Shrinking data balls in metric indexes

Bilegsaikhan Naidan and Magnus Lie Hetland
Department of Computer and Information Science,
Norwegian University of Science and Technology,
Sem Sælands vei 7-9, NO-7491 Trondheim, Norway
{*bileg,mlh*}*@idi.ntnu.no*

***Abstract*—Some of the existing techniques for approximate similarity retrieval in metric spaces are focused on shrinking the query region by user-defined parameter. We modify this approach slightly and present a new approximation technique that shrinks data regions instead. The proposed technique can be applied to any metric indexing structure based on the ball-partitioning principle. Experiments show that our technique performs better than the relative error approximation and region proximity techniques, and that it achieves significant speedup over exact search with a low degree of error. Beyond introducing this new method, we also point out and remedy a problem in the relative error approximation technique, substantially improving its performance.**

Keywords-approximation algorithms, experiments, similarity search, metric space.

I. INTRODUCTION

Nowadays, efficient similarity retrieval is becoming more important in various applications such as multimedia repositories (images, audio, video) because of the rapid growth of these data sets and the increasing demand for access to them. In such search applications, the relevance of a data object is often measured by some distance function that provides quantitative information about its similarity to some given sample query. For search techniques that treat the distance as a black-box relevance measure, the main challenge is to quickly retrieve a small set of the most relevant objects (either all within a search radius, or the k nearest neighbors, k-NN) relying on the properties of the distance—usually by exploiting the metric axioms.

Numerous metric indexing structures have been proposed to reduce the computational cost (such as the total number of distance computations at query time) of similarity retrieval [1]. These methods primarily rely on various forms of filtering based on the triangle inequality. Triangular filtering is efficient in low-dimensional spaces. However, as the dimensionality of a space increases, the performance of these indexes degrades because of the so-called curse of dimensionality: distances grow increasingly similar, and eventually one may need to examine more or less all data objects, the equivalent of a linear scan. One promising approach to ameliorating this curse is *approximate* similarity search, where some result quality is sacrificed in order to gain performance. This is acceptable in many applications, as distance-based retrieval is generally approximate to begin with—the distance function is most likely an approximation of the user's perception of similarity, and the user probably wants *similar* objects (e.g., pictures of horses), not necessarily the *most* similar objects (i.e., the most similar horse).

Some important methods used in approximate similarity search are discarding data regions at query time (by shrinking the query ball by a user-defined factor [2–4] or by analyzing the intersection of query and data regions [5]), representing data objects as permutations of a set of pivots [6], and estimating the distance by linear regression [7]. We focus on the first approach, trying to develop a method for discard regions that overlap with the query, but that are likely to contain few relevant objects, if any. Our main contributions are:

- We propose a new approximation technique that shrinks data regions instead of the query region, and show empirically that it is superior to existing methods in many cases.
- We amend a problem in the relative error approximation technique of Zezula et al. [2]. In several experimental studies, this technique was found to be the worst one [1, 2, 5]. We point out a problem with how the method has been used, and show how the amended version has significantly improved performance wrt. the original, making it comparable even to the region proximity method [5].

The rest of the paper is organized as follows. In Section II, we briefly review related work. In Section III, we propose our approximation technique, and describe how to amend the relative error approximation technique. Section IV provides experimental results and some discussion of those results. Finally, Section V contains some concluding remarks.

II. RELATED WORK

In this section, we briefly review two metric indexing structures based on the ball-partitioning principle—the M- and SSS-trees—explain some issues in the construction of indexes and also review some approximate techniques that can be applied to those structures.

The M-tree [8] is a hierarchical dynamic metric ball tree that is designed for secondary memory. The M-tree is built in a bottom-up manner like B-trees. The insertion algorithm starts from the root and moves toward the leaves by selecting

 ISBN: 978-1-61208-247-9

nodes that are closer to the new object or that require a minimum enlargement of existing balls. The new object is finally inserted into a leaf node. This may cause the leaf to split (if the node capacity is exceeded), which may trigger splits in some of its ancestor nodes, possibly even the root. For a leaf node split, the covering radius of the split node is set to the distance from the center to the object furthest away, that is, the actual covering radius. For an internal node split, the covering radius is not computed exactly, but overestimated, as follows. For every child node, the covering radius of that node is added to the distance between the center of that child and that of the split node. Then, the covering radius of the split node is then set to the maximum of those sums.

Brisaboa et al. have proposed a static index structure so called the Sparse Spatial Selection (SSS) tree [9], in which the first object in a data set is selected as the first cluster center and then the rest of the objects become new cluster centers if they are far enough away from all current centers (i.e., the minimum distance between the object and current cluster centers is greater than αM, where α is a user-defined parameter and M is the maximum distance between any two objects); otherwise, they are assigned to the cluster associated with the nearest center. The process is recursively applied to those clusters that have not yet fallen below a given size threshold and the diameter M of each such cluster is estimated by using twice the covering radius of the node. Because of the clustering principle used in the construction phase, the internal nodes of SSS-trees will generally have smaller regions than the internal nodes of M-trees. However, there are still sparse regions at higher levels of SSS-trees.

Three approximate techniques for *k*-NN search were introduced by Zezula et al. [2]. The first, the so-called relative error approximation technique, controls approximation through a user-defined relative distance error $\epsilon \geq 0$. For a given query q and error ϵ, an approximation of a *k*th nearest neighbor O_A^k is called a $(1+\epsilon)$ *k*th nearest neighbor, compared to the *true* *k*th nearest neighbor O_N^k, if and only if $d(q, O_A^k) \leq (1+\epsilon) \cdot d(q, O_N^k)$. Thus, the search algorithm uses the radius $r_q/(1+\epsilon)$ instead of the covering radius of the current *k*-NN candidate set r_q to check overlap between query and data regions and candidate object qualification as well. An example of this approach is given in Figure 1a. The second, the so-called good fraction approximation technique, uses a distance distribution to provide an early termination criterion which leads to an approximate *k*NN search. In the third, the so-called small chance of improvement approximation technique, the search algorithm is based on the fact that the dynamic radius of the result set initially decreases rapidly and eventually will slow down. Thus the search stops as soon as the decrease of the radius becomes sufficient.

The PAC method [3] is an extension of $(1+\epsilon)$ nearest neighbor search by a user-specified confidence parameter $\delta \in (0, 1)$. The search algorithm stops immediately if the result satisfies the $(1+\epsilon)$ nearest neighbor with a confidence of at least δ.

Probabilistic LAESA [4] is a probabilistic technique for range search that provides user-customizable limits (θ) for the probability of false dismissals. More specifically, the radius r_q is scaled down by a factor $(1+\epsilon)$ ($\epsilon > 0$) during the filtration of indexed objects. The upper bound for $(1+\epsilon)$ is $r_q\sqrt{1-(1-\theta)^{1/p}}/(\sqrt{2}\sigma)$, where p is the number of pivots and σ^2 is the variance of the distance distribution of the data set. Figure 1b shows an example of this technique.

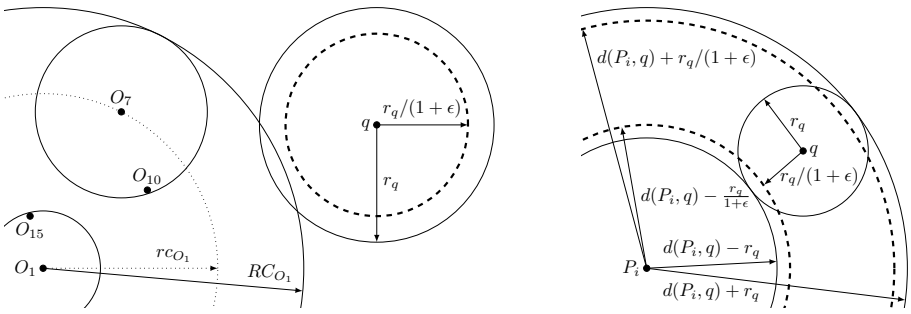
The region proximity technique [5] estimates the probability of the intersection of the query and data regions containing objects relevant to the query. The data region is discarded if the estimated probability is less than a user-specified threshold.

III. Our approach

First, we explain the basic principles of the so-called *best first* strategy [10] for *k*-NN search. In essence, we maintain a set of at most k (initially zero) candidates throughout the search. We also maintain a covering radius for this candidate set. This covering radius is infinite as long as we have fewer than k candidates. The algorithm processes the most promising metric regions first by maintaining a priority queue of pair of distances to regions and pointers to those regions. The following actions are repeated until the lower bound of the distance from the query to the region about to be processed is greater than the current dynamic query radius. The most promising region is popped from the queue. The objects of the current region are checked with the current dynamic query radius and, if necessary, the list of candidate *k*-NN is updated. If we have k candidate, this leads the reduction of the dynamic query radius. Those subregions of the current region that intersect with the query region reinserted into the queue along with the lower bound distances to them from the query.

We now look at our approach. Metric indexes may have highly sparse data regions at higher levels, with the ball radii covering large amounts of empty space—especially for high dimensionalities. During a search, in order to discard those regions that might not contain relevant objects for the query we could use any version of of the $(1+\epsilon)$ NN technique (for instance, the relative error approximation). Our suggested approach is similar to the relative error approximation technique and the key difference is to divide the *data ball radius* by $(1+\epsilon)$, rather than the query radius. See Figure 2 for an example. In the figure, we have shown what happens if we divide both the query radius and the data radius by $(1+\epsilon)$. In the first example (Figure 2a), the query lies close to the data ball, on the outside. In this case, our method lets us eliminate the region simply because it increases the lower bound more. In the second example (Figure 2b), the query is just *inside* the data ball. In this case,

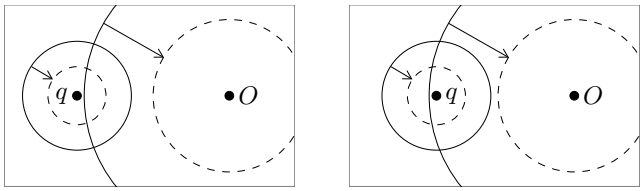
 ISBN: 978-1-61208-247-9



(a) (b)

Figure 1: Examples of data and query ball regions in $\mathbb{R}^2$ with $\mathcal{L}_2$. (a) Relative error approximation in the M-tree [8] with two levels (i.e., top level with center O_1 and bottom levels with centers O_1 and O_7). RC_{O_1} represents the covering radius of top region centered at O_1 while rc_{O_1} represents its "true" covering radius. (b) Probabilistic LAESA.

shrinking the query radius will *never* lead to an elimination, whereas our method does.



(a) (b)

Figure 2: Examples of data and query regions in $\mathbb{R}^2$ with $\mathcal{L}_2$ (a) query q is outside the data region with center O and (b) q is inside the data region.

These examples demonstrate the twofold intuition behind our method: First, ball trees are generally built using some form of clustering. If the data set itself is clustered, and the clustering algorithm is good, this will presumably lead to the center region of a ball being more densely populated than its periphery. Even if this is not the case, by setting the radius to the maximum of all center–object distances, the radius is sensitive to outliers, and the more extreme they are, the fewer there are likely to be. Even if we do not assume a Gaussian distribution, it is not unreasonable to guess that our distance histogram will have the majority of its values clustered roughly around the mean, with fewer occurrences of very high and low distances (ignoring self-distances, $d(x,x)$). Assume that we have a global distance histogram somewhat like that in Figure 3, for example. We also assume that the center–object distances are distributed roughly according to the global distance histogram. Chávez et al. call this behavior as a "reasonable approximation" [11, p. 304]. In this case, it seems that it would be safe to shrink large data balls more than smaller ones, as the number of objects lost would be smaller. The same could be said about the smallest balls, of course; however, we would probably want to examine most of those, if they are close to the query, as they more precisely represent the objects inside them.

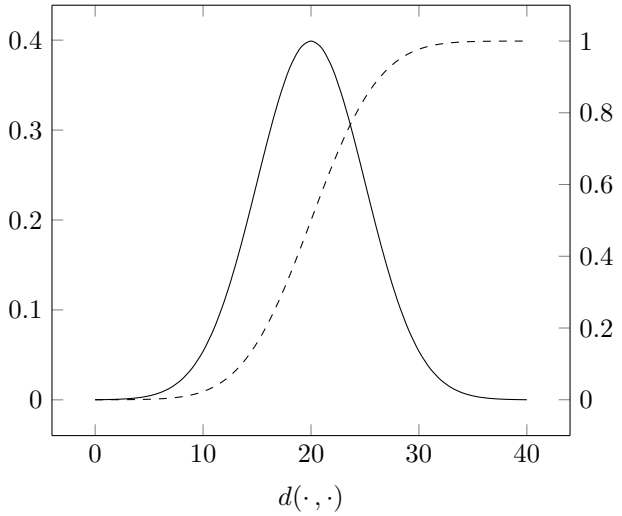


Figure 3: Distance histogram (solid) and cumulative distance histogram (dashed).

Second, shrinking the data balls affords us some elimination possibilities that simply do not exist with the original query-shrinking approach, that is, when the query falls inside the data ball. We have performed some tentative experiments to explore the relative importance of these two factors. At parameter settings that yield similar levels of error, our method generally uses fewer distance computations than the relative error method (see Section IV). We estimate that the proportion of the saved distance computations caused by cases where the query is inside the data ball to vary from about $1\,\%$ to over $50\,\%$ (data not shown).

Another contribution of this paper is that we point out and remedy a problem in the relative error approximation technique. Several experimental studies have showed that the performance of the relative error method (as originally described) is very poor [1, 2, 5]. According to Zezula et al., "the chief reason for the markedly poor performance of

the Relative Error Approximation method (with respect to the others) is that precise nearest neighbors algorithms find good candidates for the result sets soon on, and then spend the remainder of their time mostly in refining the current results" [1, p. 157]. We claim, instead, that the main reason for this performance issue is found in the pruning criterion for a candidate given object O, given by Equation 16 on page 280 of the original paper by Zezula et al. [2]:

$$\frac{r_q}{d(q,O)} < 1 + \epsilon \,,$$

or, equivalently,

$$\frac{r_q}{1+\epsilon} < d(q,O) \,,$$

where r_q is the covering radius of the current k-NN candidate set.

Now, the radius shrinking is intended to reduce the number of distance computations needed by excluding regions of low relevance. There is no need to use it here, as the distance $d(q,O)$ has already been computed, and we simply wish to know whether the object O is an improvement over the candidates we have found so far. We can determine this by simply comparing $d(q,O)$ directly to r_q. Indeed, if

$$\frac{r_q}{1+\epsilon} < d(q,O) < r_q$$

we will lose an improvement to our candidate set, involving an object whose distance we have *already computed*. This can be particularly important early on, where we wish to add good candidates (thereby reducing the dynamic search radius) as quickly as possible. In our experiments, we use this improved version of the relative error approximation technique, checking each candidate object against the actual covering radius of the candidate set.

IV. Experiments

In this section, we evaluate the performance and result quality of our technique against the amended version of the relative error approximation and the region proximity techniques on synthetic and real-world data sets. For all data sets we use the Euclidean distance.

- Uniform 10: Synthetic. 100 000 uniformly generated 10-dimensional vectors.
- Clusters 10: Synthetic. 100 000 clustered 10-dimensional vectors with 10 cluster centers. The centers were randomly chosen from a uniform distribution and objects in the clusters were generated from the multivariate normal distribution around each of the cluster centers with a variance of 0.1.
- Corel: 60 000 feature vectors with 64 dimensions extracted from the Corel image data set.
- NUS [12]: 269 648 color histograms extracted from Flickr images. Each histogram is 64-dimensional.

Amato et al. claimed that there was no practical difference between the proximity and the PAC-NN technique [5, p. 225]. Also PAC-NN is designed only for approximate NN retrieval ($k = 1$). Therefore, PAC-NN is not considered for our experiments.

We have applied our method, region proximity, and the amended version of the relative error technique on M- and SSS-trees. The maximum arities of the trees were set to 30 for the synthetic and 15 for the real-world data sets. We selected 1000 queries from the respective data set at random and the remaining objects in the data set used for indexing. We compared the search performance and result quality of three techniques by varying the result size threshold (k), using the values 1, 5, 10, 20, 40 and 80. We report only the results with 10 NN because the results with the other result size thresholds were quite similar. For the relative error approximation technique, the relative error α was varied in the interval $[0.001, 2.0]$ with step size 0.1 following the experimental settings of Zezula et al. [2]. For the region proximity technique, the proximity value was varied in the interval $[0.003, 0.06]$ with step size 0.003 following the experimental settings of Amato et al. [5]. For our technique, the data region stretching factor was varied between 0.1 and 2.0 with step size 0.1.

For each query, we counted the number of distance computations needed for the approximate search (the most commonly used criterion for measuring the performance of metric indexing structures), normalized by the number needed for an exact search, and measured a slightly modified version of the error on the position [1, 5]. The original version of the error on the position has some drawbacks. First, it gives an error value that is normalized by the size of data set. The normalized values are harder to interpret. For example, we can not directly see the absolute position difference between exact and approximate results. Second, the error should not be normalized by the approximate result's size and should take missing objects in the result set into account.

Let us have a look at a simple example: Let n be data set size and the result size threshold k be 80 ($k < n$) and the approximate result be only true 3 NNs. For some reason, the approximate result did not retrieve other 77 NNs (for instance, the search algorithm was terminated early). Then if we apply the original formula on this example, the error on the position is $((1-1)+(2-2)+(3-3))/(3n) = 0$. The error value 0 means that the approximate result has no error and as we see that it should not be 0 in this case. The modified version takes these situations into account, and the error is increased by n as a *fine* for every missing object and then the error is only normalized by k. This modified version of the error on the position yields the average absolute position difference between every point of the exact and approximate results.

In general, approximation techniques will produce results that vary both in performance and accuracy. In order to make a fair comparison between different techniques we

 ISBN: 978-1-61208-247-9

have to compare their speed-up factors with the same error or vice versa. In some cases, it would be difficult to achieve this goal, as neither performance measure is a deterministic function of the parameter settings. In order to compare the results properly, we plot them as a lines with one point for each parameter setting, with the coordinates for each point given by the mean error and mean normalized distance count for all queries. On the y-axis, the value 10^{-1} means that the indexing structure performed 10 times as fast as an exact search. On the x-axis, the value 10^{1} means that the average absolute position difference between exact and approximate result is 10 (for instance, if the result size threshold is 1, then the 11th NN is reported instead of the NN).

In Figure 4, the errors on the position vs normalized distance counts for $10\,\mathrm{NN}$ on M-trees are shown. Note that both axis are logarithmic. For the results of NUS 10 NN in Figure 4, our technique achieved a speed-up by a factor of more than 4 over the exact search, with the position error less than 10, while the region proximity technique achieved almost same speed-up with relatively high position error 10^4 (i.e., reported 10 objects from around $10\,000$ NNs for the query). For the other data sets, our technique achieved about 2.5 speed-up over exact search with the position error less than 10.

Figure 5 shows the results for SSS-trees. The most interesting results are once again obtained with the NUS data set. For NUS 10 NN, the maximum value of the error was 48.39 (with normalized distance count 0.135) for our technique while for the region proximity technique the error was 1751.39 (with normalized distance count 0.136). The results show that our technique is faster than two other competing techniques with a low degree of the error on the position. The relative error approximation technique outperforms the region proximity technique on the real-world data sets while on the synthetic data sets it does not.

In addition to the experiments presented, we have also performed some tentative experiments involving various tradeoffs between query- and data-ball shrinking. So far, this has not yielded substantial improvements.

V. Conclusions

We have proposed an approximate similarity search technique for metric spaces and we have amended a problem in the relative error approximation technique. We have empirically evaluated our technique, showing that it outperforms the amended version of relative error approximation and the region proximity techniques.

Acknowledgements

We wish to thank Øystein Torbjørnsen and Svein Erik Bratsberg for helpful discussions.

References

[1] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search : The Metric Space Approach*. Springer, 2006.

[2] P. Zezula, P. Savino, G. Amato, and F. Rabitti, "Approximate similarity retrieval with M-Trees," *The VLDB Journal*, vol. 7, no. 4, pp. 275–293, 1998.

[3] P. Ciaccia and M. Patella, "PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces," in *Proceedings of the 16th International Conference on Data Engineering, ICDE* (IEEE Computer Society, ed.), pp. 244–255, 2000.

[4] E. Chávez and G. Navarro, "A probabilistic spell for the curse of dimensionality," in *Revised Papers from ALENEX'01*, pp. 147–160, 2001.

[5] G. Amato, F. Rabitti, P. Savino, and P. Zezula, "Region proximity in metric spaces and its use for approximate similarity search," *ACM Transactions on Information Systems, TOIS*, vol. 21, no. 2, pp. 192–227, 2003.

[6] E. Chavez Gonzalez, K. Figueroa, and G. Navarro, "Effective proximity retrieval by ordering permutations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[7] O. Edsberg and M. L. Hetland, "Indexing inexact proximity search with distance regression in pivot space," in *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, 2010.

[8] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB*, pp. 426–435, Morgan Kaufmann, 1997.

[9] N. Brisaboa, O. Pedreira, D. Seco, R. Solar, and R. Uribe, "Clustering-based similarity search in metric spaces with sparse spatial centers," in *Proceedings of SOFSEM'08*, no. 4910 in LNCS, pp. 186–197, 2008.

[10] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[11] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Computing Surveys*, vol. 33, pp. 273–321, September 2001.

[12] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *Proc. of ACM Conference on Image and Video Retrieval (CIVR'09)*, 2009.

 ISBN: 978-1-61208-247-9

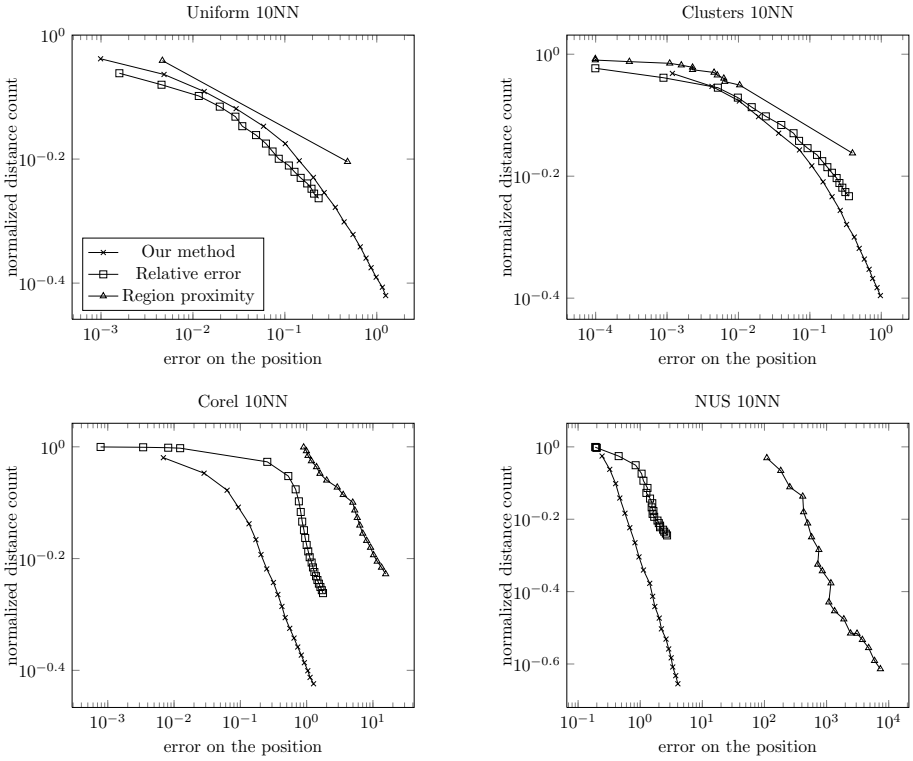


Figure 4: Performance vs. result quality of approximation on the synthetic and real-world data sets with M-trees.

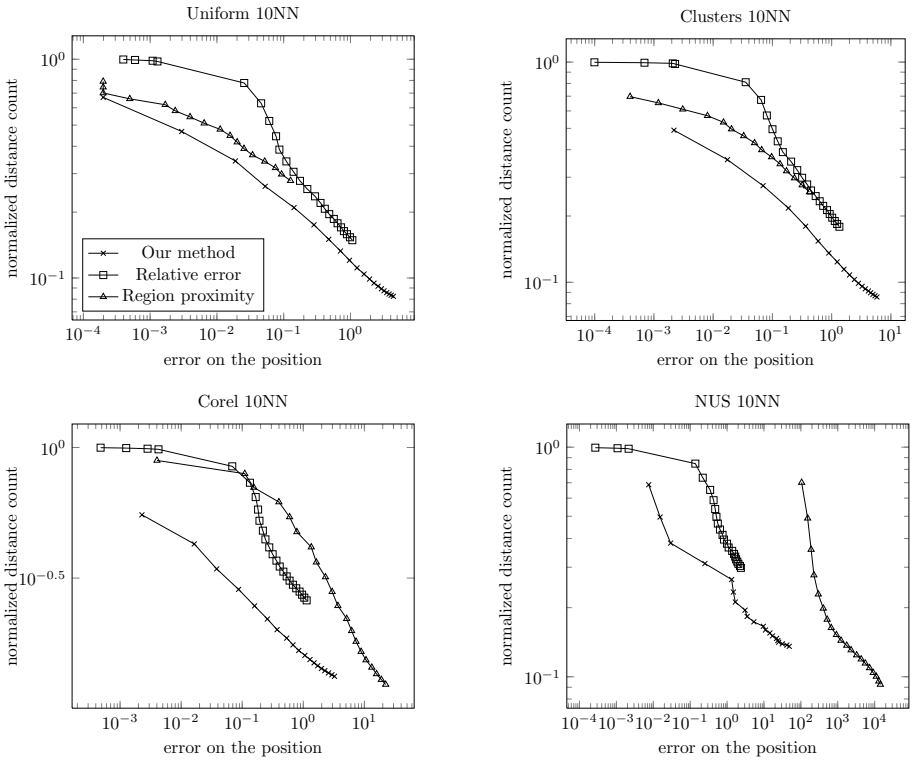


Figure 5: Performance vs. result quality of approximation on the synthetic and real-world data sets with SSS-trees.

 ISBN: 978-1-61208-247-9