Real-Time Dense Map Matching with Naive Hidden Markov Models: Delay versus Accuracy

CS 229 Final Project, *Stanford University*
Y. Ahres (yahres@stanford.edu), A. Jambulapati (jmblpati@stanford.edu)
L. Janssen (ljanssen@stanford.edu), J. Kangaspunta (juhana@stanford.edu)

Abstract **- In this paper, an algorithm for map matching is proposed, based on a Viterbi algorithm running over a Hidden Markov Model. The algorithm performs similarly to other existing algorithms for map matching, but runs faster and forgoes the use of memory-intensive shortest path algorithms. The results for the algorithm on data collected around Palo Alto are provided, and various potential improvements are listed at the end.**

1 Introduction

For smartphone users it has become a quotidian experience to find out where you're vehicle is located at a certain point in time when driving your vehicle. However, matching noisy mobile GPS signals onto the correct road in a real time fashion is still research of interest. On a commercial level these problems arise in privately-owned taxi services, where the driver's phone acts as an GPS device. Map matching algorithms must be fast and accurate such that drivers can calculate the fare of the service and can display the position of the driver to the customer. Accuracy versus output delay is most of the time a trade-off, and this work focuses on methods from cutting edge research in map matching accuracy to increase the trade-off ratio for real world implementations.

Matching GPS signals to a graph-based street can be done in multiple ways. First of all, the map matching can proceed in a real time fashion or on batched data basis. In the literature both can be devided in four main groups: geometric [Bernstein96], topological [White00], probabilistic [Ochieng04] and advanced methods not falling into any these categories. The geometric approach only concerns about the shape of the link between nodes, hence discarding information about how the segments are connected. A straightforward search to the closest node is an example of matching the observation to the map. Topological map matching extends that by taking the relationship between entities in the vector-based street map into account. The class of probabilistic map matching involves a confidence region around an observation. The matching then is based on probabilistic features in this region. The Hidden Markov Model (HMM) developed by [Baum66] belongs to this class. Superb recent work in HMM for real time map matching has been in the area of variable sliding windows with sparse data and shortest path graph search [1], pre-smoothing raw data and shortest path graph search [Szwed14]. Regarding batch data HMM, [Newson09] found an almost flawless model arguing that in batch mode the problem can be regarded as solved.

Our work focuses on the gap in recent work. The aim is to create a real time HMM that replicates a computationally undesired shortest path search algorithm without the explicit knowledge whether a path exist to that road segment given the frequency of the observations. Also, the focus is on dense 1Hz mobile GPS data which is applicable to many applications. Modelling the complexities of the problem and finding good features is key. By fusing and extending novel approaches we focus on the delay versus accuracy metric to evaluate the model.

The article proceeds as follows: Somel definitions are explained first in section 2. After that our methods for obtaining the data, the algorithm, and the parameter learning is elaborated upon in consecutive sections. We provide results and conclude with some notes on work in progress.

2 Definitions

Def 1: The graph. Let the directed graph $\mathcal{G}$ be a localized digital road map with nodes $V \in \mathcal{G}$ and arcs $E_{ij} = V_i \to V_j$. Then each curved street in real life is approximated by straight arcs with start and end node $A = V \times V$ such that $A \subseteq V$.
Def 2: Sliding window. The sliding window is defined to have fixed size T. A buffer of size $0 \leq N < T$ imposes an output delay $\propto N$. Due to this buffer the Viterbi algorithm can use 'future' observations to enhance accuracy.
Def 3: State space. We define an observation as a package of data (including GPS data) that the device has provided to the algorithm. For each observation o_t we define the arcs A within great-circle distance ρ of o_t as the state space $\mathcal{S}_t$: $\{\mathcal{S}_t : A \in \{\| o_t - V \|_g \leq \rho\}\}$. For $t > T - N$ we define the optimal path as $A^* = \{A_1^*, A_2^*, \ldots, A_{t-T+N}^*\}$. The optimal path of the algorithm we denote as $\tilde{A}^*$.
Def 4: naive HMM. The sequential HMM is defined as the tuple

$$\lambda_t \equiv \bigcup_{i \in [t-T,t]} \left(\mathcal{S}_i, o_i, P_i^{(tr)}, P_i^{(em)}, A^0\right)$$

Where $P_t^{(tr)}$ is the set of transition probabilities $p_t(A_i \to A_j | A_i)$, $\forall A \in \mathcal{S}_{t-1} \cup \mathcal{S}_t$ and $P_t^{(em)}$ is the set of emission probabilities $p_t(o_t | A \in \mathcal{S}_t)$. A^0 is the assumed end arc. Note that this naive HMM representation does not take into account whether $A_i \to A_j$ is a link in $\mathcal{G}$.

[1] For more information see [Hart68] algorithm [Goh12]

3 Data

Data is gathered by driving around in the Palo Alto and Stanford area, using the Sensor Log [Alan14] application on an iPhone5. 39943 GPS observations including difficult sections such as complex junctions, highway branches and fly-overs. This data is obtained from the 4 routes depicted in figure 1. The test data is gathered in the Palo Alto area and has 1000 observations. Our set of map information was taken from OpenStreetMap.org.

The retrieved latitude and longitude GPS data is assisted with WiFi hotspot location and cellular triangularization by the iPhone's hardware. In addition, the raw data provides the implied heading and speed of the vehicle through the iOS location API.

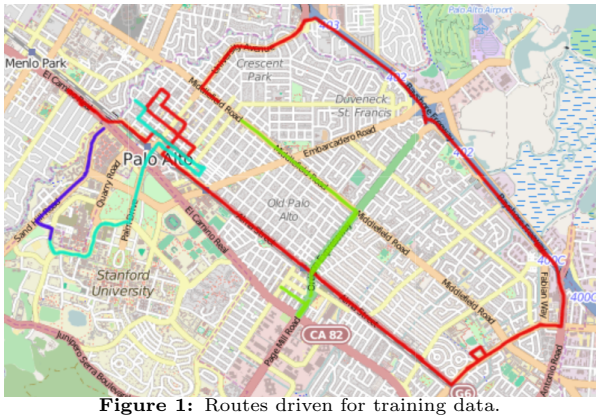


Figure 1: Routes driven for training data.

4 The Algorithm

Using Viterbi algorithm[2] at time t we search through the naive HMM to find the local optimal path $\tilde{A}^*_L = \{\tilde{A}^*_{t-N}, \ldots, \tilde{A}^*_t\}$ conditioned on $\{\tilde{A}^*_{t-T}, \tilde{A}^*_{t-T+1}, \ldots, \tilde{A}^*_{t-N-1}\}$. In short, the Viterbi implemented is a recurrence relation up to $t-N-1$ where the optimal path is defined as follows:

$$R(A|o_t; A^0) = P_t^{(em)} \max_{A \in \mathcal{S}_{t-1}} \left(P_t^{(tr)} R(A|o_{t-1}; A^0) \right)$$

$$\tilde{A}^*_L = \underset{\cup_{i \in [t-N,t]} A \in \mathcal{S}_i}{\operatorname{argmax}} R(A|o_t; A^0)$$

Figure 2 displays how the sliding window with buffer propagates. The red states, or in other words "arcs", are the states conditioned on as the optimal path for $t-T$ up to $t-N-1$.

[2]Comprehensive overview see e.g. [Forney73]

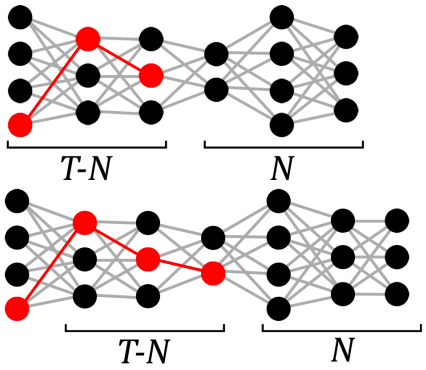


Figure 2: Sliding window with buffer size $N = 3$

4.1 Emission Probabilities

To score our possible states, we implement some emission features:

Emission feature 1: Segment-Location distance. This measure allows our model to determine the distance between our GPS estimated location and a possible segment. We define it as $(\sigma\sqrt{2\pi})^{-1} exp(\frac{-d(o,E)^2}{2\sigma^2}$, where σ is a predefined standard deviation parameter and $d(o, E)$ is the minimum distance between our observed node o and the segment E.

Emission feature 2: Segment-Velocity orientation distance. This measure determines the angle between our vehicle's estimated direction and a given road segment's direction. To compute this, we define a coordinate plane over our map, and convert each road segment into a unit vector $f(E)$ and our observation into a unit vector $f(o)$. Our measure is then $f(o) * f(E)$.

Emission feature 3: Speed Limit constraint. This measure prevents our model from transitioning to a road with a drasticaly different speed limit when our vehicle is traveling at high speeds. This is done to replace the use of shortest path in our algorithm, as most roads that are "nearby" to our current segment share the same characteristics to our current segment. We calculate the measure by setting it to 0 if our current speed is over 40 miles per hour and if our current speed is 20 percent or higher over the segment being tested, and set to 1 otherwise.

The choice of these measures contain some implicit assumptions, however. The Segment-Location distance measure assumes that the observed GPS error is normally distributed. This assumption is a good approximation for the actual truth, but it is not exact. In fact, raw GPS data is Rayleigh distributed, as when projected the error is normal in both latitude and longitude. However, the GPS data collected by most smartphones is augmented with data from wifi beacons and cell towers. These other factors complicate the raw distribution, and convert it to a more normally distributed curve. As seen below, while our raw data is not a perfect fit for a normal distribution, a normality assumption is accurate enough for use in this model.

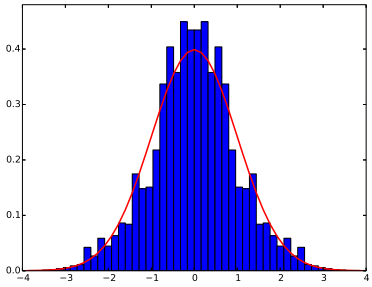


Figure 3: GPS error against Gaussian.

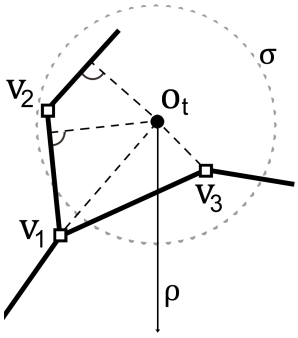
In addition, the second feature assumes that the map projection used is a perfect match for the three-dimensional earth. However, while the Mercator projection used is not a perfect fit for the earth, the error that the projection introduces is so minimal that it does not affect the results significantly.

4.2 State Transitioning

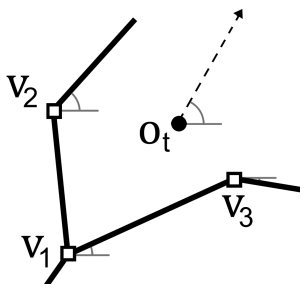
To reveal the hidden states in the HMM we have to model the state transitioning probabilities in the set $P_t^{(tr)}$ for each $A \in \mathcal{S}_t$ as a-priori $\mathcal{S}_t$ is unknown. We define 2 transition features that should replicate the shortest path algorithm well enough to give improbable transitions a low probability. For some i, j let $p_t(A_i \to A_j | A_i) = w_1\omega_1 + w_2\omega_2$ such that $\| w^{(tr)} \|_1 = 1$ where $w^{(tr)} = \text{vec}(w_1, w_2)$ defines the weight vector.

Transition feature 1: Backtrack prevention. To prevent our model from reneging on a previous decision and going back to a previously visited state, we add a weight to our model to improve the model's tolerance of uncertain predictions by underweighting segments in the state space that have been previo usly traversed. The measure is defined as 1 for all unvisited segments in the state space and 0 for all visited nodes minus the current one.

Transition feature 2: Distance discrepancy. We also add a feature to prevent the model's prediction from changing too drastically compared to the distance traveled in real life. To calculate this measure, we first compute the distance between our current observed GPS node and the previously observed GPS node $d(o_t, o_{t-1})$, and then compute the distance between the midpoints of our previous segment and the segment we are calculating the measure for, and call it $d(E_{o_{t-1}}, \hat{E})$. Then, our measure is $(|d(o_t, o_{t-1}) - d(E_{o_{t-1}}, \hat{E})| + 1)^{-1}$.



ϕ_1 metric and ρ.

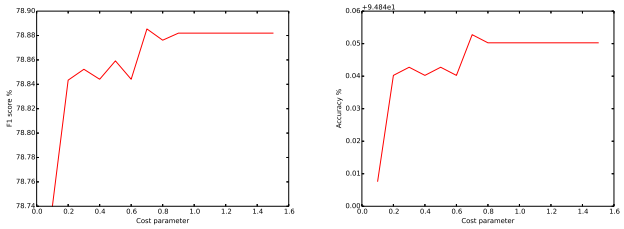


ϕ_2 metric.

4.3 Learning Weights

The weights vectors $w^{(tr)}$ and $w^{(em)}$ need to be obtained to give each feature the correct weight when constituting $p_t(A_i \to A_j | A_i)$ and $p_t(o_t | A_i \in \mathcal{S}_t)$ for some A_i. Guessing the weights first and running the Viterbi algorithm we label $A_i \to A_j$ as 1 and $A_i \nrightarrow A_j$ as -1 for each $A_i \in \mathcal{S}_{t-1}$ and $A_j \in \mathcal{S}_t$. Post-correcting manually for classification errors gives the correct labeling of feasible transitions. Holding the emission weights constant, we run SVM with l_1-regularization on the obtained labels and the feature scores. To do so, we use the `scikit-learn 0.15.1` python package that parses to the `LIBSVM` package to run the SVM algorithm. Since in general $|\mathcal{S}_t| >> 2$ we have unbalanced labeling data as only the transitioned state in the state space is assigned the label 1. As only $\approx 10.5\%$ is labeled 1 we run an SVM model that takes this into account. Figure 6 displays a balanced impression of the training data for obtaining $w^{(em)}$ where we randomly deleted samples labeled -1 just for the sake of data visualization.

Running 10-fold cross validation to tune the cost parameter C we find that $C = 1$ is optimal for both $w^{(tr)}$ and $w^{(em)}$ SVM models. Figures 5a and 5b depict the minute differences in F_1 scores and accuracy percentages for varying $C \geq 0.1$ values.



F_1 score. Accuracy %.

Figure 5: Tuning metrics.

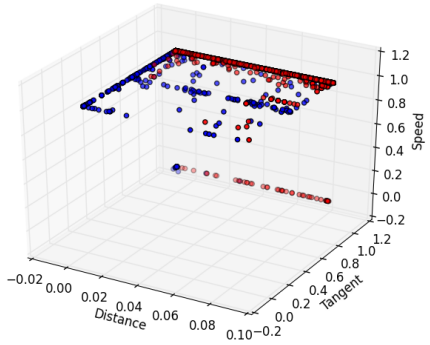


Figure 6: Impression of train data. Red is label 1.

5 Results

Give error analysis in math notation given before:

$$\mathcal{E} = \frac{1}{M} \sum_{i=T-N}^{M} I[\tilde{A}_i^* \neq A_i^*]$$

Where the subscript i denotes the ith segment in the ordered set.

The accuracy of the algorithm was evaluated by using two test sets of raw GPS observations gathered by driving in San Francisco and Palo Alto. The total number of observations in these test sets is 1227 and the environment spans from downtown streets in San Francisco to freeway. The driving routes for collecting the test sets were designed to replicate common driving paths of taxi-service companies and include difficult areas, such as freeway overpasses and areas between high buildings where GPS data is exceptionally noisy. The observations were collected at rate 1Hz.

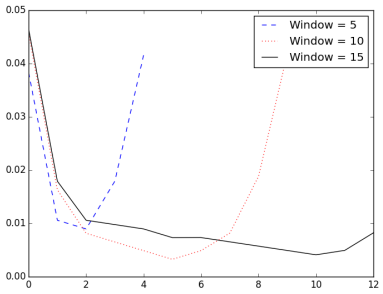


Figure 7: Convexity of classification error for fixed window size and variable buffer. Blue dashed is a sliding window of size 5, red dashed is a window of 10, and black is a window of 15. The y-axis is the classification error percentage, and the x axis is the buffer size.

Figure 7 displays the combined errors for the two test sets given different sliding window and buffer size configurations. As we can see from the figure, the errors are generally very low and the algorithm performs well in the test scenario having 0.046 classification error with the worst configuration and 0.0033 classification error with the best configuration.

As displayed in Figure 7, there is a significant tradeoff between buffer size (delay) and accuracy. Evaluating the empirical test results, we can estimate that error as a function of buffer size is convex, and the minimum error is achieved approximately when buffer size is half of window size. This observation can be explained intuitively, since when buffer size is close to half of the window size, we have an equal amount of information about future observations as we have from past observations. Therefore, the most confident estimations of the states lie in the middle of the Hidden Markov Model.

Because of the tradeoff between delay and accuracy, it can be concluded that no absolute window and buffer parameters exist for this model and it is left for the implementer to decide how much accuracy can be sacrificed for lower values of delay.

6 Conclusions and Future Work

In this paper, we proposed a new approach for real-time map matching and analyzed its performance on ground truth data. This HMM-based approach relied on simple, yet powerful features that allowed us to outperform current map-matching algorithms both in accuracy and delay. Weve also studied the tradeoff within this HMM-model between output delay and performance. This showed us that, by inducing artificial delay, we can buffer the next values and compute the current state with a better confidence. This study aimed to provide a framework for future applications where the minimum delay and accuracy of map-matching is crucial.

In the future, we may improve the algorithm devised here further. A discrete Kalman Filter may be added to process our GPS data and smooth out potential error spikes. In addition, we may also try using different statistical distributions, such as a hyperbolic secant distribution, to fit the GPS error better and improve the accuracy of our distance measure.These are relatively simple tweaks to the model, but given the limited amount of time provided for the project, there was no time remaining to implement them.

In the longer term, we may also add a maneuver detecting feature to detect turn behavior at intersections. This would allow us to further reduce the possibility of backtracks at intersections. However, adding such a feature would be very difficult to implement well, as the wide variety of intersections means that the measure will likely be much more complex than anything explored in this paper.

Finally, we may consider detecting the GPS standard deviation actively and change the model's weights accordingly. However, this is again nontrivial to implement well, as the only way to obtain the standard deviation is to take many GPS samples while the car is stationary. The error obtained in this way is not an entirely reliable measure for the standard deviation, meaning that any potenial gains in accuracy from increased granularity would be at least partially offset by the error in the resulting estimate.

7 Bibliography

[Julier04] Julier, S.J. and Uhlmann, J.K.. Unscented filtering and nonlinear estimation. . p. 401-422 2004

[Goh12] Goh, C.Y. and Dauwels, J. and Mitrovic, N. and Asif, M.T. and Oran, A. and Jaillet, P.. Online map-matching based on Hidden Markov model for real-time traffic sensing applications. . p. 776-781 2012

[Szwed14] Szwed, P. and Pekala, K.. An Incremental Map-Matching Algorithm Based on Hidden Markov Model. . p. 579-590 2014

[Quddus03] Mohammed A. Quddus and Washington Yotto Ochieng and Lin Zhao and Robert B. Noland. A general map matching algorithm for transport telematics applications. . p.

157-167 2003

[Quddus07] Mohammed A. Quddus and Washington Y. Ochieng and Robert B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions . . p. 312-328 2007

[Bernstein96] Bernstein, D and Kornhauser, A. An Introduction to Map Matching for Personal Navigation Assistants. . 1996

[White00] Christopher E. White and David Bernstein and Alain L. Kornhauser. Some map matching algorithms for personal navigation assistants. . p. 91-108 2000

[OCheing04] Ochieng, WY and Quddus, MA and Noland, RB. Map matching in complex urban road networks. . p. 1-18 2004

[Baum66] Baum, Leonard E. and Petrie, Ted. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. The Institute of Mathematical Statistics. p. 1554-1563 1966

[Hart68] Hart, P.E. and Nilsson, N.J. and Raphael, B.. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. . p. 100-107 1968

[Newson09] Newson, Paul and Krumm, John. Hidden Markov Map Matching Through Noise and Sparseness. ACM. p. 336-343 2009

[Krakiwsky88] Krakiwsky, E.J. and Harris, C.B. and Wong, R.V.C.. A Kalman filter for integrating dead reckoning, map matching and GPS positioning. . p. 39-46 1988

[Alan14] Alan, H.F. and Arnrich, B. and Ersoy, C. and Cinaz, B.. Sensor Log: A mobile data collection and annotation application. . p. 1375-1378 2014

[Forney73] Forney, G.D., Jr.. The viterbi algorithm. . p. 268-278 1973