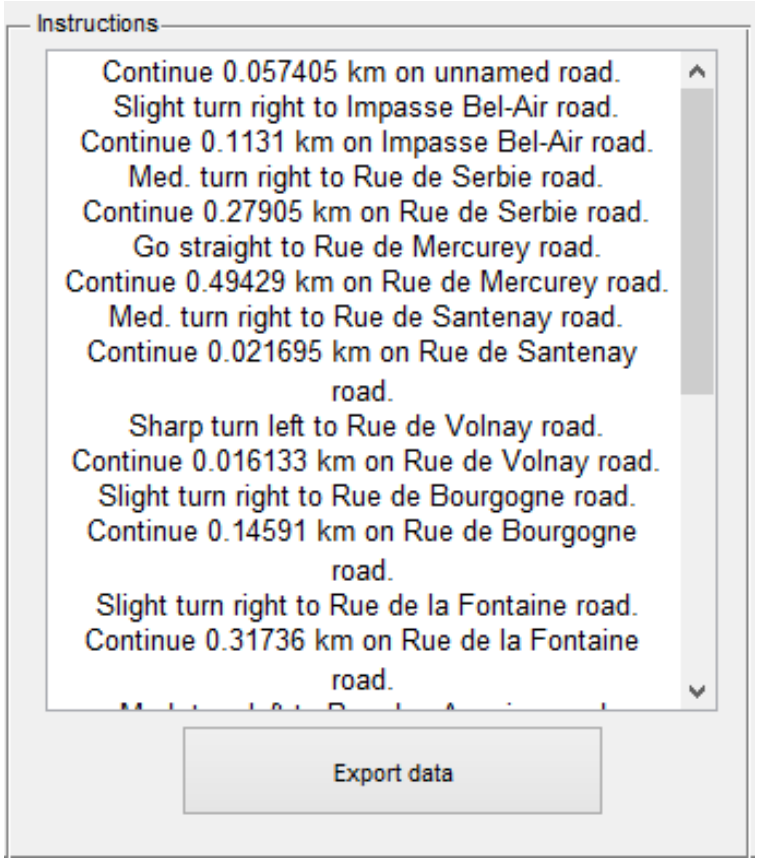
Find in a Range

- Using Point 1 data
- Selection of classes
- Distance
- Slow process



Generation of the instructions

- Generating instructions from shortest path
- Setting a scroll bar
- Exporting this data
- Default text when missing data



Conclusions

Thank you for attention!

Questions?