A Heuristic Homotopic Path Simplification Algorithm

Shervin Daneshpajouh[1] and Mohammad Ghodsi[1,2,⋆]

[1] Department of Computer Engineering,
Sharif University of Technology,
Tehran, Iran
[2] School of Computer Science,
Institute for Research in Fundamental Sciences (IPM),
Tehran, Iran
daneshpajouh@ce.sharif.edu, ghodsi@sharif.edu

Abstract. We study the well-known problem of approximating a polygonal path P by a coarse one, whose vertices are a subset of the vertices of P. In this problem, for a given error, the goal is to find a path with the minimum number of vertices while preserving the homotopy in presence of a given set of extra points in the plane. We present a heuristic method for homotopy-preserving simplification under any desired measure for general paths. Our algorithm for finding homotopic shortcuts runs in $O(m \log(n + m) + n \log n \log(nm) + k)$ time, where k is the number of homotopic shortcuts. Using this method, we obtain an $O(n^2 + m \log(n + m) + n \log n \log(nm))$ time algorithm for simplification under the Hausdorff measure.

Keywords: Computational Geometry, Simplification, Homotopy, Path, Line, Curve, Heuristic.

1 Introduction

Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ be the points of a given polygonal path, where n is the size of P. We assume that the input path is not self-intersecting, which is a common assumption[1,2,3]. A polygonal path $Q = \{q_1 = p_0, q_2, \ldots, q_b = p_{n-1}\}$ with $b < n$ is an approximation of P. In the *restricted* version of the path simplification problem, the vertices of Q should be a subsequence of the vertices of P(Fig.1). We call a segment $p_i p_j$ with $i < j$ a *link* or *shortcut*. Let $P(p, q)$ denote the sub-path of P from point p to point q. For two vertices p_i, p_j, we use $P(i, j)$ as a shorthand for $P(p_i, p_j)$.

Let S be a set of points $s_0, s_1, \ldots, s_{m-1}$ in the plane that do not lie on the path P. We say that Q preserves the homotopy if it is deformable to P without passing over any point of S. In Fig.1, there are some points between the simplified path Q and the original path P. Therefore, Q can not be deformed to P without

⋆ This work has been partially supported by IPM under contract #CS1389-2-01.

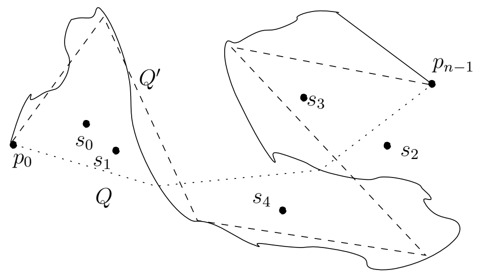

Fig. 1. The simplified path Q with fewer number of vertices does not preserve the homotopy. The simplification Q' preserves the homotopy and is a valid simplification.

passing over those points and consequently does not preserve the homotopy. In this figure, the simplified path Q' is a valid simplification.

The error of a simplification Q under an error function α is represented by $error_\alpha(Q)$. This simplification error is defined to be $\max_{i=0}^{b-2} error_\alpha(q_i q_{i+1})$, where $error_\alpha(q_i q_{i+1})$ is the error of $q_i q_{i+1}$ under the error function α.

There are two optimization goals for this problem: (1) min-b, where for a given error threshold ϵ, the goal is to find a simplification with the minimum number of vertices for which the error is at most ϵ, and (2) min-ϵ, where for a given number b, the goal is to find a simplification of at most b vertices that has the minimum simplification error. Having the solution of the min-b problem, we can solve the min-ϵ problem using a binary search. In this paper, we consider the min-b version of the problem.

1.1 Motivation, Previous Results and Our Result

Line simplification, also known as path, curve and chain simplification in the literature, is a fundamental problem in various disciplines and has been studied in computational geometry [4,5,6,7] , geographic information systems (GIS) [8,9,10] and digital image analysis [11,12,13]. In many applications of these disciplines, processing and presentation of data is very time consuming. Therefore, it is necessary to compress the very large input data. Map information, like polygonal subdivisions and contours are examples of such large data that needs simplification. In these applications, map information and features such as country borders, sea borders and cities are represented as a set of polygonal lines and vertices. Using simplification, we can reduce the total amount of input data and consequently reduce computation time. In these applications and many others, e.g. *river routing* in circuit board design, homotopy preservation is an important requirement. Homotopy preservation makes sure that, after the simplification process, cities or areas on both sides of the input path stay at the same side of the simplified line as of the original one.

There are many results on line simplification under different error criteria, though most of them do not generate homotopic results. Guibas *et al.* [14] proved that, for some error function, the problem of minimum-link approximation of a given simple-polygon for which the output is non-self-intersecting and the problem of homotopy-preserving simplification of a given subdivision, are NP-Hard. Estkowski and Mitchell [15] show that the general problem of homotopy-preserving subdivision simplification is MIN PB-complete and presented some heuristic approaches to handle it.

The first algorithm for the problem of minimum link homotopy-preserving simplification was presented by De Berg *et al.* [1,2]. They studied the min-b version of the problem under the Hausdorff measure and presented an $O(n(n+m)\log n)$ time algorithm. Their algorithm preserves the homotopy and finds the minimum number of links for x-monotone paths. They generalized their method to handle general polygonal paths and presented a heuristic method which does not always guarantee to find the minimum link simplification. Daneshpajouh *et al.* [16] improved the running time on x-monotone paths and presented an optimal $T_F(n) + O(m\log(nm) + n\log n\log(nm) + k)$ time algorithm, where k and T_F are the number of homotopic shortcuts and the complexity of the computation of the error measure under the error function F respectively. For the general path they presented an optimal algorithm that finds strongly homotopic paths in $T_F(n) + O(n(m+n)\log(nm))$. A path is called strongly homotopic if every edge of it be homotopic. It can be shown that their algorithm for general paths does not always find the optimal homotopic path. Note that, there may be some non-homotopic shortcuts that together make a homotopic paths.

In this paper, we present a heuristic algorithm for the minimum-link homotopy-preserving simplification problem. First, we present a new method for finding homotopic shortcuts in $O(m\log(n+m) + n\log n\log(nm) + k)$ time, where k is the number of homotopic shortcuts. Then, we compute the min-link simplification under the desired measure. Using our result, we obtain an $O(n^2 + m\log(n+m) + n\log n\log(nm))$ time algorithm for the problem under the Hausdorff measure. Our method guarantees the result to be homotopic to the input path. Although, our algorithm, like De Berg *et al.* methods, does not always guarantee to find the minimum number of links. The results presented here improve the running time of the previous methods and for general paths by a factor $O(\log n)$.

The remainder of this paper is organized as follows. In Section 2, we present our algorithm for finding homotopic shortcuts. In Section 3, we show how our algorithm can be used for solving the min-link simplification problem under the Hausdorff measure. In Section 4, we offer the conclusion.

2 Homotopic Shortcut Identification Algorithm

In this section, we present our algorithm for identifying homotopic shortcuts. Let P be the input path and S a set of extra points in the plan. Our algorithm builds a graph G_S containing possible shortcuts that can be in the final solution regardless of the error function. Note that the graph G_S can have at most $n(n-1)/2$ edges, where n is the size of P.

The algorithm has two phases. In the first phase we do a preprocessing on the input path P and the set S and build a simple polygon $\Psi(P,S)$. We call $\Psi(P,S)$

the permitted region. In the second phase, having the permitted region, we find the homotopic shortcuts, and build G_S.

2.1 Preprocessing Phase

The preprocessing phase consists of the following operations:

– Dividing the convex hull of P into two polygons L and R.
– Breaking L and R into simple polygons Δ_{ij}.
– Computing the *relative convex hull* for the extra points inside Δ_{ij} and some points added by our algorithm.
– Building $\Psi(P, S)$.

In the following, we describe each step in detail.

We know that in the restricted version of path simplification, the simplified path Q should use a subset of the vertices of P. Therefore, all possible shortcuts of $q_iq_j, 0 \leq i < j \leq n-1$, lie inside the convex hull of P. From now on, we refer to the convex hull of P as $CH(P)$. Before starting the first step, we omit all points in S that do not lie inside $CH(P)$.

The convex hull of a set of points is represented by an ordered set of points. First, we assume that p_0 and p_{n-1} are in $CH(P)$. Later, we show how we handle degenerate cases in general paths in which one or both of these points are not in $CH(P)$. The polygonal path P divides polygon $CH(P)$ into two polygons L and R. Similarly, $CH(P)$ is split, at p_0 and p_{n-1} into two chains $CH(P)_l$ and $CH(P)_r$. The computation we do here on polygon L is analogous for polygon R. So, we only describe the computation on polygon L. As we need to compute the convex hull of P, the computation of this first step takes $O(n \log n)$ time.

Obviously, some points of $CH(P)$ are points of P too. Therefore, L is not necessarily simple. In the second step of the algorithm, we divide polygon L into some simple linear-size polygons Δ_{ij}. For each edge of $CH(P)_l$ there is a corresponding shortcut p_ip_j. We build Δ_{ij} by combining p_ip_j and $P(i, j)$. The identification and construction of all Δ_{ij}s can be done in linear time.

In the third step, we first distribute the points in S among Δ_{ij} by preprocessing Δ_{ij} for point location. We do this in $O((n+m) \log n)$ time[17]. Let the subset of points in S that fall in polygon Δ_{ij} be denoted by S_{ij}. Now, we compute the *relative convex hull* of polygon Δ_{ij} and the set of points $S^*_{ij} = S_{ij} \cup \{p_i, p_j\}$. The *relative convex hull*, also known as the geodesic convex hull, of a simple polygon X and a set of points S inside X is the shortest cycle Y within polygon X that surrounds the points of S. From now on we call the relative convex hull of a simple polygon X and a set of points S, $RCH(X, S)$. We use the method presented by Toussaint [18] to compute $RCH(X, S)$. This method works on simple polygons. As Δ_{ij} is a simple polygon we can apply this method. If S_{ij} is empty then we define the output of $RCH(\Delta_{ij}, S^*_{ij})$ to be the shortcut p_ip_j. The computation of $RCH(\Delta_{ij}, S^*_{ij}\})$, for all polygons Δ_{ij} and the set of points S, can be done in $O((n + m) \log(n + m))$ time.

In the fourth step, we build $\Psi(P, S)$. Let ω be an ordered set of points, *i.e.* $\omega = \{p_i, p_{i+1}, \ldots, p_{j-1}, p_j\}$. We define ω^R to be the reverse set of points of ω,

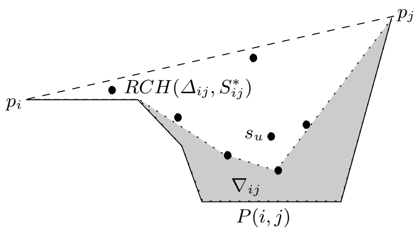

Fig. 2. The polygon Δ_{ij} consists of the edges of sub-path $P(i,j)$ and $p_ip_j \in CH(P)_l$. The relative convex hull of S^*_{ij} inside Δ_{ij} is the white area between the dotted and dashed lines, and ∇_{ij} is shown in gray.

i.e. $\omega^R = \{p_j, p_{j-1}, \ldots, p_{i+1}, p_i\}$. First, we build $\Psi(P,S)_l$. For each Δ_{ij} and $RCH(\Delta_{ij}, S^*_{ij})$ we create a polygon ∇_{ij} (see Fig.2):

$$\omega = RCH(\Delta_{ij}, S^*_{i,j}) \backslash \{p_i, p_j\}$$

$$\nabla_{ij} = P(i,j) \cup \omega^R$$

Then, we take the union of all the ∇_{ij}s and create $\Psi(P,S)_l$. We run these steps on $CH(P)_r$ too and merge the two resulting polygons $\Psi(P,S)_l$ and $\Psi(P,S)_r$ and build the simple polygon $\Psi(P,S)$. This step can be done in $O(n+m)$ time.

Now, we return to our assumption that the starting and ending points of P lie on $CH(P)$. In some degenerate cases, the starting or ending points of P may not lie on $CH(P)$. See Fig.4 where the start of P has a spiral shape. For such a case, let p_i be the first points on $CH(P)$ in the sequence of points after p_0 and let Δ_{ij} be the polygon that contains p_0. Then, we let

$$\omega = RCH(\Delta_{ij}, \{S^*_{ij} \cup \{p_0, p_1, ...p_{i-1}\}\}) \backslash \{p_i, p_j\}$$

$$\nabla_{ij} = P(i,j) \cup \omega^R$$

In this way, we remove a subset of path P that lies inside Δ_{ij}. We run the next steps of the algorithm as described before and compute $\Psi(P,S)$. Note that after running the whole algorithm and finding the minimum link path Q, we have to add the sequence of $\{p_0, p_1, ...p_{i-1}\}$ to Q. A similar approach can be applied at the end of the path if it has a spiral shape.

It is easy to see that the following lemma is correct.

Lemma 1. *For a given polygonal path P and a set of points S, the simple polygon $\Psi(P,S)$ which is build using the above method, does not contain any extra point $s \in S$.*

Now, we prove the following lemma.

Lemma 2. *For a given polygonal path P and a set of extra points S, all the shortcuts p_ip_j of P that lie inside the polygon $\Psi(P,S)$ are homotopic.*

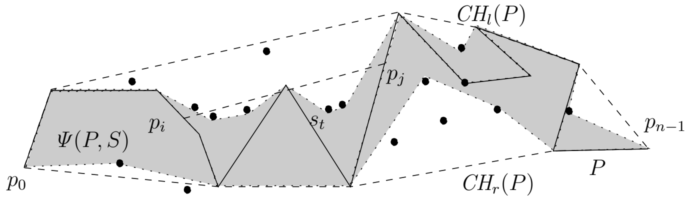

Fig. 3. A polygonal path P and a set of extra points S. Polygon $\Psi(P,S)$ is shown in gray. The shortcut $\overrightarrow{p_ip_j}$, (dashed segment) inside $CH(P)$ that intersects border of $\Psi(P,S)$, can not be a homotopic shortcut, because it can not be deformed to $P(i,j)$ without passing over point $s_t \in S$.

Proof. From Lemma 1, we know that every point $s_i \in S$ lies outside of $\Psi(P,S)$. Consequently, every shortcut p_ip_j that lies inside $\Psi(P,S)$ can easily be deformed to $P(i,j)$ without passing over any $s_i \in S$. Therefore, all the shortcuts p_ip_j of P that lie inside the polygon $\Psi(P,S)$ are homotopic.

We observe that many shortcuts that do not lie inside $\Psi(P,S)$ are not homotopic (See Fig.3). But, there may exist some shortcuts which can be homotopic and do not lie inside $\Psi(P,S)$. Therefore, we have the following lemma.

Lemma 3. *There exist some homotopic shortcuts $p_ip_j, 0 \leq i < j \leq n-1$ of a path P that intersects some $RCH(\Delta_{cd}, S^*_{cd})$ and are homotopic to $P(i,j)$.*

2.2 Finding Eligible Shortcuts

Having $\Psi(P,S)$ we identify all eligible shortcuts $p_ip_j, 0 \leq i < j \leq n-1$. We call a shortcut p_ip_j eligible if it lies inside $\Psi(P,S)$. In other words, if p_ip_j intersects any edge of polygon $\Psi(P,S)$ in any point rather than p_i and p_j, then it is not eligible.

To solve this problem, we can check all the intersection points of all possible shortcuts using naïve algorithms in $O(n^2(n+m))$. To solve it efficiently, we look at it as a visibility problem. We say that if a point p_j is visible from p_i inside $\Psi(P,S)$ then p_ip_j is an eligible shortcut. The problem of visibility of a set of points inside a polygon has been extensively studied. The best previous result was presented by Ben-Moshe *et al.* [19]. Their algorithm takes a polygon and a set of points inside the polygon as input, and returns the list of visible pairs in $O(x + y\log y \log xy + k)$-time using $O(x+y+k)$ space where x, y and k are the number of vertices of the polygon, the number of points inside the polygon and the number of visible pairs respectively.

The only thing that needs to be considered is the complexity of the algorithm with respect to the conditions of our problem. The number of points in $\Psi(P,S)$ can be $O(n+m)$ in the worst case, where n is the number of points in the input path P and m is the number of extra points. By applying these parameters in the algorithm of Ben-Moshe *et al.* , we achieve $O(m + n\log n \log(nm) + k)$ time

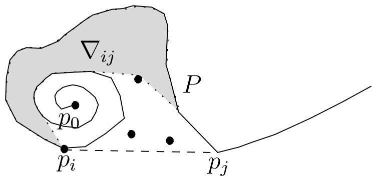

Fig. 4. The point p_0 does not lie on *CH*(P). Therefore, when computing the relative convex hull inside $\varDelta_{ij}$ we consider $P(0, i)$ too and compute *RCH*($\varDelta_{ij}, S^*_{ij} \cup \{p_0, p_1, ...p_{i-1}\}$). The gray area shows ∇_{ij}.

complexity. Note that in worst case, k can be $O(n^2)$, *e.g.* , when there is no extra point S inside *CH*(P). Therefore, we expect a much smaller k for a realistic input data and a sub-quadratic time in real applications.

Using the output of this algorithm, we build the graph G_S. The vertices of G_S are vertices of P and the edges of G_S are the k eligible shortcuts identified by our presented method. Hence, we can conclude all this in the following theorem:

Theorem 1. *Given a general polygonal path P with n vertices and a set S of m points, it is possible to compute a graph G_S containing a set of k homotopic shortcuts in $O(m \log(n + m) + n \log n \log(nm) + k)$ time.*

3 Homotopic Simplification under the Hausdorff Measure

In this section we show how a homotopic simplification under the Hausdorff measure can be computed using the result of our algorithm.

Chan and Chin [21] presented a method for optimal min-b simplification of a polygonal chain under the Hausdorff error measure, $error_H$, in $O(n^2)$ time [21]. The method builds two graphs G_1 and G_2. G_1 contains shortcut p_ip_j if the $error_H(p_ip_j)$ is less than ϵ and G_2 contains shortcut p_ip_j if the $error_H(p_jp_i)$ is less than ϵ. Then, the algorithm intersects these two graphs and creates a new graph G_3. The shortest polygonal path can be found by searching the shortest path in this graph[22]. This method does not preserve the homotopy of the path.

Here, using the method of Chan and Chin, we build G_3 containing all edges with $error_H < \epsilon$. Independently, we create graph G_S using the algorithm from Theorem 1. Finally, we intersect G_3 and G_S and obtain graph G_ϵ. Finding the shortest path in the unweighted graph G_ϵ gives us the homotopic simplified path under Hausdorff measure. We can conclude with the following theorem.

Theorem 2. *Given a general polygonal path P with n vertices, a set S of m points, and an error tolerance $\epsilon > 0$, it is possible to compute a homotopic simplification of P that approximates P within the error tolerance ϵ in $O(n^2 + m \log(n + m) + n \log n \log(nm))$ time.*

4 Conclusion

We have presented a heuristic method for maintaining homotopy in the simplification of a given general path. The given algorithm computes the graph G_S which contains the homotopic shortcuts, in $O(m \log(n+m)+n \log n \log(nm)+k)$ time. Our method always guarantees the simplified path to be homotopic to the original one.

Using the proposed algorithm, we studied the problem under the Hausdorff measure and improved the previous best-known result by the factor $O(\log n)$. The method presented here is quite general and can be directly applied to other line simplification measures (like Fréchet, Area and Angle) and other related problems. It remains open whether there is a quadratic or near-quadratic time algorithm for finding optimal homotopic simplification for general paths.

References

1. de Berg, M., van Kreveld, M., Schirra, S.: A New Approach to Subdivision Simplification. In: Twelfth International Symposium on Computer Assisted Cartography, vol. 04, pp. 79–88 (1995)
2. de Berg, M., van Kreveld, M., Schirra, S.: Topologically correct subdivision simplification using the bandwidth criterion. Cartography and GIS 25, 243–257 (1998)
3. Buzer, L.: Optimal Simplification of Polygonal Chain for Rendering. In: 23rd ACM Symposium on Computational Geometry (SoCG), pp. 168–174 (2007)
4. Goodrich, M.T.: Efficient piecewise-linear function approximation using the uniform metric. Discrete Computational Geometry 14, 445–462 (1995)
5. Agarwal, P.K., Harpeled, S., Mustafa, N.H., Wang, Y.: Near-linear time approximation algorithms for curve simplification. Algorithmica 42, 203–219 (2005)
6. Aronov, B., Asano, T., Katoh, N., Mehlhorn, K., Tokuyama, T.: Polyline fitting of planar points under min-sum criteria. International Journal of Computational Geometry and Applications 16, 97–116 (2006)
7. Abam, M.A., de Berg, M., Hachenberger, P., Zarei, A.: Streaming algorithms for line simplifications. In: Proc. ACM Symposium on Computational Geometry (SoCG), pp. 175–183 (2007)
8. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Canadian Cartographer 10(2), 112–122 (1973)
9. Hershberger, J., Snoeyink, J.: Speeding up the Douglas-Peucker line simplification algorithm. In: Proceeding of 5th International Symposium on Spatial Data Handling, pp. 134–143 (1992)
10. Li, Z., Openshaw, S.: Algorithms for automated line generalization based on a natural principle of objective generalization. International Journal of Geographic Information Systems 6, 373–389 (1992)
11. Kurozumi, Y., Davis, W.A.: Polygonal approximation by the minimax method. Comput. Graph. Image Process 19, 248–264 (1982)
12. Hobby, J.D.: Polygonal approximations that minimize the number of inflections. In: Proceeding of the 4th ACM-SIAM Symposium on Discrete Algorithms, pp. 93–102 (1993)

13. Asano, T., Katoh, N.: Number theory helps line detection in digital images. In: In Proceeding of 4th Annual International Symposium on Algorithms and Computing, vol. 762, pp. 313–322 (1993)
14. Guibas, L.J., Hershberger, J.E., Mitchell, J.S.B., Snoeyink, J.S.: Approximating polygons and subdivisions with minimum link paths. International Journal of Computational Geometry and Applications 3(4), 383–415 (1993)
15. Estkowski, R., Mitchell, J.S.: Simplifying a polygonal subdivision while keeping it simple. In: Proceedings of the 17th Annual Symposium on Computational Geometry, pp. 40–49 (2001)
16. Daneshpajouh, S., Abam, M.A., Deleuran, L., Ghodsi, M.: Computing Strongly Homotopic Line Simplification in the Plane. In: European Workshop on Computational Geometry (2011)
17. Preparata, F.P., Shamos, M.I.: Computational Geometry - an introduction. Springer, New York (1985)
18. Toussaint, G.T.: An optimal algorithm for computing the convex hull of a set of points in a polygon. In: Proceeding of Signal Processing III: Theories and Applications, EURASIP 1986, Part 2, pp. 853–856 (1986)
19. Ben-Moshe, B., Hall-Holt, O., Katz, M., Mitchell, J.: Computing the visibility graph of points within a polygon. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 27–35 (2004)
20. Guibas, L.J., Hershberger, J.: Optimal shortest path queries in a simple polygon. Journal of Comput. Syst. Sci. 39, 126–152 (1989)
21. Chan, W.S., Chin, F.: Approximation of polygonal curves with minimum number of line segments. In: Proceeding of 3rd Annual International Symposium on Algorithms and Computing, vol. 650, pp. 378–387 (1992)
22. Imai, H., Iri, M.: Polygonal approximations of a curve-formulations and algorithms. In: Toussaint, G.T. (ed.) Computational Morphology, pp. 71–86 (1988)