Scalable k-NN graph construction[*]

Jingdong Wang[†] Jing Wang[‡] Gang Zeng[‡] Zhuowen Tu[† §] Rui Gan[‡] Shipeng Li[†]
[†]Microsoft Research Asia [‡]Peking University
[§]Lab of Neuro Imaging and Department of Computer Science, UCLA
cis.wangjing@pku.edu.cn {g.zeng, rui_gan}@ieee.org
{jingdw, zhuowent, spli}@microsoft.com

Abstract

The k-NN graph has played a central role in increasingly popular data-driven techniques for various learning and vision tasks; yet, finding an efficient and effective way to construct k-NN graphs remains a challenge, especially for large-scale high-dimensional data. In this paper, we propose a new approach to construct approximate k-NN graphs with emphasis in: efficiency and accuracy. We hierarchically and randomly divide the data points into subsets and build an exact neighborhood graph over each subset, achieving a base approximate neighborhood graph; we then repeat this process for several times to generate multiple neighborhood graphs, which are combined to yield a more accurate approximate neighborhood graph. Furthermore, we propose a neighborhood propagation scheme to further enhance the accuracy. We show both theoretical and empirical accuracy and efficiency of our approach to k-NN graph construction and demonstrate significant speed-up in dealing with large scale visual data.

1 Introduction

The fields of machine learning, computer vision, data mining, bioinformatics, and internet search have witnessed a great success of applying data-driven techniques, in which neighborhood graphs are widely adopted. Some examples include image and object organization [18, 31, 32], object retrieval [20, 33], face synthesis [23], shape retrieval and approximate nearest neighbor search [1, 35], manifold learning and dimension reduction [2, 34, 37], and other machine learning tasks [26, 27, 43, 45, 47].

Two types of neighborhood graphs are often used: k-nearest-neighbor (k-NN) graphs (a node is connected to its k nearest neighbors) and ϵ-nearest-neighbor (ϵ-NN) graphs (two nodes are connected if their distance is within ϵ). The ϵ-NN graph is geometrically motivated [5], but it easily results in disconnected components [2]. Hence, it is not suitable in many situations [7]. In contrast, k-NN graphs have been shown to be especially useful in practice [7]. Therefore, this paper focuses on the problem of constructing k-NN graphs.

A naive solution to neighborhood graph construction is exhaustively comparing all pairs of points, which takes $\Theta(n^2d)$ time with n denoting the number of data points and d denoting the dimensionality. This is

[*]A conference version appeared in [44]

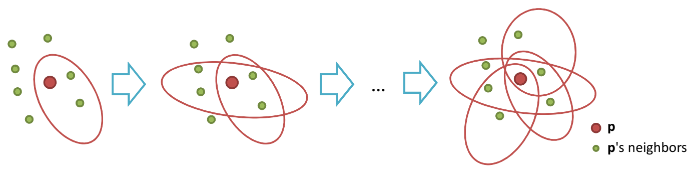
prohibitively slow and unsuitable for large-scale applications. Early research efforts have been conducted to construct exact k-NN graphs [4, 9, 29, 40]. However, the time complexity of those methods either grows exponentially with respect to the dimensionality, or grows super-linearly with respect to the number, which makes them impractical for large scale and high-dimensional problems. Nowadays most research efforts have been turned to approximate neighborhood graph construction.

A straightforward solution is to apply approximate nearest neighbor search methods to construct neighborhood graphs. One can first build an indexing structure to organize the data points, and then regard each data point as a query and find approximate NNs by searching the structure. An example approach uses locality sensitive hashing to help construct neighborhood graphs [39]. However, neighborhood graph construction is generally simpler than nearest neighbor search because any NN search algorithm can solve neighborhood graph construction, but not vice versa [7, 39]. In other words, neighborhood graph construction only cares about the existing data points, while NN search has to consider out-of-sample points.

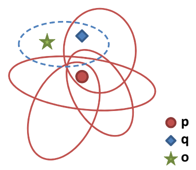
Approximate neighborhood graph construction methods have been proposed by following the divide-and-conquer methodology [4]. The process consists of two stages: recursively divide the data set into small subsets and merge the neighborhood graphs from subsets. Using the overlapped divisions [7] or the third subset [39], the neighborhood graphs from all the subsets are merged together to get an approximate neighborhood graph. Due to the overlapped divisions, both the two approaches suffer from the high time complexity [7]. The time cost relies much on the overlapping ratio, and tends to be quadratic with respect to the number of points for highly accurate neighborhood graphs.

In this paper, we propose a new approach to construct approximate k-NN graphs with emphasis in efficiency and accuracy and justify our approach in theory and empirical experiments with large scale vision data. We first present a multiple random divide-and-conquer approach to construct an approximate neighborhood graph. We randomly and hierarchically partition the data points into subsets so that the neighboring points have a high probability to lie in the same subset, and connect each point with its nearest neighbors within each subset to get a base approximate neighborhood graph. This random partition process is repeated several times to increase the chance that neighboring points are connected in at least one random partition. The multiple random partitions can be viewed as a way to exploit overlaps that are also utilized in [7, 42], but differently our approach is more efficient and the time cost grows only linearly with respect to the number of random divisions.

Furthermore, we propose a neighborhood propagation scheme, i.e., propagating the local approximate neighborhoods to the wider area, to achieve a more accurate neighborhood graph in a higher speed. We observe that after several repetitions of random partitions most of the neighboring relationships generated by the new random partition have already appeared in the previous random partitions and that the discovered approximate neighborhoods of true neighboring points will instead have relatively large overlaps. The two points suggest to propagate the local neighborhood to a wider range, expecting a higher speed in connecting neighboring points. Experimental results show that the multiple division scheme is superior over existing neighborhood construction algorithms in terms of speed and accuracy, neighborhood propagation can further improve the performance a lot and the improvement is more significant when requiring larger neighborhoods.



(a)



(b)

Figure 1: (a) As the number of random divisions increases, more neighbors of $\mathbf{p}$ are covered. (b) Illustration of neighborhood propagation from $\mathbf{p}$ to $\mathbf{o}$ by accessing the neighborhood of $\mathbf{q}$

.

2 Related work

The construction of exact k-NN graphs has been extensively studied in the literature and a number of algorithms were proposed to avoid the high complexity $O(n^2)$. In [4], a divide-and-conquer method taking $O(n \log^{d-1} n)$ time was presented, while an algorithm with expected $O(c^d \log n)$ time (for some constant c) was introduced in [9] and a worst-case $O((c'd)^d n \log n)$ time (for some constant c') algorithm was proposed in [40]. The time complexity of these methods grows exponentially with data dimension d, which makes them quite inapplicable in high-dimensional problems. Recently, [29] proposed a method, which empirically requires $O(n^{1.27})$ distance calculations in low-dimensional cases and $O(n^{1.90})$ calculations in high-dimensional cases. This algorithm works well on low-dimensional data, but becomes inefficient in high-dimensional cases. In spite of a rich previous literature, no efficient algorithm for high-dimensional exact k-NN graphs has been proposed. Thus, research efforts are moved to approximate neighborhood graph construction.

A straightforward solution of constructing an approximate k-NN graph is to apply a nearest neighbor search algorithm, in which an indexing structure is usually first made to organize the data points and a search method based on the structure is adopted to handle the upcoming queries. Representative examples of search methods are partition-tree-based methods such as kd-trees [3, 16, 21, 22], and random projection trees (rp-tree) [11], and hashing based methods such as locality sensitive hashing [12, 39]. However, as pointed in [7], the existing search methods generally suffer from an unfavorable trade-off between the complexity of the indexing structure and the accuracy of the search. Moveover, the search methods generally ignore the fact that in graph construction, each query must be one of the data points. As a consequence, unnecessary efforts are put on giving a good result for general queries, which makes them not so efficient comparing to other algorithms which focus only on graph construction.

Some approximate neighborhood graph construction methods were proposed recently by following the divide-and-conquer methodology [4]. The approach in [7] divides the data points into two or three overlapped subsets with a predefined overlapping ratio, and unites the subgraphs constructed from subsets together, which is followed by a refinement step, inspecting the neighbors of the neighbors (or the second-order-neighbors). Our neighborhood propagation is related to this refinement step, but very different because the refinement does not discriminate the points in the second-order neighborhood and may check some distant points. In contrast, our neighborhood propagation inspects the points within higher-order neighborhood in the best-first order, which is more reasonable. The approach in [42] divides the data into two non-overlapped subsets and additionally samples another subset which overlaps with both the above two subsets, and finally merges the three graphs together. The accuracy of both methods rely much on the overlapping ratio, but the time cost grows almost exponentially with the ratio, which makes them difficult to balance the efficiency and the accuracy especially in large scale problems. The approach proposed in [22] is very related to our approach, but is clearly different from ours as it adopts randomly rotated kd-trees to perform an approximate search and then performs a second-order neighborhood expansion.

There are some other approaches of neighborhood graph construction. [10] proposed a parallel fast algorithm using Morton ordering, but the method works well only on low-dimensional data. [17] presented an incremental way of building neighborhood graphs, but the algorithm is mainly for relative neighborhood graph and is inefficient in large scale problems. [27] gave an extensive theoretical analysis on how to choose a proper k in k-NN graphs for a better performance in real applications, which is a problem different from our approach.

The multiple random division scheme has been exploited in other problems. Random kd-trees are constructed in [36] to boost the indexing efficiency. Random forest [6] is developed to build an ensemble classifier that consists of many decision trees to improve the classification performance. Differently, this paper proposes to adopt this multiple random division technique to build a neighborhood graph.

3 Approach

Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \ \mathbf{x}_2, \ \cdots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^d$, the goal is to build a k-nearest-neighbor graph $G = (V, E)$. The problem is formally defined as follows.

Definition 1 (k-NN Graph)**.** *G is a directed graph, where $V = \mathcal{X}$, and $\langle \mathbf{x}_i, \mathbf{x}_j \rangle \in E$ if and only if $\rho(\mathbf{x}_i, \mathbf{x}_j)$ is among the k smallest elements of the set $\{\rho(\mathbf{x}_i, \mathbf{x}_l) | l = 1, \dots, i-1, i+1, \dots, n\}$, where ρ is a metric such as Euclidean distance and cosine distance.*

We first present a multiple random divide-and-conquer approach to build base approximate neighborhood graphs and then ensemble them together to achieve an approximate neighborhood graph. Then we introduce a neighborhood propagation technique to propagate the local neighborhoods to a wider range in order to achieve a more accurate neighborhood graph in a higher speed.

3.1 Multiple random divide-and-conquer

A base approximate neighborhood graph is an unconnected graph, in which each subgraph corresponds to a group of possibly neighboring points. In other words, a base approximate neighborhood graph corresponds to a partitioning of the data points. We adopt the divide-and-conquer methodology to recursively partition the points into small subsets, forming a random partition tree. To make nearby points lie in the same subset, we can use hyperplanes or hyperspheres to partition the data set. Specifically, we divide the point set $\mathcal{X}$ into two nonoverlapped subsets, $\mathcal{X}_l$ and $\mathcal{X}_r$, so that $\mathcal{X}_l \cup \mathcal{X}_r = \mathcal{X}$ and $\mathcal{X}_l \cap \mathcal{X}_r = \emptyset$, and recursively conduct the division process on the subsets until the cardinality of a subset is smaller than a fixed value g. Then a brute-force manner is adopted to build a neighborhood graph (subgraph) for each subset of points. Different from the division with the overlapping in [7, 39], this process is very efficient as the partitioning process takes $O(nd \log n)$ and building subgraphs only takes $O(ndg)$.

A single random division yields a base approximate neighborhood graph containing a serial of isolated subgraphs, and it is unable to connect a point with its neighboring points lying in different subgraphs. Thus, we propose to exploit multiple random divisions to find more neighbors for each point. Considering

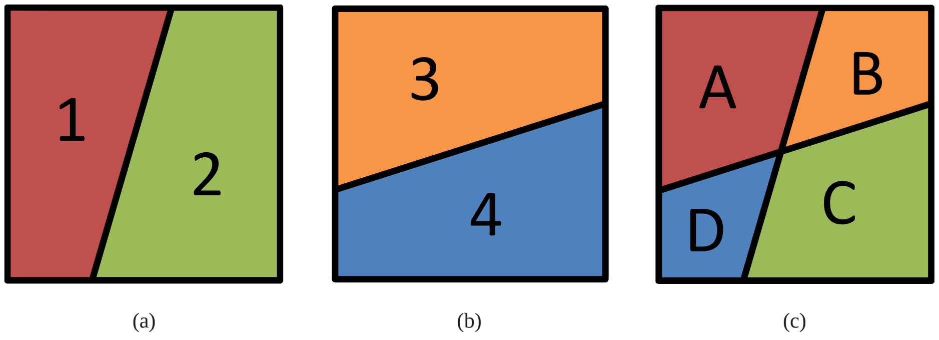


Figure 2: Overlaps of subsets in different divisions served as bridges to connect isolated subgraphs.

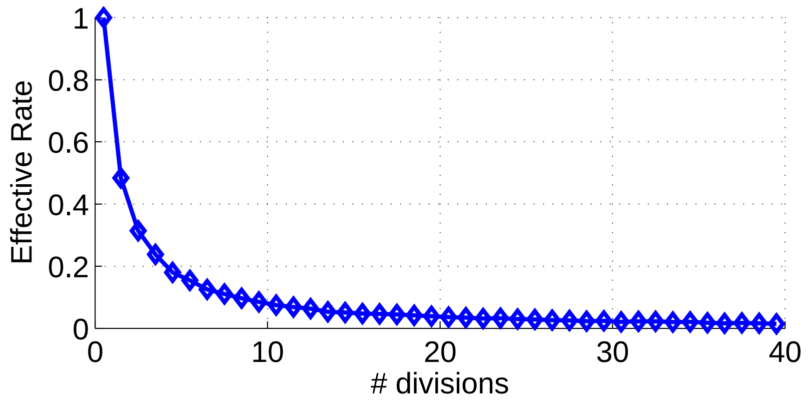


Figure 3: Effective rate drops as the increase of the number of random divisions

.

a point $\mathbf{p}$, each random division can be interpreted as enumerating a set of neighboring points around $\mathbf{p}$. As illustrated in Fig. 1(a), the neighborhood of $\mathbf{p}$ identified in each division is represented by a red ellipse. We denote the set of points in the neighborhood of $\mathbf{p}$ in the m-th division by $\mathcal{N}_{\mathbf{p}}^m$, and the union of multiple sets of neighboring points is then written as $\tilde{\mathcal{N}}_{\mathbf{p}}^m = \cup_{i \leqslant m} \mathcal{N}_{\mathbf{p}}^i$. By increasing the number of random divisions, the union $\tilde{\mathcal{N}}_{\mathbf{p}}^m$ will cover more true neighbors of $\mathbf{p}$, which are represented by small green points in Fig. 1(a), so that the quality of the combined neighborhood graph is improved. We denote the base approximate neighborhood graphs resulted from each division by $\{G_1, \cdots, G_M\}$, and the adjacent list by $Adj^m[\mathbf{x}_i]$ in G_m for point $\mathbf{x}_i$. The combination is achieved efficiently by uniting its adjacent lists $\{Adj^m[\mathbf{x}_i]\}_{m=1}^M$ together and retaining k nearest neighbors.

In essence, uniting the neighborhood subgraphs exploits the overlaps among the subgraphs from multiple random divisions. Let's consider a situation of two random divisions illustrated in Fig. 2. In Fig. 2(a) and Fig. 2(b), the first division yields two isolated subsets S_1 and S_2, and the second division yields two isolated subsets S_3 and S_4. In Fig. 2(c), one can see that S_3 overlaps with S_1 over S_A and S_3 overlaps with S_2 over S_B. This implies that S_3 serves as a bridge to connect isolated subgraphs constructed from S_1 and S_2. S_4 also serves as the same role. In turn, S_1 and S_2 can also be regarded as the same roles to connect subgraphs over S_3 and S_4. In the implementation, the data points are divided into many parts for many times so that there are sufficient overlaps to make better connections among subgraphs.

3.2 Neighborhood propagation

Let's consider the point $\mathbf{p}$ again. As discussed before, by increasing the number of random divisions, the union $\tilde{\mathcal{N}}_{\mathbf{p}}^{m}$ becomes larger and covers more true neighbors of $\mathbf{p}$. Although this increases the accuracy of the neighborhood graph, it also makes the contribution of a new random division $\tilde{\mathcal{N}}_{\mathbf{p}}^{m} - \tilde{\mathcal{N}}_{\mathbf{p}}^{m-1}$ smaller. In other words, the progress toward the true neighborhood graph becomes slower when m becomes larger. This is validated by the experimental result shown in Fig. 3. The effective rate for the m-th division is defined as $r_m = \frac{\Sigma_{\mathbf{p}}|\tilde{\mathcal{N}}_{\mathbf{p}}^{m} - \tilde{\mathcal{N}}_{\mathbf{p}}^{m-1}|}{\Sigma_{\mathbf{p}}|\tilde{\mathcal{N}}_{\mathbf{p}}^{m}|}$, which indicates the contribution made by the m-th division. On the other hand, suppose $\mathbf{q}$ is a point in the previously-identified neighborhood of $\mathbf{p}$, and $\mathbf{p}$ and $\mathbf{q}$ has a common true neighboring point $\mathbf{o}$. Then with the increase of the number of random divisions, the probability that $\mathbf{o}$ is identified as a neighboring point of $\mathbf{p}$ or $\mathbf{q}$ increases, i.e., $P(\mathbf{o} \in \mathcal{N}_p \ or \ \mathbf{o} \in \mathcal{N}_q)$ becomes larger. This suggests a way to find the neighboring point $\mathbf{o}$ for $\mathbf{p}$ ($\mathbf{q}$), by accessing its neighbor $\mathbf{q}$ ($\mathbf{p}$) and expanding the neighborhood of $\mathbf{q}$ ($\mathbf{p}$). In the situation illustrated in Fig. 1(b), $\mathbf{o}$ has not been identified as $\mathbf{p}$'s neighbor, but it has been identified as $\mathbf{q}$'s neighbor through the subset denoted by the blue dashed ellipse. Consequently, a neighborhood propagation path from $\mathbf{p}$ to $\mathbf{o}$ through $\mathbf{q}$ is accessible.

In light of the above analysis, we present a neighborhood propagation scheme. For each point $\mathbf{p}$, we access its previously-identified neighborhood and conduct more accesses gradually by propagating the neighborhoods in a best-first manner. Specifically, we first expand $\mathbf{p}$'s neighborhood, and push all the neighbors into a priority queue, in which the point the nearest to $\mathbf{p}$ will be positioned at the top. Then we iteratively pop the top point from the queue and push all its unvisited neighbors into the queue. The best-first strategy makes true neighbors be first discovered with higher probability. The propagation process stops when the queue is empty or the maximum number of visited points, T, is reached. During the process, all the visited points are considered as candidate neighbors of $\mathbf{p}$ in which the better ones will replace the current neighbors. The propagation process is performed for all the points. The process is very efficient and the cost is linear with n and d.

3.3 Analysis

In the following, we present theoretic analysis to show why random divisions and neighborhood propagation work well and complexity analysis of our approach. The detailed proofs can be found in Section 6.

Lemma 1. *Suppose a random hyperplane partitions the data points so that a certain point $\mathbf{x}_i$ and one of its true neighbors $\mathbf{x}_j$ have the probability P_{ij} to be in the same subset. Then with a single random partition tree, $\mathbf{x}_j$ is discovered as $\mathbf{x}_i$'s neighbor with the probability P_{ij}^{h}, where h is the depth of the tree. With L random partitions, $\mathbf{x}_j$ is discovered as $\mathbf{x}_i$'s neighbor with the probability $1-(1-P_{ij}^{h})^{L}$.*

Lemma 2. *The probability that the neighboring relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$ is discovered by the L-th random partition tree but not discovered by the previous $L-1$ random partition trees is $P_{r_L} = (1 - P_{ij}^{h})^{L-1} P_{ij}^{h}$.*

This lemma indicates that the true NN points newly found in the L-th partition tree become fewer when L increases and presents a theoretic justification of Fig. 3.

Lemma 3. *Considering two neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ having the same neighboring point $\mathbf{x}_n$, after $L-1$ partition trees, the probability that $\mathbf{x}_j$ can be discovered as the neighbor of $\mathbf{x}_i$ through $\mathbf{x}_n$ is $P_{p_L} = (1-(1-P_{in}^{h})^{L-1})(1-(1-P_{jn}^{h})^{L-1})$.*

It can be easily seen that P_{p_L} becomes larger when L increases. This property can be generalized to the case that $\mathbf{x}_j$ is discovered as the neighbor of $\mathbf{x}_i$ through t intermediate points (a longer neighborhood propagation path) and the probability is $P_{p_L} = \prod_{k=0}^{t}(1-(1-P^h_{n_k n_{k+1}})^{L-1})$, with $n_0 = i$ and $n_{t+1} = j$.

Let's compare the probabilities of discovering a new true nearest neighbor from a partition tree and neighborhood propagation. Under the condition that $\mathbf{x}_i$ and $\mathbf{x}_j$ are not discovered as neighbors in previous $L-1$ trees, we have conclusions: (1) The probability P^h_{ij} to discover the neighboring relationship in the next tree stay the same as L grows; (2) The probability P_{pL} to discover the relationship by propagation keeps increasing; (3) The probability with the next tree will be smaller than that with propagation when L reaches one constant. This shows that neighborhood propagation can speed up neighborhood discovery. It should be noted that neighborhood propagation is even more advantageous because the above analysis does not cover all the cases, for instance, when $\mathbf{x}_i$ and $\mathbf{x}_j$ have more same neighbors. In summary, we can have the following theorem.

Theorem 1. *Suppose $P = \min\{P_{ij} | \langle \mathbf{x}_i, \mathbf{x}_j \rangle \in E(G)\}$, where G is the exact k-NN graph. With L random divisions and a first-order neighborhood propagation, a true k-NN point, $\mathbf{x}_j$, of $\mathbf{x}_i$ is discovered with at least the probability $1-(1-P^h)^{2L}(2-(1-P^h)^L)$ under the assumption that $\mathbf{x}_i$ and $\mathbf{x}_j$ have at least one same neighboring point.*

The following discusses the time complexity. Our approach takes $O(Mdn\log n)$ time in multiple random divisions, where M denotes the number of divisions, and $O(Tdn\log T)$ time in neighborhood propagation, where T denotes the maximum number of visited points. In a large scale and high-dimensional problem, M and T are relatively very small so that the whole complexity of both the multiple random divisions and the combined method can be written as $O(dn\log n)$. The algorithm presented in [42] is denoted by VirmajokiF04 and its complexity is reported as $O(d^2 n^{1.58}\log n)$. The two algorithms in [7], which are named as Glue and Overlap, take $O(dn^{1.22}\log n)$ time and $O(dn^{1.36}\log n)$ time when the overlapping ratio α is set as 0.2 that is suggested in [7]. By comparison, the theoretic time complexity of our approach is smaller that those of other methods.

3.4 Algorithm details

Random division. Our implementation chooses the random principal directions to perform random divisions to make the diameter of each subset small enough. It is theoretically shown in [41] that the principal-direction-based way to hierarchically partition the points can guarantee that the diameters of the subsets are reduced quickly. This implies that the points in the same subset tend to be nearer to each other. The principal directions are obtained by using principal component analysis (PCA). To generate random principal directions, rather than computing the principle direction from the whole subset of points, we compute the principal direction over the points randomly sampled from each subset. In our implementation, the principle direction is computed by the Lanczos algorithm [25]. Compared with other space partitioning ways, e.g., random projections [11], our experiments show that the principal-direction-based way is more efficient and effective.

Speedup. In the process of multiple random divisions and neighborhood propagation, the distances between a pair of points may be computed more than once, which would cost too much especially for high-dimensional cases. To avoid the re-computations, a hash table is adopted to store the pairs of points whose

Table 1: The running time of the brute-force method.

	Caltech 101	Ukbench	Imagenet	TinyImage
Time (min)	2034	2067	5737	5823

distances have been computed. When requiring the distance of a pair of points, we check if their distance has been evaluated through the hash table. The time overhead is very low because the operations over the hash table cost $O(1)$, while the re-computation cost is $O(d)$.

Besides, we introduce a pairwise updating scheme. This is motivated by the observation that it is highly possible that $\mathbf{u}$ is also among the k nearest neighbors of $\mathbf{v}$ if $\mathbf{v}$ is among the k nearest neighbors of $\mathbf{u}$. When considering $\mathbf{v}$ to update the neighborhood of $\mathbf{u}$, we also immediately use $\mathbf{u}$ to update the neighborhood of $\mathbf{v}$.

4 Experiments

Data sets. We demonstrate the proposed neighborhood graph construction algorithm over SIFT features and GIST features. The SIFT features are collected from the Caltech 101 data set [14] and the recognition benchmark images [28]. We extract maximally stable extremal regions (MSERs) for each image, and compute a 128-dimensional SIFT feature for each MSER. For each image set, we randomly sample $1000K$ SIFT features as our data set.

Besides, we conduct the experiments on the TinyImage set [38] and the ImageNet data [13] to justify our approach. Similar to [24], we use a global GIST descriptor to represent each image, which is a 384-dimensional vector describing the texture within localized grid cells. The dimension of the GIST descriptor is higher than that of the SIFT feature, and hence to achieve high accuracy is more difficult and challenging. We also sample $1000K$ GIST features from each of the two data sets.

Evaluation scheme. We adopt the accuracy measurement to evaluate the quality of the approximate graph. The accuracy of an approximate k-NN graph G' (with regard to the exact neighborhood graph G) is defined as $\mathrm{accuracy}(G') = \frac{|E(G')\cap E(G)|}{|E(G)|}$, where $E(\cdot)$ denotes the set of direct edges in the graph and $|\cdot|$ denotes the cardinality of the set. The accuracy is within the range $[0, 1]$, and a higher accuracy means a better graph. The exact neighborhood graph G is computed by the brute-force method, and the running time on four data sets are given in Tab. 1.

We report the results of our approaches based on multiple random divide-and-conquer and the combination of it with subsequently followed neighborhood propagation. The recursive division is repeated till the cardinality of a subset is smaller than 500. The neighborhood propagation is triggered when the effective rate of the m-th random division, r_m, defined in Sec 3.2, is less than a threshold which is set as 0.05. With the help of the hash table, the effective rate can be easily calculated and will not affect the efficiency of the algorithm.

By comparison, we present the performances of the three divide-and-conquer approaches in [7, 42], (named ChenFS09Glue, ChenFS09Overlap and VirmajokiF04, respectively) and the multisorting algorithm in [39] (named UnoST09) that leverages locality sensitive hashing. The results of ChenFS09Glue and ChenFS09Overlap in [7] are reported by running the online implementation available[1]. We implemented

[1] http://www.mcs.anl.gov/~jiechen/research/software/knn.tar.gz

Table 2: Local label consistency ratio for face organization.

	$\mathcal{G}_0$	$\mathcal{G}_1$	$\mathcal{G}_2$
Local label consistency ratio	0.564	0.561	**0.689**

other algorithms and adjusted the parameters to make the performance as good as possible. We also present the performance of building neighborhood graphs by searching random kd-trees, which performs better than other partition trees as the construction of kd-trees is very cheap. All algorithms are run on a 2.66GHz desktop PC with a single thread.

Results. The performance comparison is shown in Fig. 4. The horizontal axis corresponds to construction time (in seconds), and the vertical axis corresponds to the accuracy. We test all the above algorithms on the four data sets described before, and compute the accuracy based on different numbers of neighbors, denoted as k. Each column of Fig. 4 corresponds to one data set and each row corresponds to a choice of k. The performance of multiple random divisions in our approach is shown as the blue circle line in each figure, and the performance of the combination of it with the neighborhood propagation is shown as the red circle line.

From the results, we can clearly see the superiority of our algorithms over other algorithms. In Caltech 101, the approach of multiple random divisions can achieve an accuracy of 90% in the 1-NN graph, at least three times faster than other algorithms, and when applying neighborhood propagation, our approach is at least six times faster. If the targeted accuracy becomes higher, or the number of neighbors becomes larger, the improvement becomes more significant.

In the high-dimensional data sets such as TinyImage and Imagenet, the divide-and-conquer algorithms in [7, 42] turn out to be more efficient than kd-tree search, but still at least three times worse than our multiple random division method when the required accuracy is above 60%. After adopting neighborhood propagation, the superiority becomes even more significant than in the low dimensional cases. For TinyImage when $k = 50$, our approach achieves an accuracy of 90% in 6000 seconds, but within the same time, the accuracy of other methods is at most 50%.

Comparing our approach with the brute-force method, we can see that our approach achieves 95% in accuracy using about 1% of the brute-force time for the 128-dimensional SIFT features, and achieves 90% in accuracy using about 2% of the brute-force time for the 384-dimensional GIST features, which shows that constructing an approximate neighborhood graph indeed saves a significant amount of time with only a minor loss in accuracy.

5 Applications

Face images organization. We first present an application that adopts a neighborhood-based distance measure to organize face images. The rank-order distance has been shown good to evaluate the distance between faces [46]. The rank-order distance over two face images is computed by comparing their neighboring faces, which requires first constructing a k-NN face graph. The data set with about $500K$ face images in our experiment consists of a labeled face dataset with 13374 labeled images from LFW [19] and a distracter face dataset collected from the Web. We first compute a 100-NN graph $\mathcal{G}_1$ using our approach. Then we conduct the neighborhood propagation step again to obtain a new k-NN graph $\mathcal{G}_2$, but with the rank-order distances

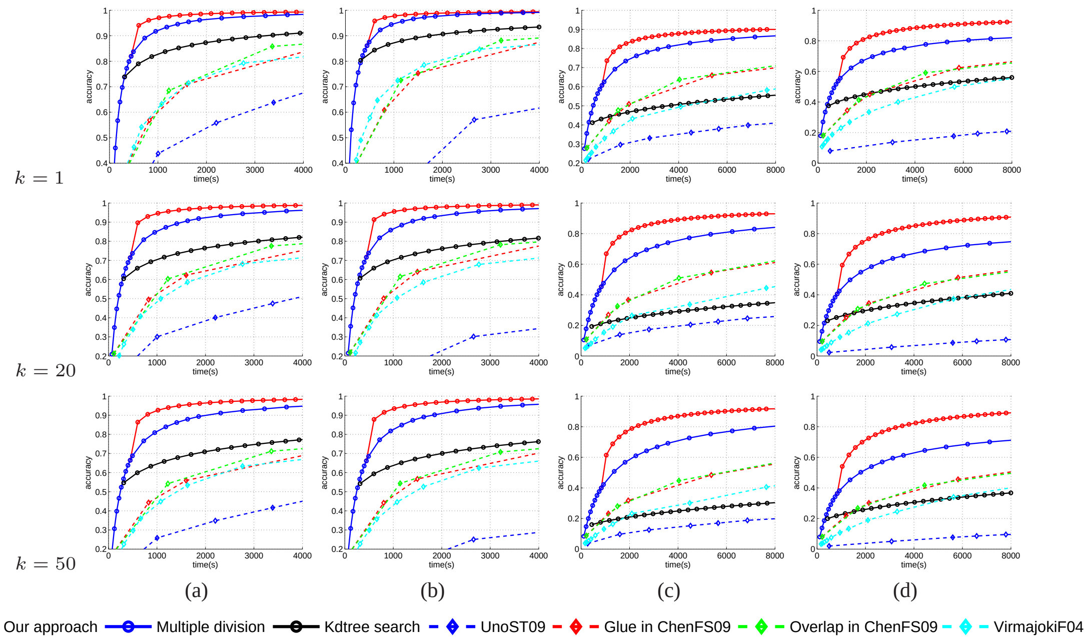


Figure 4: Performance comparisons over (a) Caltech 101, (b) Recognition Benchmark, (c) Imagenet, (d) TinyImage.

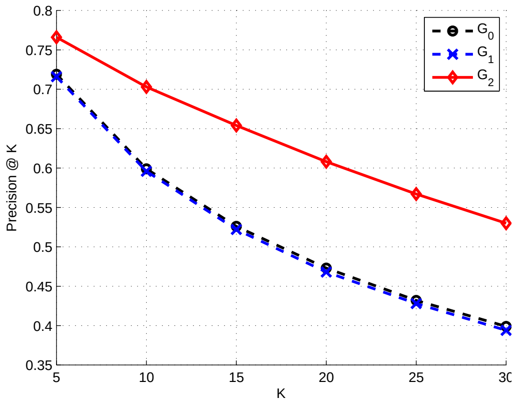
that are computed using the neighborhood from $\mathcal{G}_1$. For comparison, we also report the results over the exact k-NN graph $\mathcal{G}_0$ as the baseline.

For evaluation, we adopt three metrics: local label consistency ratio, $\text{Precision}@K$ and $\text{nDCG}@K$. Local label consistency ratio aims to evaluate if the label of the face is dominant among the labels of the 10 neighboring faces, and is evaluated as 1 if the number of faces with the same label is larger than 6 and otherwise 0. $\text{Precision}@K$ is computed as the proportion of the same faces among the top K neighbors, and $\text{nDCG}@K$ is the normalized discounted cumulative gain over the top K neighbors which has been widely used in various ranking tasks.

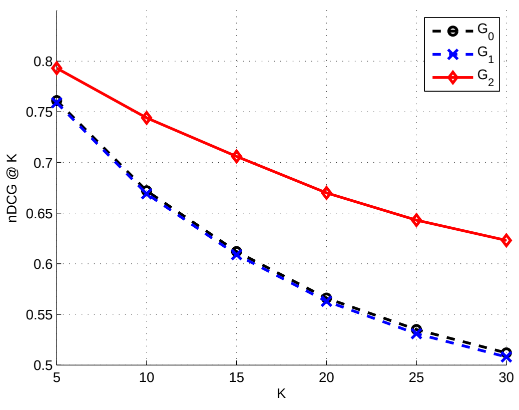
Tab. 2 shows the comparison of the average local label consistency ratio and Fig. 5 shows the comparisons of the average $\text{Precision}@K$ and $\text{nDCG}@K$. We have two observations. On the one hand, the neighborhood graph is very powerful and useful, with which we can get a better distance measure for face images organization. On the other hand, the performances of $\mathcal{G}_0$ and $\mathcal{G}_1$ are almost the same, which shows that the k-NN graph constructed from our approach is very accurate.

Object discovery. Discovering objects from large unlabeled image collections has been a challenging problem [8]. We show that the proposed approach can effectively construct a matching graph with a coarse similarity measure for fast random divisions and a fine similarity measure for accurate neighborhood propagation and that objects can be effectively discovered over such a graph.

The data set consists of 5062 labeled images from the Oxford $5K$ data set [30] and $50K$ distracter images downloaded from Flickr. SIFT features are extracted from each image. We build two vocabularies, respectively with $1K$ and $1M$ visual words. Each image is represented by two features, one $1K$-dimensional histogram over $1K$ visual words that indicates word occurrences and is then weighted by tf-idf, and one bag-of-words representation over $1M$ visual words with attaching the spatial position for each word. The low dimensional features are used to build random divisions for fast neighborhood graph construction, and



(a)



(b)

Figure 5: Comparison between different ranking results. (a) $\text{Precision}\ @K$, (b) $\text{nDCG}\ @K$.

Table 3: Performance for object discovery.

	#images	Precision	Recall	F-measure
All Souls	78	0.785	0.654	0.713
Ashmolean	25	0.947	0.720	0.818
Balliol	12	0.800	0.333	0.471
Bodleian	24	0.100	0.542	0.169
Christ Church	78	0.360	0.692	0.474
Cornmarket	9	0.833	0.556	0.667
Hertford	54	0.829	0.630	0.716
Keble	7	0.667	0.571	0.615
Magdalen	54	1.000	0.130	0.230
Pitt Rivers	6	1.000	0.833	0.909
Radcliffe Camera	221	0.820	0.661	0.732
Average		0.740	0.574	0.592

the high dimensional features with its spatial information are then used to compute image matching with spatial verification [30] for accurate neighborhood propagation, yielding a matching graph. We run affinity propagation [15] over this matching graph to over-segment the image set so that similar views of the same object are grouped together. To join these over-segments for grouping images with the same object, we then construct a graph with over-segments as nodes, and define the similarity between two over-segments as the proportion of images that in both over-segments are 50-reciprocal nearest neighbors [33], which is fast computed over the matching graph. We finally apply affinity propagation again over the over-segment graph to obtain the final clustering result.

To evaluate the performance, for each labeled object group, we find the cluster containing the most images of that object and compute precision, recall and F-measure of that cluster. The result is given in Tab. 3. We can see that most objects can be discovered with an F-measure over 0.6.

6 Conclusions

In this paper, we address the problem of constructing k-NN graphs for large scale visual descriptors. Our approach consists of two steps: multiple random divide-and-conquer and neighborhood propagation. We show both theoretical and empirical accuracy and efficiency of our approach. As an ongoing work, we are investigating a learning scheme to automatically trigger neighborhood propagation.

Appendix

Proofs

Lemma 4. *Suppose a random hyperplane partitions the data points so that a certain point $\mathbf{x}_i$ and one of its true neighbors $\mathbf{x}_j$ have the probability P_{ij} to be on the same side. Then with a single random partition tree, $\mathbf{x}_j$ is discovered as $\mathbf{x}_i$'s neighbor with the probability P_{ij}^h, where h is the depth of the tree. With L random partitions, $\mathbf{x}_j$ is discovered as $\mathbf{x}_i$'s neighbor with the probability $1-(1-P_{ij}^h)^L$.*

Proof. In a random partition tree, two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ will discover each other as neighbors if they they lie on the same side of h random hyperplanes, where h is the height of the tree. According to the assumption, for each hyperplane $\mathbf{x}_i$ and $\mathbf{x}_j$ will lie one the same side with probability P_{ij}. Because hyperplanes are independent with each other, $\mathbf{x}_i$ and $\mathbf{x}_j$ will discover each other as a neighbor with the probability P_{ij}^h. Moreover, the partition trees are also independent with each other, so that with L trees, the discovery probability will be $1-(1-P_{ij}^h)^L$. □

This property can easily be validated using the similar manner to LSH [12]. Although there is a slight difference between random partition trees and LSH that a random partition tree for each level may use different projections, the statement that a true neighboring point of $\mathbf{x}$ is discovered with the probability P_{ij}^h still holds because discovering a true neighboring point of $\mathbf{x}$ must pass h projections. In the case of Euclidean distance, the stable distribution [12] can be used to build a binary partition and the probability P_{ij} can be also easily computed. For example, if using $h_{\mathbf{a},b}(\mathbf{x}) = \lfloor \frac{\mathbf{a}^T\mathbf{x}+b}{w} \rfloor$ where each entry of $\mathbf{a}$ satisfies a Gaussian distribution and b is a uniform random variable over $[0,w]$. It can be demonstrated that $P(h_{\mathbf{a},b}(\mathbf{x}_i) = h_{\mathbf{a},b}(\mathbf{x}_j)) = \int_0^w \frac{1}{d_{ij}} f_p(\frac{t}{d_{ij}})(1-\frac{t}{w})dt$, where d_{ij} is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and $f_p(t)$ denotes the probability density function of the absolute value of Gaussian distribution. The probability will be larger than $\frac{1}{2}$ when $w \geqslant 2d_{ij}$. Then we can bi-partition the data according to the median of $h_{\mathbf{a},b}(\mathbf{x})$ so that P_{ij} will be larger than $\frac{1}{2}$ when $w \geqslant 2d_{ij}$. In the case of cosine distance [12], a random projection can be used to build the random partition tree and P_{ij} can be computed as $1-\frac{d_{ij}}{\pi}$.

Lemma 5. *The probability that the neighboring relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$ is discovered by the L-th random partition tree but not discovered by the previous $L-1$ random partition trees is $P_{r_L} = (1-P_{ij}^h)^{L-1}P_{ij}^h$.*

Proof. According to Lemma. 4, for each of the previous $L-1$ random partition trees, the probability that $\mathbf{x}_i$ and $\mathbf{x}_j$ fail to discover each other as neighbors is $1-P_{ij}^h$, and in the L-th tree the neighboring relationship will be discovered with the probability P_{ij}^h. Then the lemma can be proved with the basic multiplication principle. □

Lemma 6. *Considering two neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ having the same neighboring point $\mathbf{x}_n$, after $L-1$ partition trees, the probability that $\mathbf{x}_j$ can be discovered as the neighbor of $\mathbf{x}_i$ through $\mathbf{x}_n$ is $P_{p_L} = (1-(1-P_{in}^h)^{L-1})(1-(1-P_{jn}^h)^{L-1})$.*

Proof. $\mathbf{x}_j$ can be discovered as the neighbor of $\mathbf{x}_i$ through $\mathbf{x}_n$ if and only if the neighboring relationship between $\mathbf{x}_i$ and $\mathbf{x}_n$ and the relationship between $\mathbf{x}_j$ and $\mathbf{x}_n$ have both been discovered. From Lemma. 4, we know the probability of the two events are $(1-(1-P_{in}^h)^{L-1})$ and $(1-(1-P_{jn}^h)^{L-1})$, by simply multiplying them, we get $P_{p_L}=(1-(1-P_{in}^h)^{L-1})(1-(1-P_{jn}^h)^{L-1})$. $\square$

Theorem 2. *Suppose $P=\min\{P_{ij}|\langle\mathbf{x}_i,\mathbf{x}_j\rangle\in E(G)\}$, where G is the exact k-NN graph of the data set. With L random divisions and a first-order neighborhood propagation a true k-NN point of $\mathbf{x}_i$, $\mathbf{x}_j$ is discovered with at least the probability $1-(1-P^h)^{2L}(2-(1-P^h)^L)$ under the assumption that $\mathbf{x}_i$ and $\mathbf{x}_j$ have at least one same neighboring point $\mathbf{x}_n$.*

Proof. With two ways $\mathbf{x}_i$ will find $\mathbf{x}_j$ as a neighbor. First is to discover $\mathbf{x}_j$ in at least one of the L partition trees, and the probability of it is $1-(1-P_{ij}^h)^L$ according to Lemma. 4. Second is fail to discover $\mathbf{x}_j$ in partition trees, but discover it during the first-order neighborhood propagation. From Lemma. 5,6 the probability is $(1-P_{ij}^h)^L(1-(1-P_{in}^h)^L)(1-(1-P_{jn}^h)^L)$. To sum them up, the probability that $\mathbf{x}_i$ discovers $\mathbf{x}_j$ with L partition trees and a first-order neighborhood propagation is

$$\begin{aligned}
& 1-(1-P_{ij}^h)^L+(1-P_{ij}^h)^L(1-(1-P_{in}^h)^L)(1-(1-P_{jn}^h)^L) \\
& \geqslant 1-(1-P_{ij}^h)^L+(1-P_{ij}^h)^L(1-(1-P^h)^L)^2 \\
& \quad (P_{in},P_{jn}\geqslant P) \\
& = 1-(1-P_{ij}^h)^L+(1-P_{ij}^h)^L(1-2(1-P^h)^L+(1-P^h)^{2L}) \\
& = 1-(1-P_{ij}^h)^L(2(1-P^h)^L-(1-P^h)^{2L}) \qquad (1) \\
& \because\ 0<P<1 \\
& \therefore\ 2(1-P^h)^L-(1-P^h)^{2L}>0 \\
(1) & > 1-(1-P^h)^L(2(1-P^h)^L-(1-P^h)^{2L}) \\
& = 1-(1-P^h)^{2L}(2-(1-P^h)^L)
\end{aligned}$$

$\square$

Face image organization

Fig. 6 shows some examples of face organization results when using Euclidean distance and Rank-order distance. For each face image, we show its 9 nearest neighbors in the graph, and it can be clearly seen that with Rank-order distance, the label consistency within each neighborhoods is enhanced.

Object Discovery

Fig. 7 shows some examples of the detected clusters in the experiments of object discovery.

References

[1] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, pages 271–280, 1993.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

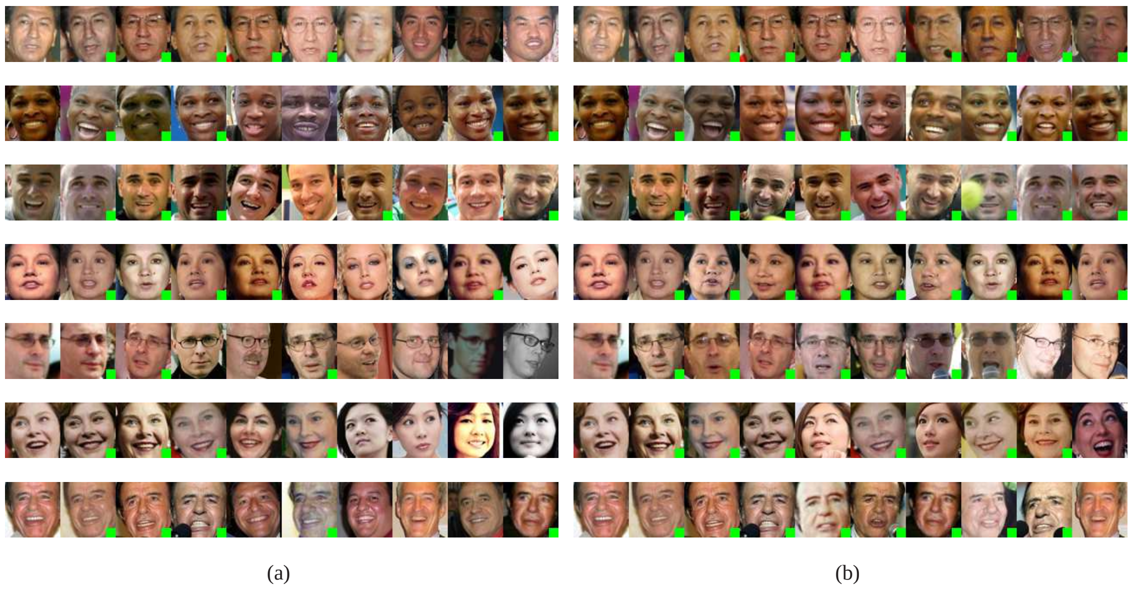


Figure 6: Image ranking result comparison between rank-order distance and Euclidean distance. The first image in each row are the query and the images with green dot at bottom right are the correct images containing the same face as the query. (a) gives the results of using Euclidean distance and (b) gives the results of using Rank-order distance.

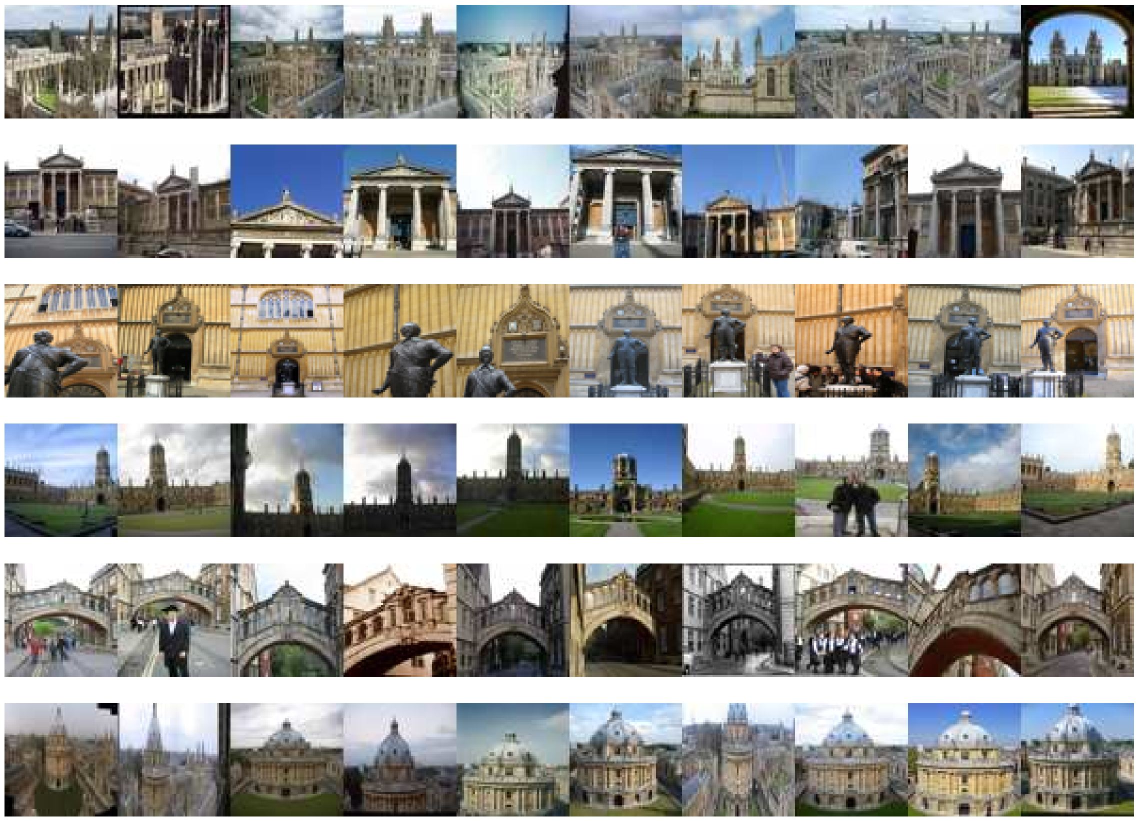


Figure 7: Object discovery results: each row represents one discovered object cluster and the images are randomly picked out from the cluster.

[4] J. L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

[5] J. L. Bentley, D. F. Stanat, and E. H. W. Jr. The complexity of finding fixed-radius near neighbors. *Inf. Process. Lett.*, 6(6):209–212, 1977.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] J. Chen, H. ren Fang, and Y. Saad. Fast approximate nn graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10:1989–2012, 2009.

[8] O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(2):371–377, 2010.

[9] K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *FOCS*, pages 226–232, 1983.

[10] M. Connor and P. Kumar. Fast construction of k-nearest neighbor graphs for point clouds. *IEEE Trans. Vis. Comput. Graph.*, 16(4):599–608, 2010.

[11] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, pages 537–546, 2008.

[12] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262, 2004.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. L. 0002. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[14] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR 2004 Workshop on Generative-Model Based Vision*, 2004.

[15] B. Frey and D. Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315(5814):972–976, 2007.

[16] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[17] H. Hacid and T. Yoshida. Incremental neighborhood graphs construction for multidimensional databases indexing. In *Canadian Conference on AI*, pages 405–416, 2007.

[18] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *CVPR*, pages 3432–3439, 2010.

[19] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[20] H. Jegou, C. Schmid, H. Harzallah, and J. J. Verbeek. Accurate image search using the contextual dissimilarity measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):2–11, 2010.

[21] Y. Jia, J. Wang, G. Zeng, H. Zha, and X.-S. Hua. Optimizing kd-trees for scalable visual descriptor indexing. In *CVPR*, pages 3392–3399, 2010.

[22] P. W. Jones, A. Osipov, and V. Rokhlin. Randomized approximate nearest neighbors algorithm. *Proceedings of the National Academy of Sciences*, 108(38):15679–15686, 2011.

[23] I. Kemelmacher-Shlizerman, E. Shechtman, R. Garg, and S. M. Seitz. Exploring photobios. *ACM Trans. Graph.*, 30(4):61, 2011.

[24] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[25] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[26] W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, pages 679–686, 2010.

[27] M. Maier, M. Hein, and U. von Luxburg. Cluster identification in nearest-neighbor graphs. In *ALT*, pages 196–210, 2007.

[28] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pages 2161–2168, 2006.

[29] R. Paredes, E. Chávez, K. Figueroa, and G. Navarro. Practical construction of -nearest neighbor graphs in metric spaces. In *WEA*, pages 85–97, 2006.

[30] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[31] J. Philbin, J. Sivic, and A. Zisserman. Geometric latent dirichlet allocation on a matching graph for large-scale image datasets. *International Journal of Computer Vision*, 95(2):138–153, 2011.

[32] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *ICVGIP*, pages 738–745, 2008.

[33] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. J. V. Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *CVPR*, pages 777–784, 2011.

[34] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[35] T. B. Sebastian and B. B. Kimia. Metric-based shape retrieval in large databases. In *ICPR (3)*, pages 291–296, 2002.

[36] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, 2008.

[37] J. B. Tenenbaum, V. de Silva, and J. C. Langford. Global geometric framework for nonlinear dimensionality reduction. *SCIENCE*, 290:2319–2323, 2000.

[38] A. B. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.

[39] T. Uno, M. Sugiyama, and K. Tsuda. Efficient construction of neighborhood graphs by the multiple sorting method. *CoRR*, abs/0904.3151, 2009.

[40] P. M. Vaidya. An o(n log n) algorithm for the all-nearest.neighbors problem. *Discrete & Computational Geometry*, 4:101–115, 1989.

[41] N. Verma, S. Kpotufe, and S. Dasgupta. Which spatial partition trees are adaptive to intrinsic dimension? In *UAI*, pages 565–574, 2009.

[42] O. Virmajoki and P. Fränti. Divide-and-conquer algorithm for creating neighborhood graph for clustering. In *ICPR (1)*, pages 264–267, 2004.

[43] J. Wang, F. Wang, C. Zhang, H. C. Shen, and L. Quan. Linear neighborhood propagation and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1600–1615, 2009.

[44] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li. Scalable k-nn graph construction for visual descriptors. In *CVPR*, pages 1106–1113, 2012.

[45] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *NIPS*, 2003.

[46] C. Zhu, F. Wen, and J. Sun. A rank-order distance based clustering algorithm for face tagging. In *CVPR*, pages 481–488, 2011.

[47] X. Zhu. Semi-supervied Learning Literature Survey. *Computer Sciences Technical Report, 1530, University of Wisconsin, Madison*, 2007.