Article

A New Algorithm for Large-Scale Geographically Weighted Regression with K-Nearest Neighbors

Xiaoyue Yang [1], Yi Yang [1,*], Shenghua Xu [2], Jiakuan Han [1], Zhengyuan Chai [1] and Gang Yang [1]

[1] School of Marine Technology and Geomatics, Jiangsu Ocean University, Lianyungang 222005, China
[2] Research Center of Geospatial Big Data Application, Chinese Academy of Surveying and Mapping, Beijing 100036, China
[*] Correspondence: yangyi@jou.edu.cn

Abstract: Geographically weighted regression (GWR) is a classical method for estimating nonstationary relationships. Notwithstanding the great potential of the model for processing geographic data, its large-scale application still faces the challenge of high computational costs. To solve this problem, we proposed a computationally efficient GWR method, called K-Nearest Neighbors Geographically weighted regression (KNN-GWR). First, it utilizes a k-dimensional tree (KD tree) strategy to improve the speed of finding observations around the regression points, and, to optimize the memory complexity, the submatrices of neighbors are extracted from the matrix of the sample dataset. Next, the optimal bandwidth is found by referring to the spatial clustering relationship explained by K-means. Finally, the performance and accuracy of the proposed KNN-GWR method was evaluated using a simulated dataset and a Chinese house price dataset. The results demonstrated that the KNN-GWR method achieved computational efficiency thousands of times faster than existing GWR algorithms, while ensuring accuracy and significantly improving memory optimization. To the best of our knowledge, this method was able to run hundreds of thousands or millions of data on a standard computer, which can inform improvement in the efficiency of local regression models.

Keywords: GWR; K-Nearest Neighbors; KD tree; K-means clustering; large-scale geographic data

check for updates

Citation: Yang, X.; Yang, Y.; Xu, S.; Han, J.; Chai, Z.; Yang, G. A New Algorithm for Large-Scale Geographically Weighted Regression with K-Nearest Neighbors. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 295. https://doi.org/10.3390/ijgi12070295

Academic Editor: Wolfgang Kainz

Received: 17 May 2023
Revised: 17 July 2023
Accepted: 18 July 2023
Published: 21 July 2023

1. Introduction

Owing to the influence of geographical location, changes in the relationship or structure between variables is an important issue in spatial data analysis. The geographically weighted regression (GWR) model, originally developed by Fotheringham and Brunsdon [1,2], is a common tool for exploring spatial non-smoothness. Currently, GWR is widely used in a variety of fields, such as geology [3–5], ecology [6,7], house price modeling [8–11], epidemiology [12–14], and environmental science [7,15]. The popular GWR packages are Spgwr [16], MGWR (PySAL) [17], GWmodel [18], and FastGWR [19]. With the expansion of demand for geographic information technology and resources in various industries [20,21], geographic data with high spatial and temporal resolution have seen explosive growth while promoting innovation in GIS methods [22–26]. Many studies have explicitly reported the computational limitations of GWR when oriented to large-scale geographic data [19,27] in order to fully utilize the geographical information.

Each regression point in GWR is regressed individually based on the distance matrix, and the optimal bandwidth selection is performed before that, which makes it computationally intensive and requires a large amount of memory usage. However, the application of GWR to extract information from large-scale geographic data is difficult. Harris estimated that it would take two or more weeks to complete the experiment using the Spgwr package for a dataset of one hundred thousand points (and five predictor variables) [28]. Li pointed out that the maximum number of records that can be handled by the current open-source GWR software is approximately fifteen thousand observations on a standard desktop [19]. In Yu's experiment [29], random sampling of a house price dataset was forced to reduce the calculation cost owing to the computational demand of building a GWR model for the 68,906 house price dataset; therefore, 3437 data were selected for the GWR calculation. Although this method addresses the problem of computational volume, it can lead to incomplete exploitation of data information. Feuillet provided a method to deal with a large sample dataset by building GWR sub-models [30]. Wang proposed an improved method based on a computational unified device architecture (CUDA) parallel architecture that can handle GWR corrections for millions of data points [31]. Tasyurek reverse-nearest-neighbor geographically weighted regression (RNN-GWR) only calculates updated data points when dealing with frequently updated datasets, which can result in a higher computational efficiency with guaranteed calculation results [32]. Although many efforts have been made, there is still a lack of an effective GWR algorithm that can analyze large numbers of geographic data in a limited timeframe. Currently, data sets with millions of observations are becoming increasingly common. During the analysis of the GWR regression process, we found that for each regression point, a certain distance (bandwidth) radius range of observation points is analyzed, and therefore observations outside of this range are not considered in the calculation. Additionally, the current optimal bandwidth search range is typically global, and reducing unnecessary calculations can alleviate the computational burden of GWR. Thus, implementing new improvements to the GWR algorithm are necessary to address computational bottlenecks, enable its application in extremely large datasets, and fully exploit data information.

Currently, the nearest-neighbor indexing method demonstrates better performance in the computational optimization of some models, with the k-dimensional tree (KD tree) being the most commonly used. Meenakshi suggested a new index structure KD tree with a linked list (k-dLst tree) for retrieving spatial records with duplicate keys [33]. To improve the computationally expensive state of performing repeated distance evaluations in the search space, a special tree-based structure (called a KD tree) was used to speed up the nearest-neighbor search [34]. Chi-Ren Shyu developed a web server (named ProteinDBS) for the life science community to search for similar protein tertiary structures in real time, which returned search results in hierarchical order from a database of over 46,000 chains in a few seconds and showed considerable accuracy [35]. Böhm performed a range search by indexing K-Nearest-Neighbor join queries [36]. Muja proposed the Fast Library for Approximate Nearest Neighbors (FLANN) library, which reduced the time for nearest-neighbor search by an order of magnitude [37,38]. The FLANN library has been applied in many studies to improve computational efficiency [37,39]. The advantages of nearest-neighbor indexing for optimal computation have been demonstrated in these studies.

When dealing with large amounts of data, clustering methods can help to better understand the distribution patterns and structure of the data, thus revealing hidden information in the data [40,41]. K-means, a widely-used clustering algorithm, partitions a dataset into K clusters and assigns each data point to the nearest cluster. This method has been extensively implemented to analyze data in various domains [40,42]. Macqueen argued that spatial clustering uses spatial location and relationships as feature terms to discover spatial clustering relationships [43,44]. Li used K-means clustering to analyze the impact of building environmental factors on the variation of rail ridership in the study area and proposed differentiated planning guidance for different regions [45]. Hernández employed the K-means clustering algorithm to identify distinct clusters of tourism types within large geographic areas [46]. Deng applied a combination approach of geographically weighted regression and K-means clustering to partition the study area into distinct regions and devise regional policies to mitigate PM2.5 concentrations [47]. The aforementioned studies all indicate that K-means clustering has significant advantages in discovering the spatial distribution of geographic data.

The main objective of this study is to develop a method that can quickly calibrate GWR to overcome the challenges of processing large-scale geographic data. This paper contributes to the previous literature as follows. (**1**) Using a K-D tree to accelerate the speed of searching for observation points around the regression point. In the GWR model, searching for neighboring points usually requires traversing the entire dataset, leading to significant time consumption as the dataset size increases. Therefore, this paper attempts to incorporate the KD tree into the GWR model to take advantage of its fast search capabilities to quickly identification of surrounding observation points in each local regression process. (**2**) Transforming the large matrix into a small matrix in the regression process. When using the Bi-square kernel function to calculate the weight matrix, the full sample matrix can cause significant memory and time consumption. Therefore, the matrices of independent variables, dependent variables, and weight matrices involved in the regression are transformed into corresponding small matrices according to the bandwidth. (**3**) Optimizing the search range of optimal bandwidth. The optimal bandwidth in GWR is extracted in the process of global traversal, which is time-consuming. This may be caused by considering only the local scale of operation of the model represented by the bandwidth and ignoring the spatial relationships implied by the bandwidth at local spatial locations. Therefore, this paper attempts to use K-means to cluster the geographic location data and refer to the obtained clustering results to limit the search range of the optimal bandwidth. Finally, this paper incorporates the KD tree index and K-means clustering into the GWR model, restructuring the matrices of independent variables, dependent variables and weight matrices involved in the calculation, and proposes a new model called KNN-GWR. The potential of using KNN-GWR models for geographically weighted regression on large-scale geographic data was explored in this paper, using simulated data and a dataset of house prices in selected regions of China.

The remainder of this paper is organized as follows. In Section 2, we introduce the proposed algorithm. In Section 3, we compare the performance of KNN-GWR with other GWRs using simulated datasets and house price datasets. In Section 4, conclusions and discussions are presented.

2. Method

2.1. Geographically Weighted Regression

GWR is used as a local fitting technique, and its regression coefficient varies with the geographical location. The mathematical expression is as follows:

$$y_i = \beta_0(u_i, v_i) + \sum_{j=1}^{p} \beta_j(u_i, v_i) x_{ij} + \varepsilon_i \tag{1}$$

where (u_i, v_i) represent the coordinates of the ith point in space, $\beta_0(u_i, v_i)$ denotes the intercept value, and $\beta_j(u_i, v_i)$ is the spatial variation coefficient of the ith independent variable. $i \epsilon \{1, 2, \cdots, n\}$, $j \epsilon \{1, 2, \cdots, p\}$, y_i is the response variable at location i, x_{ij} is the jth predictor variable, and ε_i is the error term. n is the size of the sample dataset and p is the number of independent variables. The GWR calibration in matrix form is given by:

$$\beta_i = (X^T W_i\, X)^{-1} X^T W_i Y. \tag{2}$$

where X is a $n \times (p+1)$ matrix of the independent variables, Y is a $n \times 1$ matrix of the response variable, and the matrix of X and Y can be calculated, respectively, by:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \tag{3}$$

And

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{4}$$

where W_i is a space weight matrix and is expressed as:

$$W_i = \begin{bmatrix} w_{i1} & 0 & \cdots & 0 \\ 0 & w_{i2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{in} \end{bmatrix} \tag{5}$$

where W_i is an $n \times n$ diagonal matrix. It is calculated using a specified kernel function and bandwidth (bw). For example, the adaptive bi-square weighting function is as follows:

$$w_{ij} = \begin{cases} \left[1 - \left(\frac{d_{ij}}{bw}\right)^2\right]^2 & d_{ij} \le bw \\ 0 & d_{ij} > bw \end{cases} \tag{6}$$

where d_{ij} represents the spatial distance between points i and j. bw represents the adaptive bandwidth, and means that the number of neighbors around regression point i is constant, but the distance is variable.

Where the coefficients β_i of the model can be expressed in matrix form:

$$\beta_i = \begin{bmatrix} \beta_{i0} \\ \beta_{i1} \\ \vdots \\ \beta_{ip} \end{bmatrix} \tag{7}$$

To calibrate the GWR model, a cross-validation (CV) approach is typically used to iterate the bandwidth. In other words, the optimal bandwidth is selected by minimizing the following CV scores:

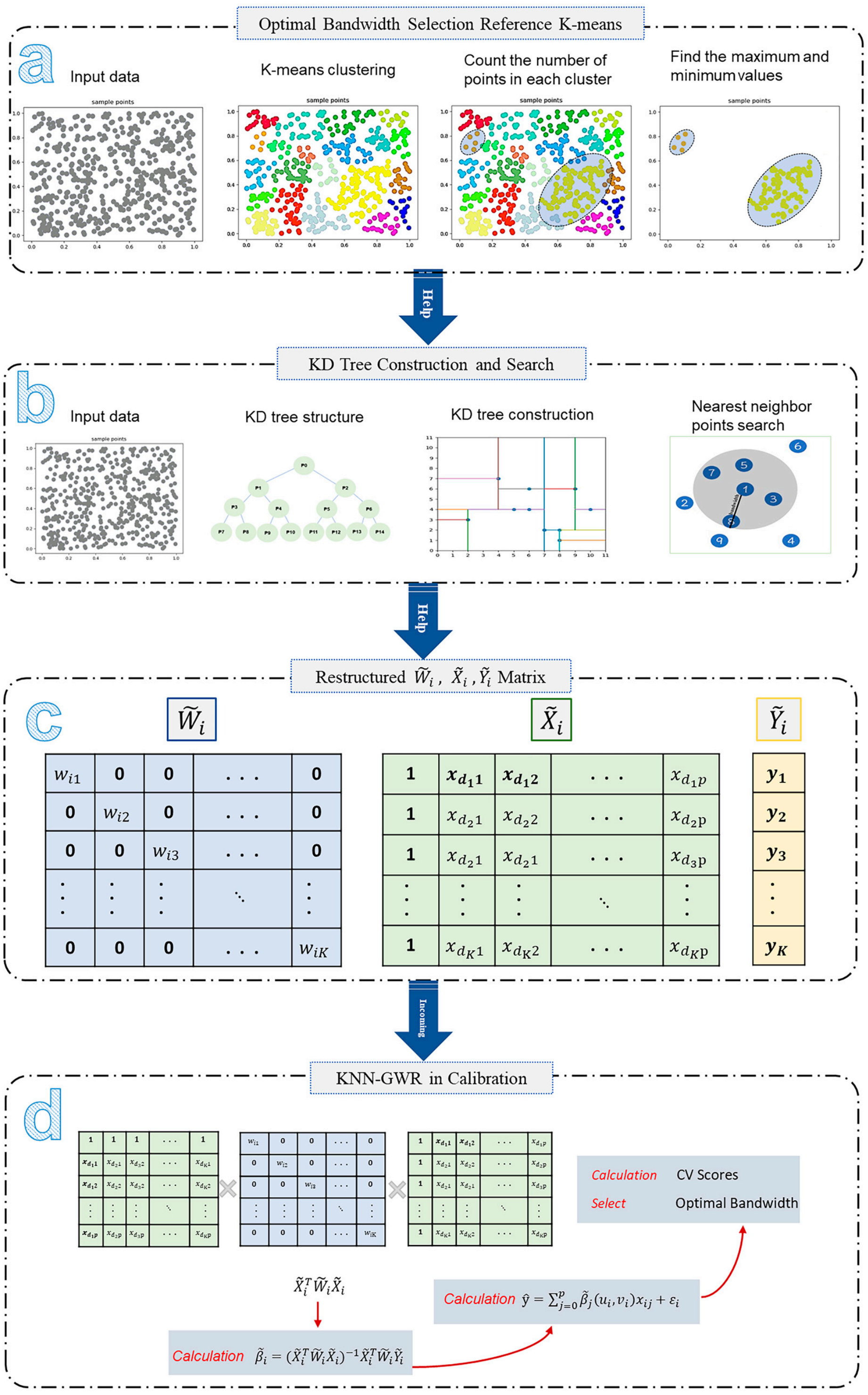
$$CV = \sum_{i=1}^{n} \left[y_i - \hat{y}_{\neq i}(bw)\right]^2, \qquad bw \in (1, n) \tag{8}$$

In general, the traversal of bw ranges from one to n. Because each regression point in the GWR does not participate in its own local regression calculation, $bw \neq 1$. When $bw = n$, the operation scale of the GWR model is global, so $bw \neq n$.

2.2. Geographically Weighted Regression with K-Nearest Neighbors

To speed up the computation and optimize the storage of the weight matrix in the GWR, the KD tree was adopted to find the observation points around the regression point. In this manner, hundreds of thousands or even millions of calculations are filtered. Thus, the large-scale regression problem can be transformed into an acceptable calc value. This method allows GWR to run hundreds of thousands or even millions of data records on an ordinary computer. The flow of the algorithm is illustrated in Figure 1.

The algorithm can be divided into four parts. **Part a**: Optimal bandwidth selection reference K-means. K-means clustering is performed on the incoming data to explain the spatial relationships in the dataset. **Part b**: KD tree construction and search. Establishing the KD tree for the incoming data and searching the observation points around the regression point according to the GWR bandwidth optimization rule based on the results obtained in Part a. **Part c:** Restructured $\widetilde{W}_i$, $\widetilde{X}_i$, $\widetilde{Y}_i$. The weight matrix $\widetilde{W}_i$, the independent variable matrix $\widetilde{X}_i$ and the dependent variable matrix $\widetilde{Y}_i$ of the GWR regression process are reconstructed according to the requirements of each regression point based on the findings in Part b. The reconstructed results are $\widetilde{X}_i$, $\widetilde{Y}_i$, and $\widetilde{W}_i$, respectively. **Part d:** KNN-GWR in calibration. Running the model based on the results of Part c and outputting the results of the optimal bandwidth run to complete the model diagnosis.



Optimal Bandwidth Selection Reference K-means
a
Input data
K-means clustering
Count the number of points in each cluster
Find the maximum and minimum values
Help
KD Tree Construction and Search
b
Input data
KD tree structure
KD tree construction
Nearest neighbor points search
Help
Restructured $\widetilde{W}_i$, $\widetilde{X}_i$, $\widetilde{Y}_i$ Matrix
c
$\widetilde{W}_i$
$\widetilde{X}_i$
$\widetilde{Y}_i$
Incoming
KNN-GWR in Calibration
d
Calculation CV Scores
Select Optimal Bandwidth
$\widetilde{X}_i^T\widetilde{W}_i\widetilde{X}_i$
Calculation $\hat{y} = \sum_{j=0}^{p}\tilde{\beta}_j(u_i,v_i)x_{ij} + \varepsilon_i$
Calculation $\tilde{\beta}_i = (\widetilde{X}_i^T\widetilde{W}_i\widetilde{X}_i)^{-1}\widetilde{X}_i^T\widetilde{W}_i\widetilde{Y}_i$

Figure 1. Flow chart of K-nearest neighbor geographically weighted regression. (**a**) Optimal bandwidth selection reference k-means. (**b**) KD Tree construction and search. (**c**) Restructured $\widetilde{W}_i$, $\widetilde{X}_i$, $\widetilde{Y}_i$. (**d**) KNN-GWR in calibration.

2.2.1. Optimal Bandwidth Selection Reference K-Means

We counted the optimal bandwidth in some studies, which was usually within a smaller range [19,48,49]; the specific range of values depended on the research field, specific dataset, the researcher's experience and judgment, and in general was usually ≤ 200. However, GWR usually searches for the optimal bandwidth on a global scale, which is less necessary in local regression. Murakami pointed out the use of rank reduction and pre-compression to eliminate the effect of data size in the regression of large datasets [49]. Geographically adjacent observation points were considered to have highly similar characteristics; therefore, in the KNN-GWR, they should be grouped into the same category or group as much as possible when performing geographic analysis. In this study, K-means clustering was utilized to analyze the latitude and longitude coordinates of geographic data, thereby facilitating a more comprehensive examination of the distribution patterns and spatial relationships of the data. For K-means clustering of big datasets, a set of values is defined and referred to as KC values, where the optimal number of clusters will be generated from this set of KC values, with $KC \in \{k_{min}, k_{sec}, k_{mean}, k_{four}, k_{max}\}$. Using the heuristic approach, the square root value of half the data size is k_{mean} [50–52], where k_{mean} is obtained by the following formula:

$$k_{mean} = \sqrt{\frac{n}{2}} \tag{9}$$

where n denotes the number of observation points.k_{sec} and k_{four} are the results of moving k_{mean} up or down by one number. k_{min} and k_{max} are the results of moving k_{mean} up or down by two different numbers.

The sum of squared errors was used to calculate the clustering error of the sample to select the optimal KC value [53,54].

The formula is as follows.

$$SSE = \sum_{g=1}^{k} \sum_{s \epsilon c_g} |s - m_i| \tag{10}$$

where C_g is the g-th cluster, s is a sample point in C_g, and m_i is the center of C_g.

The pseudocode of the Determine Optimal Bandwidth Range algorithm is given in Algorithm 1. The clusters obtained by the K-means clustering method are labeled as $A_1, A_2, \ldots, A_{k_{optimal}}$, respectively. The number of observation points in each cluster was counted using the *Num()* function to represent the cluster size. For example, *Num*(A_1) represents the number of observation points in Cluster A_1. The *Num()* function helps find the largest and smallest clusters, denoted as A_{max} and A_{min}. A_{max} and A_{min} represent the upper and lower limits of the spatial distribution of all observation points based on spatial location features. The cross-validation (CV) method was chosen to determine the optimal bandwidth. The formula is as follows.

$$CV = \sum_{i=1}^{n} \left[y_i - \hat{y}_{\neq i}(bw)\right]^2, \quad bw \in \left[Num(A_{min}),\ Num(A_{max})\right] \tag{11}$$

Algorithm 1: Optimal Bandwidth Selection Reference K-means

```
1. According to the number of data n, determine KC ∈ $\{k_{min}, k_{sec}, k_{mean}, k_{four}, k_{max}\}$
2. Perform K-means clustering analysis based on KC
3. Find the optimal KC value $k_{optimal}$ depending on the $SSE$
4. Obtain clustering results $A_1, A_2, \ldots, A_{k_{optimal}}$
5. Defined $(Num(A_{min}), Num(A_{max})) = Num(A_1, A_2, \ldots, A_{k_{optimal}})$
6. Function $Num(A_1, A_2, \ldots, A_{k_{optimal}})$
7. Count the number of data points in each cluster of $A_1, A_2, \ldots, A_{k_{optimal}}$, denoted as
   $Num(A_1), Num(A_2), \cdots, Num(A_{k_{optimal}})$
8. Compare the size of $Num(A_1), Num(A_2), \cdots, Num(A_{k_{optimal}})$ and return the maximum
   value $(Num(A_{max}))$ and minimum value $(Num(A_{min}))$
9. Output $Num(A_{min})$ and $Num(A_{max})$ to provide a reference for the optimal bandwidth
```

2.2.2. KD Tree Construction and Search

Using the bi-square kernel function in GWR, the surrounding observations need to be found to participate in the calculation during the local regression of regression point *i*. For two-dimensional geographic data, traditional linear search methods involve brute-force computation of the distances between all pairs of points in the dataset. Assuming a sample size of n, the time complexity of this method is $O(n^2)$. Since there is no adjacency between point *i* and point (5, 6, 7, 8, 9) in the limit range of R, as shown in Figure 2. Therefore, the distance calculation between point *i* and point (5, 6, 7, 8, 9) is not necessary. Further, by reducing the number of visited nodes and the corresponding distance calculation, the speed of finding surrounding points can be improved. In this study, a KD tree was created for all regression points based on spatial coordinates in order to effectively filter invalid calculations when searching for observations around regression points.

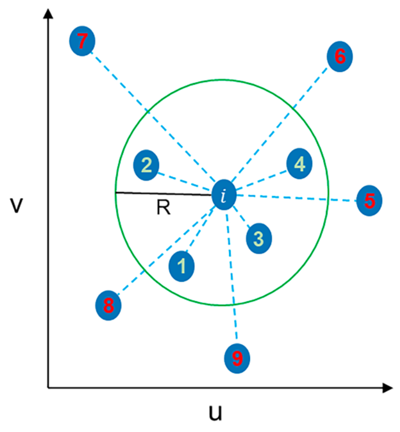


Figure 2. An example of finding observations within a certain range around the regression point. 1, 2, 3, and 4 are points within the bandwidth distance from point *i*, and 5, 6, 7, 8, and 9 are points outside the bandwidth distance from point *i*.

A K-dimensional tree (KD tree) built based on distance is an extended version of a binary tree for speeding up the dataset search. If the points include the (u, v) dimension, they are first sorted according to the value of a dimension, such as u. Then, the middle point is selected as the split (parent) node, and all points are divided into two parts according to the coordinates on the u axis. The u value of the left subspace is smaller than that of the parent node, and the u value of the right subspace must be greater than or equal to the u value of the parent node. This process continues repeatedly until both subregions have no sample points (the terminating node is a leaf node), and finally, the region division of the KD tree is formed. Using this approach, sample points are saved on the appropriate nodes. In this case, the time complexity for performing a perimeter point lookup is $O(n\log(n))$. After the tree structure is established, this algorithm starts from the root node and recursively searches the left and right child nodes to find all points within the radial range of the regression point. The lookup of the observation points around the regression point was completed quickly after the KD tree was constructed, which accelerated the process of reconstructing the matrix in the KNN-GWR method.

2.2.3. Geographically Weighted Regression with K-Nearest Neighbors and Model Evaluation

Press stated that the desirable characteristics of an algorithm are to be fast, computationally inexpensive, and to use as little memory as possible with high computational accuracy [55]. When the bi-square kernel function is selected as the kernel function for GWR, only the observation points within the bandwidth participate in the regression calculation. Here, the zero rows and columns of the weight matrix were removed to minimize the computational cost. In the local regression calculation of regression point *i*, the construction of the KD tree speeds up the search for the points surrounding regression point *i*.

Considering the above, the KNN-GWR algorithm was set up in this study. The expression of the KNN-GWR algorithm is expressed as follows:

$$y_i = \overset{\sim}{\beta}_0(u_i, v_i) + \sum_{j=1}^{p} \overset{\sim}{\beta}_j(u_i, v_i)x_{ij} + \varepsilon_i \tag{12}$$

The formula for calculating $\overset{\sim}{\beta}_i$ is shown below.

$$\overset{\sim}{\beta}_i = \left(\tilde{X}_i^T \tilde{W}_i \tilde{X}_i\right)^{-1} \tilde{X}_i^T \tilde{W}_i \tilde{Y}_i \tag{13}$$

The corresponding $\tilde{X}_i$ and $\tilde{Y}_i$ matrices are determined based on the bandwidth of the regression point and KD tree data structure. The matrices of $\tilde{X}_i$ and $\tilde{Y}_i$ can be calculated by

$$\tilde{X}_i = \begin{bmatrix} 1 & x_{d_1 1} & \cdots & x_{d_1 p} \\ 1 & x_{d_2 1} & \cdots & x_{d_2 p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{d_K 1} & \cdots & x_{d_K p} \end{bmatrix} \tag{14}$$

And

$$\tilde{Y}_i = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix} \tag{15}$$

where $\tilde{X}_i$ is composed of independent variables from the observation points within the bandwidth around the regression point i ($\tilde{X}_i$ is a $K \times (p+1)$ matrix). $\tilde{Y}_i$ is composed of the dependent variable of the observation points within the bandwidth around the regression point i ($\tilde{Y}_i$ is a $K \times 1$ matrix). K is a constant equal to the number of observation points around the regression point. $[1, 2, \ldots, K]$ is the point number in the bandwidth range after sorting by distance. $\tilde{W}_i$ is a space weight matrix and is expressed as:

$$\tilde{W}_i = \begin{bmatrix} w_{i1} & 0 & \cdots & 0 \\ 0 & w_{i2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{iK} \end{bmatrix} \tag{16}$$

The equation for the weight matrix w_{ij} is expressed as

$$w_{ij} = \left[1 - \left(\frac{d_{ij}}{d_{iK}}\right)^2\right]^2 \tag{17}$$

where d_{ij} represents the spatial distance between points i and j and d_{iK} represents the spatial distance between points i and K.

The hat matrix $\tilde{S}$ is the projection matrix from the observed y to the fitted $\hat{y}$, where each row $\tilde{S}_i$ of the hat matrix is:

$$\tilde{S}_i = X_i \left(\tilde{X}_i^T \tilde{W}_i \tilde{X}_i\right)^{-1} \tilde{X}_i^T \tilde{W}_i \tag{18}$$

where

$$\hat{y}_i(bw) = \widetilde{S}_i \widetilde{Y}_i \tag{19}$$

where $\hat{y}_i(bw)$ is the fitted value of y_i using a bandwidth. The residual vector is then

$$e = y_i - \hat{y}_i(bw) \tag{20}$$

and the residual sum of squares is

$$RSS = \sum_{i=1}^{n} e^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{21}$$

The estimation accuracy of the GWR model was evaluated by comparing the fitted dependent variable values with the dependent variable values of the sample data, as R^2.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2} \tag{22}$$

The calculated runtime ratio of the KNN-GWR algorithm to other GWR packages is described by the runtime increase rate (*RIR*).

$$RIR = \frac{Runtime_{others}}{Runtime_{KNN-GWR}} \tag{23}$$

The pseudo-code of KNN-GWR algorithm is given in Algorithm 2.

Algorithm 2: KNN-GWR Algorithm

```
I    Optimizing bandwidth search by minimizing CV scores
1.   Given the initial data $bw_1$, $X$, $Y$, $U$ and $V$
2.   Start the loop $bw = bw_1, bw_2, \cdots, bw_o, \cdots, bw_t$:
3.       Build KD tree spatial index
4.       Starting loop with $i = 1, 2, \cdots, n$:
5.           Take the data from the result of 3 steps:
6.           Reconfiguration $\widetilde{W}_i$, $\widetilde{X}_i$, and $\widetilde{Y}_i$
7.           Calculate $\widetilde{\beta}_i$ , $\hat{y}_i$
8.       End of loop
9.       Calculate CV Scores
10.  End of loop
II   Optimal bandwidth selection based on minimum CV criterion
11.  Start the loop $i = 1, 2, \cdots, n$:
12.      Reconfiguration $\widetilde{W}_i$, $\widetilde{X}_i$, and $\widetilde{Y}_i$
13.      Calculate $\widetilde{\beta}_i$ , $\hat{y}_i$
14.  End of loop
15.  Return $\widetilde{\beta}$, $\hat{y}$
16.  Spatial analysis and model diagnosis using $\widetilde{\beta}$, $\hat{y}$
```

2.2.4. Computational Complexity of GWR and KNN-GWR in Calibration

Time Complexity

In the GWR regression calculation, the time complexity of $X^T W_i X$ matrix is $O\left((p+1)^2 n\right)$ and the big O is an asymptotic notation used to describe the upper bound of an algorithm's efficiency. Calculating its inverse, $\left(X^T W_i X\right)^{-1}$, requires $O\left((p+1)^3\right)$ [19,30,56] (p is the number of independent variables, $(p+1) \leq 10$ in common). Therefore, the time complexity of computing β_i is $O\left((p+1)^2 n\right)$ [19,31]. The calculation of β_i is repeated at each regression point location n times; therefore, the total time complexity of the GWR calibration for a known bandwidth is $O\left((p+1)^2 n^2\right)$. If the golden partition is used to find the optimal bandwidth, the time complexity is close to $O(logn)$, and the total time complexity of the bandwidth selection and model calibration is $O\left((p+1)^2 n^2 logn\right)$ [19]. The time complexity of KNN-GWR is explained in the same manner as above. In the KNN-GWR model, $\tilde{X}_i$ is a $K \times (p+1)$ matrix and $\tilde{W}_i$ is a $K \times K$ matrix. The time complexity of the $\tilde{X}_i^T \tilde{W}_i$ matrix is $O((p+1)K)$, and the time complexity of the $\tilde{X}_i^T \tilde{W}_i \tilde{X}_i$ matrix is $O\left((p+1)^2 K\right)$. Calculating its inverse, $\left(\tilde{X}_i^T \tilde{W}_i \tilde{X}_i\right)^{-1}$, takes $O\left((p+1)^3\right)$, which can be neglected. $\overset{\sim}{\beta}_i$ is computed at each regression point n times; therefore, the total time complexity of the KNN-GWR calibration for a known bandwidth is $O\left((p+1)^2 n\right)$. If the golden partition is used to find the optimal bandwidth, the time complexity is close to $O(\log(Num(A_{max}) - Num(A_{min})))$, and the total time complexity of the bandwidth selection and model calibration is $O\left((p+1)^2 Knlog(Num(A_{max}) - Num(A_{min}))\right)$. Therefore, in theory, the time complexity of KNN-GWR is much lower than that of GWR. The time complexity of each algorithm is listed in Table 1.

Table 1. Comparison of time complexity among KNN-GWR, FastGWR, MGWR (PySAL), GWmodel, and Spgwr.

Name of Algorithm	Time Complexity
KNN-GWR	$O\left((p+1)^2 \text{Knlog}(Num(A\_max) - Num(A - min))\right)$
FastGWR	$O\left((p+1)^2 n^2 logn\right)$ [19]
MGWR(PySAL), GWmodel and Spgwr	$O\left((p+1)^2 n^2 logn\right)$ [19,31]

Note: K is the data of the observation points around the regression point in the bandwidth, n is the number of observation points, and p is the number of independent variables.

Memory Complexity

The KNN-GWR algorithm improves the matrix and dataset storage model of the GWR operation by adopting distance sorting and a data structure. The weight matrix W_i of the GWR method is stored as an $n \times n$ diagonal matrix, which requires n^2 storage spaces during the operation, and the memory complexity is $O(n^2)$. In the process of GWR regression calculation, $X^T W_i X$, $(X^T W_i X)^{-1}$, and $(X^T W_i X)^{-1} X^T W_i$ inherit the memory complexity of the weight matrix W_i. The KNN-GWR algorithm avoids this situation during the calculation. The weight matrix $\tilde{W}_i$ is calculated by selecting only the points within the GWR bandwidth ($\tilde{W}_i$ is a $K \times K$ matrix), and the $\tilde{X}_i$ and $\tilde{Y}_i$ matrices are reconstructed accordingly ($\tilde{X}_i$ is a $K \times (p+1)$ matrix; $\tilde{Y}_i$ is a $K \times 1$ matrix). The memory complexity of the KNN-GWR algorithm is affected by K, and the memory complexity is $O(K^2)$. (typically, $K << n$). The memory complexity of each algorithm is presented in Table 2.

Table 2. Comparison of memory complexity between KNN-GWR and other GWR packages.

Number of Data Points	KNN-GWR $O(K^2)$	FastGWR $O(n(p+1))$	MGWR(PySAL), GWmodel, and Spgwr $O(n^2)$
1000	33.8 KB	19.5 KB	3.8 MB
10,000	33.8 KB	195.3 KB	380 MB
100,000	33.8 KB	1.9 MB	37.25 GB
1,000,000	33.8 KB	19.1 MB	3.8 TB
10,000,000	33.8 KB	190.7 MB	364 TB

Note: The data are 32–bit floating points, where K is numerically equal to 93, n is the number of observation points, and p is the number of independent variables.

3. Experiment

3.1. Data Source

To explore the actual performance of KNN-GWR, two datasets were used in the experiment—the simulated dataset [31] and the house price dataset in China from the Anjuke.com website (accessed on 25 March 2022). A simulated dataset was used to verify the viability of the model. The Chinese house price dataset was used to test the scalability of the KNN-GWR model.

3.1.1. Simulated Dataset

The test data were distributed in a square area with a side length of l units and the data sample points were evenly distributed in this area. The sample size of each line was m, the total number of samples was $n = m \times m$, and the distance between adjacent sample points was $\Delta l = l/(m-1)$. Considering the lower left corner of the square area as the origin of the coordinate system, the calculation formula for the sample point position is as follows:

$$(u_i, v_i) = (\Delta l \times mod(i-1, m), \Delta l \times int(i-1, m)) \tag{24}$$

where $mod\ (a,\ b)$ and $int\ (a,\ b)$ are the remainder and integer part of a divided by b [57]. The experimental data samples were generated using a predefined GWR model with the following equation:

$$y_i = \beta_0(u_i, v_i) + \beta_1(u_i, v_i)x_{i1} + \beta_2(u_i, v_i)x_{i2} + \beta_3(u_i, v_i)x_{i3} + \beta_4(u_i, v_i)x_{i4} + \varepsilon_i \tag{25}$$

To standardize the regression coefficient β, all regression coefficients β were limited to the $(0,\ \beta_{max})$ interval (β_{max} was a fixed constant) [31]. The regression coefficient beta of the model follows the following five functions.

$$\begin{cases} \beta_0(u_i, v_i) = \frac{2\beta_{max}}{l^2}\left(\frac{l^2}{2} - (l-u_i)^2 - (l-v_i)^2\right) \\ \beta_1(u_i, v_i) = \frac{\beta_{max}}{2}\left(\left(sin\frac{u_i\pi}{l}\right)^2 + \left(sin\frac{v_i\pi}{l}\right)^2\right) \\ \beta_2(u_i, v_i) = \frac{\beta_{max}}{2}\left(2 - \left(\left(tan\left(\frac{u_i\pi}{2l} - \frac{\pi}{4}\right)\right)^2 + \left(tan\left(\frac{v_i\pi}{2l} - \frac{\pi}{4}\right)\right)^2\right)\right) \\ \beta_3(u_i, v_i) = \beta_{max}e^{-\frac{1}{2l}\left(\left(\frac{l}{2}-u_i\right)^2 + \left(\frac{l}{2}-v_i\right)^2\right)} \\ \beta_4(u_i, v_i) = \frac{16\beta_{max}}{l^4}\left(\frac{l^2}{4} - \left(\frac{l}{2} - u_i\right)^2\right)\left(\frac{l^2}{4} - \left(\frac{l}{2} - v_i\right)^2\right) \end{cases} \tag{26}$$

This paper prepared 8 sets of simulated datasets with different numbers: 1000, 5000, 10,000, 15,000, 20,000, 50,000, 100,000, and 1,000,000.

3.1.2. House Price Dataset in China

The study area ranged from 108°21′ to 122°42′ E and 23°33′ to 42°40′ N. Fourteen provinces or municipalities were mainly involved (Anhui, Jiangxi, Henan, Hubei, Hunan, Shanxi, Hebei, Jiangsu, Zhejiang, Fujian, Beijing, Tianjin, and Shanghai). The total area was approximately 17,454,000 square kilometers comprising two world-class urban agglomerations, Beijing–Tianjin–Hebei and Yangtze River Delta, as shown in Figure 3.

The house price dataset used in this study was collected from www.anjuke.com (accessed on 25 March 2022). The experimental dataset was selected from residential land price statistics with geographic location, including the number of bedrooms (NBeds), bathrooms (NBaths), floor area (Area), and age of the house (Age). Detailed information on the experimental dataset is shown in Figure 4. In this study, 123,691 research data were sequentially divided into 7 different datasets by random sampling: 1000, 5000, 10,000, 15,000, 20,000, 50,000, and 100,000. Seven sets of datasets were applied to MGWR (PySAL), GWmodel, Spgwr, and KNN-GWR, which satisfy the following expression.

$$y_i = \beta_0 + \beta_1 Area_i + \beta_2 NBaths_i + \beta_3 NBeds_i + \beta_4 Age_i + \varepsilon_i \tag{27}$$

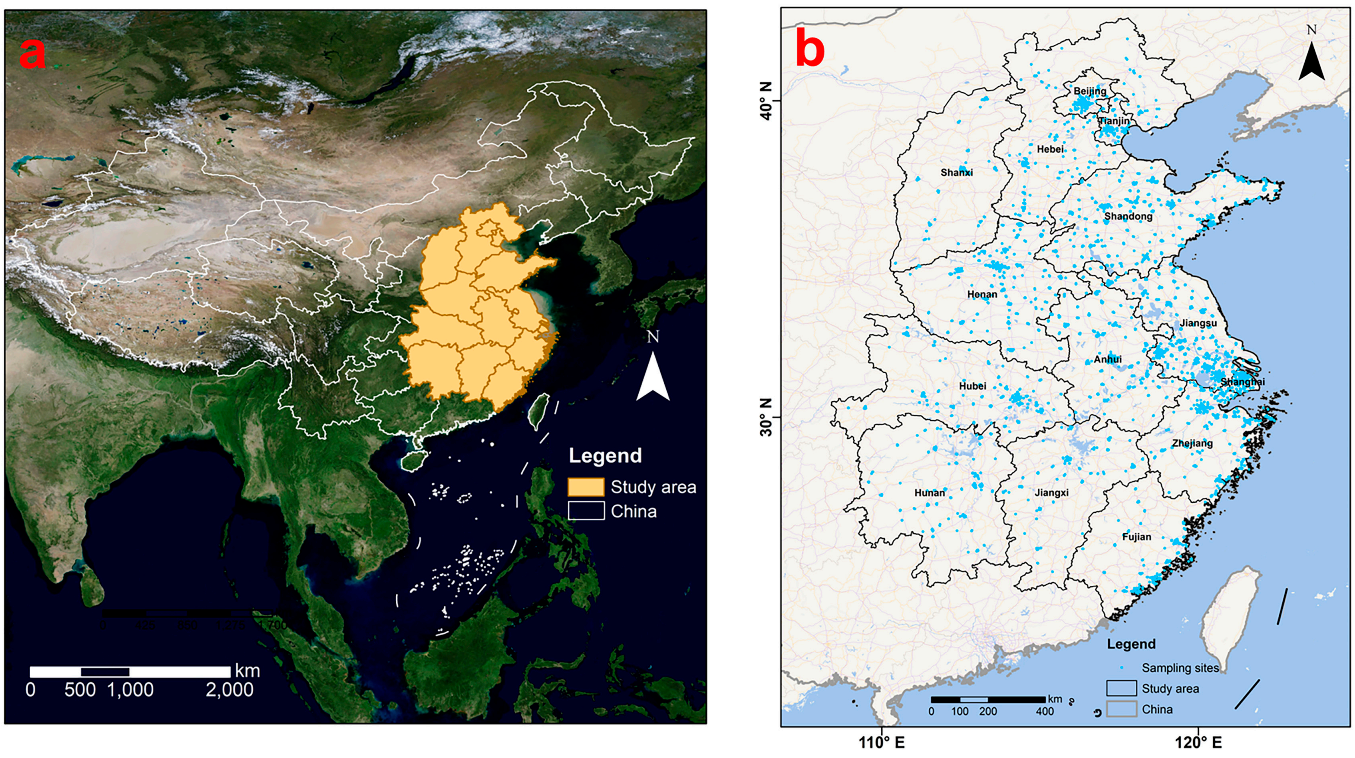


Figure 3. Study area. (**a**) Geographical location of each province of the study area in China. (**b**) Distribution of house price dataset in the study area.

3.2. Testing Specifications and Environment

The differences in runtime between the KNN-GWR and other GWR packages (MGWR (PySAL), GWmodel, and Spgwr) were compared. Details of the equipment used in the experiments are listed in Table 3.

Table 3. Device parameters for running study GWR models.

	KNN-GWR, MGWR(PySAL), GWmodel, Spgwr
CPU	Intel(R) Core(TM) i7-1065G7 CPU @ 1.30 GHz, 8 cores
Memory	16,384 MB RAM

3.3. Results

3.3.1. Case One: Simulated Dataset

In Figure 5, the coefficients $\beta(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4)$ and the respective dependent variable values Y for the simulated data are presented, along with the coefficients $\widetilde{\beta}\left(\widetilde{\beta}_0, \widetilde{\beta}_1, \widetilde{\beta}_2, \widetilde{\beta}_3, \widetilde{\beta}_4\right)$ and predicted values of the dependent variable *Predicted Y* obtained by applying the KNN-GWR method. As shown in Figure 5, the five coefficients β selected by the model are closely related to the locations of the sample points and exhibit spatial non-stationarity. The spatial distribution of the coefficients $\widetilde{\beta}\left(\widetilde{\beta}_0, \widetilde{\beta}_1, \widetilde{\beta}_2, \widetilde{\beta}_3, \widetilde{\beta}_4\right)$ obtained by the KNN-GWR method is quite similar to the distribution of the simulated data, indicating that it addresses the spatial non-stationarity.

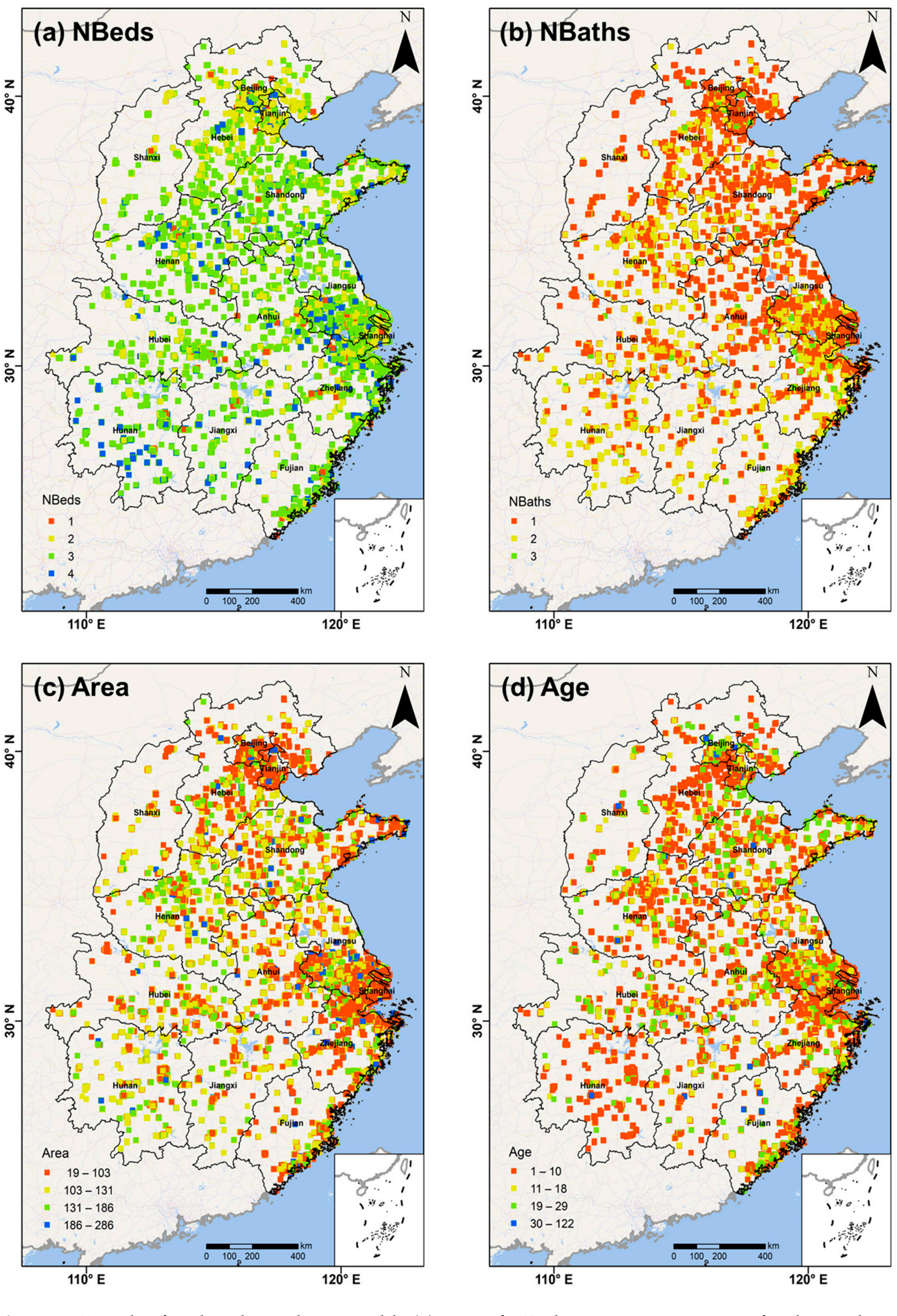


Figure 4. Details of each independent variable (**a**) Map of NBeds parameter estimates for the predictor variables. (**b**) Map of NBaths parameter estimation for predictor variables. (**c**) Map of Area parameter estimates for the predictor variables. (**d**) Map of Age parameter estimation for the predictor variables.

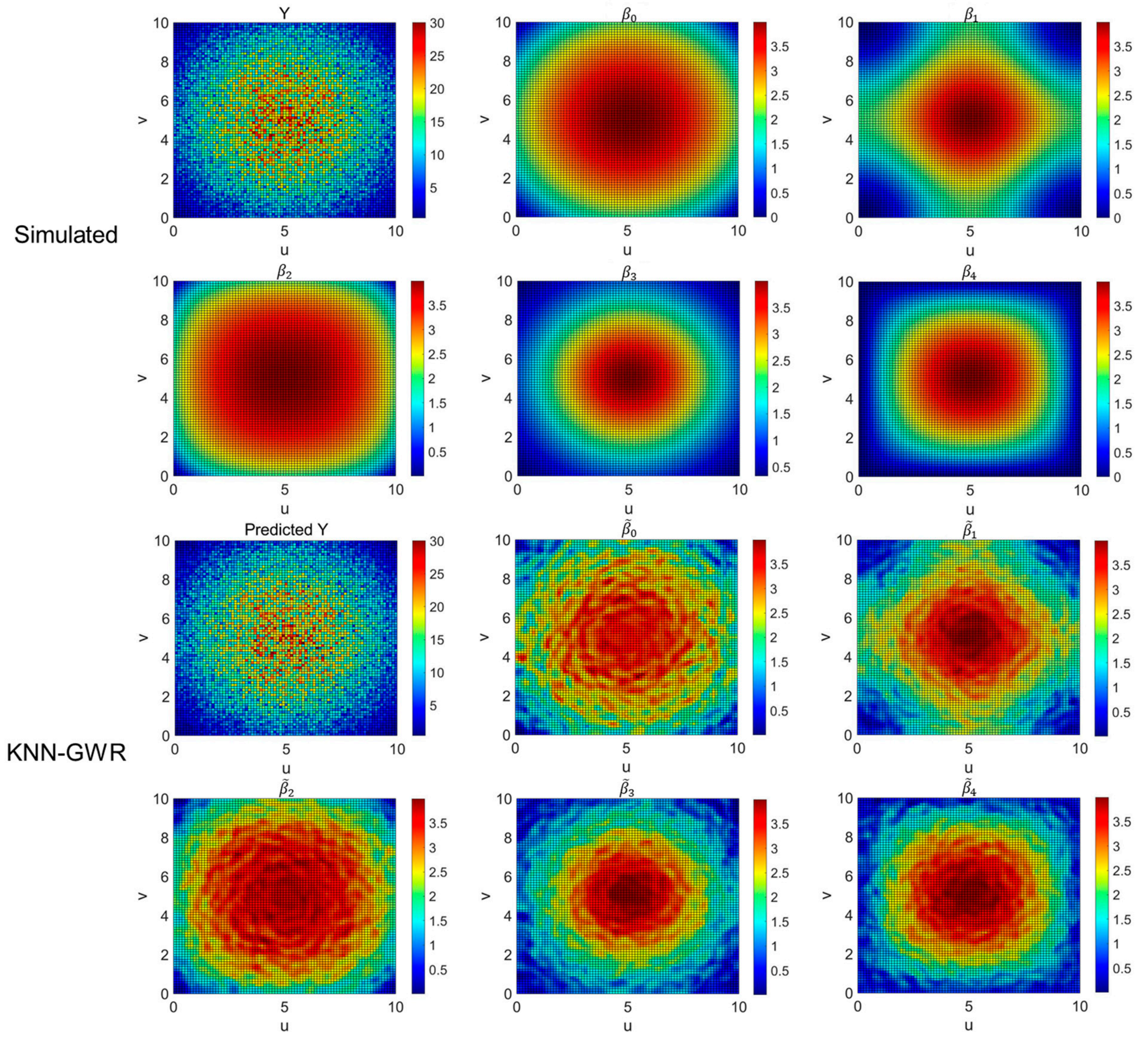


Figure 5. Estimated regression coefficients β of KNN-GWR method on simulated dataset.

Due to the transformation of a large matrix into a small matrix, the regression efficiency of KNN-GWR is significantly improved compared with that of GWR. As shown in Figure 6 and Table 4, the average running times of KNN-GWR at 1000, 5000, and 50,000 points are 0.34, 3.59, and 31.77 s, respectively. However, the average running time of MGWR(PySAL) increased from 3.71 s at 1000 points to 178.52 s at 10,000 points, and then reached 6327.91 s at 50,000 points. Figure 7 showed the comparison of KNN-GWR with other GWR packages in terms of runtime increase ratio. The results indicated that for 10,000 observations, KNN-GWR was approximately 49.7 times faster than MGWR(PySAL), approximately 58.9 times faster than GWmodel, and approximately 3224.2 times faster than Spgwr. Spgwr cannot obtain the results because the data are larger than 10,000 observations. In addition, when the data increased to 50,000 observations, KNN-GWR was approximately 199.2 times faster than MGWR(PySAL) and approximately 260.7 times faster than GWmodel. KNN-GWR implements calibration operations at amounts of data greater than 50,000, whereas in other GWR packages memory bottlenecks are caused by storing redundant computations.

Table 4. Information on the runtime and R^2 of four software packages (KNN-GWR, MGWR(PySAL), GWmodel, and Spgwr) in simulated dataset experiments.

Number of Data Points	KNN-GWR		MGWR(PySAL)		GWmodel		Spgwr	
	Runtimes (s)	R^2	Runtimes (s)	R^2	Runtimes (s)	R^2	Runtimes (s)	R^2
1000	0.34	0.995	3.71	0.995	1.33	0.994	21.65	0.994
5000	3.11	0.996	43.51	0.995	41.51	0.997	1514.71	0.997
10,000	3.59	0.996	178.52	0.994	211.41	0.998	11,574.78	0.997
15,000	6.56	0.993	342.51	0.996	598.91	0.992	n/a	n/a
20,000	8.52	0.994	758.81	0.996	1234.06	0.993	n/a	n/a
50,000	31.77	0.997	6327.91	0.996	8283.33	0.995	n/a	n/a
100,000	83.06	0.998	n/a	n/a	n/a	n/a	n/a	n/a
10,000,000	865.13	0.999	n/a	n/a	n/a	n/a	n/a	n/a

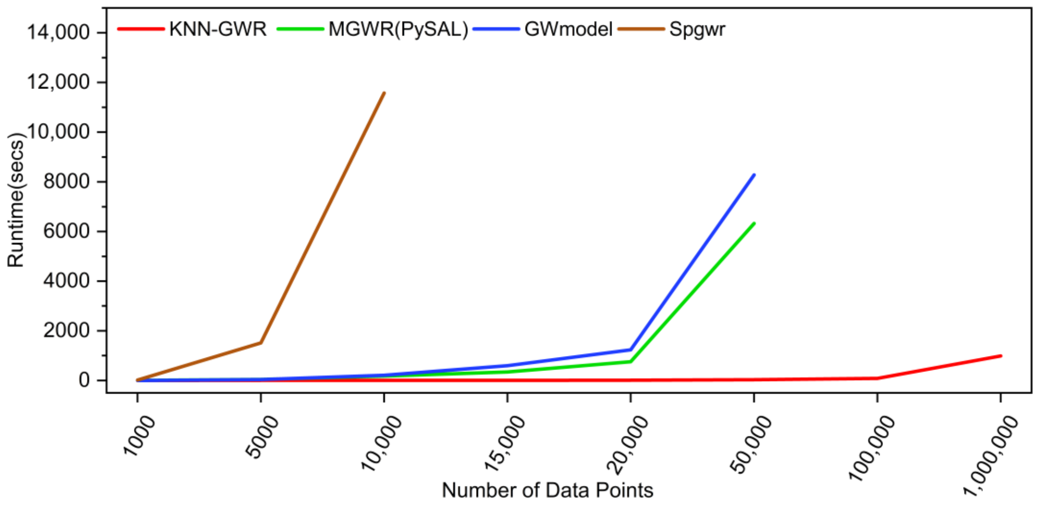


Figure 6. Comparison of the computational speeds of four software packages(KNN-GWR, MGWR (PySAL), GWmodel, and Spgwr) in simulated dataset experiments.

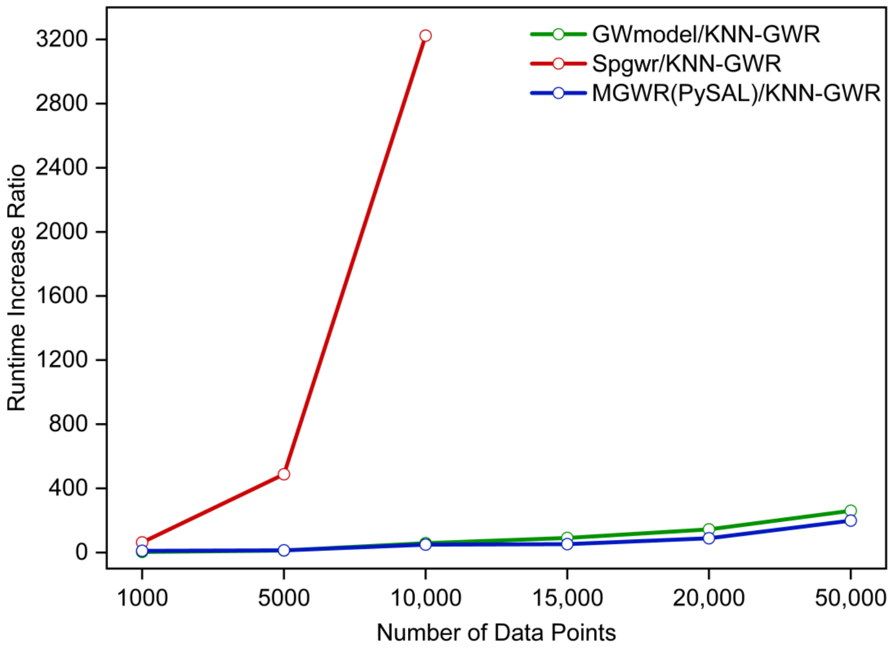


Figure 7. Comparison of KNN-GWR with other GWR packages in terms of runtime increase ratio (tested using simulated dataset).

The advantage of geographically weighted regression methods for analyzing geographic data was retained in the KNN-GWR method, and the storage of the full sample matrix by other GWR software models was changed in the KNN-GWR method to store only the matrix consisting of the observations involved in the regression.

3.3.2. Case Two: House Price Dataset in China

The local model can discover the rich information in geographic data and we performed KNN-GWR modeling on the house price data collected from www.anjuke.com. KNN-GWR generated local parameter estimates that reflect information on the spatial heterogeneity affecting house prices. The performance of KNN-GWR and its spatial non-smoothness are explored visually through local coefficient estimation of mapped variables. Figure 8a–d show the spatial patterns of the KNN-GWR model estimated coefficients.

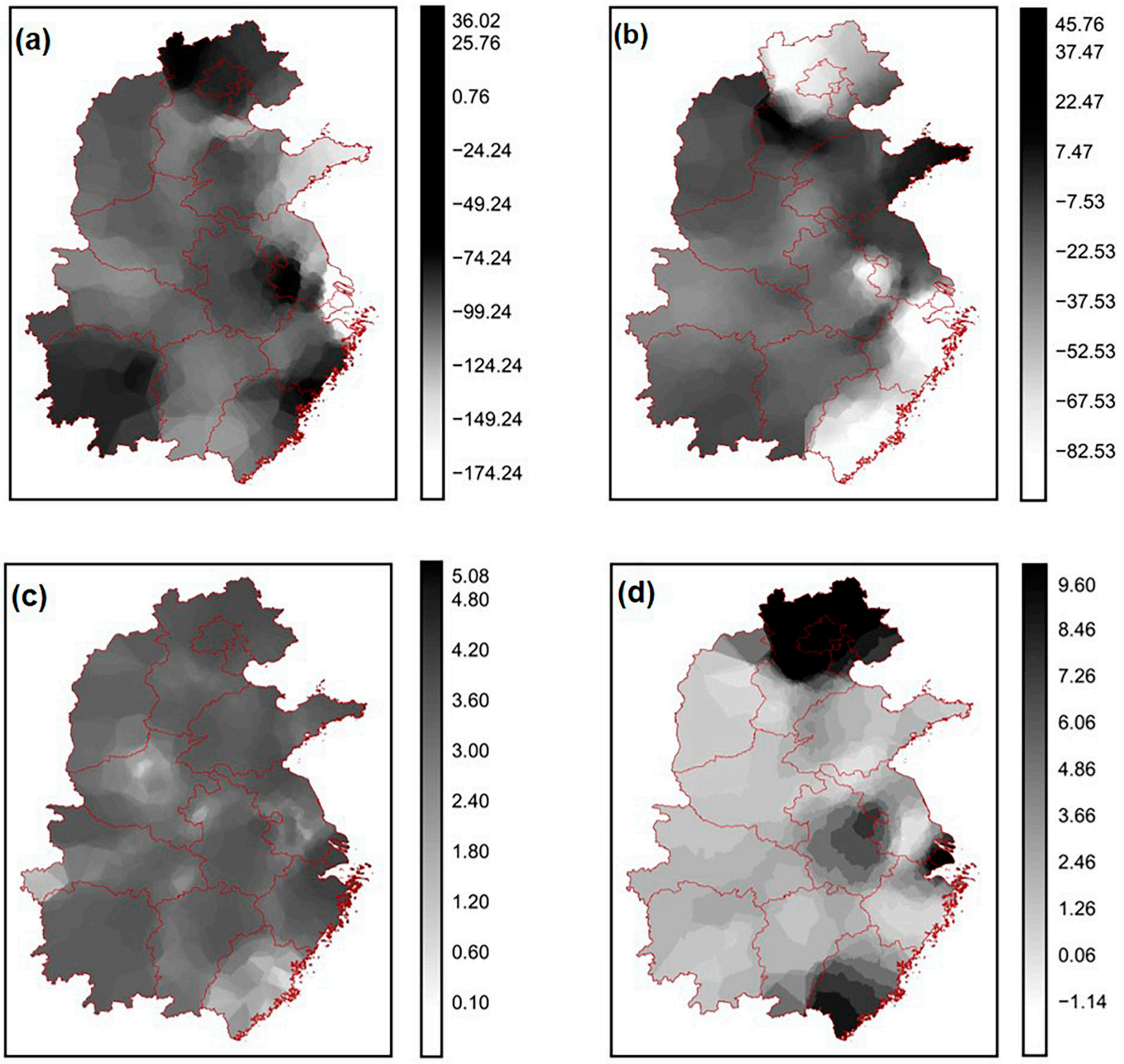


Figure 8. Space mapping for: (**a**) number of bedrooms in the house, (**b**) number of bathrooms, (**c**) area of the house, and (**d**) age of the house, by KNN-GWR modeling.

The results indicate that Area is positively correlated with residential land price, and the larger the residential area, the higher the residential land price (Figure 8c). And it seems reasonable that the effects of age are greater in Beijing–Tianjin–Hebei and the Yangtze River Delta. As these cities are better developed, housing construction has slowed down in recent years, resulting in higher house prices in some areas. In addition, the number of bathrooms is negatively correlated with house prices in some coastal cities, possibly because higher housing prices have dampened the greater demand for the number of bathrooms.

Compared with other GWR packages, the KNN-GWR method demonstrated high efficiency in processing housing price datasets in the study area. As shown in Figure 9 and Table 5, as the number of data increases, the running time of KNN-GWR remains relatively stable, while the running time of all the other GWR packages increases rapidly. For instance, with 50,000 observations, the KNN-GWR method requires 51.21 s, while MGWR (PySAl) and GWmodel require 4116.12 and 7445.82 s, respectively. This indicates that KNN-GWR is capable of processing large-scale geographic datasets relatively quickly, while other GWR packages struggled to achieve the same level of efficiency. In addition, as shown in Figure 10, KNN-GWR has an efficiency improvement of a thousand times compared to other GWR software packages. This indicates that KNN-GWR overcomes the memory limitations and process larger datasets more efficiently than other GWR packages. Overall, these findings demonstrate the usefulness of the KNN-GWR method in analyzing large-scale geographic data and its potential to outperform in computational efficiency and memory optimization compared to existing GWR algorithms.

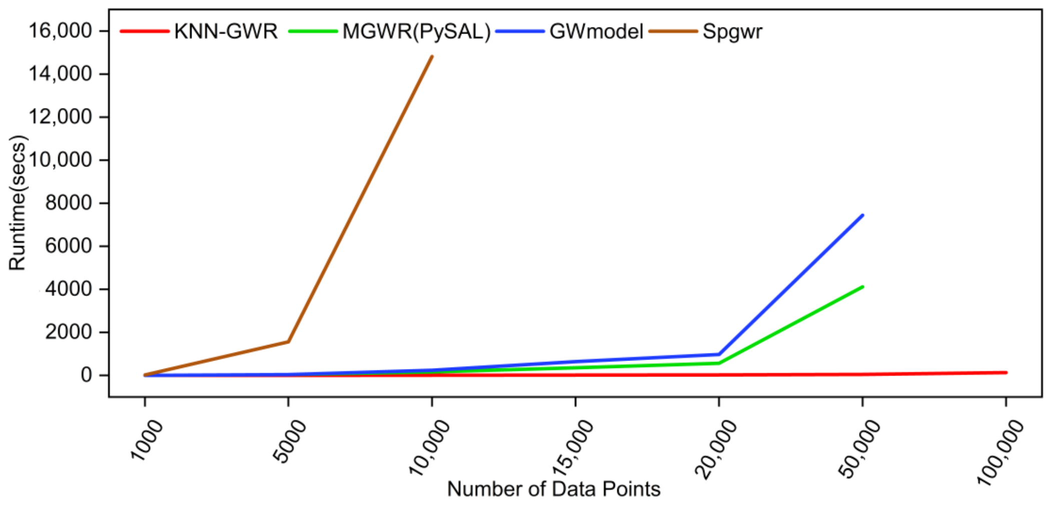


Figure 9. Comparison of the computational speeds of four software packages (KNN-GWR, MGWR(PySAL), GWmodel, and Spgwr) in the house price dataset in China.

Table 5. Information on the runtime of four software packages (KNN-GWR, MGWR(PySAL), GW-model, and Spgwr) in the house price dataset in China.

Number of Data Points	KNN-GWR	MGWR(PySAL)	GWmodel	Spgwr
1000	0.73	3.43	1.45	24.70
5000	4.51	44.92	45.26	1564.89
10,000	8.73	164.20	239.99	14,812.23
15,000	18.32	359.21	646.92	n/a
20,000	25.21	563.13	978.95	n/a
50,000	51.21	4116.12	7445.82	n/a
100,000	138.33	n/a	n/a	n/a

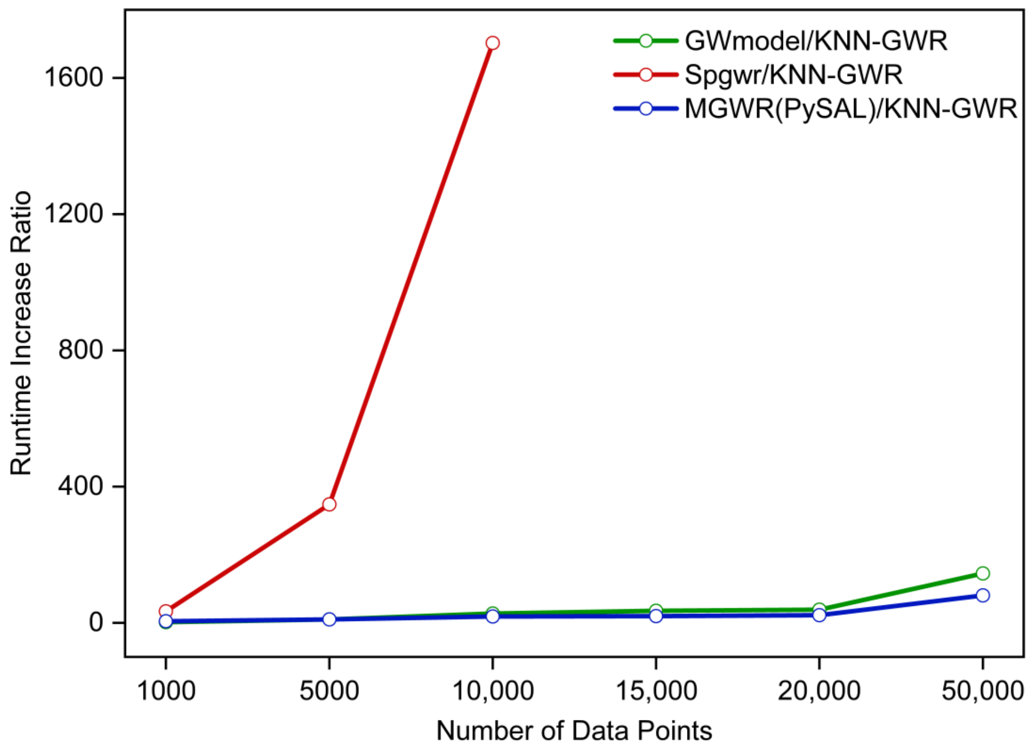


Figure 10. Comparison of KNN-GWR with other GWR packages in terms of runtime increase ratio (tested using house price dataset in China).

4. Discussion

The KNN-GWR proposed in this paper is an optimized version of GWR. Specifically, it optimized the computational efficiency and memory of GWR by strategies such as constructing KD tree, referencing geographical clustering and reconstructing regression matrices. In essence, there is no difference between KNN-GWR and GWR in exploring spatial non-stationarity. However, KNN-GWR is better suited for processing geographical big data applications and address the changes associated with it.

Therefore, in our experimental results, KNN-GWR has a good fitting degree, just like GWR. Compared with GWR, the advantages of KNN-GWR in computational optimization are as follows. First, the KD tree is used to establish a spatial data index structure that can quickly identification the observation points around the regression point. Second, KNN-GWR reconstructed the weight matrix $W_i(n \times n)$, independent variable matrix X $(n \times (p+1))$, and dependent variable matrix $Y(n \times 1)$ in the local regression calculation for each regression point (the reconstructed matrices are $\widetilde{W}_i\ (K \times K)$; $\widetilde{X}_i\ (K \times (p+1))$; $\widetilde{Y}_i\ (K \times 1)$). In this process, observation points that are not within the bandwidth in the local regression of each regression point are directly removed according to the spatial data index structure established by the KD tree. Memory complexity is reduced from $O\left(n^2\right)$ to $O\left(K^2\right)$, where n is the number of observation points. Third, KNN-GWR references the spatial clustering relationship obtained by K-means, which in turn helps to find the optimal bandwidth. To optimize GWR performance, increasing the hardware configuration is one approach; however, it is often necessary to optimize performance of GWR without relying on hardware enhancements.

To demonstrate the practicability of KNN-GWR, simulated data and the Chinese house price dataset were used for the experiments. The results show that GWR cannot handle regression tasks of large amount of data; in contrast, KNN-GWR has great potential to handle a large amount of geographic data, which can largely alleviate the dilemma of GWR in terms of data size. In this paper, we collected house price data from 14 provinces and municipalities and applied it to verify the practical significance of the model in a real geographical environment. When analyzing the house price data of 14 provinces and municipalities, it was found that the factors affecting the house price in coastal cities, more developed areas and slow-developing areas differed. For example, the Age variable has strong effects on house prices in Beijing and Shanghai, but relatively weak effects in other provinces. In terms of runtime, taking simulated data as an example, for 50,000 observations, KNN-GWR is several hundred times faster than MGWR(PySAL), and the GWmodel. In terms of the memory required for calculation, the memory requirement of KNN-GWR is affected by the number K of the nearest neighbors. Generally speaking, the required memory is very small. The memory assignment of KNN-GWR $O(K^2)$ was compared with that of FastGWR $O(nk)$, MGWR(PySAL) $O(n^2)$, GWmodel $O(n^2)$, and Spgwr $O(n^2)$ (the number of independent variables of the database is four, and the adjusted bandwidth is 93). For 50,000 observations, KNN-GWR required 33.8 KB of memory, FastGWR required 976.6 KB, and MGWR(PySAL), GWmodel, and Spgwr required 9.31 GB of memory. KNN-GWR has the same ability as GWR to explore spatial non-stationarity. However, in terms of memory usage, KNN-GWR stores a small matrix consisting of observations involved in the regression rather than a full sample matrix.

5. Conclusions

Geographically weighted regression has become a classical method for exploring the spatial non-stationarity in geographic data. However, it faces computational challenges when applied to geographical big data. Previous studies have adopted strategies such as parallelism to optimize GWR, which can handle applications with millions of geographical spatial data, but still encounter problems of memory and computational efficiency. This paper aims to address this limitation, and the main research results are summarized as follows:

(1) The KD tree is proposed to organize geographical spatial data, which can quickly identify the observation points around the regression points in local regression calculations. This greatly optimizes the time-consuming search process in the geographically weighted regression model and significantly improves the computational efficiency.

(2) This paper reconstructed the weight matrix, the independent variable matrix, and the dependent variable matrix based on the characteristics of the kernel function in GWR. This achieves the transformation from a large matrix $(n \times n)$ to a small matrix $(K \times K)$, avoiding the large memory consumption caused by the large matrix $(n \times n)$ during regression calculation in classical geographically weighted regression.

(3) The spatial clustering relationships of the geographic data obtained by the K-means clustering method are referenced to help narrow the search for the optimal bandwidth. This reduces the computational waste in the process of determining the optimal bandwidth in GWR.

Author Contributions: Conceptualization, Xiaoyue Yang; methodology, Xiaoyue Yang, Yi Yang and Shenghua Xu; software, Xiaoyue Yang and Yi Yang; formal analysis, Xiaoyue Yang; data curation, Xiaoyue Yang, Zhengyuan Chai and Gang Yang; writing—original draft, Xiaoyue Yang; writing—review and editing, Yi Yang, Shenghua Xu and Jiakuan Han; visualization, Xiaoyue Yang and Jiakuan Han; supervision, Shenghua Xu, Jiakuan Han, Zhengyuan Chai and Gang Yang; funding acquisition, Yi Yang. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 42001343; and the National Natural Science Foundation of China, grant number 42071384.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Fotheringham, A.S.; Brunsdon, C.; Charlton, M.E. *Geographically Weighted Regression*; John Wiley & Sons: Hoboken, NJ, USA, 2002.
2. Brunsdon, C.; Fotheringham, A.S.; Charlton, M.E. Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity. *Geogr. Anal.* **1996**, *28*, 281–298. [CrossRef]
3. Shi, T.; Hu, X.; Guo, L.; Su, F.; Tu, W.; Hu, Z.; Liu, H.; Yang, C.; Wang, J.; Zhang, J.; et al. Digital mapping of zinc in urban topsoil using multisource geospatial data and random forest. *Sci. Total Environ.* **2021**, *792*, 148455. [CrossRef]
4. Jiang, W.; Rao, P.; Cao, R.; Tang, Z.; Chen, K. Comparative evaluation of geological disaster susceptibility using multi-regression methods and spatial accuracy validation. *J. Geogr. Sci.* **2017**, *27*, 439–462. [CrossRef]
5. Kumar, S.; Lal, R.; Liu, D. A geographically weighted regression kriging approach for mapping soil organic carbon stock. *Geoderma* **2012**, *189–190*, 627–634. [CrossRef]
6. Davies, T.J.; Regetz, J.; Wolkovich, E.M.; McGill, B.J.; Kerkhoff, A. Phylogenetically weighted regression: A method for modelling non-stationarity on evolutionary trees. *Glob. Ecol. Biogeogr.* **2018**, *28*, 275–285. [CrossRef]
7. Mellin, C.; Mengersen, K.; Bradshaw, C.J.A.; Caley, M.J. Generalizing the use of geographical weights in biodiversity modelling. *Glob. Ecol. Biogeogr.* **2014**, *23*, 1314–1323. [CrossRef]
8. Yang, L.; Chau, K.W.; Szeto, W.Y.; Cui, X.; Wang, X. Accessibility to transit, by transit, and property prices: Spatially varying relationships. *Transp. Res. Part D Transp. Environ.* **2020**, *85*, 102387. [CrossRef]
9. Wu, C.; Ren, F.; Hu, W.; Du, Q. Multiscale geographically and temporally weighted regression: Exploring the spatiotemporal determinants of housing prices. *Int. J. Geogr. Inf. Sci.* **2018**, *33*, 489–511. [CrossRef]
10. Fotheringham, A.S.; Crespo, R.; Yao, J. Geographical and Temporal Weighted Regression (GTWR). *Geogr. Anal.* **2015**, *47*, 431–452. [CrossRef]
11. Huang, B.; Wu, B.; Barry, M. Geographically and temporally weighted regression for modeling spatio-temporal variation in house prices. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 383–401. [CrossRef]
12. Hong, Z.; Mei, C.; Wang, H.; Du, W. Spatiotemporal effects of climate factors on childhood hand, foot, and mouth disease: A case study using mixed geographically and temporally weighted regression models. *Int. J. Geogr. Inf. Sci.* **2021**, *35*, 1611–1633. [CrossRef]
13. Hong, Z.; Hao, H.; Li, C.; Du, W.; Wei, L.; Wang, H. Exploration of potential risks of Hand, Foot, and Mouth Disease in Inner Mongolia Autonomous Region, China Using Geographically Weighted Regression Model. *Sci. Rep.* **2018**, *8*, 17707. [CrossRef]
14. Mainardi, S. Modelling spatial heterogeneity and anisotropy: Child anaemia, sanitation and basic infrastructure in sub-Saharan Africa. *Int. J. Geogr. Inf. Sci.* **2012**, *26*, 387–411. [CrossRef]
15. Lu, X.Y.; Chen, X.; Zhao, X.L.; Lv, D.J.; Zhang, Y. Assessing the impact of land surface temperature on urban net primary productivity increment based on geographically weighted regression model. *Sci. Rep.* **2021**, *11*, 22282. [CrossRef]
16. Bivand, R.; Yu, D.; Nakaya, T.; Garcia-Lopez, M.-A. *Package SPGWR*; R Software Package; R Foundation for Statistical Computing: Vienna, Austria, 2022.
17. Oshan, T.; Li, Z.; Kang, W.; Wolf, L.; Fotheringham, A. mgwr: A Python Implementation of Multiscale Geographically Weighted Regression for Investigating Process Spatial Heterogeneity and Scale. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 269. [CrossRef]
18. Gollini, I.; Lu, B.; Charlton, M.; Brunsdon, C.; Harris, P. GWmodel: An R Package for Exploring Spatial Heterogeneity Using Geographically Weighted Models. *J. Stat. Softw.* **2015**, *63*, 1–50. [CrossRef]
19. Li, Z.; Fotheringham, A.S.; Li, W.; Oshan, T. Fast Geographically Weighted Regression (FastGWR): A scalable algorithm to investigate spatial process heterogeneity in millions of observations. *Int. J. Geogr. Inf. Sci.* **2019**, *33*, 155–175. [CrossRef]
20. Sudmanns, M.; Tiede, D.; Lang, S.; Bergstedt, H.; Trost, G.; Augustin, H.; Baraldi, A.; Blaschke, T. Big Earth data: Disruptive changes in Earth observation data management and analysis? *Int. J. Digit. Earth* **2019**, *13*, 832–850. [CrossRef]
21. Ma, Y.; Wu, H.; Wang, L.; Huang, B.; Ranjan, R.; Zomaya, A.; Jie, W. Remote sensing big data computing: Challenges and opportunities. *Future Gener. Comput. Syst.* **2015**, *51*, 47–60. [CrossRef]
22. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
23. Lü, G.; Batty, M.; Strobl, J.; Lin, H.; Zhu, A.X.; Chen, M. Reflections and speculations on the progress in Geographic Information Systems (GIS): A geographic perspective. *Int. J. Geogr. Inf. Sci.* **2018**, *33*, 346–367. [CrossRef]
24. Apte, J.S.; Messier, K.P.; Gani, S.; Brauer, M.; Kirchstetter, T.W.; Lunden, M.M.; Marshall, J.D.; Portier, C.J.; Vermeulen, R.C.H.; Hamburg, S.P. High-Resolution Air Pollution Mapping with Google Street View Cars: Exploiting Big Data. *Environ. Sci. Technol.* **2017**, *51*, 6999–7008. [CrossRef]
25. Lee, J.-G.; Kang, M. Geospatial Big Data: Challenges and Opportunities. *Big Data Res.* **2015**, *2*, 74–81. [CrossRef]
26. Mendi, A.F.; Demir, Ö.; Sakaklı, K.K.; Çabuk, A. A New Approach to Land Registry System in Turkey: Blockchain-Based System Proposal. *Photogramm. Eng. Remote Sens.* **2020**, *86*, 701–709. [CrossRef]
27. Finley, A.O. Comparing spatially-varying coefficients models for analysis of ecological data with non-stationary and anisotropic residual dependence. *Methods Ecol. Evol.* **2011**, *2*, 143–154. [CrossRef]
28. Harris, R. Grid-enabling Geographically Weighted Regression: A Case Study of Participation in Higher Education in England. *Trans. GIS* **2010**, *14*, 43–61. [CrossRef]
29. Yu, D. Modeling Owner-Occupied Single-Family House Values in the City of Milwaukee: A Geographically Weighted Regression Approach. *GISci. Remote Sens.* **2007**, *44*, 267–282. [CrossRef]

30. Feuillet, T.; Commenges, H.; Menai, M.; Salze, P.; Perchoux, C.; Reuillon, R.; Kesse-Guyot, E.; Enaux, C.; Nazare, J.A.; Hercberg, S.; et al. A massive geographically weighted regression model of walking-environment relationships. *J. Transp. Geogr.* **2018**, *68*, 118–129. [CrossRef]
31. Wang, D.; Yang, Y.; Qiu, A.; Kang, X.; Han, J.; Chai, Z. A CUDA-Based Parallel Geographically Weighted Regression for Large-Scale Geographic Data. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 653. [CrossRef]
32. Tasyurek, M.; Celik, M. RNN-GWR: A geographically weighted regression approach for frequently updated data. *Neurocomputing* **2020**, *399*, 258–270. [CrossRef]
33. Meenakshi; Gill, S. k-dLst Tree: K-d Tree with Linked List to Handle Duplicate Keys. In Proceedings of the Emerging Trends in Expert Applications and Security, Singapore, 17–18 February 2018; pp. 167–175.
34. Chen, Z.Y.; Liao, I.Y.; Ahmed, A. KDT-SPSO: A multimodal particle swarm optimisation algorithm based on k-d trees for palm tree detection. *Appl. Soft Comput.* **2021**, *103*, 107156. [CrossRef]
35. Shyu, C.R.; Chi, P.H.; Scott, G.; Xu, D. ProteinDBS: A real-time retrieval system for protein structure comparison. *Nucleic Acids Res.* **2004**, *32*, W572–W575. [CrossRef] [PubMed]
36. Böhm, C.; Krebs, F. The k-Nearest Neighbour Join: Turbo Charging the KDD Process. *Knowl. Inf. Syst.* **2004**, *6*, 728–749. [CrossRef]
37. Muja, M.; Lowe, D.G. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 2227–2240. [CrossRef]
38. Muja, M. Fast approximate nearest neighbors with automatic algorithm configuration. *Proc. Viss.* **2009**, *1*, 331–340.
39. Boukerche, A.; Zheng, L.; Alfandi, O. Outlier detection: Methods, models, and classification. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–37. [CrossRef]
40. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666. [CrossRef]
41. Fahad, A.; Alshatri, N.; Tari, Z.; Alamri, A.; Khalil, I.; Zomaya, A.Y.; Foufou, S.; Bouras, A. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 267–279. [CrossRef]
42. Zhao, W.-L.; Deng, C.-H.; Ngo, C.-W. k-means: A revisit. *Neurocomputing* **2018**, *291*, 195–206. [CrossRef]
43. Macqueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, 21 June–18 July 1967; Volume 1.
44. Selim, S.Z.; Ismail, M.A. K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, *PAMI-6*, 81–87. [CrossRef]
45. Li, S.; Lyu, D.; Huang, G.; Zhang, X.; Gao, F.; Chen, Y.; Liu, X. Spatially varying impacts of built environment factors on rail transit ridership at station level: A case study in Guangzhou, China. *J. Transp. Geogr.* **2020**, *82*, 102631. [CrossRef]
46. Hernández, J.M.; Bulchand-Gidumal, J.; Suárez-Vega, R. Using accommodation price determinants to segment tourist areas. *J. Destin. Mark. Manag.* **2021**, *21*, 100622. [CrossRef]
47. Deng, X.; Gao, F.; Liao, S.; Li, S. Unraveling the association between the built environment and air pollution from a geospatial perspective. *J. Clean. Prod.* **2023**, *386*, 135768. [CrossRef]
48. Murakami, D.; Tsutsumida, N.; Yoshida, T.; Nakaya, T.; Lu, B. Scalable GWR: A Linear-Time Algorithm for Large-Scale Geographically Weighted Regression with Polynomial Kernels. *Ann. Am. Assoc. Geogr.* **2020**, *111*, 459–480. [CrossRef]
49. Murakami, D.; Griffith, D.A. Spatially varying coefficient modeling for large datasets: Eliminating N from spatial regressions. *Spat. Stat.* **2019**, *30*, 39–64. [CrossRef]
50. Mardia, K.V.; Kent, J.T.; Bibby, J.M. *Multivariate Analysis*; Academic Press: London, UK, 1979.
51. Carlis, J.; Bruso, K. Rsqrt: An Heuristic for Estimating the Number of Clusters to Report. *Electron. Commer. Res. Appl.* **2012**, *11*, 152–158. [CrossRef]
52. Hassanat, A.B.; Abbadi, M.A.; Altarawneh, G.A.; Alhasanat, A.A. Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach. *Comput. Sci.* **2014**, *12*, 33–39.
53. Sugar, C.A.; James, G.M. Finding the Number of Clusters in a Dataset. *J. Am. Stat. Assoc.* **2003**, *98*, 750–763. [CrossRef]
54. Tibshirani, R.; Walther, G.; Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *J. R. Stat. Soc. Ser. B* **2001**, *63*, 411–423. [CrossRef]
55. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes: The Art of Scientific Computing*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2007.
56. Harris, P.; Fotheringham, A.S.; Crespo, R.; Charlton, M. The Use of Geographically Weighted Regression for Spatial Prediction: An Evaluation of Models Using Simulated Data Sets. *Math. Geosci.* **2010**, *42*, 657–680. [CrossRef]
57. Chen, F.; Mei, C.-L. Scale-adaptive estimation of mixed geographically weighted regression models. *Econ. Model.* **2021**, *94*, 737–747. [CrossRef]