Data-Driven Trajectory Smoothing

Frederic Chazal
INRIA Saclay – Île-de-France
Orsay, France
frederic.chazal@inria.fr

Daniel Chen
Stanford University
Stanford, CA 94305
danielc@cs.stanford.edu

Leonidas Guibas
Stanford University
Stanford, CA 94305
guibas@cs.stanford.edu

Xiaoye Jiang
Stanford University
Stanford, CA 94305
xiaoyej@stanford.edu

Christian Sommer
MIT
Cambridge, MA 02139
csom@mit.edu

ABSTRACT

Motivated by the increasing availability of large collections of noisy GPS traces, we present a new data-driven framework for smoothing trajectory data. The framework, which can be viewed of as a generalization of the classical moving average technique, naturally leads to efficient algorithms for various smoothing objectives. We analyze an algorithm based on this framework and provide connections to previous smoothing techniques. We implement a variation of the algorithm to smooth an entire collection of trajectories and show that it perform well on both synthetic data and massive collections of GPS traces.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling

General Terms

Theory, Smoothing, Trajectories, Data-Driven Techniques

1. INTRODUCTION

Motivation.

Large scale collections of trajectory data, such as GPS traces, are becoming widely available, and many applications [47, 48, 26, 28, 14, 46, 45] that involve understanding and extracting information out of such data are emerging. However, using such data in practice is often challenging as the noise levels often exceed those for which many algorithms are designed. One solution to cope with this problem is a class of tools known as smoothing techniques, which preprocess the data by removing the noise, and thereby revealing the important underlying paths. Indeed, in practice, standard smoothing techniques, such as kernel smoothers [22], smoothing splines [23], generalized additive models [23], and Kalman filters [44] are often used to preprocess GPS data in preparation for road network generation [7] and other model extraction tasks [25].

ACM SIGSPATIAL GIS '11, November 1-4, 2011. Chicago, IL, USA

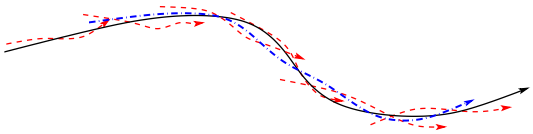

Figure 1: The underlying *path* in solid black, a collection of *traces* in dashed red, and the *query trajectory* in dashed-dotted blue. Note that the noisy traces only partially sample the underlying path and that there may not exist a single trace sampling the same portion of the path as the query trajectory.

However, these methods do not leverage the special structure of trajectory data and the opportunities for statistical estimation given by the large collection of trajectories at the same time. In this paper, we explore a new smoothing framework specifically tailored for trajectory data. Moreover, our approach is *data-driven* rather than *model-driven*, and hence is suitable for the setting where we have a large collection of traces sampling underlying paths which may belong to vastly different classes of curves. In this paper, we formalize the data-driven smoothing paradigm by developing both a model for the input data, and a novel and elegant algorithmic framework that results in practical and efficient algorithms.

Input and Output.

We assume that there exist: (1) An underlying *path* which we cannot observe directly, (2) a database of *traces* partially sampling the path and (3) a *query trajectory* sampling from the path that we seek to smooth using the traces (See Figure 1). The goal is then to use the collection of traces to output a smoothed version of the query trajectory. Note that there may not be a single trace that samples the entire portion of the path corresponding to the query trajectory. In the particular example of GPS data, this model allows the query trajectory to correspond to a route in the road network that we do not have in the database. However, the route can be split into many sub-routes, where each sub-route is covered by a trace in the database. If the particular application requires that the entire database of traces be

smoothed, we can iterate over the traces, and assign each as the query trajectory.

Algorithmic Framework.

Inspired by [42], we propose a simple and elegant method that begins by embedding each point p of the traces and query trajectory into a high dimensional space as $\hat{p}$ using its *delay* coordinates, which are the coordinates of a window of sample points preceding and following p (see Section 3). We call this process *lifting* p, the high dimensional space the *lifted space*, and the embedded image $\hat{p}$ the *lifted image* of p. We then move each point $\hat{p}$ in the lifted image of the query trajectory towards several of its nearest neighbors in the lifted space (Laplacian smoothing). The exact set of nearest neighbors selected for this step depends on the smoothing objective. Finally, we recover the original traces by averaging the relevant coordinates of the lifted images.

Analysis and Experiments.

We prove that, under natural smoothing objectives, the process of using delay coordinates achieves much better results than considering the individual sample points by themselves. We show that under some weak probabilistic assumptions, a variant of the algorithm moves a noisy query trajectory towards the portion of the underlying path that it is sampled from. Under certain assumptions, the technique can be shown to be an unbiased estimator of the underlying path with high probability. Furthermore, we are able to dramatically reduce the variance of the error from that of the original data by smoothing with respect to a large database of traces.

We also show that our method can be viewed of as a generalization of standard moving average techniques for smoothing time series data. When there is no database of traces available to smooth a query trajectory, then our method is equivalent to the moving average technique. The result of such a smoothing technique optimizes an objective function that weighs the distance between the query trajectory and a constant velocity trajectory in a natural manner.

Experimentally, we consider both synthetic noisy data, as well as a collection of real GPS traces from Moscow and a collection of taxi cab traces from an unidentified major city. We adapt our framework for all three datasets and show that they result in trajectories that are significantly smoother and less noisy than the input trajectories.

Related Work.

Our work is most similar to, and motivated by that of Cao and Krumm [8], who also smooth GPS traces before using them to construct a routable road map. Their approach, which is based on simulating forces of physical attraction, is also data-driven rather than model-driven. Our method is arguably simpler and easier to implement, and is equipped with rigorous analysis showing that it is an unbiased estimator of the underlying path. Practically, our method, like that of Cao and Krumm, also allows different directions of travel on a road to be identified, but since we do not use repelling forces, our method works for both left-hand and right-hand drive systems, and does not artificially push lanes of opposite directions away from the centerline.

There have also been many developments in smoothing, noise-removal and outlier detection techniques in the GIS community: Nearest neighbor approaches for outlier detection are well studied [40, 41, 43, 11, 12]. Chen et al. [13] leverage spatio-temporal redundancy to cleanse RFID data. Guilbert and Lin [18] developed a B-spline smoothing method with positional constraints for marine charts, where spatial conflicts induced by normal B-spline smoothing can present navigational safety issues. Liu et al. [31], explore various random walk based approaches for spatial outlier detection. Arge, et. al. [5], presented a method to clean sonar point clouds. They show theoretical analysis that their method is I/O efficient, and hence is practical for massive data sets.

Other related work in the GIS community includes that of Hönle et al. [24], who presented an experimental study on various compression algorithms for trajectories, of Panangadan and Talukder [35], who use a data driven approach for tracking applications, of Johanssan and Jern [29], who integrate statistical filtering techniques into a geographic visualization tool, and of Hall et al. [19], who investigate various heuristics to clean up personal GPS data.

In the broader research community, related work includes principal curves [21], which are smooth curves passing through point clouds. They can be considered to be a nonlinear generalization of principal components. Cheng et al. [15] use a data-driven approach to clean entries in a probabilistic database. Chazal et al. [9] show connections between the distance to k-nearest neighbors and a natural distance function between probability measures. Their results can be used for robust topological and geometric reconstruction from data with heavy, but known, noise distributions. There has also been work on computing mean shapes [38, 30] which study collections of shapes by considering their distributions in a shape-space.

2. PRELIMINARIES

Recall that we have an underlying *path*, a database of *traces* and a *query trajectory*. We model the underlying path as a curve $f : \mathbb{R} \to \mathbb{R}^D$. We cannot observe this underlying path; instead, what we observe is the database of traces, which we model as noisy, discrete and finite vectors $\{f^{(i)}\}_i$ with the following properties:

1. Each vector is of the form $f^{(i)} := (f^{(i)}_{\alpha_i}, f^{(i)}_{\alpha_i+1}, f^{(i)}_{\alpha_i+2}, \ldots, f^{(i)}_{\alpha_i+m})$ where $f^{(i)}_x \in \mathbb{R}^D$ and $f^{(i)}_x = f(x) + \epsilon^{(i)}_x$ where $\epsilon^{(i)}_x$ corresponds to an error term.

2. Each $\epsilon^{(i)}_x$ is an independent and identically distributed error term with mean $\mathbf{0}$ and with $E\|\epsilon^{(i)}_x\|^2 = \sigma^2$. Here, $E\|\epsilon^{(i)}_x\|^2$ is the trace of the covariance matrix of $\epsilon^{(i)}_x$, which characterizes the variances of $\epsilon^{(i)}_x$. In addition, we assume that there exists an upper bound M on the magnitude of the error, e.g., $\|\epsilon^{(i)}_x\| \leq M$.

The query trajectory that the algorithm seeks to smooth is of the form $g := (g_{\alpha_i}, g_{\alpha_i+1}, g_{\alpha_i+2}, \ldots, g_{\alpha_i+l})$ where $g_x \in \mathbb{R}^D$ and $g_x = f(x) + \eta_x$ where $\eta^{(i)}$ corresponds to an error term that has the same distribution $\epsilon^{(i)}_x$. The query trajectory may or may not come from the collection $\{f^{(i)}\}_i$. The smoothing objective is then to reduce the variance of the deviation errors of g with respect to the underlying path f by using the collection $\{f^{(i)}\}_i$, and output the smoothed trajectory as $\breve{g}$.

3. ALGORITHMIC FRAMEWORK

The input to the algorithm consists of the traces $\{f^{(i)}\}_i$ and a query trajectory g sampling an underlying path f. The algorithmic framework consists of three main steps:

1. *Delay Embedding (Lifting):* For the delay embedding, we choose an integer parameter $n \geq 0$ that controls the length of the embedding. In a sense, this parameter controls the importance we place on the sequential information over the positional information. Now, given a vector $h = (h_\alpha, h_{\alpha+1}, \dots h_{\alpha+m})$, we embed each $h_x \in \mathbb{R}^D$ into the *lifted space* $\mathbb{R}^{D(2n+1)}$ by mapping it to its *lifted image* $\hat{h}_x$:

$$\begin{aligned}\hat{h}_x = (&h_{x-n}(1), h_{x-n}(2), \dots, h_{x-n}(D);\\ &h_{x-n+1}(1), h_{x-n+1}(2), \dots, h_{x-n+1}(D);\\ &\dots, h_{x+n}(1), h_{x+n}(2), \dots, h_{x+n}(D))\end{aligned}$$

   where $h_{x-m}(i)$ is defined to be the ith coordinate of $h_{x-m} \in \mathbb{R}^D$. For example, a vector $((1,1),(2,2,),\dots,(k,k))$ would be lifted to $((1,1,2,2,3,3),(2,2,3,3,4,4),\dots,(k-2,k-2,k-1,k-1,k,k))$ for $n=1$. One caveat is the lifted vectors are now shorter, but in practice this is not much of an issue, since n can be set to value that is much smaller than the length of the vectors. We apply this lifting procedure to all points on $f^{(i)}$ and g to obtain $\hat{f}^{(i)}$ and $\hat{g}$.

   The advantages of using this lifting procedure are twofold. First, the sequential information around the point of interest encodes important attributes of the path such as speed and direction. This allows opposite directions of travel to be differentiated, for example. Second, the delay embedding also allows us to take advantage of statistical concentration to reduce noise even further. In Section 4, we analyze this effect in detail.

2. *Moving Towards Nearest Neighbors:* We embed each point of the traces and query trajectory into the lifted space, forming a point cloud $\mathcal{P}$ in $\mathbb{R}^{(2n+1)D}$. Inspired by [9], in which points are moved along the gradient of the so-called distance-to-measure function, the smoothing technique works in the lifted space $\mathbb{R}^{(2n+1)D}$ by moving the point $\hat{g}_x$ to its nearest neighbor or to the barycenter of its k nearest neighbors in the point cloud $\mathcal{P}$, i.e.,

$$\tilde{g}_x = \frac{1}{k} \sum_{\mathtt{x} \in \mathrm{knn}(\hat{g}_x)} \mathtt{x}$$

   where $\mathtt{x}, \hat{g}_x \in \mathbb{R}^{(2n+1)D}$.

3. *Recovering the Trajectory:* With each point $\hat{g}_x$ moved to a new position (denoted by $\tilde{g}_x$) in the last step, we now recover a new trajectory $\breve{g}$ from the points $\hat{g}_x$'s. Let

$$\begin{aligned}\tilde{g}_x = (&\tilde{g}_{x-n}(1), \tilde{g}_{x-n}(2), \dots, \tilde{g}_{x-n}(D);\\ &\tilde{g}_{x-n+1}(1), \tilde{g}_{x-n+1}(2), \dots, \tilde{g}_{x-n+1}(D);\\ &\dots, \tilde{g}_{x+n}(1), \tilde{g}_{x+n}(2), \dots, \tilde{g}_{x+n}(D))\end{aligned}$$

   There are several alternatives for this procedure. One simple strategy is to use the center of $\tilde{g}_x$, which is $(\tilde{g}_x(1), \tilde{g}_x(2), \dots, \tilde{g}_x(D))$ as the coordinates for $\breve{g}_x$. We note that $(\tilde{g}_x(1), \tilde{g}_x(2), \dots, \tilde{g}_x(D))$ may appear in different $\tilde{g}_x$'s, which are not necessarily the same. Thus another strategy is that we can take the average of all $(\tilde{g}_x(1), \tilde{g}_x(2), \dots, \tilde{g}_x(D))$ that appears in different $\tilde{g}_x$'s as the coordinates for $\breve{g}_x$.

Our method is easy to summarize: we simply move the points of the lifted trajectory $\hat{g}_x$ to the barycenter of several of its nearest neighbors in the lifted points of $\{f^{(i)}\}_i$. There are several variations of this framework: the particular one we will analyze is where we only look for one nearest neighbor from each trace $f^{(i)}$. We reproject each lifted point to a point in the original space by averaging over all values of the smoothed coordinates of its image in the original embedding. For ease of reference, we outline the algorithmic framework in Algorithm 1.

Algorithm 1 Smoothing Framework

Input: Collection of traces $\{f^{(i)}\}_i$ and a query trajectory g sampling an underlying path f.
Output: Smoothed trajectory $\breve{g}$.
1: Embed $\{f^{(i)}\}_i$ and g in a high dimensional space as in Section 3 to obtain $\{\hat{f}^{(i)}\}_i$ and $\hat{g}$.
2: Move each point in $\hat{g}$ towards several of its nearest neighbors in the points of $\{f^{(i)}\}_i$.
3: Recover trajectory $\breve{g}$ in original dimension by averaging relevant coordinates in $\hat{g}$.

4. ANALYSIS

4.1 Smoothing Curves Using Delay Coordinates

In the discrete setting of this problem, we have information about each trace in $\{f^{(i)}\}_i$ at discrete time steps. In practice, as in the example of a collection of GPS traces, the time stamps for different traces will not necessarily be synchronized, i.e. $\alpha_i - \alpha_j$ is an integer for all i, j. For ease of analysis, however, we first consider a simple case where each $f^{(i)}$ is synchronized with the query trajectory g. The analysis for the general asynchronized case will be provided in Theorem 4.5.

In the synchronized case, we can dramatically smooth the query trajectory (see Lemma 4.2, Theorem 4.3 and Proposition 4.4). This result is derived using the Hoeffding inequality as described in Theorem 4.1, which was proved first by Chernoff [16] and Okamoto [33] for the special case of binomial random variables:

THEOREM 4.1 (HOEFFDING INEQUALITY). *Let $X_1, \cdots, X_n$ be independent bounded random variables such that X_i falls in the interval $[a_i, b_i]$ with probability one, and $S_n = X_1 + \cdots + X_n$. Then for any $t > 0$ we have*

$$\mathbb{P}\Big\{S_n - \mathbb{E}S_n \geq t\Big\} \leq \exp\Big(-2t^2 / \sum_{i=1}^{n}(b_i - a_i)^2\Big)$$

When we embed a specific point g_0 on g, the lifted image $\hat{g}_0$ is $(g_{-n}, \cdots, g_n)$. We show that conditioned on $f^{(i)}$ being defined on indices $-n$ to n, with high probability, the lifted image on $\hat{f}^{(i)}$ whose delay coordinates have the same indices is the closest one. We have the following lemma:

LEMMA 4.2. *Assume that there exists a constant* $S \geq 64\sigma^2$ *such that the underlying path* f_k *satisfies*

$$\sum_{k=-n}^{n} \|f(k) - f(k+d)\|^2 \geq Sd^2n$$

for every $d \geq 1$. *Let* $f^{(i)} = f + \epsilon^{(i)}$ *satisfy* $\|\epsilon^{(i)}\| \leq M$ *and* $\mathbb{E}\|\epsilon^{(i)}\|^2 = \sigma^2$. *Then, if* $n \geq M^4/\sigma^4$, *with probability at least* $1 - 8\exp(-n\sigma^4/M^4)$, *the nearest lifted image on* $\hat{f}^{(i)}$ *to* $(g_n, g_{-n+1}, \cdots, g_n)$ *is* $(f^{(i)}_{-n}, f^{(i)}_{-n+1}, \cdots, f^{(i)}_n)$.

PROOF. For a piece of trace $f^{(i)}$ whose defining interval includes $[-n, n]$, we consider the probability that $(g_{-n}, \cdots, g_n)$ is closest to $(f^{(i)}_{-n}, f^{(i)}_{-n+1}, \cdots, f^{(i)}_n)$. For fixed d, we have

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{k=-n}^{n} \|g_k - f^{(i)}_k\|^2 \geq \sum_{k=-n}^{n} \|g_k - f^{(i)}_{k+d}\|^2\Big) \\
&= \mathbb{P}\Big(\sum_{k=-n}^{n} \|\eta_k - \epsilon^{(i)}_k\|^2 \geq \sum_{k=-n}^{n} \|f_k + \eta_k - f_{k+d} - \epsilon^{(i)}_{k+d}\|^2\Big) \\
&= \mathbb{P}\Big(\sum_{k=-n}^{n} 2(\eta_k - \epsilon^{(i)}_{k+d})^T(f_{k+d} - f_k) \\
&\qquad + \sum_{k=-n}^{n} (\epsilon^{(i)}_{k+d} - \epsilon^{(i)}_k)^T(2\eta_k - \epsilon^{(i)}_k - \epsilon^{(i)}_{k+d}) \\
&\qquad \geq \sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big) \\
&\leq \mathbb{P}\Big(\sum_{k=-n}^{n} (\eta_k - \epsilon^{(i)}_{k+d})^T(f_{k+d} - f_k) \\
&\qquad \geq \frac{1}{4}\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big) \\
&\quad + \mathbb{P}\Big(\sum_{k=-n}^{n} (\epsilon^{(i)}_{k+d} - \epsilon^{(i)}_k)^T(2\eta_k - \epsilon^{(i)}_k - \epsilon^{(i)}_{k+d}) \\
&\qquad \geq \frac{1}{2}\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big)
\end{aligned}$$

Since $(\eta_k - \epsilon^{(i)}_{k+d})^T(f_{k+d} - f_k)$ are independent random variables that are bounded by $2M\|f_{k+d} - f_k\|$, by concentration inequality, we know that

$$\begin{aligned}
\mathbb{P}\Big(\sum_{k=-n}^{n} 2(\eta_k - \epsilon^{(i)}_{k+d})^T(f_{k+d} - f_k) \geq \frac{1}{4}\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big) \\
\leq \exp\left(-\frac{(\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2)^2}{\sum_{k=-n}^{n} 64M^2\|f_k - f_{k+d}\|^2}\right)
\end{aligned}$$

where the right hand side can be simplified as

$$\exp\Big(-\frac{1}{64M^2}\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big)$$

Note that

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{k=-n}^{n} (\epsilon^{(i)}_{k+d} - \epsilon^{(i)}_k)^T(2\eta_k - \epsilon^{(i)}_k - \epsilon^{(i)}_{k+d}) \\
&\qquad \geq \frac{1}{2}\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big) \\
&\leq \mathbb{P}\Big(\sum_{k=-n}^{n} 2\|\eta_k\|^2 + 2\|\epsilon_k\|^2 \geq \frac{1}{2}\sum_{k=-n}^{n} (f_k - f_{k+d})^2\Big)
\end{aligned}$$

Since $E\eta_k^2 = \sigma^2$, then by concentration inequality again, we know the right hand side can be bounded by

$$\exp\Big(-\frac{(\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2 - 8(2n+1)\sigma^2)^2}{16(2n+1)M^4}\Big)$$

Under the assumption that there exist a constant S such that $\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2 \geq Sd^2n$ where $S \geq 36\sigma^2$, we have

$$\exp\Big(-\frac{1}{64M^2}\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2\Big) \leq \exp\Big(-\frac{Sn}{64M^2}d^2\Big)$$

and by using the fact that $S \geq 64\sigma^2$ we have

$$\begin{aligned}
&\exp\Big(-\frac{(\sum_{k=-n}^{n} \|f_k - f_{k+d}\|^2 - 8(2n+1)\sigma^2)^2}{16(2n+1)M^4}\Big) \\
&\qquad \leq \exp\Big(-\frac{4(2n+1)\sigma^4}{M^4}(2d^2-1)^2\Big)
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{k=-n}^{n} (g_k - f^{(i)}_k)^2 \geq \sum_{k=-n}^{n} (g_k - f^{(i)}_{k+d})^2\Big) \\
&\leq \exp\Big(-\frac{Sn}{64M^2}d^2\Big) + \exp\Big(-\frac{4(2n+1)\sigma^4}{M^4}(2d^2-1)^2\Big) \\
&\leq \exp\Big(-\frac{n\sigma^2}{M^2}d\Big) + \exp\Big(-\frac{n\sigma^4}{M^4}d\Big)
\end{aligned}$$

By summing up with respect to all d's, we know that the probability that $(f^{(i)}_{-n}, f^{(i)}_{-n+1}, \cdots, f^{(i)}_n)$ is closest to $(g_{-n}, \cdots, g_n)$ would be at least

$$1 - \sum_{d\neq 0}\exp\Big(-\frac{n\sigma^2}{M^2}d\Big) - \sum_{d\neq 0}\exp\Big(-\frac{n\sigma^4}{M^4}d\Big)$$

Provided that $n \geq M^4/\sigma^4$, the above quantity would be at least $1 - 8\exp(-n\sigma^4/M^4)$. Thus we conclude that the failure probability that the nearest lifted image on $\hat{f}^{(i)}$ is not $(f^{(i)}_{-n}, f^{(i)}_{-n+1}, \cdots, f^{(i)}_n)$ decays exponentially fast with the number of delayed coordinates we use. □

To interpret the conditions of the lemma, we note that $\sum_{k=-n}^{n} \|f(k) - f(k+d)\|^2 \geq Sd^2n$ requires that the path is always moving in some general direction. Note that if each component of f is strictly monotonic in some direction $\mathbf{r}$ with $f' \cdot \mathbf{r}$ always greater than a constant, then it is clear that we have $\sum_{k=-n}^{n} \|f(k) - f(k+d)\|^2 \geq Sd^2n$ for all $d \geq 1$ and $n \geq 1$. However, this is a much stronger condition than necessary, as f satisfying the condition can also have a kink or a zigzag pattern, provided that the size of the kink is not too large. Moreover, we require that $S \geq 64\sigma^2$, which means that the moving speed should not be too slow when compared with the level of noise.

Lemma 4.2 bounds the probability that the delay coordinates of the nearest lifted image $\hat{f}^{(i)}$ to $(g_{-n}, g_{-n+1}, \cdots, g_n)$ are not centered around index 0 for any i. Thus, if a collection of N traces are given, we can use the union bound to bound the probability of identifying a wrong lifted image whose delay coordinates are not centered around 0, which leads to the following theorem:

THEOREM 4.3. *Let $\{f^{(i)}\}_i$ be a database of N noisy traces $\{f^{(i)}\}_i$, each following the assumptions of Lemma 4.2. Then with probability at least $1 - 8N\exp(-n\sigma^4/M^4)$, the nearest lifted image on each $\hat{f}^{(i)}$ to $(g_n, g_{-n+1}, \cdots, g_n)$ is $(f^{(i)}_{-n}, f^{(i)}_{-n+1}, \cdots, f^{(i)}_n)$, $\forall i$.*

With the above theorem, it is easy to see that with high probability, we would use $\frac{1}{N}\sum_i f_0^{(i)}$ as the smoothed coordinate for g_0. We now analyze the use of $\frac{1}{N}\sum_i f_0^{(i)}$ as an estimator for f_0. Recall that if $\hat{\theta}$ is an estimator of θ, then the bias and variance of $\hat{\theta}$ is defined to be

$$\text{Bias}(\hat{\theta}) = \mathbb{E}\hat{\theta} - \theta, \qquad \text{Var}(\hat{\theta}) = \mathbb{E}(\hat{\theta} - \mathbb{E}\hat{\theta})^2.$$

Conditioned on the event A that the nearest lifted image on $\hat{f}^{(i)}$ is identified for each i, which holds with probability at least $1 - 8N\exp(-n\sigma^4/M^4)$, the conditional bias of the estimator $\frac{1}{N}\sum_i f_0^{(i)}$ is zero and the conditional variance of this estimator is σ^2/N because

$$\mathbb{E}(\frac{1}{N}\sum_i f_0^{(i)}|A) = f_0, \qquad \text{Var}(\frac{1}{N}\sum_i f_0^{(i)}|A) = \sigma^2/N.$$

In summary, the algorithm has the following property in the synchronized case.

PROPOSITION 4.4. *With high probability, the smoothed coordinate of g_0 is an estimator for f_0 with zero bias. Furthermore, the variance of such an estimator can be reduced by a factor of N if we smooth the query trajectory g with respect to N relevant traces.*

In the above theoretic analysis of the algorithm, we assume that each trace $f^{(i)}$ is synchronized with the query trajectory g. However, in real applications, this is rarely true. By carrying out a similar analysis as in Lemma 4.2, we can show the following:

THEOREM 4.5. *Assume that there exist constants S_1, S_2 and B such that $S_1 \geq B\sigma^2$ for which the path f_k satisfies*

$$S_1 d^2 n \leq \sum_{k=-n}^{n} \|f(k) - f(k+d)\|^2 \leq S_2 d^2 n$$

for every $d \geq 1$, where S_1 and S_2 further satisfies $2S_1 \geq S_2$. If $n \geq M^4/\sigma^4$, then there exists a constant C which depends only on B, such that with probability at least $1 - 8\exp(-Cn\sigma^4/M^4)$, the lifted image on $\hat{f}^{(i)}$ centered at the index k closest to 0 is one of the two nearest neighbors of $(g_{-n}, \cdots, g_n)$.

PROOF. See Appendix A. □

As we mentioned before, $\sum_{k=-n}^{n} \|f(k) - f(k+d)\|^2$ is a stable measure of the moving speed of the path. In the case where S_1 and S_2 are far apart, one can still show that the lifted image on $\hat{f}^{(i)}$, whose delay coordinates are centered at the index k closest to 0, is one of the $l-$nearest neighbors of $g_{-n}, \cdots, g_n$, where l depends on the difference between S_1 and S_2. In the asynchronized case, our estimator is not unbiased, but still succeeds in reducing the variance.

4.2 Connection with Moving Average

Our approach can be viewed of as a generalization of standard moving average techniques for smoothing trajectory data. Consider the case where a database of traces does not exist, and we move each point in $\hat{g}$ towards several of its nearest neighbors as described in the second step of the smoothing framework. It is easy to see that we are essentially performing a moving average operation where we average the nearby coordinates of g to obtain a smoothed trajectory. More generally, we can assign weights on different dimensions in the lifted space, i.e., assigning weights $w(r)$ on the rth dimension, we smooth g_t as $\sum_{r=-n}^{n} w(r) g_{t+r}$.

We can further interpret the result of the moving average when a carefully choosen weighted kernel w is used. For example, if we choose an exponentially weighted kernel $w(r) = \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r|}$, then it is possible to interpret the moving average result as an optimal solution which balances the geometry of the query trajectory g and the velocity constant moving trend in a continuous sense. More precisely, the moving average answers the following question:

QUESTION 4.6. *Given a noisy trajectory $g : \mathbb{R} \to \mathbb{R}$. Can we generate a new trajectory $y : \mathbb{R} \to \mathbb{R}$ such that at each time $t \in \mathbb{R}$, $y(t)$ is close to $g(t)$, while the speed $y'(t)$ is also close to a constant S?*

Here, we focus on the one dimensional case, but it is straightforward to generalize the analysis to higher dimensions because different dimensions are smoothed independently in the moving average smoothing technique. We formulate this question as an optimization problem on a continuous finite interval $[0, 1]$:

$$\min_y \int_0^1 (y'(t) - S)^2 dt + \lambda \int_0^1 (y(t) - g(t))^2 dt \qquad (1)$$

If boundary conditions on $y(0)$ and $y(1)$ are given and we restrict the searching range of the trajectory y to the class of C^2 continuous functions on $[0, 1]$, then using a calculus on variations approach we have the following lemma.

LEMMA 4.7. *The optimal C^2 continuous trajectory y for the optimization problem (1) is characterized by the differential equation*

$$y''(t) - \lambda y(t) + \lambda g(t) = 0 \qquad (2)$$

with given boundary conditions on $y(0)$ and $y(1)$.

PROOF. See Appendix B. □

We can solve the differential equation (2) exactly using Laplace transforms:

LEMMA 4.8. *All the solutions to the differential equation (2) have the form*

$$y(t) = c_1 e^{-\sqrt{\lambda}t} + c_2 e^{\sqrt{\lambda}t} + \int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} g(r) dr \qquad (3)$$

In particular,

$$y(t) = \int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} g(r) dr \qquad (4)$$

is the unique solution to (2) with the particular boundary values

$$y(0) = \int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r|} g(r)dr$$
$$y(1) = \int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-1|} g(r)dr.$$

PROOF. See Appendix C. □

When g is a noisy sample of the underlying path f, i.e., $g(t) = f(t) + \epsilon(t)$, the moving average technique generally yields a biased estimator [36] except for the case where the underlying trajectory f is linear. Specifically, when f is linear, and ϵ is a mean zero continuous process, e.g., the generalized derivative of a Wiener process, then using $y(t) = \int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} g(r)dr$ as an estimator for $f(t)$ has zero bias because

$$\mathbb{E}\Big(\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} g(r)dr\Big)$$
$$= \mathbb{E}\Big(\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} f(r)dr\Big) + \mathbb{E}\Big(\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} \epsilon(r)dr\Big)$$
$$= f(t)$$

In the above equation, $\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} f(r)dr = f(t)$ is only true for the case where f is linear. Moreover, the variance for such an estimator is

$$\mathrm{Var}\Big(\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} g(r)dr\Big)$$
$$= \mathbb{E}\Big(\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} g(r)dr - f(t)\Big)^2$$
$$= \mathbb{E}\Big(\int_{\mathbb{R}} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|} w(r)dr\Big)^2 = \int_{\mathbb{R}} \Big(\frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|r-t|}\Big)^2 dr$$
$$= \frac{\sqrt{\lambda}}{4}$$

where the second to last equation follows from the properties of a Wiener process [34]. Thus, when λ is large, we are essentially only using a few nearby coordinates around to smooth g; such a smoothing technique still has large variance which comes from the noises within g itself. When λ is small, however, such a smoothing technique works quite well.

As a final remark, we note that the moving average can also be thought of as smoothing the query trajectory with respect to a collection of traces that consists of one single linear trace. To see this, consider the continuous formulation of the Algorithm 1 in the one dimensional case: for each point $t \in \mathbb{R}$, we consider a small piece of g around t, e.g., the piece of g on $[t - T/2, t + T/2]$. The algorithm can then be interpreted as finding the piece of the standard trajectory f closest to this piece of g in the L^2 sense. Therefore we formulate a continuous version of the algorithm as the following:

$$\min_s \int_{-T/2}^{T/2} (g(t+r) - f(s+r))^2 dr \tag{5}$$

or, more generally, we can consider a weighted L^2 measure:

$$\min_s \int_{-T/2}^{T/2} w(r)(g(t+r) - f(s+r))^2 dr \tag{6}$$

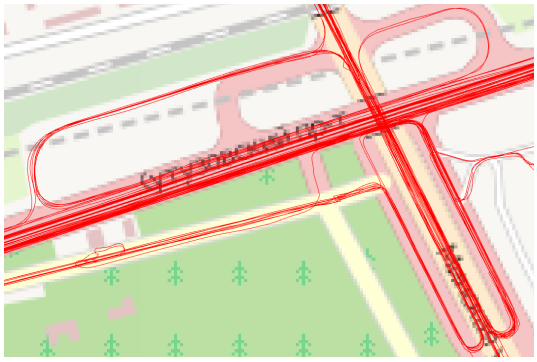

Figure 2: Noisy Traces.

If the standard trajectory f is linear, then without loss of generality we can assume $f(s) = s$. The result of the above optimization problem is exactly the moving average because the optimal solution s for the problem (6) satisfies

$$\frac{d}{ds} \int_{-T/2}^{T/2} w(r)(g(t+r) - (s+r))^2 dr = 0,$$

which implies that the output of the smoothed $g(t)$ is

$$\frac{\int_{-T/2}^{T/2} w(r) g(t+r) dr}{\int_{-T/2}^{T/2} w(r) dr} \tag{7}$$

which is a moving average operation on g, as long as the weights $w(r)$ are choosen to be a symmetric kernel, i.e., $\int_{-T/2}^{T/2} w(r) r dr = 0$.

5. VARIATIONS AND HEURISTICS

Our smoothing framework naturally leads to algorithms for various smoothing objectives. One particular application we are interested in is to smooth an entire collection of GPS traces within a certain area as a noise-removal preprocessing step for algorithms that extract further information from the traces. To do this, we run our algorithm once for each trace in the collection, each time picking that trace as the query trajectory, and hence smooth all of traces in the collection. We use the small example of noisy traces shown Figure 2 to illustrate the effect of this application. Note that in this particular example, we are able to detect inaccuracies in the map, such as the shape of the upper-left cloverleaf from the collection of GPS traces.

5.1 Adapting the Smoothing Framework for Many Underlying Paths

Unlike in the analysis and in the synthetic example, the GPS traces are usually noisy samples, not of a single underlying path, but of many different underlying paths derived by the road map and local traffic laws. If we blindly apply the original algorithm, the result would be curves that appear smooth, but because traces far away from the query trajectory also affect it, the shape of the curves are sometimes distorted to a degree where the curve no longer follows the road network, see Figure 3. Note that in particular, near the bottom of the figure, one of the traces is moved to a region that is roughly in the middle of two roads. This is because the nearest neighbors chosen by this particular trace include several from different roads.

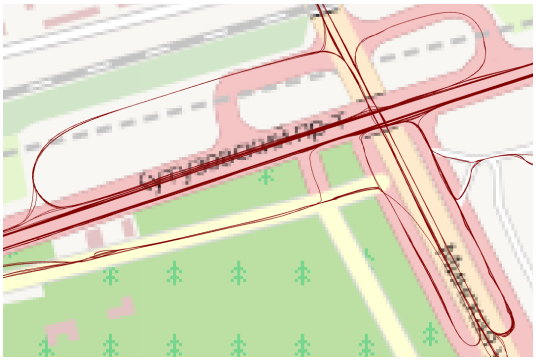

Figure 3: Effect of unmodified algorithm.

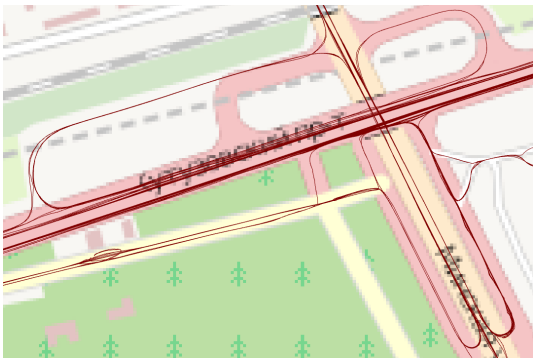

Figure 4: Effect of thresholded algorithm.

Nevertheless, we are able to adapt the algorithm by only selecting nearest neighbors within a certain distance threshold to ensure that they are not from traces sampling a different path, see Figure 4. To analyze the effect of this thresholding, we have the following theorem:

THEOREM 5.1. *Let* $\|\epsilon\| \leq M$ *and* $\|\eta^{(i)}\| \leq M$. *Assume that the query trajectory* g *is sampled from* f *with* $g_k = f_k + \epsilon_k$ *and* $h^{(i)} = h + \eta^{(i)}$ *is another trace in the database where the lifted image* $\hat{f}$ *and* $\hat{h}$ *have distance at least* D, *i.e.,* $\sum_{k=-n}^{n} \|f_k - h_k\|^2 \geq D$. *Then, with probability at least* $1 - \exp(-D/128M^2)$, *the distance between the lifted image* $\hat{h}^{(i)}$ *and* $\hat{g}$ *is greater than* $D/2$.

PROOF. See Appendix D. □

Thus, if the lifted images of two points from two different trajectories are sufficiently far apart, the thresholding ensures that their samples will not affect each other's smoothing results with high probability. Note that it is often possible to achieve the necessary separation between two road sections in the lifted space by simply increasing the parameter n.

Note that with the smoothed traces, it becomes much easier to distinguish different lanes and directions of travel than from the raw data, while the general shape of the curves is much better preserved than in Figure 3, in particular in the elongated cloverleaf pattern near the bottom of the page. This effect of the thresholding process is consistent throughout the entire data set.

5.2 Reparameterizing Data

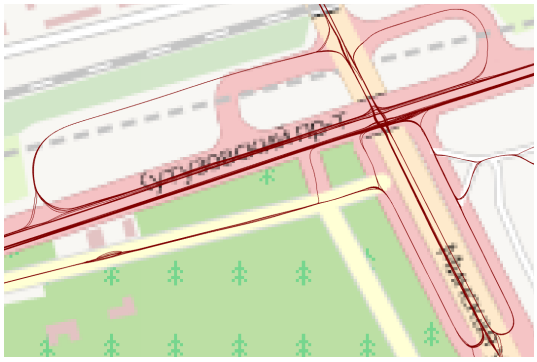

Figure 5: Effect of thresholded algorithm after reparameterizing data.

We have noticed that the smoothing algorithm sometimes forms tiny loops or cusps — these are a result of a vehicle slowing down dramatically and hence resulting in a point in the lifted space that is much further than it appears in the original space. Such features can be used to detect such anomalies in vehicle speed and direction that would have been less apparent in the original space.

If we are not interested in differentiating traces with different speed — as in the case of road network generation, we can reparameterize the input curves for constant speed, see Figure 5. Here, we see that the smoothed traces appear to have even less variance than the result in Figure 4. This can be attributed to the fact that without reparameterization, traces that appear to be close may have very different speeds, and hence their delay embedded points fall outside the threshold for inclusion into the smoothing algorithm.

5.3 Iterative Heuristic

In the event that we are smoothing an entire collection of traces, we can apply an *iterative* heuristic in which after we perform the smoothing step as analyzed in our framework, we iterate over the smoothed trajectories to further reduce error. That is, after we smooth each trajectory in the collection of traces, we use the smoothed trajectories as the new collection of traces to smooth. The effect of this smoothing approach is discussed in Section 6.4.

6. EXPERIMENTS AND EVALUATION

The smoothing step of the algorithms relies heavily on algorithms for the nearest neighbor (NN) problem: Given a set of P points, one would like to preprocess P into a data structure such that given query point q, one could efficiently find the closest point in P to q. This is a classical problem in computational geometry [39, 17], but in high dimensions, theoretical results that improve on brute force search remain elusive. However if query answers are allowed to be approximate, there exist efficient solutions when the underlying space is Euclidean [27, 20, 37, 4]. For more general metric spaces, there are algorithms for nearest neighbor search with a logarithmic query time, albeit with a constant that depends exponentially on the intrinsic dimension of the data [6]. In practice, efficient C++ libraries exist, such as the ANN Library by David Mount and Sunil Arya [32], which we use in our implementations. Our experiments were

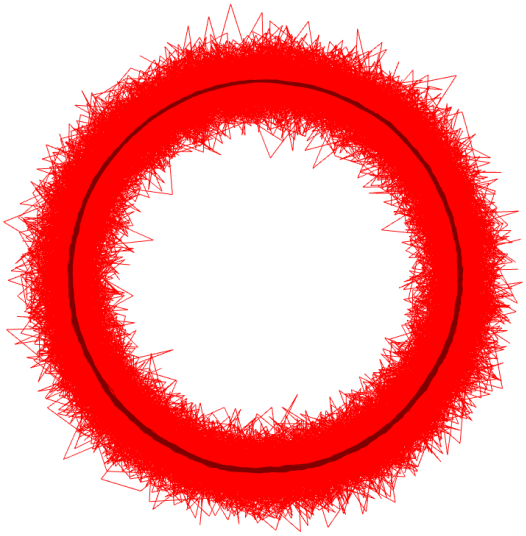

Figure 6: Synthetic data: The original trajectories are in red while the trajectories smoothed using the algorithm are in dark red.

done in the context of smoothing an entire collection of trajectories, as this allows us to see the effects of the iterative heuristic. When the embedding dimension is low, our algorithm is fast in practice: On a 2.33GHz Macbook Pro with 3GB of RAM, we can smooth 7145 traces from the Taxicab dataset with a total of 54510 points using a lifted space of dimension 9 in several seconds. The result of this is shown in Figures 10 and 13. The smoothing of the synthetic data set also took several seconds. For the Moscow dataset, we used a much higher embedding dimension and number of nearest neighbors, so the smoothing procedure took several hours. When the embedding dimension is high, a brute force nearest neighbor searching technique in lieu of the ANN library will most likely perform better. We note however, that the algorithm is fairly robust to parameter selection, and we can achieve results that are similar to the ones shown in Figures 9 and 12 in much less time.

6.1 Data Sets

To test the efficacy of the algorithm, we used two data sets: a synthetic collection of noisy and partial samples of a circular arc, and two real world data sets: One of 1331 GPS traces tagged "Moscow" from OpenStreetMap [2], and another of 7145 taxicab traces in an unindentified major city.

Synthetic Data Set.

To generate the synthetic data set, we discretized the circular arc from $\theta = 0$ to 3π on the unit circle into a sequence of 100 points. Then, to generate a individual trace, we selected a interval of this sequence uniformly at random, and then added an independent Gaussian noise with mean 0 and standard deviation of 0.1 to each point in this sequence. We repeated this process 100 times to generate 100 noisy and partial samples of the circular arc. This data set is depicted in Figure 6.

Moscow Data Set.

To evaluate the performance of the algorithm on real world data sets, we used a set of 1331 GPS traces downloaded from OpenStreetMap, as shown in Figure 8. This set of GPS traces was obtained by cutting a larger collection of GPS traces in Moscow to the window between 55.74° and 55.76° latitude and 37.58° and 37.62° longitude. Our visualization was implemented using the QGraphicsScene framework of Qt, which would not display our traces until we scaled them up. We chose to scale longitude by a factor of 3486 and latitude by a factor of 24855, which also resulted in a reasonable projection for the Moscow area. The following discussion in this section will be using these scaled units when referring to the Moscow data set. We then reparameterize the traces by sampling the original trajectories so that consecutive sample points are one units away from one another. A closeup of the data set can be seen in Figure 8, where it is evident that the traces are very noisy and have large variations in sampling error and density.

Figure 7: Moscow Data Set.

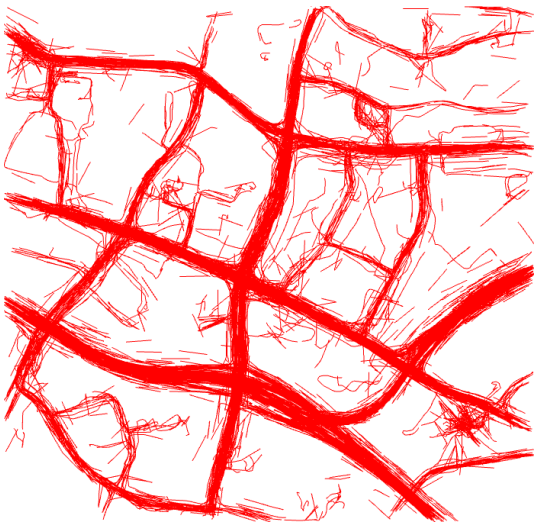

Figure 8: Taxicab Data Set.

Taxicab Data Set.

A third data set we used to evaluate the algorithm is a

	Synthetic	Moscow	Taxicab
Mean Dist Orig	0.0795	0.000143 $<$ 15.9m	0.000196
Mean Dist Smoothed	0.00553	0.000114 $<$ 12.7m	0.000156
Mean Dist Iter Smoothed	0.00461	0.000111 $<$ 12.3m	0.000174

Table 1: Fréchet distance to routes obtained using map matching on the original trajectories. The distances for the Moscow and Taxicab data sets are using longitude/latitude coordinates. Approximate upper bounds in meters for the Moscow data set are shown.

set of 7145 traces obtained by cutting a larger of collection of Taxicab GPS traces from an unidentified major city to a window of height 0.01612° latitude and width 0.01662° longitude. The taxicab traces were of varying sampling density and we only retained the sub-traces where consecutive sample points are less than 5 seconds apart and of distance less than 1000 meters apart. The latitude and longitude were both scaled by a factor of 1000, and consecutive samples were then taken every 0.1 units to reparameterize the traces.

6.2 Parameter Selection and Evaluation

We evaluated the results using both visual inspection and average distance to the underlying route. We selected parameters to produce reasonable results for both methods of evaluation. For visual inspection, we compared the smoothed traces to the underlying road network and verified that roads are not significantly distorted, pulled apart from each other, or contracted. To compute the average distance to the underlying route, we used the approximate Fréchet map matching algorithm in [10]. This algorithm finds a $(1+\epsilon)$-approximate closest path in the map with respect to the Fréchet distance. For two given curves $\pi_1 : [0, p_1] \to \mathbb{R}^d$ and $\pi_2 : [0, p_2] \to \mathbb{R}^d$, the Fréchet distance is defined as

$$\mathbf{d}_{\mathcal{F}}(\pi_1, \pi_2) := \inf_{\substack{\alpha:[0,1]\to[0,p_1] \\ \beta:[0,1]\to[0,p_2]}} \max_{t\in[0,1]} \|\pi_1(\alpha(t)) - \pi_2(\beta(t))\|_2$$

Intuitively, one can think of the Fréchet distance as the shortest leash possible for a man and a dog to walk on two curves, allowing their speeds to vary, but never walking backwards. For the ground truth map, we used maps downloaded from CloudMade [1] and we set $\epsilon = 0.1$. We note that the map matching is performed on the *original* trajectories and show that are smoothed trajectories are even closer to the matched routes than the original ones, even though the algorithm knows nothing about the underlying map. We measure distance to matched routes by using the average distance of the projection to the matched route.

The parameters we selected for the synthetic data set were $n = 21$ and 728 nearest neighbors while we selected $n = 64$ and 4096 nearest neighbors for the Moscow data set and $n = 9$ and 495 nearest neighbors for the Taxicab data set. We did not need to select a threshold for the synthetic data set, while we used a threshold of 64 units for the Moscow data set and 2 units for the Taxicab data set.

6.3 Results

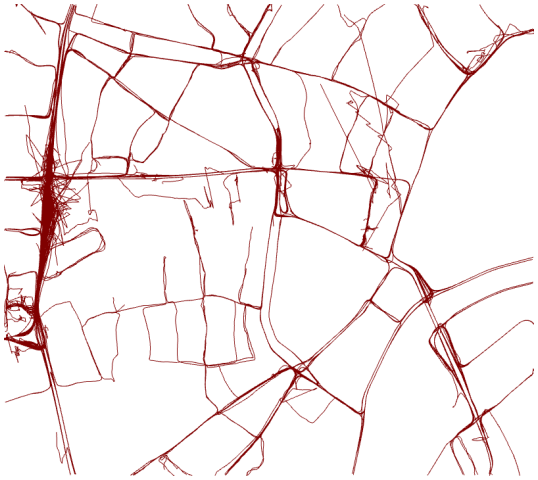

Figure 9: Smoothed Moscow Data Set.

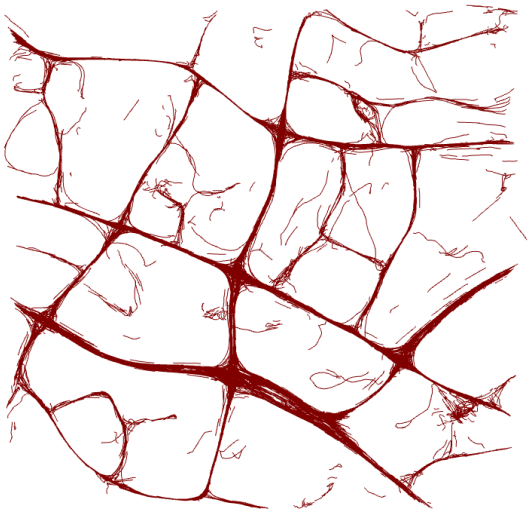

Figure 10: Smoothed Taxicab Data Set.

Visually, our results can be see in Figures 6, 9 and 10. We note that in the synthetic data set, our smoothing method produces traces along a smooth circle without contracting or otherwise shifting the centerline of the unit circle. We also note that in the Moscow and Taxicab data sets, error is reduced without distorting the curvature of individual roads. For the Moscow data set, we can now differentiate different directions of travel, but they are not artificially shifted apart. The Taxicab data set had much shorter traces, along with higher noise, so this effect was not as apparent with the parameters we selected. The results of the numerical performance metrics can be seen in Table 1. We note that for the Moscow and Taxicab data sets, on average, the traces were already very close to the map, and given that the map we used may not be entirely accurate, our results present a reasonable reduction in error.

6.4 Iterative Heuristic

We also tested the iterative heuristic by running the smoothing algorithm through four iterations, using the same parameters as before. The results are shown in Table 1

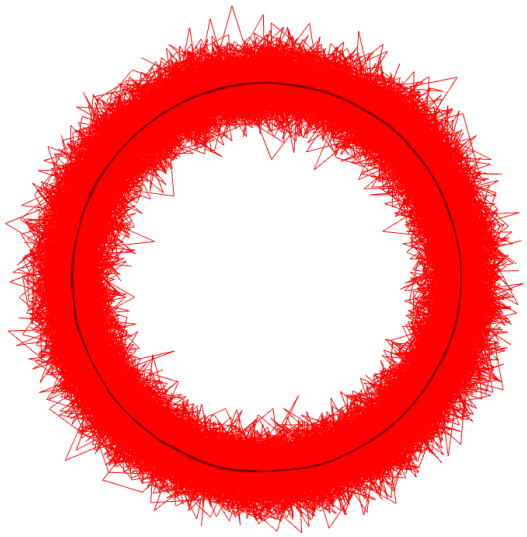

Figure 11: Synthetic data after 4 smoothing iterations.

Figure 12: Moscow data set after 4 smoothing iterations.

and Figures 11, 12, and 13. Interestingly, although the Taxicab data set appears much smoother after the iterative heuristic, the average distance to the matched route actually increased slightly. Possible reasons for this phenomenon include drift in the iterative smoothing process, inaccurate maps, or biased sampling from the original data set. Nevertheless, the iteratively smoothed collection of traces appears much cleaner, which can be useful for many applications.

7. CONCLUSION

We have presented a simple and practical algorithmic framework for smoothing a query trajectory with respect to a collection of traces. Under natural assumptions, our method can be proven to reduce variance among a noisy collection of traces. Our framework can also be viewed of as a generalization of the classical moving average technique, and we show additional connections to that method as well. Experimentally, we show that the algorithm works well on

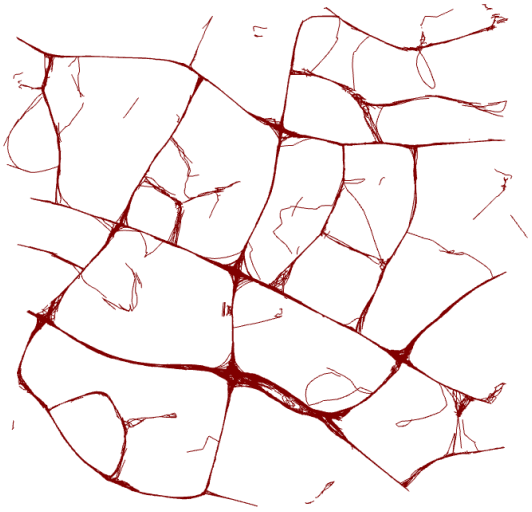

Figure 13: Taxicab data set after 4 smoothing iterations.

both synthetic data, and on two different collections of GPS traces. We also explore variations in the algorithm, and demonstrate how they result in different smoothing behavior. There are several directions for future work — these include inferring a reasonable number of delay coordinates and threshold value from the data itself without the use of a ground truth map, as well as examining special structure in the lifted images that might facilitate faster nearest neighbor queries.

8. ACKNOWLEDGMENTS

This work was supported in part by ARO grant W911NF-10-1-0037, the AHPCRC Center at Stanford (ARO grant W911NF-07-2-0027), NSF CCF grant 1011228, a Google Faculty Research grant, GIGA grant ANR-09-BLAN-0331-01, and the European project CG-Learning No. 255827. We would also like to thank the associated research team COMET.

9. REFERENCES

[1] Cloudmade. http://cloudmade.com/.

[2] Openstreetmap. http://www.openstreetmap.org/.

[3] D. Agrawal, P. Zhang, A. E. Abbadi, and M. F. Mokbel, editors. *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*. ACM, 2010.

[4] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51:117–122, January 2008.

[5] L. Arge, K. G. Larsen, T. Mølhave, and F. van Walderveen. Cleaning massive sonar point clouds. In Agrawal et al. [3], pages 152–161.

[6] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *International Conference on Machine Learning*, 2006.

[7] R. Bruntrup, S. Edelkamp, S. Jabbar, and B. Scholz. Incremental map generation with GPS traces. In

Proceedings IEEE Intelligent Transportation Systems, pages 574–579, 2005.

[8] L. Cao and J. Krumm. From gps traces to a routable road map. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 3–12, New York, NY, USA, 2009. ACM.

[9] F. Chazal, D. Cohen-Steiner, and Q. Mérigot. Geometric Inference for Probability Measures. *Journal on Foundations of Comp. Math. (see also INRIA research report RR-6930)*, 2011.

[10] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the fréchet distance. In *ALENEX*, 2011.

[11] D. Chen, C.-T. Lu, Y. Kou, and F. Chen. On detecting spatial outliers. *GeoInformatica*, 12:455–475, 2008. 10.1007/s10707-007-0038-8.

[12] F. Chen, C.-T. Lu, and A. P. Boedihardjo. Gls-sod: a generalized local statistical approach for spatial outlier detection. In B. Rao, B. Krishnapuram, A. Tomkins, and Q. Yang, editors, *KDD*, pages 1069–1078. ACM, 2010.

[13] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun. Leveraging spatio-temporal redundancy for rfid data cleansing. In *International Conference on Management of Data*, pages 51–62, 2010.

[14] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 900–911, 2011.

[15] R. Cheng, J. Chen, and X. Xie. Cleaning uncertain data with quality guarantees. *Proceedings of The Vldb Endowment*, 1:722–735, 2008.

[16] H. Chernoff. A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 1952.

[17] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, August 1988.

[18] E. Guilbert and H. Lin. B-spline curve smoothing under position constraints for line generalisation. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, GIS '06, pages 3–10, New York, NY, USA, 2006. ACM.

[19] M. M. Hall, A. N. Alazzawi, A. A. Abdelmoty, and C. B. Jones. Improving the quality of gps-based personal gazetteers.

[20] S. Har-Peled. A replacement for voronoi diagrams of near linear size. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:94, 2001.

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

[22] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[23] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. London: Chapman & Hall, 1990.

[24] N. Hönle, M. Grossmann, S. Reimann, and B. Mitschang. Usability analysis of compression algorithms for position data streams. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 240–249, New York, NY, USA, 2010. ACM.

[25] H. Huang, Y. Zhu, X. Li, M. Li, and M.-Y. Wu. Meta: A mobility model of metropolitan taxis extracted from gps traces. In *WCNC*, pages 1–6, 2010.

[26] L. Huang, Q. Li, and Y. Yue. Activity identification from gps trajectories using spatial temporal pois' attractiveness. In *Proceedings of the ACM SIGSPATIAL International Workshop on Location Based Social Networks*, 2010.

[27] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[28] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. *CoRR*, abs/1002.0963, 2010.

[29] S. Johansson and M. Jern. Geoanalytics visual inquiry and filtering tools in parallel coordinates plots. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, GIS '07, pages 33:1–33:8, New York, NY, USA, 2007. ACM.

[30] D. G. Kendall. Shape Manifolds, Procrustean Metrics, and Complex Projective Spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984.

[31] X. Liu, C.-T. Lu, and F. Chen. Spatial outlier detection: random walk based approaches. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 370–379, New York, NY, USA, 2010. ACM.

[32] D. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/ mount/ANN/.

[33] M. Okamoto. Some inequalities relating to the partial sum of binomial probabilities. *Annals of the Institute of Statistical Mathematics*, 1958.

[34] B. Oksendal. *Stochastic differential equations (3rd ed.): an introduction with applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[35] A. Panangadan and A. Talukder. A variant of particle filtering using historic datasets for tracking complex geospatial phenomena. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 232–239, New York, NY, USA, 2010. ACM.

[36] D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *The Annals of Statistics*, 22:1346–1370, 1994.

[37] Y. Sabharwal, N. Sharma, and S. Sen. Nearest neighbors search using point location in balls with applications to approximate voronoi decompositions. *J. Comput. Syst. Sci.*, 72:955–977, September 2006.

[38] G. Sánchez, J. Lladós, and K. Tombre. A mean string algorithm to compute the average among a set of 2d shapes. *Pattern Recogn. Lett.*, 23:203–213, January 2002.

[39] M. I. Shamos and D. Hoey. Closest-point problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:151–162, 1975.

[40] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers. *Intell. Data Anal.*, 6:451–468, October 2002.

[41] S. Shekhar, C. tien Lu, and P. Zhang. A unified approach to detecting spatial outliers. *Geoinformatica*, 7:139–166, 2003.

[42] F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer Berlin / Heidelberg, 1981. 10.1007/BFb0091924.

[43] C. tien Lu, D. Chen, and Y. Kou. Multivariate spatial outlier detection. *International Journal on Artificial Intelligence Tools*, 13:801–812, 2004.

[44] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[45] H. Yoon, Y. Zheng, X. Xie, and W. Woo. Smart itinerary recommendation based on user-generated gps trajectories. In *Ubiquitous Intelligence and Computing - 7th International Conference, UIC 2010, Xi'an, China, October 26-29, 2010. Proceedings*, pages 19–34, 2010.

[46] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In Agrawal et al. [3], pages 99–108.

[47] L. Zhang, F. Thiemann, and M. Sester. Integration of gps traces with road map. In *Proceedings of the International Workshop on Computational Transportation Science*, 2010.

[48] Y. Zheng and X. Xie. Learning travel recommendations from user-generated gps traces. *ACM Transaction on Intelligent Systems and Technology*, 2, 2011.

APPENDIX

A. PROOF OF THEOREM 4.5

PROOF. When the sampled time points on $f^{(i)}$ are not synchronized with that on g. Let k be the time points on $f^{(i)}$ which is closest to 0. We consider two cases: either $|x| < 1/4$ or $|x| \geq 1/4$. We will show that in any of the two cases, the lifted image on $\hat{f}^{(i)}$ whose delay coordinates centered at k will be one of the two nearest neighbors of $g_{-n}, \cdots, g_n$.

Generally, we consider the probability that the lifted image $g_{-n}, \cdots, g_n$ is closer to $f^{(i)}_{k-n}, f^{(i)}_{k-n+1}, \cdots, f^{(i)}_{k+n}$ than $f^{(i)}_{k+d-n}, \cdots, f^{(i)}_{k+d+n}$. Then we have

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{j=-n}^{n}\|g_j - f^{(i)}_{k+j}\|^2 \geq \sum_{j=-n}^{n}\|g_j - f^{(i)}_{k+j+d}\|^2\Big)\\
&= \mathbb{P}\Big(\sum_{j=-n}^{n}\|f_j + \eta_j - f_{k+j} - \epsilon^{(i)}_{k+j}\|^2\\
&\qquad \geq \sum_{j=-n}^{n}\|f_j + \eta_j - f_{k+j+d} - \epsilon^{(i)}_{k+j+d}\|^2\Big)\\
&= \mathbb{P}\Big(\sum_{j=-n}^{n} 2(\eta_j - \epsilon^{(i)}_{k+j})^T(f_j - f_{k+j})\\
&\qquad + \sum_{j=-n}^{n} 2(\eta_j - \epsilon^{(i)}_{k+j+d})^T(f_{k+j+d} - f_j)\\
&\qquad + \sum_{j=-n}^{n} (\epsilon^{(i)}_{k+j+d} - \epsilon^{(i)}_{k+j})^T(2\eta_j - \epsilon^{(i)}_{k+j} - \epsilon^{(i)}_{k+j+d})\\
&\qquad \geq \sum_{j=-n}^{n}\|f_j - f_{k+j+d}\|^2 - \sum_{j=-n}^{n}\|f_j - f_{k+j}\|^2\Big)
\end{aligned}$$

Denote $\sum_{j=-n}^{n}\|f_j - f_{k+j+d}\|^2 - \sum_{j=-n}^{n}\|f_j - f_{k+j}\|^2$ by t, then we have the above probability is bounded by

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{j=-n}^{n} 2(\eta_j - \epsilon^{(i)}_{k+j})^T(f_j - f_{k+j}) \geq \frac{t}{3}\Big)\\
&\quad + \mathbb{P}\Big(\sum_{j=-n}^{n} 2(\eta_j - \epsilon^{(i)}_{k+j+d})^T(f_{k+j+d} - f_j) \geq \frac{t}{3}\Big)\\
&+ \mathbb{P}\Big(\sum_{j=-n}^{n} (\epsilon^{(i)}_{k+j+d} - \epsilon^{(i)}_{k+j})^T(2\eta_j - \epsilon^{(i)}_{k+j} - \epsilon^{(i)}_{k+j+d}) \geq \frac{t}{3}\Big)
\end{aligned}$$

Since $2(\eta_j - \epsilon^{(i)}_{k+j})(f_j - f_{k+j})$ are mean zero independent random variables and they are bounded by $2M\|f_j - f_{k+j}\|$, by concentration inequality, we know that

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{j=-n}^{n} 2(\eta_j - \epsilon^{(i)}_{k+j})^T(f_j - f_{k+j}) \geq \frac{t}{3}\Big)\\
&\qquad \leq \exp\left(-\frac{t^2}{96\sum_{j=-n}^{n} M^2(f_j - f_{k+j})^2}\right) \quad (8)
\end{aligned}$$

For the same reason, we have

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{j=-n}^{n} 2(\eta_j - \epsilon^{(i)}_{k+j+d})^T(f_{k+j+d} - f_j) \geq \frac{t}{3}\Big)\\
&\qquad \leq \exp\left(-\frac{t^2}{96\sum_{j=-n}^{n} M^2(f_j - f_{k+j+d})^2}\right) \quad (9)
\end{aligned}$$

Note that

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{j=-n}^{n} (\epsilon^{(i)}_{k+j+d} - \epsilon^{(i)}_{k+j})^T(2\eta_j - \epsilon^{(i)}_{k+j} - \epsilon^{(i)}_{k+j+d}) \geq \frac{t}{3}\Big)\\
&\qquad \leq \mathbb{P}\Big(\sum_{j=-n}^{n} 2\|\eta_j\|^2 + 2\|\epsilon_{k+j}\|^2 \geq \frac{t}{3}\Big) \quad (10)
\end{aligned}$$

Now we consider the two cases:

1. The nearest time point k to 0 satisfies $|k| < 1/4$. In such a case, let $|d| \geq 1$, then under the assumption that $2S_1 \geq S_2$ we can easily verify that

$$t = \sum_{j=-n}^{n} \|f_j - f_{k+j+d}\|^2 - \sum_{j=-n}^{n} \|f_j - f_{k+j}\|^2$$
$$\geq S_1(k+d)^2 - S_2k^2 \geq \frac{1}{2}S_1d^2$$

Thus, using concentration inequality we know (8) and (9) are both bounded by $\exp(-C_1S_1d^2n/M^2)$ where C_1 is a constant. (10) can also be bounded by $\exp(-C_2\sigma^4dn/M^4)$ for the same reason as in the proof of Lemma 4.2. By summing up the tails, we know that with probability at least $1 - 8\exp(-C\sigma^4n/M^4)$, the lifted image whose delay coordinates centered at x on $f^{(i)}$ is the nearest neighbor of $g_{-n}, \cdots, g_n$.

2. The nearest time point k to 0 lies outside of $[-1/4, 1/4]$. In such a case, without loss of generality, we can assume $k < 0$. When $d < -1$ or $d > 2$, we have

$$t = \sum_{j=-n}^{n} \|f_j - f_{k+j+d}\|^2 - \sum_{j=-n}^{n} \|f_j - f_{k+j}\|^2$$
$$\geq S_1(k+d)^2 - S_2k^2 \geq \frac{1}{2}S_1d^2$$

Thus, for any $d < -1$ or $d > 2$, (8) and (9) are bounded by $\exp(-C_1S_1d^2n/M^2)$; (10) is bounded by $\exp(-C_2\sigma^4dn/M^4)$ for certain constants C_1 and C_2, by summing up of the tails, we know that with at least $1-8\exp(-C\sigma^4n/M^4)$, the lifted image whose delay coordinates centered at x on $f^{(i)}$ will be one among the two nearest neighbor of $g_{-n}, \cdots, g_n$.

□

B. PROOF OF LEMMA 4.7

PROOF. Denote the functional

$$\int_0^1 (y'(t) - S)^2 dt + \lambda \int_0^1 (y(t) - g(t))^2 dt$$

as $A(y)$. Suppose $\eta : [0,1] \to \mathbb{R}$ is a C^2 function and vanishes at the end points 0 and 1. Then we must have:

$$A(y) \leq A(y + \epsilon\eta)$$

for any value ϵ close to 0. Therefore the derivative of $A(y + \epsilon\eta)$ with respect to ϵ must vanish at $\epsilon = 0$. So

$$\begin{aligned}
\frac{dA}{d\epsilon}|_{\epsilon=0} &= \int_0^1 \frac{d}{d\epsilon}(y'(t) + \epsilon\eta'(t) - S)^2|_{\epsilon=0}dt \\
&\quad + \lambda \int_0^1 \frac{d}{d\epsilon}(y(t) + \epsilon\eta(t) - g(t))^2|_{\epsilon=0}dt \\
&= \int_0^1 \eta'(t)(y'(t) - S)dt + \lambda \int_0^1 \eta(t)(y(t) - g(t))dt \\
&= \eta(t)(y'(t) - S)|_0^1 - \int_0^1 \eta(t)y''(t)dt \\
&\quad + \lambda \int_0^1 \eta(t)(y(t) - g(t))dt \\
&= \int_0^1 \eta(t)(\lambda y(t) - \lambda g(t) - y''(t))dt
\end{aligned}$$

Thus,

$$\int_0^1 \eta(t)(\lambda y(t) - \lambda g(t) - y''(t))dt = 0$$

for any twice differentiable function η that vanishes at the endpoints.

We can now apply the fundamental lemma of calculus of variations: If

$$\int_a^b \eta(t)H(t)dt = 0$$

for any sufficiently differentiable function $\eta(t)$ within the integration range that vanishes at the endpoints of the interval, then it follows that $H(t)$ is identically zero on its domain. Hence

$$\lambda y(t) - \lambda g(t) - y''(t) = 0.$$

□

C. PROOF OF LEMMA 4.8

PROOF. The general solution to the differential equation (2) is given by $y(t) = y_p(t) + y_h(t)$ where $y_p(t)$ is a particular solution of the nonhomogeneous equation (2) and $y_h(t)$ is the general solution of the associated homogeneous equation

$$y''(t) - \lambda y(t) = 0.$$

The solution to the homogeneous equation is given by $c_1e^{-\sqrt{\lambda}t} + c_2e^{\sqrt{\lambda}t}$. Now we verify that $y_p(t) = \int_R \frac{\sqrt{\lambda}}{2}e^{-\sqrt{\lambda}|r-t|}g(r)dr$ is a particular solution to (2). To show this, we note that

$$\begin{aligned}
y_p(t) = \frac{\sqrt{\lambda}}{2}e^{-\sqrt{\lambda}t}\int_{-\infty}^{t} e^{\sqrt{\lambda}r}g(r)dr \\
+ \frac{\sqrt{\lambda}}{2}e^{\sqrt{\lambda}t}\int_t^{\infty} e^{-\sqrt{\lambda}r}g(r)dr
\end{aligned}$$

which implies that

$$\begin{aligned}
y_p'(t) = -\frac{\lambda}{2}e^{-\sqrt{\lambda}t}\int_{-\infty}^{t} e^{\sqrt{\lambda}r}g(r)dr \\
+ \frac{\lambda}{2}e^{\sqrt{\lambda}t}\int_t^{\infty} e^{-\sqrt{\lambda}r}g(r)dr
\end{aligned}$$

and

$$y_p''(t) = \frac{\lambda^{3/2}}{2} e^{-\sqrt{\lambda}t} \int_{-\infty}^{t} e^{\sqrt{\lambda}r} g(r) dr + \frac{\lambda^{3/2}}{2} e^{\sqrt{\lambda}t} \int_{t}^{\infty} e^{-\sqrt{\lambda}r} g(r) dr - \lambda g(t).$$

The last identity is equivalent to

$$y_p''(t) = \lambda y(t) - \lambda g(t).$$

□

D. PROOF OF THEOREM 5.1

PROOF. We consider the probability that $(g_{-n}, \cdots, g_n)$ and $(h_{-n}^{(i)}, \cdots, h_n^{(i)})$ is less than D:

$$\begin{aligned}
&\mathbb{P}\Big(\sum_{k=-n}^{n} \|g_k - h_k^{(i)}\|^2 \le D/2 \Big) \\
&= \mathbb{P}\Big(\sum_{k=-n}^{n} \|\eta_k^{(i)} - \epsilon_k\|^2 + \|f_k - h_k\|^2 \\
&\qquad + 2(\epsilon_k - \eta_k^{(i)})(f_k - h_k) \le D/2 \Big) \\
&\le \mathbb{P}\Big(\sum_{k=-n}^{n} 2(\epsilon_k - \eta_k^{(i)})(f_k - h_k) \le D/2 - D \Big) \\
&\le \exp\Big(-2(D/2 - D)^2 / \sum_{k=-n}^{n} 64 M^2 \|f_k - h_k\|^2 \Big) \\
&= \exp\Big(-(D/2 - D)^2 / 32 M^2 D \Big) \\
&= \exp\Big(-D/128 M^2 \Big)
\end{aligned}$$

□