Comparative investigation of parallel spatial interpolation algorithms for building large-scale digital elevation models

Jingzhi Tu [1] , **Guoxiang Yang** [Corresp., 1] , **Pian Qi** [1] , **Zengyu Ding** [1] , **Gang Mei** [Corresp. 1]

[1] School of Engineering and Technology, China University of Geoscience (Beijing), Beijing, Beijing, China

Corresponding Authors: Guoxiang Yang, Gang Mei
Email address: yanggx@cugb.edu.cn, gang.mei@cugb.edu.cn

The building of large-scale Digital Elevation Models (DEMs) using various interpolation algorithms is one of the key issues in geographic information science. Different choices of interpolation algorithms may trigger significant differences in interpolation accuracy and computational efficiency, and a proper interpolation algorithm needs to be carefully used based on the specific characteristics of the scene of interpolation. In this paper, we comparatively investigate the performance of parallel Radial Basis Function (RBF)-based, Moving Least Square (MLS)-based, and Shepard's interpolation algorithms for building DEMs by evaluating the influence of terrain type, raw data density, and distribution patterns on the interpolation accuracy and computational efficiency. The drawn conclusions may help select a suitable interpolation algorithm in a specific scene to build large-scale DEMs.

Comparative Investigation of Parallel Spatial Interpolation Algorithms for Building Large-scale Digital Elevation Models

Jingzhi Tu[1], Guoxiang Yang[1], Pian Qi[1], Zengyu Ding[1], and Gang Mei[1]

[1]**School of Engineering and Technology, China University of Geosciences (Beijing), Beijing, China**

Corresponding author:

Guoxiang Yang, Gang Mei[1]

Email address: yanggx@cugb.edu.cn; gang.mei@cugb.edu.cn

ABSTRACT

The building of large-scale Digital Elevation Models (DEMs) using various interpolation algorithms is one of the key issues in geographic information science. Different choices of interpolation algorithms may trigger significant differences in interpolation accuracy and computational efficiency, and a proper interpolation algorithm needs to be carefully used based on the specific characteristics of the scene of interpolation. In this paper, we comparatively investigate the performance of parallel Radial Basis Function (RBF)-based, Moving Least Square (MLS)-based, and Shepard's interpolation algorithms for building DEMs by evaluating the influence of terrain type, raw data density, and distribution patterns on the interpolation accuracy and computational efficiency. The drawn conclusions may help select a suitable interpolation algorithm in a specific scene to build large-scale DEMs.

INTRODUCTION

Digital Elevation Model (DEM) is a numerical representation of topography made up of equal-sized grid cells, each with a value of elevation. One of the most important scientific challenges of digital elevation modeling is the inefficiency of most interpolation algorithms in dealing with a large amount of data produced by large-scale DEM with a fine resolution. To solve the problem, one of the common strategies is to parallelize interpolation algorithms on various High Performance Computing (HPC) platforms.

For different large-scale DEM, different parallel spatial interpolation algorithms are usually specifically selected, because a variety of spatial interpolation algorithms exist that behave differently for different data configurations and landscape conditions. Consequently, the accuracy of a DEM is sensitive to the interpolation technique, and it is significant to understand how the various algorithms affect a DEM. Therefore, this study is being conducted.

Spatial interpolation is a category of important algorithms in the field of geographic informa-
tion. Siu-Nganlam (1983) had a review of various interpolation algorithms, including most distance-
weighting methods, Kriging, spline interpolation, interpolating polynomials, finite-difference meth-
ods, power-series trend models, Fourier models, distance-weighted least-squares, and least-squares
fitting with splines. Many spatial interpolation algorithms are used to build DEMs, for example, the
Shepard's method (IDW) (Shepard, 1968), the Kriging method (Krige, 1953), the Discrete Smooth-
ing Interpolation (DSI) method (Mallet, 1997), the Radial Basis Function (RBF)-based method
(Powell, 1977), and the Moving Least Squares (MLS)-based method (Lancaster and Salkauskas,
1981).

Much research work (Gumus and Sen, 2013; Chaplot et al., 2006; Aguilar et al., 2005; Khairnar
et al., 2015; Polat et al., 2015; Rishikeshan et al., 2014) has been conducted to evaluate the effects of
different interpolation methods on the precision of DEM interpolation. In the comparative investi-
gation of spatial interpolation algorithms for building DEMs, quite few studies specifically focused
on the impact of data samples and terrain types on interpolation accuracy, among them, Gumus and
Sen (2013) compared the accuracy of various interpolation methods at different point distributions,
the interpolation performance of IDW is worse than other algorithms for the same data distribution.
For the same algorithm, in the case of using all points and grid, their experimental results show that
the best interpolation performances are Modified Shepard's (MS) for random distribution; Multi-
quadric Radial Basis Function (MRBF) for curvature distribution, and Inverse Distance Weighted
(IDW) for uniform distribution.

Chaplot et al. (2006) and Aguilar et al. (2005) evaluated the effects of landform types and the
density of the original data on the accuracy of DEM production, their results show that interpola-
tion algorithms perform well at higher sampling densities, and MRBF provided significantly better
interpolation than IDW in rough or non-uniform terrain. At lower sampling densities, when the
spatial structure of height was strong, Kriging yielded better estimates. When the spatial structure
of height was weak, IDW and Regularized Spline with Tension (RST) performed better. On the
other hand, MRBF performed well in the mountainous areas and Ordinary Kriging (OK) was the
best for multi-scales interpolations in the smooth landscape. In addition, Zhang (2013) established
a descriptive model of local terrain features to study the correlation of surface roughness indicators
and spatial distribution indicators for DEM interpolation algorithms. (Chaplot et al., 2006). Ghan-
dehari et al. (2019) illustrated that the Bi-quadratic and Bi-cubic interpolation methods outperform
Weighted Average, Linear, and Bi-linear methods at coarse resolutions and in rough or non-uniform
terrain. Aguilar et al. (2005) pointed out that MRBF is better than Multilog function for low sample
densities and steeper terrain.

With the increasing size of DEMs, it is increasingly necessary to design parallel solutions for
existing sequential algorithms to speed up processing. When adopting an interpolation method to
deal with a large DEM, the computational cost would be quite expensive, and the computational
efficiency would especially be unsatisfied.

The techniques in HPC are widely used to improve computational efficiency in various science
and engineering applications such as surface modeling (Yan et al., 2016), spatial point pattern

73 analysis (Zhang et al., 2017), urban growth simulation (Guan et al., 2016), Delaunay Triangulation
74 (DT) for GIS (Coll and Guerrieri, 2017), spatial interpolation (Wang et al., 2017; Cheng, 2013;
75 Mei, 2014; Mei et al., 2017; Mei, 2014; Mei et al., 2016; Ding et al., 2018b), and image processing
76 (Wasza et al., 2011; Lei et al., 2011; Yin et al., 2014; Wu et al., 2018).

77 One of the effective strategies to solve the problem is to perform the DEM interpolation in
78 parallel on various parallel computing platforms such as shared-memory computers, distributed-
79 memory computers, or even clusters. The parallelization of DEM interpolation can be developed
80 with the computational power of modern multicore Central Processing Units (CPUs) and many-
81 core Graphics Processing Units (GPUs). For example, Zhou et al. (2017) proposed a parallel Open
82 Multi-Processing (OpenMP)- and Message Passing Interface (MPI)-based implementation of the
83 Priority-Flood algorithm that identifies and fills depressions in raster DEMs. Yan et al. (2015)
84 accelerated high-accuracy surface modeling (HASM) in constructing large-scale and fine resolu-
85 tion DEM surfaces by the use of GPUs and applied this acceleration algorithm to simulations of
86 both ideal Gaussian synthetic surfaces and real topographic surfaces in the loess plateau of Gansu
87 province. Tan et al. (2017) presented a novel method to generate contour lines from grid DEM
88 data, based on the programmable GPU pipeline, that can be easily integrated into a 3D GIS system.
89 Chen et al. (2010) demonstrated a new algorithm for reconstructing contour maps from raster DEM
90 data for digital-earth and other terrain platforms in real-time entirely based on modern GPUs and
91 programmable pipelines.

92 The RBF, Kriging, MLS and Shepard's interpolation algorithms are the most frequently used
93 spatial interpolation algorithms, among which, the Kriging method can be regarded as an instance
94 of RBF framework (Peng et al., 2019). Therefore, in this paper, we comparatively investigate the
95 performance of the RBF-based, MLS-based, and Shepard's interpolation algorithms for building
96 DEMs by evaluating the influence of terrain type, raw data density, and distribution patterns on the
97 interpolation accuracy and computational efficiency.

98 The rest of the paper is organized as follows. Section 2 briefly introduces the basic principles
99 of eight interpolation methods. Section 3 concentrates mainly on our parallel implementations of
100 the eight interpolation methods and creation of the testing data. Section 4 introduces some of the
101 experimental tests performed on the CPU and GPU. Section 5 discusses the experimental results.
102 Finally, Section 6 states conclusions from the work.

BACKGROUND

104 In this section, we briefly introduce eight spatial interpolation algorithms.

MLS-based Interpolation Algorithms

106 The MLS method obtains the fitting surface by solving the equation group derived from mini-
107 mizing the sum of the squares of the errors between the fitting data and the given node data.

Original MLS Interpolation Algorithm

109 The MLS approximation is used to approximate field variables and their derivatives. In a
110 domain Ω, the MLS approximation $f^h(x)$ of the field variable $f(x)$ in the vicinity of a point $\bar{x}$ is

111 given as

$$f^h(x) = \sum_{j=1}^{m} p_j(x) \cdot a_j(\bar{x}) = P^T(x) \cdot a(\bar{x}) \tag{1}$$

112 where $p_j(x), j = 1, 2, \cdots, m$ is a complete basis function with coefficients $a_j(\bar{x})$. At each point $\bar{x}$,

113 $a_j(\bar{x})$ is chosen to minimize the weighted residual L_2- norm (L_2- norm refers to $\|x\|_2$, where

114 $x = [x_1, x_2, \cdots, x_n]^T$, and $\|x\|_2 = \sqrt{\left(|x_1|^2 + |x_2|^2 + |x_3|^2 + \cdots + |x_n|^2\right)}$):

$$J = \sum_{I=1}^{N} w(\bar{x} - x_I)\left[P^T(x_I)a(\bar{x}) - f_I\right]^2 \tag{2}$$

115 where N is the number of nodes in the compact-supported neighborhood of $\bar{x}$ and f_I refers to the

116 nodal parameter of f at $x = x_I$. Nodes refer to data points in the compact-supported neighborhood

117 of $\bar{x}$. Compact-supported, i.e. point $\bar{x}$ is only related to the nodes of its neighborhood, x_I is one of

118 the nodes in the compact-supported neighborhood. And $w(x - x_k)$ is the compact-supported weight

119 function. The most commonly used weight functions are the spline functions, for example, the

120 cubic spline weight function (Eq. (3)):

$$w(\bar{s}) = \begin{cases} \frac{2}{3} - 4\bar{s}^2 + 4\bar{s}^3, & \bar{s} \leq \frac{1}{2} \\ \frac{4}{3} - 4\bar{s} + 4\bar{s}^2 - \frac{4}{3}\bar{s}^3, & \frac{1}{2} < \bar{s} \leq 1 \\ 0, & \bar{s} > 1 \end{cases} \tag{3}$$

121 where $\bar{s} = \frac{s}{s_{max}}$ and $s = \bar{x} - x_I$.

122   The minimum of J with respect to $a(\bar{x})$ gives the standard form of MLS approximation:

$$f^h(x) = \sum_{I=1}^{N} \phi_I(x) f_I = \Phi(x) F \tag{4}$$

123 ***Orthogonal MLS Interpolation Algorithm***

124   For a given polynomial basis function $p_i(x), \quad i = 1, 2, \cdots, m$, there is an orthonormal basis

125 function $q_i(x, \bar{x})$ that satisfies:

$$q_1(x, \bar{x}) = p_1(x)$$

$$q_i(x, \bar{x}) = p_i(x) - \sum_{j=1}^{i-1} \alpha_{ij}(x, \bar{x}) q_j(x, \bar{x}), i = 2, 3, \cdots, m \tag{5}$$

126 where $\alpha_{ij}(x, \bar{x})$ is the coefficient that makes $q_i(x, \bar{x})$ perpendicular to $q_j(x, \bar{x})$.

$$\alpha_{ij}(\bar{x}) = \frac{\sum_{k=1}^{N} w_k(\bar{x}) p_i(x_k) q_j(x_k, \bar{x})}{\sum_{k=1}^{N} w_k(\bar{x}) q_j^2(x_k, \bar{x})} \tag{6}$$

127   Because the coefficient matrix is a diagonal matrix, the solution for $a_i(x)$ does not require

128 matrix inversion, i.e.

$$a_i(\bar{x}) = \frac{\sum_{k=1}^{N} w_k(\bar{x}) q_i(x_k, \bar{x}) f_k}{\sum_{k=1}^{N} w_k(\bar{x}) q_i^2(x_k, \bar{x})} \tag{7}$$

129 where a_i and $a_j(\bar{x})$ (Eqs. (1)) have the same definition. f_k and f_I (Eqs. (2)) have the same definition,

130 i.e., the nodal parameter of f at $x = x_k$. Finally, a_i and the orthonormal basis function $q_i(x, \bar{x})$ are

131 fitted into Eqs. (1) to obtain the orthogonal MLS approximation $f^h(x)$.

132   When the number or order of basis functions increases, only a_{m+1} and α_{m+1} need to be calcu-

133 lated in Gram-Schmidt orthogonalization (Steve, 2011); recalculation of all entries in the coefficient

134 matrix is not needed. This could reduce the computational cost and the computational error.

135 ### *Lancaster's MLS Interpolation Algorithm*

136   A singular weight function is adopted to make the approximation function $f^h(x)$ constructed

137 by the interpolation type MLS method satisfy the properties of the Kronecker δ function:

$$\omega(x, x_k) = \begin{cases} \left\| (x - x_k)/\rho_k \right\|^{-\alpha}, & \|x - x_k\| \le \rho_k \\ 0, & \|x - x_k\| > \rho_k \end{cases} \tag{8}$$

138   Let $p_0(x) \equiv 1, p_1(x), \cdots, p_{\bar{m}}(x)$ denote the basis function used to construct the approximation

139 function, where the number of basis functions is $\bar{m} + 1$. To implement the interpolation properties,

140 a new set of basis functions is constructed for a given basis function. First, $p_0(x)$ are standardized,

141 i.e.,

$$\tilde{p}_0(x, \bar{x}) = \frac{1}{\left[\sum_{k=1}^{N} \omega(x, x_k) \right]^{1/2}} \tag{9}$$

142   Then, we construct a new basis function of the following form:

$$\tilde{p}_i(x, \bar{x}) = p_i(\bar{x}) - \sum_{k=1}^{N} \frac{\omega(x, x_k)}{\sum_{l=1}^{N} \omega(x, x_l)} P_i(x_k), i = 1, 2, \cdots, \bar{m} \tag{10}$$

143 ### **RBF-based Interpolation Algorithm**

144   The RBF operates as a spline, essentially fitting a series of piecewise surfaces to approximate

145 a complex terrain.

146   Let $X = \{x_1, x_2, \cdots, x_N\}$ be a set of pairwise distinct points in a domain $\Omega \subseteq R^d$ with associated

147 data values $f_i, i = 1, 2, \cdots, N$. We consider the problem of construction a d-variety function $F \in$

148 $C^k(R^d)$ that interpolates the known data. Specifically, we require $F(x_i) = f_i, i = 1, 2, \cdots, N$. If we

149 take F in the form.

$$F(x) = \sum_{j=1}^{N} w_j \varphi\left(\|x_i - x_j\|_2\right) \tag{11}$$

150   where $\varphi : [0, \infty] \to R$ is a suitable continuous function, the interpolation conditions become:

$$\sum_{j=1}^{N} w_j \varphi \left(\|x_i - x_j\|_2 \right) = f_i, \quad i = 1, 2, \cdots, N \tag{12}$$

Shepard's Interpolation Algorithms

152   Shepard (1968) proposed a series of interpolation algorithms on the basis of weighting aver-
153   ages. These algorithms are termed Shepard's method. The essential idea behind Shepard's method
154   is to estimate expected values of the interpolation point by weighting averages of the nearby discrete
155   points as follows:

156   Let $(x_i, y_i), i = 1, 2, \cdots, N$ be the interpolation point and f_i be the corresponding value at inter-
157   polation point (x_i, y_i). The expected value f at any point can be expressed as

$$f(x) = \sum_{i=1}^{N} \frac{w_i(x) f_i}{\sum_{j=1}^{N} w_j(x)} \tag{13}$$

158   where $w(x)$ is a weight function.

159   The differences between the different variants of Shepard's method are in the selection of
160   different weighting functions. In this subsection, four common variants of Shepard's method will
161   be briefly introduced (Eqs. (14) - (19)).

Variant A of Shepard's Interpolation Algorithm

163   First, select the influence radius $R > 0$ and let the weight function be

$$w(r) = \begin{cases} \frac{1}{r}, & 0 < r \leq \frac{R}{3} \\ \frac{27}{4} \left(\frac{r}{R} - 1 \right)^2, & \frac{R}{3} < r \leq R \\ 0, & r > R \end{cases} \tag{14}$$

164   Then, a variation of Shepard's interpolation will be obtained.

Variant B of Shepard's Interpolation Algorithm

166   When employing the following weight function (Eq. (15)), a new variation of Shepard's inter-
167   polation will be obtained.

$$w(\bar{s}) = \begin{cases} \frac{2}{3} - 4\bar{s}^2 + 4\bar{s}^3, & \bar{s} \leq \frac{1}{2} \\ \frac{4}{3} - 4\bar{s} + 4\bar{s}^2 - \frac{4}{3}\bar{s}^3, & \frac{1}{2} < \bar{s} \leq 1 \\ 0, & \bar{s} > 1 \end{cases} \tag{15}$$

Inverse Distance Weighted (IDW) Interpolation Algorithm

169   If the weight function is selected as

$$w_i(x) = \frac{1}{d(x, x_i)^{\alpha}} \tag{16}$$

170   the IDW interpolation is obtained. Typically, $\alpha = 2$ in the standard IDW. Where $d(x, x_i)$ is the
171   distance between the interpolation point x_i and the nearby discrete point x.

172 ***AIDW Interpolation Algorithm***

173 The Adaptive Inverse Distance Weighted (AIDW) is an improved version of the standard IDW

174 (Shepard, 1968) originated by Lu and Wong (2008). The distance-decay parameter α is no longer

175 a prespecified constant value but is adaptively adjusted for a specific unknown interpolated point

176 according to the distribution of the nearest neighboring data points.

177 The parameter α is taken as

$$
\alpha\left(\mu_R\right)=\begin{cases}\alpha_1, & 0.0\le\mu_R\le 0.1\\ \alpha_1\left[1-5\left(\mu_R-0.1\right)\right]+5\alpha_2\left(\mu_R-0.1\right), & 0.1\le\mu_R\le 0.3\\ 5\alpha_3\left(\mu_R-0.3\right)+\alpha_2\left[1-5\left(\mu_R-0.3\right)\right], & 0.3\le\mu_R\le 0.5\\ \alpha_3\left[1-5\left(\mu_R-0.5\right)\right]+5\alpha_4\left(\mu_R-0.5\right), & 0.5\le\mu_R\le 0.7\\ 5\alpha_5\left(\mu_R-0.7\right)+\alpha_4\left[1-5\left(\mu_R-0.7\right)\right], & 0.7\le\mu_R\le 0.9\\ \alpha_5, & 0.9\le\mu_R\le 1.0\end{cases}\tag{17}
$$

$$
\mu_R=\begin{cases}0, & R\left(S_0\right)\le R_{\min}\\ 0.5-0.5\cos\left[\pi\left(R\left(S_0\right)-R_{\min}\right)/R_{\max}\right], & R_{\min}\le R\left(S_0\right)\le R_{\max}\\ 1, & R\left(S_0\right)\ge R_{\max}\end{cases}\tag{18}
$$

178 where the α_1, α_2, α_3, α_4, α_5 are the to-be-assigned five levels or categories of distance decay value.

179 $R_{\min}$ or $R_{\max}$ refer to a local nearest neighbor statistic value, and $R_{\min}$ and $R_{\max}$ can generally be

180 set to 0.0 and 2.0, respectively. Then,

$$
R\left(S_0\right)=\frac{2\sqrt{N/A}}{k}\sum_{i=1}^{k}d_i\tag{19}
$$

181 where N is the number of points in the study area, A is the area of the study region, k is the number of

182 nearest neighbor points, d_i is the nearest neighbor distances and S_0 is the location of an interpolated

183 point.

METHODS
184

Implementations of the Spatial Interpolation Algorithms
185

186 We have implemented the spatial interpolation algorithms of RBF (Ding et al., 2018b), MLS

187 (Ding et al., 2018a), IDW (Mei, 2014), and AIDW (Mei et al., 2017) in our previous work. To

188 evaluate the computational performance of the GPU-accelerated interpolation, we implement and

189 compare (1) the sequential implementation, (2) the parallel implementation developed on a multi-

190 core CPU, (3) the parallel implementation using a single GPU, and (4) the parallel implementation

191 using multiple GPUs.

192 There are two key ideas behind the presented spatial interpolation algorithm:

193 (1) We use an efficient k-Nearest Neighbor (kNN) search algorithm (Mei et al., 2016) to find

194 the local set of data points for each interpolated point.

195 (2) We employ the local set of data points to compute the prediction value of the interpolated

196 point using different interpolation methods.

197   Mei and Tian (2016) evaluated the impact of different data layouts on the computational ef-
198 ficiency of the GPU-accelerated IDW interpolation algorithm. They implemented three IDW ver-
199 sions of GPU implementations, based upon five data layouts, including the Structure of Arrays
200 (SoA), the Array of Structures (AoS), the Array of aligned Structures (AoaS), the Structure of Ar-
201 rays of aligned Structures (SoAoS), and a hybrid layout, then they carried out several groups of
202 experiments to evaluate the impact of different data layouts on the interpolation efficiency. Based
203 on their experimental results, the layout SoA is shown in Listing 1.

```
struct Pt {
    float x[N];
    float y[N];
    float z[N];
};
struct Pt myPts;
```

Listing 1. The layout SoA

212   The *k*NN (Cover and Hart, 1967) is a machine learning algorithm often used in classification,
213 the *k*-Nearest Neighbor means that each data point can be represented by its *k* nearest neighbor
214 points. In all of the presented interpolation algorithms, for each interpolation point, a local set of
215 data points is found by employing the *k*NN search procedure and the found local sets of data points
216 are then used to calculate the prediction value of the interpolation point. For large size of DEM,
217 the *k*NN search algorithm can effectively improve the speed of interpolation by searching only the
218 points near the interpolation points (Mei et al., 2016).

219   Assuming there are m interpolated points and n data points, the process of the *k*NN search
220 algorithm is as follows:

221   Step 1: The *k* distances between the *k* data points and each of the interpolated points are
222 calculated; for example, if the k is set to 5, then there are 5 distances needed to be calculated; see
223 the row (A) in Figure 1.

224   Step 2: The *k* distances are sorted in ascending order; see the row (B) in Figure 1.

225   Step 3: For each of the rest (m-*k*) data points,

226   (1) The distance *d* is calculated, for example, the distance is 4.2 ($d = 4.2$);

227   (2) The *d* with the *k*th distance are compared: if $d <$ the *k*th distance, then replace the *k*th
228 distance with the *d* (see row (C));

229   (3) Iteratively compare and swap the neighboring two distances from the *k*th distance to the
230 1st distance until all the *k* distances are newly sorted in ascending order; see the rows (C)–(E) in
231 Figure 1.

Creation of the Testing Data

233   Two sets of DEM data were downloaded from the Geospatial Data Cloud (`http://www.`
234 `gscloud.cn//`). More specifically, two 30-m resolution DEMs for two 20 km × 20 km regions
235 in Hebei and Sichuan provinces were selected. The topography of Hebei province is mainly plain,
236 while the topography of Sichuan province is mainly mountainous. Two sets of DEM data are

Original	0.3	8.6	1.5	5	6.2	(A)
Sorted	0.3	1.5	5	6.2	8.6	(B)
Replaced	0.3	1.5	5	6.2	4.2	(C)
Swapped	0.3	1.5	5	4.2	6.2	(D)
Desired	0.3	1.5	4.2	5	6.2	(E)

Figure 1. An illustration of the process of the *k*NN search algorithm

derived from remote sensing satellites and compiled by the CNIC (Computer Network Information Center, Chinese Academy of Sciences). More details on the selected DEMs are presented in Figure 2.

Data points and interpolated points (listed in Table 1 and Table 2) are produced as follows:

(1) The selected DEMs is imported into the software ArcGIS.

(2) A square region S is delimited in selected DEMs. For example, the two 20 km × 20 km regions shown in Figure 2.

(3) Generating the *x* and *y* coordinates of randomly determined points by random number generation algorithms in the square region S, and then accessing the corresponding *z* coordinates from the DEM (the randomly determined points are the data points *P*1). Evenly distributed (regularly distributed) data points are randomly extracted using the Linear Congruential Random Number Method (Lehmer, 1949), and normally distribution (irregularly distributed, mathematical expectation μ=10000, standard deviation σ=3333) data points are randomly extracted using the Box-Muller Method (Box and Muller, 1958). For example, we set Size 1, the extracted regularly distributed data points $P1 = 249990$ (Table 1), and density is $P1/S_0$ (S_0 is the area of S, and S_0 is a fixed value, where S_0=20 km × 20 km).

(4) The square region S is triangulated into a planar triangular mesh using the Delauney algorithm (Watson, 1981), the mesh nodes are considered to be the interpolation points, with known *x* and *y* coordinates and unknown *z* coordinates, the unknown *z* coordinates is the estimated value to be obtained by interpolation. According to the randomly sampled points obtained in Step 3, we use the interpolation method mentioned in Section 2 to interpolate. Then, the corresponding exact elevation of the interpolation point is obtained by accessing the *z* value of the DEM at the associated *x* and *y* coordinates. Finally, the *z* values at the mesh points are used as control for testing the accuracy of the interpolated *z* values.

To quantitatively determine regular and irregular point sampling, Average Nearest Neighbor analysis (Ebdon, 1985) is applied. In the proposed method, Nearest Neighbor Ratio (NNR) is used to evaluate the distribution pattern of sample points: if the NNR > 1, the distribution pattern shows clustered; if the NNR < 1, the distribution pattern shows dispersed. As listed in Table 3, the NNR of regularly-distributed, approximately 1.001, is greater than 1, the distribution pattern is dispersed
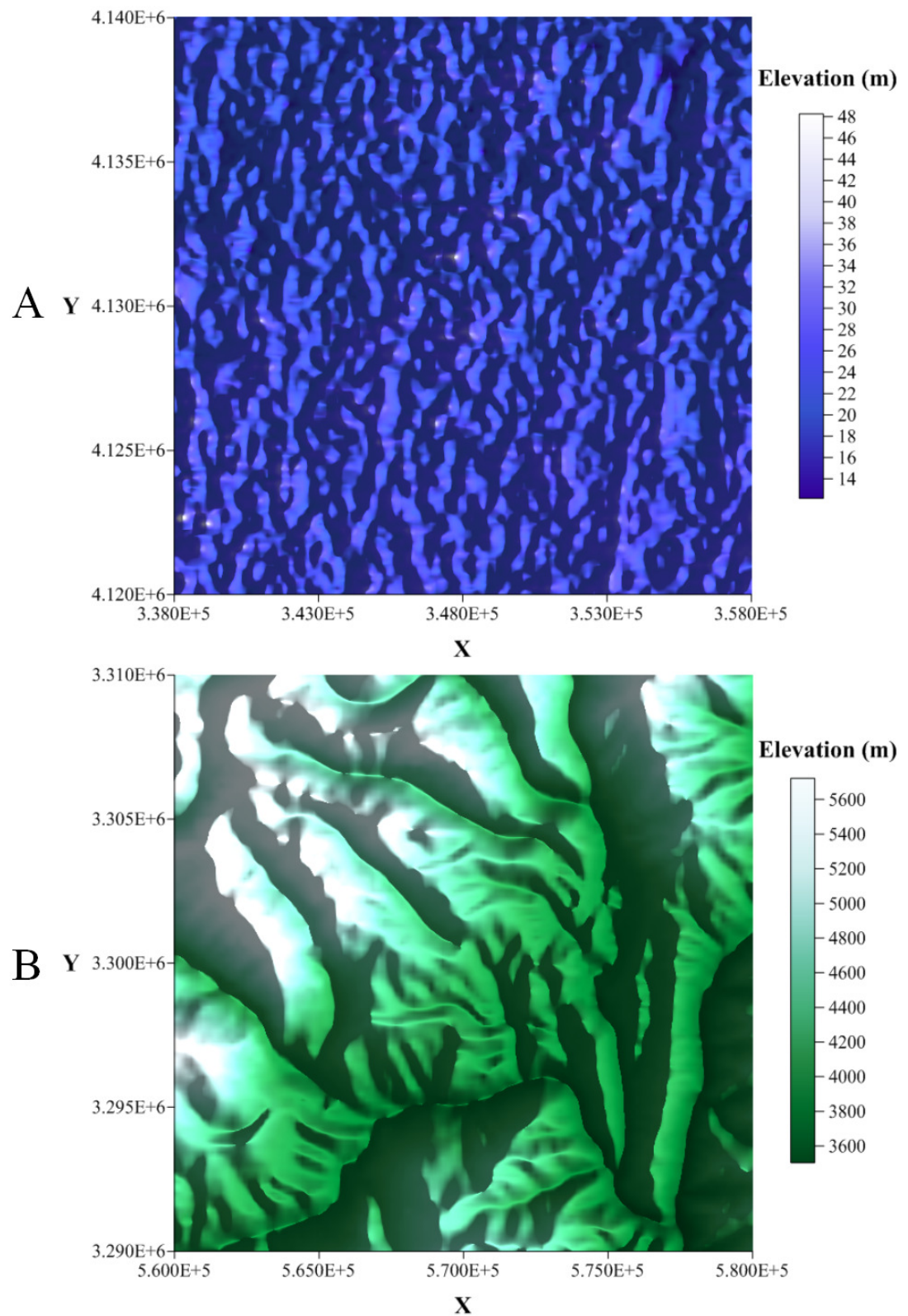
Figure 2. The selected Zone 1 and Zone 2. ((A) 2.5D model of the Zone 1 study area and (B) 2.5D model of the Zone 2 study area.)

Table 1. Ten used groups of experimental testing data in the Flat zone

Data set		Number of Data Points	Number of Interpolated Points
Regularly-distributed	Size 1	249990	259496
	Size 2	499975	529080
	Size 3	999883	1036780
	Size 4	1499750	1540373
	Size 5	1999566	2000520
Irregularly-distributed	Size 1	249920	259496
	Size 2	499751	529080
	Size 3	998840	1036780
	Size 4	1497397	1540373
	Size 5	1995531	2000520

Table 2. Ten used groups of experimental testing data in the Rugged zone

Data set		Number of Data Points	Number of Interpolated Points
Regularly-distributed	Size 1	249994	259496
	Size 2	499970	529080
	Size 3	999884	1036780
	Size 4	1499746	1540373
	Size 5	1999544	2000520
Irregularly-distributed	Size 1	249924	259496
	Size 2	499728	529080
	Size 3	998867	1036780
	Size 4	1497444	1540373
	Size 5	1995443	2000520

266 (Figure 3(A)), that is regularly-distributed; the NNR of irregularly-distributed, approximately 0.78,

267 is less than 1, the distribution pattern is clustered (Figure 3(B)), that is irregularly-distributed.

Zone 1 (Flat Zone)

269   The first selected region is located in Hengshui City, Hebei Province. The DEM of this region

270 has the identifier ASTGTM_N37E115 and is derived from the Geospatial Data Cloud (http://www.gscloud.cn/).

271 The location and elevation of this region is illustrated in Figure 2. In the region, the highest eleva-

272 tion is 48 m and the lowest is 8 m. We translated the X coordinate by 348,000 and the Y coordinate

273 by 4,130,000 to obtain a 20 km×20 km square area centered on the origin. Five sets of benchmark

274 test data were generated in this region; see Table 1.

Zone 2 (Rugged Zone)

276   The second selected region is located in Ganzi Tibetan Autonomous Prefecture, Sichuan

277 Province. The DEM of this region has the identifier ASTGTM_N29E099 and is derived from

278 the Geospatial Data Cloud (http://www.gscloud.cn/). The location and elevation of this region is

Figure 3. The distribution patterns determined by the Average Nearest Neighbor analysis. ((A) Regularly distributed and (B) irregularly distributed.)

Table 3. The NNR of regular and irregular point sampling

Data set		Flat Zone	Rugged Zone
Regularly-distributed	Size 1	1.001731	1.001170
	Size 2	1.001219	1.001291
	Size 3	1.001437	1.001173
	Size 4	1.001987	1.001758
	Size 5	1.002431	1.001869
Irregularly-distributed	Size 1	0.783242	0.781741
	Size 2	0.782947	0.784534
	Size 3	0.783653	0.784086
	Size 4	0.784653	0.784056
	Size 5	0.783745	0.784888

279   illustrated in Figure 2. In the region, the highest elevation is 5,722 m and the lowest is 3,498 m. We
280   translated the X coordinate by 570,000 and the Y coordinate by 3,300,000 to obtain a 20 km×20
281   km square area centered on the origin. Five sets of benchmark test data are generated in this region;
282   see Table 2.

Criteria for Comparison

In this paper, we evaluate the interpolation algorithms described in Section 2 by: (1) comparing the interpolation accuracy and efficiency when the terrain is gentle and rugged, and (2) comparing the interpolation accuracy and efficiency when data points are evenly distributed and nonuniformly distributed.

The accuracy of each interpolation method is analyzed by comparing the elevation values predicted by the interpolation algorithms with the real DEM elevation value. The efficiency of each interpolation method is compared by benchmarking the running time of different implementations developed in sequence, on a multicore CPU, on a single GPU, and on multiple GPUs.

RESULTS

Experimental Environment

To evaluate the computational performance of the presented various parallel interpolations, we conducted ten groups of experimental tests in both the flat zone and the rugged zone on a powerful workstation equipped with two Quadro M5000 GPUs. The specifications of the workstations are listed in Table 4.

Table 4. Specifications of the workstation and the software used for the experimental tests

Specifications	Details
CPU	Intel Xeon E5-2650 v3
CPU Frequency	2.30 GHz
CPU RAM	144 GB
CPU Core	40
GPU	Quadro M5000
GPU Memory	8 GB
GPU Core	2048
OS	Windows 7 Professional
Compiler	Visual Studio 2010
CUDA Version	v8.0

Test Results of Interpolation Accuracy for Different Interpolation Algorithms

In this paper, we adopt the Normalized Root-Mean-Square-Error (NRMSE) as the metric to measure the interpolation accuracy of the different interpolation algorithms. The NRMSE is defined in Eq. (20).

Normalized Root-Mean-Square-Error (NRMSE):

$$NRMSE = \frac{1}{\max\limits_{1 \le i \le N_i} |f_a|} \sqrt{\frac{1}{N_i} \sum_{i=1}^{N_i} |f_n - f_a|^2} \tag{20}$$

where N_i is the number of interpolated points, f_a is the theoretically exact solution of the ith interpolated point (the elevation of the DEM at this point), and f_n is the predicted value of the ith interpolated point.

The interpolation accuracy of the ten groups of experimental tests is listed in Table 5. The numerical value shown in Table 5 is NRMSE, which means that the smaller the numerical value, the higher the interpolation accuracy.

Table 5. Interpolation accuracy of the parallel interpolation algorithms implemented on a single GPU

Data set			Original MLS	Orthogonal MLS	Lancaster's MLS	kNN RBF	kNN AIDW	kNN IDW	kNN Shepard1	kNN Shepard2
Flat zone	Regularly-distributed	Size 1	7.49E-5	7.49E-5	7.50E-5	9.23E-5	1.06E-4	1.07E-4	1.05E-4	1.03E-4
		Size 2	6.25E-5	6.25E-5	6.03E-5	6.85E-5	7.92E-5	7.98E-5	7.81E-5	7.80E-5
		Size 3	5.52E-5	5.52E-5	5.23E-5	5.67E-5	6.17E-5	6.19E-5	6.15E-5	6.23E-5
		Size 4	5.16E-5	5.16E-5	4.88E-5	5.24E-5	5.45E-5	5.46E-5	5.47E-5	5.58E-5
		Size 5	4.91E-5	4.91E-5	4.64E-5	4.99E-5	5.05E-5	5.05E-5	5.08E-5	5.20E-5
	Irregularly-distributed	Size 1	1.96E-4	1.96E-4	1.86E-4	2.14E-4	1.90E-4	1.95E-4	1.98E-4	2.02E-4
		Size 2	1.53E-4	1.53E-4	1.48E-4	1.71E-4	1.57E-4	1.60E-4	1.62E-4	1.65E-4
		Size 3	1.20E-4	1.20E-4	1.15E-4	1.36E-4	1.28E-4	1.31E-4	1.32E-4	1.33E-4
		Size 4	1.07E-4	1.07E-4	1.02E-4	1.21E-4	1.15E-4	1.17E-4	1.18E-4	1.19E-4
		Size 5	9.50E-5	9.50E-5	9.14E-5	1.07E-4	1.05E-4	1.05E-4	1.06E-4	1.07E-4
Rugged zone	Regularly-distributed	Size 1	2.23E-4	2.23E-4	2.58E-4	4.41E-4	9.21E-4	9.26E-4	9.43E-4	9.69E-4
		Size 2	1.23E-4	1.23E-4	1.35E-4	2.35E-4	6.13E-4	6.16E-4	6.35E-4	6.63E-4
		Size 3	9.09E-5	9.09E-5	9.07E-5	1.37E-4	4.13E-4	4.12E-4	4.33E-4	4.58E-4
		Size 4	8.13E-5	8.13E-5	7.99E-5	1.08E-4	3.31E-4	3.30E-4	3.50E-4	3.71E-4
		Size 5	7.62E-5	7.62E-5	7.48E-5	9.39E-5	2.85E-4	2.83E-4	3.02E-4	3.21E-4
	Irregularly-distributed	Size 1	3.37E-3	3.37E-3	3.02E-3	3.99E-3	4.06E-3	4.12E-3	4.11E-3	4.07E-3
		Size 2	1.98E-3	1.98E-3	1.88E-3	2.96E-3	3.49E-3	3.55E-3	3.57E-3	3.52E-3
		Size 3	1.03E-3	1.03E-3	1.10E-3	1.56E-3	2.02E-3	2.05E-3	2.04E-3	2.02E-3
		Size 4	8.15E-4	8.15E-4	8.21E-4	1.16E-3	1.70E-3	1.70E-3	1.68E-3	1.67E-3
		Size 5	6.33E-4	6.33E-4	6.59E-4	9.78E-4	1.35E-3	1.36E-3	1.36E-3	1.37E-3

As listed in Table 5, the most accurate interpolation algorithm is the MLS interpolation algorithm. For the small size (Size 1), compared with other two algorithms, the MLS algorithm is 13.1%-49.4% more accurate than the RBF algorithm, and it is 2.1%-75.8% more accurate than the Shepard's algorithm. On the other hand, for the same algorithm, when the distribution pattern is the same, its accuracy in the flat area is higher than that the rugged area. For example, for the MLS algorithm, when the distribution pattern is nonuniformly distributed, the accuracy of the Lancaster' MLS algorithm in the flat area is approximately 90% higher than that of the Lancaster' MLS algorithm in the rugged area.

As shown in Figure 4 and Figure 5, the NRMSEs of various interpolation methods for the regularly distributed are less than 50% of the NRMSEs of various interpolation methods for the irregularly distributed. The above behavior becomes even more obvious in the rugged zone than in the flat zone. Thus, the regular distribution provides a more accurate solution for both the rugged and the flat areas.
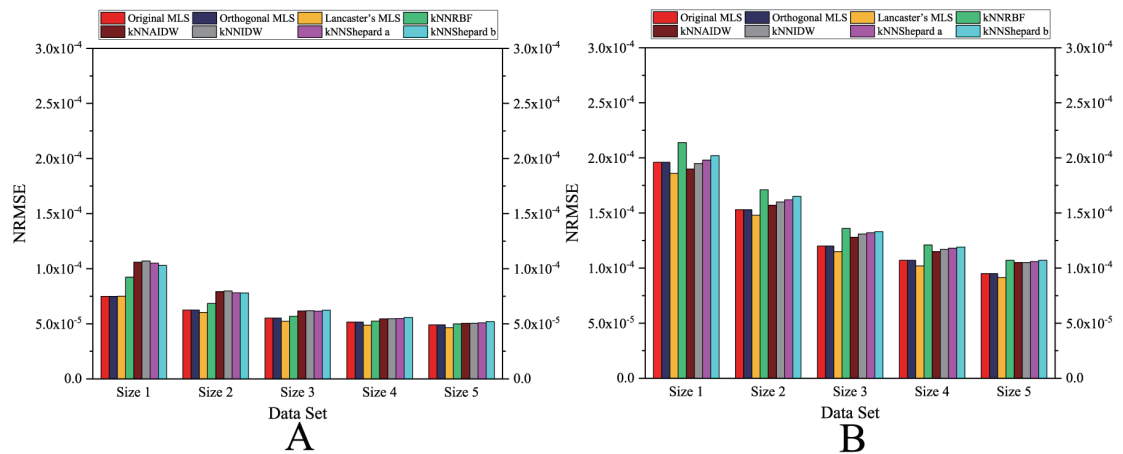
Figure 4. Interpolation accuracy of GPU-accelerated interpolation algorithms in the Flat zone. ((A) Regularly distributed and (B) irregularly distributed.)
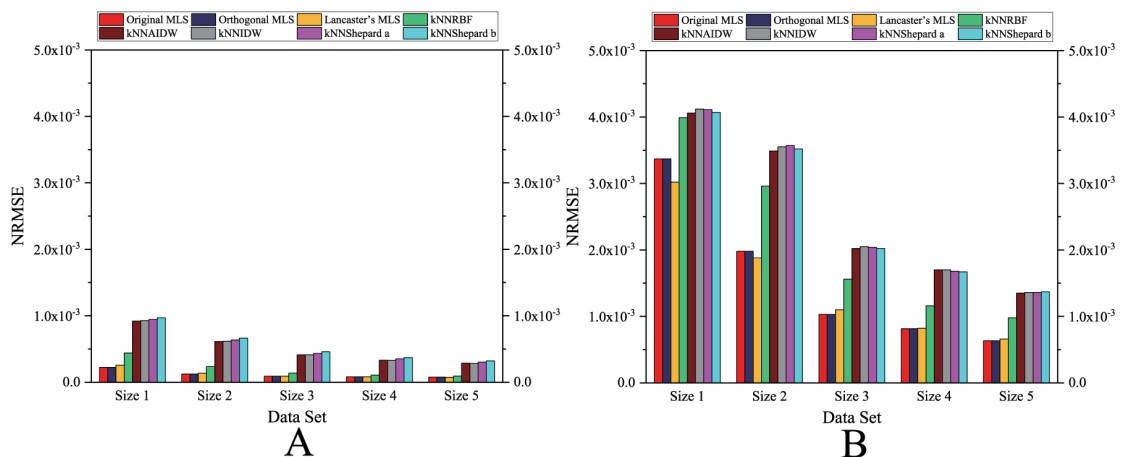
Figure 5. Interpolation accuracy of GPU-accelerated interpolation algorithms in the Rugged zone. ((A) Regularly distributed and (B) irregularly distributed.)

Test Results of Computational Efficiency for Different Interpolation Algorithms

In our experimental tests, the value of k is 20. Those twenty groups of experimental tests were performed on the workstations mentioned above. The running times and corresponding speedups of each group of experimental tests are presented in the following section. The speedup is defined in Eq. (21).

$$speedup = \frac{T_{seq}}{T_{par}} \tag{21}$$

where T_{seq} is the running time of sequential implementation, and T_{par} is the running time of parallel implementation.

Computational Efficiency of Sequential Implementations

As listed in Table 6, for the sequential version, when giving the same sets of data points and interpolation points, the order of computational time from fastest to slowest is: the Shepard's

Table 6. Running time (ms) of sequential implementations

Data set			Original MLS	Orthogonal MLS	Lancaster's MLS	*k*NN RBF	*k*NN AIDW	*k*NN IDW	*k*NN Shep-ard1	*k*NN Shep-ard2
Flat zone	Regularly-distributed	Size 1	1571.33	1501.67	1613.00	4194.33	1520.67	1239.00	1290.67	1270.33
		Size 2	3253.33	3238.33	3330.33	8547.33	3100.67	2475.67	2618.33	2583.00
		Size 3	6355.67	6063.33	6487.67	16610.67	6154.67	4957.33	5196.33	5125.33
		Size 4	9462.00	9036.67	9670.33	24856.67	9161.33	7359.00	7754.67	7674.00
		Size 5	12403.33	11854.00	12725.33	32370.33	12050.67	9643.33	10230.67	10058.00
	Irregularly-distributed	Size 1	1458.33	1392.00	1500.00	4028.67	1409.00	1104.33	1177.33	1157.67
		Size 2	3042.33	2919.67	3115.00	8291.33	2923.00	2300.33	2430.67	2397.33
		Size 3	6067.00	5738.00	6129.00	16299.33	5783.67	4559.00	4834.67	4776.33
		Size 4	8856.00	8491.33	9142.00	24286.00	8636.33	6779.33	7211.33	7105.00
		Size 5	11706.00	11214.00	12031.33	31744.00	11354.00	8922.00	9498.00	9372.67
Rugged zone	Regularly-distributed	Size 1	1576.00	1497.67	1605.33	4148.00	1512.67	1204.67	1278.00	1264.00
		Size 2	3211.33	3131.00	3285.33	8452.33	3117.33	2620.33	2695.33	2582.67
		Size 3	6354.33	6064.67	6500.33	16649.33	6139.67	4898.00	5200.33	5127.67
		Size 4	9444.67	9026.67	9662.33	24811.67	9187.00	7293.33	7710.33	7660.33
		Size 5	12416.67	11853.33	12711.33	32372.67	12008.33	9606.33	10205.67	10062.00
	Irregularly-distributed	Size 1	1503.00	1408.00	1516.00	4060.33	1424.00	1117.33	1191.67	1214.67
		Size 2	3032.33	2883.33	3110.33	8274.33	2925.67	2277.00	2424.00	2391.33
		Size 3	5943.33	5704.67	6089.33	16226.67	5746.33	4534.00	4800.33	4735.67
		Size 4	8920.00	8524.33	9132.33	24262.00	8654.67	6781.67	7224.00	7115.67
		Size 5	11632.33	11147.33	11925.33	31612.00	11282.33	8885.33	9435.67	9320.33

interpolation method, the MLS interpolation, and the RBF interpolation. The computational time of Shepard's interpolation method is approximately 20% less than the MLS interpolation method, and it is approximately 70% less than the computational time of the computational time of RBF interpolation method.

Computational Efficiency of Parallel Implementations

As shown in Figure 6-11, the parallel version developed on multi-GPUs has the highest speedup in the three parallel versions. Except for the RBF interpolation method, the maximum speedups of other interpolation algorithms are greater than 45.

As shown in Figures 12 and 13, for the parallel version developed on multi-GPUs, the order of the computational time from fastest to slowest is: the Shepard's interpolation, the MLS interpolation, the RBF interpolation method. The computational time of Shepard's interpolation method is 3%-30% less than the computational time of the MLS interpolation method, and it is 70%-85% less than the computational time of the RBF interpolation method.

DISCUSSION

The interpolation accuracy and computational efficiency are two critical issues that should be considered first in any interpolation algorithms. The interpolation accuracy should first be satisfied; otherwise, numerical analysis results would be inaccurate. In addition, the computational efficiency should be practical.

More specifically, in the subsequent section we will analyze (1) the interpolation accuracy of the presented eight GPU-accelerated interpolation algorithms with different data sets and (2) the computational efficiency of the presented eight interpolation algorithms.
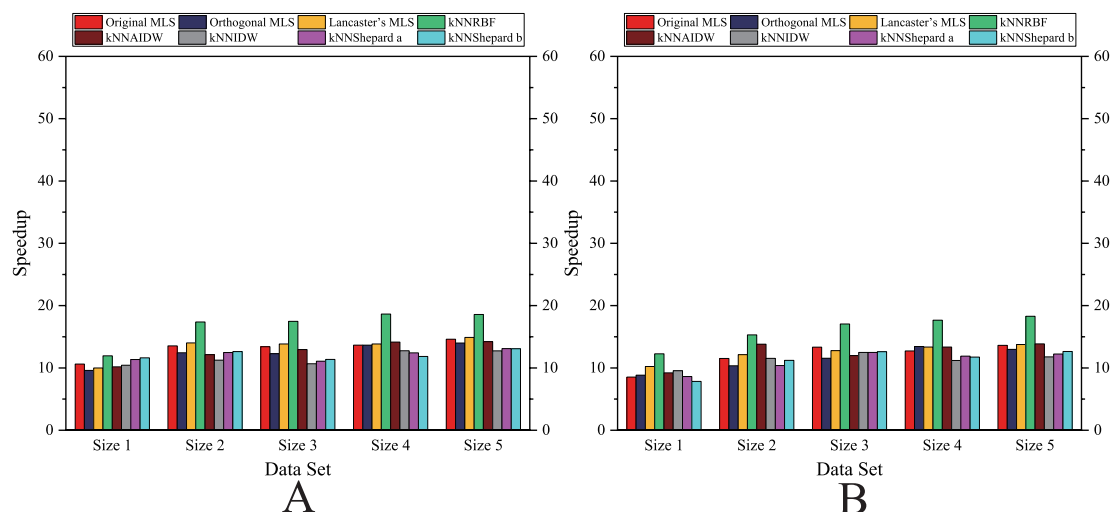
Figure 6. Comparison of the speedups of the parallel implementations developed on a multicore CPU in the Flat zone. ((A) Regularly distributed and (B) irregularly distributed.)
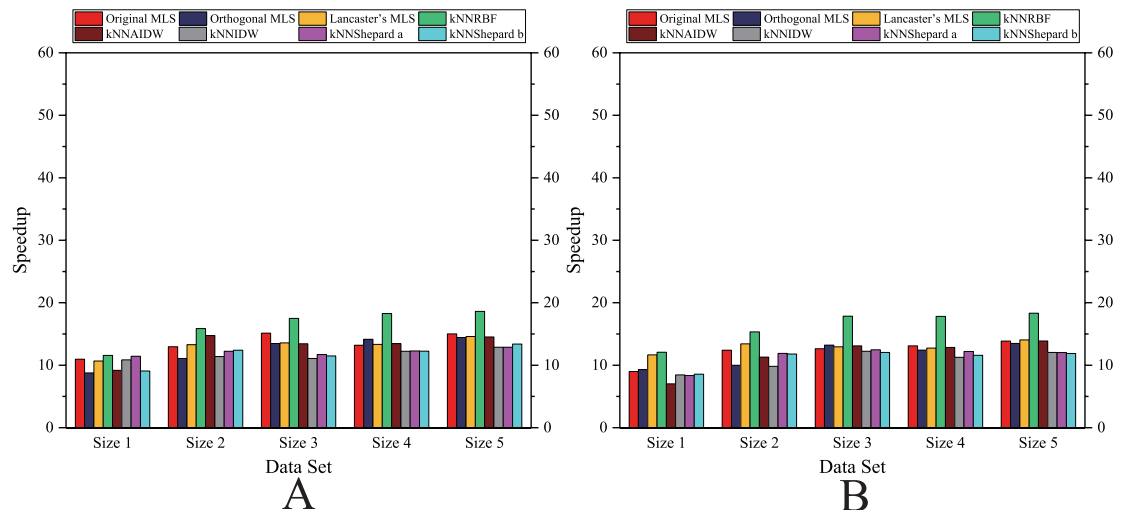
Figure 7. Comparison of the speedups of the parallel implementations developed on a multicore CPU in the Rugged zone. ((A) Regularly distributed and (B) irregularly distributed.)

Comparison of Interpolation Accuracy

To better compare the accuracy of the described interpolation algorithms, in the case of the highest sample density (Size 5) and the lowest sample density (Size 1), we listed those algorithms with the highest accuracy (i.e., the minimum NRMSE) in Table 7.

As listed in Table 7, for lower sample density (Size 1), the Original MLS algorithm has the best interpolation performance in regularly distributed. However, for higher sample density (Size 5), in general, the improved MLS algorithm Lancaster's MLS has higher interpolation accuracy than the Original MLS. In particular, the Original MLS has best accuracy in the rugged zone with irregularly distributed interpolation points.

On the other hand, for Shepard's interpolation algorithms, the *k*NNAIDW is an improved
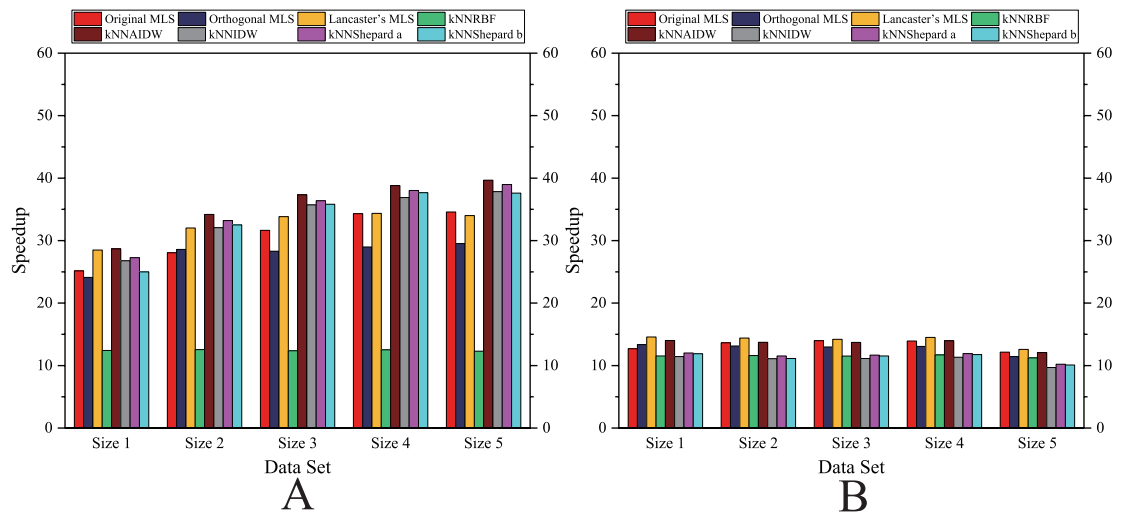
Figure 8. Comparison of the speedups of the parallel implementations developed on a single GPU in the Flat zone. ((A) Regularly distributed and (B) irregularly distributed.)
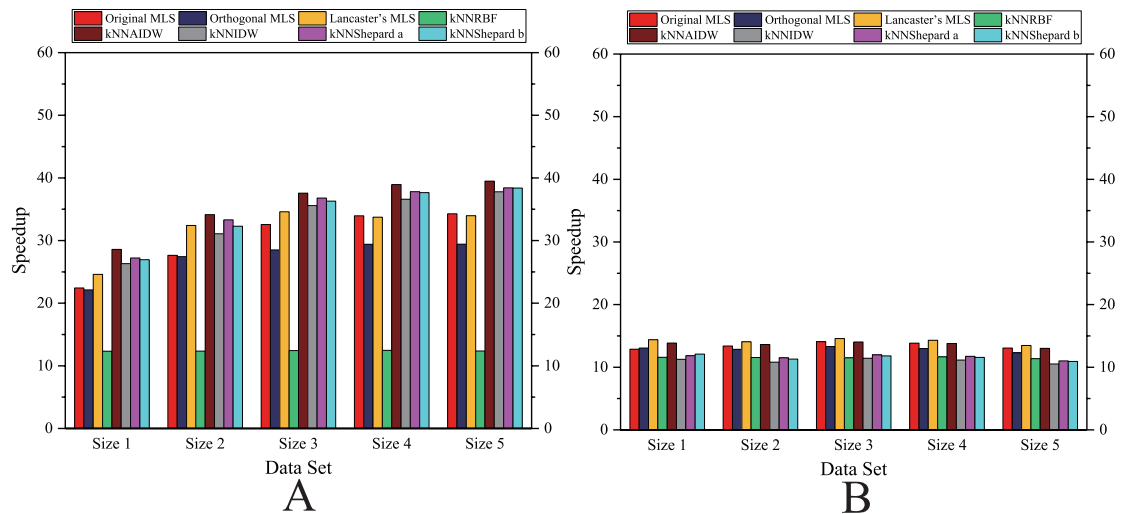
Figure 9. Comparison of the speedups of the parallel implementations developed on a single GPU in the Rugged zone. ((A) Regularly distributed and (B) irregularly distributed.)

version of the IDW, which can adaptively determine the power parameter according to the spatial points' distribution pattern. Therefore, in Shepard's interpolation algorithms, the *k*NNAIDW has higher accuracy in most situations. Although under some specific conditions, the *k*NNShepard2 and *k*NNIDW have higher accuracy than *k*NNAIDW, the accuracy of *k*NNAIDW is quite similar to them.

As listed Table 7. For the same flat zone, when the data points are uniformly distributed, the order of the interpolation accuracy from high to low is: the MLS interpolation algorithm, RBF, and Shepard's interpolation method; when the data points are normal distribution, the order of the interpolation accuracy from high to low is: the MLS interpolation algorithm, Shepard's interpolation method, and RBF. For the same rugged zone, regardless of the density and distribution of the data
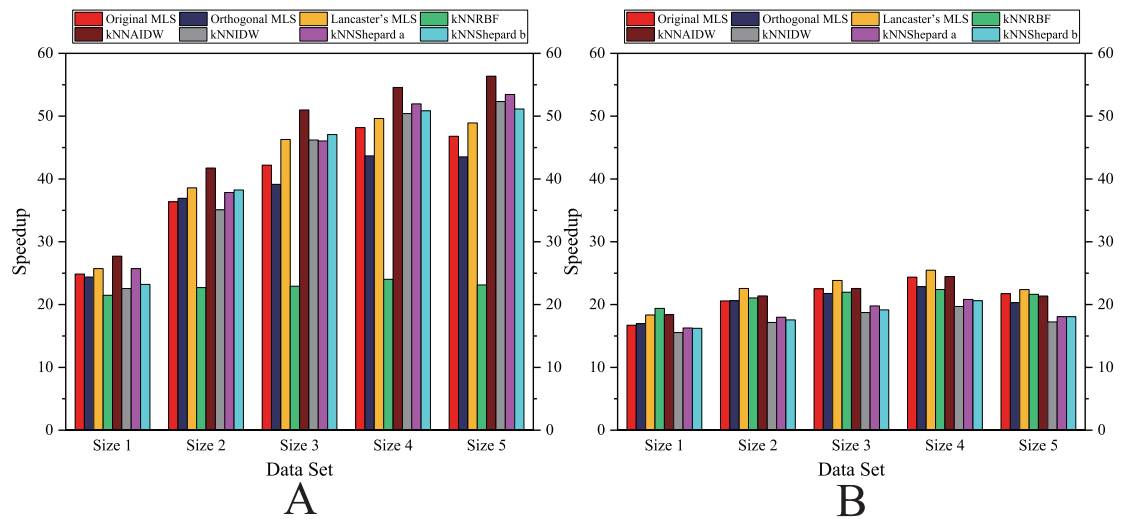
Figure 10. Comparison of the speedups of the parallel implementations developed on multi-GPUs in the Flat zone. ((A) Regularly distributed and (B) irregularly distributed.)
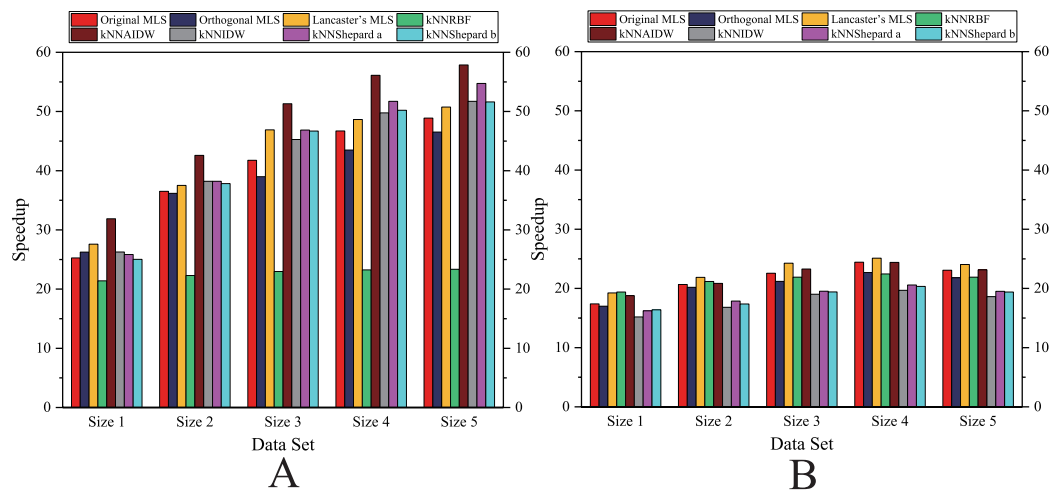
Figure 11. Comparison of the speedups of the parallel implementations developed on multi-GPUs in the Rugged zone. ((A) Regularly distributed and (B) irregularly distributed.)

373   points, the interpolation accuracy order from high to low is: the MLS interpolation algorithm, RBF,
374   and Shepard's interpolation method.

375   To further verify the above conclusions obtained from NRMSE, we investigated the relative
376   error of the interpolated results for the same set of data points and interpolation points (i.e., Size 1).
377   The algorithm with the highest accuracy (i.e., the minimum NRMSE) is used to represent the kind
378   of algorithm.

379   As shown in Figure 14 and Figure 15, the Y axis is the lgN (N is the count of relative error),
380   and the X axis is the relative error e. The e is defined in Eq. (22).

$$e_i = \left| \frac{f_n - f_a}{f_a} \right| \times 100\% \tag{22}$$
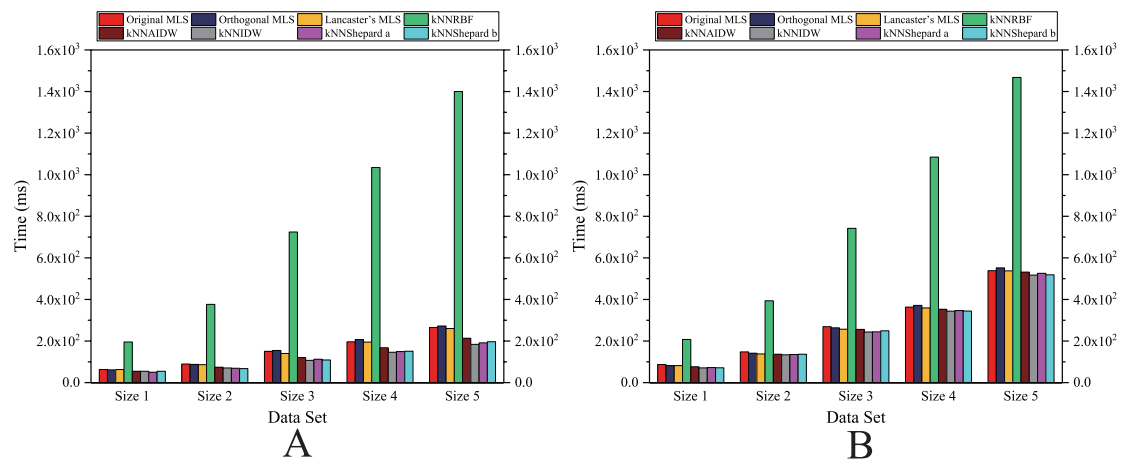
Figure 12. Comparison of the running time of the parallel implementations developed on multi-GPUs in the Flat zone. ((A) Regularly distributed and (B) irregularly distributed.)
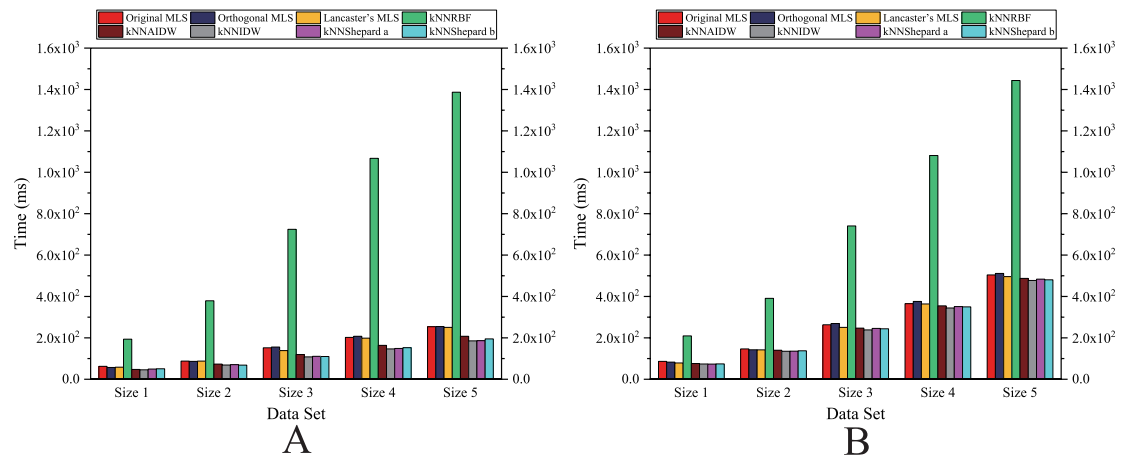
Figure 13. Comparison of the running time of the parallel implementations developed on multi-GPUs in the Rugged zone. ((A) Regularly distributed and (B) irregularly distributed.)

Table 7. The algorithm with the highest accuracy in congeneric algorithms and its corresponding NRMSE

Data set			MLS Algorithm	RBF Algorithm	Shepard's Interpolation Algorithm
Flat zone	Regularly-distributed	Size 1	Original MLS (7.49E-5)	kNNRBF (9.23E-5)	kNNShepard2 (1.03E-4)
		Size 5	Lancaster's MLS (4.64E-5)	kNNRBF (4.99E-5)	kNNAIDW (5.05E-5)
	Irregularly-distributed	Size 1	Lancaster's MLS (1.86E-4)	kNNRBF (2.14E-4)	kNNAIDW (1.90E-4)
		Size 5	Lancaster's MLS (9.14E-5)	kNNRBF (1.07E-4)	kNNAIDW (1.05E-4)
Rugged zone	Regularly-distributed	Size 1	Original MLS (2.23E-4)	kNNRBF (4.41E-4)	kNNAIDW (9.21E-4)
		Size 5	Lancaster's MLS (7.48E-5)	kNNRBF (9.39E-5)	kNNIDW (2.83E-4)
	Irregularly-distributed	Size 1	Lancaster's MLS (3.02E-3)	kNNRBF (3.99E-3)	kNNAIDW (4.06E-3)
		Size 5	Original MLS (6.33E-4)	kNNRBF (9.78E-4)	kNNAIDW (1.35E-3)

381   where f_a is the theoretically exact solution of the ith interpolated point (the elevation of the DEM

382   at this point), f_n is the predicted value of the ith interpolated point, and e_i is the relative error of the

383 ith interpolated point.

384   As listed in Table 8 and Table 9. For better evaluation of relative error, we also calculated the

385 mean relative error E. The E is defined in Eq. (23)

$$E = \frac{\sum\limits_{i=1}^{N_i} e_i}{N_i} \qquad (23)$$

386 where N_i is the number of interpolated points.

387 ***In the Flat Zone***
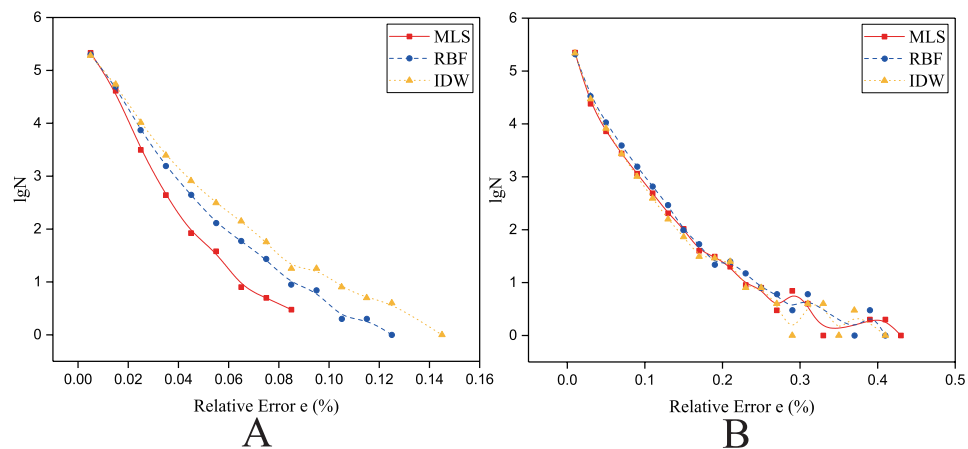
Figure 14. Frequency distribution of the Relative Error for the parallel implementation developed on a single GPU in the Flat zone. ((A) Regularly distributed and (B) irregularly distributed. The size of data points: Size 1.)

Table 8. The algorithm with the highest accuracy in congeneric algorithms and its corresponding mean relative error in the Flat zone

Distribution	Mean Relative Error E (%)		
Regularly-distributed	Original MLS	kNNRBF	kNNShepard b
	0.0069	0.0078	0.0084
Irregularly-distributed	Lancaster's MLS	kNNRBF	kNNAIDW
	0.0144	0.0162	0.0148

388   As shown in Figure 14, for the flat region, when the data points are evenly distributed, the

389 frequency statistical curve of the MLS is the highest when it is close to zero, the lowest when it is

390 far away from zero, and the relative error distribution range is smaller, which means that the error

391 of MLS method is small. The characteristics of the frequency statistical curve of Shepard's method

392 are completely opposite to those of MLS, which means that the error of MLS method is large. For

the RBF interpolation algorithm, the characteristic of the frequency statistics curve is a transitional phase between those for the MLS and those for Shepard's method. The above curve features and E (Table 8) illustrate that the interpolation accuracy is from high to low in this condition: the MLS interpolation algorithm, RBF, and Shepard's interpolation method.

When the data points are normally distributed, the relative error distribution ranges of all the interpolation methods are larger than that for the uniformly distributed data points. As shown in Figure 14, the characteristics of the frequency statistics curve of RBF are obvious, the frequency statistical curve of RBF is above the frequency statistical curves of MLS and Shepard's method, which means that the error of RBF method is larger. The characteristics of frequency statistical curves of MLS and Shepard's method are very similar, and the relative error distribution range of MLS is the largest. As listed in Table 8, in the flat zone, the accuracy of MLS is slightly higher than Shepard's method when the data points are normally distributed.

In the Rugged Zone
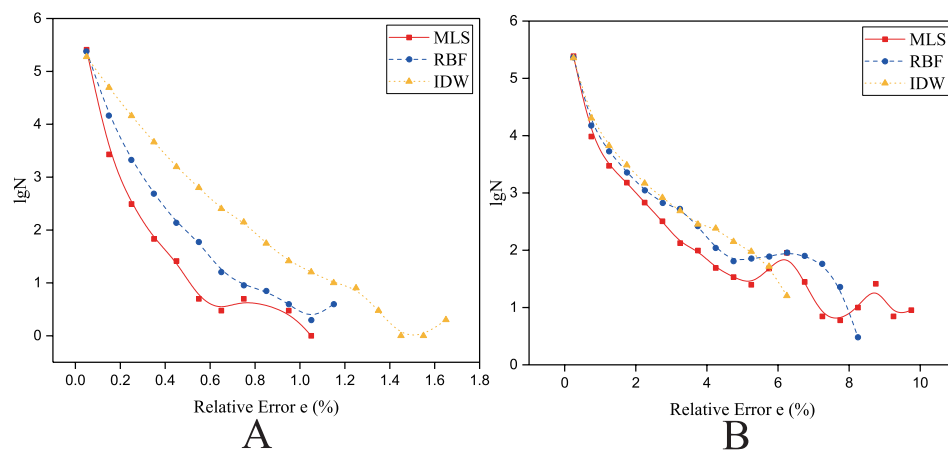
Figure 15. Frequency distribution of the Relative Error for the parallel implementation developed on a single GPU in the Rugged zone. ((A) Regularly distributed and (B) irregularly distributed. The size of data points: Size 1)

Table 9. The algorithm with the highest accuracy in congeneric algorithms and its corresponding mean relative error in the Rugged zone

Distribution	Mean Relative Error E (%)		
Regularly-distributed	Original MLS	kNNRBF	kNNAIDW
	0.0514	0.0582	0.0904
Irregularly-distributed	Lancaster's MLS	kNNRBF	kNNAIDW
	0.3078	0.3493	0.3703

As shown in Figure 15, for the rugged region, regardless of whether the data points are uni-

407 formly distributed or normally distributed, the characteristics of frequency statistical curves of MLS,

408 RBF and Shepard's method are similar to those illustrated in Figure 14. However, in Figure 15(B),

409 it is a little different in that most of the frequency statistical curve of Shepard's method is higher

410 than the RBF's. As listed in Table 9, the interpolation accuracy is from high to low: the MLS

411 interpolation algorithm, RBF, and Shepard's interpolation method.

412   According to the above Figures and Tables, some summary conclusions are obtained as fol-

413 lows:

414   For the same region, when the density of data points is almost the same, the interpolation

415 accuracy when the data points are evenly distributed is higher than the interpolation accuracy when

416 the data points are nonuniformly distributed.

417   As listed in Table 5 and Table 7, when the data points are evenly distributed, the gap of the ac-

418 curacy between the three variations of the MLS method, RBF, and Shepard's interpolation methods

419 increases with the decrease of point density.

420   As shown in Figure 14 and Figure 15, when the data points are nonuniformly distributed, the

421 maximum relative errors of MLS is larger than other algorithms', however, MLS method has lower

422 NRMSE and E. A small number of larger relative errors has little effect on the overall interpola-

423 tion accuracy. A large number of small and medium relative errors are the key to determine the

424 interpolation accuracy of the algorithm.

425   As listed in Table 5, compared with the uniform distribution, when the points are nonuniformly

426 distributed the difference in the accuracy of the interpolation algorithms is not as sensitive to the

427 changes of point density.

428   Compared with the three variations of the MLS method and the RBF method, Shepard's in-

429 terpolation method is quite suitable for cases where the data points have a smooth trend. When

430 interpolating for the data points with an undulating trend, the accuracy of Shepard's interpolation

431 method will be poor. When the density of data points is small, this rule becomes more obvious.

Comparison of Computational Efficiency

433   The parallel implementations developed on multi-GPUs is the most efficient, therefore, the

434 parallel implementations developed on multi-GPUs are discussed below.

In the Flat Zone

436   As illustrated in Figure 12, for the flat region, except for the kNNRBF, when the number of

437 data points is not much different, the nonuniformly distributed data point set requires significantly

438 more interpolation time than the uniformly distributed data point set, and with the increase of the

439 number of points, interpolation time does increase as well.

440   As illustrated in Figure 10, the speedups achieved by the RBF interpolation method is gener-

441 ally small, and its speedups are not much different in various cases. However, when the size of

442 data point set is Size 1 and the data point set is nonuniformly distributed, the speedup of the RBF

443 interpolation method is larger than other methods, which means that the benefits of parallelism are

444 lower in this case.

445   As indicated above, the distribution pattern of data points strongly influences the interpolation

446 efficiency.

In the Rugged Zone

447 ***In the Rugged Zone***

448   As illustrated in Figure 11 and Figure 13, the running time and the speedups in the rugged

449 region are almost the same as those in the flat region. In other words, the characteristics of the

450 terrain elevation of data points have a weak influence on computational efficiency.

451 **Influence of kNN Search on Computational Efficiency**

452   According to Section 3, in the interpolation procedure, the kNN search may affect the entire

453 computational efficiency of interpolation.

454   To specifically evaluate the influence of the kNN search on the computational efficiency of

455 the entire interpolation procedure, we investigated the computational cost of the kNN search for

456 relatively large numbers of data points, i.e., for the dataset of Size 5 (listed in Figure 16).

457   Note that we employ four sets of data points with Size 5, including (1) the set of uniformly

458 distributed data points and the set of nonuniformly distributed data points in the flat region and (2)

459 the set of uniformly distributed data points and the set of nonuniformly distributed data points in

460 the rugged region.
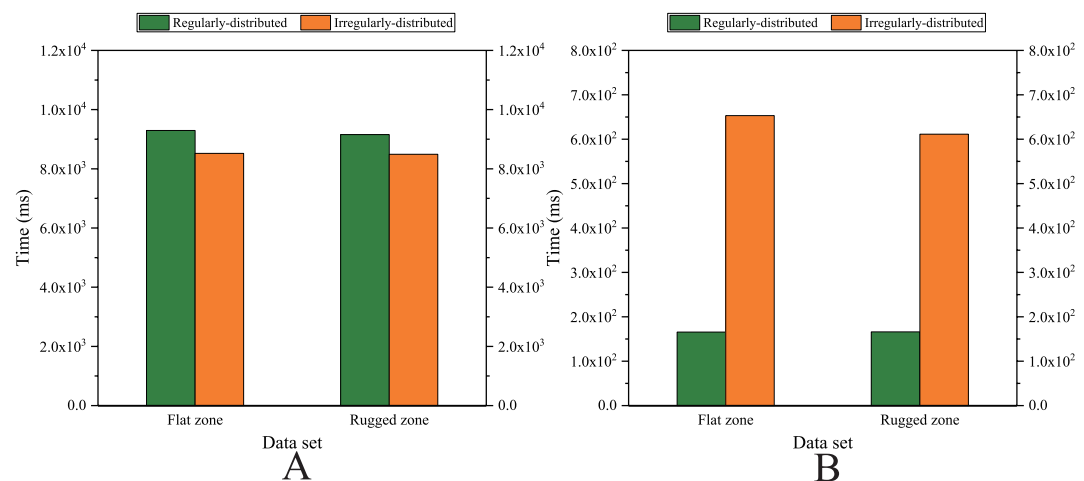
Figure 16. Comparison of the running time cost in the kNN search procedure. ((A) Sequential version on single CPU and (B) Parallel version on single GPU.)

461   As listed in Table 10, for the sequential version, regardless of whether the data points are

462 uniformly distributed or nonuniformly distributed, the kNN search costs approximately 75% of

463 the computational time of the entire interpolation procedure for the three variations of the MLS

464 interpolation algorithm and the AIDW interpolation algorithm, whereas the kNN search costs less

465 than 30% of the computational time for the RBF interpolation algorithm and approximately 90% in

466 the other three variations of Shepard's method. It should also be noted that for the same size of data

467 points, whether they are uniformly or nonuniformly distributed, there is no significant difference in

468 the computational cost of the kNN search; that is, the distribution pattern of data points is of weak

469 influence on the computational efficiency of the kNN search in the sequential version.

470   As listed in Table 11, for the parallel version developed on a single GPU, when the sizes of

471 data points are almost the same, it would cost much more time in the kNN search when the data

Table 10. Proportion of the kNN search time to the running time of the sequential implementations. (The proportion is $\frac{T_{kNN}}{T_{run}} \times 100\%$, where T_{kNN} is the kNN search time, and T_{run} is the running time of the corresponding sequential implementations.)

Data set		Original MLS	Orthogonal MLS	Lancaster's MLS	kNN RBF	kNN AIDW	kNN IDW	kNN Shep-ard1	kNN Shep-ard2
Flat zone	Regularly-distributed	74.9%	78.4%	73.0%	28.7%	77.1%	96.4%	90.8%	92.4%
	Irregularly-distributed	72.8%	76.0%	70.8%	26.8%	75.1%	95.5%	89.7%	90.9%
Rugged zone	Regularly-distributed	73.7%	77.2%	72.0%	28.3%	76.2%	95.3%	89.7%	91.0%
	Irregularly-distributed	73.0%	76.2%	71.2%	26.9%	75.3%	95.6%	90.0%	91.1%

472 points are nonuniformly distributed than when the data points are uniformly distributed. Moreover,
473 when the data points are nonuniformly distributed, the proportion of the kNN search time to the total
474 time is approximately 10% to 20% more than the proportion when the data points are uniformly
475 distributed under the same conditions.

Table 11. Proportion of the kNN search time to the running time of the parallel implementations developed on a single GPU. (The proportion is $\frac{T_{kNN}}{T_{run}} \times 100\%$, where T_{kNN} is the kNN search time, and T_{run} is the running time of the corresponding parallel implementations.)

Data set		Original MLS	Orthogonal MLS	Lancaster's MLS	kNN RBF	kNN AIDW	kNN IDW	kNN Shep-ard1	kNN Shep-ard2
Flat zone	Regularly-distributed	46.2%	41.3%	44.3%	6.3%	54.6%	65.0%	63.1%	62.0%
	Irregularly-distributed	67.8%	66.8%	68.3%	23.1%	69.5%	71.0%	70.3%	70.4%
Rugged zone	Regularly-distributed	45.8%	41.2%	44.4%	6.3%	54.6%	65.3%	62.5%	63.3%
	Irregularly-distributed	68.7%	67.4%	69.0%	22.0%	70.5%	72.3%	71.4%	71.7%

476 On the GPU, for the same interpolation method and the same data size, the proportion of the
477 kNN search time relative to the total time when the data points are nonuniformly distributed is larger
478 than that when the data points are uniformly distributed, and the achieved speedups are small.

479 However, on the CPU, the proportion of kNN search time when the data points are nonuni-
480 formly distributed relative to the total time is similar to that when the data points are uniformly
481 distributed, and the achieved speedups are similar. This is because there are a large number of
482 logical operations, such as switches in the kNN search, and the GPU is inherently not as suitable
483 for performing logical operations as the CPU.

484 In the kNN search procedure, the number of points in the search range is slightly smaller than
485 k after determining a certain level. After the level is expanded, the number of points in the search
486 range will be more than k. In this case, the k nearest neighbors should be selected and the redundant
487 neighbors should be ignored by first sorting and then discarding. Unfortunately, there are a large

488   number of logical operations in sorting.

489   In this procedure of sorting and discarding, when the point density is intensive in a region, the

490   number of found nearest neighbors would be far more than the expected k, and much computational

491   time would thus be required to sort the found neighbors.

492   For areas with sparse data points, it takes more time to find enough k points by expanding the

493   region level. Therefore, in contrast to a uniform distribution, when the data point set is nonuni-

494   formly distributed, the kNN search needs more computational time and its proportion of the total

495   time is also greater.

CONCLUSION

497   In this paper, we present the development of the sequential version, the parallel version on a

498   multicore CPU, the parallel version on a many-core GPU, and the parallel version on multi-GPUs

499   for each of the eight variations of the MLS, RBF, and Shepard's interpolation algorithms. We also

500   evaluated the interpolation accuracy and computational efficiency for the above four versions of

501   each variation when building large-scale DEMs. We have obtained the following observations.

502   (1) The distribution pattern of data points and the landscape conditions strongly influences the

503   interpolation accuracy. The distribution pattern of data points strongly influences the interpolation

504   efficiency, and the landscape conditions have a weak influence on the interpolation efficiency.

505   (2) For the same flat region, when the density of points is large, there is no obvious difference

506   in terms of the interpolation accuracy for all interpolation methods. When the data points are

507   uniformly distributed and the density of points is small, the order of the interpolation accuracy

508   from high to low is: the MLS interpolation algorithm, RBF, and Shepard's interpolation method.

509   When the data points are nonuniformly distributed and the density of points is small, the order

510   of the interpolation accuracy from high to low is: the MLS interpolation algorithm, Shepard's

511   interpolation method, and RBF.

512   (3) For the same rugged region, regardless of the density and distribution of the data points,

513   the interpolation accuracy order from high to low is: the MLS interpolation algorithm, RBF, and

514   Shepard's interpolation method. When the data points are uniformly distributed, the above rules

515   are more obvious than those when data points are nonuniformly distributed.

516   (4) The Shepard's interpolation method is only suitable for application in cases where the data

517   points have smooth trends. When the data points have uniformly rugged trends, the accuracy of

518   Shepard's interpolation method is rather unsatisfactory, especially in the case when the density of

519   data points is small.

520   (5) For the same set of data points and interpolation points, the order of computational expense

521   from high to low is: the RBF interpolation method, the MLS algorithm, and Shepard's method

522   Moreover, for the same size of data points and interpolation points, the computational efficiency in

523   the case when the data points are nonuniformly distributed is worse than when the data points are

524   uniformly distributed.

525   (6) For the same interpolation method, the impact of kNN search on the computational effi-

526   ciency of the CPU versions and the GPU versions is different. Specifically, the percentage of the

527 computational time of kNN search relative to the computational time of the entire interpolation
528 procedure in the CPU versions is much smaller than in the GPU versions.

529 ACKNOWLEDGMENTS

530 This research was supported by the Natural Science Foundation of China (Grant Numbers
531 11602235 and 41772326), and the Fundamental Research Funds for the Central Universities (Grant
532 Numbers 2652018097, 2652018107, and 2652018109). The authors would like to thank the editor
533 and reviewers for their contributions to the paper.

534 REFERENCES

535 Aguilar, F. J., Aguera, F., Aguilar, M. A., and Carvajal, F. (2005). Effects of terrain morphology,
536   sampling density, and interpolation methods on grid DEM accuracy. *Photogrammetric Engineer-*
537   *ing and Remote Sensing*, 71(7):805–816.

538 Box, G. E. P. and Muller, M. E. (1958). A Note on the Generation of Random Normal Deviates.
539   *Annals of Mathematical Statistics*, 29(2):610–611.

540 Chaplot, V., Darboux, F., Bourennane, H., Leguedois, S., Silvera, N., and Phachomphon, K. (2006).
541   Accuracy of interpolation techniques for the derivation of digital elevation models in relation to
542   landform types and data density. *Geomorphology*, 77(1-2):126–141.

543 Chen, Z., Shen, L., Zhao, Y., and Yang, C. (2010). Parallel algorithm for real-time contouring from
544   grid DEM on modern GPUs. *Science China-Technological Sciences*, 53:33–37.

545 Cheng, T. (2013). Accelerating universal Kriging interpolation algorithm using CUDA-enabled
546   GPU. *Computers & Geosciences*, 54:178–183.

547 Coll, N. and Guerrieri, M. (2017). Parallel constrained Delaunay triangulation on the GPU. *Inter-*
548   *national Journal of Geographical Information Science*, 31(7):1467–1484.

549 Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE*
550   *Transactions on*, 13(1):21–27.

551 Ding, Z., Mei, G., Cuomo, S., Tian, H., and Xu, N. (2018a). Accelerating multi-dimensional
552   interpolation using moving least-squares on the GPU. *Concurrency and Computation-Practice*
553   *& Experience*, 30(24).

554 Ding, Z., Mei, G., Cuomo, S., Xu, N., and Tian, H. (2018b). Performance Evaluation of GPU-
555   Accelerated Spatial Interpolation Using Radial Basis Functions for Building Explicit Surfaces.
556   *International Journal of Parallel Programming*, 46(5):963–991.

557 Ebdon, D. (1985). *Statistics in Geography*. Blackwell Publishing, Hoboken, 2nd edition edition.

558 Ghandehari, M., Buttenfield, B. P., and Farmer, C. J. Q. (2019). Comparing the accuracy of es-
559   timated terrain elevations across spatial resolution. *International Journal of Remote Sensing*,
560   40:5025–5049.

561 Guan, Q., Shi, X., Huang, M., and Lai, C. (2016). A hybrid parallel cellular automata model for
562   urban growth simulation over GPU/CPU heterogeneous architectures. *International Journal of*
563   *Geographical Information Science*, 30(3):494–514.

564 Gumus, K. and Sen, A. (2013). Comparison of spatial interpolation methods and multi-layer neu-

ral networks for different point distributions on a digital elevation model. *Geodetski Vestnik*, 57(3):523–543.

Khairnar, H. D., Shingare, P. S., Kale, S., and Ieee (2015). *Accuracy Evaluation of Cartosat-1 DEM using different Interpolation Techniques for Pune Area*. 2015 International Conference on Industrial Instrumentation and Control.

Krige, D. G. (1953). A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *OR*, 4(1):18–18.

Lancaster, P. and Salkauskas, K. (1981). Surfaces generated by moving least squares methods. *Math Compt*, 37(155):141–158.

Lehmer, D. H. (1949). Mathematical Methods in Large-Scale Computing Units. *Proc. of 2nd Symp. on Large-Scale Digital Calculating Machinery*, 26:141–146.

Lei, W., Xiong, R., Ma, S., Liang, L., and Ieee (2011). *GPU Based Fast Algorithm for Tanner Graph Based Image Interpolation*. IEEE International Workshop on Multimedia Signal Processing.

Lu, G. Y. and Wong, D. W. (2008). An adaptive inverse-distance weighting spatial interpolation technique. *Computers and Geosciences*, 34(9):1044–1055.

Mallet, J. L. (1997). Discrete modeling for natural objects. *Mathematical Geology*, 29(2):199–219.

Mei, G. (2014). Evaluating the Power of GPU Acceleration for IDW Interpolation Algorithm. *Scientific World Journal*.

Mei, G. and Tian, H. (2016). Impact of data layouts on the efficiency of GPU-accelerated IDW interpolation. *SpringerPlus*, 5(1):104.

Mei, G., Xu, L., and Xu, N. (2017). Accelerating adaptive inverse distance weighting interpolation algorithm on a graphics processing unit. *Royal Society Open Science*, 4(9).

Mei, G., Xu, N., and Xu, L. (2016). Improving GPU-accelerated adaptive IDW interpolation algorithm using fast kNN search. *Springerplus*, 5.

Peng, X., Wu, Q., Cai, Y., Lou, L., Yu, Y., and Li, Q. (2019). The application of radial basis function interpolation in reactor core power distribution on-line monitoring. *Annals of Nuclear Energy*, 132:752–762.

Polat, N., Uysal, M., and Toprak, A. S. (2015). An investigation of DEM generation process based on LiDAR data filtering, decimation, and interpolation methods for an urban area. *Measurement*, 75:50–56.

Powell, M. J. D. (1977). Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241–254.

Rishikeshan, C. A., Katiyar, S. K., Mahesh, V. N. V., and Ieee (2014). *Detailed evaluation of DEM interpolation methods in GIS using DGPS data, series = 2014 6th International Conference on Computational Intelligence and Communication Networks*.

Shepard, D. (1968). A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. *Proceedings of the 1968 ACM National Conference*, pp.:517–524.

Siu-Nganlam, N. (1983). Spatial interpolation methods: A review. *American Cartographer*, 10(2):129–150.

Steve, J. L. (2011). *Linear Algebra with Applications (8th Edition)*. Prentice Hall.

Tan, L., Wan, G., Li, F., Chen, X., and Du, W. (2017). GPU based contouring method on grid DEM data. *Computers & Geosciences*, 105:129–138.

Wang, H., Guan, X., and Wu, H. (2017). A Hybrid Parallel Spatial Interpolation Algorithm for Massive LiDAR Point Clouds on Heterogeneous CPU-GPU Systems. *Isprs International Journal of Geo-Information*, 6(11).

Wasza, J., Bauer, S., Hornegger, J., and Ieee (2011). *Real-time Preprocessing for Dense 3-D Range Imaging on the GPU: Defect Interpolation, Bilateral Temporal Averaging and Guided Filtering*. 2011 Ieee International Conference on Computer Vision Workshops.

Watson, D. F. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *Computer Journal*, 24(2):167–172.

Wu, J., Deng, L., and Jeon, G. (2018). Parallel constrained Delaunay triangulation on the GPU. *IEEE Transactions on Industrial Informatics*, 14:426–436.

Yan, C., Liu, J., Zhao, G., Chen, C., and Yue, T. (2016). A high accuracy surface modeling method based on GPU accelerated multi-grid method. *Transactions in Gis*, 20(6):991–1003.

Yan, C., Zhao, G., Yue, T., Chen, C., Liu, J., Li, H., and Su, N. (2015). Speeding up the high-accuracy surface modelling method with GPU. *Environmental Earth Sciences*, 74(8):6511–6523.

Yin, K., Sun, F., Zhou, S., and Zhang, C. (2014). PAR Model SAR Image Interpolation Algorithm on GPU with CUDA. *Iete Technical Review*, 31(4):297–306.

Zhang, G., Zhu, A., and Huang, Q. (2017). A GPU-accelerated adaptive kernel density estimation approach for efficient point pattern analysis on spatial big data. *International Journal of Geographical Information Science*, 31(10):1–30.

Zhang, J. (2013). *Research on DEM Interpolation Algorithm Adaptability with Local Terrain Features*. International Conference on Geoinformatics.

Zhou, G., Liu, X., Fu, S., and Sun, Z. (2017). Parallel identification and filling of depressions in raster digital elevation models. *International Journal of Geographical Information Science*, 31(6):1061–1078.

Figure 1

An illustration of the process of the kNN search algorithm

Figure 2

The selected Zone 1 and Zone 2.

(A) 2.5D model of the Zone 1 study area and (B) 2.5D model of the Zone 2 study area.

Figure 3

The distribution patterns determined by the Average Nearest Neighbor analysis.

(A) Regularly distributed and (B) irregularly distributed.

Figure 4

Interpolation accuracy of GPU-accelerated interpolation algorithms in the Flat zone.

(A) Regularly distributed and (B) irregularly distributed.

Figure 5

Interpolation accuracy of GPU-accelerated interpolation algorithms in the Rugged zone.

(A) Regularly distributed and (B) irregularly distributed.
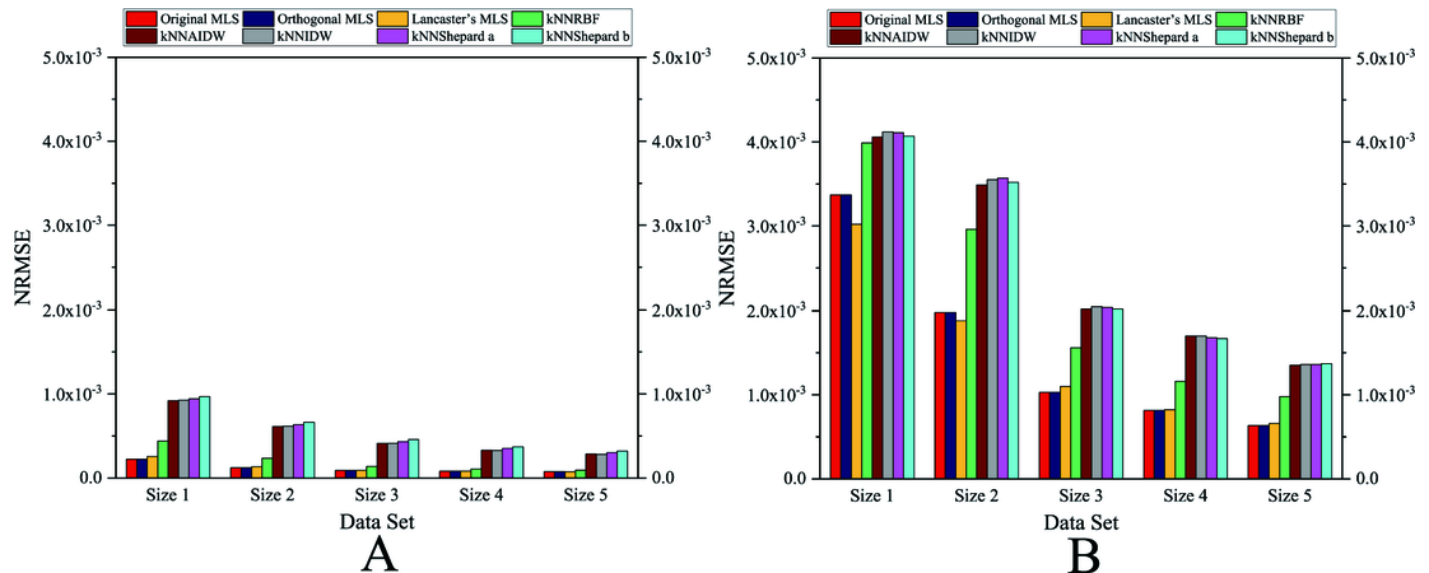
Figure 6

Comparison of the speedups of the parallel implementations developed on a multicore CPU in the Flat zone.

(A) Regularly distributed and (B) irregularly distributed.
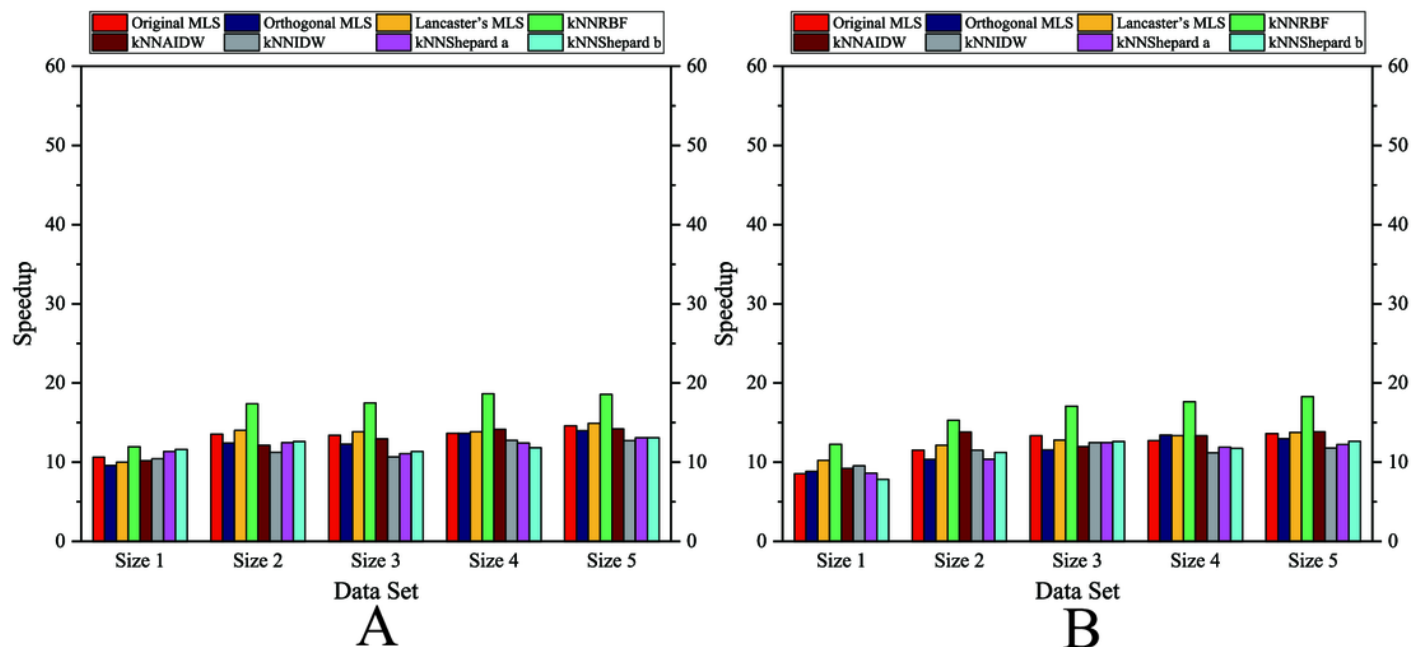
Figure 7

Comparison of the speedups of the parallel implementations developed on a multicore CPU in the Rugged zone.

(A) Regularly distributed and (B) irregularly distributed.
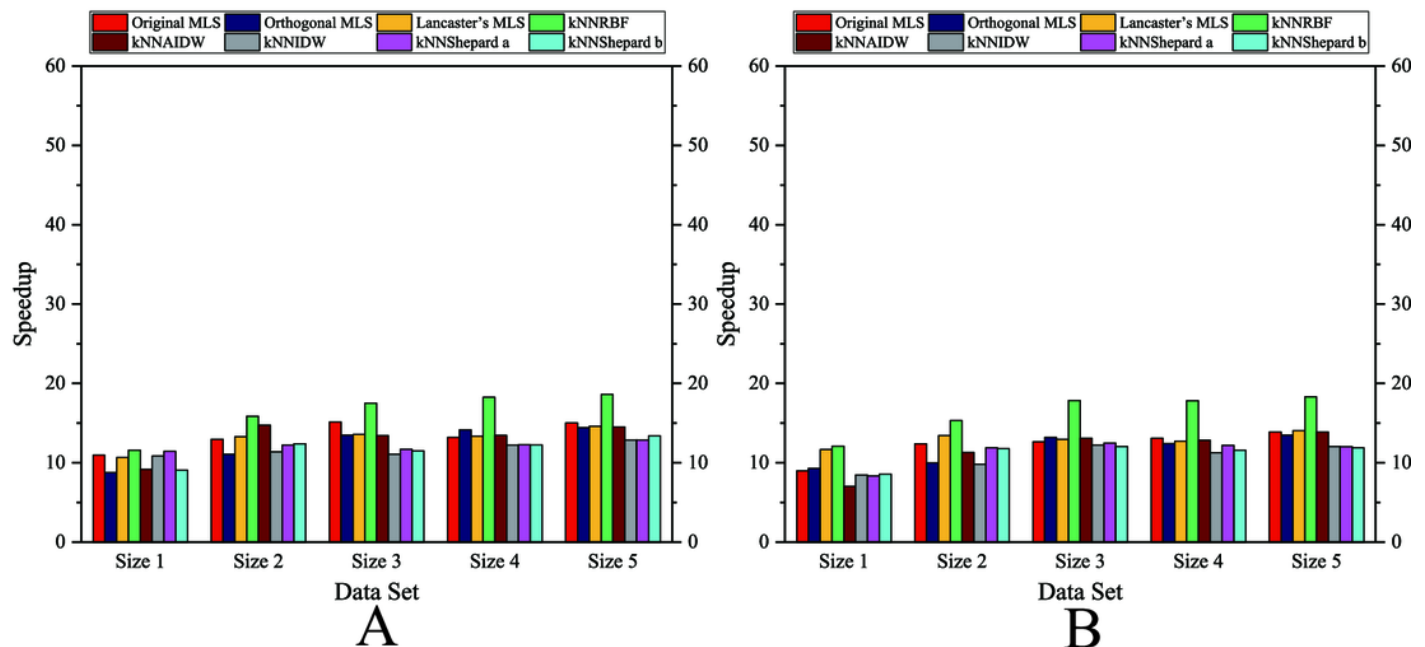
Figure 8

Comparison of the speedups of the parallel implementations developed on a single GPU in the Flat zone.

(A) Regularly distributed and (B) irregularly distributed.
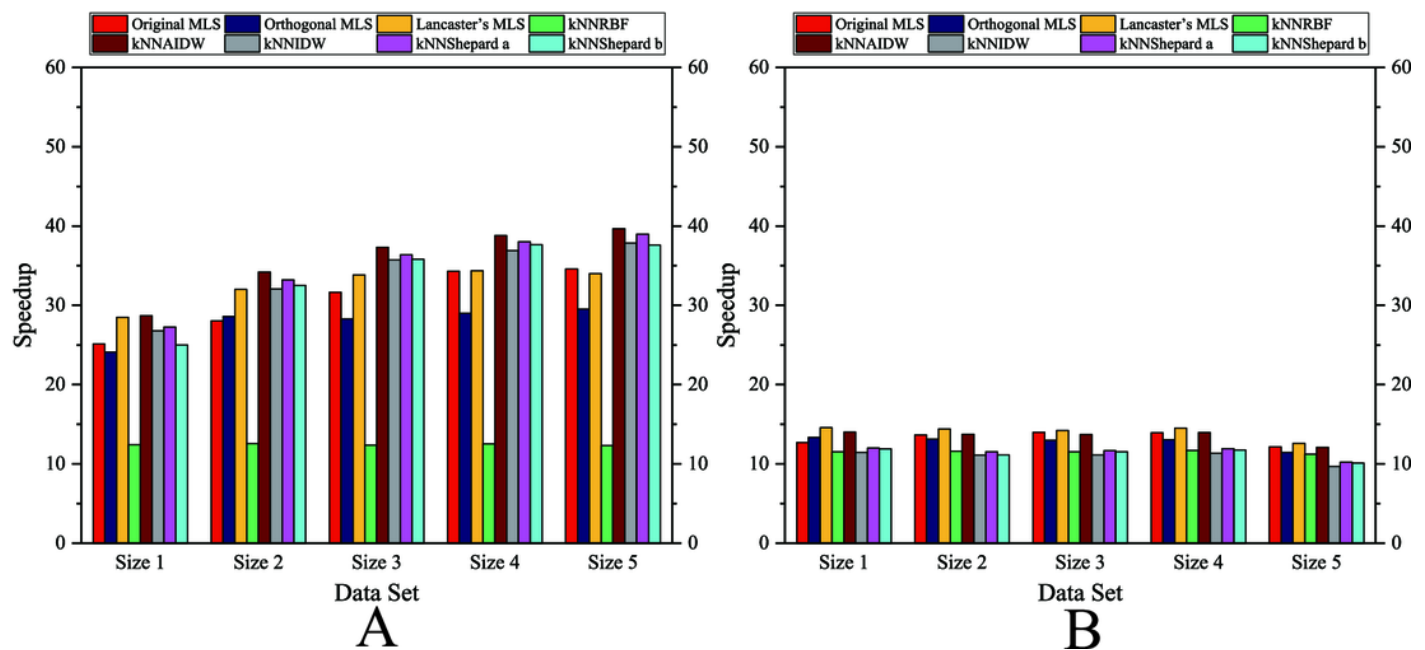
Figure 9

Comparison of the speedups of the parallel implementations developed on a single GPU in the Rugged zone.

(A) Regularly distributed and (B) irregularly distributed.
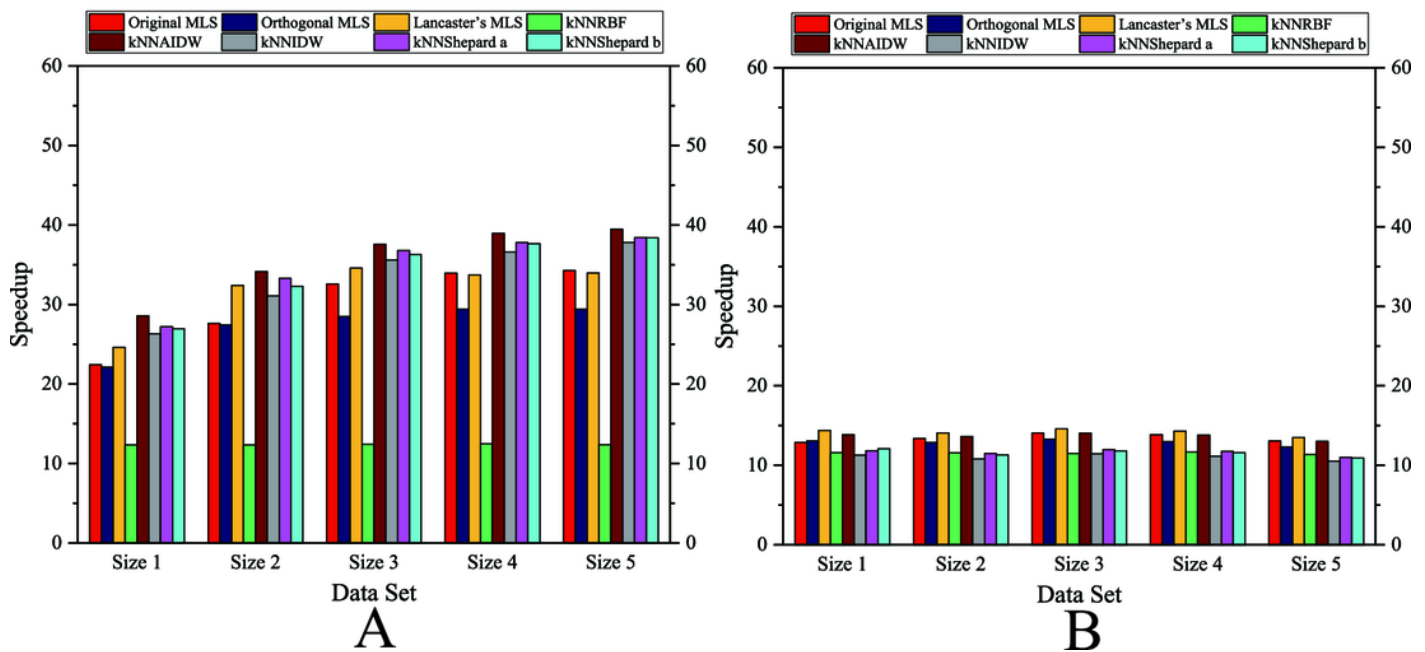
Figure 10

Comparison of the speedups of the parallel implementations developed on multi-GPUs in the Flat zone.

(A) Regularly distributed and (B) irregularly distributed.
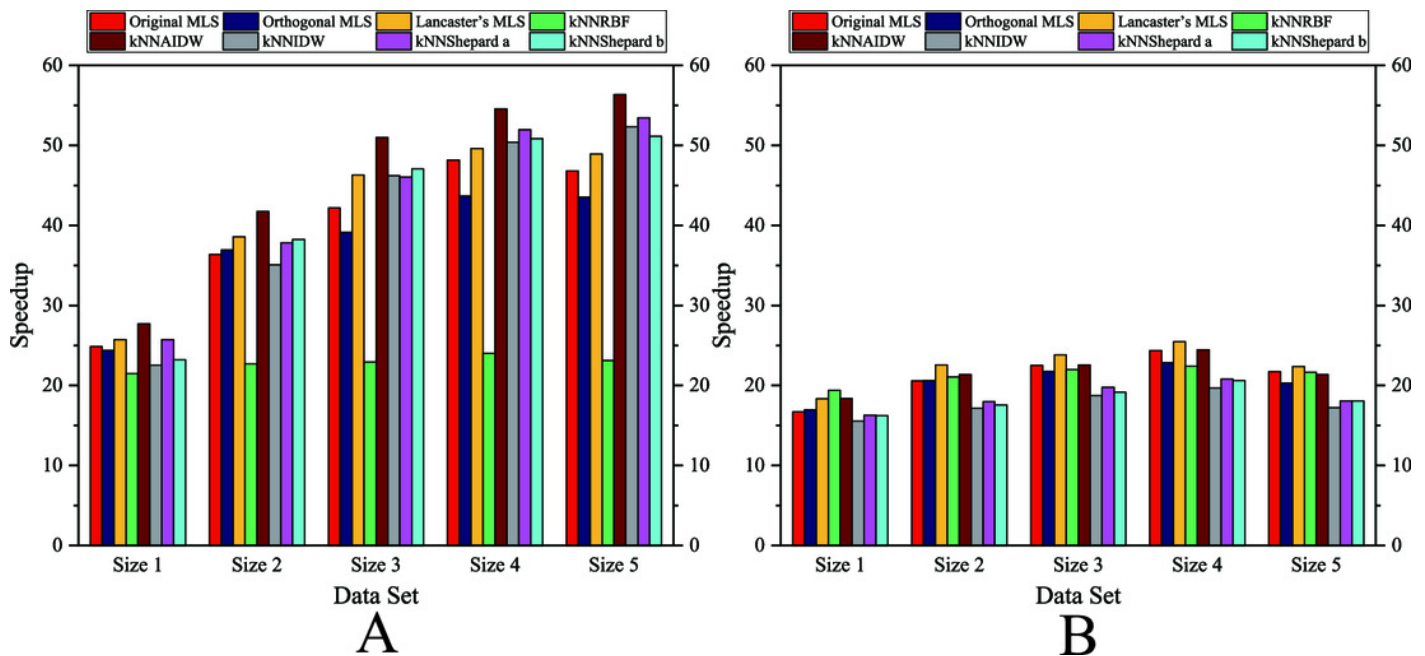
Figure 11

Comparison of the speedups of the parallel implementations developed on multi-GPUs in the Rugged zone.

(A) Regularly distributed and (B) irregularly distributed.
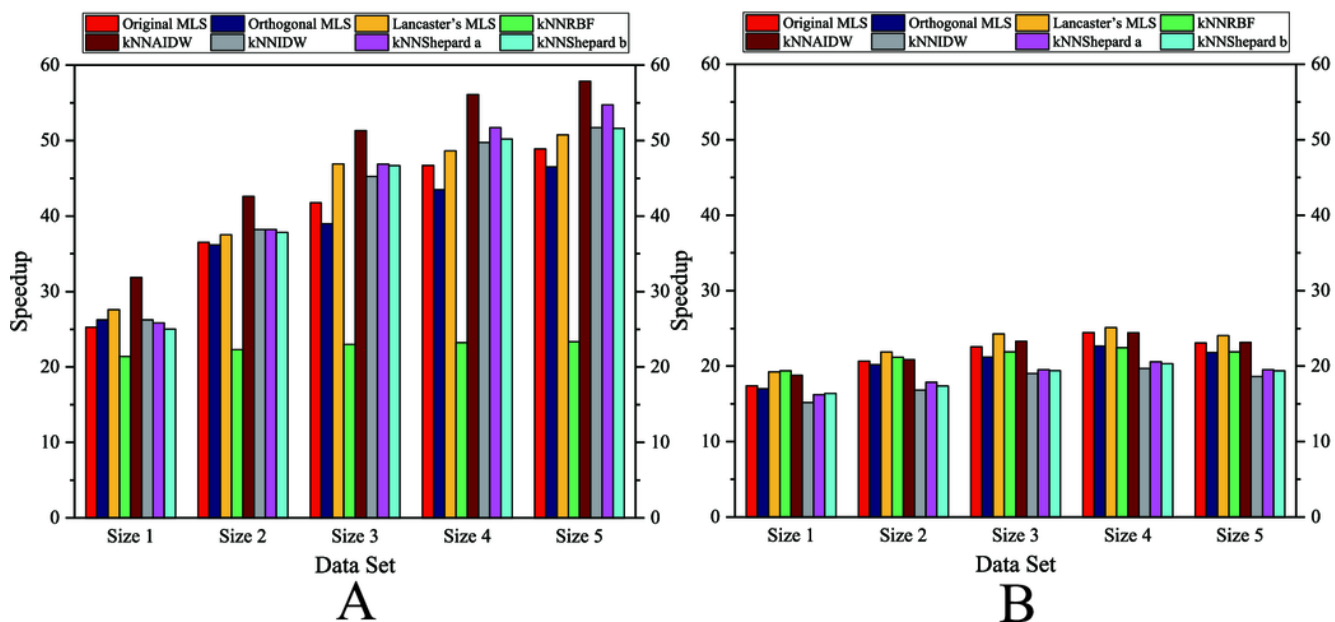
Figure 12

Comparison of the running time of the parallel implementations developed on multi-GPUs in the Flat zone.

(A) Regularly distributed and (B) irregularly distributed.
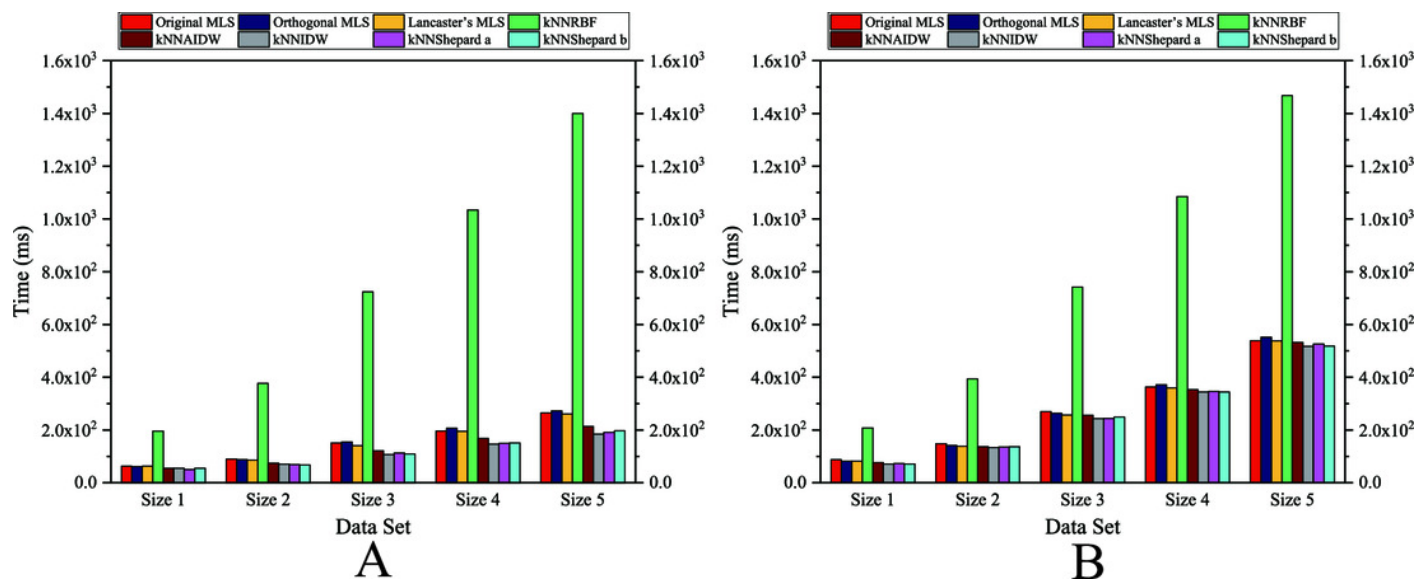
Figure 13

Comparison of the running time of the parallel implementations developed on multi-GPUs in the Rugged zone.

(A) Regularly distributed and (B) irregularly distributed.
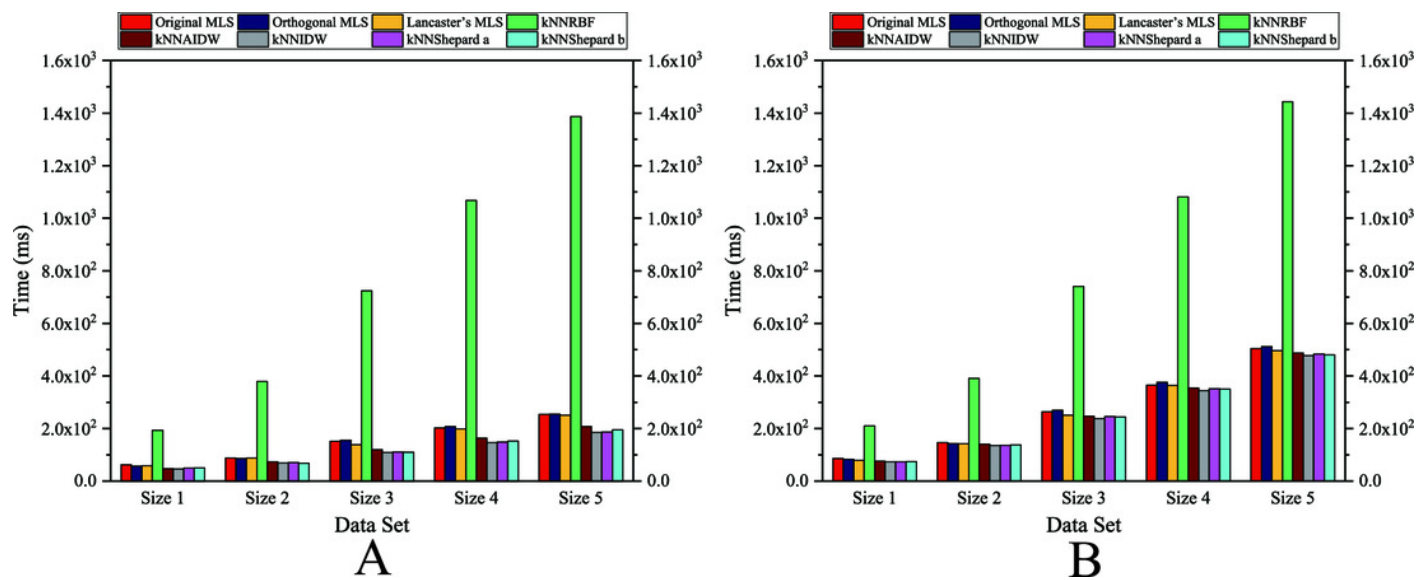
Figure 14

Frequency distribution of the Relative Error for the parallel implementation developed on a single GPU in the Flat zone.

(A) Regularly distributed and (B) irregularly distributed. The size of data points: Size 1.
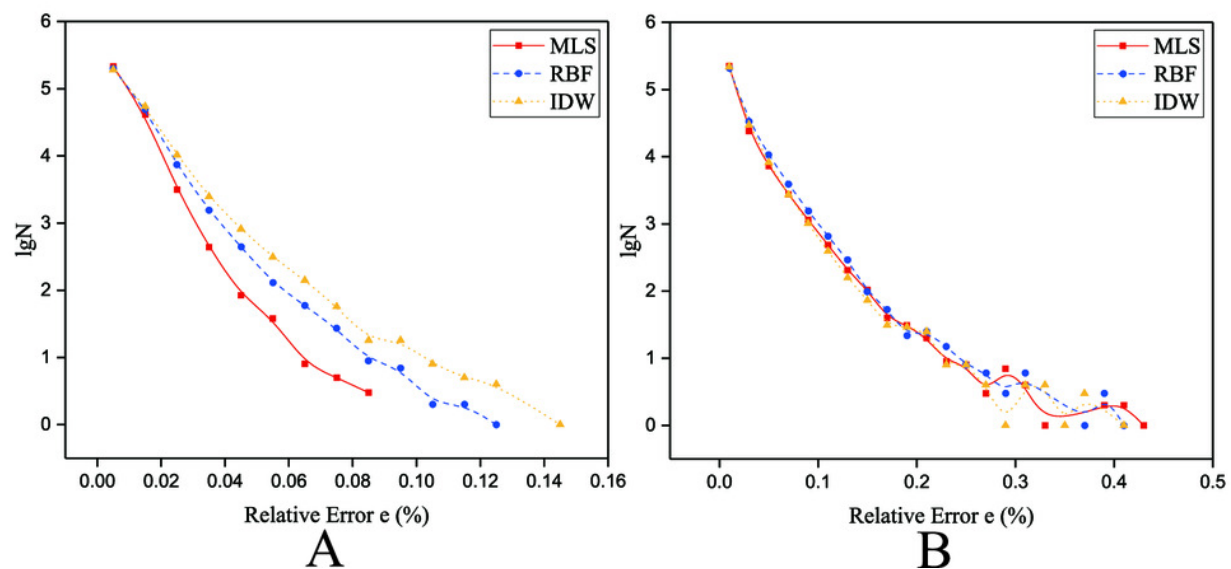
Figure 15

Frequency distribution of the Relative Error for the parallel implementation developed on a single GPU in the Rugged zone.

(A) Regularly distributed and (B) irregularly distributed. The size of data points: Size 1

Figure 16

Comparison of the running time cost in the *k*NN search procedure.

(A) Sequential version on single CPU and (B) Parallel version on single GPU.