Kernelized Map Matching for Noisy Trajectories

Ahmed Jawad , Kristian Kersting
Knowledge Discovery Dept., Fraunhofer IAIS, 53754 Sankt Augustin, Germany
ahmed.jawad@iais.fraunhofer.de , kristian.kersting@iais.fraunhofer.de

Abstract

Map matching is a fundamental operation in many applications such as traffic analysis and location-aware services, the killer apps for ubiquitous computing. In the past, several map matching approaches have been proposed. Roughly speaking they can be categorized into four groups: geometric, topological, probabilistic, and other advanced techniques. Surprisingly, kernel methods have not received attention yet although they are very popular in the machine learning community due to their solid mathematical foundation, tendency toward easy geometric interpretation, and strong empirical performance in a wide variety of domains. In this paper, we show how to employ kernels for map matching. Specifically, ignoring map constraints, we first maximize the consistency between the similarity measures captured by the kernel matrices of the trajectory and relevant part of the street map. The resulting relaxed assignment is then "rounded" into an hard assignment fulfilling the map constraints. On synthetic and real-world trajectories, we show that kernels methods can be used for map matching and perform well compared to probabilistic methods such as HMMs.

1 Introduction

Map matching is a fundamental operation for many real-world applications that are currently changing how we as a society use computers in daily life. Following [Mitchell, 2009]: after decades of analysing "historical" data, 'location-based services' are a major driving force for analysing "real-time" and "real-space" data that record personal activities, conversations, and movements in an attempt to improve human health, guide traffic, and advance the scientific understanding of human behaviour. Therefore, it is not surprising that map matching — the problem of mapping a temporal sequence of coordinates onto a given network — has received a lot of attention.
Roughly speaking existing approaches for map matching can be categorized into four groups: geometric, topological,probabilistic, and other advanced techniques. Surprisingly, however, kernel methods such as support vector machine and Gaussian processes have not received attention yet although they are very popular in the machine learning community due to their solid mathematical foundation, tendency toward easy geometric interpretation, and strong empirical performance in a wide variety of domains. In a nutshell, kernel methods first process a dataset into a kernel matrix that roughly expresses the idea that two data points are "equivalent as far as some function f of the data can tell". By representing the data in terms of a kernel matrix, the data can be of various types, and also heterogeneous types such as trees and graphs. This makes kernel approaches very flexible. In a second step, a variety of kernel algorithms that have been developed can be used to analyse the data, using only the information contained in the kernel matrix. Kernels are, also, readily extendible therefore every kernel matrix provides an opportunity to integrate its knowledge with existing kernels in the field. An investigation of using kernel methods for map matching motivated by their well-known strength was the seed that grew into our proposal for kernelized map matching. Through this article, we make the following contributions in map matching research:

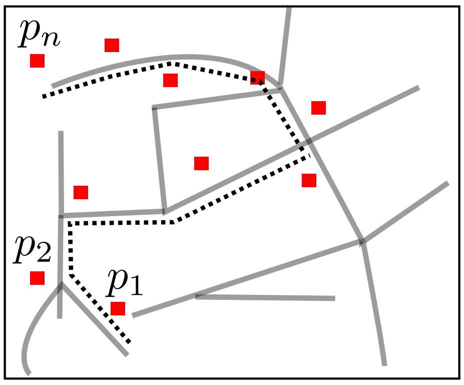


Figure 1: illustration of the map matching problem: Given a graph $G = (V, E)$, denoted as gray edges, and a trajectory $T = \{p_1, p_2, ..., p_n\}$, denoted as red squares, find the ground truth path (dashed line) starting from T.

1. Establishing a link between 'map matching' and 'kernel methods'.
2. Specifically building and investigating an instantiation of this link: a simple yet effective kernelized map matching approach, called 'Kernelized Map Matching' (KMM).
3. To do so, we also develop a simple kernel to capture spatio-temporal correlations under one umbrella.

Specifically, triggered by the observation that matching a trajectory of coordinates would be easy if the observed coordinates were noise-free — the coordinates would simply constitute the solution — one may propose to treat the map matching problem as a regression problem. That is, we treat a trajectory as a function t for which we observe a

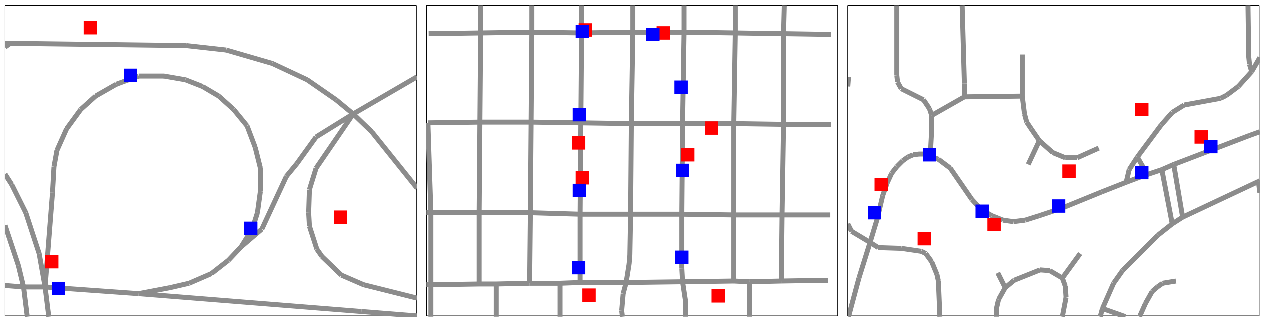

(a) KMM: over a roundabout (b) KMM: over grid structure (c) KMM: Robustness to noise

Figure 2: illustrations of KMM's performance on challenging real-world situations. The red points show the input trajectory and the blue points the output path by KMM. In all three cases, KMM recovers the exact groundtruth paths. (coloured)

sequence of noisy values, the coordinates $t(i) + \epsilon$ at inputs $i = 1, 2, 3, \ldots, k$, the temporal order of coordinates. The task is now: estimate the noise-free function t from the noise observations. In contrast to most traditional regression tasks, however, outputs are structured due to the physical constraints in the world and in turn there are non-linear dependencies among coordinates. Although kernel methods are powerful tools for modelling non-linear dependencies, and hence seem to be relevant for map matching, most kernel (regression) models focus on the prediction of a single output. Although generalizations to multiple outputs can often be achieved by training independent models for each one or tying parameters across dimensions, this fails to account for output correlations [Weston and Vapnik, 2002]. Consequently, we propose a different approach, namely to "embed" the output of F, i.e., the coordinates of trajectory into the same space as the trajectory and the network and in turn reducing the noise while still capturing the multi-output, non-linear dependencies present in trajectories. Specifically, ignoring map constraints, we first embed the trajectory and hence reduce noise. The resulting relaxed assignment is then "rounded" into an hard assignment fulfilling the map constraints. On synthetic and real-world trajectories, we show that this approach, called kernelized map matching, can be used for map matching and performs well compared to state-of-the-art hidden Markov models. Fig. 2 shows KMM performance over some of the tasks considered difficult among map matching experts [Newson and Krumm, 2009].
We proceed as follows. We start off by reviewing related work and mathematical background. In Section 3, we then introduce kernelized map matching. Before concluding, we present the results of our experimental evaluation in Section 4.

2 Related work:

The work presented in this paper is conceptually built upon three areas of research, namely map matching, structural embeddings and kernel methods. We briefly describe the background work in each of these areas respectively.

Quddus *et al.* [Quddus *et al.*, 2007] provide a good survey of existing map matching algorithms. Following them, existing approaches can be broadly categorized into four categories (or a combinations of them): geometric [Fox *et al.*, 2003], topological , probabilistic [Quddus *et al.*, 2007] and other advanced techniques [Taylor *et al.*, 2001]. Specifically, geometric map matching approaches utilize the shape of the spatial road network without considering the continuity or connectivity of it. Topological approaches use the connectivity and other topological features to restrict the candidate matches for solution points. Finally, probabilistic and advanced approaches are referred to as those "using more refined concepts such as a Kalman Filter, a fuzzy logic model or the application of hidden Markov model". Another dimension along which map matching approaches can be distinguished is "incremental vs. global" [Lou *et al.*, 2009].
In the present paper, we take the traditional, data-driven view of map matching (refer to [Lou *et al.*, 2009] for more details) which is defined as

> **Given** *a trajectory T and a street network G,* **find** *a path P in G that matches T with its real or ground truth path.*

Next to traditional map matching approaches, our approach builds upon ideas from structural embeddings [Quadrianto *et al.*, 2009], [Guo *et al.*, 2008]. The idea is to view trajectories as a special type of graph called Euclidean graph [Pemmaraju and Skiena, 2003].

Definition 1 (Euclidean graph)*: A Euclidean graph $G(V, E)$ is a graph in which*

1. *the vertices represent points in the Euclidean plane $\mathbb{R}^2$, and*
2. *the edges are assigned lengths equal to the Euclidean distance between those points, i.e., a straight line between corresponding vertices.*

In order to view a trajectory as an Euclidean graph, we need to capture the spatial and temporal aspects of the trajectory inside it. The spatial aspect of a trajectory is already captured in the Euclidean graph through $\mathbb{R}^2$ coordinates of the graph's vertices. In order to capture the temporal aspects of a trajectory, we time stamp the vertices of the graph in the order of the trajectory. The poly-line structure of street network graph with its vertices embedded in $\mathbb{R}^2$ is, indeed, an Euclidean graph. To make it explicit, we can add additional vertices to every corner of the lines constituting a road segment. Consequently, map matching, can be viewed as a problem of

> "*matching, i.e., finding similarity between two Euclidean graphs*".

Indeed, Euclidean graphs can actually be found in many machine learning problems such as map matching [Brakatsoulas *et al.*, 2005], shape analysis, protein structure analysis, time series similarity analysis, among others. However, —to the best of our knowledge— ,the problem of matching Euclidean graphs, i.e., "finding similarity between two Euclidean graphs" has not been considered yet. Instead, the problem has been approximated by casting it into a traditional "embedding" respectively "graph matching" problem and in turn dropping important information. Specifically, *graph matching* is typically formulated as a problem where we only seek a node-to-node matching between the input and output graphs. Euclidean graph matching is different as the nodes of the input graph (trajectory) can be mapped to any point lying on the edges of the output graph (street network graph).Embedding [Lee and Verleysen, 2007], on the other hand, is a generic problem: find a set of points in a (typically) low-dimensional space having similar properties and relationships as the points in the original input space. So far, however, it has only been considered for matching graphs in the traditional sense discussed above, see e.g. [Lee and Verleysen, 2007], [Guo *et al.*, 2008; Quadrianto *et al.*, 2009].Hence, this approach suffers from the same issues as the standard graph matching approach. Nevertheless, they employ kernel methods. We use a simple embedding technique called Multidimensional scaling(MDS) [Cox and Cox, 2000] defined as following:

Definition 2 Multidimensional Scaling(MDS)*: MDS is an embedding technique which uses the so-called stress function (the sum of squared differences between the similarity criteria of input and latent spaces) to come up with embedding X for input Y. Lets say K_x and K_y denote the similarity matrices for output X and input Y. Then the objective function F_o using MDS can be described as follows:*

$$F_o = \underset{X}{\operatorname{argmin}} \sum_{i,j} (K_x(i,j) - K_y(i,j))^2 \qquad (1)$$

Embedding approaches i.e, MDS and other, are called 'kernelized embedding' [Guo *et al.*, 2008] when they use kernel matrices as similarity criteria. Kernel Methods help with embedding in special cases because they are more suitable to non-vectorial data sets (text, images, protein sequences, graphs, trajectories, etc..) [Shawe-Taylor and Cristianini, 2004]. Our method is inspired by these techniques but generalizes them to the Euclidean graph case.

3 Kernelized Map Matching

Recall from the introduction that map matching would be easy if the observed coordinates were noise-free. In this case the observed coordinates would simply constitute the solution. In general, we cannot expect to reduce the noise completely. Consequently, we propose a two steps approach:
(1) To reduce the noise, embed the trajectory from $\mathbb{R}^2$ into $\mathbb{R}^2$ using kernel methods to capture the multi-output, non-linear dependencies present in trajectories.
(2) To account for remaining noise, "round" the embedding into an hard matching. We will now explain each of the steps in turn.

3.1 Euclidean Graph Matching

We proceed with some notation and problem description. $G(V,E)$ denotes a trajectory (input graph) where vertices are indexed by time t , i.e. $V = \{v_t\}_{t=1}^{|V|}, v_t \in \mathbb{R}^2$, while $G^{'}(V^{'},E^{'})$ denotes street network graph (output graph) where $V^{'} = \{v_i^{'}\}_{i=1}^{|V^{'}|}, v_{i'} \in \mathbb{R}^2$ are the set of nodes for street segments and $E^{'} = \{e_{i,j}^{'}\}_{i,j=1}^{|V^{'}|}, e_{i,j}^{'} \in \{\theta v_i + (1-\theta)v_j, \theta \in [0,1]\}$, if $e_{i,j}^{'}$ corresponds to a street segment otherwise it is empty. Notice that $e_{i,j}^{'}$ is defined as a convex combination of vertices, i.e a straight line between the nodes .In general, a street segment is defined as a poly-line however we can always divide it into a set of straight lines considering each corner of lines as a vertex to match our representation. Figure 3 illustrates the meaning of a street segment in $V^{'}$ as a convex combination. Φ denotes the

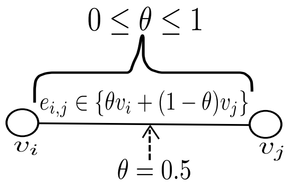


Figure 3: An edge in $G = (V, E)$ as a convex combination of two vertices

vector of mappings between a trajectory G and street network $G^{'}$. Unlike traditional graph matching where nodes of one graph are only matched to nodes of another graph, we need a mapping Φ which matches nodes of our input graph(trajectory) to any location over street segments in $E^{'}$: i.e. $\Phi_t = \Phi : v_t \mapsto e^{'} \times [0,1]$ where the interval $[0,1]$ specifies a value of θ pointing the exact location of mapping over a street segment in E'.
The truth value of vertex assignments to street segments can be stored in a binary matrix Δ, i.e. $\Delta \in \{0,1\}^{|V|\times|E^{'}|}$ subject to $\Delta^\top \mathbf{1} = \mathbf{1}$. Here $\mathbf{1}$ denotes a column vector of all ones. The last constraint enforces that each trajectory point can only be mapped to one street edge at a time.

Definition 3 *(Map Matching) Find a correspondence Φ between vertices of trajectory G and locations on street network G' such that the two matched sets, V and Φ, look most similar according to an objective criteria F_o. The problem is solved through finding an assignment of V to points lying on the edge set E' that minimizes the criteria F_o.*

The function F_o is of special interest and should have a form which preserves the relationships among input points while translating them to output space. We further assume that F_o is a decomposable function of a summation of basis functions, denoted by f_{Φ_i,Φ_j}.Thus to minimize F_o, we need to minimize the individual entries of summation. We also introduce the assignment matrix Δ in F_o. The assignment matrix Δ, enforces the Euclidean graph (or street network) constraints on the output i.e

$$F_o = \underset{\mathbf{\Phi}}{\operatorname{argmin}} \sum_{l}^{|E^{'}|} \sum_{i,j}^{|V|} \Delta_{i,l} \cdot f_{\Phi_i,\Phi_j} \qquad (2)$$
$$s.t \;\; \Delta \in \{0,1\}^{|V|\times|E^{'}|}, \Delta^\top \mathbf{1} = \mathbf{1}$$

Eq. (2) describes the map matching problem as a so-called integer linear program (ILP) which are generally known to be NP-hard except for a few classes of them. Such mathematical programs have a discrete and a continuous part. In our case, the discrete part chooses the correct combination of trajectory point ($v \in V$) versus street segment ($e' \in E'$). The total number of combinations is $V^{E'}$, which easily gets intractable even for a modest number of trajectory points and street network segments. Relaxation is a standard technique to reduce the complexity of ILP problems. In relaxation, we drop the discrete part of the problem to make it a standard linear programming problem where the optimization can be carried out in polynomial time. The result of the optimization is then rounded into a hard assignment fulfilling the discrete constraints to get an approximate solution. In our case, discrete constraints amount to street network constraints and relaxation means ignoring these constraints. Eq. (3) describes the unconstrained F_o

$$F_o = \underset{\Psi}{\operatorname{argmin}} \sum_{i,j}^{|V|} f_{\Psi_i, \Psi_j} \tag{3}$$

Notice that we have changed the output vector Φ to intermediate output Ψ. The mapping Ψ is different from Φ as it maps a a vertex v_t to $\mathbb{R}^2$: i.e $\Psi_t = \Psi : v_t \mapsto \mathbb{R}^2$.Now the solution of the optimization will be an approximation of the path used by trajectory instead of the original path. Afterwards we can convert this approximate path into the street network path through a rounding step. Hence, we provide a two step framework for the solution of Eq. (2)

- Optimize the relaxed objective function to approximate the trajectory path
- Provide a rounding scheme for assignment of step a output to street network.

We proceed by describing the details of these two steps in turn.

Optimization Step

Eq. (3) describes F_o as a summation of entries f_{Ψ_i, Ψ_j}. Now we come to details of an individual entry f_{Ψ_i, Ψ_j}. For this purpose, we define two Kernel matrices K_G and $K_{G'}$, where $K_{G_{i,j}}$ is the kernel function between trajectory vertices v_i and v_j: i.e $K_{G_{i,j}} = k_G(v_i, v_j)$ and $K_{G'_{i,j}}$ is the kernel function between the mappings Ψ_i, Ψ_j of the vertices v_i and v_j i.e $K_{G'_{i,j}} = k_{G'}(\Psi_i, \Psi_j)$. The widths of the Kernels K_G and $K_{G'}$ are denoted by σ_G and $\sigma_{G'}$. Now we define f_{Ψ_i, Ψ_j} as the 'difference of kernels' function.

$$f_{\Psi_i, \Psi_j} = (k_G(v_i, v_j) - k_{G'}(\Psi_i, \Psi_j))^2 \tag{4}$$

Now we can substitute the value of f_{Ψ_i, Ψ_j} in Eq. (3).

$$F_o = \underset{\Psi}{\operatorname{argmin}} \sum_{i,j}^{|V|} (k_G(v_i, v_j) - k_{G'}(\Psi_i, \Psi_j))^2 \tag{5}$$

Eq. (5) comes from an embedding technique known as Multidimensional scaling described in Eq. (1). We further note that Eq. (5) is like a regression equation with multiple outputs where we want to preserve the correlation among inputs during our structured prediction process and it can be used for embeddings across different spaces and structures, however our case is a special case where the input and output spaces are the same. To encode our prior knowledge that the solution of the embedding lies in the spatial neighbourhood perturbed by Gaussian noise, we add a regularization term which fuses the input and output space into one. We propose a kernel function $k_{GG'}$ between the respective points of our input and output graphs and define it as

$$k_{GG'}(\lambda, \sigma_N, i) = e^{-((v_i - \Psi_i)^2 - \lambda\sigma_N^2)^2 / 2\sigma_N^2} \tag{6}$$

where λ is a stiffness parameter of the kernel function while σ_N , the kernel width, is the estimated standard deviation of noise in trajectory. We finalize our objective function F_o as following:

$$F_o = \underset{\Psi}{\operatorname{argmin}} \sum_{ij} (K_{G_{i,j}} - K_{G'_{i,j}})^2 - \sum_i k_{GG'}(\lambda, \sigma_N, i) \tag{7}$$

Eq. (7) is the final objective function which we are using in kernelized map matching. However alternative embedding approaches can also be used in principal. We can take the derivative of F_o in Eq. (7) w.r.t Ψ and can find the answer. We use 'scaled conjugate gradient' algorithm for optimization [Shewchuk, 1994]. For assignment purposes, we apply a rounding scheme on the result of continuous optimization. We implement the optimization step in sliding window style of width k_w, because it makes the solution real time, localized and efficient.

Spatio-temporal Kernel over Euclidean Distances

The kernels $K_{G_{i,j}}$ and $K_{G'_{i,j}}$, that we have used are similar to RBF kernels. The only difference with an RBF kernel is that we use 'the sum of euclidean distances for all successive pairs of points between i and j' instead of using 'euclidean distance between i and j' directly. This little trick allows us to capture the spatial and temporal correlation among trajectory points in our kernel matrices. To elaborate further: for two points p_i and p_j in a trajectory where $j > i$, the sum of successive euclidean distances is denoted by $\delta_{i,j}$, and is defined as:

$$\delta_{i,j} = \sum_{i \leqslant m < j} |p_m, p_{m+1}|_2$$

The kernel $K_{i,j}$ is simply an RBF kernel with $\delta_{i,j}$ as the core part instead of the direct euclidean distance between i and j.

$$K_G(i,j) = e^{-\delta_{i,j}/\sigma_k} \tag{8}$$

A kernel is a valid kernel if there exists an embedding space for input points where it can be applied as a known kernel. One can show that our spatio-temporal kernel is a valid kernel because if we project a trajectory onto a straight line such that every successive distance is preserved and then take the RBF kernel over the points of this projection, it will be same as the above mentioned kernel. By virtue of being 'valid', the spatio-temporal kernel described can be integrated with other kernels in future to reflect more domain knowledge.

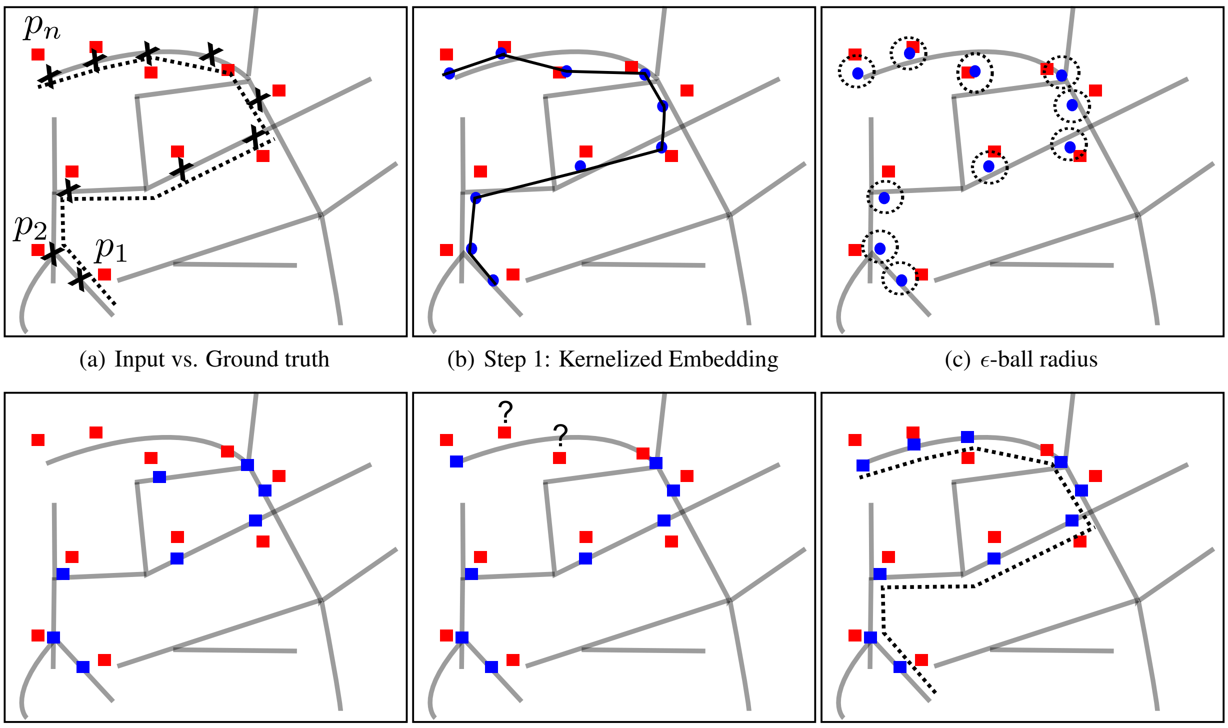
Rounding Step

After the noise is reduced by the optimization, we have to assign the points to street network. The simplest rounding scheme can be a nearest neighbour based one. However, we provide a more sensible scheme, which is based upon the following assumptions:

1. **Assumption 1**: For nearby points, the Euclidean distance between points resembles shortest street network graph distance;
2. **Assumption 2**: Most of the map matching algorithms use a radius ϵ to restrict the assignment possibilities.

For assigning a point Ψ_i to the street network, we pick all edges inside ϵ-neighbourhood and find nearest neighbours of Ψ_i on these edges. The resulting points are our candidates for the solution Φ_i. We denote the set of candidate solutions for Φ_i as C_{Φ_i}. According to our assumption 1, Ψ_i should be assigned to a point $c \in C_{\Phi_i}$ which minimizes the difference between euclidean distance and shortest path distance between the solution Φ_i and Φ_{i-1}. For this purpose, we take an RBF kernel K_Ω between graph distance (denoted by $\Omega(x,y)$) and Euclidean distance (denoted by $d(a,b)$) as following:

$$K_{\Omega_i} = e^{-(d(\Psi_i, \Psi_{i-1}) - \Omega(c \in C_{\Phi_i}, \Phi_{i-1}))^2} \tag{9}$$



(a) Input vs. Ground truth (b) Step 1: Kernelized Embedding (c) ϵ-ball radius

(d) Step 2: Rounding (e) Conflict Resolution (f) KMM Output

Figure 4: Step-by-step illustration of Kernelized Map Matching. (a) Map matching input with ground truth values. Red squares denote the input Trajectory; Gray edges, the street network graph; dashed lines, the ground truth path and crosses the ground truth points. (b) Approximation of ground-truth path by kernelized embedding in a relaxed setting. Blue circles denote the optimization output Ψ. (c) Imposition of $\epsilon-$ball radius to reduce the number of candidate matches for rounding. Dashed-circles denote $\epsilon-$balls. (d) Hard assignment of Ψ to street network graph in the rounding step on the basis of RBF kernels Blue squares denote the assignment. (e) A conflict and its resolution. The graph distance is greater than $\alpha\times$ Euclidean distance, both points are discarded to get a vote from neighbours. (f) The final output points and path computed by KMM. (Best viewed in color)

K_Ω stipulates the assumption 1 for two consecutive points. However to avoid K_Ω output going away from input point, we add a regularizer term which is also an RBF kernel between a Ψ_i and the candidate in question. Now for all elements $c \in C_{\Phi_i}$ we calculate the following term

$$-e^{-(c\in C_{\Phi_i}-\Psi_i)^2} \cdot K_{\Omega_i} \tag{10}$$

The candidate point which gives the minimum value for this term is chosen as solution Φ_i. In most of the cases, the comparison term between candidates, i.e. regularized K_{Ω_i} produces the right result however it is possible that after assignment the graph distance between Φ_i and Φ_{i-1} is far greater than the Euclidean distance between them, which introduces a conflict and violates our observation 1. To check these we take a constant α and multiply euclidean distance by it. After assignment, if the graph distance $\Omega(\Phi_i, \Phi_{i-1})$ is greater than $\alpha \times d(\Psi_i, \Psi_{i-1})$ (α-condition); we employ a resolution scheme, inspired by [Newson and Krumm, 2009]. We discard Φ_i and Φ_{i-1} and consider Φ_{i-2} as our previous point instead of Φ_i. After the assignment Φ_{i+1} , we again see whether the condition 1 is violated or not, if it happens again, we discard Φ_{i+1} and Φ_{i-2} and go ahead in the same fashion. We continue doing so until α-condition is not violated any more after an assignment of a point denoted by $\Phi_{\hat{i}}$. Once we find such a point $\Phi_{\hat{i}}$, we can again resume the standard comparison. However, before resuming the comparison, we assign all the discarded points on the shortest path between $\Phi_{\hat{i}}$ and the corresponding previous point. A conceptual work-flow of KMM steps together with explanations for each step are provided in Fig. 4

Parameter Selection

KMM has four input parameters, namely, σ_n, standard deviation of noise; λ, stiffness parameter for regularization term of F_o; α, the rounding step parameter and k_w, width of sliding window for objective function F_o.We discuss selection process for these parameter as following:
σ_N is required as an input to F_o in the regularization term. Our method for estimating σ_N is carried out on a holdout sample of 20 ground truth points and the approach is same as [Newson and Krumm, 2009]. Alternative strategies include estimators from other map matching solutions or empirical standards for given application. λ is the stiffness parameter of regularization term. A reasonable range for choice of λ is between 0.5 and 1.Clearly, $\lambda > 1$ make the standard deviation of Ψ more than the trajectory while $\lambda < 0.5$ is too restrictive on output. In our experiments, have chosen λ=0.6. k_w is the sliding window width for execution of optimization step. Figure 6 shows the results for $k_w = \{3, 4, .., 6\}$ over a synthetic data set. We have chosen $k_w = 5$ for most of our experiments. According to our observation, $k_w > 6$ proves a bit time consuming during optimization while $k_w < 3$ is not sufficient to capture the local geometry. α- is the parameter 'governing relationship between Graph and Euclidean distance'. We set it to 1.5 which means that Graph distance should not be greater than 1.5$\times$Euclidean distance. A reasonable range is $1 \leq \alpha \leq 2$. σ_{K_G} and $\sigma_{K_{G'}}$ are learned through optimization over a holdout sample of 20 consecutive input.

Algorithm 1: KMM

```
Input  : $G$,$G'$, $K_G$,$K_{G'}$,$K_{GG'}$,$K_\Omega$,$\lambda$,$k_w$, $\alpha$,$\sigma_N$
Output: $\Phi(V)$ - The Output Path $P$
// $G,G'$- Euclidean Graphs
// $K_G,K_{G'},K_{GG'},K_\Omega$ -Graph Kernels
// $\lambda$ -Regularizer
// $k_w$- sliding window width
// $\alpha$- constant for Euclidean versus
   Graph distance validation
for $i \leftarrow 1 : |V| - k_w$ do
    foreach sliding window do
        compute $K_G, K_{G'}$
        $\Psi \leftarrow$ Optimize $F_o$ w.r.t $\Psi$

$i \leftarrow 1$ , $i_{prev} \leftarrow 1$ , $prev_{dist} \leftarrow 0$
while $i < |V|$ do
    $i \leftarrow i + 1$
    $i_{prev} \leftarrow max(1, i_{prev})$
    $min_{obj} \leftarrow \infty$
    if $i_{prev} = i - 1$ then
        $prev_{dist} \leftarrow 0$
        $con_{dist} \leftarrow d(\Psi_i, \Psi_{i-1})$
    else
        $con_{dist} \leftarrow$
        $d(\Psi_i, \Psi_{i-1}) + prev_{dist} + d(\Psi_{i_{prev}}, \Psi_{i_{prev}+1})$
    for $c \in C_{\Phi_i}$ do
        $obj_{val} \leftarrow -e^{-(c-\Psi_i)^2}.K_{\Omega_i}$
        if $obj_{val} < min_{obj}$ then
            $min_{obj} \leftarrow obj_{val}$
            $\Phi_i \leftarrow c$

    if $\Omega(\Phi_i, \Phi_{i_{prev}}) > \alpha \times d(\Psi_i, \Psi_{i_{prev}})$ then
        $i_{prev} \leftarrow i_{prev} - 1$
        $prev_{dist} \leftarrow con_{dist}$
    else
        Assign all points between $i$ and $i_{prev}$ to
        shortest path between $\Phi_i$ and $\Phi_{i_{prev}}$

Output path $P$ by connecting consecutive $\Phi$
```

4 Experimental Evaluation

In this section we report the results from a series of experiments which we conducted in order to empirically investigate the following questions:

(Q1) Can kernel methods be used for map matching?

(Q2) If so, how do they perform compared to state-of-the-art methods?

(Q3) Is kernelized embedding indeed reducing the noise?

(Q4) Does the rounding step in KMM contribute to the error reduction?

To this aim, we implemented KMM in scientific Python running on a standard Intel-Quadcore 2GHz computer.

Overall, we decided for two experimental setups. Our first experimental setup evaluates and compares KMM's accuracy performance on a real-world dataset recently used in [Newson and Krumm, 2009] to evaluate an hidden Markov model based map matching approach and hence addresses **Q1**, **Q2**. To address **Q4** we compare the performance of our rounding step with a randomized rounding step. The second setup investigates **Q3** by comparing the result of KMM's embedding step to baseline "closest point on edge" using synthetically generated dataset.

4.1 Q1+Q2+Q4: Real-World Dataset

In our first experiment, we compared KMM's performance to the performance of Krumm and Newson's recent hidden Markov model based approach [Newson and Krumm, 2009]. To measure performance, we used the Route Mismatch Fraction measure already used by Newson and Krumm. Route Mismatch Fraction(RMF) is the total length of false positive and false negative route divided by length of original route [Newson and Krumm, 2009]:

$$d_+ = \text{length of erroneously added route}$$
$$d_- = \text{length of erroneously subtracted route}$$
$$d_o = \text{length of original route}$$
$$RMF = \frac{(d_+ + d_-)}{d_o}$$

We used Krumm and Newson's dataset. It consists of a 50-mile route in Seattle sampled at 1 Hz, giving one trajectory of 7531 time stamped latitude/longitude pairs along with manually matched ground-truth path. The street network comprises around $150,000$ road segments. [Newson and Krumm, 2009] presented results for different sampling intervals and noise degradations of this data. We take six base cases where HMM model performance is good, i.e 5,10 seconds sampling intervals vs. 30,40,50 meters noise. Because we want to have a statistical comparison (average, significance,etc.) with HMM, we perform 25 experiments for comparison with each base value in the following way. For one sampling interval, say 10, we choose 5 different starting points from the initial 5 points of the trajectory and then sampled at the given rate. Following this procedure, we prepared experimental datasets for each sample by adding 5 instances of noise for one standard deviation (e.g 30), i.e

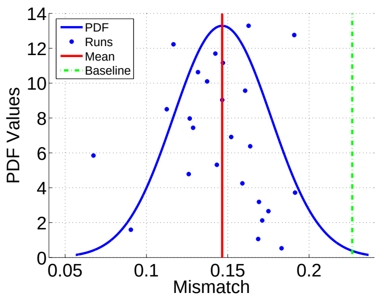
25 datasets/combination = 5 samples $\times$5 noise instances.

Fig. 5 summarizes the experimental results showing the RMF errors. As one can see, in 5 out of 6 cases KMM estimated a lower route mismatch fraction. The statistical significances of the results are shown in Table 1. In 4 out of 5 cases where we are better, the differences in mean values are significant (t-test, $p = 0.05$). Averaged over all experiments, a Wilcoxon test identifies KMM to be significantly better.
To address **Q4**, we compared the performance of our rounding step with a randomized rounding step, i.e random assignment of embedding output Ψ to a street network point in ϵ-ball, over the real-world dataset described above. Table 2 summarizes the results. As one can see our rounding step always performs better, and in most cases outperforms the randomized rounding step by a margin of 4-10 percent in error.

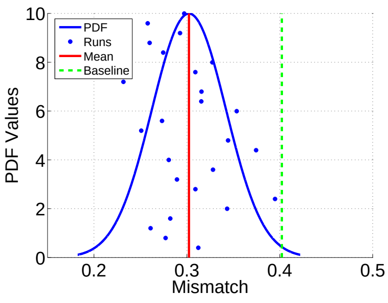
To summarize, the results clearly answer questions **Q1** ,**Q2** and **Q4** affirmatively.

4.2 Q3: Synthetic Datasets

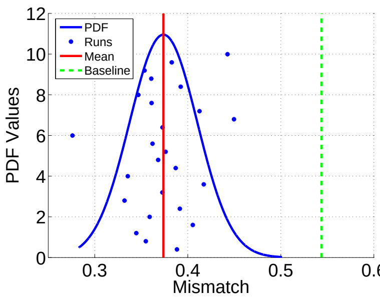
In order to investigate how well the embedding step reduces noise, we generate ground truth points with the help of synthetic data. To generate the data, we used Thomas Brinkhoff's data generator [Brinkhoff, 2000]. It allows to generate trajectories according to some underlying road network for different speed and sampling time variation setting. Additionally, we assumed normal noise on the



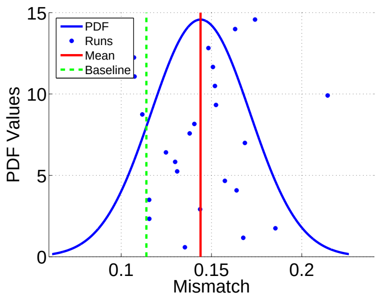
(a) NL 30, SR 5



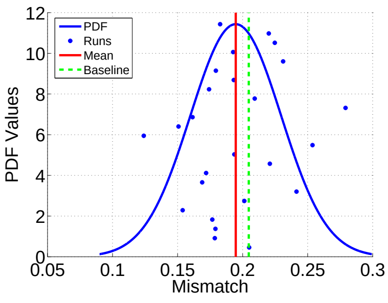
(b) NL 40, SR 5



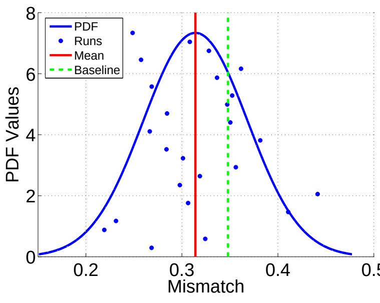
(c) NL 50, SR 5



(d) NL 30, SR 10

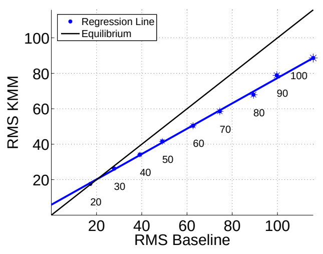


(e) NL 40, SR 10

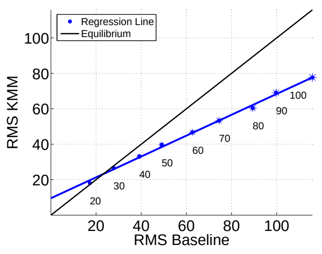


(f) NL 50, SR 10

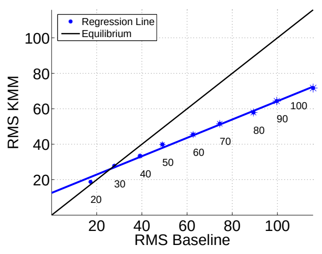
Figure 5: Route Mismatch Fraction of KMM vs. HMM for different noise level (NL, in meters) and sampling rates (SR, in seconds). We ran 150 experiments i.e 25 experiments/per NL-SR setting. One blue dot denotes the route mismatch fraction for an experiment, blue graphs the estimated normal distributions, red lines the means, and dashed green lines the route mismatch fraction for an HMM as reported by Newson and Krumm. As one can see, in 5 out of 6 cases KMM estimates a lower route mismatch fraction. In 4 out of 5 cases, the differences in mean values are significant (t-test, $p = 0.05$). Averaged over all experiments, a Wilcoxon test identifies KMM to be significantly better. (Best viewed in color)



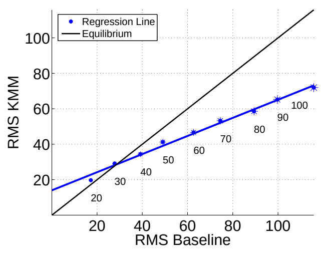
(a) K=3



(b) K=4



(c) K=5



(d) K=6

Figure 6: Scatter plots of the root mean squared(RMS) error of KMM and of the baseline, closest point on edge, for different window sizes (K) on the Oldenburg dataset for different noise levels $(20, 30, 40, \ldots, 100)$ as denoted by the numbers associated with the dots. For better comparison with the equilibrium (solid straight line), we also show the linear regression line of the values. As one can see KMM performs much better with increasing noise levels. The regression lines have consistently a smaller slope than the "equilibrium" line. The turning point is around noise level 25. (Best viewed in color)

generated observations. it is well known navigation systems produce noise that is normally distributed with average of noise (σ_N) ranging from 10-100 meters. For instance, Krumm and Newson [Newson and Krumm, 2009] discussed the different types and amount of noise in real-world movement data and have shown that they can be described well by a Gaussian shape.

Specifically, we generated 100 trajectories of average length 50 from the street network of Oldenburg, Germany. Then, we added noise σ_N for $\sigma_N = 20, 30, \ldots, 100$. This resulted in an overall set of 900 trajectories. As baseline for comparison we used a "project to the closed point in the street network" approach. To get a fair comparison, we also used the "project to the closed point in the street network" as "rounding" method for KMM. We report on the root means squared difference in meters achieved by the two methods.

The results are summarized in Fig. 6. As one can see, in all cases the embedding indeed reduced noise considerably. Moreover, it performs better with increasing noise levels as the regression lines have consistently a smaller slope than the "equilibrium" line. The turning point is around noise level 25. This clearly answers question **Q3** affirmatively. To summarize, the results of our experiments indicate that kernel methods can indeed be used for map matching and achieve results comparable to state-of-the-art techniques.

5 Conclusion

In this paper, we have recognized that kernel methods have an important role to play in map matching. Specifically, we have established — to the best of our knowledge — the first link between map matching and kernel methods and instan-

Statistics for sampling rate=5			
Noise	σ_{error}	student t	Wilcoxon signed rank
30	0.029	1	0.00012
40	0.03992	1	0.0027
50	0.036	1	0.00122
Statistics for sampling rate=10			
Noise	σ_{error}	student t	Wilcoxon signed rank
30	0.027	1	0.0004
40	0.034	0	0.047
50	0.054	1	0.00941

Table 1: Statistics table for comparison with HMM

Noise Ratio	KMM Error	ERR SR=5	ERR SR=10
30	0.1463	0.1783	0.1480
40	0.3022	0.3221	0.2350
50	0.3739	0.4671	0.3979

Table 2: Comparison of average KMM error with average Randomized Rounding (ERR) error over different Noise Ratio and Sampling Rates (SR)

tiated the link by developing an easy-to-implement kernelized map matching (KMM) approach. By accounting for spatial and temporal correlations among the trajectories using kernels, map matching becomes less sensitive to noisy observations. This allowed us to employ a simple rounding scheme to compute the hard assignments of trajectory points to points lying on the network. Our experimental results on both simulated as well as real-world datasets show that kernel methods can indeed be used for map matching and can achieve better performance than state-of-the-art hidden Markov models.

The link established between map matching and kernels provides many interesting avenues for future work; we have only started to explore it. Indeed, one should study alternative kernels and map features, cost functions and hyper-parameter optimization criteria. For instance, so-called Fisher kernels are kernels derived from hidden Markov models. In turn, we may utilize any HMM based map matching approach. Testing KMM within a real-world system tracking system is another interesting avenue. Proving the hardness of map matching problems along with guarantees on approximation are interesting venues of future work. Finally, KMM directly generalize to the case of 3D trajectories.

Acknowledgements

The authors would like to thank Paul Newson and John Krumm for making their data available, to anonymous reviewers for helpful suggestions and Novi Quadrianto for taming the notation and ideas in this paper . This work was supported by the Fraunhofer ATTRACT fellowship STREAM and a "Higher Education Commission" (HEC)-Pakistan scholarship.

References

[Brakatsoulas *et al.*, 2005] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864. VLDB Endowment, 2005.

[Brinkhoff, 2000] Thomas Brinkhoff. Generating network-based moving objects. In *SSDBM*, page 253, Washington, DC, USA, 2000. IEEE Computer Society.

[Cox and Cox, 2000] Trevor F. Cox and M.A.A. Cox. *Multidimensional Scaling, Second Edition*. Chapman and Hall/CRC, 2 edition, 2000.

[Fox *et al.*, 2003] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003.

[Guo *et al.*, 2008] Y. Guo, J. Gao, and P.W. Kwan. Twin kernel embedding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1490–1495, 2008.

[Lee and Verleysen, 2007] John A. Lee and Michel. Verleysen. *Nonlinear dimensionality reduction*. Springer, New York; London, 2007.

[Lou *et al.*, 2009] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *GIS*, pages 352–361, New York, NY, USA, 2009. ACM.

[Mitchell, 2009] Tom Mitchell. Mining our reality. *Science*, 326(5960):1644–1645, 2009.

[Newson and Krumm, 2009] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *GIS*, pages 336–343, New York, NY, USA, 2009. ACM.

[Pemmaraju and Skiena, 2003] Sriram Pemmaraju and Steven Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* . Cambridge University Press, 2003.

[Quadrianto *et al.*, 2009] Novi Quadrianto, Le Song, and Alex J. Smola. Kernelized sorring. In *NIPS 21*, pages 1289–1296. 2009.

[Quddus *et al.*, 2007] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. 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.

[Shawe-Taylor and Cristianini, 2004] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, illustrated edition edition, 2004.

[Shewchuk, 1994] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA, 1994.

[Taylor *et al.*, 2001] George Taylor, Geoffrey Blewitt, Doerte Steup, Simon Corbett, and Adrijana Car. Road reduction filtering for gps-gis navigation. *Transactions in GIS*, 5(3):193–207, 2001.

[Weston and Vapnik, 2002] Chapelle O. Elisseeff A. Scholkopf B. Weston, J. and V. Vapnik. Kernel dependency estimation. *Advances in neural information processing systems*, 2002.