Choosing Thresholds for Density-Based Map Construction Algorithms *

Mahmuda Ahmed
Dept. Computer Science
Univ. of Texas at San Antonio
mahmudaahmed@gmail.com

Brittany Terese Fasy
Dept. Computer Science
Tulane University
bfasy@tulane.edu

Matt Gibson
Dept. Computer Science
Univ. of Texas at San Antonio
gibson@cs.utsa.edu

Carola Wenk
Dept. Computer Science
Tulane University
cwenk@tulane.edu

ABSTRACT

Due to the ubiquitous use of various positioning technologies in smart phones and other devices, geospatial tracking data has become a routine data source. One of its uses that has gained recent popularity is the construction of street maps from vehicular tracking data. Due to the inherent noise in the data, many map construction algorithms are based on thresholding a density function. While kernel density estimation provides a firm theoretical foundation for computing the density from the measurements, the thresholds are generally picked in a heuristic, and often brute-force way, which results in slow algorithms with no guarantees on the map construction quality.

In this paper, we formalize the selection of thresholds in a density-based street map construction algorithm. We propose a new thresholding technique that uses persistent homology combined with statistical analysis to determine a small set of thresholds that captures all relevant topological features. We formally prove that when the samples are drawn uniformly from the street map, a constant number of thresholds suffices to recover the street map. We also provide algorithms to compute the thresholds for different sampling assumptions. Finally, we show the effectiveness of our algorithms in several experiments on artificially generated data and on real GPS trajectory data.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]

*This work has been partially supported by the National Science Foundation grant CCF-1301911.

SIGSPATIAL '15, November 03 - 06, 2015, Bellevue, WA, USA

ACM 978-1-4503-3967-4/15/11 ...$15.00
DOI: http://dx.doi.org/10.1145/2820783.2820810

General Terms

Theory, Algorithms, Experimentation, Measurement

Keywords

Map Construction, Density, Thresholding

1. INTRODUCTION

Due to the ubiquitous use of positioning technologies in smart phones and other devices, geospatial tracking data has become a highly utilized resource [3, 7]. Applications of tracking data include map matching [6, 9, 21, 22, 23], map updates [28], and traffic detection [27]. One of its uses that has gained recent popularity is the construction of street maps from vehicular tracking data [4, 8, 10, 12, 14, 18, 19, 26]. Map construction algorithms can be grouped into three categories [3]: point clustering, incremental track insertion, and intersection linking. The input is a set of tracks, each consisting of a sequence of position measurements. The desired output is a representation of the street map, often represented as an embedded graph.

Due to inherent noise in data, many map construction algorithms follow a density-based point clustering approach [8, 12, 26]. These algorithms proceed in two steps: (1) density estimation and (2) map extraction. The first step involves estimation of data density over the (discretized) plane, using non-parametric methods such as histogram computation or kernel density estimation. The input data is often a set of point measurements or of curves obtained by interpolating input trajectories between those measurements. A kernel density estimate [24] places kernels (such as Gaussian kernels) at data points and sums the kernels. This avoids the inherent bias in histogram methods that stems from an arbitrary choice of histogram bins. In the second map construction step, the map has to be extracted from the density. Since a density is a function over the plane, it can also be viewed as a grayscale image, and hence map extraction is closely related to image segmentation. Usually, *thinning* or *skeletonization* algorithms are applied to a thresholded density in order to extract a set of pixels that lie on the graph, the so-called *skeleton*. A common post-processing step is to compute an embedded graph from the skeleton; see, for example, Biagioni and Eriksson [8] for a density-based map construction pipeline with multiple post-processing steps.

In this work, we concentrate on skeleton extraction from the data density using thresholds. Choosing the right threshold has always been a challenge, and although multi-level thresholding is quite well studied in the general context of image segmentation, it has only recently been introduced for map construction [8, 13, 26]. In practice, due to multiple street categories as well as noise, multi-level thresholding is generally very effective. However, while kernel density estimation provides a firm theoretical foundation for computing the density from the measurements, the thresholds for segmentation and skeletonization are picked in a heuristic, and often brute-force, way, which results in slow algorithms with no guarantees on the map construction quality.

Our Contribution. In this paper, we formalize how many threshold levels are necessary for a density-based streetmap construction algorithm. In Section 3, we propose a new thresholding technique that uses persistent homology to determine a small set of thresholds that captures all topological features. This technique is very robust in removing insignificant thresholds by performing a statistical analysis of the features, and it works for any sampling strategy. In Section 4, we provide a formal model of different sampling assumptions involving point measurements or trajectories. We prove that for uniform sampling a constant number of thresholds suffices to recover the street map, as the number of samples tend to infinity. We also provide algorithms to compute the thresholds for the different cases. In our experiments in Section 5, we apply our algorithms to three datasets: two artificially generated and one real-world set of taxi tracking data collected in Berlin, Germany.

2. PRELIMINARIES

Given a set of trajectories, each represented as a sequence of position samples, the map construction task is to construct a road map that best represents all input trajectories. In this paper, we represent the map as an undirected graph. Density-based map construction algorithms first construct a density over the plane from the input trajectories, either directly from the set of all input position samples or from curves representing the input trajectories. In order to extract the map from the density, the algorithms then proceed by thresholding the density to obtain a fattened graph, which is subsequently thinned.

Historically, many algorithms attempt to use only a single threshold; see [11, 12]. On the other hand, Steiner and Leonhardt [26] use the watershed transform and Biagioni and Eriksson [8] iterate over a large subset of all observed function values of the density. Algorithms that use a single threshold may miss features that arise on scales other than the chosen threshold. Heuristic multi-threshold approaches, therefore, may use a very large number of thresholds in order to ensure that all topological features of interest are captured. This, however, tends to be rather inefficient.

The focus of this paper is the optimization of the number of thresholds considered for density-based map construction algorithms. Intuitively, it should be sufficient to identify a (small) set of canonical thresholds that are critical for topological changes. We present two algorithms for choosing such threshold sets. One of our approaches uses *persistent homology*, a tool from computational topology, to identify critical thresholds. Persistent homology, and the associated *persistence diagrams*, have been used in image segmentation [20, 25], for map comparison [2], and for map generalization [5]. In this section, we describe the generic framework for density-based map construction algorithms, and we provide the necessary background on persistence diagrams.

2.1 Density-Based Map Construction

Let the underlying street-map G be an undirected graph embedded in a compact rectangular region $D \subseteq \mathbb{R}^2$. We slightly abuse notation to denote with G the graph (consisting of edges and vertices) as well as the subset of $\mathbb{R}^2$. An embedding for G may represent vertices as single points and edges as curves connecting these points, or it may represent vertices and edges as thicker regions in an attempt to more closely model the geometry of streets and their widths. We now discretize D using an $m_1 \times m_2$ grid to obtain a set $\widetilde{D}$ of $m_1 \cdot m_2$ pixels, such that $D = \cup\widetilde{D}$. We then define $\widetilde{G}$ as the set of pixels containing G: $\widetilde{G} = \left\{ q \in \widetilde{D} \mid q \cap G \neq \emptyset \right\}$. The input data is given as a set $X = \{X_1, X_2, \ldots, X_n\}$ of n position samples in D. The output of the map construction algorithms should be an estimate of $\widetilde{G}$.

As mentioned above, there are several map construction algorithms that use a density as the main tool. They usually follow the generic algorithm outline below:

Algorithm 1 Generic Density-Based Map Construction

Input: X, $\widetilde{D}$
1: $\hat{p} \leftarrow$ density estimate for X over $\widetilde{D}$
2: $T \leftarrow$ set of thresholds
3: **for all** $\tau \in T$ **do**
4: $\quad \widetilde{D}^\tau \leftarrow \hat{p}^{-1}([\tau, \infty))$
5: Use $\{\widetilde{D}^\tau\}_{\tau \in T}$ to estimate $\widetilde{G} \subset \widetilde{D}$.
6: **return** $\widetilde{G}$

We now give more details of this algorithm:

Line 1: Density Estimation. There are several different ways to define and compute a density for X, usually using non-parametric methods such as histogram computation or kernel density estimation. For a statistical density estimate, the data set X is usually assumed to be taken i.i.d. from an unknown distribution p_h, and there exist many methods to estimate p_h. A *kernel density estimate* requires the use of a *kernel*; while there are many kernels to choose from, we use the Gaussian kernel $K(x, y; \sigma) := \exp(\frac{-||x-y||^2}{2\sigma^2})$, which is widely used in practice. Here, $\sigma > 0$ is a *bandwidth* parameter. The *kernel density estimate (KDE)* at any point $x \in D$ is defined as:

$$\hat{p}_\sigma(x) := \frac{1}{2\pi n\sigma^2} \sum_{X_i \in X} K(x, X_i; \sigma),$$

where $n = |X|$. Computing the function values at the centers of every pixel in $\widetilde{D}$ and triangulating the domain creates a piecewise linear approximation of $\hat{p}_\sigma$. In this paper, we assume that the discretization is fine enough that the L_∞-distance between the KDE and its approximation is negligible. In Section 3, we use a KDE to estimate the density.

A histogram is defined by partitioning the domain D into bins, which in the plane is usually a regular grid, and then counting for each bin how many data points fall into this bin. These counters can be divided by the total number $n = |X|$ of data points to obtain a density. In Section 4 we

use such a histogram-based density. In practice, histograms are often blurred with a Gaussian filter to account for noise.

Line 2: Computing Thresholds. Various methods may be employed to select thresholds. Biagioni and Eriksson [8] use a subset of all density values; they store the density as a 16-bit image and sample the 65,536 values to define the set $T = \{1, 2, 3, \ldots, 16\} \cup \{2^i | 5 \leq i \leq 16\}$ of 28 thresholds.

We propose two different methods to pick a small number of thresholds. In Section 3, we use persistent homology to determine a small threshold set that captures all topological features. In Section 4, we introduce different sampling models, and we prove that a constant number of thresholds suffices to recover the street map when using Chebyshev's inequality to pick thresholds.

Line 5: Discretized Graph. Given a set T of thresholds from Line 2, one now needs to use these thresholds to determine which pixels of $\widetilde{D}$ are in our estimate of $\widetilde{G}$. Intuitively, $\widetilde{D}^\tau$ is the set of all pixels x such that $\hat{p}_x \geq \tau$. Each such set will have some topological properties that should be preserved in the final discretized graph, and [8] proposes a technique to "merge" all $\widetilde{D}^\tau$ together in a manner that preserves these topological properties. We use a similar technique to [8] in our approach proposed in Section 3. In Section 4, we propose a new technique for this step which further classifies the pixels in $\widetilde{G}$ (e.g., by identifying intersection pixels in a road network).

Due to the noise in X, a pixel $x \in \widetilde{D} \setminus \widetilde{G}$ that is very close to a pixel in $\widetilde{G}$ may have a very high density value. This causes standard thresholding techniques to produce edges which are much wider than the actual edge. As an optional step, several "thinning" algorithms have been proposed to reduce the width of the edges.

Post-Processing: The Graph. As a final step, one may want to convert the classification of pixels into some representation of an embedded graph. There are many different ways to extract an undirected graph, and to define lane directions or turn lanes, see Biagioni and Eriksson's pipeline [8] for multiple steps that each provide a more detailed representation of the graph.

2.2 Persistence Diagrams

We define a piecewise-linear function f by defining function values at the vertices and linearly interpolating over the edges and triangles of a compact triangulated domain $\widetilde{D} \subset \mathbb{R}^2$. The super-level set at level τ consists of all points $x \in \widetilde{D}$ such that $f(x) \geq \tau$; see Figure 1. The super-level sets of interest are these thresholded images, and persistent homology captures the topology of these sets. To compute the persistence diagram of f, we track the births and deaths of one-cycles or loops (referred to as *cycles* for the remainder of this paper) realized in the super-level sets $\widetilde{D}^\tau = f[\tau, \infty)$ for $\tau \in [-\infty, +\infty]$. These cycles are called (one-dimensional) *features*. As we sweep τ from $+\infty$ to $-\infty$, we note also that connected components can appear and merge together as we decrease τ; however, these are zero-dimensional features and will be disregarded. In super-level sets of this density function, cycles correspond to the city-blocks of a road network, hence our motivation for considering them.

Each (generating) cycle γ is associated with a height τ_b, where the cycle first appears (specifically, τ_b is a height for which a cycle homologous to $\gamma \in \widetilde{D}^\tau$ does not exist in $\widetilde{D}^{\tau-\varepsilon}$ for any arbitrarily small ε). The cycle γ can die in two ways: it can merge with another cycle γ' or it can be filled in (merge with the trivial cycle). Merging between two simple, non-intersecting cycles γ and γ' occurs when all vertices lying between them are included in the super-level set. Furthermore, persistence theory creates a hierarchy of features: when two cycles with birth height $\tau_1 < \tau_2$ merge, we say that the cycle that born on τ_1 dies. Equivalently, we say that the eldest cycle survives.

For example, the density illustrated in Figure 2a. For simplicity of illustration, assume the function values are integers between two and nine. The super-level sets $\widetilde{D}^\tau$ for $\tau \in \{2, \ldots, 9\}$ are shown in Figure 3. Here, we observe the first birth event of a one-dimensional feature (cycle) occurs at $\tau = 6$ (Figure 3d) and it dies (fills in) at $\tau = 4$ (Figure 3f).

The persistence diagram represents the births and deaths of all the cycles by assigning each feature a point in the plane $\mathbb{R}^2$, where the y-coordinate represents the birth height of the cycle and the x-coordinate represents its death height. Consider again the example in Figure 3. The cycle described above has birth height 6 and death height 4, and we obtain the corresponding persistence point $(4, 6)$ in the persistence diagram. Figure 2b shows the persistence diagram for this filtration, which also includes persistence points $(2, 5)$, $(3, 4)$, and $(1, 3)$. In addition, for technical reasons, we also include all points (x, x) with infinite multiplicity.

In general, one can assume the longer the life-span of a cycle the more significant it is. Components with longer life spans are represented in the persistence diagram by points that are "*far*" from the line $y = x$. We note here that, by construction, all points in the persistence diagram will lie "above" the line $y = x$ since the death height must be less than the birth height. The persistence diagram encodes the topology of all super-level sets of a function; i.e., we can read off the homology (specifically, the number of cycles) of any super-level set from the persistence diagram.

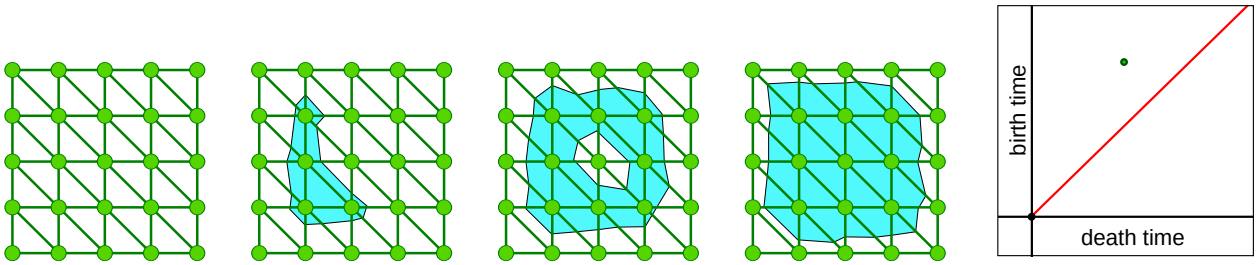
3. THRESHOLDING WITH PERSISTENCE

In this section, we propose a new thresholding technique based on persistent homology that can be plugged into Line 2 of Algorithm 1. In the experimental section, we show that using only the few select thresholds has a significant improvement in runtime over the algorithm of [8] and provides an acceptable representation of the underlying road network.

Generally speaking, the algorithm of this section removes noisy features from the diagram based on their life-span, and then computes a small set of thresholds that captures the significant features (with high enough persistence). Given a smoothing parameter σ, a confidence level α, and a set $X = \{X_1, X_2, \ldots, X_n\}$ of samples taken i.i.d. from a probability distribution over a road network, we first compute the density $\hat{p}_\sigma$. Then, we (1) construct the persistence diagram for $\hat{p}_\sigma$, (2) choose a persistence cut-off so that we can determine which points in the diagram are significant, and (3) determine a the minimal set of thresholds that capture all significant persistence points.

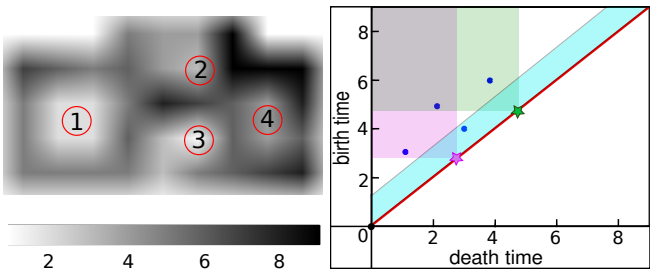
3.1 Computing Persistence Diagrams

In this step, we compute the persistence diagram $D = D(\hat{p}_\sigma) = \{(d_i, b_i)\}_i$ from the super-level set filtration of $\hat{p}_\sigma$. As we mentioned above, we are interested only in the one-dimensional features, as these features represent city blocks in a road network. We compute the persistence diagram using the R package TDA [16].



(a) $\widetilde{D}$ (b) Super-level set (c) One Cycle (d) No Cycles (e) Persistence Diagram

Figure 1: The super-level set is shown as a cyan shaded region. We start off with $\tau = +\infty$ and the super-level set is empty (see Figure 1a). As we decrease τ, we see the birth of components (zero-dimensional features) as we pass local maximum. These components grow (1b) as we continue to decrease τ. Eventually, components may join together to form cycles (one-dimensional features, as illustrated in 1c), which ultimately gets filled in, at which point, we say that a cycle dies (1d). A birth-death pair is called a persistence point. The persistence diagram (1e) plots a persistence point on a $2D$-plane where birth height is along the Y-axis and death height is along the X-axis.



(a) Density Estimate (b) Persistence Diagram

Figure 2: On the left, we have a density estimate. On the right is a persistence diagram for the super-level set filtration of this density. There are four features realized in the super-level sets, but only three thresholds are needed to capture all loops in the underlying graph.

3.2 Noisy Feature Removal

A confidence set for a persistence diagram D is an estimated diagram $\widehat{D}$ along with bound c_α for the distance between D and $\widehat{D}$ that holds with probability $1-\alpha$:

$$\mathbb{P}(W_\infty(D, \widehat{D}) > c_\alpha) \geq 1 - \alpha.$$

One way of interpreting these confidence sets is as above: each point in $\widehat{D}$ with persistence at least c_α represents a distinct persistence point in the *true diagram* D, with probability $1-\alpha$. For this reason, we call the value c_α the *persistence cut-off*. We illustrate this in Figure 2b, where the cyan region corresponds to the set of points that are not statistically distinguishable from noise.

Given the persistence diagram, we now describe a procedure of distinguishing which points are statistically significant. Then, in Section 3.3, we define a set of thresholds that will capture all statistically significant features.

Note: In the selection methods, we use the stability of persistence diagrams in order to compute a confidence band. That is, if D is the diagram corresponding to a function p and $\widehat{D}$ is the persistence diagram for $\hat{p}$, where p and $\hat{p}$ are defined over the same domain then: $||p - \hat{p}||_\infty \geq W_\infty(D, \widehat{D})$. Furthermore, we can find an upper bound on the LHS of this inequality in order to bound the distance between persistence diagrams D and $\widehat{D}$.

Deterministic Selection.

Given a confidence level α, we can compute c_α such that $\mathbb{P}(||p_\sigma - \hat{p}_\sigma||_\infty > c_\alpha) \leq \alpha$, where p_σ is the true density convolved with the spherical kernel of bandwidth σ. Then, we can use c_α to classify each persistence pair (b_i, d_i) as either statistically significant (interpreted to be topological signal) or not (interpreted to be topological noise). We choose c_α using Lemma 11 of [17]; in particular,

$$c_\alpha = \left(\frac{K(0,0;\sigma)}{\delta}\right)^2 \sqrt{\frac{1}{2n}\log\left(\frac{2N}{\alpha}\right)},$$

where N is the size of the grid.

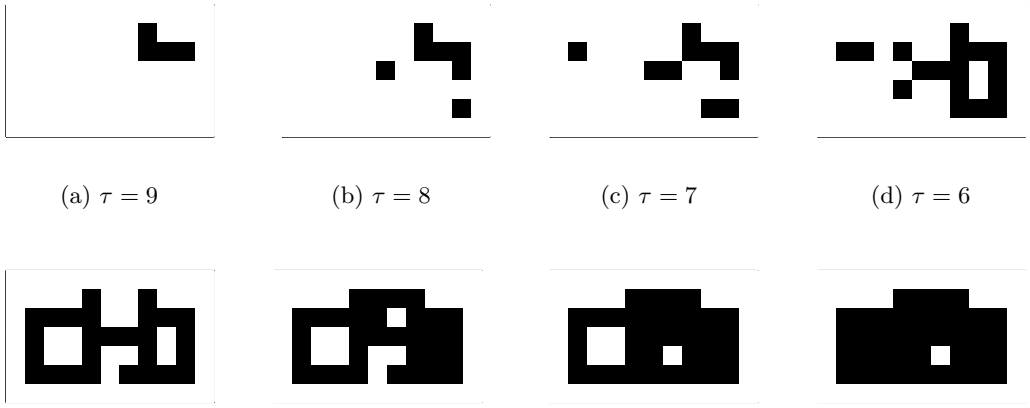
Bootstrapped Selection.

Alternatively, we compute an asymptotic confidence set using a technique such as the Bootstrap as give in Theorem 12 of [17]. In this formulation, we treat the initial data set as an empirical distribution, and sample n points i.i.d. from this empirical distribution (with replacement). This is the so-called *bootstrap sample* $X^* = \{X_1^*, \ldots, X_n^*\}$. Using a large number of bootstrapped samples, we can compute c_α such that (in the limit, as $n \to \infty$), $\mathbb{P}(||p_\sigma - \hat{p}_\sigma||_\infty > c_\alpha) \leq \alpha$. We refer the interested reader to [17, 29].

3.3 Extracting Thresholds

Let P denote the set of points in the persistence diagram that were deemed to be statistically significant. Recall that each point (τ_d, τ_b) corresponds with a feature that has a birth height τ_b and a death height τ_d. We will now describe how we use the persistence diagram to compute a set of thresholds T so that each feature can be detected using some threshold $\tau \in$ T.

For any threshold τ, consider the point (τ, τ) along the line $y = x$, and further consider the axis-parallel rectangle consisting of all points (x, y) such that $x \leq \tau$ and $y \geq \tau$. See Figure 2b. We call such a rectangle an *anchored rectangle*; we say that the rectangle is anchored at (τ, τ).



(a) $\tau = 9$ (b) $\tau = 8$ (c) $\tau = 7$ (d) $\tau = 6$

(e) $\tau = 5$ (f) $\tau = 4$ (g) $\tau = 3$ (h) $\tau = 2$

Figure 3: Super-level sets $\widetilde{D}^\tau$ for $\tau \in \{2, \ldots, 9\}$. The first birth event occurred at $\tau = 6$ and the feature (hole) died (cycle filled up) at $\tau = 4$. See the corresponding persistence diagram in Figure 2b.

Lemma 3.1. *A feature is detected by a threshold τ if and only if the rectangle anchored at (τ, τ) covers the feature's point in the persistence diagram.*

Proof. First assume that a feature point (τ_d, τ_b) is contained in the rectangle anchored at (τ, τ). Then this feature, as it was born at a height $\tau_b \geq \tau$ and died at a height $\tau_d \leq \tau$, will be present at threshold τ.

Now suppose that (τ_d, τ_b) is not contained in this rectangle. Then we have either $\tau_b < \tau$ or $\tau_d > \tau$. If $\tau_b < \tau$, then the feature will not have been born at height τ and therefore will not be detected. Similarly, if $\tau_d > \tau$, then the feature died prior to height τ and will not be detected. □

The consequence of Lemma 3.1 is that the problem of computing the minimum number of thresholds to detect all features can be reduced to computing the minimum number of anchored rectangles that cover all points in the persistence diagram. This leads to an algorithm (2). Note that we assume that the input points are in general position; that is, the set of all birth/death heights are distinct.

Algorithm 2 Threshold Extraction

Input: P
1: $\mathrm{T} \leftarrow \emptyset$
2: **while** $P \neq \emptyset$ **do**
3: Let τ_{b^*} denote the minimum birth height over all points in P.
4: Let $P' \subseteq P$ be the set of all points contained in the rectangle anchored at (τ_{b^*}, τ_{b^*}).
5: Let τ_{d^*} be the maximum death height over all points in P'.
6: $\mathrm{T} \leftarrow \mathrm{T} \cup \{(\tau_{b^*} + \tau_{d^*})/2\}$
7: $P \leftarrow P \setminus P'$.
8: **return** T

Intuitively, step 6 shifts the rectangle slightly so that the rectangle still covers exactly the points in P' without leaving any point of P directly on its boundary. Correctness of Algorithm 2 follows very similarly to standard approaches for computing a minimum hitting set of a set of intervals, and similarly can be implemented in $O(|P| \log |P|)$ time.

Lemma 3.2. *Given a set P of significant features, Algorithm 2 returns in $O(|P| \log |P|)$ time a minimum cardinality subset of thresholds* T *so that each feature of P is detected by at least one threshold in* T.

3.4 Variant I of Algorithm 1

We now recall Algorithm 1. In order to fully describe the map construction algorithm that we propose in this section, we must explicitly state how we are performing Lines 1, 2, and 5. In Line 1, we use a discretized KDE which can be computed with the TDA package in $O(n^\omega)$-time (matrix-multiplication time), although this can be improved to nearly linear time. We then compute the persistence diagram for $\hat{p}_\sigma$, as well as a confidence set for it using one of the algorithms defined in Section 3.2. This returns a set P such that each significant feature appears in P with probability $1 - \alpha$. We then give P as input to Algorithm 2, and the set T returned by Algorithm 2 can be used in Line 2 of Algorithm 1. Using T, Algorithm 1 produces a map that contains all of the significant features of the super-level sets. To see this, consider any significant feature. By Lemma 3.2, there is a threshold $\tau \in \mathrm{T}$ such that the rectangle anchored at (τ, τ) covers the feature point in the persistence diagram. Therefore when Algorithm 1 considers τ at Line 4, the feature will appear in $\widehat{D}^\tau$ and therefore will appear in the discritized graph when Line 5 is executed using an approach such as in [8]. The main strength of our approach compared to that of [8] is that we use a smaller subset of the thresholds, resulting in a significantly improved runtime of Algorithm 1.

The computational complexity of computing persistence diagrams and hence our proposed thresholding technique does not depend on data density, it only depends on the dimension of the matrix that represents the density function. This technique is also easily parallelizable by considering a set of smaller rectangular regions computed from the origi-

nal area of interest. In order to capture all features here the restriction is each city block must appear in at least one of those smaller rectangles.

Theorem 3.3. *When using Algorithm 2 in Line 2 of Algorithm 1, the resulting map contains all of the significant features. Moreover, if Algorithm 1 uses any set of thresholds* T' *that satisfies* $|\mathrm{T}'| < |\mathrm{T}|$*, then the resulting map will not contain some significant feature.*

4. CHEBYSHEV-STYLE THRESHOLDS

While the previous method for selecting thresholds is very general, sometimes knowing more information about the sampling model is essential for understanding the data. We begin by formally defining noisy samples from a discretized graph $\widetilde{G}$ as follows: Let P_1 be a uniform distribution over $\widetilde{G}$ and P_2 be a uniform distribution over $\widetilde{D} - \widetilde{G}$. Define the mixed distribution

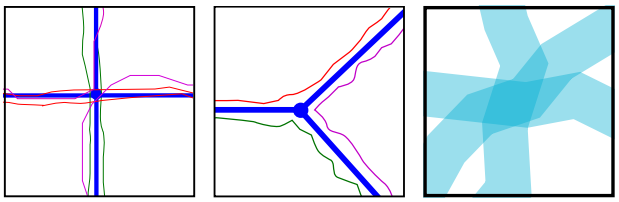
$$P = (1-\beta)P_1 + \beta P_2 \,, \tag{1}$$

where $\beta \in [0,1]$ represents the percentage of "salt and pepper" noise. Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of n independent and identically distributed samples from P. In this section, we define the density $\hat{p}$ over $\widetilde{D}$ as a normalized histogram and we analyze properties of $\hat{p}(q)$ for very large n.

We show that as the number of samples n tends to infinity, the bound $\tau_1 \le \hat{p}(q) \le \tau_2$ can be computed very tightly and with high probability using Chebyshev's inequality. Then, one or multiple thresholds are picked that separate $\widetilde{G}$ and $\widetilde{D} - \widetilde{G}$, or different sub-classes of pixels on $\widetilde{G}$. We prove that the number of thresholds necessary to separate the classes is a constant, which depends on the vertex-degrees, and in a more general case on the number of different road categories.

We now consider different sampling cases for P_1 and describe how to compute such thresholds. In Section 4.1 we consider the simplest case where the graph is uniformly sampled. In Section 4.2 we consider a more realistic sampling case that samples uniformly along trajectories. Consider, for example, a 4-way intersection as in Figure 4a, where l trajectories go from west to east and l from north to south. In this case, the mid-sections of streets are sampled l times, but the pixels in the actual intersection sampled $2l$ times. This cases the street pixels to be sampled uniformly but intersections are sampled more often.

4.1 Uniform Sampling



(a) 4-way intersection. (b) 3-way intersection. (c) Different intensity levels.

Figure 4: (a),(b) Intersections are more heavily sampled than mid-sections of streets. (c) Darker shades of blue indicate higher density. Different density levels are introduced when two (or more) streets overlap.

Let $X = \{X_1, \ldots, X_n\}$ be the set of i.i.d. samples from the distribution P, as defined in (1). Then, for a fixed pixel $q \in \widetilde{D}$, this sampling is a Bernoulli trial with success probability $p_1 = p_1(q)$, i.e., p_1 is the probability to sample q and $1 - p_1$ is the probability to sample some other pixel. More formally, the indicator random variables $X_i^q = [X_i = q]$, for $1 \le i \le n$, are Bernoulli. Let $Y^q = \sum_{i=1}^{n} X_i^q$ be the number of times q is sampled. We define the estimated density $\hat{p}(q) = \frac{1}{n} Y^q$, and observe that Y^q follows the binomial distribution $B(n, p_1)$. Thus, we have the following two formulas for μ and σ:

$$\mu = E[\hat{p}(q)] = \frac{1}{n} E[Y^q] = \frac{1}{n} n p_1 = p_1$$

and

$$\sigma^2 = \mathrm{Var}[\hat{p}(q)] = \frac{1}{n^2} \mathrm{Var}[Y^q] = \frac{1}{n^2} n p_1 (1-p_1) = \frac{p_1}{n}(1-p_1).$$

Applying Chebyshev's inequality, we obtain the inequality $\mathbb{P}(|\hat{p}(q) - \mu| \ge \varepsilon) \le \sigma^2/\varepsilon^2$, or

$$\mathbb{P}(\mu - \varepsilon < \hat{p}(q) < \mu + \varepsilon) \ge 1 - \frac{\sigma^2}{\varepsilon^2} \,. \tag{2}$$

We will use (2) to obtain thresholds $\tau \le \mu - \varepsilon$ or $\tau \ge \mu + \varepsilon$ to classify a pixel q correctly with confidence $1 - \frac{\sigma^2}{\varepsilon^2}$. In order to compute τ we need to estimate μ and ε from the input data. Here, $\mu = p_1$, so it remains to estimate p_1 and ε. Let $\widetilde{D}^{>0} = \hat{p}^{-1}([1,\infty]) = \{q \in \widetilde{D} \mid \hat{p}(q) > 0\}$. Because of the mixed distribution (1) there are two different cases, namely $q \in \widetilde{G}$ and $q \in \widetilde{D} - \widetilde{G}$.

For the first case, let $q \in \widetilde{G}$. Then, from the mixed distribution with noise percentage β it follows that $p_1 = \frac{1-\beta}{|\widetilde{G}|}$ and we estimate $\hat{p}_1 = \frac{1-\beta}{|\widetilde{D}^{>0}|} = \hat{\mu}$. Observe that for $\beta = 0$, all samples are drawn from $\widetilde{G}$ and hence $p_1 = \frac{1}{|\widetilde{G}|}$ and $\hat{p}_1 = \hat{\mu} = \frac{1}{|\widetilde{D}^{>0}|}$. We estimate ε by setting the confidence $\alpha = 1 - \frac{\sigma^2}{\varepsilon^2}$ as required by (2), which yields $\varepsilon = \sqrt{\frac{\sigma^2}{1-\alpha}}$. And from $\sigma^2 = \frac{1}{n} p_1 (1-p_1)$ we estimate $\hat{\sigma} = \frac{(1-\beta)}{n|\widetilde{D}^{>0}|}(1 - \frac{1-\beta}{|\widetilde{D}^{>0}|})$. Hence, we correctly recognize $q \in \widetilde{G}$ as being in $\widetilde{G}$ if we choose a threshold at most $\hat{\mu} - \hat{\varepsilon}$.

In the second case we have $q \in \widetilde{D} - \widetilde{G}$. Using a similar analysis as in the other case, we estimate $\hat{\mu} = \hat{p}_1 = \frac{\beta}{|\widetilde{D}| - |\widetilde{D}^{>0}|}$, $\hat{\varepsilon} = \sqrt{\frac{\hat{\sigma}^2}{1-\alpha}}$, and $\hat{\sigma}^2 = \frac{\beta}{n(|\widetilde{D}| - |\widetilde{D}^{>0}|)}(1 - \frac{\beta}{|\widetilde{D}| - |\widetilde{D}^{>0}|})$ for given α and β. And we correctly recognize $q \in \widetilde{D} - \widetilde{G}$ as being in $\widetilde{D} - \widetilde{G}$ if we choose a threshold at least $\hat{\mu} + \hat{\varepsilon}$.

Combining the two different classes of pixels we can pick a common threshold τ, and we obtain the following theorem.

Theorem 4.1. *Let* $\beta \in [0,1]$ *be the noise percentage, and* $\alpha \ge 0$ *be the desired confidence. Let* $X = \{X_1, X_2, \ldots, X_n\}$ *be* n *samples drawn i.i.d. from a mixed distribution as described in (1). Let* $|\widetilde{D}^{>0}|$ *be the number of different pixels that have been sampled. Then each pixel* $q \in \widetilde{D}$ *can be correctly classified with confidence* β *by using any threshold* $\tau \in [\hat{\mu}_2 + \hat{\varepsilon}_2, \hat{\mu}_1 - \hat{\varepsilon}_1]$*, where*

$\hat{\mu}_1 = \frac{1-\beta}{|\widetilde{D}^{>0}|}$, $\hat{\varepsilon}_1 = \sqrt{\frac{\hat{\sigma}_1^2}{1-\alpha}}$, $\hat{\sigma}_1^2 = \frac{1-\beta}{n|\widetilde{D}^{>0}|}(1 - \frac{1-\beta}{|\widetilde{D}^{>0}|})$,

$\hat{\mu}_2 = \frac{\beta}{|\widetilde{D}| - |\widetilde{D}^{>0}|}$, $\hat{\varepsilon}_2 = \sqrt{\frac{\hat{\sigma}_2^2}{1-\alpha}}$, $\hat{\sigma}_2^2 = \frac{\beta}{n(|\widetilde{D}| - |\widetilde{D}^{>0}|)}(1 - \frac{\beta}{|\widetilde{D}| - |\widetilde{D}^{>0}|})$

Note that as n tends to infinity we have

$$\lim_{n\to\infty}\hat{\mu}_1 = \frac{1-\beta}{|\widetilde{G}|} \qquad \lim_{n\to\infty}\hat{\mu}_2 = \frac{\beta}{|\widetilde{D}|-|\widetilde{G}|}$$

$$\lim_{n\to\infty}\hat{\varepsilon}_1 = \lim_{n\to\infty}\hat{\varepsilon}_2 = 0$$

Therefore, as long as the number of noisy pixels is sufficiently small, i.e., β is picked such that $\frac{\beta}{|\widetilde{D}|-|\widetilde{G}|} < \frac{1-\beta}{|\widetilde{G}|}$, then there is a large enough n such that $\hat{\mu}_2+\hat{\varepsilon}_2 < \hat{\mu}_1-\hat{\varepsilon}_1$ and the different pixel classes are separated. In words, we can choose one threshold in $(\hat{\mu}_2+\hat{\varepsilon}_2, \hat{\mu}_1-\hat{\varepsilon}_1)$ in order to classifiy the pixels as in $\widetilde{G}$ or not.

4.2 Uniform Trajectory Sampling

In the uniform sampling case we assumed all pixels in $\widetilde{G}$ are equally likely, i.e, $p_1 = \frac{1-\beta}{|\widetilde{G}|}$. But in practice, the position samples on the street-maps originate from vehicle trajectories and therefore intersections are sampled more often than mid-sections of streets. Consider the four-way intersection shown in Figure 4a. Here, if each street has three position samples close to the intersection area then there are six position samples in total. A similar analysis holds for a three-way intersection (Figure 4b); here, each street is covered by two samples and the intersection area is covered by three trajectories.

In this section, we assume that each input trajectory is uniformly sampled along its path in G, and $X = \{X_1, \ldots, X_n\}$ is the set of these samples (over all trajectories). And we assume that the embedding of G splits $\widetilde{G}$ into street pixels and intersection pixels as follows: For $d \geq 3$, $\widetilde{I}_d \subseteq \widetilde{G}$ is the set of *intersection pixels* representing vertices of degree d, and $\widetilde{S} \subseteq \widetilde{G}$ is the set of *street pixels*, where street pixels include vertices of degree one or two. Then $\widetilde{G}$ is the disjoint union $\widetilde{G} = \widetilde{S} \cup \bigcup_{d=3}^{k} \widetilde{I}_d$, where k is the maximum degree of G. This distinction between intersection pixels and street pixels naturally arises from the embedding. And in our uniform trajectory sampling model, each of these classes will have a different sampling probability.

The uniform trajectory sampling model with noise percentage β implies the following generalized definition of p_1 for a fixed pixel $q \in \widetilde{G}$:

$$p_1 = p_1(q) = \begin{cases} (1-\beta)\rho, & \text{if } q \in \widetilde{S} \\ (1-\beta)\frac{d}{2}\rho, & \text{if } q \in \widetilde{I}_d \text{ for any } d \geq 3 \end{cases}$$

Thus, pixels in $\widetilde{I}_d$ are $\frac{d}{2}$-times more likely to be sampled than pixels in $\widetilde{S}$, which yields $\rho = (|\widetilde{S}| + \sum_{d=3}^{k}\frac{d}{2}|\widetilde{I}_d|)^{-1}$. Although the neighborhood of a vertex consists of multiple small areas with different density levels (see Figure 4c), we model only a single density level in the intersection area.

In order to estimate $|\widetilde{S}|$ and $|\widetilde{I}_d|$, and with that ρ and p_1, we need to classify the pixels of $\widetilde{D}^{>0}$ into *street pixels* $\widetilde{D}_S$, and *degree-d intersection pixels* $\widetilde{D}_{I_d}$. Here, we use an algorithm similar to [1]. We consider $\widetilde{D}^{>0}$ as a grid graph with an edge (q_1, q_2) between pixels $q_1, q_2 \in \widetilde{D}^{>0}$ iff q_1 and q_2 are vertically or horizontally adjacent. For each pixel q, let c_q be the number of connected components of $\widetilde{D}^{>0} \cap (B_R(q) - B_r(q))$ for parameters $R > r > 0$, see Figure 5. Here $B_r(q)$ is the ball centered at the (center of) q with radius r. Then we define $\widetilde{D}_S = \{q \in \widetilde{D}^{>0} \mid c_q \leq 2\}$ and $\widetilde{D}_{I_d} = \{q \in \widetilde{D}^{>0} \mid c_q = d\}$ for $3 \leq d \leq k$.

This yields $\hat{\rho} = (|\widetilde{D}_S| + \sum_{d=3}^{k}\frac{d}{2}|\widetilde{D}_{I_d}|)^{-1}$. For street pixels we then have $\hat{\mu}_S = (1-\beta)\hat{\rho}$, $\hat{\varepsilon}_S = \sqrt{\frac{\hat{\sigma}_S^2}{1-\alpha}}$ and $\hat{\sigma}_S^2 = \frac{1}{n}\hat{\mu}_S(1-\hat{\mu}_S)$. For intersection pixels with degree d we have $\hat{\mu}_{I_d} = (1-\beta)\frac{d}{2}\hat{\rho}$, $\hat{\varepsilon}_{I_d} = \sqrt{\frac{\hat{\sigma}_{I_d}^2}{1-\alpha}}$ and $\hat{\sigma}_{I_d}^2 = \frac{1}{n}\hat{\mu}_{I_d}(1-\hat{\mu}_{I_d})$. $\hat{\mu}_2$ and $\hat{\varepsilon}_2$ stay the same as in the uniform sampling case described in Theorem 4.1. This yields the following result.

Theorem 4.2. *If $\hat{\mu}_2+\hat{\varepsilon}_2 < \hat{\mu}_S-\hat{\varepsilon}_S$, $\hat{\mu}_S+\hat{\varepsilon}_S < \hat{\mu}_{I_3}-\hat{\varepsilon}_3$, $\hat{\mu}_{I_3}+\hat{\varepsilon}_3 < \hat{\mu}_{I_4}-\hat{\varepsilon}_4$, $\ldots$, $\hat{\mu}_{I_{k-1}}+\hat{\varepsilon}_{k-1} < \hat{\mu}_{I_k}-\hat{\varepsilon}_k$, then the set of thresholds $T = \{\hat{\mu}_2+\hat{\varepsilon}_2, \hat{\mu}_S-\hat{\varepsilon}_S, \hat{\mu}_{I_3}-\hat{\varepsilon}_3, \ldots, \hat{\mu}_{I_k}-\hat{\varepsilon}_k\}$ induces a partition that correctly classifies each pixel $q \in \widetilde{D}$ either as a street pixel, an intersection pixel of degree d, or not on the graph. Here, k is the maximum degree of G.*

Note that, as n tends to infinity, we have

$$\lim_{n\to\infty}\hat{\mu}_2 = \frac{\beta}{|\widetilde{D}|-|\widetilde{G}|} \qquad \lim_{n\to\infty}\hat{\mu}_S = \frac{1-\beta}{|\widetilde{S}| + \sum_{d=3}^{k}\frac{d}{2}|\widetilde{I}_d|}$$

$$\lim_{n\to\infty}\hat{\mu}_{I_d} = \frac{d}{2}\frac{1-\beta}{|\widetilde{S}| + \sum_{d=3}^{k}\frac{d}{2}|\widetilde{I}_d|}$$

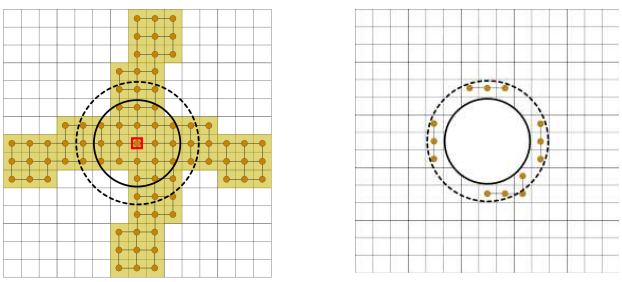
$$\lim_{n\to\infty}\hat{\varepsilon}_2 = \lim_{n\to\infty}\hat{\varepsilon}_S = \lim_{n\to\infty}\hat{\varepsilon}_{I_3} = \ldots = \lim_{n\to\infty}\hat{\varepsilon}_{I_k} = 0$$

Therefore, as long as the number of noisy pixels is sufficiently small, i.e., β is picked such that $\frac{\beta}{|\widetilde{D}|-|\widetilde{G}|} < \frac{1-\beta}{|\widetilde{S}|+\sum_{d=3}^{k}\frac{d}{2}|\widetilde{I}_d|}$ then there is a large enough n such that $\hat{\mu}_2+\hat{\varepsilon}_2 < \hat{\mu}_S-\hat{\varepsilon}_S < \hat{\mu}_{I_3}-\hat{\varepsilon}_3 < \ldots < \hat{\mu}_{I_k}-\hat{\varepsilon}_k$ and the pixel classes are separated. Note that a constant number of $|T| = k$ tresholds suffice to classify the pixels. This approach can also be generalized to multiple road categories.

4.3 Variant II of Algorithm 1

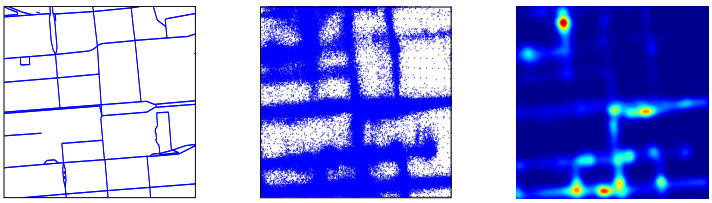
We now explain how our proposed techniques from this section fits within the general framework of Algorithm 1 for map construction. In order to fully describe the map construction algorithm, we must explicitly state how we are performing Lines 1, 2, and 5. The density estimate for Line 1 depends on if we are in the uniform sampling model or the uniform trajectory sampling models and have been described in Section 4.1 and Section 4.2 respectively.

Now consider how to compute the set of trajectories in Line 2 of Algorithm 1. This also depends on the sampling model, but the scenarios are quite similar. First, consider the uniform sampling model. Theorem 4.1 implies that with confidence $1-\alpha$, the set $\widetilde{G}$ is exactly the set of all pixels with



(a) $B(q,r)$ and $B(q,R)$ (b) $q \in I_4$

Figure 5: Partition of pixels.



(a) Ground-truth (b) Berlin-taxi (c) KDE

Figure 6: We show the ground truth (left) for the Berlin-taxi data set (middle). The graph structure is apparent in the corresponding KDE (right).

density larger than τ, for any $\tau \in [\hat{\mu}_2 + \hat{\varepsilon}_2, \hat{\mu}_1 - \hat{\varepsilon}_1]$. Therefore, for Line 2, we let T be any threshold in $[\hat{\mu}_2+\hat{\varepsilon}_2, \hat{\mu}_1-\hat{\varepsilon}_1]$. Then, for Line 5, we simply let our final discretized graph be the set of all pixels with density larger than τ.

For uniform trajectory sampling, we want to not only construct a map, but also to classify the map pixels. To do so, we choose our thresholds for Line 2 and determine the final partitioning of the pixels according to Theorem 4.2.

5. EXPERIMENTS

In this section, we apply our proposed thresholding techniques presented in Sections 3 and 4. We use two different types of input datasets: actual vehicular tracking data in the *Berlin-taxi* [15] dataset and the artificially generated *Berlin-synthetic-1* and *Berlin-synthetic-2* datasets. After computing the set of thresholds we obtain a set of thresholded images, and for visualization purposes we use the thinning algorithm in [8] to compute thinned skeletons.

5.1 Datasets

We have three different data sources. We obtained a ground-truth map of a $600m \times 600m$ region of the downtown of Berlin, Germany from OpenStreetMap.org (April 2013 database); see Figure 6a. The Berlin-taxi dataset is actual taxi-tracking data consisting of $292,997$ point samples; see Figure 6b. The sampling rate for this dataset is every 60-180s. The third type of data we generated artificially using two different sampling techniques. Berlin-synthetic-1 consists of $3,505,306$ points sampled uniformly from the fattened ground-truth graph. Here, we fattened the graph by 10 meters each side; i.e, every street is 20 meters wide. We incorporated 2% salt and pepper noise by sampling 2% unique pixels outside the graph area; see Figure 8a. Berlin-synthetic-2 consists of $854,120$ point samples originating from $2,000$ trajectories. A trajectory sample is generated using brownian-bridge motion (with variance 10 meters) between two consecutive points in a path of the ground-truth map; see Figure 9a. And a path is selected uniformly from a set of $k-$link-length paths (a $k-$link length path consists of $k+1$ intersection vertices). For this dataset we use k between one to eight.

5.2 Experiments with Real Data

We computed a KDE using a Gaussian kernel[1] with variance $\sigma^2 = 10$ for the Berlin-taxi dataset. Then, we computed persistence diagrams and extracted three different persistence cut-offs 0, $5 \cdot 10^{-8}$, and $3 \cdot 10^{-7}$ that defined threshold sets T_2, T_3, and T_4, respectively; see Table 1. Threshold set T_1 is the default set used by the authors in [8].

[1]To compute KDEs and persistence diagrams, we use the Topological Data Analysis 'TDA' package written in R. For a 600×600 grid, this computation took about 1.5 hours.

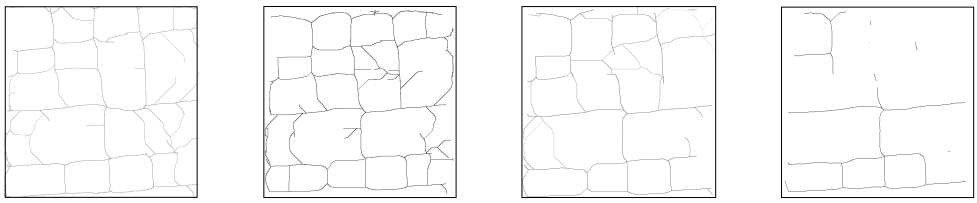
Threshold Set	Persistence Cut-off	# of Thresholds	# of Cycles	Runtime (mins)
T_1	n/a	28	18	83
T_2	0	22	20	72
T_3	$5.0 * 10^{-8}$	4	14	18
T_4	$3.0 * 10^{-7}$	1	2	1

Table 1: Result of persistence based thresholding. Threshold set T_1 is arbitrarily chosen and sets T_2, T_3, and T_4 are computed using Algorithm 2. We observe that set T_3 produces the correct number of cycles (there are, in facat, 14 cycles in the ground truth) and has significant speedup over sets T_1 and T_2.

After computing a threshold set T, for visualization purposes we plug the KDE image and T into the skeletonization algorithm of [8], which runs a thinning algorithm on each individual thresholded image and combines the results into a single skeleton. The resulting skeletons are shown in Figure 7. We observe that our ground-truth map (Figure 6a) has 14 cycles and the skeleton computed using T_1 has 18, our three threshold sets produces skeletons with 20, 14 and 2 cycles, respectively. Threshold set T_2 has six less thresholds but produces a very similar result to the skeleton generated using T_1. We also observe that our persistence cut-off based noise removal is very effective as T_3 produces skeleton that is very similar to the ground-truth (see Figure 6a) and has less noise compared to the skeletons computed using T_1 and T_2. Each additional threshold used in the skeltonization step takes a substantial amount of runtime, so our algorithm runs quite fast compared to the algorithm using the full set of thresholds.

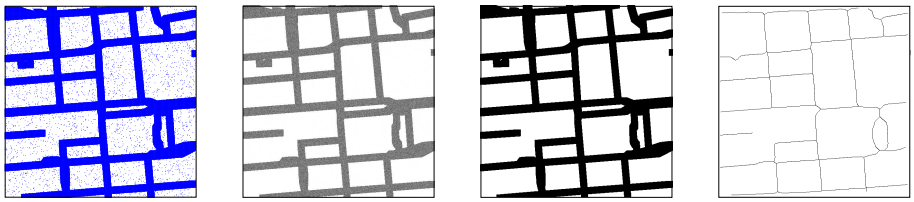
5.3 Uniform Sampling Case

In this part of the experiments we verify our thresholding technique described in Section 4. We use two artificially generated datasets Berlin-synthetic-1 and Berlin-synthetic-2. In each case we use confidence level $\alpha = 0.90$. Our results for the uniform point sampling case using Berlin-synthetic-1 is summarized in Figure 8. Here, we compute the histogram and compute the threshold using the formula given in Theorem 4.1. In this case we use $2.9570 \cdot 10^{-6}$, here $\hat{\mu}_1 = 7.6192 \cdot 10^{-6}$ and $\hat{\varepsilon}_1 = 4.6622 \cdot 10^{-6}$ and we picked $\hat{\mu}_1 + \hat{\varepsilon}_1$ as our threshold. We observe that using just a single threshold computed with our algorithm results in a



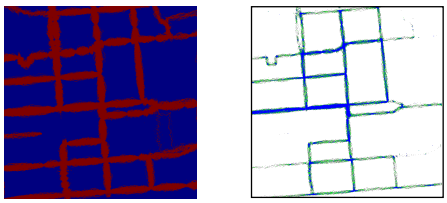
(a) Thresholds 1 (b) Thresholds 2 (c) Thresholds 3 (d) Thresholds 4

Figure 7: Skeletons computed using four different sets of thresholds.



(a) Input (b) Density Image (c) Thresholded Image (d) Skeleton

Figure 8: The Berlin-synthetic-1 dataset has 2% salt and pepper noise.



(a) Berlin-Synth.-2 (b) Thresholded

Figure 9: Berlin-Synthetic-2 dataset; point-samples originated from trajectories. On the right hand side we see intersections are classified as pixels with high density.

threshold image of very good quality (see Figure 8c) as well as a high-quality skeleton (see Figure 8d). To compare the quality of the thresholded image we compute the Dice coefficient between the fattened ground-truth image (discretized in a 1×1 grid) and the thresholded image. Intuitively the Dice coefficient gives a way of measuring the accuracy of our selected pixels with the ground truth road map where 1 is perfect and 0 means all pixels were classified incorrectly. We found at 90% confidence that the Dice coefficient is 0.9999, which suggests our thresholded image is very close to the ground-truth. For confidence levels 80%, 50% and 20% the Dice coefficient is 0.9971, 0.9735 and 0.9594, respectively.

Next, we use the Berlin-Synthetic-2 dataset to verify if our thresholding technique can classify pixels into different classes (streets vs. intersections). The dataset and results are summarized in Figure 9. Here, first we computed a histogram for the input data and applied four thresholds, as described in Theorem 4.2. The value of thresholds are $1.2281 \cdot 10^{-5}$, $1.4639 \cdot 10^{-5}$, $1.9518 \cdot 10^{-5}$ and $2.4398 \cdot 10^{-5}$ to classify streets and intersections with degree three, four and five respectively. We observe that streets and intersections are well separated from the background, and the intersections are generally well separated from the streets. To motivate the use of a sampling model with multiple road categories, we observe that some streets are sampled more frequently than others, such as the street in the middle of Figure 3b.

6. CONCLUSION

Previous research [8, 11, 12, 26] has proposed map construction algorithms utilizing densities as the key tool. These algorithms follow the general outline of Algorithm 1, described in Section 2.1, and pick one or more thresholds to extract graph pixels from the density. Most of the previous results use heuristics verified by experiments.

In this paper, we propose two methods that formalize the computation of a small set of carefully-chosen thresholds to be used within the framework of Algorithm 1 so that the constructed map will satisfy desirable properties. Our first approach is based on persistent homology with the goal of computing a small set of density thresholds T that will capture all relevant features. Our set T is essentially a subset of the thresholds used by [8], decreasing the runtime of Algorithm 1 by a factor of four in practice, while still producing a map of similar quality. We proposed computing a minimum cardinality set of thresholds which collectively captures all significant features, but we remark that there are other objective functions that one might want to consider. For example, one might want to fix the number of thresholds used and then maximize the number features captured. Similarly, one might want to assign weights to each point according to their significance and then maximize the weights of the covered points using a fixed number of thresholds. Such variants can also be computed efficiently using dynamic programming. Our second approach provides formal guarantees on the number of thresholds needed when the samples follow the uniform distribution model or the uniform trajectory model. In these settings, we prove that one can compute thresholds that can classify pixels not only

in a binary manner (i.e., on/off the street), but additionally classify street pixels according to their sampling density (i.e., street or intersection).

There are several directions for future work. One would be to strengthen the claims made in this paper. We provide a first look at providing formal proofs regarding the properties of a set of thresholds with certain assumptions on the sampling distribution. It would be interesting to provide similar guarantees in less-restrictive scenarios. persistent homology, it would be interesting to determine if the persistence cut-offs could be chosen in a statistically sound manner. Another area of future work would be to investigate new methods for combining multiple super-level sets in a map construction algorithm.

7. REFERENCES

[1] Aanjaneya, M., Chazal, F., Chen, D., Glisse, M., Guibas, L. J., and Morozov, D. Metric graph reconstruction from noisy data. In *Proc. SoCG* (2011), pp. 37–46.

[2] Ahmed, M., Fasy, B. T., and Wenk, C. Local persistent homology based distance between maps. In *ACM SIGSPATIAL* (2014), pp. 43–52.

[3] Ahmed, M., Karagiorgou, S., Pfoser, D., and Wenk, C. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica 19*, 3 (2015), 601–632.

[4] Ahmed, M., and Wenk, C. Constructing street networks from GPS trajectories. In *Proc. European Symp. Alg.* (2012), pp. 60–71.

[5] Alexeev, V., Bogaevskaya, V., Preobrazhenskaya, M., Ukhalov, A., Edelsbrunner, H., and Yakimova, O. An algorithm for cartographic generalization that preserves global topology. *J Math. Sci. 203*, 6 (2014), 754–760.

[6] Alt, H., Efrat, A., Rote, G., and Wenk, C. Matching planar maps. *J. Alg.* (2003), 262–283.

[7] Biagioni, J., and Eriksson, J. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *J. Trans. Research Board 2291* (2012), 61–71.

[8] Biagioni, J., and Eriksson, J. Map inference in the face of noise and disparity. In *ACM SIGSPATIAL* (2012), pp. 79–88.

[9] Brakatsoulas, S., Pfoser, D., Salas, R., and Wenk, C. On map-matching vehicle tracking data. In *Proc. VLDB Conf.* (2005), pp. 853–864.

[10] Cao, L., and Krumm, J. From GPS traces to a routable road map. In *Proc. ACM SIGSPATIAL* (2009), pp. 3–12.

[11] Chen, C., and Cheng, Y. Roads digital map generation with multi-track GPS data. In *Proc. Workshops Edu. Tech. Training Geoscience Remote Sensing* (2008), IEEE, pp. 508–511.

[12] Davies, J. J., Beresford, A. R., and Hopper, A. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing 5*, 4 (Oct. 2006), 47–54.

[13] Dirnberger, M., Neumann, A., and Kehl, T. NEFI: Network Extraction From Images. arXiv 1502.05241, 2015.

[14] Edelkamp, S., and Schrödl, S. Route planning and map inference with global positioning traces. In *Comput. Sci. Persp.* Springer, 2003, pp. 128–151.

[15] Efentakis, A., Brakatsoulas, S., Grivas, N., Lamprianidis, G., Patroumpas, K., and Pfoser, D. Towards a flexible and scalable fleet management service. In *ACM SIGSPATIAL* (2013), pp. 79–84.

[16] Fasy, B. T., Kim, J., Lecci, F., and Maria, C. Introduction to the R package TDA. arXiv 1411.1830, 2014.

[17] Fasy, B. T., Lecci, F., Rinaldo, A., Wasserman, L., Balakrishnan, S., and Singh, A. Confidence sets for persistence diagrams. *Ann. Statistics 42*, 6 (2014), 2301–2339.

[18] Ge, X., Safa, I., Belkin, M., and Wang, Y. Data skeletonization via Reeb graphs. In *Proc. Ann. Conf. Neural Inf. Proc. Syst.* (2011), pp. 837–845.

[19] Karagiorgou, S., and Pfoser, D. On vehicle tracking data-based road network generation. In *ACM SIGSPATIAL GIS* (2012), pp. 89–98.

[20] Letscher, D., and Fritts, J. Image segmentation using topological persistence. *Computer Analysis of Images and Patterns 4673* (2007), 587–595.

[21] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., and Huang, Y. Map-matching for low-sampling-rate GPS trajectories. In *ACM SIGSPATIAL GIS* (2009), pp. 352–361.

[22] Newson, P., and Krumm, J. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL* (2009), pp. 336–343.

[23] Quddus, M., Ochieng, W., and Noland, R. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Trans. Research Part C: Emerging Tech.* (2007), 312–328.

[24] Silverman, B. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[25] Skraba, P., Ovsjanikov, M., Chazal, F., and Guibas, L. Persistence-based segmentation of deformable shapes. In *Proc. IEEE Comput. Vis. Pattern Recognition Workshops* (June 2010), pp. 45–52.

[26] Steiner, A., and Leonhardt, A. Map generation algorithm using low frequency vehicle position data. In *Proc. Ann. Meeting Trans. Research Board* (January 2011), pp. 1–17.

[27] Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., and Eriksson, J. Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Proc. ACM Conf. Embed. Netw. Sens. Sys.* (2009), pp. 85–98.

[28] Wang, Y., Liu, X., Wei, H., Forman, G., Chen, C., and Zhu, Y. Crowdatlas: Self updating maps for cloud and personal use. In *Proc. Int. Conf. Mobile Sys. App. Services* (2013).

[29] Wasserman, L. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer, 2013.