Aggregation and Map-Matching of Mobile Cellular Network Traces

Norbert Goebel   Adrian Skuballa   Martin Mauve   Kalman Graffi

Heinrich Heine University, Düsseldorf, Germany
Computer Science Department
Computer Networks Group

TECHNICAL REPORT TR-2016-001
HEINRICH HEINE UNIVERSITY, DÜSSELDORF, GERMANY
COMPUTER SCIENCE DEPARTMENT

JULY 2016

Aggregation and Map-Matching of Mobile Cellular Network Traces

Norbert Goebel Adrian Skuballa Martin Mauve Kalman Graffi
Department of Computer Science, University of Düsseldorf, Germany
{goebel, mauve, graffi}@cs.uni-duesseldorf.de , adrian.skuballa@uni-duesseldorf.de

***Abstract*—The trace-based simulation of *Vehicle-to-X* (V2X) applications relies on sound traces in a format usable by the simulation framework. However traces normally use gps coordinates and traffic simulators operate on graph based road networks. Furthermore positions gathered using gps are error prone. Thus some post-processing of the traces has to be conducted to map-match and prepare them for the V2X simulation framework. In this paper, we introduce our post-processing techniques used to make traces generated by the *Rate Measurement Framework* (RMF) available to our trace-based simulation environment. Furthermore we present a walk-through usage example of our tool-chain.**

I. Introduction

A fundamentally important requirement for trace-based simulations are sound traces in a format usable by the simulation model. While high quality network traces of high volatile networks can be obtained by various means, they often need to be aggregated to be useful for the simulation model. Network traces for trace-based simulation of V2X communication add the challenge of map-matching, where *Global Positioning System* (GPS) coordinates need to be matched onto a road network graph.

We utilize the RMF [1] to obtain high quality network traces of mobile cellular networks while being on the road.

During each RMF measurement drive, the following data is collected and stored with corresponding timestamps in nanoseconds:

- Available data rate, drop rate and delay for each communication direction with a frequency of up to 5 Hz.
- The position of the measurement vehicle in 1 Hz intervals
- Modem status information regarding the current network condition, like the used network cell sector and the used frequency band, gathered with a frequency of 5 to 10 Hz.

The collected data has to be post-processed before it can be used as a basis for our simulation framework [2] for "Coupled Simulation of Mobile Cellular Networks, Road Traffic and V2X Applications using Traces". The simulation framework requires a graph based simulation database with tuples (e, o, g, t, a, d, l, c). Each tuple contains the **e**dge ID, the **o**ffset on the edge, the **g**roup ID, the measurement **t**ime, the **a**vailable data rate, the backbone **d**elay, the **l**oss probability and an ID representing the used **c**ell sector. Hence a major task is the matching of coordinates in latitude and longitude onto the road network graph, which uses edges and offsets to define positions. Thereby each edge has a start and an endpoint and each point in between can be defined using the offset. As a side condition, the edgeIDs have to be convertible on the fly during the simulation to those used by the traffic simulator *Simulation of Urban Mobility* (SUMO), which has its own naming convention and itself transform *OpenStreetMap* (OSM) XMLs to its format. As SUMO at that point adds many small edges for lane switching, which are not present in the real world, we can not use the SUMO net graph directly for the map-matching process.

At least the following two post-processing steps, which are described in the paper at hand, had to be implemented:

- map-match the error prone GPS positions onto a road network graph
- map the measured network characteristics to positions on the same road network graph

Further challenges solved in the paper at hand are:

- the combination of multiple measurement traces of the same area for the simulation
- finding representatives for measurement groups
- choosing specific network conditions for the simulation by selecting traces which fit user defined network characteristics as closely as possible

We start the remainder of this paper by presenting the related work in Section II, followed by a short summary of the prerequisites for the post-processing in Section III. Further on we explain the map-matching process and the aggregation of multiple traces in Section IV. We show how the map-matched gps-data is transformed into (road-edge, offset) tuples and how a minimal SUMO net-graph containing only edges with measurements is built for the simulation. In Section V we present an example of the usage of the complete post-processing tool-chain. We conclude the paper with Section VI.

II. Related Work

A. OpenStreetMap (OSM)

OpenStreetMap [3] is a free, community driven world map. It offers decent map coverage for most parts of the world. The map data is accessible as map tiles and as XML files, which define the underlying data used for map rendering. The main components of an OSM XML file are defined by *node* and *edge* tags. While a *node* tag can contain many attributes, we just use the unique *id* and the position given as *lat*itude and *lon*gitude. Each *way* tag can store multiple attributes, too, and the most important one is its unique *id*.

```
<node id="2409285517" lat="51.1622775"
   lon="6.8725373"/>
<node id="296343732" lat="51.1622536" lon
   ="6.8726286"/>

<way id="1011096">
  <nd ref="2409285517"/>
  <nd ref="296343732"/>
  <tag k="highway" v="secondary"/>
  <tag k="lanes" v="2"/>
  <tag k="lit" v="yes"/>
  <tag k="lit_by_gaslight" v="no"/>
  <tag k="maxspeed" v="50"/>
  <tag k="name" v="Benrather␣Schlossallee
     "/>
  <tag k="postal_code" v="40597"/>
</way>
```

Listing 1. OpenstreetMap tags example with stripped creationtime, version and usernames.

In addition, each way can have multiple *child* tags, which are either *node (nd)* tags or *key-value* tags. Every way representing a road segment has at least two *nd* tags, which reference *node ids* representing the consecutive course of the road's segments. It optionally stores multiple *key-value* tags defining characteristics like speed limit, number of lanes or street type. An example excerpt of an OSM XML file with only one way consisting of two nodes is shown in Listing 1. We stripped the creation timestamps, version information and user-names in the tags shown, as they are not relevant for the paper at hand.

B. Map-Matching

The process of determining the most probable path a vehicle has traveled on a road network while analyzing a series of GPS positions is called *map-matching*. While GPS positions, as described in [4], are error prone, many use cases like navigation systems rely on the knowledge of the whereabouts of the vehicle on the road network. Thus many different map-matching algorithms were developed in the last two decades. Navigation systems have to determine the current position of the vehicle using an *online* map-matching algorithm. Such algorithms can only use the current and previous GPS measurements for the calculation of the current position on the road network graph. This is the main reason why their estimation is especially error prone on intersections. *Offline* map-matching algorithms, on the other hand, are used after a complete GPS trace has been recorded. They thus have an omniscient view on the complete measurement series and can use all GPS positions for the map-matching of every measurement. This can significantly raise the quality of the estimated vehicle's path on the road network. As conducting the network measurements using the RMF does not rely on map-matched positions on the road network, we focused on *offline* map-matching algorithms. While many offline map-matching algorithms incorporate data from a dead reconning device, the authors in [5] presented an offline map-matching algorithm that only relies on GPS coordinates and a directed graph representing the road network. Their algorithm uses the *mutiple hypothesis technique (MHT)* and works by minimizing the sum of the euclidean distances of all GPS measurements to the estimated routes. Starting with the N closest links of the road network to the first GPS coordinate, it keeps evaluating the N best routes — those with the smallest cumulative errors — while matching additional GPS coordinates. At junctions, additional routes for every outbound edge is added to the list of routes. After a new calculation of the distances, the routes with the highest distances are removed from the list of best routes until only N routes are left. The authors have shown that their algorithm is fast and yields good matching results. Thus we opted for this algorithm in our implementation.

C. Cellular Network Simulation

Introduced in [6], *Trace Based UMTS Simulation* (TBUS) forms the base for our cellular network simulation model. We extended the simulation model with a fair cell share model and implemented it as an *Objective Modular Network Testbed in C++* (OMNeT++) module in [2]. Therein we also presented our complete simulation framework using the *V2X Simulation Runtime Infrastructure* (VSimRTI) as the simulation core, SUMO for traffic simulations, OMNeT++ with our library *libtbus* for the simulation of the cellular network and the *V2X application simulator* (VSimRTI_App). We used the simulation framework with measurements gained with the RMF introduced in [1] and the map-matching process described in the paper at hand to simulate an *Emergency Warning Application* (EWA) using *European Telecommunications Standards Institute* (ETSI) standards and an enhanced, mobile cellular network aware EWA version. Using the simulation results, we were able to show that ETSI-defined timeouts for EWA applications have to be altered for cellular network communications and that an offloading of traffic to the geoserver significantly lowers the bandwidth needed, without sacrificing traffic safety.

III. Prerequisites and Format of the Network Traces

As a prerequisite to our post-processing, at least one network trace has to be gathered using the RMF [1]. Each trace generates four files which are needed as input for the post-processing:

- *cellinfo.txt* contains a consecutive list of timestamped status information of the modem (like used network cell)
- *downstream-data.csv* contains timestamped network characteristics observed in the downstream direction
- *upstream-data.csv* contains timestamped network characteristics observed in the upstream direction
- *gpsd.log* contains logged gps data like position and time

Furthermore, a genuine OSM XML with the mapping data of the regions the measurements were conducted in is needed. In the remainder of this paper we call this file `genuine.osm.xml`.

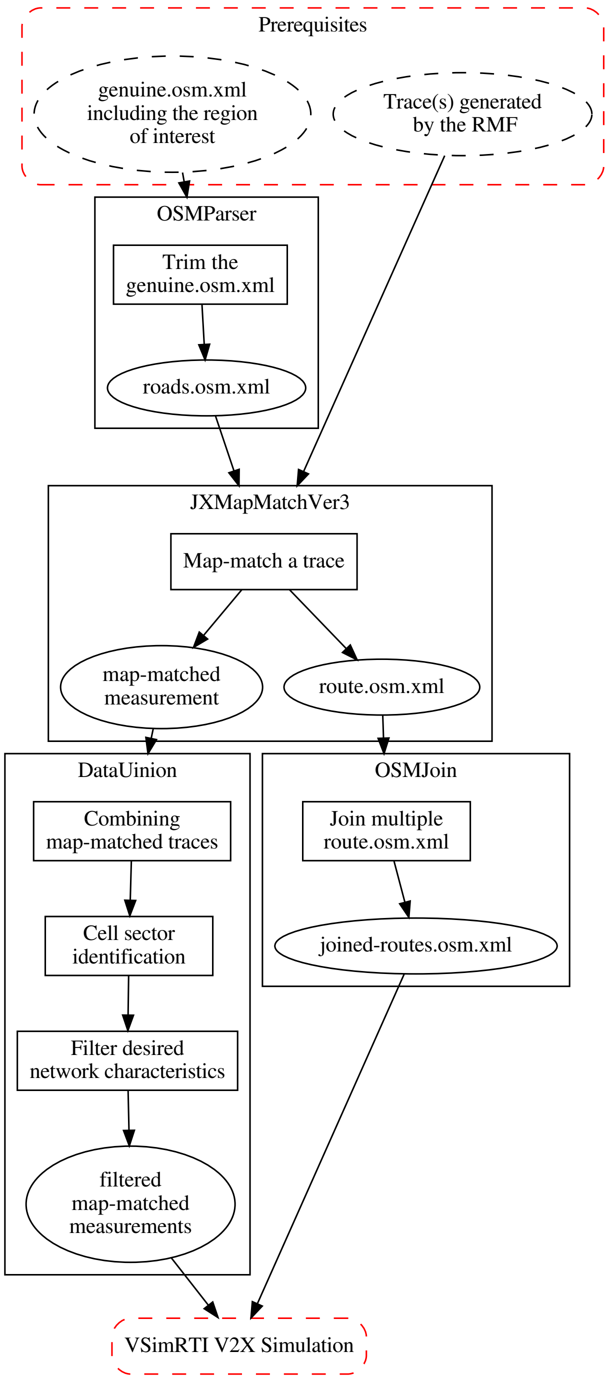

Prerequisites
genuine.osm.xml including the region of interest
Trace(s) generated by the RMF
OSMParser
Trim the genuine.osm.xml
roads.osm.xml
JXMapMatchVer3
Map-match a trace
map-matched measurement
route.osm.xml
DataUinion
Combining map-matched traces
Cell sector identification
Filter desired network characteristics
filtered map-matched measurements
OSMJoin
Join multiple route.osm.xml
joined-routes.osm.xml
VSimRTI V2X Simulation

Figure 1. Overview of the post-processing work-flow.

The software needed for the implementation should run on any Linux or Windows platform. Working *java*, *sqlite3*, *SUMO* and (for the very last step) *VSimRTI* installations are required.

IV. Post-processing Steps

We split the post-processing of our network traces into multiple steps. This significantly reduces the workload if multiple measurements of the same area need to be processed and also allows flexible aggregation or selection of measurements. Figure 1 gives an overview about the multiple steps in our post-processing chain, which we describe in detail in the following subsections. The dashed red lines surrounding the prerequisites and the VSimRTI V2X simulation are not part of the post-processing work-flow. The post-processing itself is split into four major parts with headings naming the corresponding programs developed for the task at hand. Bubbles enclosed by rectangles mark steps in the work-flow. Round bubbles surround output files generated by the previous process.

*A. Trimming the OSM File (*OSMParser*)*

In the first post-processing step, the *OSMParser* is used to shrink a genuine OpenStreetMap XML file (`genuine.osm.xml`) by filtering all information, that is not related to roads and by removing all parts that do not belong to the measurement region or the region of interest. The region of interest hereby is a rectangular region defined by latitude and longitude ranges. These ranges can either be defined by hand through a command line switch, or they are extracted from the *gpsd.log* file(s) of measurements passed to the *OSMParser*. Furthermore, the *OSMParser* performs some operations to assure that a bijective function between OSM coordinates and the (edgeID, offset) tuple in the traffic simulator SUMO exists. As SUMO's *netconvert* merges nodes with the same coordinates, but different nodeIDs into one node, it might create intersections in the SUMO road-graph which are not present in the OSM data. Thus the *OSMParser* searches for nodes with identical latitude/longitude coordinates and increments the latitude of each but the first "duplicate" nodes in steps of 0.00000001 degrees (approximately one millimeter). Additionally, the *OSMParser* splits OSM ways consisting of more than two nodes into separate ways consisting of exactly two nodes to circumvent a length calculation problem of netconvert.

The output generated by the *OSMParser* is an OSM compatible XML-File, which we refer to as `roads.osm.xml`. It only consists of nodes and ways. The nodes have unique nodeIDs and distinct coordinates and are referred to by at least one way. The ways have unique wayIDs, too, and refer to exactly two of these nodes. Each edge is directed. An excerpt of a `roads.osm.xml` with a single way and its two nodes is shown in Listing 2.

The `roads.osm.xml` output of the *OSMParser* then is fed to SUMO's netconvert, which generates a corresponding SUMO road network graph `roads.net.xml` from it. An excerpt showing the SUMO edge generated from the way shown in Listing 2 is shown in Listing 3. It shows that netconvert uses the wayID and the nodeIDs defining the way to build the unique edgeID for SUMO's network graph.

This network graph is used for the traffic simulations by SUMO in the VSimRTI V2X simulations. As the nodeIDs and wayIDs can change in OSM over time (e.g. between releases) `roads.osm.xml` and `roads.net.xml` are only useful if used in conjunction. `roads.net.xml` files generated

```
<node id="1573051002" lat="51.1227132"
      lon="6.7750606"/>
<node id="1836570477" lat="51.1227595"
      lon="6.7750414"/>
<way id="1000596">
  <nd ref="1573051002"/>
  <nd ref="1836570477"/>
  <tag k="highway" v="residential"/>
  <tag k="maxspeed" v="30"/>
  <tag k="name" v="Am␣Schwimmbad"/>
  <tag k="old_way_id" v="23645367"/>
  <tag k="oneway" v="yes"/>
  <tag k="old_oneway" v="yes"/>
</way>
```

Listing 2. Excerpt of a `roads.osm.xml`.

```
<edge id="1000596
    _1573051002_1836570477_1573051002"
   from="1573051002" to="1836570477" priority=
       "-1">
  <lane id="1000596
      _1573051002_1836570477_1573051002_0"
      index="0" speed="8.33" length="4.02"
      shape="2843.76,42.01␣2842.87,45.93"/>
  <lane id="1000596
      _1573051002_1836570477_1573051002_1"
      index="1" speed="8.33" length="4.02"
      shape="2840.55,41.27␣2839.65,45.20"/>
</edge>
```

Listing 3. Excerpt of a `roads.net.xml`.

using a different `roads.osm.xml` are not compatible for the ongoing map-matching and simulation process. But if the measurement and simulation area does not change, the generated `roads.osm.xml` and `roads.net.xml` files can be used for multiple map-matching processes and simulations even if different traces are used.

The significant size reduction of the `genuine.osm.xml` file reduces loading and execution time of the actual map-matching process and the simulation.

B. *Map-Matching (*JXMapMatchVer3*)*

Given that a network trace and corresponding `roads.osm.xml` and `roads.net.xml` files exist, the second post-processing step, the actual map-matching, can be carried out. We developed a semi automatic graphical Java Application called *JXMapMatch* to do the actual map-matching. In a first step, it map-matches every collected GPS position (*gpsd.log*) onto the road network graph described by the `roads.osm.xml` using a modified version of the algorithm introduced in [5] and explained in Section II. After the algorithm has calculated the most probable route taken by the measurement vehicle, the graphical editor allows manual changes of the route. This enables an easy correction of map-matching errors caused by bad gps accuracy or faulty mapping data. Finally, the actual measurement data is matched onto the road network graph by interpolating the position based on the timestamps of each measurement

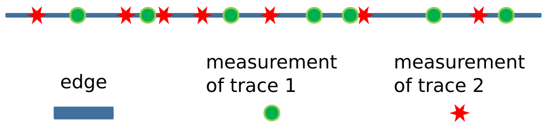


Figure 2. Example of an edge that is passed two times.

and the map-matched route taken by the vehicle. Each map-matching process using *JXMapMatch* generates multiple output files:

1) multiple KML files are generated, which allow the easy visualization of the measurements using Google earth [7]
2) map-matched measurements (`map-matched.csv`)
3) an even smaller version of the `roads.osm.xml` file containing only the roads used in the map-matched traces is generated, which we refer to as `route.osm.xml`

C. *Identifying Network Cell Sectors and Combining Multiple Measurements (*DataUnion*)*

The third post processing step, implemented as *DataUnion*, addresses two challenges:
1) it identifies the different cell sectors of the mobile cellular network
2) and it allows the combination of multiple measurements

1) Identifying Network Cell Sectors: For the trace-based simulation it is important to know the cellular network cell sector the tracing vehicle was communicating with at a specific time. As this information is not readily available, it must be derived from the logged modem information. The RMF logs the *Location Area Code (LAC)* and the *CellID* when communicating with a 2G cellular network cell and, when using a 3G network connection, the *Channel* and the *Primary Scrambling Code (PSC)*. On the one hand, two neighboring network cells must not use the same settings, as their communication would interfere. On the other hand, the number of combinations of LACs and CellIDs respectively channels and PSCs is limited. Thus multiple cells with the same settings exist. *DataUnion* therefore first groups measurements with the same connection settings. Afterwards it checks the distances of the measurements in any group and forms subgroups if measurements are farer apart than a parameterized distance. We set the default for this distance to 5 km as mobile cellular network cells can be quite large and the 5 km distance worked well in all our traces. Each such subgroup is assigned a unique CellID by *DataUnion*, which is associated with the corresponding mobile cellular network characteristics.

2) Combining Multiple Measurements: Often the map-matched measurements of multiple traces need to be combined to enlarge the simulation area, whereas in other situations it is desirable to aggregate multiple measurements of the same area.

Figure 2 sketches an example situation where an edge is passed two times during measurements. Merging the traces of Figure 2 based on their offset on the edge would lead to the sequence 21212212112121. Imagine two different situations for the measurements where the first pass was conducted during rush hour and the second pass was made in the middle of the night. Obviously, the measured network characteristics of these traces can deviate largely. As a consequence, multiple switches between extremely different network characteristics are possible. This can have a significant influence on the simulation outcome and lead to unrealistic network simulations. Hence *DataUnion* instead allocates a unique *groupID* g to every measurement. Thereby the groupID g is equal for all measurements mapped to the same edge and belonging to the same passing of the edge. For every network measurement the post-processing thus results in the desired tuple (e, o, g, t, a, d, l, c) required by our simulation model.

*D. Selecting desired network characteristics (*DataUnion*)*

A commonly asked question when analyzing algorithms target the best, average and worst case scenarios. Defining equivalent realistic cases for cellular network communication simulations is quite challenging as the same thoughts disallowing easy combination of multiple measurements have to be applied, too. Our solution for this is a two step filter option build into *DataUnion*. It allows the selection of a specific groupID per edge for the simulation. In the first step, for each group of measurements sharing the same groupID, a representative value either for the delay, the available data rate or the loss rate is selected as:

- **min**: the minimal value of the group
- **max**: the maximal value of the group
- **avg**: the average of the group
- **med**: the median of the group
- **minstde**: the maximum of "the average value subtracted by the standard deviation" and the "min" value
- **maxstde**: the minimum of "the sum of the average value and the standard deviation" and the "max" value
- **avgstde**: the arithmetic average of all values within the range of minstde and maxstde. We specifically added this option to minimize the impact of measurement outliers as often only 3 to 10 measurements are forming a value group.
- **ww**: the way weighted average of the value group. In the simulation, a measurement is used from its measured position until a position with another measurement is passed. Thus this representative selection weights the measurements depending on the length their value is valid on their edge.

In the second step, a single groupID per edge is selected based on the representative and one of the following, selectable algorithms:

- **all**: no filtering is performed
- **med-**: the groupID with the median representative value is chosen for each edge. If two medians exist, the first is chosen.
- **med+**: the groupID with the median representative value is chosen for each edge. If two medians exist, the second is chosen.

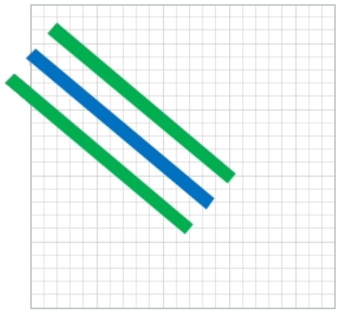


Figure 3. SUMO creating negative coordinates for one lane (green) from the blue edge.

- $p \in [0, 1]$: the groupID with the representative closest to $minrep + p \cdot (maxrep - minrep)$ is chosen.

After these two filter steps exactly one groupID is chosen per edge as long as not "all" is selected. All generated tuples (e, o, g, t, a, d, l, c) are stored in the graph-based simulation database. As switches between traces using this algorithm only occur when changing edges, it is possible to choose the network characteristics that are most interesting for the desired simulation scenario, but still retain realistic network simulations.

*E. Joining multiple `route.osm.xml` files (*OSMJoin*)*

While the measurements itself are combined and aggregated by *DataUnion*, the `route.osm.xml` files corresponding to these traces need to be merged, too, as the traffic simulator SUMO needs to know the ways and nodes of all routes taken by any of the combined traces. We thus implemented *OSMJoin*, which allows the joining of multiple `route.osm.xml` files while keeping the same nodeIDs and wayIDs used in the `roads.net.xml` used by SUMO. Furthermore, *OSMJoin* helps to avoid a simulation bug caused by SUMO, which transforms the geographical coordinates of the given `roads.net.xml` to its own coordinate system consisting of positive numbers only. During the creation of the internal route graph, SUMO converts OSM ways with multiple lanes into multiple parallel ways with one lane. This, as shown in Figure 3, can lead to negative coordinates, resulting in severe problems during the VSimRTI simulation. Figure 4 shows how *OSMJoin* solves this problem by adding two single lane horizontal ways above and below the area defined by the given `route.osm.xml` file(s). These two ways are not connected to any other way and the coordinates of the defining nodes are lower/higher than the coordinates of all nodes. The used offset is parameterized. This solves the negative coordinates problem in the simulation and also assures that the two new ways are not used during the simulation, as each route in SUMO needs to be at least two ways long.

Afterwards, the `joined-routes.osm.xml` is used by netconvert to create the SUMO road network graph `joined-routes.osm.xml`.

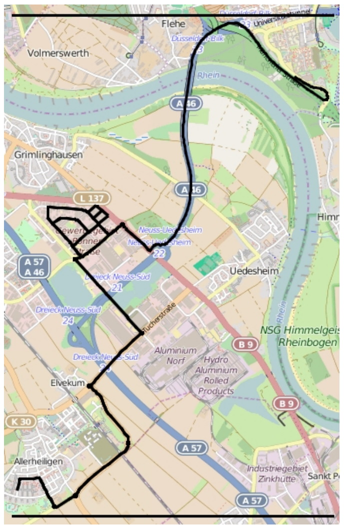


Figure 4. OsmJoin generated minimal OSM graph with top and bottom single lane ways (Map ©OpenStreetMap [3])

V. Exemplary Workflow

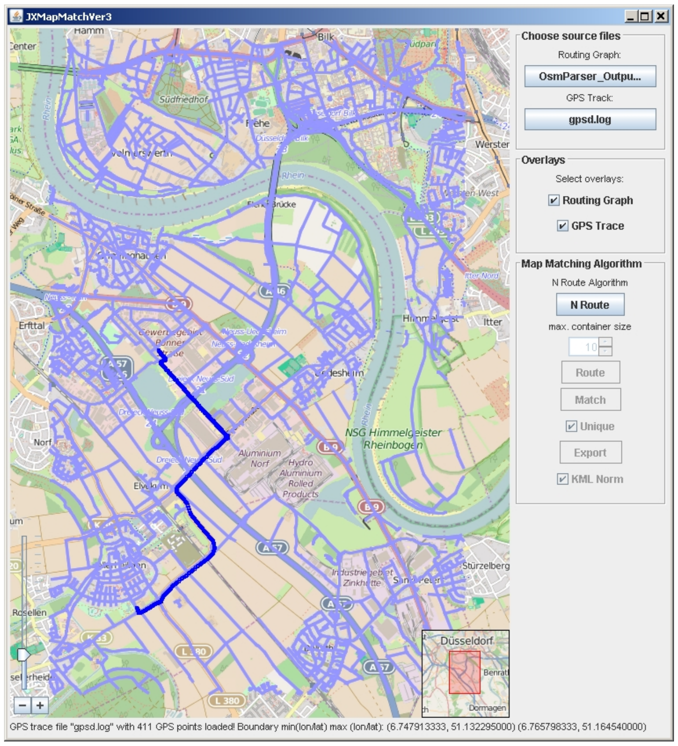
In this section we show an exemplary workflow of our post-processing tool-chain, which is available for download at github [1,2,3,4]. All listings used in this section show single line commands executed in a Linux bash shell. To enhance readability we manually split the commands into multiple lines — one line for each parameter. We also assume that the called program either resides in the current directory or in the path. Furthermore, the input files are accessible in subdirectories to the current working directory in our examples following a naming scheme of *YYYYMMDD/HHMMSS*. In the shown examples we combine four traces conducted on 3rd August 2015.

A. OSMParser and netconvert

The OSMParser is used to shrink a `genuine.osm.xml` file by deleting all non-road information and by constraining to the area of interest. In the following example the OSMParser parses a genuine OSM XML file (Parameter *-oi*) of the City of Düsseldorf and its surroundings. The name of the output file is defined by *-oo* and all gpsd.log files preceded by the *-g* parameter are used to constrain the area of interest to the rectangle formed by (minlon,minlat) and (maxlon,maxlat) contained in the gps logs.

```
java OSMParser
    -oi duesseldorf-regbez-latest.osm.xml
    -oo roads.osm.xml
    -g 20150803/090007/gpsd.log
    -g 20150803/091227/gpsd.log
    -g 20150803/092532/gpsd.log
    -g 20150803/093139/gpsd.log
```

[1]https://github.com/hhucn/OSMParser
[2]https://github.com/hhucn/JXMapMatchV3
[3]https://github.com/hhucn/OSMJoin
[4]https://github.com/hhucn/DataUnion



GPS trace file "gpsd.log" with 411 GPS points loaded! Boundary min(lon/lat) max (lon/lat): (6.747913333, 51.132295000) (6.765798333, 51.164540000)

Figure 5. Screenshot of JXMapMatchVer3 (Map ©OpenStreetMap [3]).

Even though the multi-node ways of the genuine OSM XML file (303MB in this example) are split to multiple two-node ways in the output file, the output file is significantly smaller (<3MB).

As stated in Section IV-A, the generated `roads.osm.xml` needs to be converted to a `roads.net.xml`. This can be accomplished by calling netconvert:

```
netconvert
  --osm-files roads.osm.xml
  -o roads.net.xml
```

B. *Map-Matching with* JXMapMatchVer3

In Section IV-B, we explained how the map-matching process works. As *JXMapMatchVer3* is a GUI-based application, no command line parameters are needed. Thus we give a short description of the workflow using the screenshot shown in Figure 5. The GUI is split into a MapView on the left and a navigation pane on the right. The MapView shows OpenStreetMap Map Tiles in the background and (when `roads.osm.xml`, `roads.net.xml` and a gpsd.log are loaded) the road network graph (lilac) and the logged gps positions (blue).

C. *Aggregating map-matched measurements with* DataUnion

After several traces have been map-matched by *JXMapMatchVer3* using identical `roads.osm.xml` and `roads.net.xml` files, their traces can be combined by *DataUnion*. As explained in Section IV-D, the data can be combined leaving multiple groupIDs per edge. An exemplary command is:

```
java DataUnion
    -o combined-map-matched.csv
    -g all
    -i 20150803/090007/map-matched.csv
    -i 20150803/091227/map-matched.csv
    -i 20150803/092532/map-matched.csv
    -i 20150803/093139/map-matched.csv
```

The aggregation of the same traces with active filtering could look like:

```
java DataUnion
    -o filtered-map-matched.csv
    -g med-
    -t datarate
    -r ww
    -i 20150803/090007/map-matched.csv
    -i 20150803/091227/map-matched.csv
    -i 20150803/092532/map-matched.csv
    -i 20150803/093139/map-matched.csv
```

In this example, for each edgeID with multiple assigned groupIDs only one groupID is saved in the `filtered-map-matched.csv` output. The selection bases on the data rates. The representative for each group is the way weighted average of the group. The group with the lower median is selected as the only group for this edge in the filtered output.

D. Joining multiple `route.osm.xml` by OsmJoin

A usage example for *OsmJoin*, which combines the routes of four traces to `joined-routes.osm.xml` is:

```
java OsmJoin
    -o joined-routes.osm.xml
    -i 20150803/090007/route.osm.xml
    -i 20150803/091227/route.osm.xml
    -i 20150803/092532/route.osm.xml
    -i 20150803/093139/route.osm.xml
```

E. Generate Necessary Files for the Simulator

Finally, the aggregated measurements need to be converted to a sqlite database which is used by the OMNeT++ module of the simulation framework. Let the measurements be contained in `filtered-map-matched.csv`, then the *sqlite* database can be created with the commands:

```
awk --field-separator="," 'BEGIN{i=0} /down/ {
    print i "," $24 "," $3 "," $11 "," $12 ","
     $23 "," $15 "," $17 "," $16; i=i+1}'
    filtered-map-matched.csv > filtered-
    download.csv
awk --field-separator="," 'BEGIN{i=0} /up/ {
    print i "," $24 "," $3 "," $11 "," $12 ","
     $23 "," $15 "," $17 "," $16; i=i+1}'
    filtered-map-matched.csv > filtered-upload
    .csv
cat << EOF | sqlite3 test_edge.sqlite
pragma user_version=1;
.separator ,
.import filtered-download.csv download
.import filtered-upload.csv upload
EOF
```

Furthermore, the `joined-routes.osm.xml` needs to be prepared for SUMO and VSimRTI, which we use for our V2X simulations in [2]. For the generation of the SUMO net file we use netconvert with the following parameters:

```
netconvert
  --osm-files joined-routes.osm.xml
  -o joined-routes.net.xml
```

The conversion for VSimRTI is performed by *scenario-convert*, which is distributed with VSimRTI. We used VSimRTI Version 0.15 for our simulations, and thus used the following command to create the database:

```
java -jar scenario-convert-0.15.0.jar
    --osm2sumo
    -d joined-routes.db
    -i joined-routes.osm.xml
    -n
```

VI. Conclusion

In this paper we introduced the methods used to post-process our position-based mobile cellular network traces generated using the RMF. The post-processed output builds the simulation basis for our V2X simulation framework presented in [2]. We explained how the conversion from gps positions to positions in a road network graph works and introduced our map-matching implementation. Furthermore, we introduced our aggregation approach for multiple traces by a multitude of filtering options. In Section V we presented a walk through example of the complete tool chain. The complete post-processing tool chain presented in the paper at hand is available as sources (see Section IV) under MIT licenses. The results of the post-processing are already actively used in V2X simulations with the use of VSimRTI and our OMNeT++ module presented in [2].

References

[1] N. Goebel, T. Krauthoff, K. Graffi, and M. Mauve, "Moving Measurements: Measuring Network Characteristics of Mobile Cellular Networks on the Move," in *2015 IEEE Vehicular Networking Conference (VNC) (IEEE VNC 2015)*, Kyoto, Japan, Dec. 2015.

[2] N. Goebel, R. Bialon, M. Mauve, and K. Graffi, "Coupled simulation of mobile cellular networks, road traffic and V2X applications using traces," in *IEEE ICC 2016 - Mobile and Wireless Networking Symposium (ICC'16 MWN)*, Kuala Lumpur, Malaysia, May 2016.

[3] OpenStreetMap, "OpenStreetMap," [Accessed Feb. 14th 2016], Copyright and License: www.openstreetmap.org/copyright. [Online]. Available: http://www.openstreetmap.org/

[4] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 853–864. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083592.1083691

[5] F. Marchal, J. Hackney, and K. W. Axhausen, "Efficient Map-Matching of Large GPS Data Sets - Tests on a Speed Monitoring Experiment in Zurich, volume 244 of Arbeitsbericht Verkehrs und Raumplanung," Tech. Rep., 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/citations?doi=10.1.1.113.4605

[6] N. Goebel, M. Koegel, M. Mauve, and K. Graffi, "Trace-based simulation of c2x-communication using cellular networks," in *11th Annual Conference on Wireless On-demand Network Systems and Services (WONS 2014) - Special Session on VANETs and ITS3*, Apr. 2014, pp. 108–115. [Online]. Available: http://dx.doi.org/10.1109/WONS.2014.6814730

[7] Google, "Google earth," August 2007. [Online]. Available: http://earth.google.de/