atmosphere

Article

A System Coupled GIS and CFD for Atmospheric Pollution Dispersion Simulation in Urban Blocks

Qunyong Wu [1,2,3,*], Yuhang Wang [1,2,3], Haoyu Sun [1,2,3,4,*], Han Lin [1,2,3] and Zhiyuan Zhao [1,2,3]

1 Academy of Digital China (Fujian), Fuzhou University, Fuzhou 350108, China; 225527049@fzu.edu.cn (Y.W.); hlin@fzu.edu.cn (H.L.); zyzhao@fzu.edu.cn (Z.Z.)
2 Key Laboratory of Spatial Data Mining and Information Sharing of Ministry of Education, Fuzhou University, Fuzhou 350108, China
3 National Engineering Research Centre of Geospatial Information Technology, Fuzhou University, Fuzhou 350108, China
4 Xiaomi Inc., Nanjing 210019, China
* Correspondence: qywu@fzu.edu.cn (Q.W.); sunhaoyu3@xiaomi.com (H.S.)

Abstract: Atmospheric pollution is a critical issue in public health systems. The simulation of atmospheric pollution dispersion in urban blocks, using CFD, faces several challenges, including the complexity and inefficiency of existing CFD software, time-consuming construction of CFD urban block geometry, and limited visualization and analysis capabilities of simulation outputs. To address these challenges, we have developed a prototype system that couples 3DGIS and CFD for simulating, visualizing, and analyzing atmospheric pollution dispersion. Specifically, a parallel algorithm for coordinate transformation was designed, and the relevant commands were encapsulated to automate the construction of geometry and meshing required for CFD simulations of urban blocks. Additionally, the Fluent-based command flow was parameterized and encapsulated, enabling the automatic generation of model calculation command flow files to simulate atmospheric pollution dispersion. Moreover, multi-angle spatial partitioning and spatiotemporal multidimensional visualization analysis were introduced to achieve an intuitive expression and analysis of CFD simulation results. The result shows that the constructed geometry is correct, and the mesh quality meets requirements with all values above 0.45. CPU and GPU parallel algorithms are 13.3× and 25× faster than serial. Furthermore, our case study demonstrates the developed system's effectiveness in simulating, visualizing, and analyzing atmospheric pollution dispersion in urban blocks.

Keywords: CFD; 3DGIS; urban blocks; atmospheric pollution dispersion simulation; GIS-coupled

check for updates

Citation: Wu, Q.; Wang, Y.; Sun, H.; Lin, H.; Zhao, Z. A System Coupled GIS and CFD for Atmospheric Pollution Dispersion Simulation in Urban Blocks. *Atmosphere* **2023**, *14*, 832. https://doi.org/10.3390/atmos14050832

Academic Editors: Hui Liu, Mei Li, Mingyue Lu and Hua Shao

Received: 24 March 2023
Revised: 20 April 2023
Accepted: 25 April 2023
Published: 5 May 2023

1. Introduction

Atmospheric pollution has long been a paramount concern that it can cause serious harm to human health and lead to substantial economic losses. The simulation of atmospheric pollution dispersion is necessary for the loss reduction. Computational Fluid Dynamics (CFD) models are powerful computational and modeling tools that have been widely used in simulation environments in many different fields, with the most common application being the simulation of atmospheric dispersion of hazardous emissions at an urban scale. For example, some studies have conducted simulation experiments on gas leakage using CFD and have shown that an increase in ambient temperature leads to an increase in the diffusion rate of the leaked gas in both vertical and horizontal directions [1,2]. The study by Cormier et al. [3] demonstrates how CFD can predict the consequences of liquefied natural gas (LNG) releases and indicates that CFD is an effective tool that can enhance the safety of LNG facilities. Tauseef et al. [4] and Yoshie et al. [5] simulated gas dispersion after natural gas leakage incidents using CFD models and studied the effects of obstacles, such as vegetation, buildings, and roads, on gas dispersion in the environment. Siddiqui et al. [6] used a CFD model to simulate gas leakage in indoor pipelines, and they delineated the hazardous zone indoors based on the simulation results. To study gas diffusion in soil caused by leakage accidents in urban buried natural gas pipelines, some researchers proposed suggestions for the proper utilization of soil near the pipelines by using fluid mechanics theory, porous media theory, and CFD models [7,8]. Using a CFD model, Feißel et al. [9] examined how vehicle emissions affect air quality and suggested a simplified prediction method. Schalau et al. [10] developed and validated a modified k-ε-turbulence model in OpenFOAM v5.0 for heavy gas dispersion in built-up areas under atmospheric conditions, and the result demonstrates that the model is reliable in prediction. Comparing CFD simulations of pollutant dispersion with Gaussian-type models, some researchers have found that CFD is better suited for simulating pollutant dispersion in areas with high building density [11,12]. Artificial intelligence and data-driven models are gaining attention in various applications due to the availability of large data. Hybrid models, such as LSTM-ALO, ANFIS-GBO, ELM-PSOGWO, SVR-SAMOA, and ANN-EMPA, have been proposed and applied to agriculture and environmental prediction [13–17]. Nowadays, deep learning models, such as the Short-Term Memory network (LSTM), the Convolutional Neural Network (CNN), and the Recurrent Neural Network (RNN), have found wide application in predicting atmospheric pollutants across different spatial and temporal scales [18–26]. Some researchers have combined LSTM and CNN models, considering the spatial and temporal correlations in data, to improve the performance of the prediction model [27,28].

Fluent software, as the most widely used CFD package, has accumulated various physical models and developed more efficient numerical techniques. However, due to the inclusion of numerous models and parameters to meet the demands of different fields, Fluent software is complex and has a high learning curve. To address this issue, researchers have explored custom development based on Fluent software and achieved certain results in various fields. For example, by studying the command flow log files of Gambit, a pre-processing software for Fluent, some researchers have conducted secondary development that improved modeling efficiency and made it easier for researchers to use Fluent software, enabling them to focus on model solving and analysis of computational results [29–32]. To reduce simulation workload for users, Xiao et al. [33] implemented batch processing for specific computational problems using parameters extracted from Fluent's journal files. Li et al. solved the derived equations of motion electromagnetic field control through Fluent's User-Defined Scalars (UDS) [34].

However, CFD calculation outputs often lack geospatial coordinate information, making it difficult to analyze spatiotemporal characteristics. Fortunately, Three-dimensional Geographic Information Systems (3DGIS) offer advantages in spatial data management and visual analysis, allowing for real-world objects to be abstracted into three-dimensional data for immersive display and powerful spatial analysis. Combining GIS and CFD models allows for exploitation of existing GIS data and the visualization and mapping capabilities of GIS. In recent years, there has been growing interest in the coupling of CFD and GIS. Murakami [35] concluded that GIS data structures are well suited for constructing complex CFD geometric models. Chu et al. [36] extracted the coordinates and heights of building polygons from GIS software and used them as input to the CFD pre-processing software for constructing a CFD geometric model. Coirier et al. [37] proposed a method to construct urban block models required for CFD simulations based on GIS data and conducted atmospheric pollution dispersion simulations using the model. Wong et al. [38] investigated the efficiency of using GIS data in CFD models and the sensitivity of the CFD results to different GIS data formats and discovered that raster format was more efficient, and contour data demanded considerable effort. Hooff [39] constructed a high-quality CFD geometric model of an urban residential area by importing GIS data into the CFD pre-processing software and simulated wind and thermal environments based on the geometric model. Maohui Zheng et al. [40] combined GIS and CFD software to simulate the dispersion of toxic gases in urban blocks and concluded that coupling GIS software with CFD software provides better visualization of the pollutant dispersion in complex buildings. Yunkai

Xu [41] proposed a method to build complex urban building complexes for CFD simulation based on ArcGIS software and completed the loose coupling between Fluent software and ArcGIS software through intermediate data files. Xiaoyu Jiao [42] proposed a research scheme for coupling GIS and CFD for traffic pollutant dispersion simulation. Shouzhi Chang et al. [43] integrated CFD and GIS to identify and construct urban ventilation corridors (UVCs) in Changchun City, China. Wenzhong Wang et al. [44] investigated the import of Digital Elevation Model (DEM) data from GIS into the CFD pre-processing software ICEM CFD for three-dimensional model construction and meshing. They then simulated the target wind farm area based on the mesh file and analyzed the obtained wind velocity flow field vector map.

Although the coupling of GIS and CFD has been widely applied in various fields, such as atmospheric pollution dispersion simulation, and some achievements have been made, the coupling is mostly achieved through loose coupling, which involves complex interoperation and data conversion between independent software. Therefore, this study aims to present a tight coupling model of GIS and CFD for atmospheric pollution dispersion simulation. To achieve this, we utilized the following approaches. Firstly, to overcome the time-consuming pre-processing involved in CFD-based simulations of atmospheric pollution dispersion in urban blocks, we designed a high-performance CPU/GPU coordinate parallel transformation algorithm and encapsulated geometry construction and meshing commands using a parametric design method. This approach eliminated the data barrier between GIS and ICEM CFD, enabling rapid and automatic geometry construction and meshing. Secondly, to address the issue of CFD software requiring complex parameter inputs to cater to the demands of different fields, we examined CFD models related to atmospheric dispersion and proposed a Fluent GUI (Graphical User Interface) command flow parameterization design method and a TUI (Text-based User Interface) command flow encapsulation method. This made it possible to solve CFD atmospheric pollution dispersion models. Thirdly, to address the issue of CFD model outputs lacking geographic spatial coordinate information, which makes it difficult to display and analyze the spatiotemporal characteristics of pollution dispersion, we propose the use of multi-angle spatial partitioning and spatiotemporal multidimensional visualization analysis. This enables intuitive expression and analysis of simulation results, facilitating the understanding of pollution dispersion in urban blocks. Finally, leveraging the proposed methods and the tight coupling concept of GIS and CFD, we developed a user-friendly prototype system for simulating, visualizing, and analyzing atmospheric pollution dispersion in urban blocks, which significantly reduces the research threshold of CFD in pollution source dispersion. The system makes it easier for researchers or decision-makers to study and analyze air pollution in urban blocks and is expected to be an efficient tool to reduce property and human casualties caused by sudden pollution source leaks. For illustrative purposes, the three-dimensional characteristics of pollutant dispersion will be analyzed for a mixed residential and commercial area in Fuzhou.

2. Methodology

2.1. Introduction of ICEM CFD and Fluent Software

ICEM CFD and Fluent are two CFD software tools developed by ANSYS, Inc. Our coupling strategy relied on these two software tools to generate high-quality meshes and simulate fluid flow and pollutant dispersion in urban blocks. ICEM CFD is a pre-processing tool and primarily designed for mesh generation and geometry preparation, and it offers a range of mesh types, including structured and unstructured meshes. Fluent, on the other hand, is a powerful CFD solver that can simulate complex fluid flow and heat transfer problems, including atmospheric pollution dispersion. It features a variety of turbulence models, such as the k-ε turbulence model, to accurately capture the behavior of turbulent flows. Both ICEM CFD and Fluent are widely used in academic research and industrial applications for simulating fluid flow and heat transfer problems.

2.2. *Automatic Geometry Construction and Meshing*

Construction and meshing of urban block geometry is the foundation for atmospheric pollution dispersion simulations. Traditional CFD pre-processing software utilizes manual geometric modeling or a large number of coordinate inputs for geometry construction. For urban blocks containing considerable building complexes, this process is complex, time-consuming, and cannot meet the demands of near-real-time simulations. In this paper, we propose a parametric design method to generate a command flow file for geometry construction and meshing, and we ultimately achieve automatic construction and meshing of urban block geometry. The technical roadmap is illustrated in Figure 1.

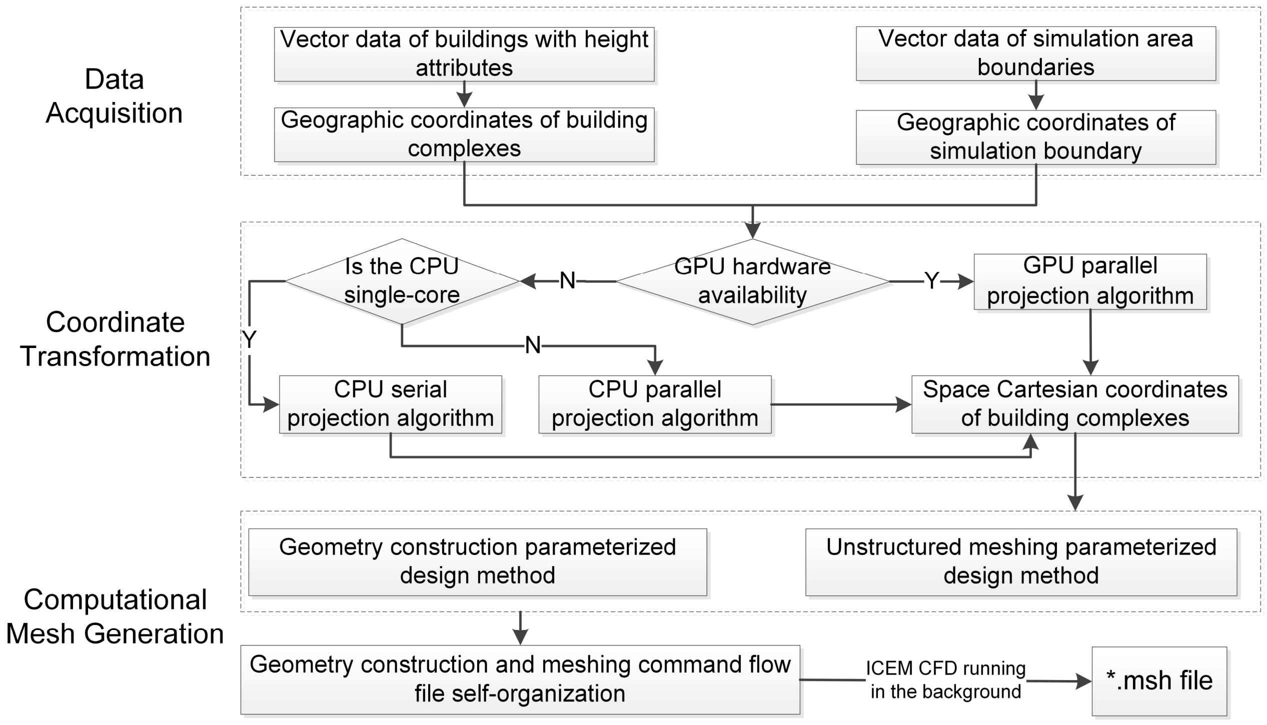

Data Acquisition
Vector data of buildings with height attributes
Vector data of simulation area boundaries
Geographic coordinates of building complexes
Geographic coordinates of simulation boundary
Coordinate Transformation
Is the CPU single-core
N
GPU hardware availability
Y
GPU parallel projection algorithm
Y
N
CPU serial projection algorithm
CPU parallel projection algorithm
Space Cartesian coordinates of building complexes
Computational Mesh Generation
Geometry construction parameterized design method
Unstructured meshing parameterized design method
Geometry construction and meshing command flow file self-organization
ICEM CFD running in the background
*.msh file

Figure 1. Technical roadmap of automatic construction and meshing of urban block geometry model. "*" represents a wildcard character, and "*.msh" denotes a mesh file format utilized to define the computational grid in CFD simulations.

2.2.1. Extraction of Geometry Coordinates

The research objects of urban block geometry are the building complexes within the entire simulation area. As the shapes of individual buildings can be complex, they are simplified as regular polyhedrons composed of vertices, edges, and faces, with the number of vertices varying depending on the shape of the building (In GIS, some features that appear to be a straight line at first glance are actually composed of multiple vertices when zoomed in). In this study, we assume that the buildings are solid with flat roofs. The constituent elements of a building with a rectangular parallelepiped shape are shown in Figure 2.

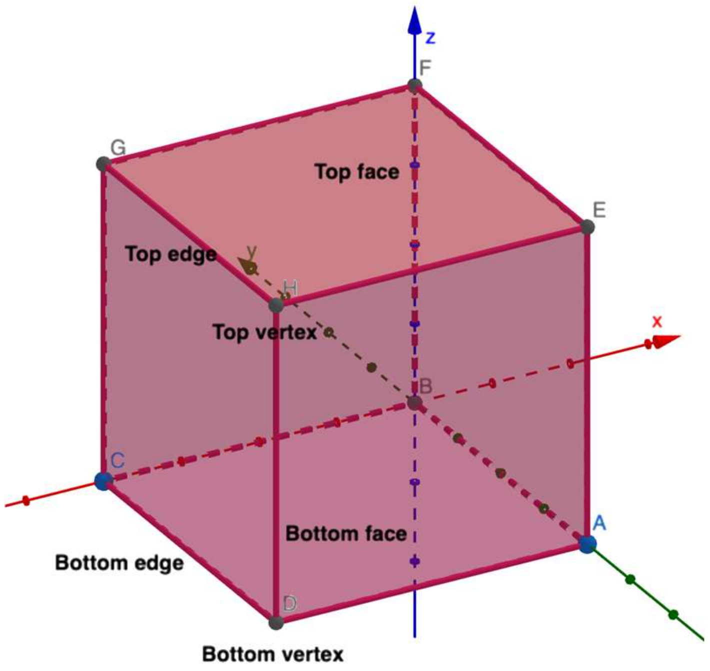


Figure 2. The elements of the building. The red line represents the x-axis, the green line represents the y-axis, and the blue line represents the z-axis. Letters A to H indicate the vertices of this building.

The geometry coordinates of the building complex in the urban block are obtained in the following three steps: (1) Based on the spatial topological containment relationship, using the simulation area boundary vector data as query object and the building vector data as queried object, all building objects within the simulation area are obtained, denoted as $B_i = \{B_i | i = 0, 1, 2 \ldots m\}$;. (2) Each obtained building object is composed of faces made up of vertices, with each vertex represented by its longitude and latitude coordinates (λ, φ). All vertex coordinates that make up each building object are saved in clockwise order, and the height attribute Z of each building object is read. The coordinate of each vertex is denoted as $P_j = \{\lambda_j, \varphi_j, Z | j = 0, 1, 2 \ldots n\}$;. (3) Repeat step 2 for all building objects in the simulation area to obtain all geometry coordinate data $B_iP_j = \{\lambda_{ij}, \varphi_{ij}, Z_i | i = 0, 1, 2 \ldots m, j = 1, 2 \ldots n\}$.

2.2.2. Parallel Approach-Based Coordinate Conversion of Building Complex

The process of computing the corresponding plane coordinates (X, Y) from the given geographic coordinates (λ, φ) in a spherical coordinate system using the Gaussian projection is called the forward Gaussian projection. The formula for the forward Gaussian projection is as follows:

$$\theta = \begin{cases} \arctan\left(\frac{b^2}{a^2} \times \frac{\tan\varphi}{\cos(\lambda - 3zoneID + 1.5)}\right) \\ \arctan\left(\frac{b^2}{a^2} \times \frac{\tan\varphi}{\cos(\lambda - 6zoneID + 3)}\right) \end{cases} \tag{1}$$

$$X = \int_0^{\theta} \mathrm{a} / \sqrt{\cos\theta^2 + \left(a^2/b^2 \times \sin\theta^2\right)} \mathrm{d}\theta \tag{2}$$

$$Y = \mathrm{r}\cos\theta\tan(\lambda - 6 \times zoneID + 3) + 500000 \tag{3}$$

where a and b are the semi-major and semi-minor axes of the Earth ellipsoid, respectively, φ is the latitude (radian value), λ is the longitude (radian value), $zoneID$ is the zone number of the current location, and X, Y represent the corresponding plane coordinates to be computed.

As the forward Gaussian projection involves a definite integral solution process, and the simulation area of urban block typically contains a large number of building vertices, using conventional CPU-based serial algorithms based on Gaussian projection for the conversion of all acquired B_iP_j data into plane coordinates under the Cartesian coordinate system can be time-consuming. To improve the efficiency of coordinate conversion, two approaches of coordinate conversion, namely, CPU parallel and GPU parallel, are investigated in this section.

The CPU serial algorithm converts the coordinate data of building vertices one by one. Only after the conversion of one vertex is completed, can the conversion of the next vertex proceed. As a result, each vertex conversion occupies only a small amount of CPU resources, leaving a significant portion of CPU resources idle. To address this issue, we propose a CPU-based parallel coordinate conversion algorithm that processes the conversion of coordinates for multiple buildings in parallel, with each building being treated as a single unit. This approach enables the simultaneous conversion of coordinates for multiple buildings at the same time. Using buildings as parallel units also effectively avoids the problem of reduced parallel efficiency caused by frequent thread switching due to excessively small parallel units.

As can be seen in Figure 3a, the building object is used as a parallel unit, and a cached thread pool technique is applied to call all available threads from the thread pool to perform forward Gaussian projection on the read geographic coordinates of multiple buildings simultaneously. A lock mechanism is then used to ensure the thread-safe storage of the converted coordinates until the conversion of all the read building geographic coordinates is completed.

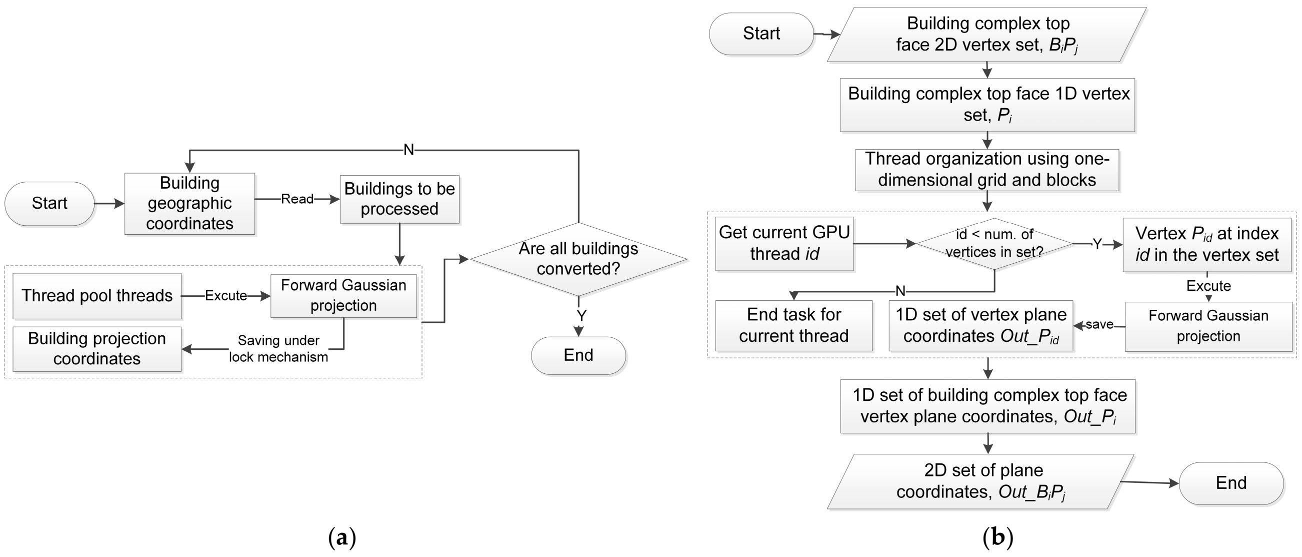
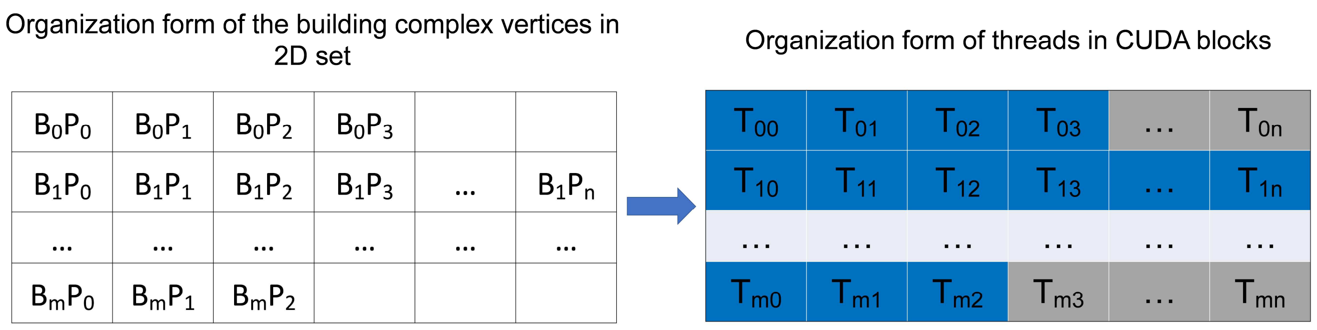

Start
Building geographic coordinates
Read
Buildings to be processed
N
Are all buildings converted?
Y
End
Thread pool threads
Excute
Forward Gaussian projection
Building projection coordinates
Saving under lock mechanism
(a)
Start
Building complex top face 2D vertex set, B_iP_j
Building complex top face 1D vertex set, P_i
Thread organization using one-dimensional grid and blocks
Get current GPU thread *id*
id < num. of vertices in set?
Y
Vertex P_{id} at index *id* in the vertex set
Excute
Forward Gaussian projection
save
1D set of vertex plane coordinates Out_P_{id}
N
End task for current thread
1D set of building complex top face vertex plane coordinates, Out_P_i
2D set of plane coordinates, $Out\_B_iP_j$
End
(b)

Figure 3. Two parallel approaches. (**a**) Flowchart of CPU parallel-based coordinate conversion of building complexes; (**b**) Flowchart of GPU parallel-based coordinate conversion of building complexes.

Although CPUs serve as the control core of an operating system, they are not well-suited for large-scale computationally intensive parallel operations. In contrast, GPUs are specifically designed for concurrent computation of large-scale data with highly uniform and non-dependent types. Therefore, based on CUDA, a NVIDIA's general-purpose parallel computing architecture, a GPU parallel coordinate conversion algorithm is proposed.

The CUDA framework employs a grid-based and block-based organization of GPU threads, with each grid and block organized in a two- or three-dimensional matrix. Nevertheless, as the number of vertices in each building varies, organizing GPU threads into a two-dimensional matrix may result in unused threads during coordinate conversion of buildings with a low number of vertices, as demonstrated in Figure 4.


Organization form of the building complex vertices in 2D set
Organization form of threads in CUDA blocks

B_0P_0	B_0P_1	B_0P_2	B_0P_3		
B_1P_0	B_1P_1	B_1P_2	B_1P_3	…	B_1P_n
…	…	…	…	…	…
B_mP_0	B_mP_1	B_mP_2			

T_{00}	T_{01}	T_{02}	T_{03}	…	T_{0n}
T_{10}	T_{11}	T_{12}	T_{13}	…	T_{1n}
…	…	…	…	…	…
T_{m0}	T_{m1}	T_{m2}	T_{m3}	…	T_{mn}

Figure 4. Illustration of wasteful GPU resources caused by storing building group coordinates in a two-dimensional set. The letter T represents threads. The blue cells on the right-hand image represent active threads, while the gray cells represent idle threads.

To address this issue, as can be seen in Figure 3b, the two-dimensional set B_iP_j needs to be converted into a one-dimensional set for storage. This involves reformatting the data of each row, which contain all vertex data for a single building, to a format where each row only contains data for a single vertex. Then, GPU threads are organized using a one-dimensional grid and one-dimensional block. In addition, the CUDA also provides four built-in variables, $gridDim$, $blockDim$, $blockIdx$, and $threadIdx$, to determine the unique identifier for each thread. Specifically, each thread with a unique ID executes the conversion of vertex coordinate at the corresponding ID in the one-dimensional set. The detailed algorithm is demonstrated in Algorithm 1.

2.2.3. Parametric Design Method for Geometry Construction and Meshing

The ICEM CFD software does not provide any secondary development APIs, but it does offer a command flow file for batch processing. In order to automate geometry construction and meshing, we have encapsulated the relevant commands using a parametric design method to generate command flow files for the relevant operations.

As the urban block buildings studied in this paper are treated as regular cubes, the construction of the simplified building in ICEM CFD software requires three steps: (1) creating the top vertices and bottom vertices, (2) creating the top, bottom, and side edges by connecting adjacent vertices, and (3) creating the top, bottom, and side faces of the building by connecting adjacent edges. To illustrate the script command for creating a point, the script command flow is as follows:

$$\text{ic\_point}\{\}\text{GEOMpnt.000}, 0, 0 \tag{4}$$

where the command "ic_point {}" creates a point in the GEOM group with the name pnt.00 at the location (x, y, z) of (0, 0, 0). All variables, except for the "ic_point {}" command, can be changed. Therefore, all variables in the command flow can be parameterized and encapsulated in a "CreatePoint" method. Similarly, we can encapsulate methods for "edge", "face", "group", and other command flows. The detailed flowchart of the automatic construction method of the geometry is shown in Figure 5.

Algorithm 1. Algorithm steps of the parallel GPU coordinate conversion of building complex vertices in CUDA-based framework. GPU parallel calculation of building vertex geographic coordinates to plane coordinates

```
Input: Coordinates of all building vertices λφZ_in ← {$B_iP_j$}, Number of all building vertices n
Output: Space Cartesian coordinates of all building vertices after conversion XYZ_out ← {$XYZ_i$}
function __global__ Kernel(XYZ_out, λφZ_in, n)//GPU-side functions
  id ← (blockIdx.x * blockDim.x) + threadIdx.x//Get thread id
  if id < n do//Perform coordinate conversion for threads that meet the condition
    //Perform forward Gaussian projection
    λφZ ← λφZ_in[id]
    X, Y, Z = GaussianForward(λφZ)
    XYZ_out[id] ← X, Y, Z//Store the converted coordinates
  end if
end function
function __host__ buildsProjection(XYZ_out, λφZ_in, n)//CPU-side functions
  //Apply for space on the device end
  cudaMalloc(dev_λφZ_in, ... ), cudaMalloc(dev_ XYZ_out, ... )
  //Copy data from host side to device side
  cudaMemcpy(dev_λφZ_in, λφZ_in, ... )
  cudaMemcpy(dev_XYZ_out, XYZ_out, ... )
  block ← blockMax/2, grid = (n − 0.5)/block + 1//Design CUDA thread organization
  Kernel<<<grid,block>>>(dev_XYZ_out, dev_λφZ_in, n)//Call core functions
  //Copy the calculation results from the device side back to the host side
  cudaMemcpy(XYZ_out, dev_XYZ_out, ... )
  //Release space requested on the device side
  cudaFree(dev_λφZ_in), cudaFree(dev_XYZ_out)
end function
```

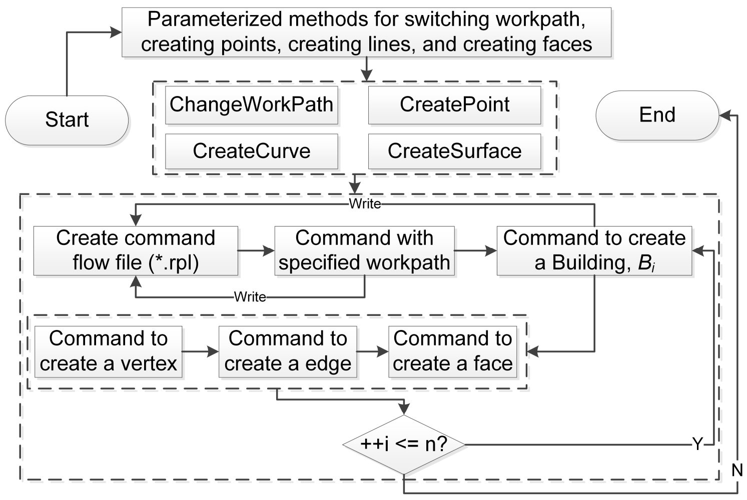

Parameterized methods for switching workpath, creating points, creating lines, and creating faces
Start
ChangeWorkPath
CreatePoint
CreateCurve
CreateSurface
End
Write
Create command flow file (*.rpl)
Command with specified workpath
Command to create a Building, B_i
Write
Command to create a vertex
Command to create a edge
Command to create a face
++i <= n?
Y
N

Figure 5. Flowchart of the automatic construction method of geometry of building complex.

ICEM CFD offers two meshing methods for geometry: structured and unstructured meshing. Structured meshing requires a predefined grid topology with a fixed number of nodes and elements, and it has strict geometric rules, such as the edges of adjacent cells needing to be aligned. This means that a significant amount of manual adjustments and corrections are required during structured meshing, increasing the difficulty and complexity of automated operations and making it unsuitable for large-scale simulations. On the other hand, unstructured meshing does not require predefined grids and does not need to follow strict geometric rules, making it easier to generate meshes for a variety of different geometries. It has a fixed sequence of steps and is easier to parameterize. Therefore, despite the fact that the building vector data in this study is composed of regular polyhedrons, unstructured meshing is more suitable due to the complexity and scale of the urban block as a whole and the need for automation. The unstructured meshing process typically involves the following steps: (1) creating a geometry, (2) setting meshing parameters, (3) saving a TETIN file representing the geometry (*.tin), (4) running the tetrahedral mesh generator, (5) smoothing the mesh elements, (6) saving the final unstructured mesh (*.uns), and (7) generating the mesh (*.msh). The necessary commands are shown in Table 1.

Table 1. Unstructured mesh related operations corresponding to the commands.

Operations	Commands
setting the global meshing parameters	ic_set_meshing_params global
setting geometric parameters for a specific geometry family	ic_geo_set_family_params
generating a tetrahedral mesh	ic_run_tetra
smoothing the mesh elements	ic_smooth_elements
saving the generated unstructured mesh	ic_save_unstruct
executing a user-defined script or external program	ic_exec

For instance, to generate a TETIN file that represents the geometry, the command ic_save_tetinfileonly_visible is used. Here, "file" is a placeholder for the name of the TETIN file that will be generated (*.tin). The parameter "only_visible" indicates whether only the visible parts of the geometry will be included in the TETIN file. The "only_visible" option indicates that, during meshing or output, only the visible parts of the geometry will be saved according to the current view, while the parts that are invisible or obstructed by other objects in the geometry will not be saved or included in the output file. The default value for this parameter is 0, which saves all parts of the geometry. Similar to the above-mentioned strategy, the command can be encapsulated in the "SaveModel" method, and other commands can also be encapsulated in their corresponding methods.

2.3. Fluent-Based Solution for Atmospheric Dispersion Models

The CFD software ANSYS Fluent has complex parameter input designed to meet the needs of various fields, and it does not provide secondary development APIs. In this section, we examine CFD models related to atmospheric dispersion and propose a Fluent GUI (Graphical User Interface) command flow parameterization design method and a TUI (Text-based User Interface) command flow encapsulation method to indirectly provide secondary development APIs for Fluent. Finally, we implement automated generation of model solver command flow files for the calculation of atmospheric pollution dispersion models in Fluent. The technical roadmap is illustrated in Figure 6.

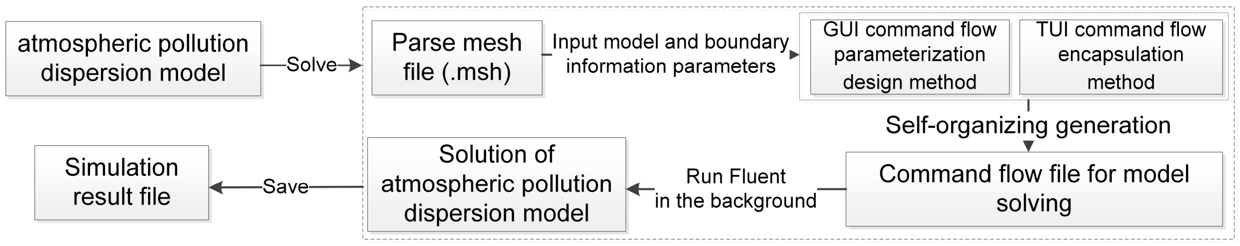

atmospheric pollution dispersion model
Solve
Parse mesh file (.msh)
Input model and boundary information parameters
GUI command flow parameterization design method
TUI command flow encapsulation method
Self-organizing generation
Command flow file for model solving
Run Fluent in the background
Solution of atmospheric pollution dispersion model
Save
Simulation result file

Figure 6. Technical road of CFD atmospheric dispersion simulation for urban blocks.

2.3.1. GUI Command Flow Parameterization Design Method

By manually operating the Fluent software, a Journal File (*.jou) can be generated to record a sequence of actions performed in the GUI. For example, a command flow generated by specifying a boundary condition for a particular zone is:

$$(\mathrm{cx-gui-docx-set-list-selections\text{“}BoundaryConditionsTable1List2(Zone)\text{”}\text{‘}(1)}) \quad (5)$$

This command selects the second item (index 1) in the list named "Boundary ConditionsTable1List2 (Zone)" in the Fluent GUI. While this index number is variable and not available to external programs, the name of the corresponding bounding area (e.g., inlet1) can be determined. Thus, the index can be replaced with the name, which is the key to parametric design of the GUI and allows for the parameterized encapsulation of Fluent-related operational command flows.

2.3.2. TUI Command Flow Encapsulation Method

The Fluent GUI command flow is not well organized and requires a call window display, making its execution relatively inefficient. In contrast, Fluent TUI commands can be directly entered and executed in Fluent's Console or saved as a log file (*.jou) in a text editor for later execution by Fluent. A TUI command is composed of a directory and a command, where the directory is selected first, followed by executing the corresponding command under that directory. Fluent TUI provides 14 categories of first-level directories, each corresponding to a functional description listed in Table 2.

Table 2. Fluent TUI first-level directory commands.

Directory Name	Functional Descriptions
adapt/	contains commands related to mesh adaptation
adjoint/	contains commands related to adjoint solver
define/	contains commands to define materials, boundary conditions, and other simulation parameters
display/	contains commands to control the display of the Fluent GUI
file/	contains commands to import or export files
mesh/	contains commands related to mesh generation and manipulation
parallel/	contains commands related to parallel
plot/	contains commands to create plots and animations
report/	contains commands to generate reports, such as force or mass reports
server/	contains commands to control the Fluent server
solve/	contains commands related to the solution of the fluid problem
surface/	contains commands related to surface modeling
turbo/	contains commands related to turbomachinery simulation
views/	contains commands to create, save, and restore views of the Fluent GUI

For example, the TUI command used to define a boundary condition is:

$$/\mathrm{define}/\mathrm{boundary}-\mathrm{conditions}/\mathrm{velocity}-\mathrm{inletinlet1n\ n\ y\ y\ n\ 10\ n\ 101325} \tag{6}$$

where "/define/boundary-conditions/" refers to the directory in which the command is located, "velocity-inlet" specifies the type of boundary condition, and "inlet1" is the name of the boundary. The input options for the command vary depending on the model being used. For instance, if the viscous model is set to Laminar and all other models are turned off, the necessary options would be the boundary name, velocity method (which has three options: "Magnitude and Direction", "Components", and "Magnitude, Normal to boundary"), reference frame (which has two options: "Absolute" and "Relative to Adjacent Cell Zone"), velocity magnitude, and initial gauge pressure. In the command, "n" and "y" represent "no" and "yes," respectively. For example, "n n y" indicates the third one of velocity methods, and "y" indicates the first one of reference frames. The velocity magnitude is set to 10 m/s, and "n" preceding the velocity magnitude indicates no use of a velocity profile. Similarly, "n" preceding the initial gauge pressure indicates no use of a profile for the initial gauge pressure, which is set to 101,325 Pa. Note that, when selecting the first reference frame option, the command is simply "y", not "y n", and when selecting the second reference frame option, the command is "n y." The other commands follow the same pattern.

As we can see, there are two major challenges in using TUI commands: (1) the parameters are not fixed and may vary depending on the specific models being used, and (2) the parameters are often represented in the form of "n" and "y", which can be difficult to interpret. In this study, an object-oriented approach is used to encapsulate the TUI command flow of each function into easy-to-use functional objects.

As an example of the encapsulation process for the TUI command of setting boundary conditions mentioned above, the process is illustrated in Figure 7. Simulating urban air pollution dispersion involves four models: energy equation, turbulence model, radiation model, and species model. Based on our research on Fluent TUI commands, ten interfaces need to be designed. The boundary conditions involve five types: velocity-inlet, pressure-inlet, mass-flow-inlet, pressure-outlet, and mass-flow-outlet, which correspond to five methods under each of the ten interfaces. After designing the interfaces, an implementation class is provided to implement all the interfaces. However, due to the complexity of Fluent software operations, each method in the implementation class involves numerous parameters, so the parameters of each method are encapsulated into a parameter class. Finally, a boundary condition setting class (BoundaryConditionSet) is designed, which is accessible for developers.

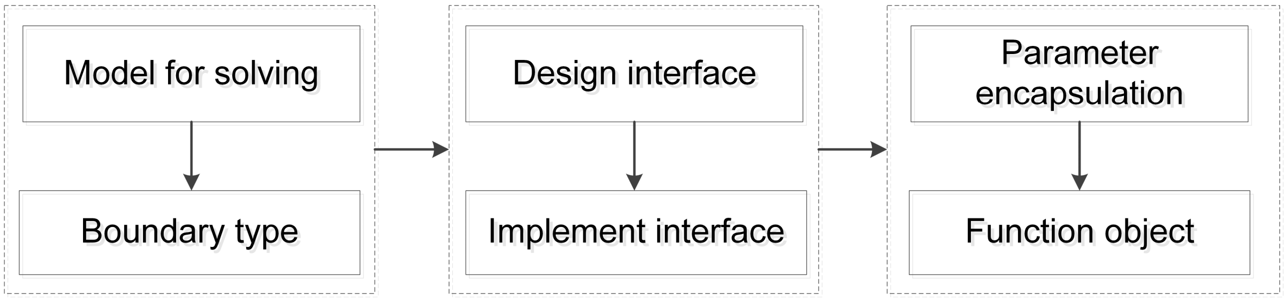


Figure 7. TUI encapsulation process of boundary condition.

2.4. Visualization and Analysis of Simulation Outputs

After the simulation is finished, Fluent stores the relevant calculation values of the entire study area in a binary text file, including node numbers, XYZ coordinates, pressure values, XYZ velocity components, velocity magnitude, velocity angle, and mass fractions of different materials at the corresponding coordinate positions. As a result of these simulation outputs being in text format, they are typically used for visualization and analysis outside of GIS, and they have not been effectively integrated within GIS for impact assessment. However, by coupling GIS and CFD, it is possible to perform spatiotemporal multi-dimensional visualization and spatial analysis on simulation outputs.

2.4.1. Conversion of Simulation Outputs to 3DGIS Data

The simulation outputs are typically saved as text data and cannot be directly imported into 3DGIS. To integrate the simulation outputs into 3DGIS, Inverse Gaussian projection is needed to convert the outputs into 3DGIS-compatible data.

Due to the large number of nodes contained in each time step of the simulation outputs, a considerable amount of conversion time is required. Therefore, the CPU parallel and GPU parallel algorithms mentioned in Section 2.2.2 are required to reduce the conversion time. The resulting converted CSV data is then transformed into a three-dimensional point Shapefile format using the geoprocessing tool interface (IGeoProcessor) in the ArcObjects library. The specific workflow is shown in Figure 8.

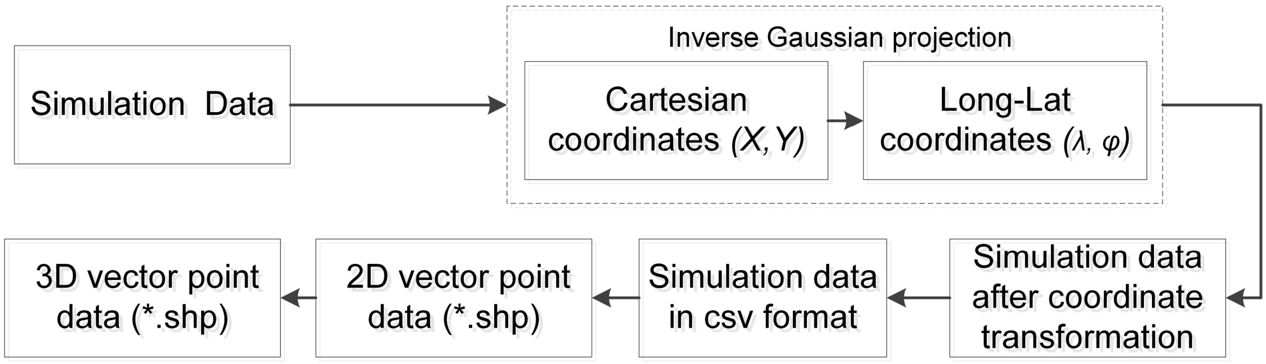

Simulation Data
Inverse Gaussian projection
Cartesian coordinates (X, Y)
Long-Lat coordinates (λ, φ)
Simulation data after coordinate transformation
Simulation data in csv format
2D vector point data (*.shp)
3D vector point data (*.shp)

Figure 8. Workflow of the simulation outputs to 3DGIS data. "*" represents a wildcard character, and "*.shp" represents a file format used for storing shapefile data in geographic information systems (GIS) software.

2.4.2. Multi-Angle Spatial Partitioning of Three-dimensional Spatial Simulation Outputs

After converting each node of the simulation output to a three-dimensional vector point, a large number of discrete three-dimensional points are spread across the entire study area, as shown in Figure 9. Visually, only the three-dimensional vector points distributed on the surface of the study area can be observed (blue-green points in Figure 9), while the internal three-dimensional points are difficult to observe, as they are obscured by the external points. To address this issue, our developed system implements the horizontal and arbitrary direction partitioning of three-dimensional spatial simulation outputs, which is especially useful for managing large-scale three-dimensional spatial datasets generated from atmospheric dispersion simulations. This approach enables more efficient processing of complex three-dimensional spatial data and facilitates more accurate and effective spatial analysis. The flowchart is shown in Figure 10.

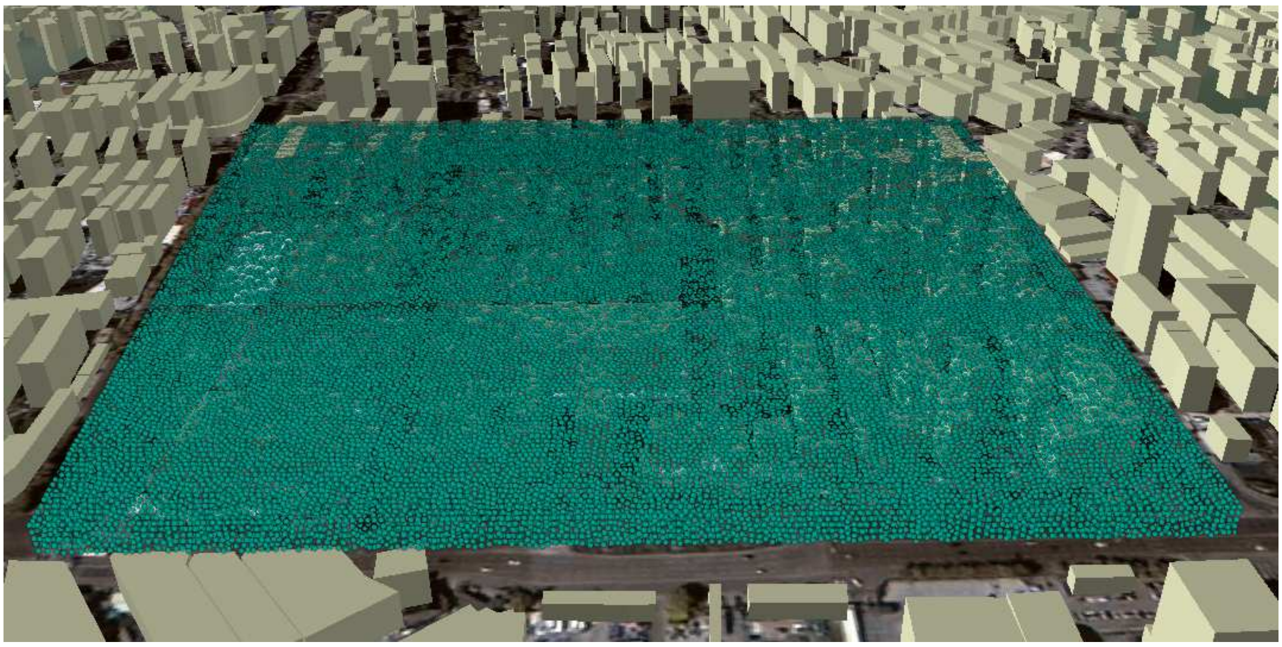


Figure 9. An example of 3DGIS vector point data converted from dispersion data at 100 s. Each blue-green point represents the converted node of simulation outputs.

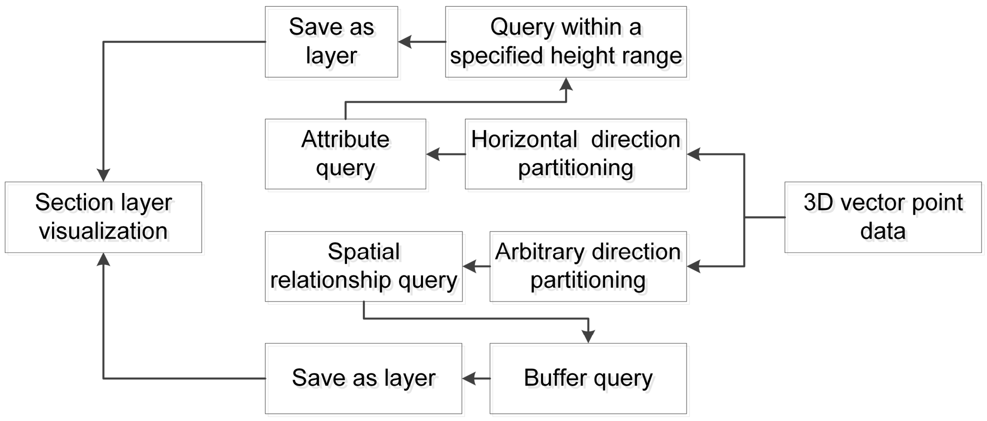

Save as layer
Query within a specified height range
Attribute query
Horizontal direction partitioning
Section layer visualization
3D vector point data
Spatial relationship query
Arbitrary direction partitioning
Save as layer
Buffer query

Figure 10. Flowchart for multi-angle partitioning method.

The attribute query function in GIS can be used to obtain point objects located on the horizontal cutting plane. However, since three-dimensional points have volume, simply querying objects that satisfy the height attribute at the level of the horizontal cutting plane is not enough. Instead, we need to query objects that fall within a certain height range around the horizontal cutting plane, with the default range being the diameter size of the three-dimensional point. The default diameter size of three-dimensional points in GIS is typically 2 or 4 pixels, which is automatically adjusted based on the geographic extent and scale of the data. Red three-dimensional vector points represent the horizontal section data of simulation data in Figure 9 at a height of 1.5 m, as shown in Figure 11a.

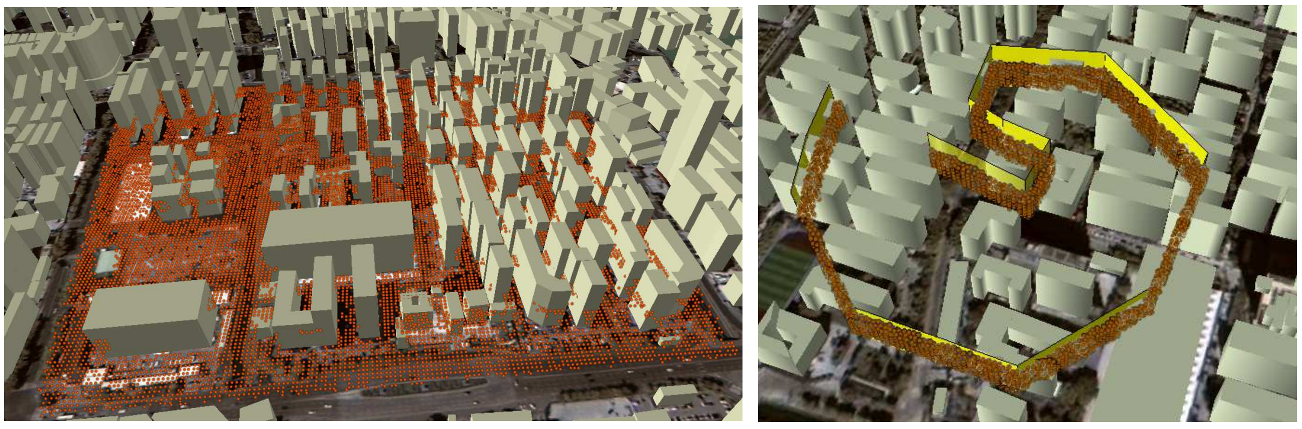


Figure 11. (**a**) Horizontal section data at a height of about 1.5 m. The red vector points represent the simulated data that meet the height attribute of 1.5 m in the query; (**b**) Cross-sectional data cut in any direction. Yellow cutting plane represents a plane created by the user for arbitrary direction partitioning of simulation data, and the brownish-red vector points represent the simulation data queried through the yellow cutting plane in space.

However, using attribute query to perform arbitrary directional partitioning can be very complex or even impossible. Fortunately, GIS offers spatial topology query, which allows us to obtain point objects in cutting planes oriented at any angle with respect to the dataset. The specific steps are as follows: (1) create a baseline for the cutting plane; (2) set a symmetric buffer on both sides of the cutting plane baseline to form the cutting plane base, with the buffer distance defaulting to the diameter size of the discrete points; (3) stretch the cutting plane base to form the partitioning volume; (4) conduct a spatial containment query of the simulated vector points with the partitioning volume; (5) save the query results as a section layer. As shown in Figure 11b, the brownish-red three-dimensional vector points represent the cross-sectional data of the simulation data of Figure 9 at the yellow cutting plane.

2.4.3. Spatiotemporal Multidimensional Visualization and Analysis

By tightly coupling CFD with 3DGIS, it becomes possible to directly access and visualize the converted simulation results data in 3DGIS, enabling spatial analysis of the simulated data. However, since atmospheric pollution dispersion is a process-based phenomenon, the temporal expression of the simulation results is equally important as their spatial position. Therefore, this paper investigates two mixed forms of layer visualization and Image animation visualization to add a temporal dimension to the three-dimensional spatial simulation result data and achieve a spatiotemporal multidimensional visualization of the simulation result.

The steps for layer visualization are as follows: (1) partition the three-dimensional simulation outputs of all moments based on a spatial partitioning method to obtain the section data of interest; (2) render the section data at each moment using symbol methods, such as the gradient color method; (3) load and display the section data of interest at each moment in a time sequence. However, the layer-based loading approach does not provide a dynamic view of the spatial distribution of pollutants over time. Thus, image animation visualization is proposed based on layer visualization to solve this problem. The detailed steps are as follows: (1) set the visibility of the rendered section layers at each moment to ensure that only the layers at the same moment are visible and save the layer data at this moment in the current view as images named with the time of the current moment; (2) play the saved images in a time sequence.

2.5. Implementation of the CFD Coupled with 3DGIS

In this study, a prototype system is developed for simulating, visualizing, and analyzing atmospheric pollution dispersion based on the tight coupling model of GIS and CFD. The user interface is shown in Figure 12. The system is built using the .NET open-source development platform and leverages the ArcObjects component object set to create three-dimensional virtual geographic scenes.

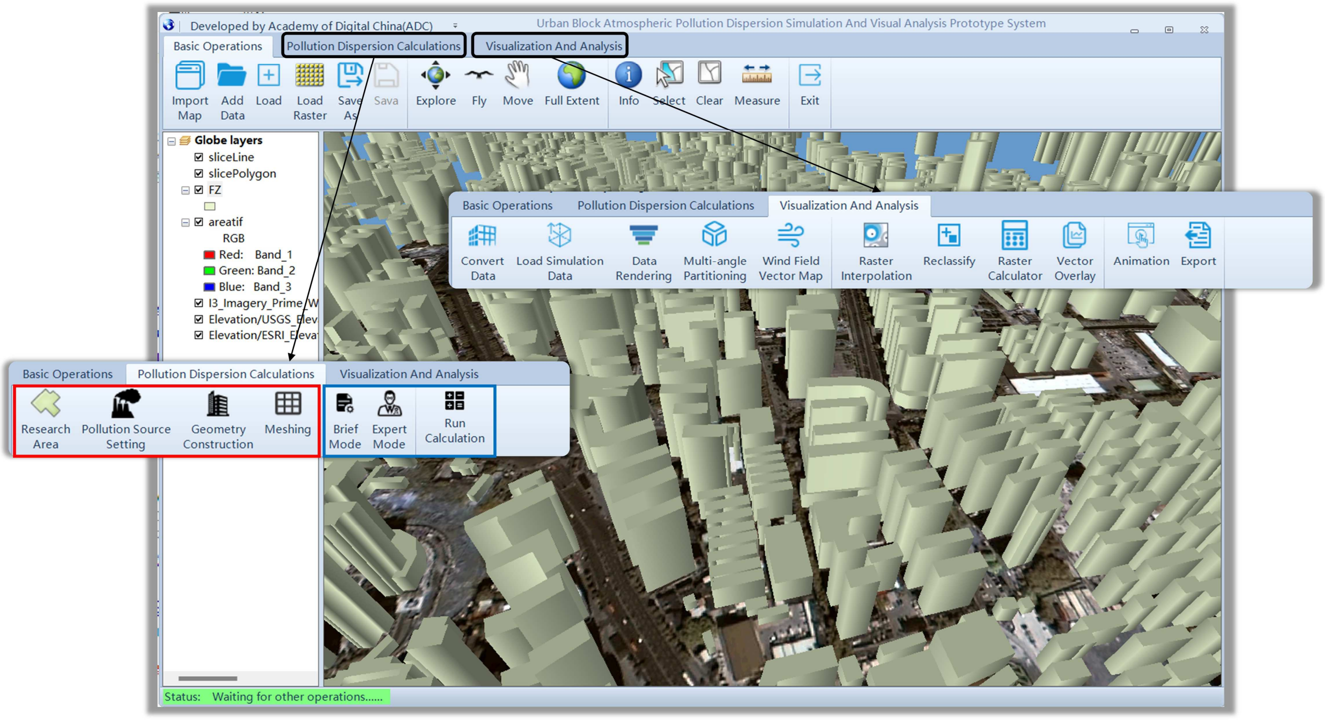
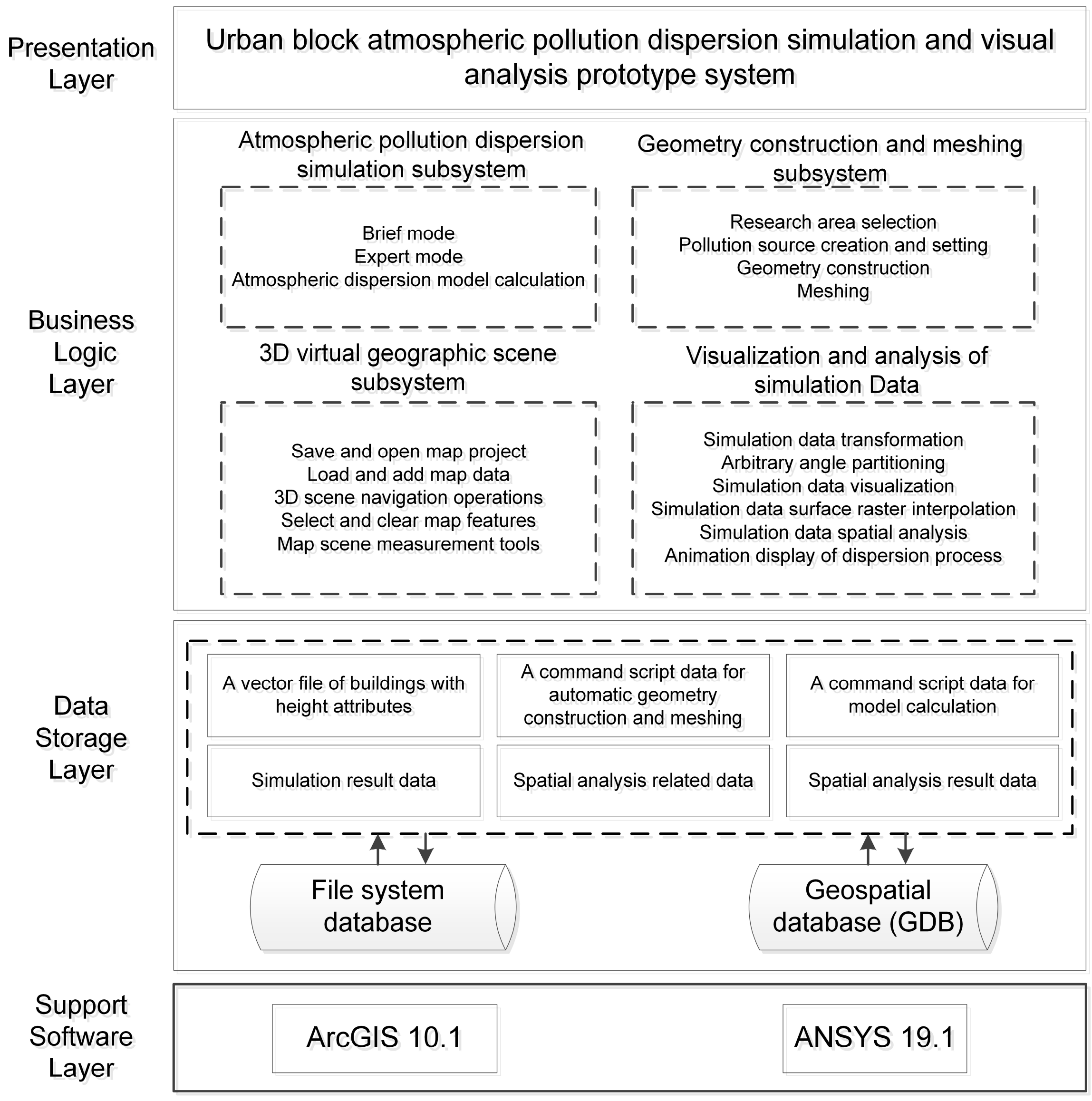


Figure 12. Three-dimensional virtual geography scene and operation interface.

Figure 13 presents the entire architecture of the prototype system, which is designed with a three-layer structure of C/S architecture (C/S architecture stands for Client/Server architecture. It refers to a computing model where the client-side and server-side of a system are separated into distinct layers, with each layer responsible for different functions. The client-side is typically responsible for user interface and presentation, while the server-side is responsible for data storage and processing). The data storage layer mainly uses local file system databases and commonly used geospatial databases in GIS, including building vector data with height attributes, ANSYS ICEM CFD command flow files (*.rpl) for automating geometry construction and meshing, ANSYS Fluent command flow files (*.jou) for atmospheric pollution dispersion simulation calculations, simulation result data from calculation, related data for spatial analysis of simulation results, and other data obtained from spatial analysis. The business logic layer mainly realizes the construction of three-dimensional virtual geographic scenes, the automatic construction of urban block geometry and mesh, the simulation calculation of atmospheric pollution dispersion in urban blocks, and the spatiotemporal multidimensional visualization and analysis of the simulation result data. The representation layer is a desktop client application built on the .NET platform.



Presentation Layer
Urban block atmospheric pollution dispersion simulation and visual analysis prototype system
Business Logic Layer
Atmospheric pollution dispersion simulation subsystem
Brief mode
Expert mode
Atmospheric dispersion model calculation
Geometry construction and meshing subsystem
Research area selection
Pollution source creation and setting
Geometry construction
Meshing
3D virtual geographic scene subsystem
Save and open map project
Load and add map data
3D scene navigation operations
Select and clear map features
Map scene measurement tools
Visualization and analysis of simulation Data
Simulation data transformation
Arbitrary angle partitioning
Simulation data visualization
Simulation data surface raster interpolation
Simulation data spatial analysis
Animation display of dispersion process
Data Storage Layer
A vector file of buildings with height attributes
A command script data for automatic geometry construction and meshing
A command script data for model calculation
Simulation result data
Spatial analysis related data
Spatial analysis result data
File system database
Geospatial database (GDB)
Support Software Layer
ArcGIS 10.1
ANSYS 19.1

Figure 13. The entire architecture of the prototype system.

The system is comprised of four major modules, namely: the creation of three-dimensional virtual geographic scenes, the automatic geometry construction and meshing for urban blocks (indicated by the red box in Figure 12), simulation of pollution dispersion (represented by the blue box in Figure 12), and spatiotemporal multi-dimensional visualization and analysis of simulation results. A more detailed breakdown of system functions is shown in the business logic layer of Figure 13.

Based on the approach investigated in Section 2.3, we designed two parameter-setting modes for atmospheric pollution dispersion simulation, namely, the brief mode and the expert mode. The former has pre-configured settings for specialized parameters, making it suitable for users who do not have specific requirements for the assumptions of fluid mechanics. For example, in the brief mode, CFD models only enable the energy equation, the standard k-epsilon model, and species transport model. After setting up the pollutant source and boundary of the study area (Figure 14), users can perform atmospheric pollution dispersion simulation by simply using the brief mode to specify parameters, such as pollutant source material, velocity-inlet boundary with its velocity magnitude, mass flux of pollutant source material, and the duration of the simulation. This simplifies the model parameter setting process and significantly reduces the learning curve for users. The latter provides more flexibility in parameter settings, allowing the user to select and configure parameters according to specific situations, which is ideal for users with adequate knowledge of fluid mechanics.

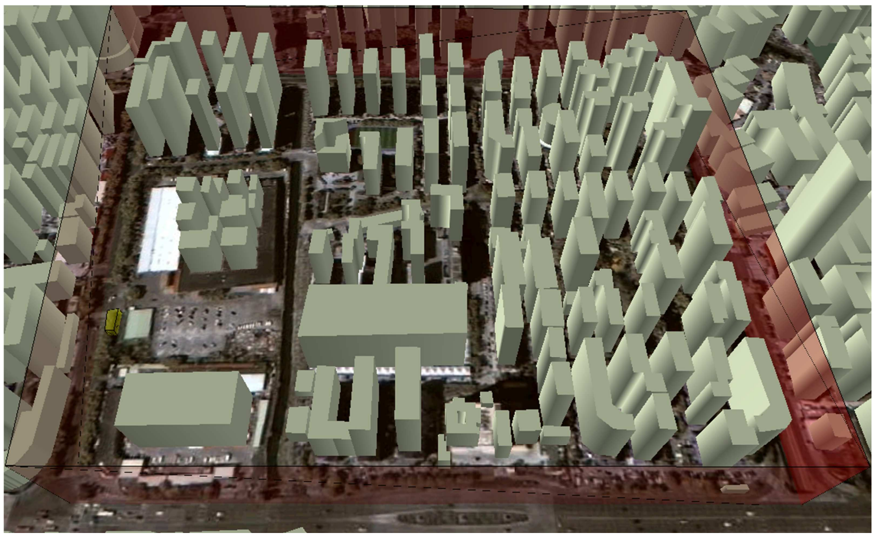
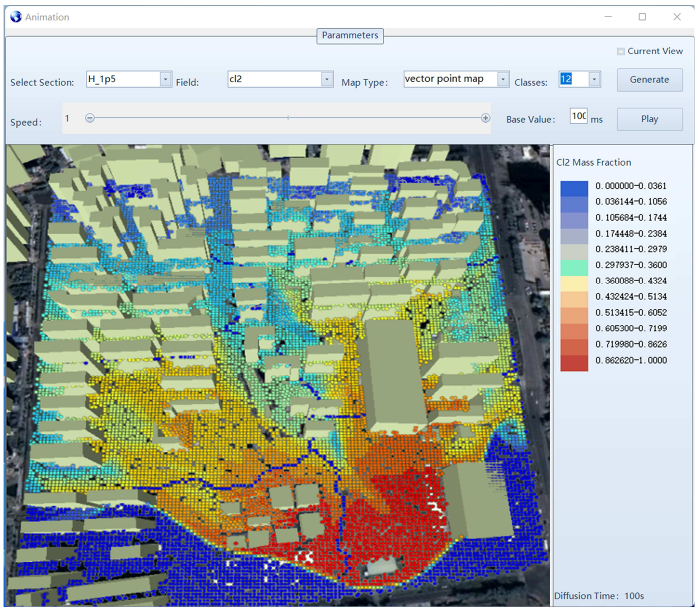


Figure 14. A sample of the study area and pollutant source. The red represents the boundary of the study area, while the yellow-green cube on the left side of the image indicates the pollutant source.

By tightly coupling GIS and CFD, users or decision makers can perform a variety of visualization and analysis operations on simulated results, including: (1) gradual color rendering, (2) conversion of simulated vector data to raster data using raster interpolation algorithms such as the inverse distance weight method (IDW) and Kriging method, (3) attribute query and spatial relationship query, (4) overlay analysis of the simulated data and residential block polygon data, (5) visualization of wind field vector maps at a specific moment, (6) multi-angle spatial partitioning, and (7) spatiotemporal multidimensional visualization and analysis. For instance, users can use the spatial partitioning method proposed in Section 2.3.2 to partition the simulation data for all moments, and then they can create animations for specified sections using the Animation Demo Window (Figure 15).


Animation

Parammeters

Current View

Select Section: H_1p5
Field: cl2
Map Type: vector point map
Classes: 12
Generate

Speed: 1
Base Value: 100 ms
Play

Cl2 Mass Fraction

🟦	0. 000000–0. 0361
🟦	0. 036144–0. 1056
🟪	0. 105684–0. 1744
⬜	0. 174448–0. 2384
⬜	0. 238411–0. 2979
🟩	0. 297937–0. 3600
🟨	0. 360088–0. 4324
🟧	0. 432424–0. 5134
🟧	0. 513415–0. 6052
🟧	0. 605300–0. 7199
🟥	0. 719980–0. 8626
🟥	0. 862620–1. 0000

Diffusion Time: 100s

Figure 15. User interface of animation generation and display.

3. Experiments and Results

3.1. Study Area and Data

A rectangular area bordered by "Sangao Road", "Gangtou Road", "Daping Road", and "Lianjiang Road" was selected as the target region for atmospheric pollution dispersion simulation in Fuzhou, China. The area consists of typical components found in an urban block, including seven residential areas, two schools, one shopping mall, and one integrated service department (which is a large building that houses various services). The satellite image of the area and road network of the area can be seen in Figure 16a. The experimental data comprises vector data of buildings in Fuzhou city and each residential area in the simulation area. The building vector data attribute contains the number of floors of each building, and the residential areas vector data attribute includes the name of each residential area and its population, as shown in Figure 16b.

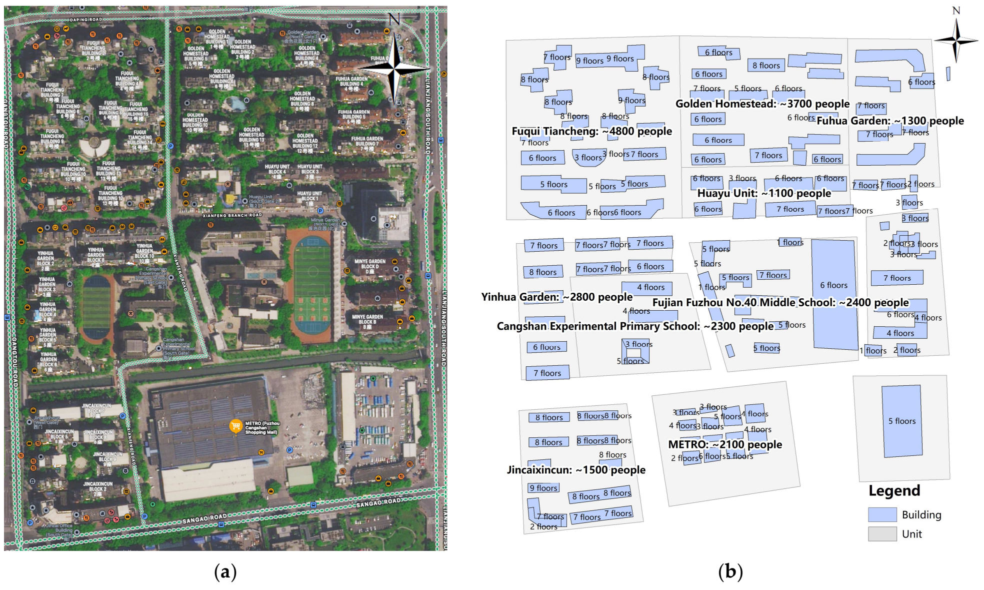

N
Fuqui Tiancheng: ~4800 people
Golden Homestead: ~3700 people
Fuhua Garden: ~1300 people
Huayu Unit: ~1100 people
Yinhua Garden: ~2800 people
Fujian Fuzhou No.40 Middle School: ~2400 people
Cangshan Experimental Primary School: ~2300 people
METRO: ~2100 people
Jincaixincun: ~1500 people
Legend
Building
Unit
(a)
(b)

Figure 16. (**a**) Simulated regional satellite imagery. The non-English terms in the figure refer to the Chinese translations of the English terms. (**b**) Vector data for the simulation area.

3.2. Comparison of The Efficiency of Coordinate Conversion Algorithm for Building Complex

This study focuses on a simulation area containing 116 buildings and 4237 bottom vertices, where the conversion of building vertices from geographic to plane coordinates was implemented using CPU serial, CPU parallel, and GPU parallel Gaussian projection algorithms. The CPU and GPU hardware-related parameters are shown in Table 3.

Table 3. Hardware-related parameter information of CPU and GPU.

Parameter	CPU	GPU
Model	Intel(R) Xeon(R) Gold 5228	NVIDIA GeForce RTX 2080 Ti

Other properties	Number of cores: 24;
	Number of logical processors: 48;
	L1 cache: 1.5 MB;
	L2 cache: 24.0 MB;
	L3 cache: 33.0 MB
		Maximum number of blocks in each dimension of the grid: 2,147,483,647, 65,535, 65,535;
		Maximum number of threads in each dimension of a block: 1024, 1024, 64

As shown in Figure 17, the CPU serial algorithm required 4162.576 s (69.4 min) to perform the forward projection, whereas the CPU parallel algorithm took only 314.871 s (5.2 min), indicating an efficiency improvement of almost 13.3 times. On the other hand, the GPU parallel algorithm took 170.277 s (2.3 min) to perform the forward projection, resulting in an efficiency improvement of almost 25 times compared to the CPU serial algorithm and twice as much compared to the CPU parallel algorithm. Overall, these results demonstrate a significant enhancement in efficiency.

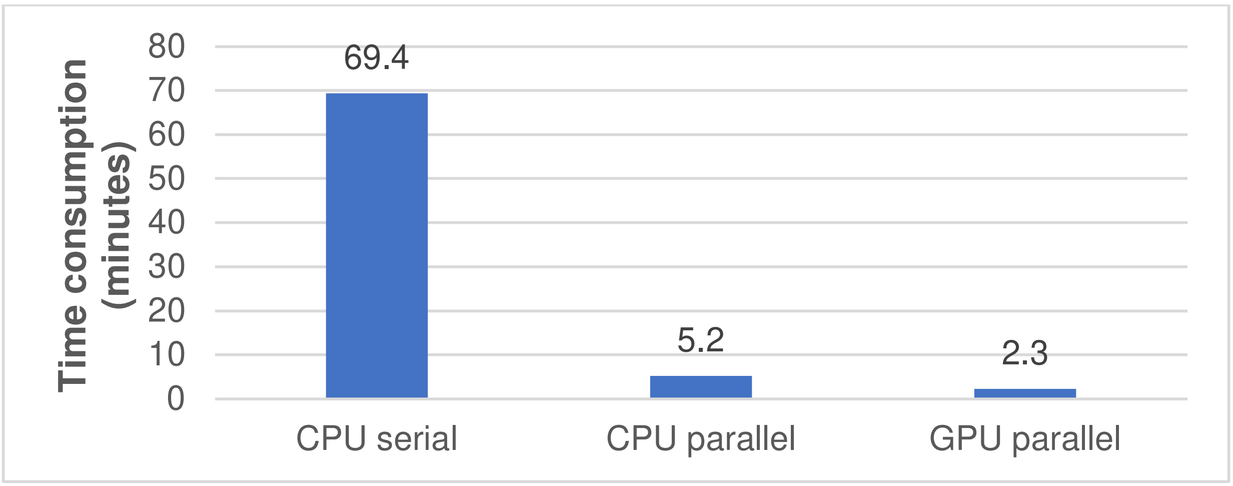


Figure 17. Time consumed by different algorithms.

3.3. Validation of Geometry Construction and Meshing

Based on the geometry construction and meshing approach proposed in this study, the generated command flow file can be executed to automatically construct the geometry of buildings within the simulation area and perform meshing on the generated geometry. The constructed buildings are shown in Figure 18, where the left figure is the building vector data from the experimental data, and the right figure is the automatically generated building geometry suitable for CFD. The comparison shows the capability of the geometry construction function implemented in our system.

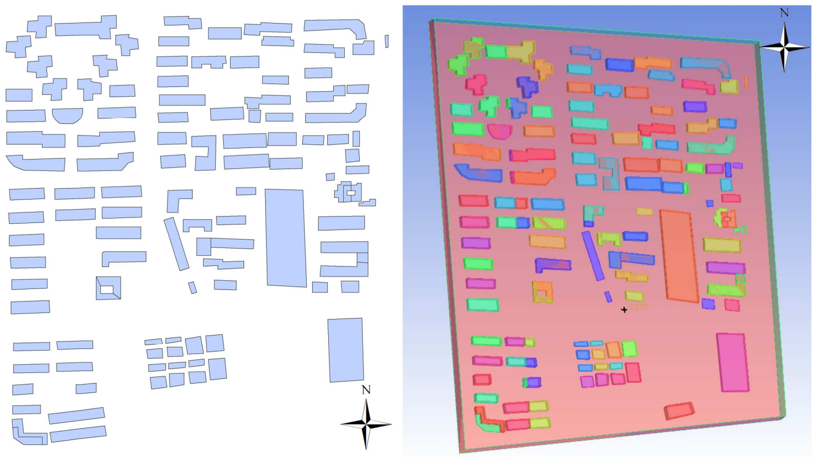


Figure 18. Validation of automatic building construction in the simulation area. The different colors in the figure are only used to distinguish different geometries.

The results of meshing and the bar chart of mesh quality metrics are shown in Figures 19 and 20, respectively. In this study, the mesh quality was evaluated using the Determinant metric, which is a measure of the distortion of each mesh element. The horizontal axis represents the quality of the mesh, which is calculated based on the determinant value of the Jacobian matrix, and the vertical axis represents the number of mesh elements with the corresponding mesh quality. Higher determinant values indicate better mesh quality. The bars with upward arrows indicate a higher number of elements. By clicking on a specific bar in the ICEM CFD software, the exact number of elements can be obtained. The bar chart shows that all mesh qualities are above 0.45, which meets the mesh quality requirement (0.3) of CFD simulation. Therefore, the proposed automatic meshing approach in this paper is reliable and effective.

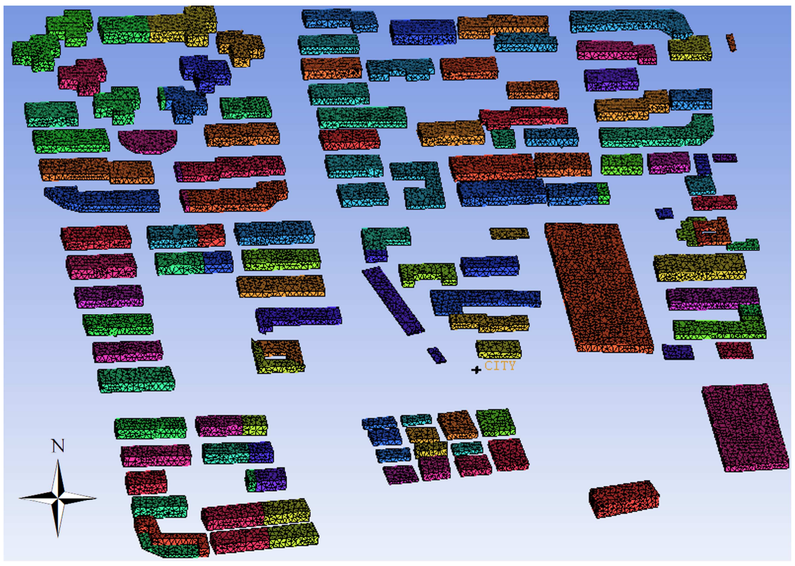


Figure 19. The results of unstructured meshing. The different colors in the figure are only used to distinguish different geometries.

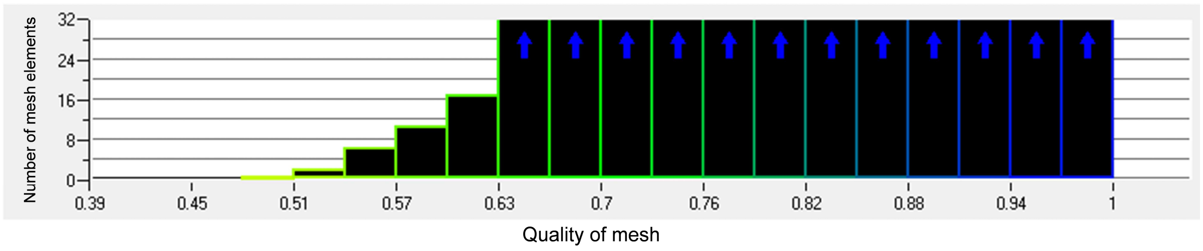


Figure 20. The bar chart of mesh quality metrics. The green outlined bars gradually transitioning to blue outlined bars represent an increasing level of mesh quality. The bars with upward arrows indicate that the number of mesh elements greatly exceeds the corresponding value on the vertical axis.

3.4. Case Study of Chlorine Dispersion Simulation

The correctness and usability of the automated geometry construction and meshing method proposed in this paper have been verified in the previous section. Therefore, in this section, we utilize the automatically constructed geometry of buildings in the simulation area to simulate the dispersion of chlorine. Assuming that, in the simulated area of Fuzhou City, at the intersection of "Sangao Road" and "Lianjiang South Road" near the Metro Cash & Carry (Fuzhou Cangshan Store), an accident occurred with a truck carrying a large amount of chlorine, causing the leakage and dispersion. The wind speed at the time of the accident was about 8 m/s, with a southeast wind direction, and the impact of chemical reactions is not considered. The simulation area and boundary surfaces are shown in Figure 21. The dimensions of the study area are as follows: the length of the inlet-S boundary is 555 m, the length of the inlet-E boundary is 596 m, the length of the outlet-W boundary is 684 m, and the length of the outlet-N boundary is 546 m. The height of the study area is set to the height of the tallest building within the area, which is 20 m in this case.

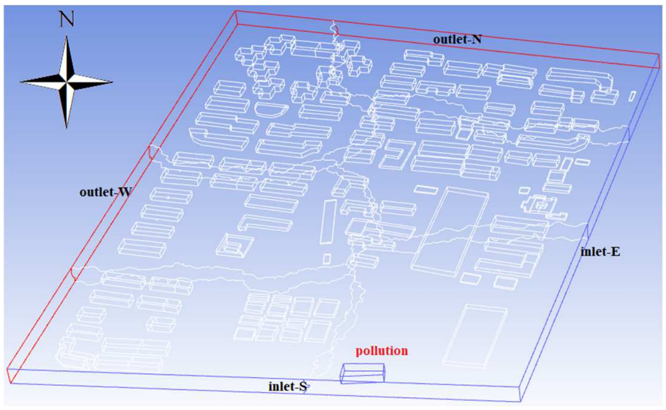


Figure 21. Simulation area and boundary. Inlet boundaries are indicated by blue and outlet boundaries are indicated by red.

3.4.1. Simulation Parameter Setting and Model Solving

To validate the effectiveness of the prototype system implementing the aforementioned methods, we focused on the system's performance, rather than the complexity of the CFD simulation. Therefore, we opted for simple parameter settings using the brief mode of the system. Specifically, the model is set up with the energy equation, standard k-ε turbulence model, and species transport model. Gravity is considered in the -Z direction with a magnitude of -9.81 m/s^2. The solver type is pressure-based with the solver velocity attribute set to absolute velocity, and the time option is set to transient. Chlorine was added to the materials as a fluid, and the mixture consists of two gases, air and chlorine. The inlet-S and inlet-E surfaces were set to the velocity-inlet boundary condition, while the pollution truck was set to mass-flow-inlet boundary condition. To simplify the flow field, we chose to specify constant values for important parameters, such as mass flux and velocity magnitude, instead of using profiles as boundary conditions. Specifically, the velocity magnitude of the inlet-S was set to 8 m/s with air as the only species, while the velocity magnitude of the inlet-E was set to 0.6 m/s with air as the only species. The mass flux of the pollution truck was set to 6 kg/(m^2-s) with chlorine gas as the only species. The turbulent intensity was set to 5% and the turbulent viscosity ratio was set to 10 for both the velocity-inlet and mass-flow-inlet. The outlet-W and outlet-N surface are set as outflow. The top and bottom surfaces of the study area and the building are set up as no-slip walls. The detailed parameters for the boundary conditions are shown in Table 4. The SIMPLEC pressure–velocity coupling scheme is selected as the solution method. The simulation results are saved as binary text files for the entire simulation area at each time step. The simulation runs for 120 s with a fixed time step of 1 s, and a maximum of 20 iterations are performed at each step.

Table 4. Boundary condition parameters for study area.

Zona Name	Type	Boundary Conditions

Pollution truck	mass-flow-inlet	Mass Flow Method: Mass Flux;
		Reference Frame: Absolute;
		Mass Flux: 6 kg/(m^2-s);
		Initial Gauge Pressure: 0;
		Direction method: Normal to Boundary;
		Temperature: 300 k

inlet-S, inlet-E	velocity-inlet	Velocity Method: Magnitude, Normal to boundary;
		Reference Frame: Absolute;
		Initial gauge Pressure: 0;
		Temperature: 300 k

outlet-W, outlet-N	outflow	Flow Rate Weighting: 1

top and bottom surface, building	wall	Wall Motion: Stationary;
		Shear Condition: No Slip;
		Roughness Models: Standard;
		Temperature: 300 k

Based on the approach proposed in Section 2.2, our system can automatically generate the command flow file for parameter setting described above and other related computing operations. The ANSYS Fluent software can be invoked in the background by our developed system to execute the command flow file and complete the simulation calculation. The residual plot after the simulation calculation is displayed in Figure 22, which shows relatively small iterative calculation errors for the continuity equation, energy equation, turbulence equation, velocity equation, and species transport model. Hence, the correctness of the simulation calculation in this case can be verified.

3.4.2. Analysis of Simulation Results

In this case, the state of chlorine dispersion was simulated for each second of the 120-s time period. Figure 23 shows the chlorine gas dispersion states at every 10-s interval, starting from the first second up to 100th second, at a height of 1.5 m on the horizontal section. The figure reveals that, as time progresses, the overall range of chlorine dispersion becomes increasingly larger. The high-concentration chlorine range (represented by the red area) rapidly expands in the first 50 s, but it reaches an equilibrium state after that under the prevailing conditions in the simulated area. This is due to the balance between the release and consumption of chlorine within this range, which is caused by the wind dispersing the chlorine and reducing its concentration. In contrast, the low and medium concentration chlorine ranges continue to expand, and their trend towards an equilibrium state is not evident. Therefore, timely actions should be taken during the early stages of chlorine leakage to prevent continuous leakage. Shortening the dispersion time can achieve a reduced range of low and medium concentrations of chlorine dispersion. Additionally, since the range of high-concentration chlorine will reach an equilibrium state quickly, its area should be quickly locked down, and rescue efforts should be focused on those within this range.

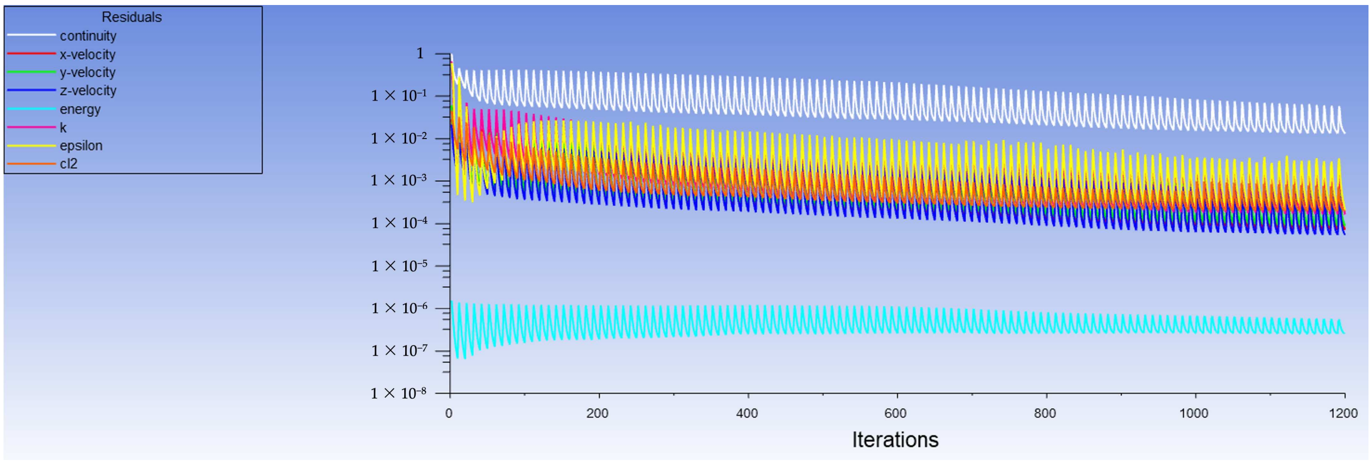


Figure 22. Residual plots of simulation.

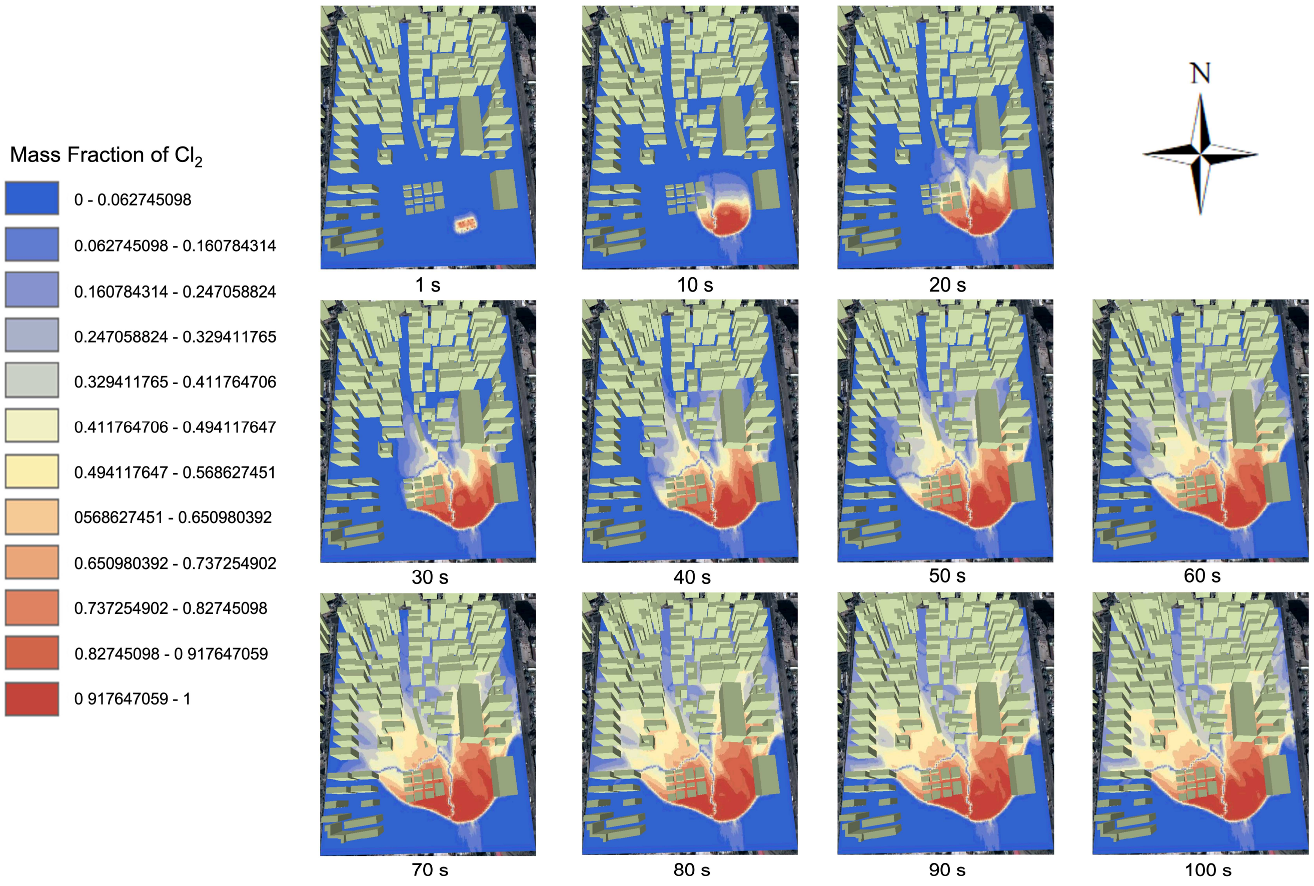


Figure 23. Chlorine dispersion every 10 s at the height of 1.5 m.

The spatiotemporal multidimensional visualization presented in Figure 23 was obtained through the following steps: (1) horizontal partitioning was performed on the simulation data at a height of 1.5 m using the horizontal partitioning method; (2) the chlorinous mass fraction field of the section data was converted into raster data using the raster interpolation method; (3) image animation visualization was used for visualization. In addition to applying the aforementioned methods to the simulation data, users can also perform other spatial analysis methods on the simulation results in the system, such as reclassifying the pollutant mass fraction based on pollution standards to classify the study area into regions of different pollution levels.

4. Discussion and Conclusions

Our team has been devoted to researching the application of GIS in atmospheric pollution accidents. For example, Huang et al. [45] integrated the atmospheric dispersion model CALPUFF with GIS, effectively integrating the numerical simulation results of the CALPUFF model into a three-dimensional terrain scene. They also realized the dynamic diffusion of model simulation results in the time dimension, as well as hierarchical classification in the spatial dimension. The disaster simulation information is data calculated by the model at a certain moment and a certain area, such as the concentration of pollutants and the probability of casualties. Tang et al. [46] organized the disaster information in GeoJSON format and designed a visualization method for sudden atmospheric pollution simulation disasters based on GeoJSON, achieving the visualization of atmospheric pollution disasters on mobile devices. However, we opted for a CFD model that is more suitable for simulating atmospheric pollution dispersion within urban blocks [11,12], rather than coupling GIS with a Gaussian model, and we successfully integrated CFD model outputs into GIS for spatiotemporal multidimensional visualization analysis, which effectively solved the issue of the lack of geospatial information in CFD model outputs. Furthermore, we introduced, for the first time, the tight coupling of GIS with CFD software ICEM CFD and Fluent, not only integrating the CFD model outputs into GIS, but also allowing users to rapidly construct the building complex geometry and meshing required for simulating pollution dispersion within urban blocks, which resolved the complexity and time-consuming nature of constructing the necessary geometry for CFD simulation of urban blocks. We also encapsulated the Fluent-based command flow, indirectly providing Fluent with secondary development APIs, and we achieved the automatic generation of model calculation command flow files to solve atmospheric pollution dispersion model, addressing the issue of Fluent's high entry barrier due to its complex parameters tailored to various fields. The specific research results are as follows:

(1) We propose an approach for automating the construction of geometry and meshing required for CFD simulations of urban blocks. Specifically, we design both CPU and CUDA-based GPU parallel algorithms to convert building vertex coordinates in spherical coordinate systems to Cartesian coordinates in the plane quickly. We also propose using a parametric design method to encapsulate geometry construction and meshing commands to achieve automatic and rapid construction of geometry and unstructured meshing. Through experiments on the study area, we validate that the constructed geometry is correct, and the mesh quality meets the requirements with all values above 0.45. Additionally, the CPU and GPU parallel algorithms are 13.3× and 25× faster than serial, respectively.

(2) We investigated CFD models related to atmospheric dispersion and proposed a parameterization design method for Fluent GUI command flow and an encapsulation method for TUI command flow, providing secondary development APIs for Fluent customization. By designing simple parameter interaction interfaces, users enabled solving CFD atmospheric dispersion models in urban blocks.

(3) We propose a spatio-temporal multidimensional visualization and analysis method based on three-dimensional GIS for simulation results. Specifically, we developed a method for converting CFD simulation results into three-dimensional GIS data and achieve the coupling of CFD simulation results with three-dimensional GIS. By using three-dimensional GIS attribute and spatial topological relationship queries, we enable multi-angle spatial partitioning of simulation results. We also propose two

methods for achieving spatiotemporal multidimensional visualization and animation of simulation results: layer visualization and image animation visualization.

(4) We integrated the above-mentioned methods to develop a system coupled GIS and CFD for atmospheric pollution dispersion simulation in urban blocks. This system provides a user-friendly and easy-to-use tool for relevant departments and researchers to simulate the dispersion of atmospheric pollutants in urban areas, as well as to easily explore deep-level information.

Due to the limitation of research conditions, the research in this paper has the following shortcomings:

(1) The geometry is constructed only considering the buildings within the urban blocks, but not the topography, public facilities and green vegetation of the urban blocks.
(2) The coupling of 3DGIS and CFD in this study is achieved through a tight integration of ArcGlobe and ANSYS software based on the .NET Framework technology framework. This involves converting GIS data into the required format for geometric construction in ICEM CFD and integrating Fluent simulation results back into GIS. All of thiese data are stored locally on the user's computer. Consequently, our approach requires local access and conversion of shared data, which reduces efficiency and necessitates the installation of third-party software, thereby making the environment configuration complex.. In the future, open-source GIS libraries and CFD libraries can be chosen to fully integrate 3DGIS and CFD based on the method proposed in this paper.
(3) For large simulation areas, the huge amount of vector point data generated by the simulation results can significantly decrease the efficiency of three-dimensional visualization, resulting in a poor user experience. In the future, it may be possible to explore the use of popular front-end two-dimensional/three-dimensional map engines, such as Cesium and Three.js, to enable three-dimensional visualization and related analysis of simulation results on the web, which could improve efficiency and user experience.

Author Contributions: Conceptualization and methodology, Q.W. and H.S.; Software and validation, H.S. and Y.W.; Investigation, H.L. and Z.Z.; Writing—original draft preparation, Q.W. and Y.W.; Writing—review and editing, Q.W., Y.W., H.L. and Z.Z.; Supervision, Q.W. and H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Key Project of Science and Technology of Fujian Province (2021H0036), the National Natural Science Foundation of China (42201500), and The Central Guided Local Development of Science and Technology Project (2022L3003).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Wu, C.; Wu, H.; Liu, B. Numeral Simulation of the Effects of Wind on Pipeline Natural Gas Leakage and Diffusion. *Ind. Saf. Environ. Prot.* **2013**, *39*, 46–49.
2. Yang, J. Diffusion of Natural Gas Concentration Field under Humidity Gradient. Master's Thesis, Chongqing Jiaotong University, Chongqing, China, 2014. Available online: https://kns.cnki.net/kcms2/article/abstract?v=3uoqIhG8C475KOm_zrgu4lQARvep2SAkbl4wwVeJ9RmnJRGnwiiNVnB5v6Bh-uVuVe-OXi2WINw7RONPKDGVl5QT9n3mabxH&uniplatform=NZKPT (accessed on 11 April 2023).
3. Cormier, B.R.; Qi, R.; Yun, G.; Zhang, Y.; Sam Mannan, M. Application of Computational Fluid Dynamics for LNG Vapor Dispersion Modeling: A Study of Key Parameters. *J. Loss Prev. Process Ind.* **2009**, *22*, 332–352. [CrossRef]
4. Tauseef, S.M.; Rashtchian, D.; Abbasi, S.A. CFD-Based Simulation of Dense Gas Dispersion in Presence of Obstacles. *J. Loss Prev. Process Ind.* **2011**, *24*, 371–376. [CrossRef]
5. Yoshie, R.; Jiang, G.; Shirasawa, T.; Chung, J. CFD Simulations of Gas Dispersion around High-Rise Building in Non-Isothermal Boundary Layer. *J. Wind Eng. Ind. Aerodyn.* **2011**, *99*, 279–288. [CrossRef]

6. Siddiqui, M.; Jayanti, S.; Swaminathan, T. CFD Analysis of Dense Gas Dispersion in Indoor Environment for Risk Assessment and Risk Mitigation. *J. Hazard. Mater.* **2012**, *209–210*, 177–185. [CrossRef]
7. Han, G. Research on the Leakage of Buried Gas Pipeline and Diffusion Regularity. Master's Thesis, Chongqing University, Chongqing, China, 2014. Available online: https://kns.cnki.net/kcms2/article/abstract?v=3uoqIhG8C475KOm_zrgu4lQARvep2SAkbl4wwVeJ9RmnJRGnwiiNVmSXvA452kamwJJZqoomuqhF3Ht0QXL8gR67RqjCspBj&uniplatform=NZKPT (accessed on 11 April 2023).
8. Fu, J.; Xiong, X.; Li, Y.; Wei, C.; Wang, Q. Numerical Study on Leakage and Diffusion of Buried High Sulfur Natural Gas Pipelines. *Contemp. Chem. Ind.* **2014**, *43*, 1923–1926. [CrossRef]
9. Feißel, T.; Büchner, F.; Kunze, M.; Rost, J.; Ivanov, V.; Augsburg, K.; Hesse, D.; Gramstat, S. Methodology for Virtual Prediction of Vehicle-Related Particle Emissions and Their Influence on Ambient PM10 in an Urban Environment. *Atmosphere* **2022**, *13*, 1924. [CrossRef]
10. Schalau, S.; Habib, A.; Michel, S. A Modified K-ε Turbulence Model for Heavy Gas Dispersion in Built-Up Environment. *Atmosphere* **2023**, *14*, 161. [CrossRef]
11. Sabatino, S.D.; Buccolieri, R.; Pulvirenti, B.; Britter, R.E. Flow and Pollutant Dispersion in Street Canyons Using FLUENT and ADMS-Urban. *Environ. Model. Assess.* **2008**, *13*, 369–381. [CrossRef]
12. Toja-Silva, F.; Chen, J.; Hachinger, S.; Hase, F. CFD Simulation of CO_2 Dispersion from Urban Thermal Power Plant: Analysis of Turbulent Schmidt Number and Comparison with Gaussian Plume Model and Measurements. *J. Wind Eng. Ind. Aerodyn.* **2017**, *169*, 177–193. [CrossRef]
13. Yuan, X.; Chen, C.; Lei, X.; Yuan, Y.; Muhammad Adnan, R. Monthly Runoff Forecasting Based on LSTM–ALO Model. *Stoch. Environ. Res. Risk Assess.* **2018**, *32*, 2199–2212. [CrossRef]
14. Adnan, R.M.; Mostafa, R.R.; Elbeltagi, A.; Yaseen, Z.M.; Shahid, S.; Kisi, O. Development of New Machine Learning Model for Streamflow Prediction: Case Studies in Pakistan. *Stoch. Environ. Res. Risk Assess.* **2022**, *36*, 999–1033. [CrossRef]
15. Adnan, R.M.; Mostafa, R.; Kisi, O.; Yaseen, Z.M.; Shahid, S.; Zounemat-Kermani, M. Improving Streamflow Prediction Using a New Hybrid ELM Model Combined with Hybrid Particle Swarm Optimization and Grey Wolf Optimization. *Knowl.-Based Syst.* **2021**, *230*, 107379. [CrossRef]
16. Adnan, R.M.; Kisi, O.; Mostafa, R.R.; Ahmed, A.N.; El-Shafie, A. The Potential of a Novel Support Vector Machine Trained with Modified Mayfly Optimization Algorithm for Streamflow Prediction. *Hydrol. Sci. J.* **2022**, *67*, 161–174. [CrossRef]
17. Ikram, R.M.A.; Ewees, A.A.; Parmar, K.S.; Yaseen, Z.M.; Shahid, S.; Kisi, O. The Viability of Extended Marine Predators Algorithm-Based Artificial Neural Networks for Streamflow Prediction. *Appl. Soft Comput.* **2022**, *131*, 109739. [CrossRef]
18. Wu, Q.; Lin, H. A Novel Optimal-Hybrid Model for Daily Air Quality Index Prediction Considering Air Pollutant Factors. *Sci. Total Environ.* **2019**, *683*, 808–821. [CrossRef]
19. Li, X.; Luo, A.; Li, J.; Li, Y. Air Pollutant Concentration Forecast Based on Support Vector Regression and Quantum-Behaved Particle Swarm Optimization. *Environ. Model. Assess.* **2019**, *24*, 205–222. [CrossRef]
20. Niu, M.; Zhang, Y.; Ren, Z. Deep Learning-Based PM2.5 Long Time-Series Prediction by Fusing Multisource Data—A Case Study of Beijing. *Atmosphere* **2023**, *14*, 340. [CrossRef]
21. Ong, B.T.; Sugiura, K.; Zettsu, K. Dynamically Pre-Trained Deep Recurrent Neural Networks Using Environmental Monitoring Data for Predicting PM2.5. *Neural Comput. Appl.* **2016**, *27*, 1553–1566. [CrossRef]
22. Esager, M.W.M.; Ünlü, K.D. Forecasting Air Quality in Tripoli: An Evaluation of Deep Learning Models for Hourly PM2.5 Surface Mass Concentrations. *Atmosphere* **2023**, *14*, 478. [CrossRef]
23. Zhang, Z.; Ren, J.; Chang, Y. Improving Intra-Urban Prediction of Atmospheric Fine Particles Using a Hybrid Deep Learning Approach. *Atmosphere* **2023**, *14*, 599. [CrossRef]
24. Li, X.; Peng, L.; Yao, X.; Cui, S.; Hu, Y.; You, C.; Chi, T. Long Short-Term Memory Neural Network for Air Pollutant Concentration Predictions: Method Development and Evaluation. *Environ. Pollut.* **2017**, *231*, 997–1004. [CrossRef]
25. Kuremoto, T.; Kimura, S.; Kobayashi, K.; Obayashi, M. Time Series Forecasting Using a Deep Belief Network with Restricted Boltzmann Machines. *Neurocomputing* **2014**, *137*, 47–56. [CrossRef]
26. Wang, B.; Qian, F. Three Dimensional Gas Dispersion Modeling Using Cellular Automata and Artificial Neural Network in Urban Environment. *Process Saf. Environ. Prot.* **2018**, *120*, 286–301. [CrossRef]
27. Huang, C.-J.; Kuo, P.-H. A Deep CNN-LSTM Model for Particulate Matter (PM2.5) Forecasting in Smart Cities. *Sensors* **2018**, *18*, 2220. [CrossRef]
28. Pak, U.; Kim, C.; Ryu, U.; Sok, K.; Pak, S. A Hybrid Model Based on Convolutional Neural Networks and Long Short-Term Memory for Ozone Concentration Prediction. *Air Qual. Atmos. Health* **2018**, *11*, 883–895. [CrossRef]
29. Cao, D. The Secondary Development of Fluent and Research and Application in Computer-Aided Optimization and Design of Pump about It. Master's Thesis, North China Electric Power University, Beijing, China, 2002. Available online: https://kns.cnki.net/kcms2/article/abstract?v=3uoqIhG8C475KOm_zrgu4m9eu-VXu9H75RhMZCEMue9h8LplqMYx9zGVlbBb7nVMKWq_bvhtNu58msc-qJ3_mXCXmN6p2uBD&uniplatform=NZKPT (accessed on 11 April 2023).
30. Zhang, Z.; Gao, B.; Zheng, C.; Wang, Y. Analysis of Transient Pressure Induced by High-Speed Train Passing a Tunnel Based on the Second Development of Fluent. *J. Railw. Eng. Soc.* **2005**, 41–44, 54.
31. Li, H.; Bi, X.; Huang, X.; Suo, W. Shape Parametric Constructing Model of Radiator Core Based on the Secondary Development of Fluent. *Veh. Power Technol.* **2008**, 36–39, 43. [CrossRef]

32. Li, H.; Wang, G.; Wang, D.; Tan, B. Journal File of the Secondary Development of the Fluent Based on the Ventilating System of the Subway. *Comput. Syst. Appl.* **2015**, *24*, 233–238.
33. Xiao, H.; Gao, C.; Dang, Y.; Yang, Y.; Wang, G. Secondary Development of FLUENT and Application in Numerical Simulation of Aerodynamic Characteristics for Rockets. *Aeronaut. Comput. Tech.* **2009**, *39*, 55–57.
34. Li, X.; Lu, J.; Tan, S.; Li, K.; Du, P. Simulation on Moving Magnetic Field of Electromagnetic Rail launch Based on Fluent Secondary Development. *Proc. CSEE* **2020**, *40*, 6364–6371. [CrossRef]
35. Murakami, S. Environmental Design of Outdoor Climate Based on CFD. *Fluid Dyn. Res.* **2006**, *38*, 108–126. [CrossRef]
36. Chu, A.K.M.; Kwok, R.C.W.; Yu, K.N. Study of Pollution Dispersion in Urban Areas Using Computational Fluid Dynamics (CFD) and Geographic Information System (GIS). *Environ. Model. Softw.* **2005**, *20*, 273–277. [CrossRef]
37. Coirier, W.J.; Kim, S. CFD Modeling for Urban Area Contaminant Transport and Dispersion: Model Description and Data Requirements. In Proceedings of the Sixth Symposium on the Urban Environment, The 86th AMS Annual Meeting, Atlanta, GA, USA, 2 February 2006.
38. Wong, D.W.; Camelli, F.; Sonwalkar, M. Integrating Computational Fluid Dynamics (CFD) Models with GIS: An Evaluation on Data Conversion Formats. In *Geoinformatics 2007: Geospatial Information Science*; SPIE: Bellingham, WA, USA, 2007; Volume 6753, pp. 368–378. [CrossRef]
39. van Hooff, T.; Blocken, B. Coupled Urban Wind Flow and Indoor Natural Ventilation Modelling on a High-Resolution Grid: A Case Study for the Amsterdam ArenA Stadium. *Environ. Model. Softw.* **2010**, *25*, 51–65. [CrossRef]
40. Zheng, M.; Jin, M.; Guo, f. Modeling and Simulation of Toxic Gas Dispersion in Urban Streets Supported by GIS. *Geomat. Inf. Sci. Wuhan Univ.* **2013**, *38*, 935–939. [CrossRef]
41. Xu, Y. Application Research of Atmospheric Pollution in the City Coupled with CFD Software and Geographic Information System (GIS). Master's Thesis, Chongqing Jiaotong University, Chongqing, China, 2014. Available online: https://kns.cnki.net/kcms2/article/abstract?v=3uoqIhG8C475KOm_zrgu4lQARvep2SAkbl4wwVeJ9RmnJRGnwiiNVoWUqW9aqtc51NN9UurKIfZzT152-rm_NjKmlPFLP6yC&uniplatform=NZKPT (accessed on 8 March 2023).
42. Jiao, X. Application Research of Pollutant Diffusion of Traffic in the City Coupled with CFD Software and Geographic Information System(GIS). Master's Thesis, Chongqing Jiaotong University, Chongqing, China, 2016. Available online: https://kns.cnki.net/kcms2/article/abstract?v=3uoqIhG8C475KOm_zrgu4lQARvep2SAkkyu7xrzFWukWIylgpWWcEoXyvc0-RZfX-EKX-ZhSaLva1axFC5O_ABrlQpUWkqcN&uniplatform=NZKPT (accessed on 8 March 2023).
43. Chang, S.; Jiang, Q.; Zhao, Y. Integrating CFD and GIS into the Development of Urban Ventilation Corridors: A Case Study in Changchun City, China. *Sustainability* **2018**, *10*, 1814. [CrossRef]
44. Wang, W.; Chen, H.; Wang, X.; Du, S.; Xv, T. Application of GIS and CFD Software in Sind Farm. *Hydropower New Energy* **2020**, *34*, 29–32. [CrossRef]
45. Wu, Q.; Huang, J.; Sheng, L.; Tan, S.; Wang, Q.; Sun, Z. Spatio-temporal and Multi-dimensional Visualizations for the Simulation Result of CALPUFF Model. *J. Geo-Inf. Sci.* **2015**, *17*, 206–214.
46. Wu, Q.; Tang, S.; Huang, J. A GeoJSON-based mobile visualization method for emergency air pollution simulation disaster condition. *J. Nat. Disasters* **2015**, *24*, 165–171. [CrossRef]