Decentralized Human Trajectories Tracking Using Hodge Decomposition in Sensor Networks

Xiaotian Yin* Chien-Chun Ni† Jiaxin Ding † Wei Han * Dengpan Zhou †
Jie Gao † Xianfeng David Gu †

ABSTRACT

With the recent development of localization and tracking systems for both indoor and outdoor settings, we consider the problem of analyzing and representing the huge amount of natural trajectories from human movements that we expect to gather in the near future. In this paper we argue the topological representation, which records how a target moves around the natural obstacles in the underlying environment, can be sufficiently descriptive for many applications and efficient enough for both storing, comparing and classifying these natural human trajectories. Technically, the representation uses the homotopy type of the trajectory. By using harmonic one-forms and Hodge decomposition, we pre-process the sensor network with a purely decentralized algorithm such that the homology class of a trajectory can be obtained by a simple integration along the trajectory. This supports real-time classification of trajectories up to the homology accuracy with minimum communication cost. We test the effectiveness of our approach by showing how to classify randomly generated trajectories in a multi-level arts museum layout as well as how to distinguish real world taxi trajectories in a large city.

Keywords

Homology, Hodge decomposition, trajectory analysis, distributed computation, sensor networks.

Categories and Subject Descriptors

I.3.5 [**COMPUTER GRAPHICS**]: Computational Geometry and Object ModelingGeometric algorithms, languages, and systems; I.3.7 [**COMPUTER GRAPHICS**]: Three-Dimensional Graphics and RealismVisible line/surface algorithms; G.2.2 [**Mathematics of Computing**]: DISCRETE MATHEMATICSPath and circuit problems

*Mathematics Department, Harvard University. Email: {xyin, weihan}@math.harvard.edu

†Computer Science Department, Stony Brook University. Email: {chni,jiading,dpzhou, jgao, gu}@cs.sunysb.edu

SIGSPATIAL'15 November 03-06, 2015, Bellevue, WA, USA
ACM ISBN 978-1-4503-3967-4/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2820783.2820844.

1. INTRODUCTION

Powered by recent technology advancements, real-time target tracking is becoming a reality for both outdoor and indoor scenarios. For example, we can track visitor's trajectories in a museum by RFID enhanced entrance ticket, or track vehicles traveling in large metropolitan areas and tracked by roadside units. In all cases, human movement patterns are never random and driven by social needs. Thus, any new discoveries of these real world mobilities are going to be fundamentally interesting and practically useful.

The focus of this paper is to develop efficient schemes for collecting, processing and analyzing such tracking data in *real time* within the network of sensors performing the tracking tasks. Trajectory data is a unique type of sensing data: they are sequential (temporal), and typically bulky. The straightforward representation of a target visiting the vicinity of k sensors uses a sequence of the IDs of these sensors or their geographical locations, generating data size of $O(k)$. Even with simplification techniques, the data size cannot be reduced much especially for trajectories that are geometrically complicated. Most of existing algorithms handle trajectory data with limitations. In one approach, the target detection events are separately stored at individual sensors. This minimizes the amount of communication between sensors during processing. However, the global knowledge of a trajectory is not present and retrieving this information involves a non-trivial query in the network touching all sensors on the path. In another approach, one can possibly construct the trajectory explicitly and propagate this information to the other sensor nodes, facilitating the query for such information. However the data size of a trajectory grows with its length. Thus to pass around a trajectory explicitly, the processing cost, in particular the communication cost, can be prohibitive.

The contribution of this paper is to provide a framework to process and analyze the trajectory data in-network with low communication cost, and also supports queries, comparisons and classifications of trajectories. One of the crucial ideas is to replace the *geometric* representation of a trajectory by the *topological* representation of a trajectory. In a geometric representation of a trajectory, we consider it as a curve in the plane, or possibly a curve in space (when the height data is non-trivial). In a topological representation, we represent a trajectory by how it moves around other specified obstacles in the same domain. These obstacles are often real obstacles (buildings, lakes, or furnitures, walls), but can also be artificially labeled by users to adapt to particular application scenarios or to increase resolution. It is true that this topological information is not sufficient to pin down the exact geometry of the specific path, e.g., we cannot tell which lane the vehicle has taken or how many inches the visitor was to each art pieces. But a fair amount of high-level geometric information is well preserved, such as the sequence of roads followed by the vehicles or the sequence

of paintings the visitor toured. This is enough for many applications that only care about the coarse grained geometry. In fact, the problem of trajectory clustering is precisely to filter out the unnecessarily fine geometric details and keep only the most prominent features of the trajectories such that one can define movement patterns and behaviors including flocking, convoys, herds, leadership, commuting, and various others.

When the topological representation of a trajectory is sufficient and preferred, the benefit of this representation is huge. As we will show later, one can succinctly encode the topological types of a trajectory into a compact way: with only a triple (s, t, h), where s, t are the starting and ending positions and h is a vector of numerical values indicating the topological type. Here the dimension of h is the number of obstacles in the domain. Further, we can pre-process the network such that identifying the topological type of a real time trajectory, i.e., the decoding part, can be done with minimum communication requirements – by simply integrating the weights defined on the sensor network edges crossed of the target trajectory. This allows us to quickly perform clustering and trajectory comparison in real time, and with a little help from a backend server, reconstruct the given path up to topological accuracy.

In the following we briefly review the main technique developed in this paper as well as previous work. We then report our algorithm and evaluation on real data sets.

2. OVERVIEW

Path topology: homotopy v.s. homology. There are various definitions of the topological structures of paths. The most used ones are homotopy and homology types. Both count "holes" in a surface (or more general topological space) by drawing loops on the surface and checking whether the loop encloses a hole. But intuitively homology considers loops as closed curves without orientation, homotopy sees loops as oriented parametric curves, and in that case it matters how the holes are ordered.

The homotopy equivalence relation naturally partitions a group of paths into equivalent classes, providing well defined clusters [1, 13]. In theory the number of homotopy types is infinite, since one can loop around a hole infinitely many times. But in practice only a finite number of homotopy types are of interest.

All paths of the same homotopy types belong to the same class by homology. Thus homology is a weaker classification. The benefit of using homology is that it is much easier to compute in many cases. In our case we will use homology as the framework we develop will also work for a sensor network deployed on a general surface (closed or with boundaries) with high-order topological features such as handles. Consider a donut surface. There are two cycles, α and β, that surround the handle in different ways. These two cycles are clearly non-homotopic as they cannot be deformed to one another. Handles can possibly show up as we live in a three-dimensional space. Consider highways that can go above one another; sensors deployed in a multi-floor building with staircases connecting different floors; tunnels and overhead bridges in a downtown area. In these cases a sensor network can be considered as a sampling of a general surface in 3D which may have handles and/or boundaries.

The problem of testing whether two paths that with the same starting and end positions on a 2D domain are homotopy equivalent has been studied in the centralized setting. Suppose the obstacles are represented by polygons of total size n and the paths are given as polygonal curves. An $\Theta(n \log n)$ running time algorithm is available for simple paths and an $O(n^{3/2} \log n)$ time algorithm is known for self-intersecting paths [2]. For paths defined on a general surface, the homotopy test boils down to testing whether the cycle connecting the two paths is contractible (i.e., shrink to a point). This problem can be solved in a centralized setting in linear running time [5] when the surface and the paths are both available. However, this algorithm cannot be directly applied to the distributed setting of sensor networks, in which a detected target trajectory is locally stored on the sensors near the trajectory and no one sensor has the information about the entire path, not to say the information of the other paths.

For a distributed solution a number of schemes could be possible. For example, in a 2D setting we can connect each obstacle i by a path λ_i to the outer boundary. For each trajectory we record the sequence of intersection with λ_i, together with the direction of crossing. First this scheme only works for 2D domain and does not apply to general cases. Further, the storage needed for this representation is proportional to the number of times that the target trajectory intersects these paths λ_i, which depends on how λ_i are selected and can be suboptimal.

What we choose to adopt in this paper is a different approach using the differential form solution. We define flow vectors, named the *harmonic forms*, on a planarized sensor network graph such that a trivial cycle (i.e., non-hole enclosing) will integrate to zero while non-trivial cycles will integrate to non-zero values. Thus the representation of the homology type is simply the tuple (s, t, h) where h contains the integrated values of the harmonic forms along the trajectory. Two paths of the homology type will give the same integrated values. The computation is simple. The communication cost is minimum and can be piggybacked on the handover cost for sensors along the trajectory of the target. If the target trajectories only represent r different types, a hash function can be used to reduce the size of the integrated differential form to be $O(\log r)$, matching the lower bound. While differential form was used before for range query in a distributed setting [15], the application we introduce in this paper is novel for target tracking and trajectory classification.

Harmonic forms. Consider a graph G with a planar embedding on a surface Σ. The *differential one-form* is a function ξ defined on directed edges. The value $\xi(a, b)$ for an edge ab is the negation of the value $\xi(b, a)$ for edge ba. Now we consider the dual graph $\tilde{G}$. Each face of the graph G corresponds to a vertex in the dual graph. An edge is placed on two vertices in the dual graph if and only if the two corresponding faces in the primal graph share one edge. A differential one-form ξ on the graph G can be extended to the dual $\tilde{G}$. The value on an edge in the dual graph is the value of the corresponding edge in the primal graph. A differential one-form is called a *harmonic one-form* if it satisfies two properties:

1. it is *divergence-free*: $\forall u$ of G, $\sum_{v \in N(u)} \xi(u, v) = 0$, where $N(u)$ is the set of neighbors of u in G;
2. it is *curl-free*, that is, for any vertex $\tilde{u}$ of the dual graph $\tilde{G}$, $\sum_{\tilde{v} \in N(\tilde{u})} \xi(\tilde{u}, \tilde{v}) = 0$, where $N(\tilde{u})$ is the set of neighbors of $\tilde{u}$ in $\tilde{G}$.

The first property means that a harmonic one-form does not have any sources or sinks. If we consider a harmonic one-form as a flow vector defined on each edge, we have the flow conservation property at each vertex – what flows in equals what flows out. The second property means that the integration of a harmonic one-form along any face of G is zero, i.e., in the dual graph there are no sources or sinks either.

With the help of a harmonic one-form we can easily test whether two paths are homologous. In particular, we connect the two paths α, β with the same starting and end positions as a cycle $\alpha - \beta$ (with β in reverse direction). If the two paths are homologous, the cycle encloses enclosing no holes/handles. By the definition of the

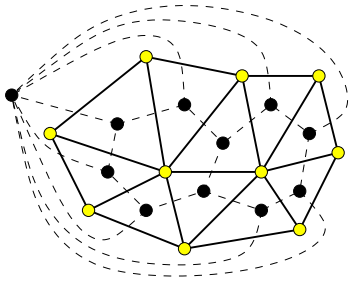


Figure 1. A planar graph G and its dual graph $\tilde{G}$. Each face of the graph G corresponds to a vertex (shown as dark circles) in the dual graph.

harmonic one-form, if we integrate the weights of the edges along the cycle in clockwise order, the integration must be zero. This represents an extremely simple homology test, by using only the knowledge of the harmonic one-form, which can be locally stored on the edges of the network. The communication cost is proportional to the total length of the paths. This is the minimum possible as an algorithm must at least read in the input.

Computing the harmonic forms for homotopy testing is done in a preprocessing phase, using *Hodge decomposition* to be explained next.

Hodge Decomposition. We assume that the sensors can locally extract a planar graph G from the communication graph. The extracted graph G stays on an unknown surface Σ where faces that correspond to holes (i.e., boundaries) are marked. Notice that Σ may have no boundaries. Distributed algorithms that planarize a connectivity graph in wireless networks [9, 10, 16] and identify boundaries [3, 4, 6–8, 12, 14, 17, 18] have been extensively studied in the past few years. If the sensors are densely deployed in the domain, the holes in the sensor network naturally correspond to obstacles that forbid sensor deployment.

To compute a harmonic one-form, we first create an arbitrary one-form ω, say by randomly assigning weights on the edges of the graph G. This one-form is by no means harmonic. The theory of Hodge decomposition says that for any one-form, we can decompose it into three components: $\omega = \alpha + \beta + \gamma$, where

1. γ is a harmonic one-form, it is divergence-free and curl-free;
2. α is a gradient flow, i.e., there is a potential function f defined on the vertices of G such that $\alpha(u, v) = f(u) - f(v)$. A gradient flow is curl-free; and
3. β is a curl flow, i.e., the gradient flow in the dual graph. There is a potential function η defined on faces such that $\beta(u, v) = \eta(x) - \eta(y)$, where x/y is the face to the right/left of edge uv. A curl-free is divergence-free.

The Hodge decomposition basically says that if we take out the gradient flow and curl flow that contribute to having sources/sinks and curls in the flow, we are left with a harmonic one-form. In this paper we develop a purely decentralized algorithm that runs in iterative, gossip style operations that solve for the two components α, β. After we subtract them from the one-form ω we obtain one harmonic one-form.

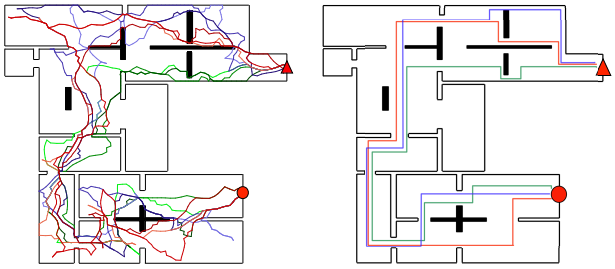
Trajectory Classification. The harmonic one-form is closely related to the topology of the domain, i.e., the number of holes, k. In particular, the family of harmonic one-forms only has dimension k. By running the Hodge decomposition multiple times with different random initial values and testing whether the harmonic forms are linear dependent on each other – an operation that each sensor can individually test – we can find k independent harmonic one-forms as the harmonic one-form basis $\Omega = \{\omega_1, \omega_2, \cdots, \omega_k\}$. Each trajectory P is represented as $(s, t, h(P))$, where $h(P)$ is a vector of dimension k containing the integration of all k one-forms along P.

For the application of testing whether two trajectories P_1, P_2 are homologous, we simply check if their representations are the same. We note that our method is purely combinatorial – no location information is needed beyond having the network represented by a combinatorial triangulated domain. This nice property is shared by the algorithm by Ghrist [11] which uses winding number and angle measurements to test whether a point is inside a cycle. Our method is also purely decentralized. All the network nodes run the same set of codes and none of them does anything special.

3. EVALUATION

In this section, we evaluate our path homology detection algorithm under two real world data. Firstly, we test our algorithm with random generated trajectories on the floor plan of a gallery. Secondly, we analysis the taxi trajectories collected in Shenzhen City, China, and classify them into different categories.

3.1 Trajectories in Gallery



(a) (b)

Figure 2. A demonstration of trajectories of three different homology types in a museum. Trajectories with the same homology travel rooms with the same sequence and go through the same direction of the walls. Trajectories with same homology are labeled with the same color.

We first choose the domain to be the floor plan of a museum, as shown in Fig. 2. In this domain, we uniformly distribute 3625 nodes with average degree of 5.565; the domain (museum) is separated by obstacles (walls) into 15 rooms, entrance and exit are marked as circle and triangle, respectively.

To test the ability of detecting trajectories with different homology types, random trajectories are generated in a two level manner. We first decide room sequences to walk through, then construct the trajectories based on the room sequences. For the first level, we treat each room in the domain as a node, and connect two room nodes if they share the same door. In this way, the domain is converted into a topology graph. The walk through room sequences are randomly chosen from the entrance to the exit. For the second level, we randomly choose a point for each room in the sequence as intermediate node to pass through, then connect these nodes in order with shortest paths.

Fig. 2 shows different trajectories with three different homology types grouped by our algorithm. In this figure, sample trajectories enter the domain from the red circle and exit from the red triangle. Trajectories with the same homology type imply that they all pass through the obstacle with the same order and same direction. In this museum example, our algorithm is able to differentiate all 32 possible different homology trajectories (5 obstacles with 2^5 possibilities for simple path).

3.2 Taxi Trajectories in Shenzhen

Table 1. Taxi data in Shenzhen

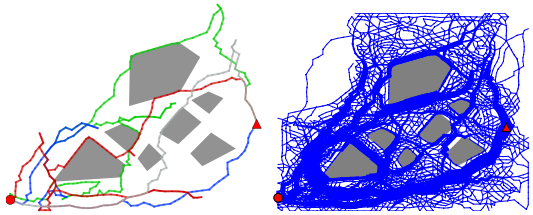
Trajectory Data Before Processing			
longitude	latitude	#traj.	sample points
111.92∼ 116.76	21.52∼ 23.47	9386	288
Trajectory Data After Processing			
longitude	latitude	#traj.	avg. pts per traj.
114.11∼ 114.14	22.54 ∼ 22.57	243	21.6

The taxi trajectory data are collected from 9386 taxies in Shenzhen, sampled per 5 minutes during one day. To simplify our data, we choose a small district of this city with sample points that trajectories passed by as the sample area. The sample data description is illustrated in Table 1.

We choose two frequent visited locations as source and destination points, marked as circle and triangle in Fig. 3. Among all the trajectories, we only choose the ones that going through the source and destination in this analysis. In total, we choose 243 trajectories, and each trajectory is represented by the combination of the shortest path between the sampled points.

In this domain, we marked 7 areas surrounded by main roads as holes in the experiment, shown in Fig. 3. The trajectories of all taxies are plotted in Fig. 4, with line width representing the number of taxies passing through. Notice that the trajectories of taxis have their specific characteristics. Intuitively, the trajectory of a taxi consists of two types of segments: with customers and without customers. The segment of the trajectory from the location where a customer is picked up to the respective destination is likely to follow a near shortest path. The segment for which the taxi carries no customers is possibly more random and may have loops or detours. This assumption matches with what we observed from the data – the trajectory from our chosen source to destination could deviate a lot from the shortest path.

Figure 3. The holes are plotted on the real map. The source and destination are marked as circle and triangle, respectively.



(a) 4 trajectories with different homology types (b) Trajectory flow

Figure 4. In Figure 4(a), 4 trajectories of different homology types are plotted. In Figure 4(b), all trajectories are plotted as a flow where the width represents the number of taxies going through the path.

Acknowledgment

The authors are partially supported by grants from AFOSR (FA9550-14-1-0193) and NSF (CCF-1535900, DMS-1418255, DMS-1221339, CNS-1217823).

4. REFERENCES

[1] K. Buchin, M. Buchin, M. Van Kreveld, M. Löffler, R. I. Silveira, C. Wenk, and L. Wiratma. Median trajectories. In *Proceedings of the 18th annual European conference on Algorithms: Part I*, ESA'10, pages 463–474, Berlin, Heidelberg, 2010. Springer-Verlag.

[2] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. In *Proceedings of the eighteenth annual symposium on Computational geometry*, SCG '02, pages 160–169, New York, NY, USA, 2002. ACM.

[3] V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic and Geometric Topology*, 7:339–358, 2007.

[4] V. de Silva and R. Ghrist. Homological sensor networks. *Notices American Mathematical Society*, 54(1):10–17, 2007.

[5] T. K. Dey and S. Guha. Transforming curves on surfaces. *J. Comput. Syst. Sci.*, 58(2):297–325, 1999.

[6] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *Mobile Networks and Applications*, volume 11, pages 187–200, 2006.

[7] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2004.

[8] S. Funke and C. Klein. Hole detection or: "how much geometry hides in connectivity?". In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 377–385, 2006.

[9] S. Funke and N. Milosavljević. Network sketching or: "how much geometry hides in connectivity? - part II". In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 958–967, 2007.

[10] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanners for routing in mobile networks. *IEEE Journal on Selected Areas in Communications Special issue on Wireless Ad Hoc Networks*, 23(1):174–185, 2005.

[11] R. Ghrist, D. Lipsky, S. Poduri, and G. S. Sukhatme. Surrounding nodes in coordinate-free networks. In *Proceedings of the Seventh International Workshop on the Algorithmic Foundations of Robotics*, pages 409–424, 2006.

[12] R. Ghrist and A. Muhammad. Coverage and hole-detection in sensor networks via homology. In *Proc. the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 254–260, 2005.

[13] R. Knepper, S. Srinivasa, and M. Mason. An equivalence relation for local path sets. In J.-C. L. David Hsu, Volkan Isler and M. C. Lin, editors, *The Ninth International Workshop on the Algorithmic Foundations of Robotics*, Tiergartenstrasse 17 69121 Heidelberg Germany, December 2010. Springer.

[14] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 1000–1009, 2006.

[15] R. Sarkar and J. Gao. Differential forms for target tracking and aggregate queries in distributed networks. In *Proc. of the 16th Annual International Conference on Mobile Computing and Networking (MobiCom'10)*, pages 377–388, September 2010.

[16] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using ricci flows. In *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, pages 97–108, April 2009.

[17] O. Saukh, R. Sauter, M. Gauger, and P. J. Marrón. On boundary recognition without location information in wireless sensor networks. *ACM Trans. Sen. Netw.*, 6:20:1–20:35, June 2010.

[18] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 122–133, September 2006.