Parallel Algorithms for Shape Matching

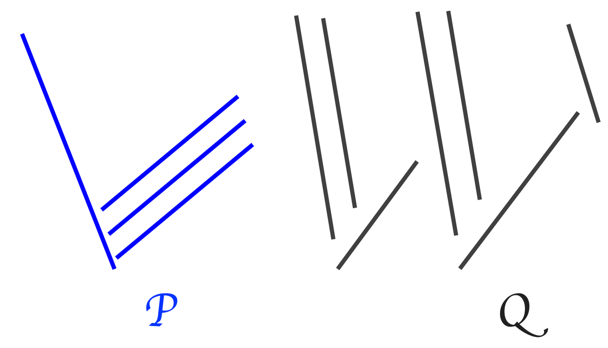
Helmut Alt, Ludmila Scharf

Freie Universität Berlin

Computing the Hausdorff Distance between Sets of Line Segments

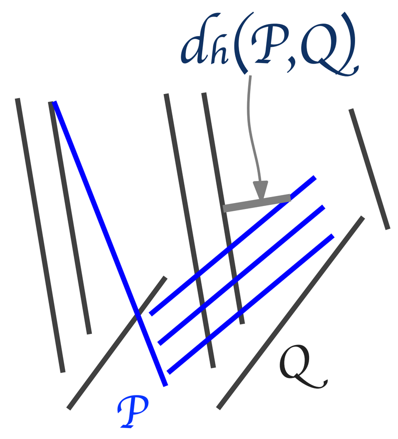
Problem

Given two shapes $\mathcal{P}$ and $\mathcal{Q}$ as sets of line segments, determine their (dis-)similarity with respect to Hausdorff distance.



Definition – Hausdorff Distance

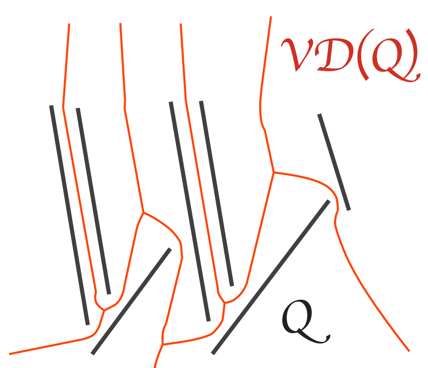
The (directed) **Hausdorff distance** between two sets of line segments $\mathcal{P}$ and $\mathcal{Q}$ is $d_h(\mathcal{P},\mathcal{Q}) = max_p\ min_q\ d(p, q)$, where $d(p, q)$ denotes the Euclidean distance between two points $p \in \mathcal{P}$ and $q \in \mathcal{Q}$.



Definition – Voronoi Diagram

Given a set $\mathcal{Q}$ of geometric objects (points, line segments, etc.), the **Voronoi cell** of an object $q \in \mathcal{Q}$ is the set of points in the plane, that are closer to q than to any other object in $\mathcal{Q}$.

The **Voronoi diagram** of $\mathcal{Q}$ ($\mathcal{VD}(\mathcal{Q})$) is the set of boundaries of the Voronoi cells of all objects in $\mathcal{Q}$.



Theorem

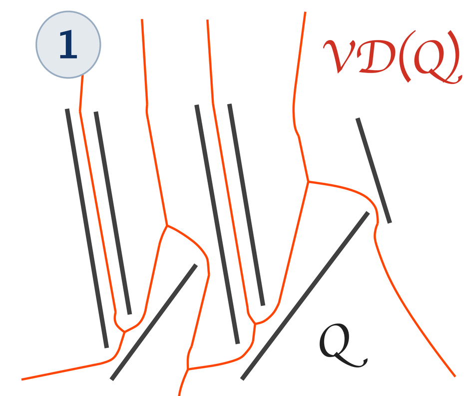
Given two sets $\mathcal{P}$ and $\mathcal{Q}$ of n line segments, such that no two segments of the same set intersect, except possibly at the endpoints, the Hausdorff distance $d_h(\mathcal{P},\mathcal{Q})$ can be computed in $O(log^2 n)$ time on $O(n)$ processors using $O(n\ log\ n)$ storage in the CREW-PRAM model.

Algorithm

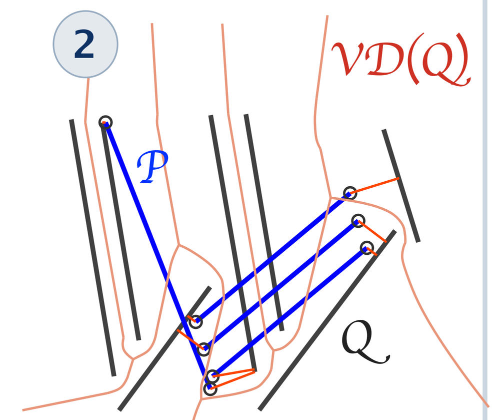
Sequential version due to [ABB95].

Observation: The Hausdorff distance from $\mathcal{P}$ to $\mathcal{Q}$ is attained either at an endpoint of $\mathcal{P}$ or at an intersection point of $\mathcal{P}$ and the Voronoi diagram of $\mathcal{Q}$.

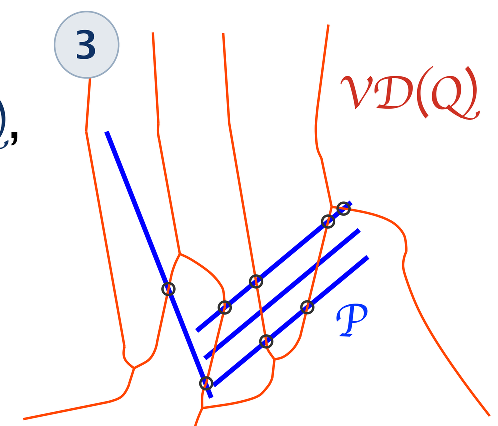
1. Construct the Voronoi diagram $\mathcal{VD}(\mathcal{Q})$ of $\mathcal{Q}$. (parallel construction in [GDY93])



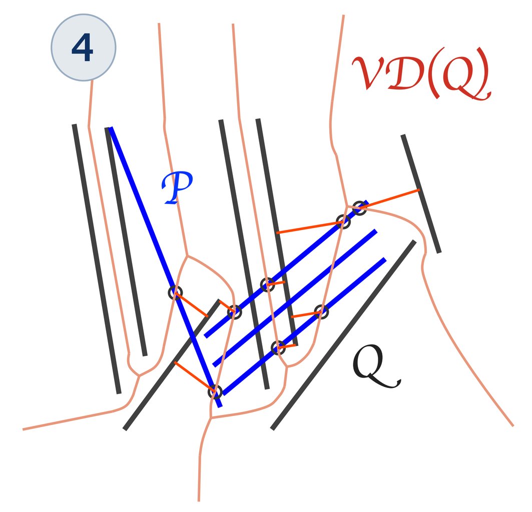
2. For each endpoint p of a segment in $\mathcal{P}$ find its closest segment in $\mathcal{Q}$ using $\mathcal{VD}(\mathcal{Q})$ and compute the distance from p to that segment. (parallel point location in [TV89])



3. Determine the "critical points" on the edges of $\mathcal{VD}(\mathcal{Q})$, the intersection points with $\mathcal{P}$ with the highest and the lowest x-coordinate. (parallel computation is the contribution of this work)



4. For each critical point q compute the distance from q to its nearest segment in $\mathcal{Q}$.



5. Return the maximal distance of endpoints and critical points.

References

[ABB95] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. Annals of Mathematics and Artificial Intelligence, 13; 251–265, 1995.

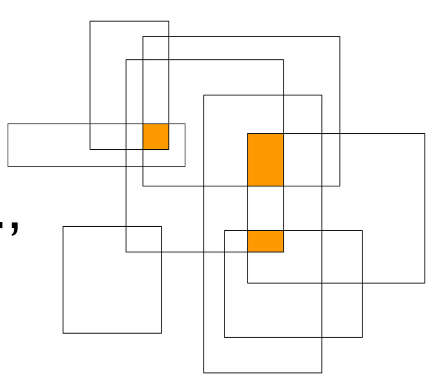
[GDY93] M. T. Goodrich, C. O'Dunlaing, and C.-K. Yap. Constructing the Voronoi diagram of a set of line segments in parallel. Algorithmica, 9(2):128–141, 1993.

[TV89] R. Tamassia and J. S. Vitter. Optimal parallel algorithms for transitive closure and point location in planar structures. SPAA '89, 399–408, 1989. ACM.

Computing the depth of an arrangement of axis-parallel rectangles

Definition

Given a set of n axis-parallel rectangles, find the **depth of their arrangement**, i.e., the maximum number of the rectangles containing a common point.

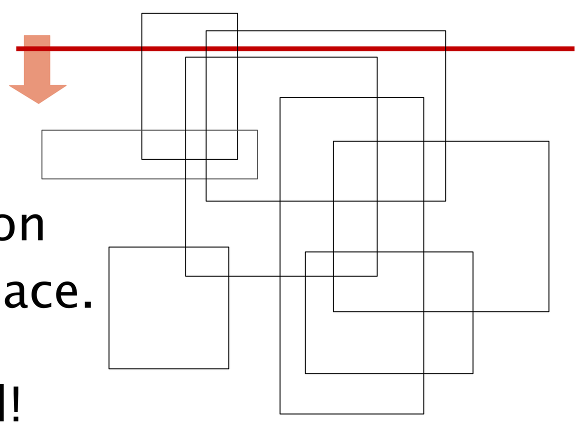


Application: e.g., find a transformation maximizing the similarity of two geometric shapes.

Sequential Algorithm

Use a **plane sweep** technique to process all rectangles and find the maximum number sharing a common point in time $O(n\ log\ n)$ using $O(n)$ space.

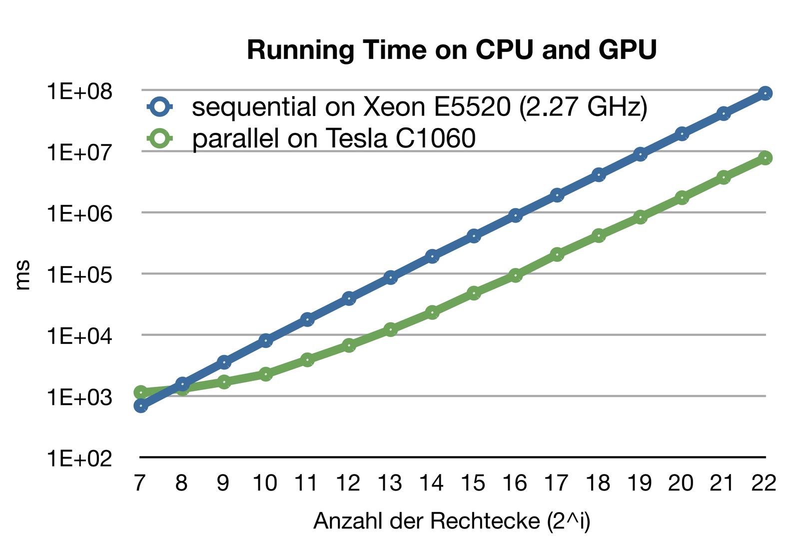
Plane sweep is inherently sequential!

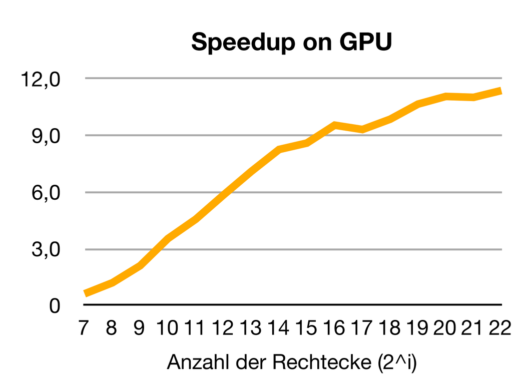


Sequential running time is a "folklore" knowledge based on [Ben77].

Implementation

Proof-of-concept implementation with OpenCL





Parallel Algorithm

Model: **E**xclusive **R**ead **E**xclusive **W**rite **PRAM** – a shared memory RAM with problem size dependent number of processors.

Idea:

- Construct a segment tree for all horizontal rectangle sides in parallel.
- Store the information about **all** sweep events for each node in a **history list** of that node.
- Process all sweep events for each level of the tree in parallel.

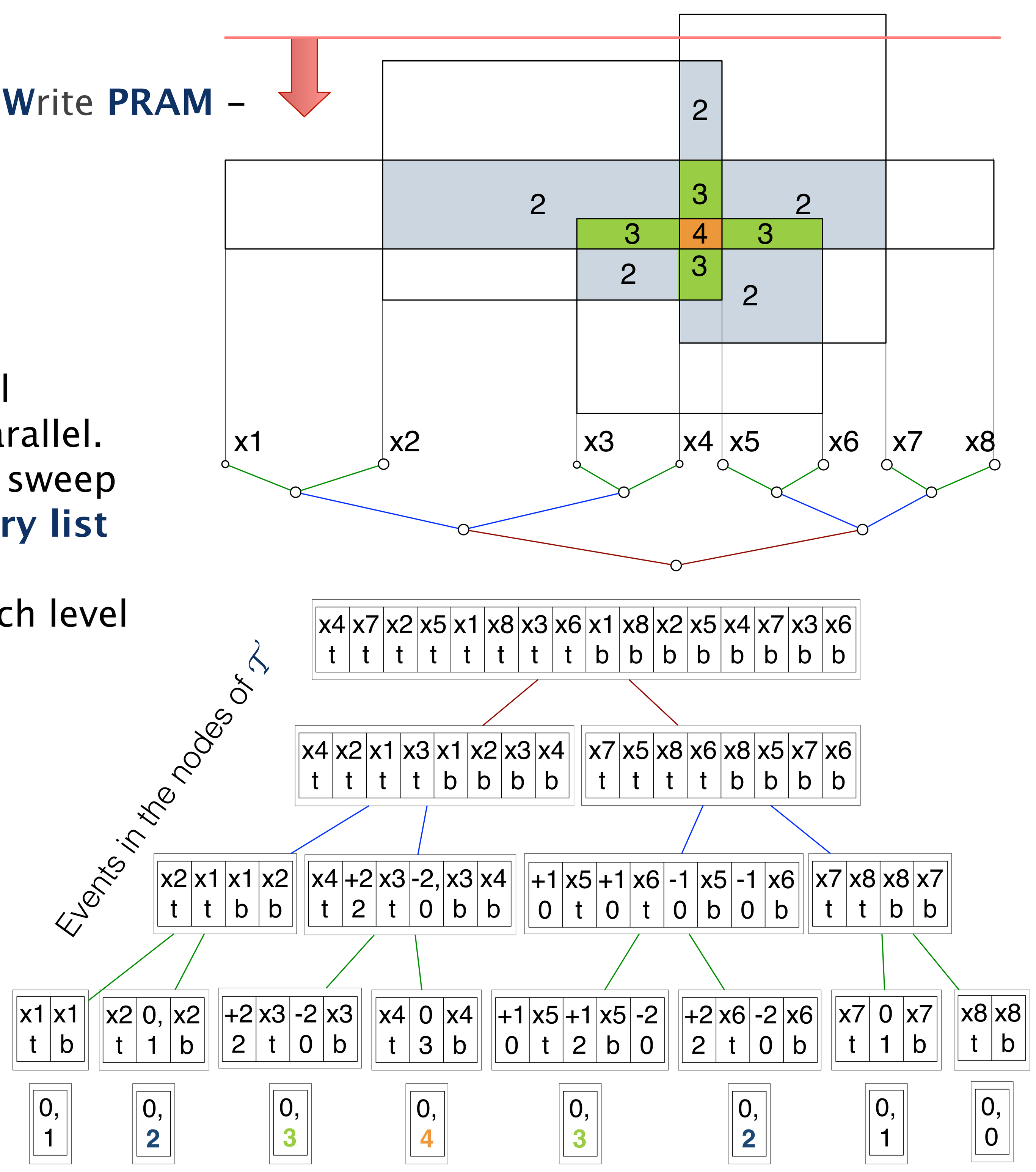
Performance:

Time $O(log^2 n)$

Processors $O(n)$

Space $O(n)$.

Randomized time $O(log\ n\ log^* n)$ with high probability.



References

Preliminary version of this work is published in:

[AS09] H. Alt, L.Scharf. Computing the depth of an arrangement of axis-aligned rectangles in parallel. EuroCG, 2010.

[Ben77] J.L. Bentley. Algorithms for Klee's rectangle problems. Unpublished notes, 1977.