

Mapping high dimensional features onto Hilbert curve: applying to fast image retrieval

Giap Nguyen, Patrick Franco, Remy Mullot, Jean-Marc Ogier
L3i Laboratory, University of La Rochelle, France
giap.nguyen@univ-lr.fr

Abstract

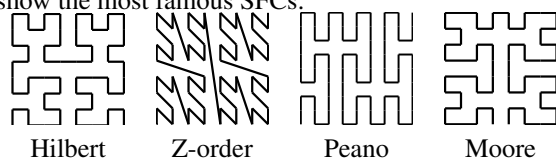
The use of high-dimension features is unavoidable in many applications of image retrieval and techniques of dimension reductions are not always efficient. The space-filling curve reduces the number of dimensions to one while preserving the neighborhood relation. In this paper, Hilbert curve, the most neighborhood preserving space-filling curve, is used in shape retrieval. The retrieving is accelerated by ordering the shapes in 1-D dynamical data structure, which enables rapid insertions of new images without changing existing data. A proposal of fast mapping facilitate the computing of 1-D Hilbert indexes from high dimensional features.

1. Introduction

In content-based image retrieval, the image is usually described by high-dimensional features, such as, histograms, SIFT features and whose dimensionality could be very big, 128 for SIFT and attain thousands for histograms, and this may cause the time-consuming when we index or search images in a large image collection.

To overcome this problem, many researchs focus on the dimension reduction, such as feature selection, PCA, clustering or neural networks. However, these techniques themselves are time-consuming. Next, reduced features could be indexed with a multidimensional indexing structure, such as, k-d tree, R-tree.

In this paper, the space-filling curve (SFC) [8] is used to map high-dimensional features with one-dimensional (1-D) indexes while preserving the locality of features. A SFC is a way to visit every point of a square, a cube or more generally, a n -D hypercube. The following figures show the most famous SFCs.



The important property of the SFC is locality-preserving: two close points are ordered nearly on curve. Position on curve a point is called its index (1-D). Thanks to this property, some or all neighbors of an indexed multi-dimensional feature can be rapidly found by taking the features corresponding to previous indexes and next indexes, thus, there is no distance calculation needed to find similar images.

Unlike other dimension reduction techniques, indexes are invariable, the index of one image stay independent from other image indexes, hence, there is no change on existing images when new images are inserted and the time for insertion is linearly increasing.

Acknowledge that different SFCs preserve the locality differently. The Hilbert SFC, recognized to be the most locality-preserving SFC [3, 4, 7], is used in the below experimentation.

For fast mapping, we propose an inverse of the Butz algorithm to map n -D features to 1-D Hilbert curve indexes. It is then applied in a symbol search system that experiments the curve performance.

2. Hilbert curve: a proposal of algorithm for fast high dimensional mapping

There are 2 effective algorithms to do the mapping task. Bially propose a general mapping technique which is applicable for every self-similar SFCs [1]. Being a table-based approach, the drawback of this approach is the processing time and the memory occupation for the table, so it is not a good choice for high-dimensional mapping.

Butz algorithm [2], used only for the Hilbert curve mapping, avoids this problem. There is no table used, the transformations are coded by a series of binary operations. However, Butz proposes only the mapping from indexes (1-D) to their points (n -D) while in general, we need an inverse mapping, which does the dimension reduction.

We propose below an inverse of Butz algorithm to map n -D points to their indexes on curve. This approach works well in the high-dimensional case and inherits the rapidity of Butz algorithm.

Butz algorithm maps an index which has the binary representation $\rho_1^1 \rho_2^1 \dots \rho_n^1 \rho_1^2 \rho_2^2 \dots \rho_n^2 \dots \rho_1^m \rho_2^m \dots \rho_n^m$ to a n -D point (a_1, a_2, \dots, a_n) .

The notation ρ^i is used to stand for $\rho_1^i \rho_2^i \dots \rho_n^i$. The "principal position" of ρ^i is the last position in ρ^i such that $\rho_j^i \neq \rho_n^i$ (if all bits of ρ^i are equal, the principal position is the n th). The remaining entities necessary to define the algorithm are given in Table 1.

Table 1. Entites used in Butz algorithm.

J_i : An integer between 1 and n equal to the subscript of the principal position of ρ^i . I the following four examples of ρ^i for the case $n = 5$, the values of J_i are 5, 2, 4, and 5, respectively (the principal positions are circled):

1	1	1	1	①
1	①	1	1	1
0	0	1	①	0
0	0	0	0	①

σ^i : A byte of n bits, such that $\sigma_1^i = \rho_1^i, \sigma_2^i = \rho_2^i \oplus \rho_1^i, \sigma_3^i = \rho_3^i \oplus \rho_2^i, \dots, \sigma_n^i = \rho_n^i \oplus \rho_{n-1}^i$, where \oplus stands for EXCLUSIVE-OR operation.

τ^i : A byte of n bits obtained by complementing σ^i in the n th position and then, if and only if the resulting byte is of odd parity, complementing in the principal position. Hence, τ^i is always of even parity. Note that the parity of σ^i is given by the bit ρ_n^i and that a mask for performing the second complementation may be set up in the same process which calculates J_i .

$\tilde{\sigma}^i$: A byte of n bits obtained by shifting σ^i right circular a number of positions equal to

$$(J_1 - 1) + (J_2 - 1) + \dots + (J_{i-1} - 1)$$

There is no shift in σ^1

$\tilde{\tau}^i$: A byte of n bits obtained by shifting τ^i in exactly the same way.

ω^i : A byte of n bits where

$$\omega^i = \omega^{i-1} \oplus \tilde{\tau}^{i-1}, \quad \omega^1 = 00\dots 00$$

and where \oplus indicates the EXCLUSIVE-OR operation on corresponding bits.

α^i : A byte of n bits where $\alpha^i = \omega^i \oplus \tilde{\sigma}^i$.

If α^i has the binary representation $\alpha_1^i \alpha_2^i \dots \alpha_n^i$ then a_j has the binary representation $\alpha_j^1 \alpha_j^2 \dots \alpha_j^m$. It is easy to recognize that (a_j) s are the rows of the matrix which

has columns are (α_i) s.

Our proposal: n -D to 1-D mapping.

Giving a point (a_1, a_2, \dots, a_n) , we can easily get back the (α_i) s because they are columns of the matrix which has rows (a_j) s.

To do the inverse mapping, we use following properties of the binary EXCLUSIVE-OR operation:

- $x \oplus y = y \oplus x$
- if $x \oplus y = z$ then $x \oplus z = y$

We can see these properties from its definition:

\oplus	0	1
0	0	1
1	1	0

From these properties and $\alpha^i = \omega^i \oplus \tilde{\sigma}^i$, we have: $\tilde{\sigma}^i = \alpha^i \oplus \omega^i$. The calculation of ρ_i is given in Table 2.

Table 2. Our proposal: inversion of Butz algorithm.

σ^1	: $\sigma^1 = \alpha^1$ because: $\alpha^1 = \omega^1 \oplus \tilde{\sigma}^1 \Rightarrow \tilde{\sigma}^1 = \alpha^1 \oplus \omega^1$ $\omega^1 = 00\dots 00 \Rightarrow \tilde{\sigma}^1 = \alpha^1$ There is no shift in $\sigma^1 \Rightarrow \sigma^1 = \tilde{\sigma}^1$
ρ^i	: A byte of n bits, such that $\rho_1^i = \sigma_1^i, \rho_2^i = \sigma_2^i \oplus \rho_1^i, \rho_3^i = \sigma_3^i \oplus \rho_2^i, \dots, \rho_n^i = \sigma_n^i \oplus \rho_{n-1}^i$.
J_i	: see Table 1.
τ^i	: see Table 1.
$\tilde{\tau}^i$: see Table 1.
ω^{i+1}	: see Table 1.
$\tilde{\sigma}^{i+1}$: A byte of n bits where $\tilde{\sigma}^{i+1} = \alpha^{i+1} \oplus \omega^{i+1}$.
σ^{i+1}	: A byte of n bits obtained by shifting σ^i left circular a number of positions equal to $(J_1 - 1) + (J_2 - 1) + \dots + (J_i - 1)$

3. Application to shape retrieving in large database

We apply now our mapping method in a shape retrieval test. The standard Zernike moments [6] are used as shape features. B-tree is used to order the shapes by their Hilbert indexes. This famous data structure minimizes the index search time and makes clear the SFC performance.

3.1. Shape retrieving procedure

In the following experimentation, the shape retrieving include two phases:

Database constructing Zernike moments of each shape are computed and mapped with its Hilbert index. Shapes are then incrementally added to a B-tree where increasing order of indexes is maintained.

Shape searching is then resumed by index searching in the before-said B-tree, the request index corresponding to the given input shape. Shapes having closest indexes are issued as result.

Because the HC preserves neighborhoods, shapes ordered successively have much probability to be similar and the shapes having indexes close to input shape index have also much probability to be similar to input shape. Thereto, we can find out shapes very fast since they are kept in a B-tree.

3.2. Experimentation condition

Test images are collected from 3 standard databases: **MPEG-7** Core Experiment CE-Shape-1 (<http://www.imageprocessingplace.com>), **LEMS** 1070-Shape Database (<http://vision.lems.brown.edu>), **GREC**'2011 Symbol Recognition Contest (<http://grec2011.cau.ac.kr>).

The following table and images resumes these shape databases:

Database	MPEG	Shape	GREC
Number of images	1400	1070	12900
Number of categories	70	60	150
Average image size	128x116		588x294



These images are then randomly merged, the new image database contains 15370 images of different natures: solid shapes (MPEG & LEMS), lines and arcs (GREC). They are separated in 3 partitions for 3 purposes: 14300 first images are used in database building test, dynamical insertion test uses 1010 others images containing 800 GREC images. 60 left images are merged with 60 used images, these images are then used in the search test.

In our tests, fast computed Zernike moments [5] until the order 9 are used, hence, the feature space is 30-dimensional. To apply the n -D to 1-D mapping, these features are normalized into [0,1023] and coded by 10-bits.

The indexing and search tests are compared to 2 kinds of traditional multi-dimensional indexing:

- *k-d tree*: the image features are indexed directly by a 30-d tree (30 dimensional k-d tree).
- *dimReduc*: the dimensionality is reduced to 10 by hierachical clustering, then, reduced features are indexed on a 10-d tree. The clustering is repeated after each adding of 143 images (10%).

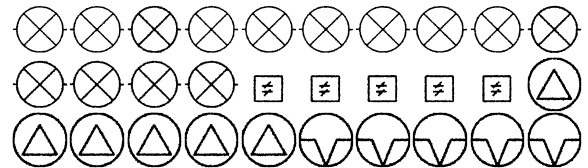
The tests are realized on a standard computer:

Processor	Intel Core 2 Duo T9800 2.93GHz x 2
Memory	3.9 GB
OS	Ubuntu 11.10 64-bit

3.3. Shape database building test

Zernike moments of each image (from 14300 images) are computed and mapped with its Hilbert index. The shape information is then inserted into B-tree.

We can recognize that images contain similar shapes are usually ordered closely. For example, following shapes have order from 10171 to 10202,

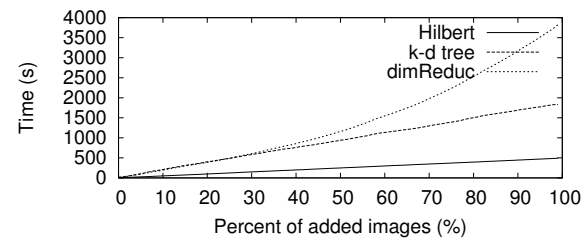


This reflects the relation between the shapes and the indexes: successive indexes usually correspond to similar shapes. This fact eases the search process.

Another way to measure the performance of Hilbert curve is the cluster size. A cluster is a series of similar shapes ordered successively. On 1000 ordered images from random position, the average cluster size is 3.62 (there are 276 clusters). Also, the following table shows size of 10 randomly taken cluster.

10	7	4	1	6	1	8	17	3	2

The database building time, including time for Zernike moments computing, mapping them with Hilbert indexes and insertion into the B-tree, is 489.31 seconds. The cumulative adding time (via percentage of images added) is shown in the following graph:




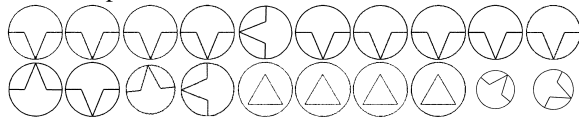
We can see a regular linear increase of added time, there is no abnormal change.

The databases are dynamics, we can add new images without modifying existing entities. The time for

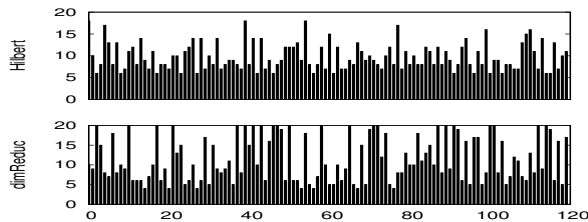
adding 1010 new images into database is only 34.5 seconds. The average adding time for each image is very small (0.0345s). Therefore, the database updating is simple.

3.4. Shape searching

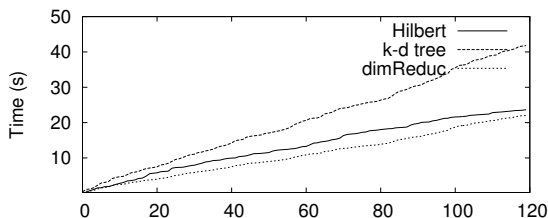
For each image in 120 test images, the Zernike moments are mapped with the corresponding index, the search of this index will issue 20 closest shape images. For example, the search results for  are



Besides many good results, we can recognize some results which haven't the same shape as the input. The inaccuracy can be explained by the lack of similar images but it is also from the neighbor missing of the curve. The original space, which of Zernike feature, is multi-dimensional but the points on Hilbert curve are only one dimensional. The map can be considered as a dimensional reduction which causes the neighbor missing. In our test, the precision average is 9.77/20 (the precision average of dimReduc is 11.5/20, the search on k-d tree is completely precise). In more details, the following graph shows the accuracy over 120 test searches.



In return for the neighbor missing, the shapes can be increasingly ordered by Hilbert indexes, therefore, the search is much accelerated. In our test, the search time is only $O(\log_N(n))$ where n is database size and N is capacity of B-tree node. In our test, the total search time for 120 search tests is 23.65 seconds and the average search time is 0.197. The following graph shows search time details:



4. Conclusion

In images processing applications, multi-dimensional features are frequently used and spatial relation between these features decides the data

essence. SFC fits well with these applications by reducing the feature dimension to 1 and preserving the spatial relation of features. The use of SFC in multi-dimensional feature applications eliminates the distance calculation time and reduces the search time once features are linearly ordered. Besides, organization of images is dynamical, images are independent, there is no computation on existing images when new images are inserted while in usual image retrieval system, a new training process is needed.

The new fast and light on memory usage n-D to 1-D mapping can be efficiently use in applications which need of multi-dimensional feature distance calculation, including application which limited by low-dimension SFCs. The integration of this mechanism in a high accurate system, like CBIRs, could be interesting.

Our test application shows the good performance of Hilbert SFC in shape ordering and search time. Of course, in a preliminary test, the search results are far from a high accuracy image search system. This simple application suggest a complete automatic image organization - instant search system, which improves the found results, for example, by combining and refining the results. Application of SFC is not limited to global image features, we can apply SFCs for local features.

References

- [1] T. Bially. Space-filling curves: Their generation and their application to bandwidth reduction. *Information Theory, IEEE Transactions on*, 15(6):658 – 664, Nov. 1969.
- [2] A. Butz. Alternative algorithm for hilbert's space-filling curve. *IEEE Transactions on Computers*, 20(4):424–426, 1971.
- [3] C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In *Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '89, pages 247–252, New York, NY, USA, 1989. ACM.
- [4] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *Image Processing, IEEE Transactions on*, 5(5):794–797, 1996.
- [5] S. Hwang and W. Kim. A novel approach to the fast computation of zernike moments. *Pattern Recognition*, 39(11):2065–2076, 2006.
- [6] A. Khotanad and Y. Hong. Invariant image recognition by zernike moments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(5):489–497, 1990.
- [7] B. Moon, H. Jagadish, C. Faloutsos, and J. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *Knowledge and Data Engineering, IEEE Transactions on*, 13(1):124 –141, jan/feb 2001.
- [8] H. Sagan. *Space-filling curves*, volume 2. Springer-Verlag New York, 5004 Glen Forest Drive Raleigh, NC 27612 USA, 1994.