JOURNAL OF SPATIAL INFORMATION SCIENCE
Number 6 (2013), pp. 119–153 doi:10.5311/JOSIS.2013.6.102

RESEARCH ARTICLE

Using trend clusters for spatiotemporal interpolation of missing data in a sensor network

Annalisa Appice[1], Anna Ciampi[1], Donato Malerba[1], and Pietro Guccione[2]

[1]Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro, Italy
[2]Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Italy

Received: July 15, 2012; returned: September 17, 2012; revised: October 15, 2012; accepted: November 17, 2012.

Abstract: Ubiquitous sensor stations continuously measure several geophysical fields over large zones and long (potentially unbounded) periods of time. However, observations can never cover every location nor every time. In addition, due to its huge volume, the data produced cannot be entirely recorded for future analysis. In this scenario, interpolation, i.e., the *estimation* of unknown data in each location or time of interest, can be used to supplement station records. Although in GIScience there has been a tendency to treat space and time separately, integrating space and time could yield better results than treating them separately when interpolating geophysical fields. According to this idea, a spatiotemporal interpolation process, which accounts for both space and time, is described here. It operates in two phases. First, the exploration phase addresses the problem of interaction. This phase is performed on-line using data recorded from a network throughout a time window. The *trend cluster discovery* process determines prominent data trends and geographically-aware station interactions in the window. The result of this process is given before a new data window is recorded. Second, the estimation phase uses the *inverse distance weighting* approach both to approximate observed data and to estimate missing data. The proposed technique has been evaluated using two large real climate sensor networks. The experiments empirically demonstrate that, in spite of a notable reduction in the volume of data, the technique guarantees accurate estimation of missing data.

Keywords: spatiotemporal data mining, interpolation, clustering, sampling, time-series regression, trend discovery

1 Introduction

Interpolation is a key technique used to supplement, smooth, and standardize observational data. Historically it has been considered a crucial task in spatial data analysis. Consequently, a plethora of spatial interpolation methods exist in the literature, both deterministic, such as inverse distance weighting [39] and radial basis functions [28], and stochastic, such as Kriging [5]. These techniques have been largely used to support transformations between different discrete and continuous representations of a geophysical field: to change over from irregular point or line data to a raster representation or to re-sample between different raster resolutions [33].

So far spatial interpolation methods have been mainly designed to estimate any unknown measure of a geophysical field at unsampled locations in the space. The estimate is often based upon actual measures which are spatially sampled in a geographic information system (GIS). The rationale behind this purely spatial estimate of a field is well-captured by Waldo Tobler's First Law of Geography: "everything is related to everything else, but near things are more related than distant things." Any spatial interpolator accounts for this law, including within its formulation the consideration of a stronger correlation between data points which are closer than for those that are farther apart.

The recent ubiquity of sensing technologies is providing a huge availability of spatiotemporal data. As a consequence, spatiotemporal interpolation is becoming an emerging research area which aims at solving critical problems that could not be solved using only spatial interpolation methods. Even though spatial interpolation theory has a long history, the problem of interpolating a dynamical (i.e., evolving with time) geophysical fields need to be addressed in a space-time domain. By following a spatiotemporal approach which has recently emerged in GIScience literature (e.g., [15], [10], [47]), traditional spatial techniques for the interpolation of geophysical fields have been supplemented by temporal methods, in order to handle spatiotemporal information properly. Two main strategies have been followed to interpolate data in a continuous space-time domain: the spatial-interpolation-primitive strategy and the temporal-interpolation-primitive strategy. The spatial-interpolation-primitive strategy reduces spatiotemporal data to a sequence of snapshots, so that spatial interpolations can be employed [27]. In this way, the interpolator is once again spatially defined and temporal interpolation is carried out on interpolation results obtained at the same location over consecutive snapshots [2]. In the temporal-interpolation-primitive strategy, a time series of data is temporally interpolated at every location, then the interpolated values are used as a sampled measurement in ordinary spatial interpolation functions [24]. This strategy is expensive to implement because temporal interpolation functions need to be calculated at every spatial location with at least one measurement at a time. On the other hand, the use of a spatiotemporal interpolation-primitive strategy has recently been proposed in the literature. In particular, the authors of [26] have advocated the importance of interpolating data by accounting for the existence of a temporal pattern on spatial measures of time-evolving geophysical fields. This consideration highlights the importance of detecting the spatiotemporal knowledge that models the observed data. This knowledge can be considered to achieve robust spatiotemporal interpolation functions. In particular, the interpolation approach in [26] (adopted to interpolate air-pollution activity) is based on a naive spatiotemporal interpretation of Tobler's Law, according to which pollution data that is closer in the *space-time* domain is expected to be more alike than those that are farther apart. Based upon this idea, the formulation in [26] can be considered a view limited to the expectation of constant or nearly constant trends in time.

In this paper, the spatiotemporal strategy is pursued and an interpolator, called TRECI (trend cluster based spatiotemporal interpolator), is defined. This method is designed to interpolate geophysical fields, whose spatial and temporal evolution is periodically monitored through a remote sensing scenario. The prominent spatial and temporal patterns which emerge from the evolution of a field are modeled by means of *trend clusters*. An inverse distance weighting interpolation is tailored to estimate the unknown field value at any location and at any time. The estimate is based on knowledge derived from the observed trend clusters.

Trend cluster discovery is achieved as an on-line process. According to the definition in [1], the input (data snapshots) of a system that performs the process on-line does not arrive as a batch, but as a sequence of input portions (i.e., time-defined windows of snapshots). The system reacts in response to each incoming window giving in output the set of discovered trend clusters. A trend cluster [4] is a cluster of sensors which transmits data whose temporal variation, called a trend polyline, is similar along a time horizon. Trend clusters discovered from each window of snapshots are permanently stored in a database. They provide a compact model of prominent spatial and temporal dynamics in data. In particular, the cluster segmentation of sensors determines a *partitioning* of the networked surface, while each trend associated with a cluster describes the evolution of that cluster over time. Intuitively, the sensors grouped in a cluster provide a discrete representation of the region covered by the cluster. A shape-dependent sampling technique is used to reduce the number of sensors needed to accurately describe the extent of each cluster. A polynomial is fitted on each trend time series, in order to capture the evolution of intra-cluster measures over time. For each trend cluster, both the sensors sampled in the cluster area and the polynomial approximation of the time series are stored in a database. This database constitutes the knowledge base for any future (spatiotemporal) interpolation.

The inverse distance weighting (IDW) interpolation scheme is used to estimate values at any spatial location. The choice of a deterministic interpolator like IDW is motivated by the comparative studies [19, 25, 26] pursued over the years. These studies conclude that stochastic Kriging methods approach the theoretical limit for the variance of the estimation error (due to the Gauss-Markov theorem), but only at the cost of significantly increasing the processing time [23]. On the other hand, deterministic methods, and in particular the inverse distance weighting methods, share the strength of being relatively accurate and fast interpolators. Specifically, simplicity, speed in calculation, ease of programming, and acceptable results for many types of geophysical data are all features that have led to the approach being widely adopted in remote sensing intensive applications. In TRECI, the IDW mechanism outputs a weighted average of the nearby points of the trend cluster representation for the observed data in the spatiotemporal proximity of the unknown point.

Therefore, the main contributions of this paper are:

1. the use of trend cluster discovery to model a trend-based partitioning of the networked surface; this partitioning encapsulates the spatiotemporal knowledge required to interpolate the field at any location and at any time;
2. the determination, for each trend cluster, of a shape-dependent sample of clustered sensors and of a polynomial trend time series, stored in the database; and

3. the exploitation of the inverse distance weighting interpolation which provides, for each geographic location (x, y) and time point t, a weighted average of the key sensors which are close to (x, y) at a time horizon comprising t.

The paper is based on a preliminary work [3], where the idea of a spatiotemporal integrated trend cluster interpolation was presented. However, the previous paper is extended in the following directions:

1. an extensive discussion of the related works on spatiotemporal interpolators (see Section 2);
2. a revised and detailed illustration of the trend cluster discovery algorithm (see Sections 3 and 4.1);
3. the evaluation of simple linear interpolation as an alternative to the polynomial interpolation (see Section 4.3.2); and
4. new experiments, including an additional sensor data network use to evaluate how TRECI is able to capture on-line the spatiotemporal knowledge required for any future interpolation. Experiments also highlight that TRECI is able to work for large networks, producing accurate interpolation even when the transmission frequency is lowered and/or the sparseness of the transmitting sensors is increased (see Section 5).

The paper is organized as follows. In Section 2 we discuss the related works regarding spatial and spatiotemporal interpolators together with the main motivations of this work. In Section 3 we introduce the snapshot data model for sensor data and we define the trend clusters. In Section 3.3 we provide the formulation of the spatiotemporal interpolation task in a sensor network. In Section 4 the TRECI spatiotemporal interpolation is described. Results from the two real datasets and discussions are reported in Section 5 and conclusions are drawn in Section 6.

2 Related work and contribution

Studies of spatial interpolation were initially encouraged by the geographical and geophysical analysis of ore mining, water extraction or pumping, and rock inspection [6]. In these application fields, interpolation methods are required as the main resource to recover unknown information and account for problems like missing data, energy saving, sensor default; as well as to provide support for data summarization, and investigation of spatial correlation between observed data [22]. More recently, interpolation methods have been needed to handle spatiotemporal data, potentially in a streaming scenario. This paper contributes to the investigation of spatiotemporal interpolators in a remote sensing scenario.

2.1 Spatial interpolators

The spatial interpolative primitives, integrated in the majority of geographic information systems, estimate a geophysical quantity at any geographic location where the field measure is not available. The interpolated value is derived by making use of the knowledge of the nearby observed data and, sometimes, of some hypotheses or supplementary information on the data field. Inverse distance weighting (IDW) [39], radial basis functions (RBF) [28] and Kriging [5] are the most common spatial techniques adopted in these cases. These techniques are used to deal with the irregular sampling of the investigated area [16, 41] or with the difficulty of describing the area by the local atlas of larger and irregular manifolds. IDW and RBF, which are both deterministic interpolators, use mathematical functions to calculate an unknown field value in a geographic location based either on the degree of similarity (IDW) or the degree of smoothing (RBF) in relation to neighboring data points. Both methods share with Kriging the idea that the collection of field observations can be considered as a product of a correlated spatial random field with specific statistical properties. In Kriging this correlation is used to derive a second-order model of the field (the variogram). The variogram represents an approximate measure of the spatial dissimilarity of the observed data. IDW interpolation is based on a linear combination of nearby observations with weights proportional to a power of the distances. It is a heuristic but efficient approach justified by the typical power-law of the random field spatial correlation. In this sense, IDW accomplishes the same strategy adopted by the more rigorous formulation of Kriging [19, 25, 26].

Several studies have arisen from these base spatial interpolation approaches. In [44], the missing data of a dense network are recovered by a Kriging interpolator. By considering that the computational complexity of a variogram is cubic in the size of the observed data [6], the variogram calculus, in this study, is sped-up by processing only the areas with information holes, rather than the global data. In [42], IDW and 1-nearest neighbors have been used to interpolate a grid of rainfall data and re-sample data at multiple resolutions. In [30], IDW is again investigated and formulated in an adaptive way which depends on the varying distance-decay relationship in the area under examination. The weighting parameters are varied according to the spatial pattern of the sampled points in the neighborhood. The method proves more efficient than ordinary IDW and in several cases also better than Kriging. These studies contribute to highlighting IDW as a deterministic, quick, and simple interpolation method which also provides accurate interpolation results. On the other hand, Kriging is based on the statistical properties of the random field and, hence, is expected to be more accurate regarding the general characteristics of the observations and the efficacy of the model. In any case, the accuracy of Kriging is highly dependent on a reliable estimation of the variogram [16, 38] and the variogram computation cost scales as the cube of the number of observed data [5]. This cost is prohibitive in evolving sensing environments, where statistical properties of a monitored field may change over time. In data mining, the change of the underlying properties over time is usually called concept drift [46]. It is noteworthy that the concept drift, expected in evolving data, can be a serious complication for Kriging. In fact, it may impose the repetition of costly computation of the variogram each time the statistical properties of the field change significantly. On the other hand, experimental studies reported in the literature (e.g., [30]) show that the accuracy of an IDW interpolator often approaches the accuracy of a Kriging interpolator, especially for smooth fields [12]. These considerations motivate the use of an interpolator which is accurate enough and whose learning phase can be reasonably run on-line with the streaming activity.

2.2 Spatiotemporal interpolators

Recently more research efforts have focused on joining traditional temporal data mining techniques with spatial interpolators. The main purpose of these studies is to transfer mature temporal data mining techniques into a joint spatiotemporal set of interpolation methods able to catch the geophysical nature of data that is both spatially and temporally correlated. The interpolation methods are based on the idea that the sequence of observations coming from a sensor can be regarded as outcomes of a stochastic process corrupted by random noise. Hence, the model of such processes can be described (and then predicted) by means of relatively few parameters [9, 14, 40].

Initial studies offered a partial integration of the spatial and temporal methods by firstly performing spatial interpolation and then reducing temporal interpolation to the application of simple methods (such as linear or spline interpolation, [32, 36]) to the sequence of snapshots of spatially interpolated data [26]. The alternative has also been explored, i.e., time series of data were temporally interpolated for each relevant location and then were used as sampled observations for the application of a traditional spatial interpolator [24].

The true integration of the spatial and temporal components is a relatively new research field. It is based on the application of a dynamic model, like the Kalman filter [18], or the Markov Random field [17], to consecutive snapshots of data, so that the spatial interpolation takes place according to a set of temporally changing parameters. In [20], Kriging is used for the spatial interpolation of medical images, but the statistical model of the variogram is updated according to a Kalman filter of the temporal observations. In [11] the impact of an irregular grid of sensors on data compression is analyzed and the nearest neighbor is proposed as the interpolator scheme to obtain a better data compression. Instead temporal interpolation is adopted to assess the possible sensor clock misalignment, but no solution is provided to provide an estimate in any spatiotemporal location of the sensed area. In [37] a methodology for the spatial and temporal interpolation of air quality data is illustrated. The methodology has two steps. First, non-stationary time series analysis methods are used to interpolate the data sets over periods where measurements are missing and to decompose the time series into trend and harmonic components. Then a preliminary analysis of spatial relations within the data sets and a spatiotemporal model of log-transformed data are computed. The model consists of trend and noise and represents the spatiotemporal variations in the data applied to predict the air pollution variations at unsampled points across time and space.

2.3 Paper contribution

The spatiotemporal interpolation algorithm proposed in this paper provides a regressive time dynamic model of the random field. This model is a polynomial model of a spatiotemporal aggregate (named trend cluster) of the data field. The IDW interpolator is adapted to exploit such a time-variant spatial model and perform an estimation of data in any location of the space-time domain. Hence, the novelty of the proposed method consists in the usage of summarized spatiotemporal information from the sensed observations to infer accurate IDW interpolated values. According to the discussion reported in Section 2.1, the choice of an IDW interpolator allows the efficient and accurate recovery of the original data discarded by the summarization process, as well as the achievement of relatively accurate estimated measures where they are not available, supposing the general hypotheses on the random field under examination to be reasonable (i.e., a power-law decaying correlation function for spatially stationary data [16] and a slow time-varying evolution of the statistics of the observations).

It is worth noting that a distinctive characteristic of the proposed algorithm is that it has been specifically designed to deal with a continuous stream of spatiotemporal data.

This represents a major difference with respect to the traditional spatiotemporal interpolators that are based on the analysis of a volume of spatiotemporal data that, although big enough, are always bounded in time. In a sensing application, a (large) amount of geo-referenced data arrives continuously at a high rate and is possibly subjected to data distribution drifts. This work advocates the necessity of an interpolator which accounts for this intrinsic dynamism in a sensor data stream.

3 Basics and problem formulation

The streaming environment considered in this work basically defines an in situ-sensing scenario which can be represented by the following premises.

1. The sensor stations are enumerated with a progressive number in the network and geo-referenced to geographic (latitude and longitude) coordinates.
2. The sensor stations routinely measure the random field at consecutive time points and transmit measured data to a central server; stations are synchronized at the transmission time.
3. The spatial location of each networked station is known, distinct, and invariant in the network, while the transmission status of a station may change in time since a sensor can be switched on or off at any time.

Concerning premise (2), it is noteworthy that for many networks used to monitor geophysical fields (e.g., pollution, weather, irradiation) measures occur at regular time intervals. In any case, the processing framework we propose here is more general and fits in both periodic and patchy transmissions. Based upon these premises, the snapshot model and the count-based window model can be adopted to represent the random field observations as they are sampled at a specific time through the network. The trend cluster model is chosen to represent the spatiotemporal knowledge of the sensor data dynamics and to formulate the interpolation task.

3.1 Sensor data stream and window model

Let T be an equally-spaced discretization of the time line and K the network of remote sensor stations which routinely measure the numeric geophysical random field Z at the consecutive time points of T. The scenario we consider is that of the measures of Z, sampled by K at the time points of T and sent to a server station. Field measures feed a sensor data stream that, in principle, can be stored at the server for any future analysis. The communication costs of forwarding the field measures from the sensor stations to the server station depend on the network topology and the communication protocol. The analysis of these protocols is out of the scope of this study.

The set of measures of Z sampled by K at a time point t is called field raw or snapshot.

Definition 3.1 (Snapshot or field raw)**.** *A snapshot of Z timestamped at t (with $t \in T$) is the pair $\langle K_t, z_t(\cdot)\rangle$ where K_t ($K_t \subseteq K$) is the set of switched-on stations which produced a data item at t and $z_t(\cdot)$ is the field function :*

$$z_t : K_t \mapsto Z, \tag{1}$$

which assigns the sensor geographic coordinates $(x, y) \in K_t$ to the reading of Z at t.

Though finite, K_t may vary with t as sensors may be switched on or off at a specific time across the network. At the server station a snapshot can be buffered in an associative structure which maps a station of K_t to a value of Z.

Definition 3.2 (Sensor data stream)**.** *A sensor data stream $z(T, K)$ is the unbounded sequence of snapshots which arrives continuously, possibly at high rate, from a remote sensor network K at the consecutive time points of T.*

The count-based window [9] is a well-known data model, largely adopted in data stream mining, concerning the data analysis of the consecutive data segments of a stream.

Definition 3.3 (Count-based window model)**.** *Let w be a window size. A count-based window model decomposes $z(T, K)$ into non-overlapping windows, each one composed of w consecutive snapshots, that is:*

$$z(T, K) = \underset{t_1 \to t_w}{\mathrm{W}} z(T, K), \underset{t_{w+1} \to t_{2w}}{\mathrm{W}} z(T, K), \ldots, \underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{W}} z(T, K), \ldots \tag{2}$$

where the (i)-th window $\underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{W}}$ comprises snapshots $z(T, K)$, acquired from the time t_{iw+1} to $t_{(i+1)w}$ of T.

Once a count-based window model is adopted to process a sensor data stream, a buffer is expected to consume snapshots as they are produced by the remote sensor network. It "pours" snapshots, window-by-window, into a data-stream knowledge discovery process.

3.2 Trend cluster pattern

A trend cluster is a kind of spatiotemporal pattern [4] that can describe simultaneously both spatial clusters and data trend dynamics, which emerge in a numeric sensor data stream $z(T, K)$.

Definition 3.4 (Trend cluster)**.** *Let $z(T, K)$ be a sensor data stream, a trend cluster in $z(T, K)$ is the triple:*

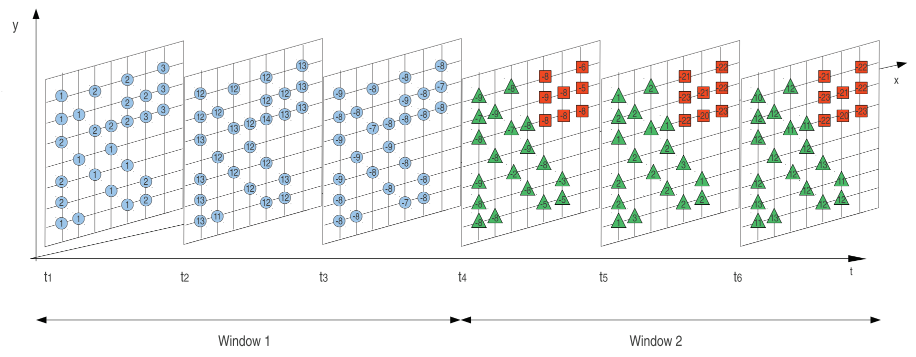
$$(t_i \to t_j, c, \hat{z}), \tag{3}$$

where

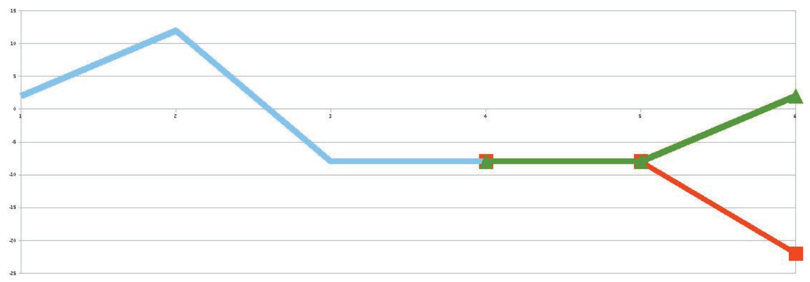
1. *$t_i \to t_j$ is a time horizon such that $t_i, t_j \in T$ and $t_i < t_j$;*
2. *c enumerates spatially close sensors whose series of field readings in $z(T, K)$ are similar to each other in the time horizon from t_i to t_j (or equivalently field data vary according to a similar trend time series prototype from t_i to t_j); and*
3. *$\hat{z}$ is the trend prototype of c from t_i to t_j; it is a time series of cluster aggregate values $\hat{z}(t)$, one value for each time point $t \in [t_i, t_j]$; each $\hat{z}(t)$ is the aggregate (median) of the measures of Z sampled across c at t, that is,*

$$\hat{z}(t) = \underset{p \in c}{\mathrm{median}}\, z_t(p). \tag{4}$$

In a count-based window model of the sensor data stream, the trend clusters can be discovered with the time horizon defined on the time window segmentation (see Figure 1). Trend clusters can be discovered to segment a data window and provide a compact and informative spatiotemporal representation of the field evolution. Trend clusters can be stored in a database for future analysis (e.g., interpolation) in place of the windowed snapshots, which are discarded.



(a) Clusters



(b) Trends

Figure 1: Trend clusters discovered in a count-based window model of a sensor data stream with $w = 3$. The blue cluster groups circle sensors whose measures vary as the blue time series from t_1 to t_3. The red cluster groups square sensors whose measures vary as the red time series from t_4 to t_6. The green cluster groups triangular sensors whose measures vary as the green time series from t_4 to t_6.

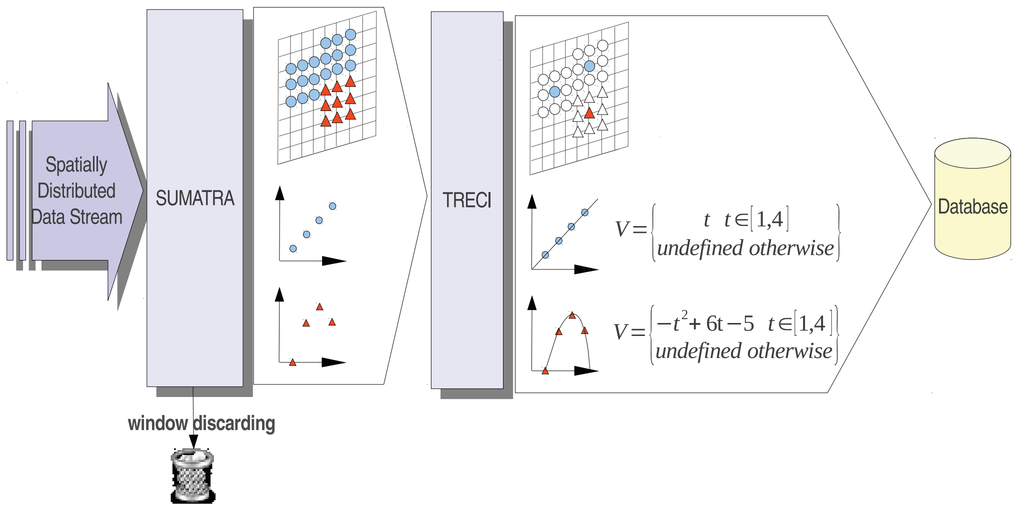
3.3 Interpolation problem formulation

Given the geophysical numeric random field Z, whose values are routinely measured through a remote sensor network K, the goal of the interpolation task is to estimate an (unknown) measure of such a field at any location of the network region and at any time of the whole monitoring period. We point out that the estimation of the field measure at locations out of the region covered by the network or at times out of the observation interval is called *prediction* and it requires further hypotheses on the data. This study does not cover the prediction task.

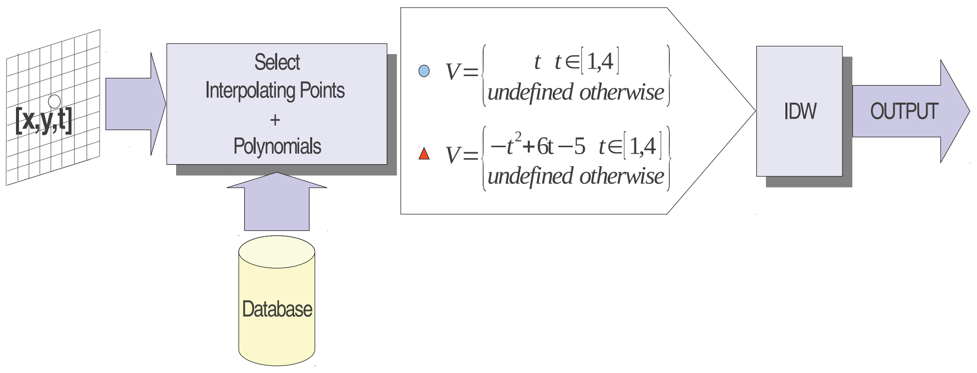
To address this spatiotemporal formulation of the interpolation task, a common evolution trend is supposed for observations at nearby sensors. Based upon such a hypothesis, the task of interpolating unknown data of a field is carried out by using the spatial clusters and data trend dynamics that underlie the field data sampled by a remote sensor network. This idea is contrary to that of interpolating directly row data measured at a specific time.

Let (x, y) be a pair of geographic coordinates and t be a time point. The estimate of Z at the space-time point (x, y, t) can then be achieved by the inverse distance weighted sum of the scatter values which surround (x, y) in the trend cluster representation of a processed sensor data stream.

4 Trend cluster aware inverse distance weighting


Spatially Distributed Data Stream
SUMATRA
window discarding
TRECI
$V=\begin{cases} t & t\in[1,4] \\ \text{undefined otherwise} \end{cases}$
$V=\begin{cases} -t^2+6t-5 & t\in[1,4] \\ \text{undefined otherwise} \end{cases}$
Database

(a) TRECI ONLINE


[x,y,t]
Select Interpolating Points + Polynomials
Database
$V=\begin{cases} t & t\in[1,4] \\ \text{undefined otherwise} \end{cases}$
$V=\begin{cases} -t^2+6t-5 & t\in[1,4] \\ \text{undefined otherwise} \end{cases}$
IDW
OUTPUT

(b) TRECI OFFLINE

Figure 2: TRECI: on-line summarization step and off-line interpolation step.

The interpolation algorithm TRECI operates in two phases. The on-line phase (see Figure 2(a)) consumes the data snapshots as they arrive from the remote sensor network and analyzes the buffered data snapshots, window-by-window, in order to determine a trend cluster segmentation of the data window. For each window, trend clusters model the spatial variation of the field along the time horizon of the window. Since this model is stored in a database, while windowed data are definitely discarded, trend clusters represent the data knowledge for the future off-line interpolation phase. For each trend cluster, a shape-dependent sample of clustered sensors (key sensors) is extracted. The sampling algorithm is designed to keep only the information which is useful to sketch the real spatial extent of the clustered region. At the same time, a (polynomial) regression model of the time law underlying the trend time series is determined and only regression coefficients are stored in the database as a model of the trend. The off-line phase (see Figure 2(b)), which is repeatable, retrieves the spatiotemporal knowledge surrounding the space-time point to be interpolated from the database. This knowledge is used to determine an IDW based estimate of the field. Details of the trend cluster discovery, the shape-based sampling, the time-series (polynomial) regression and the spatiotemporal IDW interpolation are reported in the following subsections.

4.1 Trend cluster discovery

The trend cluster discovery is performed by the algorithm SUMATRA whose implementation is now integrated as a component of the TRECI on-line framework. SUMATRA, originally presented in [4], is a sensor data stream mining algorithm able to process in real-time a window of data snapshots and to discover trend clusters as accurate and compact summarization patterns of windowed data. In its initial formulation SUMATRA stemmed from the non-realistic assumption that the number of sensors in the network does not change, while it is a matter of fact that sensors may be (temporally) switched off by causing missing values at some time points. On the other hand, it is also plausible that new sensors are switched on in the network, while old sensors are definitely switched off at the time of a long streaming activity. Unlike [4], TRECI accounts for the changeable topology of the sensor network and enhances the trend cluster discovery algorithm to estimate missing data on-the-fly and self-adapt to changes in the network topology. Consequently, the spatial closeness relation, according to which the sensors are virtually linked in the network structure, is now defined to self-adapt to the spatial density of those sensors actually operative at any time.

The core issues of the algorithm, including the spatial closeness relation between sensors; the trend similarity between sensors; the intra-cluster trend variability; and details of the trend cluster discovery algorithm itself are illustrated in the following subsections.

4.1.1 Core issues

As suggested in [31] the geographic distance between sensors can be used to determine if a sensors are spatially close or not.

Definition 4.1 (Spatial closeness relation)**.** *Let d be a distance threshold. Sensor A is close to sensor B, if A is at worst d far from B.*

It is reasonable to suppose that the threshold d is a function of the sensors spatial density. The value of the threshold d is automatically determined, at each window. For example, it is possibly to use the 90° percentile of the box plot of distances between the nearest sensors (neighboring distances) that are operative in the window:

$$d_i = \underset{90\%, p \in \mathbb{K}_i}{\text{box}} \, distance_{min}(p), \tag{5}$$

where:

1. i indicates the i-th w-sized window $\underset{t_{(i-1)w+1} \to t_{iw}}{\text{W}}$ in the sensor stream $z(K, T)$;
2. $\mathbb{K}_i$ is the set of sensors which measure at least once the field in $\underset{t_{(i-1)w+1} \to t_{iw}}{\text{W}}$, that is, $\mathbb{K}_i = \bigcup_{j=1,\ldots,w} K_{(i-1)w+j}$; and
3. $distance_{min}(p)$ is the distance between p and its nearest neighbor in $\mathbb{K}_i$, that is, $distance_{min}(p) = \min_{q \in \mathbb{K}_i, p \neq q} distance(p, q)$.

Note that the 90° percentile of the box plot of the neighboring distance is used in place of the maximum to avoid the selection of outlier values in the set of the considered neighboring distances.

The definition of the spatial closeness relation is used to characterize the spatial connectivity, according to which a pair of sensors can be grouped in a spatial cluster of sensors.

Definition 4.2 (Intra-cluster spatial-connectivity)**.** *Let c be a set of sensors and p and q be two sensors in c. Then p is spatially-connected to q across c iff:*

1. *p is spatially close to q (according to Definition 4.1), or*
2. *$r \in c$ exists such that p is spatially close to r and r is spatially connected to q across c.*

Then a spatially-aware cluster of sensors is defined.

Definition 4.3 (Spatially-aware cluster)**.** *Let c be a set of sensors; c represents a spatially-aware cluster iff for each $p, q \in c$ then p is spatially connected to q across c.*

By considering the time dimension of data, a measure of the trend similarity between sensors is defined.

Definition 4.4 (Trend similarity measure)**.** *Let p and q be two sensors which routinely sample data of the field Z. The trend similarity between the series of data of Z sensed from p and q along the time horizon $t_{(i-1)w+1} \to t_{(i)w}$ is computed as:*

$$tsim(p, q, Z, t_{(i-1)w+1} \to t_{iw}) = \max_{j=1,\ldots,w} |z_{(i-1)w+j}(p) - z_{(i-1)w+j}(q)|. \tag{6}$$

To compute the trend similarity measure between operative sensors whose data may be missing at a certain time, the median of the observed neighbor values in the snapshot is used. A missing data item is expected in the case of a sensor which is switched on in the window, but whose measured data item is not available at some time point of the window.

Definition 4.5 (Intra-cluster trend variability)**.** *Let c be a spatially-aware cluster of sensors. Let $\hat{z}$ be the trend prototype of c with time horizon $t_{(i-1)w+1} \to t_{iw}$ ($\hat{z}$ is computed according* *to Definition 1 like the time series of medians of field data sensed from c at each time point of* $t_{(i-1)w+1} \rightarrow t_{iw}$*). The intra-cluster trend variability of c along* $t_{(i-1)w+1} \rightarrow t_{iw}$ *is:*

$$trendVariablility(c, Z, t_{(i-1)w+1} \rightarrow t_{iw}) = \max_{j=1,\ldots,w} clusterVariability(c, Z, t_{(i-1)w+j}), \quad (7)$$

where:

$$clusterVariability(c, Z, t) = \max_{p \in c} |z_t(p) - \hat{z}(t)|, \quad (8)$$

with $z_t(p)$ *as the field value measured at the location of* p *and at the time* t *and* $\hat{z}(t)$ *the trend prototype value timestamped at t in* $\hat{z}$*.*

By considering a user-defined domain similarity threshold δ, the trend cluster discovery algorithm detects a spatially-aware cluster of sensors having the intra-cluster trend variability upper-bounded from δ.

4.1.2 Algorithm

The top-level description of the trend cluster discovery is reported Algorithm 1. The discovery process is triggered once the new window $\underset{t_{(i-1)w+1} \rightarrow t_{iw}}{\mathrm{W}}$ (Figure 3(a)) is completely streamed (Figure 3(b)) in $z(T, K)$.

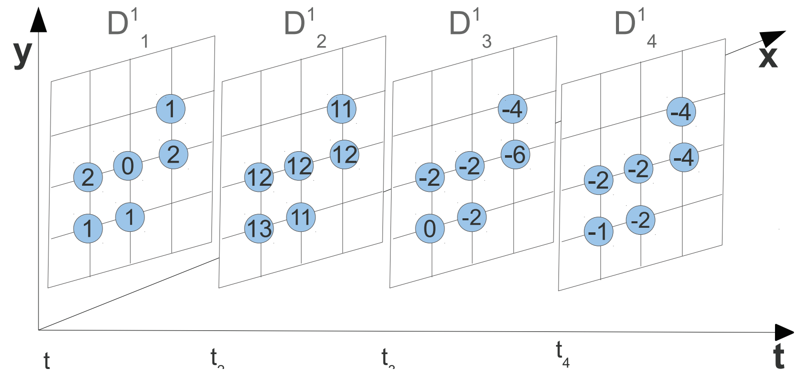
The computation starts by assigning $k = 1$, where k enumerates the computed trend clusters. An unclustered sensor p is randomly chosen as the seed of a new empty cluster c_k. Then, p is added to c_k (green cluster in Figure 4(a)) and the trend prototype $\hat{z}_k$ is constructed (by calling *trendPrototype*($\cdot$)). Both c_k and $\hat{z}_k$ are expanded by using p as the seed of the expansion process (by calling *expandCluster*($\cdot, \cdot, \cdot$)). The expanded trend cluster $[i, c_k, \hat{z}_k]$ is added to the pattern set $\underset{t_{(i-1)w+1} \rightarrow t_{iw}}{\mathrm{TC}}$. k is incremented by one and the clustering process is iteratively repeated until all the sensors are assigned to a cluster (Figure 4(e) and Figure 4(f)).

The expansion process is described in Algorithm 2. The expansion of $[c_k, \hat{z}_k]$ is driven by a seed node p and is recursively defined. First, the neighborhood $\eta(p)$ is constructed (by calling *neighborhood*($\cdot, \cdot$)) by considering the unclustered sensors which are spatially close to p (see Definition 4.1) and measure values of the field which are trend-similar to those measured by p (see Definition 4.4). Then the candidate cluster $tempC = c_k \cup \eta(p)$ and the associated trend prototype $\widehat{tempZ}$ are computed. The intra cluster trend variability of $[tempC, \widehat{tempZ},]$ is computed (by calling *trendClusterVariability*($\cdot, \cdot, \cdot$)). Two cases are distinguished:

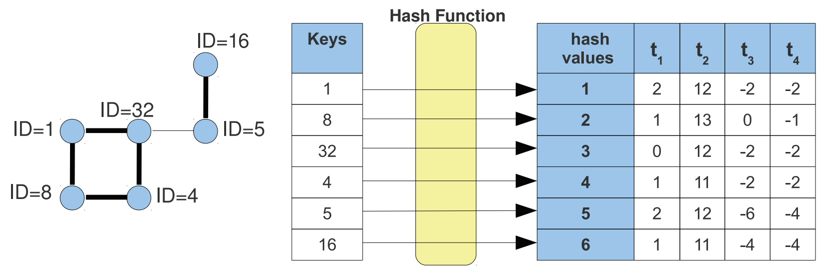
1. If $trendClusterVariability(tempC, \widehat{tempZ}, t_{(i-1)w+1} \rightarrow t_{iw}) \leq \delta$, then sensors of $\eta(p)$ are clustered into c_k (green cluster in Figure 4(b) and the last computed $\widehat{tempZ}$ is assigned to $\hat{z}_k$.
2. Otherwise the addition of each sensor of $\eta(p)$ to c_k is evaluated sensor-by-sensor.

In both cases, sensors newly clustered in c_k are iteratively chosen as seeds to continue the expansion process (grey circle in Figure 4(c)). The expansion process stops when no new sensor is added to the cluster (green cluster in Figure 4(d)).

The time complexity of trend cluster discovery is mostly governed by the number of $neighborhood()$ invocations. At worst, one neighborhood is computed for each sensor and



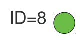
(a) A window of snapshots is completed in the stream



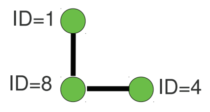
Keys	hash values	t_1	t_2	t_3	t_4
1	1	2	12	-2	-2
8	2	1	13	0	-1
32	3	0	12	-2	-2
4	4	1	11	-2	-2
5	5	2	12	-6	-4
16	6	1	11	-4	-4

(b) Window data storage

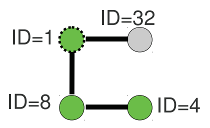
Figure 3: An example of a data window.



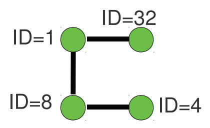
(a) Cluster seed selection



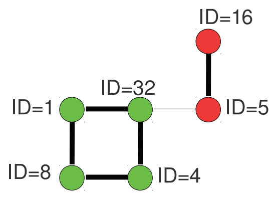
(b) Strong Neighborhood



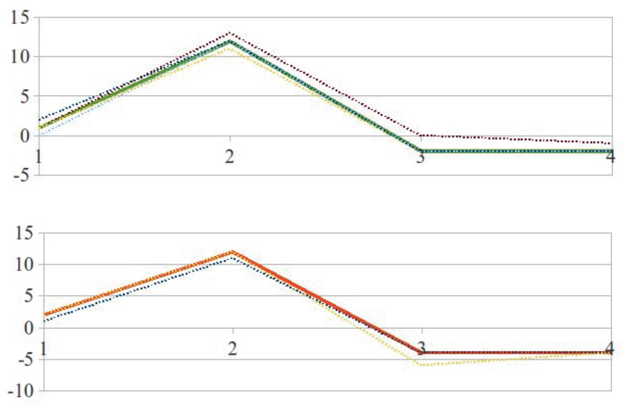
(c) Expansion seed



(d) Complete cluster



(e) Clusters



(f) Trend Prototypes

Figure 4: An example of trend cluster discovery by processing the data window shown in Figure 3.

evaluated in space and time. By using an indexing structure to execute such a neighborhood query and a quickselect algorithm (having linear time complexity) to compute the median aggregate, the time complexity of the trend cluster discovery in a window of k sensors and w snapshots is, at worst,

$$O(k(\underbrace{wlogk}_{neighbourhood()} + \underbrace{kw}_{trendPrototype()} + \underbrace{kw}_{trendClusterVariability()})).$$

4.2 Cluster shape-based sensor sampling

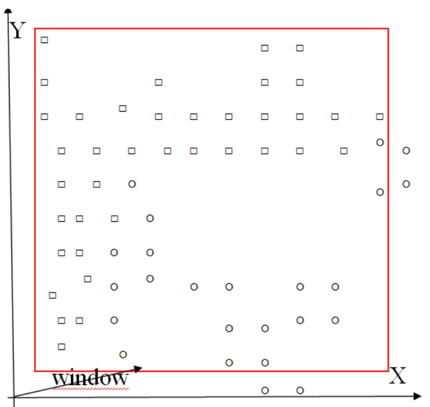
Let c be a cluster of sensors. The goal is to find a shape-based sample s of the key sensors grouped in c, such that s is stored in the database in place of s and is representative of the region covered from c for the purpose of interpolation. Random sampling is the simplest way to address this task, but it poses two issues. How can we choose the number of sensors to be sampled? How can we guarantee that the randomly selected sensors maintain the information on the cluster (region) shape? To answer both questions a sampling algorithm which resorts to a *quadtree* decomposition of the clustered region is presented. The quadtree decomposition is an adaptive sampling method largely used in image processing [21, 48]. The method is tailored to identify the key sensors of the cluster that are centroids in the densely populated subareas of the cluster itself. Thus, the number of sampled sensors and their location in space depend on how the cluster shape is spread across the space.

The sampling of a cluster is recursively performed according to Algorithm 3. First the minimum boundary rectangle Q of the cluster c is computed (Algorithm 3, line 2, see Figure 5(a) and Figure 5(d)). The minimum bounding rectangle, also known as minimum bounding box, is the rectangle enveloping c that is unambiguously identified by its left inferior vertex ($\min(x)$, $\min(y)$) and right superior vertex ($\max(x)$, $\max(y)$) with:

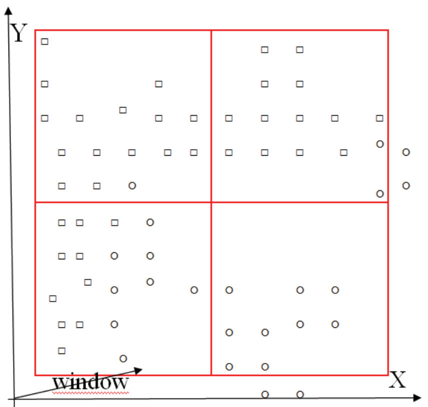
$$\begin{array}{rclcrcl} min(x) &=& \min\limits_{x} \{x|(x,y) \in c\} & & min(y) &=& \min\limits_{y} \{y|(x,y) \in c\} \\ max(x) &=& \max\limits_{x} \{x|(x,y) \in c\} & & max(y) &=& \max\limits_{y} \{y|(x,y) \in c\} \end{array} \tag{9}$$

Algorithm 1 TrendClusterDiscovery($\underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{W}}, \delta) \mapsto \underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{TC}}, \delta)$

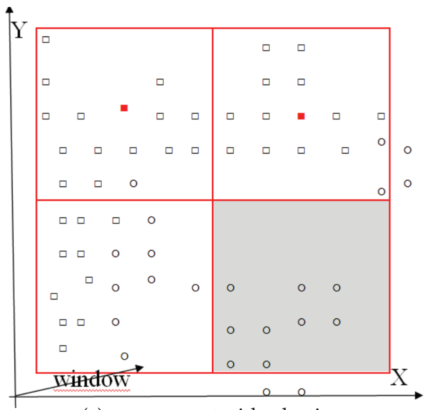
```
Require: $\underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{W}}$: the i-th data window in $z(T, K)$;
Require: $\delta$: the domain similarity threshold
Ensure: $\underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{TC}}$: the set of trend clusters $[i, c_k, \hat{z}_k]$ discovered in $\underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{W}}$
1: $k \leftarrow 1$
2: for all $p \in \mathbb{K}_i$ do
3:   if $p$ is UNCLUSTERED then
4:     $[c_k, \hat{z}_k] \leftarrow$ expandCluster($\{p\}$,trendPrototype($\{p\}$), $p$)
5:     append($\underset{t_{(i-1)w+1} \to t_{iw}}{\mathrm{TC}}$, $[t_{(i-1)w+1} \to t_{iw}, c_k, \hat{z}_k]$)
6:     $k \leftarrow k + 1$
7:   end if
8: end for
```



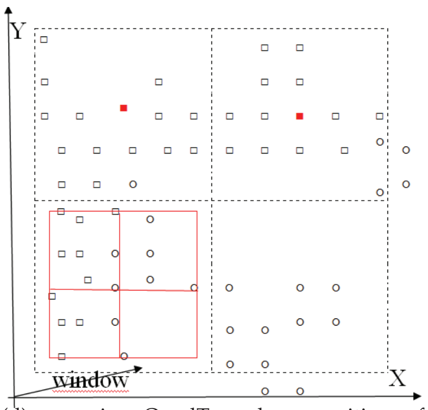
(a) MBR $density(c, Q) = 65.5\%$



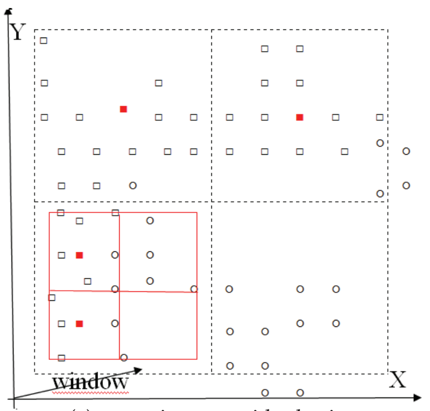
(b) QuadTree decomposition



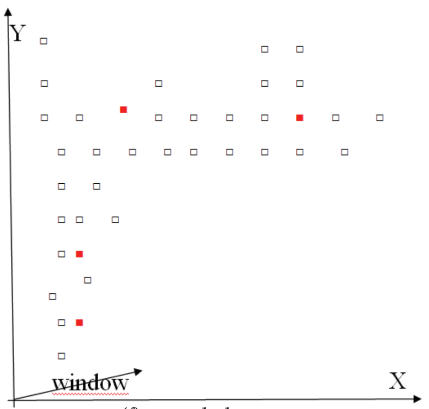
(c) sensor centroid selection



(d) recursive QuadTree decomposition of MBR rectangle



(e) recursive centroid selection



(f) sampled sensors

Figure 5: An example of quadtree-based sensor sampling computed on the cluster of squares with $\theta = 75\%$.

Algorithm 2 expandCluster $(c_k, \hat{z}_k, p) \mapsto [c_k, \hat{z}_k]$

```
Require: $c_k$: the current spatial cluster;
Require: $\hat{z}_k$: the trend prototype of $c_k$;
Require: $p$: the seed sensor for the expansion
Ensure: $[c_k, \hat{z}_k]$: the expanded trend cluster
1: $\eta(p) \leftarrow$ neighborhood$(u)$
2: $[tempC, \widehat{tempZ}] \leftarrow [c_k \cup \eta(p)$, trendPrototype$(c_k \cup \eta(p))]$
3: if trendClusterVariability$(tempC, \widehat{tempZ}, t_{(i-1)w+1} \rightarrow t_{iw}) \leq \delta$ then
4:     $[c_k, \hat{z}_k] \leftarrow [tempC, \widehat{tempZ}]$
5:     for all $q \in \eta(p)$ do
6:         $[c_k, \hat{Z}_k] \leftarrow$ expandCluster$(c_k, \hat{z}_k, q)$
7:     end for
8: else
9:     for all $q \in \eta(p)$ do
10:        $[tempC, \widehat{tempZ}] \leftarrow [c_k \cup q$, trendPrototype$(c_k \cup q)]$
11:        if trendClusterVariability$(tempC, \widehat{tempZ}, t_{(i-1)w+1} \rightarrow t_{iw}) \leq \delta$ then
12:            $[c_k, \hat{z}_k] \leftarrow$ expandCluster$(tempC, \widehat{tempZ}, q)$
13:        end if
14:    end for
15: end if
```

Algorithm 3 function sampling(c, θ) **return** $keysC$

```
Require: $c$ {cluster of sensors}
Require: $\theta$ {density threshold}
Ensure: $keysC$ {sample of key sensors extracted from $c$}
1: $keysC \Leftarrow \emptyset$
2: $Q \Leftarrow$ mbr$(c)$
3: if cardinality$(XY, Q) \neq 0\%$ then
4:     if density$(c, Q) > \theta\%$ then
5:         $keysC \Leftarrow s \cup \{$centroid$(c)\}$
6:     else
7:         $Set < c > \Leftarrow$subClusterQuadtree$(c)$
8:         for all $C_i \in Set < c >$ do
9:             $keysC \Leftarrow keysC \cup$ sampling$(C_i)$
10:        end for
11:    end if
12: end if
```

Then the density of the cluster c inside Q (Algorithm 3, lines 3–4, Figure 5(a)) is computed according to a density measure defined as follows:

$$density(c, Q) = \frac{\sharp(c \cap Q)}{\sharp Q} \times 100, \tag{10}$$

where $\sharp(c \cap Q)$ denotes the number of sensors clustered in c which are spatially contained in Q and $\sharp Q$ is the number of sensors of the network falling in Q. The spatial relation of containment between a 2D location (x, y) and a rectangle $[(x_i, y_i), (x_s, y_s)]$ is defined as follows:

$$(x, y) \subseteq Q \ \Leftrightarrow \ x_i \leq x \leq x_s \ \wedge \ y_i \leq y \leq y_s. \tag{11}$$

If $density(c, Q)$ is equal to zero, then Q is empty and it can be discarded for the sampling (see the gray colored quadrant in Figure 5(c)). If $density(c, Q)$ is greater than $\theta\%$ (by default $\theta = 75\%$), then $C \cap Q$ can be considered a dense sub-area of C and its centroid node is sampled (Algorithm 3, lines 4–5, see red-colored sensors in Figure 5(c) and Figure 5(e)). Otherwise Q is decomposed into four sub quadrants (see Figure 5(d)), that is, Q_1, Q_2, Q_3 and Q_4, and then c is coherently decomposed in the four subclusters falling in those quadrants, that is $C_1 = c \cap Q_1$, $C_2 = c \cap Q_2$, $C_3 = c \cap Q_3$ and $C_4 = c \cap Q_4$ (see Figure 5(b) and Figure 5(d)). The sampling is then recursively applied to each subcluster C_i (Algorithm 3, lines 8–10).

The quadrant decomposition of Q is defined orthogonally to the axes according to $x = \frac{max(x)+min(x)}{2}$ and $y = \frac{max(y)+min(y)}{2}$, such that:

$$\begin{array}{llll}
Q_1 : [& \left(min(x), \frac{max(y)+min(y)}{2}\right) & , \left(\frac{max(x)+min(x)}{2}, max(y)\right) &]. \\
Q_2 : [& \left(\frac{max(x)+min(x)}{2}, \frac{max(y)+min(y)}{2}\right) & , (max(x), max(y)) &]. \\
Q_3 : [& \left(\frac{max(x)+min(x)}{2}, min(y)\right) & , \left(max(x), \frac{max(y)+min(y)}{2}\right) &]. \\
Q_4 : [& (min(x), min(y)) & , \left(\frac{max(x)+min(x)}{2}, \frac{max(y)+min(y)}{2}\right) &].
\end{array} \tag{12}$$

The centroid of a set of sensors C_s is computed. First, the centroid location (Figure 5(c)) $(\widehat{x}_c, \widehat{y}_c)$ of c is determined as follows:

$$\widehat{x}_c = \frac{1}{\sharp c} \sum_{(x,y) \in c} x, \quad \widehat{y}_c = \frac{1}{\sharp c} \sum_{(x,y) \in c} y. \tag{13}$$

Then the sensor of c which is the nearest neighbor to $(\widehat{x}_c, \widehat{y}_c)$ is selected as the key sensor (centroid sensor) for the sampling. The centroid of c (see Figure 5(c) and Figure 5(e)) is the point location defined as follows:

$$centroid(c) = \underset{(x,y) \in c}{\text{argmin}} \left\{ EuclideanDistance((x, y), (\widehat{x}_c, \widehat{y}_c)) \right\}. \tag{14}$$

Once no further decomposition is possible the selected sample of sensors is output (Figure 5(f)). The consideration of the extracted sample of sensors, in place of each original cluster, drastically reduces the number of sensors processed during the interpolation phase. In this way, the algorithm speeds-up the off-line interpolation phase.

This recursive subdivision algorithm has a time complexity of O(n), where n is the size of the cluster. It allows the selection of a variable number of centroids from c. Each centroid is strategically located in a dense area of c, so that the necessary information to sketch the cluster shape is preserved.

Algorithm 4 function polynomial(T, Z, w) **return** $poly$

```
– Main routine($T, Z, w$) return $poly$
Require: $(T, Z, w)$ {the time series for the polynomial fitting}
Ensure: $poly$ {coefficients of the polynomial fitting $(T, Z, w)$}
1: $p \Leftarrow$ forwardPolynomial(costantPolynomial($V$),$T, Z, w, 1$)
– forwardPolynomial($previusP, T, Z, w, D$) return $p$
1: if $D \leq w - 1$ then
2:     $poly \Leftarrow previousP$
3: else
4:     $newP \Leftarrow$ straightLine(residual($T^D$), residual($Z$))
5:     if f-test($newP$) then
6:         $poly \Leftarrow previuousP$ {The forward addition of the variable $T^D$ to the polynomial is
           not statistically significant for the fitting of the time series $(T, Z, w)$ }
7:     else
8:         $poly \Leftarrow$forwardPolynomial($newP, T, Z, w, D + 1$)
9:     end if
10: end if
```

4.3 Trend based interpolator

Let (T, Z, w) be a time series of length w (i.e., the prototype of a trend cluster), such that, $(T, Z, w) = \langle t_1, z_1\rangle, \langle t_2, z_2\rangle, \dots, \langle t_w, z_w\rangle$. A temporal interpolator is determined to estimate the value of Z within the cluster at each unsampled time point within the interval w. TRECI computes a polynomial interpolator which fits in the time evolution of the data in a trend prototype time series. This interpolator is locally computed for each trend cluster.

The computation of a polynomial over locally related geographic data resembles the concept of a local relationship formulated in [13]. In [13] the existence of a local relationship is addressed by a competitive method based on the local entropy map. The local entropy is applied to determine a polynomial of two or more variables which change over the geographic space. Similarly to [13] the changes throughout the space of the data relationship by means of parameters (expressed in this case by polynomial coefficients) are considered. However, differently from [13], the zones where the changes occur do not need to be discovered. They are identified by the trend clusters, so that polynomial coefficients are piecewise learned within each cluster.

It is finally remarkable that any prior assumption on polynomial order is avoided by resorting to a stepwise regression procedure or by a considering simple linear interpolator. Details on both polynomial and linear interpolators are provided in the following.

4.3.1 Polynomial interpolator

The goal is to estimate the unknown coefficients of a polynomial $poly(T) : T \mapsto Z$, defined as follows:

$$poly(T) = \alpha + \beta_1 t + \beta_2 t^2, + \dots + \beta_D t^D, \tag{15}$$

such that $D < w$ and $poly(t)$ fits the time series (T, Z, w), according to the minimization of a cost function. These coefficients will be stored in a database in place of the fitted time series.

The degree D is automatically chosen ($1 \leq D < w$) by the forward selection strategy [7], tailored for the polynomial construction. This strategy is combined with a test to estimate the ability of a polynomial to fit the time series. Once the (unknown) estimate at a generic time position t^* is required, the polynomial interpolator computes:

$$z(t^*) = poly(t^*). \tag{16}$$

The polynomial is built stepwise according to Algorithm 4. We start with $D = 1$ and compute the (straight-line) polynomial of Z in T (see line 1 of the main routine in Algorithm 4). At each iteration, the ability of the current D-degree polynomial (named $newp$) to fit the time series (T, Z, w) is evaluated according to the partial F-test. The F-test [7], specifically applied in this case, allows the evaluation of the statistical significance in the improvement in the time series fitting, due to the addition of the term T^D to the currently constructed polynomial. If this improvement is not statistically significant, the polynomial $previousP$, that is, the polynomial previously constructed with degree $D-1$ (the constant polynomial if $D = 1$) is kept and no higher-degree variable is added to the final polynomial (stopping criterion, as reported in line 6 of Algorithm 4). On the contrary, D is incremented by one and the polynomial of degree D is forward computed by means of the straight-line regression between the residual of the dependent variable Z and the residual of the D-degree variable T^D (see line 8 of Algorithm 4 for the recursive call of the function $forwardPolynomial()$ and line 4 of Algorithm 4 for the computation of a straight line between residuals).

The residual of a variable is computed as the difference between the variable and the polynomial of degree $D-1$ estimating that variable. In particular, the residual of the dependent variable Z is the difference between the variable itself and the current polynomial in T of degree $D-1$ and fitting (T, Z, w). Similarly, the residual of the independent variable T^D is the difference between the variable itself and the polynomial in T of degree $D-1$ fitting T^D.

The procedure is iterated until $D = w - 1$ (see line 1 in Algorithm 4) or the F-test (see line 5 of Algorithm 4) are satisfied.

An example of the stepwise construction of a polynomial performed according to Algorithm 4 is reported in Example 4.1.

Example 4.1. (Forward selection based construction of a polynomial). *Let us consider the case in which we intend to build the polynomial p of degree $D = 2$,*

$$poly\colon\ Z = \alpha + \beta T + \gamma T^2, \tag{17}$$

through a sequence of parametric straight-line regressions. To this aim, we start by regressing V on the 1-degree variable T and building the straight line:

$$\hat{Z} = \alpha_1 + \beta_1 T. \tag{18}$$

The slope α_1 and intercept β_1 are computed on the time series (T, Z, w). This equation does not fit the series exactly. By adding the 2-degree variable T^2, the fitting might improve. However, instead of starting from scratch and building a new polynomial with both T and T^2, the forward strategy is exploited in the polynomial construction. First the parametric linear polynomial is built for T^2 if T is given, that is, $\hat{T^2} = \alpha_2 + \beta_2 T$. Then the residuals are defined on both the independent variable T^2 and the dependent variable Z, that is:

$$\begin{array}{rcl} T^{2'} & = & T^2 - (\alpha_2 + \beta_2 T). \\ Z' & = & Z - (\alpha_1 + \beta_1 T). \end{array} \tag{19}$$

Finally, the straight-line regression is determined between residuals Z' and $T^{2'}$ in the time series, that is,

$$\hat{Z}' = \alpha_3 + \beta_3 T^{2'}. \tag{20}$$

By substituting the straight-line regressions of Equation 19, the latter equation is reformulated as follows:

$$Z - (\alpha_1 + \beta_1 T) = \alpha_3 + \beta_3 (T^2 - (\alpha_2 + \beta_2 T)). \tag{21}$$

This equation can be written equivalently as:

$$Z = (\alpha_3 + \alpha_1 - \alpha_2\beta_3) + (\beta_1 - \beta_2\beta_3)T + \beta_3 T^2.$$

It is proved that the polynomial reported in the last equation coincides with the polynomial model built with Z, T and T^2 (in Equation 17), that is:

$$\alpha = \alpha_3 + \alpha_1 - \alpha_2\beta_3. \tag{22}$$
$$\beta = \beta_1 - \beta_2\beta_3. \tag{23}$$
$$\gamma = \beta_3. \tag{24}$$

A final consideration concerns the time complexity of this forward selection based computation of a polynomial of degree D, that is O($w \times \frac{D(D-1)}{2}$). This result can be interestingly combined with the consideration reported in [8, p. 90], according to which the degree of a polynomial adequately fitting w values should rarely exceed $\frac{w}{3}$.

4.3.2 Linear interpolator

The polynomial interpolation is conceptually simple since it allows a further reduction of the size of the time series stored in the database (at least if $D \leq w - 1$). However, its computation can be expensive for high polynomial degrees. The linear interpolation is a simpler lazy interpolator which, in several cases, performs as well as the polynomial interpolator without requiring a learning phase. The time series (T, Z, w) is stored as a sequence of timestamped values in the database (hence, there is no reduction of its size). Once the (unknown) estimate at a generic time position t^* is required, the linear interpolator determines:

$$z(t^*) = z(t_i) \cdot (1 - \tau(t_i, t_{i+1})) + z(t_{i+1}) \cdot \tau(t_i, t_{i+1}), \tag{25}$$

where t_i and t_{i+1} are the timestamps of two consecutive time series values (in (T, Z, w)), such that $t^* \in [t_i, t_{i+1}]$ and $\tau(t_i, t_{i+1}) = \frac{t^* - t_i}{t_{i+1} - t_i}$.

The linear interpolator is, in general, less precise than any other interpolator, apart from the nearest neighbor interpolator, which takes as output just the nearest value in the time series. Intuitively, it is less precise than any other polynomial interpolation (of degree greater than 1) since it approximates the ideal "continuous" function implied by the time series with a sequence of segments. These segments have discontinuities in the first derivative (improbable in any continuous physical quantity). The discontinuity problem does not occur for the polynomial. More precisely, the difference between the interpolators can be appreciated in the frequency domain by comparing their transfer functions [36]. In practice, interpolation can be regarded as the passage of the discrete signal (T, V) through a linear time-invariant and time-continuous system (named *filter*) and successive sampling at the desired time position t^*. The accuracy of an interpolator can be achieved by comparing its transfer function (i.e., the Fourier transform of its impulse response) with that of the ideal interpolator. The transfer function of the ideal interpolator is a low-pass ideal filter with a transfer function constant up to the Nyquist frequency and zero. Formally, $H_{id}(f) = \text{rect}(f/fs)$, rect() being the rectangle function, equal to 1 inside $[-1/2, 1/2]$ and zero otherwise and $fs = 1/\Delta T$ the sampling frequency. The transfer function of a linear interpolator corresponds to a $\text{sinc}^2(f \cdot \Delta T)$ that has a low pass effect in the fundamental domain. This function does not completely remove the replicas, since its secondary maximum is at -26dB. The transfer function of a polynomial interpolator varies with the degree of the polynomial. For example, the transfer function of a third-order polynomial resembles that of a spline interpolation, known to be very smooth in the fundamental domain and with its secondary maximum at -44dB, [36].

4.4 Spatiotemporal inverse distance weighting interpolation

Inverse distance weighting (IDW) [43] is adapted here in order to estimate off-line the unknown field measure at any space-time point (x^*, y^*, t^*). First, the past window W which hosts t^* is identified. Once W is identified, the key sensors of the summary (trend clusters) which are stored in the database for the window W and the regression model (polynomial coefficients or time series values) of the trend prototypes associated to each key sensor are retrieved.

Let c be a key sensor with $c \in keys(W)$, (x_c, y_c) be the space position of c, $z_c(t^*)$ be the value estimated for c at time point t^* by using either the polynomial interpolator (see Section 4.3.1) or the linear interpolator (see Section 4.3.2) of the trend prototype of c in W. Then the interpolated value $\hat{z}(x^*, y^*, t^*)$ is computed as follows:

$$\hat{z}(x^*, y^*, t^*) = \begin{cases} z_c(t^*) & \text{if } \exists c \in keys(W) \\ & \text{with } (x^*, y^*) \equiv (x_c, y_c), \\ \dfrac{\displaystyle\sum_{c \in keys(W)} w_{(x^*,y^*)(x_c,y_c)} \times z_c(t^*)}{\displaystyle\sum_{c \in keys(W)} w_{(x^*,y^*)(x_c,y_c)}} & \text{otherwise.} \end{cases} \tag{26}$$

The idea behind Equation 26 is that the interpolation at an unsampled point location is a function of the known values around it. In particular, it depends on them in a relation inversely proportional to the distance, i.e., the nearer a known value is, the stronger its influence. According to this idea, the weights $w_{(x^*,y^*)(x_c,y_c)}$ are defined by the inverse of a power of the Euclidean distance:

$$w_{(x^*,y^*)(x_c,y_c)} = \mathbf{d}((x^*, y^*)(x_c, y_c))^{-p}. \tag{27}$$

The IDW interpolation is dependent on the *power parameter* p, which is a positive and real number. Typically higher values of p provide more influence to the observation located closest to the unsampled position. For $p \to \infty$, IDW converges to the 1-nearest neighbor interpolation, while for $p \to 0$ it becomes an arithmetic mean. Therefore, the optimal value of this parameter is dependent on the features of the random field under study. Here p has been chosen by following the rationale in [29,35] for which a geophysical random field has well-known self-similarity properties. Among these the correlation function of a field resembles a power-descending law of the distance $\mathbf{d}$, that is, $R_x(\mathbf{d}) \simeq |\mathbf{d}|^{-\alpha}$. By considering p as the tuner of the relative influence of neighbors in interpolation $p = \alpha$ is chosen. On the other hand, the value of α is known to be related to the fractal (or Hausdorff) dimension ν of a field by a simple relation [35], that is, $\nu = n+1-\alpha/2$, where n is the field dimension. Since in the literature [29] the fractal dimension of several geophysical fields has been estimated between 2 and 3, by assuming $\nu = 2.5$, $\alpha = p = 3$ is achieved.

Although Equation 26 considers the entire set of key sensors sampled across the networked space, it is reasonable to suppose that an *influence boundary* can be set so that the key sensors which are outside this area should not be taken at all in the computation. Thus, a spheric area is fixed around the unsampled location (x^*, y^*); the key sensors contribute to the interpolation only if they are inside this spherical region. The center of the *interpolation sphere* is (x^*, y^*) and the radius is a boundary parameter b. Based on these considerations Equation 27 is reformulated as follows:

$$w_{(x^*,y^*),(x_c,y_c)} = \begin{cases} \mathbf{d}((x^*,y^*),(x_c,y_c))^{-p} & \text{if } \mathbf{d}((x^*,y^*),(x_c,y_c)) \leq b \\ 0 & \text{otherwise} \end{cases}. \quad (28)$$

In TRECI, an automatic mechanism is implemented to choose b at each window. This mechanism guarantees that, independently of (x^*, y^*), at least one centroid is within the boundaries. The idea of automatically detecting b_W at the window W as the maximum among the distances computed between each pair of closest centroids in the set $keys(W)$ was inspired to this requirement. Formally, let $c \in keys(W)$ be a key sensor of a cluster in W, $min_c[W]$ be the minimum Euclidean distance between c and any other key sampled sensor $c' \in centroids(W)$, that is:

$$min_c[W] = \min_{c' \in centroids(W) \wedge c' \neq c} \mathbf{d}(\hat{c}_{C_k}, \hat{c}_{C_k h}). \quad (29)$$

Then b_W is computed as the maximum of the $min_c[W]$ by varying c, that is:

$$b_W = \max_{c \in centroids(W)} min_c[W]. \quad (30)$$

5 Experimental results

The TRECI framework is written in Java and interfaces with a database managed by a MySQL DBMS. The on-line component (summarizer) and the off-line component (space-time interpolator) of TRECI were evaluated on an Intel(R) Core(TM) 2 DUO CPU $P61100$ @2.00GHz with 3.7 GB of RAM Memory, running Ubuntu Release 12.04 (precise) $32-$bit, Kernel Linux 3.2.0-26-generic-pae.

The goals of the experiments are twofold. The first goal is to demonstrate that TRECI-ONLINE is able to give an accurate and compact summarized representation of a sensor network data stream. This result is achieved by the combination of the trend clusters with the quadtree-based sampling of clusters and (polynomial) regression models of trend prototypes. The second goal is to evaluate how TRECI-OFFLINE is effective in integrating the trend cluster knowledge in the IDW interpolation and estimating the field value at every space-time point.

5.1 Experimental setting: Data and measures

Experiments were performed with two sets of publicly available climatology network data. Details on the real sensor network data streams and the evaluation measures considered for the empirical evaluation in this study are reported below.

5.1.1 Data

The *South American Air Temperature* sensor data stream [45] (SAT) collected monthly-mean air temperature measurements (in degrees Celsius) between 1960 and 1990 over a $0.5^\circ \times 0.5^\circ$ of latitude/longitude regular grid of South America, for a total of 6477 stations. The topology of this network does not change in time and no measures are missing in the stream. By using a boxplot the recorded temperature values range between -7.6 and 32.9^oC.

The *Global Historical Climatology Network* sensor data stream [34] (GHCN) collected monthly-mean air temperature (in degrees Celsius) for 7280 land stations worldwide. The period of record varies from station to station, with several thousand extending back to 1890 up to 1999. Stations are irregularly installed across the Earth; the network configuration changes in time since new stations have been installed during the time of measurements, while old stations have been discarded. So the stream has several missing values. The average number of stations measuring the field per snapshot is 3646.16. By using a box plot it was calculated that temperature values range between -20.75^oC and 49.25^oC.

5.1.2 Evaluation measures

The performance of TRECI was evaluated in terms of the amount of memory consumed to store the trend cluster summarization of the stream and of the interpolation error. The amount was measured in kB. The interpolation error was measured as the root mean squared error of the interpolated stream values (which were used as the ground truth). Additional statistics collected in this evaluation study included the number of trend clusters, the number of sensors grouped per cluster, the number of sampled key sensors, the degree of the polynomial regression model, as well as the average nearest neighbor distance. In particular, the average nearest neighbor distance was computed by averaging, for each window and for each key sensor in the window, the distance between the selected key sensor and its nearest neighbor key sensor. Hence, the average nearest neighbor distance computed on a window is an average indicator of the sensors' density in the window. Note that the measures are averaged per window, except for RMSE, which is averaged per snapshot.

5.2 Results

The results of the empirical evaluation of both the summarizer and the interpolator integrated in TRECI are now illustrated and discussed. Experiments were run by considering window size $w = 12$ (months) due to the expected yearlong periodicity of the temperature field. Trend cluster discovery was run with a domain similarity threshold that is about 10% of the field dynamics in the stream ($\delta = 4^o$C in SAT and $\delta = 7^o$C in GHCN). In the evaluation of the on-line phase (summarizer) the goal was to determine how much the quadtree-based sampling of clusters and the polynomial-based regression modeling of trend prototypes improve the summarization compactness of the trend cluster storage in the database. In the evaluation of the off-line phase (interpolator) the goal was to establish how accurate IDW is compared to a simple (but common in the literature) interpolation technique like the 1-nearest neighbor (1NN). The accuracy of the interpolator was also evaluated by varying either the percentage of sensors switched off in the network or the percentage of snapshots cut-off in the stream. In these cases the summarizer processed only a sub-set of the stream, while the interpolator was used to estimate both available and unavailable data.

5.2.1 TRECI-ONLINE (summarization) evaluation

Each stream was entirely (i.e., without dropping any data) processed on-line for this part of the evaluation study. The summarization power of trend cluster discovery was evaluated when combined with sampling and polynomial regression modeling.

Quadtree sampling The summarization power of combining trend cluster and sampling was evaluated. For each trend cluster, the key sensors were sampled and stored in the database as representative of the cluster, while, as in SUMATRA, the time series of w values were stored as representative of the trend. This summarizer, which combines SUMATRA with the quadtree sampling, is called Qs-S. Qs-S is compared to SUMATRA (when no sampling is applied), Ss-S (when for each cluster just the central sensor is stored in the database) and Rs-S (when for a cluster a random choice of a sample of sensors is made for the storage in the database). In order to make this comparison the random selection takes the same number of samples selected for Qs-S.

The average number of trend clusters discovered per window (nc), the average number of sensors sampled per window (nss), the average memory size of the trend cluster representation of each windowed stream as it is stored in the database (size) and the average RMSE of each streamed snapshot reconstructed from the trend cluster knowledge stored in the database (RMSE) are reported in Table 5.2.1. The stream reconstruction is a special case of the space-time interpolation, where each interpolation point (x, y, t) is the space-time location of each sensor actually transmitting at the time t.

The results obtained for both the networks confirm that, as expected, the sampling significantly reduces the size of the stream. Obviously, the observed reduction is more impressive whenever a single centroid sensor is sampled for each cluster, but this size reduction is at the expense of the accuracy (see Qs-S versus Ss-S). On the other hand, the quadtree decomposition for the sampling is highly beneficial. First, the sample size is automatically tuned. Second, the strategic selection of those sensors that keep the information on the cluster shape guarantees an error that is close to the error performed when all the sensors are stored in the database (see SUMATRA versus Qs-S). In conclusion, the results suggest that the use of quadtree sampling obtains the best trade-off between the size and the error of the summary. This theory is sustained by the results obtained with an ideal network (like SAT), whose topology is regular and does not change in time and where no expected data are missing. At the same time, it is sustained by results with an irregular sparse grid (like GHCN), where the topology changes in time and expected data are often missing.

Data	Measure/window	Stream	SUMATRA	Qs-S	Ss-S	Rs-S
SAT	nc	-	82.4	82.4	82.4	82.4
SAT	nss	-	6477	666	82.4	666
SAT	size [kB]	316.26	16.677	5.328	4.188	5.328
SAT	rmse [^{o}C]	-	1.23	1.74	4.30	3.94
GHCN	nc	-	1064.3	1064.3	1064.3	1064.3
GHCN	nss	-	3646.2	1895.6	1064.3	1895.6
GHCN	size [kB]	178.03	59.093	55.674	54.050	55.674
GHCN	rmse [^{o}C]	-	1.87	2.38	3.71	2.41

Table 1: Quadtree sampling evaluation: number of clusters (nc), number of (sampled) sensors (nss), size (kB) and error (RMSE). nc, nss, and size are averaged per window, RMSE is averaged per snapshot.

Polynomial regression model The summarization power of the quadtree sampling was also evaluated in combination with the use of the polynomial regression model to represent the trend prototypes.

Data	Measure/window	SUMATRA	Qs-S	TRECI
SAT	poly deg	-	-	5.37
SAT	size [kB]	16.677	5.328	3.686
SAT	rmse [^{o}C]	1.23	1.74	1.84
SAT	time (on-line) [secs]	19.43	21.35	21.64
SAT	time (off-line) [secs]	5.89	22.42	23.21
GHCN	poly deg/window	-	-	3.99
GHCN	size [kB]	59.093	55.674	28.086
GHCN	rmse [^{o}C]	1.87	2.38	3.02
GHCN	time (on-line) [secs]	19.23	25.33	26.68
GHCN	time (off-line) [secs]	6.22	23.16	23.52

Table 2: Polynomial evaluation: Degree of learned polynomials (deg), size (kB) and error (RMSE) of the (summarized) stream. Degree and size are averaged per window. RMSE is averaged per snapshot.

The results, collected in Table 5.2.1, show that the polynomials computed to model the trend prototypes have an average degree which is definitely lower than w (w=12) and close enough to the expected threshold of $w/3$ suggested in [8]. The storage of the (polynomial) regression model of a trend prototype in place of the time series computed by the trend cluster discovery process further reduces the size of the stream summary (see size of TRECI versus Qs-S versus SUMATRA in Table 5.2.1) stored in the database. The behavior observed is that the summarization capability is at the expense of the accuracy (see RMSE of TRECI versus Qs-S in Table 5.2.1), but the RMSE is well below the upper bound of the domain similarity threshold δ.

Computation time To complete the analysis of the on-line phase of the framework presented, the time spent per window (in seconds) to learn the interpolation model was analyzed. The computation time was collected for both SUMATRA, Qs-S (SUMATRA, quadtree sampling and linear interpolator) and TRECI (SUMATRA, quadtree sampling and polynomial interpolator) when they process the entire streams (no sensor has been switched off). As expected, in both networks, the computation time slightly increased from SUMATRA to Qs-S and then to TRECI. This phenomenon is due to the amount of additional processing required to sample sensors in a cluster as well as to determine the polynomial interpolator of a trend. In any case, the on-line computation time always remains below 27 seconds per window, where a window collects twelve snapshots for a very a large network (more than six thousands sensors in each network). It is reasonable to conclude that the proposed on-line processing, even if it is applied to monthly averaged climate data, scales well for a faster real time data stream, for which snapshots of large sets of observations can arrive within minutes or seconds.

Data	Measure/window	SUMATRA	Qs-S	TRECI
SAT	time (on-line) [secs]	19.43	21.35	21.64
GHCN	time (on-line) [secs]	19.23	25.33	26.68

Table 3: Computation time evaluation: Computation time spent per window (in secs) to learn an interpolation model in the on-line phase

5.2.2 TRECI OFFLINE (interpolation) evaluation

The error of the spatiotemporal IDW interpolation was evaluated in experimental settings obtained by switching off sensors and/or jumping snapshots in the summarization phase. The IDW was also compared to the simple 1-NN, and the conditions under which the polynomial interpolator outperforms the trend linear interpolator were investigated.

p	SAT	GHCN
1	2.28	4.17
2	1.92	4.01
3	1.84	3.99
4	1.58	**3.77**
5	1.54	3.89
6	1.52	3.98
7	1.5186	4.0
8	**1.5181**	4.09
9	1.5196	4.13
10	1.5217	4.16

Table 4: Interpolation error (RMSE) by varying p. The power parameter p ranges between 1 and 10. The error is averaged per snapshot.

IDW versus p The IDW interpolation is a function of the power parameter p and $p = 3$ has been considered as a reasonable choice in Section 4.4. The influence of this parameter on the interpolation error was analyzed on the entire networks, with p growing from 1 to 10. The results are collected in Table 5.2.2. For GHCN (a sparse and realistic network), the RMSE curve initially decreases, with p reaching a minimum between 3 and 4 (coherently with the above considerations). For SAT (a regular network), a minimum is reached at $p = 8$, but it can be noted that the decreasing of RMSE from $p = 3$ to $p = 7$ is negligible. So the choice of setting $p = 3$, according to the considerations in Section 4.4 , is not necessarily the best. In any case, this choice achieves acceptable accuracy both in sparse and in regular networks without any expensive trial and error approach.

Spatiotemporal IDW Several experimental settings were considered to analyze the accuracy of the spatiotemporal IDW interpolator. First, 50% of the sensors were switched off in each snapshot. Second, 50% of snapshots were jumped in the streaming line. Finally, 50% of the sensors were switched off and 50% of the snapshots are jumped at the same time. The error performed in interpolating the stream as a whole was compared to the baseline case where no sensor is switched off and no snapshot is jumped.

The results collected in Table 5.2.2 show that the interpolation error is below δ in SAT, as well as in GHCN. In particular, the error reaches δ in GHCN just when both new spatial and new temporal points are considered in the interpolation phase (i.e., both sensors are switched off in the network and snapshots are jumped in the streaming line of the summarization phase). This result confirms the effectiveness of the interpolation even when a sparse network (like GHCN) is processed. Although the network becomes more sparse by switching off sensors, the data are sufficient for an accurate interpolation. In conclusion, TRECI can be considered a robust tool for interpolating a random field of a sparse and incomplete network at any location across the space and at any time point in the past.

Stream	Baseline	Sensor switching-off	Time point jumping	Sensor switching-off and time point jumping
SAT	1.84	2.70	2.63	3.27
GHCN	3.02	6.32	4.67	7.24

Table 5: Interpolation error (RMSE) computed on the stream in its entirety when 50% of the sensors have been switched off in the network and/or 50% of the snapshots have been jumped in the streaming line for the summarization phase. Measures are in ^{o}C.

IDW versus 1-NN How the accuracy of the spatiotemporal IDW compares with the accuracy of the simpler 1-NN was investigated. This comparison was performed by varying the percentage of sensors which were switched off in the network from 0% (the network is considered in its entirety) to 20%, 40%, 60% and 80% and using the polynomial interpolator of trend prototypes. Collected measures included the average number of switched-on sensors per window (nONs), the average number of trend clusters per window (nc), the average number of sampled key sensors per window (nss), the average nearest neighbor distance between the sampled key sensors per window (nnd), the average number of sensors per cluster (nsc), and the average RMSE per snapshot.

The results are collected in Table 5.2.2. The analysis of these results confirms that IDW greatly improves 1-NN interpolation accuracy. As expected, by increasing the sparsity

Stream	%	nONs	nc	nss	nnd	nsc	rmse(idw)	rmse(1nn)
SAT	0%	6477	82.4	666	0.8^o	78.5	1.74	8.88
SAT	20%	5182	75.7	514.3	0.88^o	68.4	2.40	8.79
SAT	40%	3886	72.2	417.8	1.01^o	53.8	2.52	9.09
SAT	60%	2591	67.3	311.0	1.21^o	38.4	2.74	8.06
SAT	80%	1295	62.3	144.1	1.73^o	24.7	4.28	10.24
GHCN	0%	3646.16	1064.34	1895.57	1.84^o	3.438	3.02	9.28
GHCN	20%	2880.72	928.26	1562.42	2.01^o	3.10	4.80	10.94
GHCN	40%	2151.18	780.475	1213.93	2.25^o	2.76	5.84	12.41
GHCN	60%	1427.72	587.17	826.16	2.69^o	2.43	6.79	14.74
GHCN	80%	723.28	387.91	444.77	3.76^o	1.86	7.68	16.49

Table 6: IDW versus NN interpolation performance: Percentage of sensors (%) which are switched off per snapshot, average number of switched-on sensors per window (nONs), average number of trend clusters per window (nc), average number of sampled key sensors per window (nss), average nearest neighbor distance between the sampled key sensors (nnd) per window, average number of sensors per cluster (nsc) and average RMSE per snapshot computed on the whole stream.

of the network, the average number of discovered clusters (nc), decreases as well as the number of sampled key sensors (nss) and the number of sensors per cluster (nsc). It is noticeable that the average distance between each pair of nearest key sensors increases as an effect of the higher sparsity of sensors across the network. The RMSE is monotonic with the percentage of switched-off sensors. This is an expected result since, by reducing the quality of the interpolation knowledge base (trend clusters in this case), the accuracy of the interpolator decreases as well. In any case, the RMSE is always below the domain similarity threshold δ, except in the extreme case that 80% of the sensors is switched off. This is an acceptable result since, in that case, the network becomes too sparse and the amount of available information for the interpolation becomes too poor.

IDW: Polynomial interpolator versus trend linear interpolator The final considerations concerned the analysis of the spatiotemporal IDW when the polynomial interpolator is compared to the linear interpolator. The interpolation error, computed on the entire stream, was evaluated under a progressive downsampling of the snapshots in a window. So, an initial one-stepped downsampling step (all snapshots in the window are used for the summarization) was followed by a two-, three-, and four-stepped downsampling (in each window, 6, 4, and 3 snapshots were taken instead of the original twelve snapshots for the summarization phase).

The results collected in Table 5.2.2 show that the storage size of trend clusters is reduced when using the polynomial regression model. Consequently, we plan to use the polynomial interpolator of trend instead of the linear interpolator. This consideration is especially true for low values of the downsampling step. On the other hand, the use of a linear interpolator within IDW outperforms the use of the polynomial interpolator, apart from the experimental settings, where the downsampling step is 4 for the SAT network, 3 and 4 for the GHCN network.

Once again the polynomial regression model can be considered a good compromise between the effectiveness in summarizing trend data and the accuracy in spatiotemporal interpolation. In particular, its performance was significantly better in bad measurement conditions, for example, when fewer snapshots of the stream are processed in the on-line phase.

Stream	ns	dsf	size(linear)	rmse(linear)	size(polynomial)	rmse(polynomial)
SAT	0	1	5.32	1.74	3.68	1.84
SAT	6	2	1.48	2.16	1.29	2.63
SAT	8	3	0.78	2.74	0.75	3.5
SAT	9	4	0.48	3.5	0.46	2.6
GHCN	0	1	43.15	2.37	28.08	3.02
GHCN	6	2	11.32	4.74	8.31	5.18
GHCN	8	3	5.59	6.96	5.12	5.71
GHCN	9	4	3.38	10.39	3.17	6.24

Table 7: IDW performances, trend linear interpolator versus trend polynomial interpolator: The number of jumped snapshots per window (ns), the downsampling factor (dsf), the average trend cluster summary size per window (kB), the average RMSE per snapshot.

6 Conclusion

Trend clusters are stream patterns which compactly represent numeric spatiotemporal data by means of spatial clusters having prominent data trends in time. They were originally defined to summarize data measures of a geophysical numeric field which were collected throughout a remote sensor network. In this paper, the trend cluster discovery is integrated as an online step in the spatiotemporal interpolation process which permits the estimation of a field at any location of the networked space and at any time point in the past.

A shape-dependent sampling technique is used to smartly reduce the number of sensors which are needed to accurately describe the extent of each cluster. Moreover, a polynomial regression model is fitted on each trend time series, in order to establish the law according to which the intra-cluster measures presumably evolve in time. For each trend cluster the sensors sampled in the cluster area and the polynomial law connected to them are stored in a database as a knowledge base for any future (spatiotemporal) interpolation.

The inverse distance weighting principle is combined with the polynomial interpolator of trend prototype to obtain spatial and temporal estimated values of the field.

The interpolator has been extensively evaluated in two large real climate sensor networks. The results prove the efficacy of the proposed solution that, in spite of a notable reduction in the volume of sensed data, guarantees the accurate estimation of unknown data. The efficacy of the interpolation method is also proved when the density of the sensors in the network or the frequency of the field measures are reduced.

Acknowledgments

This work has been carried out in fulfillment of the research objectives of both the projects: "EMP3: Efficiency Monitoring of Photovoltaic Power Plants" funded by the "Fondazione Cassa di Risparmio di Puglia," and the PRIN 2009 Project "Learning Techniques in Relational Domains and Their Applications" funded by the Italian Ministry for Universities and Research (MIUR). Authors wish to thank Lynn Rudd for help in correcting the paper.

References

[1] ALBERS, S. Online algorithms: A survey. *Mathematical Programming 97*, 1–2 (2003), 3–26. doi:10.1007/s10107-003-0436-0.

[2] BORAK, J. S., AND JASINSKI, M. Effective interpolation of incomplete satellite-derived leaf-area index time series for the continental united states. *Agricultural and Forest Meteorology 149*, 2 (2009), 320–332. doi:10.1016/j.bbr.2011.03.031.

[3] CIAMPI, A., APPICE, A., GUCCIONE, P., AND MALERBA, D. Integrating trend clusters for spatio-temporal interpolation of missing sensor data. In *Proc. 11th International Symposium of Web and Wireless Geographical Information Systems, W2GIS* (2012), pp. 203–220. doi:10.1007/978-3-642-29247-7_15.

[4] CIAMPI, A., APPICE, A., AND MALERBA, D. Summarization for geographically distributed data streams. In *Proc. 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, KES* (Berlin, 2010), vol. 6278 of *Lecture Notes in Computer Science*, Springer, pp. 339–348. doi:10.1007/978-3-642-15393-8_39.

[5] CRESSIE, N. The origins of Kriging. *Mathematical Geology 22*, 3 (Apr. 1990), 239–252. doi:10.1007/BF00889887.

[6] CRESSIE, N. *Statistics for spatial data*. Wiley, New York, 1993. doi:10.1111/j.1365-3121.1992.tb00605.x.

[7] DRAPER, N., AND SMITH, H. *Applied regression analysis*. No. pt. 766. Wiley, 1981.

[8] FABBRIS, L. *Statistica multivariata. Analisi esplorativa dei dati*. Istruzione scientifica. McGraw-Hill Companies, 1997.

[9] GABER, M., ZASLAVSKY, A., AND KRISHNASWAMY, S. Mining data streams: A review. *ACM SIGMOD Record 34*, 2 (2005), 18–26. doi:10.1145/1083784.1083789.

[10] GALTON, A. *Qualitative spatial change*. Oxford University Press, 2000.

[11] GANESAN, D., RATNASAMY, S., WANG, H., AND ESTRIN, D. Coping with irregular spatio-temporal sampling in sensor networks. In *Proc. 2nd Workshop on Hot Topics in Networks (HotNets-II)* (2003). doi:10.1145/972374.972396.

[12] GUCCIONE, P., APPICE, A., CIAMPI, A., AND MALERBA, D. Trend cluster based Kriging interpolation in sensor data networks. In *Modeling and Mining Ubiquitous Social Media*, M. Atzmueller, A. Chin, D. Helic, and A. Hotho, Eds., vol. 7472 of *Lecture Notes in Computer Science*. Springer, Berlin, 2012, pp. 118–137. doi:10.1007/978-3-642-33684-3_7.

[13] GUO, D. Local entropy map: A nonparametric approach to detecting spatially varying multivariate relationships. *International Journal of Geographical Information Science 24*, 9 (2010), 1367–1389. doi:10.1080/13658811003619143.

[14] HARVEY, A. C. *Forecasting, structural time series models and the Kalman filters*. Cambridge University Press, Cambridge, UK, 2001.

[15] HORNSBY, K., AND EGENHOFER, M. J. Qualitative representation of change. In *Proc. International Conference on Spatial Information Theory, COSIT* (Berlin, 1997), S. C. Hirtle and A. U. Frank, Eds., vol. 1329 of *Lecture Notes in Computer Science*, Springer, pp. 15–33.

[16] ISAAKS, E. H., AND SRIVASTAVA, R. *An Introduction to Applied Geostatistics*. Oxford University Press, Oxford, UK, 1989.

[17] IZENMAN, A. *Modern Multivariate Statistical Techniques*. Springer, 2008. doi:10.1007/978-0-387-78189-1.

[18] KALMAN, R. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering 82*, Series D (1960), 35–45.

[19] KARYDAS, C., GITAS, I., KOUTSOGIANNAKI, E., LYDAKIS-SIMANTIRIS, N., AND SILLEOS, G. Evaluation of spatial interpolation techniques for mapping agricultural topsoil properties in crete. In *Proc. European Association of Remote Sensing Laboratories, EARSeL* (2009), vol. 8, pp. 26–39.

[20] KERWIN, W. S., AND PRINCE, J. L. The Kriging update model and recursive space-time function estimation. *IEEE Transactions on Signal Processing 47*, 11 (1999), 2942–2952. doi:10.1109/78.796430.

[21] KIM, B., AND TSIOTRAS, P. Image segmentation on cell-center sampled quadtree and octree grids. In *Proc. IS&T/SPIE Electronic Imaging* (2009), pp. 72480L–72480L–9.

[22] LAM, N. Spatial interpolation methods: A review. *The American Cartographer 10* (1983), 129–149. doi:10.1559/152304083783914958.

[23] LI, J., AND HEAP, A. D. *A Review of Spatial Interpolation Methods for Environmental Scientists*. Geoscience Australia Record 2008/23, 2008.

[24] LI, L. *Spatiotemporal Interpolation Methods in GIS: Exploring Data for Decision Making*. ETD collection for University of Nebraska-Lincoln, 2009.

[25] LI, L., AND REVESZ, P. A comparison of spatio-temporal interpolation methods. In *Proc. GIScience* (2002), M. Egenhofer and D. Mark, Eds., vol. 2478 of *Lecture Notes in Computer Science*, Springer, pp. 145–160.

[26] LI, L., ZHANG, X., HOLT, J., TIAN, J., AND PILTNER, R. Spatiotemporal interpolation methods for air pollution exposure. In *Proc. Ninth Symposium on Abstraction, Reformulation, and Approximation, SARA* (2011), M. R. Genesereth and P. Z. Revesz, Eds., AAAI.

[27] LIAO, D., PEUQUET, D. J., DUAN, Y., WHITSEL, E. A., DOU, J., SMITH, R., LIN, H. M., CHEN, J., AND HEISS, G. GIS approaches for the estimation of residential-level ambient PM concentrations. *Environmental Health Perspectives 114*, 9 (2006), 1374–1380. doi:10.1289/ehp.9169.

[28] LIN, G., AND CHEN, L. A spatial interpolation method based on radial basis function networks incorporating a semivariogram model. *Journal of Hydrology 288* (2004), 288–298. doi:10.1016/j.jhydrol.2003.10.008.

[29] LOVEJOY, S., AND SCHERTZER, D. Scale, scaling, and multifractals in geophysics: Twenty years on. In *Nonlinear Dynamics in Geosciences*. Springer New York, 2007, pp. 311–337.

[30] LU, G. Y., AND WONG, D. W. An adaptive inverse-distance weighting spatial interpolation technique. *Journal of Computers and Geosciences 34* (2008), 1044–1055. doi:10.1016/j.cageo.2007.07.010.

[31] MALERBA, D., APPICE, A., VARLARO, A., AND LANZA, A. Spatial clustering of structured objects. In *Proc.15th International Conference on Inductive Logic Programming, ILP* (Berlin, 2005), S. Kramer and B. Pfahringer, Eds., vol. 3625 of *Lecture Notes in Computer Science*, Springer, pp. 227–245.

[32] MIRTA, S. K., AND KAISER, J. F., Eds. *Handbook for Digital Signal Processing*. John Wiley & Sons, Inc., New York, NY, 1993.

[33] MITAS, L., AND MITASOVA, H. Spatial interpolation. In *Geographical Information Systems: Principles, Techniques, Management, and Applications*, P. Longley, M. Goodchild, D. Maguire, and D. Rhind, Eds., vol. 1. Wiley, 1999, pp. 481–492.

[34] NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION (NOAA). Global historical climatology network data. ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/v2.

[35] PRIGARIN, S., HAHN, K., AND WINKLER, G. Estimation of fractal dimension of random fields on the basis of variance analysis of increments. *Numerical Analysis and Applications 4* (2011), 71–80. doi:10.1134/S1995423911010071.

[36] PROAKIS, J., AND MANOLAKIS, D. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice-Hall. Prentice Hall, 1996.

[37] ROMANOWICZ, R., YOUNG, P., BROWN, P., AND DIGGLE, P. A recursive estimation approach to the spatio-temporal analysis and modelling of air quality data. *Environmental Modeling Software 21*, 6 (June 2006), 759–769. doi:10.1016/j.envsoft.2005.02.004.

[38] ŞEN, Z., AND ŞALHN, A. D. Spatial interpolation and estimation of solar irradiation by cumulative semivariograms. *Solar Energy 71*, 1 (2001), 11–21. doi:10.1016/S0038-092X(01)00009-3.

[39] SHEPARD, D. A two-dimensional interpolation function for irregularly-spaced data. In *Proc. 1968 23rd ACM national conference* (New York, NY, 1968), ACM, pp. 517–524.

[40] SHUMWAY, R., AND STOFFER, D. *Time Series Analysis and Its Applications: With R Examples*. Springer, Berlin, 2010.

[41] STEIN, M. L. *Interpolation of Spatial Data: Some Theory for Kriging*, 1 ed. Springer, Berlin, June 1999.

[42] TEEGAVARAPU, R. S. V., MESKELE, T., AND PATHAK, C. S. Geo-spatial grid-based transformations of precipitation estimates using spatial interpolation methods. *Computers & Geosciences 40* (Mar. 2012), 28–39. doi:10.1016/j.cageo.2011.07.004.

[43] TOMCZAK, M. Spatial interpolation and its uncertainty using automated anisotropic inverse distance weighting (IDW)—cross-validation/jackknife approach. *Journal of Geographic Information and Decision Analysis 2*, 2 (1998), 18–30.

[44] UMER, M., KULIK, L., AND TANIN, E. Spatial interpolation in wireless sensor networks: Localized algorithms for variogram modeling and Kriging. *Geoinformatica 14*, 1 (Jan. 2010), 101–134. doi:10.1007/s10707-009-0078-3.

[45] UNIVERSITY OF DELAWARE. South american air climate data. http://climate.geog.udel.edu/~climate/html_pages/sa_air_clim.html.

[46] WIDMER, G., AND KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine Learning 23* (1996), 69–101.

[47] WORBOYS, M. F. Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science 19*, 1 (2005), 1–28. doi:10.1080/13658810412331280167.

[48] YONG, J., XIAO-LING, Z., AND JUN, S. Unsupervised classification of polarimetric SAR image by quad-tree segment and SVM. In *Proc. 1st Asian and Pacific Conference on Synthetic Aperture Radar, APSAR* (2007), pp. 480–483. doi:10.1109/APSAR.2007.4418655.