Daniel Biella, Nelson Baloian, Wolfram Luther

GPS Drawing on Street Networks: Extracting Routes from Polygonal Coverings

This contribution provides important fundamentals of digital geometry and algorithms based on polygonal coverage and chain codes to generate GPS drawings. Illustrative examples created with the GPSVisualizer and GoopleMaps show the difficulty in finding suitable road networks.

Thanks to the organizers and authors of the proceedings

Universidad de Chile
Departamento de Ciencias de la Computación

AUA American University of Armenia

UNIVERSITÄT DUISBURG ESSEN

First of all, as co-editor of the CODASSCA 2020 conference proceedings, I would like to thank all authors and the American University of Armenia that we can organize a conference in a new form despite the Covid19 crisis and that 24 nice contributions have come together, which you can access via our open access proceedings.
This contribution provides important fundamentals of digital geometry and algorithms based on polygonal coverage and chain codes to generate GPS drawings. Illustrative examples created with the GPSVisualizer and GoopleMaps show the difficulty in finding suitable road networks.

Motivation

Project PRASEDEC 2014-2016
UDE – UCH

Algorithmic approach to GPS Painting

Fig. 1. Walk across the campus of University of Chile: Public domain under Creative Commons 4.0 license

We worked with the data collected from a walk in Parque O'Higgins, Tupper, Beauchef, Blanco Encalada, Jose Miguel Carrera, Domeyko, and Almirante Latorre on Tuesday, September 3, 2013 between 17 and 18h using several smart phones.

There are various reasons for the three authors to write this article. One time we were all impressed by the experiences at the University of Chile when, as part of the PRASEDEC project under the direction of Nelson Baloian and Bernd Noche, professor and logistician at the University of Duisburg-Essen, we worked on recording routes in the university district using a mobile phone and GPS. The path is shown in Figure 1; the accuracy of the position determination depends strongly on the signal of the involved satellites, the width of the road, the buildings and the vegetation along the way (see reference Zogg). In this figure we have used the thickness of the trajectory to represent the accuracy of the GPS signal and the color for the type of surrounding. Interestingly, the path indicates a road side change that did not take place at all due to a series of huge buildings and a change of sat-ellites providing the GPS signal. A second reason was the fact that publications on GPS painting contain one or the other relevant algorithmic approach, but lacks relevant references from the previous millennium on digital curves and geometry. Often, they reinvent the wheel, which then doesn't even run smoothly (see Waschk, Janser et al.).

Examples

Fig. 2a. Trace of the walk, b. a star, c. a light bulb (length 3.4 km) in Brooklyn, and d) a circle in Yerevan

- The trace shows the accuracy of the GPS coordinates (parameter linewidth: higher values mean lower accuracy). The color used represents type of environment—forest, trees, small and high buildings, free areas, streets and crossings
- Rosner mentions the challenge to paint a star and a circle which is quite simple with a regular street grid at hand (cf. Figure 2b, Brooklyn, NY)
- Special arcs in the Brooklyn road network can be used to draw a lightbulb
- A grid-covering algorithm was be applied to the city map of Yerevan.

Further examples

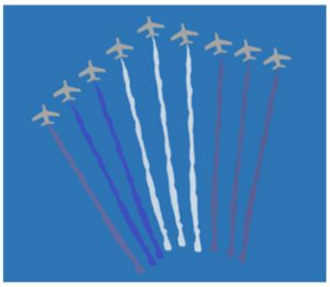


Fig.3. Parade on July 14, 2018 Paris Asynchronous aleatory contrails
Source: LIBERATION, (together with AFP), photo by Marc Chaumeil

An example of the painting being generated by moving persons is the synchronous collaborative creation of a colored drawing in the sky by air force pilots with the contrails of nine military jets and the asynchronous creation of contrails being dispersed by the wind, or traces left in the snow by skiers. Notice the wrong color on the aircraft on the left.

So it is obvious to represent covered ways on land, on water and in the air by means of waypoints and their coordinates determined by GPS as a route of straight line segments. Drawings are created with the help of covered ways (Coeurjolly et al.) or, by running different destinations and sights in a city via suitable ways, which connect them in a certain order. The route shows an interesting object or fulfils other criteria, like that of a shortest possible way or the adherence to an order of the visited sights, or finally only the condition that one arrives at the starting point again or uses recommended ways. When using road networks, it is necessary to select a repository for realization that meets certain requirements about road categories and quality criteria and to scale and align the work accordingly.

Concepts in the world of drawing, canvas, pencil, strokes and metaphorical correspondences in the world of streets and maps

Artist	Person generating the GPS painting
Brush	Means of transport, satellite signals
Canvas	Map section, polygonal area on land, at sea or in the air
Artwork	Objects of digital geometry: digitized letters or traces by brush strokes as products of travel or symbols for it
Text and letters	Text in paintings is not linear. Visual languages with appropriate grammars can be used to describe text flow in two or three dimensions. A letter or icon can be produced by a pattern or strokes or by splines, which can be scaled and rotated
Style elements Signs, colors and forms	Style elements for curves, polygons, interior areas and the contours of domains, such as fill color and line attributes, uncertainty modeling and visualization. Thickness and opacity illustrate regions of uncertainty across the spatial domain, supplemented by annotations.
Collaborative generation	Travel from or to a location, airport or port, carried out simultaneously or with a time delay
Quality	Quality criteria and dimensions such as path length, duration or proximity to the given overlay painting and attractiveness of the route

The canvas for GPS drawings can be created in two different ways: a) extracting roads, building footprints and other features in the areas between them based on satellite images and GPS data and mapping them via a suitable projection or b) deriving a road network incrementally and constructing it based on a camera tour or the trajectories of many vehicles and their uploaded GPS data. Mattyus et al. share details of new deep-learning and weakly supervised training models and make data and specialized map-editing, -reviewing and -verifying services available to the global mapping community through Map with AI. Then we consider how to find the shortest possible ways between places by using routes from a road network or how to have pictorial elements such as circles or straight lines in sufficient number and density within a city maps to compile a route from them.

Algorithm 1. Street approximation of a rectifiable Jordan arc **c**(t) with start and end points S and E in two dimensions:

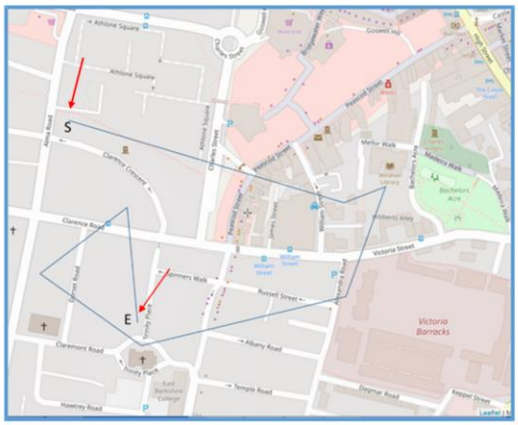


Fig. 4. Polygonal line built from straight line segments: a Jordan arc

Task: Construct a Jordan curve from sections of the road network, as close as possible to the initial trajectory or on a given side thereof

The task can be solved with borrowings from digital geometry and raster algorithms like Bresenham's for routes, polygons and circular arcs. Assume that there is a trajectory of straight lines that connects a starting point with an end point, typically crossings on a city map. The street map divides the plane into polygons, ideally rectangles, which can be used to cover the curve in such a way that there is a sequence of polygonal areas the way passes through in a certain order, entering and leaving each polygon. From the sequence of intersection points connected by boundary parts of each polygonal area traversed, a Jordan curve is constructed from sections of the road network, as close as possible to the initial trajectory or on a given side thereof. Now, we want to highlight a first algorithm.

Step 1

- Construct a nearest polygonal covering $P:=\{P_1, P_2, \ldots.\}$ in which each member is bordered by segments of allowed streets (straight line segments or circular arcs) of a given category

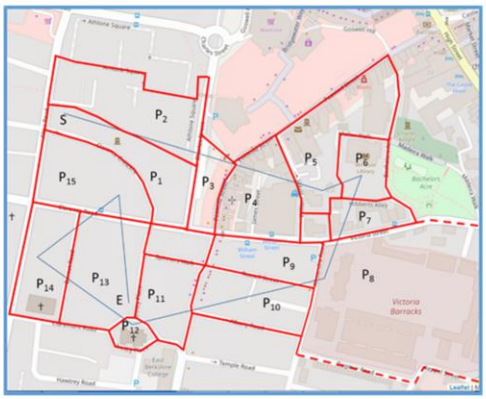


Fig. 5. Polygonal covering P_1 to P_{15} of a Jordan arc

Street approximation of a rectifiable Jordan arc c of t with start and end points S and E in two dimensions

Construct a nearest polygonal covering P in which each member is bordered by segments of allowed streets (straight line segments or circular arcs) of a given category

Step 2

- Collect an ordered list $L_c = \{\mathbf{c}(t_i), i=1,\ldots,n\}$ of intersection points ($t_i < t_{i+1}$) of $\mathbf{c}(t)$ with the covering P (avoid corner points).

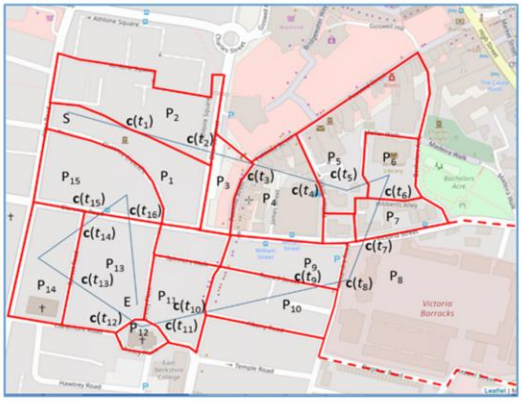


Fig. 6. Polygonal covering of a Jordan arc with intersection points $\mathbf{c}(t_i)$

Collect an ordered list L c of c of t i of intersection points with the covering P (avoid corner points).

Further steps

- Start with $\mathbf{c}(t_1)$, and set $P_1=P_{j(0)}$ that contains a part of $\mathbf{c}(t)$, $t<t_1$.
- For i:=1 to n: determine $P_{j(i)}$ with border point $\mathbf{c}(t_i)$, which contains a part of $\mathbf{c}(t)$; $t>t_i$, $P_{j(i-1)}$ contains a part of $\mathbf{c}(t)$, $t<t_i$, resulting in a list L_p = $\{(P_{j(i-1)}, P_{j(i)}), i=1,\ldots, n, j=j(i)\}$ of adjacent polygons.
- Try to assemble a path built from parts of the boundary $\partial P_{j(i)}$, i=0,1,2,…., such that the optimality criteria (boundary parts of each member of the polygonal covering used, double crossed sections eliminated, shortest or closest Jordan path constructed) are fulfilled.

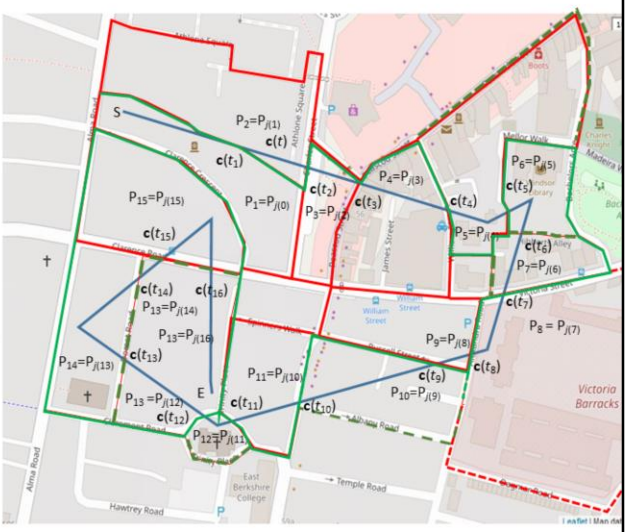


Fig. 7. Extracted routes from polygonal covering – inner and outer variants

Start with c t one, and set P one equal P j zero that contains a part of c with t smaller than t one.

For I equal to1 to n: determine P j of i with border point c of t i, which contains a part of c for t greater than t i ,

P j of i minus 1 contains a part of c with t smaller than t i, resulting in a list L p of adjacent polygons P j of I minus 1 and P j of i.

Try to assemble a path built from parts of the boundary of P j of I for I equal to 0, 1, 2,…., such that the optimality criteria (boundary parts of each member of the polygonal covering used, double crossed sections eliminated, shortest or closest Jordan path constructed) are fulfilled.

If there is no connected continuous path, try to spread to adjacent polygons or assume polygons with simply connected inner domains with respect to the four-neighborhood topology, the edges of which can be traversed in both directions. If the arc passes the same polygon twice or is part of a closed Jordan curve, try to use roads/trails inside or outside.

A second algorithm

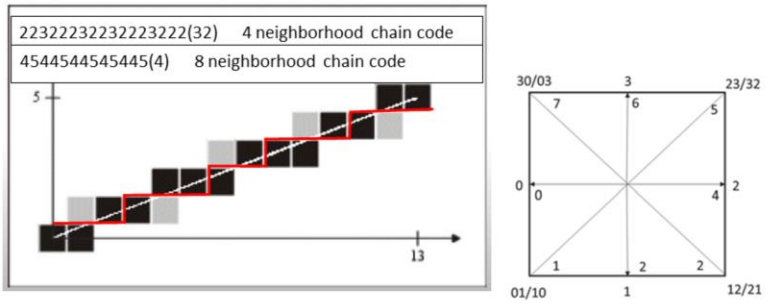


Fig. 8. Digitizing a straight line with the Bresenham algorithm, and 4/8 neighbors chain code

Another approach is even closer to the well-known Bresenham algorithm for straight lines.

The description of a digital straight line uses the so-called chain code of the four or eight neighbor topology. Suppose you are standing at an intersection, preferably on a suitable edge of a regular rectangular road network with roads running from west to east and south to north, you can leave the point of view in four directions: to the west, coded with a 0, to the south, east and north, each coded with 1, 2, or 3 (cf. Figure 8)

The idea is to superpose a regular grid over the curve and then substitute the curve with a staircase construction using the Bresenham algorithm so that the road sections run as close as possible around the curve and always change sides; each step or change of direction is described with the chain co-de, starting point and chain code sequence describe the digitized initial curve in a unique way.

Finally the chain code is applied to a rectangular real map section.

Step 1: Choose a grid to digitize a drawing, take five fingers as an example

Step 2: Compute four neighborhood chain code

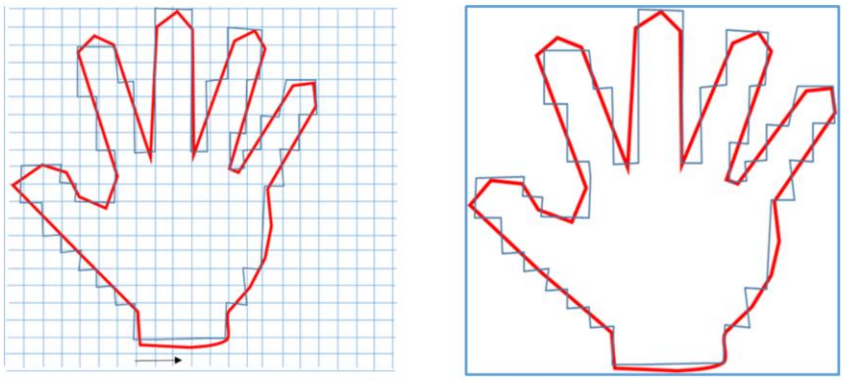


Fig. 9. Five fingers and chain code

Chain code:

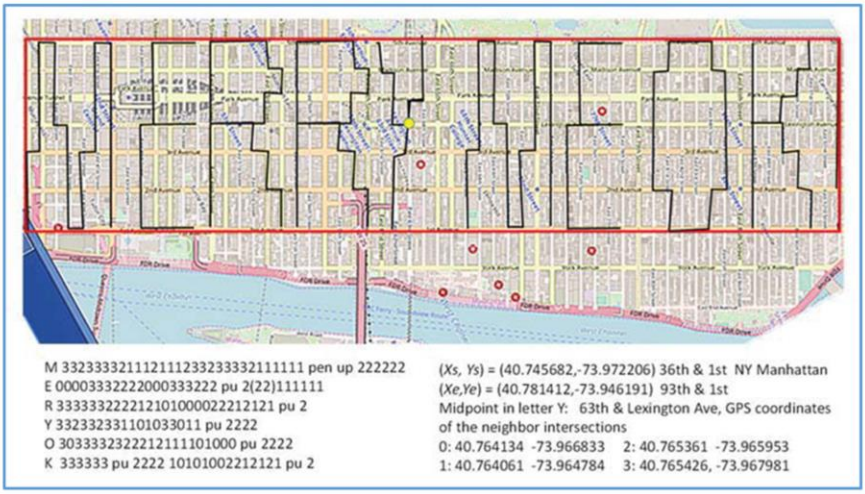
22222332332333332332332330011011010332333233300111011110333333330011111110333303300111211121110030300112112121212121211

Step 3: Select canvas

Step 4: Visualize chain code

Fig. 10. Visualized chain code on Chicago canvas – network needs scaling

Further example



M 3323333211121112332333321111111 pen up 222222
E 0000333222200033322 2 pu 2(22)111111
R 3333332222121010000222121 pu 2
Y 332332331101033011 pu 2222
O 30333323222121111101000 pu 2222
K 333333 pu 2222 101010022121121 pu 2

(*Xs, Ys*) = (40.745682,-73.972206) 36th & 1st NY Manhattan
(*Xe,Ye*) = (40.781412,-73.946191) 93th & 1st
Midpoint in letter Y: 63th & Lexington Ave, GPS coordinates of the neighbor intersections
0: 40.764134 -73.966833 2: 40.765361 -73.965953
1: 40.764061 -73.964784 3: 40.765426, -73.967981

Fig. 11. Digitized text "MERRY ME OK" with four-chain code on rectangular (0, 57) x (0, 6) Manhattan canvas – chain code for used letters - start and end points - example for GPS coordinates of an intersection

Digitized text "MERRY ME OK" with four-chain code on rectangular (0, 57) times (0, 6) on Manhattan canvas –

chain code for used letters - start and end points - example for GPS coordinates of an intersection

Another algorithmic approach - Algorithm 3: GPS drawing

- Choose a regular grid (X, Y) and coordinates $0 \le x \le X$, $0 \le y \le Y$, with a given mesh size.
- Digitize the painting as polyline and describe it with a four-neighbor chain code.
- Identify the start and end points of the curve with grid point coordinates (latitude, longitude) (*Xs, Ys*), (*Xe, Ye*). Place it so that it does not leave the canvas.
- Mark segments that are not drawn with pen up (pu).
- Identify as canvas an area with a street network of the specified category that is as regular as possible with $(X + 1)\cdot(Y + 1)$ crossing points, whose GPS coordinates are given, and determine start and end point intersections. To this end, create two-name lists of all rectangular streets in the (X, Y)-rectangle.
- To get an array IP of all intersections of two streets situated in the rectangle in Google Maps, submit the request in the format: [[Street Name A] & [Street Name B], City].
- The API will return latitude and longitude coordinates of the intersection.
- Then add the four nearest neighbors in the direction 0, 1, 2, 3 to each intersection in IP.
- Describe the artwork according to the affected meshes using the list of GPS intersection coordinates of the constructed path.
- Visualize the digitized path on the canvas. The length of the path can be optimized if, in addition to the horizontal and vertical sections of the path, existing cross connections between the intersections (in the eight-neighbor topology) are used.

Choose a regular grid (X, Y) and coordinates x from interval 0 capital x, y from interval 0 capital Y, with a given mesh size.
Digitize the painting as polyline and describe it with a four-neighbor chain code.
Identify the start and end points of the curve with grid point coordinates (latitude, longitude) (Xs, Ys), (Xe, Y e). Place it so that it does not leave the canvas.
Mark segments that are not drawn with pen up
Identify as canvas an area with a street network of the specified category that is as regular as possible with (X + 1) times (Y + 1) crossing points, whose GPS coordinates are given, and determine start and end point intersections. To this end, create two-name lists of all rectangular streets in the (X, Y)-rectangle.
To get an array IP of all intersections of two streets situated in the rectangle in Google Maps, submit the request in the format: [[Street Name A] & [Street Name B], City].
The API will return latitude and longitude coordinates of the intersection.
Then add the four nearest neighbors in the direction 0, 1, 2, 3 to each intersection in IP.
Describe the artwork according to the affected meshes using the list of GPS intersection coordinates of the constructed path.
Visualize the digitized path on the canvas. The length of the path can be optimized if, in addition to the horizontal and vertical sections of the path, existing cross

connections between the intersections (in the eight-neighbor topology) are used.

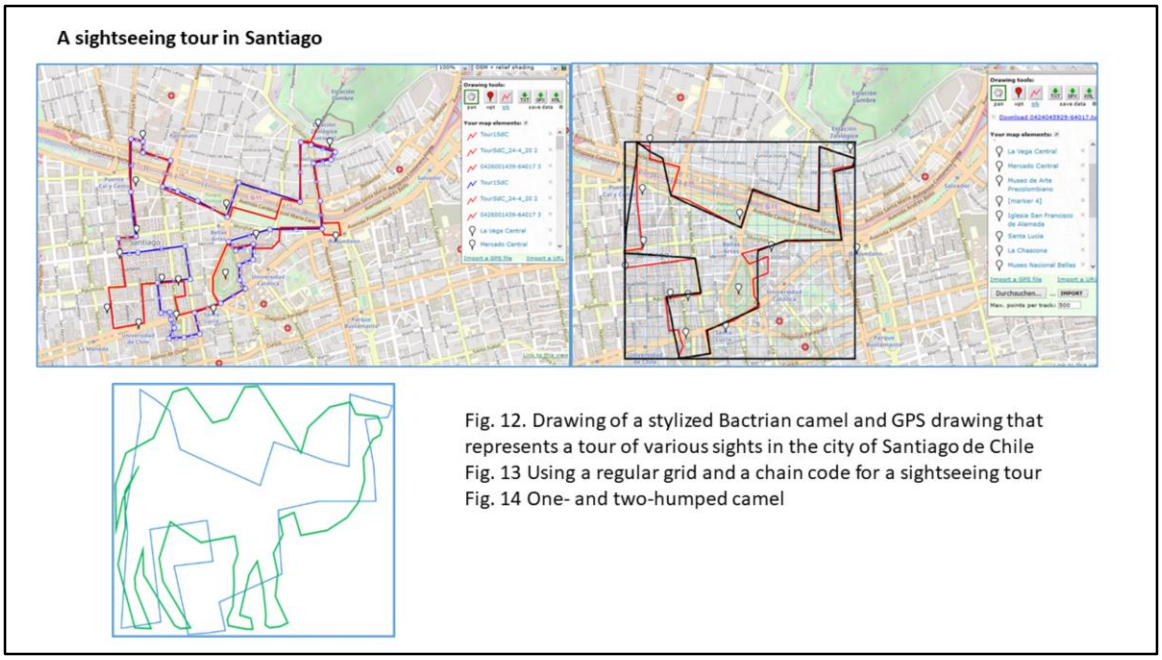


Fig. 12. Drawing of a stylized Bactrian camel and GPS drawing that represents a tour of various sights in the city of Santiago de Chile
Fig. 13 Using a regular grid and a chain code for a sightseeing tour
Fig. 14 One- and two-humped camel

The following pictures show examples and results for special curves or figures. Figure 12 shows a drawing of a stylized Bactrian camel and GPS drawing that represents a tour of various sights in the city of Santiago de Chile. Figure 13 uses a regular grid and a chain code for a sightseeing tour and the third figure shows a stylized one- and two-humped camel

It becomes clear that the actually difficult task is to find suitable sections of road maps of settlement areas that meet certain criteria, i.e. that contain dense regular grids supplemented by circular arcs or diagonals that are needed for a drawing.
In the following we would like to show some examples and explain which software can be used to create the drawings. Finally, we will explain why there are no regular street networks of any size, and that it is useful to look at the architectural plans of historical cities if you want to follow and draw special shapes.

Fractal cities

Fig.15. Palmanova as an ideal city in the shape of a star after Georg Braun and Frans Hogenberg ~1600

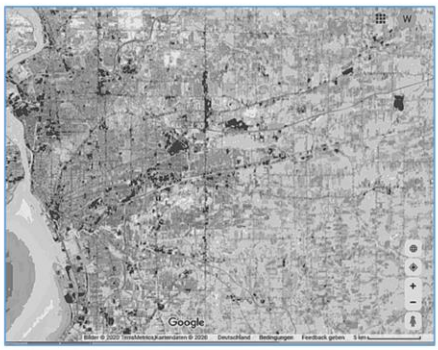


Fig. 16 shows the urban settlement (population area and green belt) of Buffalo, NY USA with streets (transportation network), buildings, and green areas. Regular grids (of limited size) are only located in the city center. Box dimension is 1.729 (1990, Batty)

N(d) denotes the number of boxes of side length d to cover a plane object O. Upper (lower) counting box dimension D(O) is defined as lim sup (inf) $_{d\to 0}$log N(d)/log(1/d), and both are upper bounds for the strong Hausdorff dimension H(O).

The authors Batty and Longley apply fractal geometry to understand shapes of cities and construction principles developed by city planners and architects in contrast to the historic growth and expansion (cf. page 242, Table 7.1. The preliminary evidence for fractal cities). They observe that a simple geometric pattern repeated on different scales and self-similarity play an important role. Strong geometric city layouts such as the octagonal Palmanova in Italy are worth mentioning, as are circular towns (e.g., Karlsruhe) or regular cell growth (e.g., Savannah), whereas Beijing has a fractal dimension close to two, but only a roughly regular grid. The observed fractal growth of large cities indicates that no regular street grids of any size can exist and that Euclidean objects cannot be approximated with equal accuracy in any size.

Conclusion

There is a rich set of tools of digital geometry methods and freely accessible software to implement GPS drawings. The efficient localization of suitable road networks using intelligent learning approaches, scaling and adaptation of the canvas, and an evaluation of the user experience are priority topics for a relevant research question and planned for future work.

Acknowledgement

The material was partly published together with Nelson Baloian and Daniel Biella in the CODASSCA 2020 Proceedings "Collaborative Technologies and Data Science in Artificial Intelligence Applications". The figures were created with the GPSVisualizer and GoopleMaps.

GPSVisualizer is a sketchbook of GPS artists and proposes a standardized input form that automatically draws your GPS data (or various exchange formats KML/ KMZ file, etc.) overlaid upon a variety of background maps and imagery, using either the Google Maps API or Leaflet, an open-source mapping library and freehand drawing utility that allows users to interactively draw on a map creating their own GPX or KML file. We used this software to produce figures 2, 4-7, and 10-13.

Google Maps can be used to build highly customizable and scalable maps with their own content and imagery and to import features from KML files, spreadsheets and other files formats (CVM, KML, KMZ, GPX, and XLSX). It supports the user in creating complex applications and powerful visualizations of the data on a modern web platform with a comprehensive user interface (see Figure 16).

Wolfram Luther University, Duisburg-Essen, luther@inf.uni-due.de, www.scg.inf.uni-due.de

I like to thank you for your interest and I am happy to answer any questions you may have

References

Batty, M., Longley, P.: Fractal Cities: A Geometry of Form and Function. Academic Press, San Diego, CA and London (1994) http://www.fractalcities.org/

Coeurjolly, D., Zerarga, L.: Supercover model, digital straight line recognition and curve reconstruction on the irregular isothetic grids. Computers & Graphics 30(1) 46–53 (2005)

Google Maps https://maps.google.com

GPS Visualizer https://www.gpsvisualizer.com/draw/

Janser, A., Luther, W., Otten, W.: Computergraphik und Bildverarbeitung, Vieweg 1996

Mattyus, G., Luo, W., and Urtasun, R.: Deep road mapper: Extracting road topology from aerial images. In The IEEE Inter. Conf. on Computer Vision (ICCV), Oct 2017. Machine Vision and Applications 28, 679–694 (2017)

Palmanova https://de.wikipedia.org/wiki/Palmanova#/media/Datei:Palmanova1600.jpg

Rosner, D., K., Saegusa, H., Friedland, J., Chambliss, A.: Walking by Drawing. Proceedings CHI 2015, Crossings, Seoul, Korea, April 18–23, 397–406 (2015)

Waschk, A., Krüger: J.: Automatic route planning for GPS art generation. Computational Visual Media 5(3) 303–310 (2019)

Zogg, J.-M.: GPS Essentials of Satellite Navigation Compendium. U-blox AG (2009)