Decomposing a Simple Polygon into Trapezoids

Fajie Li and Reinhard Klette

Computer Science Department, The University of Auckland
Auckland, New Zealand

Abstract. Chazelle's triangulation [1] forms today the common basis for linear-time Euclidean shortest path (ESP) calculations (where start and end point are given within a simple polygon). This paper provides an alternative method for subdividing a simple polygon into "basic shapes", using trapezoids instead of triangles. The authors consider the presented method as being substantially simpler than the linear-time triangulation method. However, it requires a sorting step (of a subset of vertices of the given simple polygon); all the other subprocesses are linear time.

Keywords: *computational geometry, simple polygon, Euclidean shortest path, rubberband algorithm*

1 Introduction

Simple polygons form a class of very complex 2D shapes. For example, consider a "fractal tree" where all line segments are expanded into "thin rectangles", forming a simple polygon this way. This paper proposes a decomposition of any simple polygon into trapezoids. Subsequent algorithms, such as calculating a Euclidean shortest path (ESP), or further processing for pattern recognition or shape analysis purposes, may have benefit from this. In [2] the authors claimed to have an $\mathcal{O}(n \log n)$ rubberband algorithm for calculating ESPs in simple polygons. Actually, this needs to be corrected: the algorithm given in [2] has worst-case complexity $\mathcal{O}(n^2)$. However, using the trapzoid decomposing as given in this paper it is possible to calculate step sets more efficiently [which allows to modify the algorithm given in [2] into one of $\mathcal{O}(n \log n)$ time complexity].

Section 2 provides necessary definitions and theorems. Section 3 presents the trapzoid decomposing algorithm and examples and a time analysis. Section 4 concludes the paper.

2 Basics

Let $\Pi^\bullet$ be the (topological) closure of a simple polygon Π, $\partial\Pi$ be its frontier, $V(\Pi)$ the set of vertices of Π, and $E(\Pi)$ the set of edges of Π. For $v \in \partial\Pi$, v_x and v_y are the x- and y-coordinates of v, respectively. Let v_1, v_2, $v_3 \in \partial\Pi$. Vertices are ordered either by a clockwise or counter-clockwise scan through $\partial\Pi$. Let $\rho(v_1, \Pi, v_2)$ be the polygonal path from v_1 to v_2, contained in $\partial\Pi$. Let

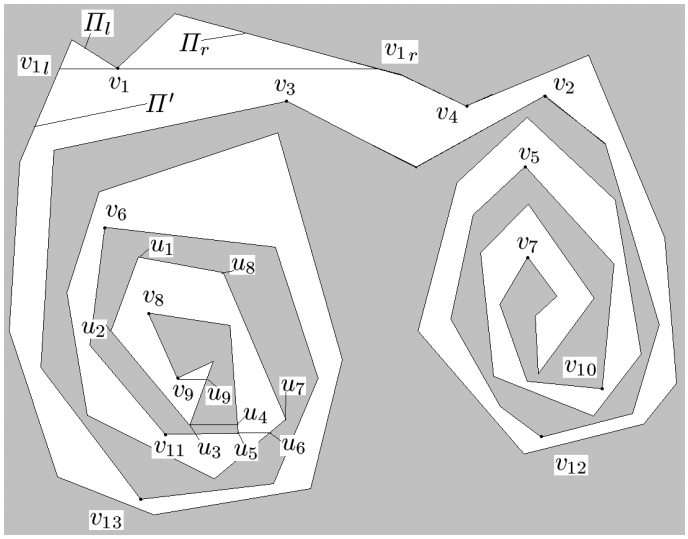

Fig. 1. Illustration of the given definitions.

$\rho(v_1, \Pi, v_2, \Pi, v_3)$ be the polygonal path from v_1 to v_2 and then to v_3 around $\partial\Pi$.

Let u, v, $w \in \partial\Pi$ such that $u_y = v_y = w_y$; consider $\rho(u, \Pi, v, \Pi, w) \subset \partial\Pi$. Let Π' be a simple polygon obtained by adding a 'closing edge' uw to $\rho(u, \Pi, v, \Pi, w)$, denoted by $\Pi' = uw + \rho(u, \Pi, v, \Pi, w)$. Π' is called an *up-(down-) polygon* with respect to v if, for each $p \in \Pi'^{\bullet}$, $p_y \geq (\leq)\ v_y$.

For example, in Figure 1, Π_l and Π_r are up-polygons with respect to v_1. Π' is a down-polygon with respect to v_1.

Let Π, Π' be two simple polygons with $\Pi'^{\bullet} \subset \Pi^{\bullet}$; let $\{v_0, v_1, v_2, \ldots, v_{n-1}\}$ and $\{v'_0, v'_1, v'_2, \ldots, v'_{n-1}\}$ be such orders of the vertices of Π and Π', respectively, that the following becomes true: For a sufficiently small $\varepsilon > 0$, we have $d_e(v'_0, v_0) = \varepsilon$ for the Euclidean distance between v'_0 and v_0, edge $v'_i v'_{i+1}$ is parallel to edge $v_i v_{i+1}$, and $v'_i v_i$ bisects the angle $\angle v_{i-1} v_i v_{i+1}$, where $i = 0, 1, 2, \ldots, n$, and the addition or subtraction of indices is taken *mod* n. In this case, Π' is called an *inner simple polygon of* Π, denoted by $\Pi(v_0, \epsilon)$. – In Figure 2, $u_0u_1u_2u_3u_4u_5u_0$ is an inner polygon of $v_0v_1v_2v_3v_4v_5v_0$.

Let v_{i-1}, v_i, v_{i+1} and v_{i+2} be four consecutive vertices of Π. If $(v_{i-1})_y < (v_i)_y$, $(v_i)_y = (v_{i+1})_y$ and $(v_i)_y > (v_{i+2})_y$ [or $(v_{i-1})_y > (v_i)_y$, $(v_i)_y = (v_{i+1})_y$ and $(v_i)_y < (v_{i+2})_y$], and the point $(0.5 \cdot (v_{ix} + v_{i+1_x}), v_{iy} + \varepsilon)$ [or $(0.5 \cdot (v_{ix} + v_{i+1_x}), v_{iy} - \varepsilon)$] is inside Π, then $v_i v_{i+1}$ is called an *up- (down-) stable edge* of Π with respect to the xy-coordinate system used for representing Π.

If $(v_{i-1})_y < (v_i)_y$, $(v_i)_y = (v_{i+1})_y$ and $(v_i)_y > (v_{i+2})_y$, and the point$(0.5 \cdot (v_{ix} + v_{i+1_x}), v_{iy} - \varepsilon)$ is in Π, then $v_i v_{i+1}$ is called a *maximal edge* of Π with respect to the chosen xy-coordinate system.

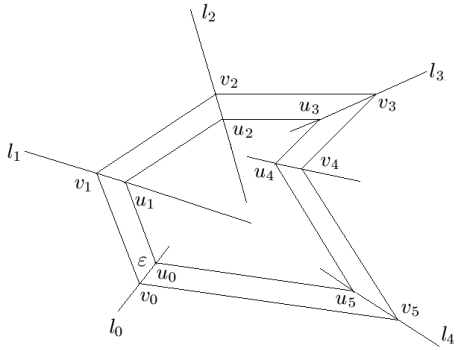

Fig. 2. Example of an inner polygon.

Let v_{i-1}, v_i and v_{i+1} be three consecutive vertices of Π. If $(v_{i-1})_y < (v_i)_y$, $(v_i)_y > (v_{i+1})_y$ [or $(v_{i-1})_y > (v_i)_y$, $(v_i)_y < (v_{i+1})_y$], and there exist points $u_i \in V(\Pi(v_0, \varepsilon))$ [this is the set of vertices of $\Pi(v_0, \varepsilon)$] such that $\triangle v_{i-1}v_iv_{i+1}$ does not (!) contain u_i, then v_i is called an *up- (down-) stable point* of Π with respect to the xy-coordinate system used for representing Π.

In Figure 1, v_2, v_3, v_5, v_6, v_7 and v_8 are up-stable points. v_1, v_4, v_9, v_{10}, v_{11}, v_{12} and v_{13} are down-stable points.

If $(v_{i-1})_y < (v_i)_y$, $(v_i)_y > (v_{i+1})_y$ [or $(v_{i-1})_y > (v_i)_y$, $(v_i)_y < (v_{i+1})_y$] and there exist points $u_i \in V(\Pi(v_0, \varepsilon))$ [this is the set of vertices of $\Pi(v_0, \varepsilon)$] such that $\triangle v_{i-1}v_iv_{i+1}$ does (!) contain u_i, then v_i is called an *maximal (minimal) point* of Π with respect to the chosen xy-coordinate system.

In Figure 1, u_1 is a maximal point and u_3 is a minimal point. – As common, a simple polygon is called *monotonous* if it has both a unique up- and down-stable point or edge. In Figure 1, $u_1u_2u_3u_4u_5u_6u_7u_8u_1$ is a monotonous simple polygon.

Since the discussion of our algorithm in the case of an up- or down-stable edge is analogous to that of an up- or down-stable point, we will just detail the case of up- or down-stable points.

Let v be an up-stable point of Π. Let S_v be the set of minimal points u of Π such that there exists a polygonal path from v to u around $\partial\Pi$ and $u_y < v_y$. Let $u' \in S_v$ such that $u'_y = \max\{u_y : u \in S_v\}$. If there exists a point $w \in \partial\Pi$ such that the segment $u'w \subset \Pi^\bullet$ and $w_y = u'_y$, then $u'w$ is called a *cut edge* of Π. The polygonal path $\rho(v, \Pi, u')$ is called a *decreasing polygonal path from v to $u'w$*. v is called an up-stable point *with respect to* the cut edge $u'w$, and $u'w$ is called a cut edge *with respect to* the up-stable point v. u' is called a *closest* minimal point with respect to v around the frontier of Π.

In Figure 1, $v_8v_9u_9u_3$ is a decreasing polygonal path from v_8 to u_3u_4, and u_3 is a closest minimal point with respect to v_8.

If $u, w \in \partial\Pi$ such that $u_y = v_y = w_y$, $u_x < v_x < w_x$, and $uv, vw \in \Pi^\bullet$, then u and w are called the *left and right intersection points* of v, respectively. – In Figure 1, v_{1l} and v_{1r} are the left and right intersection points of v_1, respectively.

Let v be a maximal point of Π. Let S_v be the set of down-stable points u of Π such that there exists a polygonal path from v to u around $\partial\Pi$ and $u_y < v_y$. Let $u' \in S_v$ such that $u'_y = \max\{u_y : u \in S_v\}$. u' is called a *closest* down-stable point with respect to v around the frontier of Π.

In Figure 1, v_9 is a closest down-stable point with respect to the maximal point above the segment v_9u_9.

3 The Algorithm

This section describes the decomposition algorithm and its three subroutines, which are Procedures 1–3. Procedure 1 is used to update the left or right intersection points of up-stable points. It will be applied by Procedure 2 to remove up-stable points. Procedure 3 applies Procedures 1 and 2 to compute the left and right intersection points of all up-stable points, from bottom to top. This procedure will be called in Step 3 of the main algorithm which processes both up- and down-stable points, from top to bottom.

In Procedure 1, let Π be a simple polygon such that $V(\Pi)$ does not have any up-stable point.

Procedure 1: update left or right intersection points

1. Decompose Π into a set of trapezoids, denoted by T_Π (note: straightforward, because there is no up-stable point).

2. For each edge $e \in E(\Pi)$.

2.1. If e is a cut edge, then do the following:

2.1.1. Let v_e be the up-stable point with respect to e.

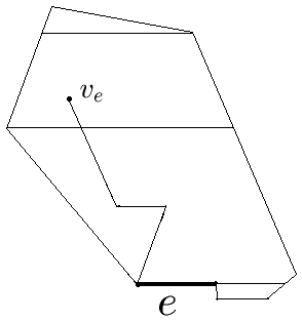

Fig. 3. Example for Step 2.1.1 of Procedure 1.

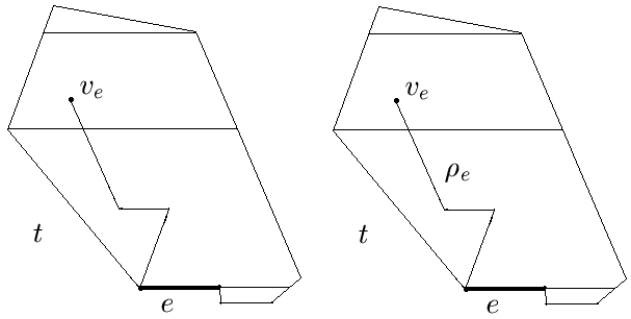

Fig. 4. Example for (left) Step 2.1.2 and (right) Step 2.1.3 of Procedure 1.

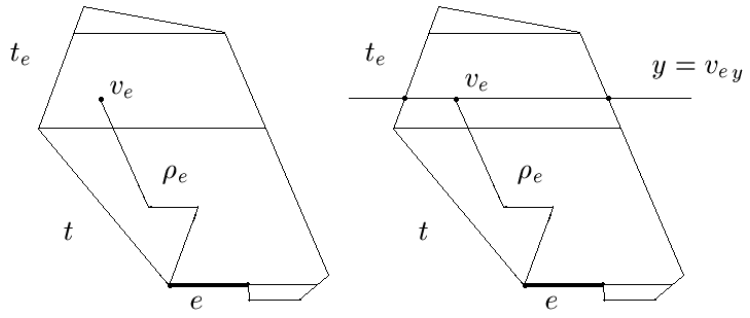

Fig. 5. Example for (left) Step 2.1.5 and (right) Step 2.1.6 of Procedure 1.

2.1.2. Find a trapezoid $t \in T_{\Pi}$ such that the bottom edge of t and edge e have a common end point in $V(\Pi)$.

2.1.3. Let ρ_e be the decreasing polygonal path from v_e to e.

2.1.4. Find a stack of trapezoids, denoted by T_e ($\subseteq T_{\Pi}$), such that for each $t \in T_e$, $t^{\bullet} \cap \rho_e \neq \emptyset$.

2.1.5. Let t_e be the topmost trapezoid in T_e.

2.1.6. Compute the intersection points between $y = v_{ey}$ with those two edges on the left and right side of t_e.

2.1.7. Update the left and right intersection points of v_e by comparing the results of Step 2.1.6 with the initial left and right intersection points of v_e.

For the following Procedure 2, let I be an interval of real numbers, and M_I be a subset of maximal points of Π such that for each element $v \in M_I$, $v_y \in I$. Suppose M_I is sorted according to y-coordinate decreasingly.

Procedure 2: remove up-stable points

1. Let $M_I = \{v_1, v_2, \ldots, v_k\}$. 2. For each $i \in \{1, 2, \ldots, k\}$, do the following:

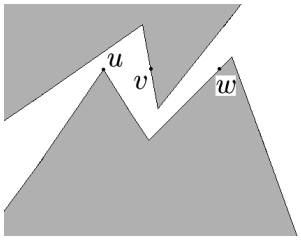

Fig. 6. Example for Step 2.1.7 of Procedure 1: u is an up-stable point, v is the result of Step 2.1.6, w is the initial right intersection point of u. Thus, v is the right intersection point of u.

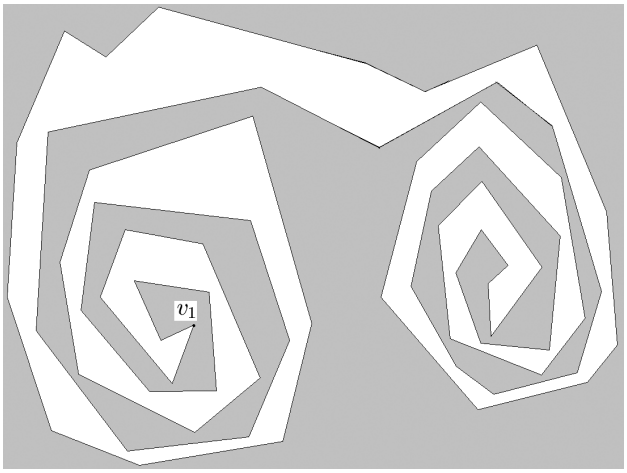

Fig. 7. Example for Step 1 of Procedure 2.

2.1. Find a closest down-stable point with respect to v_i around the frontier of Π, denoted by u_i.

2.2. Find a point $w_i \in \partial\Pi$ such that $\rho(u_i, \Pi, v_i, \Pi, w_i)$ is the shortest polygonal path in $\partial\Pi$ such that $u_{iy} = w_{iy}$.

2.3. Update Π by replacing $\rho(u_i, \Pi, v_i, \Pi, w_i)$ by the edge $u_i w_i$.

2.4. Let $\Pi_i = u_i w_i + \rho(u_i, \Pi, v_i, \Pi, w_i)$.

2.5. Now let Π_i be the input of Procedure 1 and update the left and right intersection points of all possible up-stable points.

Procedure 3: compute all left or right intersection points

1. Let $U = \{v_1, v_2, \ldots, v_k\}$ be the sorted set of up-stable points of Π such that $v_{1y} < v_{2y} < \cdots < v_{ky}$.

2. For each $i \in \{1, 2, \ldots, k\}$, do the following:

2.1. Find a closest minimal point with respect to v_i by following the frontier of Π, denoted by u_i.

2.2. Let $I = [a, b]$ where $a = u_{iy}$ and $b = v_{iy}$.

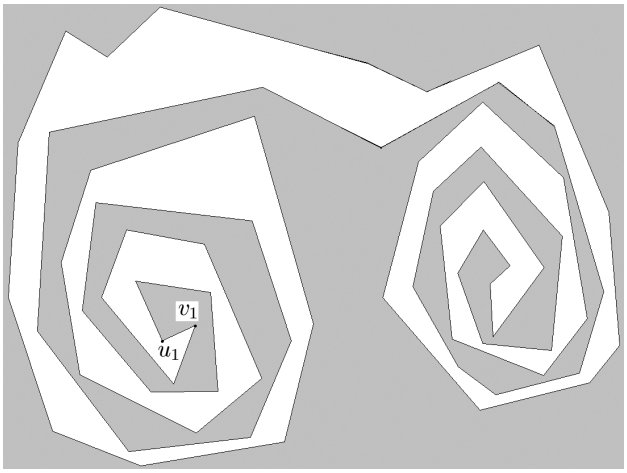

Fig. 8. Example for Step 2.1 of Procedure 2.

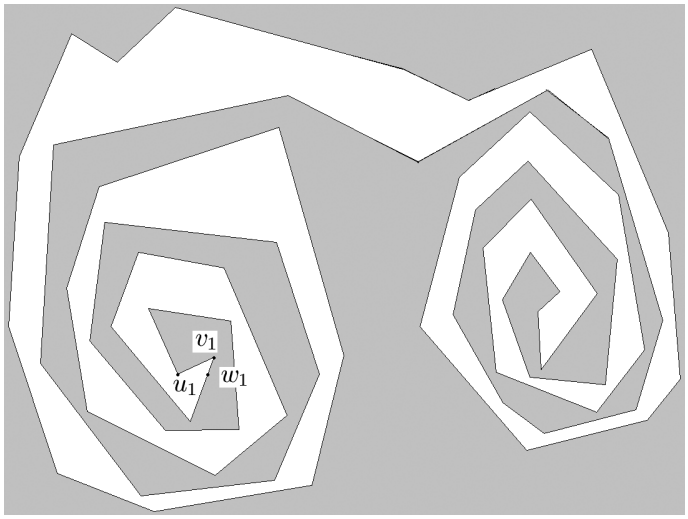

Fig. 9. Example for Step 2.2 of Procedure 2.

2.3. Compute M_I.

2.4. If $M_I \neq \emptyset$, then apply Procedure 2 and go to Step 2.1.

2.5. Otherwise, find a point w_i such that $\rho(u_i, \Pi, v_i, \Pi, w_i)$ is the shortest polygonal path of $\partial\Pi$ with $u_{iy} = w_{iy}$.

2.6. Set initial left and right intersection points of v_i as follows:

2.6.1. If $v_{i-1_y} = v_{iy} = v_{i+1_y}$ then let the initial left and right intersection points of v_i be v_{i-1} and v_{i+1}, respectively.

2.6.2. Otherwise, find two points w_{il}, w_{ir} such that $\rho(w_{il}, \Pi, u_i, \Pi, v_i, \Pi, w_{ir})$ is the shortest polygonal path of $\partial\Pi$ with $w_{ily} = v_{iy} = w_{iry}$.

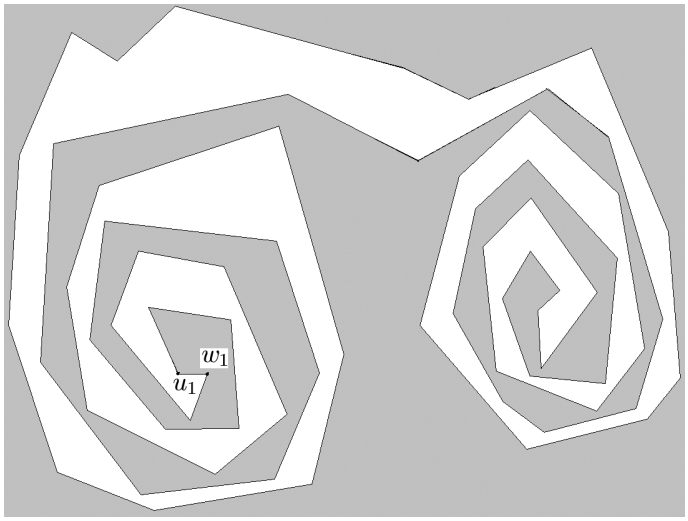

Fig. 10. Example for Step 2.3 of Procedure 2.

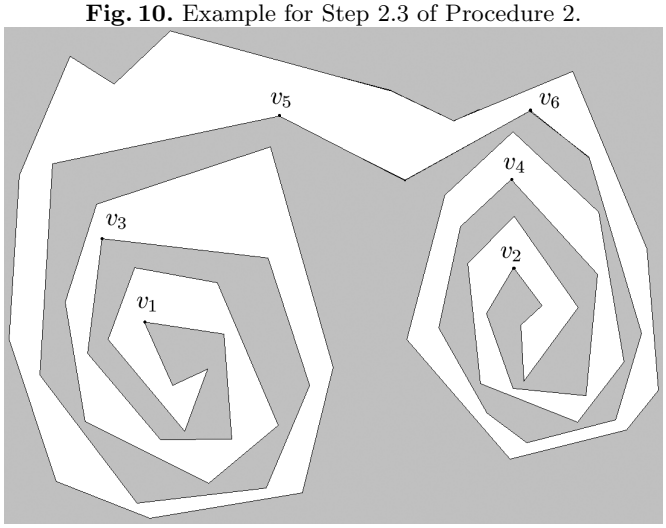

Fig. 11. Example for Step 1 of Procedure 3.

2.6.3. Let the initial left and right intersection points of v_i be w_{il} and w_{ir}, respectively.

2.7. Update Π by replacing $\rho(u_i, \Pi, v_i, \Pi, w_i)$ by the edge $u_i w_i$.

2.8. Now let Π be the input for Procedure 1; update the left and right intersection points of all possible up stable points.

Main Algorithm: Decomposition Algorithm

1. Let $S = \emptyset$ and $T = \emptyset$.

2. Compute the set of up- and down-stable points, denoted by V.

3. Apply Procedure 3 to compute the left and right intersection points, for all up-stable points in V.

4. Sort V for decreasing y-coordinates.

5. For each $i \in \{1, 2, \ldots, k\}$, with $k = |V|$, do the following:

5.1. *Case 1.* v_i is a down-stable point.

5.1.1. Find two points v_{il}, $v_{ir} \in \partial\Pi$ such that $v_{ily} = v_{iy} = v_{iry}$ and $v_{ilx} < v_{ix} < v_{irx}$.

5.1.2a. Let $\Pi_l = v_{il}v_i + \rho(v_{il}, \Pi, v_i)$ (up-polygon).

5.1.2b. Let $\Pi_r = v_iv_{ir} + \rho(v_i, \Pi, v_{ir})$ (up-polygon).

5.1.2c. Let $\Pi' = v_{il}v_{ir} + \rho(v_{il}, \Pi, v_{ir})$ (down-polygon).

5.1.3. Let $S = S \cup \{\Pi_l, \Pi_r\}$.

5.1.4. Let $\Pi = \Pi'$.

5.1.5. Let $i = i + 1$ and go to Step 5.

5.2. *Case 2.* v_i is an up-stable point.

5.2.1. Let v_{il}, v_{ir} be the left and right intersection points of v_i, respectively.

5.2.2a. Let $\Pi_l = v_{il}v_i + \rho(v_{il}, \Pi, v_i)$ (down-polygon).

5.2.2b. Let $\Pi_r = v_iv_{ir} + \rho(v_i, \Pi, v_{ir})$ (down-polygon).

5.2.2c. Let $\Pi' = v_{il}v_{ir} + \rho(v_{il}, \Pi, v_{ir})$ (up-polygon).

5.2.3. Let $S = S \cup \{\Pi'\}$.

5.2.4. Let $\Pi = \{\Pi_l, \Pi_r\}$.

5.2.5. Let $i = i + 1$ and go to Step 5.

6. For each $j \in \{1, 2, \ldots, n\}$, with $n = |S|$, do the following:

6.1. For each (monotonous) polygon $\Pi_j \in S$, decompose it into a stack of trapezoids, denoted by T_j.

6.2. Let $T = T \cup T_j$.

7. Output T.

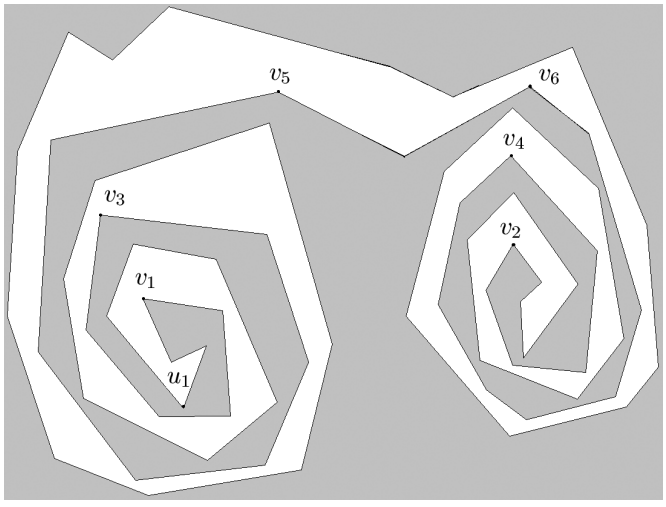

Fig. 12. Example for Step 2.1 of Procedure 3.

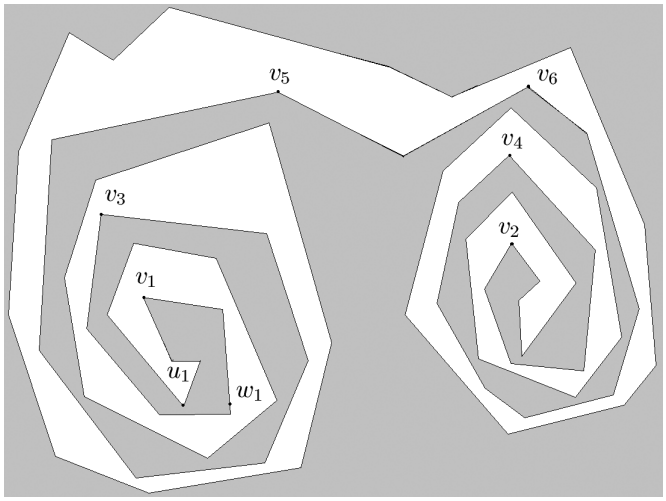

Fig. 13. Example for Step 2.5 of Procedure 3.

3.1 Time Complexity of the Algorithm

Lemma 1. *The set of up- (or down-, maximal) stable points of Π can be computed in $\mathcal{O}(n)$, where $n = |V(\Pi)|$.*

Proof. For a sufficiently small number $\epsilon > 0$, the "start" vertex v_0' of an inner polygon $\Pi(v_0, \epsilon)$ can be computed in $\mathcal{O}(n)$, where $n = |V(\Pi)|$ (see [3]). For three consecutive vertices u, v, $w \in V(\Pi)$ such that $u_x < v_x < w_x$, if $u_y < v_y$, $v_y > w_y$ and $v_y' > v_y$, then v is a up stable point (if $u_y < v_y$, $v_y > w_y$ and $v_y' < v_y$, it follows that v is a maximal point; if $u_y > v_y$, $v_y < w_y$ and $v_y' < v_y$, then v is a down-stable point). □

Lemma 2. *Procedure 1 can be computed in $\mathcal{O}(n \log n)$, where $n = |V(\Pi)|$.*

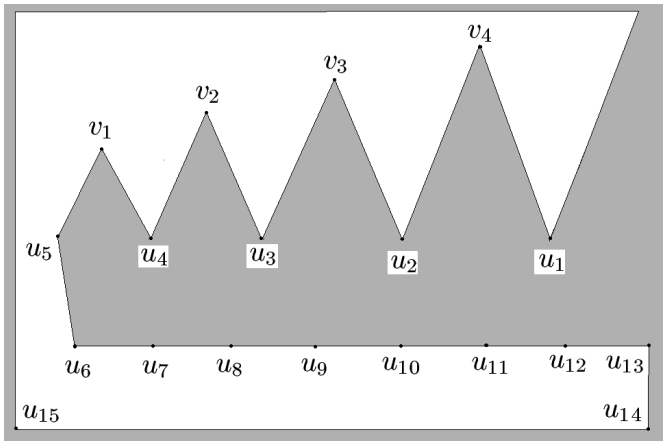

Fig. 14. Example 1 for Step 2.6.2 of Procedure 3.

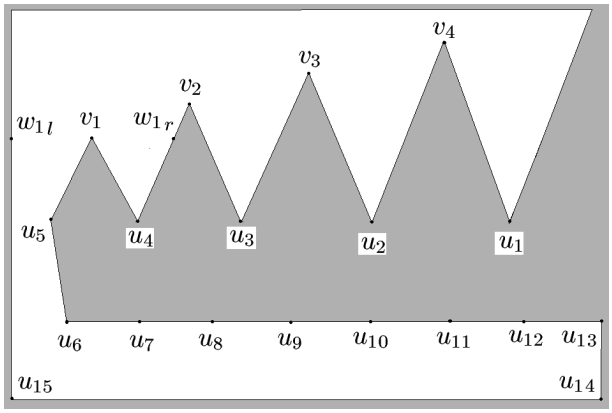

Fig. 15. Example 2 for Step 2.6.2 of Procedure 3.

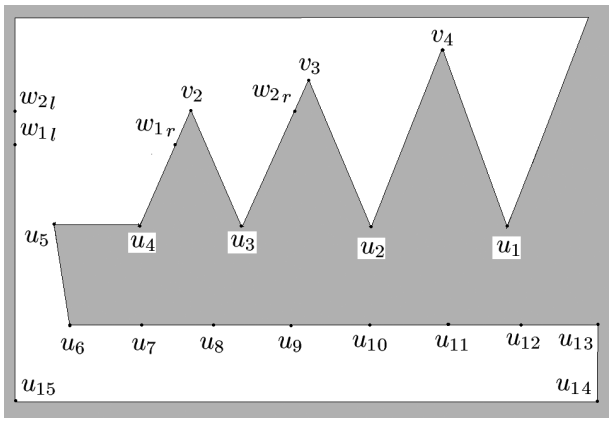

Fig. 16. Example 3 for Step 2.6.2 of Procedure 3.

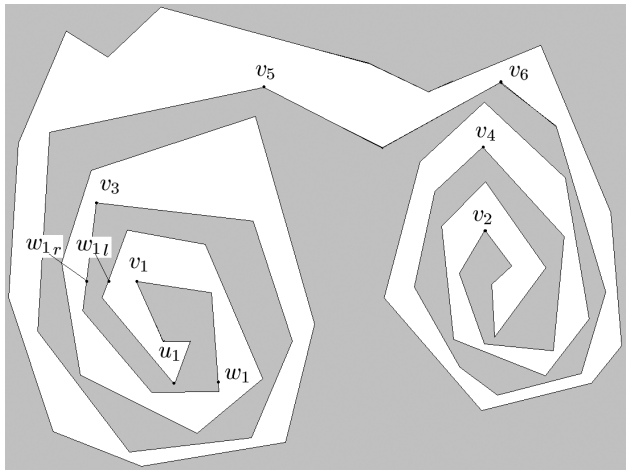

Fig. 17. Example for Step 2.6.3 of Procedure 3.

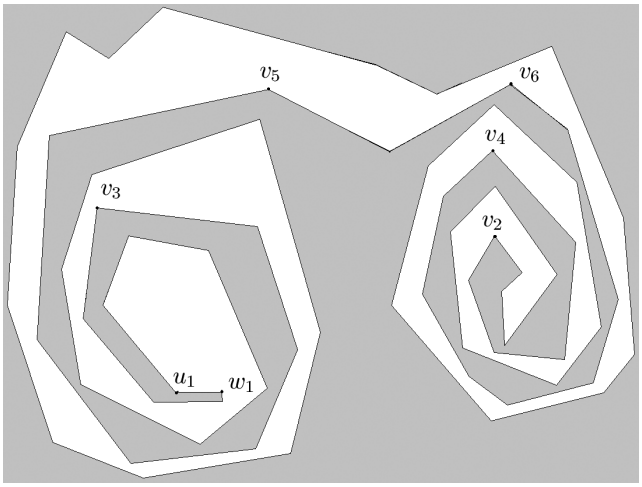

Fig. 18. Example for Step 2.7 of Procedure 3.

Proof. If Π is monotonous, then (obviously) it can be decomposed into a stack of trapezoids in $\mathcal{O}(|V(\Pi)|)$. Otherwise, by assumption, Π can only have a finite number of down-stable points. Analogous to Step 5.1 in the decomposition algorithm, Π can be decomposed into a set of trapezoids in $\mathcal{O}(|V(\Pi)|)$. Thus, Step 1 can be computed in $\mathcal{O}(|V(\Pi)|)$.

All steps following Step 2, except Step 2.1.4, can be computed in $\mathcal{O}(1)$.

Step 2.1.4 can be computed in $\mathcal{O}(nlogn)$, where $n = |V(\Pi)|$:

Let S_d be the set of all down-stable points of Π.

A1. Sort S_d according to y-coordinates; we obtain $S_d = \{v_1, v_2, \ldots, v_k\}$ such that $v_{1y} \leq v_{2y} \leq v_{3y} \leq \ldots \leq v_{ky}$.

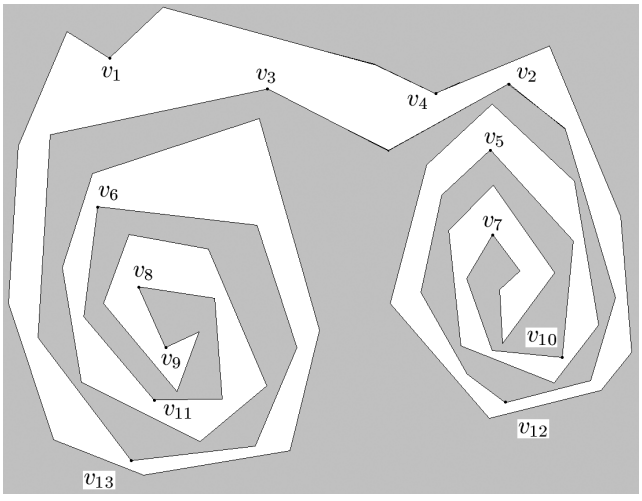

Fig. 19. Example for Step 2 of the decomposition algorithm.

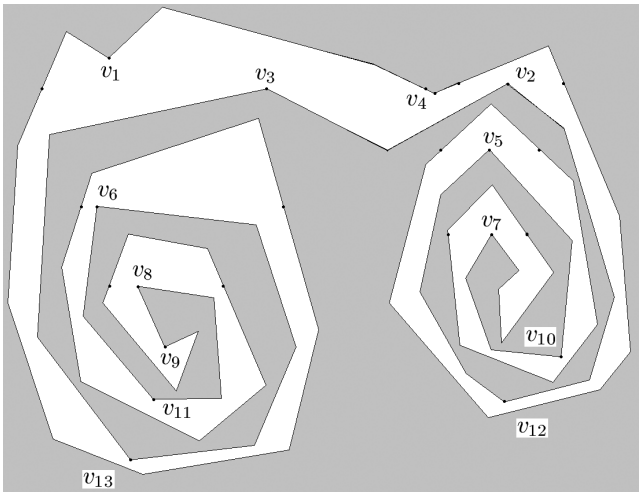

Fig. 20. Example for Step 3 of the decomposition algorithm (note the left and right intersection points).

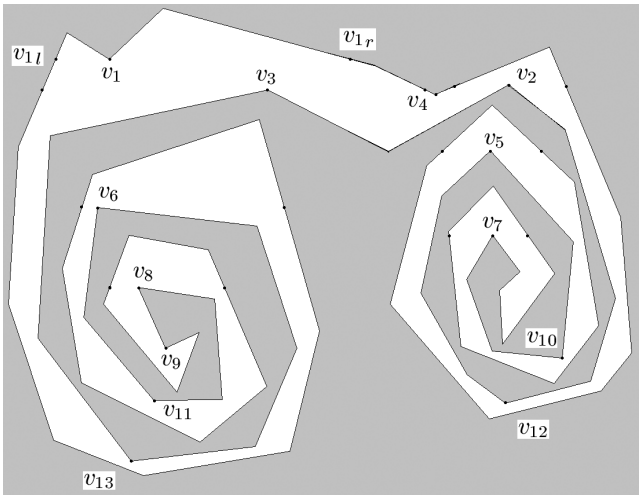

Fig. 21. Example for Step 5.1.1 of the decomposition algorithm.

A2. Let $m = \min\{|v_{i-1_y} - v_{iy}| : |v_{i-1_y} - v_{iy}| > 0, i = 2, 3, \ldots, k\}$, and $M = \max\{|v_{i-1_x} - v_{ix}| : |v_{i-1_x} - v_{ix}| > 0, i = 2, 3, \ldots, k\}$.

A3. Transform Π by rotating it by angle θ anticlockwise about the origin (i.e., for each point $p = (x, y) \in \Pi^\bullet$, update it by point (x', y'), where $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$).

Without loss of generality, assume that $m = |v_{1_y} - v_{2_y}| > 0$. We have that

$$\begin{aligned} |v'_{1_y} - v'_{2_y}| &= |(v_{1x} - v_{2x})\sin\theta + (v_{1_y} - v_{2_y})\cos\theta| \\ &\geq |(v_{1_y} - v_{2_y})\cos\theta| - |(v_{1x} - v_{2x})\sin\theta| \\ &\geq m\cos\theta - M\sin\theta \to m(\theta \to 0) \end{aligned}$$

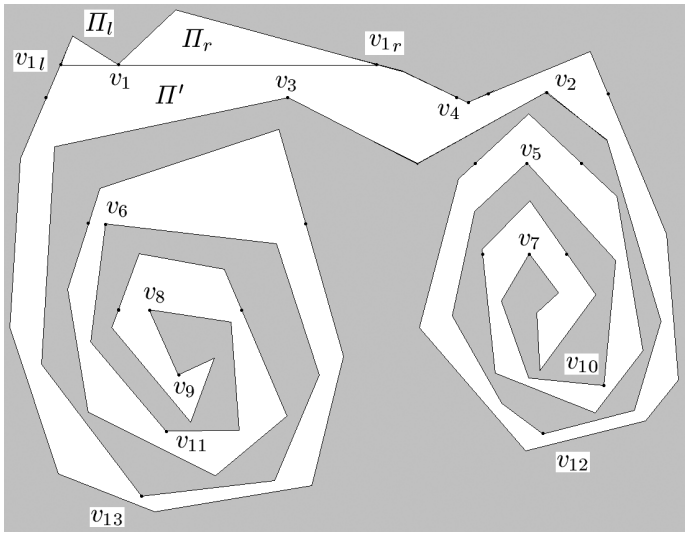

Fig. 22. Example for Steps 5.1.2a–c of the decomposition algorithm.

Therefore, there exists a sufficiently small angle $\theta > 0$ such that, after rotating, each down-stable point is still a down-stable point, and all down-stable points have unique y-coordinates. This implies that, for each trapezoid $t \in T_\Pi$, there exist at most two trapezoids $t_1, t_2 \in T_\Pi$ such that t_1 and t_2 has a common vertex with t, respectively. (In this case, t_1 and t_2 have a common vertex which is a down-stable point. See Figure 25.) Since Step A2 can be computed in $\mathcal{O}(|S_d|)$ and Step A3 can be computed in $\mathcal{O}(1)$, it follows that Step 2.1.4 can be computed in $\mathcal{O}(k)$, where k is the number of vertices of the decreasing polygonal path from v_e to e, after sorting (Step A1) in $\mathcal{O}(|S_d|log|S_d|)$. □

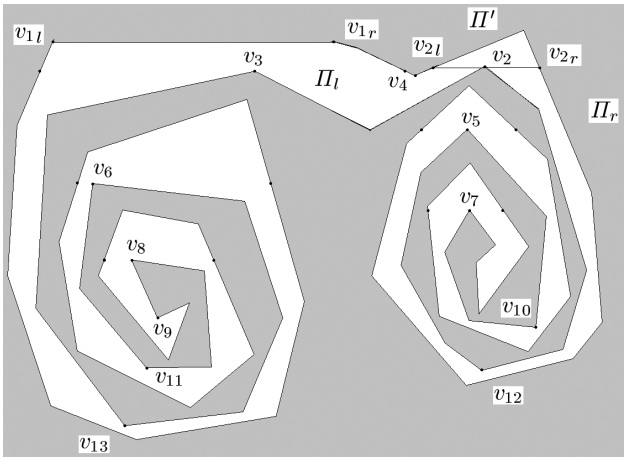

Fig. 23. Example for Steps 5.2.1 and 5.2.2a–c of the decomposition algorithm.

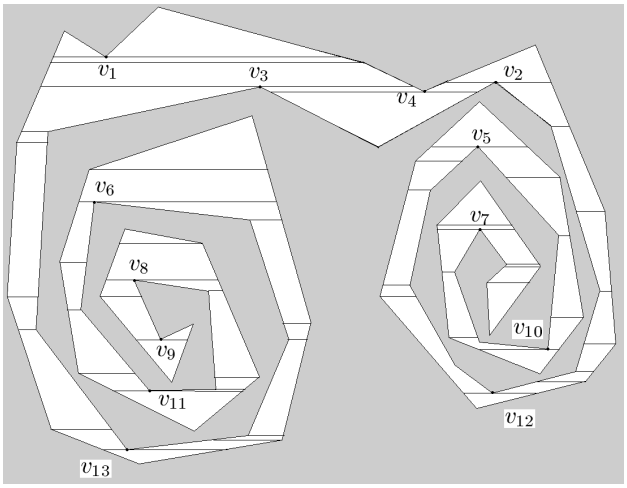

Fig. 24. Output of the decomposition algorithm for the polygon of Figure 1.

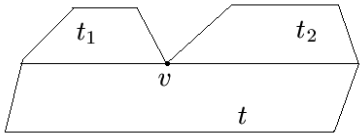

Fig. 25. Illustration for the proof of Lemma 2.

Lemma 3. *Procedure 2 can be computed in $\mathcal{O}(n\ log\ n)$ time, where n is the number of vertices of the original simple polygon Π.*

Proof. By Lemma 1, Step 1 can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of vertices of the original simple polygon Π. Step 2.1 can be computed in $\mathcal{O}(n_u)$, where n_u is the number of vertices of $\rho(u_i, \Pi, v_i)$. Step 2.2 can be computed in $\mathcal{O}(n_w)$, where n_w is the number of vertices of $\rho(v_i, \Pi, w_i)$. Steps 2.3 and 2.4 can be computed in $\mathcal{O}(1)$. By Lemma 2, Step 2.5 can be computed in $\mathcal{O}(n_i\ log\ n_i)$, where $n_i = |V(\Pi_i)|$. Thus, Step 2 can be computed in $\mathcal{O}(n\ log\ n)$ altogether, where n is the number of vertices of Π. □

Lemma 4. *Procedure 3 can be computed in $\mathcal{O}(n\ log\ n)$ time, where n is the number of vertices of the original simple polygon Π.*

Proof. By Lemma 1, Step 1 can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of vertices of the original simple polygon Π. Step 2.1 can be computed in $\mathcal{O}(n_u)$, where n_u is the number of vertices of $\rho(u_i, \Pi, v_i)$. Step 2.2 can be computed in $\mathcal{O}(1)$. Step 2.3 can be computed in $\mathcal{O}(|M_I|)$. By Lemma 4, Step 2.4 can be computed in $\mathcal{O}(n_u\ log\ n_u)$, where $n_u = |V(\rho(u_i, \Pi, v_i))|$. Step 2.5 can be computed in $\mathcal{O}(n_w)$, where n_w is the number of vertices of $\rho(u_i, \Pi, w_i)$. Step 2.6.1 can be computed in $\mathcal{O}(1)$. Step 2.6.2 can be computed in $\mathcal{O}(n_i)$, where $n_i = |V(\rho(u_i, \Pi, w_{il}))| + |V(\rho(w_i, \Pi, w_{ir}))|$. Step 2.6.3 can be computed in $\mathcal{O}(1)$.

Step 2.7 can be computed in $\mathcal{O}(1)$. By Lemma 2, Step 2.8 can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of the vertices of the updated Π. Therefore, Procedure 3 can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of the vertices of the original simple polygon Π. □

Theorem 1. *The given decomposition algorithm has time complexity $\mathcal{O}(n\ log\ n)$, where n is the number of vertices of the original simple polygon Π.*

Proof. Step 1 can be computed in $\mathcal{O}(1)$. By Lemma 1, Step 2 can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of vertices of the original simple polygon Π. By Lemma 4, Step 3 can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of the vertices of the original simple polygon Π. Step 4 can be computed in $\mathcal{O}(|V|\ log\ |V|)$. Step 5.1.1 can be computed in $\mathcal{O}(n_i)$, where $n_i = |V(\rho(v_{il}, \Pi, v_i))| + |V(\rho(v_i, \Pi, v_{ir}))|$. Steps 5.1.2a – 5.1.5 can be computed in $\mathcal{O}(1)$. Thus, Step 5.1 can be computed in $\mathcal{O}(n_i)$, where $n_i = |V(\rho(v_{il}, \Pi, v_i))| + |V(\rho(v_i, \Pi, v_{ir}))|$. Step 5.2 can be computed in $\mathcal{O}(1)$. By Lemma 3, Step 6.1 can be computed in $\mathcal{O}(|\Pi_j|)$. Thus, Step 6 can be computed in $\mathcal{O}(n)$, where n is the number of the vertices of the original simple polygon Π. Step 7 can be computed in $\mathcal{O}(1)$. Therefore, the decomposition algorithm can be computed in $\mathcal{O}(n\ log\ n)$, where n is the number of the vertices of Π. □

4 Conclusions

The report described an $\mathcal{O}(n\ log\ n)$ time algorithm for the decomposition of any simple polygon into trapezoids. (See Figure 24 for an example.) Of course, these can then be further divided into triangles. The algorithm was described on these eight pages (compared to 40 journal pages of paper [1]). (For a longer version of this paper, with some examples illustrating processing steps, see CITR-TR-199.) The given algorithm will allow to improve work reported in [2]. - The authors conjecture that the $\mathcal{O}(nlogn)$ sorting step in the given algorithm can also be replaced by incorporating some type of a (linear-time) scan, and future studies will show.

References

1. B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, **6**:485–524, 1991.
2. F. Li and R. Klette. Finding the shortest path between two points in a simple polygon by applying a rubberband algorithm. In Proc. PSIVT'06, pages 280-291, Hsinchu, Taiwan, LNCS, Springer, Berlin 2006.
3. D. Sunday. Algorithm 3: Fast winding number inclusion of a point in a polygon. See *www.geometryalgorithms.com/Archive/algorithm*$_0$103/ (last visit: April 2007).