Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 50 – 57

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Data Preparation to Simulate Public Transport in Micro-Simulations using OSM and GTFS

Glenn Cich[a,*], Jan Vuurstaek[a], Luk Knapen[a], Ansar-Ul-Haque Yasar[a], Tom Bellemans[a], Davy Janssens[a]

[a]*Transportation Research Institute (IMOB), Hasselt University, Wetenschapspark 5 bus 6, 3590 Diepenbeek, Belgium*

Abstract

Research on demand-responsive collective transportation facilities that can act as feeder services to time-table based public transportation (PT) requires detailed and accurate information about the PT infrastructure, including the attachment of bus stops to the appropriate network link. Due to the size of the infrastructure, the data integration shall be automated. This paper describes the effort to prepare data from publicly available *OpenStreetMap* (OSM) and *General Transit Feed Specification* (GTFS) sources. Procedures are proposed (i) to build a network derived from OSM suitable for simulations in transportation, (ii) to extract bus stops from GTFS and remove anomalies and (iii) to find candidate network links to attach them.

Keywords: OpenStreetMap, General Transit Feed Specification, Micro-Simulation, Data Cleaning, Public Transport

1. Problem Context

A *flow* in a transportation network is the set of passages from a source S to a target T using one of the available paths connecting S to T and for which the travel period overlaps a given set of periods (e.g. every Tuesday morning between 07:30h and 09:30h). *Thin flows* consist of small amounts of passages. Serving them by time-table based public transportation (TTB-PT) can be costly or lead to under-used services. One of the research objectives of the Smart-PT project is to determine the viability of companies offering mini-bus based collective transportation on-demand as a replacement for TTB-PT. Such services are expected to partly act as feeder services to TTB-PT. The stochastic nature of the demand and the small capacity of the provided services require daily optimization of the driven routes. The result is highly sensitive to small variations in the demand and in the characteristics of the local situation. Hence, an accurate high resolution representation of the TTB-PT is required. Furthermore, the use of open and recent data is preferred. Therefore, an automated tool to integrate *OpenStreetMap* (*OSM*) derived transportation networks and *General Transit Feed Specification* (*GTFS*) is required. This paper discusses the first stage of an integration effort.

[*] Corresponding author. Tel.: +32-11-269-111 ; fax: +32-11-269-199.
E-mail address: glenn.cich@uhasselt.be

doi:10.1016/j.procs.2016.04.098

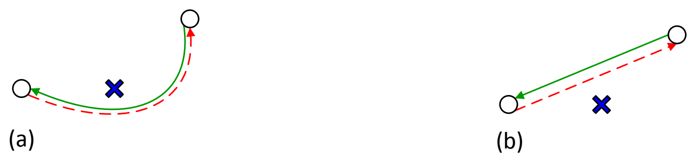

Fig. 1: (a) Situation with geometry: bus stop is connected to the (correct) solid green link. (b) Situation where the geometry is replaced by a straight line segment: bus stop is connected to the (wrong) dashed red link.

Figure 1 shows that geometrically complete and accurate data is required to attach a bus stop to the correct side of the road. On the other hand, the resulting network shall be as simple as possible because the algorithm that assigns bus stops to network links is based on combinatorial optimization.

This paper is organized in the following way: in Section 2, we give an overview of the related work. Sections 3 and 4 are organized in the same way: first terminology is discussed, then the used algorithms followed by the results. Section 3 discusses OSM and Section 4 discusses GTFS. In Section 5, an algorithm is described that is able to find candidate locations for GTFS bus stops. Finally, we draw a conclusion and briefly discuss future work.

2. Related Work

In[1], Zilske et al. describe a process in order to use OSM in the *micro-simulator* MATSim. They mention issues related to getting high quality input data (in this case maps) to use in MATSim. Maps are in different (non-standard) formats, difficult to get and in most cases not referenced to each other. The authors converted an OSM dataset to a MATSim compatible input format and attempted to integrate OSM and GTFS in order to simulate public transport.

In[2,3], the authors describe methods in order to assess the quality of the OSM network. Eight quality indicators are discussed: geometric/positional accuracy, attribute accuracy, completeness, logical consistency, semantic accuracy, temporal accuracy, lineage and usage. The quality indicator in which we are interested for this paper is the attribute accuracy. It describes the accuracy/correctness of the attributes in OSM. In[2], Haklay conducts an analysis of positional accuracy and the completeness of the OSM dataset. In order to analyze this, the Ordnance Survey for the region of London, UK is used. In[3], Girres and Touya assess the attribute accuracy by studying the matching between lake names in the region of l'Alpes d'Huez. They noticed that only 55% of the lake names are as informed as their base truth. However, when OSM describes a lake name, there is a nearly identical matching. Note that for such methods a base truth is required.

In[4], Mooney and Corcoran describe the annotation process in OSM. The main issue in the annotation process is the lack of discipline and automatic checking with respect to defining attribute names and with respect to assigning attributes to objects. Contributors can specify an unlimited amount of *tag* elements and there are no context restrictions regarding the attribute values of those *tags*. The authors studied (i) the assignment of attribute values to *tag* elements, (ii) the type of contribution by the contributors and (iii) the use of the OSM Map Features page.

In[5], Barron et al. attempt to assess the quality of OSM without the use of any base truth. They use the OSM-Full-History-Dump in order to assess the quality. They propose a framework consisting of a set of tools to assess the quality of several OSM characteristics. For example, they attempt to assess completeness of the road network by comparing the evolution in link lengths. When the length of the links stabilizes, they assume that the links of that area are finished. In contrast, when the link lengths change a lot, it means that the links are not close to completion.

Our paper focuses on data cleaning as a prerequisite in the process of OSM-GTFS integration.

3. OpenStreetMap

3.1. Terminology

In order to conduct the cleaning and preparation steps, the OSM data (which is described in `XML`) is read into `Java` classes. We assume the reader is familiar with the OSM terminology. The `Java` classes we created are (i) *GeneralPoint*, (ii) *GeneralLink*, (iii) *GeneralTransportInfrastructure* and (iv) *GeneralRoad*.

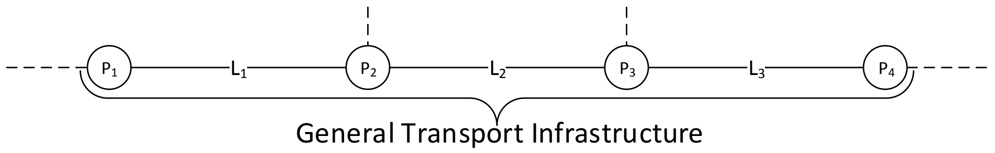

Fig. 2: A representation of the terminology, with P_* the GeneralPoints and L_* the GeneralLinks of the presented GeneralTransportInfrastructure.

Definition 3.1. A *GeneralPoint* represents a node in the OSM dataset. It merely defines the shape of a GeneralLink.

Definition 3.2. A *GeneralLink* represents a link which has no one-on-one mapping with the OSM dataset. A GeneralLink consists of n GeneralPoints where $n \geq 2$, because every GeneralLink needs at least a start and end GeneralPoint. GeneralLinks can only meet each other in the start and/or end GeneralPoints, which means that a GeneralLink does not have any junctions in the intermediate GeneralPoints that serve to define the road geometry.

Definition 3.3. A *GeneralTransportInfrastructure* is the base class for a GeneralRoad. It holds a hash map of attributes (key-value pairs) and a list of GeneralLinks. A GeneralTransportInfrastructure consists of n GeneralLinks where $n \geq 1$. The hash map of attributes represents the attribute values of tags in the OSM dataset.

Definition 3.4. A *GeneralRoad* is a class, extended of GeneralTransportInfrastructure, which represents a way with the tag "highway" in the OSM dataset. While a GeneralTransportInfrastructure has general functionality, a GeneralRoad has specific functionality, such as information about the number of lanes, the speed limits, direction etc.

A representation of the definitions can be seen in Figure 2. Note that we are also able to extend this software for other transportation infrastructures such as railways.

3.2. Algorithms

In order to combine the OSM data with a GTFS dataset, we will *clean*, *reduce* and *prepare* the OSM data.

Definition 3.5 (cleaning, reduction, preparation)**.** *Cleaning* is the process in which we will *enrich*, *correct* and *complete* OSM objects. *Reduction* is the process in which OSM objects that are not useful for any purpose will be removed. *Preparation* is the process in which application specific actions are performed.

3.2.1. Cleaning

The first cleaning step is resetting inconsistent individual and group values to predefined values. An invalid *individual* value is inconsistent with its range specification. Straightforward rules are developed such as "total number of lanes > 0". Detecting inconsistent *group* values is done by comparing values; these rules are more complex such as "total number of lanes = number of forward lanes + number of backward lanes".

A second cleaning step is *auto completing* OSM tags using predefined rules. This is done based on existing OSM tags. For example when "total number of lanes = 4", "number of backward lanes = 2" and "number of forward lanes = UNDEFINED", we can assume that the "number of forward lanes = 2".

A third cleaning step (also enriching OSM tags) is adding and completing existing tags using a set of rules. Note that these rules are country dependent. In Belgium, the maximum speed of a "primary" road is 120[km/h]. When there is a road with type "primary" and "max speed = UNDEFINED" we assign 120 to the max speed. Due to lack of space it is not possible to exhaustively list the set of rules used in the first three steps.

A fourth cleaning step is removing GeneralLinks with a length of zero while maintaining the topology of the network.

A fifth cleaning step first merges GeneralTransportInfrastructures and then GeneralLinks that are separated by *useless* splits. Such useless splits can be generated by previous cleaning steps or by mistakes during data entry.

Definition 3.6. A *useless split of a GeneralTransportInfrastructure* is a GeneralPoint which has exactly two GeneralLinks connected to it, where these two GeneralLinks are in two separate GeneralTransportInfrastructures that have the same attributes, but a different list of GeneralLinks.

The condition stating *exactly two GeneralLinks* is sufficient because all GeneralLinks in a GeneralTransportInfrastructure need to have identical attribute values, including the direction.

Definition 3.7. A *useless split of a GeneralLink* is a GeneralPoint that has exactly two GeneralLinks (belonging to the same GeneralTransportInfrastructure) connected to it.

In fact there is no reason for two GeneralTransportInfrastructures having identical attributes to be separated by a split. Note that it is very important that we first merge GeneralTransportInfrastructures and then GeneralLinks. Otherwise, we might introduce new useless splits of GeneralLinks while merging GeneralTransportInfrastructures.

3.2.2. Reduction

The first reduction step is removing road types which we do not want in the output dataset. The cleaning tool allows the user to specify (i) a list of types to keep in the dataset and (ii) a list of types to drop from the dataset.

It is impossible to specify every road type that exists in the OSM dataset due to the lack of attribute key and value validation as mentioned in Section 2. That is why we implemented a feature that will convert every type that is not contained in one of both lists to the “unclassified” type. This is done in order to avoid dropping road types that might be useful.

Another reduction is done by removing *sinks*, *sources*, *black holes*, *white holes* and *islands* in order to ensure that the transportation network constitutes a *strongly connected graph*.

Definition 3.8 (sink, source)**.** A node is a *sink* if it has no outgoing edges, i.e. you can enter it, but cannot leave it. A node is a *source* if it has no incoming edges, i.e. you can leave it, but cannot enter it.

Definition 3.9 (black hole, white hole)**.** A sub network is called a *black hole* if you can enter this sub network, but cannot leave it. A sub network is called a *white hole* if you can leave this sub network, but cannot enter it.

Definition 3.10. A sub network is called an *island* if you cannot enter and leave this sub network.

It was observed by interactive visual inspection that the part of the network constituting sources, sinks, black holes, white holes and islands represents a small amount of road segments consisting mainly of walking roads. Therefore, it was judged that dropping those parts is a justified solution.

3.2.3. Preparation

As discussed in Section 1, the goal of this data preparation is connecting GTFS stops to the OSM network. Note that we also need to take into account the side of the road. Therefor, we split every GeneralTransportInfrastructure with direction BOTH into two identical GeneralTransportInfrastructures, one with direction FORWARD and one with direction BACKWARD. By doing this, candidate locations for GTFS stops can be assigned separately on both sides of the road.

3.3. Results

For the experiments we used the part of the OSM network delimited by the minimal bounding box that contains all the bus stops of “De Lijn” (PT provider for buses and trams of Flanders) which includes the northern part of Belgium and the southern part of the Netherlands. The following pipeline of steps is used: (i) Remove GeneralRoads which are not needed, (ii) Change type of GeneralRoads which do not occur in one of the two lists (iii) reset incorrect individual values, (iv) auto complete GeneralRoads, (v) reset incorrect individual values, (vi) reset incorrect group values, (vii) enrich GeneralRoads with rules, (viii) remove GeneralLinks with zero length, (ix) merge GeneralTransportInfrastructures, (x) merge GeneralLinks, (xi) remove GeneralLinks/GeneralTransportInfrastructures not belonging to the strongly connected graph and (xii) convert GeneralTransportInfrastructures with direction BOTH into a FORWARD and BACKWARD GeneralTransportInfrastructure. Note that we reset incorrect values twice because we might introduce new mistakes when we auto complete the data. Suppose that “total number of lanes = 2”, “total number of forward lanes = 4” and “total number of backward lanes = UNDEFINED”; in this case the individual values are correct. However, if we auto complete the backward lanes, it will result in -2, which is an incorrect individual value. In Table 1 an overview of the different steps is given. The main reduction in amount of objects happens in Step (i) in which GeneralTransportInfrastructures are deleted which are not needed. Other decent reductions happen in Step (ix) and (x) in which GeneralTransportInfrastructures and GeneralLinks are merged. At the end of Step (xi) the amounts of GeneralTransportInfrastructures, GeneralLinks and GeneralPoints are reduced by respectively 25.28%, 32.60% and 28.54%. Finally, in Step (xii) there is a large increase in both GeneralTransportInfrastructures and GeneralLinks due to the conversion of GeneralTransportInfrastructures with direction BOTH to FORWARD and BACKWARD.

Table 1: Results of the OSM data preparation pipeline. The symbol "#" represents the amount of remaining objects and the symbol "δ" represents the amount of modified objects. GeneralTransportInfrastructures is abbreviated as GTI, GeneralLinks as GL and GeneralPoints as GP.

Description	Step	#GTI	#GL	#GP	δGTI
Initial	/	776 483	1 336 260	969 907	/
Remove not needed GeneralRoads	(i)	612 402	1 043 840	837 735	0
Change type of "unknown" GeneralRoads	(ii)	612 402	1 043 840	837 735	67 989
Reset incorrect individual values	(iii)	612 402	1 043 840	837 735	0
Auto complete GeneralRoads	(iv)	612 402	1 043 840	837 735	136 071
Reset incorrect individual values	(v)	612 402	1 043 840	837 735	0
Reset incorrect group values	(vi)	612 402	1 043 840	837 735	571 310
Enrich GeneralRoads with rules	(vii)	612 402	1 043 840	837 735	612 402
Remove GeneralLinks with zero length	(viii)	612 402	1 043 815	837 731	0
Merge GeneralTransportInfrastructures	(ix)	580 211	1 043 815	837 731	0
Merge GeneralLinks	(x)	580 211	905 397	699 313	0
Remove objects not belonging to the strongly connected graph	(xi)	580 211	900 702	693 068	0
Convert BOTH to FORWARD and BACKWARD	(xii)	1 132 382	1 764 648	693 068	0

4. General Transit Feed Specification

4.1. Terminology

We will not describe the GTFS dataset in this paper. For more information we refer to `https://developers.google.com/transit/gtfs/`.

4.2. Algorithms

4.2.1. Unresolved References Removal - Simplification

The GTFS dataset uses files which are connected to each other by identifiers. Hence, there can be missing links when an identifier occurs in one file but not in the other files. Data records containing *unresolved references* are deleted from the dataset.

The data preparation covered by this section is used in order to be able to connect GTFS stops to the OSM network. Hence, duplicate trips (= trips which serve the same stops in exactly the same order) are useless because they need to be processed multiple times. We decided to delete these duplicate trips because we are not interested in the time dimension (only in the sequence of stops). This reduction will decrease the complexity of the OSM-GTFS integration.

While analyzing the GTFS data, we encountered situations where the same GTFS stop is served multiple times, separated, in the majority of the cases, by a short interval (in most of the cases one minute). We decided to delete those stops because this will only make the OSM-GTFS integration more complex, without having an influence on the actual results of the assignments.

4.2.2. Trips Containing Stop Visits in Inconsistent Orders

Each stop has a unique identifier *id* and a *name* (in most but not all cases a combination of municipality and street name). If two stops share the same name, they are on the same bidirectional network GeneralLink (road segment) at opposite sides of the street (and hence, they correspond to opposite GeneralLink travel directions). As a consequence, each stop is uniquely identified by an *id*, but also by a tuple $\langle name, dir\rangle$ where $dir \in \{FORWARD, BACKWARD\}$ and

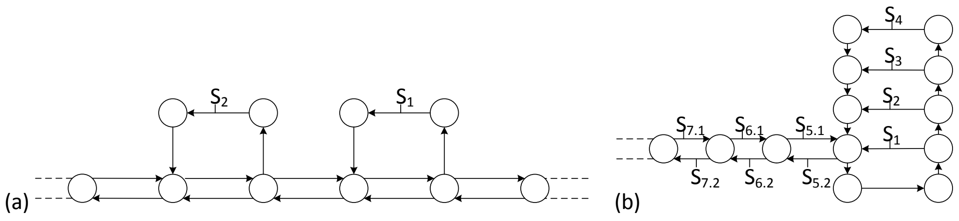

Fig. 3: Two parts of a network: the circles represent road junctions, the arrows represent lanes and the symbols S_i and $S_{i,j}$ identify stops; the location on the GeneralLink is denoted by the small stroke perpendicular to the GeneralLink. (a) A real case where stops S_1 and S_2 are visited in both orders. The trip moving from the left to the right contains two loops. (b) The example explained in the text.

name identifies a network GeneralLink. People can board and alight a vehicle at only one side (in continental Europe the right hand side).

Assume two stops S_1 and S_2 so that both $\langle S_1, S_2\rangle$ and $\langle S_2, S_1\rangle$ are subsequences of some, not necessarily different, trips (i.e. the stops are used consecutively and in both orders). Cases where S_1 and S_2 are used in both sequence orders but not as consecutive stops in a trip are not considered. The GeneralLinks corresponding to the respective stops need to be crossed in the same direction in all cases (because of the boarding and alighting side restriction).

In case S_1 and S_2 share the same name, they correspond to a single GeneralLink and to opposite directions (because they are different stops). Both $\langle S_1, S_2\rangle$ and $\langle S_2, S_1\rangle$ imply a U-turn in the trip (which is possible).

In case S_1 and S_2 have different names and none of both names are shared names, it is sure that in all cases exactly the same stops are used (since none of the stops has a counterpart at the opposite side of the street). This case is shown in Figure 3a. This case was found in reality and necessarily induces a cycle in the route.

Finally, if the stops have different names and at least one of the names is a shared name, then it is assumed that an error was made while creating the GTFS database and a *voting* concept is used to select one of the cases to be kept. In this case the number of trips for both occurrences $\langle S_1, S_2\rangle$ and $\langle S_2, S_1\rangle$ are counted. The case that is found in the majority of trips is assumed to be the right one. The other trips are corrected by swapping the "wrong" tuple. The reduction step will be explained using the synthetic example shown in Figure 3b. For that case the *shared stop name* assumption means that exactly one name is associated with each of the pairs $\{S_{5.1}, S_{5.2}\}$, $\{S_{6.1}, S_{6.2}\}$ and $\{S_{7.1}, S_{7.2}\}$. Suppose we have the following four trips: $\{S_4 - S_{5.2} - S_{6.2} - S_{7.2}, S_3 - S_{5.2} - S_{6.2} - S_{7.2}, S_2 - S_{5.2} - S_{6.2} - S_{7.2}, S_1 - S_{6.2} - S_{5.2} - S_{7.2}\}$. The first step consists of finding pairs of *consecutive* stops used in both orders. In this case we find a list of ordered pairs $\langle S_{5.2}, S_{6.2}\rangle$ and $\langle S_{6.2}, S_{5.2}\rangle$. The next step is counting the number of trips in which the pairs occur in every trip of the dataset. In our case, we will compare $\{S_{5.2}, S_{6.2}\}$ and $\{S_{6.2}, S_{5.2}\}$ with every trip in the dataset. The tuple $\{S_{5.2}, S_{6.2}\}$ occurs three times, while $\{S_{6.2}, S_{5.2}\}$ occurs only one time. We assume that the case having the highest occurrence frequency is the correct one and hence, the last trip will be corrected. Suppose that there is not a majority, but the occurrences of both tuples are equal; in this case none of the trips are corrected. Note that we only tested these cleaning steps for buses and trams (GTFS for "*De Lijn*"). The single side boarding/alighting assumption does not hold for trains (at least not in Belgium).

4.3. Results

For the experiments, we used the following pipeline of reduction steps: (i) find unique trips, (ii) remove data which is not connected, (iii) remove duplicate consecutive stops and (iv) correct stops which are used in both directions. In Table 2, an overview is given of the percentage of deleted objects in every file. For the case study "De Lijn", the following results where found: the algorithm detected 19.12% of trips (i.e. the unique ones) in which duplicate consecutive stops occur and 0.57% of the trips were corrected due to the inconsistent orders.

Table 2: Overview of deleted objects in % per file for a set of GTFS files.

GTFS File	Agencies	Calendar Dates	Routes	Stops	Stop Times	Trips
De Lijn	0.00	42.62	15.06	19.13	97.45	97.50
TEC	0.00	22.22	0.00	33.33	93.11	93.20
MIVB	0.00	0.00	32.56	39.12	99.84	99.79
Connexxion	0.00	27.55	0.00	22.06	97.20	96.85
EBS	54.17	79.44	5.41	37.12	81.44	84.04

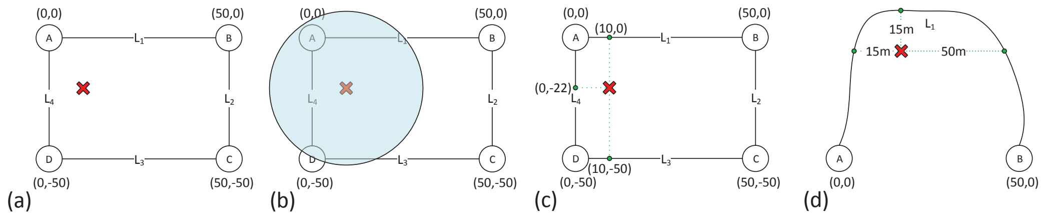

Fig. 4: (a) Four GeneralLinks and a GTFS stop. (b) Three GeneralLinks fall in the radius. (c) Projections on the GeneralLinks are calculated. (d) Three projections on the same link of which two projections share the shortest distance. A random point from the two shortest will be chosen.

5. Finding Candidate Locations for GTFS Stops

5.1. Algorithm

The goal of this algorithm is finding a set of candidate locations, called *projected stops*, for every GTFS bus stop. One of these projected stops will be chosen as the representative of the GTFS stop in the assignment algorithm which is currently in development. We determine these projected stops by the use of a `PostGIS` database. In this database, we have imported all the cleaned OSM data (as described in Section 3.2).

To find projected stops, we determine the radius in which needs to be searched and the maximum amount of projected stops we want. For this step the GIS functionality of the `PostGIS` database is used.

For every GTFS stop, we will calculate projected stops. This is achieved by finding every GeneralLink in a radius R of the GTFS stop. This query will return between zero and maximum X nearest GeneralLinks as a result. Both R and X can be configured in the software. When no GeneralLinks are found in the radius R, the algorithm will double the radius R and will attempt to find GeneralLinks again. This process will continue until at least one GeneralLink is found. In order to find a candidate location, we also need the nearest point on the link geometry on which the GTFS stop is projected. This is done by another query which can find the projection of a GeneralPoint (coordinates of the GTFS stop) on a GeneralLink (found by the previous query).

In Figure 4, an example of these steps is given. For this example, we choose a radius of 50 meters and a maximum amount of projected stops of three. In Figure 4a, we can see the starting situation with four GeneralLinks and a GTFS stop (cross). In Figure 4b, we see the radius which is specified and it is clear which GeneralLinks are found. The final step is finding the exact location of the projection, this can be seen in Figure 4c. Note that it is also possible that two or more projections on the same link with the same distance are found. In such cases a random point out of these closest points will be chosen. This can be seen in Figure 4d.

5.2. Results

In[6], Haklay et al. investigated the positional accuracy of OpenStreetMap roads in the Greater London area. In *complete areas* the average error is 9.57[m] with a standard deviation of 6.51[m]. In incomplete areas the average error is 11.72[m] and the standard deviation is 7.73[m]. In[6], completeness is defined as "*a measure of the lack of data*" and examined for specific areas by visual inspection of maps and by comparing (by means of GIS) the total road length found in OSM and in reference maps respectively. From several non-authoritative website sources it was found that the accuracy threshold $\overline{d}$ at 95% for GPS devices (used to locate the bus stops) can be assumed to be 20[m].

Based on both error limits, we decided to use 30[m] as the radius to find matching GeneralLinks for a bus-stop. For the experiments we chose a maximum of ten projected stops per GTFS stop. In Table 3, an overview of the number of GTFS stops having a specific number of projected stops is given. For instance, there are 16 196 GTFS stops which have two projected stops.

Table 3: Number of GTFS stops having the number of projected stops indicated in the column header.

Amount of Projected Stops found per GTFS stop	1	2	3	4	5	6	7	8	9	10
Occurrence Frequency	161	16 196	255	3 276	161	5 450	95	2 030	66	2 964

In total there are 127 066 projected stops found for a total of 30 654 GTFS stops, which is on average 4.15 projected stops per GTFS stop; this means we did not underestimate the amount of projected stops found per GTFS stop. In none of the cases a doubling of the radius was needed, which means that 30[m] was well chosen. One can observe that the amount of even occurrences is significantly higher than the amount of odd occurrences. This is due to the fact that we converted GeneralTransportInfrastructures with a direction of BOTH (which is the majority) to a FORWARD and a BACKWARD GeneralTransportInfrastructure.

6. Conclusion and Future Work

Accurate detailed data is required for micro-simulation aimed at the evaluation of collective transportation facilities. The data preparation consists of (i) the creation of an automatic tool to import OSM and GTFS data and (ii) the development of an algorithm to automatically assign about 30k bus stops to the OSM network (about 500k links). This paper describes the first stage of the OSM-GTFS integration, i.e. the algorithms to clean, reduce and prepare the OSM and GTFS data. Due to the size and the update frequency of the OSM and GTFS data, integrating them interactively is not an option. Ongoing and future research focuses on the problem of automatic bus stop assignment.

Acknowledgements

The research reported was partially funded by the IWT 135026 Smart-PT: Smart Adaptive Public Transport (ERA-NET Transport III Flagship Call 2013 “Future Traveling”).

References

1. Zilske, M., Neumann, A., Nagel, K.. Openstreetmap for traffic simulation. In: *M. Schmidt, G. Gartner (Eds.), Proceedings of the 1st European State of the Map OpenStreetMap conference, no. 11-10*. Vienna, Austria: M. Schmidt, G. Gartner (Eds.); 2011, p. 126–134. URL: `sotm-eu.org/userfiles/proceedingssotmEU2011.pdf`.
2. Haklay, M.. How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets. *Environment and Planning B: Planning and Design* 2010;**37**(4):682–703. URL: `http://epb.sagepub.com/lookup/doi/10.1068/b35097`. doi:`10.1068/b35097`.
3. Girres, J.F., Touya, G.. Quality Assessment of the French OpenStreetMap Dataset. *Transactions in GIS* 2010;**14**(4):435–459. URL: `http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9671.2010.01203.x/abstract`. doi:`10.1111/j.1467-9671.2010.01203.x`.
4. Mooney, P., Corcoran, P.. The Annotation Process in OpenStreetMap. *Transactions in GIS* 2012;**16**(4):561–579. URL: `http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9671.2012.01306.x/abstract`. doi:`10.1111/j.1467-9671.2012.01306.x`.
5. Barron, C., Neis, P., Zipf, A.. A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis. *Transactions in GIS* 2014; **18**(6):877–895. URL: `http://onlinelibrary.wiley.com/doi/10.1111/tgis.12073/abstract`. doi:`10.1111/tgis.12073`.
6. Haklay, M., Basiouka, S., Antoniou, V., Ather, A.. How Many Volunteers Does it Take to Map an Area Well? The Validity of Linus Law to Volunteered Geographic Information. *Maney Publishing (c) The British Cartographic Society* 2010;**47**(4):315–322.