Crowdsourcing turning-restrictions from map-matched trajectories

Alexandros Efentakis[a,*], Nikos Grivas[a], Dieter Pfoser[b], Yannis Vassiliou[c]

[a]*Research Center "Athena", Artemidos 6, Marousi 15125, Greece*
[b]*Department of Geography and GeoInformation Science, George Mason University, 4400 University Drive, MS 6C3Fairfax VA 22030-4444, USA*
[c]*School of Electrical and Computer Engineering, National Technical University of Athens, Iroon Polytechniou 9, Politechnioupoli Zographou 157 80 Athens, Greece*

Abstract

The availability of GPS-enabled devices has generated massive amounts of GPS tracking data produced by vehicles traversing the road-network. While initially used for improving traffic estimation and routing, only recently has this data been used for map-construction efforts. This work focuses on the specific aspect of identifying turning restrictions in the underlying road-network graph. We propose a novel, efficient and straightforward method to deduce turning restrictions for OpenStreetMap data, by mining historic map-matched trajectories from an existing fleet-management service. Our extensive experimental evaluation and verification process utilizing online map-services, satellite imagery, street view and public map-data APIs proves the efficiency and reliability of the proposed method.

Keywords: Crowdsourcing, Turning Restrictions, Map-matching, OpenStreetMaps

1. Introduction

Street maps and transportation networks are of fundamental importance in a wealth of applications. In recent years, Volunteered Geographic Information (VGI) [11] efforts such as OpenStreetMap (OSM) [19] have complemented commercial map datasets and provided map coverage for areas of limited commercial interest. On the other hand, the commoditization of GPS technology and integration in mobile phones, coupled with the advent of low-cost fleet management and positioning software has triggered the generation of vast amounts of tracking data. As a size indicator, one may consider the contribution of tracking data in OpenStreetMap, which is steadily increasing in size and currently amounts to 2.6 trillion points [31]. Besides the use of such data in traffic assessment and forecasting [7], i.e., map-matching vehicle trajectories to road networks

*Corresponding author
Email addresses: efentakis@imis.athena-innovation.gr (Alexandros Efentakis), grivas@imis.athena-innovation.gr (Nikos Grivas), dpfoser@gmu.edu (Dieter Pfoser), yv@cs.ntua.gr (Yannis Vassiliou)

to obtain travel times [4], there has also been a recent surge of actual map-construction algorithms that derive not only travel time attributes but actual road-network geometries from tracking data (see [1] for an overview and comparative study).

The present effort tackles a problem situated between map-matching and map-construction, since it focuses on improving *existing map datasets* by *crowdsourcing turning restrictions*. To this end, we use the map-matched trajectories produced by the SimpleFleet system of [7], which proposed an optimal workflow for combining state-of-the-art research about road networks, Floating Car Data (FCD) and map-matching algorithms in the context of low-cost fleet management solutions. The specific service was implemented for three European cities, namely Athens (Greece), Berlin (Germany) and Vienna (Austria) and during its implementation, we created road-network graphs from OpenStreetMaps data, collected huge amounts of Floating Car Data from fleet vehicles, applied state-of-the-art map-matching algorithms to align the observed GPS traces to the road-network graph and consequently produced high-quality historic speed profiles along with frequently updated live-traffic information. This combination of live-traffic information and speed profiles was then used to provide up-to-date, live-traffic, shortest-path and isochrone computation (refreshed every 5 minutes), using the shortest-path implementation of [10]. Moreover, the follow-up work of [9] clearly showcased the impact of traffic fluctuations in a geomarketing context, by combining the live-traffic isochrone functionality of this system with demographic data.

SimpleFleet used OpenStreetMap (OSM) data for constructing the road-network graphs. However, operating this service for more than a year and for the three urban regions has revealed an *inherent limitation of the OSM dataset*. It contains limited information for turning restrictions, i.e., a transition from one network edge to another (via an intersection vertex) that is prohibited due to local traffic rules. Although OSM supports turning restrictions by using an additional relation tag (Relation:restriction [30]), only a small number of users contribute to this information. This is particularly evident, considering that OSM includes more than 2.1 billion Nodes, Ways and Relations [29] and less than 230,000 relations actually represent turning restrictions [30]. Our individual test cases confirm this observation. For the Athens area and its 277K vertices road network, only 214 turning restrictions have been recorded by OSM users. This observed lack of data is mainly attributed to the fact that there are no public datasets for traffic signs easily found (if any), satellite imagery cannot testify to the existence of such restrictions and contributing turning restrictions even for a single road to the OSM dataset may be extremely time-consuming.

Despite the fact that turning restrictions are especially important for any public mapping service, there is only a limited number of scientific literature addressing them, since *"no publicly-available realistic turn data exist"* [6]. Turning restrictions severely impact the quality of computed shortest-paths provided by routing engines considerably more than traffic: While ignoring traffic returns a suboptimal, yet valid route to the user, ignoring turning restrictions provides erroneous paths that may lead to accidents. Thus, providing a semi-automatic method for identifying turning restrictions is extremely important for any public mapping service.

During our research on related work, we found a significant body of work focusing on Floating Car Data (FCD) (see [44, 43] for a partial overview on GPS related research). The only previous works relevant to solving (or even acknowledging) our

actual problem also use FCD for calculating turn delays [3, 24, 37, 41, 42]. However, no scientific literature exists that utilizes map-matched (MM) trajectories to derive turning restrictions. As such, this work presents and significantly extends our previous results [8] by employing a more rigorous and systematic verification process. Consequently, it is the first approach that automatically identifies and infers turning restrictions based on historic map-matched trajectory datasets. This approach has several benefits in that map-matched trajectories are (i) more condensed, i.e., instead of random locations in the plane we use edge sequences in the road-network graph and (ii) less ambiguous and susceptible to errors, i.e., movement is interpolated using the actual road-network. Although this work focuses on OpenStreetMaps, it may also be used for any road network dataset, i.e., for cases in which the road network evolves faster than commercial map updates. As recent examples such as the Apple Maps incident [1] have shown, even commercial datasets from technology giants are prone to failure. Hence, providing a method that improves existing map-datasets by using the map-matched trajectories created by vehicle drivers traversing the road network may be extremely useful, even for commercial road-network vendors and providers.

The outline of this work is as follows. Section 2 describes previous research in relation to our work. Section 3 describes our scientific contribution towards identifying turning restrictions in the OSM dataset by utilizing historic map-matched trajectories. Section 4 summarizes the results of our approach. Finally, Section 5 gives conclusions and directions for future work.

2. Related work

Recently, real-time Floating Car Data (FCD) collected by GPS-enabled vehicles has become the mainstream in traffic study because of its cost-effectiveness, flexibility and being the "*the only significant traffic data source with the prospect of global coverage in the future*" [21]. Typically a GPS trajectory describing a vehicle movement consists of a sequence of measurements with latitude, longitude and timestamp information. However, this data is inherently imprecise *"due to measurement errors caused by the limited GPS accuracy and the sampling error caused by the sampling rate"* [32]. Therefore the observed GPS traces need to be aligned with the road-network graph through a process commonly referred to, as *map-matching*. Map-matching algorithms accept as input a vehicle trajectory and output the path (i.e., an ordered sequence of road-network graph edges) that this vehicle has potentially traversed (according to its movement pattern), along with travel time information, i.e., how long did it take for the specific vehicle to traverse the calculated path. For the remainder of this work, the term *map-matched trajectory* refers only to the calculated vehicle's path and not to the associated travel time information. In the SimpleFleet service of [7] we employed the Fréchet-based curve-matching algorithm of [4, 39] and an implementation [22] of the ST-matching algorithm of [25]. Both implementations were adapted to handle incoming FCD in a streaming fashion.

[1] http://techcrunch.com/2012/09/28/tim-cook-apologizes-for-apple-maps-points-to-competitive-alternatives/

Despite their inherent imprecision and the usually low sampling-rates of most available public datasets, there was an explosion of research focusing on GPS trajectories in recent years (cf. [44, 43]). Unfortunately, so far, only limited effort focused on road-network construction and map-updates, such as in this case turning restrictions and road intersections. This comes as a surprise, since intersections are important components of urban road-networks and contribute significantly to the total travel time cost [28, 38]. Previous works [28] conclude that intersection delays, i.e., *"the turn cost associated with the continuation of travel between edges via an intersection node"* are responsible for as much as the $17 - 35\%$ of the total travel time according to a conducted survey in the Copenhagen urban area [40] .

Existing work mostly focuses on estimating intersection delays based on the available Floating Car Data. To calculate intersection delays, researchers have utilized the historical mean method [37, 42], piecewise linear interpolation [3, 41] and the principal curves method [20] to overcome data sparseness of Floating Car Data and calculate turn-delay tables for the Beijing urban region [24].

Although turn-costs and intersection delays are a generalization of turning restrictions, i.e., a turning restriction is a turn with delay set to ∞, previous works are fundamentally different from our approach at several levels. First, for previous approaches to calculate turn-cost for a particular turn, many vehicles need to actually traverse it. Contrarily, we identify turning restrictions by focusing on turns with no available tracking data. Second, typically GPS trajectories are used. We use map-matched trajectories. Third, most publicly available GPS datasets are either simulated [23], focus on a specific city [5, 27] or cover only limited time periods (a day, week or month for [23, 27, 5] respectively). In the present work, we use *real traffic data* from three different European cities and fleets of $2,000 - 5,000$ vehicles per city, covering a full 12-month period. Since we get almost identical results for all cities (see Section 4), our method was showed to be both realistic and robust. Lastly, since previous methods are based on data-mining techniques, they may only verify results by dividing the original GPS datasets into a training and a test set. Contrarily, we use a plethora of multiple alternative techniques and a rigorous validation method to verify our findings *against the actual ground-truth* conditions of the tested road-networks. Such techniques include (i) a visual inspection of our results with public mapping services, (ii) using satellite imagery and street view of the particular areas for manual visual inspection, (iii) batch retrieving of street view images for human evaluation in public crowdsourced marketplace platforms and (iv) utilizing two separate public mapping APIs from different vendors to verify our results. Using all those verification methods, we can ensure our method's robustness and obtain significant insights about its specific performance characteristics, as well as its potential improvements. This work is mainly based on the [8] publication but it also introduces a significantly expanded verification process and experimentation section.

3. Crowdsourcing Turning Restrictions

The core contribution of this work is to propose a solid and straightforward methodology for *identifying turning restrictions for existing road-network datasets from historic map-matched trajectories*. The main motivation for this effort is twofold. (i) First,

(a) Prohibited U-turn (b) Prohibited left turn (c) Prohibited right turn (d) No entry sign

Figure 1: Prohibitory traffic signs for turning restrictions

we want to improve the OpenStreetMaps (OSM) road-network dataset. Despite the overall high quality of OSM, it has very limited information for turning restrictions, which inhibits its use for mission-critical applications (ii) To propose a novel way of taking advantage of available GPS tracking data to improve commercial road-network datasets with minimum effort. Although the use of online shortest-path APIs provided by huge technology vendors is free for casual users and a limited number of queries, for vehicle fleets with thousands of vehicles the free-usage limits of those APIs simply cannot suffice. For a typical vehicle fleet of 5,000 taxis (like the one monitored in Berlin for our SimpleFleet service) the free usage limit of 2,500 requests of the Google directions API [15], cannot even service one shortest-path request per vehicle and day. The maximum billing for such a large-fleet could be as high as $50,000$ per day and amounts to an average of 20 shortest-path computations per vehicle.

Another disadvantage of using those public shortest-path APIs is, that huge companies do not want to share their itineraries (as revealed by the corresponding shortest-path requests) with their competitors, e.g., Amazon would not like to reveal the locations of its customer base to Google. Hence, most of those large companies have to create their own in-house solutions buying the corresponding road-network datasets from vendors like TomTom (which again is costly) or use OpenStreetMaps instead. Since, those companies have a lot of tracking data available from their vehicle fleets, it makes sense to take advantage of their drivers' local knowledge to identify restricted or even "unattractive" turns by mining those tracking data. Again, the methodology proposed in this work could be easily adapted to such a scenario for commercial road-network datasets. In the following section we will show how to extract turning restrictions from historic map-matched trajectories and in the process, we will also describe the OpenStreetMaps road-network dataset and its properties.

3.1. Definitions and Preliminaries

In the discussion that follows, a road-network is represented as a directed weighted graph $G(V, E, w)$, where V is a finite set of vertices / nodes, $E \subseteq V \text{x} V$ are the edges of the graph and w is a positive weight function $E \rightarrow R^{+}$. Typically the weight w represents the travel time required to traverse the edge. In other cases, w may refer to the length of the edge in meters (for travel distances metric).

The degree of a vertex u, denoted as $deg(u)$, is the number of edges incident to the vertex. *Intersection vertices* are the road-network vertices with vertex-degree larger

Table 1: Turning restrictions added in OSM per year for the cities covered by our service

city	2009	2010	2011	2012	2013	Total
Athens	-	11	1	75	127	**214**
Berlin	8	26	101	386	147	**668**
Vienna	33	36	99	307	324	**799**

Table 2: OSM road-networks of the three cities covered by our service

			# intersection vertices		# intersection vertices for roads $\leq$ 10	
city	**# vertices**	**# edges**	**total**	**%**	**total**	**%**
Athens	277,719	329,444	100,422	26%	34,921	13%
Berlin	89,598	103,486	51,935	58%	21,119	24%
Vienna	100,579	112,478	44,874	45%	16,104	16%

than two, i.e., $I = \{v_i \in V, deg(v_i) > 2\}$. A *turning restriction* is an ordered sequence of two or more network edges connected via intersection vertices that is prohibited due to local traffic rules. Drivers are alerted for existing turning restrictions through standardized traffic signs (see Fig. 1). In this paper, we only cover those edge sequences that consist of *a single ordered pair of two edges connected via a single intersection vertex*. This constellation represents the majority of turning restrictions in typical road-networks. Note that turning restrictions do not refer to one-way streets, because (i) even a single edge may be marked as unidirectional and (ii) turning restrictions may refer to roads that are bidirectional, but it is only their sequential traversal that is prohibited. Moreover, unidirectional streets are easily modeled in every directed graph representation, whereas *turning restrictions are a distinguishing characteristic of road-networks*, which differentiates them from other types of networks.

3.2. OpenStreetMap and Turning Restrictions Coverage

OpenStreetMap (OSM) provides unlimited and free access to the entire map dataset under an Open Database License [2]. This massive amount of data may be downloaded in full, but is also available through APIs and Web services. Users may participate in the OSM community by providing their local knowledge-based feedback and edit the map. Although OSM contains a relation tag (Relation:restriction [30]) for describing turning restrictions, only a small number of OSM users are aware of this property. This fact was easily confirmed for the cases of three European cities (Athens, Berlin, Vienna) covered by our service. The results for September 2013 presented in Table 1 show that the available data for turning restrictions is low when compared to road-network sizes of Table 2. We obtained similar or worse results for other European cities, especially for countries with less extensive coverage (e.g., Albania, Montenegro).

[2] http://opendatacommons.org/licenses/odbl/1.0/

Using map-matched trajectories, our method for discovering turning restrictions identifies turns that, although allowed in the original map dataset, are rarely executed by drivers. Exhibiting such an unusually low frequency, such turns have a very high probability to be actually prohibited. Essentially, our approach is based on crowdsourcing driver behavior as evidenced by their tracking data.

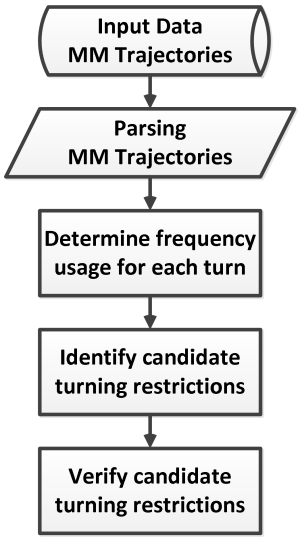


Figure 2: Methodology for identifying OSM turning restrictions by using historic map-matched trajectories

3.3. Methodology

Our basic methodology for inferring/identifying OSM turning restrictions can be described by the simplified diagram of Figure 2. The following sections will elaborate on the independent stages of this process.

3.3.1. Input data (Map-matched trajectories)

In the SimpleFleet system of [7], GPS-traces of fleet vehicles for the three European cities arrive in a streaming fashion. For each geographic area, we monitored $2,000-5,000$ vehicles producing a GPS position sample every $60-180$s. GPS trajectories for each vehicle are subsequently map-matched. The end-result of this process is an ordered sequence of road edges that each vehicle has traversed. The traffic datastore of the service, including the OSM road-network graphs, Floating Car Data and map-matched trajectories, is implemented using a PostgreSQL/PostGIS database [34, 33] (one database instance per city).

As a vehicle traverses the road-network, it traverses roads of varying importance (cf. Table 3 for the distribution of the OSM road-networks per their respective category). Hence, the typical usage of each road is directly linked to its respective category. To reduce the bulk of data stored in the corresponding datastores, a separate process eliminates map-matched edges that belong to edges of less important roads, i.e., those

Table 3: Road categories for the OSM road-networks

CategoryID	Road category	Athens	Berlin	Vienna
1	motorway	4287	1420	2410
2	motorway link	3747	2012	4386
3	trunk	1343	111	171
4	trunk link	567	0	227
5	primary	16210	5203	8913
6	primary link	1257	347	422
7	secondary	42881	21250	12894
8	secondary link	0	45	0
9	tertiary	58722	9678	11576
10	tertiary link	0	6	0
11	unclassified	13484	2792	3060
12	road	395	28	0
13	residential	186459	58338	67482
14	living street	92	2256	937

Table 4: Typical size of compressed MM trajectory archives

Size	Athens	Berlin	Vienna
per day	22.3 MB	224 MB	76.3 MB
per month	0.67 GB	6.74 GB	2.29 GB

that correspond to road-categories greater than 10 (OSM categories for unclassified, road, residential and living street - see Table 3). Depending on the time-period and the traffic patterns of each city, about 12−15% of the map-matched records are eliminated using this "sanitation" process.

Since map-matched records are mainly used to offer real-time traffic information, older data is periodically removed from the respective PostgreSQL datastores (every 5 minutes) and archived into comma-delimited (csv) files for offline use. At the end of each day, a batch process compresses the text files created during the day. The compressed files are then sent to a backup server for permanent archiving. Table 4 indicates the typical size of compressed archives produced per day and month for each city.

After a whole year of data collection (Oct. 2012 - Sept. 2013), several GBs of compressed historic map-matched trajectories are archived for each of the cities covered by the service. The real challenge is how to utilize this significant wealth of data to infer turning restrictions for the respective OSM road-networks.

3.3.2. Parsing map-matched trajectories and optimizations

The scope of our work is to identify specific turns (i.e., ordered pairs of edges connected via an intersection vertex) that, in an unusual way, are infrequently executed by vehicles. This frequency will be determined by parsing the compressed archives of the historic map-matched trajectories produced for the three cities during the one-year

Table 5: Total counted instances for all monitored turns between Oct 2012 and September 2013

city	# intersection vertices for roads ≤ 10	#examined turns	# total instances	# instances per inters. vertex for roads ≤ 10
Athens	34,921	75,552	144,451,729	4,137
Berlin	22,119	44,636	2,054,969,090	97,304
Vienna	16,104	36,484	610,902,632	37,935

operational period of our service. Since, the respective OSM road-networks comprise hundreds of thousands of vertices and edges (see Table 2), we need to somehow limit the number of turns that need to be examined.

The first optimization is to identify those pairs of consecutive edges that connect at intersection vertices. There is no need to study vertices of degree 2 (with just one incoming and one outgoing edge) or lower, since in those cases the driver has no choice but to travel in one direction (no actual intersection). In this way, we can effectively limit the number of candidate turns since the number of intersection vertices is much smaller (less than 60% or even less) than the number of total vertices (see Table 2).

The second optimization relates to the archived data. Since we only store map-matched trajectories that include major roads, i.e., OSM categories ≤ 10, we are also only interested in those intersection vertices connected to such roads. In making this assumption, we might miss some intersection vertices (strictly connected to unimportant roads). However, turn restrictions on minor roads have not only little to no impact on the overall traffic, but are also not always clear to express and enforce. On the other hand, intersections involving major roads are more likely to be used by vehicle drivers and, thus, have an overall dominant impact on traffic. The process of minimizing the number of turns we need to monitor is described in the pseudocode of Algorithm CALCULATEMONITOREDTURNS. Table 2 shows that major roads' intersection vertices are less than 25% of total vertices of all cities covered by our service.

CALCULATEMONITOREDTURNS($G(V, E, w)$)

```
1  monitoredTurns = {}
2  totalTurns(V) = calculateAllTurnsOf(G)
3  for each turn(v) ∈ totalTurns(V)
4      if v.degree > 2  &  v.connectedToMajorRoad == TRUE
5          monitoredTurns = monitoredTurns ∪ turn(v)
6  return monitoredTurns
```

These two optimizations considerably reduce the number of unique turns/pairs of consecutive edges we need to monitor, which is a considerable improvement (see Columns 2,3 of Table 5). Since the OSM road-networks of each city are stored in the respective PostgreSQL datastores, Algorithm CALCULATEMONITOREDTURNS for determining intersection vertices of interest and their corresponding turns may be easily implemented using plain SQL commands.

calculateMonitoredTurnsUsage(*monitoredTurns*, *MmResults*, *examTimePeriod*)

```
1  for each monitoredTurn ∈ monitoredTurns
2      freqCounter[monitoredTurn] = 0
3  for each day ∈ examTimePeriod
4      for each vehicleId ∈ MMResults(day)
5          for each turn ∈ MMResults(day, vehicleId)
6              if turn ∈ monitoredTurns
7                  freqCounter[turn] = freqCounter[turn] + 1
8  return freqCounter[monitoredTurns]
```

As a companion to the monitored-turns detection algorithm, we implemented a custom Java application that (i) parses the compressed archives of historic map-matched trajectories (see Section 3.3.1), (ii) counts the instances encountered for each monitored turn and (iii) stores the results in the respective PostgreSQL datastores. Algorithm calculateMonitoredTurnsUsage describes this process and the results, i.e., the total counted instances for all monitored turns during our one-year testing period, are given in Table 5. Results show that on average for every intersection vertex connected to major roads (i.e., their respective road category ≤ 10), we have counted turn instances, ranging from 4,137 (Athens) up to 97,304 (Berlin). These results represent a sufficiently large number of measurements per intersection vertex.

3.3.3. Identifying candidate turning restrictions

At this point in our approach, we have identified the turns we need to monitor and counted the number of times each monitored turn has been traversed by a vehicle. We now need to examine, which of those turns are *rarely used*. Since, both, turns and results of the enumeration process are stored in the respective datastores, it is easy to group results/turns by *entrance* edge and direction (for bidirectional edges). Each such group contains all possible turns a vehicle may execute after following a specific entrance edge (and direction). Likewise, each turn belongs to a single, specific group of turns. Since we know the number of instances encountered for each one of the turns belonging to the same group, it is easy to calculate the usage percentage or each one (cf. Algorithm calculateTurnPercentPerGroup). An example group for a specific entrance edge is shown in Figure 3.

calculateTurnPercentPerGroup(*monitoredTurns*, *freqCounter*(*monitoredTurns*))

```
1  groupsOfTurns = group(monitoredTurns, entranceEdge, direction)
2  for each group ∈ groupsOfTurns
3      freqCounter[group] = 0
4      for each monitoredTurn ∈ group
5          freqCounter[group] = freqCounter[group] + freqCounter[monitoredTurn]
6  for each group ∈ groupsOfTurns
7      for each monitoredTurn ∈ group
8          percent[monitoredTurn] = freqCounter[monitoredTurn] / freqCounter[group]
9  return percent[monitoredTurns]
```

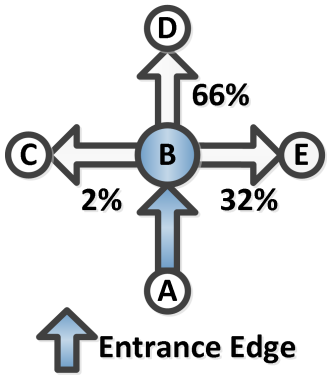


Figure 3: A simple example of grouping turns per entrance edge (A, B) at an intersection vertex (B) for calculating usage percentage per turn

Table 6: Number of candidate turning restrictions discovered for 5% and 2.5% thresholds

		# turning restrictions		turning restrictions (%)	
city	**# turns**	**5%**	**2.5%**	**5%**	**2.5%**
Athens	75,552	5,287	3,596	7.00%	4.76%
Berlin	44,636	2,653	1,582	5.94%	3.54%
Vienna	36,484	1,739	1,261	4.77%	3.46%

As we notice in the example group of Figure 3, most drivers (66%) continue straight when they traverse the entrance edge (A, B) leading to the intersection vertex B. Some others (32%) prefer to turn right. But a very small percentage of them (2%) turn left. This is a very strong indication that this low frequency-usage of the left-turn actually represents erroneous map-matched trajectories (even the most efficient MM algorithms have a small error rate). Next, we made the assumption that turns with frequency usage percentage lower than a 5% threshold are most likely prohibited. The choice of this threshold was established after an initial set of experiments resulting in encouraging turning restriction results. Table 6 shows the respective number of the candidate turning restrictions discovered for each city for, both, 5% and 2.5% thresholds.

However, estimating candidate turning restrictions is not enough. For each such turn, we need to additionally calculate its direction in comparison to its entrance edge. Figure 4 shows the angles assigned to each possible turn direction and *U-left* and *U-right* turns will be collectively referred to as *U-turns* in the remainder of this paper. Note that in our case, *U-turns* do not correspond to the typical case $(A, B) \rightarrow (B, A)$, which are implicitly prohibited in most road networks but in the case of a turn from edge (A,B) to edge (B,C) when the angle between edges (A,B) and (B,C) are at the center, bottom part of the angle circle of Figure 4. Note, that the direction calculation is comparatively easy, since we have already stored the angular direction of each edge in the respective datastore as needed for the isochrone functionality of our service [7]. Table 7 shows the categorization of the discovered candidate turning restrictions with

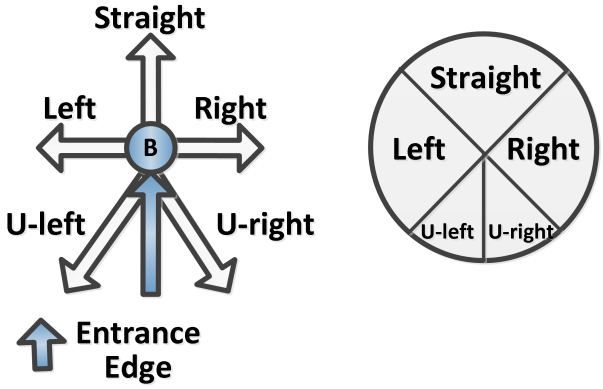


Figure 4: Categorization of turns according to directions.

respect to their direction. As expected, most of them (particularly in Berlin and Vienna) represent left-turns. As our verification results will show, the direction of the candidate turning restrictions has a very strong impact on the turning restrictions' validity.

To additionally improve the validity assessment of identified turning restrictions, we devised the verification methods described in the following section.

3.4. Verification process

Although identifying candidate turning restriction based on the frequency usage of their respective turns is an important criterion, we still need to verify our results in comparison to the actual road-network conditions. To this effect, we will use many alternative methods that may be used independently or in different combinations according to the budgetary and time constraints imposed for a particular test case. Those verification methods include: (i) Using a mapping application for visualizing turning restrictions in comparison to a web mapping service, (ii) Using manual inspection of candidate turning restrictions by utilizing publicly available satellite imagery and Google Street View (wherever available), (iii) Batch retrieving multiple street view images per candidate turning restriction for human evaluation in a public crowdsourced marketplace platform and (iv) Cross-checking our results with two separate map vendors' public APIs. Those alternative verification processes offer multiple advantages over typical data-mining techniques, since *we are able to cross-reference our results in comparison to the actual road-network conditions*. Moreover, the new verification processes proposed in this work (i.e., manual inspection of candidate turning restrictions using publicly available satellite imagery and the batch retrieving of street view images for human evaluation), two extensions of the original work in [8], turned out to be the most intuitive methods for providing the most accurate feedback for the qualitative evaluation of our approach and how to further calibrate its accuracy.

3.4.1. Visualizing results with a mapping application

Our first verification method entails the visualization of the candidate restricted turns in a mapping application. An intuitive way to do this is to (i) cross reference

Table 7: Categorization of candidate turning restrictions per direction for 5% threshold

city	# turning restrictions	straight	left	right	U-turn
Athens	5,287	6.5%	45.4%	41.6%	6.5%
Berlin	2,653	1.6%	64.6%	18.6%	15.2%
Vienna	1,739	10.5%	44.8%	30.0%	14.7%

(a) Turning restrictions visualization in Athens

(b) Turning restrictions visualization in Berlin

(c) Turning restrictions visualization in Vienna

Figure 5: Visualizing turning restriction with QGIS

each such turn with the appropriate traffic sign (depending on the direction of the turn according to Table 7) located at the corresponding intersection vertex coordinates and (ii) rotate each such traffic sign according to the entrance edge direction and to "simulate" what the driver witnesses before entering the corresponding intersection vertex. Since the second of these criteria cannot be achieved through either Google Maps or Google Earth [12], we used QGIS [35], which is a popular, free and open source GIS application that runs in all major operation systems. In addition, QGIS may visualize geometry features directly retrieved from PostGIS enabled databases (such as our datastores) and, thus, we can avoid an unnecessary export process of our data. Moreover, we used a Google Maps Layer in QGIS as the background map layer to compare results with an external mapping service. Figure 5 shows some typical examples of the results of this visualization process for some of the candidate turning restrictions.

- Figure 5(a) depicts an intersection familiar to most local drivers in the center of Athens. This type of restrictions were easily verified by our personal experience and they effectively demonstrate how easily, critical turning restrictions are discovered through our method.
- Figure 5(b) shows a case of a prohibited U-turn in the Berlin area. There, many disallowed U-turns are missing from the OpenStreetMap dataset.
- Figure 5(c) shows that the Google Maps layer visually confirms the discovered turning restriction.

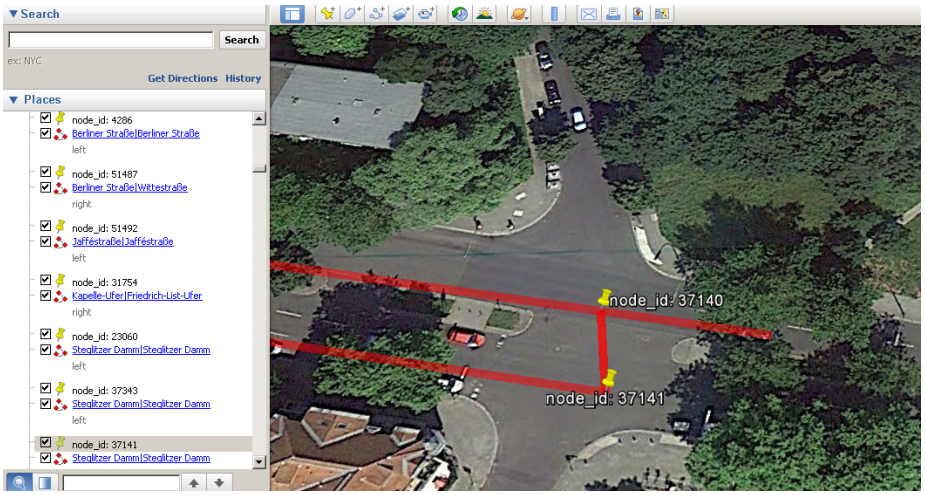


Figure 6: The KML file created for Berlin

Although our first verification method is an elegant, straightforward and efficient way for visualizing and cross-referencing our results, it has one major drawback. It still requires local knowledge of the examined area to confirm the turning restrictions. As such, it may be very useful for experienced end-users with extensive local knowledge, such as fleet managers or taxi drivers, but it does not benefit the typical (Web) user dealing with areas unknown to him. To this effect, the following sections will provide alternative means of verification hat do not require local knowledge of a specific area.

3.4.2. Manual inspection of satellite imagery and Street View

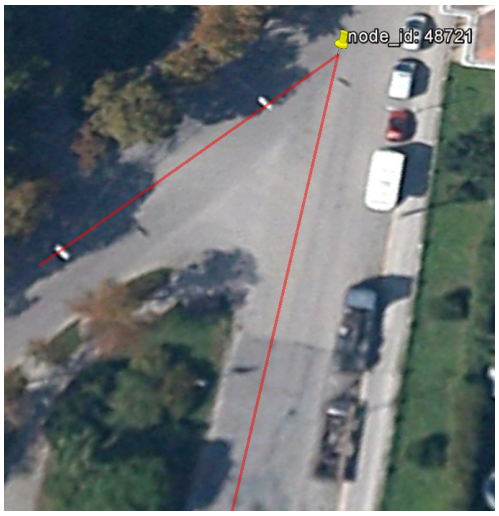
Our second verification method is to export the candidate turning restrictions in a format that may be used by Google Earth to cross-reference results with data collected by Google, such as the actual satellite imagery and Street View. We used the export functionality of PostGIS to create KML [13] representations, the data format used in Google Earth, of the line geometries of each candidate turning restriction and create one large KML file per city. In addition, during our KML export, we add the name of the edges/roads participating in each restriction in alphabetical order for easy lookup, as shown in Figure 6. We can get an even clearer impression of the respective intersections should the selected areas be available in Google Street View [14]. Some of the results of this visualization process are shown in Figure 7.

Cross-referencing results with satellite imagery has revealed several interesting details about our method. First of all, it showed that most U-turn restrictions may be easily verified (see Figures 7(c), 7(d)). However, the most important aspect is that those verified U-turn restrictions do not even have a traffic-sign assigned in the actual road network, since *no actual driver would effectively use them*, i.e., they are too dangerous. Nevertheless, these turning restrictions should still be added on the digital representation of the road-network graph for accurate shortest-path computation by routing engines. Therefore, our method not only discovers true turning restrictions

(a) A turning restriction in Athens verified by satellite imagery

(b) A turning restriction in Berlin verified by satellite imagery

(c) Another turning restriction in Berlin verified by satellite imagery



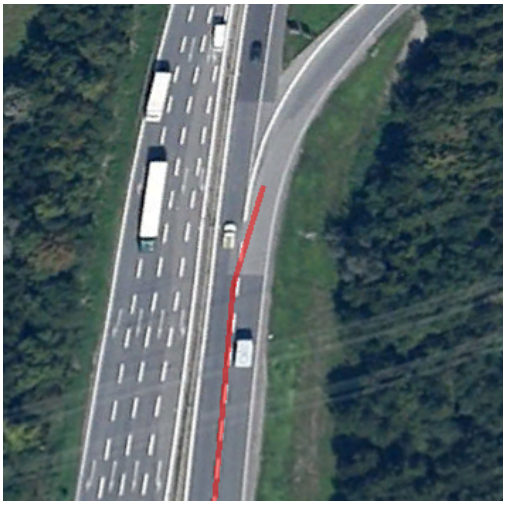
(d) A turning restriction in Wienna verified by satellite imagery

Figure 7: Verifying turning restrictions on Google Earth

(as represented by traffic-signs), but also *discovers those restrictions not explicitly prohibited by a traffic-sign due to their extremely low probability of actually being used*. This is an interesting discovery, since even if we were given full access to a world-wide traffic-sign database (if such a database existed), our method could still pinpoint additional existing turning restrictions.

Our method also produces some false-positives as revealed by satellite imagery. Examples are shown in Figure 8. Sometimes, even satellite imagery is not enough for revealing useful details and we need to resort to Google Street View for a more accurate depiction of the local intersection. Using Street View revealed several details for the examined intersections ranging from existing traffic-signs (as shown in Figure 9) to

(a) A false-positive turning restriction in Vienna as revealed by satellite imagery



(b) Another false-positive turning restriction in Vienna as revealed by satellite imagery

Figure 8: False positives produced by our method for "straight" turns

Figure 9: A false-positive turning restriction in Athens, as revealed by Google Street View

errors on the OSM mapping process (as shown in Figure 10)

When we closely inspected false-positives to identify a common pattern, we realized that many of them represent "straight" turns, i.e., turns with small angles between entrance and exit edge (see Figure 8). Such turns are mostly encountered on highways for exiting the main road or performing U-turns. The fact that the vehicles we monitored (during the one-year period) rarely used such turns is mainly attributed to the fact that we dealt with fleets using professional drivers (trucks in Athens, taxis in Vienna

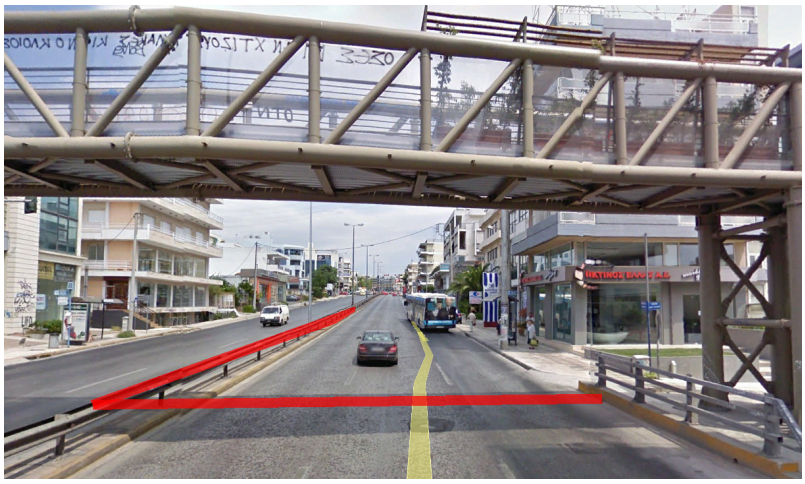


Figure 10: An error in OSM maps that confirms the existence of the discovered turning restriction, as revealed by Google Street View

```
https://maps.googleapis.com/maps/api/streetview?size=
600x300&location=46.414382,10.013988&heading=1518&
pitch=0&key=Google_API_key
```

Figure 11: A sample Google Street View Image API request

and Berlin), who (i) know precisely how to reach their destination and never have to backtrack when on a highway and (ii) usually use highways only for distant trips and not nearby destinations. In this sense, we should probably treat these "straight" turns as less promising to represent actual turning restrictions, a fact that will also be verified during the final phase of our verification process.

3.4.3. Batch retrieving of Street View images and crowdsourcing

In the previous section, we have identified the main qualitative characteristics of the candidate turning restrictions, using manual inspection of the discovered turns on Google Earth and Google Street View. Unfortunately, inspecting all candidate turning restrictions for a set of cities one-by-one is infeasible by a small number of map-specialists. One popular method to overcome this issue, is to offload the inspection of those candidate restrictions to the "crowd", by using popular crowdsourcing marketplace platforms like Amazon Mechanical Turk[2]. To facilitate such a process, we take advantage of the fact that Google offers the Street View Image API [18] that allows the bulk retrieval of Street View images for a set of predefined locations. In the remainder of this section, we will show how we will use this specific API in the context of crowdsourcing the inspection of the previously discovered candidate turning restrictions.

The Google Street View Image API allows developers to embed a static (non-interactive) Street View image into their web page. The respective viewport is defined with URL parameters sent through a standard HTTP request and is returned as a static

image. A sample request is shown on Figure 11. The most important parameters in such a request are [16]:

- Location. Where exactly is the location of the Street View Image. It can be either a text string (e.g., an address) or a lat/lng value.
- Size. Specifies the output size of the image in pixels. For free usage, the width and height of the image cannot exceed 640 pixels.
- Heading. Indicates the compass heading of the camera. It accepts values are from $0 - 360$ (both values indicating North, 90 indicating East and 180 South).
- Pitch. Specifies the up or down angle of the camera relative to the Street View vehicle. It defaults to 0.

To take advantage of the Google Street View Image availability for the discovered turning restrictions, we follow the methodology of Figure 12, where red arrows illustrate the location and heading of the corresponding image requested for each location. For each turning restriction (such as the left turning restriction A to C through B of Figure 12), we use the Google Street View Image API to retrieve exactly 5 images. The first image is at location A (tail vertex of entrance edge (A, B)), the second image is located in the middle of entrance edge (A, B) (location AB) and the last three images are at location B (the intersection vertex). The difficult part is computing the `heading` parameter of the Google Street View Image API to simulate what a random vehicle driver would see when entering the entrance vertex and is driving towards this specific turn. The `heading` parameter in pictures 1,2 and 3 is calculated strictly by locations A and B and is the angle that a driver must have on location A in order to face location B. The fifth street view image at location B uses a `heading` calculated by locations B and C and is the angle that a driver must have on location B in order to face location C (i.e., to view the exit edge) and the forth image uses a heading between the headings of photos 3 and 5, to simulate a panoramic view of the turn.

To batch-retrieve the necessary five photos per each candidate turning restriction for our cities of interest, we wrote a Java command line application that takes the locations A, B and C from our datastore, computes the location AB (the middle point of entrance edge (A, B)) and calculates the necessary headings as described in the previous paragraph, by adapting the `computeHeading()` [17] method offered by the Google Maps Javascript API, that allows the computation of the `heading` parameter of the Google Street View Image API between two locations. We also used the Sprockets for Java [36] library for accessing the Google Street View Image API from Java. Unfortunately, this specific process was only possible for the cities of Athens and Berlin, since Vienna is not yet covered by Google Street View [3].

Sample results of this process are shown in Figures 13 and 14. The retrieved images indeed confirm the existence of these specific turn restrictions, either because the former right turn is blocked for maintenance, or the latter left turn leads to a private

[3] `https://en.wikipedia.org/wiki/Google_Street_View_in_Europe`

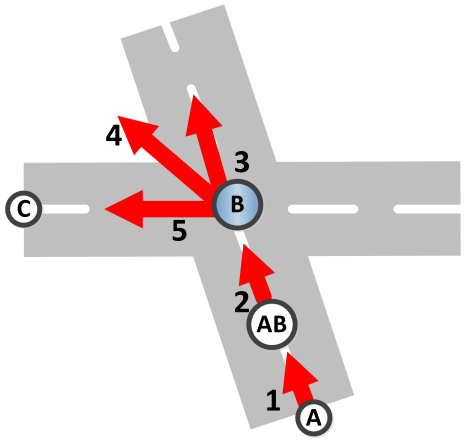


Figure 12: A candidate turning restriction and the locations & headings of the retrieved Google Street View images

property (dead end). These images also show, that the described methodology for retrieving five images for each candidate turning restriction is very efficient in simulating what a vehicle driver sees when entering each of those turns. Then, the corresponding images could easily be evaluated by humans that may confirm (or not) the existence of those restrictions and hence, they could easily be used in a crowdourced marketplace platform for human evaluation such as the Amazon Mechanical Turk. Again, the fact that we narrowed down a limited number of candidate turning restrictions by mining the map-matched trajectories, effectively limited the number of requests to the Google Street View Image API and thus, the whole process for Athens and Berlin requires less than a few hours to complete. Moreover, the limited number of candidate turning restrictions for human evaluation limits the corresponding costs (and the time required) when the corresponding images per turn would be uploaded to a crowdourced marketplace platform for human evaluation. Note that although we have not performed this human evaluation due to budgetary and time constraints, a random sampling of the candidate turning restrictions for Athens and Berlin has shown that in most cases it is very easy to confirm (or not) the existence of a turning restriction by the retrieved Street View images and hence, we believe that the proposed methodology for combining batch-retrieved Street View images with turning restrictions could be very accurate when used in a public crowdsourced platform.

3.4.4. Sourcing external mapping services

Although the evaluation methods presented in the previous sections provide a convincing, qualitative way of validating results and provided significant insight into potential drawbacks of our method, they still require human intervention to confirm results. Hence, it would be best if we could further verify and quantify our findings through an purely *automated process*. In this section, we propose a method that compares our results with those provided by external Web-based routing APIs, the Google

(a) Photo 1 at location A (tail vertex of entrance edge (A, B))

(b) Photo 2 at location AB (middle of entrance edge)

(c) Photo 4 at location B (intersection)

(d) Photo 5 at location B (intersection)

Figure 13: Batch street view images for a specific right-turn at Berlin. The images reveal that the right-turn is blocked for maintenance

Directions [15] and the Bing Maps Routes [26] APIs. Using two in instead of just one (the Google Directions API as in [8]), we will not only provide a more credible verification of our results, but also assess the difference of the results provided by separate map vendors.

The Google Directions API and the Bing Maps Routes API are two public REST APIs that allow the calculation of directions between locations using HTTP requests. For both services, users may search for directions using different transportation modes, include driving, transit, walking, and cycling. Directions may specify origins, destinations, and via-waypoints either as text strings or as latitude/longitude coordinates.

The Google Directions API allows only 2,500 directions requests per $24h$ period from a single IP address (free service). Contrarily, the Bing Maps Routes API offers a free 90-day Trial Key that allows one to evaluate Bing Maps for development and

(a) Photo 1 at location A (tail vertex of entrance edge (A, B))

(b) Photo 2 at location AB (middle of entrance edge)

(c) Photo 3 at location B (intersection)

(d) Photo 4 at location B (intersection)

Figure 14: Batch street view images for a specific left-turn at Berlin. The images reveal that the left-turn leads to a dead-end

may be used for up to 10,000 transactions/routes calculations within a 30-day span during the evaluation period. In both cases, due to the aforementioned limits, by first identifying (a rather limited number) of candidate turning restrictions as our method does, we can verify our results within the limits allowed for free users (Bing Maps) and within a few days (Google Directions). In any such HTTP request to the APIs, certain parameters are required, while others are optional. The most important required parameters (relative to our problem) are:

- Origin (Google Directions) or waypoint.n (Bing Maps) - The origin location FROM which we want to calculate directions.
- Destination (Google Directions) or waypoint.n (Bing Maps)- The destination location TO which we want to calculate directions.

Two additional, optional parameters useful to our purpose are:

- Mode (Google Directions) or travelMode (Bing Maps). Both use driving as the default value - Specifies the mode of transport to use when calculating directions.

```
http://maps.googleapis.com/maps/api/directions/json?
origin={A_coordinates}&destination={C_coordinates}&
waypoints=via:{B_coordinates}&sensor=false
```

Figure 15: A sample Google Directions API request

```
http://dev.virtualearth.net/REST/v1/Routes?wayPoint.0=
{A_coordinates}&viaWaypoint.1={B_coordinates}&waypoint.
2={C_coordinates}&key=BingMapsKey
```

Figure 16: A sample Bing Maps Routes API request

- Waypoints (Google Directions) or viaWaypoint.n (Bing Maps) - For defining additional intermediate locations that the route must travel through.

Given the above and with reference to Figure 3 in which the Turn($A \rightarrow C$ *via* B) has a low frequency usage, an HTTP request to verify this candidate turning restriction would be similar to Figure 15 for Google Directions API and Figure 16 for Bing Maps Routes API. Both requests return a JSON object with the proposed route by the respective APIs. The process for verifying the turning restriction is described in Algorithm VerifyRestriction, which compares the distance / length (in meters) calculated by the APIs with the sum of lengths of edges (A, B) and (B, C). If the API provided distance is significantly greater than the sum of lengths of edges (A, B) and (B, C), then we may safely assume that indeed there is a turning restriction and the respective API has to follow a much longer route than simply $(A, B) \rightarrow (B, C)$.

VerifyRestriction($Turn(A \rightarrow C$ *via* $B)$)

1 $apiPath = DirectionsAPICall(A \rightarrow C$ *via* $B)$
2 **if** $length(apiPath) >> length(A \rightarrow B) + length(B \rightarrow C)$
3 $\quad TurningRestriction(A \rightarrow C$ *via* $B).verified =$ TRUE

In order to access both APIs, we implemented a Java command-line application that retrieves turns below a threshold frequency usage (5% in our case) from the datastores, constructs an appropriate request string similar to Figure 15 and 16 for each turn, and retrieves the distance of the route returned by each API. To avoid overloading the API servers and rejected requests, we enforced a $500ms$ gap between requests. The distance results returned from both APIs are also stored in the respective PostgreSQL datastore for easy access and querying.

An obvious problem to this approach for verifying results, is the usage limits of both APIs (especially the rather limited number of requests allowed by the Bing Maps Routes API for free users). Although we are dealing with road networks with hundreds of thousands of vertices, edges and possible turns, through our optimizations (see Section 3.3.2) and by restricting the usage of the APIs to strictly confirm the candidate prohibited turns found by our proposed method, we only need to evaluate a few thousands turns. The obtained results are presented in the following section.

Table 8: Differences on the results between Google Directions and Bing Maps Routes API

city	Total	Google < Bing	Google > Bing	Google = Bing
Athens	5,287	1,457 (27.6%)	3,705 (70.1%)	125 (2.4%)
Berlin	2,653	762 (28.7%)	1,794 (67.6%)	97 (3.7%)
Vienna	1,739	526 (30.2%)	1,186 (68.2%)	27 (1.6%)

4. Evaluation Results

Having developed the automatic verification method described in Section 3.4.4, for our crowdsourced candidate turning restrictions, the following section summarizes the assessment results produced by the respective method.

4.1. Verified turning restrictions

Our verification method, based on using the Google Directions API and Bing Maps Routes APIs, were described in Section 3.4.4. Before presenting the actual assessment results, we want to highlight the differences with respect to the data returned by the two APIs, shown in Table 8. Several interesting results emerge. Surprisingly, on only very rare occasions (less than 4%) the two APIs return the same route and route length. Besides algorithmic differences, this also shows the considerable differences between the map datasets used in each case. For most instances (> 67%), the Google Directions API returns a longer path than Bing Maps. This needs to be taken into account when trying to verify our candidate turning restrictions against the two APIs.

Tables 9 and 10 show the number of restrictions verified for both 5% and 2.5% implicit usage thresholds for each API, for both APIs and for either of the two APIs, as well as their respective percentages in comparison to the total candidate restrictions. We notice that the majority of the candidate restrictions are successfully verified by one of the mapping services' APIs. In fact, for the case of Athens and Vienna, more than 74% of the extracted turning restrictions are verified by at least one of the APIs. For Berlin, the verified restrictions are 66% using the 5% threshold and 72% in case of the 2.5% threshold. Another observation is that by moving from the 5% to the 2.5% threshold, the verified restrictions' percentage increases slightly, but, we are also missing a significant number of restrictions (compare columns "Either" for 5% and 2.5%). This means, that there is a sizable number of existing (and verified) restrictions "contained" in the turn usage interval between 2.5% and 5%, which testifies to our choice of the threshold value of 5%. Hence, all following results are based on this 5% threshold. Note, that usually the paths returned by both APIs are significantly larger (85-90% of the verified restrictions give at least two-times larger paths) than the sum of lengths (A, B) and (B, C) for the verified restrictions, which is also a very strong indication of the validity of our verification method.

In Section 3.4.2 we observed a correlation between the entrance - exit direction angle of each candidate turning restriction with the validity of the restriction. Table 11 shows now the results of an explicit comparison of the number of verified restrictions (either API) using 5% threshold to this direction difference.

The results clearly confirm our empirical observations of Section 3.4.2. *Almost all (99%) of U-turn turning restrictions discovered by our method are verified by the*

Table 9: Number of verified restrictions for 5% implicit threshold

	candidate turning restrictions	# verified			
city	**Total**	**Google**	**Bing**	**Both**	**Either**
Athens	5,287	3,521 (67%)	2,624 (50%)	2,222 (42%)	3,923 (74%)
Berlin	2,653	1,546 (58%)	1,081 (41%)	886 (33%)	1,741 (66%)
Vienna	1,739	1,167 (67%)	683 (39%)	811 (47%)	1295 (74%)

Table 10: Number of verified restrictions for 2.5% implicit threshold

	candidate turning restrictions	# verified			
city	**Total**	**Google**	**Bing**	**Both**	**Either**
Athens	3,596	2,470 (69%)	1,921 (53%)	1,643 (46%)	2,748 (76%)
Berlin	1,582	1,040 (66%)	736 (47%)	632 (40%)	1,144 (72%)
Vienna	1,261	879 (70%)	629 (50%)	550 (44%)	958 (76%)

Table 11: Number of verified restrictions per angle for the 5% threshold

	Left		**Right**		**Straight**		**U-turns**	
City	**Total**	**Verified**	**Total**	**Verified**	**Total**	**Verified**	**Total**	**Verified**
Athens	2,402	1,829 (76%)	2,199	1,589 (72%)	342	163 (48%)	344	342 (99%)
Berlin	1,714	1,058 (62%)	494	264 (53%)	43	20 (47%)	402	399 (99%)
Vienna	779	584 (75%)	521	348 (67%)	183	108 (59%)	256	255 (100%)

APIs. In contrast, "straight" turning restrictions (with small direction changes between entrance and exit edges) show a rather small verification rate of only about 50%. This is a strong indication that this particular type of turning restriction is more susceptible to errors and therefore additional means of verification are required. Moreover, the fact that many of these restrictions are highway exits is clearly confirmed by the rather limited number of those restrictions encountered in Berlin. In this case, the road network considered did not include the inner-city highways, which we did consider for Athens and Vienna. On the other hand, left turns show slightly better results than right turns. However both cases exhibit a similar mean verification percentage as shown in Table 9.

Finally, Table 12 compares the total turns, examined turns, candidate and verified turning restrictions to the turning restrictions of the OSM datasets for the three respective cities. The results are very encouraging. Instead of examining hundreds of thousands of turns (Column 2) and focusing only on intersection vertices connecting major roads and utilizing historic map-matched trajectories, we discovered only a few thousand candidate turning restrictions (Column 4) that needed verification. By using the Google Directions and the Bing Maps Routes API, we could verify most of the identified candidate turning restrictions (Column 5). Also, the number of verified turning restrictions (Column 5) is significantly larger than the restrictions existing in the original datasets (Column 6). Especially for Athens, the number of verified turning restrictions is 18× larger than those existing in the current OSM dataset. Even for Vienna and Berlin the number of the verified prohibited turns is still 1.5 – 2.6× larger than

Table 12: Total turning restrictions results for 5% implicit threshold in comparison to existing OSM's restrictions

city	total turns	examined turns	candidate turning restrictions	verified turning restrictions	OSM turning restrictions
Athens	900,397	75,552	5,287	**3,923**	214
Berlin	252,271	44,636	2,653	**1,741**	668
Vienna	256,185	36,484	1,739	**1,295**	799

those existing in the OSM dataset. Our results lead us to assume that for countries and respective cities with worse map coverage, e.g., Albania, Montenegro, the proposed approach could significantly improve map datasets.

4.2. False positives?

Another important question is what we really can infer for those turning restrictions that were not verified by either API. Here, we refer to the unverified left and right turns for which the distances returned by the Google Directions and Bing Maps Routes API are quite similar to the sum of lengths of their constituent edges (A, B) and (B, C). We examined several of these cases and found that for a small number of routes (almost 1% for all three cities of those unverified left and right restrictions) that the distances returned by the API is *less* than 90% of the sum of lengths of the constituent edges (A, B) and (B, C), i.e., there is a shorter route than making a simple turn. When examining these anomalies, we found that most of the times there was an inconsistency between OSM and the commercial map dataset. For these cases, as to whether the turn is actually allowed or not is very debatable.

Still, even if we assume that all unverified left and right turning restrictions are indeed permitted, i.e., our method produces false-positives, we cannot ignore the fact that only *a very small percentage of the professional drivers we monitored actually use them*. Hence, a good-quality shortest-path solution *would still have to penalize (by increasing the respective turn cost) such "unappealing" turns*. As such, even unverified turning restrictions are still useful in revealing typical drivers' patterns and behaviors.

5. Conclusion and Future Work

This work proposed a new and efficient, semi-automatic way to infer and identify turning restrictions for OpenStreetMap data by utilizing historic map-matched trajectories from an existing fleet management service. Our experimentation covered three major European cities and a period of twelve months. Overall, $66-74\%$ of the turning restrictions we identified, was successfully verified through a rigorous verification method, including visual inspection with a mapping application, satellite imagery, batch retrieving of street view images and the use of public mapping APIs. However, the most important outcome of our work is, that we have identified and verified $2-18\times$ more turning restrictions than those existing in the current OSM dataset. This impressive feat testifies to the credibility of our method.

To the best of our knowledge, this is the first work to utilize historic map-matched trajectories for such a task. This is after all, the main contribution of our work, since the few existing works addressing the related subject of intersection delays base their research on raw GPS trajectories. In addition, most previous works use either simulated data or data covering smaller time periods (up to a month) and were focused on a particular geographic area. Our results are based on three European cities, originate from three medium to large vehicle fleets of 2,000-5,000 vehicles each, and cover an entire year of operation. The results (in terms of discovered data) for the three areas were almost identical, which further testifies to the robustness and validity of our method. In addition, by comparing our results with two external mapping APIs (the Google Directions and the Bing Maps Routes APIs) we show the validity of our approach.

We can propose several interesting directions for future work. Since the proposed method is able to identify and confirm turning restrictions in the OSM data, we can expand it to automatically contribute those verified restrictions back to the OSM project. In this way, the outcome of our work could be shared by the mapping community and, thus, increase its impact. Our results could further improve the quality of existing map-matching algorithms. Many of these algorithms use partial shortest-path calculations to align the raw GPS traces to the road network graph. Up until now, those shortest-path computations do not take turning restrictions into account. Since our approach identifies such restrictions, those newly found constraints could be integrated back into the map-matching algorithms to further improve their results. In this way, for the first time, a self-improving, evolutionary map-matching algorithm might become a reality.

Acknowledgments

The research leading to these results has received funding by the NGA NURI grant HM02101410004 and from the European Union 7th Framework Programme "Simple-Fleet" (`http://www.simplefleet.eu`, grant agreement No. FP7-ICT-2011-SME-DCL-296423.

The authors would additionally like to thank Sotiris Brakatsoulas for his work on the map-matched trajectories, in the original work of [8].

References

[1] M. Ahmed, S. Karagiorgou, D. Pfoser, C. Wenk, A comparison and evaluation of map construction algorithms using vehicle tracking data, GeoInformatica 19 (3) (2014) 601–632.
URL `http://dx.doi.org/10.1007/s10707-014-0222-6`

[2] Amazon, Mechanical Turk, `https://www.mturk.com/mturk/help?helpPage=overview` (2016).

[3] J. Ban, R. Herring, P. Hao, A. M. Bayen, Delay pattern estimation for signalized intersections using sampled travel times, Transportation Research Record (2009) 109–119.

[4] S. Brakatsoulas, D. Pfoser, R. Salas, C. Wenk, On map-matching vehicle tracking data, in: Proc. 31st VLDB Conference, 2005, pp. 853–864.

[5] Complex Engineered Systems Lab, Taxi Trajectory Open Dataset [Online], http://sensor.ee.tsinghua.edu.cn/datasets.php (2010).

[6] D. Delling, A. V. Goldberg, T. Pajor, R. F. Werneck, Customizable route planning, in: Proceedings of the 10th international conference on Experimental algorithms, SEA'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 376–387.
URL http://dl.acm.org/citation.cfm?id=2008623.2008657

[7] A. Efentakis, S. Brakatsoulas, N. Grivas, G. Lamprianidis, K. Patroumpas, D. Pfoser, Towards a flexible and scalable fleet management service, in: Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS '13, ACM, New York, NY, USA, 2013, pp. 79:79–79:84.
URL http://doi.acm.org/10.1145/2533828.2533835

[8] A. Efentakis, S. Brakatsoulas, N. Grivas, D. Pfoser, Crowdsourcing turning restrictions for OpenStreetMap, in: Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014., 2014, pp. 355–362.
URL http://ceur-ws.org/Vol-1133/paper-56.pdf

[9] A. Efentakis, N. Grivas, G. Lamprianidis, G. Magenschab, D. Pfoser, Isochrones, Traffic and DEMOgraphics, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL'13, ACM, New York, NY, USA, 2013, pp. 548–551.
URL http://doi.acm.org/10.1145/2525314.2525325

[10] A. Efentakis, D. Pfoser, Optimizing landmark-based routing and preprocessing, in: Proceedings of the 6th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS '13, New York, NY, USA, 2013, pp. 25–30.
URL http://doi.acm.org/10.1145/2533828.2533838

[11] M. F. Goodchild, Citizens as voluntary sensors: spatial data infrastructure in the world of web 2.0, International Journal of Spatial Data Infrastructures Research 2 (2007) 24–32.

[12] Google, Google Earth [Online], https://www.google.com/earth/ (2014).

[13] Google, Keyhole Markup Language [Online], https://developers.google.com/kml/documentation/kmlreference (2014).

[14] Google, Street View [Online], https://www.google.com/maps/views/streetview?gl=us (2014).

[15] Google, The Directions API [Online], https://developers.google.com/maps/documentation/directions/ (2014).

[16] Google, Google Street View Image API. Introduction, https://developers.google.com/maps/documentation/streetview/intro (2016).

[17] Google, Maps JavaScript API. Geometry Library, https://developers.google.com/maps/documentation/javascript/geometry (2016).

[18] Google, Street View Image API , https://developers.google.com/maps/documentation/streetview/ (2016).

[19] M. Haklay, P. Weber, Openstreetmap: User-generated street maps, IEEE Pervasive Computing 7 (4) (2008) 12–18.

[20] T. Hastie, W. Stuetzle, Principal curves, Journal of the American Statistical Association 84 (406) (1989) 502–516.
URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478797

[21] R. Herring, P. Abbeel, A. Hofleitner, A. Bayen, Estimating arterial traffic conditions using sparse probe data, Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems, September 19-22, Madeira Island, Portugal (2010) 929–936.

[22] L. Kabrt, Travel Time Analysis, http://code.google.com/p/traveltimeanalysis/source/browse (2010).

[23] Laboratory for Software Technology, Computer Science Department, ETH Zurich, Realistic Vehicular Traces [Online], http://www.lst.inf.ethz.ch/research/ad-hoc/car-traces/index.html#traces (2011).

[24] X. Liu, F. Lu, H. Zhang, P. Qiu, Estimating beijing's travel delays at intersections with floating car data, in: Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS '12, ACM, New York, NY, USA, 2012, pp. 14–19.
URL http://doi.acm.org/10.1145/2442942.2442946

[25] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, Y. Huang, Map-matching for low-sampling-rate gps trajectories, in: Proc. 17th ACM SIGSPATIAL GIS conf., GIS '09, ACM, New York, NY, USA, 2009, pp. 352–361.
URL http://doi.acm.org/10.1145/1653771.1653820

[26] B. Maps, Routes API [Online], http://msdn.microsoft.com/en-us/library/ff701705.aspx (2014).

[27] Microsoft Research, T-Drive trajectory data sample [Online], http://research.microsoft.com/apps/pubs/?id=152883 (2011).

[28] O. A. Nielsen, R. D. Frederiksen, N. Simonsen, Using expert system rules to establish data for intersections and turns in road networks, International Transactions in Operational Research 5 (6) (1998) 569 – 581.
URL http://www.sciencedirect.com/science/article/pii/S0969601698000422

[29] OpenStreetMap, Stats - OpenStreetMap wiki [Online], http://wiki.openstreetmap.org/wiki/Stats#OpenStreetMap_Statistics_Available (2011).

[30] OpenStreetMap, Relation:restriction [Online], http://wiki.openstreetmap.org/wiki/Relation:restriction (2014).

[31] OpenStreetMap Foundation, Bulk GPX track data [Online], http://blog.osmfoundation.org/2013/04/12/bulk-gpx-track-data (2014).

[32] D. Pfoser, C. S. Jensen, Capturing the uncertainty of moving-object representations, in: Proceedings of the 6th International Symposium on Advances in Spatial Databases, SSD '99, London, UK, UK, 1999, pp. 111–132.
URL http://dl.acm.org/citation.cfm?id=647226.719082

[33] PostGIS, Spatial and Geographic objects for PostgreSQL [Online], http://postgis.net/ (2015).

[34] PostgreSQL, The world's most advanced open source database [online], http://www.postgresql.org/ (2015).

[35] QGIS, A Free and Open Source Geographic Information System [Online], http://www.qgis.org/ (2014).

[36] Sprockets, Google Places API and Google Street View Image API in Java, https://github.com/pushbit/sprockets (2016).

[37] L. Sun, An approach for intersection delay estimate based on floating vehicles, Dissertation for Master Degree. Beijing: Beijing University of Technology(in Chinese), 2007.

[38] F. Viti, H. J. van Zuylen, Modeling queues at signalized intersections, Transportation Research Record: Journal of the Transportation Research Board (1883) (2004) 68–77.
URL http://trb.metapress.com/content/U20Q1L6121J658RK

[39] C. Wenk, R. Salas, D. Pfoser, Addressing the need for map-matching speed: Localizing global curve-matching algorithms, in: Proc. 18th SSDBM conf., 2006, pp. 379–388.

[40] S. Winter, Modeling costs of turns in route planning, GeoInformatica 6 (4) (2002) 345–361.
URL http://dx.doi.org/10.1023/A%3A1020853410145

[41] H. Zhang, F. Lu, L. Zhou, Y. Duan, Computing turn delay in city road network with gps collected trajectories, in: Proceedings of the 2011 International Workshop on Trajectory Data Mining and Analysis, TDMA '11, ACM, New York, NY, USA, 2011, pp. 45–52.
URL http://doi.acm.org/10.1145/2030080.2030090

[42] M. Zhao, X. Li, Deriving average delay of traffic flow around intersections from vehicle trajectory data, Frontiers of Earth Science 7 (1) (2013) 28–33.
URL http://dx.doi.org/10.1007/s11707-012-0341-z

[43] Y. Zheng, Trajectory data mining: An overview, ACM Trans. Intell. Syst. Technol. 6 (3) (2015) 29:1–29:41.
URL http://doi.acm.org/10.1145/2743025

[44] Y. Zheng, X. Zhou (eds.), Computing with Spatial Trajectories, Springer, 2011.
URL http://dblp.uni-trier.de/db/books/collections/zheng2011.html