Crowdsourcing turning restrictions for OpenStreetMap

Alexandros Efentakis
Research Center "Athena"
Artemidos 6, Marousi 15125, Greece
efentakis@imis.athena-innovation.gr

Sotiris Brakatsoulas
Research Center "Athena"
Artemidos 6, Marousi 15125, Greece
mprakats@ceid.upatras.gr

Nikos Grivas
Research Center "Athena"
Artemidos 6, Marousi 15125, Greece
grivas@imis.athena-innovation.gr

Dieter Pfoser*
Department of Geography and
GeoInformation Science
George Mason University
4400 University Drive, MS 6C3
Fairfax VA 22030-4444
dpfoser@gmu.edu

ABSTRACT

The abundance of GPS tracking data due to the emergence and popularity of smartphones has fuelled significant research around GPS trajectories and map-matching algorithms. Unfortunately, none of this previous research addresses the issue of identifying turning restrictions in the underlying road network graph. Our latest research endeavour remedies this, by proposing a novel, straightforward and effective way to infer turning restrictions for OpenStreetMap data, by utilizing historic map-matching results from an existing fleet management service. Our experimental evaluation based on the results acquired for three European cities within an one-year period, proves the robustness and credibility of our method.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Design

Keywords

Crowdsourcing, Turning Restrictions, Map-matching, OpenStreetMap

1. INTRODUCTION

Our latest research efforts of [22] aimed towards combining state-of-the-art research about road networks, Floating Car Data, map-matching, historic speed profile computation, live-traffic assessment and time-dependent shortest-path computation to provide an efficient, yet economical fleet management solution. This process was documented in [5] and its result, our fully functional SimpleFleet fleet management system and its accompanying demo [7] now cover the urban regions of three European metropolitan cities namely: Athens (Greece), Berlin (Germany) and Vienna (Austria). Creating the actual service required several intermediate steps such as: Creating road network graphs from OpenStreetMaps data, collecting a large amount of Floating Car Data (FCD) from fleet vehicles, applying state-of-the-art map-matching algorithms on this data for aligning the GPS traces to the road network graph and consequently producing high-quality historic speed profiles along with frequently updated live-traffic assessment. This combination of live-traffic information and speed profiles was subsequently used to provide up-to-date live-traffic shortest-path and isochrone computation (refreshed every 5 minutes). In addition, our recent work of [6] has already combined the live-traffic isochrone functionality of this system with demographic / business data to showcase the impact of traffic fluctuations in a geomarketing context. There, one can see in a quantitative way the considerable importance of traffic and how it should affect geomarketing decisions.

It is obvious that the success of every fleet management solution depends highly on the quality of the road network graph. Although OpenStreetMap (OSM) is a crowdsourced, high-quality, frequently updated road network dataset, an entire year of running the SimpleFleet service for its three urban regions has revealed a inherent limitation of the OSM dataset: Its limited information for turning restrictions, i.e., a transition from one network edge to another (via an intersection node) that is prohibited due to local traffic rules. It is not that OSM data does not support turning restrictions: an additional relation tag (Relation:restriction [19]) is defined for describing such restrictions. The problem is that only a small number of users contribute to this information. While OSM includes more than 2.1 billion Nodes, Ways and Relations [17], less than 230,000 relations actually represent turning restrictions [19]. This is even more obvious, when we look at our individual test cases: For the city of Athens and its road network of 277K nodes, only 214 turning restrictions have been recorded by OSM users. This lack of data is to be expected: there are no public datasets for traffic signs easily found (if any), satellite imagery cannot reveal this information and adding turning restrictions even for a single road is extremely time-consuming. Keep in mind that turning restrictions do not include one-way streets. Such streets are easily modeled in every directed graph representation, are easily recognized by users and OSM data has extensive coverage for them.

*On leave from Research Center "Athena", Greece.

Turning restrictions on road networks are especially important for any routing / isochrone service. While a lot of scientific literature has focused on time-dependency on road networks (due to the fluctuation of traffic) and consequently the implementation of efficient shortest-path algorithms that address this issue, there is a limited number of works that deal with turning restrictions. This is mostly due to the fact that *"no publicly-available realistic turn data exist"*[4]. Note that turning restrictions have a more dramatic impact on shortest paths provided by mapping services than traffic: While ignoring traffic returns a valid, yet suboptimal route to the user, ignoring turning restrictions returns erroneous paths that may lead to accidents. As a result, providing an efficient method for automatically identifying turning restrictions is extremely important.

When searching existing scientific literature for solving this issue, we found a significant body of work based on Floating Car Data (FCD) in various areas (see [29] 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 [1, 13, 23, 27, 28]. It was really a surprise, when we did not find any literature devoted to the results of the map-matching (MM) algorithms, as though those results may strictly be used for travel-time estimation. In this sense, we beg to differ, because the main focus of this work is to automatically identify / infer turning restrictions in the OSM dataset by utilizing historic map-matching results, i.e., we crowdsource the identification of turning restrictions to local vehicle drivers by mining the map-matching results produced by them, when traversing the road network graph. This is also the true novelty of our contribution: instead of using the GPS trajectories directly, we use the map-matching results derived from them. Our approach makes sense: In comparison to raw GPS traces, map-matching results are: a) more condensed, since instead of random locations in the plane we have edge sequences and b) less error-prone (if an efficient map-matching algorithm is used) since they are interpolated with the actual road network. Therefore it is logical to utilize those historic results to extrapolate this additional meaningful information, instead of using raw FCD like most previous works. To the best of our knowledge, we are the first to utilize map-matching results for such a task. Although our method uses OSM data, it may also be used for any road network dataset, in cases where the road network evolves faster than typical map updates. In those cases, identifying added turn-restrictions, as soon as they occur, is extremely important.

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-matching results. 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 operating vehicles equipped with GPS-enabled devices 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*"[10]. 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"* [20]. Therefore the observed GPS positions often need to be aligned with the road network graph. This process is called map-matching. As a result, a map-matching (MM) algorithm accepts as input a vehicle's GPS trajectory and outputs a path / ordered sequence of road network graph edges that this vehicle has traversed, along with travel time information, i.e., how long did it take for the specific vehicle to traverse the calculated path. In our SimpleFleet service [5] we used two different MM algorithms: the Fréchet-distance-based curve matching algorithm of [2, 25] and the [11] implementation of the ST-matching algorithm [14]. However, both implementations were significantly enhanced to handle live incoming FCD streams.

Despite their inherent imprecision and the usually low sampling rate of available datasets, latest years saw an explosion of research around GPS trajectories ([29] presents a partial overview of GPS related research). Nevertheless, so far, only a limited portion of this research focused on road network intersections. This is a major oversight, since intersections are important components of urban road networks and contribute much to the total travel time cost [16, 24]. [16] concludes that intersection delays i.e., the turn cost associated with the continuation of travel between edges via an intersection node [26] contribute to 17-35% of the total travel time, according to a conducted survey in the Copenhagen urban area.

The few research works around road network intersections that actually exist, focused on estimating intersection delays based on the available Floating Car Data. Some researchers have utilized the historical mean method to calculate the intersection delays ([23], [28]), while other authors employ piecewise linear interpolation ([1], [27]). Additional works employed the principal curves method [9] to overcome data sparseness of Floating Car Data and calculate turn delays tables for the region of Beijing [13].

Although turn costs / 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 in several levels. First of all, for previous approaches to calculate turn cost for a specific turn, many vehicles need to actually use it. On the contrary, we identify turning restrictions by focusing on turns with no available data. Second, previous approaches use GPS trajectories; we use map-matching results. Third, since they are based on data mining techniques they may only verify results by dividing the original GPS dataset into a training and a test set. We use an independent mapping service to verify our findings. Lastly, most publicly available GPS datasets are either simulated [12], focused on a specific city [3, 15] or for limited time periods (a day, week or month for [12, 15, 3] respectively). Contrarily, our conclusions are based on three different European cities and fleets of $2,000-5,000$ vehicles per city covering a full 12-month period. Since our results are almost identical for each city (see Sec. 4) it is obvious that our method is both realistic and robust.

In the next section we will describe the basic methodology with all the practical details of our implementation.

3. CROWDSOURCING TURNING RESTRICTIONS

In this section we are going to present basic information about the OpenStreetMap dataset and the methodology we used to infer information about turning restrictions from historic map-matching results, as well as the verification process used.

3.1 Preliminaries

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

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

Figure 1: Prohibitory traffic signs for turning restrictions

The degree of a vertex u, denoted as $deg(u)$, is the number of edges incident to the vertex. *Intersection nodes* are the road network vertices with node degree larger 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 nodes that is prohibited due to local traffic rules. Such turning restrictions are represented in the road network as traffic signs (see Fig. 1). For the remainder of the paper, we only deal with those edge sequences that consist of a single ordered pair of two edges connected via a single intersection node, since those represent the majority of turning restrictions on typical road networks. Note that turning restrictions do not refer to one-way streets, because a) even a single edge may be marked as unidirectional and b) turning restrictions may refer to roads that are bidirectional but it is only their sequence that is prohibited. In addition, unidirectional streets are easy to model in every directed graph representation, whereas turning restrictions is a distinguishing trait of road networks that differentiates them from most other real-world networks.

A GPS trajectory describing a vehicle movement, consists of a sequence of measurements with latitude, longitude and timestamp information for the same vehicle ID. *Map-matching* is the process of aligning such a trajectory against the underlying road network graph. Consequently, a map matching algorithm accepts as input a single vehicle's trajectory and outputs a path / ordered sequence of road network graph edges that this vehicle has traversed, 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 the paper, when we are talking about map-matching results we only refer to the calculated vehicle's path and not the associated travel time information.

3.2 OpenStreetMap and turn restrictions

OpenStreetMap [18] is a free, editable map of the entire world. Unlike proprietary datasets, the OpenStreetMap license allows free access to the full map dataset. This massive amount of data may be downloaded in full but is also available in other useful formats such as mapping, geocoding or other web services. Users participate in the OpenStreetMap (OSM) community by providing feedback and editing the map. Although OpenStreetMap contains an appropriate relation tag (Relation:restriction [19]) for describing turning restrictions, only a small number of OSM users contribute to this information, due to its inherent difficulty. This statement was easy to confirm for the three European cities (Athens, Berlin, Vienna) covered by our service. Results retrieved in September 2013 are shown in Table 1.

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 <=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%

Table 1 shows that available data for turning restrictions is particularly low, especially in comparison to the sizes of the OSM road networks of the three cities covered by our service, as shown in Table 2. We got similar results (or even worse) for other European cities, especially for countries with less detailed maps (e.g., Albania, Montenegro). Given the above issue, we decided to take advantage of the large amount of historic map-matching results created through our fleet management service and infer / identify turning restrictions in the OSM road network. Our method is straightforward and efficient: We aim to discover turns that, although they are allowed in the original dataset, in practice they are rarely, if ever, used by the vehicle drivers. Such turns that exhibit such low usage frequency have a very high probability to be actually prohibited.

3.3 Methodology

The basic methodology for inferring / identifying OSM turning restrictions by using historic map-matching results created by our fleet management service may be described by the following simplified diagram of Fig. 2. The independent stages of this process will be thoroughly discussed in the following sections.

3.3.1 Input data

In our SimpleFleet service [5], GPS traces of fleet vehicles for the three European cities, arrive in a streaming fashion. Specifically, for each of those urban areas, we are dealing with 2,000-5,000 fleet vehicles producing a data point (GPS position sample) every 60 -180s. GPS trajectories for each vehicle are subsequently map-matched, in order to align those GPS traces to the underlying OSM road network graph. The result of this process is an ordered sequence of road edges that a vehicle has traversed. The traffic datastore of the service, including the OSM road network graphs, Floating Car Data and map-matching results is implemented by PostGIS enabled PostgreSQL databases (one database instance per city) for permanent storage.

As a vehicle moves in the road network, it traverses roads of varying importance to traffic (refer to Table 3 for the distribution of the OSM road networks per their respective category for the three SimpleFleet cities). However, not all these roads are important to

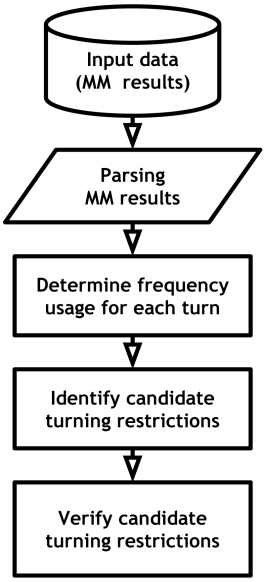


Figure 2: Methodology for identifying OSM turning restrictions by using historic map-matching results

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

traffic, so it made sense to reduce the bulk of data stored. Towards this goal, a separate process eliminates map-matched edges that belong to edges of road category greater than 10 (i.e., OSM categories for unclassified, road, residential and living street - see Table 3). Depending on the time period and the traffic patterns in each city, about 12-15% of the map-matched records are subsequently dropped after the map-matching process.

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

After one year of running the service (from Oct 2012 to end of September 2013), we have accumulated several Gb of compressed historic map-matching results for each of the cities covered by our service. The challenge is how to utilize this significant wealth of data to infer turning restrictions for the respective OSM road networks. This process will be described in the following sections.

Table 4: Typical size of compressed MM results 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

Table 5: Total counted instances for all examined 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

3.3.2 Parsing map-matching results and optimizations

The basic focus of our work is to identify those turns (i.e., ordered pair of edges connected via an intersection node) that exhibit unusually low frequency of usage by vehicles. The frequency of usage will be determined by parsing the compressed archives of the historic map-matching results produced for the three cities during the operational period of our service (Oct 2012 to end of September 2013). Since, the respective OSM road networks comprise of hundreds of thousand of nodes and edges (see Table 2) we need to somehow limit the possible 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 accumulate information for vertices of degree 2 (with just one incoming and one outgoing edge) or lower, since in those vertices the driver has no choice but to continue in one direction. The second optimization had to do with the available input data. Since map-matching results only include larger roads that correspond to OSM categories ≤ 10 (see Section 3.3.1), we are only interested in those intersection vertices connected to such roads. This way we miss some intersection vertices (strictly connected to smaller categories roads) but this is a necessary compromise to minimize our scope. In addition, intersection vertices that connect to major roads are more likely to be used by many vehicle drivers and as a result they significantly influence traffic behaviour. Table 2 shows that the number of intersection vertices connecting major roads are less than 25% of total vertices for all cities covered by our service.

As a result of those two optimizations, the number of unique turns / pairs of consecutive edges we need to examine is significantly smaller than the available road network edges, which is a considerable improvement (see Table 5). Since the OSM road networks of each city are stored in the respective PostgreSQL datastores, determining intersection vertices of interest and their corresponding turns is easily accomplished with plain SQL commands.

After determining the unique turns for examination, we implemented a custom Java app that parses the compressed archives of historic map-matching results in our disposal (see Section 3.3.1), counts the instances encountered for each turn and stores results in the respective PostgreSQL datastores. The total counted instances for all examined turns during our one-year testing period (starting Oct 2012) are shown in Table 5. Results also show, that on average for every intersection vertex connected to major roads (i.e., their road category $\leq$ 10), we have 4,137 (Athens) - 97,304 (Berlin) counted instances of turns, which means that we have a sufficiently large number of measurements per intersection vertex.

3.3.3 *Identifying candidate turning restrictions*

After enumerating instances for every unique turn we needed to examine, we must discover which of those turns are rarely used. Since both turns and results of the enumerating 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 follow after traversing a specific entrance edge (and a specific direction). Subsequently, each turn belongs to only a single 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 percentage of usage for each one. An example group for a specific entrance edge is shown in Fig. 3.

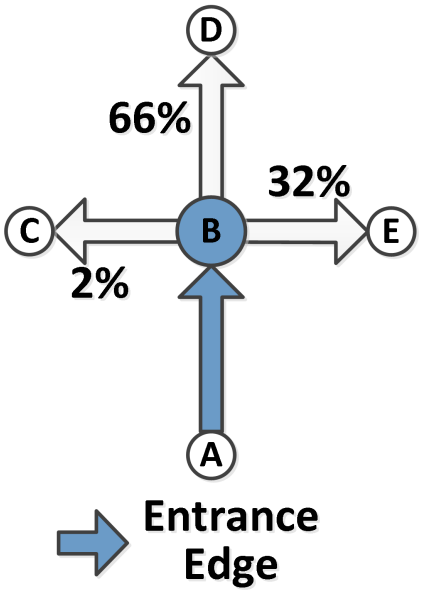


Figure 3: A simple example of grouping turns per entrance edge (A, B) at an intersection vertex (B) for calculating usage percentage per turn

As we notice in the example group of Fig. 3, most drivers continue straight when they traverse the entrance edge (A, B) that leads to the intersection vertex B. Some others prefer to turn right. But a very small percentage of them (2%) turn left. This is a very strong indication that this low percent actually represents erroneous map-matching results (even the most efficient MM algorithm has a small error rate) and indeed this particular left-turn is prohibited, even if OpenStreetMap lacks this information.

Next, we made the rather logical assumption that turns with lower frequency percentage than an implicit 5% threshold are probably prohibited. Of course this threshold is arbitrary but as results will show, it is pretty accurate as well. Table 6 shows the number of the candidate turning restrictions we have 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, to conclude if it is a straight, right, left or U-turn. The direction calculation is very easy, since we have already stored the inclination of each edge in the respective datastore, since this information was needed for the isochrone functionality of our service. Table 7 shows the percentages of the turns direction categorization for the candidate prohibited turns we have extracted. As expected, most of them (particularly in Berlin and Vienna) represent left-turns.

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%

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%

Still, after determining the candidate prohibited turns for each city covered by our service, we still need to find additional means to verify the validity of our claims. This process is discussed in the following section.

3.4 Verifying results

There are two basic ways to validate the discovered candidate turning restrictions. Firstly, we can visualize each one of those candidate restrictions. Secondly, we can use an external mapping service and cross-check if we get similar results to ours. We used both ways: Results are presented in the following sections.

3.4.1 *Visual Inspection*

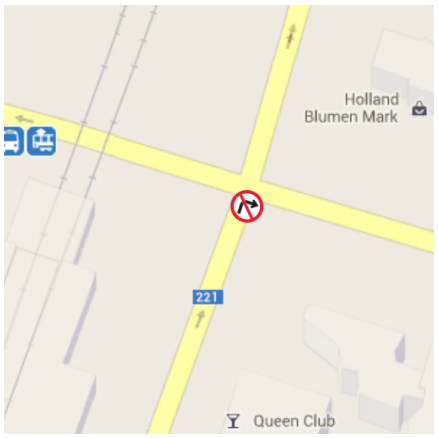
In order to confirm our results, we need to visualize the candidate restricted turns. Each such turn may be represented with the appropriate traffic sign (depending on the direction of the turn according to Table 7) located in the corresponding intersection vertex coordinates. For that purpose, we used QGIS [21], a popular, free and open source GIS application that runs in all major operation systems. We used a Google Maps Layer in QGIS as the background map layer, in order to compare results to an external mapping service. Figure 4 shows some typical examples of the results of our visualization process for some of the candidate turning restrictions:

- Figure 4(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 important and critical turning restrictions are discovered through our method.
- Figure 4(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 4(c) shows that the Google Maps layer visually confirms the discovered turning restriction.

Conclusively, visual inspection of our findings shows that OpenStreetMap data (despite its high quality) is still missing some vital information, especially as turning restrictions are concerned. By taking advantage of historic map-matching results, we have tracked and visualized many such problematic cases. As a result, our methodology may be used to further enrich the collection of turn-restrictions available in OpenStreetMap. Yet, to further verify and quantify results, an additional high quality mapping service could be used for cross-checking the validity of our findings. This process will be described in the following section.

(a) Turning restrictions visualization in Athens

(b) Turning restrictions visualization in Berlin



(c) Turning restrictions visualization in Vienna

Figure 4: Visualizing turning restriction with QGIS

3.4.2 *Sourcing an external mapping service*

Although visual inspection is a convincing, qualitative way to validate results, it would be best if we could further verify and quantify our findings through an automatic process. One solution to this problem is to use the Google Directions API [8]. Although Google Maps is not guaranteed to be perfect, yet, it is a global, popular, commercial, alternative solution to the crowdsourced user-generated data of OpenStreetMap.

The Google Directions API is a service that calculates directions between locations using HTTP requests. Users may search for directions for several transportation modes, include transit, driving, walking or cycling. Directions may specify origins, destinations and waypoints either as text strings or as latitude/longitude coordinates. The Directions API can return multi-part directions using a series of waypoints [8].

The Google Directions API allows only 2,500 directions requests per 24 hour period from a single IP address (for free users). This is the reason, why it was important to first identify (a rather limited number) of candidate turning restrictions, so that the required requests to the API could finish in a few days. In any such HTTP request to the API, certain parameters are required while others are optional. As is standard in URLs, all parameters are separated using the ampersand (&) character. The most important required parameters (relative to our problem) are:

- Origin - The address or textual latitude/longitude value FROM which we wish to calculate directions.
- Destination - The address or textual latitude/longitude value TO which we wish to calculate directions.

Two additional, optional parameters useful to our purpose are:

- Mode (defaults to driving) - Specifies the mode of transport to use when calculating directions. The value can be "driving", "walking", "bicycling" or "transit".
- Waypoints - Specifies an array of waypoints. Waypoints alter a route by routing it through the specified location(s). A waypoint is specified as either a latitude/longitude coordinate or as an address which will be geocoded.

Since our requests concern driving directions, there is no need to specify the "mode" parameter as it defaults to "driving". As for waypoints, we only need one waypoint per request. This waypoint should not be a stopover but serves just to influence the route. This may be done by prefixing the waypoint with the prefix "via:" (in the respective API call). This way, a single-part route is returned.

```
http://maps.googleapis.com/maps/
api/directions/json?origin={A_
coordinates}&destination={C_
coordinates}&waypoints=via:{B_
coordinates}&sensor=false
```

Figure 5: A sample Google Directions API request

Given the above and with reference to Fig. 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 Fig. 5. This request returns a JSON object with the proposed route by the API. The process for verifying the turning restriction is described in Algorithm 3.1, which compares the distance (in meters) calculated by the Google Directions API with the sum of lengths of edges (A, B) and (B, C). If the Google distance is significantly greater (over 10%) than OSM's distance, then we may safely assume that indeed there is a turning restriction and the Google Directions API has to follow a much longer route than simply $(A, B) \rightarrow (B, C)$.

Algorithm 3.1: VERIFYTURN($Turn(A \rightarrow C \text{ via } B)$)

$GooglePath \leftarrow DirectionsAPICall(A \rightarrow C \text{ via } B)$
if $dist(GooglePath) >> dist(A \rightarrow B) + dist(B \rightarrow C)$
then $\{Turn(A \rightarrow C \text{ via } B)$ *is verified*

In order to access the Google Directions API, we implemented a Java command-line application that retrieves turns below a threshold frequency usage (5% in our case) from the datastore, constructs an appropriate request string similar to Fig. 5 for each turn and retrieves the distance of the route returned by the API. To avoid overloading Google's servers and getting rejected requests, we have enforced a 500 ms gap between requests. API distance results are also stored in the respective PostgreSQL datastore for easy accessing.

An obvious problem to this approach for verifying results, is the usage limits of the Google Directions API. Although we are dealing with road networks with hundreds of thousands of nodes, edges and

Table 8: Number of verified restrictions for 5% and 2.5% implicit threshold

city	candidate turning restrictions		# verified		verified (%)	
	5%	**2.5%**	**5%**	**2.5%**	**5%**	**2.5%**
Athens	5,287	3,596	3,517	2,471	67%	69%
Berlin	2,653	1,582	1,510	1,016	57%	64%
Vienna	1,739	1,261	1,172	880	67%	70%

possible turns, through our optimizations (see Section 3.3.2) and by restricting the usage of the API to strictly confirm the candidate prohibited turns found by our proposed method, we only need to check a few thousands turns. Even by not bypassing Google API's limits (by using different IP addresses) this process only takes 1-3 days per city (e.g., for Vienna it requires only a few hours). As a result, the API usage limits easily suffice for confirming our results. These results are presented in the following section.

4. RESULTS

This section summarizes the results produced by the Google Directions API verification process for all candidate turning restrictions in comparison to the original OpenStreetMap datasets.

4.1 Verified turning restrictions

The method used for comparing results of Google Directions API and the OSM distances for each turn was thoroughly explained in Section 3.4.2. In Table 8, we present the number of restrictions verified for both 5% and 2.5% implicit usage threshold, as well as their respective percentages in comparison to the total candidate restrictions. Keep in mind that usually the paths returned by the Google Directions API 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), which further proves the validity of the verification method used.

As we notice, the majority of the candidate restrictions are successfully verified by the Google Directions API. In fact, in Athens and Vienna more than 67% of the extracted turning restrictions are verified. In Berlin, the verified restrictions are about 57% for the 5% threshold and 64% for the 2.5% threshold. Another useful remark is that moving from the 5% to the 2.5% threshold, the verified restrictions' percentage is slightly increased but, in fact, we are missing a significant number of restrictions (compare columns "# verified" for 5% and 2.5%). This means, that there is a respectable amount of existing (and verified) restrictions even in the turn usage interval between 2.5% and 5%.

Finally, Table 9, compares total turns, examined turns, candidate and verified turning restrictions in comparison to the turning restrictions existing in the original OpenStreetMap datasets for the three respective cities. Results are quite impressive: Instead of examining hundreds of thousands of turns, by focusing on intersection nodes connecting major roads and utilizing historic map-matching results, we discovered only a few thousand candidate turning restrictions in need of verification. Next, by using the Google Directions API most of the candidate turning restrictions were successfully verified. But the most impressive fact of all, is that the number of verified turning restrictions is significantly larger than the restrictions existing in the original datasets. Especially in Athens, the number of verified turning restrictions is 16 times larger than those existing in the original OSM dataset. Even, in Vienna and Berlin the number of the verified prohibited turns is still 1.7 - 2.2 times larger than those existing in the original data. Our

Table 9: 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	**3,517**	214
Berlin	252,271	44,636	2,653	**1,510**	668
Vienna	256,185	36,484	1,739	**1,172**	799

results lead us to assume that in cities of European countries with less detailed maps (e.g., Albania, Montenegro) the situation will be similar to Athens or even worse.

4.2 False positives?

Another pending question is what can we really infer for those turning restrictions that were not verified by the Google Directions API. Most of the times, for those turns, the distance returned by the Google Directions API is quite similar to the sum of lengths (A, B) and (B, C). Still, there is also a non- negligible number of routes (5% for Athens, 2% for Berlin and 8% for Vienna of those unverified restrictions) where the distance returned by the API is less than 80% of the sum of lengths of edges (A, B) and (B, C). When we searched for those strange cases, most of the times there were serious inconsistencies between the two maps. In that case, if the turn is actually allowed or not is very debatable.

Still, even if we assume that all unverified turning restrictions are indeed allowed (i.e., our method produces false-positives) there is one fact we cannot ignore: *A very small percent of drivers actually use them*. In that case, a perfect shortest-path solution *would still have to penalize (by increasing the respective turn cost) such "unappealing" turns*. In this sense, even unverified turning restrictions returned by our method are still useful in revealing typical drivers' patterns and behaviors.

5. CONCLUSION AND FUTURE WORK

In this work we have proposed a new and efficient, semi-automatic way to infer / identify turning restrictions for OpenStreetMap data by utilizing historic map-matching results from an existing fleet management service, covering three major European cities, spanning a period of twelve months . Our method has proved solid: 57-67% of the turning restrictions we have extracted may be successfully verified. However, the most important result is that we have identified and verified 2-16 times more turning restrictions than those existing in the original datasets. This impressive feat proves the validity and credibility of our method.

To the best of our knowledge, we are the first to utilize historic map-matching results for such a task. This is after all, the main novelty of our work, since the few existing works that deal with the similar subject of intersection delays base their research on raw GPS trajectories. In addition, most previous works used either simulated data or data covering smaller time periods (up to a month) and were focused on a particular area. Our results are based on three European cities, originate from three medium / large fleets of 2,000-5,000 vehicles per city and cover an entire year of operation. Results for the three areas were almost identical, which further proves the robustness of our method. Moreover, by comparing our results with an external mapping service (the Google Directions API) we have shown the correctness of our approach. On a quite similar note, we also experimented with two fundamentally different map-matching algorithms and our results showed that our method produces similar results regardless of the map-matching algorithm used.

We can give the following 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 confirmed restrictions back to the OpenStreetMap project. This way, the product of our work could be shared by the related mapping community. Additionally, our results could be proven extremely useful to further improving the quality of existing map-matching algorithms. Many of those algorithms use partial shortest-path calculations to align the raw GPS traces to the road network graph. Up until now, those SP computations do not take turning restrictions into account. Since, our approach identifies such restrictions, those newly found constraints could be integrated back in the map-matching algorithms to further improve their results. That way a self-improving, evolutionary map-matching algorithm might be possible after all.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme "SimpleFleet" (http://www.simplefleet.eu, grant agreement No. FP7-ICT-2011-SME-DCL-296423).

The authors would additionally like to thank Kostas Patroumpas for his work on the map-matching algorithms and his useful insight.

6. REFERENCES

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

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, pages 853–864, 2005.

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

[4] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable route planning. In *Proceedings of the 10th international conference on Experimental algorithms*, SEA'11, pages 376–387, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] A. Efentakis, S. Brakatsoulas, N. Grivas, G. Lamprianidis, K. Patroumpas, and 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, pages 79:79–79:84, New York, NY, USA, 2013. ACM.

[6] A. Efentakis, N. Grivas, G. Lamprianidis, G. Magenschab, and D. Pfoser. Isochrones, Traffic and DEMOgraphics. In *Proc. 21st ACM SIGSPATIAL GIS conf.*, 2013.

[7] A. Efentakis and G. Lamprianidis. SimpleFleet Deliverable D6.5. SimpleFleet Online Demo [Online]. http://www.simplefleet.eu/?page_id=84, 2013.

[8] Google. The Directions API [Online]. https://developers.google.com/maps/documentation/directions/.

[9] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[10] R. Herring, P. Abbeel, A. Hofleitner, and 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*, pages 929–936, 2010.

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

[12] 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.

[13] X. Liu, F. Lu, H. Zhang, and 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, pages 14–19, New York, NY, USA, 2012. ACM.

[14] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proc. 17th ACM SIGSPATIAL GIS conf.*, GIS '09, pages 352–361, New York, NY, USA, 2009. ACM.

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

[16] O. A. Nielsen, R. D. Frederiksen, and N. Simonsen. Using expert system rules to establish data for intersections and turns in road networks. *International Transactions in Operational Research*, 5(6):569 – 581, 1998.

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

[18] OpenStreetMap. [Online]. http://www.openstreetmap.org/, 2013.

[19] OpenStreetMap. Relation:restriction [Online]. http://wiki.openstreetmap.org/wiki/Relation:restriction, 2013.

[20] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, SSD '99, pages 111–132, London, UK, UK, 1999.

[21] QGIS. A Free and Open Source Geographic Information System [Online]. http://www.qgis.org/.

[22] SimpleFleet. Democratizing Fleet Management [Online]. http://www.simplefleet.eu, 2013.

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

[24] F. Viti and H. J. van Zuylen. Modeling queues at signalized intersections. *Transportation Research Record: Journal of the Transportation Research Board*, (1883):68–77, 2004.

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

[26] S. Winter. Modeling costs of turns in route planning. *GeoInformatica*, 6(4):345–361, 2002.

[27] H. Zhang, F. Lu, L. Zhou, and 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, pages 45–52, New York, NY, USA, 2011. ACM.

[28] M. Zhao and X. Li. Deriving average delay of traffic flow around intersections from vehicle trajectory data. *Frontiers of Earth Science*, 7(1):28–33, 2013.

[29] Y. Zheng and X. Zhou, editors. *Computing with Spatial Trajectories*. Springer, 2011.