ST-COPOT: Spatio-temporal Clustering with Contour Polygon Trees

Yongli Zhang and Christoph F. Eick

Department of Computer Science, University of Houston, Houston, TX, USA
yzhang93@uh.edu ceick@uh.edu

Abstract. Spatio-temporal clustering of data streams aims to discover interesting regions in large spatio-temporal data streams efficiently using a small amount of memory and time. In this paper, we propose a serial, density-contour based clustering algorithm called ST-COPOT; the proposed algorithm employs a non-parametric density estimation approach and contouring algorithms to obtain spatial clusters from point cloud streams and spatio-temporal clusters are then formed by identifying continuing relationships between spatial clusters in consecutive time frames. In particular, our approach subdivides the incoming data into batches; next, for each batch spatial clusters are generated as regions which are enclosed by polygons that correspond to given set of density thresholds; finally, spatio-temporal clusters are generated by analyzing continuity between spatial clusters in consecutive batches. Moreover, the paper introduces a unique data structure called contour polygon tree which provides a compact representation of the spatial clusters obtained for each batch for different density thresholds and proposes a family of novel distance functions that operate on contour polygon trees to identify continuing clusters. We evaluate our approach by conducting a case study involving NYC taxi trips data. The experimental results show that ST-COPOT can effectively discover interesting spatio-temporal patterns in taxi pickup location streams.

1 Introduction

Spatio-temporal clustering is a major research field of spatio-temporal data mining and knowledge discovery, which aims to identify dense regions of spatio-temporal entities and to discover interesting spatio-temporal patterns associated with each region [1]. Moreover, as the data generation and querying rates of spatio-temporal datasets increase, it comes to the problem of geo-streaming, which processes and analyzes stream data with spatial attributes [2].

Due to the advances in remote sensors and sensor networks, different types of spatio-temporal datasets become increasingly available these days. Revealing interesting spatio-temporal patterns from such datasets is very important as it has broad applications, such as understanding climate change [3], identifying crime patterns [4], epidemics detection [5], flood risk analysis [6], and earthquake analysis [7]. Moreover, nowadays, geospatial applications such as location based
services (LBS) and Intelligent Transportation System (ITS) have been widely used. In general, geostreaming applications are growing both in quantity and scale due to recent advancements in sensing technology and the increased popularity of social media and smart phones [2]. As a result, there is an exponential growth in data generation and querying rates for these data, highlighting the importance of efficient techniques for geostreaming.

When it comes to processing geo-tagged data streams, we face the following major stream processing challenges: querying, analysis and integration, scalability, extensibility, one-time access to data, volume, and real-time analysis [8]. Density-based stream clustering algorithms have been broadly investigated in literature and have been categorized into two broad groups: density micro-clustering algorithms and density grid-based clustering algorithms [9]. However, our proposed approach ST-COPOT, as a density-contour based approach, is neither a strictly micro-clustering nor a strictly grid-based clustering approach, which will be further discussed in Section 2.

In this paper, we propose a novel spatio-temporal clustering approach for spatio-temporal point cloud streams called ST-COPOT. Our approach subdivides the incoming data into batches; then, for each batch spatial clusters are generated as regions whose area is enclosed by a polygon that corresponds to a certain density threshold. Our approach generates such contour polygons for multiple density thresholds, and a data structure called contour polygon tree is used as a compact representation of the clustering results for each batch. A similar data structure called density contour trees was first introduced and briefly discussed in [10], however, as far as we know, no further methodologies and applications have been investigated and developed in terms of this data structure since its publication. The root of each tree stores a polygon that has been generated using the lowest density threshold, and the remainder of the tree stores polygons that are contained in the polygon of the higher levels of the tree. Fig-

---

Fig. 1: Example and visualization of contour polygon trees
Figure 1 gives an example of two contour polygon trees that have been generated and depicts their associated polygons, as well. Finally, spatio-temporal clusters are formed by identifying continuing relationships between spatial clusters in consecutive batches; this is done by analyzing the respective contour polygon trees.

Besides, ST-COPOT is an enhanced version of ST-DPOLY [11], which uses only one density threshold. Since ST-COPOT uses multi-thresholds, it is more fault tolerant in terms of density threshold selection. Moreover, using multiple density thresholds can also help us to identify and analyze regional patterns, for example, we can discover local hotspots inside a regional hotspot.

This paper’s main contributions include:

- We propose a serial, density-contour based spatio-temporal clustering algorithm ST-COPOT which can generate spatio-temporal clusters from point cloud streams.
- We propose a data structure called contour polygon tree as a compact representation of clustering results for each batch. A family of novel distance functions that operate on contour polygon trees is proposed to identify continuing clusters.
- Moreover, the proposed approach is capable of creating spatio-temporal clusters at different levels of granularity, e.g. continuing polygons, continuing trees, continuing forests.
- ST-COPOT meets the one-time access requirement for streaming data processing, as the data in each batch are only read once, and it is capable of processing very large data streams.
- We evaluate our approach in a challenging real-world case study involving NYC taxi trips data. The experimental results show that ST-COPOT can effectively discover interesting spatio-temporal patterns in taxi pickup location streams.

The rest of the paper is organized as follows. Section 2 introduces related work about spatio-temporal clustering and clustering from data streams. Section 3 introduces the serial, density-contour based spatio-temporal clustering algorithm ST-COPOT in detail. Section 4 demonstrates our approach using NYC taxi trips data. Section 5 concludes the paper and discusses potential future work.

2 Related Work

Spatio-temporal clustering and hotspot discovery techniques for point objects have been well studied in the literature. Kulldorff et al. [12] introduce a spatial scan statistic for the detection of spatio-temporal cylinders where the point objects occur consistently for a significant period of time. Iyengar et al. [13] extend

\footnote{Using solely one density threshold, sometimes we might obtain a poor clustering result, and more importantly, it does not allow to view a dataset at different granularities.}
the basic scan statistics using the flexible square pyramid shape to detect clusters with restrictive shapes, and the proposed framework can model growth and shifts in location over time. Wang et al. [14] propose a spatio-temporal clustering algorithm ST-GRID which maps the spatial and temporal dimensions into multidimensional cells and then extract and merge spatio-temporal dense regions to obtain a final cluster. Birant et al. [15] propose ST-DBSCAN as an extension of DBSCAN for spatio-temporal clustering by introducing a second parameter of temporal neighborhood radius in addition to the spatial neighborhood radius. Wang et al. propose a spatio-temporal clustering approach, ST-SEP-SNN [16], which combines spatial and temporal distances into a joint spatio-temporal distance function, and then generalizes the SNN clustering algorithm to operate on the obtained joint distance function.

However, most existing spatio-temporal clustering algorithms mentioned above, are not suitable to deal with large data streams. For example, both ST-DBSCAN and ST-SEP-SNN pass over the data several times and assume that the data fit into main memory. However, a few density-based clustering algorithms have been proposed in the literature for the data streams.

Cao et al. [17] propose a density-based stream clustering approach which has the following features: no assumption on the number of clusters, the discovery of clusters with arbitrary shape and ability to handle outliers. Chen et al. [18] propose a framework for clustering stream data, which uses an online component which maps input data into a grid and an offline component which computes the grid density and clusters the grids. Wan et al. [19] also propose an online-offline approach, which is able to detect arbitrarily shaped, evolving clusters with high quality.

Overall, density-based stream clustering algorithms are categorized into two broad groups: density micro-clustering algorithms and density grid-based clustering algorithms [9]. In density micro-clustering algorithms, micro-clusters keep synopsis information about stream data, then clustering is performed on the summary information, example algorithms in this category: DenStream [17], HDenStream [20], SOSStream [21], and PreDeConStream [22]. In density grid-based clustering algorithms, the data space is divided into grids, then the clusters are formed based on the density of grids, example algorithms in this category are: D-Stream [18], PKS-Stream [23], DENGRIS-Stream [24], and ExCC [25].

However, our proposed approach ST-COPOT is neither strictly a micro-clustering nor strictly a grid-based clustering approach. ST-COPOT uses grids, but only for the contouring algorithm but not for density estimation. To the best of our knowledge, our proposed approach is the first approach that uses density contours and contour analysis to create spatio-temporal clusters from spatio-temporal data streams.
3 ST-COPOT

3.1 Overview of ST-COPOT

ST-COPOT creates spatio-temporal clusters in a serial fashion and its clustering approach consists of the following 3 phases:

1. Obtain a spatial density function through non-parametric density estimation for the spatial point cloud collected in each batch.
2. For each batch, ST-COPOT uses multiple density thresholds to identify spatial clusters which are enclosed by a polygon that is constructed from density contour lines of the spatial density function created in phase 1.
3. Identify continuing relationships between contour polygon trees in consecutive batches, and construct spatio-temporal clusters as sequences of continuing spatial clusters.

Moreover, Barbará [26] introduces four basic requirements in data stream clustering algorithms: i) Tracking cluster changes; ii) Compactness of representation; iii) Fast, incremental processing of new data points; iv) Clear and fast identification of “outliers”. Our proposed approach satisfies all these requirements, as ST-COPOT tracks continuing clusters and takes trees of spatial clusters as a compact representation for data in each batch, also it is a fast, incremental approach, moreover, our approach allows for the identification of outliers which are point objects that do not belong to any clusters.

3.2 Phase 1: Obtain Spatial Density Distribution

Our current implementation uses non-parametric kernel density estimation (KDE) [27] to estimate a 2-dimensional spatial density function \( f \). Our original approach uses Gaussian Mixture Model as spatial density estimation [28], which is really slow compared to kernel density estimation. For a bivariate random sample \( X_1, X_2, \ldots, X_n \) drawn from an unknown density function \( f \), the kernel density estimator is defined as follows:

\[
\hat{f}(x; H) = \frac{1}{n} \sum_{i=1}^{n} K_H(x - X_i)
\]

where \( x = (x_1, x_2)^T \), \( X_i = (X_{i1}, X_{i2})^T \) \( (i = 1, 2, \ldots, n) \), \( K(x) \) is the kernel which is a symmetric non-negative probability density function that integrates to one and has mean zero, \( H \) is bandwidth matrix which is symmetric and positive-definite, \( K_H(x) = |H|^{-1/2}KH^{-1/2}x \), and we are interested in estimating the density function \( f \) using \( \hat{f} \).

Our implementation uses the KernSmooth package [29] in R to estimate the spatial density distribution for a set of spatial points in each batch. In particular, KernSmooth uses the standard normal distribution as kernel:

\[
K(x) = (2\pi)^{-1} exp(-\frac{1}{2}x^Tx)
\]
And KernSmooth implements the diagonal bandwidth matrices $H$, and it supports user-specified bandwidth matrices, which is formulated as follows:

$$H = \begin{bmatrix} h_1^2 & 0 \\ 0 & h_2^2 \end{bmatrix}$$

Overall, KernSmooth uses binned approximation as the 2-dimensional kernel density estimation. Linear binning is used to obtain the bin counts and the Fast Fourier Transform is used to perform the discrete convolutions. For each location: $(x_1, x_2)$, the bivariate Gaussian kernel is centered on that location, and the heights of the kernel scaled by the bandwidths are summed—this sum, after a normalization, is the corresponding $\hat{f}$ value at that location.

### 3.3 Phase 2: Spatial Cluster and Contour Polygon Trees Extraction

For the second phase, the goal is to identify dense spatial regions in the data collection area as spatial clusters using the spatial density functions that have been created for each batch in phase 1. A spatial cluster in our approach is defined as a region which is enclosed by a polygon and whose probability density of data points is above a given threshold, and the polygon is a 2-dimensional closed density contour line of the spatial density function, which corresponds to a certain threshold. Our approach uses the contouring algorithm CONREC [30] and post-processing (e.g. close open contours, identify contour holes \(^2\)) to get such polygons.

CONREC's main input parameters are:

- A density threshold;
- A 2-dimensional array that contains the data to be contoured. Each element of the array is a sample of the surface being investigated at a point;
- Two, 1-dimensional arrays and which contain the horizontal and vertical coordinates of each sample point, which allows for a rectangular grid of samples;

As is mentioned above, the density values of the 3-dimensional surface are stored in a 2-dimensional array for each grid intersection point. This rectangular grid is considered one cell at a time. The center of each rectangle is assigned to the average values of each of the four vertices. Each rectangle is in turn divided into four triangular regions. Each of these triangular planes may be bisected by horizontal contour plane. The intersection of these two planes is a straight line segment, which forms part of the contour curve at that contour height. Then an interpolation routine is called to calculate the starting and stopping coordinates of the line segment.

The process of extracting spatial cluster mainly consists of the following 6 steps:

1. Grid the data collection area using a 2-dimensional grid.

\(^2\) Our approach allows holes inside spatial clusters.
2. Calculate a probability density for all grid intersection points using the spatial density function (given in Equation 1), and obtain a density matrix. Create a table $T$ to store locations of all grid intersection points and corresponding density matrix.

3. Pass $T$, along with a pair of density threshold $\theta_1, \hat{\theta}_1$ to CONREC, which returns two sets of contour lines.

4. Close open contour lines.

5. Classify the obtained contour lines into holes and spatial clusters;

6. Repeat step 3 to step 5 for density threshold pairs: $\theta_2, \hat{\theta}_2, \theta_3, \ldots, \theta_N, \hat{\theta}_N$, respectively.

As far as step 4 is concerned, for all the open contours, we use boundary lines of the data collection area to connect them and make them closed contours. As far as step 5 is concerned, the identified contour lines could be either spatial clusters for which the density increases as we move to the inside of the spatial cluster or hole contours for which the reverse is true. In order to distinguish contour holes from spatial clusters, we choose another threshold, which is slightly smaller than the chosen threshold. So we get two layers of contours, and contour holes are those whose density value decreases toward the center, which means the inner contour has smaller density value. Moreover, closed contours that have 0 points inside are also considered as contour holes.

After all the open contours are converted to closed contours, and all the contour holes are identified and added to the corresponding spatial cluster, we get a final list of spatial clusters which is a set of polygons which might contain holes, identifying areas where probability density of data points is above a given density threshold. Algorithm 1 gives the pseudocode how to extract spatial clusters for a specific threshold, and we repeat Algorithm 1 for multiple pairs of density thresholds to extract multilayer polygons for each batch.

At the end, we obtain contour polygon trees as the clustering result of each batch. Due to the characteristic of contouring algorithm-CONREC, all the contour lines extracted for different thresholds will not intersect. Therefore, by pair to pair comparison, easily we can find contour polygon trees. A more formal definition of contour polygon tree is given as follows:

**Definition 1.** Let $T$ be a contour polygon tree, $\text{child}(n)$ denotes the set of successor nodes of $n$ in the tree, and $n.polygon$ be the polygon associated with $n$:

\begin{align*}
\text{Contour Polygon Tree}(T) \iff \\
\forall n \in T \ (\text{not(leaf}(n)) \Rightarrow \forall \hat{n}(\text{child}(n) \Rightarrow (\hat{n}.polygon \subseteq n.polygon)) \land \\
\forall \hat{n} \in T \ \forall \hat{\hat{n}} \in T \ \exists n(\hat{n} \in \text{child}(n) \land \hat{\hat{n}} \in \text{child}(n) \Rightarrow \hat{n}.polygon \cap \hat{\hat{n}}.polygon = \emptyset)
\end{align*}

where, “$\subseteq$” represents the polygon containment relationship: $p \subseteq \hat{p} \iff p \cap \hat{p} = p$, with “$\cap$” represents the polygon intersection operator.

Moreover, we might get more than one contour polygon trees for each batch, which we call a forest of density contour trees. In general, the polygons in the
**Algorithm 1** Pseudocode for spatial cluster extraction (for one threshold)

**Input:**
- Location cloud: spatial points data collected at batch $i$;
- Spatial density function: $F_i$ for data collected in batch $i$;
- Density Thresholds: $\theta, \hat{\theta}$ (typically $\hat{\theta} = \theta \times 0.995$);
- Data collection area: given as rectangle or polygon;

**Output:**
- Set of generated spatial cluster $SC_i$ for each batch $i$;

1: Grid the data collection area;
2: Calculate log density $d$ at grid intersection points using $F_i$;
3: Create table $T$ with entry $(x, y, d)$ where $d$ is the density value at grid intersection point $(x, y)$.
4: /* Then apply CONREC: */
5: $SC_i = CONREC(\theta, T)$;
6: $\overline{SC}_i = CONREC(\hat{\theta}, T)$;
7: for each contour $C$ in $SC_i$ do
8: if $C$ is open contour then
9: Convert to closed contour;
10: end if
11: end for
12: for each contour $\overline{C}$ in $\overline{SC}_i$ do
13: if $\overline{C}$ is open contours then
14: Convert to closed contour;
15: end if
16: end for
17: for each pair of contour $C$ and $\overline{C}$ do
18: if there is $\overline{C}$ inside $C$ then
19: remove $C$ from $SC_i$, and add $C$ to hole list: $H$;
20: end if
21: end for
22: for each contour $C$ in $SC_i$ do
23: for each hole $h$ in hole list $H$ do
24: if $C$ contains hole $h$ then
25: add the hole $h$ to $C$;
26: end if
27: end for
28: end for
29: Return spatial clusters list: $SC_i$

higher level of the tree correspond to lower density threshold. Specifically, assume we use $N$ thresholds: $\theta_1 < \theta_2 < \ldots < \theta_N$, for each polygon in the tree, we associate it with a tree level value based on the corresponding density threshold of that polygon, for example, if there is a polygon in a tree which corresponds to density threshold $\theta_2$ then we associate it with level 2. But for some level values, there might be 0 polygons, as we might not get any polygon for corresponding
density threshold. Moreover, the root of a tree will always correspond to the lowest threshold $\theta_1$.

Figure 1 gives an example of visualizing contour polygon trees, as we can see, we use 3 density thresholds, using the containing relationship, we obtain two density contour trees on the right side, as we can see, the node of the tree is polygon that encloses spatial clusters and the edge indicates the containing relationship, which is, the higher level polygons contains lower level polygons. Also, we can see that the polygons of children of the same parent do not intersect. Moreover, we have 10 spatial clusters in total.

3.4 Spatio-temporal Cluster Extraction

The goal of phase 3 of is to identify continuing relationship from forests of contour polygon trees obtained for two consecutive batches. In order to accomplish this objective, we propose a set of novel distance functions, in particular, for contour polygons, contour polygon trees, and forests of contour polygon trees. Using those density functions, spatio-temporal clusters are defined and obtained at different granularities: continuing contour polygons, continuing contour polygon trees, or continuing forests of contour polygon trees.

Before we move to those definitions of distance functions, let us introduce the following notifications: let $p$ and $\overline{p}$ be contour polygons that correspond to the same density threshold, $t$ and $\overline{t}$ be contour polygon trees, $S$ and $\overline{S}$ be sets of contour polygons at the same level for polygon trees $t$ and $\overline{t}$ respectively, $N$ be the number of density thresholds we use in Phase 2, $F$ and $\overline{F}$ be forests of contour polygon trees. $p$ and $\overline{p}$, $t$ and $\overline{t}$, $F$ and $\overline{F}$ are created for 2 consecutive batches respectively.

First of all, in order to identify continuing relationships between contour polygons we need to define a distance function for polygons:

$$d_P(p, \overline{p}) = 1 - \frac{\text{area}(p \cap \overline{p})}{\text{area}(p \cup \overline{p})}$$  \hspace{1cm} (3)

where, $\text{area}(p \cap \overline{p})$ is the intersection area of $p$ and $\overline{p}$, and $\text{area}(p \cup \overline{p})$ is the union area of $p$ and $\overline{p}$. Obviously, the range of $d_P(p, \overline{p})$ is between 0 and 1. If two polygons are identical we obtain a distance of value 0, and their distance is 1 if there is no overlap between $p$ and $\overline{p}$.

Moreover, since our clustering result for each batch consists of sets of trees, distance function to assess the similarity of trees from two consecutive batches is needed. Since the root of a tree, always corresponds to the lowest density threshold, a naïve distance function could be defined as follows:

$$d_T^*(t, \overline{t}) = d(t.\text{root.polygon}, \overline{t}.\text{root.polygon})$$  \hspace{1cm} (4)

where, $t.\text{root.polygon}$ is the polygon at the root level (level 1) of tree $t$ and $\overline{t}.\text{root.polygon}$ is the polygon at the root level (level 1) of tree $\overline{t}$.

However, we are interested in obtaining distance function for comparing sets of polygons at levels 2, 3,\ldots, $N$ of the tree, firstly we define a “forward” distance
function for sets of polygons as follows:

$$d^*_S(S, \overline{S}) = \begin{cases} 
0 & \text{if } S = \emptyset \text{ and } \overline{S} = \emptyset \\
1 & \text{if } S = \emptyset \text{ or } \overline{S} = \emptyset \\
\frac{\sum_{p \in S} (\min_{p \in \overline{S}} d_S(p, p))}{|S|} & \text{otherwise.}
\end{cases}$$

(5)

where, $|S|$ is the total number of polygons in $S$, and the distance $d_S(p, \overline{p})$ is calculated using Equation 3. Basically, we try to sum up the closest distance from polygons in $\overline{S}$ to the polygons in $S$ by iterating over all the polygons in $S$. If there exists a polygon that doesn’t have any overlap with the polygons in $S$, the distance will be 1. And in order to normalize total distance, we divide the total distance by the number of polygons in $S$.

However, from Equation 5, we know that $d^*_S(S, \overline{S}) \neq d^*_S(\overline{S}, S)$, as $S$ and $\overline{S}$ might consist of different number of polygons; consequently, the last two distance functions we introduced are not symmetric. To deal with this problem, we use the same formula to calculate the backward distance: $d^*_S(\overline{S}, S)$ which sums up the closest distance from polygons in $S$ to polygons in $\overline{S}$ iterating over $\overline{S}$. By averaging the forward and backward distances to obtain the following symmetric distance for sets of polygons:

$$d_S(S, \overline{S}) = \frac{d^*_S(S, \overline{S}) + d^*_S(\overline{S}, S)}{2}$$

(6)

Now we have distance function to assess the similarity of polygons at a specific tree level; next, we propose a distance function for pairs of contour polygon trees that compares all the levels of each tree, which is defined as follows:

Let $N$ be the number of levels of $t$ and $\overline{t}$ respectively; $\text{level}(i, t)$ and $\text{level}(i, \overline{t})$ be the sets of polygons at level $i$ for $t$ and $\overline{t}$ respectively; moreover $\rho \in (0, 1]$ is a parameter called discount factor$^3$. The multi-level DPT distance function is as follows:

$$d_T(t, \overline{t}) = \frac{d^*_T(t, \overline{t}) + \sum_{j=2}^{N} \rho^{j-1} \cdot d_S(\text{level}(j, t), \text{level}(j, \overline{t}))}{1 + \sum_{j=2}^{N} \rho^{j-1}}$$

(7)

What we introduced so far puts more importance to polygons closer to the root when assessing the similarity between two DPTs; however, the above distance function $d_T(t, \overline{t})$ can be generalized to put more importance to a particular level of a contour polygon tree, which is defined as follows:

Let $N$ be the number of levels of $t$ and $\overline{t}$ respectively; $s$ be the focus granularity level with $1 \leq s \leq N$; $\text{level}(i, t)$ and $\text{level}(i, \overline{t})$ be the sets of polygons at level $i$ for $t$ and $\overline{t}$, respectively; moreover, $\rho \in (0, 1]$ is the discount factor.

$$d_{T,s}(t, \overline{t}) = \frac{\sum_{j=1}^{N} (\rho^{j-s}) \cdot d_S(\text{level}(j, t), \text{level}(j, \overline{t}))}{\sum_{j=1}^{N} \rho^{j-s}}$$

(8)

$^3$ Disagreement counts less the deeper we go down the tree.
For example, if \( N = 5 \) and \( s = 3 \), level 3 agreement is weighted by 1, level 2 and 4 agreement is weighted by \( \rho \) when computing the distance between \( t \) and \( \tilde{t} \); moreover, the following holds: 
\[
d_T(t, \tilde{t}) = d_{T,1}(t, \tilde{t}).
\]

Furthermore, using the same approach of defining \( d_S^*(S, \overline{S}) \), we define the forward distance function for two forests of contour polygon trees:

\[
d^*_F(F, \overline{F}) = \begin{cases} 
0 & \text{if } F = \emptyset \text{ and } \overline{F} = \emptyset \\
1 & \text{if } F = \emptyset \text{ or } \overline{F} = \emptyset \\
\sum_{t \in F}(\min_{t' \in F}(d_T(t, t'))) / |F| & \text{otherwise.}
\end{cases}
\]

(9)

where, \( |F| \) is the total number of trees in \( F \), and the distance \( d_T(t, \tilde{t}) \) is calculated using Equation 7 or 8. Basically, we sum up the closest distance from the trees in \( \overline{F} \) to the trees in \( F \) by iterating over all the trees in \( F \). If there exists a tree that doesn’t have any overlap with the trees in \( F \), the distance will be 1. And in order to normalize the total distance, we divide the total distance by the number of trees in \( F \).

Also, from definition above, we know that \( d^*_F(F, \overline{F}) \neq d^*_F(\overline{F}, F) \), as \( F \) and \( \overline{F} \) might consist of different number of trees, and we reuse the earlier approach to make the forest distance function symmetric:

\[
d_F(F, \overline{F}) = d_F(\overline{F}, F) = \frac{d_F^*(F, \overline{F}) + d_F^*(\overline{F}, F)}{2}
\]

(10)

In summary, we defined a family of distance functions for pairs of polygons, pairs of sets of polygons, pairs of contour polygon trees and pairs of density contour forests. Next, we use the introduced distance functions to propose computational methods that identify continuing relationships between consecutive batches at different granularities:

- If the distance between two contour polygons from two consecutive batches are less than a certain threshold \( \gamma_1 \), we conclude that the contour polygon doesn’t change significantly over two consecutive batches, and create a 'continuing' relationship between the two polygons, if the following condition holds:
  \[
  \text{Continuing}(p, \overline{p}) \iff d_P(p, \overline{p}) < \gamma_1
  \]
  (11)

- If the distance between two contour polygon trees from two consecutive batches are less than a certain threshold \( \gamma_2 \), we create a 'continuing' relationship between the two contour polygon trees, if the following condition holds:
  \[
  \text{Continuing}(t, \overline{t}) \iff d_T(t, \overline{t}) < \gamma_2
  \]
  (12)

- Similarly, if the distance between two forests of contour polygon trees from two consecutive batches are less than a certain threshold \( \gamma_3 \), we infer a 'continuing' relationship between the two forests, if the following condition holds:
  \[
  \text{Continuing}(F, \overline{F}) \iff d_F(F, \overline{F}) < \gamma_3
  \]
  (13)
3.5 Time and Space Complexities

Table 1: Time/Space Complexities of ST-COPOT

<table>
<thead>
<tr>
<th>Phase</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>$O(m^2 \times n)$</td>
<td>$O(n + m^2)$</td>
</tr>
<tr>
<td>Phase 2</td>
<td>$O(m^2)$</td>
<td>$O(m^2 + e^2)$</td>
</tr>
<tr>
<td>Phase 3</td>
<td>$O(e^2)$</td>
<td>$O(e^2)$</td>
</tr>
</tbody>
</table>

Table 1 gives the time and space complexities of ST-COPOT. Supposing the grid size we use is $m \times m$, $n$ is the total number of points, $e$ is the average number of edges a spacial cluster has, $N$ is the number of thresholds we use, $T$ is the average number of trees in each batch. For ST-COPOT, since $N$ is a constant of a small value and $T$ is also a small value, ST-COPOT has the same time and space complexities as ST-DPOLY [11]. Since $e$ is usually smaller than $m$ and it is a serial approach, the overall time complexity of ST-COPOT would be $O(m^2 \times n)$, in cases that the number of data points is much larger than the number of grid cells ($n \gg m^2$), ST-COPOT’s time complexity becomes $O(n)$.

4 Experiment and Analysis

4.1 Dataset Description

The TLC Trip Record Data [31] were collected by technology providers authorized under the Taxicab and Livery Passenger Enhancement Programs (TPEP/LPEP), which contain data for over 1.1 billion taxi trips from January 2009 through June 2016, covering both yellow and green taxis. Each individual trip record contains precise location coordinates for where the trip started and ended, timestamps for when the trip started and ended, and a few other variables including fare amount, payment method, and distance traveled.

4.2 Demonstration of ST-COPOT

For the experiment, we use yellow taxi pick-up locations collected in 2-hour interval as batches on January 8th, 2014, and we use the Manhattan metropolitan area as the data collection area, and the total number of taxi pick-ups is over 430,000. The clustering result is presented in Figure 2. As we can see, using ST-COPOT can easily track the evolution of clusters over time by looking into the continuing relationship at different granularities. For example:

- There is a region centered around Time Square in the late night and before dawn, which shows that Time Square is where many people gather around during this time window.
Fig. 2: Example of Spatio-temporal clustering of taxi pick-up location streams (2-hour interval as batch size, grid size $300 \times 300$, bandwidth 0.0015, log density: purple (6.8); black (7.0); red (7.4), thresholds for establishing continuing relationships all use 0.45, arrows: purple(continuing polygons for density threshold 1); black(continuing polygons for density threshold 2); blue(continuing trees); red(continuing forests) )
– Early in the morning, in terms of east of Midtown area, there are two sub-regions with high density, one is the southwest of Time Square which is closer to several train and bus terminals; the other subregion is further southwest, which is closer to the 34 Street Penn Station. We infer that, early in the morning, after New Yorkers get off trains, a lot of people choose to look for taxi rides, which explains the presence of high density polygons of taxi pick-ups around the train stations. Similarly, in the 6-7AM and 7-8am batch, a contour polygon tree centering around the Grand Central Terminal exhibits a similar taxi pickup pattern.

– In time window 8-11am, we obtain continuing forests of contour polygon trees, which means during these 4 hours the dense pick-up location regions are relatively stable, and there are 3 dense regions: southwest of Time Square, W 34th St, and Grand Central Station.

– 2-7pm in the afternoon, there are a lot of people who look for taxis at the W 34th St. But in the following 2 hours (8-9pm), the contour polygon tree disappears, which means if you need a taxi at around W 34th St early in the evening, it is easier to get a taxi if the time is after 8pm.

– Moreover, since all those spatial clusters are dense regions in terms of taxi pick-ups, it means it is harder to get a taxi in those regions during the time window when they are identified as spatial clusters, people who take taxis can better plan their activities beforehand using such information, which also shows the potential practical value of ST-COPOT for people who take taxis especially commuters.

4.3 Quality of Clusters

In order to evaluate the quality of the clustering results of ST-COPOT, we apply a density-based clustering validation method proposed by Moulavi [32] to the clusters obtained. Using this method, we can obtain a relative validity index for each cluster we created, the range of validity index is between -1 and 1. But one flaw of the validation method is that it does not consider the case that only a single cluster is obtained; in this case we ignore the validity index.

In the experimental result depicted in Figure 2, totally we observe 43 contour polygon trees, the number of number of spatial clusters for each level: 43 clusters for root, 26 clusters for level 2, and 6 clusters for level 3. Figure 3 gives the histogram of validity indexes obtained for each cluster. Over 77 percent of the clusters have a validity index larger than 0.5, and over 69 percent of clusters have a validity index larger than 0.6, and the average validity index is 0.6415, which shows that most of the spatial clusters obtained have good quality.

For those clusters with low validity indexes, it occurs when two clusters are close, in which case, the validation method consider them as low quality clusters and as we should merge them. However, due to the characteristics of contouring algorithm, we obtain two separate clusters that are close to each other if the region between them has relatively low density. For example, as is shown in Figure 4, both the purple and black clusters have low validity indexes, but as we can see from the density of pick-up locations, it is reasonable to split them
Validity Indexes of Clusters

Fig. 3: Histogram of validity indexes for clusters in Figure 2

Fig. 4: Example of clusters with low quality (it corresponds to the 4-5am in Figure 2)
into two trees, as there is gap where the density of points are low, which causes the splitting by the contouring algorithm. The same thing happens to the two close trees on the left of 6-7am figure in Figure 2, the validity indexes we have for these two trees are: black (-0.3362291, -0.2200243), and purple (0.07295217, -0.02499342), which further justify the case that though some clusters obtained by ST-COPOT have low validity indexes, it does not mean they have low quality.

5 Conclusion

The main objective of the presented study is the development of spatio-temporal clustering algorithms for point cloud streams. In this paper, we proposed a serial, density-contour based spatio-temporal clustering approach ST-COPOT, which first employs a non-parametric density estimation approach to obtain spatial cluster as regions enclosed by polygons that have been generated from contour lines whose density corresponds to certain density thresholds. To support these activities, the paper employs a data structure called contour polygon trees as a compact representation of clustering result of each batch. Our approach at last forms spatio-temporal clusters by identifying continuing relationships between temporally consecutive spatial clusters, contour polygon trees and forest of contour polygon trees. We also demonstrate the effectiveness of our approach by conducting a case study involving the taxi trips data of the New York city demonstrating that ST-COPOT can discover interesting spatio-temporal patterns in taxi pick-up location streams. Moreover, we are not aware of other approaches that use contouring algorithms and contour analysis to obtain spatio-temporal clusters.

As far as future work is concerned, we will try to extend our approach to support dynamic or adaptive batch sizes. In terms of selecting parameters for the proposed approach, we will investigate semi-automatic and automatic parameter selection tools to facilitate the use of ST-COPOT. We also plan to develop a parallel version of ST-COPOT. Last but not least, we plan to adapt our spatio-temporal clustering approach to identify economically attractive pickup locations: locations that produce the largest profits for the cab company.

References

8. Sellis, T.: Managing streaming spatial data