# A KD-Tree-Based Nearest Neighbor Search for Large Quantities of Data

**Shwu-Huey Yen[1] and Ya-Ju Hsieh[2]**
[1] Department of Computer Science and Information Engineering, Tamkang University
New Taipei City, Taiwan
[e-mail: [1]105390@mail.tku.edu.tw, [2]698410122@s98.tku.edu.tw]
*Corresponding author: Shwu-Huey Yen

*Received July 7, 2012; accepted March 16, 2013; published March 29, 2013*

## *Abstract*

The discovery of nearest neighbors, without training in advance, has many applications, such as the formation of mosaic images, image matching, image retrieval and image stitching. When the quantity of data is huge and the number of dimensions is high, the efficient identification of a nearest neighbor (NN) is very important. This study proposes a variation of the KD-tree - the arbitrary KD-tree (KDA) - which is constructed without the need to evaluate variances. Multiple KDAs can be constructed efficiently and possess independent tree structures, when the amount of data is large. Upon testing, using extended synthetic databases and real-world SIFT data, this study concludes that the KDA method increases computational efficiency and produces satisfactory accuracy, when solving NN problems.

***Keywords:*** Arbitrary KD-tree (KDA), Feature Point, KD-Tree, Nearest Neighbor (NN), Image Stitching

**http://dx.doi.org/10.3837/tiis.2013.03.003**

## 1. Introduction

Feature matching in two images identifies the relationship between two sets of feature points extracted from images, such that corresponding points are associated by the least distance, compared to their association from other points. It is a key problem in tasks involving computer vision. Object recognition, tracking, building panoramas and image registration are just a few examples of many other possible applications. Successful feature matching requires the robust identification of the features. Various studies have used edges, corners, or histograms as features [1], [2]. The SIFT, proposed by Lowe [3], is probably the best known feature descriptor. It uses the difference between Gaussian (DoG) and scale space to locate the feature points and produces a local descriptor with a dimension of 128. Every image probably contains thousands of SIFT features. The large amount of high dimensional data makes feature matching between two images computationally cumbersome. Therefore, the efficient location of a nearest neighbor (NN) in such a large number of candidates and the high dimensionality are critical issues. This paper investigates the use of multiple KD-trees for large quantities of high-dimensional data and proposes an Arbitrary KD-tree (KDA) to manage this type of data, to ensure more efficient computation.

The remainder of the paper is organized as follows: Section 2 provides a review of related work, KD-trees and BBFs. Section 3 describes the method and the experimental results are given in Section 4. Finally, the conclusion is stated.

## 2. Related Work

One of the most widely used algorithms for a nearest-neighbor search in multidimensional data is the KD-tree [4]. A KD-tree is a space-partitioning data structure for organizing points in a K-dimensional space. It first calculates the variance of each dimension and then partitions the data into two halves, according to the medium on the dimension in which data possesses the greatest variance. The splitting procedure repeats, until the leaf node is reached [5]. The KD-tree works well when searching for an exact nearest neighbor in low dimensional data, but quickly loses its effectiveness as dimensionality increases. One way to improve the performance is to visit more leaf nodes, i.e, backtracking. However, as Silpa-Anan *et al.* indicated [6], increasing the amount of backtracking in one tree does not result in independent searches and, so there are diminished returns. Conversely, if multiple KD-trees are constructed, using different parameters and different methods of selection for partitioning values, for example, the order of the search nodes and the search results for these KD-trees may be different. Many studies have used multiple KD-trees with this concept, in the hope of increasing the level of independence of searches. Randomized Trees, Multiple Randomized KD-Trees, a hierarchical K-means tree, a PKD-Tree and Six-KD-Trees have all been used [7], [8], [9]. However, for large data sets, the space requirements for managing large numbers of trees create their own problems.

When finding a NN from a data set arranged in a KD-tree, backtracking is essential, to increase the accuracy of the search. Lowe [3] proposed the Best-Bin-First (BBF) method on a Priority Queue, to improve the execution of backtracking. The algorithm initially performs a single traverse through the tree and adds all of the unexplored branches in each node along the path to a priority queue. To execute a backtracking, the branch that has the closest distance to the query point is extracted and removed from the priority queue and the traverse of the tree is

restarted from that branch (similarly, all the unexplored branches along the path are added to the priority queue). The backtracking stops when a pre-defined number of leaf nodes have been visited, or when every distance from the unexplored branch to the query point is greater than $d$, where $d$ = min {distance (query-point, leaf-node)} for all visited leaf-nodes. For large quantities of high-dimensional data, the backtracking stops usually due to the number of visited leaf nodes reaches the pre-defined limit. When the search ends, the data point that corresponds to the distance, $d$, is returned as the search result.

Inspired by the work in [6], Muja and Lowe proposed a multiple randomized KD-tree (KDR) [7]. In order to increase "randomization", when constructing trees, the randomized KD-Tree (KDR) is constructed by randomly choosing the split dimension from the first $k$ dimensions in which data has the greatest variance. The fixed value, $k$ = 5, was used in their study. Multiple trees with different structures are easily produced, using this method. It was demonstrated that KDR can speed the matching of high-dimensional vectors by up to several orders of magnitude, compared to a linear search.

## 3. The Proposed Method

In this section, the definition of the proposed Arbitrary KD-tree (KDA) is given. Further discussions are provided with respect to randomization and complexity of the KDA.

### 3.1 Randomization

A KD-tree for $N$ data requires ($N$-1) splitting dimensions. If a dimension is randomly selected as the splitting dimension, to construct multiple KD-trees, say $M$ trees, then, altogether, there are $M$($N$-1) random selections for these trees. When $N$ is large, these different combinations of $M$($N$-1) random selections produce "randomization" in the trees. Based on this observation, a variation of the KD-tree algorithm is proposed, called an Arbitrary KD-tree (KDA). The construction of a KDA is similar to that of a traditional KD-tree, except for the selection of the splitting dimension. Instead of looking for the dimension with the greatest variance, the KDA splits the data into two halves, at the median of a randomly selected dimension. This avoids the need to calculate variances and the trees are constructed more efficiently. In addition to allowing greater computational efficiency, a multiple KDA organizes the data into independent tree structures, which is an advantage in a NN search when the number of backtracking is high.

### 3.2 Complexity

When constructing a traditional KD-tree, it needs to locate the dimension with the greatest variation to serve as a splitting dimension. We will analysis how many additions and multiplications it takes to find such dimension. And these operations are exactly saved when a KDA is constructed instead. The variance of $n$ real numbers, $y_1, \ldots, y_n$, can be obtained by

$$\text{var} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n} = \frac{\sum_{i=1}^{n} y_i^{\,2}}{n} - \left(\bar{y}\right)^2 = \frac{\sum_{i=1}^{n} y_i^{\,2}}{n} - \left(\frac{\sum_{i=1}^{n} y_i}{n}\right)^2 . \quad (1)$$

Equation (1) takes ($n$+3) multiplications and (2$n$-1) additions. For $N$ data of $D$-dim, it needs $D$($N$+3) multiplications and $D$(2$N$-1) additions to locate the dimension of the greatest variance. A traditional KD-tree is a balanced binary tree. Without loss of generality, let us assume the

number of data, $N$, is a power of 2. As shown in the **Fig. 1**, the constructed KD-tree will have a total of ( log$N$ + 1) level, i.e., level 0, 1, …, log$N$. On the level $k$, $k$ = 0, 1, …, (log$N$)-1, there are $2^k$ (parent) nodes that it each has $N/2^k$ children. To complete the construction of the tree, each parent node needs to find out the dimension of the greatest variance among its children.

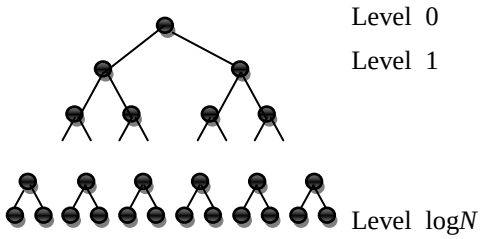



**Fig. 1.** A KD-tree of $N$ leaves assuming $N$ is a power of 2

When there are $N$ data of $D$-dim, to construct a KD-tree, it requires $\sum_{k=0}^{(\log N)-1} 2^k \cdot D \cdot (\frac{N}{2^k}+3)$ and $\sum_{k=0}^{(\log N)-1} 2^k \cdot D \cdot (2 \cdot \frac{N}{2^k}-1)$ that many multiplications and additions. From the simple algebraic equation in (2), and the derivations in (3) and (4), we conclude that it requires an order of O($DN$log$N$) additions and multiplications to construct a traditional KD-tree of $N$ data of $D$-dim. In real applications, it is very common that the number of data $N$ is over thousands. Not to mention if there are multiple KD-trees to be constructed.

$$2^0 + 2^1 + \ldots + 2^i = 2^{i+1} - 1. \tag{2}$$

$$\begin{aligned} \sum_{k=0}^{(\log N)-1} 2^k \cdot D \cdot (\frac{N}{2^k}+3) &= D \sum_{k=0}^{(\log N)-1} (N + 3 \cdot 2^k) \\ &= D(N \log N + 3 \cdot (2^{\log N} - 1)) \\ &= D(N \log N + 3N - 3), \end{aligned} \tag{3}$$

$$\begin{aligned} \sum_{k=0}^{\log N-1} 2^k \cdot D \cdot (2 \cdot \frac{N}{2^k} - 1) &= D \sum_{k=0}^{\log N-1} (2N - 2^k) \\ &= D(2N \log N - (2^{\log N} - 1)) \\ &= D(2N \log N - N + 1). \end{aligned} \tag{4}$$

## 4. Experimental Results and Analysis

The experiments were conducted using MATLAB and C in an environment of UNIX, using a 4G RAM. The test data consisted of synthetic data and SIFT features from real-world images. The synthetic data was randomly generated by uniform distribution and normal distribution. The range for the uniform distribution of data was [-1, 1] and for the normal distribution of

data, the range was $\mu = 0$ and $\sigma = 1$. Query points were randomly generated in the same way.

Tests similar to [8] were performed on the SIFT feature with a dimension of 128. One image (e.g., 'Buddha' and 'Bike' in **Fig. 3**), or a combination of multiple images (e.g., 'Graffiti' in **Fig. 3**) of the SIFT feature points are required to form a test database. The query is selected from a pool that consists of *n* randomly selected feature points from the test database, such that each dimension is disturbed by a Gaussian noise, $\sigma = 0.05$. For a given query, a linear search is first used to find the exact nearest neighbor, as the ground truth. Either KDA or KDR is used with *k* backtrackings to search for the NN. If the NN is found that is same as the ground truth, then this search is a success. The accuracy rate is the average number of successes in 100 repetitions.

When searching for the NN for a given query, the Best-Bin-First method was adopted. Since the method works on multiple trees, it first executes one search for every tree and a single priority queue records all of the distances when traversing the branches, from the roots to the leaf nodes (of every tree). The distance between the query and the medium of the splitting dimension, the location of the branch and the index of trees are recorded in the priority queue. When backtracking occurs, the branch node with the minimum distance is selected. In the following discussion, for simplicity, let *k* be the number of backtrackings, *D* is the dimensionality, *N* is the number of data, KDA*n* is *n* Arbitray KD-trees and KDR*n* is *n* randomized KD-trees. In the following tests, multiple KD-trees are focused especially on 10, 20, 40 tress.

### 4.1 Synthetic data

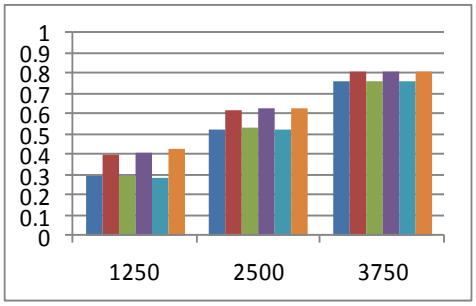
The test used $D = 100$, for $N$=5,000 data & 10,000 data, and $D$=300, for $N$=30,000 data & 60,000 data. The number of backtrackings, *k*, were 25%, 50% and 75% of *N*, respectively. An accuracy of at least 50% is of interest as this is the minimum accuracy required for practical applications. The results are shown in **Fig. 2**. From **Fig. 2**, it is seen that, regardless of the type of data, the accuracy rate never exceeds 50%, if $k = 25\%\ N$. The best performance for $k = 25\%\ N$ is an accuracy rate 0.425, for the KDR40 method in data (uniform, $D$=100, $N$=5000 and uniform, $D$=100, $N$=10,000). In the tests with synthetic data, KDR consistently outperforms KDA. Regardless of data types or data numbers, the average accuracy rates for KDR and KDA, and the improvement of KDR over KDA are summarized in **Table 1**.

**Table 1**. Overall Accuracy Rates of the Synthetic Data for KDR and KDA

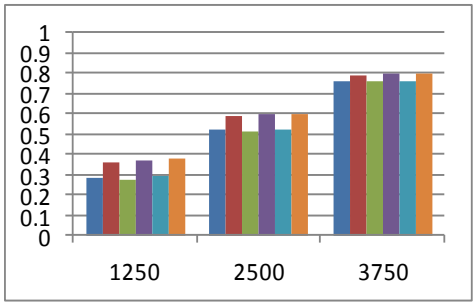
| Number of Backtrackings | KDR | KDA | (KDR-KDA)/KDA |
|---|---|---|---|
| 50% *N* | 60.57% | 51.55% | 17.50% |
| 75% *N* | 79.98% | 75.80% | 5.51% |

#### 4.1.1 Number of Trees

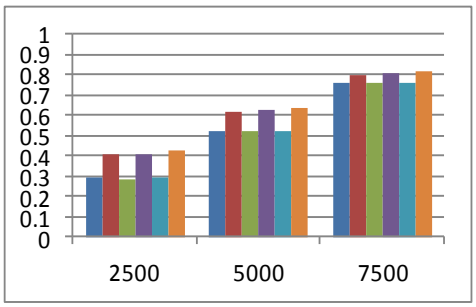
The accuracy rates in different number of trees are compared. The types of data (uniform or normal generated), the number of data, and number of backtrackings (50%*N* and 75%*N*) are considered together. For the KDR multiple tree method, more trees produce better accuracy. However, the increase is quite limited. There are average increases of 0.6% and 0.528%, from KDR10 to KDR20 and from KDR20 to KDR40, respectively. The greater number of trees in a KDA multiple tree does not guarantee greater accuracy. The average rates of increase are 0.019% and -0.024%, from KDA10 to KDA20 and from KDA20 to KDA40, respectively.



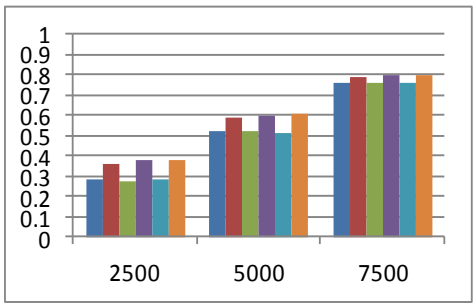

(a) $D$ = 100 and $N$ = 5,000



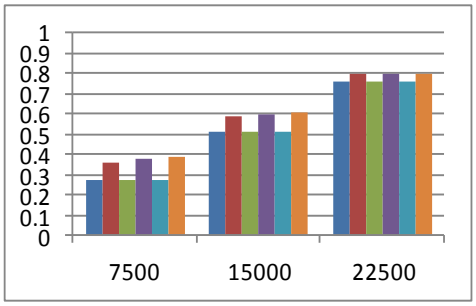

(b) $D$ = 100 and $N$ = 5,000



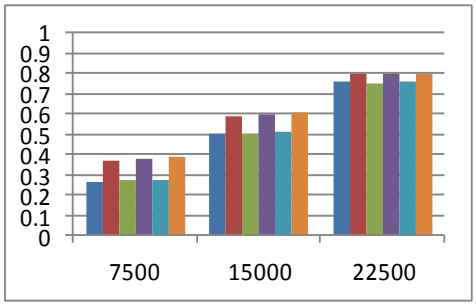

(c) $D$ = 100 and $N$ = 10,000



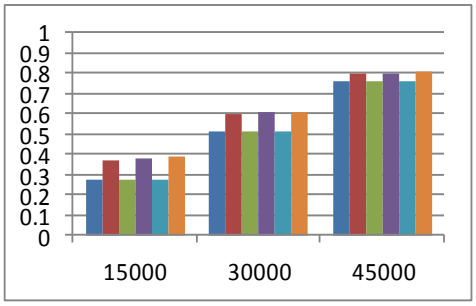

(d) $D$ = 100 and $N$ = 10,000



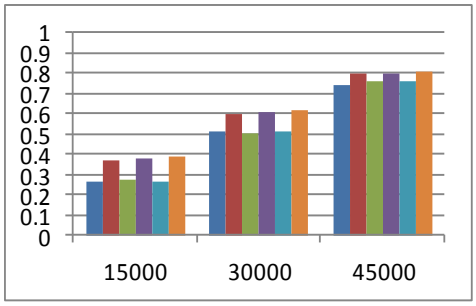

(e) $D$ = 300 and $N$ = 30,000




(f) $D$ = 300 and $N$ = 30,000

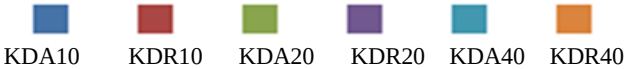



(g) $D$ = 300 and $N$ = 60,000




(h) $D$ = 300 and $N$ = 60,000

**Fig. 2.** Test Results on Synthetic Data (left column (a), (c), (e), (g): uniform-generated and right column (b), (d), (f), (h): Gaussian-generated). The vertical and horizontal axes are for accuracy (average of 100 tests) and number of backtrackings (25%, 50% and 75% of $N$), respectively, where $D$ is the dimension of the data, $N$ is the number of data, and



### 4.1.2 Uniform vs. Normal

The performance using uniform- or normal-generated data is compared, under different tree structures, for the same $k$ (50%$N$ and 75%$N$) and the same number of trees. The expression, **u** : **n** : **t**, the two shaded rows in **Table 2**, indicates that, for a total of (**u**+**n**+**t**) experiments, **u** times that uniform-generated data perform better, **n** times that normal-generated data perform better and there was a tie **t** times. From **Table 2**, uniform-generated data have better accuracy rates than normal-generated data do if KDA is used. However, as in the second row of the **Table 2**, the privilege is minor with an approximately 0.45%. As for KDR, it has better performance when data are uniform-generated and dimension is 100. But it makes no difference in normal-generated or uniform-generated data if the dimension goes up to 300 (as in the last row of the **Table 2**).

**Table 2.**The Performance Comparison on Different Types of Data (see text)

| | $D$ = 100 $N$ = 5000 | $D$ = 100 $N$ =10000 | $D$ = 300 $N$ =30000 | $D$ = 300 $N$ =60000 | Average |
|---|---|---|---|---|---|
| KDA | 6 : 0 : 0 | 6 : 0 : 0 | 6 : 0 : 0 | 6 : 0 : 0 | N/A |
| KDA acc. rate of Unif vs Norm | 0.51% | 0.43% | 0.38% | 0.49% | 0.45% |
| KDR | 6 : 0 : 0 | 6 : 0 : 0 | 1 : 3: 2 | 1 : 2 : 3 | N/A |
| KDR acc. rate of Unif vs Norm | 1.8% | 2.13% | -0.05% | -0.03% | 0.96% |

### 4.2 SIFT feature points

The test used SIFT feature points (dimension $D$ = 128) extracted from the images shown in **Fig. 3** and the results are also summarized in **Fig. 3**. In contrast to use of the synthetic data, KDA consistently outperforms KDR, when real-world data is used. Similarly, regardless of the type of data or tree structures, the accuracy rate never exceeds 50%, if $k$ = 25% $N$. The rest of discussion will focus on the backtracking number to be $k$ =50%$N$ and 75%$N$ only. **Table 3** describes the overall accuracy rates for KDA and KDR methods. It also indicates that the average improvements of KDA over KDR to be 27.68% and 7.61% when $k$ =50%$N$ and 75%$N$, respectively.

The accuracy rates in different number of trees are compared. It did not show any impact of the number of trees on the accuracy rates, in our experiments. For the KDR multiple tree method, there are average increases of -0.27% and 0.03%, from KDR10 to KDR20 and from KDR20 to KDR40, respectively. For the KDA multiple tree method, there are average increases of -0.15% and 0.02%, from KDA10 to KDA20 and from KDA20 to KDA40, respectively. Furthermore, regardless of the number of backtracking ($k$ =50%$N$ or 75%$N$), from **Fig. 3**, the average accuracy rates for KDR and KDA with respect to the number of trees are summarized in **Table 4**. To conclude the above discussion, using 10 trees for KDR and KDA is a moderate choice. To investigate the number of backtrackings relating to the performance, a more detailed comparison of KDA10 & KDR10 is given in the ensuing discussion.

**Table 3.** Overall Accuracy Rates of The SIFT Data for KDR and KDA

| Number of Backtrackings | KDR | KDA | (KDA-KDR)/KDR |
|---|---|---|---|
| 50% $N$ | 46.40% | 59.24% | 27.68% |
| 75% $N$ | 74.43% | 80.10% | 7.61% |

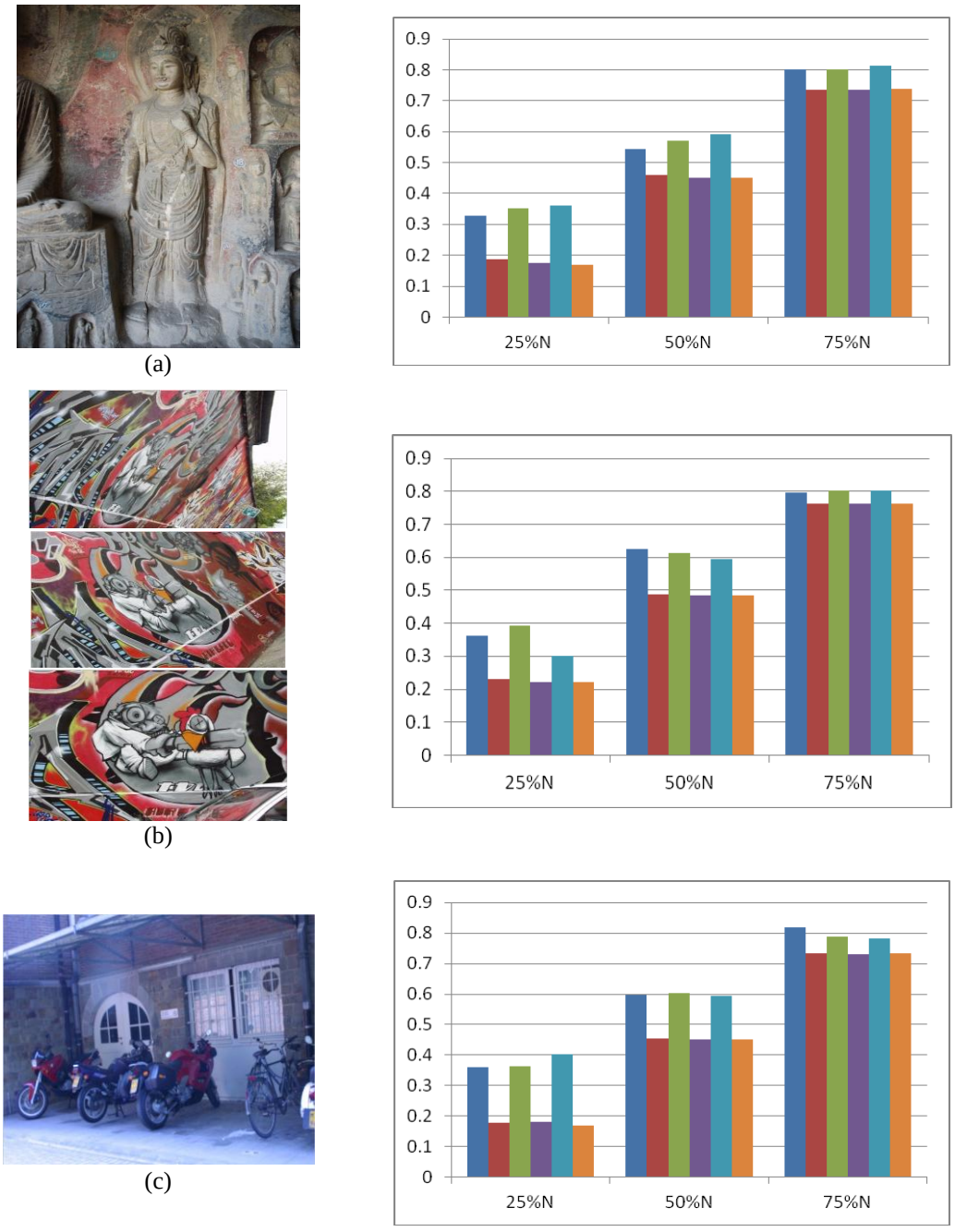


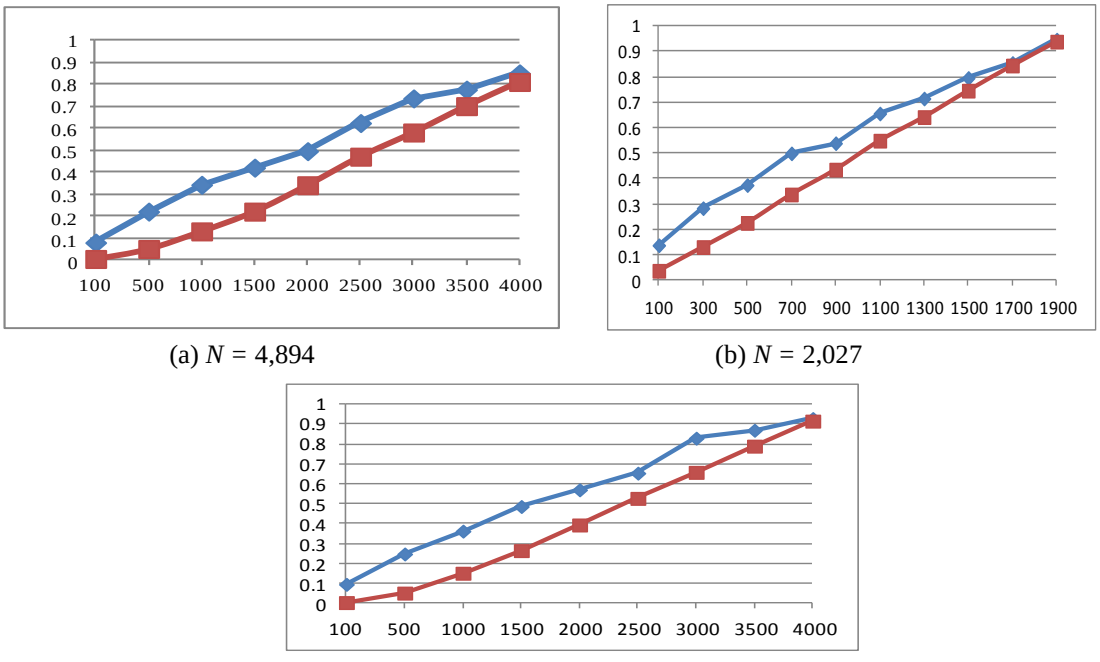

**Fig. 3.** Test Results on SIFT feature points (all the feature points are of dimension 128, $N$ is the number of SIFT features points) Left column shows the test images. Right column shows the accuracy rates of NN search results. The vertical and horizontal axes are for accuracy (average of 100 tests) and number of backtrackings in terms of the data number, respectively, where the images from top to bottom are (a) "Buddha" ($N$ = 4,894), (b) "Graffiti" ($N$ = 5,801), (c) "Bike" ($N$ = 4,359), and

**Table 4.** Accuracy Rates of the SIFT Data for KDR and KDA With Respect to the Number of Trees

| Methods | 10 trees | 20 trees | 40 trees |
|---|---|---|---|
| KDR | **60.58%** | 60.32% | 60.35% |
| KDA | **69.77%** | 69.61% | 69.66% |

### 4.3 KDA10 vs. KDR10

The performances on KDA10 and KDR10 were further compared. From **Fig. 4**, it is seen that the accuracy rate of KDA10 is 50% or more, when $k \approx 37\%N$, while KDR requires that $k \approx 51\%N$. The accuracy rate of KDA10 is 70% or more, when k ≈ 62%*N*, while KDR requires that $k \approx 72\%N$. The difference in their accuracy is not so different for $k$ to be approximately 80%*N* - 90%*N*. However, a number of backtrackings that is 80% of the data amount or more defeats the purpose of multiple KD-trees, which is a lower number of backtrackings and a high accuracy rate. For feature matching, the accuracy rate must be at least 70%. In the next experiment, we apply feature matching by KDA10 in image stitching with the number of backtrackings to be $k \approx 60\%N$.




(a) $N$ = 4,894 (b) $N$ = 2,027

(c) $N$ = 4,359

**Fig. 4.** The accuracy comparison on SIFT feature points by KDA10 and KDR10. The test images from top to bottom are (a) "Buddha" ($N$ = 4,894), (b) "Graffiti-1" ($N$ = 2,027), (c) "Bike" ($N$ = 4,359), Test images are the same with the Fig.3 except "Graffiti-1" is the first image from the set of "Graffiti" images. The vertical and horizontal axes are for accuracy (average of 100 tests) and number of backtrackings, where

### 4.4 Image Stitching Application

In accordance with the previous conclusion, KDA10 is more suited to feature matching on image stitching, when the number of backtrackings is 60% of the data amount, as suggested. The big Foku Buddha images from Huang-Ze Temple, Guangyuan City, Sichuan, China, with sizes 2848×4288, 2592×3872 were used; one as the query image and the other as the target image. $q$ Query points were manually picked from query images and then each selected point automatically locates $r$ nearest SIFT feature points, generating a total $(r \cdot q)$ query SIFT feature points. Using this method, points can be selected that are located in both the query and target images and feature matching can be restricted to $(r \cdot q)$ points only. In this experiment $q = r = 10$, i.e., a total of 100 feature matches were implemented. RANSAC was then applied to these 100 matched pairs, to solve a homography matrix between these two images. Using this homography matrix, the two images were stitched into one. The results are shown on **Fig. 5**.

(a) (b) (c)

**Fig. 5.** Image stitching using KDA10 for feature point matching. 100 NN are located by KDA10 on the target images corresponding to 100 query points of the query images. To stitch two images, techniques of RANSAC and homography are used where the images from left column to right column are (a) the query images, (b) the target images and (c) the stitched images.

## 5. Conclusion

This paper presents a KDA method for identification of a NN in a large quantity of high-dimensional data. Comparing to Lowe's KDR method [7] for the construction of one tree for $N$ data, with dimensionality $D$, KDR requires extra $O(DN\,logN)$ additions and $O(DN\,logN)$ multiplications. The proposed method was tested using synthesized, high-dimensional data and real-world SIFT data, both of which had large quantities of data. The experiments indicated that the proposed method outperforms the KDR method for the SIFT data. However, the use of synthesized, high-dimensional data does not produce the same result. It is suggested that one reason for the different results yielded using the synthesized data and the real-world data is the notorious nature of dimensionality. Beyer *et al*. [10] indicated that distances to near and far neighbors become more similar, as the dimensionality, $D$, of the data increases (i.e., loss of relative contrast):

$$\lim_{D\to\infty} \frac{dist_{\max} - dist_{\min}}{dist_{\min}} \to 0. \tag{5}$$

Thus, it is futile to use conventional similarity measurement to find the NN in high-dimensional data. Recently, some studies have noted that the difference in distances between pairs diminishes with increasing dimensionality, which severely hampers all distance-based algorithms [11], [12].

Finally, the KDA method was used for image stitching. With 100 query points and the number of backtrackings limited to 60% of the total data amount, the stitching result is satisfactory.

## Acknowledgment

Authors wish to thank the Institute of History and Philology, Academia Sinica, Taiwan, R.O.C., for providing the big Foku Buddha images of Huang-Ze Temple, Guangyuan City, Sichuan, China. In addition, authors would like to thank the grant support from National Science Council (NSC99-2221-E-032-060) of Taiwan, R.O.C.

## References

[1] A. Bosch, A. Zisserman and X. Munoz, "Image classification using random forests and ferns," in *Proc. of IEEE International Conference on Computer Vision*, pp. 1-8, 2007. Article (CrossRef Link)

[2] S. Lazebnik, C. Schmid and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2006. Article (CrossRef Link)

[3] D. Lowe, "Distinctive image features from scale invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. Article (CrossRef Link)

[4] J. H. Freidman, J. L. Bentley and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, no.3, pp. 209–226, 1997. Article (CrossRef Link)

[5] J. L. Bentley, " Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975. Article (CrossRef Link)

[6] C. Silpa - Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proc.*

*of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008. Article (CrossRef Link)

[7] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. of International Conference on Computer Vision Theory and Applications* (VISAPP), Lisbon, Portugal, Feb. 2009. Article (CrossRef Link)

[8] P. Wu, S. C.H. Hoi, N. D. Dung and H. Ying, "Randomly projected KD-Trees with distance metric learning for image retrieval," in *Proc. of International Conference on MultiMedia Modeling*, Taipei, Taiwan, pp. 371–382, 2011. Article (CrossRef Link)

[9] S. H. Yen, C. Y. Shih, T.K. Li and H. W. Chang, "Applying multiple KD-Trees in high dimensional nearest neighbor searching," *International Journal of Circuits, Systems and Signal Processing*, vol. 4, no. 4, pp. 153–160, 2010. Article (CrossRef Link)

[10] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft, "When is "nearest neighbor" meaningful?" in *Proc. of International Conference on Database Theory*, *LNCS*, vol. 1540, pp.217–235, Springer, Heidelberg, 1999. Article (CrossRef Link)

[11] M. Houle, K. Kriegel, P. Kroger, E. Schubert and A. Zimek, "Can shared-neighbor distances defeat the curse of dimensionality?" *Scientific and Statistical Database Management*, *LNCS*, vol. 6187, pp. 482-500, Springer, Heidelberg, 2010. Article (CrossRef Link)

[12] B. Pagel, F. Korn and C. Faloutsos, "Deflating the dimensionality curse using multiple fractal dimensions," in *Proc. of IEEE International Conference on Data Engineering*, pp. 589–598, 2000. Article (CrossRef Link)

**Shwu-Huey Yen** is an associate professor in the department of Computer Science and Information Engineering (CSIE), Tamkang University, New Taipei City, Taiwan. She is also an author of over 50 journal papers and conference papers. Her academic interests are signal processing, multimedia processing and medical imaging.

**Ya-Ju Hsieh** received both Bachelor degree and Master degree in CSIE department, Tamkang University, on 2008 and 2012, respectively. Currently, she works as an Associate System Engineer in Shin Kong Life Insurance Co., Ltd., Taipei, Taiwan.