

# Algorithms for Comparing Curves and Surfaces

## 2: Other ways of searching the free space

Maïke Buchin

WSCG 2017

# Free Space Diagram

- in 1992 Alt and Godau proposed the *free space diagram* as a means to compute the *Fréchet distance*
- basis for all subsequent algorithms

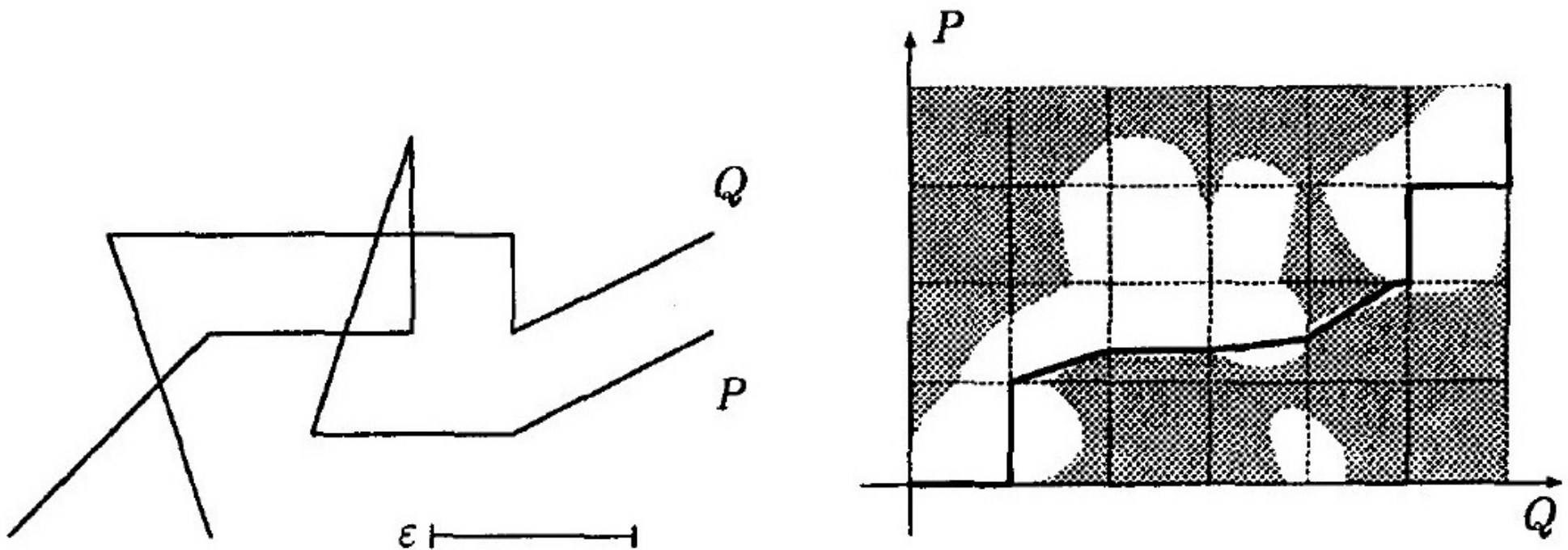


Fig. 3. Diagram for polygonal chains  $P, Q$  and the given  $\epsilon$

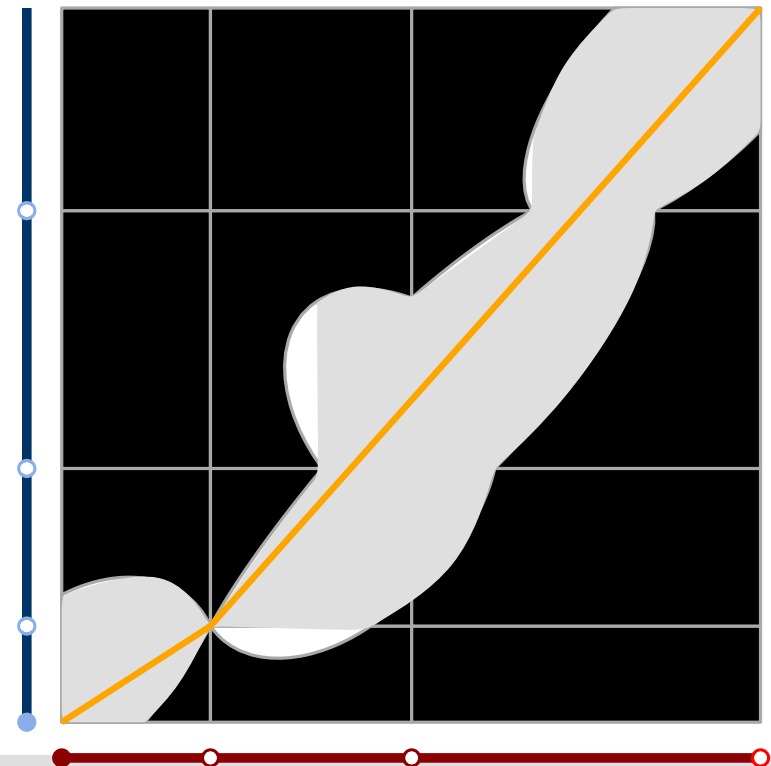
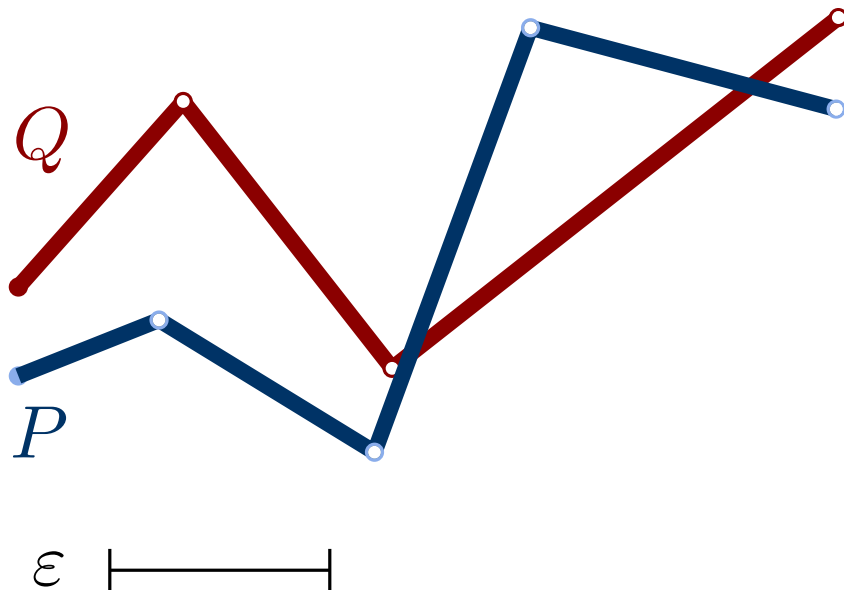
Let  $P, Q$  be two polygonal curves of size  $n$ .

- **decision problem:** Is  $d_F(P, Q) \leq \varepsilon$ ?

Solve by searching for a monotone path in the free space in  $O(n^2)$  time

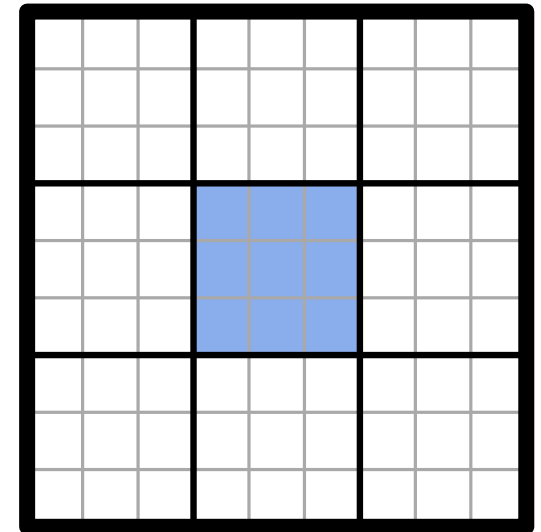
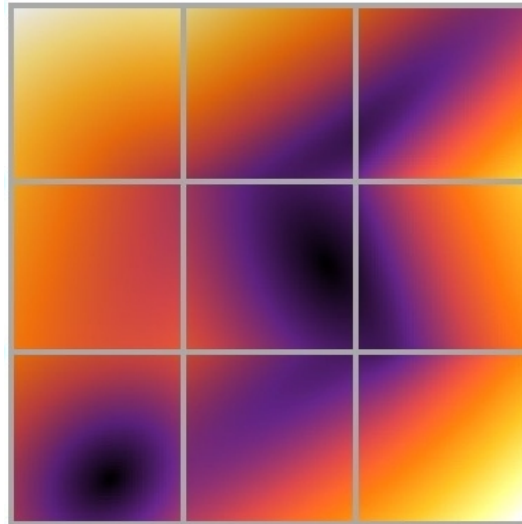
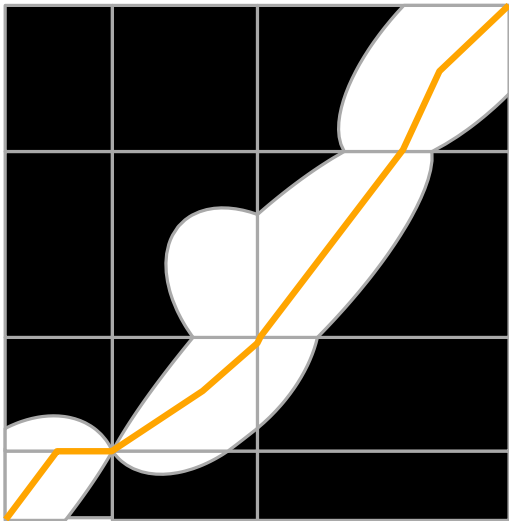
- **computation problem:** What is  $d_F(P, Q)$ ?

Solve by searching over a set of critical values in  $O(n^2 \log n)$  time

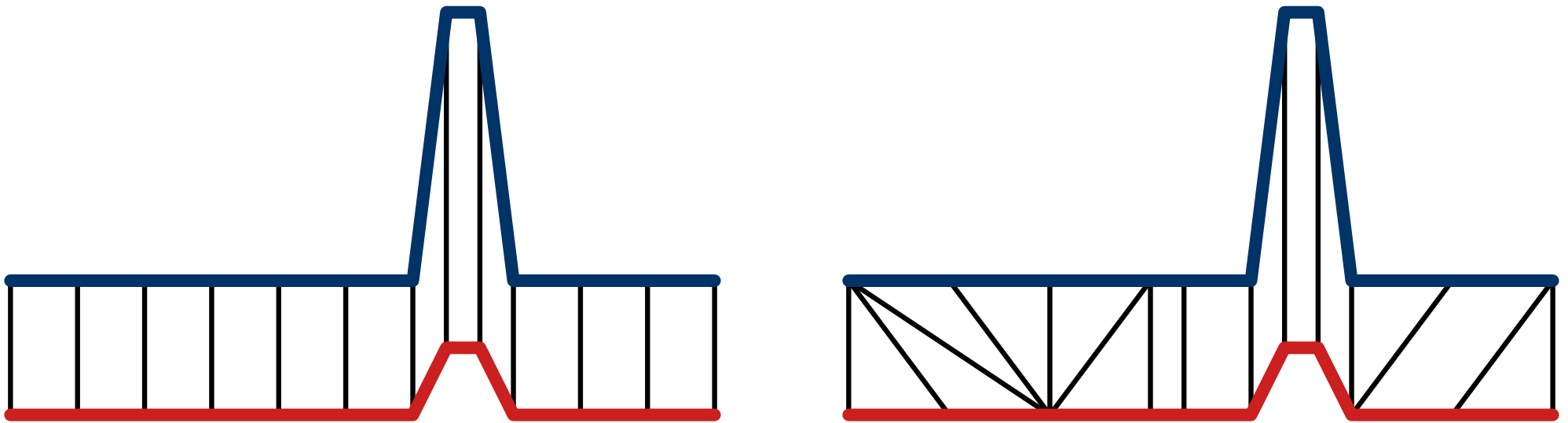


# Searching the Free Space Faster or Better

- locally correct [2012, Buchin et al.]
- lexicographic [2014, Rote]
- retractable [2013, Buchin et al.]
- faster [2014, Buchin et al.]



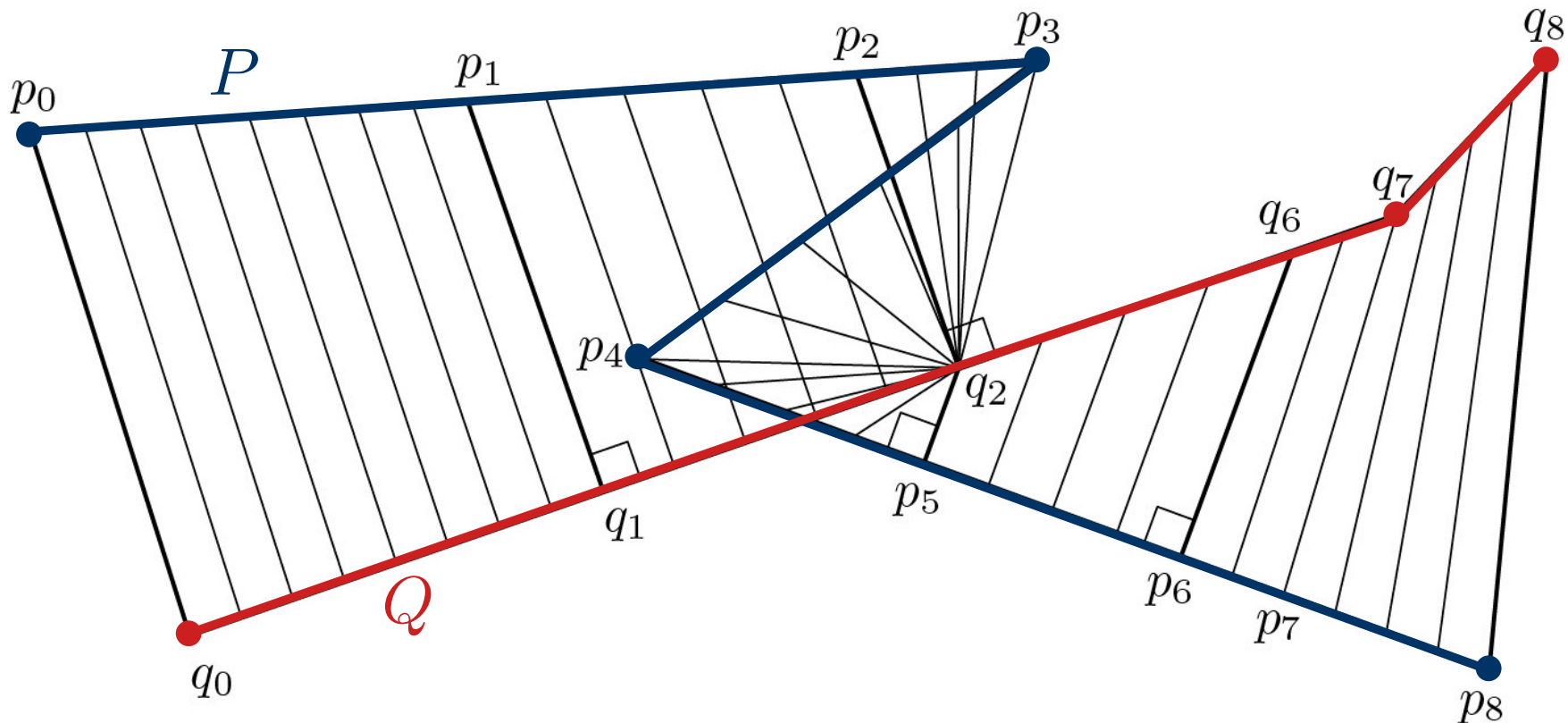
**Weakness:** many matchings may realize the Fréchet distance



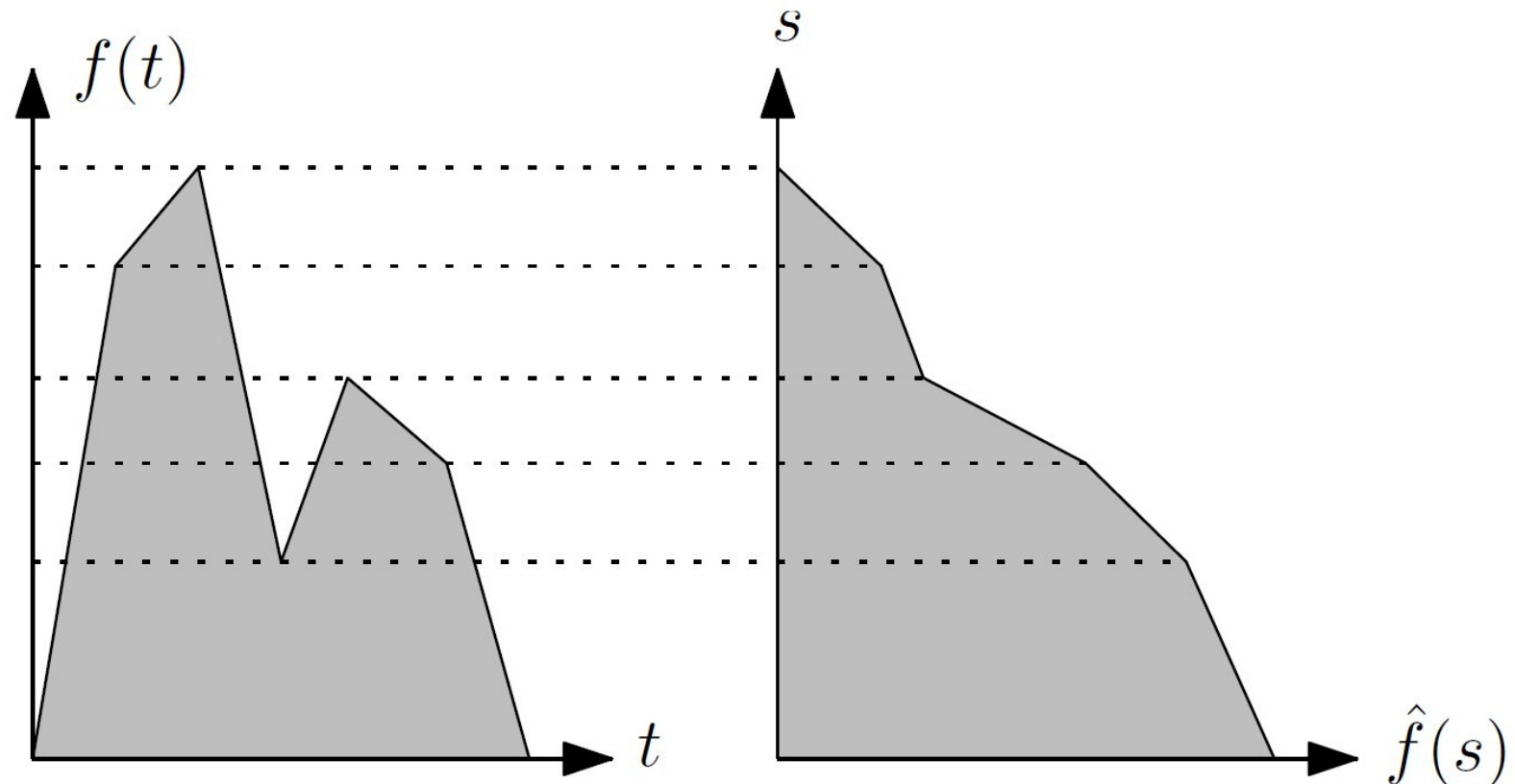
**Goal:** find optimal matchings

**now:** lexicographic Fréchet matchings

- $P: [0, L_P] \rightarrow \mathbb{R}^d$  and  $Q: [0, L_Q] \rightarrow \mathbb{R}^d$  polygonal curves
- reparametrize by  $\alpha: [0, M] \rightarrow [0, L_P]$  and  $\beta: [0, M] \rightarrow [0, L_Q]$
- simultaneous traversal  $(P(\alpha(t)), Q(\beta(t)))$  for  $0 \leq t \leq M$
- Fréchet distance minimizes  $\max_t \|P(\alpha(t)) - Q(\beta(t))\|$



- $P: [0, L_P] \rightarrow \mathbb{R}^d$  and  $Q: [0, L_Q] \rightarrow \mathbb{R}^d$  polygonal curves
- reparametrize by  $\alpha: [0, M] \rightarrow [0, L_P]$  and  $\beta: [0, M] \rightarrow [0, L_Q]$
- now consider  $f: [0, M] \rightarrow \mathbb{R}_{\geq 0}$ ,  $t \mapsto \|P(\alpha(t)) - Q(\beta(t))\|$
- and its profile  $\hat{f}: \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$ ,  $s \mapsto \mu(\{t | f(t) \geq s\})$



- $P: [0, L_P] \rightarrow \mathbb{R}^d$  and  $Q: [0, L_Q] \rightarrow \mathbb{R}^d$  polygonal curves
- reparametrize by  $\alpha: [0, M] \rightarrow [0, L_P]$  and  $\beta: [0, M] \rightarrow [0, L_Q]$
- now consider  $f: [0, M] \rightarrow \mathbb{R}_{\geq 0}$ ,  $t \mapsto \|P(\alpha(t)) - Q(\beta(t))\|$
- and its profile  $\hat{f}: \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}$ ,  $s \mapsto \mu(\{t | f(t) \geq s\})$

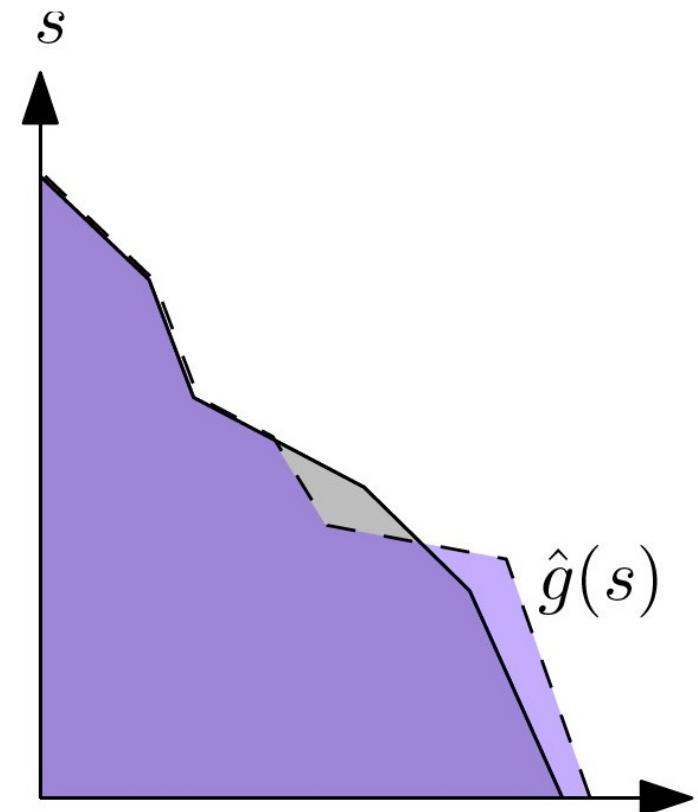
- compare lexicographically

$$f \leq_{lex} g \Leftrightarrow$$

$$\exists s_0 [\forall s \geq s_0 : \hat{f}(s) = \hat{g}(s) \wedge$$

$$\forall \varepsilon > 0 \exists s \in (s_0 - \varepsilon, s_0) \hat{f}(s) < \hat{g}(s)]$$

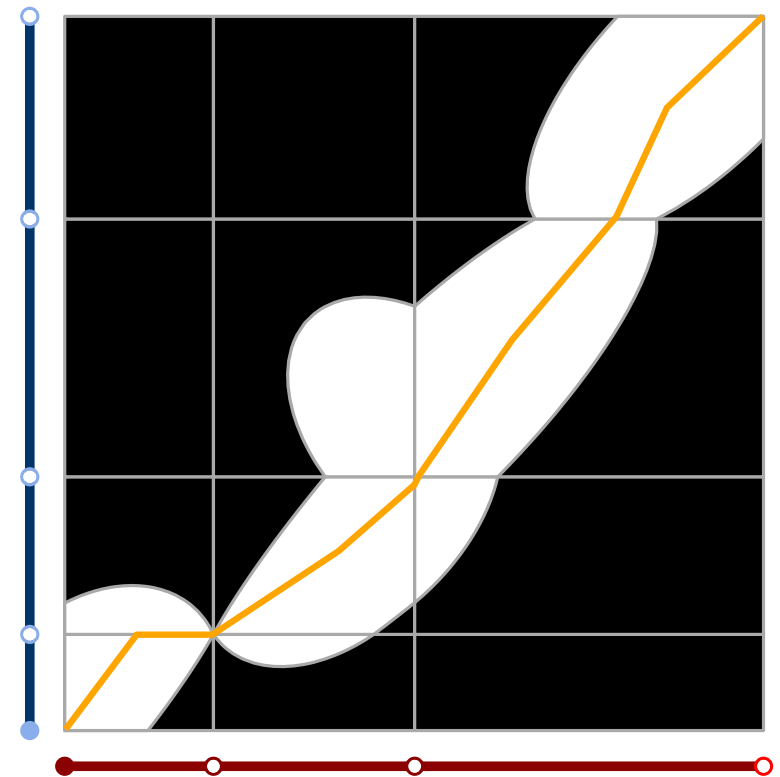
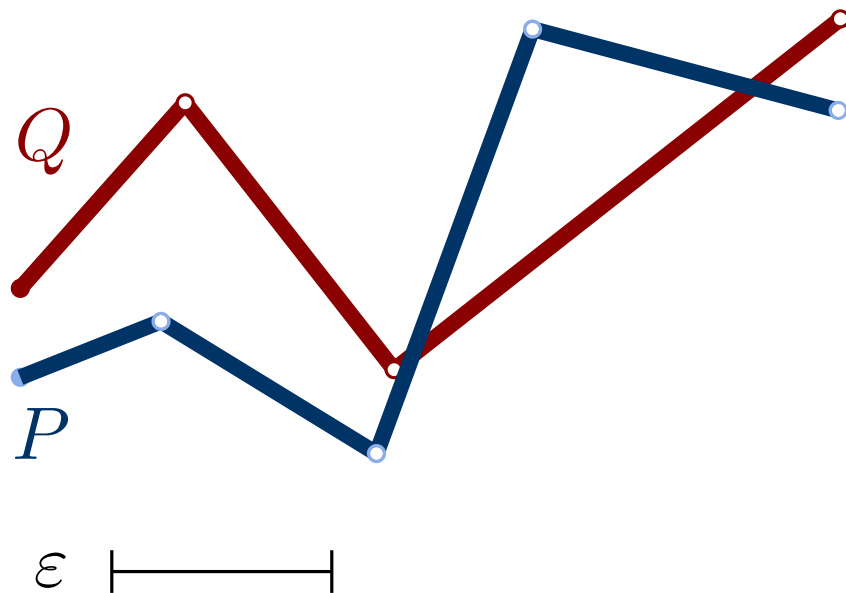
- requires normalization:  
speed of reparameterizations  $\alpha, \beta$   
is bounded by 1





## in the Free Space Diagram

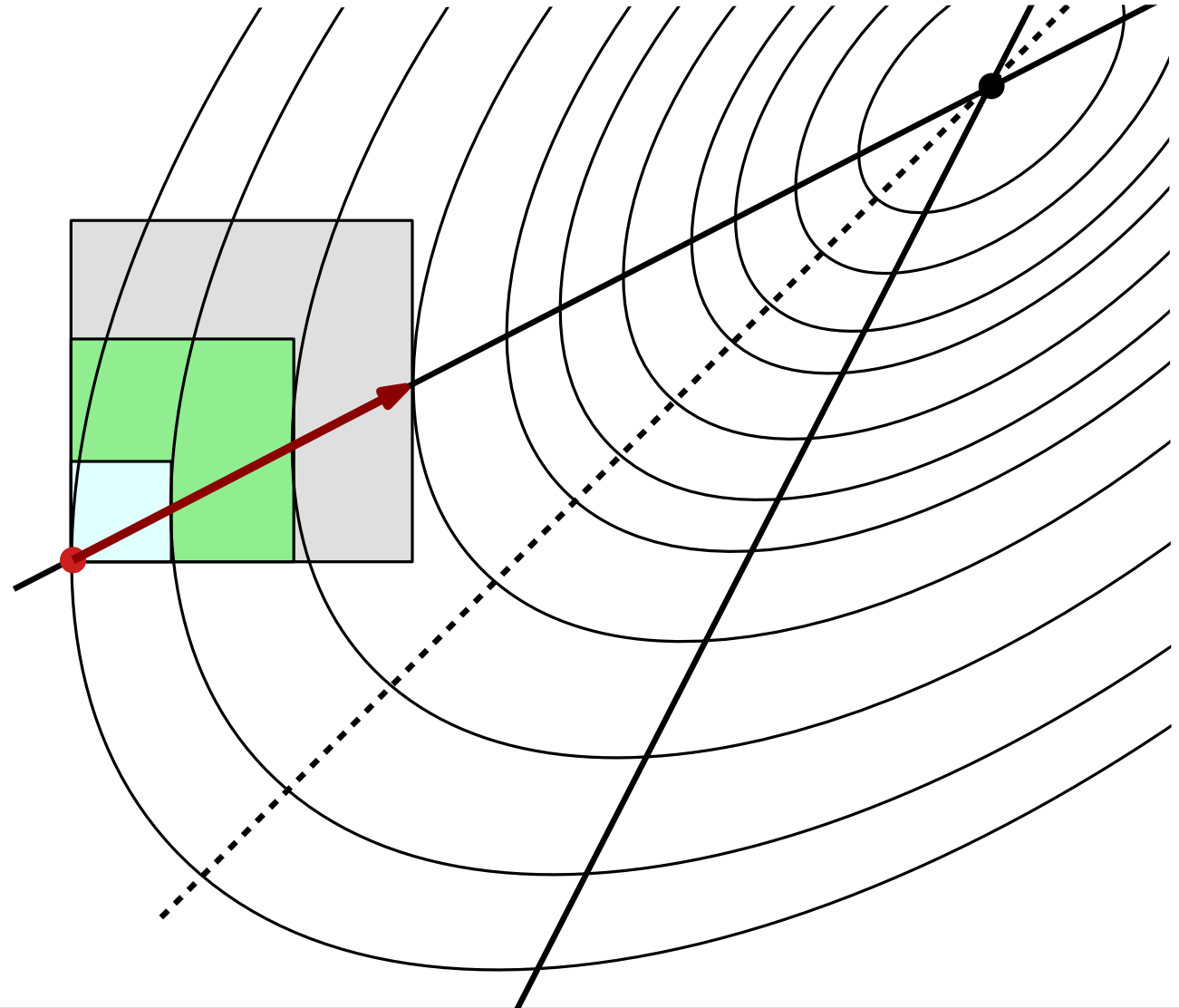
- a monotone path  $c : [0, M] \rightarrow F_\varepsilon$  with lowest profile
- speed constraint:  $c$  is Lipschitz continuous w.r.t  $L_\infty$ -norm



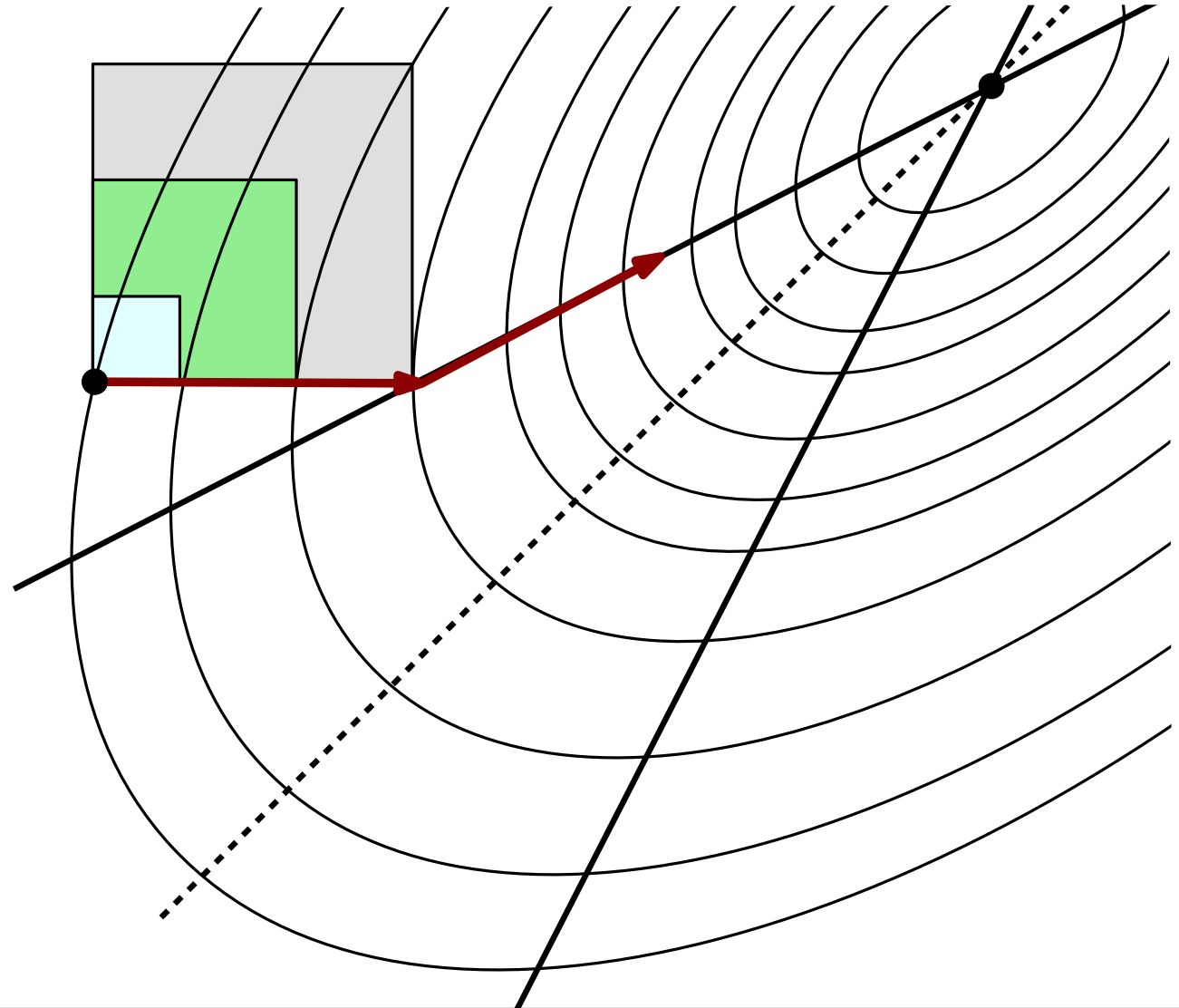
**general idea to find such a curve:**

follow steepest descent path if not constrained by critical values

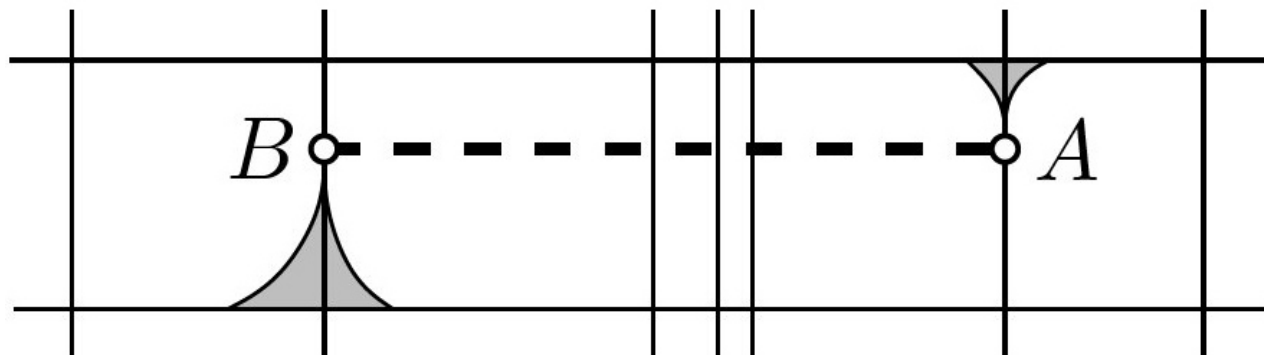
**Lemma:** A steepest descent path is a polygonal path with at most one bend before leaving a cell or reaching the minimum.



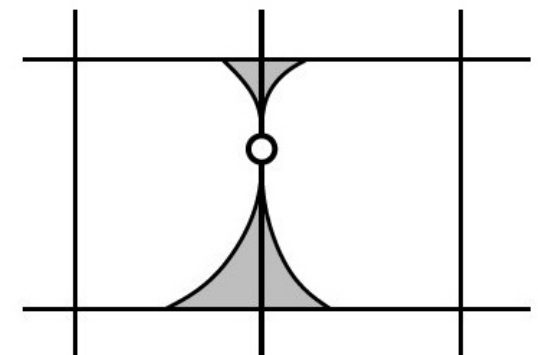
**Lemma:** A steepest descent path is a polygonal path with at most one bend before leaving a cell or reaching the minimum.



- known events

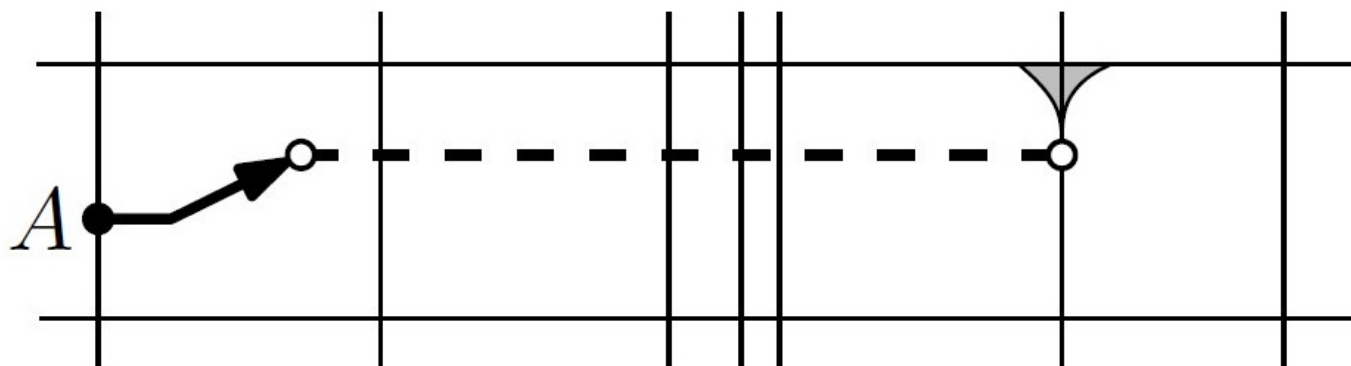


tunnel



boundary

- new events



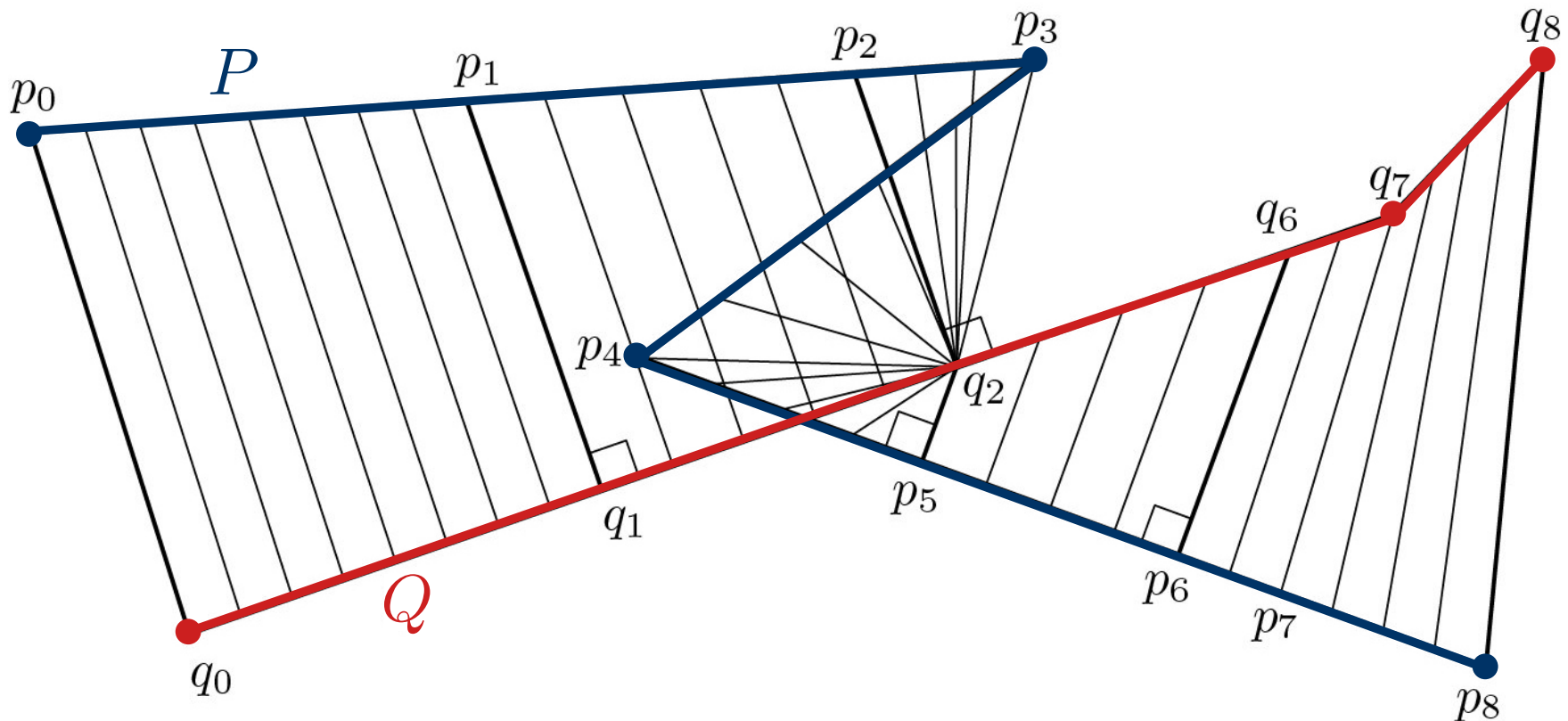
→ at any event the complexity of the curve is reduced!

# Algorithm

to find the optimal curve from A to B:

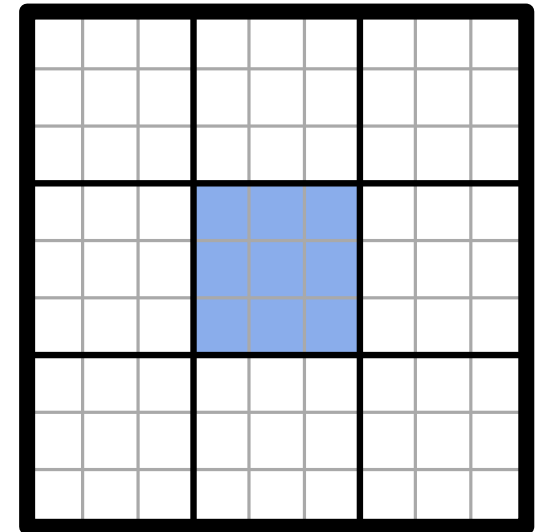
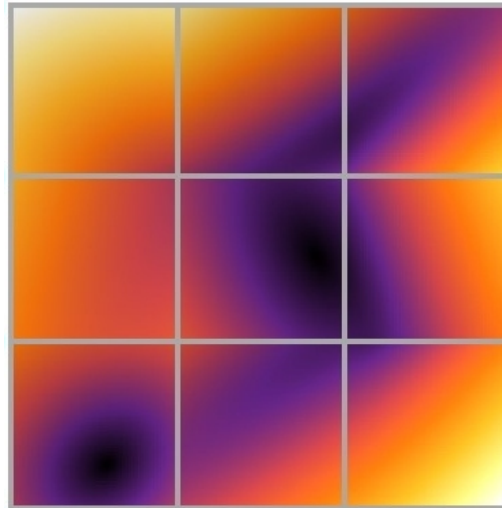
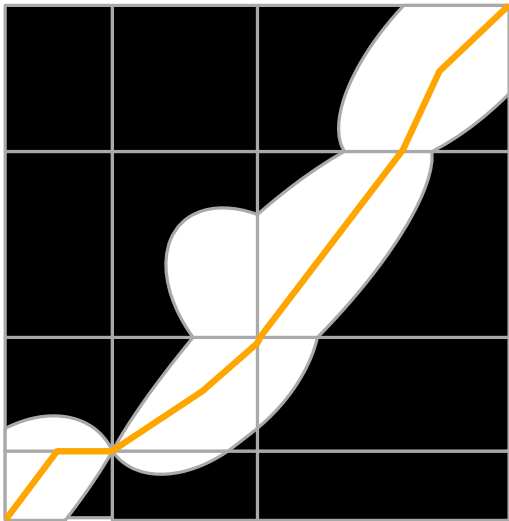
- follow steepest descent if possible
- watch for critical events

**Theorem:** A lexicographic Fréchet Matching can be computed in  $O(mn(m + n) \log mn)$  time. [Rote, 2014]



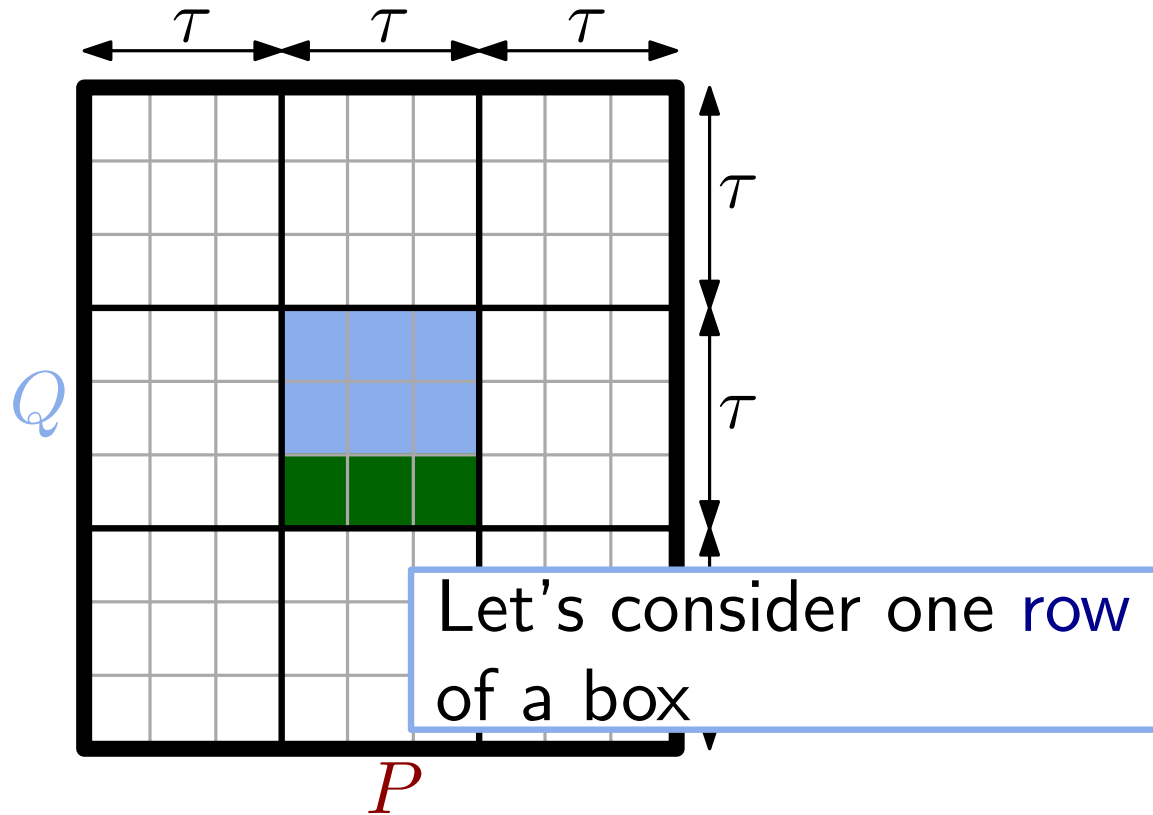
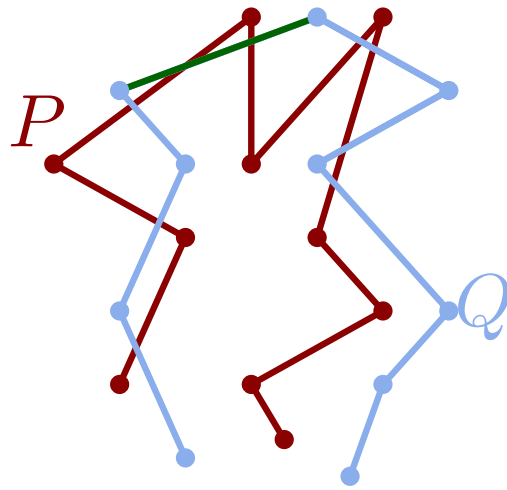
# Searching the Free Space Faster or Better

- locally correct [2012, Buchin et al.]
- lexicographic [2014, Rote]
- retractable [2013, Buchin et al.]
- faster [2014, Buchin et al.]



# Searching Faster by Preprocessing

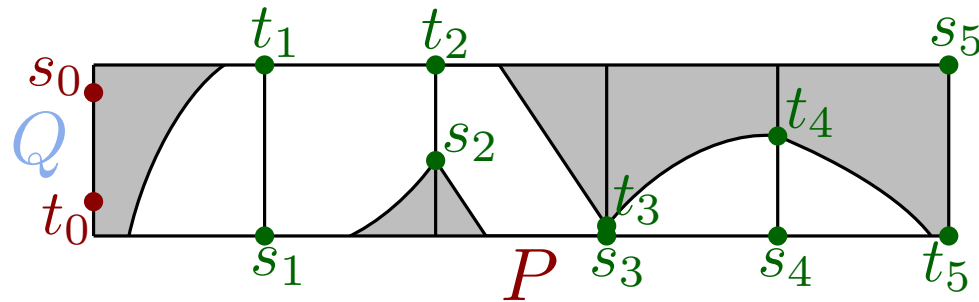
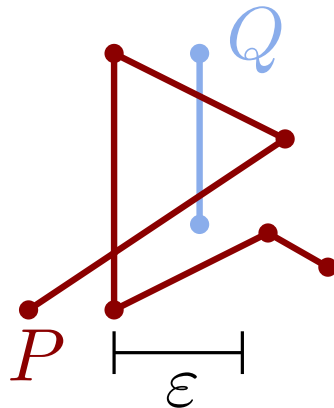
Can we preprocess to speed up the computation?



going below  $O(n^2)$

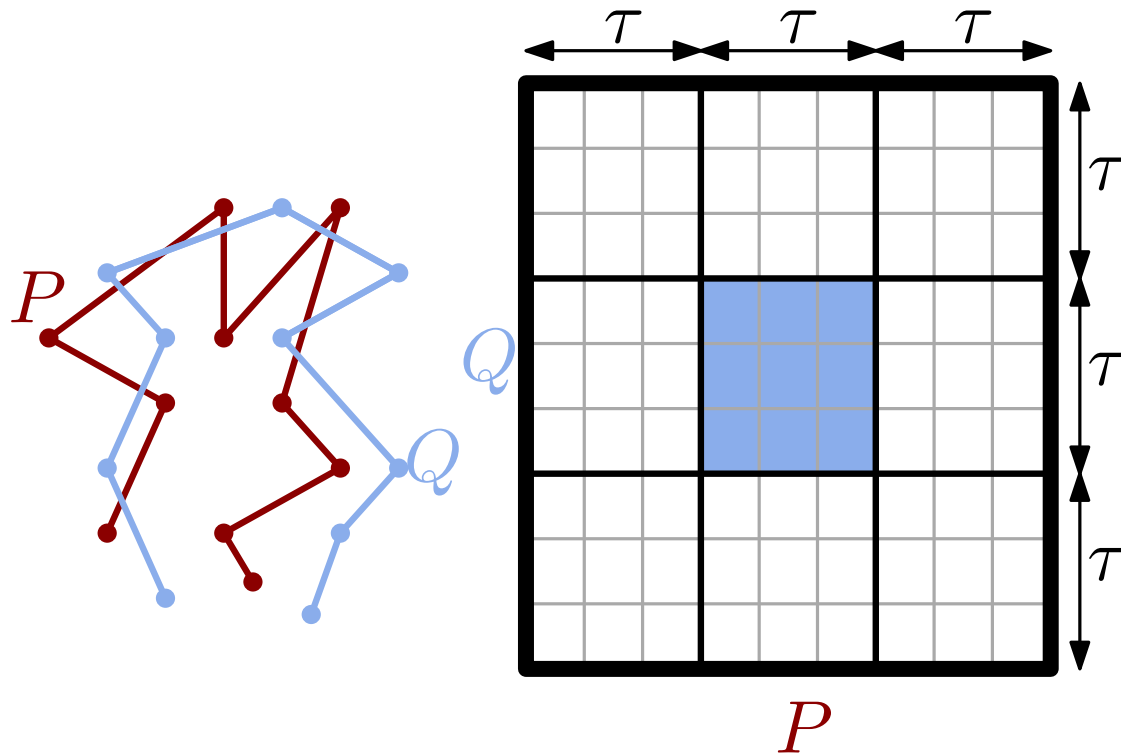
- process elementary boxes ( $\tau \times \tau$  cells) in  $o(\tau^2)$  time.
- How many combinatorial types of boxes are there?
- How can we find the combinatorial type of a box fast?





## combinatorial type of a row

- white passages (*doors*): lower end ( $y$ -value)  $s_i$ , upper  $t_i$
- $s_0, t_0$  encode reachability from the left (*reach-door*)
- row order: permutation of  $s_i, t_i$  (sorted by height)  
 $s_1 s_3 s_4 t_5 t_3 t_0 s_2 t_4 s_0 s_5 t_1 t_2$
- partial row order: permutation of  $s_i, t_i$  without  $s_0$  and  $t_0$
- row order determines how reachability propagates from left to right



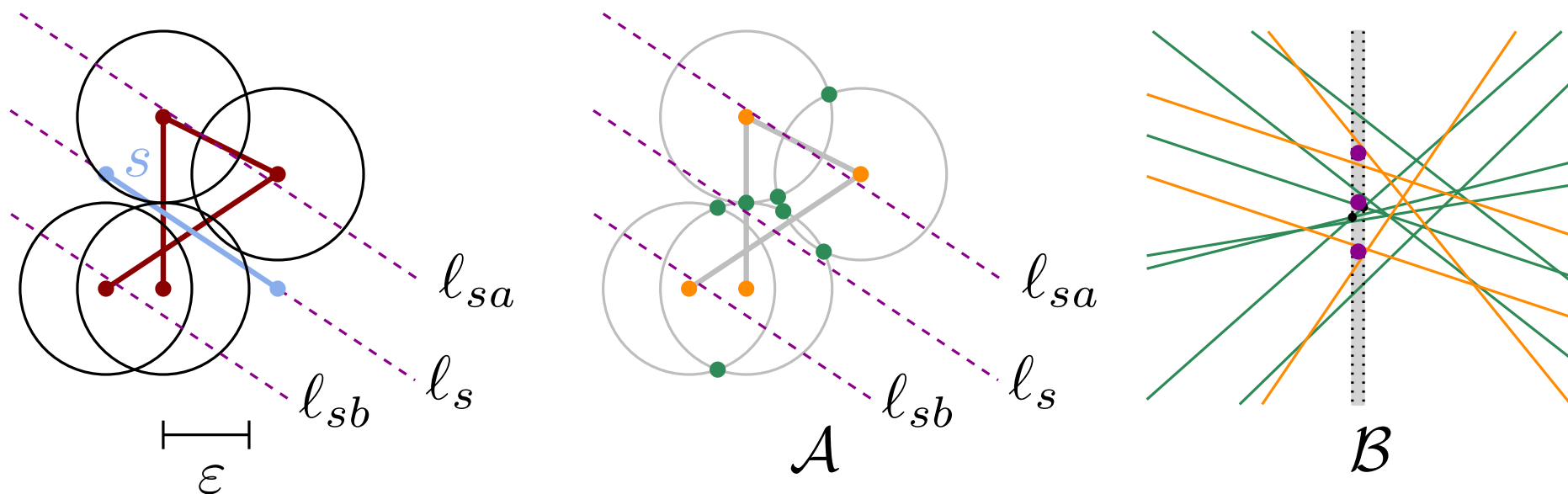
How do we determine the signature?

*boils down to*

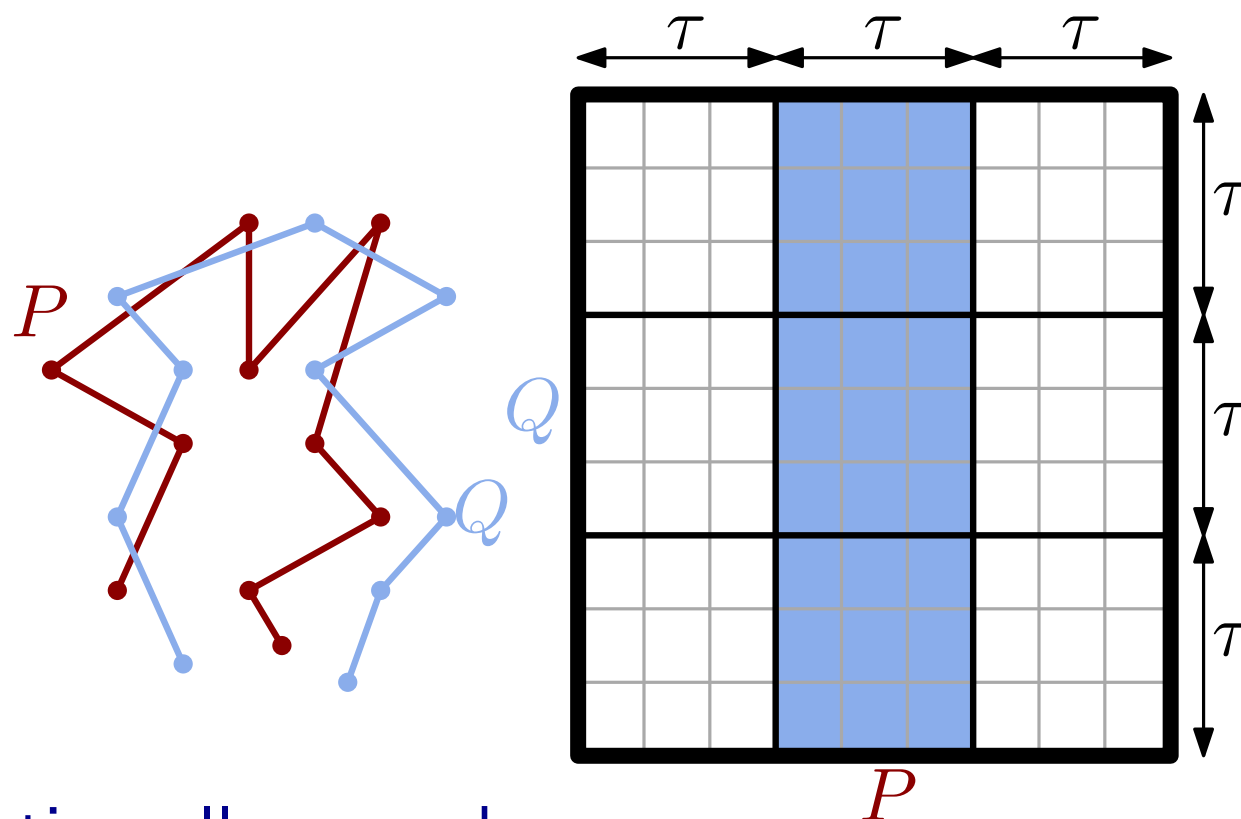
- determine partial door orders
- add reach-doors

combinatorial type of an elementary box

- signature of a box: all its row&column door orders
- partial signature of a box: all its partial door orders
- signature determines how reachability propagates from left&bottom to right&top



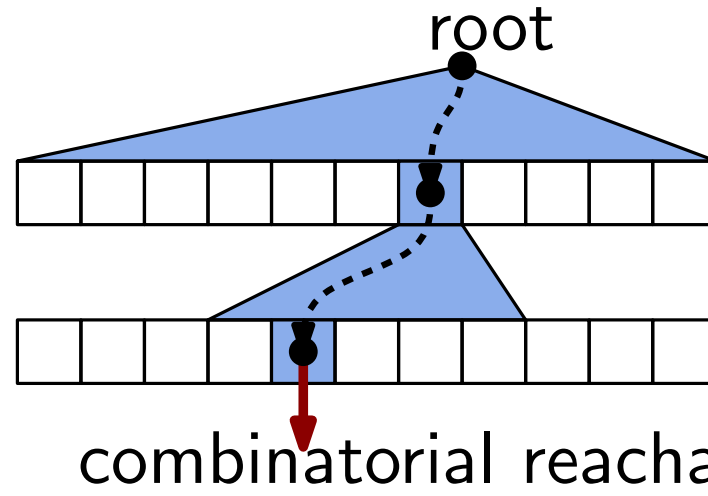
- ordered sequence of intersections of  $s$  with  $\varepsilon$ -circles determines partial door order
- sequence of intersections of  $l_s$  with circles is determined by position of  $l_s$ ,  $l_{sa}$  and  $l_{sb}$  relative to the circle centers and the intersection of circles
- with point location in the dual plane we can (after  $O(\tau^c)$  preprocessing) query partial door orders in  $O(\log \tau)$  time.



computing all row orders

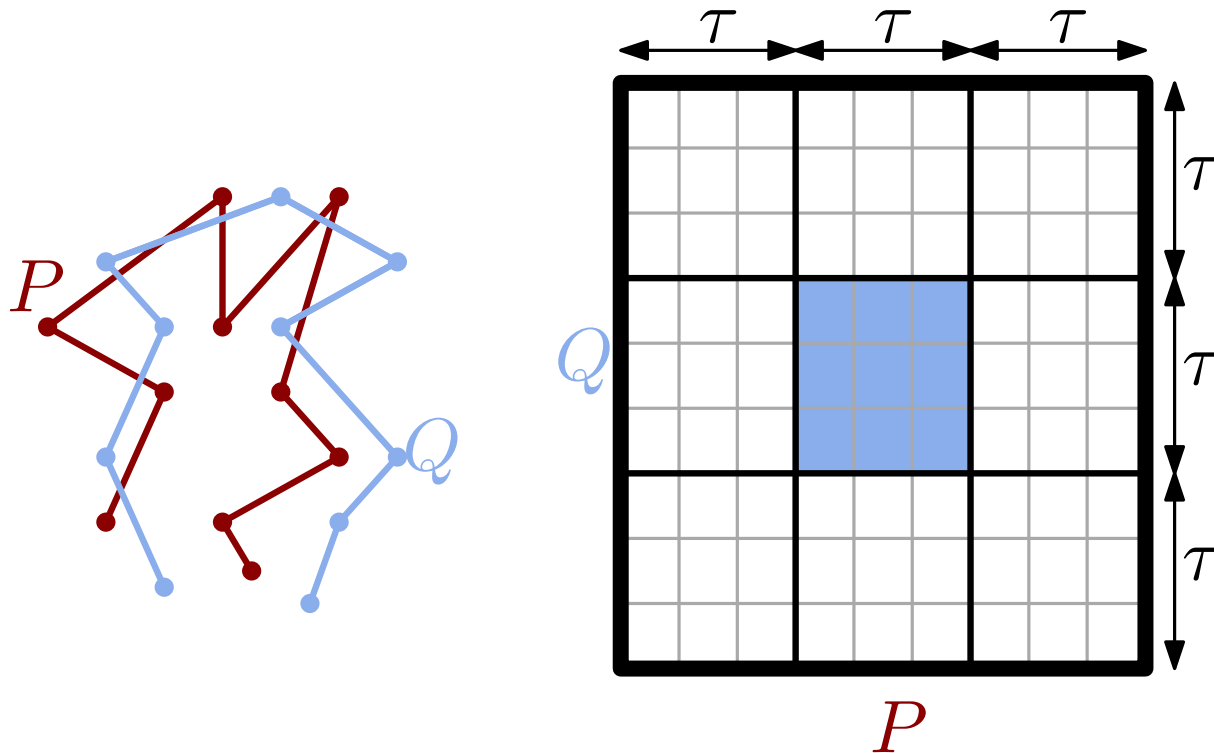
- process vertical strips of elementary boxes
- compute partial row orders in strip in time  $O(\tau^c + n \log \tau)$
- compute row orders from partial row orders in strip in time  $O(n \log \tau)$  (having stored a suitable search structure)

compute all column orders in the same way



*compute partial door-orders*  
*partial signature*  
*add reach-doors*  
*full signature*

- Takes  $O(n\tau^{c-1} + \frac{n^2 \log \tau}{\tau})$  time except for last step
- precompute **lookup table** to go from signature to reachability
- bottleneck:
  - size of lookup table = **number of different signatures**
- number of different row/column orders:  $(2(\tau + 1))!$
- number of signatures:  $(2(\tau + 1))!^{2\tau} = O(\tau^{O(\tau^2)})$
- Set  $\tau = \Theta(\sqrt{\log n / \log \log n})$ . Then ...



**Theorem:** The Fréchet distance of two polygonal curves of size  $n$  each can be decided in  $O(n^2 (\log \log n)^{3/2} / \sqrt{\log n})$  time.

[Buchin, Buchin, Meulemans, Mulzer, 2014]

# Summary

many different ways to search the free space

- nicer following steepest descent
- faster using 4 Russians

Thanks!