IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 27, 2020, accepted August 10, 2020, date of publication August 19, 2020, date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017954

FastIsomapVis: A Novel Approach for Nonlinear Manifold Learning

MAHWISH YOUSAF, TANZEEL U REHMAN, DONGLIANG LIAO, NAJI ALHUSAINI, AND LI JING

School of Computer Science and Technology, University of Science and Technology of China, Hefei 230056, China

Corresponding author: Li Jing (lj@ustc.edu.cn)

This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (A Class) under Grant XDA19020102.

ABSTRACT The classical Isomap is the most common unsupervised nonlinear manifold method and widely being used in visualizations and dimension reductions. However, when it applied to real-world datasets, it shows shortcomings for the shortest path between all pairs of data points, which are based on the nearest neighborhood G graph via the Dijkstra algorithm, which makes it a very time-consuming step. The other critical problem is the classical Isomap has a lack of topological stability on the nearest neighborhood G graph. In this paper, we propose a novel technique called the FastIsomapVis for the above problems of the classical Isomap. The FastIsomapVis uses hierarchal divide, conquer, and combine approach through two algorithms, which are randomized division tree (KD-tree) and Dijkstra Buckets Double (DKD). The primary aim of the FastIsomapVis is to increase the efficiency and accuracy of the graph. This research paper focuses on transforming the high dimensional datasets into a low dimensional Isomap visualization. The FastIsomapVis makes it easy to construct an accurate K nearest neighborhood G graph and scale high dimensional data points into low dimensional space. Our proposed method is compared to the classical Isomap to verify its effectiveness and provide highly authentic results of the high dimensional datasets. The finding of the current study shows that our proposed method is much fastened than classical Isomap.

INDEX TERMS FastIsomapVis, manifold learning, Isomap, dimensionality reduction, visualization.

I. INTRODUCTION

Information visualization is an emerging research area, which helps users in sense-making, investigation, and representation from analysis results and original data [1]. Traditional visualization techniques have proved their effectiveness for small and intermediate-sized data, but their application to big data is challenging. For example, the visualization of scatter plots, network diagrams, and heat maps requires the data to be placed out. The information becomes computationally unmanageable when it has numerous dimensions, and many data points are large. In many fields of research, such as data mining, machine learning, computer vision, information visualization [2]–[4], and the information visualization community [5]–[7], there is an urgent requirement to discover a representation of low-dimensional data to high dimensional data [8].

In the process of dimensionality reduction, the most critical point is not to affect the inherent framework of the large, high-dimensional data; for instance, it is important to keep dissimilar datasets far apart and similar datasets closer to each other, in the low dimensional space [9]. The machine learning technique provides dimensionality reduction, accompanied by minimal information loss.

Dimensionality reduction techniques consist of linear and nonlinear methods. The linear techniques can execute the high dimensional data point transformation into a low dimensional data point as a linear variable. The nonlinear methods can only use the nonlinear variables, with the representation of low dimensional and high dimensional data points, and get the calculating distance between the original nonlinear data points [8], [10].

In recent years, much more attention has been given to the great importance of studying nonlinear manifold learning dimension reduction and visualization [11], [12]. Manifold learning is a technique of nonlinear dimension reduction. Manifold learning provides the methods for nonlinear methods, which is based on the several information theory of the dimensionality sets [8].

The implementation of linear methods is usually incomplete because the information of high dimensional mostly lies

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

on or near a nonlinear low-dimensional manifold [4], [9]. While the implementation of nonlinear methods is perfect mainly for local and global structures of high dimensional datasets, for that reason, we have chosen a nonlinear Classical Isomap method [13], which is the most popular technique of nonlinear manifold learning method. The basic idea of classical Isomap is a variant of Multidimensional Scaling (MDS) metric and maps the high dimensional data into low dimensional space. MDS calculates approximate geodesic distances between the data points instead of Euclidean distances (lines 1-4). Given the N data points and D distance matrix, classical Isomap executes the representation of the shortest eigenvalues of the distance D matrix. The geodesic distances between neighboring points are approximated (lines 3-6 in the algorithm below) by input-space distance. The geodesic distances for faraway points [14] are calculated as the shortest paths in a neighborhood G graph (line 7). The resulting geodesic distances are contained in the feature matrix F, which is processed by MDS to transform into low dimensional space Y [15]. The Isomap time complexity is based on the iterative process. Therefore, the time complexity of Isomap is $O(N^3)$, where N is the number of data points. The basic Isomap algorithm is given below:

Algorithm 1: Classical Isomap

Data: ***K*:** number of nearest neighbor, and the data points ***N***
Result: ***Y*** (data transformed into low dimensional space)
D_{nxn} ←Pairwise Distance (***N***)
$G_{nxn} \leftarrow \infty$
for ($x_i \in A$) **do**
  KNN ← KNN ($\mathbf{x_i}$, ***N***, ***K***)
  for ($x_j \in K$NN) **do**
    $G_{i,j} \leftarrow D_{i,j}$
  end
  F_{nxn} ←All PairsShortestPaths(***G***)
  Y ←MDS(**F**)
end
return Y

In our motive, we have attended the problem of the classical Isomap algorithm. The main problem in the classical Isomap algorithm is the lack of topological stability in the neighborhood G graph [16], [17]. Choi and Choi [18] proposed the method for outliers, noises, and a lack of topological stability problems, such as the Robust Kernel Isomap method. They reduced the effects of outliers based on the topological structure with the help of network flow. They also proposed the kernel Isomap algorithm for lack of topological stability issues. This method can maintain the topological stability while addressing the noises and outliers. Moreover, Yousaf*et al.* [8] proposed the FastIsomap method for high computational cost and topological stability in the G neighborhood graph problems. This method used two algorithms, such as KD-tree and NN-Descent, and reduced the high computational cost of the KNN graph.

For the shortest path problem, Hong-Yuan *et al.* [19] proposed the Multi-Class Multi-Manifold-Isomap (MCMM-ISOMAP) and Isomap for classification (ISOMAP-C). The MCMM-ISOMAP and ISOMAP-C and an improved Isomap method based on the Dijkstra algorithm with method Fibonacci Heap (Fib-Dij) have been proposed to overcome the shortest path problem [14], [20], e.g., by removing the nearest neighboring graphs G of the neighborhood graph or by removing large data points in the shortest path algorithm [14], [18]–[20], which can severely affect the classical Isomap graph [21]. Lei *et al.* [22] proposed a greedy approximated algorithm of Minimum Set Coverage (MSC) [23] for the shortest path and MDS problems. This method uses a few subsets of overlapping neighborhoods for high and large dimensional datasets that closely lies in a low dimensional space.

Moreover, a critical flaw of the classical Isomap algorithm is that it is very slow due to the dependency on the shortest path algorithm. This means that such a classical Isomap technique can rarely be applied to datasets with millions of data points and hundreds of dimensions. Thus, the classical Isomap algorithm is only used for global datasets and still far from generating authentic results of high dimensional datasets.

In this paper, we proposed a novel approach, known as the FastIsomapVis method, to deal with the above issues. The FastIsomapVis method is based on the divide, conquer, and combined approach. In the 'divide' step, we use the randomized KD-tree building algorithm to handle incorrect-link problems. In the 'conquer' step, we use the DKD algorithm to solve the shortest path problem. In the 'combine' step, we have to calculate the overall accuracy of the FastIsomapVis method by using the KD-tree search method. The FastIsomapVis method not only use an effective algorithm to K nearest neighbor graph (KNN) construction at high accuracy, but it also use Randomized KD-tree method [24] to merge the subgraph data and purify the data graph with the DKD technique.

A detailed description of the techniques and algorithms is given in section 3. The FastIsomapVis method maintains the shape of the graph in the low dimensional space and also keeps dissimilar data points far away, and similar data points close to each other. The FastIsomapVis method is also much faster than classical Isomap.

Moreover, the FastIsomapVis method produces accurate results for millions of data points, on hundreds of dimensions, while reducing the time complexity of the data points. Although an approximated KNN graph can be efficiently built, there is no research data available on how the performance of graph-based search techniques will be affected if we use an approximated KNN graph instead of an exact KNN graph and DKD instead of the Dijkstra algorithm. In recent years, no such type of method has been used for the classical Isomap algorithm.

The rest of the paper is organized as follows. In Section 2, we give brief details of the classical Isomap, an approximation to the nearest neighbor search (ANN), and K nearest neighbor (KNN). In Section 3, we propose the FastIsomapVis method for large-scale and high-dimensional datasets with divide, conquer, and combined approach. In Section 4, we describe the experimental results on two large-scale datasets. Finally, in section 5, the paper is concluded.

II. RELATED WORK

There are limited visualization techniques that can be used meaningfully for millions of high-dimensional data points in the 2D domain. In most visualizations of large datasets, we have to draft an overview or make a rough collection of the data and time-polish the subsection of the datasets if the user zooms in [25]. Visualization is the primary source of exploratory graph analysis (EGA). It includes the development of suitable types of visual representations (like node-link diagrams and matrixes), which provide useful visual attribute mapping and well-organized placement of the graphical element on the screen [2]–[4]. Available visualization tools are designed for geography, network, and sensor data: it is questionable whether they can deal with large-scaled dimensional data. In recent years, machine learning has become capable of handling the problems of visualization of high-dimensional data. Our work follows the strategies of the LargeVis method, in which we first construct a KNN graph and then visualize this graph in a 2D space [9].

A. DIMENSION REDUCTION

The main emphasis in dimensionality reduction is on basic pattern findings and reducing repetitions. It can also be used for feature extraction, machine learning, and data visualization. The dimension reduction method reduces the data dimensionality with minimal information loss and executing the evaluation.

The major challenge of dimension reduction is to preserve the intrinsic structure of the information and to perform the reshaping with minor data loss. Moreover, a dimensional reduction technique consists of two categories, including linear or nonlinear methods. The principal linear methods are Principal Component Analysis [26], multidimensional scaling [3], Independent Component Analysis (ICA), Compact Matrix Decomposition (CMD), Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), and CUR Matrix Decomposition.

The main nonlinear methods are Kernel PCA, Laplacian Eigenmaps [2] Isomap [13], FastMap, Locally Linear Embedding [27]. However, these nonlinear methods are afflicted with the out-of-sample problem [28], [29]. Hinton and Maaten proposed high-dimensional local and global structure methods, which are known as t-SNE [9]. Moreover, the dimension reduction technique decreases the problem of the complexity of the data, along with enhancing the accuracy of data exploration [10].

B. CLASSICAL ISOMAP

The main advantages of the classical Isomap are globally optimal, provable convergence guarantees and appropriability for the nonlinear manifold. Due to its excellent performance, it has been used in various domains, such as pattern recognition [14], [30] image processing [12], signal processing [31] robotics [32], and computer vision graphics [33]. Conventionally, E Euclidean distance is used to compute the dissimilarities between data points, but this is based on the hypothesis of a global linear structure, regardless of whether datasets are extracted from a supervised (linear structure) method or an extremely distorted unsupervised (nonlinear) structure. Classical Isomap uses geodesic distance instead of Euclidean Distance E for data placed on a nonlinear manifold. The actual distance between the two data points will be the geodesic distance on the manifold, but this may not be the distance along the surface of the manifold.

Furthermore, the classical Isomap is to find out the optimal subspace and preserve the shape of the geodesic distance data points [34]. A summary of the classical Isomap is as follows:

1) BUILD THE NEAREST NEIGHBOR GRAPH

First, a classical Isomap method finds the neighborhood data on the manifold. The neighborhood data is symbolized as ε and K in this section [35]. If $d(x_i, x_j)$ is smaller than the specified radius ε or $x_j \in K$ are the nearest neighbors of x_i, $x_{i,}$ and x_j for each pair of the data points. Build the undirected, weighted (KNN) graph $G\ (V, E)$ of the manifold learning, based on the distance between the pair of the data points x_i in the input space. Suppose x_i and x_j are neighbors, the weight of the edge $(x_i, x_j) \in E$ is set to $d\ (x_i, x_j)$; otherwise, it is set to $+\infty$.

2) CALCULATE THE GEODESIC DISTANCE

After searching the neighborhood data on the manifold, classical Isomap computes the geodesic distance $d_G(x_i, x_j)$ matrix between all pairs of data points, using the shortest distance path graph in the G algorithm and then calculates the shortest distance path by using Floyd's algorithm and Dijkstra's algorithm [36].

3) BUILD THE LOW-DIMENSIONAL EMBEDDING

Identify low-dimensional embedding by executing Eigendecomposition on the geodesic distance density matrix. Some essential features of high dimensional datasets, such as face images, handwritten digits, and hand-gesture images are detected by Isomap [8]. Although it also has some shortcomings, the geodesic distance is suitable for nonlinear data points, and it can efficiently regenerate the structure of topological datasets.

Furthermore, while using the complex and noisy input data, the performance of the dimension reduction considerably reduced because the geodesic distance calculation is sensitive to noise [21], which might destroy the shape of local

neighboring or make disjointed edges. Therefore, it cannot be used for labeled datasets or full classification information of datasets.

Here, we briefly describe the 17 general shortest-path algorithms (GSPA) in Table 1 [37]. The Dijkstra algorithm can be used to calculate the shortest path in various fields, such as logistic distribution line [38], link-state routing (LSR) protocols [39], robot path planning [40], and Open Shortest Path First (OSPF) [41].

TABLE 1. 17 general shortest path algorithms (GSPA).

Algorithm Class	Abbreviation	Implementation Data Structures
A*-Search	ASH	A*-Search algorithm with k-array Heap
	ASBD	A*-Search algorithm with Double Buckets
	ASBA	A*-Search algorithm with Approximate Buckets of data structures
Bellman-Ford-Moore	BFM	Bellman-Ford-Moore
	BFP	Bellman-Ford-Moore with parent- - checking
Dijkstra	DKQ	Dijkstra's Naive Implementation
	DKB	Dijkstra's Buckets - - Basic Implementation
	DKD	Dijkstra's Buckets - - Double Implementation
	DKA	Dijkstra's Buckets - - Approximate
	DKM	Dijkstra's Buckets - - Overflow Bag
	DKF	Dijkstra's Heap - - Fibonacci
	DKH	Dijkstra's Heap - - k- - array
	DKR	Dijkstra's Heap - - R- - Heap
Graph Growth Model	TQQ (Two-Q)	Graph Growth - - Gallo & Pallottino algorithm with two queues data structures
	PAP	Graph Growth - - Pape
Threshold Algorithm	THR	Threshold Algorithm
Topological Ordering	GR1	Topological Ordering - - Basic
Topological Ordering	GR2	Topological Ordering - - Distance Updates

In the literature of the classical Isomap, [14], [20], and an improved classical Isomap method is proposed, based on the Dijkstra algorithm with the Fibonacci Heap (Fib-Dij). The Fib-Dij algorithm can handle the problem of the shortest path of all pairs of data points, based on the neighborhood graph. This method is much faster than Floyd's algorithm and improves the speed of the classical Isomap.

C. APPROXIMATE NEAREST NEIGHBOR (ANN) SEARCH ALGORITHMS

The nearest neighbor search has gained considerable attention in recent years. ANN search algorithms are used for highly accurate results and lower time complexity. There are two types of ANN search algorithms; Hierarchical index-based algorithms and Hashing-based algorithms. Hierarchical index-based algorithms, such as KD-tree has achieved success in ANN search problems. Randomized KD-tree and K-means trees are the most popular and newly-developed open-source hierarchical index-based algorithms [42]–[45]. Spectral hashing, Locality Sensitive Hashing (LSH), and Iterative Quantization are types of Hashing-based algorithms.

D. K NEAREST NEIGHBOR (KNN) GRAPH

Constructing the KNN graph for large-dimensional datasets is difficult for several applications, such as network analysis, manifold learning, collaborative filtering, and similarity search. However, the exact time complexity of KNN is $O(Nd)$, where N is the number of data points, and d is the number of dimensions, which is very costly. The existing techniques use three types of methods; Locality-sensitive hashing (LSH) methods, neighbor exploring (NN-descent) method [46], and space-partitioning tree algorithm [24], [42]. Space-partitioning tree approaches divide the whole dataset into different sectors and arrange the sectors into various tree frames, e.g., KD-tree, cover tree, random projection tree, and VP-tree [24]. These methods work efficiently on different types of data points, and their efficiency has been verified on data with a small number of dimensions [9]. In this paper, we propose a FastIsomapVis method for high-dimensional and large-scale data, which differs from the existing method; our method uses the accuracy of KNN graphs to provide 100% accurate results without investing in many trees.

III. FastIsomapVis

In general, given high and large-scale dimensional datasets $X = \{x_i \in R^d\} i = 1, 2, \ldots, N, ,$ we aim to denote each data point x_i with a low-dimensional vector $y_i \in R^d$, where the value of d is generally 3 or 4. The main goal of the visualization of high-dimensional data is to protect the basic shape of the datasets in the low-dimensional data. Current techniques often calculate the similarities of all data pairs $\{x_i, x_j\}$ and then protect the similarities in the low-dimensional transmutation. We have used the Euclidean distance $\sqrt{d = \|x_i - x_j\|}$ matrix in the high dimensional space instead of geodesic distances metric because graphs of KNN take a distance matrix. We have used the Euclidean distance E to compute the similarities of the datasets via a square matrix, which provides much more cost-effective and accurate results than geodesic distances. We show the main steps of the FastIsomapVis method in Fig 1.

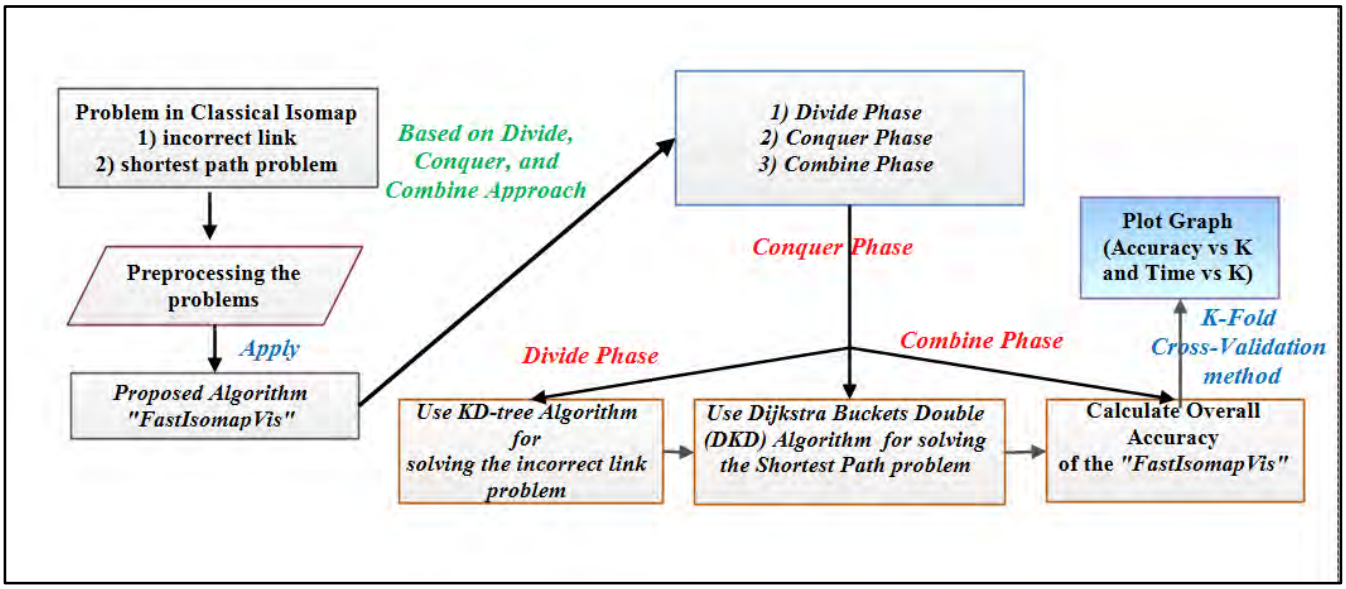

FIGURE 1. Main steps of the FastIsomapVis method.

A. HIERARCHICAL DIVIDE, CONQUER, AND COMBINE APPROACH

The divide, conquer, and combine method is a very useful approach to achieve this goal. First, the divide, conquer, and combine method splits the problem into sub-problems recursively, and then the sub-problems are divided into small-scale problems that are easy enough to solve. We combine the results of the sub-problems to get a result of the actual problem. The divide, conquer, and combine method is a better and easier way to approximate KNN graph construction. To reduce the computation time of data, we need to search for a better 'conquer' approach and avoid applying 'conquering' to the root node.

In this paper, we follow the previous work of Fu and Cai [45] and Wang *et al.* [47], both of which mentioned that multiple randomized division could make overlapping subsets. As a result, there is no need for the 'conquer' phase to be carried out on the root node. In the 'divide' step, we use the randomized KD-tree building algorithm to handle incorrect-link problems and deal with the K nearest neighbor case in a boundary case, and divide the graphs into subgraphs. The 'conquer' step involves using the DKD algorithm to calculate the shortest path for K nearest neighbor values and merging the subgraphs. Our motivation for developing a better 'conquer' approach is to decrease the number of data points, and we always pick the closest possible data points for conquering points. We make sure that the quality of conquering at each level of data points is maintained. In the 'combine' step, we have to calculate the overall accuracy of the FastIsomapVis method by using the KD-tree search method and make an actual graph of the K nearest neighbor.

B. DIVIDE PHASE

1) RANDOMIZED KD-TREE ALGORITHM (KD-TREE)

Hierarchical approaches are used for the initialization of the graph data. There are various hierarchical approaches, such as hierarchical clustering, multi-table hashing, and randomized KD-tree. The randomized KD-tree algorithm is used in our method. One part of the FastIsomapVis is based on the divide Phase. We showed the Algorithm 2 details on Randomized KD-tree (divide phase) in this section. The basic idea of the KD-tree is to provide better initialization to improve the performance of classical Isomap significantly.

The most common method for better query time-interval is to compute an Approximate K-Nearest Neighbor search (AKNN) instead of exact KNN; AKNN approaches are much more useful than exact KNN. The AKNN uses two techniques; FLANN and Randomized KD-tree. The FLANN approach can only be used for interactive visual analytics, whereas Randomized KD-tree can be used for AKNN.

The KD-tree algorithm splits a multidimensional data points repeatedly and then builds the tree that can be used for quick searching. KD-tree is the most popular and extensively used technique for KNN problems. This method splits data points by vector space and constructs a binary tree with logarithm time cost for the KNN search.

In the binary tree, data points are divided into two subdivision groups by the size of every level, in which data points have the highest variance [48]. Silpa-Anan and Hartley [42] proposed the concept of the KD-tree for KNN sear in high dimensional space. The central idea of the KD-tree is that the data point is split by dimension repeatedly and then selected randomly from a small set of data points with the highest variance. Furthermore, Muja and Lowe [44] proposed two

algorithms, such as hierarchical K-means trees and randomized KD-tree for KNN querying [48].

The nodes size of each data point is *2N-1*; the cost of at each leaf is *O (N)*. The cost of the depth of the tree is *O (logN)*. The cost of the median of each data point of a tree and left, the right child nodes of each data point of the tree are $O(N)$. Hence the total time complexity of KD-tree is *O (NlogN)*.

Algorithm 2: Randomized KD-Tree Algorithm (Divide Phase)

Data: The number of trees $\boldsymbol{T}$, the number of points in a leaf node $\boldsymbol{K}$, and the data dimension $\boldsymbol{d}$.
Result: A Randomized KD-tree set $\boldsymbol{S}$.
function *MAKE-TREE (NODE, POINT-SET)*
  if (size of $POINT - SET < K$) **then**
    return;
  else
    Choose Randomly data dimension $\boldsymbol{d}$.
    Calculate the median over the *POINT-SET* on data Dimension $\boldsymbol{d}$.
    Divide the POINT-SET equally into two subsets, LEFT-HALF and RIGHT-HALF, according to the median.
    MAKE-TREE (*NODE.LEFT-CHILD, LEFT-HALF*)
    MAKE-TREE (*NODE.RIGHT-CHILD, RIGHT-HALF*)
  end
  return
end function
for ($i \leftarrow 1$ to T) **do**
  MAKE-TREE ($ROOT - NODE_i$, D)
  Add $ROOT - NODE_i$ to S.
end

C. CONQUER PHASE

1) DIJKSTRA BUCKETS DOUBLE (DKD) ALGORITHM

In Algorithm 3, we have used the Dijkstra Buckets Double (DKD) algorithm for the shortest path problem. The divide' phase involves building the randomized division tree algorithm, as described in the previous paragraph. The 'conquer' phase is then performed, along with the DKD algorithm, to calculate the shortest path of the KNN graph.

In this section, we have to use a shortest-path algorithm, such as the Dijkstra buckets double (DKD) algorithm. Table 1 shows the details of the 17 general shortest path algorithms (GSPA). The Dijkstra buckets double (DKD) algorithm implementation, which combines two algorithms; Dijkstra Bucket Overflow Bag (DKM) and Dijkstra Buckets Approximate (DKA) [37]. It consists of random levels of buckets, which are managed by the implementation of the DKD. A matrix D bucket in the random level buckets is used. First, we calculate the cost of the shortest path of the KNN and find the shortest value of KNN by using a breath-first-search algorithm. After that, we sort the values of the KNN in random buckets by using a bubble sort algorithm. Next, we identify the minimum cost of the KNN and then make a matrix of the resulting values of the KNN. The DKD uses the Euclidean distance E instead of the geodesic distance for the shortest path of the KNN graph. The DKD is much faster than the Dijkstra algorithm and provides higher accuracy in the KNN graph.

The time cost of the breath-first-search is $O(N+E)$, where N and E are the number of nodes and edges, respectively. The time complexity of the bubble sort is $O(N)$. The cost of extract-min of the KNN is *O (logN)*. Hence the total cost of Dijkstra Buckets Double (DKD) is *O (logN)*.

D. COMBINE PHASE

In the 'combine' phase, we have to calculate the overall accuracy of the FastIsomapVis method via the KD-tree searcher method.

1) KD-TREE SEARCHER

We used the KD-tree algorithm, and it uses the KD-tree search model objects method. The KD-tree searcher method collects the results of K nearest neighborhood search. In the result section, we include the training datasets, its parameters, distance metric, and the maximal number of bucket sizes in every leaf node. The KD-tree algorithm divides N *by* K data points repeatedly and splits the N data points in K dimensional space into a binary tree mode. When we created a KD-tree searcher model object, we also identified the stored tree.

When we explored all the KNN data pints to the query data, the KD-tree searcher methods use two methods to find out K nearest neighbor values such as Rangesearch and Knnsearch. We use the Knnsearch method in our work [49], [50].

2) KNNSEARCH METHOD

Given parameters in the Knnsearch method, nearest neighbor training data indices are represented in the *Idx* nearest neighbor searchers by X; query data by Y parameter; name by *Name;* and corresponding value by *value*. We have used two Knnsearch methods for the nearest neighbor, searcher. The $Idx = the\ Knnsearch(X, Y)$ method identifies the nearest neighbor (i.e., the nearest data points, rows) in the nearest neighbor search datasets, *X,* to each point in the query data, *Y,* for KD-tree; the *Idx = the Knnsearch(X, Y, Name, Value)* method returns the indices of the nearest data points in the nearest neighbor search datasets, *X,* to *Y,* with extra options specified by one or more Name, Value pair parameters [49].

The following is the detail of Algorithm 3 of the Divide, Conquer, and Combine algorithm.

E. OUTLINE OF THE FastIsomapVis METHOD

The outline of the FastIsomapVis method can be concluded as follows:

Algorithm 3: Hierarchical Divide, Conquer, and Combine Algorithm

Data: the randomized KD-tree set S constructed within Algorithm 1, the matrix D, the approximate KNN graph is K, no. of nodes N, NSMethod is the nearest neighbor search method, C is the accuracy, and CP is the CorrectRate.
Result: An Approximate KNN graph.
the Division step
  Using the Algorithm 2 to MAKE-TREE, which shows to the tree input S
the Conquering step with DKD
  foreach $(K \in N)$
    col = COST(K)
    ml = col(FIND(col))
    m = SORT(UNIQUE(ml))
    for $(j \in length(m))$
      mx = m(j)
      temp = MIN(COST(col)[ ],2)
      temp = temp + mx
    end for
    COST(K) =col
    foreach $(K \in N)$
      COST(K, K) =r
    end foreach
  end foreach
end Conquering step
the Combining step
  Start time
  D = CREATENS (***the Conquering step***, '*NSMethod', 'kDtree','Distance', 'euclidean'*)
  KDTreeSearcher with properties (K)
  for (K ← 1 to 5) **do**
    C = KNNCLASSIFY (D,D,D, K,'eucli')
    CP = CLASSPERF (D, C)
    Time(K) = End time
    K(K) = CP.CorrectRate
    return CP
  end for
end Combining step

Step-1: Divide Phase
- We performed the Randomized KD-tree Algorithm (KD-tree) for the incorrect link problem.

Step-2: Conquer Phase
- We made a square matrix for the DKD and calculated the similarities of the low-dimensional space.
- We used the DKD method to calculate the shortest path of the KNN graph.

Step-3: Combine Phase
- We calculated the overall accuracy of the FastIsomapVis method via the KD-tree searcher method and constructed the KNN graph to the datasets.

IV. EXPERIMENTS

In this section, we evaluate the cost-effectiveness and efficiency of the FastIsomapVis method by overall experiments on high and large-scale dimensional datasets. We analyze the execution of the proposed algorithms for building the KNN graph with a randomized division tree and DKD algorithms and then visualize them in a graph.

A. DATASETS

The experimental results were implemented on nine high and large-scale dimensional of social network datasets, Facebook, Twitter, Amazon, LiveJournal, YouTube, and Orkut, which are available in [51]. 20Newsgroups [52], SIFT1M [53], and Perfume [54], [55]. The detail of the datasets is listed in Table 2.

TABLE 2. Properties of the datasets.

Dataset	Data	Dimensions	Categories
Facebook	4,039	40	10
Twitter	81,306	40	973
20Newsgroups	18,846	100	30
Amazon	334,863	100	1000
SIFT1M	100,000	128	1000
LiveJournal	3,997,963	100	5000
YouTube	1,134,890	150	5000
Orkut	3,072,441	150	5000
Perfume	560	40	20

B. RESULTS AND EXPERIMENTS ON THE CONSTRUCTION OF THE KNN GRAPH

We present unique algorithms for the construction of the K-Nearest-Neighbor (KNN) graphs, and our FastIsomapVis method is based on the below algorithms.

- Randomized Division Tree (KD-tree).
- Divide, Conquer and Combine Algorithm (KNN Graph Construction) with DKD algorithm

C. EVALUATION METHOD

There are many evaluation methods introduced for calculating the accuracy in recent years. There are two types of evaluation methods, including descriptive and predictive. We have used the predictive evaluation method for calculating accuracy. In the predictive evaluation method, the training classifier is done from the training classifier, and evaluation is done by a test classifier instances [56]. There are two predictive evaluation methods, such as the holdout set method and K-fold cross-validation method [57].

Well-known classification techniques include the KNN classifier [58], [59], SVM [60], [61], BP neural network [62], image processing [63], decision tree [64], biomedical research [65], and remote sensing [66]. The KNN classifier is a modern and non-parametric technique in classification [67]. Moreover, the predictive accuracy method and KNN classifier method are both used the KFCV method for calculating the accuracy by using Eq.1 [68]. We use the well-established KFCV method for calculating the accuracy of the "FastIsomapVis" and "Classical Isomap" algorithm.

TABLE 3. Overall accuracy of the FastIsomapVis method.

	Facebook		Twitter		20Newsgroups		Amazon		SIFT1M		LiveJournal		YouTube		Orkut		Perfume	
K	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)
1	1.00	916.34	1.00	1962.87	1.00	4.09	1.00	5.06	1.00	1.56	1.00	104.07	1.00	76.45	1.00	309.91	1.00	0.11
2	1.00	1803.71	1.00	3928.44	1.00	6.68	1.00	5.41	1.00	2.11	1.00	203.41	1.00	192.26	1.00	578.71	1.00	0.17
3	1.00	2693.08	1.00	5954.39	1.00	9.12	1.00	5.73	1.00	2.66	1.00	304.33	1.00	381.08	1.00	842.30	1.00	0.23
4	1.00	3581.71	1.00	7914.29	1.00	11.56	1.00	6.12	1.00	3.21	1.00	403.93	1.00	692.89	1.00	1117.05	1.00	0.28
5	1.00	4471.05	0.90	9895.67	1.00	14.05	1.00	6.45	1.00	3.78	0.88	503.90	0.99	965.82	0.86	1405.75	1.00	0.35
6	1.00	5360.13	0.90	12114.59	1.00	17.35	1.00	6.77	1.00	4.34	0.88	603.39	0.99	1326.26	0.85	1680.49	0.99	0.42
7	0.99	6250.56	0.81	14384.83	1.00	21.68	1.00	9.05	1.00	4.91	0.82	703.08	0.99	1545.25	0.82	1958.04	0.99	0.48
8	0.99	7138.51	0.81	16351.28	1.00	24.99	1.00	9.39	1.00	5.49	0.82	806.13	0.99	1853.34	0.81	2247.46	0.99	0.53
9	0.99	8027.64	0.73	18320.05	1.00	28.07	1.00	9.73	1.00	6.06	0.78	938.82	0.99	2385.68	0.77	2533.73	0.99	0.58
10	0.99	8918.18	0.73	20289.11	1.00	30.60	1.00	10.07	1.00	6.64	0.78	1051.08	0.99	2738.47	0.76	2814.67	0.99	0.63
11	0.99	9808.45	0.67	22258.83	1.00	33.09	1.00	10.42	1.00	7.22	0.75	1169.39	0.99	3157.59	0.74	3142.99	0.99	0.69
12	0.99	10705.27	0.67	24228.92	1.00	35.59	1.00	10.78	1.00	7.81	0.75	1285.96	0.99	3469.92	0.73	3436.23	0.98	0.74
13	0.99	11597.22	0.61	26198.13	1.00	38.44	1.00	11.12	1.00	8.40	0.73	1409.72	0.99	3764.28	0.72	3702.96	0.98	0.80
14	0.99	12489.69	0.61	28170.59	1.00	41.08	1.00	11.47	1.00	9.00	0.73	1529.83	0.99	4109.18	0.70	3971.73	0.97	0.86
15	0.99	30488.28	0.56	30139.75	1.00	43.59	1.00	11.81	1.00	9.61	0.71	1657.03	0.99	5891.25	0.68	4239.76	0.97	0.92

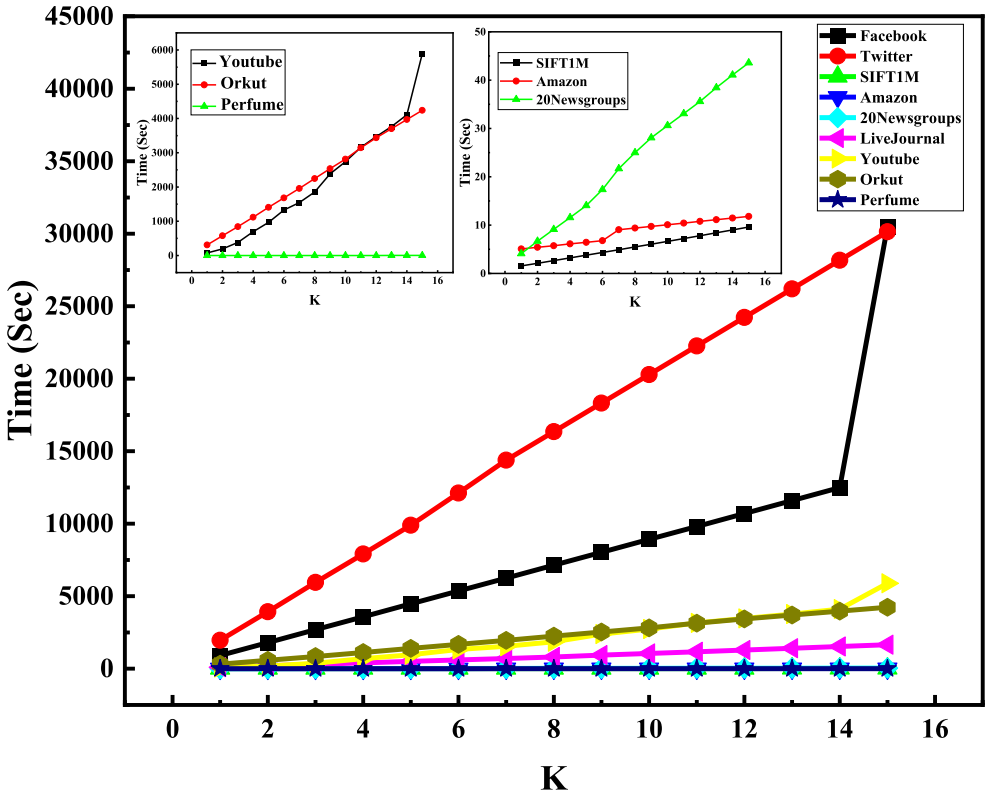

FIGURE 2. Time versus K graph of Facebook, Twitter, 20Newsgroups, SIFT1M, LiveJournal, Amazon, YouTube, Orkut, and Perfume datasets in FastIsomapVis method.

1) K-FOLD CROSS-VALIDATION METHOD (KFCV)

The K-fold cross-validation method is used for the empirically to choose the parameters. Although, for each parameter, the K-fold cross-validation method is verified by several values and selected the best one value [68].

We use the popular KFCV method for calculating the accuracy of the FastIsomapVis algorithm. The KFCV method randomly splits the available data into N portions of equally sized datasets. This KFCV procedure is performed N times, which provides N accuracy. P is denoting the accuracy of the N data points, and the C is the number of correct classifications, are defined as:

$$P = C/N \tag{1}$$

Table 3 shows the accuracy of Facebook, Twitter, 20Newsgroups, Amazon, SIFT1M, LiveJournal, YouTube, Orkut, and Perfume datasets from Eq.1.

D. ACCURACY COMPARISONS BETWEEN FastIsomapVis ALGORITHM AND CLASSICAL ISOMAP ALGORITHM

In Fig. 2 and 3, we have represented the relationship between time and accuracy for nine different high dimensional datasets. The graphs represent that our proposed algorithm consistently attains the highly accurate results in less computation time. We have used the value of K from 1 to 15 to calculate the accuracy and time for the nine different datasets.

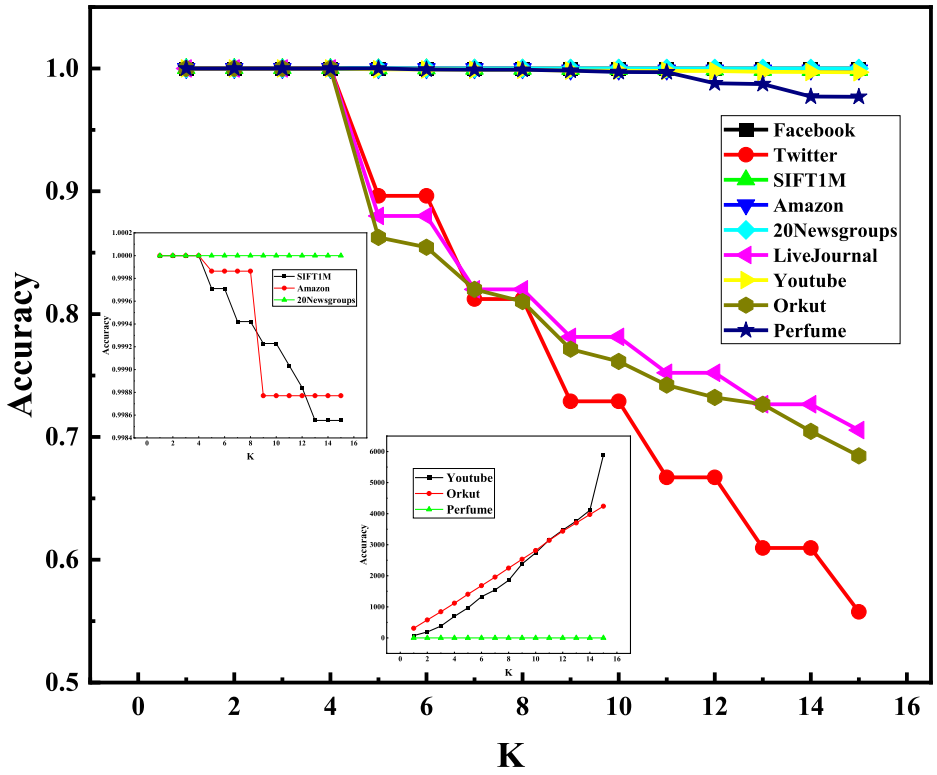

FIGURE 3. Accuracy versus K graph of Facebook, Twitter, 20Newsgroups, SIFT1M, LiveJournal, Amazon, YouTube, Orkut, and Perfume datasets in FastIsomapVis method.

For Facebook datasets having 4039 data points, we have calculated 100% accurate results for initial values of K (1 to 6) and got 0.99% results for higher values of K, as shown in Fig. 3. The time taken to calculate these results increased linearly from 916.34 to 30488.28 seconds. The reason for a slight decrease in the accuracy for higher values of K is that some neighboring data points are far away to each other.

For 20Newsgroups, SIFT1M, Amazon datasets having data points, 18,306, 100,000, and 334,863, respectively: we have achieved 100% accurate results for all the values of K, as shown in Fig. 3. The computation time to calculate these results increased linearly with the increase of K, as shown in Fig. 2.

For twitter, LiveJournal, and Orkut datasets having 81,306, 3,997,963, and 3,072,441 data points, we have observed that the accuracy is 100% for initial values of K (1 to 4), but it gradually decreases with the increase of values of K (5 to 15) as shown in Fig. 3. The reason for this strange behavior is that Twitter, LiveJournal, and Orkut datasets have large categories as compared to their dimensions and data points, as shown in Table 2. But the computation time of Orkut is less than Twitter and LiveJournal, as shown in Table 3 because some neighboring data points are far away to each other, and some data points are close to each other.

For YouTube datasets having 1,134,890 data points, we have calculated 100% accurate results for initial values of K (1 to 4) and got 0.99% results for higher values of K, as shown in Fig. 3. The time taken to calculate these results increased linearly from 76.45 to 5891.25 seconds. The reason for a slight decrease in the accuracy for higher values of K is that some neighboring data points are far away to each other, and some data points are close to each other.

For Perfume datasets having 560 data points, we have calculated 100% accurate results for initial values of K (1 to 5) and got 0.97% results for higher values of K, as shown in Fig. 3. The computation time to calculate these results increased linearly with the increase of K, as shown in Fig. 2.

In Fig. 3, the three datasets have the same accuracy; therefore, the graph values are overlapping to each other. Moreover, Facebook and Twitter datasets have the same computation time; the reason for this strange behavior of the Twitter dataset is that neighboring data points are closed to each other and directed graph. However, the overall performance of our proposed method continuously achieves the high authentic results for the nine high and large-scale dimensional datasets.

E. TIME COMPLEXITY ANALYSIS

In this section, we have analyzed the time complexity of Alg. 1, Alg. 2 and Alg. 3. As shown above, the classical Isomap (Alg. 1) time cost is $O(N^3)$, KD-tree (Alg. 2) total time complexity is $O\ (NlogN)$, and (Alg. 3) Dijkstra Buckets Double (DKD) is $O\ (logN)$. Our FastIsomapVis method is based on the divide, conquer, and combined approach. The divide, conquer, and combine method is a better and easier way to approximate KNN graph construction.

TABLE 4. Accuracy comparison of the FastIsomapVis (KD-tree, Dijkstra buckets double (DKD)) and classical Isomap algorithms.

	Facebook		Twitter		20Newsgroups		Amazon		SIFT1M		LiveJournal		YouTube		Orkut		Perfume	
K	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap	FastIso mapVis	Isomap
1	1.00	0.85	1.00	0.80	1.00	0.87	1.00	0.88	1.00	0.87	1.00	0.79	1.00	0.79	1.00	0.79	1.00	0.89
2	1.00	0.80	1.00	0.78	1.00	0.84	1.00	0.86	1.00	0.84	1.00	0.76	1.00	0.77	1.00	0.78	1.00	0.88
3	1.00	0.78	1.00	0.74	1.00	0.80	1.00	0.84	1.00	0.80	1.00	0.74	1.00	0.75	1.00	0.76	1.00	0.86
4	1.00	0.74	1.00	0.70	1.00	0.78	1.00	0.80	1.00	0.78	1.00	0.70	1.00	0.72	1.00	0.74	1.00	0.84
5	1.00	0.69	0.90	0.66	1.00	0.74	1.00	0.78	1.00	0.74	0.88	0.65	0.99	0.69	0.86	0.68	1.00	0.81
6	1.00	0.64	0.90	0.62	1.00	0.70	1.00	0.75	1.00	0.70	0.88	0.62	0.99	0.65	0.85	0.64	0.99	0.79
7	0.99	0.60	0.81	0.54	1.00	0.67	1.00	0.73	1.00	0.67	0.82	0.54	0.99	0.60	0.82	0.61	0.99	0.74
8	0.99	0.54	0.81	0.50	1.00	0.64	1.00	0.70	1.00	0.64	0.82	0.51	0.99	0.58	0.81	0.57	0.99	0.70
9	0.99	0.50	0.73	0.47	1.00	0.62	1.00	0.68	1.00	0.62	0.78	0.46	0.99	0.54	0.77	0.53	0.99	0.69
10	0.99	0.48	0.73	0.44	1.00	0.59	1.00	0.65	1.00	0.59	0.78	0.43	0.99	0.52	0.76	0.50	0.99	0.66
11	0.99	0.42	0.67	0.40	1.00	0.54	1.00	0.61	1.00	0.54	0.75	0.39	0.99	0.49	0.74	0.47	0.99	0.64
12	0.99	0.36	0.67	0.34	1.00	0.50	1.00	0.57	1.00	0.50	0.75	0.34	0.99	0.46	0.73	0.43	0.98	0.60
13	0.99	0.32	0.61	0.30	1.00	0.49	1.00	0.52	1.00	0.49	0.73	0.31	0.99	0.42	0.72	0.40	0.98	0.58
14	0.99	0.30	0.61	0.28	1.00	0.47	1.00	0.47	1.00	0.47	0.73	0.29	0.99	0.40	0.70	0.38	0.97	0.55
15	0.99	0.29	0.56	0.27	1.00	0.46	1.00	0.44	1.00	0.46	0.71	0.26	0.99	0.39	0.68	0.32	0.97	0.52

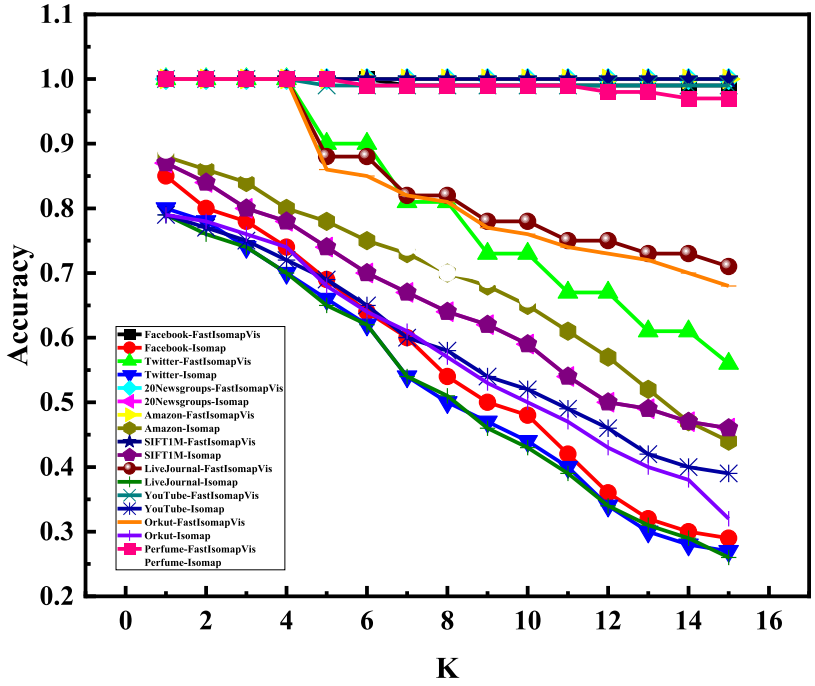

FIGURE 4. Overall accuracy performance of the FastIsomapVis and classical Isomap algorithm.

Our FastIsomapVis method reduced the computation time of large scale and high dimensional datasets and provided highly accurate results. Hence the overall time complexity of our FastIsomapVis is *O (NlogN)*. The experimental results and time complexity analysis shows that our FastIsomapVis method is much fastened and effective than the classical Isomap.

In Fig. 4, we have compared the overall accuracy of the FastIsomapVis and classical Isomap methods for nine different datasets. We calculated highly accurate results from 20Newsgroups, Amazon, and SIFT1M datasets by the FastIsomapVis method: the FastIsomapVis method provided 100% accuracy as compared to classical Isomap whereas the classical Isomap algorithm is computationally expensive for high and large-scale dimensional datasets. For Facebook, Twitter, LiveJournal, YouTube, Orkut, and Perfume datasets, it is very difficult to construct the KNN graph with 89% to 29% accuracy through the classical Isomap algorithm for the value of K (1 to 15), as shown in Table 4. It is also evident that the accuracy of classical Isomap decreases significantly with the increase of values of K. In contrast, our proposed has more stable and accurate results than the classical Isomap. Both Table 4 and Fig. 4 specify that FastIsomapVis is much more accurate and less sensitive to K than classical Isomap. Our FastIsomapVis method for the construction of the KNN graph has generated very useful results at a larger scale of millions of data points with hundreds of dimensions.

V. CONCLUSION

In this paper, we have presented a visualization technique called FastIsomapVis. FastIsomapVis straightforwardly measures datasets with millions of data points and with hundreds of dimensions. We propose a very cost-effective algorithm for an approximated K nearest neighbor graph construction,

and as well the graphs project them into the low-dimensional space. The empirical results on nine real-world datasets show that FastIsomapVis method performance is very useful and better for the construction of the KNN graph and datasets rather than classical Isomap. In our future work, we plan to use advanced techniques of a hierarchical method for the FastIsomapVis.

REFERENCES

[1] D. A. Keim, "Information visualization and visual data mining," *IEEE Trans. Vis. Comput. Graphics*, vol. 8, no. 1, pp. 1–8, Jan./Mar. 2002.
[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," presented at the 14th Int. Conf. Neural Inf. Process. Syst., Natural Synth., Vancouver, BC, Canada, 2001.
[3] W. S. Torgerson, "Multidimensional scaling: I. Theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, Dec. 1952.
[4] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[5] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw., Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.
[6] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software," *PLoS ONE*, vol. 9, no. 6, Jun. 2014, Art. no. e98679.
[7] S. Martin, W. M. Brown, R. Klavans, and K. W. Boyack, "OpenOrd: An open-source toolbox for large graph layout," *Proc. SPIE*, vol. 7868, Jan. 2011, Art. no. 786806.
[8] M. Yousaf, T. U. Rehman, and L. Jing, "An extended isomap approach for nonlinear dimension reduction," in *SN Computer Science A Springer Nature Journal*. Singapore: Springer, May 2020.
[9] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proc. 25th Int. Conf. World Wide Web (WWW)*, 2016, pp. 287–297.
[10] V. Sumithra and S. Surendran, "A review of various linear and non linear dimensionality reduction techniques," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, pp. 2354–2360, 2015.
[11] B. Zhang, "Multiple features facial image retrieval by spectral regression and fuzzy aggregation approach," *Int. J. Intell. Comput. Cybern.*, vol. 4, no. 4, pp. 420–441, Nov. 2011.
[12] R. Verma, P. Khurd, and C. Davatzikos, "On analyzing diffusion tensor images by identifying manifold structure using isomaps," *IEEE Trans. Med. Imag.*, vol. 26, no. 6, pp. 772–778, Jun. 2007.
[13] J. B. Tenenbaum, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.
[14] T. Qu and Z. Cai, "An improved isomap method for manifold learning," *Int. J. Intell. Comput. Cybern.*, vol. 10, no. 1, pp. 30–40, Mar. 2017.
[15] F. Schoeneman, V. Chandola, N. Napp, O. Wodo, and J. Zola, "Entropy-isomap: Manifold learning for high-dimensional dynamic processes," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1655–1660.
[16] M. Balasubramanian and E. L. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, p. 7, 2002.
[17] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: A comparative," *J. Mach. Learn. Res.*, vol. 10, nos. 66–71, p. 13, 2009.
[18] H. Choi and S. Choi, "Robust kernel isomap," *Pattern Recognit.*, vol. 40, no. 3, pp. 853–862, Mar. 2007.
[19] W. Hong-Yuan, D. Xiu-Jie, C. Qi-Cai, and C. Fu-Hua, "An improved isomap for visualization and classification of multiple manifolds," in *Proc. Int. Conf. Neural Inf. Process.* Seoul, South Korea: Springer, 2013, pp. 1–12.
[20] T. Qu and Z. Cai, "A fast isomap algorithm based on Fibonacci heap," in *Proc. Int. Conf. Swarm Intell.* Beijing, China: Springer, 2015, pp. 225–231.
[21] J. A. Lee and M. Verleysen, "Nonlinear dimensionality reduction of data manifolds with essential loops," *Neurocomputing*, vol. 67, pp. 29–53, Aug. 2005.
[22] Y.-K. Lei, Y. Xu, S.-W. Zhang, S.-L. Wang, and Z.-G. Ding, "Fast ISOMAP based on minimum set coverage," in *Proc. Int. Conf. Intell. Comput.* Changsha, China: Springer, 2010, pp. 173–179.
[23] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, ed. New York, NY, USA: WH Freeman and Company, 1979.
[24] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *Proc. 14th Annu. ACM Symp. Theory Comput. (STOC)*, 2008, pp. 537–546.
[25] M. Card, *Readings in Information Visualization: Using Vision to Think*. San Mateo, CA, USA: Morgan Kaufmann, 1999.
[26] I. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer, 2011.
[27] S. T. Roweis, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.
[28] S. Mehta, B.-S. Zhan, and X.-J. Shen, "Weighted neighborhood preserving ensemble embedding," *Electronics*, vol. 8, no. 2, p. 219, Feb. 2019.
[29] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 177–184.
[30] N. Yazdian, Y. Tie, A. Venetsanopoulos, and L. Guan, "Automatic ontario license plate recognition using local normalization and intelligent character classification," in *Proc. IEEE 27th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2014, pp. 1–6.
[31] S. Gepshtein and Y. Keller, "Sensor network localization by augmented dual embedding," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2420–2431, May 2015.
[32] F. T. Ramos, S. Kumar, B. Upcroft, and H. Durrant-Whyte, "A natural feature representation for unstructured environments," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1329–1340, Dec. 2008.
[33] S. Takahashi, I. Fujishiro, and M. Okada, "Applying manifold learning to plotting approximate contour trees," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 6, pp. 1185–1192, Nov. 2009.
[34] X. Li, C. Cai, and J. He, "Density-based multi-manifold ISOMAP for data classification," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Dec. 2017, pp. 897–903.
[35] M. Maier, U. V. Luxburg, and M. Hein, "Influence of graph construction on graph-based clustering measures," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1025–1032.
[36] S. Hougardy, "The Floyd–Warshall algorithm on graphs with negative cycles," *Inf. Process. Lett.*, vol. 110, nos. 8–9, pp. 279–281, Apr. 2010.
[37] F. B. Zhan, "Three fastest shortest path algorithms on real road networks: Data structures and procedures," *J. Geographic Inf. Decis. Anal.*, vol. 1, no. 1, pp. 69–82, 1997.
[38] L. Xiao-Yan and C. Yan-Li, "Application of Dijkstra algorithm in logistics distribution lines," in *Proc. 3rd Int. Symp. Comput. Sci. Comput. Technol. (ISCSCT)*, Jiaozuo, China, 2010, pp. 048–050: Citeseer.
[39] R. Abujassar and M. Ghanbari, "Efficient algorithms to enhance recovery schema in link state protocols," 2011, *arXiv:1108.1426*. [Online]. Available: http://arxiv.org/abs/1108.1426
[40] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *Proc. 2nd Int. Conf. Mechanic Autom. Control Eng.*, Jul. 2011, pp. 1067–1069.
[41] A. Eneh and U. Arinze, "Comparative analysis and implementation of Dijkstra's shortest path algorithm for emergency response and logistic planning," *Nigerian J. Technol.*, vol. 36, no. 3, pp. 876–888, 2017.
[42] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[43] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 2161–2168.
[44] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. VISAPP (1)*, 2009, vol. 2, nos. 331–340, p. 2.
[45] C. Fu and D. Cai, "EFANNA-: An extremely fast approximate nearest neighbor search algorithm based on kNN graph," 2016, *arXiv:1609.07228*. [Online]. Available: http://arxiv.org/abs/1609.07228
[46] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proc. 20th Int. Conf. World Wide Web*, vol. 2011, pp. 577–586.
[47] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-NN graph construction for visual descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1106–1113.
[48] J. Jo, J. Seo, and J.-D. Fekete, "A progressive k-d tree for approximate k-nearest neighbors," in *Proc. IEEE Workshop Data Syst. for Interact. Anal. (DSIA)*, Oct. 2017, pp. 1–5.
[49] *Global Optimization Toolbox: User's Guide (R2016a)*, Math Work Inc., Natick, MA, USA, Aug. 2016.

[50] M. Gulraj and N. Ahmad, "Mood detection of psychological and mentally disturbed patients using Machine Learning techniques," *Int. J. Comput. Sci. Netw. Secur.*, vol. 16, no. 8, p. 63, 2016.
[51] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*. RMA Design Studios, 2014.
[52] J. Rennie, "20 Newsgroups," in *Home Page for 20 Newsgroups Data Set*. San Francisco, CA, USA: Ken Lang, 2008.
[53] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.
[54] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2019.
[55] B. Karlik and Y. Al-Bastaki, "Real time monitoring odor sensing system using OMX-GR sensor and neural network," *WSEAS Trans. Electron.*, vol. 1, no. 2, pp. 337–342, 2004.
[56] M. E. Syed, "Attribute weighting in k-nearest neighbor classification," M.S. thesis, Univ. Tampere, Tampere, Finland, 2014.
[57] D. A. Gredell, A. R. Schroeder, K. E. Belk, C. D. Broeckling, A. L. Heuberger, S.-Y. Kim, D. A. King, S. D. Shackelford, J. L. Sharp, T. L. Wheeler, D. R. Woerner, and J. E. Prenni, "Comparison of machine learning algorithms for predictive modeling of beef attributes using rapid evaporative ionization mass spectrometry (REIMS) data," *Sci. Rep.*, vol. 9, no. 1, pp. 1–9, Dec. 2019.
[58] T. K. Ho, "Nearest neighbors in random subspaces," in *Proc. Joint IAPR Int. Workshops Stat. Techn. Pattern Recognit. (SPR) Struct. Syntactic Pattern Recognit. (SSPR)*. Springer, 1998, pp. 640–648.
[59] D. G. Lowe, "Similarity metric learning for a variable-kernel classifier," *Neural Comput.*, vol. 7, no. 1, pp. 72–85, Jan. 1995.
[60] V. N. Vladimir and V. Vapnik, *The Nature of Statistical Learning Theory*, ed. Berlin, Germany: Springer, 1995.
[61] V. Vapnik, "*Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.
[62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[63] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2624–2637, Nov. 2013.
[64] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2016.
[65] B. Manavalan, S. Subramaniyam, T. H. Shin, M. O. Kim, and G. Lee, "Machine-Learning-Based prediction of cell-penetrating peptides and their uptake efficiency with improved accuracy," *J. Proteome Res.*, vol. 17, no. 8, pp. 2715–2726, Aug. 2018.
[66] L. Ma, M. M. Crawford, and J. Tian, "Local manifold learning-based *k*-nearest-neighbor for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4099–4109, Nov. 2010.
[67] S. Mehta, X. Shen, J. Gou, and D. Niu, "A new nearest centroid neighbor classifier based on k local means using harmonic mean distance," *Information*, vol. 9, no. 9, p. 234, Sep. 2018.
[68] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Trans. Syst., Man Cybern., B (Cybern.)*, vol. 35, no. 6, pp. 1098–1107, Dec. 2005.

MAHWISH YOUSAF received the B.S. degree in computer science from the University of the Punjab, Lahore, Pakistan, in 2012, and the M.S. degree in information technology from the University of Gujrat, Pakistan, in 2016. She is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Science and Technology of China (USTC), Hefei, China. Her research interests include machine learning, dimension reduction, image processing, data mining, artificial intelligence, and deep learning.

TANZEEL U REHMAN received the B.S. degree in software engineering from the University of Gujrat, Pakistan, in 2018. He is currently pursuing the M.S. degree with the Department of Computer Science, University of Science and Technology of China (USTC), Hefei, China. His research interests include machine learning, cloud computing, data mining, image processing, deep learning, and artificial intelligence.

DONGLIANG LIAO was born in Wuwei, China, in 1992. He received the B.S. and Ph.D. degrees in computer science from the University of Science and Technology of China, in 2014 and 2019, respectively. He is currently working with Tencent Inc. His research interests include mobility prediction, text modeling, and data mining.

NAJI ALHUSAINI received the B.S. degree in software engineering from Northeastern University (NEU), Shenyang, China, in 2011, the M.S. degree in computer software and theory from the Hefei University of Technology, Hefei, China, in 2015, and the Ph.D. degree from the Department of Computer Science, University of Science and Technology of China (USTC), Hefei. His research interests include data mining, utility mining, pattern discovery, machine learning, and cloud computing.

LI JING received the bachelor's, master's, and Ph.D. degrees in computer science from the University of Science and Technology of China (USTC), in 1987, 1990, and 1993, respectively. He is currently a Professor with the School of Computer Science and Technology, USTC. He published more than 80 articles in high-profile domestic and international academic journals. His recent research interests include distributed systems, cloud computing, and mobile computing. In the last few years, he was mainly engaged in the promotion and application of software technology, large-scale network systems, and the research and development of distributed algorithms. He is a Senior Member of the CCF.