Chapter 5

Bounding the Fréchet distance by the Hausdorff distance

As we have already seen (c.f., Figure 3.1 and 3.2), neither the ratio between δ_H and δ_F, nor the ratio between δ_F and $\tilde{\delta}_F$ is bounded in general. However, the following result from [10] (see also [35]) shows that for certain classes of curves the three distance measures are closely related:

Theorem 5.1 (δ_H vs. δ_F for convex closed curves, Alt et. al, [10]). For any pair of *convex closed* curves P and Q,

$$\delta_H(P,Q) = \tilde{\delta}_F(P,Q) = \delta_F(P,Q).$$

In the following we will consider *κ-straight* curves; for these curves the arclength between any two points is at most a constant κ times their Euclidean distance.

Definition 5.2 (κ-Straightness). *A planar (closed) rectifiable curve $P \in \mathcal{K}^0 \cup \mathcal{K}^1$ is called* κ-straight *for some real parameter $\kappa \geq 1$, if the following holds for any two points x and y on P:*

$$\mathrm{d}_P(x,y) \leq \kappa ||x-y||,$$

where $\mathrm{d}_P(x,y)$ is the arclength of the shortest piece of P connecting x and y.

Examples for straight curves are the curves with increasing chords of [44], where $\kappa \leq 2\pi/3$, or the self-approaching curves of [6].

We will see below that the straightness condition rules out the possibility of curves with small Hausdorff distance that have a large Fréchet distance: In section 5.1 we show that the Fréchet distance of κ-straight curves is at most $(\kappa+1)$ times their Hausdorff distance (c.f., Theorem 5.3 on the next page). This result gives rise to a randomized approximation algorithm that computes an upper bound on the Fréchet distance between two κ-straight curves that is off from the exact value by a multiplicative factor of $(\kappa+1)$. The algorithm runs in $\mathcal{O}((m+n)\log^2(m+n)2^{\alpha(m+n)})$ time for given polygonal curves P, Q with m and n

vertices (c.f., Corollary 5.8 on page 49), and thus outperforms the fastest known algorithm to compute the Fréchet distance exactly, which requires $\mathcal{O}(mn \log(mn))$ time, see [13]. In section 5.3 we also provide the first non-trivial algorithm to decide for any $\kappa \geq 1$, if a given polygonal curve is κ-straight; it runs in $\mathcal{O}(n \log^2 n)$ time for a polygonal curve on n vertices (c.f., Theorem 5.9 on page 49).

5.1 The upper bound

In this section we will show that for κ-straight polygonal curves the Fréchet distance is at most a factor of $(\kappa + 1)$ away from the Hausdorff distance.

Theorem 5.3 (δ_H vs. δ_F for κ-straight polygonal curves). *For any pair of κ-straight polygonal curves $P, Q \in \mathcal{K}^0$*

$$\delta_F(P,Q) \leq (\kappa+1)\delta_H(P,Q), \text{ if } \mathsf{max}(||P(0)-Q(0)||, ||P(1)-Q(1)||) \leq \delta_H(P,Q).$$

Proof. Let $\delta = \delta_H(P,Q)$. Since both $||P(0)-Q(0)||$ and $||P(1)-Q(1)||$ are bounded by δ, we know that the points L and R in $F_\delta(P,Q)$ are white. To each $x \in [0,1]$ we assign the value $h(x)$ defined by

$$h(x) = \mathsf{max}\{y \mid \exists x' \leq x \text{ s.t. } (x', y) \in F_\delta\}.$$

h is a piecewise continuous monotone function consisting of of elliptic arcs and horizontal segments. At the points of discontinuity we add vertical segments and obtain a curve φ_δ from the graph of h (see Figure 5.1). In addition, we add the vertical segment $\overline{Lh(0)}$ such that φ_δ starts in L and ends in R.

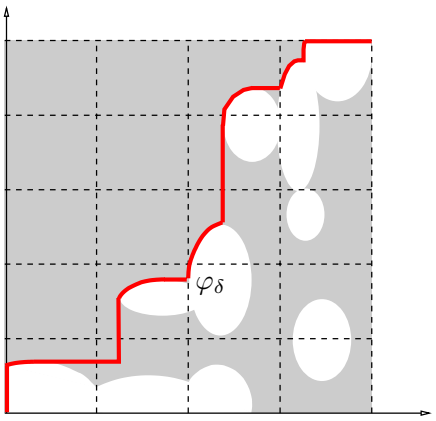


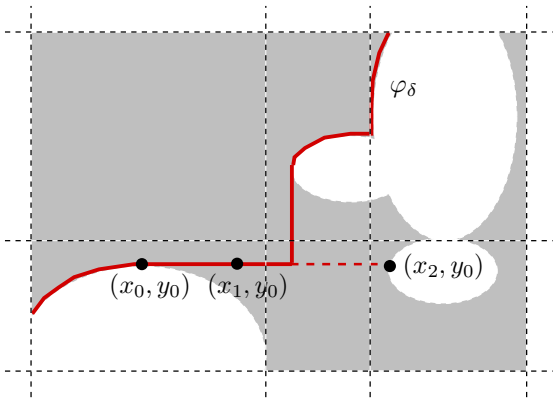
Figure 5.1: The reparametrization φ_δ in $F_\delta(P,Q)$.

The bi-monotone curve φ_δ consists of elliptical arcs and vertical and horizontal line segments. The left endpoint of a horizontal segment coincides with the upper endpoint

of an elliptical arc and the upper endpoint of a vertical segment coincides with the left endpoint of an elliptical arc. The line segments are always black (apart from the endpoint they share with an elliptical arc) and the elliptical arcs are always white.

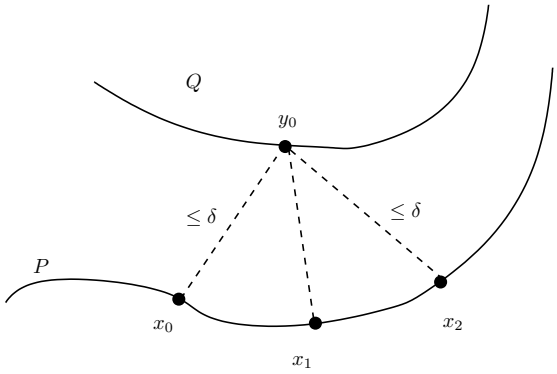
We claim that for all points (x, y) on φ_δ we have that $||P(x) - Q(y)|| \leq (\kappa + 1)\delta$, so that two corresponding reparametrizations yield a Fréchet distance of at most $(\kappa+1)\delta$. If (x, y) is white we are done since $||P(x) - Q(y)|| \leq \delta$ in that case, so let us consider a black point (x_1, y_0) on φ_δ that lies on a horizontal black segment (this implies that $x_1 < 1$) with the white left endpoint (x_0, y_0). It remains to show that $||P(x_1) - Q(y_0)|| \leq (\kappa + 1)\delta$.

We first claim that there exists a white point $(x_2, y_0) \in F_\delta$ with $x_2 > x_1$. To see this, observe that, since $\delta = \delta_H(P, Q)$, there exists for each y an x such that (x, y) is white, so for any $\epsilon > 0$ there exists an x such that $(x, y_0 + \epsilon)$ is white. Since (x_1, y_0) is white and the region above φ_δ is black (by definition), we can conclude that for any $\epsilon > 0$ there exists an $x > x_1$ such that $(x, y_0 + \epsilon)$ is white, and moreover, since F_δ is closed (by definition), that there exists an $x_2 > x_1$ such that (x_2, y_0) is white.



By applying the triangle inequality twice we obtain

$$\begin{aligned} ||P(x_1) - Q(y_0)|| \leq \ & \mathsf{min}(\ ||P(x_0) - Q(y_0)|| + ||P(x_0) - P(x_1)||, \\ & \quad ||P(x_2) - Q(y_0)|| + ||P(x_1) - P(x_2)||) \\ \leq \ & \mathsf{min}(\ \mathrm{d}_P(P(x_0), P(x_1)), \mathrm{d}_P(P(x_1), P(x_2))) + \delta. \end{aligned}$$



So with $\mathsf{min}(\mathrm{d}_P(P(x_0),P(x_1)),\mathrm{d}_P(P(x_1),P(x_2))) \leq \frac{1}{2}\,\mathrm{d}_P(P(x_0),P(x_2))$, and

$$\begin{aligned}\mathrm{d}_P(P(x_0),P(x_2)) &\leq \kappa||P(x_0)-P(x_2)|| \\ &\leq \kappa(||P(x_0)-Q(y_0)||+||P(x_2)-Q(y_0)||) \\ &\leq 2\kappa\delta,\end{aligned}$$

it follows that $||P(x_1)-Q(y_0)|| \leq (\kappa+1)\delta$.

If the black point (x_1,y_0) lies on a *vertical* black segment we can apply the same reasoning. □

Since $\delta_H(P,Q) \leq \tilde{\delta}_F(P,Q)$, and $\mathsf{max}(||P(0)-Q(0)||,||P(1)-Q(1)||) \leq \tilde{\delta}_F(P,Q)$ we can conclude:

Corollary 5.4 ($\tilde{\delta}_F$ vs. δ_F for κ-straight polygonal curves). *For any pair of κ-straight polygonal curves $P,Q \in \mathcal{K}^0$*

$$\delta_F(P,Q) \leq (\kappa+1)\tilde{\delta}_F(P,Q).$$

5.2 Computing the reparametrization

In this section we describe a divide-and-conquer algorithm that computes the reparametrization φ_δ. Our approach makes crucial use of the following result:

Theorem 5.5 (Combination Lemma, Agarwal/Sharir, [47]). Let B be a set of m_B *blue* Jordan arcs, and R be a set of m_R *red* Jordan arcs with the property that any two of these arcs intersect at most a constant number of times, and let P be a set of n points with the property that none of these points lies on one of the arcs; let $N = m_R + m_B + n$. Then the total complexity of all the regions induced by the red *and* the blue arcs that contain a point from P is $\mathcal{O}(N)$ and these regions can be computed in $\mathcal{O}(N \log N)$ time.

We show the following:

Theorem 5.6 (Computation of the upper envelope of $\Phi_\delta(P,Q)$). *Let $P,Q \in \mathcal{K}^0$ be* simple *polygonal curves with n, and m vertices, and $\delta > 0$. Then φ_δ can be computed in $\mathcal{O}((m+n)\log^2(m+n)2^{\alpha(m+n)})$* [a] *time by a randomized algorithm, and in $\mathcal{O}((m+n)\log^3(m+n)2^{\alpha(m+n)})$ time by a deterministic one.*

[a] Here, $\alpha(n) = \mathsf{min}\{k \geq 1 \mid A(k,k) \geq n\}$ is the functional inverse of the *Ackermann function* $A(k,n)$, which is defined as follows:

$$\begin{aligned}&A(1,n) = 2n, \quad n \geq 1 \\ &A(k,1) = 2, \quad k \geq 2 \\ &A(k,n) = A(k-1,A(k,n-1)), \quad k \geq 2, n \geq 2.\end{aligned}$$

See chapter 2 in [47] for a detailed discussion.

Proof. The algorithm proceeds as follows: In a first step, we determine the intersection of φ_δ with the vertical line in F_δ that corresponds to the midpoint $Q(1/2)$ of Q in $\mathcal{O}((m+n)\log^2(m+n)2^{\alpha(m+n)})$ deterministic time, and $\mathcal{O}((m+n)\log(m+n)2^{\alpha(m+n)})$ randomized time (see below); this yields a point $P(y_{1/2})$ on P, c.f., Figure 5.2.

Then we split P into $P_B := P|_{[0,y_{1/2}]}$ and $P_T := P|_{[y_{1/2},1]}$ and Q into $Q_L := Q|_{[0,1/2]}$ and $Q_R := Q|_{[1/2,1]}$, recursively compute $\varphi_\delta(P_B, Q_L)$ and $\varphi_\delta(P_T, Q_R)$, and glue them together.

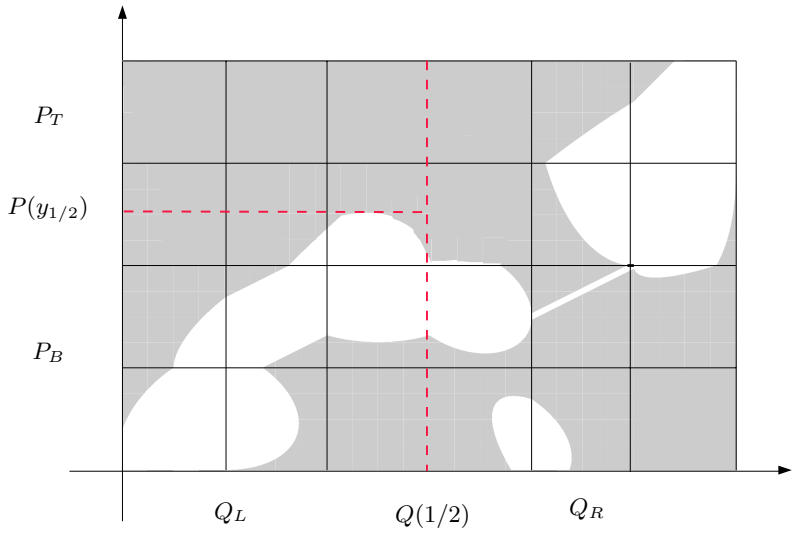
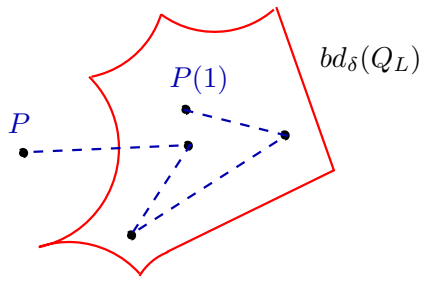


Figure 5.2: The first point on P (starting from $\mathbf{p}$) that intersects $\mathsf{bd}_\delta(Q_L)$.

In order to compute $P(y_{1/2})$, we imagine the following process: First we cut off the right half of $F_\delta(P,Q)$, i.e., we only look at $F_\delta(P,Q_L)$. Now we start sweeping a horizontal line from $y=1$ downwards to $y=0$ until it hits the first white point. This will happen at $y_{1/2}$. Another interpretation of this process is the following: We start walking on P in $P(1)$ towards $P(0)$. We walk until we arrive at a point that has distance at most δ to Q_L. This will happen at $P(y_{1/2})$. Of course we cannot afford to compute the diagram $F_\delta(P,Q)$ (or larger parts of it) explicitly, so we have to proceed in a different way.

Let $\mathbf{p} := P(1)$ be the endpoint of P. In a first step we check in $\mathcal{O}(m)$ time, if $\mathbf{p}$ is δ-close to Q_L. If this is the case, we are already finished. Otherwise we consider $\mathsf{nh}_\delta(Q_L)$, the δ-neighborhood of Q_L. This set can be described as the union of rectangles of width δ and circles of radius δ. The boundary of $\mathsf{nh}_\delta(Q_L)$ will be called $\mathsf{bd}_\delta(Q_L)$; it consists of circular arcs (of radius δ) and line segments. Since Q is simple, this boundary has complexity $\mathcal{O}(m)$ (c.f. [38] and [41]), and can be computed in $\mathcal{O}(m\log m)$ time, (c.f. [23]). The basic idea of our algorithm is based on the following simple observation:

Observation 5.7. *The first point on P (starting from $\mathbf{p}$) that intersects $\mathsf{bd}_\delta(Q_L)$ is part of the boundary of the cell C of the arrangement $\mathbf{A}$ induced by P and $\mathsf{bd}_\delta(Q_L)$, that contains $\mathbf{p}$.*



If we consider the segments and circular arcs of $\mathsf{bd}_\delta(Q_L)$ as a set of $\mathcal{O}(m)$ *red*, and the segments of P as a set of $\mathcal{O}(n)$ *blue* Jordan arcs, we can conclude with Theorem 5.5 that the cell C has complexity $\mathcal{O}(m+n)$.

So all we have to do is to compute the cell C, and check its boundary. For technical reasons, we need a point $\mathbf{p}'$ in this cell, that is neither part of P nor of $\mathsf{bd}_\delta(Q_L)$, in particular we cannot use $\mathbf{p}$ itself. Instead we compute the intersection of the line supporting the last segment of P with $P \cup \mathsf{bd}_\delta(Q_L)$ by brute force in $\mathcal{O}(m+n)$ time, and let $\mathbf{p}'$ be the midpoint between $\mathbf{p}$ and the intersection point that is closest to $\mathbf{p}$.

Since any two of the Jordan arcs of P and $\mathsf{bd}_\delta(Q_L)$ intersect at most twice, the cell of $\mathbf{A}$ that contains the point $\mathbf{p}'$ can be computed in $\mathcal{O}(\lambda_4(m+n)\log^2(m+n)) = \mathcal{O}((m+n)\log^2(m+n)2^{\alpha(m+n)})$ deterministic time ([36], see also [47], Theorem 6.11), or in $\mathcal{O}(\lambda_4(m+n)\log(m+n)) = \mathcal{O}((m+n)\log(m+n)2^{\alpha(m+n)})$ randomized time ([26], see also [47], Theorem 6.15). [b]

Complexity. Let n_X denote the number of vertices of the curve P_X for $X \in \{T, B\}$ and m_Y denote the number of vertices of the curve Q_Y for $Y \in \{L, R\}$. We have

$$n_T + n_B = n, \text{ and}$$
$$\mathsf{max}(m_L, m_R) \leq m/2, \text{ so}$$

the runtime $T(n,m)$ of the algorithm obeys the following recursion

$$\begin{aligned} T(n,m) &\leq T(n_B, m_L) + T(n_T, m_R) + d \cdot (n+m) f(n+m) \\ &\leq T(n_B, m/2) + T(n_T, m/2) + d \cdot (n+m) f(n+m), \end{aligned}$$

where $f(n,m)$ is a *monotone* function, and $d > 0$ is a suitable constant. We prove by induction on m that $T(n,m) \leq c \cdot (n+m) f(n+m) \log m$ for a suitable constant $c > 0$.

[b] For positive integers n, s, a sequence $U = \langle u_1, \ldots, u_m \rangle$ over $\{1, \ldots, n\}$ is called a *Davenport-Schinzel* sequence of order s over an n-element alphabet, if any two consecutive elements of U are distinct, and U does not contain a subsequence of length $s+2$ of the form $ababab\ldots$ for $a \neq b$. The maximum length of a Davenport-Schinzel sequence of order s over an n-element alphabet is denoted by $\lambda_s(n)$. It is known that $\lambda_4(n) = \mathcal{O}(n2^{\alpha(n)})$, see [47] for more details.

Since $T(n,1) = \mathcal{O}(n)$ the induction starts readily. Now

$$\begin{aligned} T(n,m) &\leq c \cdot (n_B + m/2) f(n_B + m/2) \log(m/2) + \\ &\quad c \cdot (n_T + m/2) f(n_T + m/2) \log(m/2) + \\ &\quad d \cdot (n+m) f(n+m) \\ &\leq c \cdot (n_B + m/2 + n_T + m/2) f(n+m)(\log m - 1) + \\ &\quad d \cdot (n+m) f(n+m) \\ &\leq c \cdot (n+m) f(n+m) \log m, \text{ if } c \geq d. \end{aligned}$$

□

With an algorithm of Alt et al. [9] we can compute $\delta = \delta_H(P,Q)$ in $\mathcal{O}((m+n)\log(m+n))$ time. Combining this with Theorems 5.3 and 5.6, we can find a $(\kappa+1)$-approximation to $\delta_F(P,Q)$, together with a reparametrization φ_{app} that witnesses this fact, within the time bounds stated in Theorem 5.6.

Corollary 5.8 (δ_F-approximation for κ-straight polygonal curves). *For any pair of κ-straight polygonal curves $P, Q \in \mathcal{K}^0$ with $\mathsf{max}(||P(0) - Q(0)||, ||P(1) - Q(1)||) \leq \delta_H(P,Q)$, we can compute a $(\kappa+1)$-approximation to $\delta_F(P,Q)$, together with a reparametrization φ_{app} by a deterministic (randomized) algorithm in $\mathcal{O}((m+n)\log^3(m+n)2^{\alpha(m+n)})$ ($\mathcal{O}((m+n)\log^2(m+n)2^{\alpha(m+n)})$) time.*

5.3 Recognizing κ-straight curves

In this section we describe an efficient algorithm to decide whether a given polygonal curve P is κ-straight. The results are summarized in the following Theorem.

Theorem 5.9 (Recognition of κ-straight curves). *Let $P \in \mathcal{K}^0 \cup \mathcal{K}^1$ be a simple (closed) polygonal curve on n vertices, and $\kappa \geq 1$. One can decide in $\mathcal{O}(n \log^2 n)$ time whether P is κ-straight.*

The Theorem is a combination of Lemmas 5.13 and 5.15; they will be proved in the next two subsections. If P is κ-straight, it is also κ'-straight for any $\kappa' \geq \kappa$. The smallest κ such that P is κ-straight is called the *detour* of P.

Definition 5.10 (Detour of a polygonal curve). *Let P be a simple polygonal curve, and $x, y \in P$ with $x \neq y$. Then*

$$\delta_P(x,y) := \frac{\mathrm{d}_P(x,y)}{||x-y||}$$

*denotes the P-*detour *between x and y. For $X, Y \subseteq P$*

$$\delta_P(X,Y) := \sup_{x \in X, y \in Y, x \neq y} \delta_P(x,y)$$

denotes the P-detour *between* X *and* Y*. Finally*

$$\delta(P) := \delta_P(P, P)$$

denotes the detour *of* P*. If* P *is understood from the context, we will write* δ *instead of* δ_P*.*

The decision problem version of the detour computation problem is equivalent to the problem of recognizing κ-straight curves.

5.3.1 Simple polygonal curves

Let us first consider the case where $P \in \mathcal{K}^0$ is a simple polygonal curve. The detour problem in that setting has been studied before by Ebbers-Baumann et al., [31]. They give an $(1+\epsilon)$-approximation algorithm for computing the detour of P that runs in $\mathcal{O}(n \log n)$ time. Their approach makes use of the fact that the detour of a curve is always attained at a vertex of the curve. We will also exploit that property in the proof of Lemma 5.12 below.

Lemma 5.11 (Ebbers-Baumann et al., [31]). Let $P \in \mathcal{K}^0$ be a simple polygonal curve, and let $L \subseteq P$ and $R \subseteq P$ be two subcurves of P. Then $\delta(L, R) = \mathsf{max}\big(\delta(L_0, R), \delta(L, R_0)\big)$.

We first show the following technical Lemma which is of independent interest, and will be used in the next section, too.

Lemma 5.12 (Bichromatic disjoint subcurve detour). *Let* $P \in \mathcal{K}^0$ *be a simple polygonal curve, and let* $\kappa \geq 1$*. Let* $L \subseteq P$ *and* $R \subseteq P$ *be two disjoint subcurves of* P *with* $|L| = m_L$ *and* $|R| = m_R$*, and let* $m = m_L + m_R$*. Assume that for two vertices* $(\mathbf{l}, \mathbf{r}) \in L \times R$ *the distance* $\mathrm{d}_P(\mathbf{l}, \mathbf{r})$ *can be computed in* $\mathcal{O}(1)$ *time, and that* $\mathsf{max}\big(\delta_P(L), \delta_P(R)\big) \leq \kappa$*. Then one can decide in* $\mathcal{O}\big(m \log m\big)$ *time whether* $\delta_P(L, R) \leq \kappa$*.*

Proof. From Lemma 5.11, we know that $\delta(L, R) = \mathsf{max}\big(\delta(L_0, R), \delta(L, R_0)\big)$. Therefore it is sufficient to test if $\delta(L_0, R) \leq \kappa$ and $\delta(L, R_0) \leq \kappa$. Let $(\mathbf{l}, \mathbf{r}) \in (L, R_0)$, and $\mathbf{m}$ be a vertex on P that separates L and R. Then

$$\begin{aligned}
\delta(\mathbf{l}, \mathbf{r}) \leq \kappa &\iff \frac{\mathrm{d}_P(\mathbf{l}, \mathbf{r})}{||\mathbf{l} - \mathbf{r}||} \leq \kappa \\
&\iff \frac{\mathrm{d}_P(\mathbf{l}, \mathbf{m}) + \mathrm{d}_P(\mathbf{m}, \mathbf{r})}{||\mathbf{l} - \mathbf{r}||} \leq \kappa \\
&\iff \frac{\mathrm{d}_P(\mathbf{l}, \mathbf{m})}{\kappa} \leq ||\mathbf{l} - \mathbf{r}|| - \frac{\mathrm{d}_P(\mathbf{m}, \mathbf{r})}{\kappa}.
\end{aligned}$$

Now look at the bi-variate function

$$C_{\mathbf{r},\mathbf{m},\kappa} : \begin{cases} \mathbb{R}^2 \to \mathbb{R} \\ \mathbf{x} \mapsto ||\mathbf{x} - \mathbf{r}|| - \mathrm{d}_P(\mathbf{m}, \mathbf{r})/\kappa. \end{cases}$$

If $\mathbf{r} = (r_x, r_y)$, the graph of $C_{\mathbf{r},\mathbf{m},\kappa}$ is a cone in $\mathbb{R}^3$ with apex $(r_x, r_y, -\mathrm{d}_P(\mathbf{m}, \mathbf{r})/\kappa)$, apex angle $\pi/2$ and its principal axis parallel to the z-axis. If $\mathbf{l} = (l_x, l_y)$ then

$$\delta(\mathbf{l}, \mathbf{r}) \leq \kappa \iff (l_x, l_y, \mathrm{d}_P(\mathbf{l}, \mathbf{m})/\kappa) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa}.$$

Consider the *lifting map*

$$\Lambda_{\mathbf{m},\kappa} : \begin{cases} L \to \mathbb{R}^3 \\ \mathbf{l} = (l_x, l_y) \mapsto (l_x, l_y, \mathrm{d}_P(\mathbf{l}, \mathbf{m})/\kappa). \end{cases}$$

The image of L under this lifting map is a polygonal curve in $\mathbb{R}^3$ whose projection to the xy-plane is L. Now we have

$$\begin{aligned} \delta(L, \mathbf{r}) \leq \kappa &\iff \Lambda_{\mathbf{m},\kappa}(L) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa}, \text{ and therefore} \\ \delta(L, R_0) \leq \kappa &\iff \Lambda_{\mathbf{m},\kappa}(L) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa} \text{ for all } \mathbf{r} \in R_0. \end{aligned}$$

If $C_{\mathbf{m},\kappa} := \mathsf{min}_{\mathbf{r} \in R_0} C_{\mathbf{r},\mathbf{m},\kappa}$ denotes the lower envelope of all $C_{\mathbf{r},\mathbf{m},\kappa}$, then

$$\delta(L, R_0) \leq \kappa \iff \Lambda_{\mathbf{m},\kappa}(L) \text{ lies below } C_{\mathbf{m},\kappa}.$$

The minimization diagram of $C_{\mathbf{m},\kappa}$, i.e., the projection of the edges of this lower envelope to the xy-plane, is the additively weighted Voronoi diagram $\mathsf{AVD}_{\mathbf{m},\kappa}(R_0)$ of the points $\mathbf{r} \in R_0$ with weights $-\mathrm{d}_P(\mathbf{m}, \mathbf{r})/\kappa$. This diagram can be computed in $\mathcal{O}(m_R \log m_R)$ time, c.f. [34]. For a vertex $\mathbf{r} \in R_0$ let $\mathsf{V}(\mathbf{r})$ denote the Voronoi cell of $\mathbf{r}$. Let $L_{\mathsf{V}}(\mathbf{r})$ be the set of maximal connected edge segments of L inside $\mathsf{V}(\mathbf{r})$. Then

$$\delta(L, R_0) \leq \kappa \iff \Lambda_{\mathbf{m},\kappa}(e) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa} \text{ for all } \mathbf{r} \in R_0 \text{ and for all } e \in L_{\mathsf{V}}(\mathbf{r}).$$

Let A denote the arrangement that results from superimposing $\mathsf{AVD}_{\mathbf{m},\kappa}(R_0)$ with the polygonal curve L. For a vertex $\mathbf{r} \in R_0$ let $\mathsf{A}(\mathbf{r})$ denote the cell of A that contains $\mathbf{r}$. Let $L_{\mathsf{A}}(\mathbf{r})$ be the set of maximal connected edge segments of L on the boundary of $\mathsf{A}(\mathbf{r})$. Then

$$\delta(L, R_0) \leq \kappa \iff \Lambda_{\mathbf{m},\kappa}(e) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa} \text{ for all } \mathbf{r} \in R_0 \text{ and for all } e \in L_{\mathsf{A}}(\mathbf{r}).$$

To see this, pick some $e \in L_{\mathsf{V}}(\mathbf{r})$ and a point $\mathbf{z}$ on e, see Figure 5.3. Since $\mathsf{V}(\mathbf{r})$ is star shaped wrt $\mathbf{r}$, the line segment s from $\mathbf{r}$ to $\mathbf{z}$ lies completely inside $\mathsf{V}(\mathbf{r})$. Assume that s crosses the segments $e_1, \ldots, e_k = e$ from $L_{\mathsf{V}}(\mathbf{r})$ in that order. In particular $e_1 \in L_{\mathsf{A}}(\mathbf{r})$, i.e., e_1 lies on the boundary of $\mathsf{A}(\mathbf{r})$. Let $\mathbf{r} = p_0, p_1, \ldots, p_k = \mathbf{z}$ denote the corresponding points of intersection with e_i for $1 \leq i \leq k$.

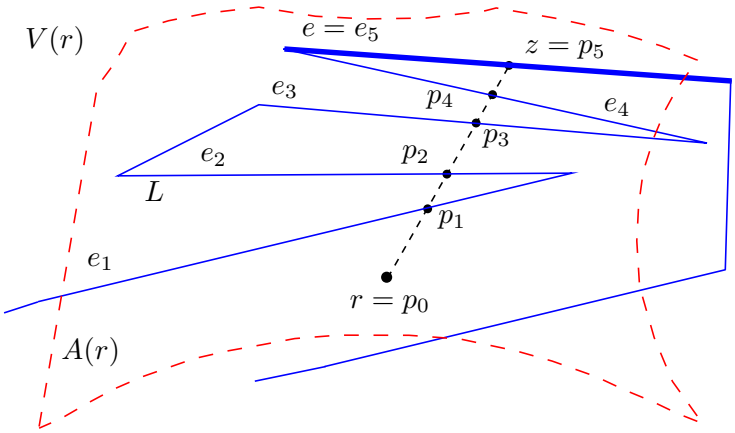


Figure 5.3: The cell $\mathsf{A}(\mathbf{r})$ in the superposition of $\mathsf{AVD}_{\mathbf{m},\kappa}(R_0)$ and L.

We have that $||\mathbf{z}-\mathbf{r}|| = |s| = \sum_i ||p_i - p_{i+1}||$ and $\mathrm{d}_P(\mathbf{z},\mathbf{r}) \leq \sum_i \mathrm{d}_P(p_i, p_{i+1})$, so

$$\delta(\mathbf{z},\mathbf{r}) \leq \frac{\sum_i \mathrm{d}_P(p_i,p_{i+1})}{\sum_i ||p_i - p_{i+1}||}.$$

Observe that for any two sequences of positive numbers $x_1,\ldots,x_n$ and $y_1,\ldots,y_n$ we have that

$$\frac{\sum_i x_i}{\sum_i y_i} \leq \mathsf{max}_i \frac{x_i}{y_i}.$$

This is easily seen, since if

$$\mathsf{max}_i x_i/y_i = x_1/y_1$$

then

$$\frac{1}{\sum_i y_i}\sum_i x_i \leq \frac{1}{\sum_i y_i}\sum_i \frac{y_i}{y_1}x_1 = x_1/y_1.$$

Thus

$$\begin{aligned}
\delta(\mathbf{z},\mathbf{r}) &\leq \frac{\sum_i \mathrm{d}_P(p_i,p_{i+1})}{\sum_i ||p_i - p_{i+1}||} \\
&\leq \mathsf{max}_i \frac{\mathrm{d}_P(p_i,p_{i+1})}{||p_i - p_{i+1}||} \\
&\leq \mathsf{max}\big(\delta(p_1,\mathbf{r}),\delta(L)\big) \\
&\leq \mathsf{max}\big(\delta(e_1,\mathbf{r}),\delta(L)\big) \\
&\leq \mathsf{max}\big(\delta(L_{\mathsf{A}}(\mathbf{r}),\mathbf{r}),\delta(L)\big).
\end{aligned}$$

Since $\mathbf{z}$ was a arbitrary point of $L_{\mathsf{V}}(\mathbf{r})$ this implies

$$\delta(L_{\mathsf{V}}(\mathbf{r}), \mathbf{r}) \leq \mathsf{max}\big(\delta(L_{\mathsf{A}}(\mathbf{r}), \mathbf{r}), \delta(L)\big).$$

So with $\delta(L) \leq \kappa$ we get

$$\delta(L_{\mathsf{V}}(\mathbf{r}), \mathbf{r}) \leq \kappa \iff \delta(L_{\mathsf{A}}(\mathbf{r}), \mathbf{r}) \leq \kappa.$$

Thus for all $\mathbf{r} \in R_0$

$$\Lambda_{\mathbf{m},\kappa}(e) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa} \text{ for all } e \in L_{\mathsf{V}}(\mathbf{r}) \iff \Lambda_{\mathbf{m},\kappa}(\tilde{e}) \text{ lies below } C_{\mathbf{r},\mathbf{m},\kappa} \text{ for all } \tilde{e} \in L_{\mathsf{A}}(\mathbf{r}).$$

The edges in $L_{\mathsf{A}}(\mathbf{r})$ are part of the boundary of the cell of A that contains $\mathbf{r}$. Therefore the total number of these edges is bounded by the total complexity of all cells of A that contain some $\mathbf{r} \in R_0$. We will now show that the total complexity of all these cells is $\mathcal{O}(m)$, and that the boundaries of all these cells (and therefore all the edges in $L_{\mathsf{A}}(\mathbf{r})$) can be computed in $\mathcal{O}(m \log m)$ time.

To this end, consider the segments of L as a set of m_L *blue* Jordan arcs, and the edges of $\mathsf{AVD}_{\mathbf{m},\kappa}(R_0)$ as a set of $\mathcal{O}(m_R)$ *red* Jordan arcs. Any two of these Jordan arcs intersect at most twice, and none of the points in R_0 lies on one of them.

The total complexity of all the regions induced by the *red* arcs that contain a point from R_0 is $\mathcal{O}(m_R)$; moreover we have an explicit description of these regions, since they correspond to the Voronoi regions of the vertices from R_0. The total complexity of all the regions (this is in fact only one region) induced by the *blue* arcs that contain a point from R_0 is $\mathcal{O}(m_L)$; again we have an explicit description of this region.

With Lemma 5.5 we can conclude that the total complexity of all the regions induced by the red *and* blue arcs that contain a point from R_0 is $\mathcal{O}(m_L + m_R) = \mathcal{O}(m)$ and that these regions can be computed in $\mathcal{O}(m \log m)$ time.

□

The following Lemma contains the main result of this section.

Lemma 5.13 (Detour of a polygonal curve). *Let $P \in \mathcal{K}^0$ be a simple polygonal curve on n vertices, and let $\kappa \geq 1$. Then one can decide in $\mathcal{O}(n \log^2 n)$ time whether $\delta(P) \leq \kappa$.*

Proof. In a preprocessing step we traverse P in $\mathcal{O}(n)$ time, starting from $P(0)$, and store in each vertex $P(i)$ the distance $\mathrm{d}_P(P(0), P(i))$. This will enable us to determine the distance on P between any pair of vertices of P in $\mathcal{O}(1)$ time (this is required in order to apply Lemma 5.12).

Now we split P into two subcurves $L := P|_{[0,\lfloor n/2 \rfloor]}$ and $R := P|_{[\lfloor n/2 \rfloor, n]}$ with $\mathbf{m} = P(\lfloor n/2 \rfloor)$ as a split vertex; observe that $\delta(P) = \mathsf{max}\big(\delta(L), \delta(R), \delta(L, R)\big)$. Then we check recursively whether $\delta(L) \leq \kappa$ and $\delta(R) \leq \kappa$. If this is not the case then $\delta(P) > \kappa$. Finally we run the algorithm from the proof of Lemma 5.12 to decide in $\mathcal{O}(n \log n)$ time, if $\delta(L, R) \leq \kappa$. The recursion only adds an additional logarithmic factor to this runtime. □

5.3.2 Simple closed polygonal curves

Let us now consider the case where $P \in \mathcal{K}^1$ is a simple closed polygonal curve. We are not aware of any related work on the problem of computing the detour for closed curves. For a *closed* rectifiable curve P and two points $x, y \in P$ with $x \neq y$ the distance on P between these two vertices, $\mathrm{d}_P(x, y)$, is the length of the *shortest* subcurve of P connecting x and y. The detour between two points on P and the detour of P itself are definied in the same manner as for open curves.

Since the statement of Lemma 5.11 is not valid anymore for closed curves we take a different approach, and use a recursive partition technique to enable us to apply our previous techniques to solve the detour problem for curves appropriately; we first prove a technical result:

Lemma 5.14 (Top-Bottom detour of a closed polygonal curve). *Let $\kappa \geq 1$, and let $P \in \mathcal{K}^1$ be a simple closed polygonal curve with four vertices t_L, t_R, b_R, b_L on P (in that order) such that*

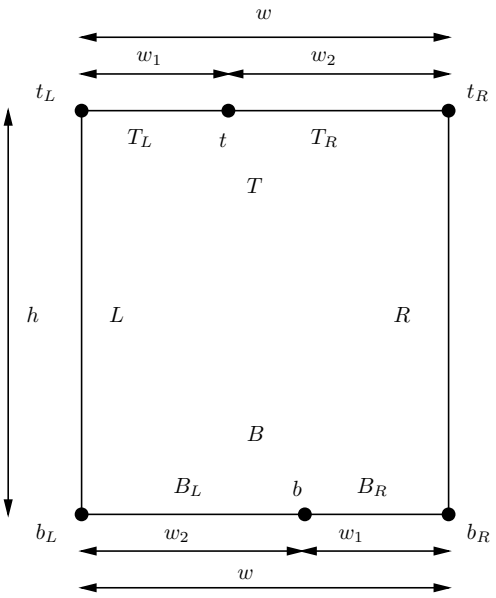
1. $\mathrm{d}_P(t_L, t_R) = \mathrm{d}_P(b_R, b_L)$,
2. $\mathrm{d}_P(t_R, b_R) = \mathrm{d}_P(b_L, t_L)$, *and*
3. $\mathsf{max}\big(\delta_P(T \cup R), \delta_P(R \cup B), \delta_P(B \cup L), \delta_P(L \cup T)\big) \leq \kappa$,

with $T := P|_{[t_L, t_R]}$, $R := P|_{[t_R, b_R]}$, $B := P|_{[b_R, b_L]}$, and $L := P|_{[b_L, t_L]}$. Let n be the total number of vertices of T and B. Assume that for two vertices $(\mathbf{l}, \mathbf{r}) \in P$ the distance $\mathrm{d}_P(\mathbf{l}, \mathbf{r})$ can be computed in $\mathcal{O}(1)$ time. Then one can decide in $\mathcal{O}\big(n \log^2 n\big)$ time whether $\delta_P(T, B) \leq \kappa$.

Proof. Let us assume wlog that T contains more than $n/2$ vertices; call this number n_T. We claim that the following algorithm correctly decides if $\delta_P(T_R, B_L) \leq \kappa$.

Algorithm *TB_Closed_Curve_Detour*$(P, t_L, t_R, b_L, b_R, \kappa)$

1. Choose a point $t \in P$ on T so that $T_L := P|_{[t_L, t]}$ and $T_R := P|_{[t, t_R]}$ contain $n_T/2$ vertices each.
2. Split B with a (possibly new) vertex b into $B_L := P|_{[b, b_L]}$ and $B_R := P|_{[b_R, b]}$, such that $\mathrm{d}_P(b_R, b) = \mathrm{d}_P(t_L, t)$ and $\mathrm{d}_P(b, b_L) = \mathrm{d}_P(t, t_R)$.
3. Check if $\delta_{P_L}(T_L, B_L) \leq \kappa$, where $P_L := T_L \cup L \cup B_L$.
4. Check if $\delta_{P_R}(T_R, B_R) \leq \kappa$, where $P_R := T_R \cup R \cup B_R$.
5. Recursively call the algorithm with
$$t'_L = t_L, t'_R = t, b'_R = b_R, b'_L = b$$
to check that $\delta_P(T_L, B_R) \leq \kappa$.
6. Recursively call the algorithm with
$$t''_L = t, t''_R = t_R, b''_R = b, b''_L = b_L$$
to check that $\delta_P(T_R, B_L) \leq \kappa$.



Correctness: First we prove that $\delta_P(P_L) \leq \kappa$ whenever $\delta_{P_L}(T_L, B_L) \leq \kappa$. To see this, observe that $\mathrm{d}_P(u,v) = \mathrm{d}_{P_L}(u,v)$ for $u, v \in P_L$ and therefore $\delta_P(T_L, B_L) = \delta_{P_L}(T_L, B_L) \leq \kappa$; similarly $\delta_P(T_R, B_R) = \delta_{P_R}(T_R, B_R) \leq \kappa$. Now since

$$\begin{aligned}
\delta_P(T_L, B_L) &\leq \kappa, \\
\delta_P(L) &\leq \delta_P(L \cup T) \leq \kappa, \\
\delta_P(L, T_L) &\leq \delta_P(L \cup T_L) \leq \delta_P(L \cup T) \leq \kappa, \\
\delta_P(L, B_L) &\leq \delta_P(L \cup B_L) \leq \delta_P(L \cup B) \leq \kappa, \\
\delta_P(T_L) &\leq \delta_P(L \cup T_L) \leq \delta_P(L \cup T) \leq \kappa, \text{ and} \\
\delta_P(B_L) &\leq \delta_P(L \cup B_L) \leq \delta_P(L \cup B) \leq \kappa,
\end{aligned}$$

we can conclude that $\delta_P(P_L) \leq \kappa$. A symmetric argument yields that $\delta_P(P_R) \leq \kappa$ if $\delta_{P_R}(T_R, B_R) \leq \kappa$.

In order to apply induction we have to argue that the subproblems created in Steps 5 and 6 of the algorithm meet the preconditions of Lemma 5.14: To this end, observe that

$$\mathrm{d}_P(t_L, t) = \mathrm{d}_P(b_R, b), \text{ and } \mathrm{d}_P(t, b_R) = \mathrm{d}_P(b, t_L).$$

With

$$T' = T_L, R' = T_R \cup R, B' = B_R, \text{ and } L' = B_L \cup L$$

it follows that

- $\delta_P(T' \cup R') \leq \kappa$, since $T' \cup R' = T \cup R$, and $\delta_P(T \cup R) \leq \kappa$,

- $\delta_P(R' \cup B') \leq \kappa$, since $R' \cup B' = P_R$, and $\delta_P(P_R) \leq \kappa$,
- $\delta_P(B' \cup L') \leq \kappa$, since $B' \cup L' = B \cup L$, and $\delta_P(B \cup L) \leq \kappa$, and finally
- $\delta_P(L' \cup T') \leq \kappa$, since $L' \cup T' = P_L$, and $\delta_P(P_L) \leq \kappa$.

Similarly,

$$\mathrm{d}_P(t, t_R) = \mathrm{d}_P(b, b_L), \text{ and } \mathrm{d}_P(t_R, b) = \mathrm{d}_P(b_L, t),$$

and with

$$T'' = T_R, R'' = R \cup B_R, B'' = B_L, \text{ and } L'' = L \cup T_L$$

it follows that

- $\delta_P(T'' \cup R'') \leq \kappa$, since $T'' \cup R'' = P_R$ and $\delta_P(P_R) \leq \kappa$,
- $\delta_P(R'' \cup B'') \leq \kappa$, since $R'' \cup B'' = R \cup B$ and $\delta_P(R \cup B) \leq \kappa$,
- $\delta_P(B'' \cup L'') \leq \kappa$, since $B'' \cup L'' = P_L$ and $\delta_P(P_L) \leq \kappa$, and finally
- $\delta_P(L'' \cup T'') \leq \kappa$, since $L'' \cup T'' = L \cup T$ and $\delta_P(L \cup T) \leq \kappa$.

The correctness of the Algorithm therefore follows inductively.

Complexity: Let n_X denote the number of vertices of the curve X, where $X \in \{T, B, T_L, T_R, B_L, B_R\}$. We have

$$\begin{aligned} n &= n_T + n_B = n_{T_L} + n_{T_R} + n_{B_L} + n_{B_R} = 2n_{T_L} + n_{B_L} + n_{B_R}, \text{ and} && (5.1) \\ n/2 &\leq n_T = n_{T_L} + n_{T_R} = 2n_{T_L} = 2n_{T_R}. && (5.2) \end{aligned}$$

Observe that in order to decide if $\delta_{P_L}(T_L, B_L) \leq \kappa$ and $\delta_{P_R}(T_R, B_R) \leq \kappa$ in Steps 3 and 4 of Algorithm *TB_Closed_Curve_Detour* we can use the result of Lemma 5.12, since

$$\begin{aligned} \mathsf{max}(\delta_{P_L}(T_L), \delta_{P_L}(B_L)) &\leq \kappa, \text{ and} \\ \mathsf{max}(\delta_{P_R}(T_R), \delta_{P_R}(B_R)) &\leq \kappa. \end{aligned}$$

This follows, since (as noted before) $\mathrm{d}_P(u, v) = \mathrm{d}_{P_L}(u, v)$ for $u, v \in P_L$, and therefore $\delta_{P_L}(T_L) = \delta_P(T_L) \leq \delta_P(T \cup B) \leq \kappa$. The other inequalities can be deduced in the same way. So the costs of Steps 3 and 4 are $\mathcal{O}((n_{T_L} + n_{B_L}) \log(n_{T_L} + n_{B_L}))$, and $\mathcal{O}((n_{T_R} + n_{B_R}) \log(n_{T_R} + n_{B_R}))$, respectively. Therefore the runtime $T(n)$ of Algorithm *TB_Closed_Curve_Detour* obeys the following recursion:

$$\begin{aligned} T(n) &\leq T(n_{T_L} + n_{B_R}) + T(n_{T_R} + n_{B_L}) + \\ &\quad c(n_{T_L} + n_{B_L}) \log(n_{T_L} + n_{B_L}) + \\ &\quad c(n_{T_R} + n_{B_R}) \log(n_{T_R} + n_{B_R}) \\ &\leq T(n_{T_L} + n_{B_R}) + T(n_{T_R} + n_{B_L}) + cn \log n \end{aligned}$$

for some constant $c > 0$. The size of each subproblem is reduced to at least $3/4$, since (5.1) and (5.2) imply that

$$n/4 \leq n_{T_L} \leq n_{T_L} + n_{B_R} = n_{T_L} + n - 2n_{T_L} - n_{B_L} = n - n_{T_L} - n_{B_L} \leq 3n/4 - n_{B_L} \leq 3n/4,$$

and the total size of the subproblems remains the same at each level of the recursion. Thus the recursion tree has logarithmic depth and the total work amounts to

$$T(n) = \mathcal{O}(n \log^2 n).$$

□

Lemma 5.15 (Detour of a closed polygonal curve). *Let $P \in \mathcal{K}^1$ be a simple closed polygonal curve on n vertices, and let $\kappa \geq 1$. Then one can decide in $\mathcal{O}(n \log^2 n)$ time whether $\delta(P) \leq \kappa$.*

Proof. In a preprocessing step we pick an arbitrary vertex P_0 on P, and handle P as if it was an *open* polygonal curve P' with starting and ending point P_0; we traverse P' in $\mathcal{O}(n)$ time, starting from that vertex, and store in each vertex v the distance $\mathrm{d}_{P'}(P_0, v)$, together with the total arclength of P. This will enable us to determine the distance on P between any pair of vertices on P, and the distance between any pair of vertices on a subcurve of P in $\mathcal{O}(1)$ time (this is required in order to apply Lemma 5.14).

Now we proceed as follows: First we split P into four consecutive pieces $P_1, \ldots, P_4$ of equal arclength. Since $\delta(P) = \mathsf{max}_{1 \leq i \leq j \leq 4}(\delta_P(P_i, P_j))$ we have that $\delta(P) \leq \kappa \iff \delta_P(P_i, P_j) \leq \kappa$ for all $1 \leq i \leq j \leq 4$.

Next we define four overlapping polygonal curves $Q_i := P_i \cup P_{i+1 \pmod 4}$ for $1 \leq i \leq 4$. Observe that $\mathrm{d}_P(u, v) = \mathrm{d}_{Q_i}(u, v)$ for $u, v \in Q_i$, and

$$\delta(Q_i) = \mathsf{max}(\delta_P(P_i), \delta_P(P_{i+1}), \delta_P(P_i, P_{i+1})) \leq \delta_P(P)$$

for all $1 \leq i \leq 4$. Now we check if $\delta(Q_i) \leq \kappa$ for $1 \leq i \leq 4$ with the algorithm from Lemma 5.13. If this is not the case we are done, since $\delta_P(P) \geq \delta(Q_i) > \kappa$. Otherwise we know that $\delta_P(P_i) \leq \kappa$ and $\delta_P(P_i, P_{i+1}) \leq \kappa$ for $1 \leq i \leq 4$, so it remains to verify whether $\delta(P_i, P_{i+2}) \leq \kappa$ for $i = 1, 2$. This can be decided in $\mathcal{O}(n \log^2 n)$ time by running the algorithm from the proof of Lemma 5.14 twice. □