2018 19th IEEE International Conference on Mobile Data Management

Corridor Learning Using Individual Trajectories

Nikolaos Zygouras
National and Kapodistrian University of Athens
nzygouras@di.uoa.gr

Dimitrios Gunopulos
National and Kapodistrian University of Athens
dg@di.uoa.gr

***Abstract*—The rapid development and commercialization of location acquisition technologies generates large trajectory datasets, that trace moving objects' trips. In this work, we propose a new trajectory mining algorithm, for discovering paths that are frequently followed by the given trajectories, named as corridors. We claim that the moving objects follow common paths-*corridors*. Detecting corridors from a collection of trajectories is extremely challenging due to the nature of the data (low sampling rates, different speeds, noisy measurements etc.). In this work we propose and evaluate a pipelined algorithm that abstracts from trajectories their underlying frequent paths.**

I. INTRODUCTION

The rapid growth of mobile devices that trace the moving objects' locations results into the generation of voluminous trajectories datasets. These datasets contain information regarding the trips that the moving objects followed in a given period of time. It is critical to develop novel algorithms that detect frequent patterns in objects' movements. The knowledge of such patterns would be a valuable asset helping to cope with the complex nature of the data and would aid in organizing and understanding the objects' movement.

In this paper, we propose an approach towards detecting frequently followed paths from a given trajectory database. We refer to such frequently followed paths as "corridors". A corridor can be thought as a route that is frequently traversed by a considerable number of moving objects. This work builds on an earlier paper [1], which focused mainly on the formulation of the problem and the presentation of the approach. Consider for instance the 5 trajectories of Fig. 1, even if they have different origins and destinations they contain subtrajectories that share common movement at the two marked areas. It is clear that it is not possible to detect such patterns if the trajectories are considered as a whole. Our proposed framework detects frequent corridors that can be used to describe how objects moved in these areas.

The **contributions** of this work are summarized below: (i) Present an efficient algorithm that summarizes vast trajectory databases detecting few corridors that represent the major movement patterns. (ii) Evaluate extensively our proposed algorithm, using synthetic benchmarks, hurricane track data and two real traffic datasets from the cities of Dublin and Porto.

II. BACKGROUND AND RELATED WORK

Trajectory Pattern Mining. Research work for discovering frequent paths in trajectory databases can be distinguished in two main categories. The first assumes that the objects are moving in a known, well structured network [2]–[4]. While the second assumes that such information does not exist or is not provided and the objects are moving in the *unconstrained space* [5]–[15]. For a more detailed description of the related work the reader may refer to [1].

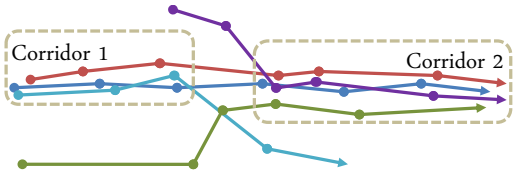

Figure 1: Example of 5 trajectories, that share two common paths-corridors.

Latent Dirichlet Allocation. (LDA) [16] is a topic modelling technique that identifies underlying topics from a collection of documents. It is used to provide interpretation why some parts of the observed data are similar. LDA learns the probabilistic distributions of hidden variables, that are introduced in order to discover patterns in the dataset. It has been extensively applied in Natural Language Processing. Beyond text it is successfully applied in a broad range of fields including image [17], music [18] and map inference [19] domains.

III. PROBLEM DEFINITION

In this work, we propose an efficient algorithm for detecting the main patterns of movement, following the definitions of [1]. The proposed framework receives as input a *trajectory database* $\mathcal{D} = \{T_1, T_2, \ldots, T_N\}$ that contains N trajectories and detects a set of *corridors* $\mathcal{C}$, that are frequently followed by the moving objects in $\mathcal{D}$. A *Trajectory* $T_i : p_1 p_2 \ldots p_{M_i}$, is a time ordered sequence of M_i points of the i^{th} moving object. Originally the trajectories' points lie on a two-dimensional Cartesian plane, $p \in \mathbb{R}^2$.

Definition 1: (*Corridor*): A *corridor* $c \in \mathcal{C}$ is an *induced* trajectory that connects two coordinates through a *dense* path that was *frequently followed* by a considerable number of moving objects in $\mathcal{D}$.

Given a trajectory database $\mathcal{D}$, a set of corridors $\mathcal{C}$ and a distance threshold θ_d a compressed dataset $\mathcal{D}|\mathcal{C}$ is constructed replacing with a pointer to the corridor $c \in \mathcal{C}$ the parts of the trajectories $T \in \mathcal{D}$ that c *covers*. A corridor c is said to *cover* a part or all the trajectory T if its distance with any possible subtrajectory of T does not exceed a threshold θ_d:

2375-0324/18/$31.00 ©2018 IEEE
DOI 10.1109/MDM.2018.00032

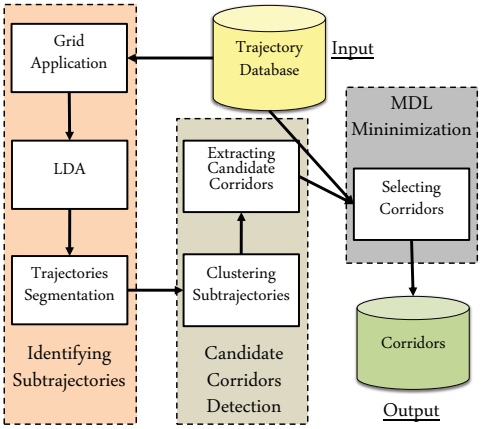

Figure 2: The architecture of the proposed framework.

$min(d(c,s)) \leq \theta_d \forall\, s \in subtrajectories(T)$. The distance $d(\cdot,\cdot)$ could be any distance measurement that measures the distance between two trajectories. In this work, we selected to use the Dynamic Time Warping (DTW) distance [20]. Our formal problem definition is given below:

> *Given a set of sparse trajectories $\mathcal{D}$ moving in the unconstrained space, assuming no time find the set of corridors $\mathcal{C}$ that minimizes the sum of $\mathcal{L}(\mathcal{C})$ and $\mathcal{L}(\mathcal{D}|\mathcal{C})$; $\mathcal{L}(\cdot)$ is the size of a data collection in bits.*

Our definition adopts the Minimum Description Length (MDL) principle [21] viewing learning as data compression. MDL states that the best hypothesis for the dataset is the one that compresses it the most. In our setting the hypothesis corresponds to the seeking corridors $\mathcal{C}$, while the dataset is the collection of trajectories $\mathcal{D}$.

IV. Learning Corridors

The overview of our framework for detecting corridors from a given trajectory database is illustrated in Fig. 2, satisfying the stated above Problem Definition. Our architecture is decomposed into the following three processing modules:

Identifying Subtrajectories: The first step towards detecting a set of corridors is to divide the raw trajectories into subtrajectories, since it is unattainable to detect frequent paths if the trajectories are considered as a whole. We first abstract the trajectories using a grid (Section IV-A1). Then we decompose the space into different sets of grid cells frequently observed together in the objects movements (Section IV-A2). Call these sets of grid cells *frequent sets of locations* [22]. Finally, we segment trajectories into subtrajectories considering their intersections with these frequent sets of locations (Section IV-A3).

Candidate Corridors Detection: Our next step is to extract for each set of similar subtrajectories a corridor that underlies them. In Section IV-B1, we describe how similar subtrajectories are grouped together. In Section IV-B2, we describe how a set of candidate corridors is induced.

MDL Minimization: The final module filters the candidate corridors selecting only those that compress the original trajectory database the most (Section IV-C). In this way, the set of returned corridors complies with the MDL principle.

A. Identifying Subtrajectories

In this Section, we propose a sequence of steps that identifies efficiently subtrajectories from a given trajectory database.

1) Partition Space into Grids: The sparsity of trajectories' GPS samples hinders the detection of interesting movement patterns. There are two main practices followed to cope with this issue. The first fills in the sparse trajectories transforming them into dense ones, using map-matching techniques. The second approach abstracts the trajectories applying a grid that maps the GPS coordinates onto the cells of the grid. In this work, we apply a *grid* of uniformly sized cells, since maps may not exist or may not be available. The collection of trajectories $\mathcal{D}$ is transformed into a new collection of trajectories $\mathcal{D}'$, where each trajectory is represented as a sequence of grid cells $T_i' : g_1 g_2 \dots g_{M_i'}$. If two or more consecutive coordinates are mapped into the same grid cell then we report only the first instance, not allowing T_i' to have consecutive points of the same grid cell, meaning that $M_i' \leq M_i$ and that $g_k \neq g_{k+1} \forall k \in \{1,\dots,M_{i-1}'\}$. Finally, the set of all accessed grid cells is denoted as G_{cells}, $g \in G_{cells}$. The size of the grid cells is dataset dependent.

2) Frequent Sets Discovery: In order to detect the frequent sets of locations we apply the LDA model over the bag of cells for each trajectory. Under the LDA model the cells that a trajectory T_i' accessed are assumed to be generated by a Dirichlet distribution θ_i over the frequent sets and each frequent set $k \in \{1,\dots K\}$ has a Dirichlet distribution ϕ_k over the cells. The grid cells F_k that are associated with the k^{th} frequent set, are those grid cells that have non negative associative probability: $F_k = \{g : p(g_{cell} = g|\phi_k) > 0, g \in G_{cells}\}$.

3) Trajectories Partitioning: In the absence of underlying information regarding the objects' movement at the area associated with each *frequent set* we extract here a set of subtrajectories that will be aggregated later in order to detect the corridors $\mathcal{C}$. Firstly, we invoke the previously created LDA model detecting the frequent sets TFS_i that are related to the i^{th} trajectory T_i': $TFS_i = \{k : p(z = k|\theta_i) > 0, k \in \{1\dots K\}\}$. This avoids looking for intersecting cells with frequent sets that cover areas not visited by the trajectory.

Our method extracts a set of subtrajectories S_k for each frequent set. Initially, it detects the intersecting cells between each trajectory T_i' and the *frequent sets* TFS_i that T_i' is associated with. If the cells of T_i' intersect with those of the k^{th} frequent set, then a subtrajectory is generated and added in S_k. The subtrajectory contains the cells between the first and the last cells of T_i' that intersect with those of F_k. If T_i' contains more than θ_{gap} consecutive cells not matched with the cells of the frequent set, then the trajectory is split further into 2 subtrajectories. We introduce θ_{gap} in order to consider the subtrajectories that do not match perfectly with the cells of the *frequent set* and avoid creating subtrajectories that contain many grid cells not matched with the cells of the *frequent set*.

B. Candidate Corridors Detection

In this Section, we describe how a set of candidate corridors is detected grouping together similar subtrajectories.

1) Grouping Subtrajectories: Here we describe how different subtrajectories are grouped together based on their similar movement at the area that is related to each frequent set. We applied the hierarchical clustering algorithm [23] to the subtrajectories of each frequent set, following a bottom-up approach. The algorithm stops merging clusters when the intra-cluster distances exceed a threshold θ_{cl}. We use DTW in order to measure the distance between two subtrajectories, measuring the distance between two cells with the Manhatan distance. Using DTW we are able to distinguish different directions of movement and assign lower distances to trajectories with similar shape and ordering of locations.

2) Corridors Induction: Given a cluster of similar subtrajectories our objective is to induce a set of candidate corridors that cover them. A directed edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed for *each* cluster of subtrajectories SC. The set of vertices $\mathcal{V}$ corresponds to the grid cells that subtrajectories accessed and the set of edges $\mathcal{E}$ connects the adjacent grid cells. A weight w is assigned to each edge $e = (v_1, v_2)$, depicting the likelihood of moving from v_1 to v_2. The weight w considers a frequency weight w_f and a direction weight w_d (line 15), as it is described below. Finally, the most likely sequence of adjacent grid cells followed by the moving object between two consecutive and not adjacent grid cells is computed posing a shortest path query over $\mathcal{G}$.

The algorithmic form for creating $\mathcal{G}$ is listed in Algorithm 1. Initially, $\mathcal{G}$ is defined as an empty graph (line 1). Then a supplementary graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is constructed, summarizing the cluster's subtrajectories. The edge weight $w'(e)$ of an edge $e = (v_1, v_2)$ of $\mathcal{G}'$ corresponds to the number of subtrajectories that moved from grid cell v_1 to v_2. Then our technique estimates how likely is a transition from node $v \in \mathcal{V}'$ to its neighboring cells, allowing vertical and horizontal movements $\mathcal{M}$ (line 5).

The frequency weight w_f provides higher weight to the most frequently visited adjacent cells. For each allowable movement $m \in \mathcal{M}$ from v we calculate in line 8 the sum of edges' weights that start or end at v_m, using $\mathcal{G}'$ and satisfying Eq. 1; $I(a, b)$ is an indicator function that outputs 1 if $a = b$ and 0 otherwise. Finally the weight for each possible movement m is normalized considering the frequencies of all the adjacent cells of v (line 15).

$$f(\mathcal{G}, v_*) = \sum_{(v_1, v_2) \in \mathcal{G}.\mathcal{E}} w_e * (I(v_1, v_*) + I(v_*, v_2)) \quad (1)$$

The direction weight w_d assigns greater weight to the adjacent cells of v with similar direction with the direction of the subtrajectories of the current SC, when they departed from cell v. Each subtrajectory that traversed v votes for the allowable next movements. The weight for each movement is set initially to 1. Then each successor node v_{succ} of v, using $\mathcal{G}'$, votes updating w_d for each possible movement. The angle of movement θ_{succ} for each successor v_{succ} is estimated using $atan2$ function (line 11). Following that we calculate how likely is a movement m from v to v_m according to v_{succ}, denoted as d. Finally, the new edges from v to its neighboring cells v_m with their corresponding weights are inserted in $\mathcal{G}$ (lines 16 and 17).

Algorithm 1: Creating the graph $\mathcal{G}$, that is used to resolve uncertainties in objects' movement between the nonadjacent points of a cluster's subtrajectories.

Data: SC
Result: $\mathcal{G}$
1 $\mathcal{G} \leftarrow Graph()$;
2 $\mathcal{G}' \leftarrow buildSupplementaryGraph(SC)$;
3 **foreach** $v \in \mathcal{V}'$ **do**
4 $\mathcal{G}.addNode(v)$;
5 $\mathcal{M} \leftarrow allowableMovements(v)$;
6 $w_f \leftarrow dict()$ $w_d \leftarrow dict()$;
7 **foreach** $m \in \mathcal{M}$ **do**
8 $w_f[m] \leftarrow f(\mathcal{G}', v_m)$;
9 $w_d[m] \leftarrow 1$;
10 **foreach** $v_{succ} \in \mathcal{G}'.successors(v)$ **do**
11 $\theta_{succ} \leftarrow atan2(v_{succ}.y - v.y, v_{succ}.x - v.x)$;
12 $d \leftarrow \frac{cos(|\theta_{succ} - \theta_m|)+1}{2}$;
13 $w_d[m] \leftarrow w_d[m] * d^{w'(v, v_{succ})}$;
14 **foreach** $m \in \mathcal{M}$ **do**
15 $weight \leftarrow w_d[m] * \frac{w_f[m]}{\sum_{m_* \in \mathcal{M}} w_f[m_*]}$;
16 $\mathcal{G}.addNode(v_m)$;
17 $\mathcal{G}.addEdge(v, v_m, weight)$;

Finally, a set of candidate corridors for each cluster of subtrajectories is extracted using $\mathcal{G}$. The cluster's subtrajectories are filled in with the missing grid cells, if two consecutive cells are not adjacent. The missing cells are derived, posing shortest path queries over $\mathcal{G}$. In this way, sparse trajectories are transformed into dense sequences of adjacent grid cells. Each detailed path is inserted into the set of candidate corridors $\mathcal{C}_{cand}$ if the minimum DTW distance from any of the already inserted paths in $\mathcal{C}_{cand}$ does not exceed the distance threshold θ_d. The candidate corridors will be evaluated in the next and final step, where only the most dominant corridors will remain.

C. MDL Minimization

Finally a greedy algorithm selects the set of corridors $\mathcal{C}$ from the set of candidate corridors $\mathcal{C}_{cand}$, $\mathcal{C} \subseteq \mathcal{C}_{cand}$, minimizing the MDL principle.

V. EVALUATION

We conduct extensive experimental studies in order to evaluate the performance of the proposed method, denoted as *COR*. In our experiments, we use real GPS trajectories collected from (i) taxis running in the city of Porto [24], (ii) buses moving in Dublin [25] and (iii) Atlantic hurricanes from 1950 till 2004 [26] and ten synthetic benchmarks. Our

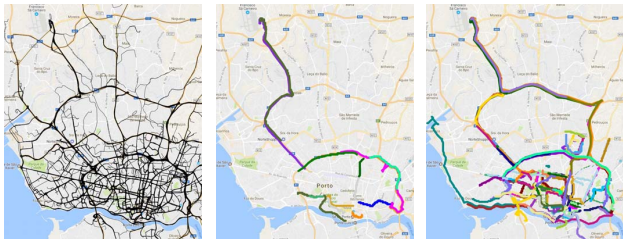

Figure 3: The set of taxis' trips at the city of Porto and the extracted most frequent corridors.

approach is compared with the technique proposed in [7] for detecting frequently visited corridors, denoted as *ZLY10*.

Real Datasets: From the two urban datasets we extracted single trips. Trips from the Porto dataset referred to taxis' paths between passengers' pick-up and drop-off, while for the Dublin dataset referred to bus routes from origin to destination. The hurricane dataset contains 794 trajectories; for the urban datasets we tested our method for different number of trajectories, varying from 1000 till 100000.

Synthetic Datasets: We used ten different synthetically generated benchmarks in our experiments. Each one of them contained different number of frequent patterns FP and noisy patterns NP. Each frequent pattern is followed by 15 trajectories, while each noisy pattern is followed by a single trajectory. The ten benchmarks were generated combining 5, 10, 20, 40 or 80 frequent patterns with 0 or 200 noisy patterns.

Setting the parameters: Below we present the default parameters' values for the conducted experiments. The training set $\mathcal{D}$ contained 25000 trajectories (N) for the two urban datasets, LDA detected 100 and 20 frequent sets (K) for the two urban and the hurricane datasets respectively and the clustering threshold (θ_{Cl}) was 20 grid cells. The experiments described bellow were conducted with these values unless it is stated otherwise. We found that the 50x50m grid cells is a good generalization for an urban context and also speeds up the processing as less processing is required comparing with more fine grained grid cells. For the hurricane dataset the edge length of the grid cells was chosen to be $200km$, as the distance between consecutive GPS samples is much larger than that of the urban datasets. Finally, the maximum distance between a trajectory and a corridor θ_d was 2 grid cells and θ_{gap} was selected to be 5 not matched neighboring cells between the trajectory and the cells of a frequent set.

Evaluation Metrics: In order to measure the quality of the proposed techniques we evaluate the following: (i) $MDL\ \ score\ =\ \frac{\mathcal{L}(\mathcal{C})+\mathcal{L}(\mathcal{D}|\mathcal{C})}{\mathcal{L}(\mathcal{D})}$: referring to the achieved compression. Our objective is to find a set of corridors $\mathcal{C}$ that minimizes the $MDL\ \ score$. (ii) $Coverage\,Length$: represents the percentage of points of the synthetic frequent patterns covered by the detected corridors. This metric is applied only in the synthetic datasets, where the patterns are known in advance. (iii) $Runtime$: the total time required to extract the corridors. (iv) $Aggr.\ Runtime$: the sum of runtimes of all the 8 threads. (v) *Corridors Visualization*: a corridor is plotted connecting the points located at the centers of its grid cells. This creates the yellow step lines of Figs. 3, 4, 5 and 6. The overlapping illustrated corridors (Figure 3), refer either to different directions of movement or to corridors with different origins and destinations that contain overlapping parts. Finally, the X most frequent corridors are the X most frequently pointed out by the reconstructed dataset $\mathcal{D}|\mathcal{C}$. Our experiments were conducted on an 8 core machine clocked at 4GHz, equipped with 16 Gbyte of RAM.

A. Qualitative Analysis

1) Data Summarization: The left part of Fig. 3 illustrates 5000 taxis' trips in the city of Porto. It is particularly difficult for a human to handle, interpret or extract meaningful information just by visualizing the raw trajectories. Our approach summarizes the provided trajectory dataset, detecting the frequently followed corridors. The central and the right image of Fig. 3 illustrate the output of our approach abstracting the 20 and 200 most frequent corridors of taxis' movements. The detected corridors capture frequent routes in the city centre as long as in peripheral roads, that taxi drivers widely use to approach the city centre from the suburbs. A meaningful corridor connects the city with the airport of Porto (located at the North-West part of the city).

The 20 most frequent corridors from the hurricane dataset are illustrated in Fig. 4. This dataset has been also used in [5, p. 601] [10, p. 137]. As we can see, we can detect patterns where the direction of the hurricane changes, similarly to [10], indicating the ability of our approach to detect meaningful patterns in completely unstructured objects movements.

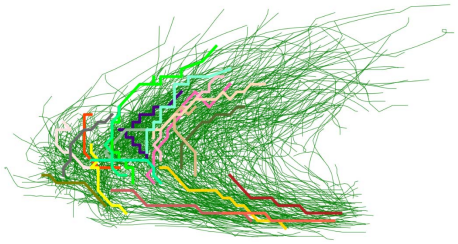

Figure 4: The 20 most frequent corridors of hurricanes.

2) Detecting Common Behavior: The movements of several taxis at the city of Porto are illustrated in the left part of Fig. 5. Although taxis have different origins and destinations they share common movement, captured by the detected corridor. The corridor contains parts of different roads and is of arbitrary shape. A similar observation is derived from the hurricanes' movements in the right part of Fig. 5. A set of 26 hurricanes and their shared corridor from east to west are illustrated. After exiting that corridor each hurricane follows different path.

3) Reconstructing Trajectories: the left part of Fig. 6 shows how a trajectory is reconstructed using a subset of the found corridors $\mathcal{C}$ that *cover* parts of its trip. The original trajectory (magenta) contained 44 grid cells. After the reconstruction only 2 cells remained and 10 pointers to corridors (black)

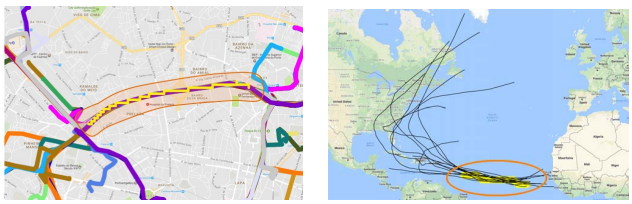

Figure 5: Taxis moving in a peripheral road of Porto (left), 26 hurricanes and their corridors (circled yellow routes).

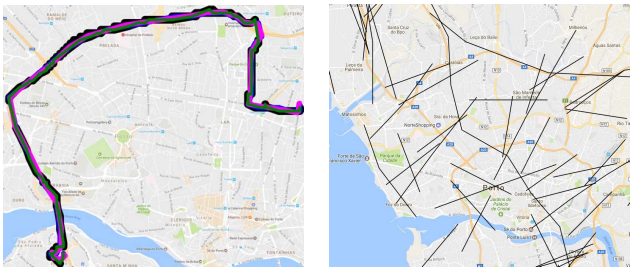

Figure 6: Example of a reconstructed trajectory (left). Clusters' representatives, for the Porto dataset, detected by *TRACLUS* [5] (right).

were used. The reconstructed trajectory (green) overlaps with the original.

4) TRACLUS: the right part of Fig. 6 illustrates the output of *TRACLUS* for the Porto dataset. As we can see, clustering approaches like TRACLUS, despite their advantages, are not able to detect actual corridors. Comparing with the output of our approach (in Fig. 3) we can see that our approach detects patterns that lie on top of the road network. On the other hand the output of TRACLUS is abstract lines and extra processing steps would be required in order to detect detailed corridors.

5) Comparison with ZLY10: Here we compare our method against the method proposed by Zhu et. al. [7]. The left part of Fig. 7 illustrates the percentage of the frequent patterns' length covered by the detected corridors from the two approaches (*COR* and *ZLY10*), while the corresponding $MDL\ scores$ are visualized at the right part of the Figure. Our technique covers larger parts of the frequent patterns achieving better $MDL\ score$ than *ZLY10*. Also, our approach is unaffected by the number of frequent patterns, or the existence of noise. Both the approaches have similar $MDL\ scores$ for the benchmarks that contains 5 frequent patterns without noisy trajectories, but as the benchmarks become more complex our approach outperforms *ZLY10*.

Also, we measured the $MDL\ scores$ of the two approaches on the urban datasets, using 5000 trajectories. *Our approach achieves* 0.42 *and* 0.36 $MDL\ score$ *for the Porto and the Dublin datasets respectively, while ZLY10 achieves* 0.93 *and* 0.85 $MDL\ scores$ *for the same datasets. ZLY10* detects frequently visited corridors ignoring whether or not the objects followed the whole or part of the corridor. Thus, it detects less and longer corridors comparing to our approach. On the other hand our approach is able to detect corridors in high level of granularity, achieving much better $MDL\ scores$.

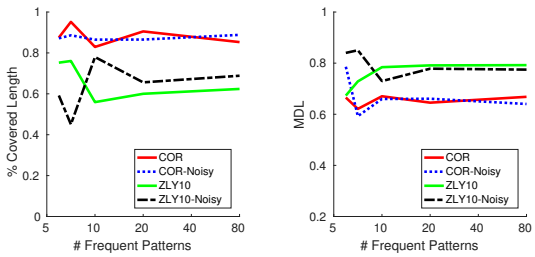

Figure 7: Percentage of synthetic frequent patterns' covered by the found corridors (left) and the $MDL\ scores$ (right).

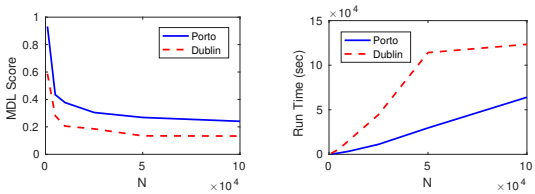

Figure 8: Performance for various number of trajectories N.

B. Varying the Parameters

Below we describe how different values of N, K and θ_{Cl} affect the $Runtime$ and the $MDL\ score$.

1) Number of Trajectories N: Fig. 8 describes how different amounts of input trajectories N affect the $MDL\ score$ (left) and the execution $Runtime$ (right). The $MDL\ score$ is increased when $\mathcal{C}$ is computed with few trajectories, but as N increases it sharply meliorates.

2) Number of Frequent Sets K: Fig. 9 presents the number of the detected corridors (left) and the $MDL\ score$ (right) for various values of K. Larger values of K reduce the number of detected corridors and deteriorate the $MDL\ score$. As K increases LDA associates the frequent sets with smaller areas, not allowing our approach to detect longer corridors.

Furthermore, Figure 10 shows how K affects the synthetic datasets. The performance for the benchmarks with few frequent patterns is unaffected by the selected K, but more complex benchmarks require increased K. Also, our approach achieves better compression for benchmarks without noise (Fig. 10-left), than for the noisy benchmarks (Fig. 10-right).

3) Performance at the Test Set: Here we evaluate the derived set of corridors $\mathcal{C}$ in a new unseen collection of trajectories, different from the dataset used to detect them. The test sets contained 10^5 buses and taxis trajectories. *The detected* $MDL\ score$ *for the Dublin dataset is* 0.16*, while for the Porto it is* 0.27. This means that there are regularities in the data that our approach is able to detect, justifying our initial claims. The found scores are similar with the scores found when the model was self-evaluated in the training set.

4) Runtime Profiling: The bar plots of Fig. 11 illustrate the required $Aggr.\ Runtime$ for the different processing phases of the proposed framework. Larger values of K decrease the

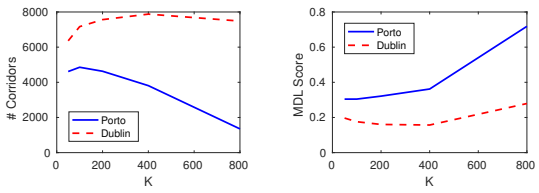

Figure 9: Results for various number of frequent sets K.

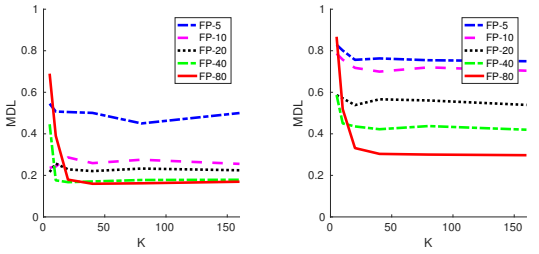

Figure 10: MDL scores, for different values of K, for the synthetic datasets without noise (left) and with noise (right).

total execution time. Even if the time required to run LDA is linearly related to the selected number of frequent sets, the processing of the rest processing components is decreased sharply, as K increases. This is happening since smaller values of K associate larger spatial areas with each frequent set, requiring more computations to extract the corridors.

Finally, the time needed to group together similar subtrajectories is slightly decreased for greater θ_{Cl} values. On the other hand the time required to detect the candidate corridors C_{cand} is increased for larger θ_{Cl} values, since more subtrajectories are added in the clusters requiring more computations between trajectories, thus smaller θ_{Cl} values are favoured.

VI. CONCLUSION

In this work, we proposed a pipelined approach for detecting a set of frequently accessed corridors from a vast collection of trajectories. Initially we applied a well known topic modelling technique to detect frequent sets of locations and then we derived the frequent corridors from the trajectories that accessed these locations.

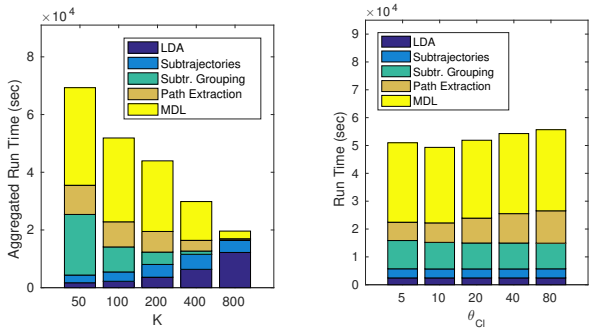

Figure 11: Aggregated runtime varying K and θ_{Cl}.

ACKNOWLEDGMENTS

European Union Horizon2020 grant 688380 "VaVeL" and a Google 2017 Faculty Research Award.

REFERENCES

[1] Nikolaos Zygouras and Dimitrios Gunopulos. Discovering corridors from gps trajectories. In *ACM SIGSPATIAL*, page 61, 2017.

[2] Chen Chen, Hao Su, Qixing Huang, Lin Zhang, and Leonidas Guibas. Pathlet learning for compressing and planning trajectories. In *ACM SIGSPATIAL*, pages 392–395, 2013.

[3] Michael R Evans, Dev Oliver, Shashi Shekhar, and Francis Harvey. Summarizing trajectories into k-primary corridors: a summary of results. In *ACM SIGSPATIAL*, pages 454–457, 2012.

[4] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. Traffic density-based discovery of hot routes in road networks. In *SSTD*, pages 441–459. Springer, 2007.

[5] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *ACM SIGMOD*, pages 593–604, 2007.

[6] Huiping Cao, Nikos Mamoulis, and David W Cheung. Mining frequent spatio-temporal sequential patterns. In *IEEE ICDM*, pages 8–pp, 2005.

[7] Haohan Zhu, Jun Luo, Hang Yin, Xiaotao Zhou, Joshua Zhexue Huang, and F. Benjamin Zhan. Mining trajectory corridors using fréchet distance and meshing grids. In *PAKDD*, pages 228–237, 2010.

[8] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *ACM SIGKDD*, pages 330–339, 2007.

[9] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. In *SAC*, pages 644–655. Springer, 2008.

[10] J. Gudmundsson, A. Thom, and J. Vahrenhold. Of motifs and goals: mining trajectory data. In *ACM SIGSPATIAL*, pages 129–138, 2012.

[11] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W Cheung. Mining, indexing, and querying historical spatiotemporal data. In *ACM SIGKDD*, pages 236–245, 2004.

[12] Htoo Htet Aung, Long Guo, and Kian-Lee Tan. Mining sub-trajectory cliques to find frequent routes. In *SSTD*, pages 92–109. Springer, 2013.

[13] Elio Masciari, Gao Shi, and Carlo Zaniolo. Sequential pattern mining from trajectory data. In *ACM IDEAS*, pages 162–167, 2013.

[14] Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. *The VLDB Journal*, 24(2):169–192, 2015.

[15] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *IEEE ICDE*, pages 900–911. IEEE, 2011.

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022, 2003.

[17] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering objects and their location in images. In *ICCV 2005.*, volume 1, pages 370–377. IEEE, 2005.

[18] Diane Hu and Lawrence K Saul. A probabilistic topic model for unsupervised learning of musical key-profiles. In *ISMIR*, pages 441–446. Citeseer, 2009.

[19] Renjie Zheng, Qin Liu, Weixiong Rao, Mingxuan Yuan, Jia Zeng, and Zhongxiao Jin. Topic model-based road network inference from massive trajectories. In *MDM*, pages 246–255. IEEE, 2017.

[20] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370, 1994.

[21] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[22] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, I. Verkamo, et al. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, 12(1):307–328, 1996.

[23] L. Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[24] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. Predicting taxi–passenger demand using streaming data. *IEEE ITS*, 14(3):1393–1402, 2013.

[25] Nikolaos Zygouras, Nikos Zacheilas, Vana Kalogeraki, Dermot Kinane, and Dimitrios Gunopulos. Insights on a scalable and dynamic traffic management system. In *EDBT*, pages 653–664, 2015.

[26] Atlantic Tropical Storm Tracking by Year. http://weather.unisys.com/hurricane/atlantic/, 2017.