Spatial Database

Linear Quadtree – Proximity Analysis of N-order, Hilbert and Column-wise algorithms

Kuo, Shao-Heng, student number 201883361

Department of Electricity and Electronic Computer Engineering

Abstract

The linear quadtree is a way to compress data size stored in quadtree. N-order, Hilbert and Column-wise curves are three of the order decision methods available in linearization process. Three of them have different behaviors in proximity between the data before linearization (2D data) and the data after linearization (1D data). This research is to find out the difference between their proximity. It was done by two methods. First, estimate the distance between points in 2D coordinates in the order of 1D data index. Second, estimate the distance between points in 1D data, which is represented by their index. In the second method, the corresponding points in 1D data of the target point in 2D data are the points which are around the target point in the range of 3 by 3 square. Roughly, the results showed that in method 1, Hilbert and Column-wise methods got better proximity, while in method 2, N-order got the best proximity. The more detailed discussion will be given in the further part.

1. Introduction to the Three Curves

The algorithm I used to accomplish the implementation of N-order, Hilbert and Column-wise curves will be briefly explained here. N-order and Hilbert curves have some similar points in their algorithm, like they are mainly aimed at processing the square having side length with a value of 2^n, so their algorithm can be imagined as an iteration process of broadening basic $2x2$ square.

N-order curve:

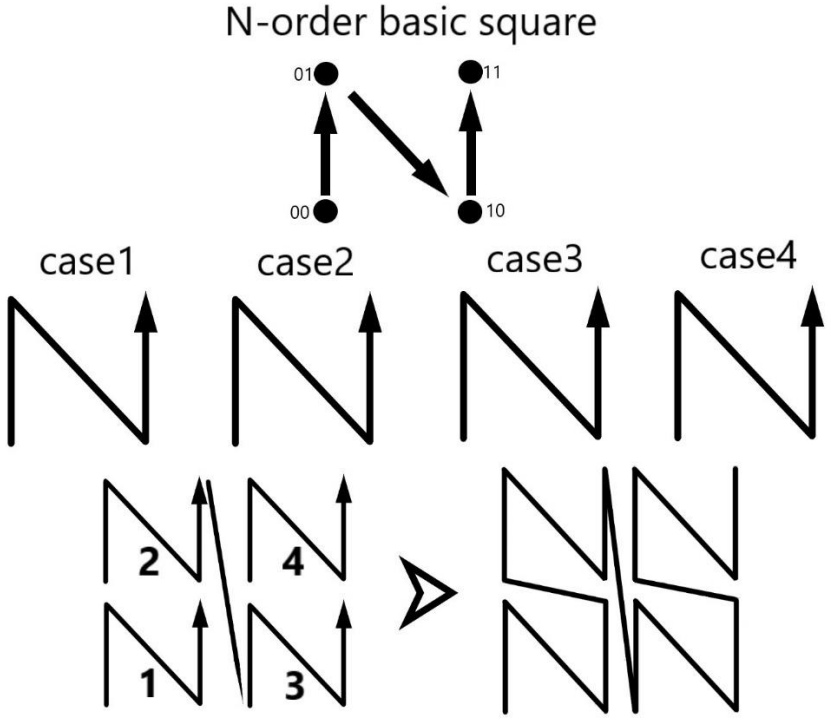


Figure 1. N-order iteration rule

If we want to draw a N-order curve, there're mainly two methods to complete it. One is to find the coordinates in 2D data from a given index of 1D data, the other is to find the index of 1D data from given coordinates in 2D data. The former will be more clearly in explaining the iteration process and will be used here.

Figure 1 shows the process of broadening the basic square to higher order of square. In the top image of Figure 1, the binary numbers are the last two digits of a given index number (represented in binary). This is due to the characteristic of N-order curve, after passing through four points, the similar N shape will repeat again. So we can determine the last two digits of index number (represented in binary), which can represent four kinds of situation. After determine the last two digits and decide which case it should be, as show in Figure 1, we can then compose them and modify their coordinates. Next step is to left-shift to the next two digits of index number and find the corresponding case. The iteration process should end when the size achieves one we wish to have.

In the process of composing and re-coordinating the basic square, the corresponding calculating equations are shown below, with the starting point $(0,0)$ at lower left,

Case1:

$$x \ and \ y \ remain \ the \ same$$

Case2:

$$x \ remains \ the \ same$$

$$y = y + n/2$$

Case3:

$$x = x + n/2$$

$$y \ remains \ the \ same$$

Case4:

$$x = x + n/2$$

$$y = y + n/2$$

where x, y are the corresponding coordinates of a given index and n is the side length of the square which we want to expand our basic square to. Now we can get all of the coordinates we need and start to draw a N-order curve.

Hilbert curve:

The Hilbert curve is similar with N-order curve, so we can apply the same iteration process described above. But the process of expanding basic square to higher order square is different from N-order. The rule of Hilbert curve is shown in Figure 2.

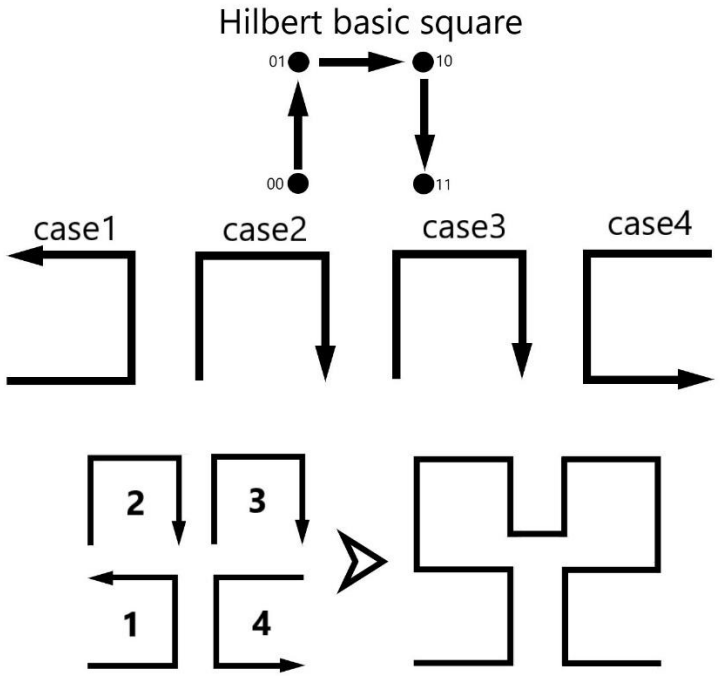


Figure 2. Hilbert iteration rule

The most different point between N-order and Hilbert is that the basic square in Hilbert needs to be overturned to fit the condition of case 1 and case 4. The corresponding calculating equations are shown below,

Case1:

$$x = y$$

$$y = x$$

Case2:

$$x \ remains \ the \ same$$

$$y = y + n/2$$

Case3:

$$x = x + n/2$$

$$y = y + n/2$$

Case4:

$$x = -y + \left(\frac{n}{2} - 1\right) + \frac{n}{2}$$

$$y = -x + (\frac{n}{2} - 1)$$

Column-wise:

Column-wise curve can be completed quite simply by adjusting x, y due to the 1D index's corresponding position in 2D data.

In odd column:

$$x = the\ \ integer\ \ part\ \ of\ \ n\ \ deviding\ \ index$$

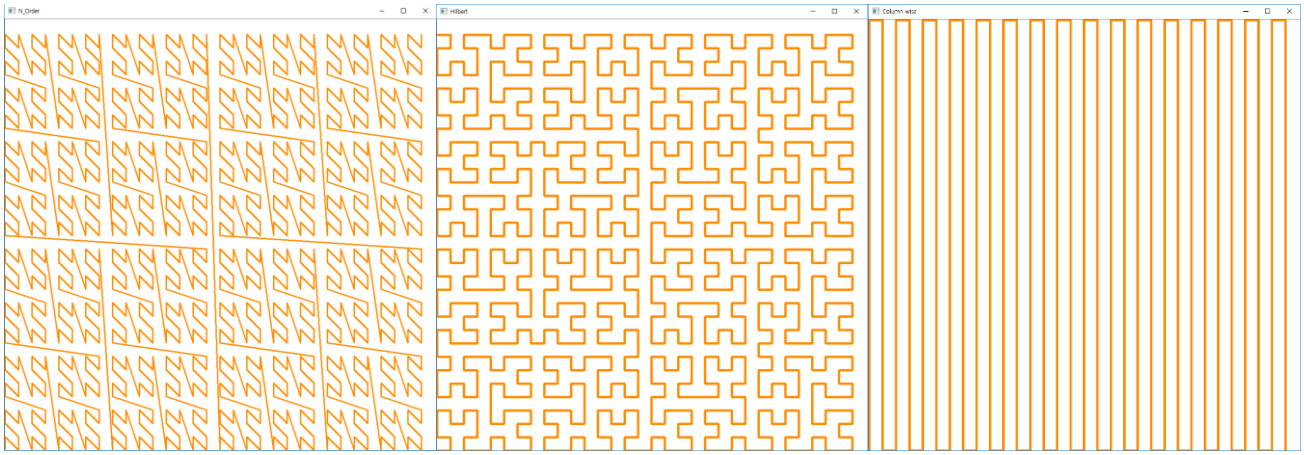
$$y = the\ \ remainder\ \ of\ \ n\ \ deviding\ \ index$$

In even column:

$$x = the\ \ integer\ \ part\ \ of\ \ n\ \ deviding\ \ index$$

$$y = (n-1)\ \ minus\ \ the\ \ remainder\ \ of\ \ n\ \ deviding\ \ index$$

An example of 32 by 32 N-order, Hilbert and Column-wise curves are shown below,



N-order, Hilbert and Column-wise curves (32 by 32)

2. Proximity Analyzing Methods

The first part gave the algorithms of each curve. Not only the index-to-coordinate can be achieved, coordinate-to-index can also be done by the same concept. When estimating the distance of points in 2D data, index-to-coordinate function is used, while estimating the distance of points in 1D data, coordinate-to-index function is used.

For the convenience of calculating the distances of points in 1D data, we assume the distances between each adjacent data in 1D data type are 1, which can be represented by their index difference, for example, the distance equals to 2 between the data with index 0 and the data with index 2. For the distances of points in 2D data, we calculate them by the coordinates acquired by our function.

Method1: Estimating distances of 2D data:

After the linearization of quadtree data, for all of the three curves described above, the distances between each adjacent data are all the same in 1D data, but it is important to find the corresponding positions in 2D data and find out their actual distance.

This method visits the data in 1D sequentially from the starting point to the last point. When visiting each data point, it calculates the actual distance between this data point and the next data point. At the end of visiting, the average 2D distance between each adjacent point pairs in 1D data will be computed.

$$D_{avg} = \frac{\sum_{i=0}^{n^2-1} D_{i \to i+1}}{n^2 - 1}$$

where D is the distance between two points and i is the 1D data index

```
The number  1 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.14
The number  2 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.31
The number  3 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.46
The number  4 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.56
The number  5 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.61
The number  6 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.64
The number  7 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.66
The number  8 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.66
The number  9 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.67
The number 10 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.67
The number 11 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.67
The number 12 average distance in ColunmWise : 1.00       Hilbert : 1.00       N_Order : 1.67
```

Figure 3. Results of 2D average distance

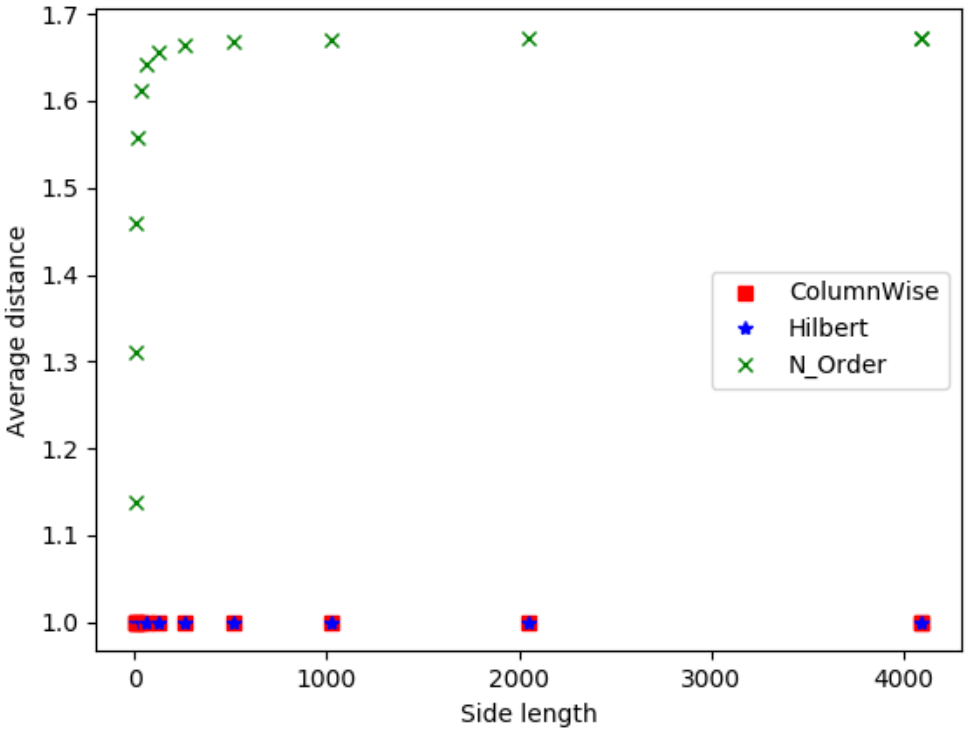


Figure 4. Results chart of 2D average distance

Figure 3 shows the results of average distance of side length varying from 2^1 to 2^{12}. The 'number' in Figure 3 refers to the power component of side length which is notated as 2^x. Figure 4 plotted the results in Figure 3.

From the results, we can find that Column-wise and Hilbert have constant average distance (with value 1) no matter how long the side length is. While in N-order, the average distance increased quickly when the side length became larger. But the average distance of N-order didn't increase unlimitedly, the average distance remained at 1.67 from side length 2^9 to 2^{12} and might be predicted to grow slowly after that.

In this method, Column-wise and Hilbert showed better proximity than N-order, when the data size became larger, their proximity difference would become more significant and will reach a limit about 1.67 times difference.

Method2: Estimating distances of 1D data:

The visiting order in method 1 is based on the sequence of 1D data points, while method 2 will find the points geometrically adjacent to each point in 2D data and then trace their corresponding points in 1D data. The method of finding adjacent points is shown below,

$$\begin{bmatrix} p_6 & p_7 & p_8 \\ p_4 & p_{target} & p_5 \\ p_1 & p_2 & p_3 \end{bmatrix}$$

where p_1 to p_8 are the desired adjacent points of p_{target}

Now we can find the corresponding index (notated as ind below) in 1D data for all of the adjacent points, and use the 1D distance calculating assumption described above, for each target point,

$$\begin{bmatrix} ind_6 & ind_7 & ind_8 \\ ind_4 & ind_{target} & ind_5 \\ Ind_1 & ind_2 & ind_3 \end{bmatrix} - \begin{bmatrix} ind_{target} & ind_{target} & ind_{target} \\ ind_{target} & ind_{target} & ind_{target} \\ ind_{target} & ind_{target} & ind_{target} \end{bmatrix}$$

so the average distance of a $n \times n$ square will be,

$$D_{avg} = \frac{\sum_{k=0}^{n^2-1} \sum_{i=1}^{8} |ind_i - ind_k|}{n^2}$$

Figure 5 and Figure 6 show the results.

```
The number  1 average distance in ColunmWise : 5          Hilbert : 5          N_Order : 5
The number  2 average distance in ColunmWise : 16         Hilbert : 16         N_Order : 15
The number  3 average distance in ColunmWise : 40         Hilbert : 42         N_Order : 37
The number  4 average distance in ColunmWise : 88         Hilbert : 97         N_Order : 83
The number  5 average distance in ColunmWise : 184        Hilbert : 210        N_Order : 176
The number  6 average distance in ColunmWise : 376        Hilbert : 440        N_Order : 366
The number  7 average distance in ColunmWise : 760        Hilbert : 903        N_Order : 747
The number  8 average distance in ColunmWise : 1528       Hilbert : 1832       N_Order : 1512
The number  9 average distance in ColunmWise : 3064       Hilbert : 3694       N_Order : 3046
The number 10 average distance in ColunmWise : 6136       Hilbert : 7421       N_Order : 6115
The number 11 average distance in ColunmWise : 12280      Hilbert : 14878      N_Order : 12256
The number 12 average distance in ColunmWise : 24568      Hilbert : 29796      N_Order : 24542
```

Figure 5. Results of 1D average distance

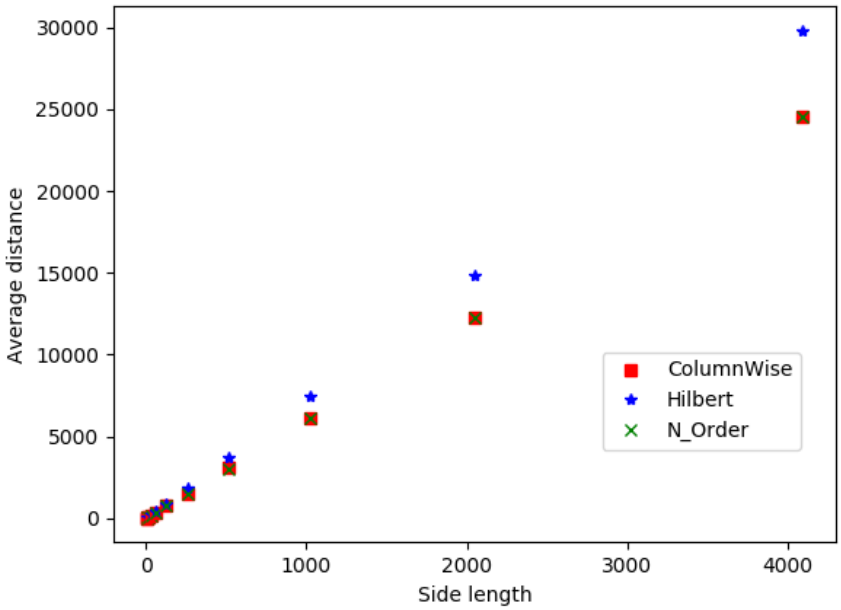


Figure 6. Results chart of 1D average distance

At side length with value 2, three of them got the same average distance. But when the side length became bigger, the average distance of Hilbert started to increase faster than the others. Column-wise and N-order almost got the same value but N-order was always slightly smaller than Column-wise in the whole process. Figure 7 shows the average distance difference percentage between Hilbert and N-order.

$$Distance\ \ difference = \frac{D_{Hilbert} - D_{N-order}}{D_{N-order}} \times 100\%$$

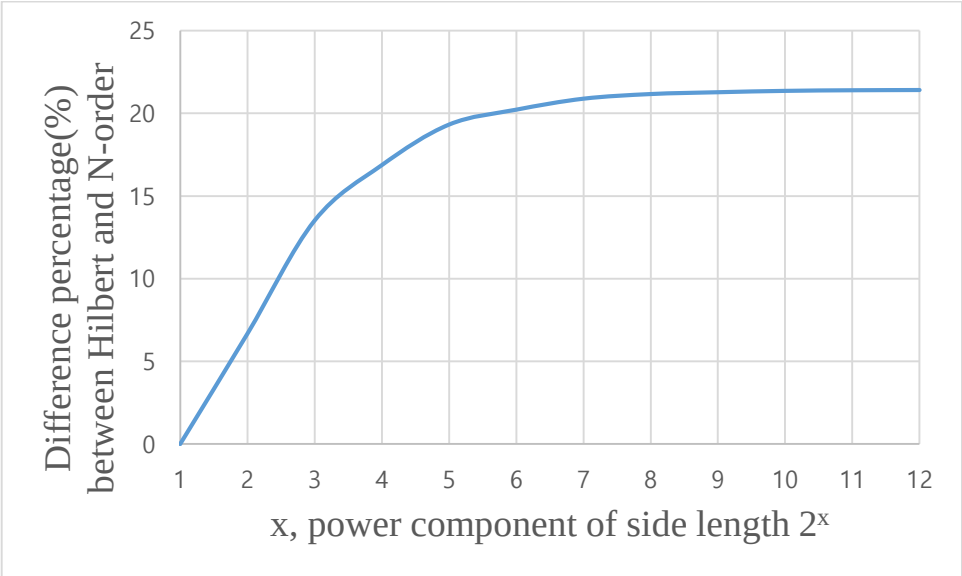


Figure 7. Average distance difference between Hilbert and N-order

In this method, Hilbert got worse proximity compared with the others, while Column-wise and N-order got the extremely near proximity.

3. Discussion and Conclusion

One imperfection in this analysis is that, in method2, we treated all of the adjacent points of the target point as they all have same distance with target point. But actually the diagonal ones have longer distance with target point, so method 2 has to be improved if we take this into consideration and then have further discussion.

To sum up the results of method1 and method2, column-wise got the best proximity in the overall results of the two methods, it reflects the characteristic of column-wise, which sequentially labels the data points of 2D data in the order of columns from left to right and there's no jumping steps between points. These may explain why column-wise got the best results during proximity analysis.

For the comparison of N-order and Hilbert curves, Hilbert got better results in method1 while N-order got better ones in method2. From the results we got in method1, N-order got 1.67 times longer average distance than Hilbert under the circumstances that side length is large, this might be explained by the frequent appearance of jumping steps in N-order. In method2, from Figure 7 we can find out that Hilbert got about 21% larger average distance in higher order squares, this difference is not as significant as that in method1. Based on the overall results, we might still say that Hilbert has a better proximity than N-order does, since Hilbert lead a much more significant better results in method1 (2D data distances).

4. References

1. Ki-Joune Li. 2018. Spatial Databases – Representation. Pusan National University, Department of Computer Science and Engineering.

2. Iterative algorithm for drawing Hilbert curve.

[https://marcin-chwedczuk.github.io/iterative-algorithm-for-drawing-hilbert-curve]. Accessed March 28, 2018.