Statistical Regular Pavings to Analyze Massive Data of Aircraft Trajectories

G. Teng[1], K. Kuhn[2], and R. Sainudiin[3]
Private Bag 4800, University of Canterbury, Christchurch 8140, New Zealand

A variety of tasks conducted by aviation system decision makers and researchers requires analyzing aircraft trajectory data. Datasets containing high frequency aircraft position information collected over large geographic areas and long periods of time are too large to store in the primary memory of personal computers. This paper introduces the use of statistical regular pavings as data structures capable of summarizing very large aircraft trajectory datasets. Recursively computable statistics can be stored for variable-sized regions of airspace. The regions themselves can be created automatically to reflect the varying density of aircraft observations, dedicating more computational resources and providing more detailed information in areas with more air traffic. In particular, statistical regular pavings are able to very quickly aggregate or separate data with different characteristics so that data describing individual aircraft or collected using different technologies (reflecting different levels of precision) can be stored separately and yet also very quickly combined using standard arithmetic operations.

[1] Doctoral Student, Department of Mathematics and Statistics.
[2] Lecturer, Department of Civil and Natural Resources Engineering.
[3] Coordinator, Laboratory for Mathematical Statistical Experiments, Christchurch Centre, and Senior Lecturer, Department of Mathematics and Statistics.

I. Introduction

Air traffic controllers monitor the positions of aircraft and offer pilots guidance to ensure safe and efficient operations. Aviation systems researchers are developing decision-support tools to assist controllers as they manage increasing numbers of aircraft. Much research is framed around analyzing and forecasting the locations of aircraft in space and time. Aircraft trajectory data are often investigated in the context of other data: for instance airspace configuration (air traffic control) data or weather data. Data regarding aircraft positions can be used to estimate, and help control, air traffic controller workloads, local environmental impacts (e.g., noise), airport or airspace throughput, the proximity of different aircraft trajectories, etc..

A wealth of aircraft position data is currently being produced. For instance, in the United States the Federal Aviation Administration (FAA) records precise data on the latitude, longitude, altitude, and assorted other data for aircraft in radar range at least every 12 s in en-route areas and every 4.2 s around airport terminals [1]. Data are expected to be produced at a higher frequency as technologies like automatic dependent surveillance-broadcast come into widespread use. At the same time, there are and will be increasing numbers of aircraft and other airborne objects to track.

One of the authors previously studied aircraft position data recorded at one FAA control center over the course of 40 days [2]. The data set takes up roughly 14 GB of space. In this same study, the author was interested in analyzing the impacts of weather on aviation, and thus collected weather data for the 40 days of interest that take up another 10 GB of space when stored in the efficient hierarchical data format (hdf5). The sizes of datasets containing information collected over extended periods of time, tracking large numbers of aircraft, are problematic. Decision makers and researchers interested in the monitoring of real-time operations in particular face a challenge: how to quickly analyze and automatically summarize more data than can be stored in the primary memory of computers.

There is a need for a method to compactly store aircraft position and related data in a format that enables speedy analyses. Some aggregation of available data will be required, but data loss due to aggregation should be minimized. There should also be techniques for performing computationally efficient mathematical operations on aggregations of different datasets. We propose using statistical regular pavings (SRPs), described subsequently, to represent aircraft trajectory data, and perform arithmetic operations with them.

Regular pavings (RPs) are a class of subsets of the Euclidean space that partition the set of interest with boxes in the space. Recursive bisections and selections on the set are done to form a RP. We can think of this class of objects as a space-partitioning data structure for organizing points in the Euclidean space. RPs can also be seen as a space of binary trees with computationally efficient recursive properties. A SRP is an extension of a RP for statistical set processing. The statistically extended RP allows us to organize and summarize the sample data. In particular, it is capable of maintaining recursively computable statistics such as the counts, means, covariances, etc., of the data it represents.

For any given flight-specific trajectory data, we can use a SRP to partition a section of airspace (which minimally bounds the trajectory data) such that the finest cell in the partition is about the size of the aircraft. Only cells at the finest resolution are allowed to contain exactly one data point. The remaining cells with sizes larger than or equal to the finest resolution are empty with no data points. Each cell that contains a data point represents the position of a specific aircraft in airspace over a particular time period. Our proposed SRP data structure for flight-specific trajectory data can thus be thought of as a binary space-partitioning tree whose leaves are $\{0,1\}$-valued in order to represent the trajectory of a particular aircraft in a given region of the airspace over some time period. When the $\{0,1\}$-valued SRP for individual flight trajectories is a tree-based partition of the airspace within the radar's range, such that the leaf cells of the tree are either 0 or 1 to indicate the absence or presence of a particular flight, respectively, over a given time period, we can exploit the recursive properties of trees to perform computationally efficient arithmetic operations over the space of flight-specific trajectories. For instance, we can perform the addition of individual flight trajectories that are in the airspace over the same period of time in order to obtain the aggregate cotrajectories in the airspace during this period of time. Such an addition operation amounts to data aggregation of individual SRP trajectories into an aggregate SRP cotrajectory for which the leaves are allowed to have possibly more than one flight in them, depending on the length of the time period. Other arithmetic operations such as subtraction of two SRPs and multiplication of an SRP by a real number can also be performed. Thus, we can condense massive data into memory-efficient SRPs and perform subsequent arithmetic operations over them for aiding downstream decisions such as cotrajectory classification with additional weather data, fine-scale pollution monitoring, etc. We do not engage in such down-stream decisions using SRPs in this paper and focus instead on the foundations of SRPs for trajectory and cotrajectory arithmetics.

We note that there are many ways in which the flight observations from radar can sequentially enter a data structure for analysis. Suppose n $\mathbb{R}^d$-valued observations $X_1, X_2, \ldots, X_n$ sequentially enter the structure. We consider the following sequential entry settings in this study.

- In the n-presensed setting, all available data can sequentially enter the structure as one burst of all n points. In other words, all of the data are available in external-memory (or radar buffer).

- In the $n_{1:m}$-presensed setting, all available data can sequentially enter the structure as m bursts of $n_1, n_2, \ldots, n_m$ points from external-memory at conveniently choosable CPU times $t_1 < t_2 < \ldots < t_m$.

- In the $n_{1:m:\ldots}$-sensing setting, the CPU times $t_1 < t_2 < \ldots < t_m$ are driven by the real time line with yet unsensed burst indices beyond current CPU time t_m. We want our methods to computationally cope with data bursts and aid in decision-making in this more realistic online or dynamic sensor setting. We can obtain an idea of the limitations of our methods in the $n_{1:m:\ldots}$-sensing setting by first studying the memory and CPU limitations in the n-presensed and $n_{1:m}$-presensed settings.

Our proposed data structure is capable of handling both the $n_{1:m}$-presensed and $n_{1:m:\ldots}$-sensing settings. Thus, our dynamic data structure grows with each new burst from the radar and shrinks with each landing event in order to efficiently represent all aircraft positions in airspace at the present time or some specified interval of time. This is elaborated further in Sec. V.

II. Related Work

A number of research efforts have been put into investigating large aircraft trajectory data sets. An interactive visualization tool called FromDaDy [3] was developed for exploratory data analysis of aircraft trajectories and efficient detection of specific features. In this sophisticated work, an aircraft trajectory is a single line, or more precisely, dots connected by a line due to discrete observations by radar detection. There are hence no duplications of trajectories for a flight unit, but rather the trajectories are spread across views [3]. Using lines as the unit object enables the user to "filter, remove and add trajectories in an iterated manner until they extract a set of relevant data" [3] by using brush, pick, and drop techniques for selection. Boolean operations such as union (or addition) or intersection of the lines can be performed efficiently as well. This aspect is comparable to our data structure where each SRP object represents an aircraft trajectory over a time interval such that efficient aggregation (union) or intersection operations may be performed over SRPs. Recall that our SRP objects can be thought as a $\{0, 1\}$-valued structure such that an aggregation of these objects in integer arithmetic would also return the total number of aircraft in any given cell over the specified period of time. Using SRPs we are not only able to aggregate the individual trajectories, but also perform 1) query operations over a cuboidal region of the airspace over some time period, and 2) arithmetic operations over the aggregated trajectories for aiding downstream decisions. However, FromDaDy [3] is powerful data visualization tool and more efficient than SRPs in terms of trajectory selection, whereby the selection shape is not restricted by rectangular query boxes. This is due to the brushing technique that allows for geometrical queries that are more complex in shape than the cuboids of SRPs. We think that [3] and our SRP objects nicely complement each other in terms of combining visualization techniques with arithmetic techniques for trajectories.

Another study [4] presents results on the geographic distribution of aircraft carbon dioxide emissions by using radar track data of the type analyzed here, as well as other data regarding the trajectories of aircraft flying through areas where radar data were unavailable. The authors were able to access data on trajectories over the course of 24 hour periods but note that "the large daily files are too large and cumbersome to load into computer memory" [4]. The authors aggregated the data using a grid-based approach that is considered later in this paper. Authors investigating the aviation system impacts of adverse weather often use a similar grid-based approach. To analyze air traffic control system performance, the FAA in the United States uses the Weather Impacted Traffic Index (WITI) [5]. WITI uses a regular grid overlaid on the United States and then compares weather and aircraft trajectory data in each grid cell. Here, [5] notes the "trade-offs regarding fineness of gridding". The finer the gridding, the heavier the needs for space and computational time, while also facing "excess information representation" [5]. Conversely, if the grid is too coarse, one then runs into the issue of a lack of information representation. We thus suggest using SRPs, which allows the finest resolution to be at the level of the size of the aircraft while not having unneeded information. We will further explore this in Sec. VI.

Other tree-based data structures are used in [6] and [7] to analyze moving objects, with the common aim of answering range queries efficiently, which typically involves getting the number of points (planes) that are contained in some query box (location). Variations and extensions of well-known R trees (R for rectangle) for spatial access and queries are used in [6]. Here, nearby points are grouped with their minimal bounding rectangle and each bounding box is know as a region. The bounding boxes or regions play a main role in deciding if there is a need to descend into the subtrees. The authors augment the R-trees with summarized information, for instance the total number of points, for each box of the tree. The authors then introduced a temporal aspect such that the query is now "the number of points in a given box for a given time interval." If we are given T timestamps, then for a given region, the corresponding summarized information associated with each time stamp is now stored in a B tree. An aggregate multi-version R-B tree is developed to perform queries for moving objects. The term "aggregate" here is used to describe the augmentation of the summarized information at each box of the tree. A new box is created when an object moves to another position that also changes the position and size of parent box. The summarized information are updated accordingly as well. The aggregate R-B tree, which holds summarized information at time stamp t is essentially a different object compared with our SRP tree that is a $\{0, 1\}$-valued structure for flight-specific trajectories. But, if we aggregate the flight-specific SRP objects, we obtain an SRP object that has summarized information of the trajectories, which is similar to the aggregate R-B tree. Now building the underlying R-tree, which is a balanced tree whereby all its leaf nodes have to be at the same height, can be a challenging task in terms of building an efficient tree such that a query search requires as few descents into the subtrees as possible. There is therefore a trade-off between information fineness and computational/memory needs for the data structure in [6]. Our SRP objects have the advantage of allowing us to summarize information at the finest resolution in airspace (i.e., each plane is enclosed by a box just big enough to fit it) such that we do not lose this crucial information during aggregation. Note that the aggregated SRP object at an instant of time t has to be a $\{0, 1\}$-valued structure to remain collision free, for if the value at a leaf box or cell is more than one at an instant of time, then it means that a collision has happened there. One of the main objectives in [7] is the development of algorithms that maintain the median of a set of moving points for both the on-line (i.e., our $n_{1:m:\ldots}$-sensing setting) and off-line setting (i.e., our n-presensed or $n_{1:m}$-presensed settings). The algorithms in [7] use kinetic kd trees on sets of moving objects for query tasks. Typically, a kd-tree is built by splitting the data at the coordinate-specific median into two subsets where splits are done by alternating at the coordinates. Two variants of kd-trees are proposed: a δ-pseudo kd-tree (which turns out to be an almost balanced tree) that allows for efficient insertion and deletion, and a δ-overlapping kd-tree (a perfectly balanced tree) where the bounding boxes of two children are allowed to overlap. Similar to the R-trees in [6], the kd-trees in [7] have nodes that already contain summarized information and are essentially different from our basic SRP objects that are specific to a flight instance.

The main motivation of the work of [6] and [7] is to seek efficient data structure for query tasks. We are also able to perform query tasks with our data structure by intersecting a query box with the aggregated SRP object and extract the needed information. However, this querying aspect of SRPs is not explored in this study since the main focus of our study is to develop memory-efficient tree-based data structures for individual and aggregate flight trajectories for purposes of arithmetical operations over them. Finally, we would like to emphasize that our SRP trees are uniquely suited to perform efficient arithmetic operations, especially on a set of aggregated SRP objects for, say, cotrajectory pattern analyses, especially in conjunction with local weather data. We think that the data structures in [3], [6] and [7] can be used in complementary ways with SRPs, for which the real strength lies in performing arithmetic operations over the space of trajectories and cotrajectories, to comprehensively deal with downstream decision problems using our SRP arithmetics bolstered by interactive visualization in [3] as well as fast querying and coarse aggregation in [6] and [7].

III. Statistical Regular Paving (SRP)

A. Regular Paving (RP)

Let $\boldsymbol{x} := [\underline{x}, \overline{x}]$ be a compact real interval with lower bound $\underline{x}$ and upper bound $\overline{x}$ where $\underline{x} \leq \overline{x}$. Denote the space of such intervals as $\mathbb{IR}$. We can then define a box of dimension d as an interval vector

$$\boldsymbol{x} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_d, \overline{x}_d] \ .$$

Let $\mathbb{IR}^d$ be the set of all such boxes. Consider a box $\boldsymbol{x}$ in $\mathbb{IR}^d$. Let the index ι be the first coordinate of maximum width, i.e.,

$$\iota = \min \left(\underset{i}{\operatorname{argmax}} (\overline{x}_i - \underline{x}_i) \right) \ .$$

A bisection or split of $\boldsymbol{x}$ at the mid-point along this first widest coordinate gives us the left and right child boxes of $\boldsymbol{x}$ as follows:

$$\boldsymbol{x}_{\mathsf{L}} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_\iota, (\underline{x}_\iota + \overline{x}_\iota)/2) \times [\underline{x}_{\iota+1}, \overline{x}_{\iota+1}] \times \ldots \times [\underline{x}_n, \overline{x}_n] \ ,$$

$$\boldsymbol{x}_{\mathsf{R}} := [\underline{x}_1, \overline{x}_1] \times \ldots \times [(\underline{x}_\iota + \overline{x}_\iota)/2, \overline{x}_\iota] \times [\underline{x}_{\iota+1}, \overline{x}_\iota + 1] \times \ldots \times [\underline{x}_n, \overline{x}_n] \ .$$

Such a bisection is said to be regular. A recursive sequence of selective regular bisections of boxes with possibly open boundaries along the first widest coordinate, starting from the root box $\boldsymbol{x}_\rho$ in $\mathbb{IR}^d$, is known as a RP [7] or n tree [8] of $\boldsymbol{x}_\rho$. An RP of $\boldsymbol{x}_\rho$ can also be seen as a binary tree formed by recursively bisecting the box $\boldsymbol{x}_\rho$. When the root box $\boldsymbol{x}_\rho$ is clear from the context, we refer to an RP of $\boldsymbol{x}_\rho$ as merely an RP. Each node of an RP is associated with a sub-box of the root box that can be attained by a sequence of selective regular bisections.

Each node in an RP is distinctly labeled by the sequence of child node selections from the root node. We label these nodes and the associated boxes with strings composed of L and R for left and right, respectively. For example, the root node associated with root box $\boldsymbol{x}_\rho$ is labeled ρ. First, we split ρ into two child nodes. These left child and right child nodes are labeled by $\rho\mathsf{L}$ and $\rho\mathsf{R}$, respectively. The left half of $\boldsymbol{x}_\rho$ that is now associated with the node $\rho\mathsf{L}$ is denoted by $\boldsymbol{x}_{\rho\mathsf{L}}$. Similarly, the right half of $\boldsymbol{x}_\rho$ that is associated with the node $\rho\mathsf{R}$ is denoted by $\boldsymbol{x}_{\rho\mathsf{R}}$. Since the nodes $\rho\mathsf{L}$ and

ρR share the same parent node, ρL and ρR are a pair of sibling nodes for which the parent node is ρ.

Let us further split the left node ρL by bisecting the associated box $\boldsymbol{x}_{\rho \mathsf{L}}$ to get its left and right child nodes ρLL and ρLR with the associated sub-boxes $\boldsymbol{x}_{\rho \mathsf{LL}}$ and $\boldsymbol{x}_{\rho \mathsf{LR}}$, respectively. Next, we split the right child node ρR similarly into its child nodes ρRL and ρRR, respectively. Let us select ρLR to do a final split and obtain its child nodes ρLRL and ρLRR. We have obtained a binary tree from four splits of the root node. A node with no child nodes is called a leaf node. Let s be the RP of $\boldsymbol{x}_\rho$ obtained by the above sequence of splits. Then the set of leaf boxes associated with its leaf nodes is

$$\ell(s) = \{\boldsymbol{x}_{\rho \mathsf{LL}}, \boldsymbol{x}_{\rho \mathsf{LRL}}, \boldsymbol{x}_{\rho \mathsf{LRR}}, \boldsymbol{x}_{\rho \mathsf{RL}}, \boldsymbol{x}_{\rho \mathsf{RR}}\}$$

and $\ell(s)$ is a partition of $\boldsymbol{x}_\rho$. A graphical representation of the obtained RP s is shown in Fig. 1.

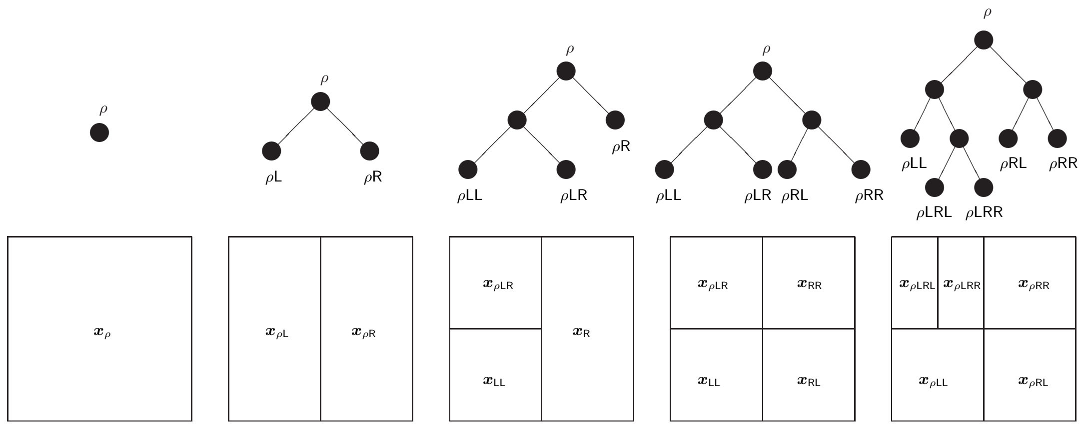


Fig. 1 A sequence of selective bisections of boxes (nodes) along the first widest coordinate, starting from the root box (root node), produces an RP.

B. Statistical Regular Pavings (SRPs)

The volume of a d-dimensional box $\boldsymbol{x}_{\rho v} = ([\underline{x}_{\rho v,1}, \overline{x}_{\rho v,1}], \ldots, [\underline{x}_{\rho v,d}, \overline{x}_{\rho v,d}])$ associated with the node ρv of an RP of $\boldsymbol{x}_\rho$ is the product of the side lengths of the box, i.e.

$$\mathrm{vol}(\boldsymbol{x}_{\rho v}) = \prod_{j=1}^{d} (\overline{x}_{\rho v,j} - \underline{x}_{\rho v,j}) \ .$$

The volume may also be associated with the depth of a node. A node has depth i if the length of the path of the node from the root node is i. Then, the volume of any d-dimensional box $\boldsymbol{x}_{\rho v}$

associated with node ρv having depth i is $\mathrm{vol}(\boldsymbol{x}_{\rho v}) = 2^{-i}\,\mathrm{vol}(\boldsymbol{x}_\rho)$ due to the recursive nature of the bisections and the restriction to only bisect at the first widest coordinate.

We use the nodes of the RP in Fig. 1 for illustration purposes. Assume that the root box $\boldsymbol{x}_\rho$ is a unit hypercube. Then the root node ρ has depth 0 and $\mathrm{vol}(\boldsymbol{x}_\rho) = 1$; the nodes $\rho\mathsf{L}$ and $\rho\mathsf{R}$ have depth 1 and volume 2^{-1}; the nodes $\rho\mathsf{LL}, \rho\mathsf{LR}, \rho\mathsf{RL}$, and $\rho\mathsf{RR}$ have depth 2 and volume 2^{-2}; and finally, the nodes $\rho\mathsf{LRL}$ and $\rho\mathsf{LRR}$ have depth 3 and volume 2^{-3}.

Suppose n points $x_1, x_2, \ldots, x_n$ have fallen into the root box $\boldsymbol{x}_\rho$ of an RP s. We can further associate each node ρv with the sample count

$$\#\boldsymbol{x}_{\rho v} := \sum_{i=1}^{n} I_{\boldsymbol{x}_{\rho v}}(x_i)$$

or the number of points that are inside its associated box $\boldsymbol{x}_{\rho v}$. Whenever a bisection happens, the number of points associated with the bisected node is also recursively updated. Each leaf node has pointers to the data that lie within its associated box. Whenever a bisection happens, the data fall into either the left or right child, depending on their location. We can use such information for statistical set processing and call this information structure a SRP as it enhances an RP by mutably caching recursively computable statistics of the data. Once the SRP has been constructed, the pointers to data from the leaf nodes are removed and only the counts remain. By an abuse of notation, we denote an RP as well as an SRP by s.

C. SRPs and Flight Trajectories

We will now apply SRPs to the analysis of aircraft trajectories. Let $X_1, \ldots, X_n$ be the time-ordered aircraft position data provided by FAA radar facilities. Data are typically provided as ordered high-dimensional tuples containing position data in $\mathbb{R}^2$ or $\mathbb{R}^3$, e.g., (latitude, longitude) or (latitude, longitude, altitude), and related data regarding aircraft heading, speed, type, etc. We will focus on position data to simplify discussion and because position data are the most relevant for many applications. Here, we are interested in constructing an SRP by recursive bisections to represent position data. A bisection only happens when a node has more than one point and this bisected node will not produce child nodes that have volume less than a predefined minimum volume $\lambda^* = 2^{-i^*}\,\mathrm{vol}(\boldsymbol{x}_{\rho v})$, where $i^* = \lfloor \log_2 \lambda^{-1}\,\mathrm{vol}(\boldsymbol{x}_{\rho v}) \rfloor$, and λ is taken to be an approximate volume of the aircraft. This splitting criteria guarantees that a leaf box with volume more than λ^* will not have any points in it and leaf boxes with volume λ^* may have more than one observation in them. The resulting SRP that encloses time-ordered aircraft position data of a particular aircraft is then an SRP trajectory. Here we will use the terms SRP and SRP trajectory interchangeably. The procedure to get an SRP trajectory is shown in Algorithm 1.

Algorithm 1: $s = \texttt{makeSRP}(\{X_1, \ldots, X_n\}, \lambda^*, \boldsymbol{x}_\rho)$

Input:

(i) data $\{X_1, \ldots, X_n\} \subseteq \mathbb{R}^d$

(ii) minimum volume λ^*

(iii) root box $\boldsymbol{x}_\rho$

Output: an SRP s

Make a new node s with box $\boldsymbol{x}_\rho$ and $\#\boldsymbol{x}_\rho \leftarrow 0$

for $j = 1$ to n `insertData`(ρ, X_j, λ^*) (see Algorithm 2)

return s

Figure 2a shows the SRP trajectory with root box $[810, 1230] \times [550, 1350]$ for aircraft position data $X_1, \ldots, X_n$, while Fig. 2b shows the shaded boxes in which points fall into. We zoom into Fig. 2a to obtain Fig. 2c and its corresponding tree in Fig. 2d. The aircraft at some position X_j is shown as a black point in Fig. 2c and is enclosed by a box with volume λ^*. We note that the observations are discrete in time, hence resulting in boxes with points that are disconnected.

IV. Arithmetic on SRPs

A. Performing Arithmetic by Overlaying SRPs

The choice of splitting along the first widest coordinate at the midpoint at each recursive level allows for efficient arithmetic operations that is possible over SRPs. Overlaying operations thus can be performed over two or more SRPs to get a new SRP, as long as each SRP has the same root box. This allows one to perform arithmetic over the space of SRPs in terms of adding and subtracting SRPs, producing an aggregate SRP. We describe this procedure in Algorithm 3 and give an illustration of an addition of two SRPs in Fig. 3.

Algorithm 2: `insertData`(ρv, X_j, λ^*)

Input:

(i) node ρv

(ii) data $X_j \in \boldsymbol{x}_{\rho v}$

(iii) minimum volume λ^*

if (box $\boldsymbol{x}_{\rho v}$ contains X_j)

  increment $\#\boldsymbol{x}_{\rho v}$ by 1

    if (ρv is a leaf node) $\wedge$ ($\frac{1}{2} \cdot \mathrm{vol}(\boldsymbol{x}_{\rho v}) \geq \lambda^*$)

        Make left node $\rho v\mathsf{L}$ with box $\boldsymbol{x}_{\rho v\mathsf{L}}$

        Make right node $\rho v\mathsf{R}$ with box $\boldsymbol{x}_{\rho v\mathsf{R}}$

        $\#\boldsymbol{x}_{\rho v\mathsf{L}} \leftarrow 0, \#\boldsymbol{x}_{\rho v\mathsf{R}} \leftarrow 0$

        Graft onto ρv as left child the node $\rho v\mathsf{L}$ and `insertData`($\rho v\mathsf{L}$, X_j, λ^*)

        Graft onto ρv as right child the node $\rho v\mathsf{R}$ and `insertData`($\rho v\mathsf{R}$, X_j, λ^*)

    if (ρv is not a leaf node)

      `insertData`($\rho v\mathsf{L}$, X_j, λ^*)

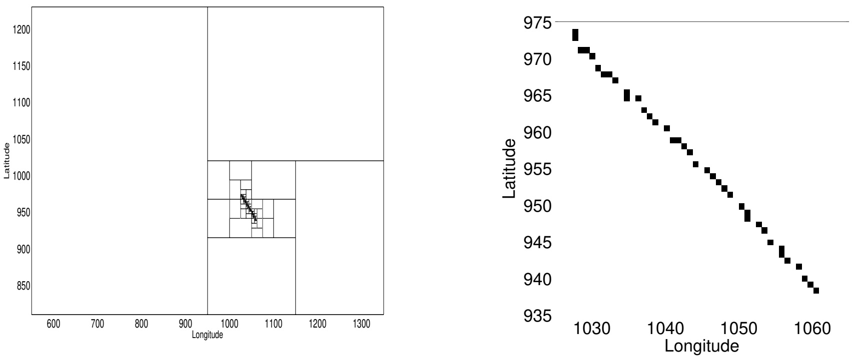
      `insertData`($\rho v\mathsf{R}$, X_j, λ^*)

B. Aggregations and Operations with SRP Trajectories

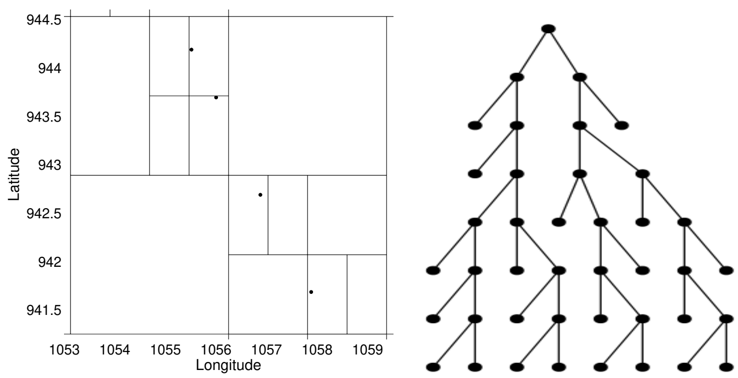
We have shown in Sec. IV A.A that it is fairly easy to combine SRPs that contain different-sized boxes using arithmetic on SRPs. This would, for instance, allow us to come up with aggregate frequency histograms when different data sets or even points have different levels of precision. This is intriguing from the point of view of aviation systems research, since aircraft size, aircraft equipage, and the fidelity of different relevant data streams are variable.

Here, we add and/or subtract SRP trajectories to produce an aggregate SRP trajectory. For a given SRP trajectory $s^{(1)}$ with root node $\rho^{(1)}$, define the scalar product $\alpha \cdot \rho^{(1)}$ to be the root node obtained from $\rho^{(1)}$ by transforming the count $\#\boldsymbol{x}_{\rho^{(1)}v}$ of every node $\rho^{(1)}v$ in $s^{(1)}$ to $\alpha \cdot \#\boldsymbol{x}_{\rho^{(1)}v}$. Now, for any real-valued α, β, we can obtain a linear combination $\alpha \cdot \rho^{(1)} + \beta \cdot \rho^{(2)}$ of two SRPs $s^{(1)}$ and $s^{(2)}$ with root nodes $\rho^{(1)}$ and $\rho^{(2)}$ by applying `AddSRP`$(\alpha \cdot \rho^{(1)}, \beta \cdot \rho^{(2)})$. When $\alpha = \beta = 1$, `AddSRP`$(\alpha \cdot \rho^{(1)}, \beta \cdot \rho^{(2)})$ is equivalent to an addition between the SRP trajectories $s^{(1)}$ and $s^{(2)}$.

Figure 4 shows high fidelity aircraft position data points enclosed by SRPs for three flights to



(a) SRP trajectory for aircraft position data. (b) Shaded boxes in the SRP trajectory.



(c) Aircraft positions enclosed by boxes. (d) The tree corresponding to (c).

Fig. 2 An SRP trajectory for aircraft position data and its corresponding tree.

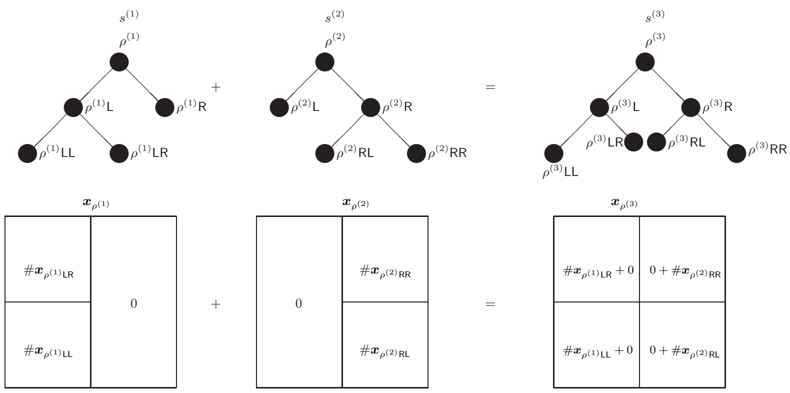


Fig. 3 An addition operation between SRPs $s^{(1)}$ and $s^{(2)}$.

which we assign fictional flight numbers ABC123, DEF456, and GHI789. The top panel in Fig. 4 shows the shaded leaf boxes, while the bottom panel shows the corresponding box boundaries at all nodes of the SRP trajectories. We give an example of adding three SRP trajectories in Fig. 5. The aggregate SRP trajectory for these three flights is shown in Fig. 5a as shaded leaf boxes and as box

Algorithm 3: $\rho^{(3)} = \texttt{AddSRP}(\rho^{(1)}, \rho^{(2)})$

Input: two nodes $\rho^{(1)}$ and $\rho^{(2)}$ with the same root box $\boldsymbol{x}_{\rho^{(1)}} = \boldsymbol{x}_{\rho^{(2)}}$

Output: a new node $\rho^{(3)} = \rho^{(1)} + \rho^{(2)}$

Make a new node $\rho^{(3)}$ with box $\boldsymbol{x}_{\rho^{(1)}}$ (or $\boldsymbol{x}_{\rho^{(2)}}$) and $\#\boldsymbol{x}_{\rho^{(3)}} \leftarrow \#\boldsymbol{x}_{\rho^{(1)}} + \#\boldsymbol{x}_{\rho^{(2)}}$

if ($\rho^{(1)}$ is a leaf node) $\wedge$ ($\rho^{(2)}$ is not a leaf node)

  Make temporary nodes L', R'

  $\boldsymbol{x}_{\mathsf{L}'} \leftarrow \boldsymbol{x}_{\rho^{(1)}\mathsf{L}}$, $\boldsymbol{x}_{\mathsf{R}'} \leftarrow \boldsymbol{x}_{\rho^{(1)}\mathsf{R}}$; $\#\boldsymbol{x}_{\mathsf{L}'} \leftarrow \#\boldsymbol{x}_{\rho^{(1)}\mathsf{L}}$, $\#\boldsymbol{x}_{\mathsf{R}'} \leftarrow \#\boldsymbol{x}_{\rho^{(1)}\mathsf{R}}$

  Graft onto $\rho^{(3)}$ as left child the node $\texttt{AddSRP}(\rho^{(2)}\mathsf{L}, \mathsf{L}')$

  Graft onto $\rho^{(3)}$ as right child the node $\texttt{AddSRP}(\rho^{(2)}\mathsf{R}, \mathsf{R}')$

if ($\rho^{(2)}$ is a leaf node) $\wedge$ ($\rho^{(1)}$ is not a leaf node)

  Make temporary nodes L', R'

  $\boldsymbol{x}_{\mathsf{L}'} \leftarrow \boldsymbol{x}_{\rho^{(2)}\mathsf{L}}$, $\boldsymbol{x}_{\mathsf{R}'} \leftarrow \boldsymbol{x}_{\rho^{(2)}\mathsf{R}}$; $\#\boldsymbol{x}_{\mathsf{L}'} \leftarrow \#\boldsymbol{x}_{\rho^{(2)}\mathsf{L}}$, $\#\boldsymbol{x}_{\mathsf{R}'} \leftarrow \#\boldsymbol{x}_{\rho^{(2)}\mathsf{R}}$

  Graft onto $\rho^{(3)}$ as left child the node $\texttt{AddSRP}(\rho^{(1)}\mathsf{L}, \mathsf{L}')$

  Graft onto $\rho^{(3)}$ as right child the node $\texttt{AddSRP}(\rho^{(1)}\mathsf{R}, \mathsf{R}')$

if (both $\rho^{(1)}$ and $\rho^{(2)}$ are not leaf nodes)

  Graft onto $\rho^{(3)}$ as left child the node $\texttt{AddSRP}(\rho^{(1)}\mathsf{L}, \rho^{(2)}\mathsf{L})$

  Graft onto $\rho^{(3)}$ as right child the node $\texttt{AddSRP}(\rho^{(1)}\mathsf{R}, \rho^{(2)}\mathsf{R})$

return $\rho^{(3)}$

boundaries at all nodes in Fig. 5b.

It is easy to access information such as frequencies (box heights) and airspace locations (the positions of the boxes) since the SRP trajectories store such information. A simple visual inspection of an aggregate trajectory SRP can be useful: areas where boxes are smaller (or have darker shades) are where frequencies are larger and indicate sections of airspace that are more frequently occupied.

We now process huge data sets containing the positions of different aircraft at different levels of precision and quickly aggregate them selectively to obtain desired aggregate trajectories. For instance, such selective aggregations could be useful when analyzing air traffic patterns for different weather conditions. Figures 6a and 6b show aggregate SRP trajectories on arrivals at a busy North American airport for periods of 6 h during a day with good weather and another with bad weather, respectively. Latitude and longitude data have been converted to Cartesian coordinates with the

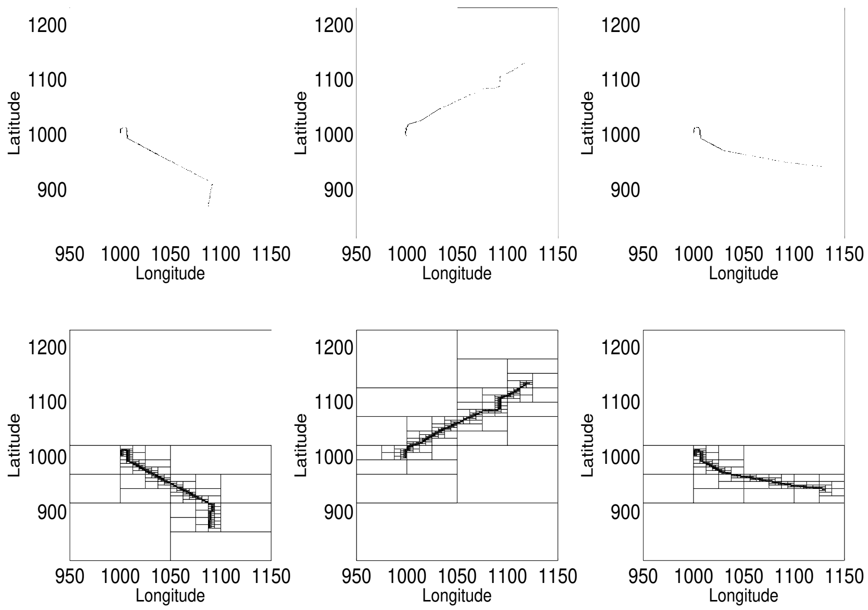
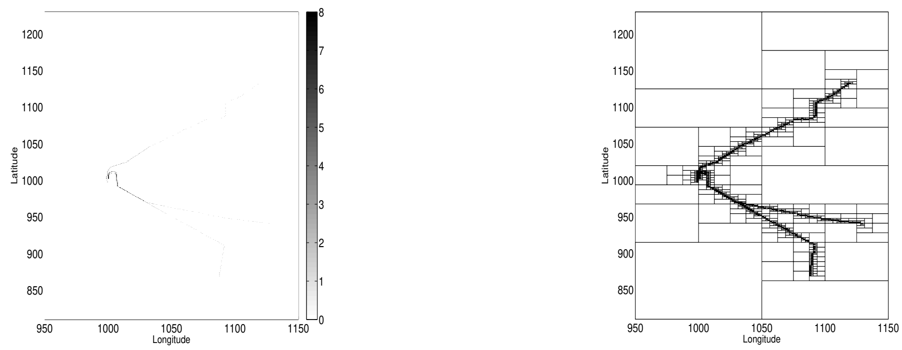


Fig. 4 SRP trajectories of flights ABC123, DEF456, GHI789.



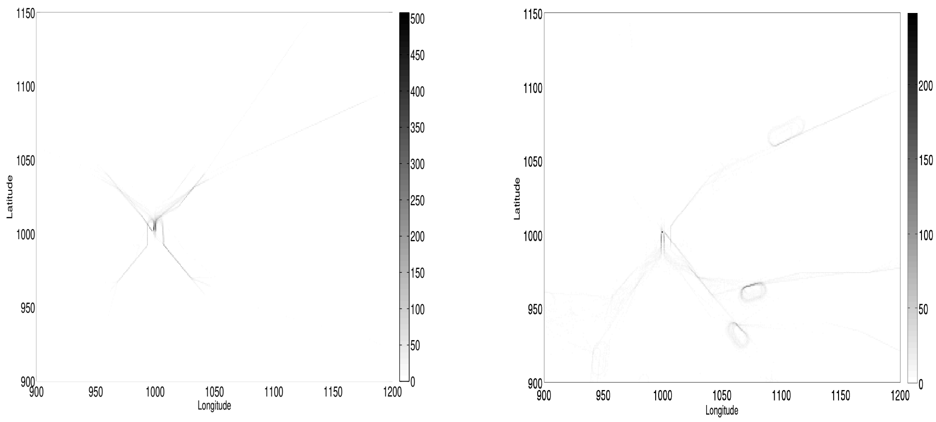
(a) Shaded boxes.

(b) Box boundaries at all nodes.

Fig. 5 Aggregate SRP trajectory for the flight trajectories of Fig. 4.

airport located approximately at point (1000, 1000). Given that information can be accessed conveniently, it would be easy to, for example, manually/qualitatively or automatically/quantitatively compare Figs. 6a and 6b, and thus compare operations when weather conditions were benign vs when they were adverse.

We can further perform arithmetic on these aggregate SRP trajectories. By an abuse of notation, let $s^{(1)}$ denote the aggregate SRP trajectory of the good weather day and $s^{(2)}$ be the aggregate SRP trajectory of the bad weather day, each with root nodes $\rho^{(1)}$ and $\rho^{(2)}$, respectively. Now, let $\alpha = 1$



(a) A good weather day.

(b) A bad weather day.

Fig. 6 Aggregate SRP trajectories on a good weather day and a bad weather day respectively.

and $\beta = -1$. Then, $\texttt{AddSRP}(\alpha \cdot \rho^{(1)}, \beta \cdot \rho^{(2)})$ is equivalent to a subtraction between the two aggregate SRP trajectories, and the resulting aggregate SRP trajectory is able to provide information on which airspace will be more/less frequented on the good weather day compared with the bad weather day. This is illustrated in Fig. 7, which shows the differences between the two weather aggregate SRP trajectories in Fig. 6. Lighter areas indicate sections of airspace more frequently occupied on the bad weather day, whereas darker areas indicate locations of airspace frequented more often on the good weather day.

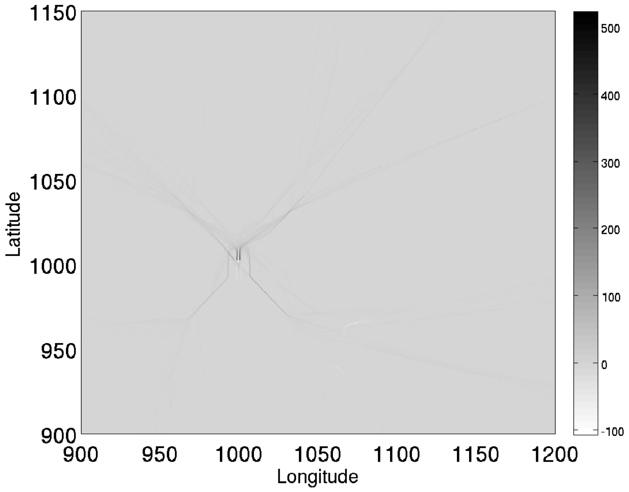
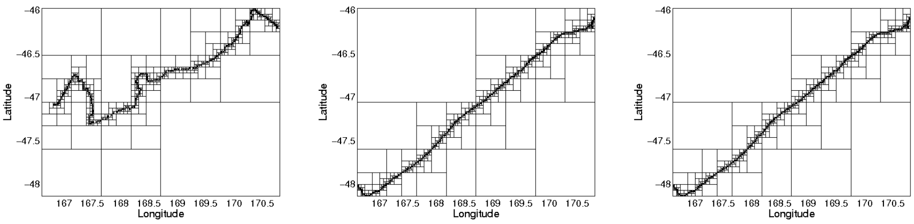


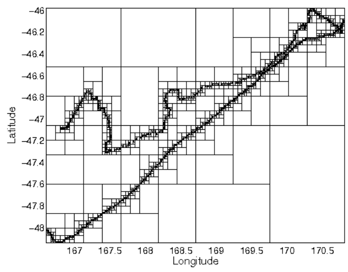
Fig. 7 Subtraction between aggregate SRP trajectories on a good and bad weather day.

We have shown that SRPs can be used to represent and operate (linearly combine) sets of aircraft trajectories. Thus, further analysis of how SRPs like those shown in Figs. 6a and 6b evolve over time and as weather patterns change is warranted. We note that bad weather days are generally different and it will likely prove quite difficult to classify and predict what will happen on a bad weather day. However, Figs. 6 and 7 could be the first step toward classification and prediction of airspace usage based on concurrent weather data.

There are potentially other applications of SRPs in other fields that require the analysis of object movements, e.g. animal migration tracking. Figure 8 shows animal track data for a single animal over three days represented by SRPs (data courtesy of Mr. Paul Sagar). Figure 8(a) shows the movement of the animal of each individual day represented by individual SRP trajectories, whilst Fig. 8(b) is the aggregate SRP trajectory obtained by adding the three individual SRP trajectories. The core functions needed to implement the algorithms here can be found in a GNU general public licensed C++ class library [10].



(a) Individual SRP trajectories.



(b) Aggregate SRP trajectory.

Fig. 8 Using SRPs to track animal migration.

V. A Dynamic Airspace Data Structure with SRPs

A. Reuniting Nodes

If an (aggregate) SRP trajectory has pairs of sibling nodes that share the same properties, for instance, no points in each node, we can reunite these nodes to obtain its parent node and thereby reduce the size of the SRP to conserve memory. If two child nodes require two units of memory, then a reunification will halve the need for memory. Furthermore, the reunited node provides the same statistical information as its child nodes, so there is no loss of information. Figure 9 shows that boxes $\boldsymbol{x}_{\rho\mathsf{LL}}$ and $\boldsymbol{x}_{\rho\mathsf{LR}}$ can be reunited to get $\boldsymbol{x}_{\rho\mathsf{L}}$ since neither $\boldsymbol{x}_{\rho\mathsf{LL}}$ nor $\boldsymbol{x}_{\rho\mathsf{LR}}$ have any points in it. Algorithm 4 describes this procedure of reuniting empty nodes.

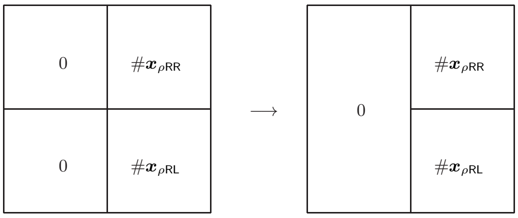


Fig. 9 Reuniting boxes $\boldsymbol{x}_{\rho\mathsf{LL}}$ and $\boldsymbol{x}_{\rho\mathsf{LR}}$ to get box $\boldsymbol{x}_{\rho\mathsf{L}}$.

Algorithm 4: `ReuniteEmptyNodes`(ρv)

Input: a node ρv

if ρv has left child `ReuniteEmptyNodes`$(\rho v\mathsf{L})$

if ρv has right child `ReuniteEmptyNodes`$(\rho v\mathsf{R})$

if ($\rho v\mathsf{L}$ and $\rho v\mathsf{R}$ are leaf nodes) $\wedge$ $(\#\boldsymbol{x}_{\rho v\mathsf{L}} = \#\boldsymbol{x}_{\rho v\mathsf{R}} = 0)$

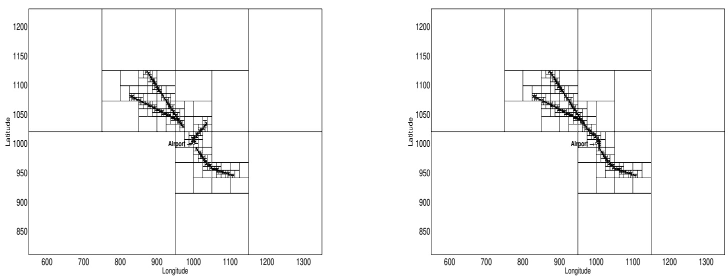
$\quad \#\boldsymbol{x}_{\rho v} \leftarrow 0$

$\quad$ Delete the two child nodes $\rho v\mathsf{L}$ and $\rho v\mathsf{R}$

B. A Dynamic Airspace Data Structure

Say we are only interested in tracking flights that are currently in the airspace. In other words, in some time interval t, we only want information of aircraft that are still in flight and have no need for any information about aircraft that have already landed. We achieve this by constructing aggregate SRP trajectories that specifically describe the airspace for a sequence of time intervals.

For example, Fig. 10a shows the aggregate SRP trajectory of three aircraft in some time interval. One of the aircraft has already landed at the airport (represented by the black dot at point [1000 $\times$ 1000]). At the next time interval, the trajectory of the landed flight is removed or subtracted from the aggregate SRP trajectory, producing empty leaf nodes that initially held information of the removed flight. Now using `ReuniteEmptyNodes`, we reunite any pair of sibling leaf nodes that are empty to obtain an aggregate SRP trajectory that only has information of the flights in the air, as seen in Fig. 10b. We thus obtain a dynamic airspace data structure where, at every time interval, the corresponding aggregate SRP trajectory only keeps information of the aircraft that are in the airspace. This is an instance of the $n_{1:m}$-presensed setting where trajectory data enters the structure at various timestamps. In other applications, another given property of the leaf nodes may be more appropriate than the property of being empty. In such cases, Algorithm 4 can be modified to reunite nodes with the given property.



(a) A plane is just about to land. (b) Trajectory of the landed plane removed.

Fig. 10 A dynamic airspace data structure using SRPs.

VI. Complexity

We compare the space or memory as well as the time requirements of our data structures with those of a regular grid.

A. Space Complexity

The setup of the construction of SRP trajectories prevents unnecessary splitting on places without any flight visitations, i.e., as long as a box is empty, there is no need to bisect that box.

This is in contrast to a regular grid construction where the root box is split until each cell has the same volume: in this case, λ, in order to represent aircraft position data. Note that we are storing recursively computable statistics for each box or cell. Thus, the number of boxes or cells reflects the memory requirements of the different data structures.

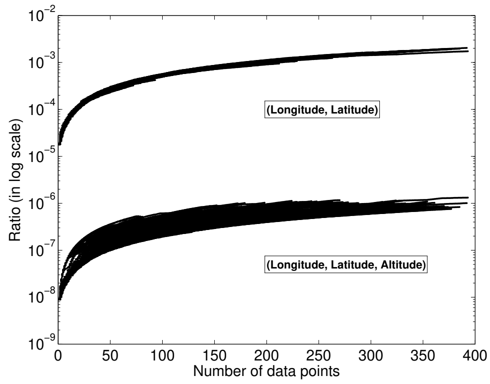
Suppose the regular grid is over a hypercube with box $\boldsymbol{x} = [0,1]^d$. Let h be the side length of a cell in the grid along each of the d coordinates. Then, the volume of the cell is $h^d = \lambda$. Now let $m = 1/h$ be the number of cells along each coordinate. Then, there are m^d many cells in a regular grid representing aircraft trajectory where

$$m^d = \left(\frac{1}{h}\right)^d = \frac{1}{\lambda} \ .$$

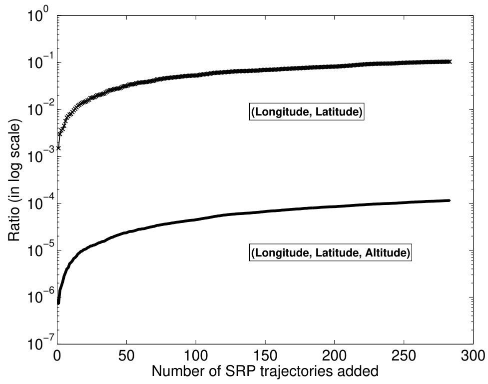
Thus, the memory requirements needed by a grid to represent aircraft position data grows exponentially with dimension d, resulting in the need for heavy computational storage requirements to produce data based on a grid. This is not desirable for massive data problems, especially in high dimensions.

The gain in memory for larger values of d is significant using SRPs as opposed to regular grids. In the graphs of Fig. 11, the ratio of the number of nodes in an SRP data structure to the number of cells in a regular grid in shown in log scale. Figure 11a shows the ratios for SRP trajectories with coordinates (latitude, longitude) and (latitude, longitude, altitude), respectively. Fig. 11b shows the ratios for aggregate SRP trajectories. Fig. 11c gives the ratios for a dynamic structure for historic data blocked into 30 min and is in a $n_{1:m}$-presensed setting, i.e., the k-th "historical" burst is from an interval of 30 min. Figure 11d has ratios for a dynamic structure constructed using data arriving from the radar or a $n_{1:m:\ldots}$-sensing setting. The kth time interval corresponds to the radar time for which we set at 10 s here. As d increases from 2 to 3, the ratio decreases by three orders of magnitude.

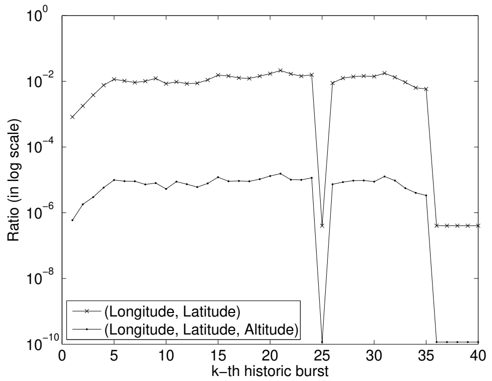
Let the root box of the SRP trajectory be the same as that of the regular grid. The SRP trajectory requires $2j+1$ nodes to represent aircraft position data, where j is the total number of leaf nodes. As data are inserted into the SRP tree, the number of splits is determined by various factors, which include $\lambda = h^d$ (and thus the maximal depth $i^* = \lceil d \log_2(1/h) \rceil$ of the SRP), the number of data points, and interestingly, the position of the aircraft relative to its history in the



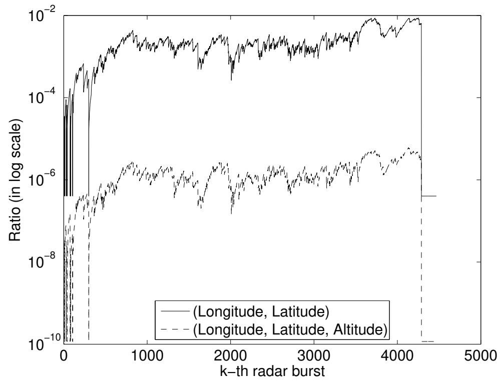
(a) Individual SRP trajectories.



(b) Aggregate SRP trajectories.



(c) Dynamic airspace structure ($n_{1:m}$-presensed).

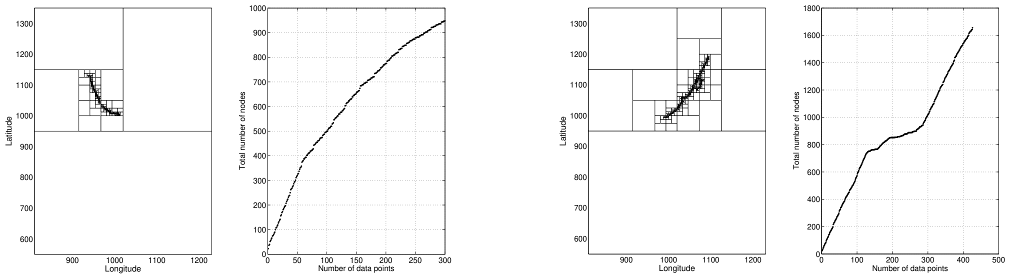


(d) Dynamic airspace structure ($n_{1:m:\ldots}$-sensing).

Fig. 11 Ratio of the number of nodes in an SRP to the number of cells in a regular grid.

SRP tree. For instance, fewer nodes are required to represent data that are clustered closely and/or if aircraft are repeatedly visiting the same areas. An example is given in Fig. 12 for two SRP trajectories for position data of aircraft A and B. A flat plateau is observed between point 150 to point 280 for aircraft B in contrast to the fairly straight curve of aircraft A. A more detailed analysis shows that the flat plateau is a result of aircraft B circling around the vicinity of $[1100, 1075]$, i.e., points are likely falling into the same boxes due to repeated visits, and thus there are fewer splits of the data structure around that vicinity.

Now suppose n, the number of data points in our set of trajectories, is so large such that $n = (1/h)^d = m^d$, and that each cell in the grid contains at least one point. The corresponding SRP will then have $m^d - 1$ splits resulting in $2m^d - 1$ nodes, i.e., $\mathcal{O}(m^d)$, which is just as memory intensive as constructing a grid. This is, however, the worst-case scenario with a completely occupied airspace. For any data point that enters the SRP, at most, i^* number of splits are required for the



(a) Trajectory of aircraft A. (b) Trajectory of aircraft B.

Fig. 12 Total number of nodes needed to represent two aircraft trajectories.

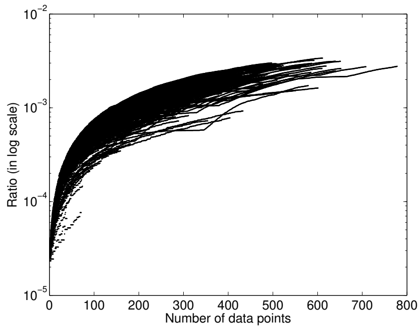
point to be enclosed in a box of depth i^*. The number of nodes required in the SRP is then

$$k \leq \min \left\{ ni^* = nd \log_2 \left(\frac{1}{h}\right)^d, 2m^d - 1 \right\} \ .$$

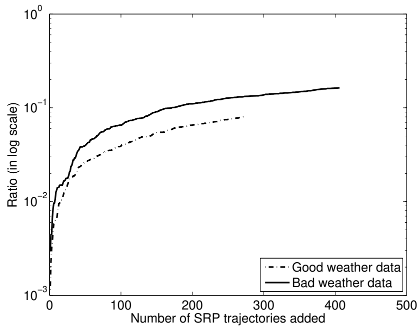
In the analyses, we are working with data from actual flights that generally fly on established routes between established waypoints. Thus, we do not expect actual aircraft trajectory data to evenly fill the entire airspace. A dynamic SRP structure as discussed in Sec. V allows the SRP tree to grow and shrink as needed such that the number of nodes required stabilize as the number of aircraft in airspace reaches a steady state. Figure 13 shows, in log scale, the ratio of the number of nodes in an SRP to the number of cells in a regular grid given λ for time-ordered position data on a good and bad weather day. Figure 13a shows the ratio (about 0.08) for individual trajectories; Fig. 13b gives the ratio (about 0.16) for aggregated trajectories on a good and bad weather day. Finally, Fig. 13c gives the ratios at various time intervals of the dynamic airspace structure in a $n_{1:m}$-presensed setting with time intervals of 30 min.

B. Time Complexity

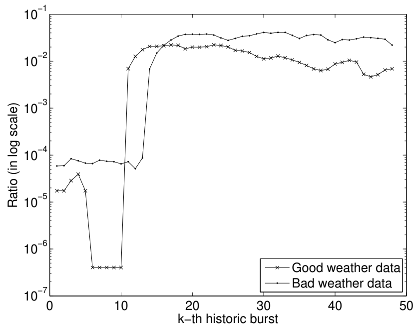
The regular grid, being a random access container, allows for instantaneous insertions and deletions in any dimension. For an SRP trajectory, if we measure time in units of nodes traversed or created, then each data point requires i^* time units to reach the box at depth i^* that encloses it. Thus, the time complexity is $\mathcal{O}(i^*) = \mathcal{O}(d \log_2 m)$, i.e., linear in d (when h is identical in each of the d coordinates). This is slower than a grid but, given that radar data arrives in bursts of 4-12 s, an SRP trajectory can easily be updated in time for online analysis. If the insertion time of a data



(a) Individual SRP trajectories.



(b) Aggregate SRP trajectories.



(c) Dynamic airspace structure

($n_{1:m}$-presensed).

Fig. 13 Ratio of the number of nodes in an SRP to the number of cells in a regular grid.

point into a regular grid requires one CPU time unit, then we will need i^* CPU time units to insert a data point that is enclosed by a box at depth i^* of the SRP trajectory. Thus, for a given data point, the ratio of its insertion time into a regular grid to its insertion time into an SRP trajectory is i^*. Figure 14 gives the CPU times needed for such online analysis corresponding to the structures developed in Fig. 11d. We observe that the timings are reasonable to handle data arriving in bursts of 10 s from the radar. All of our programs were run on a machine with dual Intel X5670 2.93Ghz 6 core Xeon CPUs, 48GB of RAM, 2 x 320GB 15K serial attached small computer system interface (SAS) hard drives, and an OpenSuSE 11.2 (x86 64) operating system.

VII. Conclusions

The potential utility of SRPs in the context of aviation systems research has been pointed out. In particular, such data structures can be used to show which sections of airspace are most often occupied in different contexts (related to weather, time-of-day, etc.).

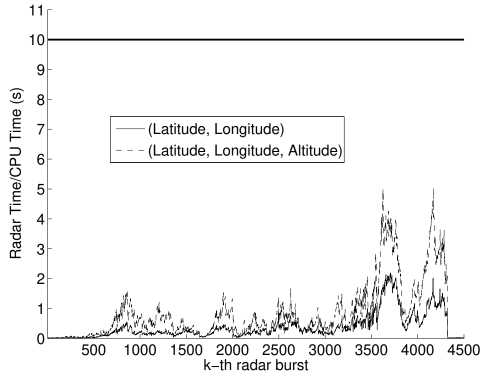


Fig. 14 CPU times used when building dynamic structures for online analysis.

In addition, it has been shown how SRPs can be used to summarize massive data sets by their unique rule that focuses computational resources on areas containing data points. The SRP has also been compared with a grid in terms of memory requirements and time complexity. It was shown that, although an SRP needs $\mathcal{O}(d \log_2 m)$ time units and is less efficient in terms of time compared to a grid, the SRP is more memory efficient when representing aircraft position data, especially in higher dimensions.

The recursive bisections allow arithmetic to be naturally extended to SRPs. Thus, arithmetic operations (addition, subtraction, multiplication, etc.) were able to be developed for aircraft trajectory data: for example, aggregating trajectories over a given block of time, or taking the difference of two such aggregate trajectories on a good day and a bad day to see how much change in traffic there is between the two days. This is a step towards real-time classification and prediction of airspace usage, which in turn will be useful to monitor and manage air traffic controller workloads, local environmental impacts of aviation systems, airport and airspace throughput, etc.

It has also been shown that it is possible to reunite sibling nodes when information in the sibling nodes is not needed anymore. This helps prune the tree associated with the SRP trajectory and prevents the tree from over-growing (unnecessarily). With this capability, the SRP was transformed into a dynamic structure that allows tracking of flights in the airspace over time.

Finally, although SRPs have been used to represent the aircraft position in two or three coordinates through time, position, velocity, fuel-level and other real-valued measures of the aircraft could just have easily been used with a higher-dimensional root box. Arithmetic over SRPs generalize to any dimension and the memory requirements become significantly less demanding.

Acknowledgments

This research was partly supported by RS's external consulting revenues from the New Zealand Ministry of Tourism, a Visiting Scientist award from Theoretical Statistics and Mathematics Unit of Indian Statistical Institute, Bangalore Centre, and a sabbatical grant from the College of Engineering, University of Canterbury, Christchurch, New Zealand.

References

[1] Lester, E.A., and Hansman, R.J., "Benefits and Incentives for ADS-B Equipage in the National Airspace System", M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, Sept. 2007.

[2] Kuhn, K., "Analysis of Thunderstorm Effects on Aggregate Aircraft Trajectories", *Journal of Aerospace Computing, Information, and Communication*, Vol. 5, No. 4, 2008, pp. 108–119. doi:10.2514/1.34830

[3] Hurter C., Tissoires B., and S. Conversy. "FromDaDy: Spreading Aircraft Trajectories Across Views to Support Iterative Queries", *IEEE Transactions On Visualization and Computer Graphics*, Vol. 15, No. 6, 2009, pp. 1017–1024. doi:10.1109/TVCG.2009.145

[4] Wilkerson J., Jacobson M., Malwitz A., Balasubramanian S., Wayson R., Fleming G., Naiman A., and Lele S., "Analysis of emission data from global commercial aviation: 2004 and 2006", *Atmos. Chem. Phys.*, Vol. 10, 2010, pp. 6391–6408. doi:10.5194/acp-10-6391-2010

[5] Callaham, M., DeArmon J., Cooper A., Goodfriend J., Moch-Mooney D., and Solomos J., "Assessing NAS Performance: Normalizing for the Effects of Weather", *In Proceedings of the 4th USA/Europe Air Traffic Management R&D Symposium*, Santa Fe, NM, 2001.

[6] Agarwal P., Gao J., and Guibas L., "Kinetic Medians and kd-Trees", *In Proceedings of the 10th Annual European Symposium on Algorithms (ESA '02)*, London, 2002, pp. 5–16.

[7] Tao Y., and Papadias D., "Historical spatio-temporal aggregation", *ACM Trans. Inf. Syst.*, Vol. 23, No. 1, 2005, pp. 61–102. doi:10.1145/1055709.1055713

[8] Jaulin, L., Kieffer M., Didrit O., and Walter E., *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

[9] Samet, H., *The Design and Analysis of Spatial Data Structures*, Addison-Wesley Longman Publishing, Boston, 1990.

[10] Sainudiin R., and Harlow J., "A C++ Class Library for Statistical Set Processing", In Rajendra Bhatia (Ed.), *Short Communication in Mathematical Software, International Congress of Mathematicians*, edited by R. Bhatia, Hindustan Book Agency, India, Aug. 2010, p. 670.