Part I

Point set pattern matching in d-dimensional space

A first and natural approach to model geometric patterns is to represent them by point sets in d-dimensional Euclidean space. Needless to say that the cases $d = 2, 3$ are the most prominent ones; in fact geometric pattern matching problems for planar and spatial point sets have received considerable attention in the literature, see, e.g., the survey by Alt and Guibas [14] and the references therein. However, although probably less interesting from a practical point of view, in this part we will investigate two matching problems for point sets in *higher dimensions*.

First, in chapter 2, we consider the question, whether two point sets P and Q, with m and n points ($m \leq n$), respectively, in d-dimensional Euclidean space are congruent, i.e., if there exists a rigid motion μ with $\mu(P) = Q$. Recall that a rigid motion is obtained by combining a translation with a rotation and (possibly) a reflection. The congruence testing problem can be seen as a special case of the general pattern matching problem described in the introduction, where the distance measure is the *disrcete metric* $\mathrm{d}_{\mathsf{discr}}$, with $\mathrm{d}_{\mathsf{discr}}(P, Q) = 0$ if $P = Q$ and $\mathrm{d}_{\mathsf{discr}}(P, Q) = 1$ otherwise, and the set of admissible transformations is the set of rigid motions of $\mathbb{R}^d$.

We present an algorithm for the d-dimensional congruence test problem that runs in $O(n^{\lceil d/3 \rceil} \log n)$ time (c.f., Theorem 2.2 on page 13). The exponential dependence on d is somewhat unsatisfactory, since the best known lower bound (which already holds in dimension one) is $\Omega(n \log n)$, but some dimension-dependence is unavoidable, for the congruence testing problem without restriction on the dimension is NP-hard as is show in [7].

Obviously the discrete metric is extremely sensitive to noise and omissions, and therefore congruity is usually a too strong notion to assess the similarity of point patterns, especially in practical applications where the patterns arise from appropriately sampled real world data. The *Hausdorff distance* is a commonly used similarity measure for geometric patterns that circumvents these problems (at least to some extent); for two sets P and Q it is the smallest δ, such that P is completely contained in the δ-neighborhood of Q, and vice versa:

Definition 1.4 (Hausdorff distance, one-sided Hausdorff distance). *Let P and Q be compact sets in $\mathbb{R}^d$, and $||z||$ denote the Euclidean norm of $z \in \mathbb{R}^d$. Then $\delta_H(P, Q)$ denotes the* Hausdorff distance *between P and Q, defined as*

$$\delta_H(P,Q) := \mathsf{max}\left(\tilde{\delta}_H(P,Q), \tilde{\delta}_H(Q,P)\right), \textit{ with}$$
$$\tilde{\delta}_H(P,Q) := \max_{x \in P} \min_{y \in Q} ||x - y||, \textit{ the } \text{one-sided Hausdorff distance } \textit{from } P \textit{ to } Q.$$

Intuitively speaking the 'pattern' P has a small one-sided Hausdorff distance to Q if it is 'similar' to a 'subpattern' of Q.

In chapter 3 we will present an efficient algorithm to measure the one-sided Hausdorff distance of a d-dimensional m-point set P to a set Q of n geometric objects of constant 'size' each. As we already noted in the introduction this also can be seen as a special case of the general pattern matching problem, where the set of admissible transformations consists of the identity only.

To be more precise we look at the case where Q is a set of n *semialgebraic* sets in $\mathbb{R}^d$, each of constant *description complexity*. We develop an algorithm to compute $\tilde{\delta}_H(P, Q)$ in $\mathcal{O}_\epsilon(mn^\epsilon \log m + m^{1+\epsilon-\frac{1}{2d-2}} n)$ randomized time (c.f., Theorem 3.8 on page 20).

Recall that a set $S \subseteq \mathbb{R}^d$ is called semialgebraic if it satisfies a *polynomial expression*, which is any finite boolean combination of *atomic polynomial expressions*, which in turn are of the form $P(\mathbf{x}) \leq 0$, where $P \in \mathbb{R}[x_1, \dots, x_d]$ is a d-variate polynomial.

The *description complexity* of a polynomial expression $\mathcal{B}$ involving the polynomials $P_1, \dots, P_N$ is the length of an encoding of that expression over a fixed finite alphabet *disregarding* the length of the encoding of the coefficients of the P_i (which we can afford, since we work in the unit-cost model anyway). The description complexity of a semialgebraic set is the minimum description complexity of an expression defining that set. When we talk about algorithms that work on a set of n semialgebraic sets each of constant description complexity in time $\mathcal{O}(T(n))$, we actually mean that for each constant $C > 0$ the runtime of these algorithms on a set of n semialgebraic sets each of description complexity at most C is $\mathcal{O}(T(n))$; the constant hidden in the $\mathcal{O}$-notation may depend on C.

The results in this part have partially been obtained in collaboration with Peter Braß. Some of the material has already been published in [24].