

MWGen: A Mini World Generator

Jianqiu Xu and Ralf Hartmut Güting

Database Systems for New Applications, Mathematics and Computer Science
FernUniversität in Hagen, Germany



July, 2012

- 1 Problem Description
- 2 Method
- 3 Conclusions

- Moving objects Databases
- Moving objects with transportation modes
 - ① Inferring outdoor transportation modes (e.g., cycling, walking, driving) from GPS data [1,2,3]: Geolife Project in Microsoft
 - ② Advanced trip planning [4], different modes (e.g., *Walk* → *Bus*) and constraints (e.g., less than two bus transfers)

[1] Y. Zheng, L. Liu, L. Wang, X. Xie: Learning transportation mode from raw gps data for geographic applications on the WWW, 2008.

[2] Y. Zheng, Y. Chen, Q. Li, X. Xie, W.Y. Ma: Understanding transportation modes based on GPS data for web applications. ACM Transaction on the Web, 4(1), 2010.

[3] L. Stenneth, O. Wolfson, P. S. Yu, B. Xu: Transportation mode detection using mobile phones and GIS information. GIS, 2011.

[4] J. Booth, A. P. Sistla, O. Wolfson, I. F. Cruz: A data model for trip planning in multimodal transportation systems. EDBT, 2009.

- Example trips
 - 1 *Indoor* → *Walk* → *Car*
 - 2 *Bus* → *Walk* → *Indoor*
- Global Work: Represent and manage moving objects with different transportation modes in a database system and provide efficient query processing.
 - 1 Data Model: J. Xu and R.H. Güting. A Generic Data Model for Moving Objects, Geoinformatica, 2012.
 - 2 **Data Generator: J. Xu and R.H. Güting. MWGen: A Mini World Generator, MDM, 2012.**
 - 3 Benchmark: J. Xu and R.H. Güting. GMOBench: A Benchmark for Generic Moving Objects, Informatik-Report 362, FernUni in Hagen, Germany, 2012.

- The goal: Generating moving objects in different environments where (1) the precise location in each environment and (2) transportation modes are managed.
- Existing data generators for moving objects
 - 1 free space: GSTD [1], BerlinMOD [2]
 - 2 road network: [3]
 - 3 indoor: [4]

[1] Y. Theodoridis, J. R. O. Silva, M. A. Nascimento: On the Generation of Spatiotemporal Datasets. SSD, 1999.

[2] C. Düntgen, T. Behr, R.H. Güting: BerlinMOD: a benchmark for moving object databases. VLDB J. , 18(6):1335-1368, 2009.

[3] T. Brinkhoff, A Framework for Generating Network-Based Moving Objects. Geoinformatica 6(2):153-180, 2002.

[4] C. S. Jensen, H. Lu, B. Yang, Indexing the Trajectories of Moving Objects in Symbolic Indoor Space. SSTD, 2009.

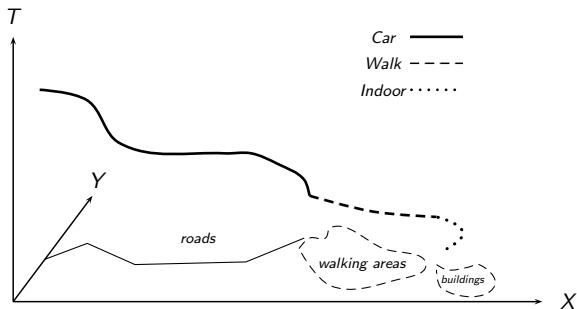
- 1 Preliminary
- 2 Framework
- 3 Trip plannings
- 4 Moving objects generation
- 5 Experimental results

- Available environments
 - 1 Road Network
 - 2 Region-based Outdoor
 - 3 Bus Network and Metro Network
 - 4 Indoor
- Transportation modes
 $TM = \{Car, Taxi, Bike, Walk, Bus, Metro, Indoor\}$

- Data representation (location and moving objects)

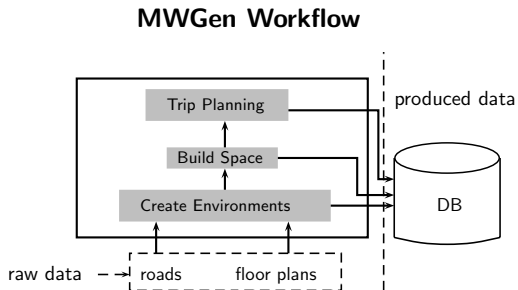
(1) $D_{genloc} = \{(oid, (loc_1, loc_2)) \mid oid \in D_{int}, loc_1, loc_2 \in D_{real}\}$

(2) $mo = \langle u_1, u_2, \dots, u_n \rangle$ where $u_i = (t, g_l^1, g_l^2, m)$, $g_l^1, g_l^2 \in D_{genloc}$, $m \in D_{TM}$



$mo = \langle u_1(t_1, Indoor_loc1, Indoor_loc2, Indoor), \dots, u_i(t_i, Pave_loc1, Pave_loc2, Walk), \dots, u_n(t_n, Road_loc1, Road_loc2, Car) \rangle$

- Framework

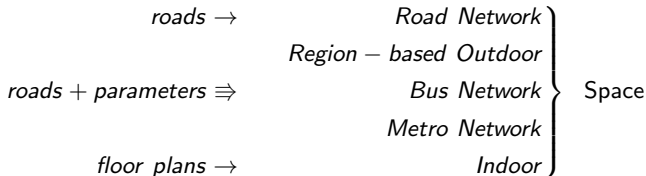


- **Input:**

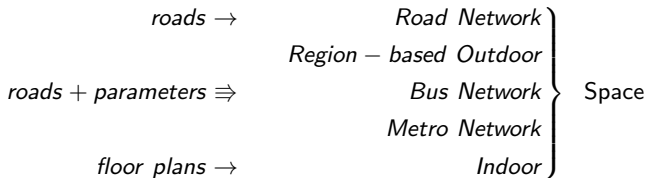
- 1 a set of roads represented by lines
- 2 floor plans
- 3 parameters such as road width

- **Output:**

- 1 Environments: Road Network, Region-based Outdoor, Bus Network, Metro Network, and Indoor
- 2 Moving objects with multiple transportation modes such as *Indoor* → *Walk* → *Bus* → *Walk*



- 1 Road Network: roads and junctions
- 2 Region-based Outdoor: pavements and zebra crossings
- 3 Bus and Metro Network: routes, stops and moving buses (metros)
- 4 Indoor: rooms, corridors, staircases and doors.



- An Environment

- 1 the object set
- 2 indices (B-tree and R-tree)
- 3 a graph for routing

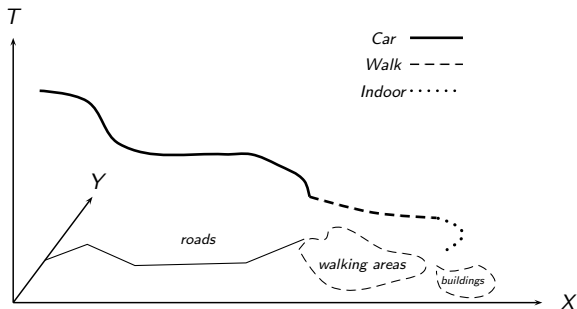
- Space

- 1 managing each environment
- 2 location mapping (e.g., bus stop)
- 3 an interface between moving objects and underlying geographic objects such as roads, bus routes and rooms

- Trip Planning
 - One environment
 - ① shortest path searching in a road network
 - ② shortest path searching for pedestrians (a large polygon with obstacles)
 - ③ routing in a bus network (combine bus and walk movements); routing in a metro network
 - ④ indoor navigation (precise path inside a building)
 - Time Complexity Analysis
 - Multiple environments (graphs and location mapping):
 - ① *Indoor* → *Walk* → *Car* → *Walk*
 - ② *Walk* → *Bus* → *Walk* → *Indoor*

Method - Moving Objects Generation

- Moving Objects Generation: paths + speed values



Method - Experimental Results

- Results

Input

Roads	Berlin (3,250); Houston (4,575)
Floor plans	office building, shopping mall, university... (8 in total)
Parameters	road width, pavement width, etc.

Output

	Berlin	Houston
X Range	[0, 44411]	[0, 133573]
Y Range	[0, 34781]	[0, 163280]
No. Vertices in P	116,516	437,279
Bus Routes	89	92
Metro Routes	10	16
Buildings	4,996	5,992

Trip No.	Berlin		Houston	
	Time (h)	Size (GB)	Time (h)	Size (GB)
4k	0.32	0.052	0.57	0.038
.
.
.
500k	39.75	6.35	74.06	4.95

Moving objects with different transportation modes:

Walk + Indoor + Car(Taxi, Bus, Metro)

Time Cost (sec) for Outdoor Trip Planning

	Berlin	Houston
Region-based Outdoor	0.78	2.4
Bus Network	0.13	0.23
Metro Network	< 0.1	< 0.1

Time Cost (sec) for Indoor Navigation

officeA	0.25	officeB	0.27
mall	0.37	cinema	0.32
hotel	1.57	hospital	0.35
university	0.123	trains station	0.1

Method - Experimental Results

Berlin: Pavements

Second (0) [New View]

Program Server Optimizer Command Help Viewers File Settings Object Creation Object Close Window

```
query allholes feed filter[.0id > 1] consume;...successful
see result in object list
Secquery allholes feed filter[.0id > 1] consume;
query allholes feed filter[.0id > 1] consume;...successful
see result in object list
Secquery allholes feed filter[.0id > 1] consume;
query allholes feed filter[.0id > 1] consume;...successful
see result in object list
Secquery allholes feed filter[.0id = 1] consume;
query allholes feed filter[.0id = 1] consume;...successful
see result in object list
Secquery allholes feed filter[.0id = 1] consume;
query allholes feed filter[.0id = 1] consume;...successful
see result in object list
Sec
```

show	hide	remove	clear
save	load	store	rename

```
** query r;
** query allholes feed filter[.0id > 1] consume;
** query allholes feed filter[.0id = 1] consume;
```

no time 42256.85608304408 / 22979.707364

query allholes feed filter[...]

```
0id : 1300
Hole : busroute
-----
0id : 1301
Hole : busroute
-----
0id : 1302
Hole : busroute
-----
0id : 1303
Hole : busroute
-----
0id : 1304
Hole : busroute
-----
0id : 1305
Hole : busroute
-----
0id : 1306
Hole : busroute
-----
0id : 1307
Hole : busroute
-----
0id : 1308
Hole : busroute
-----
0id : 1309
Hole : busroute
-----
0id : 1310
Hole : busroute
-----
0id : 1311
```

search go

Berlin: Roads + Bus Routes

The screenshot displays a software application window titled "Secondo GUI (Data Viewer)". The interface is divided into several sections:

- Command Window (Top Left):** Contains a series of commands and their outputs:

```
Sec>open database berlingena0;
open database berlingena0;...successful
no result
Sec>query r;
query r;...successful
see result in object list
Sec>query bn_busroutes(bn) extend(Geo:brgeodata(.Bus_route))
consume;
query bn_busroutes(bn) extend(Geo:brgeodata(.Bus_route))
consume;...successful
see result in object list
Sec>
```
- Action Table (Top Right):** A table with four columns: "show", "hide", "remove", and "clear". Below it, another table lists "save", "load", "store", and "rename".
- Map View (Bottom Right):** A map of Berlin showing a dense network of roads. The roads are color-coded: blue lines represent the road network, and red lines represent bus routes. The map is titled "speed" and shows coordinates "53426.165615524165 / 36371.134586".
- Search Bar (Bottom Left):** A search input field with a "go" button.

Berlin: Roads + Metro Routes

Secondo GUI (Basic View)

Program Server Optimizer Command Help Viewers File Settings Object Creation Object Close Window

show	hide	remove	clear
save	load	store	rename

```
consume;...successful
see result in object list
Sec:query #routes feed extend[Geo:brgeodata(.Mroute)]
consume;

query #routes feed extend[Geo:brgeodata(.Mroute)]
consume;...successful
see result in object list
Sec:query #routes feed extend[Geo:brgeodata(.Mroute)]
consume;

query #routes feed extend[Geo:brgeodata(.Mroute)]
consume;...successful
see result in object list
Sec>
```

no time 61065.41124780316 / 27323.562390

query #routes feed exten...

```
Mf_id : 1
Mroute : no display functi
Oid : 156479
Geo : sline, true (Defa)
-----
Mf_id : 1
Mroute : no display functi
Oid : 156480
Geo : sline, true (Defa)
-----
Mf_id : 2
Mroute : no display functi
Oid : 156481
Geo : sline, true (Defa)
-----
Mf_id : 2
Mroute : no display functi
Oid : 156482
Geo : sline, true (Defa)
-----
Mf_id : 3
Mroute : no display functi
Oid : 156483
Geo : sline, true (Defa)
-----
Mf_id : 3
Mroute : no display functi
Oid : 156484
Geo : sline, true (Defa)
-----
Mf_id : 4
Mroute : no display functi
Oid : 156485
Geo : sline, true (Defa)
```

search go

Berlin: A Close View (pavements, zebra crossings, bus routes, roads)

The screenshot displays the SecondGIS GUI (Hesse Viewer) with the following components:

- Command Window (Left):** Contains a series of SQL queries and commands for data retrieval and visualization. The queries filter for bus routes on Monday and specific pavements.
- Toolbar (Top):** Includes buttons for 'show', 'hide', 'remove', 'clear', 'save', 'load', 'store', and 'rename'.
- Map Area (Center):** Shows a detailed view of a road network in Berlin. The map features multiple colored lines representing bus routes (red, blue, yellow) and grey areas representing pavements. A zebra crossing is visible on the road.
- Legend (Left of Map):** Lists the objects being displayed, including 'pavement: Default_3' and 'pavement: Default_2'.
- Status Bar (Bottom):** Displays the date and time '2010-12-06-06:34:02.561' and coordinates '27986,42265517943 / 12867,504844'.

```
...successful
see result in object list
Sec>query all_bus_rel feed filter[.bus_day = "Monday"]
      filter[.schedule_id = 1] filter[.br_id = 6] consume ;
query all_bus_rel feed filter[.bus_day = "Monday"]
      filter[.schedule_id = 1] filter[.br_id = 6] consume
...successful
see result in object list
Sec>query subpaves1;
query subpaves1;...successful
see result in object list
Sec>query subpaves2;
query subpaves2;...successful
see result in object list
Sec>
```

query subpaves2;
oid : 13329
rid : 1
pavement : Default_3

oid : 13330
rid : 1
pavement : Default_3

oid : 13331
rid : 1
pavement : Default_3

oid : 13332
rid : 1
pavement : Default_3

oid : 13333
rid : 1
pavement : Default_3

oid : 13334
rid : 1
pavement : Default_3

oid : 13335
rid : 1
pavement : Default_3

oid : 13336
rid : 1
pavement : Default_3

oid : 13337
rid : 2
pavement : Default_3

Method - Experimental Results

The screenshot displays the IndoorViewer application interface. The main window is titled "Secundo GUI (IndoorViewer)". The interface is divided into several sections:

- Command Console (Top Left):** Contains a list of commands and their outputs. The commands include:
 - no result
 - Sec>query fernuni;
 - query fernuni:...successful
 - see result in object list
 - Sec>query trajectory(mo1 feed addcounter[id,1] filter[.id = 198] extract[IndoorTrip])
 - query trajectory(mo1 feed addcounter[id,1] filter[.id = 198] extract[IndoorTrip])..successful
 - see result in object list
 - Sec>query mo1 feed addcounter[id,1] filter[.id = 198]
 - consume;
 - query mo1 feed addcounter[id,1] filter[.id = 198]
 - consume;...successful
 - see result in object list
 - Sec>
- Command Input (Bottom Left):** A search bar with the text "query mo1 feed addco..." and a "go" button.
- 3D Visualization (Center):** A 3D wireframe model of a building floor plan. The walls and rooms are outlined in yellow. A red line indicates a path or trajectory through the building. Dotted lines represent vertical connections between floors.
- Control Panel (Right):** Contains various controls for the 3D view:
 - Buttons for "Translate" and "Rotation".
 - Sliders for X: 0.0, Y: 0.1, and Scale: 2.6.
 - Buttons for "Front View", "L.View", "TOP View", "R.View", and "Back View".
- Command Panel (Top Right):** A table with columns for "show", "hide", "remove", "clear", "save", "load", "store", and "rename".

- Conclusion

We developed a tool called MWGen that can

- 1 create the following environments road network, region-based outdoor, bus network, metro network, and indoor based on roads and floor plans;
- 2 provide trip plannings in one environment and multiple environments;
- 3 generate moving objects with multiple transportation modes based on the result of trip plannings.

- Future Work

- 1 Creating moving objects by considering human movement patterns such as home \leftrightarrow work, work \leftrightarrow work, nearest neighbor searching.

- Transportation Mode Web Page:

<http://dna.fernuni-hagen.de/secondo/TransportationMode/TM.html>

Thank you !