

# Efficient Query Processing Techniques for Spatial Time Series Data <sup>†</sup>

Pusheng Zhang

Department of Computer Science and Engineering,

University of Minnesota

200 Union Street SE, Minneapolis, MN-55455

pusheng@cs.umn.edu

May 28, 2003

## Abstract

A spatial time series dataset is a collection of time series, each referencing a location in a common spatial framework. Correlation analysis is often used to identify pairs of potentially interacting elements from the cross product of two spatial time series datasets (the two datasets may be the same). However, the computational cost of correlation analysis is very high when the dimension of the time series and the number of locations in the spatial frameworks are large. In this paper, I use a spatial autocorrelation-based search tree structure to propose new processing strategies for correlation-based similarity range queries and similarity joins. I provide a preliminary evaluation of the proposed strategies using experimental studies with Earth science datasets.

**Keywords:** Spatial Time Series Data, Query Processing, Spatial Autocorrelation

---

<sup>†</sup>This work was partially supported by NASA grant No. NCC 2 1231 and by Army High Performance Computing Research Center contract number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

# 1 Introduction

Analysis of spatio-temporal datasets [14, 15, 16, 9] collected by satellites, sensor nets, retailers, mobile device servers, and medical instruments on a daily basis is important for many application domains such as epidemiology, ecology, climatology, and census statistics. The development of efficient tools [1, 5] to explore these datasets, the focus of this work, is crucial to organizations which make decisions based on large spatio-temporal datasets. A spatial framework [19] consists of a collection of locations and a neighbor relationship. A time series is a sequence of observations taken sequentially in time [3]. A spatial time series dataset is a collection of time series, each referencing a location in a common spatial framework. For example, the collection of global daily temperature measurements for the last 10 years is a spatial time series dataset over a degree-by-degree latitude-longitude grid spatial framework on the surface of the Earth.

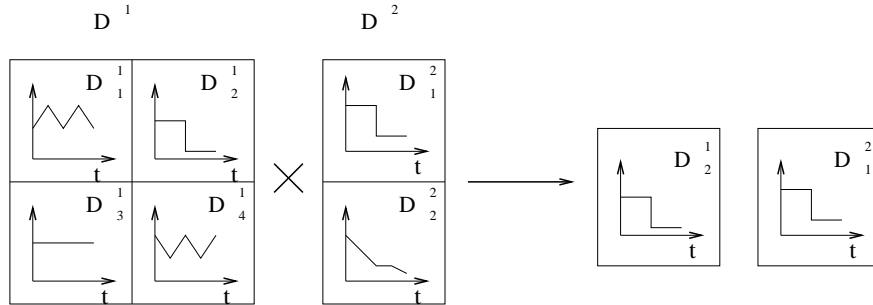


Figure 1: An Illustration of the Correlation Analysis of Two Spatial Time Series Datasets

Correlation analysis is important to identify potentially interacting pairs of time series across two spatial time series datasets. A strongly correlated pair of time series indicates potential movement in one series when the other time series moves. For example, El Nino, the anomalous warming of the eastern tropical region of the Pacific, has been linked to climate phenomena such as droughts in Australia and heavy rainfall along the Eastern coast of South America [17]. Figure 1 illustrates the correlation analysis of two spatial time series datasets  $D^1$  and  $D^2$ .  $D^1$  has 4 spatial locations and  $D^2$  has 2 spatial locations. The cross product of  $D^1$  and  $D^2$  has 8 pairs of locations. A highly correlated pair, i.e.  $(D^1_2, D^2_1)$ , is identified from the correlation analysis of the cross product of the two datasets.

However, a correlation analysis across two spatial time series datasets is computationally expensive when the dimension of the time series and number of locations in the spaces are large. The computational cost can be reduced by reducing time series dimensionality or reducing the number of time series pairs to be tested, or both. Time series dimensionality reduction techniques include discrete Fourier transformation [1], discrete wavelet transformation [5], and singular vector decomposition [8].

My work focuses on reducing the number of time series pairs to be tested by exploring spatial autocorrelation. Spatial time series datasets comply with Tobler’s first law of geography: everything is related to everything else but nearby things are more related than distant things [18]. In other words, the values of attributes of nearby spatial objects tend to systematically affect each other. In spatial statistics, the area devoted to the analysis of this spatial property is called spatial autocorrelation analysis [6]. I have proposed a naive uniform-tile cone-based approach for correlation-based similarity joins in my previous work [20]. This approach groups together time series in spatial proximity within each dataset using a uniform grid with tiles of fixed size. The number of pairs of time series can be reduced by using a uniform-tile cone-level join as a filtering step. All pairs of elements, e.g., the cross product of the two uniform-tile cones, which cannot possibly be highly correlated based on the correlation range of the two tile cones are pruned. However, the uniform tile cone approach is vulnerable because spatial heterogeneity may make it ineffective.

In this paper, I use a spatial autocorrelation-based search tree to solve the problems of correlation-based similarity range queries and similarity joins on spatial time series datasets. The proposed approach divides a collection of time series into hierarchies based on spatial autocorrelation to facilitate similarity queries and joins. I propose processing strategies for correlation-based similarity range queries and similarity joins using the proposed spatial autocorrelation-based search trees. Experimental evaluations with Earth science data [12] show that the performance of the similarity range queries and joins processing strategies using the spatial autocorrelation-based search tree structure often saves a large fraction of computational cost.

**Application Domain:** NASA Earth observation systems currently generate a large

sequence of global snapshots of the Earth, including various atmospheric, land, and ocean measurements such as sea surface temperature (SST), pressure, precipitation, and Net Primary Production (NPP). NPP is the net photosynthetic accumulation of carbon by plants. Keeping track of NPP is important because it includes the food source of humans and all other organisms and thus, sudden changes in the NPP of a region can have a direct impact on the regional ecology. These data are spatial time series data in nature.

The climate of the Earth’s land surface is strongly influenced by the behavior of the oceans. Simultaneous variations in climate and related processes over widely separated points on the Earth are called teleconnections. For example, every three to seven years, an El Nino event, i.e., the anomalous warming of the eastern tropical region of the Pacific Ocean, may last for months, having significant economic and atmospheric consequences worldwide. To investigate such land-sea teleconnections, time series correlation analysis across the land and ocean is often used to reveal the relationship of measurements of observations.

Due to large amount of data available, the performance of naive nested loop algorithms is not sufficient to satisfy the increasing demands to efficiently process correlation-based similarity queries in large spatial time series datasets. Thus we propose new algorithms to facilitate the correlation-based similarity query processing in data.

## 2 Basic Concepts

Let  $x = \langle x_1, x_2, \dots, x_m \rangle$  and  $y = \langle y_1, y_2, \dots, y_m \rangle$  be two time series of length  $m$ . The correlation coefficient [4] of the two time series is defined as:

$$\text{corr}(x, y) = \frac{1}{m-1} \sum_{i=1}^m \left( \frac{x_i - \bar{x}}{\sigma_x} \right) \cdot \left( \frac{y_i - \bar{y}}{\sigma_y} \right) = \hat{x} \cdot \hat{y}, \text{ where } \bar{x} = \frac{\sum_{i=1}^m x_i}{m}, \sigma_x = \sqrt{\frac{\sum_{i=1}^m (x_i - \bar{x})^2}{m-1}}, \bar{y} = \frac{\sum_{i=1}^m y_i}{m}, \sigma_y = \sqrt{\frac{\sum_{i=1}^m (y_i - \bar{y})^2}{m-1}}, \hat{x}_i = \frac{1}{\sqrt{m-1}} \frac{x_i - \bar{x}}{\sigma_x}, \hat{y}_i = \frac{1}{\sqrt{m-1}} \frac{y_i - \bar{y}}{\sigma_y}, \hat{x} = \langle \hat{x}_1, \hat{x}_2, \dots, \hat{x}_m \rangle, \text{ and } \hat{y} = \langle \hat{y}_1, \hat{y}_2, \dots, \hat{y}_m \rangle. \text{ Because the sum of the } \hat{x}_i^2 \text{ is equal to 1:}$$

$$\sum_{i=1}^m \hat{x}_i^2 = \sum_{i=1}^m \left( \frac{1}{\sqrt{m-1}} \frac{x_i - \bar{x}}{\sigma_x} \right)^2 = 1, \hat{x} \text{ is located in a multi-dimensional unit sphere.}$$

Similarly,  $\hat{y}$  is also located in a multi-dimensional unit sphere. Based on the definition of  $\text{corr}(x, y)$ , we have  $\text{corr}(x, y) = \hat{x} \cdot \hat{y} = \cos(\angle(\hat{x}, \hat{y}))$ . The correlation of two time series is directly related to the angle between the two time series in the multi-dimensional unit sphere. Finding pairs of time series with an absolute value of correlation above the user

given minimal correlation threshold  $\theta$  is equivalent to finding pairs of time series  $\hat{x}$  and  $\hat{y}$  on the unit multi-dimensional sphere with an angle in the range of  $[0, \theta_a]$  or  $[180^\circ - \theta_a, 180^\circ]$  [20].

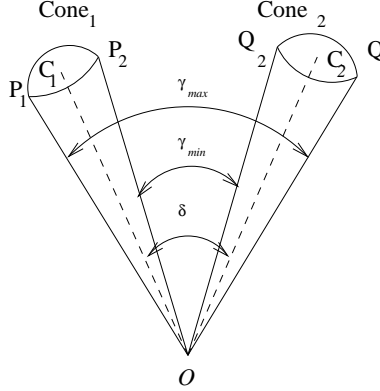


Figure 2: Angle of Time Series in Two Spherical Cones

A cone is a set of time series in a multi-dimensional unit sphere and is characterized by two parameters, the center and the span of the cone. The center of the cone is the mean of all the time series in the cone. The span  $\tau$  of the cone is the maximal angle between any time series in the cone and the cone center. The largest angle ( $\angle P_1 O Q_1$ ) between two cones  $C_1$  and  $C_2$  is denoted as  $\gamma_{max}$  and the smallest angle ( $\angle P_2 O Q_2$ ) is denoted as  $\gamma_{min}$ , as illustrated in Figure 2. We have proved that if  $\gamma_{max}$  and  $\gamma_{min}$  are in specific ranges, the absolute value of the correlation of any pair of time series from the two cones are all above  $\theta$  (or below  $\theta$ ) [20]. Thus all pairs of time series between the two cones satisfy (or dissatisfy) the minimal correlation threshold. To be more specific, if we let  $C_1$  and  $C_2$  be two cones from the multi-dimensional unit sphere structure and let  $\hat{x}$  and  $\hat{y}$  be any two time series from the two cones respectively, we have the following properties (please refer to [20] for proof details):

1. If  $0 \leq \gamma_{max} \leq \theta_a$ , then  $0 \leq \angle(\hat{x}, \hat{y}) \leq \theta_a$ .
2. If  $180^\circ - \theta_a \leq \gamma_{min} \leq 180^\circ$ , then  $180^\circ - \theta_a \leq \angle(\hat{x}, \hat{y}) \leq 180^\circ$ .
3. If  $\theta_a \leq \gamma_{min} \leq 180^\circ$  and  $\gamma_{min} \leq \gamma_{max} \leq 180^\circ - \theta_a$ , then  $\theta_a \leq \angle(\hat{x}, \hat{y}) \leq 180^\circ - \theta_a$

If either of the first two conditions is satisfied,  $\{C_1, C_2\}$  is called an All-True cone pair (All-True lemma). If the third condition is satisfied,  $\{C_1, C_2\}$  is called an All-False cone

pair (All-False lemma).

### 3 Strategies for Correlation-Based Similarity Queries

In this section, I describe the formation of a spatial autocorrelation-based search tree and strategies for processing correlation-based similarity range queries and joins using the proposed search tree.

#### 3.1 Spatial Autocorrelation-Based Search Tree Formation

I explore spatial autocorrelation, i.e., the influence of neighboring regions on each other, to form a search tree. Search tree structures have been widely used in traditional DBMS (e.g. B-tree and B+ tree) and spatial DBMS (quad-tree, R-tree, R<sup>+</sup>-tree, R<sup>\*</sup>-tree, and R-link tree [13, 15]). To fully exploit the spatial autocorrelation property, there are three major criteria for choosing a tree on the spatial time series datasets. First, a spatial tree structure is preferred to incorporate the spatial component of the datasets. Second, during the tree formation the time series calculations such as mean and span should be minimized while still need to maintain a high correlation (high clustering) among time series within a tree node. Third, threaded leaves where leaves are linked are preferred to support sequential scan of files which are useful for high selectivity ratio correlation queries. Other desired properties include depth balances of a tree and incremental updates when the time series component changes.

I choose a simple quad tree with threaded leaves which satisfies the three criteria. Other tree structures are also possible and will be explored in future work. As shown in Algorithm 1, the space is first divided into a collection of disjoint cells with a coarse starting resolution. Each cell represents a cone in the multi-dimensional unit sphere representation and includes multiple time series. Then the center and span are calculated to characterize each cone. When the cone span exceeds the maximal span threshold, this cone is split into four quadrants. Each quadrant is checked and split recursively until the cone span is less than the maximal span.

The maximal span threshold can be estimated by using an algebraic formula analyzed as

---

**Algorithm 1** Spatial\_Similarity\_Search\_Tree\_Formation

---

**Input:** 1)  $S = \{s_1, s_2, \dots, s_n\}$  :  $n$  spatial referenced time series  
where each instance references a spatial framework  $SF$ ;  
2) a maximum threshold of cone angle  $\tau_{max}$

**Output:** Similarity Search Tree with Threaded Leaves

**Method:**  
divide  $SF$  into a collection of disjoint cells  $C$   
/\* each cell is mapped to a cone. \*/  
 $index = 1$ ;  
while ( $index < C.size$ )  
   $C(index).center = Calculate\_Center(C, index)$ ;  
  /\* cone center is the average time series within the cone. \*/  
   $C(index).angle = Calculate\_Span(C, index)$ ;  
  /\* cone span is the max angle between any time series and the  
  cone center within the cone. \*/  
  if (  $C(index).angle > \tau_{max}$  )  
    split cell  $C(index)$  into four quadrants  $C_{11}, C_{12}, C_{13}, C_{14}$ ;  
    insert four quadrants into  $C$  at position  $index + 1$ ;  
    set  $C_{11}, C_{12}, C_{13}, C_{14}$  as  $C(index)$ 's children;  
  else  
     $index ++$  ;  
    insert  $C(index)$  at the end of the threaded leaf list;  
return  $C$ ;

---

follows. Given a minimal correlation threshold  $\theta$  ( $0 < \theta < 1$ ),  $\gamma_{max} = \delta + \tau_1 + \tau_2$  and  $\gamma_{min} = \delta - \tau_1 - \tau_2$ , where  $\delta$  is the angle between the centers of two cones, and the  $\tau_1$  and  $\tau_2$  are the spans of the two cones respectively. For simplicity, suppose  $\tau_1 \simeq \tau_2 = \tau$ . We have the following two properties (Please refer to [20] for proof details):

1. Given a minimal correlation threshold  $\theta$ , if a pair of cones both with span  $\tau$  is an All-True cone pair, then  $\tau < \frac{\arccos(\theta)}{2}$ .
2. Given a minimal correlation threshold  $\theta$ , if a pair of cones both with span  $\tau$  is an All-False cone pair, then  $\tau < \frac{180^\circ}{4} - \frac{\arccos(\theta)}{2}$ .

I use the above two properties to develop a heuristic to bound the maximal span of a cone. The maximal span of a cone is set to be the minimal of the  $\frac{\arccos(\theta)}{2}$  and  $\frac{180^\circ}{4} - \frac{\arccos(\theta)}{2}$ . The starting resolution can be investigated by using a spatial correlogram [6]. A spatial correlogram plots the average correlation of pairs of spatial time series with the same spatial distance against the spatial distances of those pairs. I choose the starting resolution size whose average correlation is close to the correlation which corresponds to

$$\min\left(\frac{\arccos(\theta)}{2}, \frac{180^\circ}{4} - \frac{\arccos(\theta)}{2}\right).$$

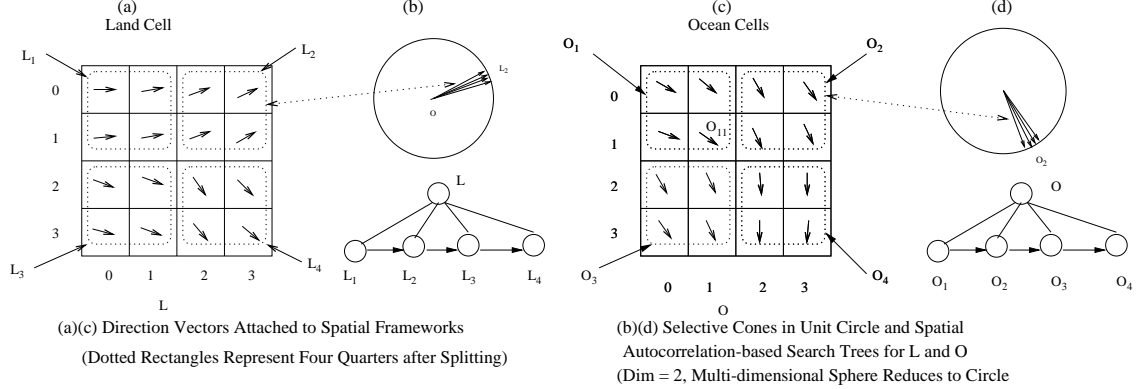


Figure 3: An Illustrative Example for Spatial Autocorrelation-Based Search Tree Formation

**Example 1 (Spatial Autocorrelation-Based Search Tree Formation)** Figure 3 illustrates the spatial autocorrelation-based search tree formation for two datasets, namely land and ocean. Each land/ocean framework consists of 16 locations on the starting resolution. The time series of length  $m$  in a location  $s$  is denoted as  $F(s) = F_1(s), F_2(s), \dots, F_i(s), \dots, F_m(s)$ . Figure 3 only depicts a time series for  $m = 2$ . Each arrow in a location  $s$  of ocean or land represents the vector  $\langle F_1(s), F_2(s) \rangle$  normalized to the two dimensional unit sphere. Since the dimension of the time series is two, the multi-dimensional unit sphere reduces to a unit circle, as shown in Figure 3 (b) and (d).

Both land and ocean cells are further split into four quadrants respectively due to the spatial heterogeneity in the cell. The land is partitioned to  $L_1 - L_4$  and the ocean is partitioned to  $O_1 - O_4$ , as shown in Figure 3 (a) and (c). Each quadrant represents a cone in the multi-dimensional unit sphere. For example, the patch  $L_2$  in Figure 3 (a) matches  $L_2$  in the circle in Figure 3 (b). All leaves are threaded, assuming that  $L_1$  to  $L_4$  and  $O_1$  to  $O_4$  are all leaves.

### 3.2 Strategies for Similarity Range Queries and Similarity Joins

The first step is to pre-process the raw data to the multi-dimensional unit sphere representation. The second step, formation of spatial autocorrelation-based search trees involves grouping similar time series into hierarchical cones using the one described in Algorithm 1. The query processing functions called may be related to similarity range query or similarity join, depending on the query types.

---

**Algorithm 2** Correlation-Based Similarity Query Algorithm

---

**Input:** 1)  $S^1 = \{s_1^1, s_2^1, \dots, s_{n_1}^1\}$ :  $n_1$  spatial referenced time series where each instance references a spatial framework  $SF_1$ ;  
2)  $S^2 = \{s_1^2, s_2^2, \dots, s_{n_2}^2\}$ :  $n_2$  spatial referenced time series where each instance references a spatial framework  $SF_2$ ;  
3) a user defined correlation threshold  $\theta$ ;  
4) query time series denote  $T_q$  ;  
5) a maximum threshold of cone angle  $\tau_{max}^1$   
6) a maximum threshold of cone angle  $\tau_{max}^2$   
**Output:** pairs of time series each from  $S^1$  and  $S^2$  or  $T_q$  and  $S^2$  with correlations above  $\theta$ ;

**Method:**

```
Pre-processing( $S^1$ ); Pre-processing( $S^2$ ); (1)
 $T_1$  = Spatial_Similarity_Search_Tree_Formation( $S^1, \tau_{max}^1$  ); (2)
 $T_2$  = Spatial_Similarity_Search_Tree_Formation( $S^2, \tau_{max}^2$  ); (3)
if range query (4)
/* assume to find highly correlated time series with  $T_q$  in  $S^2$ .*/ (5)
    Similarity_Range_Query( $T_2, T_q, \theta$ ); (6)
else if similarity join (7)
    Similarity_Join( $T_1, T_2, \theta$ ); (8)
```

---

### 3.2.1 Strategies for Range Queries

Given a query time series  $T_q$ , I want to search all highly correlated time series from the spatial time series dataset  $S^2$  with  $T_q$ . In general, strategies to process range queries include scan-based approaches and search tree-based approaches [7]. The scan-based approaches probe each individual nodes one by one. The search tree-based approach starts from the root of the tree and branches to a node's children only when certain conditions are satisfied, e.g., the minimal bounding box of the child contains the querying element. I adopt two common strategies to traverse the spatial autocorrelation-based search tree (step 1), namely threaded-leaves-only strategy and tree-based strategy. Note that the query time series  $T_q$  can be treated as a cone with a cone span 0. The threaded-leaves-only traversal only visits all the leaf nodes of the tree. The pairs of time series formed by  $T_1$  and each time series in a leaf node which satisfies the All-True lemma will be output. The pairs of time series formed by  $T_1$  and each time series in a leaf node which satisfies the All-False lemma will be ignored. An individual time series in a leaf node which fails both All-True and All-False lemmas will be visited. The tree-based traversal starts from the root and checks the All-True and All-False lemmas. The children of non-leaf nodes which fail both

---

**Algorithm 3** Similarity\_Range\_Query
 

---

**Input:** 1)  $T$ : a spatial autocorrelation-based search tree with threaded leaves;  
 2)  $T_q$ : the query time series denote;  
 3) a user defined correlation threshold  $\theta$ ;

**Output:** all time series each from  $S$  whose correlations with  $T_q$  are above  $\theta$ ;

**Method:**

```

traverse  $T$ ; for each cone  $c$  on the route do (1)
   $Filter\_Flag = Cone\_level\_Join(T_q, c, \theta)$ ; (2)
  if ( $Filter\_Flag == ALL\_TRUE$ ) (3)
    output all time series in the cone  $c$  (4)
  else if ( $Filter\_Flag != ALL\_FALSE$ ) (5)
    if  $c$  is a leaf node (6)
      for all pair  $T_q$  and  $s$  from  $c$  do (7)
         $High\_Corr\_Flag = Instance\_level\_Join(T_q, s, \theta)$ ; (8)
        if ( $High\_Corr\_Flag$ ) output  $s$ ; (9)
      else for each  $c'$  of  $c$ 's children do (10)
        Similarity_Range_Query( $c', T_q, \theta$ ) (11)
  
```

---

All-True and All-False lemmas will be visited until a leaf node is reached. For leaf nodes, the process is the same as that in the threaded-leaves-only traversal.

**Example 2 (A Similarity Range Query)** The range query with respect to  $O_{11}$  and  $L$  in Figure 3 (a) and (c) is applied as shown in Table 1. For the threaded-leaves-only traversal, all leaf cones are checked against  $O_{11}$  for correlation. The total cost is the sum of 4, which is the filtering cost, and 4, which is the refinement cost. For the tree-based traversal,  $O_{11}$  is first checked with  $L$  against the All-True and All-False lemmas. If both of them fail, all of  $L$ 's four children, which are all leaf nodes, are checked. Three of them satisfy the All-False Lemmas and one needs refinement where individual time series are checked against  $O_{11}$  for correlation. The total correlation computation is the sum of 5 and 4, which is the refinement cost. For this particular example, the tree-based traversal is more expensive than the threaded-leaves-only traversal.

Tree-Based Traversal			Threaded-leaves-only Traversal		
Ocean-Land	Filtering	Refinement	Ocean-Land	Filtering	Refinement
$O_{11} - L$	No	No			
$O_{11} - L_1$	No	4	$O_{11} - L_1$	No	4
$O_{11} - L_2$	All-False		$O_{11} - L_2$	All-False	
$O_{11} - L_3$	All-False		$O_{11} - L_2$	All-False	
$O_{11} - L_4$	All-False		$O_{11} - L_2$	All-False	

Table 1: The Range Query with Respect to  $O_{11}$  in Example Data

### 3.2.2 Strategies for Similarity Joins

Spatial join operations are usually divided into a filter step and a refinement step [15] to efficiently process complex spatial data types such as point collections. In the filter step, the spatial objects are represented by simpler approximations such as the MBR (Minimum Bounding Rectangle). There are several well-known algorithms, such as plane sweep [2], space partition [10] and tree matching [11], which can then be used for computing the spatial join of MBRs using the overlap relationship; the answers from this test form the candidate solution set. In the refinement step, the exact geometry of each element from the candidate set and the exact spatial predicates are examined along with the combinatorial predicate to obtain the final result.

---

#### Algorithm 4 Similarity\_Join

---

**Input:** 1)  $T^1$ : a spatial autocorrelation-based search tree with threaded leaves ;  
 2)  $T^2$ : a spatial autocorrelation-based search tree with threaded leaves;  
 3) a user defined correlation threshold  $\theta$ ;  
**Output:** all pairs of time series each from leaves of  $C^1$  and  $C^2$  with correlations above  $\theta$ ;  
**Method:**

```

  traverse  $T_1$  via threaded leaves; for each  $c_1$  from  $T_1$  do (1)
    traverse  $T_2$ ; for each  $c_2$   $T_2$  do (2)
       $Filter\_Flag = Cone\_level\_Join(c_1, c_2, \theta)$ ; (3)
      if ( $Filter\_Flag == ALL\_TRUE$ ) (4)
        output all pairs in the two cones (5)
      else if ( $Filter\_Flag != ALL\_FALSE$ ) (6)
        if  $c_2$  is a leaf node (7)
          for all pair  $s_1$  from  $c_1$  and  $s$  from  $c$  do (8)
             $High\_Corr\_Flag = Instance\_level\_Join(s_1, s_2, \theta)$ ; (9)
            if ( $High\_Corr\_Flag$ ) output  $s_1$  and  $s_2$ ; (10)
          else for each  $c'$  of  $c$ 's children do (11)
             $Similarity\_Join(c_1, c', \theta)$  (12)
  
```

---

For a join between two spatial autocorrelation-based search trees, we traverse one tree in a threaded-leaves-only manner and traverse the other tree in either a threaded-leaves-only manner (single loop join) or a tree-based manner(nested loop join). Other traversal combinations such as tree matching are also possible but are beyond the scope of this paper; they will be addressed in future work. For each leaf  $c_1$  in the first search tree, a process similar to the range query with respect to  $c_1$  is carried out.

**Example 3 (A Similarity Join)** The join operation between the cones in Figure 3 (a) and (c) is applied as shown in Table 2. For the nested loop join, each leaf ocean cone is checked with the land cones. The cost of the threaded-leaves-only traversal is the sum of 16, which is the filtering cost, and  $2 \times 16$ , which is the refinement cost. For the single loop join, each ocean cone is checked with the land cones starting with the root  $L$ . Its children will be visited only if neither the All-True or All-False lemmas turns out to be true. As can be seen, some All-False cone pairs and All-True cone pairs are detected in the non-leaf nodes and their descendents are not visited at all. The cost of the tree-based traversal is the sum of 12, which is the filtering cost, and  $2 \times 16$ , which is the refinement cost.

Tree-Based Traversal			Threaded-leaves-only Traversal		
Ocean-Land	Filtering	Refinement	Ocean-Land	Filtering	Refinement
$O_1 - L$	No				
$O_1 - L_1$	No	16	$O_1 - L_1$	No	16
$O_1 - L_2$	All-False		$O_1 - L_2$	All-False	
$O_1 - L_3$	All-False		$O_1 - L_3$	All-False	
$O_1 - L_4$	All-False		$O_1 - L_4$	All-False	
$O_2 - L$	All-False				
			$O_2 - L_1$	All-True	
			$O_2 - L_2$	All-True	
			$O_2 - L_3$	All-True	
			$O_2 - L_4$	All-True	
$O_3 - L$	No				
$O_3 - L_1$	All-True		$O_3 - L_1$	All-True	
$O_3 - L_2$	All-True		$O_3 - L_2$	All-True	
$O_3 - L_3$	All-True		$O_3 - L_3$	All-True	
$O_3 - L_4$	No	16	$O_3 - L_4$	No	16
$O_4 - L_4$	All-True				
			$O_4 - L_1$	All-True	
			$O_4 - L_2$	All-True	
			$O_4 - L_3$	All-True	
			$O_4 - L_4$	All-True	

Table 2: Join in Example Data

### 3.3 Completeness and Correctness

**Lemma 1 (Completeness and Correctness of the Range Query Algorithm)** *The Similarity-Range-Query algorithm is complete and correct.*

**Proof Sketch:**

Given a query time series  $T_q$ , for the threaded-leaves-only traversal, a pair of time series  $T_q$  and  $T'$  having a correlation value greater than the user given threshold can only be dismissed when it is in a pair of cones satisfying the All-False lemma or in Instance-level-Join (step 8 in Algorithm 3). The All-False lemma ensures no false-dismissal in the first case and the instance level pairwise checking will not false dismiss either. Any pair of time series found having a correlation value greater than the user given threshold either comes from an All-True cone pair (step 4 in Algorithm 3) or from individual correlation checking (step 9 in Algorithm 3). The All-True lemma ensures no false-admission in the first case and the individual checking will not false admit any pair either.

Given a query time series  $T_q$ , for the tree-based traversal, pairs formed by  $T_q$  and individual time series in a non-leaf node will be output if they satisfy the All-True lemma; pairs formed by  $T_q$  and individual time series in a non-leaf node will be dismissed if they satisfied the All-False lemma. The children will not be visited in both of these cases. This will not result in a false dismissal or false admission for any pair because of the All-True, All-False lemmas and the fact that the union of the time series sets of a non-leaf node's children is the same as the time series set of their parent. The children of a non-leaf node which does not satisfy the two lemma will be visited recursively. As in the threaded-leaves-only traversal, the leaf node will also be checked against the All-True and All-False lemmas. The completeness and correctness can be argued similarly.

**Lemma 2 (Completeness and Correctness of the Join Algorithm)** *The Similarity-Join algorithm is complete and correct.*

**Proof Sketch:**

The Similarity-Join algorithm is similar to the Similarity-Range-Query algorithm with a set of query time series organized as threaded leaves. The completeness and correctness proofs are similar to those in Lemma 1.

## 4 Performance Evaluation

I wanted to answer two questions: (1) How do the two query strategies improve the performance of correlation-based similarity range query processing? (2) How do the two query strategies improve the performance of correlation based similarity join processing?

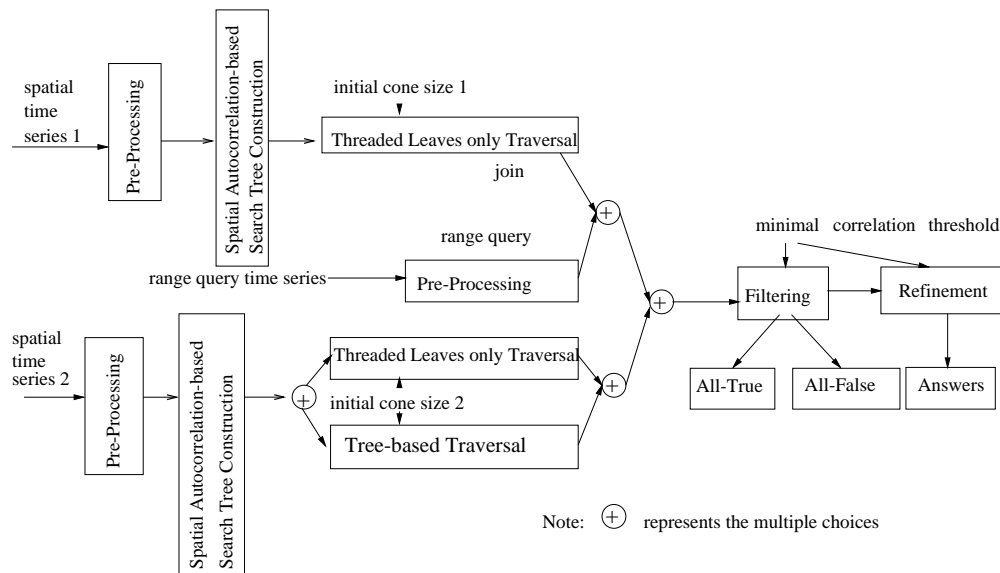


Figure 4: Experimental Design

I evaluated the performance of the proposed query processing strategies with a dataset from NASA Earth science data [12]. In this experiment, correlation-based similarity queries were carried out between the Sea Surface Temperature (SST) in the eastern tropical region of the Pacific Ocean (80W - 180W, 15N - 15S) and Net Primary Production (NPP) in the United States. The NPP time series from 2901 land cells of the United States and the SST time series from 11556 ocean cells of the eastern tropical region of the Pacific Ocean were obtained under a 0.5 degree by 0.5 degree resolution. The records of NPP and SST were monthly data from 1982 to 1993.

Figure 4 describes the experimental setup to evaluate the different strategies for similarity range query and join processing. As I noted in Section 3, there are two proposed strategies for the query processing: threaded-leaves-only traversal and tree-based traversal. I

investigated the two strategies for range similarity queries. For the similarity joins, I chose the threaded-leaves-only traversal for the outer loop, and I evaluated the two query strategies in the search tree of the inner loop.

## 4.1 Correlation-based Similarity Range Query Processing

This section describes a group of queries carried out to show the savings of the two strategies for a correlation-based range similarity queries. The SST data for the eastern tropical region of the Pacific ocean was chosen as the inner loop to construct a spatial autocorrelation-based search tree. The query time series were from the NPP data in the United States. I carried out the range queries in the spatial autocorrelation-based search tree for SST. All time series in SST, which correlates with the query NPP time series over the given minimal correlation threshold  $\theta$ , are retrieved. I chose the starting cone size for the eastern tropical region of the Pacific Ocean to be  $8 \times 8$ . (Assume that we have built the spatial autocorrelation-based search tree for the SST time series in the inner loop before we carried out the queries.)

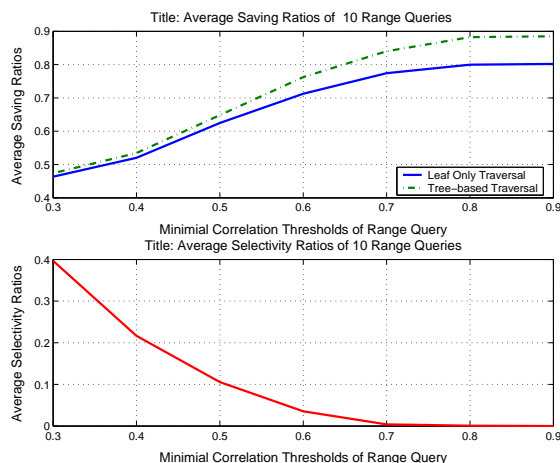


Figure 5: Savings and Selectivity Ratios for Range Query Processing

The brute force strategy scans all the time series in SST linearly. The cost of the brute force range queries is equal to  $|SST|$ , where  $|SST|$  denotes the number of time series in the SST data. Here I define the saving ratio as the percentage of cost savings of a range query processing compared to the cost of a range query using the brute force strategy measured

in the unit of number of correlation computations. And I define the average saving ratio for multiple range queries as the mean saving ratio for these range queries. I define the selectivity ratio for a range query as the fraction of query results of time series among all the time series in the dataset. And I define the average selectivity ratio for multiple range queries as the mean selectivity ratio for these range queries.

I randomly chose 10 NPP time series from the United States and carried out the correlation-based similarity range queries using the two different strategies respectively with the SST data from the eastern tropical region of the Pacific Ocean. The geographical locations of the 10 query time series were widely spread in the United States. The average selectivity ratios for the 10 queries at the different minimal correlation thresholds are illustrated in the lower plot of Figure 5. As the minimal correlation threshold increased from 0.3 to 0.9, the average selectivity ratio decreased from 0.4 to 0. The average saving ratios using the two query strategies for the 10 queries at the different minimal correlation thresholds (0.3-0.9) are presented in the upper plot of Figure 5. The solid line represents the average saving ratios for the threaded-leaves-only traversal strategy, which range from 0.46 to 0.80. The dash-dot line represents the average saving ratios for the tree-based traversal strategy, and the saving ratios range from 0.48 to 0.89.

As the selectivity ratio decreases, more and more non-leaf nodes(cones) in the search tree are identified as All-True or All-False cones in the query processing using the tree-based strategy. Thus the tree-based strategy often outperformed the threaded-leaves-only strategy as the selectivity ratio decreased.

## 4.2 Correlation-based Similarity Join Processing

This section describes a group of experiments carried out to show the net savings of the two strategies for the correlation-based similarity joins. The NPP time series dataset for the United State was chosen as the outer loop. As I discussed in the selection of parameter, the cone size for the NPP data was fixed at  $1 \times 1$ . The SST time series data for the eastern tropical region of the Pacific Ocean was chosen as the inner loop. A spatial autocorrelation-based search tree was constructed for the SST data. (Assume that we have built the spatial autocorrelation-based search trees before we carried out the similarity join

operations.)

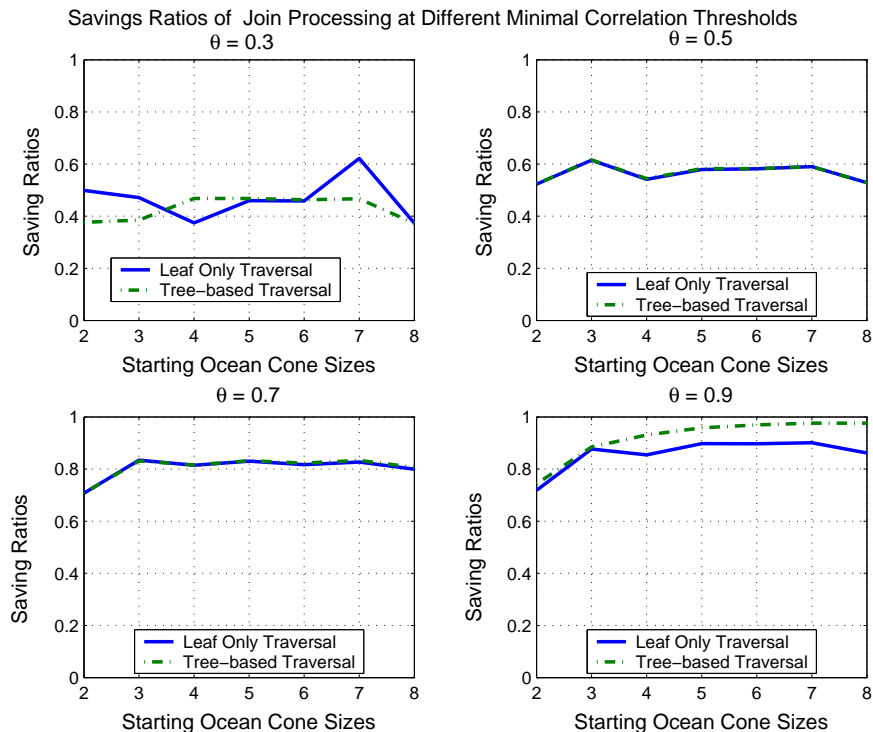


Figure 6: Savings for Join Processing

The cost of a brute force strategy is  $|NPP| \times |SST|$ , where  $|NPP|$  and  $|SST|$  are the number of the time series in NPP and SST respectively. Here I define the saving ratio as the percentage of cost savings of a join processing compared to the cost of a join using a brute force strategy measured in the unit of number of correlation computations. I define the selectivity ratio for a join as the fraction of join results of time series among the cross product of the two spatial time series datasets.

The selectivity ratios for the join processing of the NPP data and SST data are shown in Table. 3. As the minimal correlation threshold of the joins increased from 0.3 to 0.9, the selectivity ratio decreased from 0.39 to 0.

The saving ratios of the join processing using the two strategies are shown in Figure 6. Each subplot represents the saving ratios of the join processing for the two strategies using the search tree beginning with the different starting ocean cone sizes at a fixed minimal

correlation threshold. The starting cone sizes for the eastern tropical region of the Pacific Ocean vary from  $2 \times 2$  to  $8 \times 8$ . The saving ratios were presented at the different minimal correlation thresholds as shown in Figure 6.

The saving ratios of the join processing using both strategies increases as the minimal correlation threshold of the joins increases. When the selectivity ratio is high, more leaf nodes(cones) are possibly traversed in the join processing using the tree-based strategy. The threaded-leaves-only strategy often tends to outperform the tree-based strategy at a high selectivity ratio. However, the tree-based strategy often outperformed the threaded-leaves-only strategy as the selectivity ratio was decreased.

Minimal Correlation Thresholds	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Selectivity Ratios	0.39	0.22	0.11	0.04	0.005	0.0006	0

Table 3: Selectivity Ratios for the Join between NPP data and SST data

In summary, the experimental results show that the query processing using the two query strategies saves a large fraction of the computational cost. The performance of the query processing using the two strategies is robust to the starting cone sizes, and it offers stable savings for the different starting cone sizes.

## 5 Conclusion and Future Work

I investigated the processing strategies for correlation-based similarity range queries and joins using a spatial autocorrelation search tree. Experimental evaluations with Earth science data show that the performance of the query and join processing strategies using the spatial autocorrelation-based search tree structure saves a large fraction of computational cost.

In future work, I would like to explore other search tree candidates, such as k-d tree, R-tree, and R-link tree. I will also carry out a comparison study between the proposed query processing strategies with other temporal indexing techniques [1, 5] in spatial time series data.

## Acknowledgments

I am particularly grateful to my advisors, Prof. Shahsi Shekhar and Prof. Vipin Kumar, and collaborator Yan Huang at the University of Minnesota for their helpful comments and valuable discussions.

## References

- [1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search In Sequence Databases. In *Proc. of the 4th Int'l Conference of Foundations of Data Organization and Algorithms*, 1993.
- [2] L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. Vitter. Scalable Sweeping-Based Spatial Join. In *Proc. of the 24th Int'l Conf. on VLDB*, 1998.
- [3] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, 1994.
- [4] B.W. Lindgren. *Statistical Theory (Fourth Edition)*. Chapman-Hall, 1998.
- [5] K. Chan and A. W. Fu. Efficient Time Series Matching by Wavelets. In *Proc. of the 15th ICDE*, 1999.
- [6] N. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, 1991.
- [7] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison Wesley, 2002.
- [8] Christos Faloutsos. *Searching Multimedia Databases By Content*. Kluwer Academic Publishers, 1996.
- [9] R. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu, editors. *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers, ISBN: 1-4020-0033-2, 2001.
- [10] D. J. DeWitt J. M. Patel. Partition Based Spatial-Merge Join. In *SIGMOD*, 1996.
- [11] S. T. Leutenegger and M. A. Lopez. The Effect of Buffering on the Performance of R-Trees. In *Proc. of the ICDE Conf.*, pp 164-171, 1998.
- [12] C. Potter, S. Klooster, and V. Brooks. Inter-annual Variability in Terrestrial Net Primary Production: Exploration of Trends and Controls on Regional to Global Scales. *Ecosystems*, 2(1):36-48, 1999.
- [13] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases: With Application to GIS*. Morgan Kaufmann Publishers, 2001.
- [14] J. Roddick, K. Hornsby, and M. Spiliopoulou. An Updated Bibliography of Temporal, Spatial, and Spatio-Temporal Data Mining Research. In *First Int'l Workshop TSDM*, 2000.
- [15] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, ISBN:0130174807, 2003.
- [16] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.T. Lu. Spatial Databases: Accomplishments and Research Needs. *IEEE TKDE*, 11(1), 1999.
- [17] G. H. Taylor. Impacts of the El Nio/Southern Oscillation on the Pacific Northwest. [http://www.ocs.orst.edu/reports/enso\\_pnw.html](http://www.ocs.orst.edu/reports/enso_pnw.html).
- [18] W.R. Tobler. *Cellular Geography, Philosophy in Geography*. Gale and Olsson, Eds., Dordrecht, Reidel, 1979.
- [19] Michael F. Worboys. *GIS - A Computing Perspective*. Taylor and Francis, 1995.
- [20] Pusheng Zhang, Yan Huang, Shashi Shekhar, and Vipin Kumar. Correlation Analysis of Spatial Time Series Datasets: A Filter-and-Refine Approach. In *the Proc. of the 7th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2003.