Received December 3, 2020, accepted January 3, 2021, date of publication January 18, 2021, date of current version January 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051911

Fast Visualization of 3D Massive Data Based on Improved Hilbert R-Tree and Stacked LSTM Models

HUAN CHENG[1,2,3], **KAI XIE**[1,2,3], **CHANG WEN**[3,4], **AND JIAN-BIAO HE**[5]
[1]School of Electronic Information, Yangtze University, Jingzhou 434023, China
[2]National Demonstration Center for Experimental Electrical & Electronic Education, Yangtze University, Jingzhou 434023, China
[3]Western Institute of Yangtze University, Karamay 834000, China
[4]School of Computer Science, Yangtze University, Jingzhou 434023, China
[5]School of Computer Science and Engineering, Central South University, Changsha 410083, China

Corresponding author: Kai Xie (pami2009@163.com)

This work was supported in part by the 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, and in part by the National College Student Innovation and Entrepreneurship Training Program under Grant 202010489007.

ABSTRACT With the explosive growth of scientific data, significant challenges exist with respect to the interaction of large volumetric datasets. To solve these problems, we propose a visualization algorithm based on the Hilbert R-tree improved by the clustering algorithm using K-means (CUK) and a stacked long short-term memory (LSTM) model to quickly display massive data. First, we use the Hilbert R-tree optimized by the CUK to quickly store unevenly distributed data and build a fast index for the massive data. Then, we determine the position of the current point of view and use the stacked LSTM model to predict the next point of view. According to the location of two points, we divide the visible area. Finally, according to the preloading strategy, we import the data into the cache area of the graphics processing unit (GPU), which greatly realizes smoother rendering data and large-scale data interaction visualization. The experimental results showed that the proposed algorithm can quickly and accurately draw large volumetric data with high quality while guaranteeing rendering quality.

INDEX TERMS Large-scale datasets, prediction model based on a stacked LSTM model, Hilbert R-tree, load in advance.

I. INTRODUCTION

The three-dimensional (3D) visualization technique is an effective method to intuitively analyze data for researchers in medicine, remote sensing, and geological exploration. Compared with two-dimensional (2D) data visualization analysis, 3D data visualization analysis has an irreplaceable advantage. Through a 3D display, inline information between scientific data can be intuitively observed so that researchers can make rapid and accurate judgments about the information represented by the data. However, with the continuous development of advanced data acquisition technology, increasing amounts of data are being obtained [1]–[3], and many 3D visualization technologies are no longer suitable for their intended purposes [4], [5] of fluency and clarity in 3D display.

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

Although the graphics processing unit (GPU)-based [6]–[8] application used in 3D volume rendering relieves the workload of the CPU when the latter handles many 3D data operations, the GPU is always limited by the size of its memory and cannot directly process massive data that exceed the scale of the memory. The traditional volume data rendering process is used to completely load the volume data into memory in one go. The rendering module only obtains the data from memory, therefore, this method is invalid for large seismic data that exceed the memory scale. To achieve an interactive visualization of large-volume data, researchers have proposed many algorithms based on data compression [9] and bricking [10], [11]. These schemes are designed to reduce the amount of rendered data without reducing the quality. However, they cannot effectively increase the speed of data display. In addition, some algorithms that use spatial index technology (such as octree [12], R-tree [13], [14], and Hilbert

R-tree [15], [16]) to increase the query speed also have their own disadvantages which are discussed below.

An octree (OCT) is a layered data organization that uses a uniform partition. If the termination condition is not selected properly, it will produce too many subblocks and the I/O operation will rise sharply, resulting in the rendering getting stuck. The OCT-based hierarchical indexing scheme has many advantages in massive volumetric data rendering. Lamar *et al.* [4] first proposed a multi-resolution layering scheme based on an OCT for volume rendering, however, the artificial selection of the minimum size of the OCT will affect the efficiency of the OCT index. Weiler *et al.* [5] then proposed an effective extension of the algorithm to avoid discontinuities between different levels of detail. In addition, wavelet representations are also used for volume rendering, such as in the study by Muraki [17], who first introduced this method. Then, Guthe *et al.* [18] used a hierarchical wavelet representation to transform their input data and implemented real-time rendering of wavelet representation data through hardware texture mapping. Some methods enable large data visualization by compressing data to reduce the rendering data. For example, Nguyen and Saupe [19] proposed a block-wise compression algorithm that splits data into small same size blocks and then compresses them separately. It is believed that the most ideal process of data compression is compressing the data during their generation and decompressing them as needed during rendering [20]. The core concept is to reduce the amount of data that must be processed by the system each time and to realize large-volume data visualization using efficient data streams. As for the out-of-core methods [21], [22], the core of these algorithms is the performance of the data access and pre-fetching. Recently, Wang *et al.* [23] proposed an OCT-based convolutional neural network (O-CNN) model for 3D shape recognition, which is also based on voxel processing and the application of the CNN enables a high recognition rate algorithm. Although O-CNN can handle massive amounts of data, the algorithm requires significant time and can hardly meet the real-time interaction we desire to achieve. Some researchers have proposed a purely CPU-based volume rendering algorithm to avoid the limitation of GPU memory size. However, when it renders a huge amount of data, the result is obviously not ideal. Fuller *et al.* [24] proposed a CPU slicing + GPU rendering algorithm based on an OCT data visualization method, which realizes the real-time visualization of large-volume data. The algorithm is based on the visibility test [25], which uses the strategy of partitioning a block, divides the data into small cubes of the same size in the preprocessing stage, and processes the data according to their visibility.

The R-tree [26] can effectively process spatial data. In the R-tree spatial index, the spatial object and each layer node are represented by the smallest bounding box. The overlap of data rectangles will inevitably affect the efficiency of the query. A space-filling curve is a curve that reduces the dimensionality of a high-dimensional space and maps high-dimensional space data to a one-dimensional curve. Combining the Hilbert curve with the R-tree can place the adjacent data blocks in the space into the same parent node as much as possible, ensure the spatial continuity of the data, and reduce a large number of repeated operations and I/O operations. The storage utilization of the Hilbert R-tree is close to 100% [27], [28], but the node space is too large. When the spatial data are unevenly distributed, the overlap showed by Figure 1 will increase and the retrieval performance will be greatly reduced, therefore, the aim of displaying massive 3D data in real time cannot be achieved. The application of a clustering model in data visualization [29] can improve retrieval efficiency. The commonly used clustering models include the *K-means* cluster, *K-medoids* cluster, and *CURE* (Cluster Using REpresentatives). Among them, the *K-means* algorithm is simple to implement and has low time complexity; however, the clustering result is sensitive to the initially selected center point and often fails to reach the global optimum. *K-medoids* are not sensitive to noise points and have good stability; however, the choice of the initial value of *K* will affect the amount of data in the node and, thus, is not suitable for processing large datasets and irregular data, thereby affecting the efficiency of the R-tree. The *CURE* algorithm has strong robustness to isolated points and can better handle massive data; however, it has high time complexity. Huan-Yu *et al.* [30] proposed the clustering algorithm using *K-means* (*CUK*), which partially combines the *CURE* algorithm with the *K-means* algorithm to compensate for the shortcomings of existing methods, but does not use it to improve the 3D display of massive data.

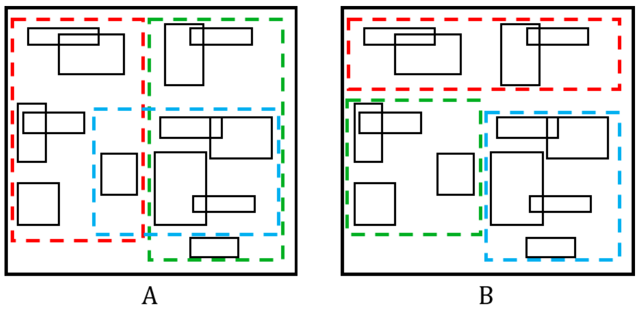


FIGURE 1. Distribution of tree node. (A) Hilbert R-tree. (B) Hilbert R-tree improved by cluster.

Although the development of indexing algorithms has reached a very high level, there is still a problem of blurring and stuck browsing images when displaying massive 3D data. Wen *et al.* [31] proposed a rapid display algorithm for massive data based on viewpoint motion that uses the Lagrange interpolation algorithm to calculate the motion trajectory of the viewpoint. However, when the viewpoint is at a stationary or jumping state, the algorithm still predicts the wrong viewpoint and, thus, there is a large deviation in the predicted viewpoint, thereby increasing the consumption of system resources.

With the aim of addressing the problems found in the above methods, this paper proposes an index method based on the Hilbert R-tree optimized by the *CUK* and a viewpoint

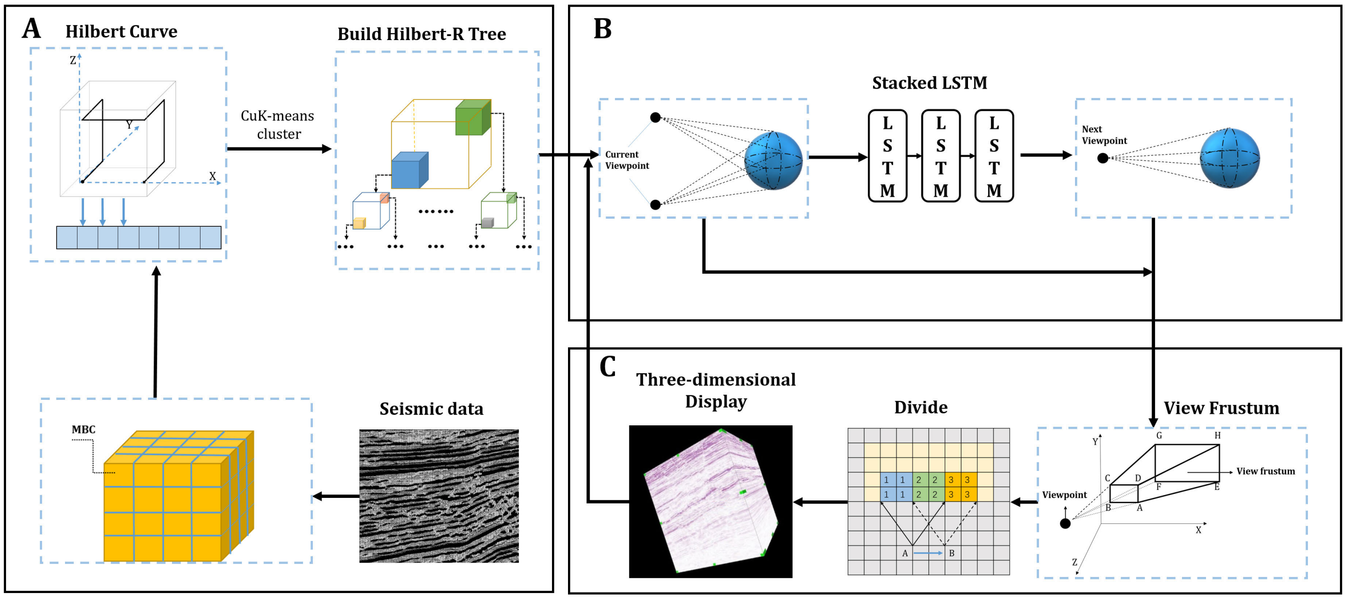


FIGURE 2. Flow of proposed algorithm.

prediction method based on the stacked LSTM model to improve the quality and browsing fluency of the 3D display of massive data.

The rest of the paper is organized as follows. Section II introduces the theoretical foundations of our method. Section III presents the details, data, and results of the experiment. Finally, Section IV presents the conclusion and future work.

II. METHOD

The fast visualization of 3D massive data with our algorithm is divided into three parts: index with the Hilbert R-tree improved by the *CUK*, viewpoint prediction based on the stacked LSTM model, and 3D visualization by preloading data. The algorithm flow is shown in Figure 2.

We begin by reading the data in the seismic data file. First, we use the Hilbert curve to reduce the dimensionality of the 3D volume data, and then we use the *CUK* to cluster the dimensionality-reduced data, recalculate the Hilbert code value of the cluster center, and build the R-tree according to the code value. In the next step, we determine the coordinate position of the current viewpoint. Then, we process information in two ways. One is to use the view frustum model to divide the spatial data according to the current coordinate position of the viewpoint and then obtain the area to be displayed. The other is to input the old viewpoint coordinate sequence information to the stacked LSTM model and use it to predict the next position coordinates of the viewpoint. We obtain the visual area, which is divided by the view frustum model, for the predicted viewpoint. Finally, by comparing the divided areas, we can divide the scope of the visual, latent, and unloaded fields. Then, the data blocks of the visual and latent fields are rendered and displayed, and the data in the unloaded field are unloaded. Next, we redetermine the coordinates of the current viewpoint and draw the next frame.

A. INDEX WITH THE HILBERT R-TREE IMPROVED BY THE CUK

To establish an efficient 3D data index, we used the R-tree based on the 3D Hilbert curve, *K-means* clustering algorithm, and *CURE* algorithm. Next, we will introduce each in detail.

1) THREE-DIMENSIONAL HILBERT SPACE-FILLING CURVE FOR VISUALIZING MASSIVE DATA

A space-filling curve is a type of curve that reduces the dimensionality of high-dimensional space data. The curve is used to map high-dimensional space data to a one-dimensional curve, thereby improving the access and query speed. Among many space-filling curves, the 3D Hilbert curve has a better filling effect on the spatial data. The Hilbert curve can preserve the spatial continuity of the data well, and adjacent points in the space are also arranged continuously in the sequence. Figure 3 shows the first-order 3D Hilbert curve and the second-order 3D Hilbert curve, respectively, and the 3D Hilbert curve of order i can be deduced by analogy.

The process of generating the 3D Hilbert curve code value as follows:

Step 1. Obtain the 3D coordinates $H(x, y, z)$ of the target point in the massive data volume.

Step 2. Convert $H(x, y, z)$ into binary coordinates $H(x_1x_2 \ldots x_N, y_1y_2 \ldots y_N, z_1z_2 \ldots z_N)$.

Step 3. According to the binary coordinates, determine the coordinates $H(x_i, y_i, z_i)$ of the point in the 3D Hilbert curve of order i. Combining the filling order of the 3D Hilbert curve of order i and $H(x_i, y_i, z_i)$, calculate the number k_i of points before this point in the curve of order i, and the Hilbert code of order i to obtain

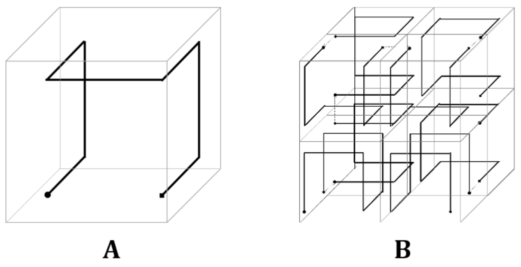


FIGURE 3. Different three-dimensional Hilbert curves. (A) First-order three-dimensional Hilbert curve. (B) Second-order three-dimensional Hilbert curve.

the value of m_i.

$$m_i = 8^{i-1} * k_i \quad (1)$$

Step 4. Summing the Hilbert code value of each order of the target point, obtain the Hilbert code value M of point $H(x, y, z)$

$$M = \sum_{i-1}^{N} m_i \quad (2)$$

2) CLUSTERING ALGORITHM BASED ON K-MEANS

When the Hilbert curve maps high-dimensional data blocks to a one-dimensional sequence, it is not completely guaranteed that the corresponding Hilbert code values of adjacent objects in space are also adjacent. Therefore, in the process of building a tree, layer-by-layer recursive filling of leaf nodes will make the volume of the node too large and even generate a large amount of overlapping space and empty space, thereby affecting the performance of the index. Therefore, the clustering algorithm is added to the process of building a Hilbert R-tree, and spatially similar data are placed under the same subtree to reduce the spatial overlap and further improve the query efficiency. This method combines the *CURE* algorithm and *K-means* clustering algorithm to form the *CUK*.

a: CURE ALGORITHM

First, the clustering algorithm selects a number of scattered objects in a class. Then these objects are multiplied by an appropriate shrinkage factor α, which is generally between 0.2 and 0.7, to achieve a better clustering effect in general, so that it is closer to the center point of the class. The algorithm is more effective in dealing with isolated points and can identify clusters of arbitrary shapes. However the *CURE* algorithm has the same shortcomings as those of the hierarchical clustering method. The time complexity ($O(n^2 logn)$) is high and the efficiency is low when processing massive data. This will lead to spend more time.

The algorithm flow is as follows:

Step 1. The original dataset should be sampled to reduce the amount of data. Suppose that the original dataset $Q = \{(x_i, y_i, z_i)|i = 1, 2, \ldots, m\}$ randomly selects n samples $W = \{(x_i, y_i, z_i)|i = 0, 1, \ldots, n-1\}$. The minimum amount of sampling data n is determined by "Chernoff bounds"

$$n_{min} = \zeta k + klog(\frac{1}{\delta}) + \sqrt{log(\frac{1}{\delta})^2 + 2\zeta log(\frac{1}{\delta})} \quad (3)$$

where ζ is the number of data points included in the smallest cluster, k is the specified number of clusters, and δ is the probability that the number of data points belonging to the cluster is less than *n/v*.

Step 2. The sampled dataset W is divided to obtain f partitions, each with a size of n/f. Then, every partition is clustered until the number of clusters in the partition is j:

$$j = \frac{n}{f \times q} \quad (4)$$

Try to ensure that j is two to three times of k by changing q $(q > 1)$.

Step 3. In the clustering process, if a cluster grows very slowly, it means that the cluster is a noise cluster. Then, we delete the noise clusters.

Step 4. We obtain the cluster center C_i of each newly generated cluster and use it as the representative of the cluster to repeat the clustering operation, and at the same time, we continuously eliminate the noise clusters that may be generated each time in this process.

$$\text{center } C_i = (\sum_{i=1}^{t} \frac{x_i}{t}, \sum_{i=1}^{t} \frac{y_i}{t}, \sum_{i=1}^{t} \frac{z_i}{t}) \quad (5)$$

where t is number of point objects.

Step 5. By calculating the distance L between the remaining data and each cluster center, we can classify each data into the cluster with the closest distance to them, thereby realizing global clustering.

$$L = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (6)$$

b: K-MEANS CLUSTERING ALGORITHM

For a sample set, the algorithm divides the sample set into k clusters according to the distance between the samples, and attempts to ensure that the data points in the clusters are compact and those between the clusters are scattered. The algorithm has low time complexity, is efficient for processing big data, and has a good clustering effect. Generally, it processes spherical datasets. However, the initial clustering center is unstable, falls easily into a local optimal solution, and is sensitive to outliers. If there are abnormal points or noise points, the center point may deviate and affect the final clustering effect.

The algorithm flow is as follows:

Step 1. Randomly select k samples from the dataset $Q = \{(x_i, y_i, z_i)|i = 1, 2, \ldots, m\}$ as the initial k centroid vectors $\{(\mu_{1x}, \mu_{1y}, \mu_{1z}), (\mu_{2x}, \mu_{2y}, \mu_{2z}), \ldots, (\mu_{kx}, \mu_{ky}, \mu_{kz})\}$.

Step 2. Initialize the cluster set $C_j = \emptyset (j = 1, 2, \ldots, k)$. Then, calculate the Euclidean distance of each point $(x_i, y_i, z_i)(i = 1, 2, \ldots, m)$ in the sample set Q from each centroid $(\mu_{jx}, \mu_{jy}, \mu_{jz})(j = 1, 2, \ldots, k)$ by using formula (6). Next, place point (x_i, y_i, z_i) into the category ρ_i with the smallest distance d_{ij}, and update $\rho_i = \rho_i \cup \{x_i, y_i, z_i\}$. Recalculate the centroid of C_j :

$$C_j = \frac{1}{C_i}(\sum\nolimits_{x \in C_i} x, \sum\nolimits_{y \in C_i} y, \sum\nolimits_{z \in C_i} z) \tag{7}$$

If none of the k centroids are changing, then go to Step 3. Otherwise, repeat Step 2.

Step 3. Output the division result $C = \{C_1, C_2, \ldots, C_k\}$.

c: CLUSTRING ALGORITHM BASED ON K-MEANS

Combining the *CURE* algorithm with the *K-means* algorithm complements the advantages and disadvantages of the two algorithms. The *CUK* first uses the *CURE* algorithm for clustering and the shrinkage factor for moving the data points closer to the cluster center, which can effectively handle isolated points and identify clusters of arbitrary shapes. After the initial clustering using the *CURE* algorithm, the center point of each cluster is selected as the initial center of *k-means*, which avoids the defect of randomly selecting the centroid.

The algorithm flow is as follows:

Step 1. Extract n data samples from the original dataset Q;divide the n data samples to obtain f regions, each of which has a size of n/f; and treat each data sample in the partition as a separate cluster

Step 2. Use the *CURE* algorithm for clustering to obtain new sub-clusters until each partition cluster reaches $n/f^* q$ new clusters. Then delete the noise points.

Step 3. Calculate the positions of the center points of the new clusters, merge the data items closest to each center point into the same cluster through the K-means method, and obtain k clusters until the position of the cluster center no longer changes.

3) BUILDING THE HILBERT R-TREE IMPROVED BY THE CUK

The Hilbert R-tree evolved on the basis of the R-tree. Each leaf node of the R-tree corresponds to a minimum bounding cube (MBC). From the leaf node upwards, a larger cube is used to surround the existing cube until the entire space is enclosed to complete the space division. The Hilbert R-tree will mechanically fill the leaf nodes, which will cause some spatial data to overlap, which, in turn, will affect the query efficiency. The clustering algorithm can reduce the space overlap and make the space allocation more reasonable, thereby reducing the number of I/O access requests to the disk and improving efficiency.

The building tree flow is as follows:

Step 1. Read the massive data volume, determine the range of the data volume in space, and construct a 3D Hilbert curve to map the 3D source volume data to a one-dimensional sequence S.

Step 2. According to formula (3), sample sequence S to obtain N data objects.

Step 3. Divide the above N data objects to obtain F partitions and perform the CUK operation in each partition to obtain the center of the clusters.

Step 4. After completing the clustering in the partition, calculate the Euclidean distance L between the center of the MBC removed in the clustering process and the center of the existing cluster by formula (6), and merge it into the corresponding cluster according to the closest distance criterion, then complete the global clustering operation.

Step 5. If the amount of data contained in the current cluster is less than or equal to the maximum capacity of the Hilbert R-tree node, consider the current cluster as a leaf node of the current layer of the Hilbert R-tree. Otherwise, sort the Hilbert code value of all data objects in the cluster in ascending order, and there are several leaf nodes from small to large according to the code value. Finally, according to the time sequence of generating the leaf nodes, the middle node and root node of the Hilbert R-tree are formed layer by layer from top to bottom, thereby obtaining an efficient Hilbert R-tree structure.

B. VIEWPOINT PREDICTION BASED ON A STACKED LSTM MODEL

To reduce the amount of data loaded into memory in each frame of the 3D display, it is necessary to load the partial data in advance according to the range of 3D objects observable from the viewpoint position. However, we observed that the position of the viewpoint of the 3D object changes, therefore, it is necessary to predict the expected position of the next viewpoint based on the previous viewpoint motion trajectory. Comparing the areas of two viewpoints, we load the data that must be displayed in advance, to avoid the problem of rendering getting stuck during 3D display when loading very large data initially. This is a typical timing forecasting problem. This study used the stacked LSTM model to predict the trajectory of the moving viewpoint.

1) LONG SHORT-TERM MEMORY NETWORK

In practice, neural networks that process time-series information include the RNN and LSTM. The gradient disappears when the sequence is too long because the RNN [32] can only solve some simple timing problems. In contrast, LSTM [33], which is one of the variants of the RNN, can solve this problem well. It is a cyclic neural network that can effectively solve long-term dependence problems. It can remove or add information to the unit state through the gate mechanism. A common LSTM cell consists of a unit, an input gate, an output gate, and a forget gate, as shown in Figure 4(A). The unit can recall the value at any time interval, and the three gates control the information flowing in and out of the unit.

1. The forget gate is used to control the ratio of the cell state C_{t-1} of the neural network at the previous time

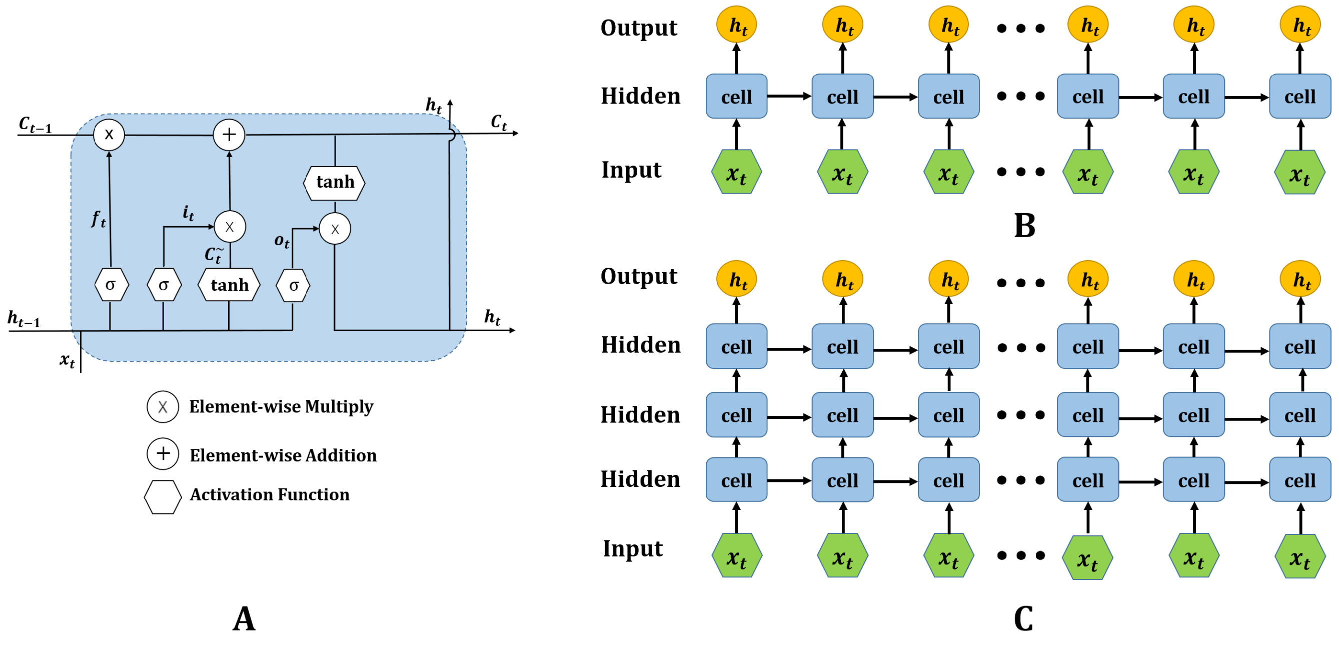


FIGURE 4. Different structures of LSTM. (A) Long short-term memory cell. (B) Long short-term memory network. (C) Stacked long short-term memory network.

t-1 to the cell state C_t at the current time t.

$$f_t = \sigma(w_f x_t + u_f h_{t-1} + b_f) \quad (8)$$

2. The input gate is used to control the ratio of the input x_t at the current time t to the cell state C_t at the current time t.

$$i_t = \sigma(w_i x_t + u_i h_{t-1} + b_i) \quad (9)$$
$$C_t^{\sim} = tanh(w \times x_t + u \times h_{t-1} + b) \quad (10)$$
$$C_t = f_t \times C_{t-1} + i_t + i_t \times C_t^{\sim} \quad (11)$$

3. The output gate is used to control the ratio of the unit state C_t at the current time t to the output value h_t of the LSTM neural network.

$$O_t = \sigma(w_o \times h_{t-1} + w_o \times x_t + b_o) \quad (12)$$
$$h_t = tanh(C_t) \times O_t \quad (13)$$

2) STACKED LSTM

The shallower neural network model may have difficulty achieving the desired effect of prediction, and the prediction ability of a single-layer LSTM structure is limited. In recent years, deep learning has performed well. Therefore, increasing the depth of the neural network model can improve performance.

A standard LSTM unit completes all functions and has three parts:

1. Mapping from the input layer to the hidden layer. The input information of each time step will be matrix-mapped, and then the mapped content will be used as the input of the forget gate, input gate, unit state and output gate.
2. Mapping from the hidden layer to the hidden layer, including the calculation of the forget gate, input gate, output gate, and unit state updates.
3. Mapping from the hidden layer to the output layer.

To increase the depth of the LSTM neural network model, on the basis of the original LSTM network, we use the output of the previous LSTM and the mapping from the input layer of the previous LSTM to the hidden layer as the input of the next layer. Multi-layer LSTM network stacking (stacked LSTM) increases the depth of the model and improves performance. Figure 4(B) and (C) shows the structure of the LSTM and the stacked LSTM [34], respectively.

C. THREE-DIMENSIONAL VISUALIZATION BY PRELOADING DATA

According to the location of the viewpoint, the data domain is divided using the frustum [35] clipping method. The data in the current display area are loaded into memory, and the GPU is used for rendering and display. Because the viewpoint changes in motion, we must use the position of the predicted viewpoint combined with the frustum clipping method to divide the potential area and the unloading area, load the potential area in advance, and unload the data blocks in the unloading area to reduce the feeling of freezing.

In the process of establishing the viewpoint motion model, it is necessary to use the viewing cone to judge the visual field. Knowing the coordinates $P_n(x_n, y_n, z_n)$ of the viewpoint at the current moment, combined with the constraints α, β and d_0 (where α is the opening angle of the viewing cone along the x direction, β is the viewing angle of the viewing cone along the y direction, and d_0 is the distance from the viewpoint to the viewing plane *EFGH*), we can construct the

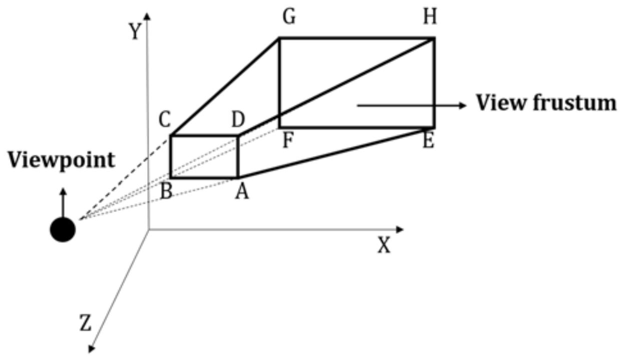


FIGURE 5. Frustum model.

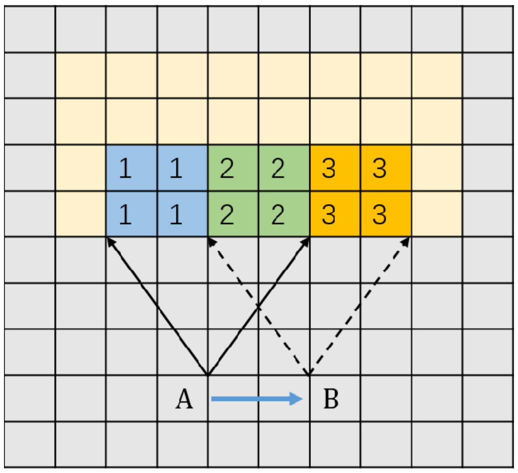


FIGURE 6. Schematic diagram of regional division.

view cone *ABCDEFGH*, Figure 5 shows a schematic view of the frustum. Thus, the spatial plane equations of the six planes, namely *ABCD, ADHE, BFGC, EFGH, CDHG* and *ABFE* of the viewing cone are obtained. Each plane can be represented by $A_i x + B_i y + C_i z + D_i = 0$.

It is specified that pointing to the inside of the viewing cone is the positive direction. The coordinates of each MBC center are divided into six equations. If the obtained results are all positive, the MBC is divided into the visible domain and imported into memory. For an MBC block, if more than half of the vertices are located inside the viewing frustum, it is considered that most of the MBC data are located inside the viewing frustum, and must be rendered and displayed.

According to the Figure 6, the model dynamically divides the entire data volume into the visible domain, potential domain, and offload domain as the viewpoint moves from A to B. The visible domain is the visible data domain (areas 1 and 2) from the current viewpoint, which must be rendered and displayed in time for the user to browse; the potential domain is the visible data domain (areas 2 and 3) under the predicted angle of view at the next moment. The data block in this area is loaded into memory in advance, which makes it convenient for the user to read and display directly during continuous browsing. Assuming that the visible domain data block is N, the potential domain data block is P, and the unloading

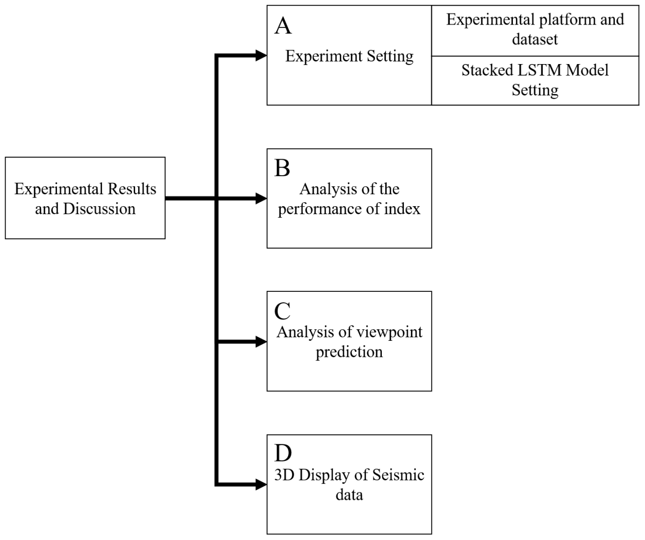

A
Experiment Setting
Experimental platform and dataset
Stacked LSTM Model Setting
Experimental Results and Discussion
B
Analysis of the performance of index
C
Analysis of viewpoint prediction
D
3D Display of Seismic data

FIGURE 7. Overview of experimental results and discussion. Experimental criterions are derived by the confusion matrix. (A) Experimental setting. (B) Analysis of the performance of index. (C) Analysis of viewpoint prediction. (D) 3D display of seismic data.

domain data block is D, then we have

$$D = N - N \cap P \tag{14}$$

When viewpoint A moves to B, data domain 1 changes from the visible domain to the offload domain, and data domain 3 becomes the visible domain simultaneously.

III. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we present the experimental details. Figure 6 shows the entire experimental procedure divided into four parts. First, we introduce some basic experimental settings, and then we analyze the performance of the index method and viewpoint prediction method according to the experimental data. Finally, we analyze the experimental results of the 3D display using our method.

A. *EXPERIMENT SETTING*

1) EXPERIMENTAL PLATFORM AND DATASET

The hardware configurations used in the experiment were as follows: an Intel Core i5-7300 processor, 8 GB of memory, and an NVIDIA GTX 1050 GPU with a memory size of 4 GB. The software systems were Windows 10, an independently developed 3D seismic data visualization system developed by PyCharm, Visual C++, and OpenGL using the cross-platform development tool QT5.

This study tested the 3D scientific data visualization system, as shown in Figure 6. The 3D data tested were the seismic data obtained from the work area in Huabei Oilfield, China, which are stored in the SEG-Y file format. We had three groups of test data with different sizes: Group A data were 469.9 MB, Group B data were 3348.48MB, and Group C data were 14643.2 MB. The size of every voxel was 4 bytes. The data included information such as the buried depth, extent, thickness, top-bottom interface, and extension

TABLE 1. Comparison of number of sub-blocks of the multiple algorithms for the given three sets of data.

Data	Number of sub-blocks				$\frac{N_4}{N_1}$	$\frac{N_4}{N_2}$	$\frac{N_4}{N_3}$
	OCT(N_1)	HRT(N_2)	KHRT(N_3)	CKHRT(N_4)			
Group A	540	382	354	316	58.5%	82.7%	89.2%
Group B	3329	2266	2073	1831	55.0%	80.8%	88.3%
Group C	14581	10538	9959	8915	61.1%	84.6%	89.5%

TABLE 2. Comparison of query time of the multiple algorithms for the given three sets of data.

Data	Method	Query the percentage of seismic data blocks		
		2%	4%	6%
Group A	OCT query time(s)	2.42	5.82	8.80
	HRT query time(s)	1.84	4.48	7.03
	KHRT query time(s)	1.41	3.42	5.64
	CKHRT query time(s)	0.88	2.13	3.25
Group B	OCT query time(s)	28.21	69.18	115.19
	HRT query time(s)	26.72	66.44	108.73
	KHRT query time(s)	17.08	46.79	76.35
	CKHRT query time(s)	9.69	26.76	41.70
Group C	OCT query time(s)	50.10	147.06	216.38
	HRT query time(s)	41.79	121.47	194.99
	KHRT query time(s)	32.15	92.61	152.33
	CKHRT query time(s)	16.02	46.67	75.86

trend of the geological body. The dataset for training the stacked LSTM model was a large number of 3D coordinate sequences. These sequences were equally spaced sampling points of the motion trajectory of the viewpoint. These sampling points are the viewpoint coordinates derived by the software when professionals continue to browse the 3D display of seismic data.

2) STACKED LSTM MODEL SETTING

We select some continuous motion trajectory viewpoint coordinates from the dataset, each of which has three dimensions (x, y, z), which we use as the inputs to the model. We connect the LSTM layer behind the input layer, where the number of LSTM units will have a different values (10, 32, 64, 100, 128, 160 and 256). We select a parameter dropout of 0.2. A dropout

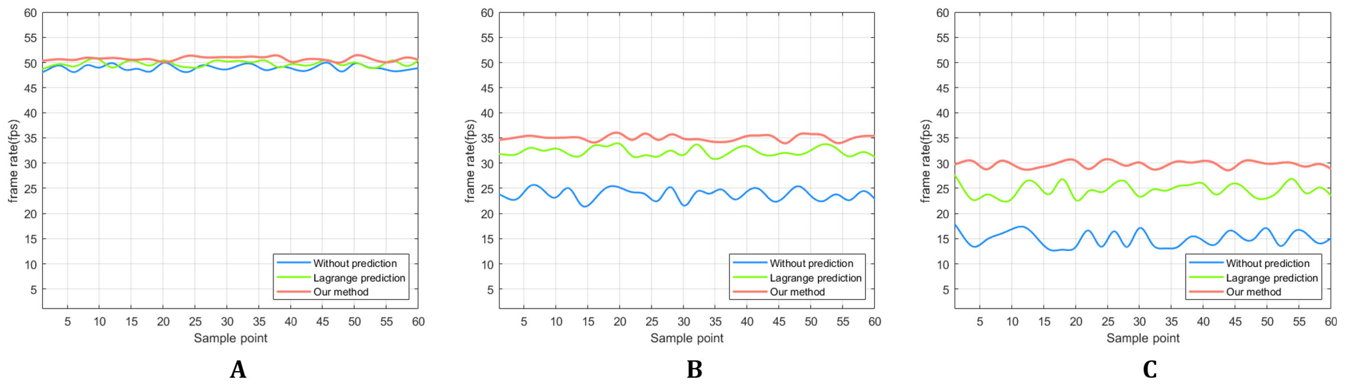


FIGURE 8. Comparison of frame rate of the multiple algorithms by selecting 60 sets of continuous FPS values in experiment of different data. (A) is group A data. (B) is group B data. (C) is group C data.

layer [36], [37] is connected after the LSTM layer with a parameter of 0.5. Then, the LSTM layer, the dropout layer, the LSTM layer, the dropout layer, the dense layer and the output layer are continuously connected behind the network. The output layer outputs a 3D vector, which is the coordinate of the next viewpoint.

B. ANALYSIS OF THE PERFORMANCE OF THE INDEX

1) COMPARISON OF NUMBER OF SUB-BLOCKS

In the 3D data visualization method based on a data block, the number of sub-blocks determines the efficiency of the data query in the 3D display process. The test methods were as follows: the OCT, Hilbert R-tree (HRT), Hilbert R-tree based on *k-means* clustering (KHRT), and Hilbert R-tree based on the *CUK* (CKHRT), which were used to establish the indexes. To prove the advantages of this method in data indexing efficiency, we used the above methods and our method to index three groups of different sized data. The results are shown in Table 1.

It can be observed from the comparison results in the table that, compared with the OCT index, the CKHRT index can reduce the number of sub-blocks by 38.9% to 45%. Compared with the traditional HRT index, this method reduces the number of sub-blocks by 15.4% to 19.2%. Compared with the KHRT index, the method reduces the amount of sub-block data by 10.5% to 11.7%. The reason this method is more effective in reducing the number of sub-blocks is that the Hilbert R-tree used here has high space utilization, and the use of the CUK will have the same or similar voxel values, which are divided into the same sub-block to achieve this effect.

2) COMPARISON OF QUERY TIME

The access efficiency of the tree determines the performance of the data index during the display process. From the three groups of seismic data (A, B, and C), the data blocks with the same proportion were selected by four methods to make multiple queries, record the query results, and calculate the average query time, as shown in Table 2.

The group B data are 3D volume data with a relatively uniform distribution, and the group C data contain data with

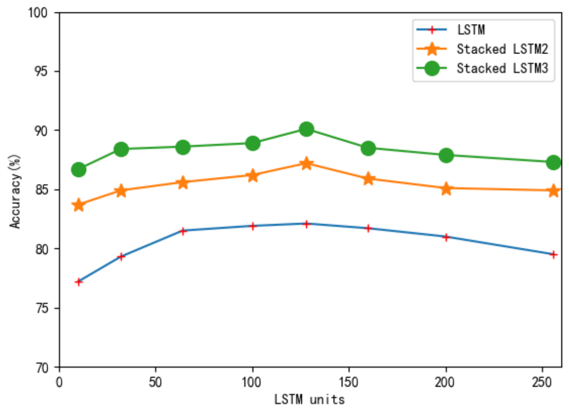


FIGURE 9. Average accuracy curve for LSTM and stacked LSTM with different LSTM units.

a scattered distribution. As can be observed from Table 1, for the group B data, compared with those obtained by KHRT, the time of the query sub-block of the Hilbert R-tree based on the CUK was reduced by 42.8% to 45.3%, and for the group C data, their query time was reduced by 49.6% to 50.2%. The sub-block index method proposed in this paper performs a CUK operation on the original data, so that data objects with close distances are gathered together to generate adjacent leaf nodes and then stored in adjacent locations, thus effectively reducing disk I/O access time and increasing index speed. Especially in the case of scattered 3D data volume, the spatial retrieval efficiency of the proposed method is more obvious.

C. ANALYSIS OF VIEWPOINT PREDICTION

Figure 9 shows the average accuracy curves for the single-layer LSTM network, two-layer LSTM network, and three-layer LSTM network with different numbers of hidden units. The following characteristics were observed. (1) The average accuracy may decrease when the number of hidden units is either very small or many. (2) The average accuracy may decrease when the number of LSTM layers increases slightly. If the LSTM layer is very large, the accuracy will decrease.(3) The accuracy of 128 LSTM units is the highest. However, we choose 32 LSTM units. Compared with

TABLE 3. Comparison of correct rate of the multiple algorithms for the given three sets of data.

Data	Method	Correct rate (%)		
		1 (min)	5 (min)	10 (min)
Group A	Lagrange interpolation	82.53	75.45	70.81
	Proposed method	92.37	88.96	87.57
Group B	Lagrange interpolation	80.49	72.18	66.93
	Proposed method	90.24	87.41	85.71
Group C	Lagrange interpolation	71.33	65.97	61.66
	Proposed method	87.55	84.67	80.02

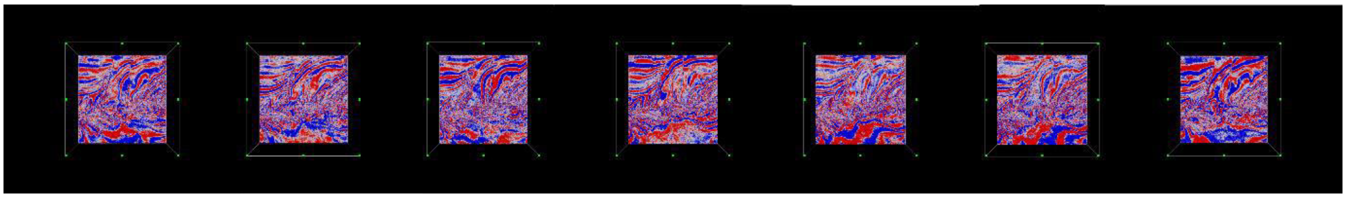


FIGURE 10. The slice of depth side line profile of seismic data.

TABLE 4. The data is sliced using four algorithms, compare the time that the sliced data fully rendering.

Method		Slice number (No.)						
		252	262	272	282	292	302	312
OCT+SLSTM	Time(s)	2.15	2.06	2.10	2.12	1.95	2.03	2.13
HRT+SLSTM	Time(s)	1.09	1.07	1.02	1.05	1.19	1.17	1.10
KHRT+SLSTM	Time (s)	1.01	1.04	1.06	1.03	1.05	1.00	1.03
CKHRT+SLSTM	Time (s)	0.45	0.44	0.45	0.47	0.46	0.45	0.45

128 LSTM units, the accuracy of the three-layer LSTM network is not much different, but the amount of network parameters is reduced a lot, which makes the real-time effect of our system better.

To verify the effect of the viewpoint motion model, we tested the three sets of data with the method without viewpoint prediction, the Lagrangian viewpoint prediction method and the stacked LSTM viewpoint prediction method. We selected 60 sets of continuous FPS values from the experiment and compared them.

The statistics are shown in Figure 8. As can be observed from the figure, although the motion trajectory becomes complicated, our algorithm can still maintain a relatively stable frame rate. In addition, the stability is better than Lagrange algorithms. The experimental results also showed that the frame rate obtained by the motion prediction algorithm is more stable than that obtained without prediction. Therefore, our algorithm preloads data blocks that are smaller and renders them smoother.

In addition, we selected the X-axis component of the sampling point for comparison, and evaluate the prediction accuracy of the piecewise quadratic Barycentric Lagrange interpolation prediction algorithm and the Lagrange interpolation algorithm. The results are shown in Table 3.

Table 3 shows that the prediction accuracy of the proposed algorithm is 9.75% to 18.78% higher than that of the Lagrange interpolation algorithm, which has better prediction performance. When the accuracy of the prediction increases, the higher the proportion of data in the correct range loaded by the computer in advance, the smaller the amount of data that the computer needs to reload into the memory before displaying the screen. Therefore, this reduces consumption of system resource. The method of motion prediction effectively reduces the fluctuation of the frame rate, avoids the stuck phenomenon during viewpoint motion, and provides the possibility for real-time visualization of large-volume 3D data.

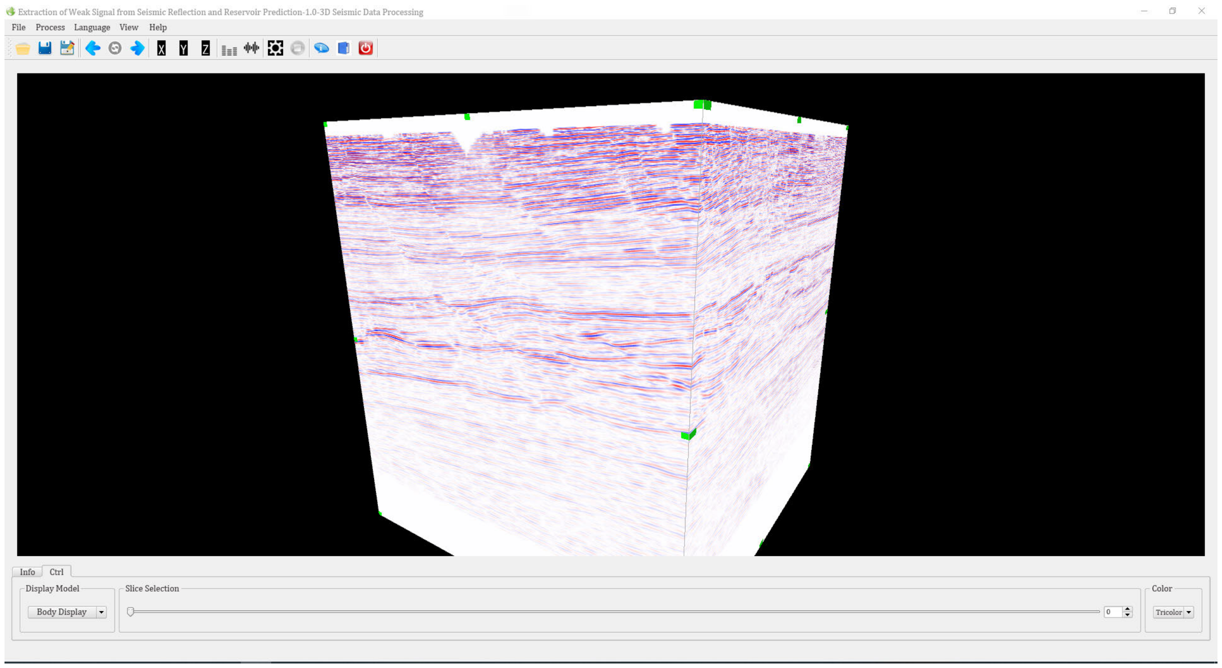

Extraction of Weak Signal from Seismic Reflection and Reservoir Prediction-1.0-3D Seismic Data Processing
File
Process
Language
View
Help
X
Y
Z
Info
Ctrl
Display Model
Body Display
Slice Selection
0
Color
Tricolor

FIGURE 11. The 3D display of seismic data of group B (3.27G).

D. THREE-DIMENSIONAL DISPLAY OF SEISMIC DATA

1) COMPARISON OF INTERACTION QUALITY

In this experiment, we combined the four indexing strategies with the stacked LSTM model to test the interactive performance of our algorithm. Taking the B group data as an example, we sliced seismic data is at intervals of 10. The depth side line profiles of the seismic data are shown in Figure 10. We recorded the time spent by the four sets of algorithms displaying the slice results, shown in Table 4. It can be clearly observed from the table that our algorithm has the best interactive performance, and its display speed increased by about 3.58, 1.43 and 1.27 times compared with those of the other three algorithms.

2) QUALITY OF 3D DISPLAY

The following figure 11 shows the result of the 3D display of the experimental data of group B using the experimental method. It can be observed from the figure that the quality of our 3D display is high, the texture is clear, the picture resolution is high, and the relationship between the stratigraphic structures can be displayed more clearly.

IV. CONCLUSION AND FUTURE WORK

In this study, we used the optimized Hilbert R-tree to establish a fast indexing of data and used a deep learning network model to predict the viewpoint to realize the pre-loaded data function and the fast 3D display of massive seismic data. In the indexing process, we establish an efficient index structure using the Hilbert R-tree improved by the *CUK* based on the data in the file. Compared with traditional indexing methods, the proposed method can prevent leaf nodes from overlapping greatly. In the display process, the frequent I/O operations between the internal memory and the disk are reduced, and the aim to improve display efficiency is achieved. In the pre-loading method based on viewpoint prediction, we use the stacked LSTM model to predict the viewpoint, crop the 3D volume data according to the current viewpoint and the predicted viewpoint, divide the unloading domain and the potential domain, preload the data to be displayed in advance, and start unloading from memory the data that will not be displayed. This method utilizes the characteristics of high prediction accuracy of the deep learning network model, reduces the error rate of the preload area and the size of the loaded data block, and realizes smoother rendering and real-time interactive display.

However, our proposed method also has some limitations. If the view point moves too fast or moves irregularly when viewing a 3D display, it is likely to increase error rate in the point of view prediction and cause browsing pauses. In addition, the number of Stacked LSTM layers and the number of units is too large, which leads to an increase in the number of parameters and in model calculations. This will result in frozen display. Therefore, we need to make an appropriate adjustment to its setting, which compromise the amount of calculation and the accuracy of prediction, according to different application conditions.

With the development of visualization research, many fields have begun to shift from the study of two-dimensional (2D) visualization to 3D visualization. Three-dimensional visualization has good visual effects and rich expressive power, which can intuitively convey more information and overcome some limitations of 2D visualization. Meanwhile, with the development of data acquisition technology, 3D data are becoming increasingly abundant and the amount of data is also increasing. Therefore, we must continuously study the 3D visualization method to optimize the data index method and the accuracy of preloading.

ACKNOWLEDGMENT

(Huan Cheng, Kai Xie, and Jian-Biao He contributed equally to this work.)

V. AUTHOR CONTRIBUTIONS

Huan Cheng conceived and initialized the research, conceived the algorithms, and designed the experiments; Kai Xie reviewed the paper; Chang Wen conducted comparative experiments; and Jian-Biao He checked the spelling and provided advice.

REFERENCES

[1] J. Beyer, A. Al-Awami, N. Kasthuri, J. W. Lichtman, H. Pfister, and M. Hadwiger, ''ConnectomeExplorer: Query-guided visual analysis of large volumetric neuroscience data,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 12, pp. 2868–2877, Dec. 2013.

[2] Y. Gu, C. Wang, T. Peterka, R. Jacob, and S. Hyun Kim, ''Mining graphs for understanding time-varying volumetric data,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 1, pp. 965–974, Jan. 2016.

[3] H. Tomasz, ''Framework for cognitive analysis of dynamic perfusion computed tomography with visualization of large, volumetric data,'' *J. Electron. Imag.*, vol. 21, no. 4, p. 3017, Nov. 2012.

[4] E. Lamar, B. Hamann, and K. I. Joy, ''MultiMultiresolution techniques for interactive texture-based volume visualization,'' *Proc. SPIE*, vol. 3960, pp. 105–114, Nov. 2000.

[5] M. Weiler, ''Level-of-detail volume rendering via 3D textures,'' in *Proc. IEEE Symp. Volume Vis.*, New York, NY, USA, Oct. 2000, pp. 7–13.

[6] J. Shen, Y. Luo, Z. Wu, Y. Tian, and Q. Deng, ''CUDA-based real-time hand gesture interaction and visualization for CT volume dataset using leap motion,'' *Vis. Comput.*, vol. 32, no. 3, pp. 359–370, Feb. 2016.

[7] J. E. Stone, M. Sener, K. L. Vandivort, A. Barragan, A. Singharoy, I. Teo, J. V. Ribeiro, B. Isralewitz, B. Liu, B. C. Goh, J. C. Phillips, C. MacGregor-Chatwin, M. P. Johnson, L. F. Kourkoutis, C. N. Hunter, and K. Schulten, ''Atomic detail visualization of photosynthetic membranes with GPU-accelerated ray tracing,'' *Parallel Comput.*, vol. 55, pp. 17–27, Jul. 2016.

[8] H. Zou, F. Lin, J. Han, and W. Zhang, ''GPU-based medical visualization for large datasets,'' *J. Med. Imag. Health Informat.*, vol. 5, no. 7, pp. 1467–1473, Dec. 2015.

[9] M. B. Rodrgí uez, ''State-of-the-art in compressed GPU-based direct volume rendering,'' *Comput. Graph Forum*, vol. 33, no. 6, pp. 77–100, Feb. 2014.

[10] M. Isenburg, P. Lindstrom, and H. Childs, ''Parallel and streaming generation of ghost data for structured grids,'' *IEEE Comput. Graph. Appl.*, vol. 30, no. 3, pp. 32–44, May 2010.

[11] J. Beyer, M. Hadwiger, S. Wolfsberger, and K. Buhler, ''High-quality multimodal volume rendering for preoperative planning of neurosurgical interventions,'' *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 6, pp. 1696–1703, Nov. 2007.

[12] I. Boada, I. Navazo, and R. Scopigno, ''Multiresolution volume visualization with a texture-based octree,'' *Vis. Comput.*, vol. 17, no. 3, pp. 185–197, May 2001.

[13] 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.

[14] P. Jin, ''Optimizing r-tree for rash memory,'' *EXPERT Syst. Appl.*, vol. 42, no. 10, pp. 4676–4686, Jun. 2015.

[15] Q. Du and X. Li, ''A novel KNN join algorithms based on Hilbert R-tree in MapReduce,'' *Iccsnt*, vol. 2013, pp. 417–420, Nov. 2014.

[16] I. Kamel and C. Faloutsos, ''Hilbert R-tree: An improved r-tree using fractals,'' in *Proc. 20th VLDB Conf.*, 1994, pp. 500–509.

[17] S. Muraki, ''Volume data, and wavelet transform,'' *IEEE Comput. Graph*, vol. 13, no. 4, pp. 50–56, Jul. 1993.

[18] S. Guthe, ''Interactive rendering of large volume data sets,'' in *Proc. IEEE Vis.*, Washington, DC, USA, Dec. 2002, pp. 53–60.

[19] K. G. Nguyen and D. Saupe, ''Rapid high quality compression of volume data for visualization,'' *Comput. Graph. Forum*, vol. 20, no. 3, pp. 49–57, Sep. 2001.

[20] M. B. Rodríguez, ''A survey of compressed GPU-based direct volume rendering,'' in *Proc. Eurographics*, May 2013, pp. 117–136.

[21] V. Pascucci and R. J. Frank, ''Hierarchical indexing for out-of-core access to multi-resolution data,'' in *Proc. Math Vis.* Berlin, Germany: Springer, 2001, pp. 225–241.

[22] W. T. Correa, J. T. Klosowski, and C. T. Silva, ''Visibility-based prefetching for interactive out-of-core rendering,'' in *Proc. IEEE Symp. Parallel Large-Data Vis. Graphic*, Oct. 2003, pp. 1–8.

[23] P. S. Wang, Y. Liu, Y. X. Guo, C. Y. Sun, and X. Tong, ''O-CNN: Octree-based convolutional neural networks for 3D shape analysis,'' *ACM Trans. Graph.*, vol. 36, no. 4, p. 72, Jul. 2017.

[24] A. R. Fuller, ''Comparison of real-time visualization of, volumetric OCT data sets by CPU-slicing and GPU-ray casting methods,'' *Proc. SPIE*, vol. 7163, Feb. 2011, Art. no. 716312.

[25] K. Xie, P. Wu, and S. Yang, ''GPU and CPU cooperation parallel visualisation for large seismic data,'' *Electron. Lett.*, vol. 46, no. 17, pp. 1196–1197, Aug. 2010.

[26] A. Guttman, ''R-trees:A dynamic index structure for spatial searching,'' in *Proc. ACMSIGM*, Boston, MA, USA, 1984, pp. 47–57.

[27] S. T. Leutenegger, M. A. Lopez, and J. Edgington, ''STR: A simple and efficient algorithm for R-tree packing,'' in *Proc. 13th Int. Conf. Data Eng.*, Engineering, UK, Dec. 1997, pp. 497–506.

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

[29] G. S. Michaels, ''Cluster analysis and data visualization of large- scale gene expression data,'' *Pac Symp Biocomput*, vol. 3, pp. 42–53, Feb. 1998.

[30] H. Y. Cui, S. Li, and L. P. Zhang, ''R-tree constructionbuilt based on CUK-MEANS algorithm,'' *J. Chin. Comput. Syst.*, vol. 37, no. 2, p. 15, Feb. 2016.

[31] C. Wen, L. Li, and K. Xie, ''Fast visualisation of massive data based on viewpoint motion model,'' *Electron. Lett.*, vol. 53, no. 15, pp. 1038–1040, Jul. 2017.

[32] J. Zhang and K. F. Man, ''Time series prediction using RNN in multidimension embedding phase space,'' in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, San Diego, CA, USA, Feb. 1988, pp. 1868–1873, doi: 10.1109/ICSMC.1998.728168.

[33] F. A. Gers, J. Schmidhuber, and F. Cummins, ''Learning to forget: Continual prediction with LSTM,'' in *Proc. 9th Int. Conf. Artif. Neural Netw.*, Edinburgh, U.K., 1999, pp. 850–855, doi: 10.1049/cp:19991218.

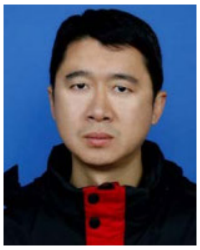
[34] Y. Heryadi and H. L. H. S. Warnars, ''Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, stacked LSTM, and CNN-LSTM,'' in *Proc. IEEE Int. Conf. Cybern. Comput. Intell.*, Phuket, U.K., Nov. 2017, pp. 84–89, doi: 10.1109/CYBERNETICSCOM.2017.8311689.

[35] Y. Jiang and P. Niu, ''View frustum culling algorithm for scene based on adaptive binary tree,'' in *Proc. 2nd IEEE Adv. Inf. Manage., Commun., Electron. Autom. Control Conf. (IMCEC)*, Xi'an, China, May 2018, pp. 1159–1164, doi: 10.1109/IMCEC.2018.8469508.

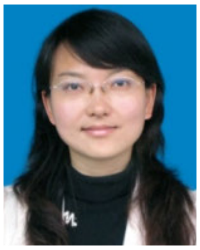
[36] A. Chatterjee, M. W. Gerdes, and S. G. Martinez, ''Statistical explorations and univariate timeseries analysis on COVID-19 datasets to understand the trend of disease spreading and death,'' *Sensors*, vol. 20, no. 11, p. 3089, May 2020.

[37] V. T. C. Pham Bluche Kermorvant and J. Louradour, ''Dropout improves recurrent neural networks for handwritingrecognition,'' in *Proc. Int. Conf. Frontiers Handw. Recognit.*, 2020, pp. 285–290.

HUAN CHENG joined the National Demonstration Center for Experimental Electrical and Electronic Education in 2019, in order to research deep learning and image processing. He is currently an Assistant Researcher with Yangtze University, Jingzhou, China. He is absorbed in image processing and artificial intelligence. His current research interests include image recognition and 3-D visualization.



KAI XIE received the M.S. degree in electronic engineering from the National University of Defense Technology, Changsha, China, in 2003, and the Ph.D. degree in pattern recognition and intelligent system from Shanghai Jiao Tong University, Shanghai, China, in 2006. He is currently a Professor with the School of Electronic Information, Yangtze University, Jingzhou, China. His current research interests include image processing and signal processing.



CHANG WEN received the B.S. degree in computer science from the Naval University of Engineering, Wuhan, China, in 2002, and the M.S. degree in computer science from Yangtze University, Jingzhou, China, in 2008. She is currently an Assistant Professor with the School of Computer Science, Yangtze University. Her current research interests include image processing and signal processing.

JIAN-BIAO HE received the B.S. and M.S degrees from the Huazhong University of Science and Technology, Wuhan, China, in 1986 and 1989, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Central South University. His research interests include artificial intelligence, the Internet of Things, pattern recognition, mobile robots, and cloud computing.