Matching GPS Traces to (Possibly) Incomplete Map Data: Bridging Map Building and Map Matching

Fernando Torre, David Pitchford, Phil Brown, Loren Terveen
GroupLens Research
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, Minnesota, USA
{torre,pitch,brown,terveen}@cs.umn.edu

ABSTRACT

Analysis of geographic data often requires matching GPS traces to road segments. Unfortunately, map data is often incomplete, resulting in failed or incorrect matches. In this paper, we extend an HMM map-matching algorithm to handle missing blocks. We test our algorithm using map data from the Cyclopath geowiki and GPS traces from Cyclopath's mobile app. Even for conservative cutoff distances, our algorithm found a significant amount of missing data per set of GPS traces. We tested the algorithm for accuracy by removing existing blocks from our map dataset. As the cutoff distance was lowered, false negatives were decreased from 34% to 16% as false positives increased from 5% to 10%. Although the algorithm degrades with increasing amounts of missing data, our results show that our extensions have the potential to improve both map matches and map data.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Geometrical problems and computations; H.2.8 [**Database Management**]: Spatial databases and GIS

General Terms

Algorithms

Keywords

map matching, map building, GPS, geowikis

1. INTRODUCTION

As GPS-equipped devices become more common, larger amounts of GPS data become available to geographic applications. An important challenge and opportunity for these applications is to make sense of all of this new geographic information. Integrating GPS data into these applications can enhance understanding not only of the map and its structure, such as road networks, but also of users, such as their behavior and familiarity with different parts of the map.

One useful technique to help make sense of GPS data is map matching. Map matching consists of finding a corresponding block in a road network for a GPS observation. This lets us situate and analyze GPS data in the context of a road network. Researchers have used this technique to help them study characteristics of a map, such as traffic data [9] (how many vehicles travel through a specific block?), and characteristics of the users, such as route choice models [4] (which blocks do users prefer to travel through?).

Unfortunately, maps are often incomplete. Roads get built and removed. Maps designed for one purpose, frequently driving, may not contain information about shortcuts, sidewalks, bike trails, and alleyways that are crucial for other purposes, such as bicyling or walking. And many times, map data has simply not been gathered yet, as is often the case with VGI (Volunteered Geographic Information) in maps such as OpenStreetMap [3]. If we simply try to map every observation to existing blocks, we can end up with many incorrectly matched blocks or with observations that do not correspond to any block. In order to overcome the limits of matching to incomplete data, we need to be able to account for missing blocks.

In this paper, we present extensions to an HHM-based algorithm for matching traces to maps. Our algorithm is able to detect missing blocks and add them as necessary. First, we discuss work related to map matching and map building. Then we explain our map matching algorithm and the extensions for detecting missing blocks. Finally, we discuss the results of testing our algorithm with our GPS and map data, the implications of our findings, and possible avenues for future work.

2. RELATED WORK

There is a great variety of existing map matching algorithms, using geometric, topological, probabilistic, and even more advanced approaches [7]. A recently popular technique for map matching has been using Hidden Markov Model (HMM) algorithms [5]. These algorithms have been shown to be robust even in the absence of high-precision GPS data, which is often the case with handheld devices. This technique treats blocks as states with transition probabilities between each other and maps the sequence of GPS traces to the most probable sequence of HMM states. As described

ACM SIGSPATIAL GIS '12, November 6-9, 2012. Redondo Beach, CA, USA

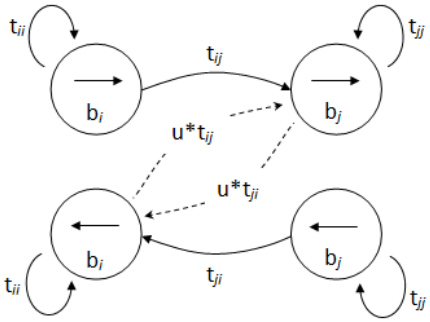


Figure 1: An example of an HMM process. Nodes represent blocks in the road network. Edges represent whether it is possible to travel from the a node's block to another node's block (that is, the two blocks are connected by an intersection). t_{ij} represents the probability of a transition from b_i to b_j, and u represents the probability of backtracking.

in Hummel's work, this approach also inherently provides means for detecting u-turns and erroneous map topology. Although others have expanded on this algorithm [6, 9], these implementations fail to address the problem of detecting missing map data.

There exist many algorithms to build road networks from GPS traces, including [8] and [2]. These algorithms are able to produce precise information (such as lanes and intersections) from high-precision GPS data. Some require the whole set of GPS traces and others are able to build the map incrementally as more GPS data is added [1].

These algorithms have two significant limitations. First, sufficient amounts of high-precision GPS data are not always easy to obtain, such as in our case, where GPS data was obtained from users voluntarily using their mobile devices. Second, these algorithms do not work in the context of road network data. Even the incremental approaches have to build on previously obtained GPS data.

3. MAP-MATCHING USING HMM

Our map-matching algorithm builds on the approach used by Thiagarajan [9], which uses HMM (see Figure 1) to model blocks as hidden states that produce visible observations (GPS traces). In an HMM process, we know: 1) what the resulting observations are, 2) the probability of a state producing a certain observation (emission probability), and 3) the probability of a state transitioning to another state (transition probability). The goal is to find the most likely set of states that could have resulted in a given set of observations.

We model the emission probability for a given block and GPS observation as a normal distribution based on the distance from the observation to the block. In our tests we use a standard deviation of 10m. We set the probability of transitioning to a connected block or staying on the same block to 1/(dmax+1), where dmax is the maximum out-degree of the transportation graph. For a more detailed description of these settings, refer to Thiagarajan's paper.

In addition, we allow U-turns by multiplying the transition probability by a backtracking probability for blocks which we just transitioned from. This is denoted in Figure 1 by u. The bigger the backtracking probability, the higher the possibility that the algorithm will consider that the user backtracked. To avoid accidental matches to incorrect nearby blocks, we keep this probability at an extremely low value (currently 1E-20).

Before running our matching algorithm, we pre-process the GPS tracks to remove outliers (observations that would suggest that the user is traveling unrealistically fast). We also discard tracks with less than 10 observations or that do not cover a long enough distance. Once pre-processing is done, we use the Viterbi decoding algorithm to find the sequence of blocks with the highest probability of producing the given observations [10].

4. FINDING MISSING BLOCKS

Extending the algorithm in the previous section to handle missing blocks means starting with an incomplete HMM and extending it as we go. We need to detect when a new state (block) is required to better fit the given observations. In essence, this is a hybrid between map-maptching and map-building: match when possible, build when needed.

Our approach is straightforward: if there are no blocks that we can transition to, add a new block. There are two cases in which this happens: (1) all blocks have an emission probability of 0 for a given observation (there are no blocks within a cutoff distance d) or (2) all blocks with an emission probability higher than 0 have transition probabilities of 0 (nearby blocks are not connected to the available paths).

The original Viterbi algorithm goes through the sequence of GPS observations and for each step calculates the possible paths that could have created the observations up to that point. With our extensions, at each step, if any of the conditions for creating a new block are met, the following actions also take place:

1. We start with the current GPS observation and move backwards in the sequence of observations until we reach either the first observation in the sequence or the observation that was last seen within s of a block in the matched path, where s is the standard deviation for GPS error.

2. We move forward until we reach an observation that is within s of any block. We now have the start and end observations of the new block to be created.

3. We create a block geometry using the GPS observations selected in the previous steps.

4. We connect the new block to nearby blocks to ensure that transitions will be possible between them.

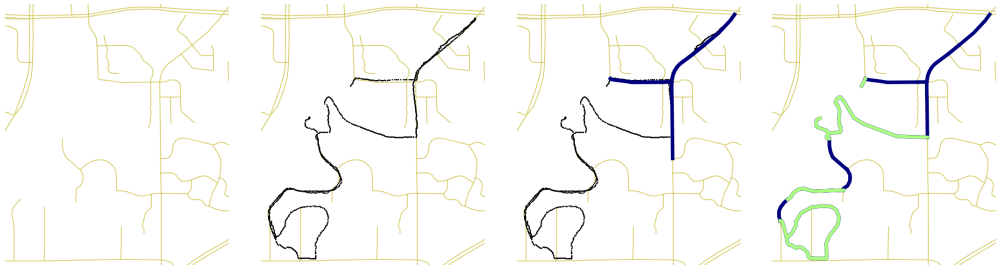
5. We simplify the block geometry to reduce noise.

6. Because the presence of a new block could have affected the emission and transition probabilities of nearby blocks, we rewind the algorithm to the first observation where the new block or any blocks connected to it (which might have been split) had any emission probabilities.

7. We continue the matching algorithm as usual, until the next time we encounter the need for adding a new block.

Figure 2(a) shows an example of an area where a GPS track was recorded. A section of the track is shown in Figure 2(b). This is an example of a case where the user backtracks and also veers off the existing blocks in the dataset.

Without creating new blocks, even if our algorithm can handle u-turns, the map matching result is still not good enough, as shown in Figure 2(c). Once detection of missing blocks is enabled, the algorithm is able to add new blocks and continue as normal until producing the result shown in Figure 2(d). In this example, a total of nine new blocks were added as a result of our algorithm.



(a) Area of map with incomplete data. (b) Section of GPS observations from a track ridden by a user. (c) Map matching with u-turns, but no detection of new blocks. (d) Final match. New blocks are shown in light green.

Figure 2: Map matching and building

d	meters added	blocks added
50	117.1	0.5
40	146.5	0.8
30	207.4	1.3
20	414.7	3.0
10	842.7	10.8

Table 1: Average number of meters and blocks added for each track for each cutoff distance.

	1 block removed		2 blocks removed	
d	false negatives	false positives	false negatives	false positives
50	33.5%	5.4%	51.5%	6.0%
40	30.7%	6.3%	48.7%	7.2%
30	25.7%	7.7%	45.1%	9.6%
20	16.1%	9.7%	36.0%	13.8%

Table 2: Percentage of test runs with false positives and false negatives.

5. EXPERIMENTS

5.1 Dataset

We tested our algorithm using the Cyclopath map of the Twin Cities, which has more than 155,000 blocks covering more than 20,000 miles. In addition to being a collaboratively edited geographic system, Cyclopath is also a computational system that uses VGI to compute bike-friendly routes. We used GPS data collected by a mobile Cyclopath Android app. After pre-processing GPS trace information and removing tracks that were too short, we had 128 GPS tracks. Each track had an average of 1450 GPS observations. The average length was 1.75 mi and the average duration was 8.5 min, with the longest ride taking 50 min.

5.2 New blocks

We first tested the algorithm by running it on the map dataset to see: (1) the amount of new blocks it could find and (2) the length of those new blocks. This gives us an idea of how much benefit we could expect from this algorithm in terms of new block information. Since the value of the cutoff distance d can significantly affect the results of the algorithm, we ran the algorithm with several different values.

Table 1 shows the results for cutoff values between 10 and 50 meters. As expected, a smaller d means the number of false positives (blocks that *should not* have been created) will increase, but the number of false negatives (blocks that *should* have been created but were not) will decrease.

In order to correctly verify which newly created blocks are false positives and which ones are true positives a gold standard is required. In principle, we could evaluate the results by visualizing each new block with aerial photos; however, this approach does not scale. In order to better evaluate our algorithm, we decided to try the following idea: remove existing blocks from our map dataset and see if the algorithm can find them with the given GPS tracks.

5.3 Tests removing existing blocks

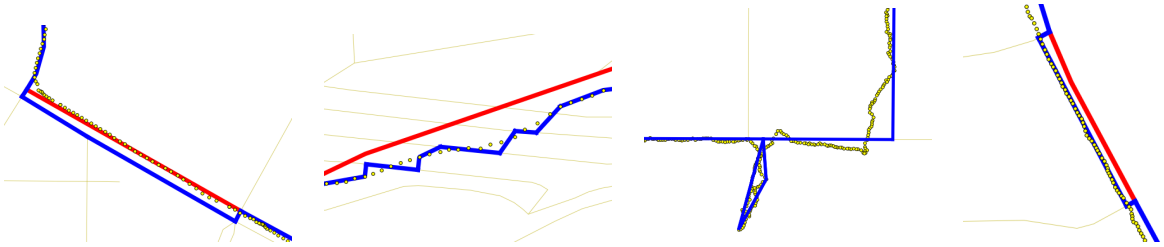
Existing blocks in our system can serve as a type of truth to aid in testing our algorithm. The idea is that if the matching algorithm matches to a certain block when run normally, if we remove that block, the algorithm should be able to recreate the block from the GPS observations. The accuracy of the algorithm for blocks in our system can be a predictor for accuracy for new, unknown bocks.

To test this, we first took every matched ride and removed the first and last blocks, as these blocks are often matched on the basis of very few observations and are therefore matches of low confidence. We then removed from each sequence those blocks that were created by our algorithm when matching, since these blocks were created from the GPS tracks we are trying to test. Finally, we ran the algorithm for each ride, once for each block in the sequence of blocks to be removed for that ride.

The results are shown in Table 2. We define false positives as test runs where more than one block was created and false negatives as test runs where no blocks were created. As expected, as the cutoff distance is decreased, the amount of false negatives also decreases as we are able to find new blocks more accurately, but the amount of false positives also increases, as new blocks get added when they shouldn't.

False negatives occur when the algorithm is able to find a different block to match to, such as in Figure 3(a). The bigger the cutoff distance, the higher the probability that another nearby block can replace the removed block in the sequence of matches. One common cause is parallel blocks, such as bike paths, which often follow along main roads. Thus, a motivation for reducing false negatives is to able to find bike-related facilities such as these.

False positives occur when the algorithm created more blocks than it should have. This is the case for overpasses, such as in Figure 3(b). It is difficult to know for certain if



(a) An example of false negatives, where observations match to nearby blocks and the true block is not created.

(b) An example of false positives, where more blocks are created than actually exist.

(c) An example of incorrect block geometry resulting from a user making a u-turn on a new block.

(d) An example of a case where the intersection is offset from the GPS observations.

Figure 3: Map building issues

an intersection exists at the point where two blocks intersect unless the user actually turns at the intersection.

In this project we did not focus on creating high-quality blocks, so that left us with some space for improvement:

Noise. One case where our current algorithm could do a better job is in simplifying geometries of new blocks, especially when a lot of noise is involved. This is often an issue when the users wanders around the end of his ride or when the users u-turns on a new block, such as in Figure 3(c).

Intersections. As discussed earlier, correctly creating blocks near intersections is also an issue. This is the case not only for overpasses, but also when the intersection is slightly offset from the observations, such as in Figure 3(d).

Bias. New geometries are biased towards the first observations used. In most map building algorithms, new observations are integrated with older observations to create the final blocks. But in our case, we create new blocks using only information from the current track.

The advantage of a geowiki is the fact that we can leverage user input in order to make up for many of these deficiencies. We could ask users to fix geometries created with noisy GPS data, decide whether an a new block intersects or not with older blocks, and update old geometries based on new GPS data. Users can also help detect false positives and false negatives. We could potentially allow users to interactively adjust the cutoff distance in order to change the amount of new blocks created by the algorithm.

5.4 Removing more than one block

In order to get an idea of how the algorithm might perform in situations with less data available, we tried the same technique as in the previous section, but removing two blocks in a row instead of just one. The results for number of tests with false negatives or false positives are shown in Table 2.

Both false positives and false negatives increased significantly when we increased the amount of missing blocks by just one. This is an indicator that our algorithm does not necessarily scale well for map data sets with too much missing data. This results from using a single set of GPS traces to create a representation of the road network.

6. CONCLUSION

With the rising popularity of systems that depend on Volunteered Geographic Information, it is important to develop algorithms that can handle missing map data. This paper provides one such bridge between map matching and map building algorithms. It is ideal for geographic applications that are in constant evolution, such as geowikis. The "match when possible, build when needed" approach is a great fit for applications with (possibly) incomplete map data.

7. ACKNOWLEDGEMENTS

We thank the members of GroupLens Research, the Cyclopath team, and the Cyclopath user community. This work was supported in part by the NSF grant IIS 08-08692 and by a GAANN fellowship.

8. REFERENCES

[1] R. Bruntrup et al. Incremental map generation with gps traces. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 574 – 579, 2005.

[2] L. Cao and J. Krumm. From gps traces to a routable road map. In *Proc. 17th ACM SIGSPATIAL GIS*, GIS '09, pages 3–12, 2009.

[3] M. Haklay. How good is volunteered geographical information? a comparative study of openstreetmap and ordnance survey datasets. *Environment and Planning B: Planning and Design*, 37(4):682–703, July 2010.

[4] J. Hood et al. A gps-based bicycle route choice model for san francisco, california. *Transportation Letters: The International Journal of Transportation Research*, 3(1):63–75, 2011.

[5] B. Hummel. Map matching for vehicle guidance. In *Dynamic and Mobile GIS: Investigating Changes in Space and Time*. CRC Press, 2006.

[6] J. Krumm et al. Map matching with travel time constraints. In *SAE World Congress*, 2007.

[7] M. A. Quddus et al. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312 – 328, 2007.

[8] S. Schroedl et al. Mining gps traces for map refinement. *Data Mining and Knowledge Discovery*, 9:59–87, 2004.

[9] A. Thiagarajan et al. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proc. 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 85–98, 2009.

[10] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260 –269, April 1967.