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

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 particular, we address two nearest neighbour problems:

The *nearest neighbour problem* is usually stated independently of the application as returning the point $p \in S, S = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ 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.

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 (for example in SIFT).

In this paper, we introduce an approximate solution to solve these problems that is based on using a space filling curve. The central idea of the proposed approach is to map the image plane to one dimension using the Hilbert curve (Fig. 1). The nearest-neighbour problem is then solved efficiently in one dimension and mapped back to the 2D-plane.

The proposed method answers the two types of nearest neighbour problems in three steps each: At first, the mapping between 2D and 1D-coordinates is 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.

The 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 (Fig. 2), 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.

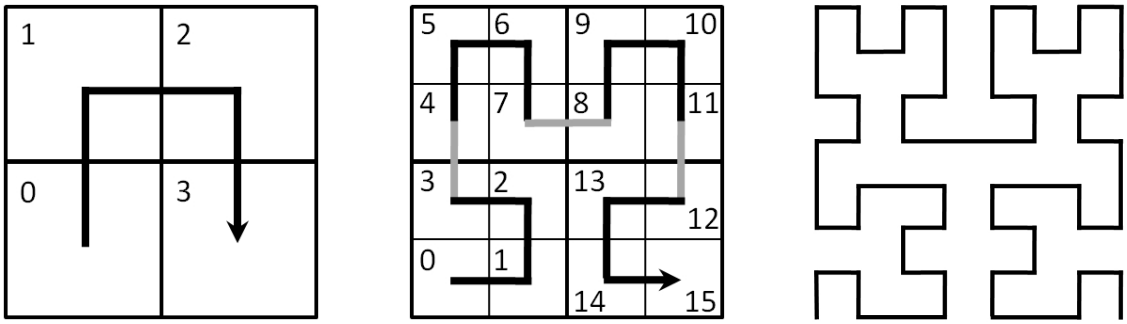

Figure 1: Hilbert curve for a 2×2 (on the left), 4×4 (middle) and 8×8 image (on the right). Pixels are counted along the curve. The second curve (and higher resolutions) can be computed (recursively) from the first one by replacing each corner by a rotated and reflected version of the basic U-shaped pattern.

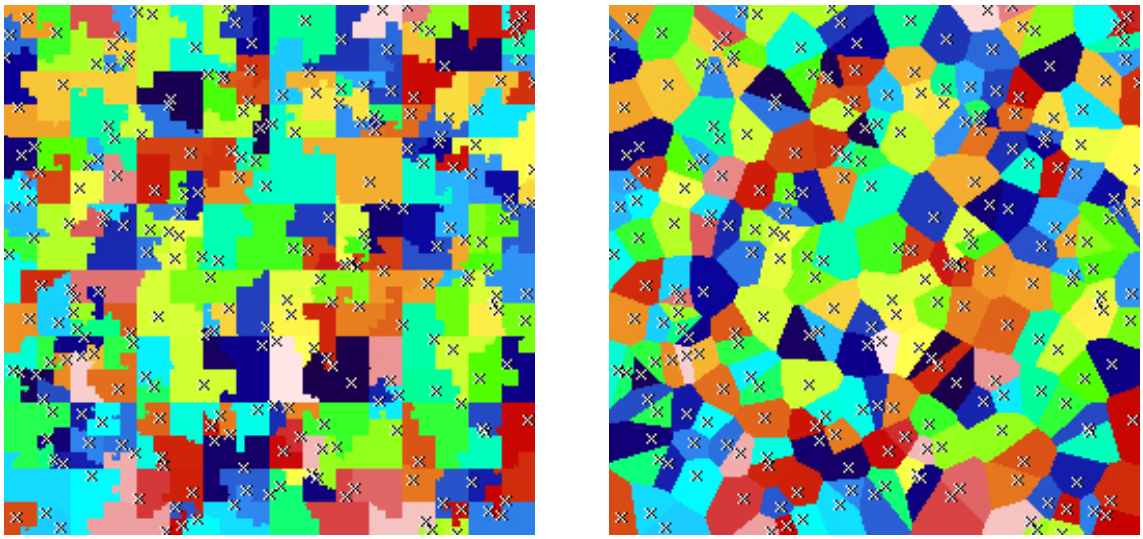

Figure 2: All nearest neighbour assingment using the proposed approximative method (left image) and an exact search (right image) for 240 keypoints (marked by crosses). The resulting cells are coloured randomly but consistent over both diagrams.