Received January 22, 2022, accepted March 5, 2022, date of publication March 8, 2022, date of current version March 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3157823

Fast 3D Visualization of Massive Geological Data Based on Clustering Index Fusion

YU-HANG ZHANG[1,2,3], CHANG WEN[1,3], MIN ZHANG[2,3,4], KAI XIE[2,3,4], AND JIAN-BIAO HE[5]

[1]School of Computer Science, Yangtze University, Jingzhou 434023, China
[2]National Demonstration Center for Experimental Electrical & Electronic Education, Yangtze University, Jingzhou 434023, China
[3]Western Institute, Yangtze University, Karamay 834000, China
[4]School of Electronic Information, Yangtze University, Jingzhou 434023, China
[5]School of Computer Science and Engineering, Central South University, Changsha 410083, China

Corresponding author: Chang Wen (wenchang2016paper@163.com)

This work was supported in part by the National Natural Science Foundation of Xinjiang Uygur Autonomous Region under Grant 2020D01A131; in part by the Fund of Hubei Ministry of Education under Grant B2019039; in part by the Graduate Teaching and Research Fund of Yangtze University under Grant YJY201909; in part by the Teaching and Research Fund of Yangtze University under Grant JY2019011; in part by the Undergraduate Training Programs for Innovation and Entrepreneurship of Yangtze University under Grant Yz2020057, Grant Yz2020059, and Grant Yz2020156; and in part by the National College Student Innovation and Entrepreneurship Training Program under Grant 202110489003.

ABSTRACT With the development of $3D$ visualization technology, the amount of geological data information is increasing, and the interactive display of big data faces severe challenges. Because traditional volume rendering methods cannot entirely load large-scale data into the memory owing to hardware limitations, a visualization method based on variational deep embedding clustering fusion Hilbert R-tree is proposed to solve slow display and stuttering issues when rendering massive geological data. By constructing an efficient data index structure, deep clustering algorithms and space-filling curves can be integrated into the data structure to improve the indexing efficiency. In addition, this method combines time forecasting, data scheduling, and loading modules to improve the accuracy and real-time data display rate, thereby improving the stability of $3D$ visualization of large-scale geological data. This method uses real geological data as the experimental dataset, comparing and analyzing the existing index structure and time-series prediction method. The experimental results indicate that when comparing the index of the variational deep embedded clustering-Hilbert R-tree ($VDEC-HRT$) with that of the K-means Hilbert R-tree ($KHRT$), the time required is reduced by 55.67%, the viewpoint prediction correctness of the proposed method is improved by 22.7% compared with Lagrange interpolation algorithm. And the overall rendering performance and quality of the system achieve the expected results. Ours experiments prove the feasibility and effectiveness of the proposed scheme in the visualization of large-scale geological data.

INDEX TERMS $3D$ visualization of massive data, deep learning, Hilbert R-tree, deep clustering, time-series forecasting, view frustum culling.

I. INTRODUCTION

Three-dimensional visualization technology has always been an indispensable part of the development of computer graphics. It is an effective method for multi-dimensional presentation and the analysis of data objects in various industries, such as medicine, remote sensing, geology, and oil and gas exploration [1]–[4]. Among them, the volume-rendering algorithm [5]–[7] is applied to geological exploration, which can clearly depict the internal level of detailed information and characteristics of the geological body and provides a data research platform for researchers in related fields.

However, when dealing with a variety of data, the development of $3D$ visualization faces several challenges. For example, traditional volume rendering algorithms are more complex and require a large amount of memory space, necessitating relatively advanced computer hardware. In addition, for large-scale volume data, the calculation speed is slow, and there arise delays in the response time of interactive displays, and lags during browsing, among other issues. Therefore, several $3D$ visualization solutions have certain limitations.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhanyu Ma.

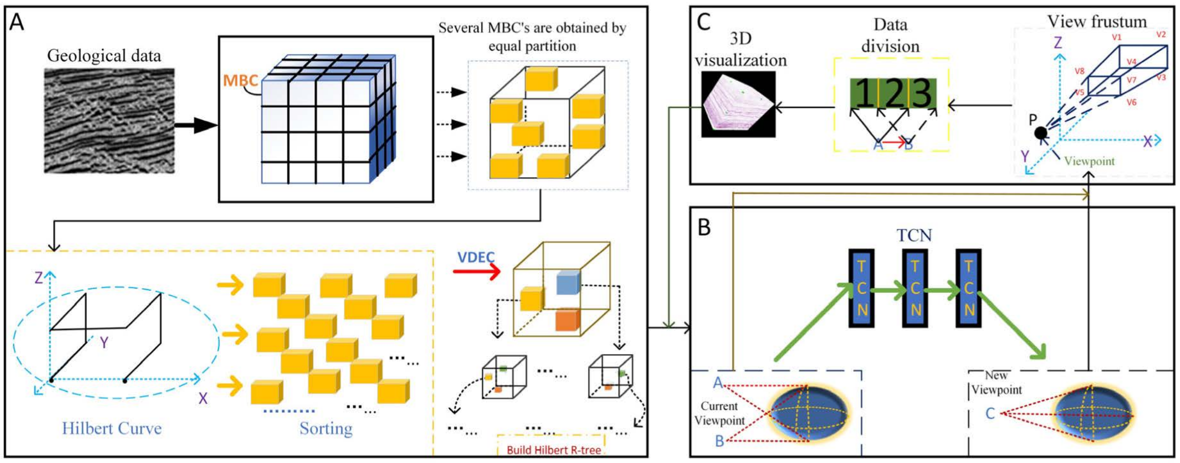

FIGURE 1. Flow of proposed algorithm. (A) Read the data, determine the range of the data body in the space, determine the size of an MBC, divide the original data body equally to obtain several MBCs, calculate the Hilbert code values of all MBC center points through the Hilbert curve formula, sort them, and finally complete the global clustering through VDEC operation. (B) TCN is used to predict the position of the next view point, so as to avoid problems such as picture stuck jump caused by centralized loading when the amount of data is too large. (C) Finally realize geological data visualization through visual cone clipping and volume data division.

Currently, a major problem of the 3*D* visualization process is that large-scale data cannot be displayed quickly and with high-quality using traditional volume rendering technology. Many related optimization solutions have been proposed. The literature [8]–[15] proposes a modified R-tree structure to improve query efficiency. The literature [16]–[18] further optimizes the structure of the Hilbert R-tree based on the R-tree; however, this method has shortcomings in the processing of large-scale data. The main problem is that when mapping with a spatial filling curve, significant overlapping space is generated in the process of building a tree structure, thus affecting the efficiency of data retrieval. Considering the problems of Hilbert R-tree in the literature [19], [20] with respect to clustering, the coverage overlap between nodes can be reduced to a certain extent by clustering, thereby enabling the formation of a compact and efficient data structure. In addition to solving the problem of fast visualization regarding data structure, there are also some solutions to address the display lag when the data object is large. The Lagrangian interpolation algorithm is used to predict the trajectories of viewpoints, and different interpolation steps are set to determine the final prediction accuracy and rendering effect. Moreover, the deep learning model [21]–[26] is also used to predict the trajectory of viewpoints, which improves the accuracy of prediction. The literature [27]–[30] introduces the visual cone clipping algorithm in the real-time rendering of large-scale terrain, which performs the necessary clipping of data objects according to the change in the viewpoint range to load data objects quickly and accurately. In addition, level-of-detail (*LOD*) technology [31]–[34] reduces the detailed information of data according to viewpoint position and object distance, thus improving the rendering efficiency.

To this end, this study proposes a fast 3*D* visualization method based on deep clustering for use in massive geological data research. Firstly, in terms of data structure, deep clustering is adopted to reduce the partial overlap of tree structure, improving the efficiency of the overall index traversal for the data. Moreover, through the construction of a deep learning model under the time-series model to realize accurate viewpoint location prediction, and in combination with an improved data scheduling scheme to accelerate the volume rendering efficiency, this strategy allows for reduction in the complexity of the operation, load, and rendering of potential data in advance, and avoids sluggish browsing. This technique results in improved efficiency of the visualization in several ways to achieve rapid 3*D* visualization of massive geological data. The experiments prove that the proposed scheme has adequate feasibility and research value.

II. METHOD

This study proposes a method of deep clustering combined with an data structure using deep learning to predict the trajectory of the viewpoint, as well as the application of improved field-of-view removal technology in the rapid 3*D* visualization of massive geological data. As shown in *Figure* 1, the algorithm for the rapid 3*D* visualization of data is mainly divided into three modules: a) the establishment of an efficient data index structure, b) the prediction of the motion trajectory of the viewpoint, and c) the scheduling and loading of volume data in the field-of-view removal technology.

First, the geological data file is read, the original geological data format is mapped onto the 3*D* structure of space, and the sub-blocks with the smallest boundary cube are divided. Then, mutual information maximization is applied to clearly distinguish the samples from these datasets, using the Hilbert curve to reduce the dimensionality and the variational deep embedded clustering (*VDEC*) to perform the clustering operation, considering the center after clustering. The value of the code builds a Hilbert R-tree. Furthermore, the next module is entered to determine the coordinates of the current viewpoint position. This process has two branches: the first to use the frustum model to crop and render the spatial data corresponding to the current viewpoint position, and the second to perform a temporal convolutional network(TCN) prediction based on the current viewpoint position. The position coordinates of the next viewpoint are predicted by the frustum model, and the potential data are unloaded and rendered by comparing the divided regions. These new viewpoint coordinates are re-determined to draw the next frame.

The following subsections introduce the relevant algorithms of each module.

A. INDEX STRUCTURE OF DATA

Considering the inefficiency of the index caused by the scaling up of data, it is insufficient to only improve only the hardware. More importantly, it is more efficient to solve the rapid $3D$ visualization of massive geological data by improving the algorithm and combining these software and hardware optimizations. In this study with the index of the module using the Hilbert R-tree structure, due to connecting the Hilbert space-filling curve in a particular way and passing through the data in high dimensional space, the goal is to encode the location coordinates sorting, straight for one-dimensional data and then finding adjacent elements with a minimum bounding rectangle box, node up, and framed space is larger, to form the Hilbert R-tree. However, when the Hilbert R-tree structure is applied to large-volume data, the spatial overlap of nodes will also occur. In other words, within the same leaf node, spatial data objects do not originally belong to the same category, but the adjacent data code values after spatial curve transformation, which forms a clustering problem. For large-scale datasets, it may be considered to first gather the data with relatively close structure using a reasonable algorithm to avoid the misclassification of adjacent code values after subsequent conversion, and for quickly index.

1) VARIATIONAL DEEP EMBEDDING CLUSTERING

To optimize the indexing technology, this study uses the $VDEC$. Compared with the traditional clustering algorithm, on the one hand, it uses the variational autoencoder (VAE) [35], [36] to reflect the input data distribution with an unsupervised algorithm. On the other hand, it learns the feature representation and cluster assignment of data into the latent variable space through deep embedding clustering (DEC), and iteratively optimizes the target, thereby improving the performance of clustering.

The VAE can approximate the true high-dimensional distribution of complex data using unsupervised algorithms and can reconstruct the data characteristics of the hidden variable space. First given the dataset $X = \{x_1, x_2, \ldots, x_n\}\epsilon R^{D\times n}$, after the prior distribution $p(z)$ of the latent feature space, the latent variable space is generated as z, and then reconstructed by $p(x|z)$ to generate $\tilde{x}$. We denote the weight and bias of the encoder as model $\emptyset$, and the weight and bias of the decoder as model θ. DEC uses autoencoders as a network architecture and clustering allocation to reinforce losses as specifications. Network parameters are initialized through the autoencoder, and a two-layer neural network is defined as:

$$\tilde{x} \sim Dropout\,(x), \quad h \sim b_1(W_1\tilde{x} + u_1)$$
$$\tilde{h} \sim Dropout\,(h), \quad y \sim b_2(W_2\tilde{h} + u_2)$$

$Dropout()$ means that the dimension of input data of any set part is 0, b represents the activation function of the encoder, and the model parameter $\emptyset$ is W_1, u_1, W_2, u_2. First, the training model of the compressed data part of the encoder is collected by rebuilding the loss of the auto-encoder and discarding the decoder part. For a given dataset $X = \{x_1, x_2, \ldots, x_n\}\epsilon R^{D\times n}$, the corresponding feature $Z_\emptyset$ can be obtained through the initial mapping $f_\emptyset$ between data space and feature space, then the algorithm iteratively improves the clustering by minimizing the KL divergence between the soft label distribution and the auxiliary target distribution. In order to effectively improve the clustering performance, the sample can be clearly distinguished from the entire dataset, and the concept of mutual information maximization [37] is introduced in the VAE to identify the most unique information of the sample.

$$I\,(x, z) = D_{kl}[p(z|x)\check{p}(x)||p(z)\check{p}(x)] \tag{1}$$

Here, $\check{p}(x)$ is the distribution of the original sample; the larger the KL divergence between $p\,(z\,|\,x)\,\check{p}\,(x)$ and $p\,(z)\,\check{p}\,(x)$, the higher the correlation between x and z. This means that for each data point x, the encoder $p(z|x)$ can encode a unique z, and the optimization goal of the feature encoder is to maximize mutual information:

$$p(z|x) = argmax[I(x, z)] \tag{2}$$

Because it is difficulty to calculate the posterior distribution, the approximate posterior distribution $q_\phi(z|x)$ is introduced to estimate the true posterior distribution $p(x|z)$, which only needs to minimize $KL[q_\phi(z|x)||p(z|x)]$:

$$\begin{aligned} &D_{kl}[q_\phi(z|x)||p(x|z)] \\ &\quad = \iint q_\phi\,(z\,|\,x)\check{p}\,(x)\,log\frac{q_\phi(z|x)}{p\,(z|x)}dxdz \end{aligned} \tag{3}$$

From equations (2) and (3), we can obtain the total minimization goal of the encoder as

$$\begin{aligned} p\,(z\,|\,x) = MIN\{&\iint \check{p}\,(x)[q_\phi\,(z\,|\,x)\,log\frac{q_\phi(z|x)}{p\,(z|x)} \\ &-p\,(z\,|\,x)\,log\frac{p\,(z|x)}{p\,(z)}]dxdz\} \end{aligned} \tag{4}$$

The data that maximizes the lower bound of the target log-likelihood change can be obtained as:

$$logP_{target}(x) = L_{\theta,\emptyset} + D_{kl}[q_\phi(z|x)||p(z|x)] \tag{5}$$

using the Bayesian formula:

$$p(z|x) = \frac{p(z)p(x|z)}{p(x)} \tag{6}$$

Using the non-negativity of KL divergence, we can obtain:

$$D_{kl}[q_\phi(z|x)||p(z|x)] \\ = logp(x) + E[logq_\phi(z|x) - logp(z) - logp_\theta(x|z)] \quad (7)$$

and depending on the following,

$$logP_{target}(x) - D_{kl}[q_\phi(z|x)||p(z|x)] \\ = E[logq_\phi(z|x)] - D_{kl}[q_\phi(z|x)||p(z)] \quad (8)$$

$$=> logP_{target}(x) \geq E[logp_\theta(x|z)] \\ - D_{kl}[q_\phi(z|x)||p(z)] \quad (9)$$

$$=> L_{VAE}(\theta, \emptyset) = \sum\nolimits_i^n E_{x\sim q_\emptyset(z|x_i)}[logp_\theta(x_i|z)] \\ - \sum\nolimits_i^n D_{kl}[q_\phi(z|x_i)||p(z)] \quad (10)$$

Define z to obey a normal distribution:

$$p(z) \sim N(z; 0, t) \quad (11)$$

The variational self-encoding network VAE is used to output the two dimensions of the parameter mean μ and the variance σ^2 vector in the hidden layer:

$$q_\phi(z|x) \sim N(z; \mu, \sigma^2 t) \quad (12)$$

The heavy parameter technique is used to sample in the distribution space Z of the latent variables to obtain z:

$$z = \mu + \sigma * \varepsilon, \quad \varepsilon \sim N(0, 1) \quad (13)$$

The sampled z is input into the generation model $p(x|z)$ to generate a new $\tilde{x}$, and the KL divergence is obtained between $q_\phi\,(z\,|\,x)$ and the prior distribution $p\,(z)$ as the loss of the coding model:

$$L_{q_\emptyset} = \frac{1}{2}\sum\nolimits_{j=1}^{J}(1 - \log\left(\sigma^2\right) + \left(\mu_j^2\right) + (\sigma_j^2)) \quad (14)$$

For the generative model, the reconstruction error of the decoder is defined as its loss:

$$L_{p_\theta} = ||x_i - \tilde{x}||^2 \quad (15)$$

The coding part of the autoencoder is trained as the learned feature information z, and the similarity between it and the cluster center is computed. We define the similarity and target distribution as follows, where j is the dimension of the latent variable space:

$$q_{ij} = \frac{(1 + ||z_i - c_j||^2)^{-1}}{\sum_j (1 + ||z_i - c_j||^2)^{-1}} \quad (16)$$

and

$$p_{ij} = \frac{q_{ij}^2 \Big/ \sum_i q_{ij}}{\sum_j q_{ij}^2 \Big/ \sum_i q_{ij}} \quad (17)$$

In addition, we define the loss function of the target variable $P$and the similarity variable Q as

$$L_c = \sum_{i=1}^{n}\sum_{j=1}^{k} p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (18)$$

According to the similarity between the feature representation z and the cluster center, as in formula (16), the cluster label of x is obtained as follows:

$$S_i = \arg\max_j q_{ij} \quad (19)$$

We use the gradient descent iterative method to optimize the overall objective function:

$$L = L_{p_\theta} + L_{q_\emptyset} + \gamma L_c \quad (20)$$

2) VARIATIONAL DEEP EMBEDDING CLUSTERING FUSION HILBERT R-TREE

The specific structural process pseudocode of the variational deep embedding clustering fusion Hilbert R-tree ($VDEC - HRT$) is shown in Algorithm 1:

B. PREDICTION OF VIEWPOINT MOVEMENT TRAJECTORY

A certain amount of data can be directly loaded into memory; however, when the data size increases, only essential data must be loaded, and the redundant data must be blocked. If the data range can be predicted in advance, it is loaded into the memory for rendering and the amount of data loaded during memory access time can be avoided, making the image smoother and more stable.

The viewpoint prediction module in this study accurately predicts the expected position of future viewpoints according to the continuously changing viewpoint positions and perspectives, which is a typical time-series prediction problem. In this study, a time series convolution network is used for prediction, and its basic network structure consists of the following three parts:

1) CAUSAL CONVOLUTION

A one-way time-constrained structure, implying that the convolution operation at the current moment c is only based on the information before and at the historical moment $c - 1$. The structure is shown in *Figure* 2.

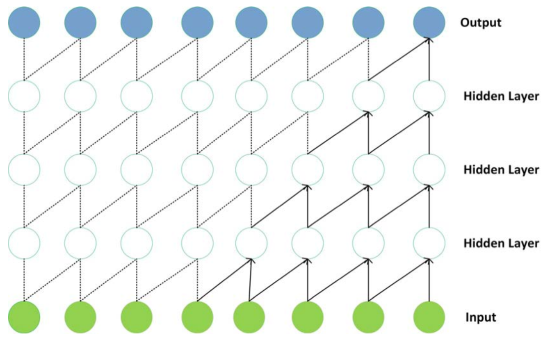

FIGURE 2. Causal convolution structure diagram.

2) EXPANSION CONVOLUTION

This mainly solves the problem of the causal convolution having too many stacked layers and is limited by the convolution kernel size. Its structure is shown in *Figure* 3, where D represents the cavity coefficients 1, 2, and 4, and k is the convolution kernel size.

Algorithm 1 Algorithm Flow of Constructing the Index Structure

Algorithm 1: $VAE - DEC$ fusion Hilbert R-tree index structure:

```
1: Determine the overall smallest boundary cube ($MBC$) according to the size occupied by the target object in the space and divide it equally to obtain n $MBC$ sub-blocks
2:     for $i$ in $1 \ldots - n$:
3:         Fill $MBC$ block with the Hilbert curve ($m = \sum_{i=1}^{n} 8^{i-1} * k_i$)
4:         Convert the Hilbert value corresponding to the center point $C_p$ of the $n$ smallest bounding cubes according to the above formula, where $center\ C_p = [\sum_{i=1}^{n}(x_i/n), \sum_{i=1}^{n}(y_i/n), \sum_{i=1}^{n}(z_i/n)]$
5:     end for
6: Sort the $MBC$ center code values and get $L_{MBC}(m_1, m_2, \ldots, m_n)$
7: Sample data objects and calculate the boundary coordinates and center coordinates of the $MBC$ block for each data object according to the size of the data volume
8: Use the data sample $x, L_{q\emptyset}, L_{p\theta}$ to pre-train the variational autoencoding model
9: $K - means$ initialize cluster center $c$
10:    for $i$ in $1 \ldots - n$ :
11:        Calculate the latent variable space of $x$ to represent $z$: $z \sim p_\emptyset(z|x)$
12:        Update the target distribution $P_{ij}$ with $z$ and equations (16) and (17)
13:        Save the cluster center label $S_{old} = S$ and update the label with:
14:            $S_i = \arg_j \max q_{ij}$
15:        for $L$ in $1 \ldots - T$ :
16:            Gradient descent method to optimize the objective function $L = L_{p\theta} + L_{q\emptyset} + \gamma L_c$
17:            if $\frac{O(S_{old} \neq S)}{N} < \delta$ (where O is the number of label changes)
18:                Return the weights of $p_\theta$ and $q_\emptyset$, the cluster center $c$
19:            end if
20:        end for
21: continue
22: Re-sort according to the Hilbert code value of the cluster center $c$ after clustering
23: Determine the maximum number of child nodes that a Hilbert R-tree node can store according to memory constraints
24:   if $SIZE_{currentcluster} \leq MAX_{node-HRT}$:
25:           All elements of the current category are regarded as nodes of the current level of the tree structure
26:   else:
27: Arrange the Hilbert code values corresponding to the data center in ascending order to form a leaf node
28: According to the time when the node is generated, the middle node and the root node of the tree are formed from bottom to top to construct an efficient Hilbert R-tree index structure
```

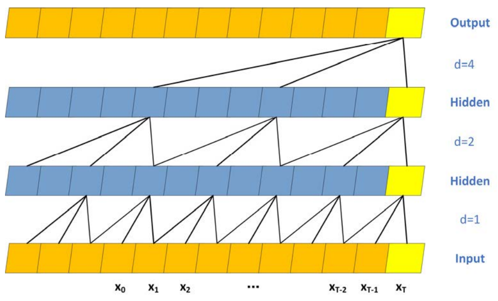

FIGURE 3. Expanded convolution structure diagram.

3) RESIDUAL CONNECTION

The basic unit of *TCN* uses causality and dilative convolution as the standard convolution layer and adds layer normalization and a linear function. Every two of these unit blocks are connected with identity mapping as a residual module so that the network model can transmit information in a cross-layer manner.

In contrast to *RNN* and other networks, *TCN* has the characteristics of parallelism, gradient stability, and flexible receptive field. The forecasting process begins by selecting N points of data are selected from the dataset. Each point corresponds to the viewpoint coordinates of the continuous motion track at different times. The current time coordinates have three dimensions (i, j, k), and the coordinates at time T_c are $P_c(i_c, j_c, k_c)$. The coordinate sequence $X = (p_1, p_2, \ldots, p_c)$ is set at the current time and the previous time as the input of the model. The convolution kernel is $F = (f_1, f_2, \ldots, f_k)$, where K is the size of the convolution kernel, the input sequence is $X = (p_1, p_2, \ldots, p_c)$, and the causal convolution at p_c can be expressed as

$$F(p_c) = \sum\nolimits_{k=1}^{K} f_k P_{c-K+k} \tag{21}$$

The output of causal convolution after introducing dilated convolution is

$$F_d(p_c) = \sum\nolimits_{k=1}^{K} f_k P_{c-(K-k)d} \tag{22}$$

The input of the hidden layer is normalized through regularization, and a nonlinear is introduced through *Relu* function:

$$F_o(F_d) = max(0, F_d) \tag{23}$$

Dropout is introduced after passing through the above convolutional layer to simplify the network and prevent overfitting.

$$F'_o \sim Dropout(F_o(F_d)) \tag{24}$$

To obtain the long time series information with the expansion convolution, the stability of the network model needs to be considered as the network layers are deepened. Therefore, identity mapping is added to increase network stability, and the output result is

$$O = F'_o(p_c) + p_c \tag{25}$$

Two of these unit blocks were used as a residual module along with the identity map:

$$O_n = Activation(F'_o(p_c) + p_c) \tag{26}$$

The deep network is stacked using residual modules, and the structure is shown in *Figure* 4. *TCN* blocks are repeatedly connected, and the output of each stack is used as the input of the next one to deepen the network layers and extract essential features. Finally the *Relu* nonlinear factor is added to the output feature, and a one-dimensional convolution layer is used to replace the coordinate position of the next viewpoint predicted by the output of the full connection layer.

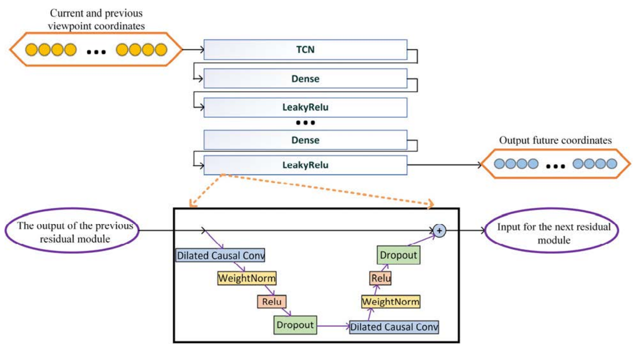
Current and previous viewpoint coordinates
TCN
Dense
LeakyRelu
•••
Dense
LeakyRelu
Output future coordinates
The output of the previous residual module
Input for the next residual module
Dilated Causal Conv
WeightNorm
Relu
Dropout
Dropout
Relu
WeightNorm
Dilated Causal Conv

FIGURE 4. Schematic diagram of TCN structure.

C. SCHEDULED LOADING OF VOLUME DATA

As the amount of data increases, the number of objects to be drawn during the loading process of 3D data also increases significantly. Visibility judgment is an effective method to reduce unnecessary drawing when loading large-scale data, thereby accelerating the image rendering speed. The frustum clipping method is a visibility judgment method, and its overhead operation is relatively small and easy to modify. Therefore, an efficient frustum clipping algorithm is conducive to fast and accurate loading of graphics objects, thereby greatly improving its display performance.

In the first part of this article, the original data are equally divided into *MBC* sub-blocks, and the original geological data structure is then recombined using the Hilbert R-tree algorithm. When judging whether the current data object is visible, it turns to judge the viewpoint and each spatial position relationship of the *MBC* data sub-blocks. When it is found that the data object is outside the viewpoint area, it will be cropped immediately, thereby shortening the time for object traversal in the structure and improving the rendering speed.

In this part of this article, the two-layer frustum clipping algorithm is used as follows:

1). First, the rough cutting algorithm is adopted. We simplify the viewing cone into a simple cone, the position relationship between the cone and the space object is judged, and the overall judgment times are reduced to improve the cutting efficiency. The flow of the rough clipping algorithm flow is shown in *Algorithm* 2:

A: Simplify the viewing cone hexahedron into a simple cone. The cross-sectional view is shown in *Figure* 5. From the viewpoint position, make the smallest cone encompassing the viewing cone and use the simple cone to check the positional relationship between it and the smallest boundary cube *MBC*.

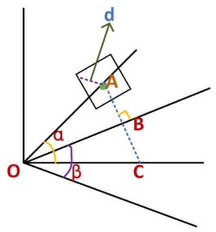

FIGURE 5. The position profile of the MBC block and simple cone (where A is the central coordinate of an MBC block, C is A point on the visual axis, AC is perpendicular to the cone surface at point B, and d is the distance from the center of MBC block to any vertex).

B: If the center of $A(x, y, z) \notin cone$ an *MBC* block is not inside the cone, $AB \geq d$, the *MBC* block is judged to be outside the cone, and the algorithm ends. Otherwise, proceed to C.

C: If $(d + zsin\theta/2)^2 \leq (x^2 + y^2)cos(\theta/2)^2$, the *MBC* block is judged to be outside the cone, and the space object contained in the *MBC* block is cut. Otherwise, the *MBC* block is judged to have intersected the cone, the fine clipping algorithm can be used for further clipping, and the algorithm ends.

2). The fine cropping algorithm is based on the previous rough cropping and further uses the standard viewing frustum truncated pyramid to finely crop the space object and improve the accuracy of the cropping.

When using the standard frustum for cutting, the *MBC* block of the data was first used to determine the spatial position of the six faces of the frustum (i.e., the top, bottom, left, right, near, and far of the hexahedron in *Figure* 6). The overall idea is that when the *MBC* block is located outside a certain plane equation of the frustum, it is invisible and the object is cropped; if the *MBC* block is not cropped in the above process, it means that the positional relationship with the frustum is inclusive or intersect, specifically in the following two cases:

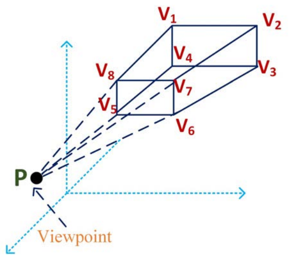

FIGURE 6. Schematic diagram of the frustum.

(1) Inclusive: When the *MBC* block is in all six planes, indicating that the space object is located in the viewpoint area, the space object is directly sent to the *GPU* drawing pipeline for rendering and display.

(2) Intersect: When the inner side of a certain plane of the frustum contains a part of the *MBC*, and the outer side of the plane also contains a part, indicating that the space object and the frustum are intersecting, then continue to judge the underlying objects and traverse all the space objects in turn.

Specifically, the viewpoint position is located at the zero point of the world coordinate system and the viewing cone model is placed along the positive *Z* axis, with the set projection matrix used to transform the vertices, allowing for the six plane equations corresponding to the truncated cone to be obtained.

After obtaining the six planes, the general approach is to calculate the distance from each vertex to each plane; however, it would be more complicated to calculate eight vertices in this way. The method in this study is to first determine the vertices n and p of the *MBC* block. Point p is the vertex closest to the plane, and point n is the vertex of the farthest diagonal (the vertex furthest from p). Suppose $p(x_{min}, y_{min}, z_{min})$ brings six plane equations $ax + by + cz + d = 0$: if $ax_{min} + by_{min} + cz_{min} + d > 0$, it can be judged that the vertex of the *MBC* block closest to the plane is outside the plane, and then the *MBC* block is outside the plane for clipping.

In the same way, if $p(x_{min}, y_{min}, z_{min})$ is inside the plane and $n(x_{max}, y_{max}, z_{max})$ is outside the plane, $ax_{min} + by_{min} + cz_{min}+d < 0$ and $ax_{max}+by_{max}+cz_{max}+d > 0$ mean that the *MBC* block intersects the frustum. If point p and point n are both inside the frustum plane equation, $ax_{min}+by_{min}+cz_{min}+d < 0$, $ax_{max}+by_{max}+cz_{max}+d < 0$, it means that the *MBC* block is inside the plane and is directly sent to the rendering pipeline for rendering and display.

Among these, the viewpoint movement process is regarded as a dynamic viewpoint model. The movement process divides the entire range of data objects into visible, potential, and unloading areas. After the predicted viewpoints were obtained through the cutting of the frustum model above, the unloading and loading rendering were performed by comparing the divided regions below. Specifically, when the viewpoint is at position A, parts 1 and 2 in the figure are the visible areas, which are rendered and displayed; the potential areas are visible areas 2 and 3 of the predicted next viewpoint B, which need to be rendered in advance, and the data in this area are marked as visible data and loaded into the *GPU* pipeline for rendering, making it convenient for the user to directly read and display the data object during continuous browsing. When the viewpoint moves from A to B, area 1 is marked as an unloaded domain to reduce memory, as shown in *Figure* 7:

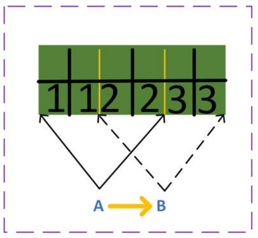

FIGURE 7. Diagram of data region division.

Because of the process of browsing the image constantly, which changes direction and coordinates of the viewpoint, a viewpoint trajectory is formed. Therefore, the current viewpoint position and its historical data in module (b) can be used to predict the next viewpoint position, and the data objects of the potential area loaded into the memory to be displayed, improving the smoothness of the image display during the entire information loading process. The overall process of combining viewpoint prediction and frustum clipping algorithms is shown in *Figure* 8:

III. EXPERIENAL RESULTS AND DISCUSSION

A. EXPERIMENTAL PLATFORM

The processor used in this experiment was Intel Core i5-9300H, the operating system was Windows 10, the chip type was NVIDIA GeForce GTX 1650, the *CPU* frequency is 2.40GHz, the memory was 8GB, and the code was written using PyCharm2019, Visual Studio 2019, and QT5.7.

The data used in the experiment are a subset of the seismic area data of an oil field in China and are stored in *SEG* − *Y* format. Divided into three groups, dataset *A* is 458.3*MB*,

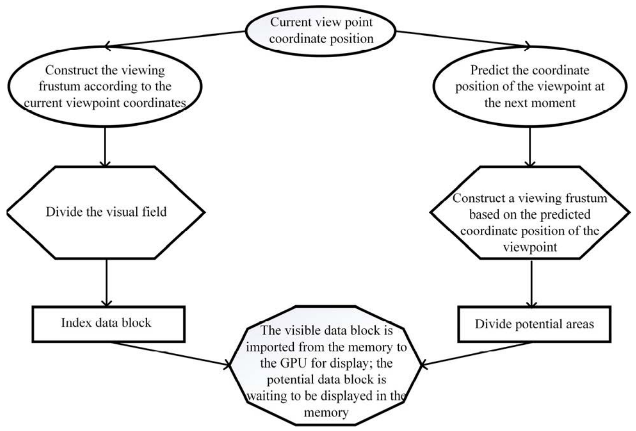
Current view point coordinate position
Construct the viewing frustum according to the current viewpoint coordinates
Predict the coordinate position of the viewpoint at the next moment
Divide the visual field
Construct a viewing frustum based on the predicted coordinate position of the viewpoint
Index data block
Divide potential areas
The visible data block is imported from the memory to the GPU for display; the potential data block is waiting to be displayed in the memory

FIGURE 8. **Viewpoint movement combined with frustum clipping flow chart.**

dataset B is 3219.5MB (the data in group B is more evenly distributed), dataset C is 13984.1MB (the data in group C is larger in amount and more scattered), and the data contains buried information such as depth, range, thickness, and the stretching trend.

B. DATA RECORD

Experiments were conducted to evaluate the effectiveness of this method by comparing the relevant modules. Specifically, this method is mainly divided into the following parts, as shown in *Figure* 9:

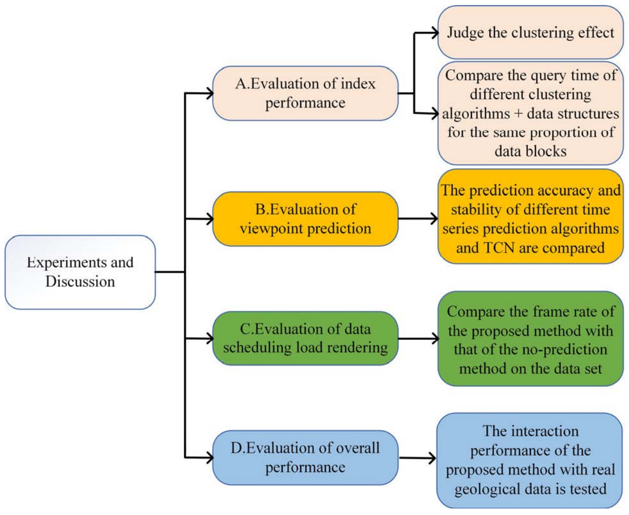
Experiments and Discussion
A.Evaluation of index performance
Judge the clustering effect
Compare the query time of different clustering algorithms + data structures for the same proportion of data blocks
B.Evaluation of viewpoint prediction
The prediction accuracy and stability of different time series prediction algorithms and TCN are compared
C.Evaluation of data scheduling load rendering
Compare the frame rate of the proposed method with that of the no-prediction method on the data set
D.Evaluation of overall performance
The interaction performance of the proposed method with real geological data is tested

FIGURE 9. **Experimental methods flow chart of test and evaluation.**

1) INDEXING EFFICIENCY OF DATA STRUCTURES

In view of the *VDEC* model partially integrated with the index data structure in this study, validation was carried out on *MNIST* handwritten dataset. First, $k - means$ was used to initialize the cluster center, and encoder was used for pre-training. The optimizer used was the Adam optimizer, the learning rate was set to 0.003, and the parameters were updated after every 10 epochs trained. The output dimension was set to 10, and the training was conducted 300 times. The hidden feature space Z was constructed according to the mean and variance returned by the encoding layer. For visualization, $T - sne$ was used to map the sampled part Z to a two-dimensional space. The aggregation distribution of data in the potential space is shown in *Figure* 10, which indicates that the representation of potential features is suitable for clustering, and the clustering accuracy of reconstructed samples based on real labels and models reaches 94.3% with a Jaccard similarity coefficient 0.959, and an NMI score of 0.956.

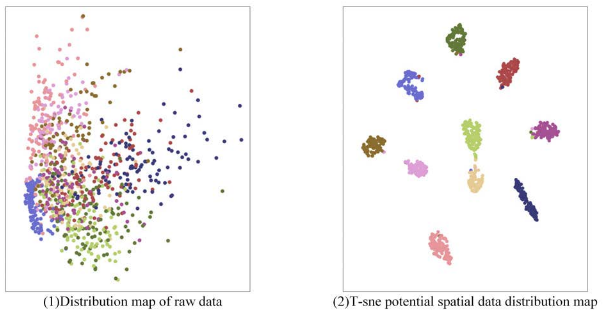

FIGURE 10. **Data clustering distribution diagram of VDEC potential space.**

In addition, to evaluate the overall index effect of data structure, we set the minimum particle size to 1MB in this part to compare it not only with the related Hilbert tree structure, but also with the index efficiency of octree (OCT) structure. Specifically, we take the same proportion data blocks (1%, 3% and 6%) for three groups of data, A, B and C respectively, and compare the OCT, HRT, $KHRT$, and the VDEC − HRT structure query data times for blocks. The recorded results are shown in *Table* 1.

TABLE 1. **Comparison of indexing efficiency of different data structures.**

Data set	Data block percentage	Method and time (s)			
		VDEC -HRT	KHRT	HRT	OCT
A	1%	0.34	0.43	1.32	3.11
	3%	0.85	1.04	4.01	7.58
	6%	1.03	3.09	6.73	11.36
B	1%	8.76	20.63	26.32	28.14
	3%	25.28	48.79	70.81	72.93
	6%	40.03	75.01	115.39	120.61
C	1%	13.96	31.49	40.16	49.98
	3%	42.18	93.88	122.73	146.72
	6%	77.76	153.25	193.65	224.38

As shown in the results, for a A small amount of data in group A, the query time of the method in this study is less than that of the previous two methods, but the comparison is not obvious. For group B data, compared using the OCT algorithm, the index time of $VDEC - HRT$ in this study is reduced by 65.34%-68.87%; compared using the HRT algorithm alone, the index time of $VDEC - HRT$ in this study is reduced by 64.30%-66.60%; compared using $KHRT$, the index time of $VDEC - HRT$ is reduced by 46.63%-57.54%. Compared using OCT, HRT and $KHRT$, the query time of group C data subblock was reduced by 65.34%-72.07%, 59.85%-65.63% and 49.26%-55.67%, respectively. Therefore, even compared with large data, our algorithm can make the data close to the original data nodes more compact through improved clustering, effectively reducing the frequent visits to disk, improving the query efficiency significantly, and reducing the index time.

2) EVALUATION OF VIEWPOINT PREDICTIONS

To evaluate the accuracy of the prediction algorithm, we compared the accuracy of the Lagrange interpolation, stacked short and long memory networks, and the proposed method for predicting data blocks with durations of 1, 4 and 8 min, respectively. The results are listed in *Table* 2.

TABLE 2. Comparison of the accuracy of data block prediction algorithm with different times.

Data set	Time(minutes)	Method and accuracy (%)		
		Lagrange	Stacked-LSTM	Our method
A	1	83.19	92.42	94.98
	4	80.12	89.31	92.56
	8	74.98	88.56	90.72
B	1	82.84	89.48	94.85
	4	73.19	86.63	92.14
	8	67.13	84.71	89.60
C	1	74.55	87.54	91.66
	4	68.91	84.76	89.73
	8	62.77	81.09	85.78

It can be concluded from the data that the accuracy of the proposed algorithm is 11.74%-23.01% higher than that of the Lagrange interpolation algorithm and 2.16%-5.51% higher than that of $StackedLSTM$. With the average frame rate unchanged, we set the steps to 8, 16 and 24 frames respectively, and randomly change the position and direction of the camera to keep the viewpoint moving at a uniform speed for 15 min. Moreover, to avoid the interference of different average frame rate experiment, we chose the average frame rates of 12, 24, and 48 frames to calculate the asynchronous length and compare the prediction accuracy. The recorded data is presented in *Table* 3.

TABLE 3. Influence of asynchronous length on prediction accuracy.

Step length	Average accuracy (%)		
	12(FPS)	24(FPS)	48(FPS)
8	96.22	95.93	95.07
16	94.87	93.04	91.63
24	93.42	91.46	87.54

The data analysis results show that the accuracy rate decreases as the selected step size increases, but the smaller the selected step size frame, the weaker the prediction effect. Therefore, the accuracy of the prediction can be improved by adjusting the step size parameters.

In addition, when the average frame rate is approximately 30 fps, X and Y components of the partial position coordinates were sampled during prediction, and the errors between the predicted position using TCN and $LSTM$ models and the actual position coordinates are compared. The time step was selected to be 15, and the curve was obtained after 15 iterations, as shown in *Figure* 11.

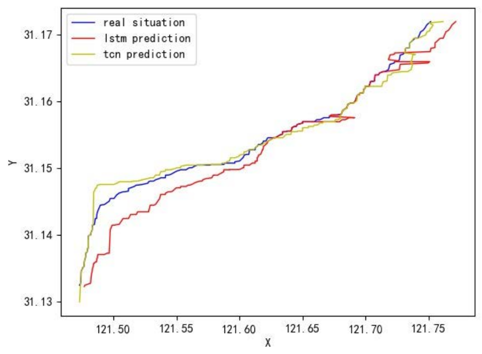

FIGURE 11. Comparison curve of prediction results.

According to the experimental results, the prediction accuracy of $ACC - TCN$ in the coordinate dataset of this study was 97.34%, and that of the $ACC - LSTM$ was 93.86%. It is evident that the prediction effect of TCN is better than that of $LSTM$ in the XY coordinate component of the viewpoint in this study, indicating that TCN is effective in the long time series tasks.

3) EVALUATION OF DATA LOAD RENDERING

In addition, to test the stability of the algorithm, frame sampling points were selected as X-axis components to evaluate the frame rates on three sets of datasets under the no-prediction and proposed predictive scheduling loading algorithms. The results are listed in *Table* 4.

TABLE 4. The frame rates of the proposed method are tested on three sets of data.

Dataset	Method (fps)	Sampling frame					
		0	10	20	30	40	50
A	Our method	52.23	54.01	53.20	53.18	52.98	53.16
	Without prediction	48.92	49.98	50.03	49.94	48.97	49.99
B	Our method	34.52	35.02	35.11	35.09	35.06	35.04
	Without prediction	24.33	24.31	24.59	25.05	24.52	24.64
C	Our method	75.52	30.01	29.96	30.08	39.97	29.88
	Without prediction	12.65	12.43	12.57	12.52	12.39	12.55

The experimental results show that compared with no-prediction, the frame rate is still higher and more stable even when the data size is large, causing the rendering effect to be smoother.

TABLE 5. Query time of data slicing for different algorithms.

Slice number	Method	Time (s)
259	OCT+TCN	2.26
	HRT+TCN	1.40
	KHRT+TCN	1.05
	VDEC-HRT+TCN	**0.49**
269	OCT+TCN	2.23
	HRT+TCN	1.25
	KHRT+TCN	1.04
	VDEC-HRT+TCN	**0.48**
279	OCT+TCN	2.09
	HRT+TCN	1.42
	KHRT+TCN	1.03
	VDEC-HRT+TCN	**0.49**
289	OCT+TCN	2.21
	HRT+TCN	1.27
	KHRT+TCN	1.03
	VDEC-HRT+TCN	**0.50**

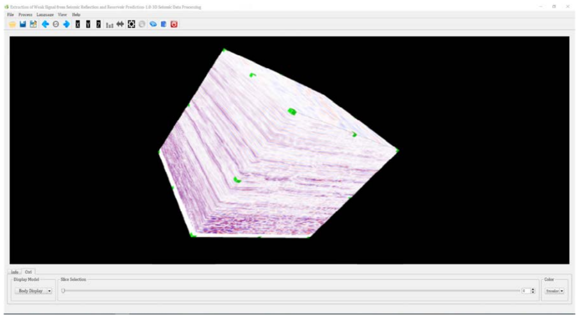

FIGURE 12. Geological body data display effect.

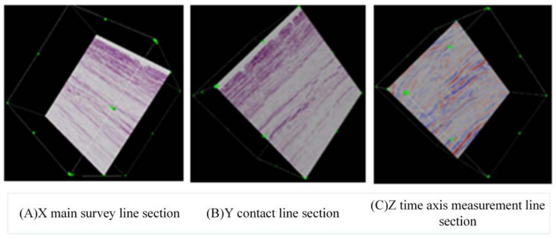
(A)X main survey line section
(B)Y contact line section
(C)Z time axis measurement line section

FIGURE 13. Geological data along the different directions of the survey line section display effect.

4) OVERALL PERFORMANCE EVALUATION

Here, with the index structure preloading model effectively, this algorithm can test the interactive performance of the entirety group *B*data. For example, after slicing geological data every 10 samples and recording the time needed for each algorithm according to the section; the results are shown in *Table* 5, which significantly reduces the query time slice compared with other algorithms.

Finally, we test the effect of the system's overall display settings that have different display modes for the entire system platform rendering. *Figure* 12 is a blue map mode of body data, *Figure* 13 is Lord with *X* as the contact line, *Y* and *Z* as rendering time axis line direction, and is the running observation of actual geological data. And it can be seen that the system platform in this study can reflect the internal structure and information of geological data with high quality.

IV. CONCLUSION

In this study, based on the original Hilbert R-tree structure, variational deep embedding clustering is integrated to improve the overlap of the data node space and directly improve the efficiency of the index structure algorithm. In addition, the new time-series convolutional network is used to predict the viewpoint coordinates, which improves the accuracy of the original prediction. Combined with the data scheduling loading module, the data that need to be drawn and displayed are preloaded into the memory. Through comparative experiments on geological datasets, it is proven that the proposed scheme can solve the problems of real-time display and lag during large-scale data visualization through the use of multiple modules, and improve the accuracy and fluency of the entire data loading process.

In the future, the proposed method needs to be improved and optimized, and the algorithm can be explored in greater depth to include more complex and larger datasets. Moreover, the method proposed in this study has only been tested on geological datasets, and the system can be extended to provide researchers with more convenient tools.

THE FORMULA OF APPENDIX

(1)

$$\begin{aligned} I(x,z) &= \iint p(x,z)\, log \frac{p(x,z)}{p(x)\,p(z)} dxdz \\ &= \iint p(z\,|\,x)\,\check{p}(x)\, log \frac{p(z|x)}{p(z)} dxdz \\ &= D_{kl}[p(z|x)\check{p}(x)||p(z)\check{p}(x)] \end{aligned}$$

(4)

$$\begin{aligned} p(z\,|\,x) &= MIN\{D_{kl}[q_\phi(z|x)||p(x|z)] - I(x,z)\} \\ &= MIN\{\iint q_\phi(z\,|\,x)\check{p}(x)\, log \frac{q_\phi(z\,|\,x)}{p(z|x)} dxdz \\ &\quad - \iint p(z\,|\,x)\,\check{p}(x)\, log \frac{p(z|x)}{p(z)} dxdz\} \\ &= MIN\{\iint \check{p}(x)[q_\phi(z\,|\,x)\, log \frac{q_\phi(z|x)}{p(z|x)} \\ &\quad - p(z\,|\,x)\, log \frac{p(z|x)}{p(z)}]dxdz\} \end{aligned}$$

(7)

$$\begin{aligned} & D_{kl}[q_\phi(z|x)||p(z|x)] \\ &= \sum q_\phi(z\,|\,x)\, log \left[\frac{q_\phi(z\,|\,x)}{p(z|x)}\right] \\ &= E[logq_\phi(z|x) - logp(z) - logp_\theta(x|z) + logp(x) + logp(z)] \\ &= logp(x) + E[logq_\phi(z|x) - logp(z) - logp_\theta(x|z)] \end{aligned}$$

(18)

$$\begin{aligned} L_c &= KL(P||Q) \\ &= -H(Q) + H(Q;P) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} p_{ij}\log p_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{k} p_{ij}\log q_{ij} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} p_{ij}\log\frac{p_{ij}}{q_{ij}} \end{aligned}$$

AUTHOR CONTRIBUTIONS

Yu-Hang Zhang conceived the algorithms, and designed the experiments; Chang Wen reviewed the paper; Min Zhang checked the spelling and made suggestions; Kai Xie conducted the comparative experiment; Jian-Biao He is responsible for data collection.

REFERENCES

[1] M. Hu, H. Hu, W. Cai, Z. Mo, N. Xiang, J. Yang, and C. Fang, "The safety and feasibility of three-dimensional visualization technology assisted right posterior lobe allied with part of V and VIII sectionectomy for right hepatic malignancy therapy," *J. Laparoendoscopic Adv. Surgical Techn.*, vol. 28, no. 5, pp. 586–594, May 2018.

[2] J. Magat, A. Fouillet, M. Constantin, K. Haliot, J. Naulin, D. El Hamrani, D. Benoist, S. Charron, R. Walton, O. Bernus, and B. Quesson, "3D magnetization transfer (MT) for the visualization of cardiac free-running purkinje fibers: An *ex vivo* proof of concept," *Magn. Reson. Mater. Phys., Biol. Med.*, vol. 34, no. 4, pp. 605–618, Aug. 2021.

[3] R. Miao, J. Song, and Y. Zhu, "3D geographic scenes visualization based on WebGL," in *Proc. 6th Int. Conf. Agro-Geoinf.*, Aug. 2017, pp. 1–6.

[4] T. C. Piedade, V. F. Melo, L. C. P. Souza, and J. Dieckow, "Three-dimensional data interpolation for environmental purpose: Lead in contaminated soils in southern Brazil," *Environ. Monitor. Assessment*, vol. 186, no. 9, pp. 5625–5638, Sep. 2014.

[5] J. Yun and K. Seokyeon, "Approximate image-space multidirectional occlusion shading model for direct volume rendering," *Appl. Sci.*, vol. 11, no. 12, pp. 5717–5733, 2021.

[6] Y. Shin, B.-S. Sohn, and H. Kye, "Acceleration techniques for cubic interpolation MIP volume rendering," *Multimedia Tools Appl.*, vol. 80, no. 14, pp. 20971–20989, Jun. 2021.

[7] G. Gu and D. Kim, "Accurate and efficient GPU ray-casting algorithm for volume rendering of unstructured grid data," *ETRI J.*, vol. 42, no. 4, pp. 608–618, Aug. 2020.

[8] S. Vaibhav *et al.*, "R-tree data structure implementation for Computer Aided Engineering (CAE) tools," *Int. J. Simul. Multidisciplinary Des. Optim.*, vol. 12, p. 6, Apr. 2021.

[9] Y. Yang, P. Bai, N. Ge, Z. Gao, and X. Qiu, "LAZY R-tree: The R-tree with lazy splitting algorithm," *J. Inf. Sci.*, vol. 46, no. 2, pp. 243–257, Apr. 2020.

[10] W. Macyna and K. Majcher, "Cost-based storage of the R-tree aggregated values over flash memory," in *Proc. Int. Conf. Ind. Enterprise Syst. Eng. (IcoIESE)*, 2019, pp. 97–102.

[11] X. Wang, W. Meng, and M. Zhang, "A novel information retrieval method based on R-tree index for smart hospital information system," *Int. J. Adv. Comput. Res.*, vol. 9, no. 42, pp. 133–145, May 2019.

[12] Y. Hong, Q. Tang, X. Gao, B. Yao, G. Chen, and S. Tang, "Efficient R-tree based indexing scheme for server-centric cloud storage system," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1503–1517, Jun. 2016.

[13] S. Yuan, D. Pi, X. Zhao, and M. Xu, "Differential privacy trajectory data protection scheme based on R-tree," *Expert Syst. Appl.*, vol. 182, Nov. 2021, Art. no. 115215.

[14] P. Goyal, J. S. Challa, D. Kumar, A. Bhat, S. Balasubramaniam, and N. Goyal, "Grid-R-tree: A data structure for efficient neighborhood and nearest neighbor queries in data mining," *Int. J. Data Sci. Anal.*, vol. 10, no. 1, pp. 25–47, Jun. 2020.

[15] Q. He, Y. Chen, Q. Dong, and Y. Wang, "Mining moving object gathering pattern based on resilient distributed datasets and R-tree index," *Neurocomputing*, vol. 393, pp. 194–202, Jun. 2020.

[16] S. D. Yang and W. Choi, "Fast Hilbert R-tree bulk-loading scheme using GPGPU," *J. KIISE*, vol. 41, no. 10, pp. 792–798, Oct. 2014.

[17] L. Song, Z. Liping, L. Shuang, and H. Xiaohong, "Spatial skyline query method based on Hilbert R-tree in multi- dimensional space," *High Technol. Lett.*, vol. 3, pp. 262–270, Sep. 2019.

[18] Q. Du, X. Li, and R. Te, "Efficient improved K-nearest neighbor join algorithms with Hilbert R-tree in mapreduce," *ICIC Exp. Lett. B Appl. Int. J. Res. Surv.*, vol. 5, pp. 925–932, Aug. 2014.

[19] A. Obulkasim and M. A. Van De Wiel, "HCsnip: An r package for semi-supervised snipping of the hierarchical clustering tree," *Cancer Informat.*, vol. 14, Mar. 2015, Art. no. CIN.S22080.

[20] H. Y. Cui, S. Li, L. P. Zhang, and H. Jing, "R-tree constructionbuilt based on CUK-MEANS algorithm," *J. Chin. Comput. Syst.*, vol. 37, no. 2, pp. 1000–1220, 2016.

[21] H. Cheng, K. Xie, C. Wen, and J.-B. He, "Fast visualization of 3D massive data based on improved Hilbert R-tree and stacked LSTM models," *IEEE Access*, vol. 9, pp. 16266–16278, 2021.

[22] M. Zhao, X. Yao, J. Wang, Y. Yan, X. Gao, and Y. Fan, "Single-channel blind source separation of spatial aliasing signal based on stacked-LSTM," *Sensors*, vol. 21, no. 14, p. 4844, Jul. 2021.

[23] S. T. Jishan and Y. Wang, "Audience activity recommendation using stacked-LSTM based sequence learning," *Mach. Learn. Comput.*, vol. 37, no. 2, pp. 98–106, 2017.

[24] U. Yayan, A. T. Arslan, and H. Yucel, "A novel method for SoH prediction of batteries based on stacked LSTM with quick charge data," *Appl. Artif. Intell.*, vol. 35, no. 6, pp. 421–439, May 2021.

[25] J. Li, F. Guo, A. Sivakumar, Y. Dong, and R. Krishnan, "Transferability improvement in short-term traffic prediction using stacked LSTM network," *Transp. Res. C, Emerg. Technol.*, vol. 124, Mar. 2021, Art. no. 102977.

[26] X. Meng, M. Liu, and Q. Wu, "Prediction of Rice yield via stacked LSTM," *Int. J. Agricult. Environ. Inf. Syst.*, vol. 11, no. 1, pp. 86–95, Jan. 2020.

[27] Z. J. Wei, W. W. Gen, C. Bin, and L. J. Chen, "A method of view-frustum culling with OBB based on octree," in *Proc. IET Conf. Wireless, Mobile Sensor Netw. (CCWMSN)*, 2007, pp. 680–682.

[28] P. Carvalho, M. Santos, and W. Schwartz, "An improved view frustum culling method using octrees for 3D real-time rendering," *Int. J. Image Graph.*, vol. 13, no. 3, pp. 1–23, 2013.

[29] R. Kodituwakku, K. R. Wijeweera, and M. A. P. Chamikara, "Anefficient-line clipping algorithm for 3D space," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, no. 5, pp. 96–101, 2012.

[30] P. R. De Carvalho, M. C. Dos Santos, W. R. Schwartz, and H. Pedrini, "An improved view frustum culling method using octrees for 3D real-time rendering," *Int. J. Image Graph.*, vol. 13, no. 3, Jul. 2013, Art. no. 1350009.

[31] O. Capra, F. Berthaut, and L. Grisoni, "Levels of detail in visual augmentation for novice and expert audiences," *Comput. Music J.*, vol. 44, nos. 2–3, pp. 92–107, Jul. 2021.

[32] J. Li, H. Wu, C. Yang, D. W. Wong, and J. Xie, "Visualizing dynamic geosciences phenomena using an octree-based view-dependent LOD strategy within virtual globes," *Comput. Geosci.*, vol. 37, no. 9, pp. 1295–1302, Sep. 2011.

[33] B. Kathariya, V. Zakharchenko, Z. Li, and J. Chen, "Level-of-detail generation using binary-tree for lifting scheme in LiDAR point cloud attributes coding," in *Proc. Data Compress. Conf. (DCC)*, Piscataway, NJ, USA, Mar. 2019, p. 580.

[34] R. S. Gargees and G. J. Scott, "Large-scale, multiple level-of-detail change detection from remote sensing imagery using deep visual feature clustering," *Remote Sens.*, vol. 13, no. 9, p. 1661, Apr. 2021.

[35] R. Wei, C. Garcia, A. El-Sayed, V. Peterson, and A. Mahmood, "Variations in variational autoencoders—A comparative evaluation," *IEEE Access*, vol. 8, pp. 153651–153670, 2020.

[36] A. Abdella and I. Uysal, "A statistical comparative study on image reconstruction and clustering with novel VAE cost function," *IEEE Access*, vol. 8, pp. 25626–25637, 2020.

[37] J. Bian, X. Hui, S. Sun, X. Zhao, and M. Tan, "A novel and efficient CVAE-GAN-based approach with informative manifold for semi-supervised anomaly detection," *IEEE Access*, vol. 7, pp. 88903–88916, 2019.