Parametric Search Visualization

Ivaylo Ilinkin
Department of Computer Science
Gettysburg College
Gettysburg, PA 17325, USA
iilinkin@gettysburg.edu

ABSTRACT

This video provides a visualization of parametric search using as the underlying application an algorithm for computing a tangent to a level in arrangement proposed by Matoušek [2]. The visualization is based on the framework of van Oostrum and Veltkamp [6], which simplifies considerably the implementation of parametric search algorithms.

Categories and Subject Descriptors

I.3.5 [**Computational Geometry and Object Modeling**]: Geometric algorithms, languages, and systems

Keywords

parametric search; framework; arrangements; tangents

1. INTRODUCTION

Parametric search is an optimization technique that has been used to provide efficient algorithms for a wide range of problems [3]. In the parametric search setting one considers a decision problem, $P(\lambda)$, that is monotone in a real-valued parameter λ, i.e. if $P(\lambda') = T$, then $P(\lambda) = T$ for $\lambda < \lambda'$. The goal is to find the largest value, λ^*, for which $P(\lambda) = T$.

Despite its success, algorithms based on parametric search are considered difficult to implement, and therefore, the technique has been primarily of theoretical interest. Indeed, there have been only two reported implementations of parametric search algorithms by Schwerdt [4] and Toledo [5].

The framework of van Oostrum and Veltkamp [6] simplifies considerably implementations of parametric search algorithms. The framework hides the complexity of synchronizing the flow of control and lets the user focus on the high-level aspects of the problem. Our video provides a visualization of a parametric search algorithm in the context of this framework and could inspire further experimental work to validate the practical aspects of parametric search.

The visualization is based on an algorithm proposed by Matoušek [2] for computing a tangent to a level in arrangement. This algorithm is, in fact, part of Matoušek's solution to a larger problem, namely, computing the center of a planar point set, and the overall solution uses multiple applications of parametric search.

SoCG'13, June 17–20, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2031-3/13/06.

2. ALGORITHM

The video provides a visualization of an algorithm that solves the following problem [2]: *Given a collection, H, of lines, a point, x, and an integer, k, compute a tangent, τ^*, through x that touches the k-level in the arrangement of H to the right of x.*[1]

The algorithm uses parametric search to sort the lines in H along the unknown tangent τ^*. It is based on the following decision problem, $P(\tau)$, that exhibits the monotonicity property: *Does the line through x with slope τ intersect the k-level?* (Figure 1).

The execution of the algorithm depends on the efficient implementation and synchronization of two sub-routines:

- *Comparing two lines at τ^**: Even though τ^* is not known, it is possible to decide the order of two lines, ℓ_i and ℓ_j, along τ^* by considering three critical lines through x: τ_i and τ_j, which are parallel to ℓ_i and ℓ_j, respectively, and τ_{ij}, which goes through the intersection point $y = \ell_i \cap \ell_j$. The order of ℓ_i and ℓ_j along τ^* can be decided by considering the location of x relative to ℓ_i and ℓ_j and by determining whether τ_i, τ_j, and τ_{ij} intersect the k-level (Figure 2). The running time is $O(1)$ if the answers to the decision problems $P(\tau_i)$, $P(\tau_j)$, and $P(\tau_{ij})$ are known.

- *Solving the decision problem*: The comparison of two lines along τ^* depends on knowing if a line τ intersects the k-level, i.e. if the answer to $P(\tau) = T$. This can be determined as follows (Figure 3): (1) find the intersection points between τ and the lines in H; (2) sort the intersection points; (3) find the level of the left-most intersection point; (4) walk along τ updating (by at most ± 1) the level of each intersection point. The running time is $O(n \log n)$ and it is dominated by the time to sort the intersection points.

3. SORTING-BASED SEARCH

During the sorting each comparison of lines ℓ_i and ℓ_j along τ^* is potentially expensive since it depends on answering the decision problem for the critical lines τ_i, τ_j, and τ_{ij}. In a parametric search implementation expensive comparisons are batched together and resolved efficiently in a binary search fashion exploiting the monotonicity property. Throughout its execution a parametric search algorithm maintains a progressively smaller interval where τ^*

[1]The level of a point, p, is defined as the number of lines in H that lie below or go through p.

lies, so that the expensive computation for solving the decision problem is run only for lines τ that lie in this interval.

For sorting-based parametric search the framework offers an implementation of *quick sort* based on the observation that it allows for convenient batching of comparisons.[2] To achieve this, *quick sort* is modified so that during the partition step it first computes for each comparison of lines ℓ_i and ℓ_j the critical lines τ_i, τ_j, and τ_{ij}, but does not evaluate the decision problem, yet. Next, the decision problem is answered for the critical line with the median slope and this single expensive comparison makes it possible to deduce the answer for half of the critical lines. Therefore, all of the $O(n)$ comparisons during the partition step can be answered using only $O(\log n)$ expensive evaluations of the decision problem. Thus, the running time per level in *quick sort* is $O(n \log^2 n)$ and $O(n \log^3 n)$ for the entire algorithm over the $O(\log n)$ expected levels.

4. IMPLEMENTATION

The framework of van Oostrum and Veltkamp [6] simplifies the task of implementing parametric search algorithms by requiring that the user provide only a small number of classes (Figure 4):

- *Polynomials class*: This is a representation of the objects that are compared during the search. In our implementation we use CGAL lines [1].
- *Comparison class*: This compares two polynomials at the unknown value τ^*. The framework collects independent comparisons in a single batch to be resolved efficiently once the answers for the associated decision problems are known.
- *Solver class*: This solves the decision problem, $P(\tau)$, which effectively answers, for any given value τ, whether $\tau < \tau^*$, $\tau = \tau^*$, or $\tau > \tau^*$. This is an expensive computation and the framework ensures that a number of decision questions are answered together in binary search fashion exploiting the monotonicity property.

5. REFERENCES

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] J. Matoušek. Computing the center of planar point sets. *Comput. Geom.: Papers from the DIMACS Special Year*, pages 221–230, 1991.

[3] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, Oct. 1983.

[4] J. Schwerdt, M. Smid, and S. Schirra. Computing the minimum diameter for moving points: an exact implementation using parametric search. In *Proc. 13th Annu. ACM Symp. Comput. Geom.*, SCG '97, pages 466–468, New York, NY, USA, 1997. ACM.

[5] S. Toledo. Extremal polygon containment problems and other issues in parametric searching. Master's thesis, Tel Aviv Univ., 1991.

[6] R. van Oostrum and R. Veltkamp. Parametric search made practical. *Comput. Geom. Theory Appl.*, 28(2-3):75–88, June 2004.

[2] An implementation of *bitonic sort* is also provided.

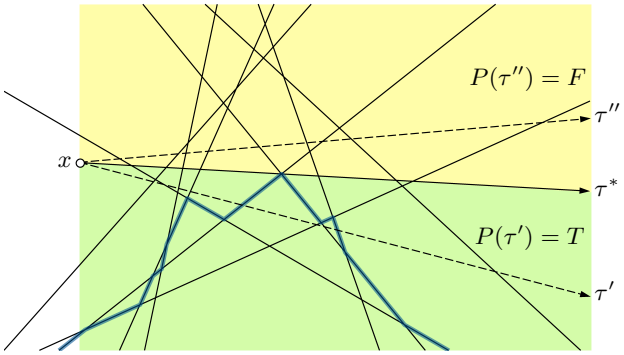

Figure 1: Monotonicity of the decision problem: $P(\tau) = T$ iff $\tau \leq \tau^*$, i.e. line τ intersects the k-level if it lies in the darker shaded region.

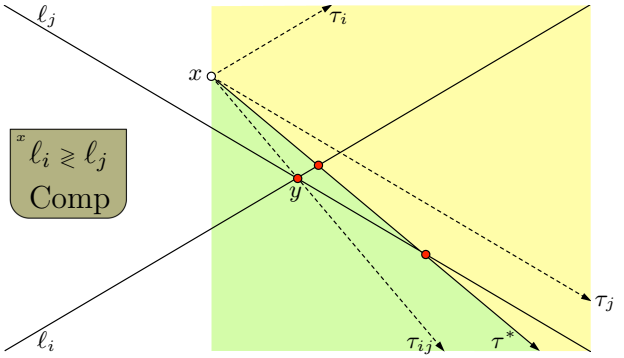

Figure 2: Comparing lines ℓ_i and ℓ_j along τ^* when x lies above ℓ_i and ℓ_j and left of y, and $P(\tau_i) = F$, $P(\tau_j) = F$, and $P(\tau_{ij}) = T$. In this case, $\ell_i < \ell_j$.

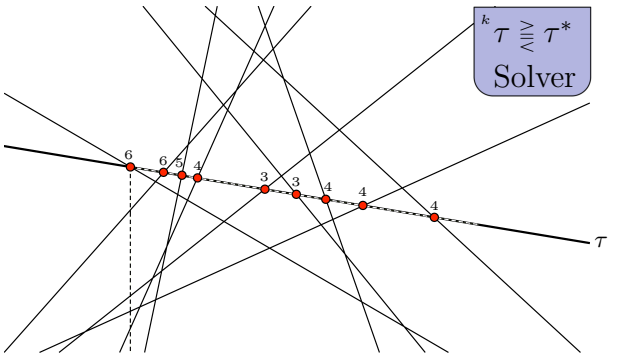

Figure 3: Solving the decision problem for a given line τ — find if an intersection point has level $\leq k$.

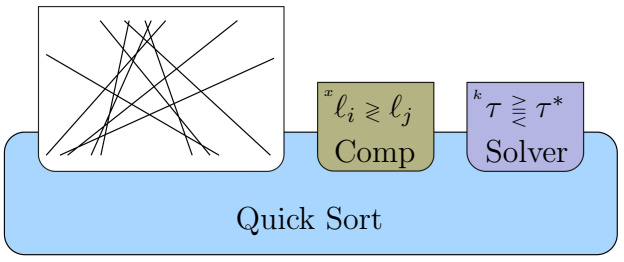

Figure 4: The parametric search framework of van Oostrum and Veltkamp.