Chapter 3

Measuring the Hausdorff distance of point sets in $\mathbb{R}^d$

In this chapter we will develop efficient algorithms to measure the one-sided Hausdorff distance of a d-dimensional point set P to a set Q of n geometric objects of constant 'size' each. To be more precise, in section 3.1, we look at the case where Q is a set of n *semialgebraic* sets in $\mathbb{R}^d$, each of constant *description complexity*, i.e., we look at the following problem:

Problem 3.1 ($\tilde{\delta}_H$-measure problem: point set vs. semialgebraic set in $\mathbb{R}^d$).

Given *a point set $P \subseteq \mathbb{R}^d$ of m points, and a set $Q \subseteq \mathbb{R}^d$ of n semialgebraic sets, each of constant description complexity.*

Compute $\tilde{\delta}_H(P, Q)$.

Of course one can always compute the distance of each point $\mathbf{x} \in P$ to each set $\mathbf{y} \in Q$ in $\mathcal{O}(1)$ time (c.f., Lemma 3.3 on the next page) and thus solve the problem in $\mathcal{O}(mn)$ time; we will refer to this algorithm as the 'brute-force approach'. We will present a randomized algorithm that computes $\tilde{\delta}_H(P, Q)$ in $\mathcal{O}_\epsilon(mn^\epsilon \log m + m^{1+\epsilon-\frac{1}{2d-2}}n)$ expected time (c.f., Theorem 3.8 on page 20). This is – to the best of our knowledge – the first result that constitutes an improvement upon the brute-force approach.

3.1 The one-sided Hausdorff distance of a point set to a semialgebraic set

In this section we look at the problem of computing the one-sided Hausdorff distance from a set of m points P in $\mathbb{R}^d$ to a set Q of n semialgebraic sets of constant description complexity each (c.f., page 8). We will first consider the corresponding *decision problem*, and apply a randomized technique afterwards to *compute* $\tilde{\delta}_H(P, Q)$. The following result will turn out to be a crucial ingredient in our algorithms:

Theorem 3.2 (Point location among real-algebraic varieties, Chazelle et al., [25]). Let T be a set of k d-variate polynomials of bounded degree. A data structure of size $\mathcal{O}_\epsilon(k^{2d-2+\epsilon})$

that allows $\mathcal{O}(\log k)$ time queries among the varieties defined by the set T can be built in $\mathcal{O}_\epsilon(k^{2d-2+\epsilon})$ randomized expected time.

We first describe how to determine $\tilde{\delta}_H(P,Q)$ by 'brute-force' in $\mathcal{O}(mn)$ time by computing all pairwise distances from the points in P to the sets in Q. Later we will use this result as a subroutine in a randomized reduction that turns the algorithm for the decision problem into an algorithm that computes $\tilde{\delta}_H(P,Q)$.

Lemma 3.3 (Computing $\tilde{\delta}_H(P,Q)$ by brute-force). *We can compute $\tilde{\delta}_H(P,Q)$ in $\mathcal{O}(mn)$ time.*

Proof. It suffices to show how the distance $\mathrm{d}(\mathbf{p},\Gamma)$ between a semialgebraic set $\Gamma \in Q$ of constant description complexity and a point $\mathbf{p} \in P$ can be computed. The claim then follows, since we can compute $\tilde{\delta}_H(P,Q)$ by simply computing the mn distances from the points in P to the sets in Q.

Let $\Gamma(\mathbf{x})$ be a polynomial expression that defines Γ. Consider the following Tarski sentence[a]:

$$\mathsf{HD}_{\Gamma,\mathbf{p}}(z) := \forall\epsilon\exists\mathbf{y}\left(\Gamma(\mathbf{y}) \wedge ||\mathbf{p}-\mathbf{y}||^2 \leq z^2+\epsilon^2\right).$$

When $\delta \in \mathbb{R}$ satisfies $\mathsf{HD}_{\Gamma,\mathbf{p}}$, this means that the distance between $\mathbf{p}$ and Γ is at most $|\delta|$. We can transform $\mathsf{HD}_{\Gamma,\mathbf{p}}$ to prenex form and eliminate quantifiers with the algorithm of Collins, c.f., [30] to get a polynomial expression which we will also denote by $\mathsf{HD}_{\Gamma,\mathbf{p}}$. The runtime of this algorithm is doubly-exponential in d and polynomial in the number of polynomials forming the expression $\Gamma(\mathbf{x})$.

The expression $\mathsf{HD}_{\Gamma,\mathbf{p}}$ defines a semialgebraic set in $\mathbb{R}$, i.e., a finite set of intervals of constant size, which we can compute in $\mathcal{O}(1)$ time with the algorithm of Theorem 3.2. From that set we can read off the smallest $\delta \geq 0$ for which $\mathsf{HD}_{\Gamma,\mathbf{p}}$ holds and return it as $\mathrm{d}(\mathbf{p},\Gamma)$.

Since the description complexity of Γ (and therefore of all other semialgebraic sets derived from it) is independent of the input size (i.e., m and n), we can compute $\mathrm{d}(\mathbf{p},\Gamma)$ in $\mathcal{O}(1)$ time (where the hidden constant depends doubly-exponential on d), and the claimed time bound follows. □

As we already mentioned, we will first consider the decision version of Problem 3.1:

Problem 3.4 ($\tilde{\delta}_H$-decision problem: point set vs. semialgebraic set in $\mathbb{R}^d$).

Given *a point set $P \subseteq \mathbb{R}^d$ of m points, and a set $Q \subseteq \mathbb{R}^d$ of n semialgebraic sets, each of constant description complexity, and some $\delta \geq 0$.*

Decide, *whether $\tilde{\delta}_H(P,Q) \leq \delta$.*

We have that $\tilde{\delta}_H(P,Q) \leq \delta$ iff for each point in P there is a point of Q that is δ-close. Therefore it is reasonable to look at the set of all points that are δ-close to Q:

[a] A *Tarski sentence* consists of a polynomial expression that is prefixed by a finite number of existential ($\exists$) and universal ($\forall$) quantifiers.

Definition 3.5 (δ-neighborhood). *Let Q be a compact set in $\mathbb{R}^d$. Then $\mathsf{nh}_\delta(Q)$ denotes the* δ-neighborhood *of Q, defined as*

$$\mathsf{nh}_\delta(Q) := \{x \in \mathbb{R}^d \mid \mathrm{d}(x, Q) \leq \delta\}.$$

Our result is based on the following simple observation:

Observation 3.6. *Let and P, Q be compact sets in $\mathbb{R}^d$, and $\delta > 0$. Then the one-sided Hausdorff distance from P to Q is at most δ iff all points of P are contained in the δ-neighborhood of Q, i.e.,*

$$\tilde{\delta}_H(P, Q) \leq \delta \iff P \subseteq \mathsf{nh}_\delta(Q).$$

The algorithm for the decision problem computes from Q a data structure that represents $\mathsf{nh}_\delta(Q)$ and allows efficient point-containment queries. Then it queries this data structure with all points of P and determines all points that are not contained in the δ-neighborhood – this will be needed to perform the randomized reduction that solves the optimization problem.

Lemma 3.7 (Computing $\tilde{\delta}_H(P, Q)$ – decision problem). *We can compute the set $X = \{\mathbf{x} \in P \mid \mathrm{d}(\mathbf{x}, Q) > \delta\}$ in $\mathcal{O}_\epsilon(mn^\epsilon + m^{1+\epsilon-1/(2d-2)}n)$ randomized expected time.*

Proof. We first describe a randomized algorithm that computes X in $\mathcal{O}_\epsilon(n^{2d-2+\epsilon} + m \log n)$ expected time, which we speed up with a simple batching technique afterwards.

Let us first argue that for $\Gamma \in Q$ the set $\mathsf{nh}_\delta(\Gamma)$ is semialgebraic and that it (i.e., a polynomial expression defining it) can be computed in $\mathcal{O}(1)$ time. To this end, let $Q(\mathbf{x})$ be a polynomial expression that defines Γ and consider the following Tarski sentence:

$$\mathsf{NH}_{\Gamma,\delta}(\mathbf{x}) := \exists \mathbf{y} \left(Q(\mathbf{y}) \wedge ||\mathbf{x} - \mathbf{y}||^2 \leq \delta^2 \right).$$

Obviously $\mathsf{nh}_\delta(\Gamma) = \{\mathbf{x} \in \mathbb{R}^d \mid \mathsf{NH}_{\Gamma,\delta}(\mathbf{x}) \text{ holds}\}$. We can eliminate the quantifier to obtain a polynomial expression which we will also denote by $\mathsf{NH}_{\Gamma,\delta}$; it will be identified with the sequence $(n^{(i)}_{\Gamma,\delta}(\mathbf{x}))_{1 \leq i \leq b}$ of d-variate polynomials that form it. Since the description complexity of Γ is independent of the input size, the set $\mathsf{NH}_{\Gamma,\delta}$ (i.e., the polynomials $n^{(i)}_{\Gamma,\delta}$) can be computed in $\mathcal{O}(1)$ time.

Let $F = \cup_{\Gamma \in Q} \{n^{(i)}_{\Gamma,\delta}\}$ denote the set of polynomials that appear in the atomic polynomial expressions forming the expressions $\mathsf{NH}_{\Gamma,\delta}$. By the above reasoning this set can be computed in $\mathcal{O}(n)$ time. With the algorithm of Theorem 3.2 we can compute a point-location data structure of size $\mathcal{O}_\epsilon(n^{2d-2+\epsilon})$ in $\mathcal{O}_\epsilon(n^{2d-2+\epsilon})$ time for the arrangement of the varieties $n^{(i)}_{\Gamma,\delta} = 0$ defined by F. The signs of all polynomials in F and therefore the validity of each polynomial expression $\mathsf{NH}_{\Gamma,\delta}$ is constant for each cell of the decomposition of $\mathbb{R}^d$ induced by these varieties. Thus a point-location query to this data structure determines whether the query point lies in $\mathsf{nh}_\delta(Q)$ and the set X can be computed by querying the data structure with all points in P. The overall runtime of this method is $\mathcal{O}_\epsilon(n^{2d-2+\epsilon} + m \log n)$.

We gain a significant speedup with a simple batching technique. To this end distinguish the following cases:

$m \leq n^{2d-2}$: We partition Q into $g = \lceil n/m^{1/(2d-2)} \rceil$ groups of at most $k = m^{1/(2d-2)} \leq n$ points each. For each group, we run the algorithm described above. The total time spent is

$$\mathcal{O}_\epsilon(g(k^{2d-2+\epsilon} + m \log k)) = \mathcal{O}_\epsilon(m^{1+\epsilon-1/(2d-2)} n).$$

$n^{2d-2} \leq m$: In that case the runtime is $\mathcal{O}(mn^\epsilon)$.

□

Theorem 3.8 (Computing $\tilde{\delta}_H(P,Q)$). *We can compute $\tilde{\delta}_H(P,Q)$ in $\mathcal{O}_\epsilon(mn^\epsilon \log m + m^{1+\epsilon-\frac{1}{2d-2}} n)$ randomized expected time.*

Proof. We follow a strategy similar to that proposed in [2]. Initially we set $\delta = 0$ and $X = P$. Then we repeat the following steps until X becomes empty:

Choose a random point $\mathbf{x} \in X$ and compute $\delta' = \tilde{\delta}_H(\mathbf{x}, Q)$ in $\mathcal{O}(n)$ time with the algorithm from Lemma 3.3. Set δ to $\mathsf{max}(\delta, \delta')$. Now compute the set $X' = \{\mathbf{x} \in X \mid \tilde{\delta}_H(\mathbf{x}, Q) > \delta\}$ in $\mathcal{O}_\epsilon(mn^\epsilon + m^{1+\epsilon-1/(2d-2)} n)$ time with the algorithm from Lemma 3.7. Finally set X to X'.

Obviously the last value of δ will be $\tilde{\delta}_H(P,Q)$. As is shown in [28], the expected number of iterations is $\mathcal{O}(\log m)$ and therefore the expected time to compute $\tilde{\delta}_H(P,Q)$ with this algorithm is $\mathcal{O}_\epsilon(mn^\epsilon \log m + m^{1+\epsilon-1/(2d-2)} n)$. □