Computational Visual Media
https://doi.org/10.1007/s41095-019-0146-z Vol. 5, No. 3, September 2019, 303–310
Research Article

Automatic route planning for GPS art generation

Andre Waschk[1] (✉), **Jens Krüger**[1] (✉)

Abstract In this paper, we present a novel approach to automated route generation of global positioning system (GPS) artwork. The term GPS artwork describes the generation of drawings by leaving virtual traces on digital maps. Until now, the creation of these images has required a manual planning phase in which an artist designs the route by hand. Once the route for this artwork has been planned, GPS devices have been used to track the movement. Using our solution, the lengthy planning phase can be significantly shortened, thereby opening art creation to a broader public.

Keywords GPS art; walking; routing; shape detection; navigation

1 Introduction

Global positioning system (GPS) art, as it is known today, is a form of location-based tracking using GPS hardware as a digital pen to create astonishing large-scale artwork. It can be considered the 21st-century version of large-scale drawings on the landscape, a concept known for millennia. As early as around 500 BCE, the Nazca culture created the Nazca Lines, which have been recognized as a UNESCO world heritage site [1].

One of the best known such works of art was created by Jeremy Wood in 2010. He was hired as a GPS artist to create a work of art, called "Traverse Me", for the University of Warwick [2]. Wood used GPS trackers to record his path as he walked across the university campus without having any immediate feedback of his taken route. After finishing his walk, he used the recorded data to visualize the path he walked and finalize his artwork.

Due to the integration of GPS into consumer electronic devices, such as mobile phones and smartwatches, recording our motions has become simpler than ever. Over the last decade, these smart devices have become more and more powerful, allowing us to compute routing solutions on the fly right when we need them.

Over time, GPS art has also attracted the interest of sports analytics. Thanks to integrated tracking hardware, runners and cyclists have become especially interested in sharing their latest workouts with other enthusiasts on platforms like STRAVA [3]. Sports devotees around the world have started running simple figures to give additional motivation to their daily workout programs.

Stephen Lund, a cyclist and GPS artist, described his process of planning and riding a GPS artwork in a Ted Talk in 2015 [4]. According to Lund, the primary process consists of glancing at a map and waiting for inspiration. The artwork presented on his website [5] is the result of this glancing-at-a-map paradigm.

In our work, we target a different situation. Instead

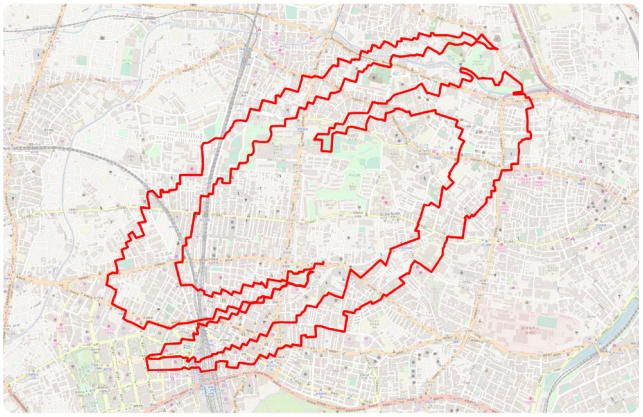

Fig. 1 The SIGGRAPH logo as a GPS route on top of Tokyo computed using our approach.

1 University of Duisburg-Essen, 47057 Duisburg, Germany. E-mail: A. Waschk, Andre.Waschk@uni-due.de (✉); J. Krüger, Jens.Krueger@uni-due.de (✉).

Manuscript received: 2019-02-27; accepted: 2019-05-19

of attempting to find a shape that naturally exists on a given map, in many applications it is desirable to find a path for a given shape (see Fig. 2) and optimize the route and its placement, size, and orientation according to the shape.

Fig. 2 The largest GPS drawing recognized by the Guinness Book of World Records [6].

2 Related work

Even though the idea of GPS art has been around for several decades, to the best of our knowledge, it has never been of particular interest to the computer graphics community.

The work most closely related to our approach is the thesis of Balduz [7]. He approached the GPS art problem by rasterizing the map and the target figure into a regular grid. He then converted the streets into a binary image where black pixels represent a street and white everything else. To compute the best possible position for the figure, he centered it above every pixel, computed the minimal sum of distances between the figure and the map image, and then selected the position with the smallest value.

The goal of Balduz's work was to use image operators to find the best possible match on the map for the given figure. His intention was to design the approach so that GPUs could be used to parallelize a brute force approach to the needed similarity computations. Neither a GPU implementation nor the computation of the actual route from the pixel cloud is solved, leaving both problems open for future work. Balduz makes no mention of the performance of the system. Although his result resembles the input figure, we consider that the discrete image-based approach has numerous drawbacks compared to a graph-based solution. Firstly, it is unclear how to handle particular characteristics of the streets, such as a one-way designation, or three-dimensional information, such as bridges and overpasses, and secondly, it is unclear how to turn the pixel cloud into a connected route automatically.

Another work that explicitly addresses GPS art is that of Rosner et al. [8], which focuses on the human–computer interaction aspect of the software supporting the route computation task. The goal of this work was to motivate users to explore the city by tracing figures sent to them by friends and annotated with little personal messages. For the route computation, Rosner et al. rely on the iOS built-in mapping system. In Section 4, we point out the problems with standard routing algorithms for the task of GPS artwork. Since Rosner et al. mainly focus on the input and community aspect of GPS art and consider routing as a black box, it should be straightforward to integrate our solution with their application.

In the field of GPS art, the most massive GPS artwork traversed by a man on foot, and recognized by the Guinness Book of World Records, is a 7.163 km long marriage proposal by Yasushi Takahashi (see Fig. 2). Besides the considerable length of the artwork, it also differs in another way from the widely used approach of GPS art generation. Whereas the commonly used approach analyzes the road network first and then recognizes and draws figures that are implied by the road itself, we can see in this proposal that the artist had a precise image in mind, the text "Marry Me", and was forced to find his way across the islands of Japan.

A related concept, known as *snakes*, is used for feature detection in images [9] and triangular meshes [10]. A different concept for the visualization of figures within graphs can be found in Ref. [11]. None of these works, however, fit the requirements of GPS art: the route has to be a sub-graph of the existing road map.

A similar idea tailored toward the usage of drones can be found in Refs. [12, 13], in which the focus is on introducing an aerial sketch of the flying drones' behavior in single images. Although the concept of a still image of time-varying data seems to be similar to our idea, drones are much less constrained in producing artistic sketches since they can freely move throughout the sky without any streets to be considered.

3 Contribution

The contribution of this paper is a novel algorithm for route generation that is specifically tailored for the generation of GPS art routes. In contrast to the widely adopted process for the creation of GPS artwork, as described by Lund [4], our approach automatically computes an eye-pleasing route and opens the artistic style of GPS art to a broader public, enabling users to generate GPS artwork for any specific figures they choose, e.g., a marriage proposal, a company/brand logo, a name, a date, or other text. More specifically, the algorithm takes as input the map region, starting position, and additional parameters defining the quality metrics of our approach, such as desired maximum length of the route to compute the final artwork. From this data, our approach computes a route that resembles the input figure much better than any off-the-shelf mapping solutions.

As we do not require additional data structures or any precomputation, we anticipate that our proposed solution can be easily integrated into existing mapping solutions such as Google Maps [14], Bing Maps [15], or other popular software and services.

4 Issues with standard solutions

The prevailing idea for the creation of routes for GPS artwork is to use existing routing algorithms that are widely available on mapping solutions and navigation systems. On closer examination, however, it becomes clear that these systems optimize the routes for goals that are very different from the ones necessary for the creation of GPS artwork routes. Figure 4 illustrates a typical problem arising from using conventional routing algorithms for the creation of GPS artwork routes. To the best of our knowledge, existing software uses variations of three criteria for the optimization of the route: time, price, or distance. Depending on the underlying criteria, the routing reduces the details of the final GPS artwork shape. Time favors streets with higher speed limits than streets that are closer to the figure. Price avoids various kinds of charges and minimizes fuel costs not related to the target shape. The last widely used optimization, distance, minimizes the length of the calculated route. Depending on the figure, this reduction in length, in turn, reduces the details of the final GPS artwork.

As can be seen in Fig. 4, the approximation of the line produced by Google's mapping solution inadequately represents the desired input, whereas our solution delivers a much closer approximation. Curiously, Google's solution is particularly problematic in cities with a checkerboard-like road pattern. In these cities, one would expect a standard distance-based solution to work, but within this kind of street pattern, numerous shortest routes exist between the two points, due to the Manhattan-Distance being the same for multiple routes. In this specific case, the routing software prefers main streets with their higher speed limits and a minimum number of turns needed to reach the destination.

An obvious mitigation attempt is to insert more points along the outline of the shape and thereby force the system to more closely approximate a given shape. Starting from a coarse shape, more intermediate points do improve the route with shortest distance optimization, but with increasing density of these points, the result quickly degenerates (see Fig. 4).

Another problem with standard routing algorithms is the resolution of points that are not directly placed on a street. All solutions we analyzed snap these off-grid points onto streets that are predetermined for the area of the original point location. The process of repositioning off-grid points is a reasonable choice for the standard use of routing software, where users mostly desire to get to a point on the street corresponding to the selected location (e.g., entrances of parks, a large building, etc.). In GPS art, however, the goal is to approximate the overall shape of a figure and not to reach every control point. Consequently, missing a control point of the shape by some distance is preferable to a lengthy detour that would disarrange the appearance of the entire shape (see Fig. 3). This situation is particularly problematic with large natural *obstacles*, such as lakes, rivers, parks, or mountains. In such situations, standard mapping solutions can introduce almost arbitrary *detours*.

To generate pleasing and desirable GPS artwork, the optimization goals are significantly different, and, to the best of our knowledge, no available routing software offers specific optimization methods for this situation, even though there exist exciting applications of this technique for the broader public (see Fig. 2).

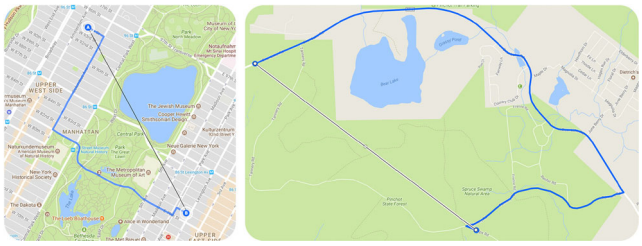

Fig. 3 Due to the underlying routing approach and targeting, standard routing approaches fail to locate control points that are off-grid. In this case, massive detours occur that are undesirable for GPS art.

Fig. 4 Approximating a single line shape with Google Maps. Using just the start and end points creates a route that significantly deviates from the diagonal line (a). Adding more points (b). With too many points, the line starts to incoprporate numerous U-turns to guarantee that each way-point is reached (c). Our solution is given in (d).

5 System overview

We present an automated system for generating routes for GPS artwork. The first input to our system is a figure drawn by the user as well as text, input via a keyboard. Our system supports basic transformations, such as scaling, rotation, and translation of the text and brush strokes made. The text is converted, using a simple font-rendering engine, to a list of line segments, such as brush strokes. Our implementation uses street data obtained from the Open Street Map (osm) project [16]. At runtime, we stream the osm data of the current area and extract the street information found within the data on the fly. For faster computation, we build a bidirectional graph of the street network, where each node represents a physical point on the street found in osm's XML data structure. These nodes cross between different streets or points between line segments to approximate curved streets. The restructuring of the data allows faster access to neighboring nodes. Thanks to our data streaming approach, users can choose any location on the globe to create GPS artwork without thinking about any data constraints. In the next step, our system computes the route via a divide and conquer approach, processing the line-list line by line. We use a single-source, multiple-target, shortest path algorithm similar to Dijkstra's algorithm [17] that minimizes a specifically tailored distance metric explained in Section 7. The system works at interactive rates and performs parameter changes, with immediate feedback. Once a satisfactory route has been computed, our system outputs the route as an overview image as well as a list of turn-by-turn navigation instructions.

6 Algorithm overview

Our approach can be summarized as follows. We start by selecting the first segment of our original figure and then the longitude and latitude of the starting position (see Fig. 5). Based on these coordinates, we determine the closest node of our graph as the starting position for our GPS path. Once the starting node has been determined, we compute the path from the start to the end of the segment by evaluating the cost function described in Section 7.

After approximating the end position of the first segment, the algorithm selects the next segment starting from the last end position. We reuse the resulting end node of the previous segment as the new starting position to guarantee a continuous path across all segments once the algorithm is finished.

Fig. 5 Left to right: placing a user-drawn figure on the map, selecting the first segment depending on the minimum segment length, and computing the path of the segment.

7 Cost function

7.1 Basics

Our approach is based on the concept of a single-source, multiple-target, shortest path algorithm similar to Dijkstra's algorithm. The result of this algorithm is the shortest path within a graph starting

from a given node. With Dijkstra's algorithm, each edge is weighted by a defined cost function, which results in a path with the lowest cost.

We defined a completely new cost function that depends on the figure drawn by the user. We incorporate several metrics within our function to compensate for different edge cases that occur due to the nature of modern streets.

The cost between two connected nodes P and N is defined by the function

$$C(\boldsymbol{P}, \boldsymbol{N}, \boldsymbol{S}, \boldsymbol{E}) = \alpha C_1(\boldsymbol{N}, \boldsymbol{E}) + \beta C_2(\boldsymbol{P}, \boldsymbol{N}) + \gamma C_3(\boldsymbol{P}, \boldsymbol{N}, \boldsymbol{S}, \boldsymbol{E})$$

where C_1, C_2, C_3 describe different conditions for the relationship between the street graph and the figure segment, which is defined by $\boldsymbol{S}$ and $\boldsymbol{E}$. The weights α, β, γ are used to define the influence of different metrics on the overall cost. The final cost of a single path that ends as close as possible to the segment's end position $\boldsymbol{E}$ is computed using Dijkstra's algorithm.

7.2 Metric: Distance minimization

Our first assumption was that to reach the end of the segment $\boldsymbol{E}$, we have only to pick the node $\boldsymbol{N}$ with the smallest distance to $\boldsymbol{E}$. By selecting such nodes, we can guarantee to at least minimize the distance to $\boldsymbol{E}$ and therefore move toward the end of the segment (see Fig. 6). We define function C_1 as

$$C_1(\boldsymbol{N}, \boldsymbol{E}) = |\boldsymbol{N} - \boldsymbol{E}|$$

This metric can also be found in various navigational systems and therefore underlies those drawbacks described in Section 4, but it is a good start for further investigation.

7.3 Metric: Path minimization

We found that by adding a path minimization metric, as used in Dijkstra's algorithm, we can increase the quality of our GPS artwork route (see Fig. 8(top)). We define C_2 as

$$C_2(\boldsymbol{P}, \boldsymbol{N}) = |\boldsymbol{P} - \boldsymbol{N}|$$

Using this metric, we prioritize nodes that are closer together, which counteracts the problems of C_1 to a large degree. However, this metric has no knowledge of the input shape and cannot be used on its own to compute any feasible route (see Fig. 7).

7.4 Metric: Distance sum minimzation

In several cases, a combination of metrics C_1 and C_2 led to a reasonable approximation of the original shape. However, this straightforward solution ignores the spatial distance between the route and the input shape. Figure 8 demonstrates such an issue, for which there is no incentive for a route to return to the original shape before the target point once it has deviated.

The underlying idea of this decider is to compute the distance between the figure segment and two neighboring nodes of the graph. We based our approach on the Riemann sum [18] and developed a function that computes the distance between two lines.

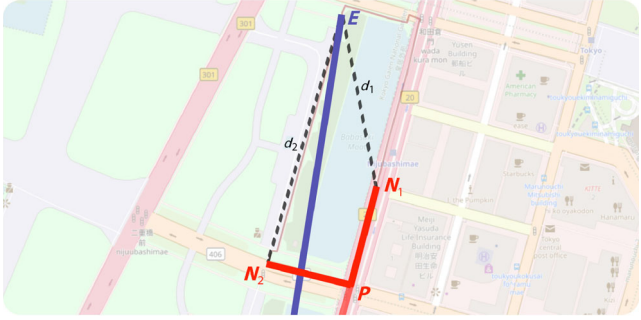

Fig. 6 A distance-based decision would favor node N_1, even though N_2 would be the better choice.

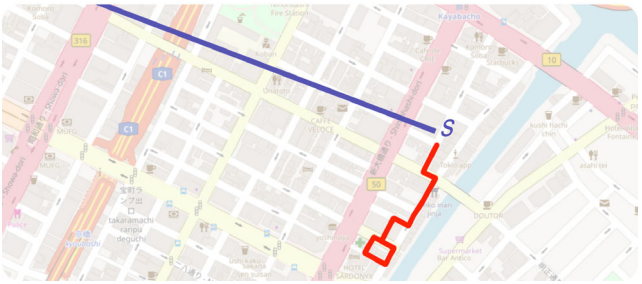

Fig. 7 Using travel minimization alone results in paths with no connection to the original segment, leading to never-ending loops.

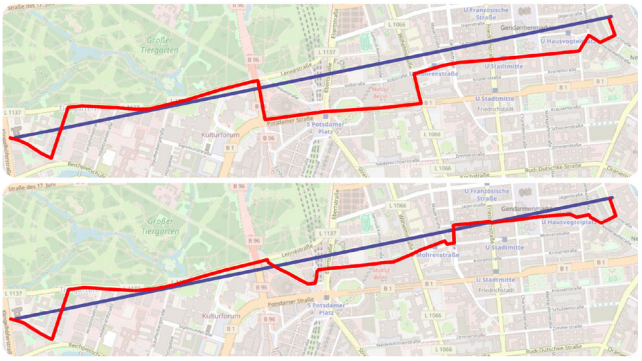

Fig. 8 Comparison of different cost functions. Top: using C_1 and C_2 to approximate the segment. Bottom: adding C_3 to the cost function.

For a given segment starting at a point $\boldsymbol{S}$ and ending at the point $\boldsymbol{E}$, we compute the distance sum for the point $\boldsymbol{P}$ and each neighbor $\boldsymbol{N}$ by evaluating the function:

$$C_3(\boldsymbol{P},\boldsymbol{N},\boldsymbol{S},\boldsymbol{E}) = \sum_{k=0}^{n} R(\boldsymbol{P},\boldsymbol{N},\boldsymbol{K})$$

where the function R describes the distance between the edge of the graph and an equally distanced point $\boldsymbol{K}$ on the segment. The point $\boldsymbol{K}$ is defined by

$$\boldsymbol{K} = \boldsymbol{S} + \frac{k}{n}|\boldsymbol{N}-\boldsymbol{S}|(\boldsymbol{N}-\boldsymbol{S})$$

and evaluation by

$$R(\boldsymbol{P},\boldsymbol{N},\boldsymbol{K}) = \begin{cases} |\boldsymbol{K}-\boldsymbol{P}|, & p<0 \\ |\boldsymbol{K}-(\boldsymbol{P}+p\boldsymbol{D})|, & 0\leqslant p\leqslant 1 \\ |\boldsymbol{K}-\boldsymbol{N}|, & p>1 \end{cases}$$

where p describes the projection of $\boldsymbol{K}$ onto the segment by computing the dot-product:

$$p = \boldsymbol{D}\cdot(\boldsymbol{K}-\boldsymbol{P})$$

with $\boldsymbol{D}$ being the directional vector of the edge starting from point $\boldsymbol{P}$:

$$\boldsymbol{D} = \boldsymbol{N}-\boldsymbol{P}$$

Based on the value p, we can decide whether point $\boldsymbol{K}$ is located on an orthogonal line between $\boldsymbol{P}$ and $\boldsymbol{E}$ or if it is found before or behind the edge. Using this information, we can evaluate the previously mentioned term and compute the distance factor of the point $\boldsymbol{K}$ and the edge of the graph.

The resulting sum (see Fig. 9) provides the best indication of the difference between the segment and the current edge and enhances the approximated path to a greater degree.

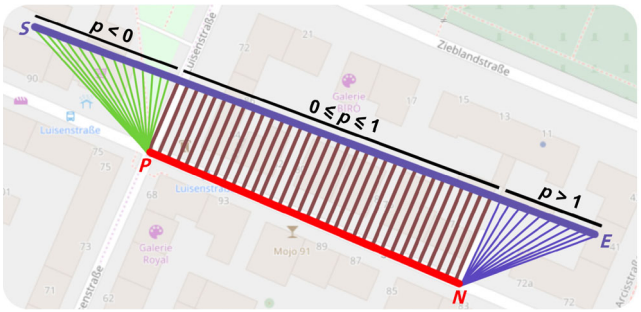

Fig. 9 Representation of C_3 for a given line segment between $\boldsymbol{S}$ and $\boldsymbol{E}$ and the current graph position $\boldsymbol{P}$.

8 Results and outlook

In this paper, we have presented a new approach to the creation of GPS artwork routes. We have demonstrated the drawbacks of widely used routing solutions and addressed their shortcomings by implementing a new way of routing optimization (see Fig. 10).

We reduce the time users have to spend planning an elaborate work of art to a minimum by offering easy-to-understand parameters. We strongly believe new interest in GPS art can be stimulated, and by integrating our approach into platforms that are already used for GPS art, a new generation of artists will be motivated.

We found our system to perform well in a broad range of figure and map setups. These setups ranged from simple shapes to lines of text with logos placed on many different maps. We tested our solution on the checkerboard pattern of modern cities and the *chaotic* street maps of historic cities, but there are obvious limits to our solution. In rural areas with very few streets, small complicated figures cannot be approximated reasonably without leaving the street network. Also, fine structures are generally hard to map to the street network (see Fig. 11).

In future, we will conduct a user study to further evaluate the quality of our algorithm in comparison to standard routing solutions. We are aiming for a better understanding of our metrics and how the resulting artworks, based on our costfunction, are

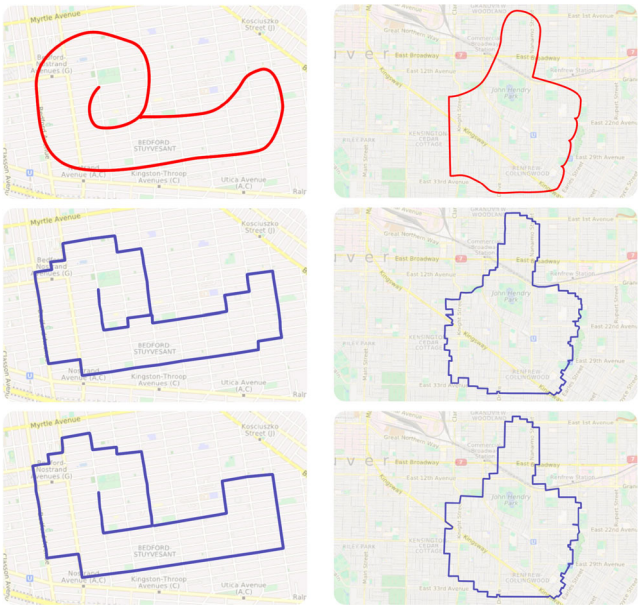

Fig. 10 Computing routes for a snail and a thumbs-up figure using our algorithm. Top: actual drawing of the figure. Middle: result using a heavily favored C_3. Bottom: mainly basing our cost function on C_1 and C_2.

Fig. 11 Limitations of automated GPS art route generation. Top: an overly detailed figure placed within a checkerboard pattern where details get lost. Bottom: placing figures within rural areas with insufficient streets.

received by users. Our current goal is a local and global optimization by implementing an automated approach for the placement, rotation, and scaling of the figure based on different conditions. We will also investigate to what degree our metric C_3 subsumes C_1 and C_2.

References

[1] UNESCO. Lines and geoglyphs of nasca and palpa. 1994. Available at https://whc.unesco.org/en/list/700/.

[2] Wood, J. Traverse me. 2010. Available at https://www.jeremywood.net/artworks/traverse_me.html.

[3] Strava. Strava. 2018. Available at https://www.strava.com.

[4] Lund, S. A creative spin: Pedaling my art. TEDx-Victoria, 2015. Available at https://www.youtube.com/watch?v=OsMMysaZRyg.

[5] Lund, S. GPS doodles. 2015. Available at https://gpsdoodles.com/.

[6] Takahashi, Y. Yassan's GPS drawing project. 2015. Available at http://gpsdrawing.info/.

[7] Balduz, P. Walk line drawing. 2017. Available at https://www.cg.tuwien.ac.at/research/publications/2017/Balduz_01/.

[8] Rosner, D. K.; Saegusa, H.; Friedland, J.; Chambliss, A. Walking by drawing. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 397–406, 2015.

[9] Kass, M.; Witkin, A.; Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision* Vol. 1, No. 4, 321–331, 1988.

[10] Lee, Y.; Lee, S. Geometric snakes for triangular meshes. *Computer Graphics Forum* Vol. 21, No. 3, 229–238, 2002.

[11] Lin, S. S.; Lin, C. H.; Hu, Y. J.; Lee, T. Y. Drawing road networks with mental maps. *IEEE Transactions on Visualization and Computer Graphics* Vol. 20, No. 9, 1241–1252, 2014.

[12] Yang, H.; Xie, K.; Huang, S. Q.; Huang, H. Uncut aerial video via a single sketch. *Computer Graphics Forum* Vol. 37, No. 7, 191–199, 2018.

[13] Roberts, M.; Hanrahan, P. Generating dynamically feasible trajectories for quadrotor cameras. *ACM Transactions on Graphics* Vol. 35, No. 4, Article No. 61, 2016.

[14] Google Inc. Google Maps. 2005.

[15] Microsoft. Bing maps. 2005. Available at https://www.bing.com/maps.

[16] OpenStreetMap Contributors. OpenStreetMap.2006. Available at https://www.openstreetmap.org.

[17] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik* Vol. 1, No. 1, 269–271, 1959.

[18] Reimann, B. XII-Ueber die Darstellbarkeit einer Function durch eine trigonometrische Reihe. In: *Bernard Riemann's Gesammelte mathematische Werke und wissenschaftlicher Nachlass.* Dedekind, R.; Weber, H. M. Eds. Cambridge: Cambridge University Press, 213–253, 2013.

Andre Waschk received his bachelor degree in applied computer science at the Universitt Duisburg-Essen in 2013 followed by a master degree in the same program in 2016.

Jens Krüger studied computer science at the Rheinisch-Westfälische Technische Hochschule Aachen where he received his diploma in 2002. In 2006 he finished his Ph.D. at the Technische Universitt Mnchen and after postdoctoral positions in Munich, and at the Scientific Computing and Imaging (SCI) Institute he became research assistant professor at the University of Utah. In 2009 he joined the Cluster of Excellence Multimodal Computing and Interaction at Saarland University to head the Interactive Visualization and Data Analysis Group. Since 2013, he has been chair of the High Performance Computing Group at the University of Duisburg-Essen. In addition, he also holds an adjunct professor title of the University of

Utah and is a principal investigator of multiple projects in the Intel Visual Computing Institute.

DOI: 10.1007/s41095-019-0146-z
URN: urn:nbn:de:hbz:464-20200812-110457-7