RETRACTED: Robot Path Planning Method Based on Indoor Spacetime Grid Model

Huangchuang Zhang $^{1,2}$, Qingjun Zhuang $^3$ and Ge Li $^{1,*}$

$^1$ School of Electronic and Computer Engineering, Peking University, Shenzhen 518055, China; zhanghuangchuang@stu.pku.edu.cn
$^2$ Pengcheng Laboratory, Shenzhen 518055, China
$^3$ Academy for Advanced Interdisciplinary Studies, Peking University, Beijing 100871, China; 1801214017@pku.edu.cn or zhuangqingjun0410@126.com

* Correspondence: geli@ece.pku.edu.cn; Tel.: +86-139-1015-0832

Abstract: In the context of digital twins, smart city construction and artificial intelligence technology are developing rapidly, and more and more mobile robots are performing tasks in complex and time-varying indoor environments, making, at present, the unification of modeling, dynamic expression, visualization of operation, and wide application between robots and indoor environments a pressing problem to be solved. This paper presents an in-depth study on this issue and summarizes three major types of methods: geometric modeling, topological modeling, and raster modeling, and points out the advantages and disadvantages of these three types of methods. Therefore, in view of the current pain points of robots and complex time-varying indoor environments, this paper proposes an indoor spacetime grid model based on the three-dimensional division framework of the Earth space and innovatively integrates time division on the basis of space division. On the basis of the model, a dynamic path planning algorithm for the robot in the complex time-varying indoor environment is designed, that is, the Spacetime-A* algorithm (STA* for short). Finally, the indoor spacetime grid modeling experiment is carried out with real data, which verifies the feasibility and correctness of the spacetime relationship calculation algorithm encoded by the indoor spacetime grid model. Then, experiments are carried out on the multi-group path planning algorithms of the robot under the spacetime grid, and the feasibility of the STA* algorithm under the indoor spacetime grid and the superiority of the spacetime grid are verified.

Keywords: spacetime grid; indoor complex environment; grid model; mobile robot; path planning

1. Introduction

In the context of digital twins [1–3], smart city [4–6] construction and artificial intelligence technologies have developed rapidly, and more and more mobile robots are performing tasks in complex and time-varying indoor environments. For example, service robots provide information services in shopping malls, food delivery robots provide commodity distribution in hotel restaurants, and rescue robots provide indoor emergency response. In these indoor application scenarios, although there are great differences in research under different application tasks, they are all inseparable from indoor environment modeling. Only by modeling the indoor environment can the robot carry out further applications (such as path planning). However, indoor environments are typically complex environments [7,8]. Compared with the open outdoor environment, the indoor environment has a narrow space area, fewer passable areas, and a large number of irregular obstacles, as well as other multi-source heterogeneous data, such as robot path data. In addition, the indoor environment has very strong time-varying characteristics [9]. Obstacles in the outdoor environment may rarely change for a long time, but the traffic state changes in different areas in the indoor space are more frequent, such as the opening and closing of indoor doors and windows. How to make mobile robots perform tasks better in complex indoor environments remains a pressing problem to be solved.
environments with time-varying characteristics is of great practical significance. In other words, it is to better solve the problems of unification of modeling, dynamic expression, visualization of operation, and wide application between robots and indoor environments.

This paper conducts in-depth research on this, summarizes the current three types of methods: geometric model method, topological model method, and grid model method, and points out the advantages and disadvantages of these three types of methods. The geometric model method [10–20] is a very simple and intuitive modeling method. By introducing the equation of time motion of obstacles [10,11,14,15,20], time-varying obstacles with regular motion can be expressed. However, it is not suitable for modeling environments with dense and complex obstacles, and the time-varying characteristics of most obstacles are difficult to express through time equations. The data structure of the topological graph of the topology model method [21–27] is very suitable for subsequent algorithm applications, such as path planning, and has small storage space and high computational efficiency. It is very difficult, however, to create a topology map in a complex environment, and it requires too much manual intervention. In addition, the ability of a time-varying topology graph to express the time-varying environment is too limited, and the high degree of abstraction also leads to the lack of information; only the information of topological edges and points can be obtained. The grid model method [28–32] is a simple and effective modeling method, which can deal with the modeling of time-varying complex and irregular obstacles. However, different mobile robots often need to build local grid maps according to their own scales, and there is a lack of unified construction standards, making it difficult to share data under the grid modeling method between robots.

Through the analysis of the above research and applications, we found that a new method is urgently needed to solve the problems of unification of modeling, dynamic of expression, visualization of operation, and wide application between robots and the indoor environment. Therefore, in view of the current pain points of robots and complex time-varying indoor environments, this paper proposes an indoor spacetime grid model based on the three-dimensional division framework of the Earth space and innovatively integrates time division on the basis of space division. On the basis of the model, a dynamic path planning algorithm for the robot in the complex time-varying indoor environment is designed, that is, the Spacetime-A* algorithm (STA* for short). Finally, the indoor spacetime grid modeling experiment is carried out with real data, which verifies the feasibility and correctness of the spacetime relationship calculation algorithm encoded by the indoor spacetime grid model. Then, experiments are carried out on the multi-group path planning algorithms of the robot under the spacetime grid, and the feasibility of the STA* algorithm under the indoor spacetime grid and the superiority of the spacetime grid are verified.

The essential difference between the method in this paper and the previous method is the method proposed in this paper relies on the three-dimensional division of the Earth space, starts from a global perspective, and then shrinks to a single indoor building. In this way, it can be seamlessly connected with the existing mature technical standards, which is conducive to the unification of modeling, and naturally has the characteristics of wide application. Moreover, due to the characteristics of the grid, in terms of operation visualization, it is only necessary to use the Cesium platform [33] to establish a real-world 3D visualization Earth platform. In addition, on the basis of global space division, this paper adopts an innovative fusion time division method for the complex time-varying characteristics of indoor space, which solves the expression dynamic. In terms of expansion, this method has certain reference and significance for digital twin cities to build an indoor and outdoor integrated navigation service platform.

The innovation of the method in this paper is that the method proposed in this paper integrates the advantages of identifiable, locatable, indexable, computable, and automatic spatial association of multi-level spacetime division of Earth mesh and time division. The modeling of the indoor complex time-varying environment is realized by using a multi-layered spacetime grid. In one scenario, the unified association and multi-level refined management of indoor multi-heterogeneous data are realized from
the data management level. In the second scenario, from the algorithm design level, the dynamic path planning algorithm for the mobile robot in the indoor complex time-varying environment is implemented.

The structure of the paper is mainly as follows: the first section is the general introduction of the paper. The second section is the focus of the article, introducing the indoor spacetime model framework, spacetime grid computing, and STA* algorithm design in detail. The third section is the experiment and results analysis. The fourth section and fifth section summarize the experimental results and provide an outlook for future research.

2. Materials and Methods

2.1. Indoor Spacetime Grid Model

The indoor spacetime grid model consists of two parts, the space division and the time part, which will be introduced in detail below.

2.1.1. Indoor Space Division Framework

The spatial part is built on the basis of GeoSOT-3D (Geographic Coordinate Sub-dividing Grid with One Dimension Integral Coding on 2^n-Tree-3D) [34–39], relying on the framework of geospatial stereo dissection. GeoSOT-3D takes the geodetic height of the CGCS2000 (China Geodetic Coordinate System 2000) geodetic coordinate system as the elevation direction. Through three Earth expansions, the Earth space is expanded to $512° \times 512° \times 512°$, $1°$ is expanded to $64′$, and $1′$ is expanded to $64″$, realizing the recursive octree of the whole degree, minute, and second [40], as shown in Figure 1. The GeoSOT-3D solid grid is divided into 32 levels [41]. The 0-level grid corresponds to the entire three-dimensional space of Earth, and the space height is $0°$ from the center of Earth and goes up to $512°$. As the level increases, the space is continuously divided into 8 equal parts, and, finally, the space up to 50,000 km and the three-dimensional space down to the center of Earth are divided into three-dimensional grids of different granularities [42,43].

![Figure 1. Earth meshing framework. (a) GeoSOT meshing framework [40]; (b) GeoSOT-3D meshing framework.](image)

The indoor space division takes the grid center of a GeoSOT-3D grid in the room as the origin of the indoor local coordinate system. At the same time, the latitude direction and the longitude direction of the center point are taken as the $x$-axis direction and the $y$-axis direction, respectively. The $z$-axis of the coordinate system adopts the CGCS2000 geodetic elevation system and is consistent with the GeoSOT-3D framework. The subdivision idea is similar to GeoSOT-3D, which is essentially an octree subdivision of the grid. The subdivision range only considers the indoor subdivision of a single building. At present, the largest single building in the world is the New Century Global Center in Chengdu,
Sichuan, China. This building is 500 m × 400 m × 300 m in length. Further, the tallest building in the world is the Burj Khalifa at 828 m. In order to cover the indoor space of any single building, the indoor local 0-level grid adopts 2048 m × 2048 m × 2048 m, and the coverage space of the 0-level grid can basically meet the indoor space modeling requirements of all single buildings on Earth. The following is the subdivision method of the indoor mesh at each level.

The 0-level is the largest-scale subdivision grid of the indoor space, and the theoretical grid size is 2048 m × 2048 m × 2048 m. In a real indoor environment, this three-dimensional grid contains the entire building unit. The 1-level is to divide the indoor three-dimensional grid into 8 three-dimensional areas. Theoretically, the size of each subdivision voxel is 1024 m × 1024 m × 1024 m. Since the actual indoor environment does not necessarily have such a large space in terms of length, width, and height, during indoor segmentation, if a grid does not exist in the indoor space, the false indoor local grid will not be divided. The N-level is based on the (N-1)-level subdivision for octagonal division. From a single dimension (x-axis, y-axis, z-axis), the size of each dimension is 1/2 of the previous level. The maximum scale level of indoor division is 18 levels, and the grid size is 7.8125 mm × 7.8125 mm × 7.8125 mm, which is a three-dimensional grid of about 8 mm. Through the indoor local grid level of 0–18, the seamless and non-overlapping local space grid of indoor space up to 2048 m and down to 8 mm is realized. Table 1 shows the comparison table of the level and grid size of indoor space grid division.

Table 1. Indoor space section hierarchy table.

<table>
<thead>
<tr>
<th>Level</th>
<th>Grid Size (m)</th>
<th>Level</th>
<th>Grid Size (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>2048</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1024</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>12</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>13</td>
<td>1/4</td>
</tr>
<tr>
<td>4</td>
<td>128</td>
<td>14</td>
<td>1/8</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>15</td>
<td>1/16</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>16</td>
<td>1/32</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>17</td>
<td>1/64</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>18</td>
<td>1/128</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indoor space grid coding method: referring to the coding method of GeoSOT-3D, but different from GeoSOT-3D, there is no quadrant code subdivision for indoor local grid subdivision. For indoor local meshing, the three-dimensional Z-sequence of each mesh is the same, according to the x-axis from small to large, the y-axis from small to large, and the z-axis from small to large. The grid code for level 0 is B, and the coding for subsequent levels uses digital records. At the same time, the coding forms used in the coding of the local space grid include octal 1-dimensional, binary 1-dimensional, binary 3-dimensional, and other coding forms, and different forms of coding have different effects in different situations.

2.1.2. Indoor Time Division Framework

The indoor time division model uses time-scale intervals up to 1 year and down to 1 s in the time domain setting. It is divided into 26 levels in total, and the data above the year level are stored in the form of data disk. Due to the difficulty brought by the original month unit to the time relationship calculation, the local time model eliminates the month unit for the convenience of time relationship calculation. Therefore, there are only five scales of day, hour, minute, and second in the indoor time division model. The division idea is: the virtual expansion of the year is to expand the year to 512 days, 1 day to 32 h, 1 h to 64 min, and 1 min to 64 s, and, in addition, consider the smooth transition of time data between adjacent years, especially the 512 days after expansion, including 64 days at the
end of the previous year and 64 days at the beginning of the next year. Taking a normal year as an example, the 1–64 days of the 512 days are the last 64 days of the previous year, the 65–429 days are the 365 days of the year, the 430–493 days are the first 64 days of the following year, and the remaining number of days is considered a virtual extension. In this way, the time data can be more effectively smoothed in the year-to-year data transition.

Indoor time division also uses binary division to divide time. The following is the division process: the 0-level indoor time division represents 512 days, but, in actual use, only the 64 days at the end of the previous year, 365 days of all days in this year, and 64 days at the beginning of the next year are considered, for a total of 493 days (494 days in leap years). The remaining days are considered virtual extensions. The 1-level indoor time division is to divide the 0-level time domain into two 256-days. In actual use, part 1, the first-level time domain, includes the data of the last 64 days of the previous year and the data of the first 192 days of the current year, a total of 256 days. Part 2, the 1-level time domain, includes 173 days of the current year (174 days in leap years) plus 64 days at the beginning of the following year, a total of 237 days (238 days in leap years), and the rest of the virtual expansion time is not considered. The 2-level indoor time division is to divide the 1-level time domain into two parts of 128 days. In actual use, the size of the real time domain needs to consider the virtual expansion and the average leap year. Similarly, the indoor time division of the N-level is to divide the time domain of the (N-1)-level into two equal parts. The relationship between the specific section level and the corresponding time domain is shown in Table 2.

### Table 2. Indoor time section hierarchy table.

<table>
<thead>
<tr>
<th>Level</th>
<th>Time Domain Size</th>
<th>Level</th>
<th>Time Domain Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>256 days</td>
<td>14</td>
<td>1 h (60 min)</td>
</tr>
<tr>
<td>2</td>
<td>128 days</td>
<td>15</td>
<td>32 min</td>
</tr>
<tr>
<td>3</td>
<td>64 days</td>
<td>16</td>
<td>16 min</td>
</tr>
<tr>
<td>4</td>
<td>32 days</td>
<td>17</td>
<td>8 min</td>
</tr>
<tr>
<td>5</td>
<td>16 days</td>
<td>18</td>
<td>4 min</td>
</tr>
<tr>
<td>6</td>
<td>8 days</td>
<td>19</td>
<td>2 min</td>
</tr>
<tr>
<td>7</td>
<td>4 days</td>
<td>20</td>
<td>1 min (60 s)</td>
</tr>
<tr>
<td>8</td>
<td>2 days</td>
<td>21</td>
<td>32 s</td>
</tr>
<tr>
<td>9</td>
<td>1 d (24 h)</td>
<td>22</td>
<td>16 s</td>
</tr>
<tr>
<td>10</td>
<td>16 h</td>
<td>23</td>
<td>8 s</td>
</tr>
<tr>
<td>11</td>
<td>8 h</td>
<td>24</td>
<td>4 s</td>
</tr>
<tr>
<td>12</td>
<td>4 h</td>
<td>25</td>
<td>2 s</td>
</tr>
<tr>
<td>13</td>
<td>2 h</td>
<td>26</td>
<td>1 s</td>
</tr>
</tbody>
</table>

#### 2.2. Spacetime Computation of Indoor Spacetime Grid

The coding corresponding to the indoor spacetime grid implies spatial location information, temporal region information, and scale information. This implicit relationship enables the spacetime encoding to be directly used for fast computation of the spacetime relationship. This section discusses from two perspectives of space and time. Finally, the spacetime combined neighborhood calculation method is introduced.

##### 2.2.1. Space Computation of Indoor Spacetime Grid

Indoor spacetime grid coding is composed of GeoSOT-3D coding, indoor spatial coding, and local time coding. Among these three codes, the main function of GeoSOT-3D coding is to play a role in global positioning so that indoor space division has a global reference, and the spatial calculation of indoor grid is mainly based on indoor space coding. Next, the spatial grid displacement method, spatial neighborhood calculation method, spatial parent grid coding calculation method, subgrid coding calculation method, and spatial distance calculation method based on indoor spatial subdivision coding are introduced, respectively.
• The spatial grid displacement method

The spatial grid displacement based on the indoor spatial subdivision coding refers to the displacement of several grid cells in the corresponding grid voxels of the indoor local spatial subdivision coding at the level level. The spatial grid displacement method is the basic algorithm of the spatial grid neighborhood calculation method because, for any spatial grid neighborhood, it can be obtained by the displacement of the spatial grid, in essence. Suppose there is an indoor space subdivision coding code at the level level, and convert it into a binary three-dimensional shaping coding \[ \text{Code}_{X}, \text{Code}_{Y}, \text{Code}_{Z} \], which represents the binary shaping coding of the x-axis, y-axis, and z-axis, respectively.

The formula for the displacement operation on the x-axis is as follows:

A grid displacement of one element in the positive x-axis occurs:
\[
\text{Code}_{\text{new}X} = \text{Code}_{X} + 2^{18-\text{level}}
\]  

(1)

A grid displacement of one element in the negative x-axis occurs:
\[
\text{Code}_{\text{new}X} = \text{Code}_{X} - 2^{18-\text{level}}
\]  

(2)

A grid displacement of one element in the positive y-axis occurs:
\[
\text{Code}_{\text{new}Y} = \text{Code}_{Y} + 2^{18-\text{level}}
\]  

(3)

A grid displacement of one element in the negative y-axis occurs:
\[
\text{Code}_{\text{new}Y} = \text{Code}_{Y} - 2^{18-\text{level}}
\]  

(4)

A grid displacement of one element in the positive z-axis occurs:
\[
\text{Code}_{\text{new}Z} = \text{Code}_{Z} + 2^{18-\text{level}}
\]  

(5)

A grid displacement of one element in the negative z-axis occurs:
\[
\text{Code}_{\text{new}Z} = \text{Code}_{Z} - 2^{18-\text{level}}
\]  

(6)

It should be noted that Formulas (1)–(6) needs to meet the following conditions:
\[
0 \leq \text{Code}_{\text{new}X} \leq 2^{18} - 1, \ 0 \leq \text{Code}_{\text{new}Y} \leq 2^{18} - 1, \ 0 \leq \text{Code}_{\text{new}Z} \leq 2^{18} - 1
\]  

(7)

There is one key thing to note. When performing positive displacement or negative displacement on the grid, it is necessary to judge whether the displaced grid is a real indoor grid because the displaced grid may have exceeded the real indoor space. Then, the grid obtained by displacement at this time has no real geographical significance. In practical application, in order to avoid the occurrence of this situation, a hash table can be used to record the code of the real indoor subdivision grid, or the real indoor subdivision grid can be stored as the key of the data table. In this way, after grid displacement occurs, it is possible to query whether the corresponding grid code is a real geographic grid.

• The spatial neighborhood calculation method

The spatial neighborhood calculation method based on indoor spatial subdivision coding refers to obtaining the codes corresponding to the grid voxels with adjacent relationship through spatial coding. The calculation of spatial neighborhood is important in many algorithms, such as path planning, breadth traversal, depth traversal, etc. For the adjacency of indoor subdivision voxels, it can be divided into three types. The first type is that the subdivision voxels are adjacent to each other: the two subdivision voxels are only displaced by a grid distance in one of the dimensions of the x-axis, the y-axis, and the z-axis, and there are 6 possibilities in total. The second type is that the edge of the subdivision voxel is adjacent: the two subdivision voxels are displaced by a grid distance in the two
dimensions of the x-axis, the y-axis, and the z-axis, and there are 12 possibilities in total. The third type is that the divided voxels are angularly adjacent: the two divided voxels are separated by a grid distance in the three dimensions of the x-axis, the y-axis, and the z-axis, and there are 8 possibilities in total. The three types are shown in Figure 2.

![Diagram of indoor space grid adjacent]  
**Figure 2.** Indoor space grid adjacent diagram. (a) Face adjacent; (b) side adjacent; (c) point adjacent.

The neighborhood space computation of trellis coding needs to be based on binary three-dimensional shaping coding. For the grid coding code at the level subdivision scale, it is necessary to convert the code into a binary three-dimensional shaping coding [\(\text{Code}_x, \text{Code}_y, \text{Code}_z\)]. The essence of the neighborhood calculation is the combination of 0 displacement or 1 grid displacement in each dimension of the grid code (the neighborhood does not contain its own grid). According to the different computing requirements of the neighborhood, it can be mainly divided into four kinds of neighborhoods: plane four neighborhoods, plane eight neighborhoods, stereo six neighborhoods, and stereo twenty-six neighborhoods, as shown in Figure 3. In addition to the above four kinds of neighborhoods, other neighborhood forms can also be obtained by the displacement calculation of the grid. The requirements for neighborhood calculation under different tasks are also different, and the specific neighborhood calculation method to be used needs to be considered in combination with the actual problem.

![Diagram of four kinds of neighborhood]  
**Figure 3.** Four kinds of neighborhood diagrams. (a) Plane four neighborhoods; (b) plane eight neighborhoods; (c) stereo six neighborhoods; (d) stereo twenty-six neighborhoods.
Similar to the grid displacement calculation, the neighborhood calculated by the neighborhood grid may also contain grids that are not geographically meaningful. Similarly, the same approach can be taken to deal with this problem.

- **The spatial parent grid and subgrid coding calculation method**

In the principle of the indoor space division framework, there is inheritance between the subgrid and the parent grid, which is well-reflected in the coding of the two. The parent grid code calculation for any grid code can be calculated by taking the grid code prefix. Assuming that $\text{Code}$ is an octal one-dimensional code under the level section level, its parent grid code is the first $\text{level} - 1$ bits of $\text{Code}$. For example, for the calculation of the parent grid of the octal one-dimensional grid $B001111$, the grid is a 6-level grid, and taking the first 5 digits of the grid will obtain $B0011$, which is the parent grid of $B001111$.

In the same way, by using the idea of taking prefix, it can be determined whether there is an ancestor relationship between two grids at different levels. In addition, the parent grid operation of codes in other hexadecimal forms can be obtained in the same way, which will not be described here.

Similarly, when the parent grid code is obtained, the child grid can be obtained by inheritance calculation. Because the subgrid is essentially the octagonal division of the parent grid, for any parent grid code, the calculation process of the subgrid code as shown in Algorithm 1.

**Algorithm 1: Subgrid Coding Algorithm.**

**Require:** Input the octal one-dimensional grid code $\text{Code}$

**Ensure:** the set $R$ of subgrid codes for this grid

1. Set $R$ to be an empty array, $l \leftarrow 0$
2. while $l \leq 7$ do
   3. $\text{Code}_{\text{new}} \leftarrow \text{stract}(\text{Code}, l)$
   4. Add $\text{Code}_{\text{new}}$ to $R$
   5. $l \leftarrow l + 1$
3. end while
6. Return the subgrid coding set $R$

For example, for the subgrid calculation of the octal one-dimensional grid $B0011$, the grid is a 6-level grid, and the subgrids of the grid can be obtained through the above algorithm: $B00110$, $B00111$, $B00112$, $B00113$, $B00114$, $B00115$, $B00116$, $B00117$.

- **The spatial distance calculation method**

For any two indoor grid codes under the same subdivision level, define the grid space distance between grids in a certain dimension as the distance between the two grids in this dimension. When calculating grid distance, it is essentially a displacement operation of the grid. Assuming that there are two grid codes $\text{Code}1$, $\text{Code}2$ at the level, the corresponding binary three-dimensional grid codes are $[\text{Code}1_x, \text{Code}1_y, \text{Code}1_z]$, $[\text{Code}2_x, \text{Code}2_y, \text{Code}2_z]$; then, these two codes are on the $x$-axis, $y$-axis, and $z$-axis. The grid distance calculation formulas are as follows:

$$
\begin{align*}
\text{dis}(\text{Code}1_x, \text{Code}2_x) &= |\text{Code}1_x - \text{Code}2_x| \gg (18 - \text{level}) \\
\text{dis}(\text{Code}1_y, \text{Code}2_y) &= |\text{Code}1_y - \text{Code}2_y| \gg (18 - \text{level}) \\
\text{dis}(\text{Code}1_z, \text{Code}2_z) &= |\text{Code}1_z - \text{Code}2_z| \gg (18 - \text{level})
\end{align*}
$$

(8)

Since the grid size of the same level of indoor meshing is the same, for any two indoor grids, the real space distance (m) between the two grids can be calculated. The spatial
distance between two grids is defined here as the distance between the centers of the two grids, and the formula for calculating the real spatial distance is as follows:

\[
\begin{align*}
\text{dis}(\text{Code}_{X1}, \text{Code}_{X2}) &= \left( |\text{Code}_{X1} - \text{Code}_{X2}| \gg (18 - \text{level}) \right) \times \frac{2048}{2048} \\
\text{dis}(\text{Code}_{Y1}, \text{Code}_{Y2}) &= \left( |\text{Code}_{Y1} - \text{Code}_{Y2}| \gg (18 - \text{level}) \right) \times \frac{2048}{2048} \\
\text{dis}(\text{Code}_{Z1}, \text{Code}_{Z2}) &= \left( |\text{Code}_{Z1} - \text{Code}_{Z2}| \gg (18 - \text{level}) \right) \times \frac{2048}{2048}
\end{align*}
\]  

(9)

2.2.2. Time Computation of Indoor Spacetime Grid

This section mainly discusses computations in the time dimension based on indoor spacetime trellis coding. It mainly includes the displacement operation method of time coding and the neighborhood query method of time coding.

- The displacement operation method of time coding

Regarding the time coding under level, the time unit displacement under the time coding is defined as the movement of the same time scale unit under level. Algorithm 2 and Algorithm 3, respectively, provide the calculation method of positive unit time displacement and negative unit time displacement calculation method of 32-bit integer time code.

\begin{algorithm}
\caption{Algorithm of positive unit time displacement.}
\begin{algorithmic}
\Require Decimal integer time encoding Code_{time}
\Ensure The time code corresponding to the next adjacent time domain of the time code Code_{new}
\State $\text{level} \leftarrow \text{Code}_{time} \gg 27$, isleap $\leftarrow (\text{Code}_{time} \gg 26) \bmod 2$, Code $\leftarrow (\text{Code}_{time} \bmod 2^{56}) + (1 \ll (26 - \text{level}))$
\State $\text{secs} \leftarrow \text{Code} \bmod 60$, Code $\leftarrow (\text{Code} \gg 6) + \text{int}(\frac{\text{Code} \mod 2^6}{60})$
\State $\text{mins} \leftarrow \text{Code} \bmod 60$, Code $\leftarrow (\text{Code} \gg 6) + \text{int}(\frac{\text{Code} \mod 2^6}{24})$
\State $\text{hours} \leftarrow \text{Code} \bmod 24$, Code $\leftarrow (\text{Code} \gg 5) + \text{int}(\frac{\text{Code} \mod 2^5}{24})$
\State $\text{days} \leftarrow \text{Code} \bmod (\text{isleap} + 365)$
\If {$\text{Code} \geq (\text{isleap} + 365 + 128)$}
\State The time code is outside its own time coverage, \textbf{return} NULL
\EndIf
\State $\text{Code}_{new} = (\text{levle} \ll 27) + (\text{isleap} \ll 26) + (\text{days} \ll 17) + (\text{hours} \ll 12) + (\text{mins} \ll 6) + \text{secs}$
\State \textbf{return} $\text{Code}_{new}$
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\caption{Algorithm of negative unit time displacement.}
\begin{algorithmic}
\Require Decimal integer time encoding Code_{time}
\Ensure The time code corresponding to the last adjacent time domain of the time code Code_{new}
\State $\text{level} \leftarrow \text{Code}_{time} \gg 27$, isleap $\leftarrow (\text{Code}_{time} \gg 26) \bmod 2$, Code $\leftarrow (\text{Code}_{time} \bmod 2^{56}) + (1 \ll (26 - \text{level}))$
\If {$\text{Code} < 0$}
\State The time code is outside its own time coverage, \textbf{return} NULL
\EndIf
\If {$\text{Code} \bmod 2^6 \geq 60$}
\State $\text{secs} \leftarrow 59$
\EndIf
\If {$\text{Code} \bmod 2^6 \geq 6$}
\State $\text{Code} \leftarrow \text{Code} \gg 6$
\EndIf
\If {$\text{Code} \bmod 2^6 \geq 24$}
\State $\text{hours} \leftarrow 23$
\EndIf
\State $\text{days} \leftarrow \text{Code} \bmod (\text{isleap} + 365)$
\State $\text{Code}_{new} = (\text{levle} \ll 27) + (\text{isleap} \ll 26) + (\text{days} \ll 17) + (\text{hours} \ll 12) + (\text{mins} \ll 6) + \text{secs}$
\State \textbf{return} $\text{Code}_{new}$
\end{algorithmic}
\end{algorithm}
The neighborhood query method of time coding

The time neighborhood of the time code refers to the next time domain and the previous time domain under the corresponding level of the time code. The essence of the neighborhood calculation of time code is to obtain the relevant time code through the displacement operation of time code. The calculation method of unit time neighborhood as shown in Algorithm 4.

**Algorithm 4:** The calculation method of unit time neighborhood.

```
Require: Input the time code Code at the level
Ensure: the set of neighborhood codes R for this time code
1: Perform a positive displacement of Code by one unit and obtain the time code Code+, add Code+ to the R set
2: Perform a positive displacement of Code by one unit and obtain the time code Code−, add Code− to the R set
3: Return time-encoded set R
```

In certain scenarios, it may be necessary to obtain a temporally encoded multi-unit neighborhood. At this time, the corresponding neighborhood set can be obtained by several times of positive unit time domain displacement or negative unit time domain displacement. For example, if the first 3 time domains and the last 3 time domains are required at a certain time point, 3 positive unit displacements and 3 negative unit displacements can be performed through time coding to obtain the time codes corresponding to these 6 neighborhood times.

2.2.3. Spacetime Computation of Indoor Spacetime Grid

The spacetime neighborhood of indoor spacetime grid is essentially the combination of temporal neighborhood and spatial neighborhood, which is a kind of neighborhood calculation in four-dimensional space. It includes the neighborhood in space and the neighborhood in time. Figure 4 is a schematic diagram of the spacetime neighborhood. The neighborhood calculation method of XY plane spacetime eight neighborhoods based on indoor spacetime grid is given Algorithm 5. The calculation of the rest of the spacetime neighborhood is the same and will not be described here.

![Figure 4. Schematic diagram of spacetime neighborhood.](image-url)
Algorithm 5: Algorithm of XY plane spacetime eight neighborhoods.

Require: Input the binary three-dimensional code \([\text{Code}_X, \text{Code}_Y, \text{Code}_Z]\) corresponding to the divided voxels in the indoor spacetime grid, and the corresponding time code \(T\text{Code}\)

Ensure: the XY plane spacetime twenty-six neighborhood grid set \(R\) of this subdivision volume

1: set \(R\) to be an empty array
2: Calculate the codes obtained by moving \(\text{Code}_X\) by one unit in the positive and negative directions on the \(x\)-axis as \(\text{Code}_X^+, \text{Code}_X^-\), set \(A = \{\text{Code}_X^+, \text{Code}_X, \text{Code}_X^-\}\)
3: Calculate the codes obtained by moving \(\text{Code}_X\) by one unit in the positive and negative directions on the \(y\)-axis as \(\text{Code}_Y^+, \text{Code}_Y^-\), set \(B = \{\text{Code}_Y^+, \text{Code}_Y, \text{Code}_Y^-\}\)
4: set \(C = \{\text{Code}_Z\}\)
5: Calculate the codes obtained by moving \(T\text{Code}\) by one unit in the positive and negative directions on the \(y\)-axis as \(T\text{Code}_Y^+, T\text{Code}_Y^-\), set \(D = \{T\text{Code}_Y^+, T\text{Code}_Y, T\text{Code}_Y^-\}\)
6: Take one code from each of \(A, B, C,\) and \(D\) sets, a total of 27 cases. Remove one of the codes with no displacement in the \(x\)-axis, \(y\)-axis, \(z\)-axis, and time-axis, and add the remaining 26 codes to \(R\)
7: Return the XY plane spacetime twenty-six neighborhood grid set \(R\)

2.3. Path Planning of Mobile Robots under Spacetime Grid
2.3.1. Mainstream Path Planning Algorithms

In complex indoor environments, whether a reasonable planning path can be found directly determines the efficiency and safety of mobile robots when performing tasks. Usually, an excellent path planning algorithm not only needs to be able to give full play to the advantages of the environment modeling method and perfectly fit with the space model but also needs to meet the conditional constraints of the real task (including the physical performance constraints of the robot itself, the external environment constraints, the task goal constraints, etc.). The comparison of the current mainstream path planning algorithms is shown in Table 3.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Whole/Part</th>
<th>Efficiency</th>
<th>Planning Ability</th>
<th>Complexity</th>
<th>Path Smoothness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Potential Field Algorithm</td>
<td>Part</td>
<td>High</td>
<td>Weak</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Neural Network Algorithm</td>
<td>Part</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Genetic Algorithm</td>
<td>Whole</td>
<td>Low</td>
<td>Strong</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Ant Colony Algorithm</td>
<td>Whole</td>
<td>Medium</td>
<td>Strong</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>A* algorithm</td>
<td>Whole</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
</tbody>
</table>

In terms of efficiency, the artificial potential field method and the A* algorithm are the most efficient. In actual path planning, real-time planning of the robot must be achieved, so efficiency is a very important reference. However, from the perspective of planning ability, the planning ability of artificial potential field method is insufficient. In comparison, the planning ability of the A* algorithm is acceptable. Although both the ant colony algorithm and the genetic algorithm have strong planning capabilities, their efficiency is not high and cannot meet the needs of real-time path planning, and the complexity of the genetic algorithm is still very high. The optimization ability of neural network algorithm is general, but the efficiency is too low and the complexity is too high, so it is not suitable for the path planning in this paper. By comparing these methods, and considering the modeling method in this paper and the requirements for real-time and accuracy of path planning, the A* heuristic search algorithm is finally selected as the basic algorithm of path planning used in this paper. For the principle of the A* algorithm, please refer to References [49–53]. This paper does not describe too much.
2.3.2. Spacetime-A* Algorithm

The original A* algorithm has certain defects in dealing with time-varying environments, and the travel time of time-varying grids (such as doors) is inconsistent. For example, the closing time of the grid corresponding to door A is 16:00–20:00. If the planning is at 15:59, according to the traditional planning method, the mobile robot will think A is passable and incorporate it into the planning category. However, if the spacetime code is also considered as a variable in the planning and navigation of the mobile robot, then the robot will have a prediction in advance, and it is very likely it will arrive at the grid of gate A at 16:03, which is inaccessible at this time. At this time, by predicting in advance, the robot can plan a more reasonable path. In addition to the time-varying characteristics of some static obstacles, other mobile robots in the same space and time also have time-varying characteristics. For example, when robot A performs the corresponding path planning task indoors, it will also occupy the present and future time and space accordingly. At this time, if robot B also wants to perform path planning, it needs to take into account the occupation of robot A on the spacetime grid. The comparison between ordinary planning and time-varying planning is shown in Figure 5.

![Figure 5. The comparison between ordinary planning and time-varying planning. (a) Spacetime latitude is not considered; (b) consider spacetime latitude.](image)

In order to solve the possible impassability problem in the time dimension, this paper introduces the moving speed \( v \) of the mobile robot on the basis of the original A* algorithm. The velocity \( v \) can provide the robot’s occupied time slice for each spacetime grid during path planning. By performing spacetime coding on the time slice, the time dimension is introduced into the original spatial path planning to ensure the planned path can satisfy the environment with time-varying characteristics. The A* algorithm based on the spacetime grid is shown in Algorithm 6.

After determining the spacetime grid code set of the planned path obtained by Spacetime-A* planning, the data of the robot path planning need to be updated in the indoor spacetime data table immediately. The planning situation of the robot is stored in the large indoor spacetime table. If there is a collision and conflict in the large spacetime data table at a later time and space (for example, a certain spacetime grid of the original planned path is suddenly blocked, or the multi-robot path planning passes through the same grid at the same time), a collision warning can be issued immediately. At this time, the spatiotemporal data table will query those robots that have not yet arrived, brake them in time, and re-plan a reasonable route.

Step1: Input start point, Start, end point, E, and speed, v, and start time, time. The starting point and starting time are encoded into the starting spacetime code, S, and its cost value into a list called “Open-List”, and the cost value of the starting spacetime code is 0 by default. The “Open-List” list stores the set of spatiotemporal codes currently waiting to be searched, and a “Close-List” list is created to store the searched spatiotemporal codes.

Step2: When the Open-List is not empty, go to (1); otherwise, go to step3:
1. Take the spacetime code with the least cost in the Open-List as the current spacetime code N. Enter (2)
2. Put the current spacetime code N into the Close-List list. Enter (3).
3. Obtain the spatial neighborhood code of the current spacetime code N through coding calculation. The neighborhood encoding loop is processed as follows: use the speed v to calculate the time slice occupied by the current spacetime code to the space neighborhood code, and obtain the corresponding spacetime code n. If the attribute corresponding to the spacetime code is inaccessible or in the Close-List, the same processing steps are performed on the next neighborhood code of N; otherwise, the following steps are performed on the neighborhood spacetime code n:
   (a) If the position of the neighborhood spacetime code n is exactly the target point E, then the current spacetime code N is used as the parent node of the neighborhood spacetime code n, and exit the loop, return to the adjacent spacetime code n, and enter step3. Otherwise, enter (b).
   (b) Calculate the actual cumulative cost g(n) from the starting spacetime code S to the adjacent spacetime code n. g(n) = g(N) + dis(N, n) represents the distance between N and n. Calculate the expected cost h(n) of the adjacent spacetime code n to the end point E, and calculate the cost f(n) = g(n) + h(n) of the adjacent spacetime code n. If the neighborhood code is not in the Open-List, set the parent node of the neighborhood code as the current code N, and add the adjacent spacetime code n to the Open-List. Otherwise, enter (c).
   (c) Compare f(n) with the cost value F(n) of the adjacent spacetime code n in the Open-List; if F(n) is not greater than f(n), return (3) to access the next adjacency code. Otherwise, the current point N is taken as the parent node of the adjacent spacetime code, and the cost of the adjacent spacetime code n in Open-List is changed to f(n).

Step3: If there is no return value, no valid path was found. Otherwise, according to the return code n, recursively search for the parent node of n until the starting point S is found, and this path is the path found by the Spacetime-A* algorithm.

3. Results
3.1. Spacetime Computation Experiment of Indoor Spacetime Grid

This experiment is a simulation experiment. The experimental data are randomly generated through uniform distribution to meet the requirements of local space subdivision space range and indoor time coding time requirements. Part of the randomly generated data is shown in Table 4.

- Spatial Computing Experiment of Indoor Space Subdivision Coding

This experiment mainly includes the calculation of the three-dimensional six-neighborhood of indoor space division and coding, the calculation of the three-dimensional twenty-six neighborhood of indoor space division and coding, and the calculation of the parent grid and subgrid of indoor space division and coding. The experimental data are randomly generated 50,000 coordinate points, and these 50,000 coordinate points are encoded, respectively, and then the above-mentioned spatial calculation is performed by using the encoding. When judging whether the neighborhood calculation is successful, the original code and the neighborhood code are decoded, and the spatial position and spatial range of the decoded grid are compared to determine whether they are adjacent to determine the correctness. When calculating the parent grid, the correctness is determined by judging whether the calculated spatial position and spatial range of the parent grid includes the
child grid. When calculating the subgrid, compare whether the calculated subgrid just fills the parent grid seamlessly without overlapping to determine the correctness. The experimental results are shown in Table 5.

**Table 4.** Schematic table of spacetime simulation data.

<table>
<thead>
<tr>
<th>X (m)</th>
<th>Y (m)</th>
<th>Z (m)</th>
<th>Day</th>
<th>Hour</th>
<th>Minute</th>
<th>Second</th>
<th>Leap Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>876.72</td>
<td>699.33</td>
<td>995.51</td>
<td>41</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>32.87</td>
<td>917.02</td>
<td>−518.85</td>
<td>414</td>
<td>21</td>
<td>34</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>71.42</td>
<td>321.91</td>
<td>536.68</td>
<td>2</td>
<td>20</td>
<td>37</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>491.53</td>
<td>−329.07</td>
<td>870.49</td>
<td>183</td>
<td>23</td>
<td>40</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>−832.96</td>
<td>−397.66</td>
<td>288.71</td>
<td>40</td>
<td>19</td>
<td>23</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>−157.24</td>
<td>−974.68</td>
<td>−245.72</td>
<td>150</td>
<td>0</td>
<td>32</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>807.37</td>
<td>511.31</td>
<td>−897.74</td>
<td>141</td>
<td>12</td>
<td>49</td>
<td>26</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 5.** Space computing experiment results table.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Spatial Calculation Method</th>
<th>Level</th>
<th>Time</th>
<th>Correct Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>Stereo six neighborhood computation</td>
<td>10</td>
<td>109.71 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Stereo twenty-six neighborhood computation</td>
<td>10</td>
<td>553.53 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Parent grid calculation</td>
<td>10</td>
<td>50.9 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Subgrid computing</td>
<td>10</td>
<td>143.11 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Stereo six neighborhood computation</td>
<td>15</td>
<td>99.28 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Stereo twenty-six neighborhood computation</td>
<td>15</td>
<td>526.22 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Parent grid calculation</td>
<td>15</td>
<td>54.86 ms</td>
<td>100%</td>
</tr>
<tr>
<td>50,000</td>
<td>Subgrid computing</td>
<td>15</td>
<td>142.16 ms</td>
<td>100%</td>
</tr>
</tbody>
</table>

The experimental results show that the computing time of various spatial computing methods has nothing to do with the level. The twenty-six neighborhood computations are the most time-consuming, averaging 10 microseconds per operation. The parent grid computes the fastest, taking an average of 1 microsecond per operation. The experimental results prove that the spatial calculation method based on indoor subdivision coding proposed in this paper is correct.

- **Time Calculation Experiment of Indoor Time Subdivision Coding**

  This experiment is mainly about the calculation of the unit time neighborhood of indoor time division coding. The experimental data are randomly generated two sets of simulation data, containing 50,000 and 100,000 time points, respectively. The experiment first generates time codes at levels 9, 14, 20, and 26 for these two sets of simulation data. Then, based on the time coding, the neighborhood calculation under the corresponding level is performed. This experiment also directly compares with the neighborhood computation method combining decoding, temporal unit neighborhood computation, and coding (referred to as decoding neighborhood computation for short) to prove the temporal neighborhood computation, directly based on temporal coding, is efficient and feasible. The experimental results of this experiment are shown in Table 6.
Table 6. Time calculation experiment results table.

<table>
<thead>
<tr>
<th>Level</th>
<th>Quantity</th>
<th>Decoding Neighborhood Computation</th>
<th>This Paper’s Method</th>
<th>Correct Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>50,000</td>
<td>237.31 ms</td>
<td>152.52 ms</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>50,000</td>
<td>247.34 ms</td>
<td>160.51 ms</td>
<td>100%</td>
</tr>
<tr>
<td>20</td>
<td>50,000</td>
<td>250.39 ms</td>
<td>175.49 ms</td>
<td>100%</td>
</tr>
<tr>
<td>26</td>
<td>50,000</td>
<td>228.42 ms</td>
<td>164.56 ms</td>
<td>100%</td>
</tr>
<tr>
<td>9</td>
<td>100,000</td>
<td>469.88 ms</td>
<td>339.98 ms</td>
<td>100%</td>
</tr>
<tr>
<td>14</td>
<td>100,000</td>
<td>449.91 ms</td>
<td>327.13 ms</td>
<td>100%</td>
</tr>
<tr>
<td>20</td>
<td>100,000</td>
<td>467.87 ms</td>
<td>324.14 ms</td>
<td>100%</td>
</tr>
<tr>
<td>26</td>
<td>100,000</td>
<td>443.89 ms</td>
<td>328.14 ms</td>
<td>100%</td>
</tr>
</tbody>
</table>

From the experimental results, it can be found that the time neighborhood calculation proposed in this paper takes about 3.2 microseconds each time, while the decoding time neighborhood calculation time is about 4.5 microseconds each time. The method in this paper improves the computational efficiency by 40%. In terms of accuracy, both methods are completely correct. The experimental results demonstrate the direct coding-based unit temporal neighborhood computation method proposed in this paper is stable and efficient at different levels and is completely correct.

3.2. Experiment of Mobile Robot Path Planning Based on Indoor Spacetime Grid Model

The selected experimental site is the corridor space on the west side of the third floor of the East Building of Remote Sensing of Peking University. First, the experimental site is modeled by the spacetime grid modeling method, and the center point of the subdivision voxel encoded by GeoSOT-3D as G002620644464066645057056 is used as the origin of indoor coordinates. Then, the experimental site is divided into 13 levels (0.25 m) indoors, and the indoor time coding adopts 26 levels (1 s). Finally, considering the safety requirements of the planned path and the physical constraints of the actual movement of the car, the neighborhood calculation in path planning uses four neighborhoods, and the distance cost function of the A* algorithm uses the Manhattan distance.

A total of four groups were carried out in this experiment, and the simulation method was adopted. In the four groups of experiments, for the convenience of calculation, the speed of the car is set to a constant speed of 0.25 m/s, that is, it passes through one grid per second.

**Experiment 1:** Ordinary A* path planning, the future traffic conditions of each grid are not considered in ordinary A* planning, which is consistent with the path planning of traditional static modeling. Arbitrarily select the start point and end point for common path planning. In ordinary path planning, only the grid traffic at the time of planning is concerned, and the experimental results are shown in Figure 6.

![Figure 6. Path Planning Experiment 1, ordinary A* path planning. Blue indicates wall, red indicates starting point, purple indicates the end point, green indicates passable area, grey indicates the path, orange indicates obstacles.](image-url)
Experiment 2: On the basis of Experiment 1, the experiment of the STA* algorithm is carried out. On the path of the experiment, set up several grids that will be impassable after 3 s. Then, the path planning of the car is carried out to verify whether the car can plan a more reasonable road through the time-varying information under the spacetime grid. The experimental results are shown in Figure 7.

![Figure 7](image_url)

**Figure 7.** Path Planning Experiment 2, the experiment of STA* algorithm. Blue indicates wall, red indicates starting point, purple indicates the end point, green indicates passable area, grey indicates the path.

Experiment 3: On the basis of Experiment 1, continue to carry out the STA* algorithm experiment. After the car departs, make a sudden impassability setting for some spacetime grids. Using the planned route data inside the spacetime grid, the unmanned car is stopped in an emergency. After the unmanned car receives the notification, it re-plans the path to verify the real-time early warning under the spacetime grid. The experimental results are shown in Figure 8.

![Figure 8](image_url)

**Figure 8.** Path Planning Experiment 3, a real-time warning experiment. Blue indicates wall, red indicates starting point, purple indicates the end point, green indicates passable area, grey indicates the path, orange indicates obstacles.

Experiment 4: Two-vehicle path planning, the safety and reliability of the STA* algorithm is verified by comparing the two-vehicle path planning under static A* and time-varying A*. This experiment is mainly to verify the collaborative ability of double-vehicle planning for trolleys under the spatiotemporal grid modeling. In the experimental map, two sets of starting points and ending points are selected as the starting point and ending point of the path planning of the two cars. When two cars start at the same time, firstly, the ordinary static A* path planning algorithm is performed on the two cars to obtain the planned paths of the two cars. After that, the STA* algorithm is performed on the two cars, respectively. When the path planning of the first vehicle is completed, the trajectory data are immediately stored in the corresponding spacetime grid, and then the second vehicle performs the STA* algorithm. The experimental results are shown in Figure 9.
Figure 9. Path Planning Experiment 4. (a) Schematic diagram of static A* algorithm path planning; (b) collision diagram for static A* algorithm path planning; (c) schematic diagram of two-vehicle path planning under the spacetime grid. Blue indicates wall, red indicates starting point, purple indicates the end point, green indicates passable area, grey indicates the path, orange indicates obstacles.

It can be found from the four sets of experimental results:

**Experiment 1 & Experiment 2**: The experimental results are shown in Figures 6 and 7. The trolley reasonably avoids the impassable area in the future, which proves the reasonable effectiveness of STA* under the spacetime grid modeling.

**Experiment 3**: A real-time warning experiment. During the travel of the car path, the trafficability of some areas on its future route is suddenly changed. Then, using the null-time grid of this part of the area, the data of the trolley that will pass afterwards are quickly obtained, and, immediately, advance braking is applied to the trolley, giving the trolley blockage warning and allowing the trolley to start planning the movement path again. The experimental results are shown in Figure 8, which demonstrates the real-time warning of the STA* path planning algorithm under the spacetime grid modeling.

**Experiment 4**: Two-vehicle collaborative path planning. Firstly, static two-vehicle path planning was performed, and both vehicles only considered their optimal paths from the static paths, resulting in collisions between the two vehicles according to the predefined path form. In contrast, for path planning under the spacetime grid, since each vehicle stores its own trajectory data in the spacetime grid when it finishes planning, this allows the next vehicle that performs STA* algorithm planning to immediately sense the future impassable grid. As a result, two routes that will have spacetime collisions will not be planned. The experimental results are shown in Figure 9. Where the yellow grid in Figure 9b is the collision point of the paths obtained from static A* planning, this experiment proves that STA* path planning with spacetime grid modeling can achieve cooperative planning of two vehicles and proves the superiority of the spacetime grid and the reliability of STA*.

4. Discussion

By analyzing the experimental results, we have the following three discussions:

1. The indoor spacetime grid model proposed in this paper does successfully integrate indoor space and time to achieve spacetime unity, which is not available in previous methods. The indoor spacetime grid model can also realize the modeling and coding of the indoor complex time-varying environment. Moreover, the data correlation
capability of the model lays the foundation for updating the dynamic information of individual grids afterwards; the accurate spacetime calculation capability lays the foundation for the path planning algorithm afterwards. In short, the model can realize the dynamic update of the object information within the model when the physical state of the real object changes, which is perfectly compatible with the time-varying characteristics of indoor space.

2. The spacetime grid modeling method can be perfectly compatible with the application of the common A* algorithm. At the same time, using the spacetime property, the STA* algorithm experiment in a time-varying environment can be realized, and the original path planning, which only considers the space level, is upgraded to the spacetime level, which is more suitable for the indoor environment with strong time-varying characteristics. It perfectly solves the problem of difficult mobile robot planning due to the complex time-varying indoor environment.

3. The approach proposed in this paper is highly scalable because it starts with a large global framework and then scales down to the interior, which allows the model to seamlessly interface with the existing mature technical standards. This is a natural advantage of the method. It starts from the entire planet, from large to small. Conversely, it is much easier to go from small to large. The spacetime grid model is kept constant and only the individual interior information of a single building is put into the model, just like filling a spacetime container with goods. This makes it easy to achieve the unified management of indoor location services for a region, a city, or even a country.

5. Conclusions

In this study, we took the construction of a digital twin city as the general background and landed on the relationship between the complex time-varying indoor environment and mobile robots, pointing out that the modeling unification, expression dynamics, operation visualization, and application extensiveness between them become the current urgent problems. To solve these problems, we propose a generalized approach to modeling indoor time-varying spatial environments by relying on the framework of geospatial three-dimensional dissection and integrating temporal dissection on the basis of spatial dissection. The model can realize that, when the physical state of the real object changes, the information of the object in the model can be updated dynamically, which is perfectly compatible with the time-varying characteristics of indoor space. Then, on the basis of this model, the dynamic path planning of the mobile robot in the indoor complex time-varying environment is realized.

However, the cost function under the STA* algorithm in this paper is still the distance function, and it can be investigated in the future whether the temporal cost can be introduced from the temporal dimension in addition to the spatially related cost to further improve the cost function of path planning in the spacetime dimension. In addition, in the future, it is necessary to provide more spatial-temporal grid expression methods for data to further improve the expression of the indoor spacetime environment. Finally, the robot path planning method based on the indoor spacetime grid model proposed in this study will be beneficial to the integration of indoor and outdoor location information services and has certain reference values for the construction of digital twin cities.

Author Contributions: Conceptualization, H.Z., Q.Z. and G.L.; methodology, H.Z. and Q.Z.; software, H.Z. and Q.Z.; validation, H.Z. and G.L.; formal analysis, H.Z. and G.L.; investigation, H.Z. and Q.Z.; data curation, H.Z.; writing—original draft preparation, H.Z.; writing—review and editing, G.L.; visualization, H.Z.; supervision, G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 62172021).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.
Remote Sens. 2022, 14, 2357

References


34. Qian, C.; Yi, C.; Cheng, C.; Pu, G.; Wei, X.; Zhang, H. GeoSOT-based spatiotemporal index of massive trajectory data. *ISPRS Int. J. Geo-Inf.* 2019, 8, 284. [CrossRef]
36. Li, S.; Sun, Z.; Wang, Y.; Wang, Y. Understanding urban growth in beijing-tianjin-hebei region over the past 100 years using old maps and landsat data. *Remote Sens.* 2021, 13, 326. [CrossRef]
38. Qian, C.; Yi, C.; Cheng, C.; Pu, G.; Liu, J. A coarse-to-fine model for geolocating chinese addresses. *ISPRS Int. J. Geo-Inf.* 2020, 9, 698. [CrossRef]
40. Cheng, C.Q.; Tong, X.C.; Chen, B.; Zhai, W.X. A subdivision method to unify the existing latitude and longitude grids. *ISPRS Int. J. Geo-Inf.* 2016, 5, 161. [CrossRef]
44. Zhang, H.; Zhu, Y.; Liu, X.; Xu, X. Analysis of obstacle avoidance strategy for dual-arm robot based on speed field with improved artificial potential field algorithm. *Electronics* 2021, 10, 1850. [CrossRef]
47. Pan, J.; Qian, Q.; Fu, Y.; Feng, Y. Multi-population genetic algorithm based on optimal weight dynamic control learning mechanism. *Jisuanji Kexue Yu Tansuo* 2021, 15, 2421–2437.
48. Cui, X.; Yang, J.; Li, J.; Wu, C. Improved genetic algorithm to optimize the wi-fi indoor positioning based on artificial neural network. *IEEE Access* 2020, 8, 74914–74921. [CrossRef]