International Journal of Engineering & Scientific Research
Vol. 10 Issue 6, June 2022,
ISSN: 2347-6532 Impact Factor: 6.660
Journal Homepage: http://esrjournal.com, Email: esrjeditor@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Comparing Time execution of Space Filling Curve using QHT RHT THT on arbitrary matrix

Tianandrasana Romeo Rajaonarison[*]
Paul AugusteRandriamitantsoa[**]

Abstract

The Space Filling curve visits all points of matrix only on one path. The Hilbert path on matrix is a scrambling technique using Space Filling Curve. Many algorithms are proposed like Quantum Hilbert Transform (QHT), Recursive Hilbert Transform (RHT) and Tensor Hilbert Transform (THT). Hilbert transform could only applied on matrix sized$2^n \times 2^n$ so the matrix should be cutted on efficiency multiple area using left-up (lu) or right-up (ru) or left-down (ld) and right-down (rd) priority. All curve are optimized using pchip (Piecewise Cubic Hermite Interpolating Polynomial) interpolation. All the result is simulated at Matlab using classic computing. The time evaluation varied by order: QHT, RHT and THT. All curve QHT has a bit similitude for lu-qht, ru-qht, ld-qht, rd-qht. The curve RHT usinglu-rht, ru-rht, ld-rht, rd-rht and THT usinglu-ttht, ru-tht, ld-tht, rd-tht have also the same characteristic between them. The THT gives us a good time performance between them.

Keywords:

SFC
QHT
RHT
THT
Matlab

Author correspondence:

Tianandrasana Romeo Rajaonarison,
Department of Telecommunication,
High School Polytechnics of Antananarivo, university of Madagascar
Email: rajaonarisonromeo11@gmail.com

1. Introduction

Space filling curve (SFC) [1] [2][3] [4] [5]consist to visit all points of the matrix in a specific order like example: Morton, Peano, Hilbert. In 1890, the mathematician G. Peano presents family curve of SFC. After that, the SFC are using at the matrix scrambling. Using quantum computer, it's possible to implement the Hilbert transform using quantum gate logic. This algorithm is named QHT (Quantum Hilbert Transform). Using classic computer, it's also possible to implement the algorithm by 2ways: using Tensor recursive RHT (Recursive Hilbert Transform) and tensor iterative THT (Tensor Hilbert Transform). Besides, the algorithm Hilbert transform is possible only to matrix sized $2^n \times 2^n$, the decomposition by left-up (lu), right-up,(ru) left-down(ld) and right-down(rd) are proposed to give us a maximum area decomposition. Simulate all the implementation in Matlab using classic computing conduct us to ask which algorithm has efficient time evaluation. This article compared also the time implementation of QHT, RHT and THT when the computer use lu, ru, ld, rd.

[*]Department of Telecommunication, High School Polytechnics of Antananarivo
[**]Department of Telecommunication, High School Polytechnics of Antananarivo

2. Research Method

I provide all code of the implementation at my githubalcinoos [5].

2.1 Quantum Circuit Gate

The different quantum circuit gate for quantum computing is presented like on the Figure 1. Each quantum gate has his own transfert function [2] [3] [4] [6] [7] [8] [9].

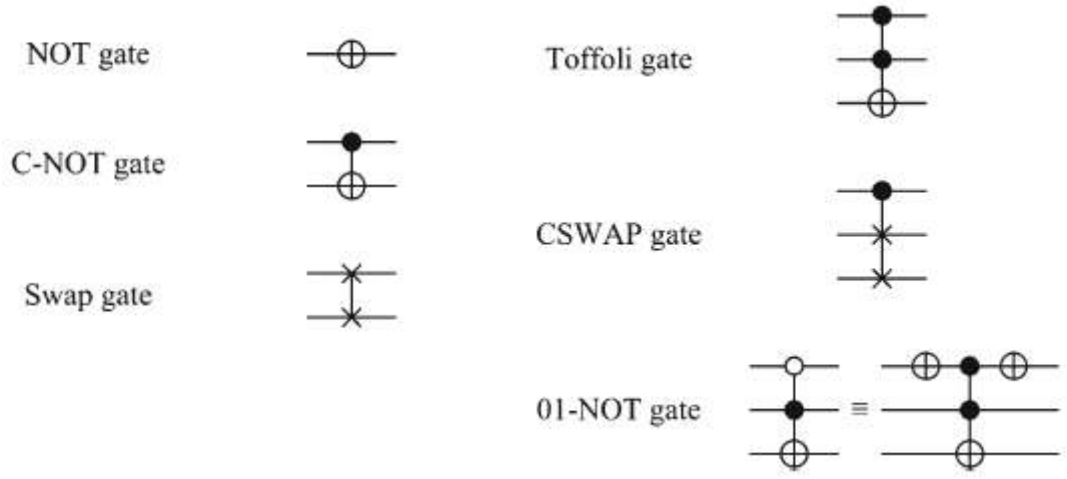


Fig1: Quantum Circuit

2.2 FRQI representation

All matrix could be represented like FRQI[2] [3] [4] [6] [7] [8] [9] defined by formula (1) and (2)

$$|I(\theta)\rangle = \frac{1}{2^n}\sum_{i=0}^{2^{2n}-1} |C_i\rangle \otimes |i\rangle \quad (1)$$

$$|C_i\rangle = \cos(\theta_i) + j\sin(\theta_i), \theta_i = \left[0; \frac{2}{\pi}\right], i = 0,1,\dots,2^{2n}-1 \quad (2)$$

$|C_i\rangle$Thecomposant of the matrix
$|i\rangle$Linearized index of the matrix
$|I(\theta)\rangle$Representation of the matrix using FRQI

2.3 Hilbert Transform

The Hilbert transform H_n is a permutation recursive represented like on Figure 2.

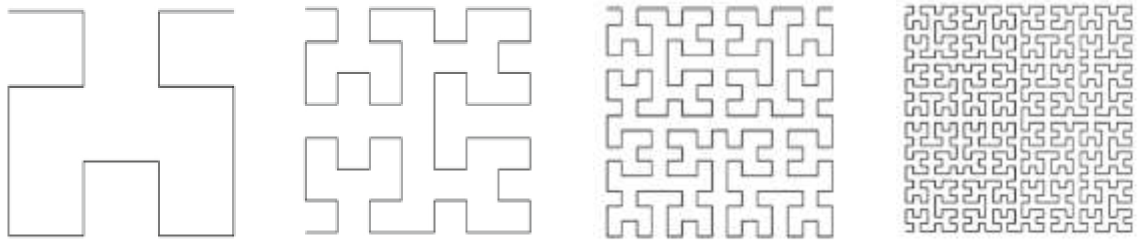


Fig 2:Graphics representation of Hilbert Transform

2.4 Quantum Hilbert Transform

The Qunatum Hilbert Transform could be implemedusig matrix conversion and algorithm if it's even or odd [2] [3] [4] [6] [7] [8] [9].

2.4.1 Transformation of Matrix

To realize the Quantum Hilbert Transform, all this matrix transform is needed: transpose, up-down, left-right, and central rotation noted respectively by $A^T, A^{ud}, A^{lr}, A^{pp}$:

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,m} \end{bmatrix}$$

So,

$$A^T=\begin{bmatrix} a_{1,1} & a_{2,1} & \dots & a_{m,1} \\ a_{1,2} & a_{2,2} & \dots & a_{m,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1,m} & a_{2,m} & \dots & a_{m,m} \end{bmatrix};\ A^{lr}=\begin{bmatrix} a_{1,m} & \dots & a_{1,2} & a_{1,1} \\ a_{2,m} & \dots & a_{2,2} & a_{2,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,m} & \dots & a_{m,2} & a_{m,1} \end{bmatrix};$$

$$A^{ud}=\begin{bmatrix} a_{m,1} & a_{m,2} & \dots & a_{m,m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ a_{1,1} & a_{1,2} & \dots & a_{1,m} \end{bmatrix};\ A^{pp}=\begin{bmatrix} a_{m,m} & \dots & a_{m,2} & a_{m,1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{2,m} & \dots & a_{2,2} & a_{2,1} \\ a_{1,m} & \dots & a_{1,2} & a_{1,1} \end{bmatrix}$$

The recursive path of Hilbert transform is defined by the Formula:

$$H_{n+1}=\begin{cases}\begin{pmatrix} H_n & 4^nE_n+H_n^T \\ (4^{n+1}+1)E_n-H_n^{ud} & (3.4^n+1)E_n-(H_n^{lr})^T \end{pmatrix} & if\ n\ even \\ \begin{pmatrix} H_n & (4^{n+1}+1)E_n-H_n^{lr} \\ 4^nE_n+H_n^T & (3.4^n+1)E_n-(H_n^T)^{lr} \end{pmatrix} & if\ n\ odd \end{cases} \tag{3}$$

Withn an integer positif and $H_1=\begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$ and$E_n=\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$

For the realization of Quantum Hilbert Transform, all these properties are considered:
If A, B, C, D, four matrix $m\times m$, so,

1. $(A+B)^{pp}=A^{pp}+B^{pp}$ (4)
2. $(A+B)^{lr}=A^{lr}+B^{lr}$ (5)
3. $(A+B)^{ud}=A^{ud}+B^{ud}$ (6)
4. $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{lr}=\begin{pmatrix} B^{lr} & A^{lr} \\ D^{lr} & C^{lr} \end{pmatrix}$ (7)
5. $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{ud}=\begin{pmatrix} C^{ud} & D^{ud} \\ A^{ud} & B^{ud} \end{pmatrix}$ (8)
6. $\begin{pmatrix} A & B \\ C & D \end{pmatrix}^{pp}=\begin{pmatrix} D^{pp} & C^{pp} \\ B^{pp} & A^{pp} \end{pmatrix}$ (9)
7. $(A^T)^{ud}=(A^{lr})^T$ (10)
8. $A^{ud}=((A^T)^{lr})^T$ (11)
9. $(A^{ud})^{pp}=(A^{pp})^{ud}=A^{lr}$ (12)
10. $(A^{lr})^{pp}=(A^{pp})^{lr}=A^{ud}$ (13)
11. Using the recursive form we could express the Hilbert Transform like this:

$$H_{n+1}=\begin{cases}\begin{pmatrix} H_n & (H_n+4^nE_n)^T \\ (H_n+3.4^nE_n)^{pp} & (H_n+2.4^n)^T \end{pmatrix} & if\ \ n\ even \\ \begin{pmatrix} H_n & (H_n+3.4^nE_n)^{pp} \\ (H_n+4^nE_n)^T & (H_n+2.4^nE_n)^T \end{pmatrix} & if\ n\ odd \end{cases} \tag{14}$$

If n is a positive integer, the initial matrix is $H_1=\begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}$ and $E_n=\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$

12. In the case of, n even, the 4 expressions should be identic:
(i) $H_n=H_n$
(ii) $4^nE_n+H_n^T=(H_n+4^nE_n)^T$
(iii) $(4^{n+1}+1)E_n-H_n^{ud}=(H_n+4^nE_n)^T$
(iv)$(3.4^n+1)E_n-(H_n^T)^{lr}=(H_n+2.4^nE_n)^T$

The expression (i) is an evident identity, so we don't need demonstration,
For the expression (ii),

$$(H_n + 4^n E_n)^T = H_n^T + (4^n E_n)^T = 4^n E_n + H_n^T$$

For the expression (iii),

$$H_n + H_n^{lr} = (4^n + 1)E_n$$
$$(H_n + H_n^{lr})^{pp} = H_n^{pp} + H_n^{ud} = [(4^n + 1)E_n]^{pp} = (4^n + 1)E_n$$
$$(4^n + 1)E_n - H_n^{ud} = H_n^{pp}$$
$$(4^n + 1)E_n - H_n^{ud} + 3.4^n E_n = H_n^{pp} + 3.4^n E_n$$
$$(4^{n+1} + 1)E_n - H_n^{ud} = H_n^{pp} + 3.4^n E_n$$
$$(4^{n+1} + 1)E_n - H_n^{ud} = H_n^{pp} + (3.4^n E_n)^{pp}$$
$$(4^{n+1} + 1)E_n - H_n^{ud} = (H_n + 3.4^n E_n)^{pp}$$

For the expression (iv),

$$H_n + H_n^{lr} = (4^n + 1)E_n$$
$$(H_n + H_n^{lr})^T = H_n^T + (H_n^{lr})^T = [(4^n + 1)E_n]^T = (4^n + 1)E_n$$
$$(4^n + 1)E_n - (H_n^{lr})^T = H_n^T$$
$$(4^n + 1)E_n - (H_n^{lr})^T + 2.4^n E_n = H_n^T + 2.4^n E_n$$
$$(3.4^n + 1)E_n - (H_n^{lr})^T = H_n^T + 2.4^n E_n = H_n^T + (2.4^n E_n)^T$$
$$(3.4^n + 1)E_n - (H_n^{lr})^T = (H_n + 2.4^n E_n)^T$$

If n even,

$$H_{n+1} = \begin{pmatrix} H_n & (H_n + 4^n E_n)^T \\ (H_n + 3.4^n E_n)^{pp} & (H_n + 2.4^n)^T \end{pmatrix}$$

In the case, nodd, all the 4 expressions should be identic:
(i) $H_n = H_n$
(ii) $(4^{n+1} + 1)E_n - H_n^{lr} = (H_n + 3.4^n E_n)^{pp}$
(iii) $4^n E_n + H_n^T = (H_n + 4^n E_n)^T$
(iv)$(3.4^n + 1)E_n - (H_n^{lr})^T = (H_n + 2.4^n)^T$
For the expression (i),$H_n = H_n$,it's a general truth, no nedd demonstration
For the expression (ii),

$$H_n + H_n^{ud} = (4^n + 1)E_n$$
$$(H_n + H_n^{ud})^{pp} = H_n^{pp} + H_n^{lr} = \left((4^n + 1)E_n\right)^{pp} = (4^n + 1)E_n$$
$$(4^n + 1)E_n - H_n^{lr} = H_n^{pp}$$
$$(4^n + 1)E_n - H_n^{lr} = H_n^{pp}$$
$$(4^n + 1)E_n - H_n^{lr} + 3.4^n E_n = H_n^{pp} + 3.4^n E_n$$
$$(4^n + 1)E_n - H_n^{lr} + 3.4^n E_n = H_n^{pp} + (3.4^n E_n)^{pp}$$
$$(4^{n+1} + 1)E_n - H_n^{lr} = H_n^{pp} + (3.4^n E_n)^{pp}$$
$$(4^{n+1} + 1)E_n - H_n^{lr} = (H_n + 3.4^n E_n)^{pp}$$

For the expression (iii), $(H_n + 4^n E_n)^T = H_n^T + (4^n E_n)^T = H_n^T + 4^n E_n = 4^n E_n + H_n^T$
For the expression (iv),

$$H_n + H_n^{ud} = (4^n + 1)E_n$$
$$(H_n + H_n^{ud})^T = H_n^T + (H_n^{ud})^T = H_n^T + (H_n^T)^{lr} = [(4^n + 1)E_n]^T = (4^n + 1)E_n$$
$$(4^n + 1)E_n - (H_n^T)^{lr} = H_n^T$$
$$(4^n + 1)E_n - (H_n^T)^{lr} + 2.4^n E_n = H_n^T + 2.4^n E_n$$
$$(4^n + 1)E_n - (H_n^T)^{lr} + 2.4^n E_n = H_n^T + (2.4^n E_n)^T$$
$$(3.4^n + 1)E_n - (H_n^T)^{lr} = H_n^T + (2.4^n E_n)^T$$
$$(3.4^n + 1)E_n - (H_n^T)^{lr} = (H_n + 2.4^n E_n)^T$$

When n odd,

$$H_{n+1} = \begin{pmatrix} H_n & (H_n + 3.4^n E_n)^{pp} \\ (H_n + 4^n E_n)^T & (H_n + 2.4^n E_n)^T \end{pmatrix}$$

2.4.2 Quantum Circuit of Hilbert Transform

Based on the recursive theorem, the Quantum Hilbert Transform is dived on 3 part:initialization, part odd and part even. Two methods could be proposed:
1. The result of the recursivepath of Hilbert on $H_n$need to compute the matrix H_n, and swap directly the pixel on $(\lfloor H_n(i,j) - 1 \rfloor / 2^n, H_n(i,j) - 1 \; mod \; 2^n)$ with(i,j)
2. Using the partition ofH_n, consists to make partition of each size 2*2, 4*4, 8*8, 16*16 … The size to the output of the image is always $2^n * 2^n$. This method needs partition of each steps: initialization, part odd and part even.

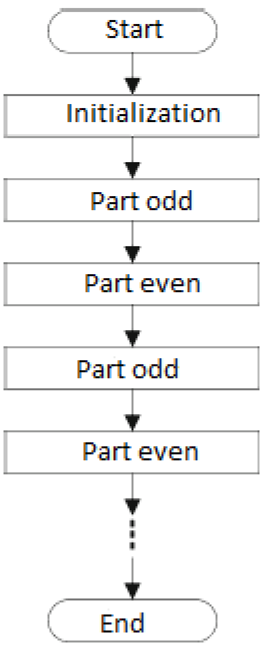


Fig 3. Quantum Hilbert Transform algorithm

The method partition(k) consists to subdivide $2^n * 2^n$size to input of $2^{n-k-1} * 2^{n-k-1}$bloc sized$2^{k+1} * 2^{k+1}$.

- For the first step k =0,matrix is partitioned to$2^{n-1} * 2^{n-1}$bloc sized $2 * 2$.
- For the second step k=1, matrix is partitioned to$2^{n-2} * 2^{n-2}$bloc sized $4 * 4$.
- For the third step k=2, matrix is partitioned to$2^{n-2} * 2^{n-2}$bloc sized $8 * 8$.

…

- For the n-th step k=n-1, matrix is partitioned to 1bloc sized$2^n * 2^n$.

2.4.3. Partition(k)

Partition(k) is dived in two steps :

(1) Swap$x_{k+1}$and x_{k+2} ; x_{k+2} andx_{k+3},….x_{n-2} andx_{n-1}

(2) Swapx_{n-1} andy_k

The quantum circuit swap gate is needed to swap 2 quantum state for having the quantum circuit on Figure 4.

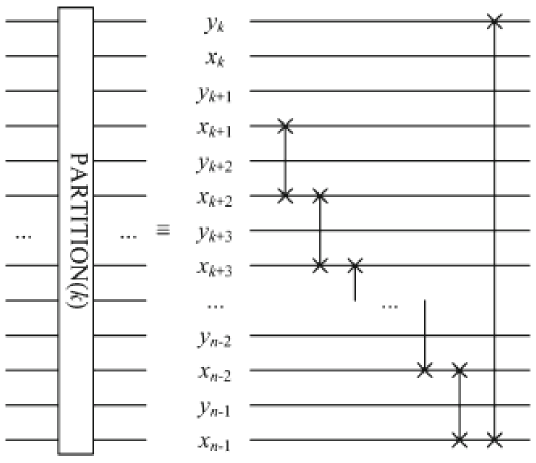
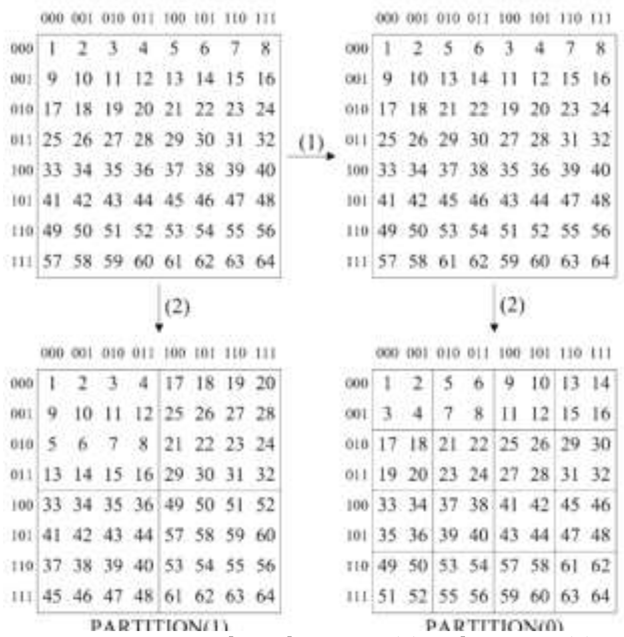


Fig4: Partition (k)

For each dimension$|y\rangle$ *and* $|x\rangle$, the matrix $|i\rangle$ could be expressed by:$|i\rangle = |y\rangle|x\rangle$


(1)
(2)
(2)
PARTITION(1)
PARTITION(0)

	000	001	010	011	100	101	110	111
000	1	2	3	4	5	6	7	8
001	9	10	11	12	13	14	15	16
010	17	18	19	20	21	22	23	24
011	25	26	27	28	29	30	31	32
100	33	34	35	36	37	38	39	40
101	41	42	43	44	45	46	47	48
110	49	50	51	52	53	54	55	56
111	57	58	59	60	61	62	63	64

	000	001	010	011	100	101	110	111
000	1	2	5	6	3	4	7	8
001	9	10	13	14	11	12	15	16
010	17	18	21	22	19	20	23	24
011	25	26	29	30	27	28	31	32
100	33	34	37	38	35	36	39	40
101	41	42	45	46	43	44	47	48
110	49	50	53	54	51	52	55	56
111	57	58	61	62	59	60	63	64

	000	001	010	011	100	101	110	111
000	1	2	3	4	17	18	19	20
001	9	10	11	12	25	26	27	28
010	5	6	7	8	21	22	23	24
011	13	14	15	16	29	30	31	32
100	33	34	35	36	49	50	51	52
101	41	42	43	44	57	58	59	60
110	37	38	39	40	53	54	55	56
111	45	46	47	48	61	62	63	64

	000	001	010	011	100	101	110	111
000	1	2	5	6	9	10	13	14
001	3	4	7	8	11	12	15	16
010	17	18	21	22	25	26	29	30
011	19	20	23	24	27	28	31	32
100	33	34	37	38	41	42	45	46
101	35	36	39	40	43	44	47	48
110	49	50	53	54	57	58	61	62
111	51	52	55	56	59	60	63	64

Fig 5: Figure explained partition(1) and partition(0)

2.4.4 Le module O (k)

Having 4 matrix A, B, C, D sized $2^{k-1} * 2^{k-1}$; The function O(k) Transforms the matrix $\begin{pmatrix} A & B \\ C & D \end{pmatrix}to\begin{pmatrix} A & D^{pp} \\ B^T & C^T \end{pmatrix}$for the same wayof the recursive form of Hilbert path when n is odd. O(k) is used so when k is odd. Divide this steps in three moduleslike:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \xrightarrow{step(1)} \begin{pmatrix} A & D \\ B & C \end{pmatrix} \xrightarrow{step(2)} \begin{pmatrix} A & D \\ B^T & C^T \end{pmatrix} \xrightarrow{step(3)} \begin{pmatrix} A & D^{pp} \\ B^T & C^T \end{pmatrix}$$

Step1:Using C-NOT gate and CSWAP gate
Step2:Using CSWAP gate
Step3:Using 01-NOT and 0-Control gate

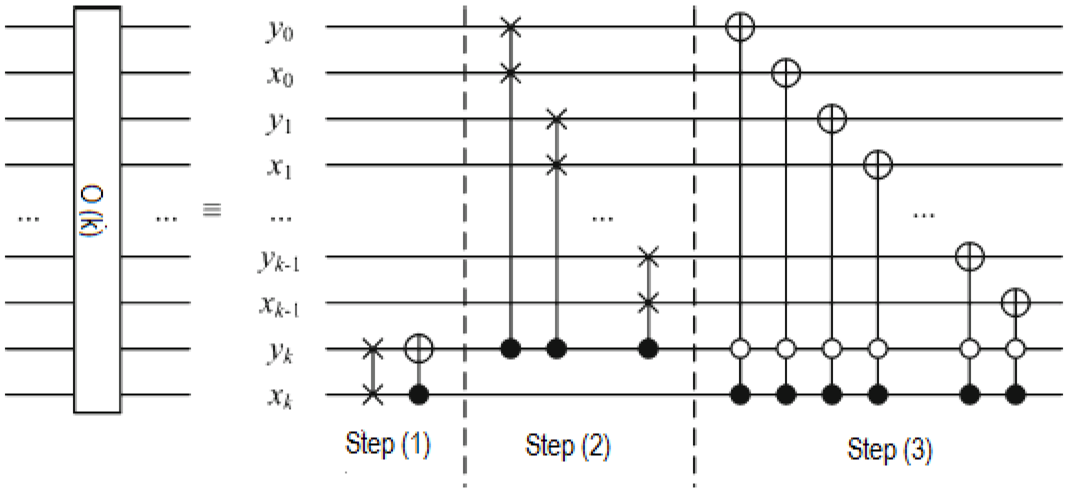


Fig 6. Odd Quantum Circuit module O(k)

2.4.5. The module E(k)

Having 4 matrix A, B, C, D sized $2^{k-1} * 2^{k-1}$; The function E(k) Transforms the matrix $\begin{pmatrix} A & B \\ C & D \end{pmatrix}to\begin{pmatrix} A & B^T \\ D^{pp} & C^T \end{pmatrix}$for the same manière of the recursif form of Hilbert path when n is odd. E(k) is used so when k is even. Divide this steps in three moduleslike:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \xrightarrow{Step(1)} \begin{pmatrix} A & B \\ D & C \end{pmatrix} \xrightarrow{Step(2)} \begin{pmatrix} A & B^T \\ D & C^T \end{pmatrix} \xrightarrow{Step(3)} \begin{pmatrix} A & B^T \\ D^{pp} & C^T \end{pmatrix}$$

Step 1 : Using C-NOT gate

Step2 :Using CSWAP gate
Step3 :Using 01-NOT and 0-Control gate

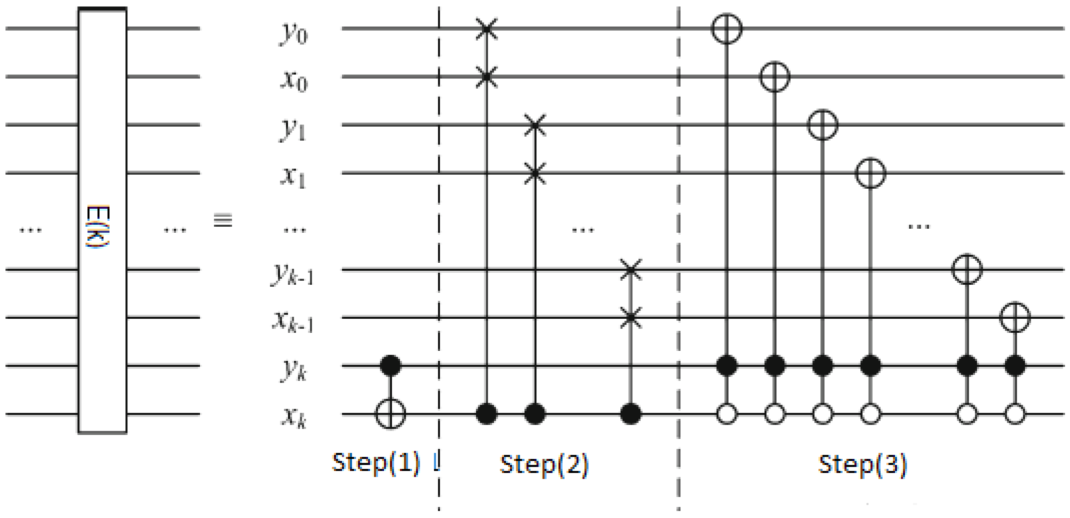


Fig 7: Even Quantum Circuit module ModuleE(k)

2.4.6. Module Initialization
The module initialization is by definition the partition (0). The Classic Hilbert transform, used like initialization the matrix H_1. Thepartition(0) will divide the matrix input $2^n * 2^n$ to$2^{n-1} * 2^{n-1}$size 2*2 which has the same size of H_1.

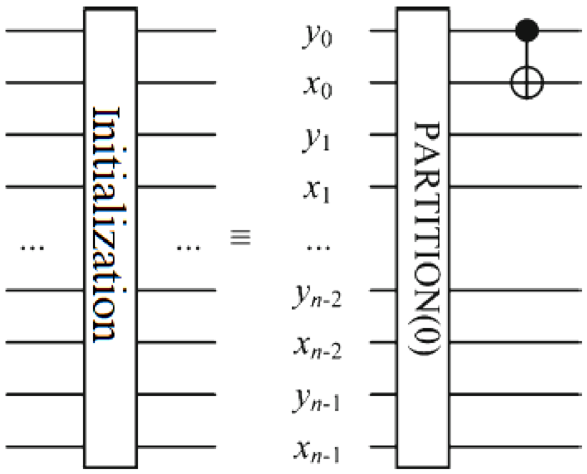


Fig 8: Quantum circuit for initialization

2.4.7. Module part even and Part Odd
For having all different pixels localizations, the module partition is before the E(k) giving the module part even and the module partition is before the O(k) giving the module part odd.

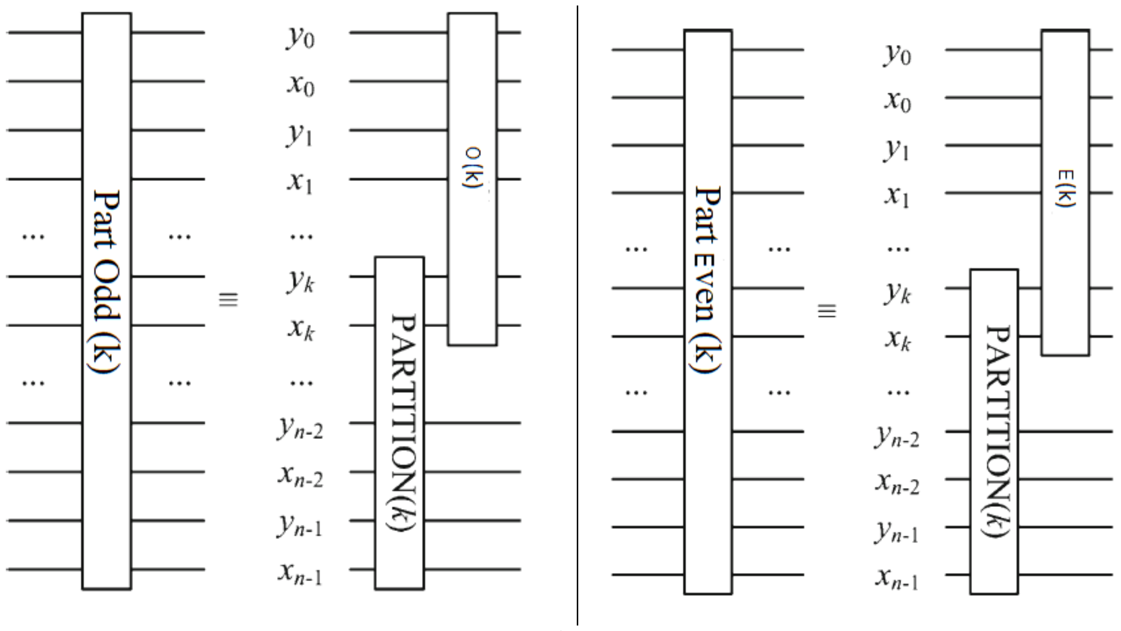


Fig 9: Quantum Circuit for part even and part odd

2.4.8. Quantum circuit ofQHT and inverse circuit

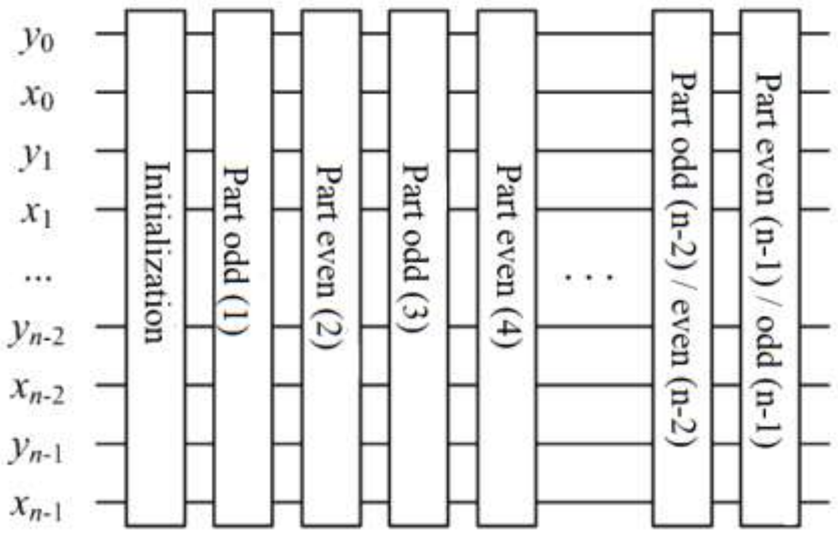
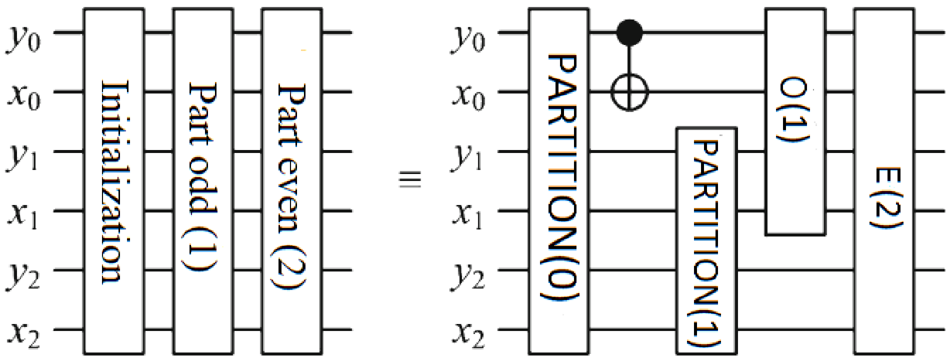


Fig 10: Quantum QHT

To realize the quantum circuit, The module of initialization – part odd – part even will be alternate. To recover the matrix on the output, the order are inversed. For n = 3, we could have :



$\equiv$

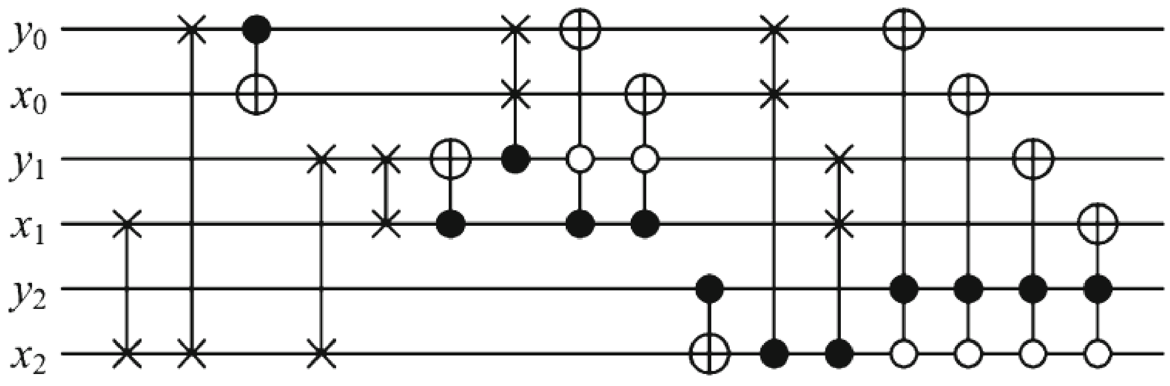


Fig11 : Quantum Hilbert Transform for n = 3

2.5. RHT and THT

The tensor product konwn as product of Kroneckeris used on algorithm like: fast Fourier transform, Strassen multiplication, parallel computational, Karatsubamultiplication and also Hilbert Transform.

Having A and B, two matrix of dimension m*n and p*q, respectively. The tensor product A and B is a bloc of matrix obtained by replacing each element$a_{i,j}$ by$a_{i,j}B$. Using tensor product, it's possible to make version recursif RHT and version simple iterativeTHT [10] [11] [12].

2.5.1. Tensor Product

The tensor product is defined by:

$$A \otimes B = \begin{bmatrix} a_{0,0}B_{p\times q} & \cdots & a_{0,n-1}B_{p\times q} \\ \vdots & \ddots & \vdots \\ a_{m-1,0}B_{p\times q} & \cdots & a_{m-1,n-1}B_{p\times q} \end{bmatrix} \quad (15)$$

$F^n$is a vector space formed by n-uplet in F to$F^{m\times n}$which is a vector space with$m \times n$ matrix. The collection of element$\{e_i^m | 0 \le i < m\}$ with$e_i^m$is a vector of i-th position and zero apart.

2.5.2. Tensor Base

HavingF^na space vector formed by n-uplet in F. The collection of element:

$\left\{e_{i_1}^{n_1} \otimes e_{i_2}^{n_2} \otimes \ldots \otimes e_{i_k}^{n_k} | 0 \le i_1 \le n_1, 0 \le i_2 \le n_2, \ldots, 0 \le i_k \le n_k\right\}$is named based tensor of $F^{n_1} \otimes F^{n_2} \otimes \ldots \otimes F^{n_k}$

The base tensor could be factorized or linearized by:

$$e_{i_1}^{n_1} \otimes e_{i_2}^{n_2} \otimes \ldots \otimes e_{i_k}^{n_k} = e_{i_1 n_2 \ldots n_k + \cdots + i_{k-1} n_k + i_k}^{n_1 n_2 \ldots n_k} \quad (16)$$

2.5.3. Direct sum

Having A and B two matrix of dimensions m*n and p*q, respectively. The direct sum of matrix A and B is $(m+p) \times (n+q)$ matrix defined by :

$$A \oplus B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} \quad (17)$$

We have also,

$$I_n \otimes B = \bigoplus_{k=0}^{n-1} B = \begin{bmatrix} B & & \\ & B & \\ & & B \end{bmatrix} \quad (18)$$

The path of Hilbert uses 3 permutations:

- « Stride permutation »
- « Reverse permutation »
- « Gray Permutation »

2.5.4. Stride permutation

« Stride permutation » noted by$L_n^{mn}\left(e_i^m \otimes e_j^n\right)$is defined by :

$$L_n^{mn}\left(e_i^m \otimes e_j^n\right) = e_j^n \otimes e_i^m \quad (19)$$

$L_n^{mn}\left(e_i^m \otimes e_j^n\right)$Refered about the tensor product of two bases. It's should also exchange the coordinate on the system.

2.5.5. Reverse permutation

« Reverse permutation » noted by$J_n$is defined by:

$$J_n e_i^n = e_{(n-1)-i}^n \tag{20}$$

J_n mapped the elements of the base e_i^n en $e_{-(i+1)modn}^n$. This permutation is indeed used on the notation of « Gray Permutation ».

« Reverse permutation » for a vector of length 2^kcould be interpreted like the complementary binary operation because:

$$J_{2^k}\left(e_{i_{k-1}}^2 \otimes \dots \otimes e_{i_0}^2\right) = e_{\overline{i_{k-1}}}^2 \otimes \dots \otimes e_{\overline{i_0}}^2 \tag{21}$$

With $\overline{i_j}$is the complementary binary of i_j and$0 \leq j < n$

2.5.6. Gray permutation

The n-bit in code gray permutation noted by $G_2^n$is defined by:

$$G_2 = I_2, G_2^n = (I_{2^{n-1}} \oplus J_{2^{n-1}})(I_2 \otimes G_2^{n-1}) \tag{22}$$

So,

$$G_2^3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Propriety

1. Associativity

$$A \otimes B \otimes C = (A \otimes B) \otimes C = A \otimes (B \otimes C)$$

$A \oplus B \oplus C = (A \oplus B) \oplus C = A \oplus (B \oplus C)$

2. Multiplication

$(A_1 \otimes A_2 \otimes \dots \otimes A_k)(B_1 \otimes B_2 \otimes \dots \otimes B_k) = A_1 B_1 \otimes A_2\ B_2 \otimes \dots \otimes A_k B_k$

$(A_1 \oplus A_2 \oplus \dots \oplus A_k)(B_1 \oplus B_2 \oplus \dots \oplus B_k) = A_1 B_1 \oplus A_2\ B_2 \oplus \dots \oplus A_k B_k$

3. Multiplication two by two

$(A_1 \otimes B_1)(A_2 \otimes B_2) \dots (A_k \otimes B_k) = (A_1 A_2 \dots A_k) \otimes (B_1 B_2 \dots B_k)$

$(A_1 \oplus B_1)(A_2 \oplus B_2) \dots (A_k \oplus B_k) = (A_1 A_2 \dots A_k) \oplus (B_1 B_2 \dots B_k)$

4. Inverse operation

$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$

$(A \oplus B)^{-1} = A^{-1} \oplus B^{-1}$

5. Distributive by right

$$\prod_{i=0}^{n-1} (I_n \otimes A_i) = I_n \otimes \left(\prod_{i=0}^{n-1} A_i \right)$$

6. Distributive by left

$$\prod_{i=0}^{n-1} (A_i \otimes I_n) = \left(\prod_{i=0}^{n-1} A_i \right) \otimes I_n$$

7. Inverse of « Stride permutation »

$(L_n^{mn})^{-1} = L_n^{mn}, L_n^n = I_n$

2.5.7 Recurrence by the Hilbert Path

The Hilbert path is possible only on matrix sized $2^n \times 2^n$. Typically, $2^n \times 2^n$of Hilbert path is obtained by recursive of $2^{n-1} \times 2^{n-1}$.

For n=1, the bloc$2 \times 2$is namedH_1 ; and n = 2, the bloc$4 \times 4$is namedH_2 represented byFigure 12. The main goal is to define as recursive formula of Hilbert transformH_n by using tensor product.

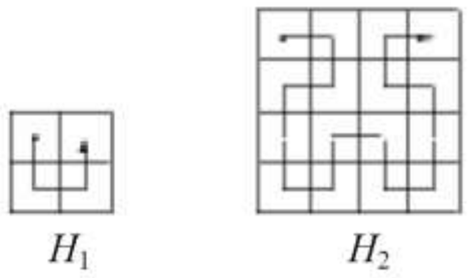
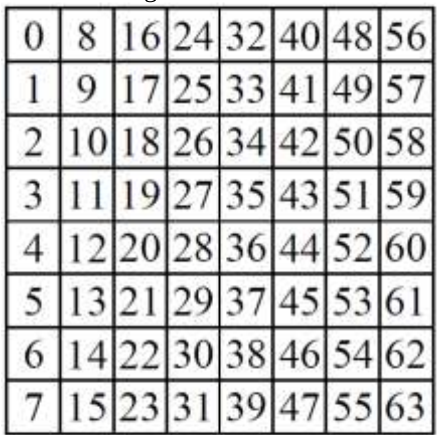


Fig 12: 2 × 2 and4 × 4Space Filling Curve

2.5.8. Recursive Hilbert Transform and Tensor Hilbert Transform:

Suppposed all points in $2^n \times 2^n$are initially saved by increased order of the column. Like illustration, the points 8 × 8of the grid is represented like the Figure 13.



0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

Fig 13: Index by increased order of column of the grid8 × 8

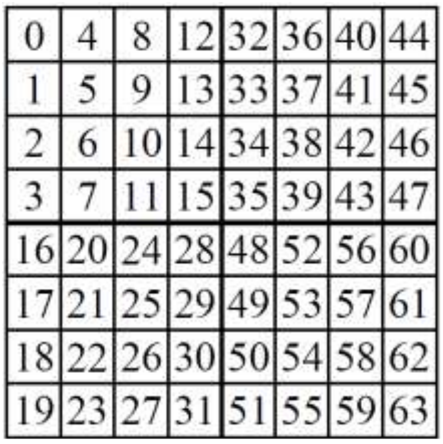
The construction of $2^n \times 2^n$curve filling the space could be subdivided in 4 steps:

- Reallocation bloc
- Permutation Gray
- Rotation andreflection
- Recursive extension

Step 1: Reallocation

Reallocate the initial column by increased order of the bloc of points $2^{n-1} \times 2^{n-1}$.

Their blocs and points are placed by increased order of the bloc. After the reallocation, the grid 8 × 8are represented like the Figure 14.



0	4	8	12	32	36	40	44
1	5	9	13	33	37	41	45
2	6	10	14	34	38	42	46
3	7	11	15	35	39	43	47
16	20	24	28	48	52	56	60
17	21	25	29	49	53	57	61
18	22	26	30	50	54	58	62
19	23	27	31	51	55	59	63

Fig 14: Reallocation of points for the grid8 × 8

Step 2: Permutation Gray

Permute each bloc to 2 × 2 « Gray permutation ». The index order (0, 1, 2,3) are transformed to (0, 1, 3, 2). The result of permutation 2 × 2are represented by the following Figure 15.

0	4	8	12	48	52	56	60
1	5	9	13	49	53	57	61
2	6	10	14	50	54	58	62
3	7	11	15	51	55	59	63
16	20	24	28	32	36	40	44
17	21	25	29	33	37	41	45
18	22	26	30	34	38	42	46
19	23	27	31	35	39	43	47

Fig 15: Gray Permutation for the grid8×8

Step3:Rotation and Reflection

Each bloc, rotation and reflection to followed orientation are applied. By the Figure 16 of Hilbert path only the part left-up and right-up change. The part left-down and right-down has the same orientation ofH_2. The part left-up uses the rotation 90° following the bloc center and reflection by mirror compared to the horizontal of the center bloc. Besides, the part right-up uses the same transformation. Mathematically, the rotation and reflection is obtained by using respectively the transposition and anti-diagonal transposition .The Figure 16 represented the rotation and reflection.

0	1	2	3	63	62	61	60
4	5	6	7	59	58	57	56
8	9	10	11	55	54	53	52
12	13	14	15	51	50	49	48
16	20	24	28	32	36	40	44
17	21	25	29	33	37	41	45
18	22	26	30	34	38	42	46
19	23	27	31	35	39	43	47

Fig 16: Rotation andReflectionfor grid8×8

Step4: Recursive extension

Finally, applying the 4×4 Hilbert path on space filling curve of each 4 blocs. The permutation also is done to the points of blocs. The Figure 17 represent the final version of Hilbert transform

0	3	4	5	58	59	60	63
1	2	7	6	57	56	61	62
14	13	8	9	54	55	50	49
15	12	11	10	53	52	51	48
16	17	30	31	32	33	46	47
19	18	29	28	35	34	45	44
20	23	24	27	36	39	40	43
21	22	25	26	37	38	41	42

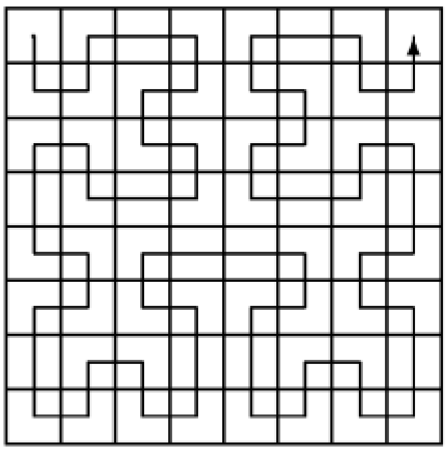


Fig 17: Hilbert Path for grid8×8

We explained the recursive formulation way for space filling curve on each step. Initially, the points are 2D grid of$2^n \times 2^n$following coloumn by the increase order. The index point $(i,j), 0 \le i,j < 2^n$is describe by the tensor base $e_i^{2^n} \otimes e_j^{2^n}$

1. The reallocation bloc could be translated by $B_n = I_2 \otimes L_2^{2^n} I_{2^{n-1}}$. By applying the formula of reallocation bloc on the coloumnbase in the increased order, we obtained also the base:

$$B_n' = e_{i_0}^2 \otimes e_{j_0}^2 \otimes e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}} \quad (23)$$

With, $i = i_0 2^{n-1} + i_1$ and$j = j_0 2^{n-1} + j_1$

Demonstration

♣ To apply initial base $e_i^{2^n} \otimes e_j^{2^n}$ onB_n, B_n' is denifed by :

$$B_n' = B_n\left(e_i^{2^n} \otimes e_j^{2^n}\right) = I_2 \otimes L_2^{2^n} I_{2^{n-1}}\left(e_i^{2^n} \otimes e_j^{2^n}\right)$$

$$B_n' = I_2 \otimes L_2^{2^n} I_{2^{n-1}}\left(e_{i_0}^2 \otimes e_{i_1}^{2^{n-1}} \otimes e_{j_0}^2 \otimes e_{j_1}^{2^{n-1}}\right) = e_{i_0}^2 \otimes e_{j_0}^2 \otimes e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}}$$

Avec, $i = i_0 2^{n-1} + i_1$ and$j = j_0 2^{n-1} + j_1$ ♦

2. « Gray permutation » for2×2 transformed (0, 1, 2,3) to (0,1,3,2). Introduced for beginning,

$$G_2 = I_2 \oplus J_2$$

$$J_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{and } G_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The « Gray permutation » is defined by $G_n = G_2 \otimes I_{2^{2(n-1)}}$. Applying$G_n$on the result of base$B_n'$, it gives so,

$$G_n' = e_{i_0'}^2 \otimes e_{j_0'}^2 \otimes e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}} \quad (24)$$

With(i_0, j_0)will transform to(i_0', j_0')following$(0,0) \to (0,0)$; $(0,1) \to (0,1)$; $(1,0) \to (1,1)$ and$(1,1) \to (1,0)$

Demonstration

♣ $G_n' = \left(G_2 \otimes I_{2^{2(n-1)}}\right) B_n' = \left(G_2 \otimes I_{2^{2(n-1)}}\right)\left(e_{i_0}^2 \otimes e_{j_0}^2 \otimes e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}}\right)$

$$G_n' = e_{i_0'}^2 \otimes e_{j_0'}^2 \otimes e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}}$$

With (i_0, j_0)will transform to(i_0', j_0')following $(0,0) \to (0,0)$; $(0,1) \to (0,1)$; $(1,0) \to (1,1)$ and$(1,1) \to (1,0)$

♦

3. Using reflection and rotation: $T_{n-1}$and$\overline{T}_{n-1}$, are the transposition and the transposition anti-diagonal of$2^{n-1} \times 2^{n-1}$. The rotation and reflection are defined by :

$$R_n = T_{n-1} \oplus I_{2^{2(n-1)}} \oplus I_{2^{2(n-1)}} \oplus \overline{T}_{n-1} \quad (25)$$

The effect of T_{n-1} and $\overline{T}_{n-1}$ on the base $e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}}$is like :

$$T_{n-1}\left(e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}}\right) = e_{j_1}^{2^{n-1}} \otimes e_{i_1}^{2^{n-1}} \quad (26)$$

$$\overline{T}_{n-1}\left(e_{i_1}^{2^{n-1}} \otimes e_{j_1}^{2^{n-1}}\right) = e_{2^{n-1}-1-j_1}^{2^{n-1}} \otimes e_{2^{n-1}-1-i_1}^{2^{n-1}} \quad (27)$$

Noted also that$T_{n-1}$is equivalent of the « stride permutation » by$L_{2^{n-1}}^{2^{2(n-1)}}$

4. Finally, applying the recurisivity of the space filling curve on the 4 blocs, the results is like the expression $I_4 \otimes H_{n-1}$.

2.5.9. RHT

By using all steps reallocation, permutation to gray code, rotation – reflection and recurrence to the other bloc, the space filling curve of Hilbert Transform could be expressed by:

$$H_1 = G_2$$

$$n > 1; H_n = (I_4 \otimes H_{n-1}) R_n G_n B_n$$

$$n > 1; H_n = (I_4 \otimes H_{n-1})\left(T_{n-1} \oplus I_{2^{2(n-1)}} \oplus I_{2^{2(n-1)}} \oplus \overline{T}_{n-1}\right)\left(G_2 \otimes I_{2^{2(n-1)}}\right)\left(I_2 \otimes L_2^{2^n} I_{2^{n-1}}\right) \quad (28)$$

If X is a vector4^ncontained the indices noted by the column by the increase order. Applying the Hilbertpath ,RHT will be defined by :

$$Y = H_n X \quad (29)$$

2.5.10. THT

Tensor Hilbert transform using versioniterative could be expressed by

$$H_1 = G_2 \text{and for} n \ge 1$$

$$H_n = \prod_{i=0}^{n-1} I_{4^i} \otimes \left[T_{n-i-1} \oplus I_{2^{2(n-i-1)}} \oplus I_{2^{2(n-i-1)}} \oplus \overline{T}_{n-i-1}\right]\left(G_2 \otimes I_{2^{2(n-i-1)}}\right)\left(I_2 \otimes L_2^{2^{n-i}} \otimes I_{2^{n-i-1}}\right)$$

With$T_{i-1}$the transposition operationwhich is equivalent to $L_{2^{i-1}}^{2(i-1)}$ and$\overline{T}_{i-1}$ the anti-diagonal transposition of the matrix$2^{i-1} \times 2^{i-1}$

Demonstration

♣ We demonstrate it by recurrence begging to the Hypothesis ofrecurrence :

$$H_1 = \prod_{i=0}^{0} I_{4^i} \otimes \left[T_{1-i-1} \oplus I_{2^{2(1-i-1)}} \oplus I_{2^{2(1-i-1)}} \oplus \overline{T}_{1-i-1}\right]\left(G_2 \otimes I_{2^{2(1-i-1)}}\right)\left(I_2 \otimes L_2^{2^{1-i}} \otimes I_{2^{1-i-1}}\right)$$

$$H_1 = I_{4^0} \otimes [T_0 \oplus I_{2^0} \oplus I_{2^0} \oplus \overline{T}_0](G_2 \otimes I_{2^0})\left(I_2 \otimes L_2^{2^1} \otimes I_{2^0}\right) = G_2$$

Supposed now,

$$H_k = \prod_{i=0}^{k-1} I_{4^i} \otimes \left[T_{k-i-1} \oplus I_{2^{2(k-i-1)}} \oplus I_{2^{2(k-i-1)}} \oplus \overline{T}_{k-i-1}\right]\left(G_2 \otimes I_{2^{2(k-i-1)}}\right)\left(I_2 \otimes L_2^{2^{k-i}} \otimes I_{2^{k-i-1}}\right)$$

Demonstrate,

$$H_{k+1} = \prod_{i=0}^{k} I_{4^i} \otimes \left[T_{k-i-1} \oplus I_{2^{2(k-i-1)}} \oplus I_{2^{2(k-i-1)}} \oplus \overline{T}_{k-i-1}\right]\left(G_2 \otimes I_{2^{2(k-i-1)}}\right)\left(I_2 \otimes L_2^{2^{k-i}} \otimes I_{2^{k-i-1}}\right)$$

By definition,

$$H_{k+1} = \left(I_4 \otimes H_{(k+1)-1}\right)\left(T_{(k+1)-1} \oplus I_{2^{2((k+1)-1)}} \oplus I_{2^{2((k+1)-1)}} \oplus \overline{T}_{(k+1)-1}\right)\left(G_2 \otimes I_{2^{2((k+1)-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)}} I_{2^{(k+1)-1}}\right)$$

$$H_{k+1} = \left(I_4 \otimes \prod_{i=0}^{k-1} I_{4^i} \otimes \left[T_{k-i-1} \oplus I_{2^{2(k-i-1)}} \oplus I_{2^{2(k-i-1)}} \oplus \overline{T}_{k-i-1}\right]\left(G_2 \otimes I_{2^{2(k-i-1)}}\right)\left(I_2 \otimes L_2^{2^{k-i}} \otimes I_{2^{k-i-1}}\right)\right)$$

$$\left(T_{(k+1)-1} \oplus I_{2^{2((k+1)-1)}} \oplus I_{2^{2((k+1)-1)}} \oplus \overline{T}_{(k+1)-1}\right)\left(G_2 \otimes I_{2^{2((k+1)-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)}} I_{2^{(k+1)-1}}\right)$$

$$H_{k+1} = \left(\prod_{i=0}^{k-1} I_{4^{i+1}} \otimes \left[T_{k-i-1} \oplus I_{2^{2(k-i-1)}} \oplus I_{2^{2(k-i-1)}} \oplus \overline{T}_{k-i-1}\right]\left(G_2 \otimes I_{2^{2(k-i-1)}}\right)\left(I_2 \otimes L_2^{2^{k-i}} \otimes I_{2^{k-i-1}}\right)\right)$$

$$\left(T_{(k+1)-1} \oplus I_{2^{2((k+1)-1)}} \oplus I_{2^{2((k+1)-1)}} \oplus \overline{T}_{(k+1)-1}\right)\left(G_2 \otimes I_{2^{2((k+1)-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)}} I_{2^{(k+1)-1}}\right)$$

$$H_{k+1} = \left(\prod_{i=1}^{(k+1)-1} I_{4^i} \otimes \left[T_{(k+1)-i-1} \oplus I_{2^{2((k+1)-i-1)}} \oplus I_{2^{2((k+1)-i-1)}} \oplus \overline{T}_{(k+1)-i-1}\right]\right)$$

$$\left(G_2 \otimes I_{2^{2((k+1)-i-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)-i}} \otimes I_{2^{(k+1)-i-1}}\right)\left(T_{(k+1)-1} \oplus I_{2^{2((k+1)-1)}} \oplus I_{2^{2((k+1)-1)}} \oplus \overline{T}_{(k+1)-1}\right)$$

$$\left(G_2 \otimes I_{2^{2((k+1)-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)}} I_{2^{(k+1)-1}}\right)$$

Like$H_0$is equivalent to :

$$\left(T_{((k+1)+1)-1} \oplus I_{2^{2(((k+1)+1)-1)}} \oplus I_{2^{2(((k+1)+1)-1)}} \oplus \overline{T}_{((k+1)+1)-1}\right)\left(G_2 \otimes I_{2^{2((k+1)-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)}} I_{2^{(k+1)-1}}\right)$$

$$H_{k+1} = \left(\prod_{i=0}^{(k+1)-1} I_{4^i} \otimes \left[T_{(k+1)-i-1} \oplus I_{2^{2^{2((k+1)-1)}}} \oplus I_{2^{2^{2((k+1)-1)}}} \oplus \overline{T}_{(k+1)-i-1}\right]\right)$$

$$\left(G_2 \otimes I_{2^{2((k+1)-i-1)}}\right)\left(I_2 \otimes L_2^{2^{(k+1)-i}} \otimes I_{2^{(k+1)-i-1}}\right)$$

$$H_{k+1} = \prod_{i=0}^{k} I_{4^i} \otimes \left[T_{k-i-1} \oplus I_{2^{2(k-i-1)}} \oplus I_{2^{2(k-i-1)}} \oplus \overline{T}_{k-i-1}\right]\left(G_2 \otimes I_{2^{2(k-i-1)}}\right)\left(I_2 \otimes L_2^{2^{k-i}} \otimes I_{2^{k-i-1}}\right) \blacklozenge$$

2.5.11 inverse THT

The inverse transformation could by expressed by :

$$H_n^{-1} = \prod_{i=0}^{n-1} I_{4^{n-1-1}} \otimes \left[\left(I_2 \otimes L_{2^i}^{2^{i+1}} \otimes I_{2^i}\right)\left(G_2 \otimes I_{2^i}\right)\left(T_i \otimes I_{2^i} \otimes I_{2^i} \otimes \overline{T}_i\right)\right]$$

Demonstration

♣ Byreccurence,

$$H_n^{-1} = \prod_{i=0}^{n-1} I_{4^{n-1-1}} \otimes \left[\left(I_2 \otimes L_{2^i}^{2^{i+1}} \otimes I_{2^i}\right)\left(G_2 \otimes I_{2^i}\right)\left(T_i \otimes I_{2^i} \otimes I_{2^i} \otimes \overline{T}_i\right)\right]$$

and,

$$H_n^{-1} = \left[\prod_{i=0}^{n-1} I_{4^i} \otimes\left[\left(T_{n-i-1} \oplus I_{2^{2(n-i-1)}} \oplus I_{2^{2(n-i-1)}} \oplus \bar{T}_{n-i-1}\right)\left(G_2 \otimes I_{2^{2(n-i-1)}}\right)\left(I_2 \otimes L_2^{2^{n-1}} \otimes I_{2^{2(n-i-1)}}\right)\right]\right]^{-1}$$

$$H_n^{-1} = \prod_{i=n-1}^{0} I_{4^i} \otimes\left[\left(I_2 \otimes L_2^{2^{n-1}} \otimes I_{2^{2(n-i-1)}}\right)^{-1}\left(G_2 \otimes I_{2^{2(n-i-1)}}\right)^{-1}\left(T_{n-i-1} \oplus I_{2^{2(n-i-1)}} \oplus I_{2^{2(n-i-1)}} \oplus \bar{T}_{n-i-1}\right)^{-1}\right]$$

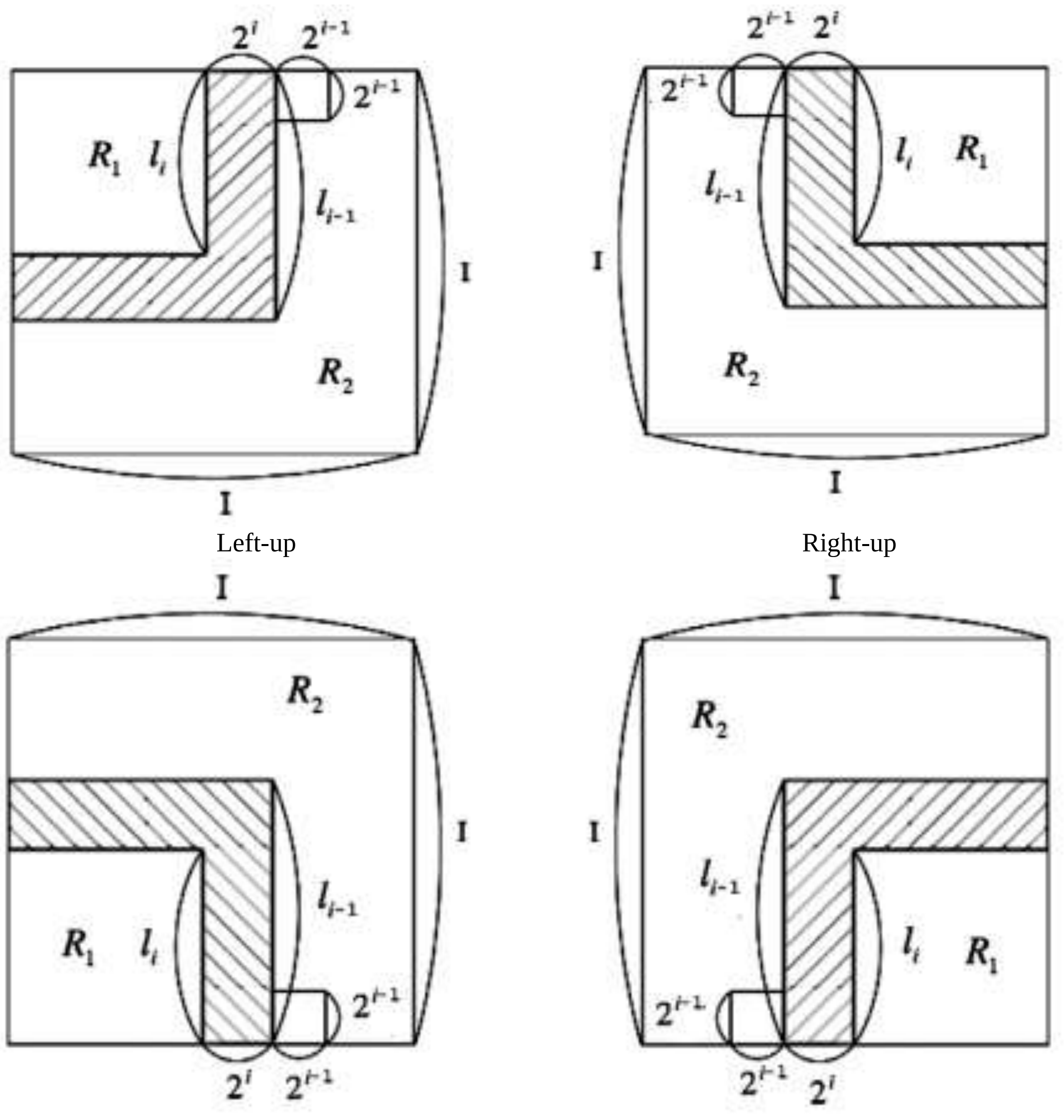
$$H_n^{-1} = \prod_{i=n-1}^{0} I_{4^i} \otimes\left[\left(I_2 \otimes \left(L_2^{2^{n-1}}\right)^{-1} \otimes I_{2^{2(n-i-1)}}\right)\left(G_2{}^{-1} \otimes I_{2^{2(n-i-1)}}\right)\left(T_{n-i-1}^{-1} \oplus I_{2^{2(n-i-1)}} \oplus I_{2^{2(n-i-1)}} \oplus \bar{T}_{n-i-1}^{-1}\right)\right]$$

$$H_n^{-1} = \prod_{i=n-1}^{0} I_{4^i} \otimes\left[\left(I_2 \otimes L_{2^{n-i-1}}^{2^{n-1}} \otimes I_{2^{2(n-i-1)}}\right)\left(G_2 \otimes I_{2^{2(n-i-1)}}\right)\left(T_{n-i-1} \oplus I_{2^{2(n-i-1)}} \oplus I_{2^{2(n-i-1)}} \oplus \bar{T}_{n-i-1}\right)\right]$$

$$H_n^{-1} = \prod_{i=0}^{n-1} I_{4^{n-1-1}} \otimes\left[\left(I_2 \otimes L_{2^i}^{2^{i+1}} \otimes I_{2^i}\right)\left(G_2 \otimes I_{2^i}\right)\left(T_i \otimes I_{2^i} \otimes I_{2^i} \otimes \bar{T}_i\right)\right]$$

♦

2.6. Efficient Decomposition of Matrix:



Left-down Right-down

Fig 18: Left-up ; right-up ; left-down ; right-down optimum area decomposition

Like the Hilbert transform is only applied on matrix$2^n \times 2^n$, with any image having different size, any decomposition following the optimum area could be possible [4] in case of : left-up, right-up, left-down, right-down optimum decomposition area [4]. So in the Figure18 :

- R1 is the matrix $2^{l_i} \times 2^{l_i}$ still studied at i-éme iteration
- Hachured area: is the area in the treatment,, to be decomposed to the area $2^i \times 2^i$ with i the maximum that possible
- R2 is the rest area to be treated at the future for each iteration

3. Results and Analysis

Note that all results are sampled so to improve the curve, the implementation of pchip(Piecewise Cubic Hermite Interpolating Polynomial) still implanted at Matlabimproves the plot.

3.1. left-up optimum area decomposition

The time comparison using left-up decomposition is presented by the Figure 19.

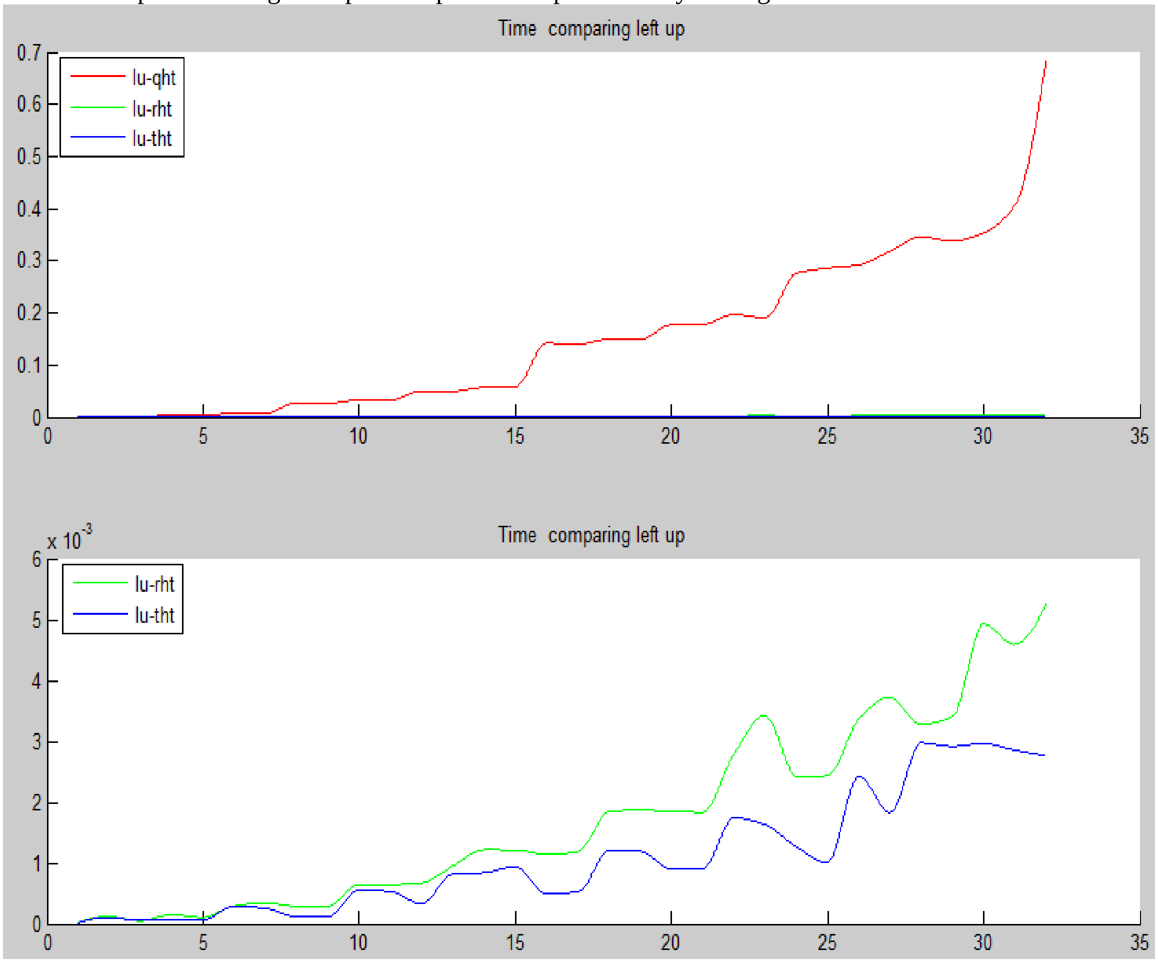


Fig 19: Comparing lu-qht, lu-rht, lu-tht

The curve obtained is like cubic, the reason of this is the same that using left-up decomposition. The size of matrix changes also to 1×1 until 32×32. The QHT time change to 0 until 0.7sec and the curve in the top shows us that implementation of the QHT on the classic computer is not efficient. To make distinction of the 2 others curves RHT and THT with time evaluation at 0 to 6×10^{-3}. The nature of the curve also is cubic with the same reason that the image with size $2^n \times 2^n$ don't spent more time than the other size. The RHT is more complicate time evaluation that THT it's due to the recursive version should allocate all the calling procedure of all sub-function. By order, QHT then, RHT, and then THT are proposed using left-up maximum area and the THT has the best time.

3.2. right-up optimum area decomposition

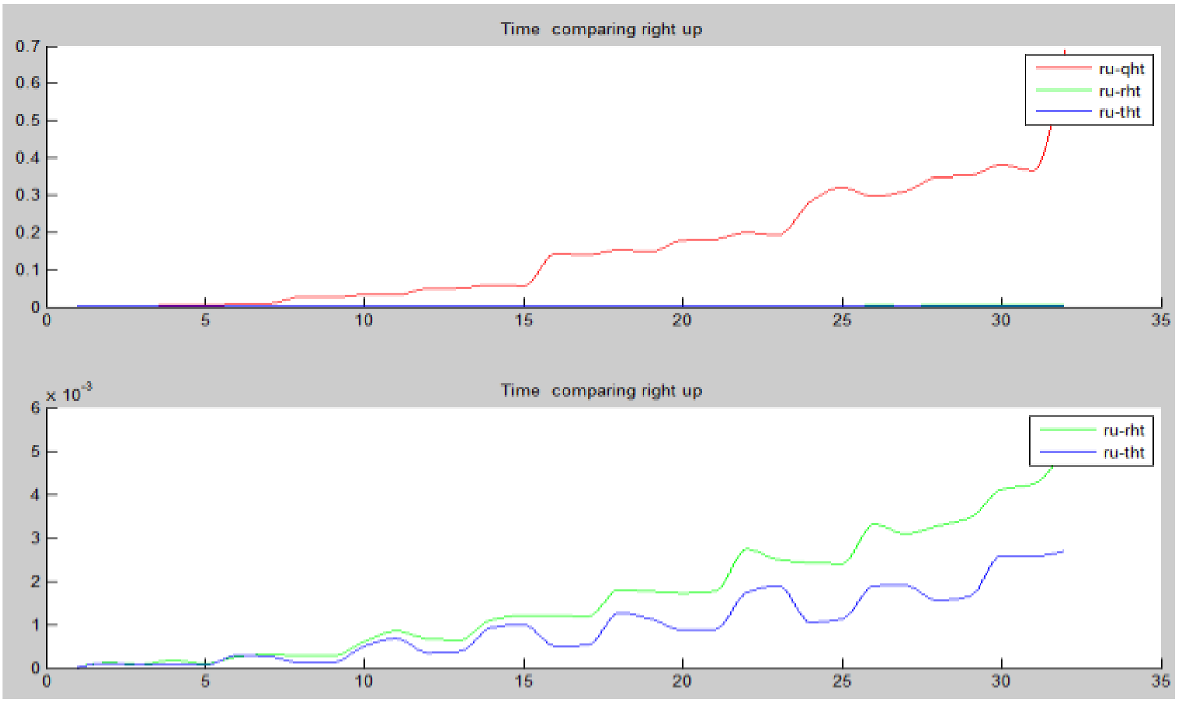


Fig 20: Comparing ru-qht, ru-rht, and ru-tht

The interpretation is more identic than: left-up interpretation. The curve obtained is like cubic it's due to the decomposition is only one decomposition if the matrix have the sized $2^n \times 2^n$ and the decomposition have spend more time when the sized is different. The size of matrix changes to 1×1 until 32×32.QHT is also the most complex to 0 until 0.7sec like the curve on top, the two curve ru-tht and ru-rht are near to zero second . The RHT is more complicate time evaluation that THT it's due to the recursive version should allocate all the calling procedure of all sub-function. QHT then, RHT, and then THT are classified by order using right-up maximum area and the THT has the best time. The time is also more identic of left-up decomposition due to the number of decomposition of the same matrix is the same.

3.3. left-down optimum area decomposition

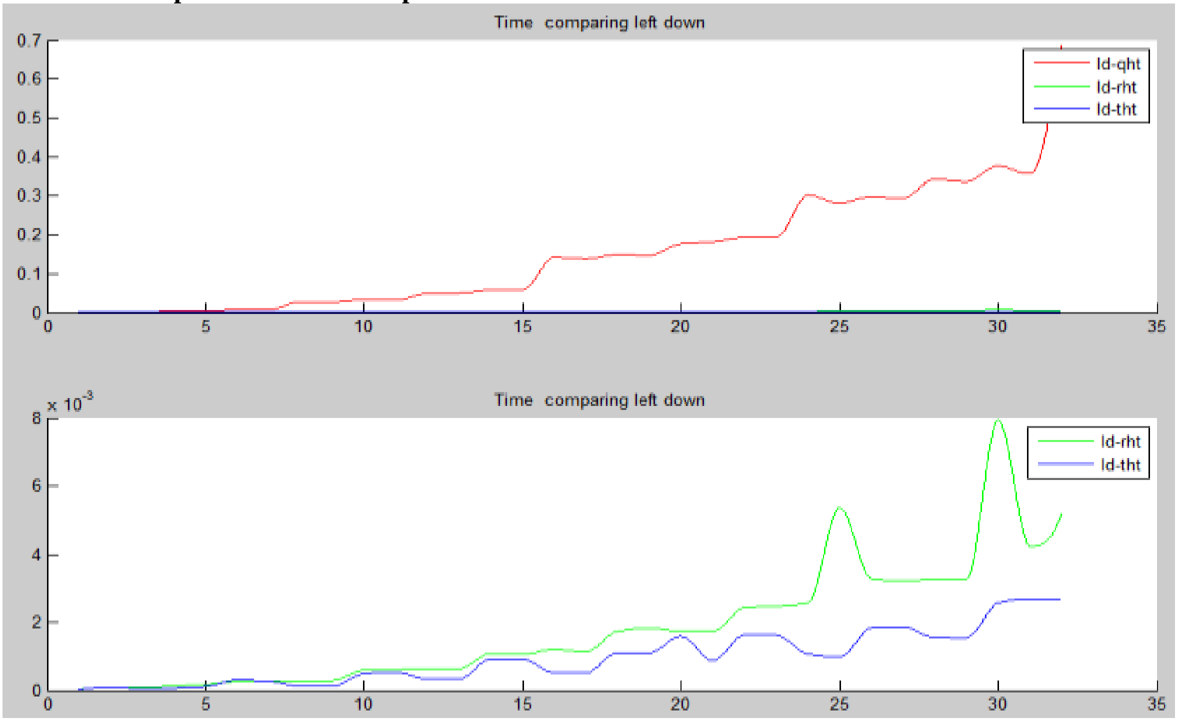


Fig 21:Comparing ld-qht, ld-rht, ld-tht

The curve obtained is like cubic, the reason of this is the same that using left-down decomposition. The size of matrix changes also to 1×1 until 32×32. The QHT time change to 0 until 0.7sec and the curve in the top shows us that implementation of the QHT on the classic computer is not efficient. To make distinction of the 2 others curves RHT and THT with time evaluation at 0 to 8×10^{-3}. The nature of the curve also is cubic with the same reason that the image with size $2^n \times 2^n$ don't spent more time than the other size. The RHT is more complicate time evaluation that THT it's due to the recursive version should allocate all the calling procedure of all sub-function. QHT then, RHT, and then THT are classified by this order using left-down maximum area and the THT has the best time.

3.4. right-down optimum area decomposition

The interpretation is more identic than: left-down interpretation. The curve obtained is like cubic it's due to the decomposition is only one decomposition if the matrix have the sized $2^n \times 2^n$ and the decomposition have spend more time when the sized is different. The size of matrix changes to 1×1 until 32×32.QHT is also the most complex to 0 until 0.7sec like the curve on top, the two curve ru-tht and ru-rht are near to zeo second . The RHT is more complicate time evaluation that THT it's due to the recursive version should allocate all the calling procedure of all sub-function. QHT then, RHT, and then THT are classified by order using right-down maximum area and the THT has the best time. The time is also more efficient for the other on the last simulation due to the cache procedure found at the 3 others methods still analyzed.

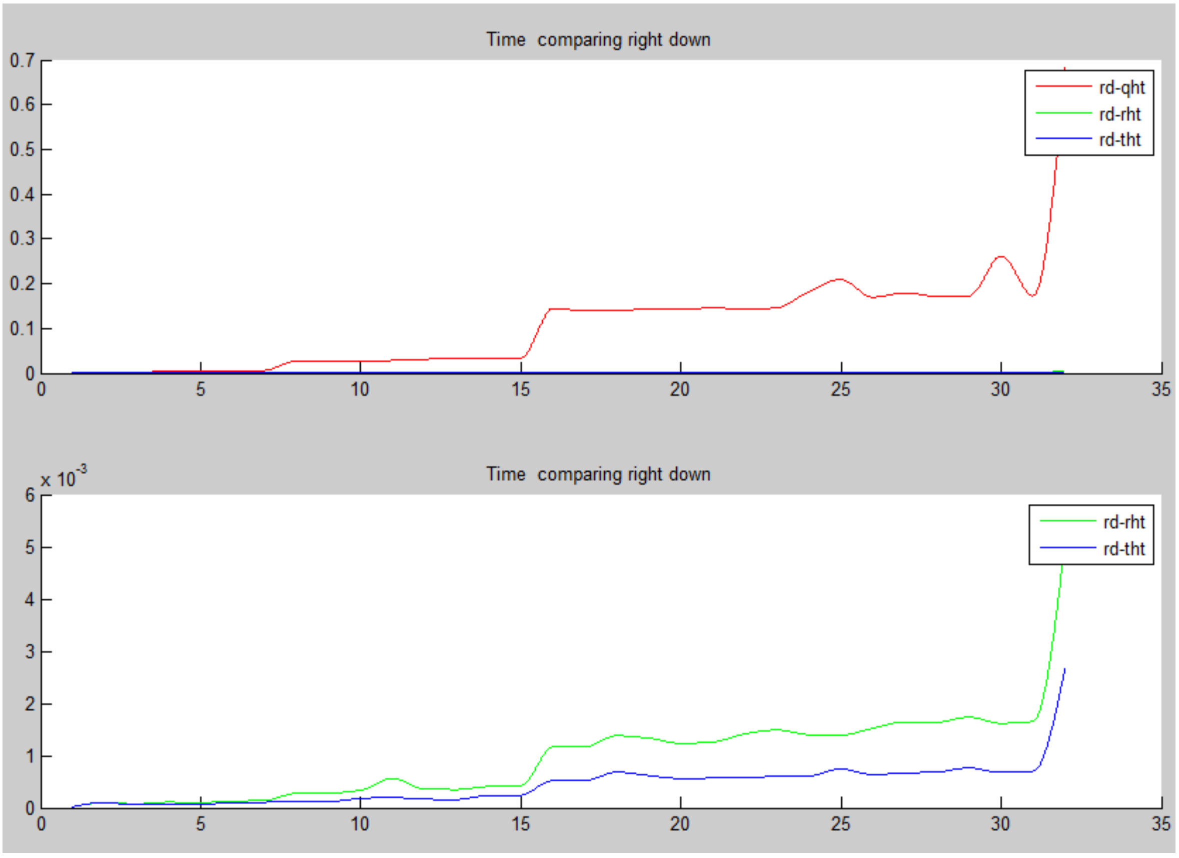


Fig 22: Comparing rd-qht, rd-rht, rd-tht

4. Conclusion

There are many methods to implements the Hilbert Transform: using the quantum version (QHT), using recursive version (RHT) and using tensor version (THT). For any square matrix, the left-up, left-down, right-up and right-down decomposition using optimum are needed. By time complexity could be arranged by order QHT then RHT then THT even for the 4 area decompositions. On the classic computing, using quantum algorithm is not efficiency. The choice should be the THT.The number of matrix obtained is the same on the 4 decomposition, the time for each is quite identic. In the future, it's very interesting to study the performance of the other scrambling technic using SFC and evaluating the performance of the quantum computing and classic computing [13] [14] [15].

References

[1] Zhengwen P. and Xin L.,"Amplification Matrix Iteration Algorithm to Generate the Hilbert-Peano Curve", *Transactions on Computer Science and Technology,* vol.3, June 2014

[2] Ri-Gui Z., Qian W., Man-Qun Z. and Chen-Yi S., "A Quantum Image Encryption Algorithm Based on Quantum Image Geometric Transformations", *Article in International Journal of Theoretical Physics,* June 2012

[3] Shen-Yi L., Chih-Shen C., Li L. and Chua-Huang H., "Tensor Product Formulation for Hilbert Space-Filling Curves", *National Science Council, Taiwan, R.O.C. under grant NSC 91-2213-E-035-015*, 2015

[4] Kuo-Liang C. A., Yi-Luen H. A andYau-Wen L., "Efficient algorithms for coding Hilbert curve of arbitrary-sized image and application to window query", *Information Sciences 177 2130–2151*, 2007

[5] https://github.com/Alcinoos/QHT_RHT_THT,2019

[6]Ri-Gui Z., Ya-Juan S. and Ping F., "Quantum image Gray-code and bit-plane scrambling, in Quantum Information Processing", *Shanghai Maritime Universit*y, May 2015

[7] Nan R. Z., Tian X. H., Li H. G., Dong J. P. and Qing H. L., "Quantum image encryption based on Generalized Arnold Transform and double random-phase encoding" *in Quantum Inf Process Nanchang University; Shanghai Jiao Tong University,* 28 January 2015

[8] Nan J., Luo W. and Wen-Y. W., "Quantum Hilbert Image Scrambling Published", *Springer Science Business and Media New York*, 30 April 2014
College of Computer, Beijing University of Technology, Beijing

[9] Rifaat Z. K. and Alharith A. A., "Novel Quantum Encryption Algorithm Based on Multiqubit
Quantum Shift Register and Hill Cipher", *Hindawi Publishing Corporation Article ID 104325*, 2014

[10] Shen-Yi L., Chih-Shen Ch., Li L., and Chua-Huang H."A closed-form algorithm for converting Hilbert Space filling curves indices", in *IAENG International Journal of computer Science, IJCS_31_01_02,* February 2010

[11]Xuefeng G., Peter V. O. and Bo C., "A Parallel N-Dimensional Space-Filling Curve Library and Its Application in Massive Point Cloud Management", *International Journal of Geo-Information*, 15 August 2018

[12]Ruisong Y. and Li L.,"An Iterated Function System based Method to Generate Hilbert-type Space-filling Curves", *I.J. Information Technology and Computer Science*, 2015

[13]Haverkort H.J., "An inventory of three-dimensional Hilbert space-filling curves*", Eindhoven University Technology*, 01 January 2011

[14]Chih-Sheng, Shen-Yi L., Min-Husuan and Chua-Huang, "A novel construction method for n-dimension Hilbert Space-Filling Curves", *IEICE Trans.Infvol E93 n0.7,* July 2010

[15]Chih-Sheng Chen, Shen-Yi Lin, and Chua-Huang Huang, "Algebraic Formulation and Program Generation of Three-Dimensional Hilbert Space-Filling Curves" *by National Science Council, Taiwan, R.O.C.*