Pathlet Learning for Compressing and Planning Trajectories

Chen Chen
Department of Electrical
Engineering
Stanford University
cchen86@stanford.edu

Hao Su
Department of Computer
Science
Stanford University
haosu@stanford.edu

Qixing Huang
Department of Computer
Science
Stanford University
huangqx@stanford.edu

Lin Zhang
Department of Electrical
Engineering
Tsinghua University
linzhang@tsinghua.edu.cn

Leonidas Guibas
Department of Computer
Science
Stanford University
guibas@cs.stanford.edu

ABSTRACT
The wide deployment of GPS devices has generated gigantic datasets of pedestrian and vehicular trajectories. These datasets offer great opportunities for enhancing our understanding of human mobility patterns, thus benefiting many applications ranging from location-based services (LBS) to transportation system planning. In this work, we introduce the notion of pathlet for the purpose of compressing and planning trajectories. Given a collection of trajectories on a roadmap as input, we seek to compute a compact dictionary of pathlets so that the number of pathlets that are used to represent each trajectory is minimized. We propose an effective approach whose complexity is linear in the number of trajectories. Experimental results show that our approach is able to extract a compact pathlet dictionary such that all trajectories can be represented by the concatenations of a few pathlets from the dictionary. We demonstrate the usefulness of the learned pathlet dictionary in route planning.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous;
D.2.8 [Software Engineering]: Metrics—complexity measures, performance measures

Keywords
GPS trajectories, linear programming, integrality conditions, pathlet learning, pathlet planning

1. INTRODUCTION
The pervasiveness of GPS devices has created many large datasets of pedestrian and vehicle trajectories. Compressing such large data sets is obviously of interest. Furthermore, compression is tightly coupled with shared structure extraction, related to the semantics of trajectories. Such higher-level trajectory understanding can benefit a variety of applications ranging from the study of population migration routes, vehicular traffic patterns, and the state of city road networks. In this paper, we seek to extract latent shared structure among many human trajectories.

Our work is motivated by the seminal work of Gonzalez et al. [5], who discovered that human trajectories show a high degree of spatial and temporal regularity, i.e., human beings have high probability of repeating similar mobility patterns. In this paper, we explore the idea of Pathlet Learning to capture the spatial and temporal regularity in human trajectories. We seek to extract a set of path segments that have semantic meanings, referred to as Pathlet Dictionary. We note that this can also be viewed as a joint trajectory segmentation problem, where the pathlet structure becomes only apparent in the context of a trajectory collection.

We formulate pathlet learning as solving an integer linear program (ILP), whose objective function minimizes the size of the pathlet dictionary as well as the number of pathlets that are used to reconstruct each trajectory. To solve this ILP on large datasets, we introduce a decoupled approach, which optimizes a lower bound of the original objective function. We test our algorithm on a large-scale real world dataset, which contains 230K trajectories of taxi cabs in Beijing. Our algorithm extracts a pathlet dictionary containing around 130K pathlets, which can reconstruct all trajectories in the dataset using ∼ 7 pathlets on average per trajectory. This number is significantly smaller than the average number of edges used to represent trajectories on a road map (which is 60.1), or the average number of “turns” that might be provided in a navigation system to realize the trajectory (which is 36.7). To show the usefulness of the extracted pathlets, we also apply them in route planning in the city of Beijing. Experimental results are competitive against those obtained from Google Maps.

The major contributions of this work are as follows:
• We introduce the notion of Pathlet Dictionary to represent spatial regularities in a trajectory dataset.
• We give an effective algorithm to learn pathlet dictionaries from large collections of trajectories.
• We demonstrate applications exploiting the extracted dictionary.
Related Works. Our work lies in the emerging field of spatial-temporal data mining, whose primary goal is to design algorithms for storing, processing, retrieving, and mining trajectory data and exploring their broad applications. We refer to [14] for a general introduction to this domain.

Our work is also closely related to trajectory compression. Several techniques have been published on this topic, e.g., greedy and shortest-path codes [6] and semantic trajectory compression [10]. These methods typically require additional computations, e.g., shortest path computation, for decoding. In contrast, we are interested in the discovery of high-order shared structures among a collection of trajectories for the purpose of compression, so that for each trajectory, the decoding step is simply combining a few pathlets.

There are also several works on detecting shared structure among collections of trajectories, focusing on other types of shared structure. In [15], the authors propose a “bag of segments” method by chopping trajectories into small pieces, and then aggregating the pieces into clusters. However, this method can only detect segments of fixed length. In contrast, our algorithms automatically extract appropriate pathlets with varying lengths by solving an optimization problem. In [11], the authors address the problem of computing the median trajectory from a set of trajectories. This, however, requires the input trajectories to have similar starting and ending points. Recently, [7] proposes to decompose a given trajectory set using segments of meaningful geometric shapes. In contrast to these techniques, we focus on a different objective, i.e., to minimize the size of the shared pathlet dictionary.

The idea of path planning by stitching path segments is also related to a couple of path planning methods that utilize information extracted from historical data [13, 3, 12]. In particular, in the seminal work of T-drive [13], the authors propose to compute a landmark graph using historical trajectory data. The path planning is carried out in two phases. The first phase computes a coarse trajectory on the landmark graph. This coarse trajectory is then refined in the second phase into a trajectory on the original map. The difference between our work and T-drive is that our pathlets encode higher-order information, beyond the first-order information available in a graph representation.

2. PROBLEM STATEMENT

In this section, we formally define the problem of pathlet learning from a collection of trajectories.

**Terminology.** We represent the trajectory $t$ of a moving object as a path on a underlying roadmap $G = (V, E)$ (i.e., a sequence of connected map edges on the road map). We refer to a set of trajectories as $T$. Such a trajectory set can be obtained by map-matching a set of GPS traces to the underlying road map. We call a path $p$ on $G$ a pathlet if it is a sub-path of one or many trajectories in $T$. We denote the set of all possible sub-paths of a trajectory $t$ as $P(t)$. It is easy to see that if $t$ contains $|t|$ edges, then $|P(t)| = \frac{|t|(|t| + 1)}{2}$.

A Pathlet Dictionary is a set of pathlets ($P$), which can reconstruct trajectories by concatenations of pathlets in the dictionary. The size of a pathlet dictionary, denoted by $|P|$, is defined as the number of its pathlets. Obviously, the largest pathlet dictionary one can have is $P = \cup_{t \in T} P(t)$.

**Definition 1.** We say a trajectory $t$ can be reconstructed by a pathlet set $P = \{p_1, \ldots, p_m\}$ if $t = \cup_{p \in P} p$, if there exists a permutation $\sigma$ such that $t = p_{\sigma(1)} p_{\sigma(2)} \cdots p_{\sigma(m)}$.

Note that given a trajectory $t$ and a pathlet dictionary $P$, there may exist multiple ways to reconstruct $t$ using subsets of $P$. In this paper, we are interested in the subset(s) of smallest size.

**Definition 2.** The description length $dl(t, P)$ of a trajectory $t$ with respect to a pathlet dictionary $P$ is given by

$$dl(t, P) = \min_{P \subseteq P} \sum_{p \in P} |p_{\sigma} \subseteq P, t = \cup_{\sigma}'(P)| + \sum_{p \in P, t = \cup_{\sigma}(P)} \infty$$

otherwise.

1

Pathlet extraction via constrained optimization. For pathlet learning, we would like to minimize the size of the extracted pathlet dictionary as well as the number of pathlets that are used to represent each trajectory. Formally speaking, given the input trajectory set $T$ and the set of all possible pathlets $P$, we define pathlet learning as solving the following optimization problem:

$$\min_{P \subseteq P} \sum_{\sigma \in T} dl(t, P) + \lambda \sum_{\sigma \in T} \sum_{t \in T} x_{t, P}$$

s.t. $x_{t, p} \leq 1$ for all $t \in T$, $\forall p \in P(t) \subseteq P$, $x_{t, p} = 1$, $x_{t, p} = 0$ otherwise.

3. ALGORITHM

We first convert the pathlet learning described in Problem 2 to an equivalent integer programming with constraints. We then introduce a modified formulation, which can be solved efficiently using dynamic programming.

**Integer programming formulation.** We associate each pathlet $p \in P$ with a binary indicator $x_{t, p} \in \{0, 1\}$, where $x_{t, p} = 1$ if $p \in P(t)$, and $x_{t, p} = 0$ otherwise. Thus, the extracted pathlet dictionary $P$ will be $\{p | p \in P, x_{t, p} = 1\}$.

To formulate the reconstruction constraint, i.e., a trajectory can be reconstructed by a concatenation of pathlets in $P(t)$, we associate each pathlet $p \in P(t)$ with a binary indicator $x_{t, p} \in \{0, 1\}$, where $x_{t, p} = 1$ if pathlet $p$ is used to reconstruct trajectory $t$ and $x_{t, p} = 0$ otherwise.

With the binary indicators $(x_{t, p})$, introduced, the constrained optimization for pathlet learning (2) can be transformed into the following integer program:

$$\min \sum_{p \in P} x_p + \lambda \sum_{t \in T} \sum_{p \in P(t)} x_{t, p}$$

s.t. $x_{t, p} \leq 1$, $\forall p \in P(t), t \in T$

$$\sum_{p \in P(t)} x_{t, p} = 1$, $\forall t \in T$

$$x_{t, p} \in \{0, 1\}$, $\forall p \in P(t), t \in T$.

(3)

Approximate solution. A standard way of solving Equation (3) is to optimize its linear programming relaxation. However, we found that this method is inappropriate for large-scale datasets as the complexity of solving a linear program does not scale linearly in terms of the number of variables [4]. To address this issue, we develop a scalable method, where the basic idea is to solve a modified objective
function which can be decoupled into independent objective terms for each trajectory.

More precisely, we consider a lower bound on the original objective function, which is derived as follows:

\[
\sum_{p \in \mathcal{T}} x_p + \lambda \sum_{t \in \mathcal{T}, p \in \mathcal{T}(t)} x_{t,p} \\
= \sum_{p \in \mathcal{T}} \left( \sum_{t \in \mathcal{T}(p)} \frac{x_p}{|T(p)|} \right) + \lambda \sum_{t \in \mathcal{T}, p \in \mathcal{T}(t)} x_{t,p} \\
\geq \sum_{p \in \mathcal{T}} \left( \sum_{t \in \mathcal{T}(p)} x_{t,p} \right) + \lambda \sum_{t \in \mathcal{T}, p \in \mathcal{T}(t)} x_{t,p} \\
= \sum_{t \in \mathcal{T}} \left( \sum_{p \in \mathcal{T}(t)} \left( 1 + \frac{1}{|T(p)|} \right) x_{t,p} \right) = f, \tag{4}
\]

where \( T(p) = \{ t| t \in \mathcal{T}, p \in \mathcal{T}(t) \} \).

Substituting Equation 4 into Problem 3, we obtain a decoupled optimization problem for each trajectory \( t \in \mathcal{T} \):

\[
\min_{x_{t,p} \in \{0, 1\}} \left( \lambda + \frac{1}{|T(p)|} \right) x_{t,p} \\
\text{s.t.} \quad \sum_{p \in \mathcal{T}(t)} x_{t,p} = 1, \quad \forall t \in \mathcal{T}. \tag{5}
\]

Note that after optimizing \( x_{t,p} \) for all input trajectories, the optimal value of \( x_p = \max_{t \in \mathcal{T}(p)} x_{t,p} \).

It turns out Problem 5 can be solved exactly using dynamic programming. Specifically, denoting the optimal objective value in Problem 5 for a sub-trajectory \( v_i v_j \) of \( t = v_i \ldots v_o \), we can compute \( f^*(v_i v_j) \) recursively as follows:

\[
f^*(v_i v_j) = \min_{k \in \{i+1, \ldots, j-1\}} \left( f^*(v_i v_k) + f^*(v_k v_j) \right) \quad i < j - 1 \\\n\lambda + 1/|T(v_i v_{j+1})| \quad \text{otherwise.} \tag{6}
\]

After obtaining the optimal value for each sub-trajectory, the optimal decomposition of \( t \) can then be obtained via back-tracking.

4. RESULTS

Our raw trajectory data is a collection of \( \sim 230K \) GPS traces of taxi cabs in Beijing, China over a month’s period (05/01-05/31/2009) [2]. Our road map, \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \), comes from Beijing OpenStreetMap [1]. We project our raw GPS traces on the road map using the map matching algorithm described in [9, 8]. The Beijing road map contains 26K nodes and 56K edges. After map matching, each projected trajectory contains an average of 60 map edges.

We randomly sample 30% of the trajectories as our evaluation dataset, and use the remaining 70% as training dataset. We choose the default \( \lambda = 1.0 \). To guarantee the reconstruction of each trajectory, we also include all map edges in the pathlet dictionary. The resulting pathlet dictionary has around 130K pathlets. We evaluate the quality of a resulting pathlet dictionary from the following aspects:

- **Size** of the pathlet dictionary, i.e., the number of pathlets in the pathlet dictionary. This characterizes the compactness of the pathlet dictionary.
- **Distribution** of pathlet length, where the length of a pathlet is defined as the number of its edges on the road map. Longer pathlets contain higher ordered information, which indicate movement patterns across multiple edges on the road map.

**Description length**, i.e., average number of pathlets to reconstruct each trajectory, which measures the efficiency of pathlet dictionary in explaining trajectories.

**Overall Pathlet Distribution.** Figure 1(a) shows the distribution of pathlet length for different \( \lambda \). As \( \lambda \) increases, the term \( \lambda \sum_{t \in \mathcal{T}, p \in \mathcal{T}(t)} x_{t,p} \) becomes dominant in the objective function. Hence, increasing \( \lambda \) favors long pathlets.

**Description Length and Pathlet Dictionary Size.** Figure 1(b) shows the effect of \( \lambda \) on the average #pathlet per trajectory and the pathlet dictionary size. It is clear that large values of \( \lambda \) lead to large pathlet dictionaries. This is expected from the definition of our objective function.

![Figure 1](image1.png)

**Description Length and Pathlet Dictionary Size.** Figure 1(b) shows the effect of \( \lambda \) on the average #pathlet per trajectory and the pathlet dictionary size. It is clear that large values of \( \lambda \) lead to large pathlet dictionaries. This is expected from the definition of our objective function.

**Figure 2: The percentage change of reconstructable trajectories in the evaluation dataset when using part of the pathlet dictionary.**

**Partial Reconstruction.** In previous results, the pathlet dictionary is a complete dictionary that can reconstruct all trajectories in the evaluation dataset. However, if we tolerate a small portion of trajectories that are not reconstructed, the size of the pathlet dictionary can be reduced substantially. As shown in figure 2, most evaluation trajectories (~80%) can be reconstructed using just the top 40% of the pathlet dictionary.

**Pathlets have semantic meanings.** We rank each pathlet by the number of trajectories that use it to reconstruct themselves in the training dataset. Figure 3(c) shows an overview of the top 1000 pathlets. In figure 3(a) and (b), each pathlet is represented by a sequence of map edges which has semantic meaning that reveals a common mobility pattern shared by multiple trajectories. For example, figure 3(a) is a pathlet corresponding to entering a highway (“2nd Ring Rd”) from a popular street (“W Chang’an Ave”).

5. APPLICATION IN ROUTE PLANNING
In this section, we show the usefulness of our pathlet dictionary in route planning. As the extracted pathlet dictionary contains high-order information on trajectory portions, it makes sense to augment the underlying roadmap by adding pathlets as additional “edges”. Therefore, the shortest path computed on the augmented graph naturally incorporates high-order mobility patterns, and hence may have high potential of generating desirable results.

More precisely, for each pathlet \( p = v_1 \cdots v_{|p|} \), we add a directed edge \( e_p = v_1 \cdots v_{|p|} \) to \( G \). We define the edge weight \( w_{ep} \) so that planned trajectories are prioritized to follow pathlets: \( w_{ep} = L_p \), where \( L_p \) denotes the sum of edge curve lengths (in kilometers) of pathlet \( p \), and parameter \( \alpha \in [0, 1] \) controls the influence of the pathlets on the resulting trajectories. In particular, when \( \alpha = 1 \), i.e., the weight of each pathlet is the same as its length, our approach is equivalent to graph shortest path.

Figure 3 shows two examples of route planning. When \( \alpha \leq 0.5 \), the planned trajectories are consistent with the top suggestions by Google Maps.

Figure 4 shows two examples of route planning using pathlets on the augmented graph by pathlets.

6. CONCLUSIONS

In this paper, we have defined the problem of pathlet learning from a collection of trajectories and gave an algorithm for its solution. Our algorithm minimizes the size of the pathlet dictionary as well as the number of pathlets that are used to represent each trajectory.

We test our algorithm on a large-scale dataset from the city of Beijing, which consists of a roadmap with 20K nodes and 46K edges and 230K trajectories with an average of 60 edges per trajectory. A pathlet dictionary of less than 150K pathlets is extracted using our algorithm. Such a pathlet dictionary can compress all trajectories in the dataset using 7 pathlets on the average. This number is significantly smaller the averaged number of edges (i.e., \( \sim 60 \)) as well as the number of road turns (which is 36.7) that each trajectory traverses on the road map. We have also demonstrated the usefulness of the pathlet dictionary in route planning.

Acknowledgments

The authors acknowledge the support of ARO grant W911NF-07-2-0027, as well as NSF grants CCF-1011228 and CCF-1161480, and a Google Research award.

7. REFERENCES