International Journal of Geo-Information

MDPI

Article

A Low-Altitude Flight Conflict Detection Algorithm Based on a Multilevel Grid Spatiotemporal Index

Shuangxi Miao [1], Chengqi Cheng [1], Weixin Zhai [2,*], Fuhu Ren [3], Bo Zhang [4], Shuang Li [5], Junxiao Zhang [6] and Huangchuang Zhang [1]

1 College of Engineering, Peking University, Beijing 100871, China; miaosx@pku.edu.cn (S.M.); ccq@pku.edu.cn (C.C.); zhanghuangchuang@pku.edu.cn (H.Z.)
2 Academy for Advanced Interdisciplinary Studies, Peking University, Beijing 100871, China
3 Beijing Institute of Big Data Research, Beijing 100871, China; fuhuren@gmail.com
4 Heilongjiang Institute of Geography Information Engineering, Harbin 150081, China; zhangbo851028@gmail.com
5 Institute of Remote Sensing and GIS, Peking University, Beijing 100871, China; lishuang0928@foxmail.com
6 Faculty of Geosciences and Environmental Engineering, Southwest Jiaotong University, Chengdu 611756, China; zjxgisswjtu@my.swjtu.edu.cn
* Correspondence: pkuzhaiweixin@gmail.com

Received: 5 April 2019; Accepted: 15 June 2019; Published: 21 June 2019

Abstract: Flight conflict detection is fundamental to flight dispatch, trajectory planning, and flight safety control. An ever-increasing aircraft population and higher speeds, particularly the emergence of hypersonic/supersonic aircrafts, are challenging the timeliness and accuracy of flight conflict detection. Traditional trajectory conflict detection algorithms rely on traversing multivariate equations of every two trajectories, in order to yield the conflict result and involve extensive computation and high algorithmic complexity; these algorithms are often unable to provide the flight conflict solutions required quickly enough. In this paper, we present a novel, low-altitude flight conflict detection algorithm, based on the multi-level grid spatiotemporal index, that transforms the traditional trajectory-traversing multivariate conflict computation into a grid conflict state query of distributed grid databases. Essentially, this is a method of exchanging "storage space" for "computational time". First, we build the spatiotemporal subdivision and encoding model based on the airspace. The model describes the geometries of the trajectories, low-altitude obstacles, or dangerous fields and identifies the grid with grid codes. Next, we design a database table structure of the grid and create a grid database. Finally, we establish a multilevel grid spatiotemporal index, design a query optimization scheme, and examine the flight conflict detection results from the grid database. Experimental verification confirms that the computation efficiency of our algorithm is one order of magnitude higher than those of traditional methods. Our algorithm can perform real-time (dynamic/static) conflict detection on both individual aircraft and aircraft flying in formation with more efficient trajectory planning and airspace utilization.

Keywords: multilevel grid spatiotemporal index; aircraft tracking; conflict detection

1. Introduction

Between 2006 and 2017, China recorded an average annual aircraft movement increase of approximately 9.7% [1]. The aviation safety of manned/unmanned, civil/military, and hypersonic/ultrasonic aircrafts is constrained by complicated environmental factors, such as low-altitude terrain, obstacles, and extreme weather conditions [2,3]. In low-traffic-density civil flights, the flight conflict results can be calculated in advance according to flight plans, whereas in high-traffic-density or non-civil

airspace applications, the speed and accuracy of flight conflict detection are challenging concerns. In fact, flight conflict detection is already a widely researched topic in the field of aviation safety.

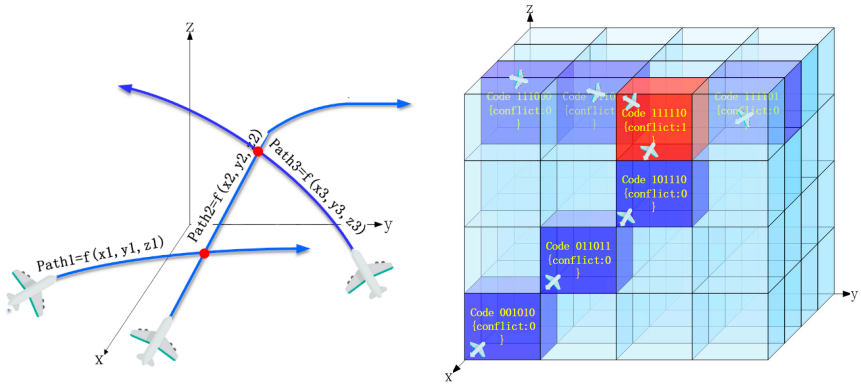
As the research in the field deepens, authors are shifting their focuses from the direct polynomial trajectory intersection of two aircrafts to the partitioning of flight regions from three-dimensional spaces to four-dimensional spaces, and from aircraft to spatiotemporal domains [4–7]. Traditional flight conflict detection methods include deterministic methods and probabilistic methods.

Deterministic methods use computational geometry to produce the conflict results and determine whether potential flight conflicts exist within the encounter geometry of two aircraft (air vehicles). For example, Fulton [8–10] used an unstructured grid method for flow field computation to detect conflicts. This method detects existing or potential conflicts among a number of airplanes at one time and, by ignoring unnecessary distance detection, reduces the amount of computation. The management of the entire airspace, however, is restricted, as this method assumes the trajectory to be a Delaunay-based plane triangle. Another example is the axis-aligned bounding box (AABB) algorithm, which approximates the original spatial entity level-by-level by constructing a tree hierarchy. AABB intersection testing requires a maximum of six comparative operations [11], which is time-consuming. By quickly and accurately identifying flight conflicts between two trajectory points that are less than a safe distance apart, many high-dimensional spatial index structures and approximate query algorithms are proposed. One of them is the k-d tree. The Bkd-tree maintains its high space utilization and excellent query and update performance regardless of the number of updates performed on it [12]. However, the method needs to be carried out by calculating the distance between the root node and the track point, which is a time-consuming process.

Probabilistic conflict detection [13–15] calculates the conflict probabilities between air vehicles by a support vector machine, genetic algorithm, or some other approach. These methods involve complicated probability computations, and their thresholds are selected subjectively. Excessively high thresholds may lead to missed detections, while excessively low thresholds could give rise to extensive mis-detection.

In brief, traditional flight conflict detection is based on assumptions, such as a limited flight angle, constant speed, and no exposure to environmental impacts [16,17], but errors caused by uncertainties in navigation or positioning, pilot operations and spatiotemporal environmental variations on the underlying surface, can directly affect the conflict detection accuracy. Deterministic methods use coordinates to solve and determine whether conflicts exist between aircrafts; traversing multivariate approaches involves a high algorithmic complexity and extensive computation; and probabilistic methods calculate flight probabilities using a complicated process. None of these methods provide the flight conflict detection speed required for an efficient dispatch.

In view of these problems, a new, low-altitude flight conflict detection algorithm, based on a multi-level grid spatiotemporal index, is proposed that transforms traditional trajectory-by-trajectory multivariate equation conflict solutions into a real-time conflict state query of distributed databases. As shown in Figure 1, the traditional flight conflict detection method is performed by calculating the intersection between two trajectories at a time, while in the new approach, two planes or the plane and the obstacle appear in the same grid at the same time during a grid conflict state.



(**a**) Traditional method (**b**) The proposed method

Figure 1. Schematic diagram of different methods.

2. Materials and Methods

In this paper, a low-altitude flight conflict detection algorithm, based on a multi-level grid spatiotemporal index is proposed, that transforms trajectory conflict detection from multivariate solutions into an efficient conflict state query of distributed grid databases. The distributed grid databases are established by discretizing trajectories, low-altitude obstacles or low-altitude dangerous fields into a multiscale discrete grid system. Essentially, it is a method of exchanging grid database storage space for traversing time. As shown in Figure 2, our algorithm is organized as follows. First, we built a flight airspace total spatiotemporal subdivision and encoding model that subdivides and encodes the flight trajectories, and low-altitude obstacles and fields, within the flight airspace to achieve a multilevel grid discretization and data identification. Next, we designed a grid database table structure and created a multilevel grid database. Finally, we established a global multiscale grid spatiotemporal index. When the amount of data had increased dramatically, a database partition table and a priority policy for a temporal or spatial index are designed to improve the efficiency of the conflict state query. The workflow is shown in Figure 2.

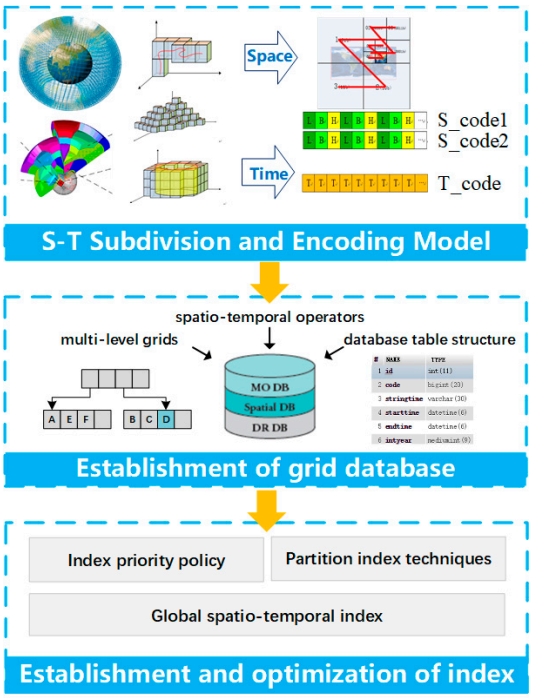


Figure 2. Workflow of the flight conflict detection algorithm based on the multilevel grid spatiotemporal index.

2.1. Spatiotemporal Subdivision and Encoding Model

Subdivision is the process of discretizing aircraft trajectories, low-altitude obstacle elements, and dangerous fields over spatial and temporal dimensions. Encoding is the process of identifying the spatiotemporal attributes of subdivision elements. Subdivision and encoding organize the discretized subdivision elements in an orderly way [18,19].

A total spatiotemporal subdivision and encoding model was built to describe the geometries of flight paths, low-altitude obstacles, and dangerous fields with multilevel, seamless, and unfolded subdivision element sets ("grids") and to identify each scale of the subdivision element codes. To build such a model, first, we need to discretize the total time-space domain into "three-dimensional (3D) space subdivision grid elements" and "time subdivision grid elements"; then, we need to re-aggregate the flight trajectories, low-altitude obstacles, and dangerous fields with subdivision grid elements (time elements and space elements) [as shown in Equation (2)], and finally identify the subdivision grid elements with codes [as shown in Equation (3)].

$$Grid_{space\ time} \overset{Subdivision}{\leftarrow} \left\{Grid_{space} + Grid_{time}\right\}, \tag{1}$$

$$\{Line\ element, Vol\ element\} \overset{Cluster}{\leftarrow} Grids, \tag{2}$$

$$Codes \overset{code}{\leftrightarrow} Grids, \tag{3}$$

$$Model_{flight\ space\ Sub-Code} = \left\{Codes = \mathrm{U}_{i=1}^{n}(Code_{space\,i} + Code_{time_i})\right\}. \tag{4}$$

The flight airspace total spatiotemporal subdivision and encoding model, as shown in the equations above, essentially involve aggregating the subdivision elements into spatial objects using the spatial finite element concept. Equation (1) indicates that the time and space attributes are recorded by the subdivision elements. Equation (2) aggregates the subdivision elements into flight paths, low-altitude obstacles, and dangerous fields. Equation (3) indicates that, within the subdivision space, each subdivision element maps a unique subdivision code. In Equation (4) the operator "+" means we can connect the spatial code and the temporal code. Equation (4) shows that the flight airspace total spatiotemporal subdivision and encoding model is an orderly set of subdivision element codes and that each subdivision code forms part of all subdivision code sets within the subdivision space.

One-dimensional coding consists of temporal coding and spatial coding, where the spatial coding contains the level information (spatial scale information) and location information, and the temporal coding contains the level information (temporal scale information) and start time.

2.1.1. Space Subdivision

GeoSOT Subdivision Technique

Space subdivision grids, which are discrete, hierarchical, and globally continuous, address both the computer management of continuous spaces and the discretization of aircraft paths and other elements [20].

To better describe the sizes of the aircraft trajectories and low-altitude obstacles, we base our study on an existing global longitude and latitude subdivision grid of 2^n one-dimension-integer arrays, GeoSOT (geographical coordinate grid subdivision by one-dimension-integer and two to the n-th power) [20]. GeoSOT subdivision essentially extends the longitude and latitude intervals from the eight basic grids up to 512° (2^9°), interpolates four levels between 1° and 2′ and another four levels between 1′ and 2″, and extends the part below 0.5′ to 1/2048″ to yield a graticule-based subdivision system of {2^8°, 2^7°, 2^6°, 2^5°, 2^4°, 2^3°, 2^2°, 2^1°, 2^0°, 2^5′, 2^4′, 2^3′, 2^2′, 2^1′, 2^0′, 2^5″, 2^4″, 2^3″, 2^2″, 2^1″, 2^0″, 2^{-1}″, 2^{-2}″, 2^{-3}″, 2^{-4}″, 2^{-5}″, 2^{-6}″, 2^{-7}″, 2^{-8}″, 2^{-9}″, 2^{-10}″, and 2^{-11}″}.

The Earth's surface space is expanded from 180° × 360°to 512° × 512°; In the eight basic grids, the 1st level 256° × 256°, the 2nd level 128° × 128°, the 3rd level 64° × 64°,the 4th level 32° × 32°. the 5th level 16° × 16°. the 6th level 8° × 8°. the 7th level 4° × 4°. the 8th level 2° × 2°.the advantage of the

expanding segmentation is that the 0–9 layers of grids are all integer degree segmented. In the same way, 1° × 1°(60′ × 60′) is expanded into 64′ × 64′, which ensures that the 9–15 level grids are all integer minutes segmented. While, 1′ × 1′ (60″ × 60″) is expanded into 64″ × 64″, which ensures that the 15–21 level grids are all integer seconds segmented.

According to the above mentioned definition of the subdivision hierarchy, GeoSOT contains 32 levels spanning from the global to the centimeter scale, and from the Earth's core up to the solid space of the Earth's synchronous orbit, which is 60,000 km above the ground level. It equally divides (same latitude and longitude scale) the Earth's surface space into multilevel grids, which constitute a global quad-tree system. The area ratios between adjacent levels of the GeoSOT grids vary equally at approximately 4: 1, as shown in Figure 3.

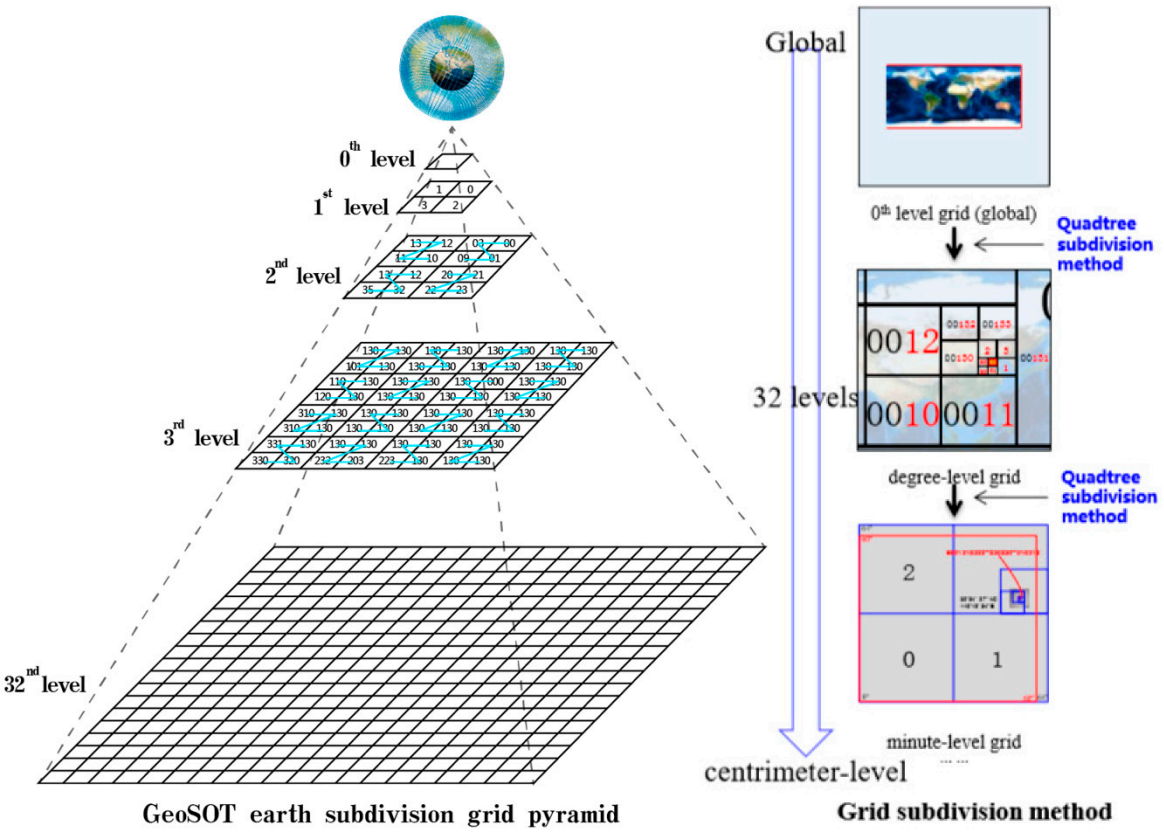


Figure 3. Sketch showing the hierarchy of the Earth's subdivision patches.

Flight Path Expression

According to the GeoSOT grid cells, when aggregating paths, one needs to discretize the path segments and aggregate them with multiscale grids as shown in Figure 4.

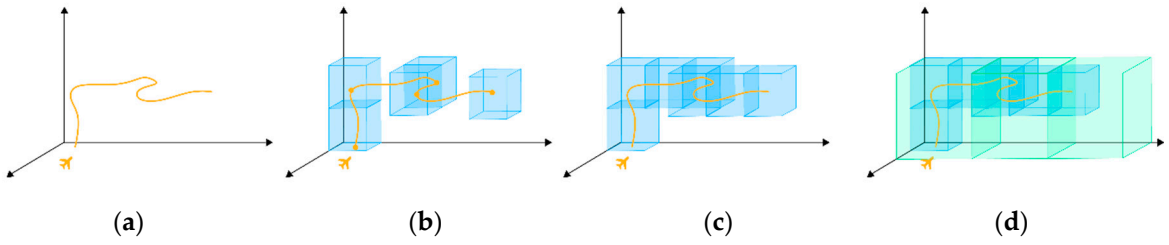


Figure 4. Path discretization. (**a**) flight path; (**b**) GeoSOT grid cells at key points along the flight path; (**c**) The path is discretized into higher level GeoSOT grids (smaller grid); (**d**) The path is discretized into lower level GeoSOT grids (bigger grid).

Grids = {spatial attributes, temporal attributes, codes} is the minimum element of a subdivision. If a point is contained in a grid space, this may be expressed by a grid approximation. A grid approximates a point when its side length is sufficiently small. After determining grid levels L1 and L2, we can discretize the grid using the following Equations:

$$\vec{Point} = \{Grid_{L1P}\}, \tag{5}$$

Then, after L1 is discretized, the path is described as follows:

$$\vec{Line} = \left\{Grid_{L1P_1}, Grid_{L1P_2}, Grid_{L1P_3}, \ldots, Grid_{L1P_n}\right\} = \left\{\mathrm{U}_{i=1}^{n} Grid_{L1p_i}\right\}, \tag{6}$$

If the path is discretized with the L2 grids, then

$$\vec{Line} = \left\{Grid_{L2P_1}, Grid_{L2P_2}, \ldots, Grid_{L2P_m}\right\} = \left\{\mathrm{U}_{j=1}^{m} Grid_{L2p_j}\right\}. \tag{7}$$

In Equation (7), if L1 < L2, then n < m.

In an airspace, the total spatiotemporal subdivision and encoding model, a multiscale model of trajectories is needed within the airspace. The following path data discretization rules need to be established:

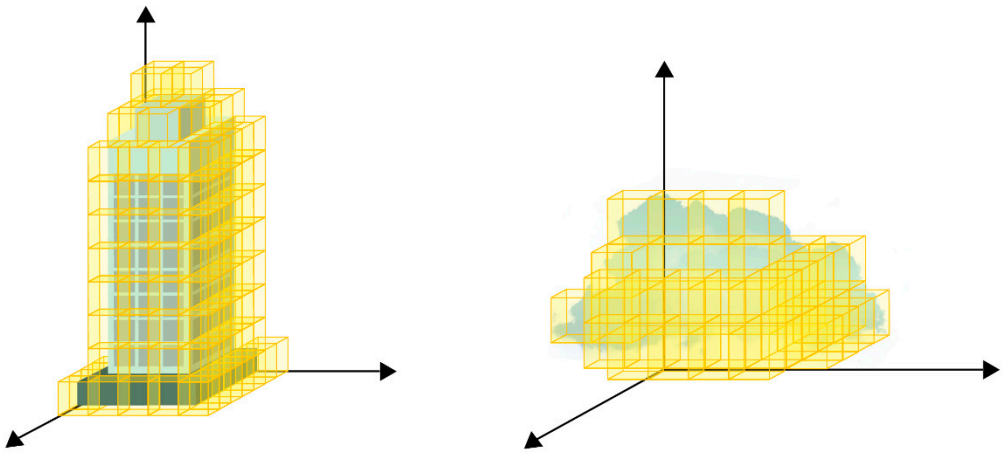
- The selected grid side length must be positively correlated to the speed (the faster the speed, the larger the side length of the grid).
- The selected trajectory-scale level must neither be too high nor too low, since low levels would make the grids too large to accurately describe the true flight trajectories and excessively high levels could lead to significant trajectory fragmentation.
- To highlight the benefits of grid-coded multiscale query and retrieval, a two-level or multilevel grid database is normally established to utilize the multiscale of subdivision grids by downward segmentation and upward aggregation. When dividing sub-tables by row, lower-level patches are included into the same sub-table as elements of the upper-level patches to improve the efficiency of data retrieval.
- To utilize the benefits of multiscale grids, the grid side length may be used as a criterion for conflict thresholds. If the flying safety distance is 1000 m, for example, the level-16 grids of GeoSOT may be used [21].

Volume Element and Field Expression

Flying safety is subject to the environmental features of the underlying surface, such as high-rises, lightning fields, or storm fields. Hence, the flight conflict detection must analyze the expressions of these volume elements and fields. Here, the expressions of high-rises and simple fields are all treated as volume-state expressions. The buildings and dangerous field are expressed by aggregating a number of grids, as shown in Figure 5.

In the same way, volume elements and fields are clustered with subdivision grids on two levels.

$$\begin{aligned} \vec{Volume} &= \left\{Grid_{L1P_1}, Grid_{L1P_2}, \ldots, Grid_{L1P_m}\right\} = \left\{\mathrm{U}_{i=1}^{m} Grid_{L1p_i}\right\} \\ &= \left\{Grid_{L2P_1}, Grid_{L2P_2}, \ldots, Grid_{L2P_m}\right\} = \left\{\mathrm{U}_{j=1}^{n} Grid_{L2p_j}\right\}, \end{aligned} \tag{8}$$



(**a**) Discretization of buildings (**b**) Discretization of dangerous fields (thunderstorm field)

Figure 5. Subdivisions of volume elements and dangerous fields.

2.1.2. Time Subdivision

Time subdivision includes the subdivision of time points and the subdivision of time spans. A time point is the duration of a moment, whereas a time period is the time spanning from the start to the end. Like space subdivision, time can be subdivided into different granularities such as years-months-days-hours-minutes-seconds-milliseconds-microseconds. In actual application, the exact time granularity must be selected as practically required.

During trajectory conflict detection, it is unnecessary to model all scales of time granularity. According to the required computational efficiency, only the second-scale subdivision of the time domain is needed in the method, One-scale and multiscale subdivision as shown in Table 1.

Table 1. The differences between one-scale and multiscale temporal subdivisions.

One-Scale Subdivision			**Multiscale Subdivision**					
1970-01-01T:00:00Z-			Year					
0 S	1 S	2 S	January	February	March	...	December	
			1st	2nd	...		364th	365th

2.1.3. Unified Spatiotemporal Encoding

Subdivision codes, as identifiers of subdivision elements must, not only express the spatiotemporal attributes of the subdivision elements, but also be applicable to the efficient query/retrieval and analysis/computation among subdivision elements. Typical encoding methods include hierarchical encoding and space-filling curve encoding [21].

As each code element of hierarchical encoding records the parent elements that has passed before arriving at the current element, the encoding scheme essentially describes the "tree" structure of the affiliations between different levels of elements within a grid system. In a coding system with a deep hierarchy and many leaf nodes, the code query efficiency is low; the filling curve encoding operation can design different space-filling curves, such as Hilbert curves, Peano curves, Sierpinski curves, and Morton (Z) curves, according to the nature of the mapping function.

In general, space-filling curves feature spatial aggregation, namely, neighboring elements in two-dimensional spaces are still neighbors with each other when mapped on the space-filling curve. This approach adequately addresses the demand to express the neighborhood between the elements and allows efficient neighborhood querying during flight conflict detection. The spatial z-order has the advantages of simplicity and intuitionism in spatial encoding compared to other approaches. Hence,

we propose a z-order binary encoding method that satisfies the requirement of direct computer storage and makes the best use of bitwise operations on binary numbers.

GeoSOT-3D Subdivision Volume Element Encoding Algorithm

The GeoSOT-3D ellipsoid subdivision space is a discretized (digitalized) abstraction of the entire geospace. The foundation for establishing a discretized GeoSOT-3D subdivision space is the GeoSOT volume element encoding structure and method. GeoSOT encoding is based on the GeoSOT-3D space subdivision theory, as shown in Figure 6.

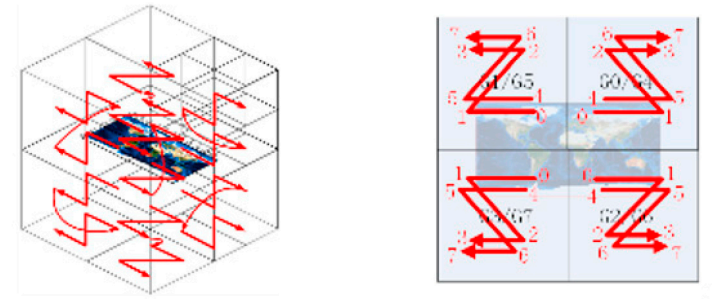


Figure 6. GeoSOT-3D space-filling curves.

GeoSOT-3D subdivides a space downward as an octree and encodes the elements in order, as shown in Figure 5a. The direction of z-order encoding is closely associated with the positions of the volume elements. In the northeast quadrant, for example, the longitude is encoded in an ascending manner, and the latitude and height are encoded on a bottom-up basis.

Binary 3D Space Encoding Algorithm

GeoSOT-3D binary encoding uses 3*32-bit binary numbers to indicate high coordinates of longitude and latitude. As shown in the charts below, the first binary bit of its encoding structure defines the south and north latitudes, east and west longitudes, and above and below the ground level. The integer degrees of latitude and longitude each occupy eight binary bits; the integer minutes of latitude and longitude each occupy six binary bits; the integer seconds of latitude and longitude each occupy six binary bits; and the fractions of a second of the integer latitude and longitude each occupy 11 binary bits. The longitude and latitude are expressed to an accuracy of 1/2048″, and the height is encoded from the Earth's core to 60,000 km above the ground level, as shown in Figure 7.

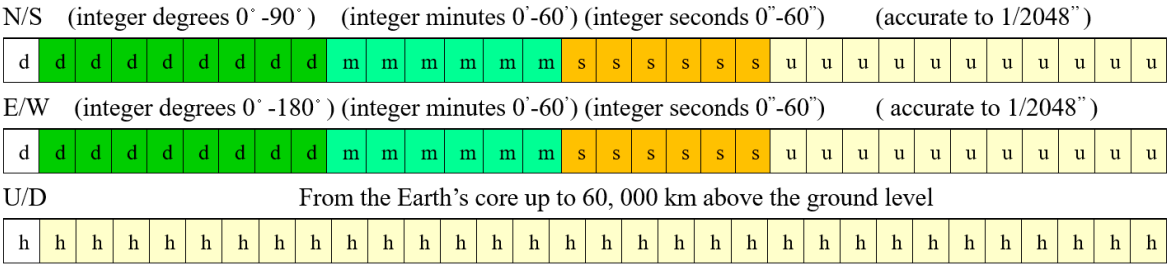


Figure 7. Geographical interpretation of GeoSOT binary three-dimensional (3D) code.

Binary Time Encoding Algorithm

In this paper, time period encoding is used to express the spatial extent of dynamic aircraft trajectories within a time period, in order to quickly detect path conflicts within the time dimension. Essentially, this method uses a maximum of 64-bit binary code to store time, by dividing the 64 bits from high bits to low ones, and stores the years, months, days, hours, minutes, seconds, milliseconds,

and microseconds. The method identifies the granularity of a time period with the length of the code. The longer the bits of a code, the smaller the time span is. The date, March 2, 2019, for example, is encoded by adding the code for "day" to the code for March 2019. The number of bits of a code reflects the size and hierarchical count of this code and, thereby, the chronology and hierarchical count of the time.

The year, month, day, hour, minute, second, and microsecond are different levels of the time scale. We use the time level of seconds in the method. In the case of "2019-03-02T10:21:50.000002", for example, the single-scale binary expression of the subdivision code is as follows:

$$\underbrace{0}_{unsigned}\ \underbrace{1000001111110001}_{year}\ \underbrace{1}_{month}\ \underbrace{0}_{day}\ \underbrace{0000101010}_{hour}\ \underbrace{010101110}_{minute}\ \underbrace{010000000000000000000010}_{second\ \&\ microsecond}$$

3. Database Table Design and Establishment and Optimization of the Index

In this paper, we propose a trajectory conflict detection algorithm, based on a multiscale grid spatiotemporal index. Designing a grid database and creating a grid spatiotemporal index are fundamental for efficient conflict state querying within grids. Hence, the design of the table structure in the grid database and the creation of a multiscale grid spatiotemporal index directly determines the efficiency of the trajectory conflict detection.

3.1. Database Table Structure Design

Traditional relational databases suffer from severe bottlenecks in expansibility, storage, and query behaviors. MongoDB, as an open-source NoSQL database written with C++ and based on file storage, supports data structures in BSON format, which is similar to the JSON format. It is loosely structured and schema-free and can store complicated types of data. Here, we created a conflict detection-oriented grid database, based on MongoDB, and use grid spatiotemporal codes as the keys for the database to make the best use of the query and retrieval behavior of binary one-dimensional encoding in efficient bitwise operation. The table structure is designed as follows:

Each of the subdivision elements maps a subdivision element code, and each subdivision element maps a number of aircraft objects (there can be more than one aircraft in one grid simultaneously). Hence, they can be organized by grid data and aircraft object data.

To enable the quick query and retrieval of grid data and the dynamics of different aircraft, two table structures, including a grid database table and objects data table, are designed. The grid database table was used to store the discretized grid data of a number of paths, low-altitude building objects, and dangerous fields. A multiscale grid code database was created to utilize the multiscale of grid codes. That is, if a conflict was detected in the parent grid, this conflict may be queried in the child grid to further confirm if such a conflict exists. If no conflict is detected in the parent grid, we will jump out of the query and conclude that no conflict exists in the child grid. The object data table is used to store the attributes of an aircraft object. The two tables are correlated with the aircraft object ID. This approach enables both the efficient query of the grid database and the query of the object attributes with high retrieval efficiency.

Database Table Structure Design of the aircraft trajectory as follows (Table 2):

Table 2. Database Table Structure Design 1 of the aircraft trajectory.

Database Table Structure	Instructions
{	
"_id": GeoSOT 3Dcode,	geosot code of aircraft tracking, building, etc.
"data_exist":1,	
"data":[	
{	
"dataid":0,	
"timecode":01010 010101 110010,	time code
"L":110.470244,	Latitude
"B":9.000059,	Longitude
"H":3307.167235,	Height
"up":1500.0,	top of aircraft tracking
"down":1500.0,	bottom of aircraft tracking
"left":75.0,	left width of aircraft tracking
"right":75.0,	right width of aircraft tracking
"v":197.2,	speed of aircraft
"type":2,	type of aircraft
"conflict":0,	conflict or not
"con_type":1,	conflict type of conflict
"OID":"PLANE2",	aircraft ID
"trackid":"track0"	id of aircraft tracking
]}	
}	

A distributed grid code database uses the GeoSOT codes as the main keys. The table contains as GeoSOT codes the longitude ("L"), latitude ("B") and height ("H") within the grid, the time codes ("timecode"), the extent of aircraft trajectories, including the height of the top of the aircraft tracking ("up"), the height of the bottom of the aircraft tracking ("down"), the left width of the aircraft tracking ("left"), and the right width of the aircraft tracking ("right"), the flying speed ("v"), the type of aircraft within the grid ("type"), whether a conflict exists ("conflict"), the type of conflict ("con-type"), the aircraft ID ("OID"), and the ID of the aircraft tracking ("trackid"). This design allows a normal query as well as a quick query of a conflict in a grid at a time point from the grid database.

The object data table is designed as follows (Table 3):

Table 3. Database Table Structure Design 2 of the object data.

Database Table Structure	Instructions
{"_id":	
{"$oid":"5ae18ec655c45e894619469c"},	
"OID":"PLANE2",	aircraft ID
"name":"J-20",	
"crew":1,	
"length(m)":21,	the number of crew
"Thrust/Weight":0.75,	
"span(m)":12.8,	
"height(m)":6.22,	
"practicalceiling(m)":15,500,	
"Maximum_range(km)":3650,	
"wing_area(m*m)":42.2,	wing_area
"empty_weight(kg)":14,500,	empty_weight
"Maximum_take-off_weight(kg)":27,415,	Max take-off_weight
"Maximum_flying_speed(ma)":1.7	Maximum_flying_speed
}	

GeoSOT codes are used as the database keys for "name", "crew (person)", "length (m)", "thrust/weight ratio", "span (m)", "height (m)", "practical ceiling (m)", "maximum range (km)",

"empty weight (kg)", "maximum take-off weight (kg)", and "maximum flying speed (ma)". Continuous trajectories are discretized into discrete grid code expressions with spatiotemporal attributes. An object attribute table is used to store the object attributes to minimize data redundancy. The data table and object table are correlated via the object "OID".

3.2. Establishment and Optimization Index

A multiscale grid spatiotemporal index is a multiscale index comprising multiscale grid time codes and multiscale space codes based on space subdivision and encoding. When establishing a multiscale spatiotemporal index, if a large number of aircraft and a large amount of grid data are involved, we need to create global multiscale grid spatial and temporal code indexes first, then partition the table according to the range of space to improve the query efficiency, and finally, determine the sequence of the spatial indexes and temporal indexes according to the sparseness of the space and time data and find the space and time domains, as shown in Figure 8.

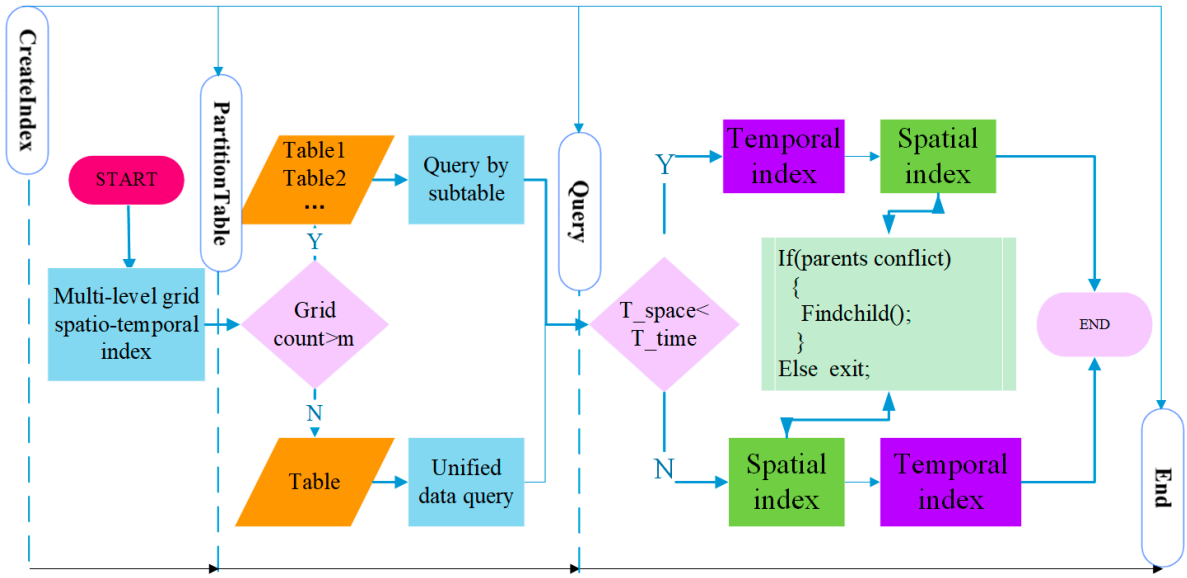


Figure 8. Multilevel grid spatiotemporal index and optimization method.

3.2.1. Global Spatial Index and Temporal Index of Multilevel Grids

Grid indexing is an indexing method that uses the space and time codes of multilevel grids as the conflict detection result, to allow a quick query. Unified spatiotemporal encoding is performed according to the total spatiotemporal subdivision model of the airspace. The performance of a grid index relies on the sizes of the grids and objects and the relationships between object densities [22,23]. For data with high spatial densities, a multilevel index grid system is usually created to provide optimum performance.

In multilevel grid indexing, when the user performs a spatial query, he/she first calculates the space grid hosting the queried object and uses this grid to quickly locate the selected space object. The grid indexing method is the most intuitive indexing method for space objects. Its corresponding algorithm is also simple enough for the quick query of the objects. This is a typical space-for-time indexing method whose high data redundancy does not prevent its wide acceptability owing to the high efficiency of grid indexing.

Multilevel grid spatial indexing is a method of querying flight conflicts in two levels of spatial grids by utilizing the multiscale structure of the GeoSOT subdivision grid. This method first queries conflicts in the parent grid and, if any conflict exists there, further queries its child grid and returns to the conflicted child grid code or, if no conflict exists, directly queries conflicts in the next parent grid. Essentially, this method is also a process of coarse filtration to fine filtration.

3.2.2. Partition Index Techniques

With the increasing quantity of large-area grid data, traditional centralized database systems are suffering from expansibility and performance bottlenecks and are far from addressing high-concurrency, low-delay real-time applications. Distributed organization and indexing techniques represent an effective solution to this problem. Logically, the distributed processing of space indexes involves cross-node segmentation and a local data storage structure.

However, when a large number of aircraft or long paths are involved and the data size of the grid database increases sharply, the efficiency of the conflict detection results will be compromised by traversing the index table. Hence, we propose the concept of a data partition, which partitions the index table and utilizes the merits of a distributed database query to provide the speed required for big data path conflict detection.

3.2.3. Space Index and Time Index Priority Policy

The spatiotemporal data distribution varies in granularity. In a grid database with sparse time elements, if a unified spatiotemporal index is used or if the space and time search method is used, the efficiency of the conflict detection results will be compromised by the unequal distribution of space and time data. For example, when the number of aircraft objects is determined and the flying time varies from 1 h to 24 h, a unified spatiotemporal index would limit the speed of the conflict detection. In view of this issue, we establish a priority policy for selecting the temporal or spatial index with the table partition result.

The time density (*T_time*) and space density (*T_space*) are defined as follows:

$$\begin{aligned} T\_time &= k_1 \frac{number_{grids}}{number_{tiime\ span}} \\ T\_space &= k_2 \frac{number_{grids}}{number_{objects}}, \end{aligned} \tag{9}$$

Here, $number_{grids}$, $number_{time\ span}$, and $number_{object}$ represent the total number of grids, time span, and number of aircraft objects, respectively; k_1 and k_2 are the weights of time and space, respectively.

3.2.4. Real-Time (Dynamic/Static) Conflict Detection

During static fight conflict detection, the paths are pre-discrete and stored in the database to form a grid database. During dynamic track conflict detection, this proposed method uses a technique, where only the new grid conflict state needs to be queried under the static conflict detection result (new paths are discretized into the grid then updated to the grid database). On the one hand, the amount of new path data is limited under normal circumstances; on the other hand, querying the specified grid code conflict state is efficient; in conclusion, in the real-time demand of flight conflict detection, both dynamic detection and static detection can be met.

4. Prototype System Implementation and Experiments

4.1. Case Description

The experimental data were sourced from the data of 300 paths of aircraft departing and landing at three airports in the southeastern coastal area of China at different times. In southeastern China, which is subject to frequent storms during the summer, flights must always be managed with high timeliness and accuracy. The aircraft data include attribute data and path data, the latter of which are presented in the format of time (s), longitude (°), latitude (°), and height (m). Some examples are as follows:

2018-08-14 15:21:07, 113.3080, 23.3244, 1097.28
2018-08-14 15:21:27, 113.3110, 23.3053, 1249.68

2018-08-14 15:21:45, 113.3275, 23.2952, 1463.04.

Sixty low-altitude obstacles and storm fields are organized, with the low-altitude obstacles appearing in the form of polygons + elevations. The coordinate string is formatted as start time, end time, longitude (°), latitude (°), the height of the top (m) of the obstacle, and the height of the bottom of the obstacle (m); in the case of a building, the start and end times are defaulted as always existing, and the height of the bottom of the obstacle is 0. Some examples are as follows:

2018-08-14 10:00:00- 2018-08-20 10:00:00, 106.1100, 34.3050, 300.0, 10000.00.

2018-08-14 10:00:00- 2018-08-20 10:00:00, 106.1200, 34.3310, 300.0, 10000.00.

2018-08-14 10:00:00- 2018-08-20 10:00:00, 106.1606, 34.3010. 300.0, 10000.00.

4.2. Prototype System Development

To verify the effectiveness of the theory proposed herein, a prototype experiment verification system was developed that supports the visualization of fundamental geophysical data and displays different scales of GeoSOT grids. To perform the experiment, first, one needs to pre-grid the path, low-altitude obstacles, and dangerous field data. Then, we need to establish a database of discretized gridded data. Next, we create a multiscale grid spatiotemporal index to obtain the aircraft trajectory conflict detection results under different algorithms and compare the query efficiencies of the conflict detection results among different algorithms.

4.2.1. Data Gridding

The path data is composed of a series of points. Before the data were added to the grid, polynomial interpolation was performed to obtain the real-time aircraft positions at second-scale time intervals. Then, based on the GeoSOT subdivision theory and according to the path data discretization rule, 300 paths were discretized using Equation (2) at a safety distance of 1 km (considering the possibility of supersonic aircraft) on level 14 (with an equator grid length of 4 km) and level 16 (with an equator grid length of 1 km) grids, and a 1-second time encoding granularity was selected. The gridding process covering both storm fields and low-altitude obstacles yielded 1,800,000 points of discretized grid data. To test the query efficiencies of our algorithm for different data sizes, we reproduced three copies of the data of the paths, low-altitude obstacles, and dangerous fields and yielded a maximum of 1,800,000 grids × 4 = 7,200,000 grid data. The query efficiencies were compared in the laboratory at 10 paths (50,000 grids), 50 paths (150,000 grids), 300 paths (1,800,000 grids), and 1200 paths (7,200,000 grids).

4.2.2. Creating a Database

After the discretized path data and low-altitude obstacle data were gridded, each grid data point was formatted according to the grid data in Tables 2 and 3 and documented. The path data were edited and imported into MongoDB, as shown in Figure 9.

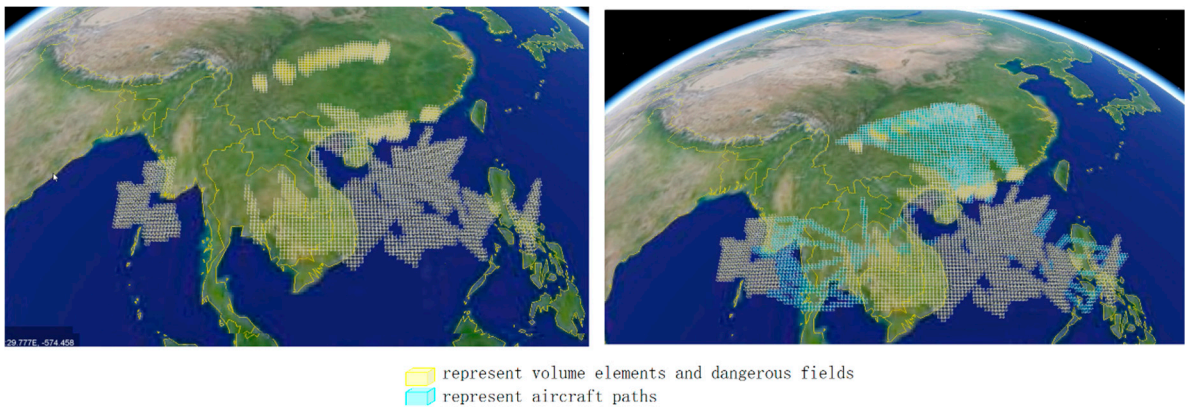


Figure 9. Discretized data expression.

4.2.3. Creating a Multiscale Grid Spatiotemporal Index

A global index was created in MongoDB to ensure the effectiveness of the partitioned index.

First, we created a joint index with the spatial codes and time codes of the grids as the keys. ({ "_id":1."**timecode**":1},{"_id":" **geosot 3Dcode**"}).

Next, a data partition table was created by analyzing the grid data size to improve the query efficiency of the distributed data. Finally, the time and space features were quantitatively compared by calculating the spatiotemporal sparseness of the data. As our experiment was conducted on civil aircrafts, which normally fly in the daytime, and whose T_time is less than T_space, according to Equation (9), the priority of the indexes was determined to be the time index before the space index. When querying the space index, if a conflict existed at a parent node, we entered into the child grid to query the conflicts there and returned the result. If no conflict existed at the parent node, we proceeded to the next parent grid. The aircraft object table was correlated to the attribute table with the OID.

4.2.4. Flight Conflict Range Search Method Based on the BKD-Tree

To compare the efficiency of the proposed method and the Bkd-tree method, we implemented the Bkd-tree in C++ using TPIE. We used a block size of 16 KB for the internal nodes (following the suggestions of reference [12] for the B-tree), resulting in a maximum fanout of 512. The leaves of a kd-tree, which were stored in 16 KB blocks as well, contain a maximum of 1364 (key, pointer) elements. We use the data set of discrete points of aircraft routes every second and set 500 m as a safe distance.

4.3. Results and Analysis

All trials were conducted on The ThinkPad T470p Signature Edition with the i7-7700HQ CPU. The machine had 16GB of memory. In the experiment, the conflict detection results between the current paths or between the paths and high-rises and storm fields were tested using a traditional polynomial equation algorithm, the range search method of the BKD-Tree, simple unified spatiotemporal code retrieval, and multiscale grid spatiotemporal indexing (our method).

To test the conflict detection times of the different methods under different data sizes, a conflict query was conducted on different data sizes of the path data, including data of 10 paths, 50 paths, 300 paths, and 1200 paths, as shown in Figure 10. The yellow grids represent volume elements and dangerous field, the cyan grids represent aircraft paths, and the red grids represent conflict areas.

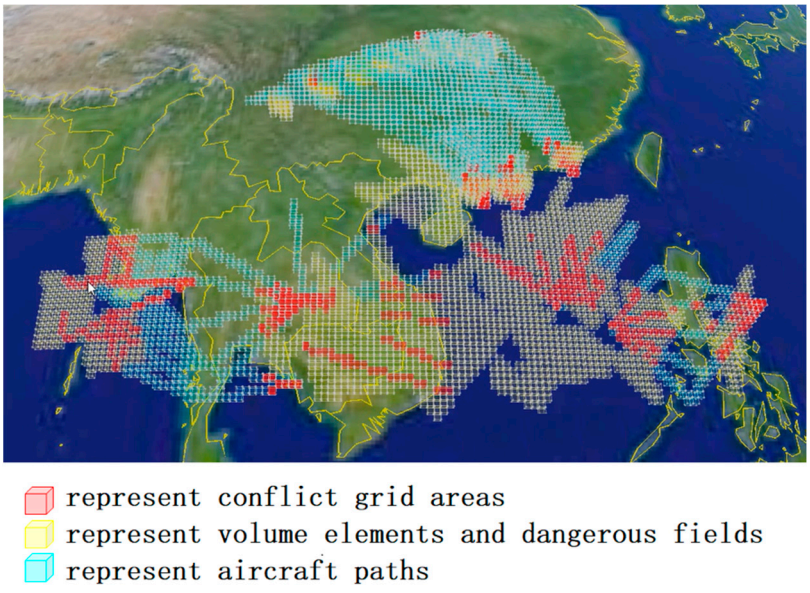


Figure 10. Results of flight conflict detection.

From Figure 11, when the traditional multipath traversing conflict detection algorithm is used, as the number of paths increases, the time needed for path conflict detection increases geometrically; when the spatiotemporal code of the grid is used as the only key, the conflict detection efficiency increases, although the time consumed will continue to increase with the increasing data size.

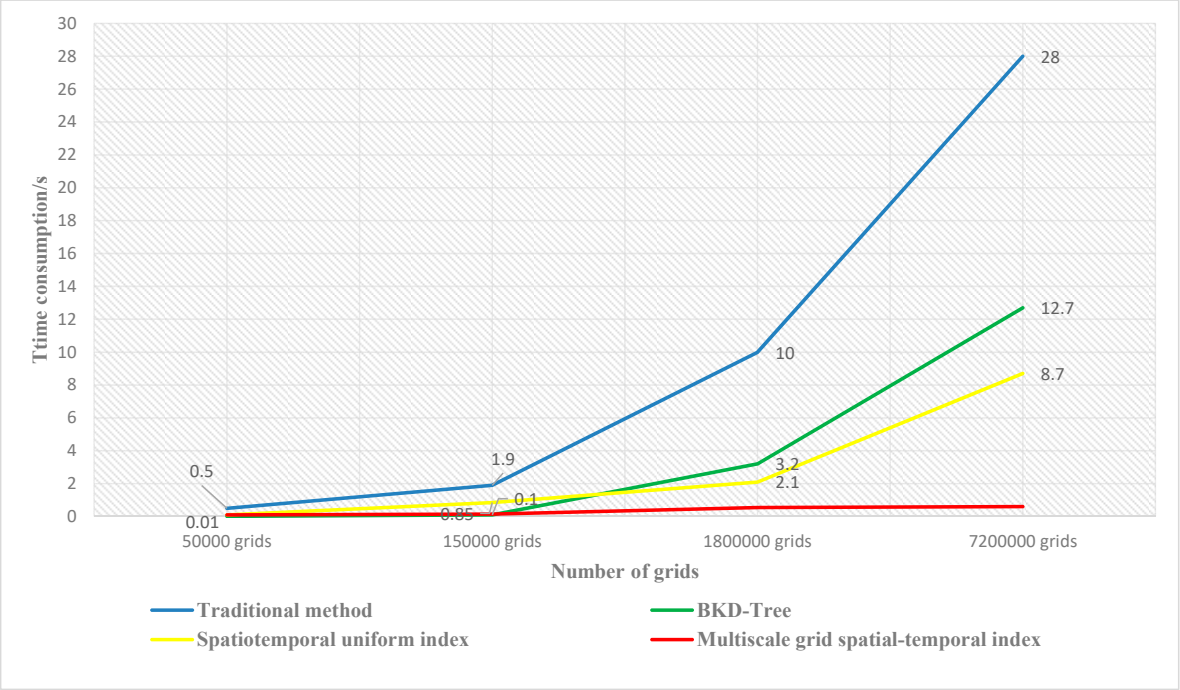


Figure 11. Comparison of flight conflict detection efficiency among different methods.

The grid index has a certain number of child nodes, but the number of child nodes of the Bkd-tree is not fixed. The need for rebalancing Bkd-tree index structures can also be prohibitively expensive than grid index. And the Bkd-tree method has the same tendency as the spatiotemporal uniform index, i.e., the large amount of distance calculation needed is an increasingly time-consuming process as the amount of data grows.

When the path conflict retrieval algorithm, based on a multiscale grid spatiotemporal index is used, as the data size increases, the time consumed for conflict detection virtually stays within 1 second. This result occurs because, as the number of paths increases, the data size traversed by the traditional algorithms will increase geometrically, causing the query time to increase geometrically as well.

If the unified grid spatiotemporal code is used as the only key for indexing, an unequal spatiotemporal distribution of the data will give rise to numerous invalid grid codes, leading to data redundancy (e.g., when encoding paths, discrete encoding to the time of the entire region, including the time when, and after, the aircraft flies past the grid) and consequently compromising the retrieval efficiency.

Our path conflict detection algorithm, based on the multiscale grid spatiotemporal index, treats the time code and space code as two separate keys. It quickly locates the grid codes in the space and time domains by utilizing the neighborhood between the neighboring grid codes and the multilevel regression of the grids and, when the data are unequally distributed in space and time, it automatically optimizes the sequence of the time index and space index. Furthermore, our data partition policy allows database table partitioning in times when the data size increases and utilizes the concurrent query capability of distributed databases. This policy ensures that the time consumed for the conflict detection query will remain basically constant despite the increase in the grid data size.

5. Conclusions and Future Studies

A new, low-altitude trajectory conflict detection algorithm, based on the multilevel grid spatiotemporal index, is presented that transforms the traditional path-by-path traversing computation into a conflict state query of distributed databases. Essentially, it is a method of exchanging "storage space" for "computational time". First, a flight airspace total spatiotemporal subdivision encoding model is built, using the GeoSOT subdivision scheme, to discretize and identify the airspace. Next, a multilevel grid time/space index priority policy is established. Finally, a data partition strategy is used to improve the grid conflict query efficiency for large data sizes. This method not only allows the computation of the conflict detection results between different paths but also supports the detection of path conflicts under complicated low-altitude spatiotemporal variations (high-rises or storm fields). Our method also provides the speed and accuracy required for the real-time (dynamic/static) detection of path conflicts among individual aircraft or aircraft flying in formation, with an efficiency one order of magnitude higher than those of traditional algorithms. In future studies, our algorithm may be extended to support train dispatching, unmanned aerial vehicle (UAV) swarm management, low-altitude flight control, and satellite orbit operations.

Author Contributions: Data curation, W.Z., B.Z., S.L. and J.Z.; Formal analysis, F.R.; Funding acquisition, H.Z.; Investigation, C.C.; Methodology, S.M.

Funding: This research was supported by the National Natural Science Foundation of China (Grant NO. 41801301) and the National Key Research and Development Program of China (Grant Nos. 2018YFB0505301 and 2017YFB0503703).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, Y.; Wan, J. Analysis of development and evolution rules of civil aviation in China based on life cycle theory. *PLoS ONE* **2019**, *14*. [CrossRef] [PubMed]
2. Gariel, M.; Hansman, R.; Frazzoli, E. Impact of GPS and ADS-B Reported Accuracy on Conflict Detection Performance in Dense Traffic. *AIAA J.* **2019**, *57*, 208–222. [CrossRef]
3. Alam, S.; Shafi, K.; Abbass, H.; Barlow, M. An ensemble approach for conflict detection in free flight by data mining. *Transp. Res. Part C* **2009**, *17*, 298–317. [CrossRef]
4. Jilkov, V.P.; Ledet, J.H.; Li, X.R. Multiple Model Method for Aircraft Conflict Detection and Resolution in Intent and Weather Uncertainty. *IEEE Trans. Aerosp. Electron. Syst.* **2019**, *55*, 1004–1020. [CrossRef]
5. Ho, F.; Geraldes, R.; Gonçalves, A.; Cavazza, M.; Prendinger, H. Improved Conflict Detection and Resolution for Service UAVs in Shared Airspace. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1231–1242. [CrossRef]
6. Wan, Y.; Tang, J.; Lao, S. Distributed Conflict-Detection and Resolution Algorithms for Multiple UAVs Based on Key-Node Selection and Strategy Coordination. *IEEE Access* **2019**, *7*, 42846–42858. [CrossRef]
7. Bickerstaff, M.A.; Hellestrand, G.R. A highly parallel architecture for real time collision detection in flight simulation. *Comput. Graph.* **1991**, *15*, 355–363. [CrossRef]
8. Chiang, Y.J.; Klosowski, J.T.; Lee, C.; Mitchel, J.S.B. Geometric Algorithms for Conflict Detection and Resolution in Air Traffic Management. In Proceedings of the 36th IEEE Conference on Decision and Control, San Diego, CA, USA, 12 December 1997; IEEE Press: Piscataway, NJ, USA; pp. 1835–1840.
9. Fulton, N.L. Airspace design: Towards a rigorous specification of conflict complexity based on computational geometry. *Aeronaut. J.* **2016**, *103*, 75–84. [CrossRef]
10. Xing, L.; Songcen, H. Delaunay method for free flight conflict detection. *J. Data Acquis. Process.* **2002**, *17*, 446–519.
11. Zhou, Y.; Suri, S. Analysis of a bounding box heuristic for object intersection. *J. Assoc. Comput. Mach.* **1999**, *46*, 833–857. [CrossRef]
12. Procopiuc, O.; Agarwal, P.; Arge, L.; Vitter, J.S. Bkd-Tree: A Dynamic Scalable kd-Tree. In Proceedings of the International Symposium on Spatial and Temporal Databases, Santorini Island, Greece, 24–27 July 2003; Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany, 2003; Volume 2750, pp. 46–65. [CrossRef]

13. Prandini, M.; Hu, J.; Lygeros, J.; Sastry, S. A probabilistic approach to aircraft conflict detection. *IEEE Trans. Intell. Transp. Syst.* **2000**, *1*, 199–220. [CrossRef]
14. Jardin, M.R. *Grid-Based Strategic Air Traffic Conflict Detection: AIAA-2005-5826*; AIAA: Reston, VA, USA, 2005.
15. Hu, J.; Prandini, M.; Sastry, S. *Aircraft Conflict Detection in Presence of Spatially Correlated Wind Perturbations: AIAA-2003-5339*; AIAA: Reston, VA, USA, 2003.
16. Sahr, K. Central Place Indexing: Hierarchical Linear Indexing Systems for Mixed-Aperture Hexagonal Discrete Global Grid Systems. *Cartogr. Int. J. Geogr. Inf. Geovis.* **2019**, *54*, 16–29. [CrossRef]
17. Jiang, X.R.; Wen, X.X.; Wu, M.; Wang, Z.; Qiu, X. A SVM Approach of Aircraft Conflict Detection in Free Flight. *J. Adv. Transp.* **2018**, *9*, 1–9. [CrossRef]
18. Sahr, K.; White, D.; Kimerling, A.J. Geodesic discrete global grid systems. *Cartogr. Geogr. Inf. Sci.* **2003**, *30*, 121–134. [CrossRef]
19. Zhao, X.S.; Wang, L.; Wang, H.B.; Li, Y. Modeling methods and basic problems of discrete global grids. *Geogr. Geo-Inf. Sci.* **2012**, *28*, 29–34.
20. Cheng, C.; Tong, X.; Chen, B.; Zhai, W. A Subdivision Method to Unify the Existing Latitude and Longitude Grids. *ISPRS Int. J. Geo-Inf.* **2016**, *5*, 161. [CrossRef]
21. Sun, Z.Q.; Cheng, C.Q. 3D integrated representation model and visualization based on the global discrete voxel—GeoSOT3D. In Proceedings of the 23rd International Conference on Geoinformatics, Wuhan, China, 19–21 June 2015; pp. 1–5. [CrossRef]
22. Longley, P. *Geographic Information Systems and Science*; John Wiley & Sons Inc.: West Sussex, UK, 2005.
23. Ben, J. An Effective Search Scheme for Gnutella-Like P2P Networks. In Proceedings of the 2nd International Workshop on Intelligent Systems and Applications, Wuhan, China, 22–23 May 2010; pp. 1–5. [CrossRef]