Fractal Approximate Nearest Neighbour Search in Log-Log Time

Martin Stommel
mstommel@tzi.de
Stefan Edelkamp
edelkamp@tzi.de
Thiemo Wiedemeyer
wiedemeyer@informatik.uni-bremen.de
Michael Beetz
beetz@cs.uni-bremen.de

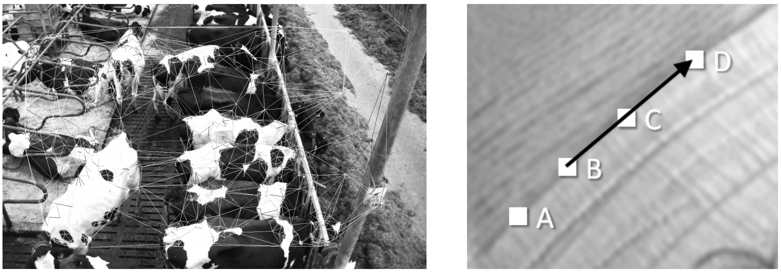
Institute for Artificial Intelligence
University of Bremen
Am Fallturm 1
28359 Bremen
Germany

Abstract

The power of fractal computation has been mainly exploited for image compression and halftoning. Here, we consider it for finding a fast approximate solution for the fundamental problem of nearest neighbour computation in the image plane. Traditional solutions use Delaunay triangulations or hierarchies (for the case of optimal solutions) or kd-trees for approximate ones. In contrast, we use a space-filling Hilbert curve which allows us to reduce the problem from 2D to 1D. The Hilbert curve has already been used to optimise high-dimensional nearest neighbour queries in the context of data base systems. In this paper, we propose a simplified solution that fits better to the computer vision context. We show that our algorithms solve two particular nearest neighbor problems efficiently. We provide practical results on the accuracy of the method and show that it is significantly faster than a kd-tree.

1 Introduction

Nearest neighbour searches in the image plane are among the most frequent problems in a variety of computer vision and image processing tasks. They can be used to replace missing values in image filtering, or to group close objects in image segmentation, or to access neighbouring points of interest in feature extraction. In image filtering, the filter result is often only computed for a sparse set of key points. This is either the case if the processing of the whole image would take too much time or if only a small set of pixels is suitable for processing (e.g. because of the aperture problem). The missing filter output must then be interpolated between the nearest keypoints. In image segmentation, nearest neighbour searches allow for a (possibly recursive) combination of close points to more complex objects, which leads to a fine-to-coarse decomposition of the image [28, 30]. In particular for object recognition, the concept of spatial proximity seems to be of fundamental importance, with a clear effect on the image statistics [29]. Figure 1 illustrates examples of common nearest neighbour problems.



a) Interpolation between sparse keypoints b) Line feature extraction

Figure 1: Nearest neighbour problems in the image plane. a) Mesh of SIFT keypoints used to detect the movements of cows in a stable. Since there are only measurements for the nodes of the mesh, pixels within the cells must be assigned to the nearest node. As a second nearest neighbour problem, the cells of the mesh can be constructed by finding two neighbouring nodes for every node. Missing measurements can then be interpolated between the corners of a cell. b) Estimation of the orientation of the edge through point B. The edge is represented by keypoints A–D. The orientation is computed by constructing the line segment of B to the approximate nearest neighbour D instead of the exact nearest neighbour A.

Traditional solutions to such nearest neighbour problems are a pixel-wise search within adaptive or fixed-size image windows, or the use of Delaunay triangulations and kd-trees. Search windows are unattractive because many irrelevant pixels must be visited. Fast results can only be achieved by using small windows or sub-sampling, often with a loss of accuracy. And although fast approximate results would often be preferred over accurate but slow computations, fixed window sizes may simply not suit the problem well. Balanced trees are more attractive because of their logarithmic run-time for a nearest neighbour search. The construction of the tree however introduces additional overhead, in the case of video sequences even repeatedly.

In this paper, we propose a fractal approach to achieve an approximate solution. The basic idea is to map the image plane to a one-dimensional space filling curve, the Hilbert curve, and perform the nearest neighbour search there. The Hilbert curve is known to keep the original 2D-relationship to a certain degree. As a result, an approximate nearest neighbour can be found by searching the nearest neighbour (in other words the successor or predecessor) in a linear list. This can be done in one step or in log-log time depending on the implementation. Since the mapping is the same for every image (assuming a fixed size), there is no repeated overhead for video sequences. The theoretical and experimental results in this paper show that the proposed method is quite powerful in a computer vision context. Surprisingly, the use of space filling curves is largely unknown in this domain. Rare counterexamples include the application in halftone methods [26], image compression [22], and edge detection [19]. Aside from simplifications of the exact method [10], the contribution of this paper consists in the transfer of the method from database systems to computer vision.

2 Related Work and Problem Definition

The *nearest neighbour problem* is usually stated independently of the application as returning the point $p \in S, S = \{(x_1,y_1),\dots,(x_n,y_n)\}$ that minimises the Euclidean distance $||p-q||_2$

to a query point $q = (x, y)$. The simple solution of a linear scan comprises a comparison of q to all elements of S, which is too time-consuming for most applications, especially those with real-time requirements.

In image processing and computer vision the search space can often be reduced if the nearest neighbour is known to lie on an edge or another detectable feature. Contour tracers [15, 25] can then lead to fast results. The drawback of such methods is that many special cases must be considered and contours may be broken because of noise. A full search for the continuation of an edge after a gap is slow and complicated. A noise-tolerant but still slow solution could be to use active contours [18], where the shape of a contour is optimised with respect to energy functionals describing the adaptedness to the image and the shape complexity. Contour tracing does not benefit from information of neighbour searches of adjacent query points, which makes the method slow for dense sets of query points.

If the nearest neighbour $p \in S$ must be found for every coordinate $q \in I$ of an image $I = \{(0,0),(0,1),(0,2),\ldots,(W,H)\}$ of width W and height H, we obtain the *all nearest neighbours problem*. This problem occurs frequently in modern saliency based approaches, where only robustly detectable image regions are processed (exemplarily [20]). Filter results for subsequent steps (e.g. object movements) are only computed for keypoints placed in those regions. In order to assign the filter results to single pixels, it might be necessary to find the nearest keypoint. The problem can be solved by a distance transform [27]. The method exploits that images consist of connected discrete pixels. The result can then be computed in two passes through the image [14, 27], and hence in linear time. The coordinates of the nearest neighbours can be recorded in the same process. The method does however not generalise to the *k-nearest neighbour problem*, where k neighbours $p_1,\ldots,p_k$ of a query point q must be found with distance $||p_i - q||_2 \leq ||p_j - q||_2, i \leq k, j > k$.

Graph-based methods [4, 8, 12] are also often used for nearest neighbour searches in computer vision, although these methods do not benefit from any image specific structuring of the data (e.g. contours). Approximate solutions can be computed by recursively subdividing the search space into smaller cells by using a kd-tree [3, 4, 21]. The approximately nearest neighbour in a bounded uniform distribution is found in $O(\lg n)$ by searching a list of cells ordered by proximity to the query point. Exact solutions can be computed by using Delaunay hierarchies [8, 12]. One recent result to the nearest neighbor problem are full Delaunay hierarchies [6]. The main difference between an ordinary Delaunay triangulation is to additionally record all edges that have been used during the incremental construction. Using a randomized incremental construction of the Delaunay triangulation, the approach uses $O(n \lg n)$ expected time for building the data structure with an expected number of $O(n)$ edges. The expected number of nodes traversed for finding the nearest neighbor by a simple greedy algorithm yield an expected query time of $O(\lg n)$. Different to many other approaches, once the graph is constructed, an additional search structure is not needed.

To simplify high-dimensional nearest neighbour queries, locality sensitive hashing has been proposed [13, 23, 31]. The idea is to map the input data on the unit hypercube in a way that preserves the original distance relationships to a certain degree, even if the Hamming distance is used to compare the mapped vectors. The methods have been shown to allow for fast vector comparisons, predominantly for SIFT-related data sets. For low-dimensional data, such methods are too coarse: The cited approaches would divide the 2D-plane into at most four areas.

Space filling curves have originally been introduced to demonstrate the existance of a point-to-point mapping of the real valued unit square to a continuous curve [24]. The Hilbert curve [17], a variant of the original Peano curve [24], is defined as the recursive subdivi-

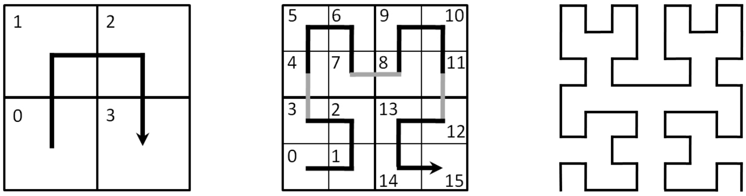


Figure 2: Hilbert curve for a 2×2 (on the left), 4×4 (middle) and 8×8 image (on the right). Pixels are counted along the curve. The second curve (and higher resolutions) can be computed (recursively) from the first one by replacing each corner by a rotated and reflected version of the basic U-shaped pattern.

sion of a square into four sub-squares with a U-shaped ordering imposed on the sub-squares (cf. Fig. 2). In the case of discrete images, the recursion stops after a certain number of iterations R. This results in a subdivision of the unit square into $W = H = 2^R$ rows and columns. There are several algorithms that can be used to compute the complete mapping in linear time (in terms of the number of pixels) [9, 16, 17]. For a specific image coordinate, the corresponding index of a point on the Hilbert curve (the *Hilbert index*) requires the recursive descent over R levels and a number of bit operations on each level, hence the complexity is $O(\lg W)$. The mapping preserves 2D-distance relationships in the sense that neighbouring points on the Hilbert curve are neighbouring in the image, too. Larger distances are preserved by the quad-tree-like recursive subdivision of the image. This has motivated to use the Hilbert curve to answer exact [10, 11] and approximate [32] nearest neighbour queries. To compensate for errors in the mapping, the Hilbert indices of adacent image coordinates must be found. This can be solved by reducing the problem to the easier Peano curve [10] or by evaluating multiple mappings [32]. Most methods consider high-dimensional input data. The generalisation of the Hilbert curve to more than two dimensions is however not as straightforward as it seems. Alber and Niedermeier [1] show that already in 3D there are 1536 structurally different curves with Hilbert property, which might be a potential for further optimisation. Aside from nearest neighbour computations, the Hilbert curve is also used to visualise high-dimensional data sets [2].

We observed that in many computer vision applications, a high accuracy of the nearest neighbour search is not needed. Our method therefore differs from the above mentioned ones in that we do not attempt to correct the inaccuracies in the mapping of 2D to 1D-distances. As a result, our method is able to find close points at low computational cost.

3 Proposed Method

Our method answers two types of nearest neighbour problems in three steps each: At first, the mapping between 2D and 1D-coordinates must be computed. For the all nearest neighbour problem, the set of keypoints S must be written as an array. The nearest neighbour assignment can then be done in two passes throught the array. For the k-nearest neighbour problem, the set of keypoints S must be stored in a priority queue. The neighbours of a query point q can then be found by using the successor function of the queue.

A Hilbert curve of recursive depth R subdivides the unit square into 2^R rows and columns. The size of the sub-squares is $2^{-R} \times 2^{-R}$. We map an image with unit square sized pixels (i.e. integer coordinates) to the unit cube by applying the scale factor 2^{-R}. To make sure that

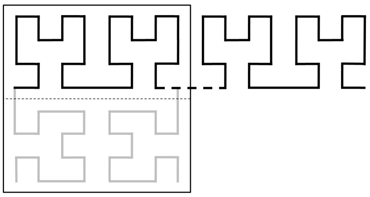


Figure 3: For elongated rectangular images it is possible to use only the upper half of the Hilbert curve or to stitch together multiple Hilbert curves.

an image with W columns and H rows fits into the unit cube, the recursive depth must be $R = \lceil \log_2 \max(W,H) \rceil$. We obtain C Hilbert indices range from 0 to $2^{2R}-1$. More recursions are redundant. With this scaling, we have a correspondence between the Hilbert indices of the 1D-curve, the sub-squares of the unit-cube, and the pixels of an image. Rectangular images and images whose side length is not a power of two do not cover the unit square completely. An image of size 640×480 would be mapped to a 1024×1024 grid in the unit cube. In this case, only 30% of the Hilbert indices correspond to pixels in the image. Since the algorithm for the all nearest neighbour computation requires a loop over all Hilbert indices, it might be advantageous to cut the Hilbert curve to the the upper half of the unit square, or to string together multiple Hilbert curves (Figure 3) in order to reduce the overhead. In the rest of the paper, however, this optimisation is not used. The result of the first step is a mapping $M(x,y) : \mathbb{N}^2 \rightarrow \mathbb{N}$ of image coordinates (x,y) to Hilbert indices, as well as the partial inverse mapping $M^{-1}(i) : \mathbb{N} \rightarrow \mathbb{N}^2$ of a Hilbert coordinate to the image coordinate (x,y). The mapping must be computed only once for a fixed image size. For the whole image this can be done in linear time with respect to the number of pixels. We use arrays to store the mapping.

The mapping is used to solve two types of problems. In the all nearest neighbour computation we find the approximately closest keypoint $p \in S$ for all pixels $q \in I$.

Theorem 1. *The all approximately nearest neighbour problem can be solved in precomputation time $O(C+n)$ and query time $O(1)$. It uses $O(C)$ space.*

Proof. Constant query time is achieved by computing a lookup table T of size $O(C)$ with the results. The indices of the lookup table are the Hilbert indices. At first, we mark the n keypoints S in the table, which is $O(n)$. In one forward pass, we compute for all indices i the distance $i-j$ to the nearest keypoint with lower Hilbert index j. To this end, we need to check if index i of the lookup table corresponds to a keypoint, i.e. if $M^{-1}(i) \in S$. In that case, we set the distance to zero. We also set it to zero if there is no preceding keypoint. Otherwise we increment the distance of the preceding table entry by one. The check and the distance increment can be done in constant time, so the general complexity of the forward pass is $O(C)$. In a backward pass, we compute the distance to the nearest keypoint with higher Hilbert index. Then we assign each index of the Hilbert curve to the nearest keypoint index from the forward and backward pass. The complexity of the backward pass is again $O(C)$. The total complexity is therefore $O(n+C)$. □

The second problem is the approximately k-nearest neighbour problem.

Theorem 2. *The approximately k-nearest neighbours can be found in precomputation time $O(n \lg\lg C)$ and query time $O(\lg\lg n + k)$. The space requirement is $O(C \lg\lg C)$.*

Proof. We build a priority queue Q where all n keypoints S are ordered by their Hilbert index $M(p), p \in S$. Using the precomputed arrays for M, the Hilbert index of a key point can be found in constant time. Inserting an element in a priority queue takes constant time plus an overhead of $O(\lg\lg C)$ for locating the right position in the queue using the successor function [7]. The complexity for inserting n elements is therefore $O(n \lg\lg C)$. By linking all adjacent elements of the queue (in $O(n)$), we can find the successor and predecessor of an existing element in constant time. The priority queue needs to be computed only once for a given set of keypoints. The precomputation time is therefore $O(n \lg\lg C)$.

For a query point q, the approximately nearest neighbour can be the successor of $M(q)$ in Q or its direct predecessor. The successor $M(p_s)$ can be found in $O(\lg\lg C)$. Its direct predecessor $M(p_p)$ is found in constant time using direct links. The comparison of the Hilbert indices ($|M(q) - M(p_s)|$ and $|M(q) - M(p_p)|$) and the nomination of an approximately nearest neighbour is also constant time. The remaining $k-1$ neighbours can be found by reading out the directly linked $(k-1)/2$ predecessors and $(k-1)/2$ successors, which are single step operations. The k nearest neighbours can therefore be found in time $O(\lg\lg C + k)$ once the queue is constructed. The space requirement is $O(C \lg\lg C)$ [7]. □

4 Experimental Results

We first measured how well 1D-neighbourhoods on the Hilbert curve correspond to 2D-neighbourhoods in an image and vice versa. To this end, we created a 256×256 image together with its Hilbert curve. We sampled a first list of 50 000 pairs of points on the Hilbert curve with a maximum difference of 10 in their Hilbert indices. For every pair we then computed the corresponding 2D-Euclidean distance in the image. As Fig. 4 (left) shows, close points in 1D are close in 2D as well. On average, the 2D-distance is the square root of the 1D distance. Median and mean are close together, so there are no serious outliers to the rule. We then sampled a second list of 50 000 pairs of image points with a Euclidean distance less than 10 and measured the corresponding 1D-distance of their Hilbert indices. Figure 4 (right) shows that close points in 2D are not necessarily close in 1D. The 1D-distances are not normally distributed and the mean deviates from the median because of far outliers (the maximum 1D-distance was about 54 000 for every 2D-distance). The relatively unaffected median of only up to 200 shows that the number of outliers is however below the 50% breaking point of the median. For our nearest neighbour search this means that the approximated result will be close to the query point in 2D if it is close on the Hilbert curve in 1D. The distance in 1D will be small because it is minimised by our algorithm. On the other hand, our method will sometimes miss the exact nearest neighbour because small 2D-distances do not necessarily translate to small 1D-distances. But because of the good 1D-to-2D-correspondence, the result will still be good.

In order to measure the accuracy of the method, we solved the all nearest neighbour problem for a small image and varying numbers of keypoints first using the proposed approximation and secondly using an exact method. It turns out that our approximation yields the same nearest neighbour as the exact method in about 50% of the queries (Fig. 5). Even in the case where the proposed method produces different results, the approximated neighbour is close (as shown above). As Fig. 6 shows, the proposed method produces a compact partitioning of the image plane. It is roughly comparable to the Voronoi diagram of the exact solution.

We benchmarked the computation time of our method for the all nearest neighbour prob-

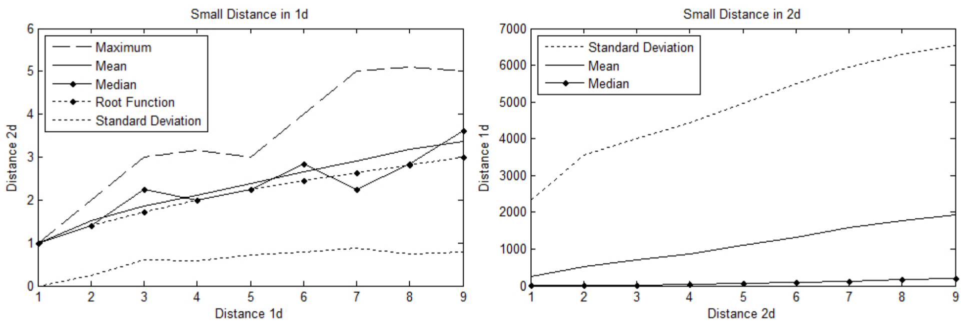


Figure 4: Relationship between corresponding 1D and 2D-distances for randomly selected pairs of close points (distance < 10) once in 1D (left diagram) and then in 2D (on the right). The left diagram shows that the approximated nearest neighbour is close to the query point in 2D if it is close in 1D (which it is expectedly).

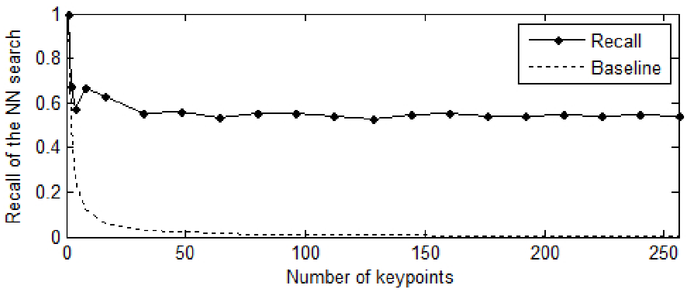


Figure 5: Probability of finding the exact nearest neighbour by applying the proposed *approximative* method to a 256 × 256 image. For a cursory reader we remark that it is quite improbable to find the nearest neighbour by chance (baseline curve).

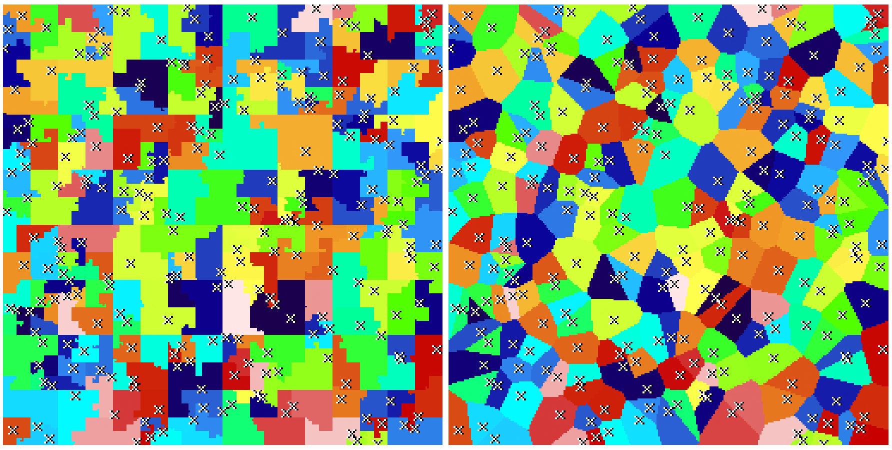


Figure 6: All nearest neighbour assingment using the proposed approximative method (left image) and an exact search (right image) for 240 keypoints (marked by crosses). The resulting cells are coloured randomly but consistent over both diagrams.

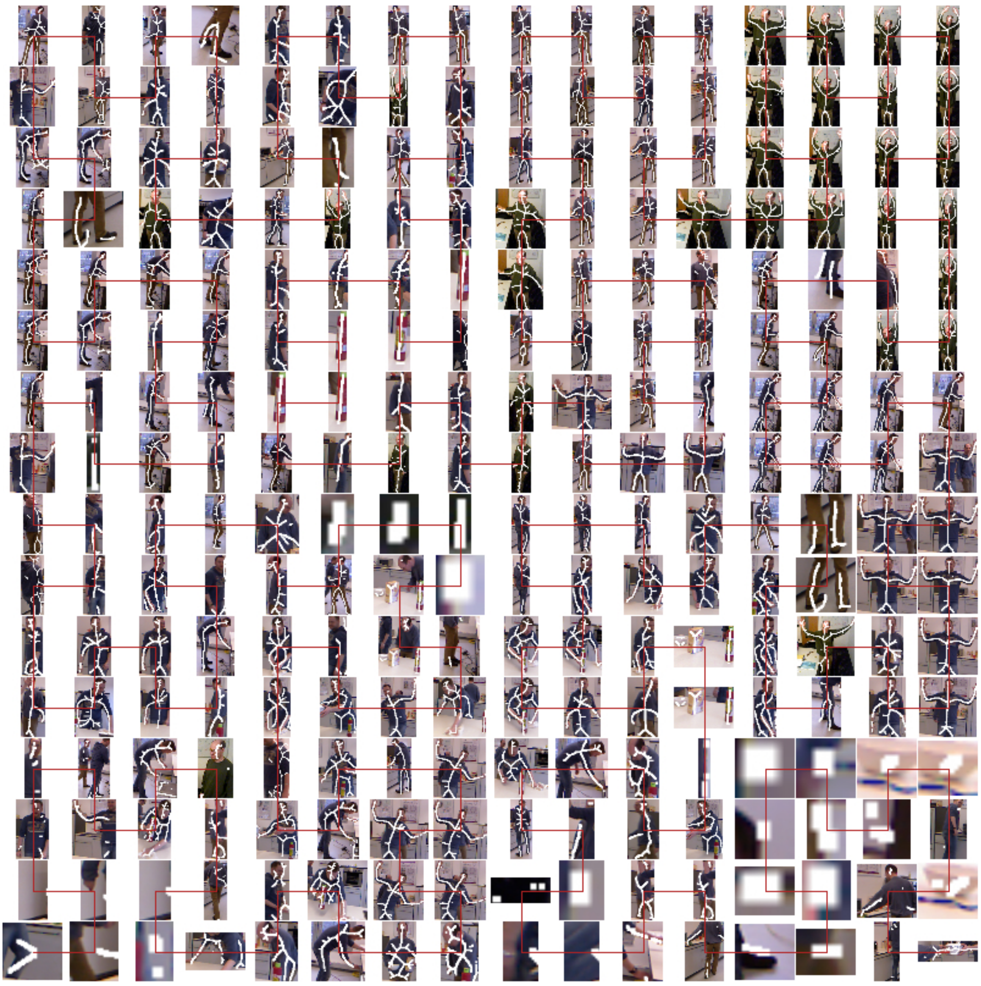


Figure 7: Using a Hilbert curve to display the clustering of a high dimensional data set: The data set consists of images of skeletons of persons in Kinect images. Each skeleton is described by a 3D-histogram over the x and y image coordinate and the angle of a branch of the skeleton. The angle was found using the proposed nearest neighbour method to vectorise the branches (similar to the method outlined in Fig. 1 a). The skeletons were clustered by computing the minimum spanning tree in a fully connected graph weighted by the cross correlation between the histograms. A preorder traversal of the tree yields a 1D-visualisation of the data. The Hilbert curve was used to obtain a more compact and space effective representation.

lem against a kd-tree [5] which provides an exact solution in $O(\lg n)$ query time. Our input data is an image with 1920×1200 query points and 4753 keypoints provided by a feature extractor. The software was run on an Intel Core-i7 2670QM processor in a single-threaded implementation. We measured a precomputation time of 1.9ms to build the kd-tree, and 536.7ms to find all nearest neighbours (averaged over 100 runs each). For the proposed method, we measured 195ms for initialisation and 29.8ms to find all nearest neighbours (again averaged over 100 runs). This is a speed-up factor of 18. For a smaller image of 1280×800 pixels and 4694 keypoints, building the kd-tree took again 1.9ms, whereas the nearest neighbour computation took 236.3ms. Since the recursive depth R is equal for both images, we achieved similar results for the proposed method: 190ms to initialise the method, and 26.2ms for the all nearest neighbour search. The speed-up factor is 9 here.

As a last result, we present an example of using the Hilbert curve to visualise a high-dimensional data set (Fig. 7). The curve was used both to determine the global arrangement of the figure, and to compute one of the underlying features.

5 Conclusion

Our method uses the Hilbert curve to compute fast approximate solutions of the all nearest neighbour problem and the k-nearest neighbour problem in the image plane. Our method has a precomputation time of $O(n)$ for adapting to a fixed image size. Depending on the problem, an additional precomputation time of $O(C+n)$ (all nearest neighbours) or $O(n \lg\lg C)$ (k-nearest neighbours) is needed to adapt to a certain set of key points. The query time is then $O(1)$ or $O(\lg\lg C + k)$, respectively. This is an advantage over a balanced tree (with runtime $\lg n$) if $n > \lg C$ (the latter being a small number for common image formats). Our experiments show that our method yields a compact and visually meaningful approximation of the Voronoi diagram in the image plane, which is sufficient for many applications. For 50% of the queries, our method yields the exact result. In a practical example of finding all nearest neighbours of a set of keypoints, our method was 9–18 times faster than a kd-tree.

Acknowledgement

This work has been funded by the German Science Foundation (DFG) as part of the project MeMoMan2.

References

[1] J. Alber and R. Niedermeier. On Multidimensional Curves with Hilbert Property. *Theory of Computing Systems*, 33(4):295–312, 2000.

[2] S. Anders. Visualisation of genomic data with the Hilbert curve. *Bioinformatics*, 25: 1231–1235, 2009.

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008. ISSN 0001-0782.

[4] S. Arya and H.-Y. A. Fu. Expected-case complexity of approximate nearest neighbor searching. In *Symposium on Discrete Algorithms*, pages 379–388, 2000.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782.

[6] M. Birn, M. Holtgrewe, P. Sanders, and J. Singler. Simple and Fast Nearest Neighbor Search. In *ALENEX*, pages 43–54, 2010.

[7] P. Emde Boas, R. Kaas, and R. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical systems theory*, 10(1):99–127, 1976.

[8] J.-D. Boissonnat and M. Teillaud. The hierarchical representation of objects: The Delaunay tree. In *ACM Symposium on Computational Geometry*, pages 260–268, 1986.

[9] A.R. Butz. Alternative algorithm for hilbert's space-filling curve. *Computers, IEEE Transactions on*, C-20(4):424–426, 1971.

[10] H.-L. Chen and Y.-I. Chang. Neighbor-finding based on space-filling curves. *Inf. Syst.*, 30(3):205–226, May 2005. ISSN 0306-4379.

[11] H.-L. Chen and Y.-I. Chang. All-nearest-neighbors finding based on the Hilbert curve. *Expert Systems with Applications*, 38(6):7462 – 7475, 2011. ISSN 0957-4174.

[12] O. Devillers. The Delaunay Hierarchy. *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002.

[13] S. Edelkamp and M. Stommel. The Bitvector Machine: A Fast and Robust Machine Learning Algorithm for Non-linear Problems. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 175–190. Springer, 2012.

[14] R. Fabbri, L. Da F. Costa, J. C. Torelli, and O. M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, 40(1):2:1–2:44, February 2008. ISSN 0360-0300.

[15] S. Gul and M. F. Khan. Automatic Extraction of Contour Lines from Topographic Maps. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 593–598, 2010.

[16] C. H. Hamilton and A. Rau-Chaplin. Compact Hilbert indices: Space-filling curves for domains with unequal side lengths. *Information Processing Letters*, 105(5):155–163, 2007.

[17] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[18] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[19] C.-H. Lamarque and F. Robert. Image analysis using space-filling curves and 1d wavelet bases. *Pattern Recognition*, 29(8):1309 – 1322, 1996.

[20] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Converence on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[21] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[22] T. Ouni, A. Lassoued, and M. Abid. Gradient-based Space Filling Curves: Application to lossless image compression. In *IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, pages 437–442, 2011.

[23] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recogn. Lett.*, 31(11):1348–1358, 2010. ISSN 0167-8655.

[24] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[25] R. Pradhan, S. Kumar, R. Agarwal, M. P. Pradhan, and M. K. Ghose. Contour Line Tracing Algorithm for Digital Topographic Maps. *International Journal of Image Processing (IJIP)*, 4(2):156–163, 2010.

[26] T. Riemersma. A balanced dithering technique. *C/C++ Users J.*, 16(12):51–58, December 1998. ISSN 1075-2838.

[27] A. Rosenfeld and J. L. Pfalz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[28] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[29] M. Stommel and K.-D. Kuhnert. Part Aggregation in a Compositional Model Based on the Evaluation of Feature Cooccurrence Statistics. In *International Conference on Image and Vision Computing New Zealand (IVCNZ)*. IEEE, 2008.

[30] M. Stommel and K.-D. Kuhnert. Visual Alphabets on Different Levels of Abstraction for the Recognition of Deformable Objects. In *Joint IAPR International Workshop on Structural, Syntactic and Statistical Pattern Recognition (S+SSPR)*, LNCS 6218, pages 213–222. Springer, 2010.

[31] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved Matching with Smaller Descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1):66–78, 2012.

[32] H. Xu. An Approximate Nearest Neighbor Query Algorithm Based on Hilbert Curve. In *Internet Computing Information Services (ICICIS), 2011 International Conference on*, pages 514–517, 2011.