Algorithms for Scientific Computing

Space-Filling Curves in 2D and 3D

Michael Bader
Technical University of Munich
Summer 2016
Classification of Space-filling Curves

Definition: (recursive space-filling curve)

A space-filling curve $f : I \rightarrow Q \subset \mathbb{R}^n$ is called recursive, if both $I$ and $Q$ can be divided in $m$ subintervals and sudomains, such that

- $f^*(I^{(\mu)}) = Q^{(\mu)}$ for all $\mu = 1, \ldots, m$, and
- all $Q^{(\mu)}$ are geometrically similar to $Q$.

Definition: (connected space-filling curve)

A recursive space-filling curve is called connected, if for any two neighbouring intervals $I^{(\nu)}$ and $I^{(\mu)}$ also the corresponding subdomains $Q^{(\nu)}$ and $Q^{(\mu)}$ are direct neighbours, i.e. share an $(n - 1)$-dimensional hyperplane.
Connected, Recursive Space-filling Curves

Examples:
- all Hilbert curves (2D, 3D, ...)
- all Peano curves

Properties: connected, recursive SFC are
- continuous (more exact: Hölder continuous with exponent $1/n$)
- neighbourship-preserving
- describable by a grammar
- describable in an arithmetic form
  (similar to that of the Hilbert curve)

Related terms:
- face-connected, edge-connected, node-connected, ...
- also used for the induced orders on grid cells, etc.
Approximating Polygons of the Hilbert Curve

**Idea:** Connect start and end point of iterate on each subcell.

**Definition:**

The straight connection of the $4^n + 1$ points

$$h(0), h(1 \cdot 4^{-n}), h(2 \cdot 4^{-n}), \ldots, h((4^n - 1) \cdot 4^{-n}), h(1)$$

is called the *n-th approximating polygon of the Hilbert curve*.
Properties of the Approximating Polygon

- the approximating Polygon connects the **corners** of the recursively divided subsquares
- the connected corners are start and end points of the space-filling curve within each subsquare
  \[ \Rightarrow \text{assists in the construction of space-filling curves} \]
- approximating polygons are constructed by recursive repetition of a so-called **Leitmotiv**
  \[ \Rightarrow \text{similarity to Koch and other fractal curves} \]
- the sequence of corresponding functions \( p_n(t) \) converges **uniformly** towards \( h \)
  \[ \Rightarrow \text{additional proof of continuity of the Hilbert curve} \]
Part I

3D Hilbert Curves
3D Hilbert Curves

- Wanted: connected, recursive SFC, based on division-by-2
  ⇒ leads to 3 basic patterns:

- in addition: symmetric forms, change of orientation
- always two different orientations of the components
  ⇒ numerous different Hilbert curves expected

Exercise: construct a 3D Hilbert curve!
3D Hilbert Curves – Iterations

1st iteration

2nd iteration
3D Hilbert Curve – Arithmetic Representation

t given in the octal system, \( t = 0_8.k_1k_2k_3k_4 \ldots \), then

\[
h(0_8.k_1k_2k_3k_4 \ldots) = H_{k_1} \circ H_{k_2} \circ H_{k_3} \circ H_{k_4} \circ \cdots \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\]

with operators

\[
H_0 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}x + 0 \\ \frac{1}{2}y + 0 \\ \frac{1}{2}z + 0 \end{pmatrix}
\]

\[
H_1 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}z + 0 \\ \frac{1}{2}y + \frac{1}{2} \\ \frac{1}{2}x + 0 \end{pmatrix}
\]

\[
H_2 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}x + \frac{1}{2} \\ \frac{1}{2}y + \frac{1}{2} \\ \frac{1}{2}z + 0 \end{pmatrix}
\]

\[
H_3 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}z + \frac{1}{2} \\ -\frac{1}{2}x + \frac{1}{2} \\ -\frac{1}{2}y + \frac{1}{2} \end{pmatrix}
\]
3D Hilbert Curve – Arithmetic Representation

(continued)

\[
H_4 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}z + 1 \\ -\frac{1}{2}x + \frac{1}{2} \\ \frac{1}{2}y + \frac{1}{2} \end{pmatrix} \quad H_5 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}x + \frac{1}{2} \\ \frac{1}{2}y + \frac{1}{2} \\ \frac{1}{2}z + \frac{1}{2} \end{pmatrix}
\]

\[
H_6 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}z + \frac{1}{2} \\ \frac{1}{2}y + \frac{1}{2} \\ -\frac{1}{2}x + 1 \end{pmatrix} \quad H_7 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}x + 0 \\ -\frac{1}{2}z + \frac{1}{2} \\ -\frac{1}{2}y + 1 \end{pmatrix}
\]

⇒ leads to algorithm analog to 2D Hilbert and 2D Peano
⇒ uses only one pattern; each in only one orientation
3D Hilbert Curves – Variants

Different approximating polygons:

- same basic pattern:
  - same order of the eight sub-cubes
- differences only noticeable from the 2nd iteration
3D Hilbert Curves – Variants (2)

Different orientation of the sub-cubes:

- same basic pattern Grundmotiv, same approximating polygon
- differences only visible from 2nd iteration
Part II

Peano Curves in Higher Dimensions
Construction of the Peano Curve

Recursive Construction:
- divide quadratic domain into 9 subsquares
- construct Peano curve for each subsquare
- join the partial curves to build a higher level curve
Approximating Polygons of the Peano Curve

Definition:
The straight connection between the $9^n + 1$ points

$p(0), p(1 \cdot 9^{-n}), p(2 \cdot 9^{-n}), \ldots, p((9^n - 1) \cdot 9^{-n}), p(1)$

is called $n$-th approximating polygon of the Peano curve
3D Peano Curves

- Concentration on “serpentine” Peano curves (no Meander-type)
- still lots of different variants
- especially interesting are dimension-recursive variants:

  in each 3D cut, the sub-cubes are again traversed in Peano order
2D Peano Curve – Dimension-Recursive Grammar

Illustration of patterns:

Construction of Grammar:

\[
P \quad \leftarrow \quad P_y \rightarrow R_y \rightarrow P_y \quad \quad P_y \quad \leftarrow \quad P \uparrow Q \uparrow P
\]

\[
Q \quad \leftarrow \quad Q_y \leftarrow S_y \leftarrow Q_y \quad \quad Q_y \quad \leftarrow \quad Q \uparrow P \uparrow Q
\]

Note: dimensional “stretching” implied via index notation (y)
Arithmetic Formulation of the Peano Function

In addition to the classical 2D-construction in the “nonal” system, there is also a dimension-splitting approach based on ternary system:

\[ t = 0_3.t_1 t_2 t_3 t_4 \ldots, \text{ then} \]

\[ p(0_3.t_1 t_2 t_3 t_4 \ldots) = P_{t_1}^x \circ P_{t_2}^y \circ P_{t_3}^x \circ P_{t_4}^y \circ \ldots \left( \begin{array}{c} 0 \\ 0 \end{array} \right). \]

with the operators

\[ P_0^x \left( \begin{array}{c} x \\ y \end{array} \right) = \left( \begin{array}{c} x + 0 \\ \frac{1}{3} y + 0 \end{array} \right), \quad P_1^x \left( \begin{array}{c} x \\ y \end{array} \right) = \left( \begin{array}{c} -x + 1 \\ \frac{1}{3} y + \frac{1}{3} \end{array} \right), \quad P_2^x \left( \begin{array}{c} x \\ y \end{array} \right) = \left( \begin{array}{c} x + 0 \\ \frac{1}{3} y + \frac{2}{3} \end{array} \right) \]

and \( P^y \) analogously.

**Key idea:** each ternary digit defines scaling in only one dimension!
Peano’s Representation of the Peano Curve

Definition: (Peano curve, original construction by G. Peano)

• each \( t \in \mathcal{I} := [0, 1] \) has a ternary representation

\[
t = (0_3.t_1 t_2 t_3 t_4 \ldots)
\]

• define the mapping \( p: \mathcal{I} \to \mathcal{Q} := [0, 1] \times [0, 1] \) as

\[
p(t) := \left( \begin{array}{c} 0_3.t_1 k_{t_2}(t_3) k_{t_2+t_4}(t_5) \ldots \\ 0_3.k_{t_1}(t_2) k_{t_1+t_3}(t_4) \ldots \end{array} \right)
\]

where \( k(t_i) := 2 - t_i \) for \( t_i = 0, 1, 2 \) and \( k^j \) is the \( j \)-times concatenation of the function \( k \)
Peano’s Representation of the Peano Curve (2)

Still to prove:

- $p$ is independent of the ternary representation
- the Peano curve $p : I \rightarrow Q$ defines a space-filling curve.

Comments:

- the direction of “switchback” can be both vertical (see definition), horizontal, or mixed;
- actually, 272 different Peano curves of the switchback type can be constructed using the same principles;
  For comparison: there are only two different 2D Hilbert curves
- in addition: Peano-Meander curves
Part III

Fractal Curves
Recall: Approximating Polygons

First approximating polygons of the Hilbert curve:

- polygon results from recursive repetition of a basic pattern → “Leitmotiv”
- note: Leitmotiv “added” to alternating sides of the polygon (compare location of entry/exit points in the illustration)
- strong similarity to Fractal curves
Example: Koch Curve
How Long are Approximating Polygons?

Example: Hilbert curve

- polygon results from recursive repetition of the Leitmotiv
- every recursion step **doubles** the length of the polygon in each subsquare
  \[ \Rightarrow \text{length of the } n\text{-th polygon is } 2^n \rightarrow \infty \text{ for } n \rightarrow \infty. \]

Corollaries:

- the “length” of the Hilbert curve is not well defined
- instead, we can give an “area” of the Hilbert curve
  \( 1, \text{the area of the unit square} \)
  \[ \Rightarrow \text{Question: what’s the dimension of a Hilbert curve?} \]
Fractal Dimension of Curves

Measuring the length of a curve:

- approx. the curve by a polygon with faces of length $\epsilon$
  $\Rightarrow$ gives a measured length $L(\epsilon)$.
  *(cmp. approximating polygons of a space-filling curve)*
- in case of recursive repeat of a Leitmotiv:
  replace each units of length $r$ by a polygon of length $q$, then
  $$L\left(\frac{\epsilon}{r}\right) = \frac{q}{r}L(\epsilon), \quad L(1) := \lambda$$

- we obtain for the length $L(\epsilon)$:
  $$L(\epsilon) = \lambda\epsilon^{1-D}, \quad \text{where} \quad D = \log_r q = \frac{\log q}{\log r}$$
Fractal Dimension of Curves (2)

Length of a recursively defined curve computed as

\[ L(\epsilon) = \lambda \epsilon^{1-D}, \quad \text{mit} \quad D = \log_r q = \frac{\log q}{\log r} \]

⇒ \( D \) is the **fractal dimension** of the curve
⇒ \( \lambda \) is the length w.r.t. that dimension

Gives “well defined” dimension:
- in all other “dimensions”, the length is 0 or \( \infty \)!
- the fractal dimension of the 2D Hilbert curve is 2, similar for the Peano curve

⇒ **Hausdorff dimension**
How Long is the Coastline of Britain?

Compare, e.g., Mandelbrot: The Fractal Geometry of Nature
Test: Length of Fractal Curves

- circle
- koch
- gosper
- britain

Graph showing the length of different fractal curves as a function of epsilon.
Exercise: What is the Area of a Fractal Curve?

Koch curve as example:

→ refine green area and compute its limit value …