RUB

Computing the cut distance of two curves

Maike Buchin, Ruhr University Bochum

Leonie Ryvkin, Ruhr University Bochum

Jérôme Urhausen, Utrecht University

RUB

Visiting Würzburg

Schloss Veitshöchheim

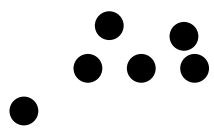
Mainbrücke, Marienburg,

Residenz, Dom...

RUB

Visiting Würzburg

Schloss Veitshöchheim



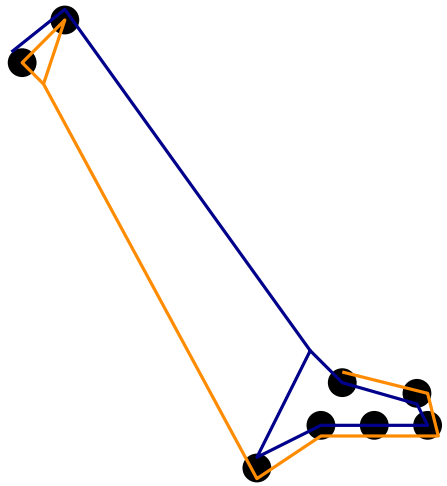
Mainbrücke, Marienburg,

Residenz, Dom...

RUB

Visiting Würzburg

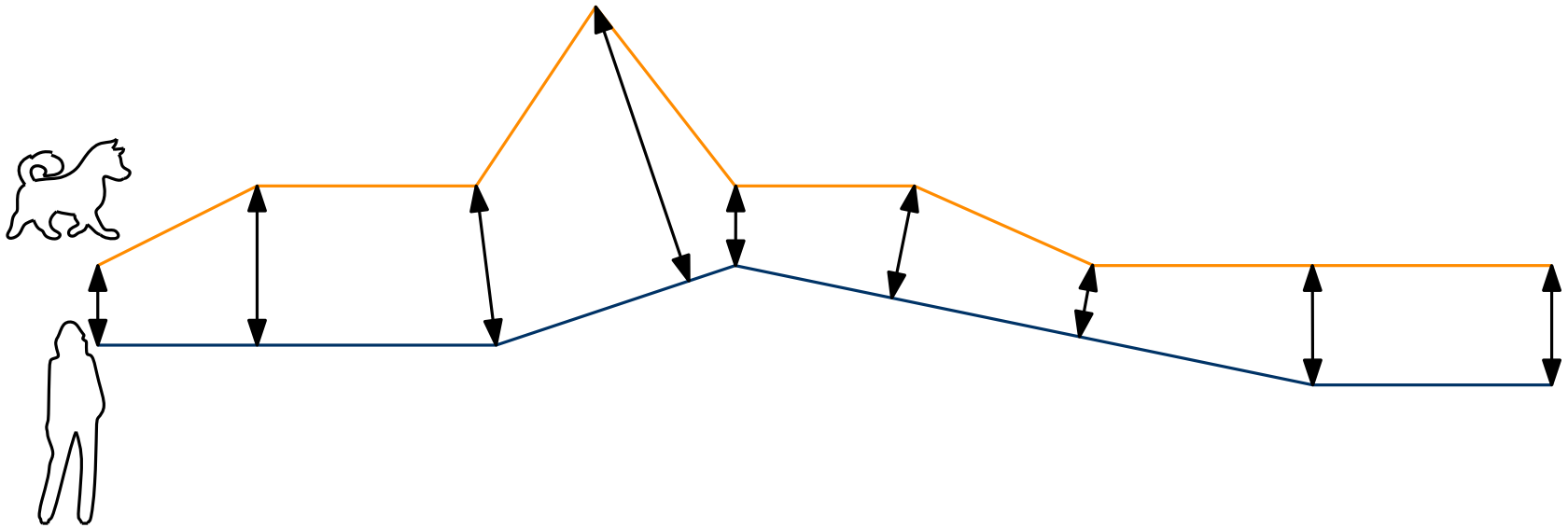
Schloss Veitshöchheim



Mainbrücke, Marienburg,

Residenz, Dom...

The Fréchet distance



$$\delta_F(P, Q) = \inf_{\tau,\sigma} \max_{t\in[0,1]} \|P(\tau(t)) - Q(\sigma(t))\|$$

τ, σ range over all orientation-preserving homeomorphisms

The cut distance

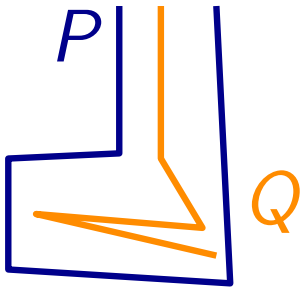
$$\delta_{\text{cut}}(k, P, Q) = \inf_{\tau,\sigma} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|$$

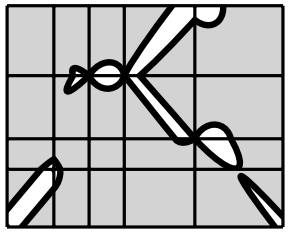
where τ, σ range over all piecewise continuous, surjective functions with $k' < k$ jump discontinuities, each.

The cut distance

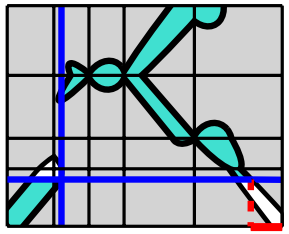
$$\delta_{\text{cut}}(k, P, Q) = \inf_{\tau,\sigma} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|$$

where τ, σ range over all piecewise continuous, surjective functions with $k' < k$ jump discontinuities, each.

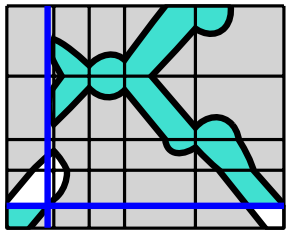




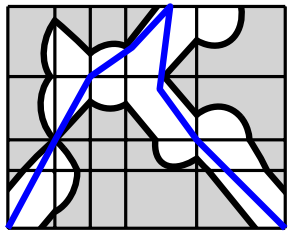
$\varepsilon_1 \geq \delta_{\text{H}}(P, Q)$



$\varepsilon_2 < \delta_{\text{cut}}(2, P, Q)$



$\varepsilon_3 \geq \delta_{\text{cut}}(2, P, Q)$

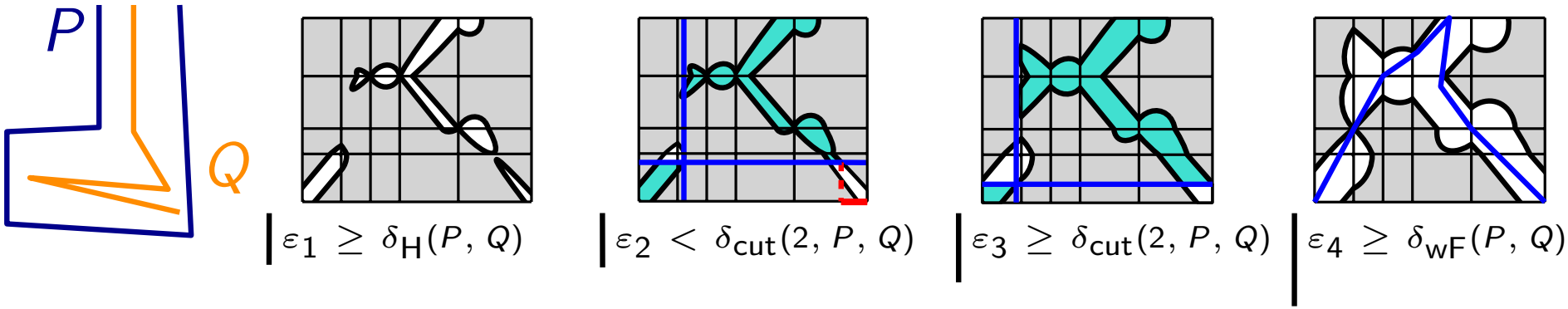


$\varepsilon_4 \geq \delta_{\text{wF}}(P, Q)$

The cut distance

$$\delta_{\text{cut}}(k, P, Q) = \inf_{\tau,\sigma} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|$$

where τ, σ range over all piecewise continuous, surjective functions with $k' < k$ jump discontinuities, each.

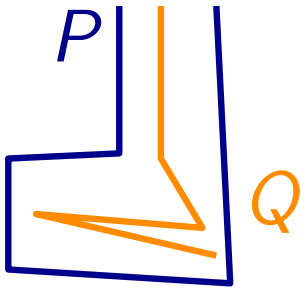


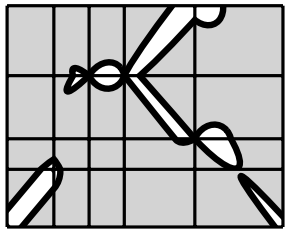
$$F_\varepsilon(P, Q) = \{(t_1, t_2) : \|P(t_1) - Q(t_2)\| \leq \varepsilon\}$$

The cut distance

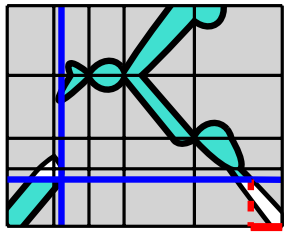
$$\delta_{\mathrm{cut}}(k, P, Q) = \inf_{\tau,\sigma} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|$$

where τ, σ range over all piecewise continuous, surjective functions with $k' < k$ jump discontinuities, each.

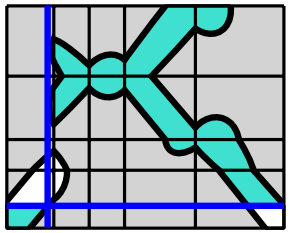




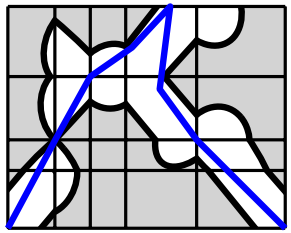
$\varepsilon_1 \geq \delta_{\mathrm{H}}(P, Q)$



$\varepsilon_2 < \delta_{\mathrm{cut}}(2, P, Q)$



$\varepsilon_3 \geq \delta_{\mathrm{cut}}(2, P, Q)$

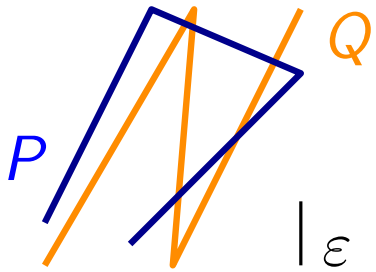


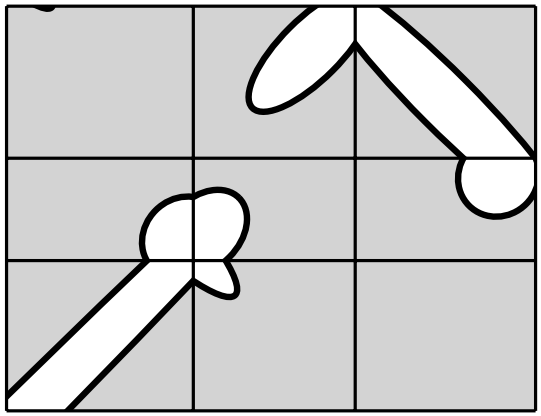
$\varepsilon_4 \geq \delta_{\mathrm{wF}}(P, Q)$

Observe that $\delta_{\mathrm{H}} \leq \delta_{\mathrm{cut}} \leq \delta_{\mathrm{wF}} \leq \delta_{\mathrm{F}}$.

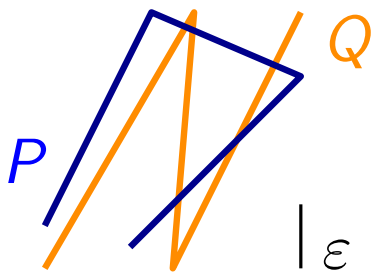
Placing cut lines

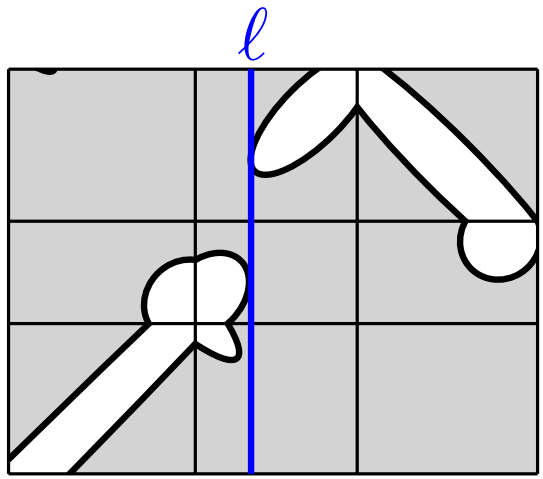
RUB



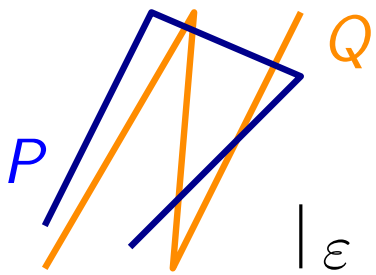


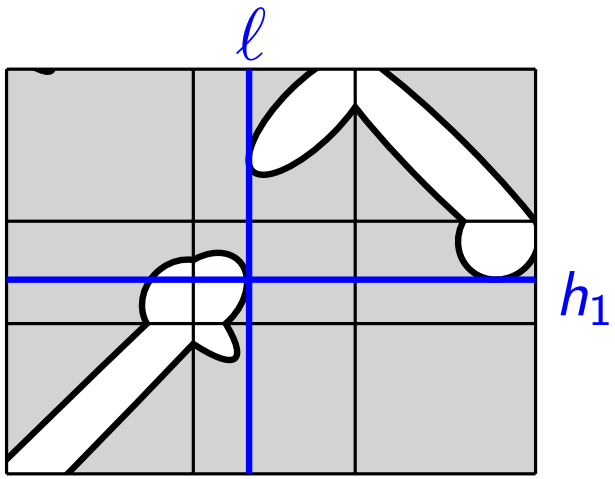
Placing cut lines



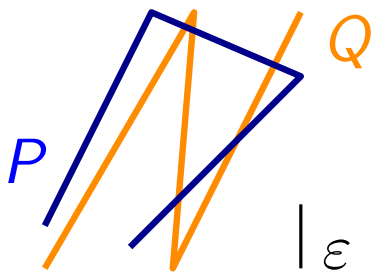


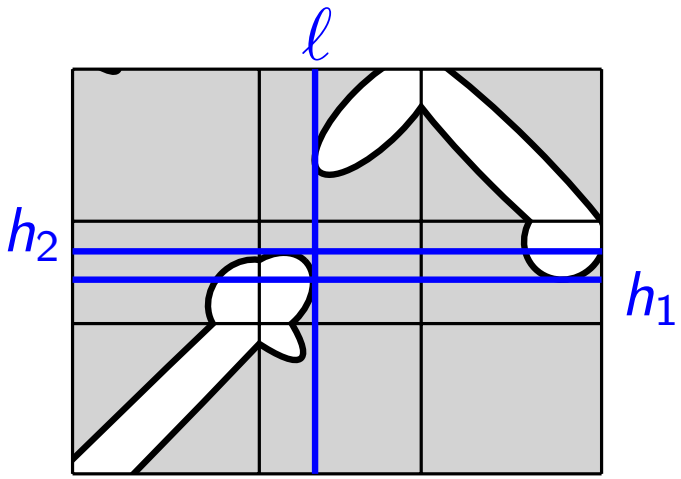
Placing cut lines



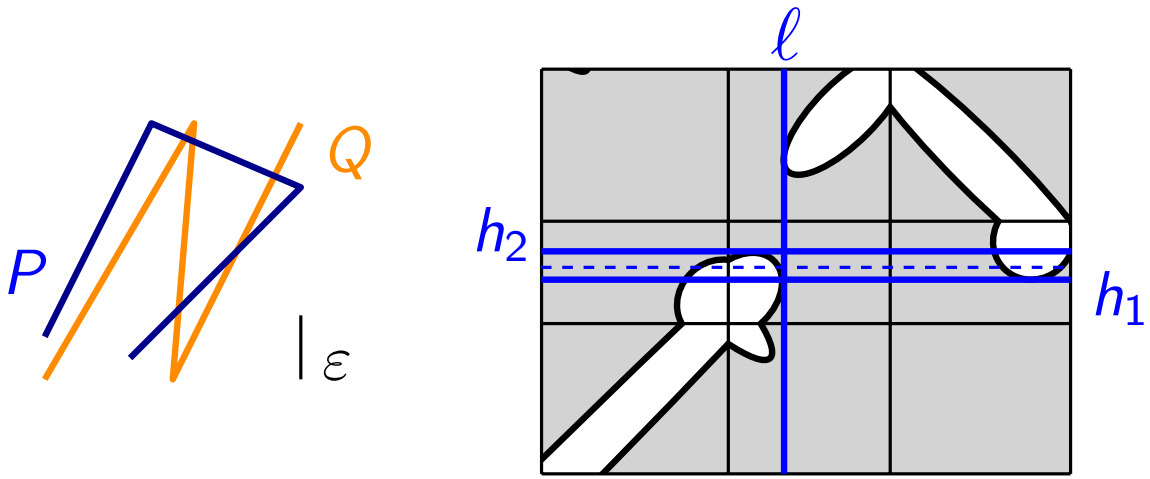


Placing cut lines



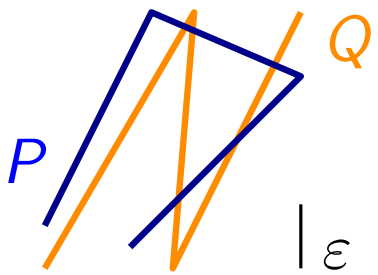


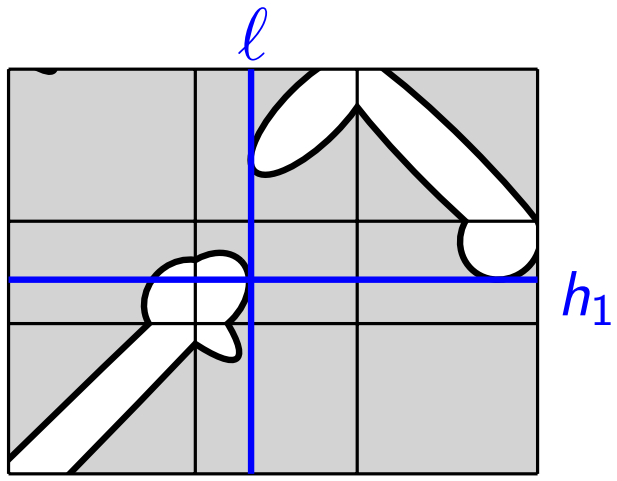
Placing cut lines



Problem: In general, there are infinitely many possible placements for cut lines.

Placing cut lines



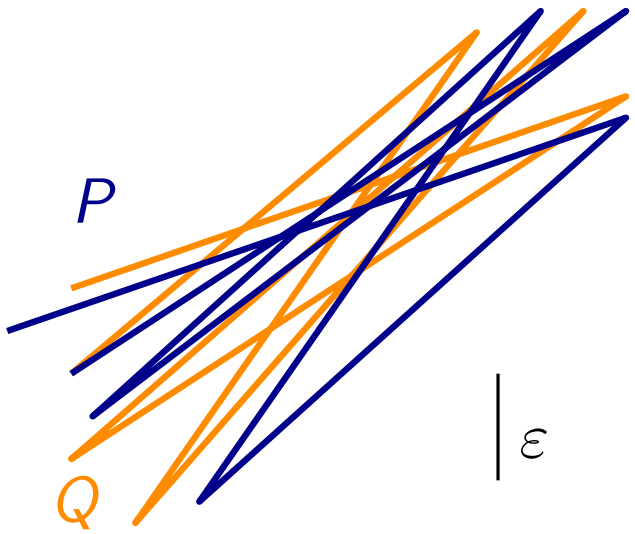


For $k = 2$, it is possible to move valid cut lines such that one of them coincides with an extremal point of a component's boundary.

Algorithm for $k = 2$

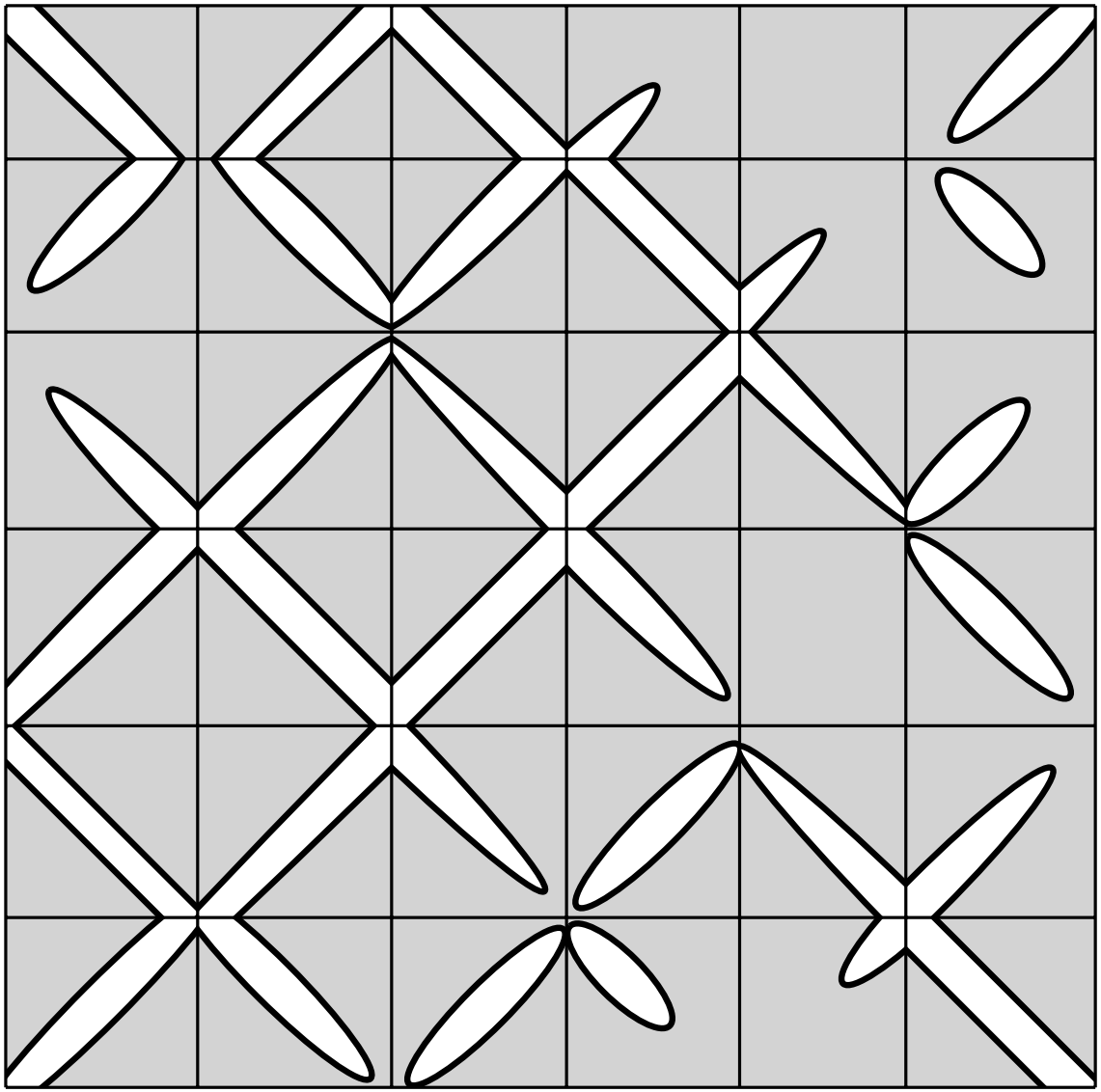
RUB

Input: polygonal curves P, Q and $\varepsilon > 0$

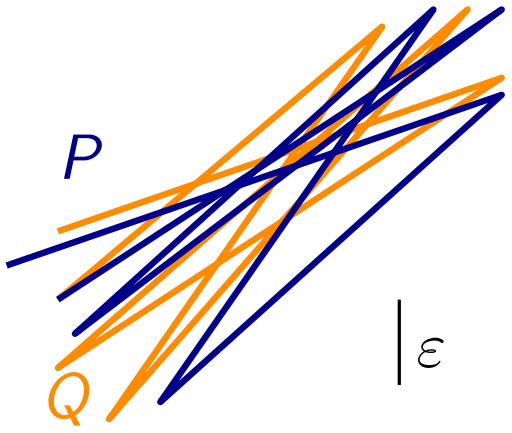


Algorithm for $k = 2$

RUB

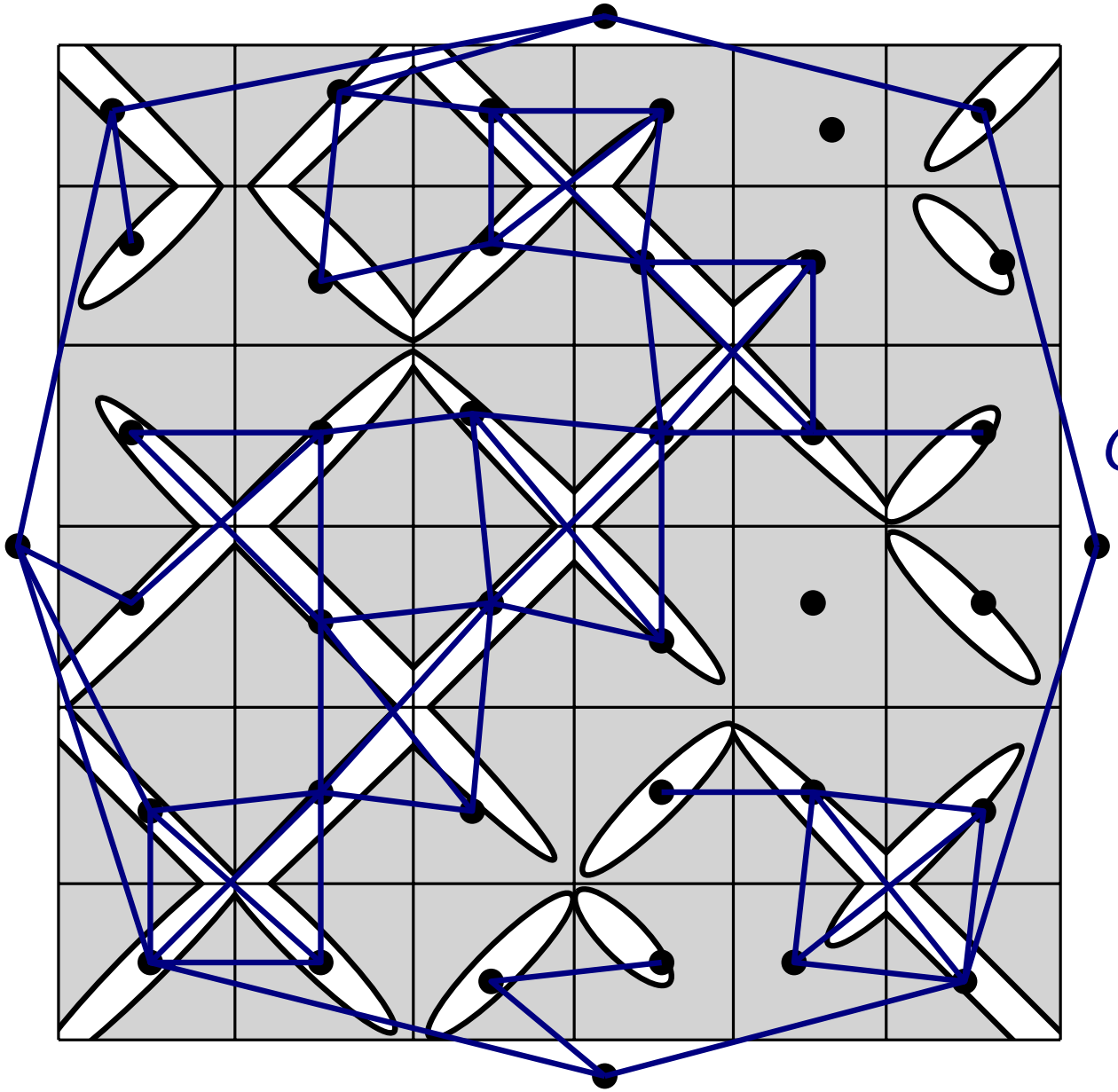


1. Compute FSD

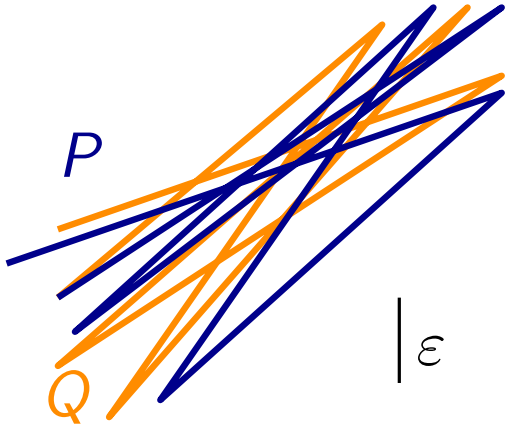


Algorithm for $k = 2$

RUB

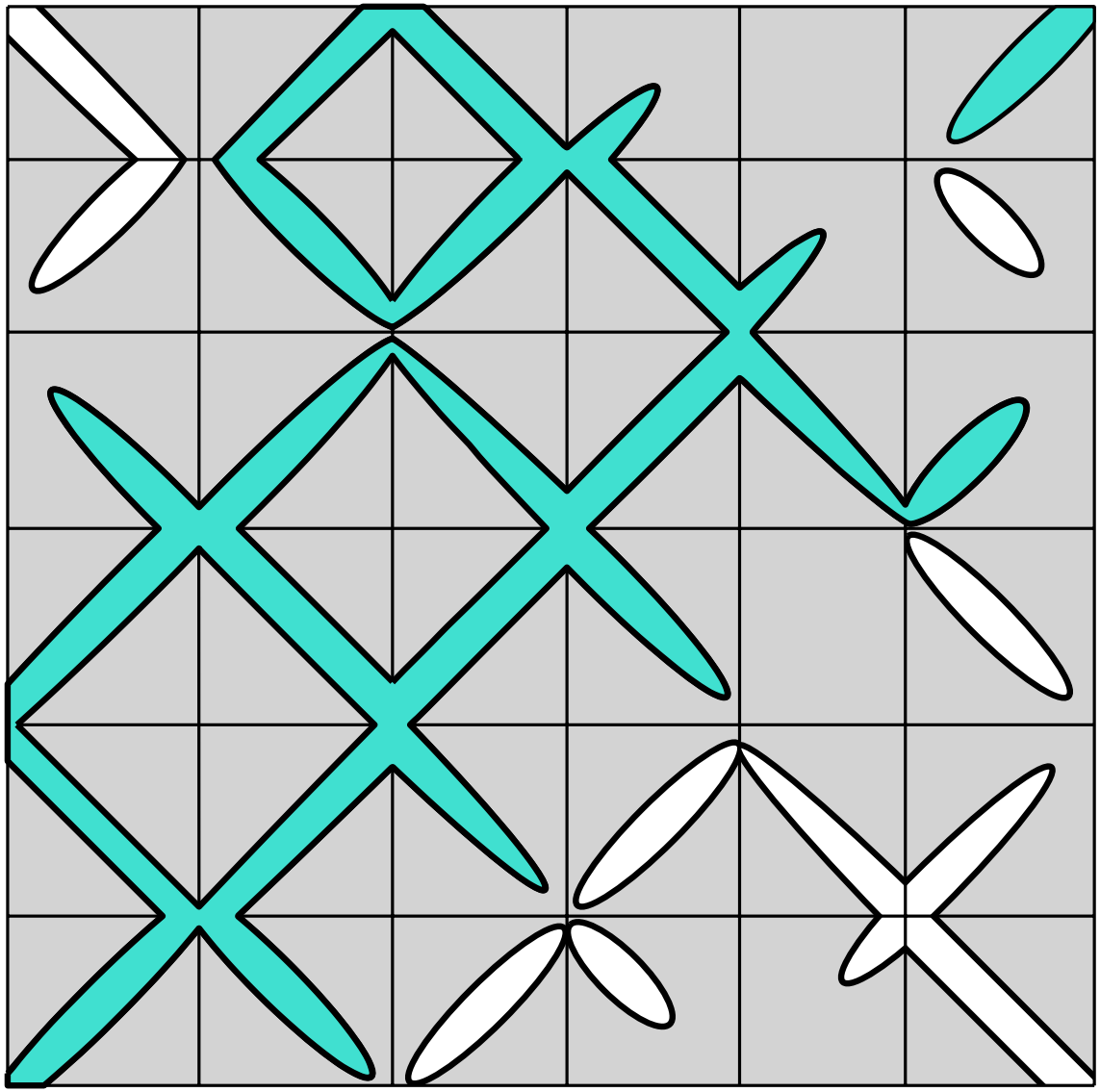


1. Compute FSD

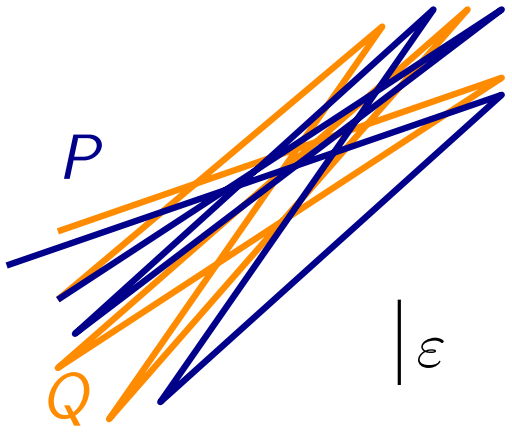


Algorithm for $k = 2$

RUB

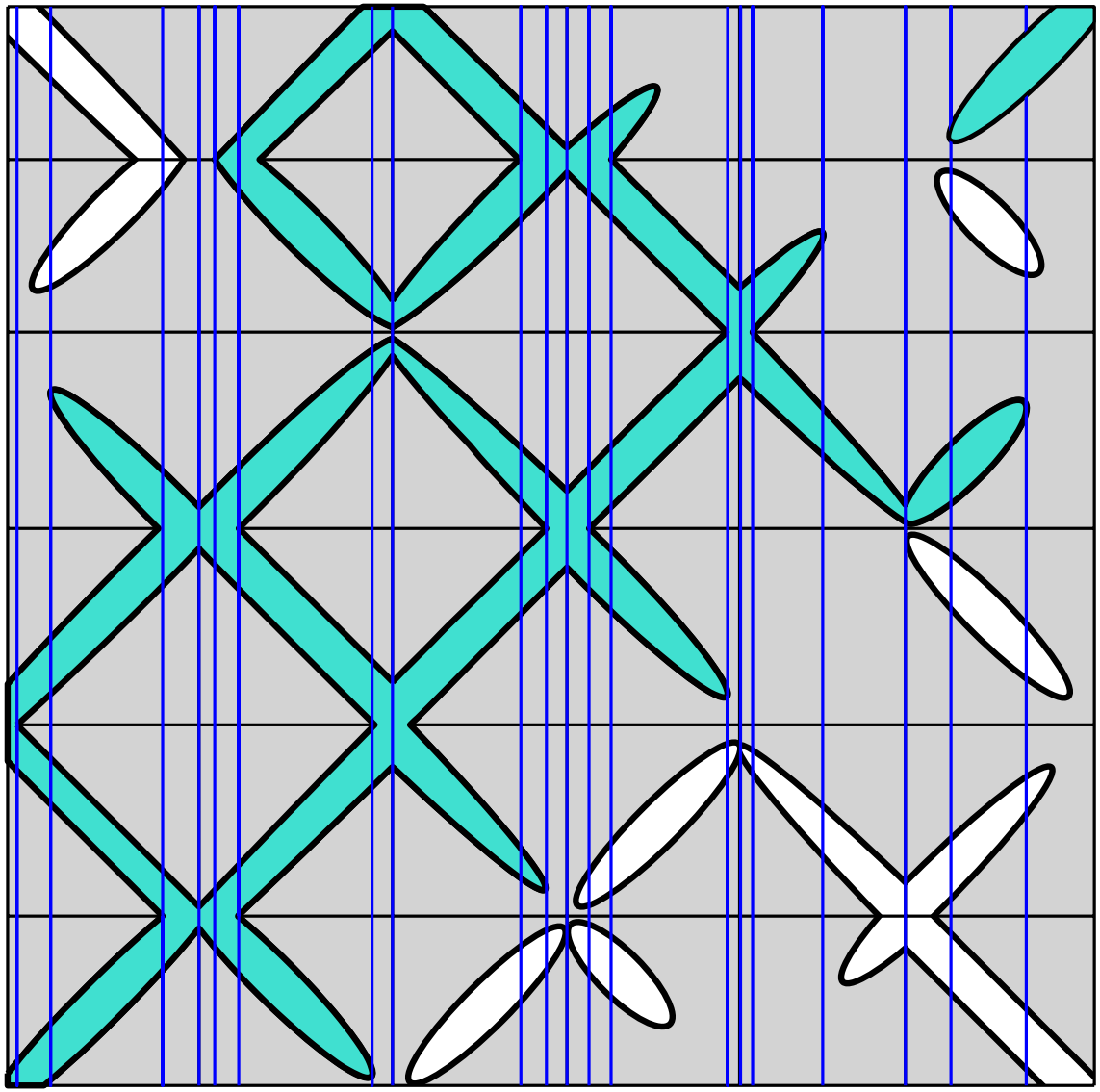


1. Compute FSD
2. Identify *candidates* a_i, b_j

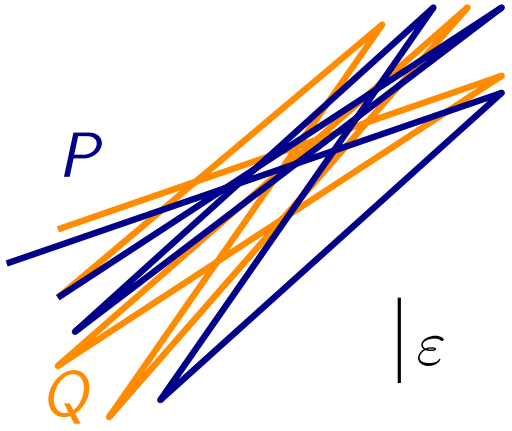


Algorithm for $k = 2$

RUB

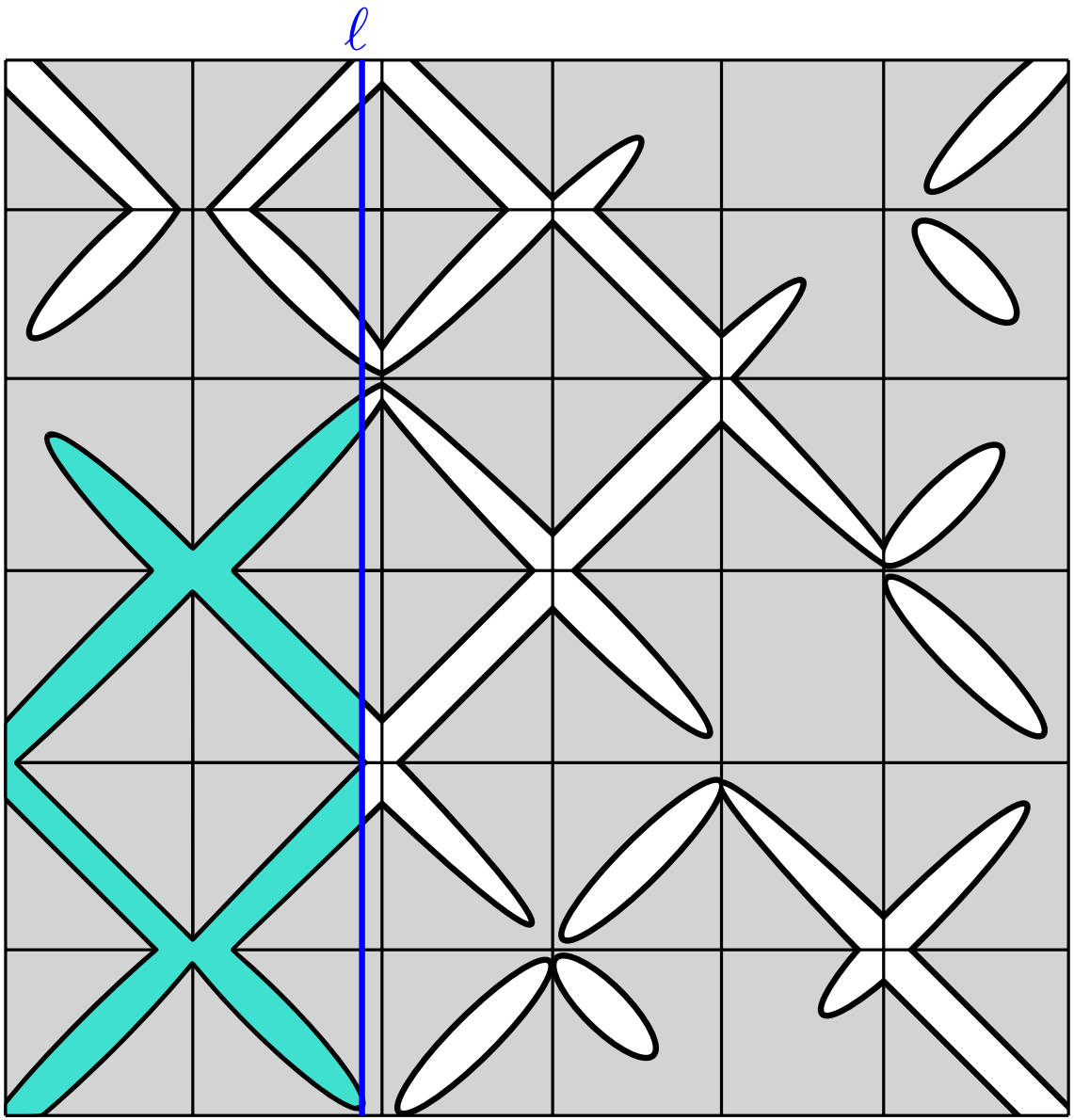


1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points

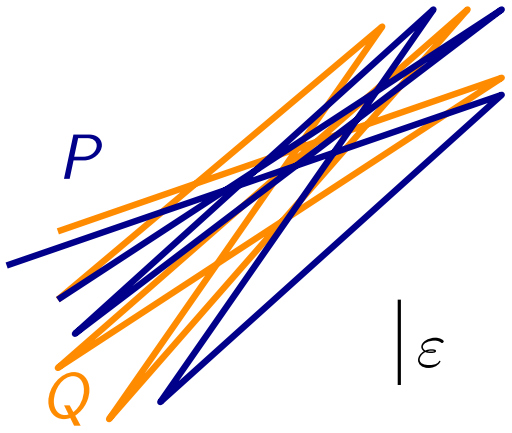


Algorithm for $k = 2$

RUB

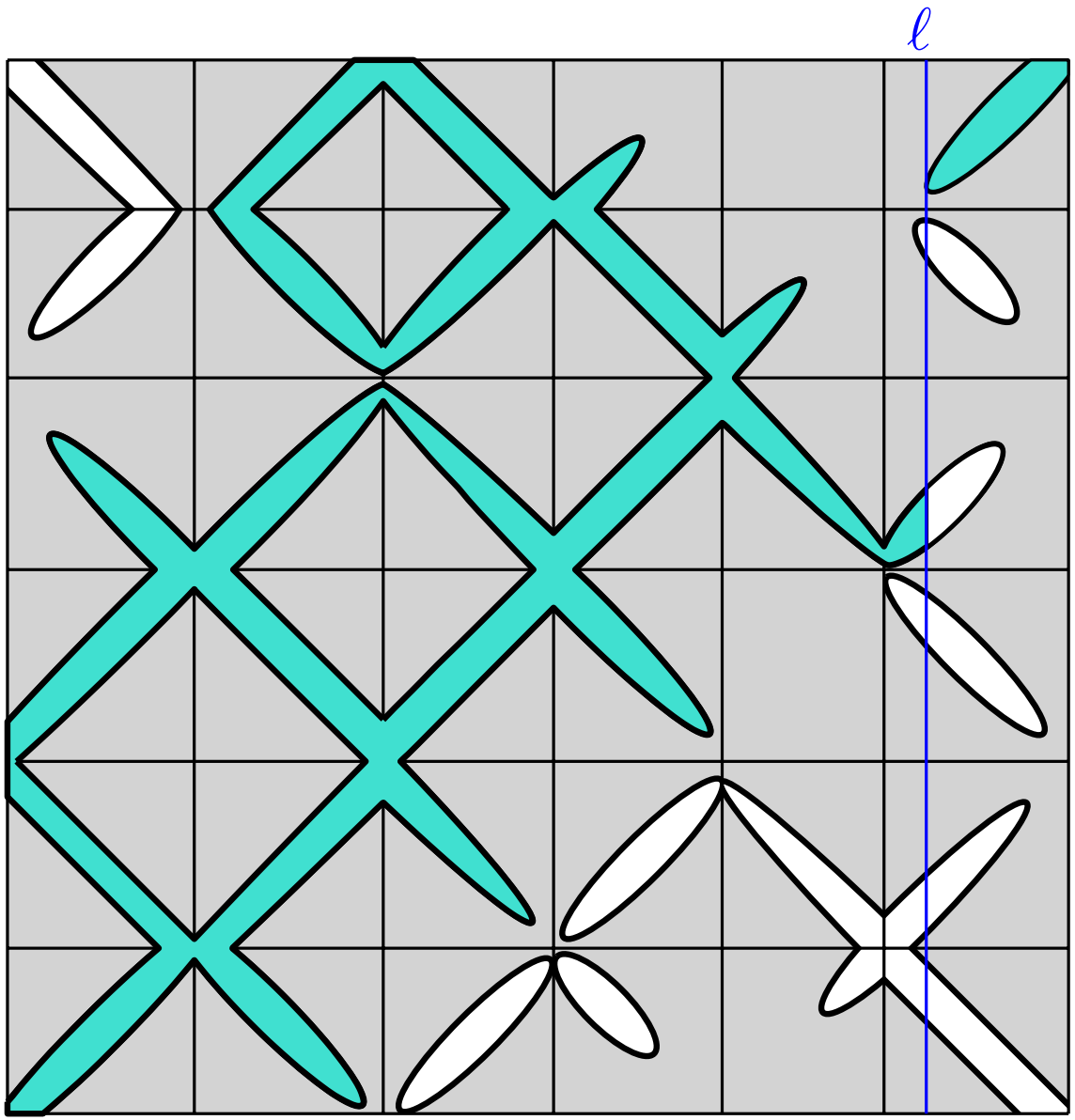


1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ

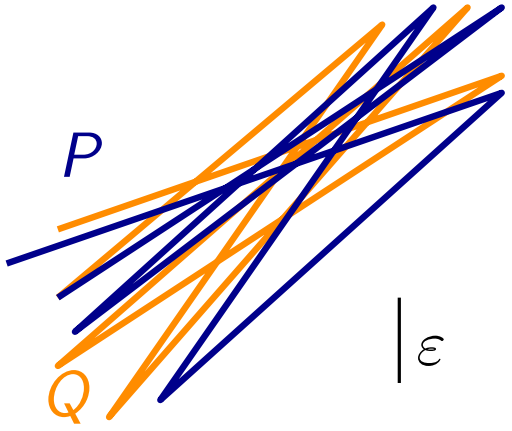


Algorithm for $k = 2$

RUB

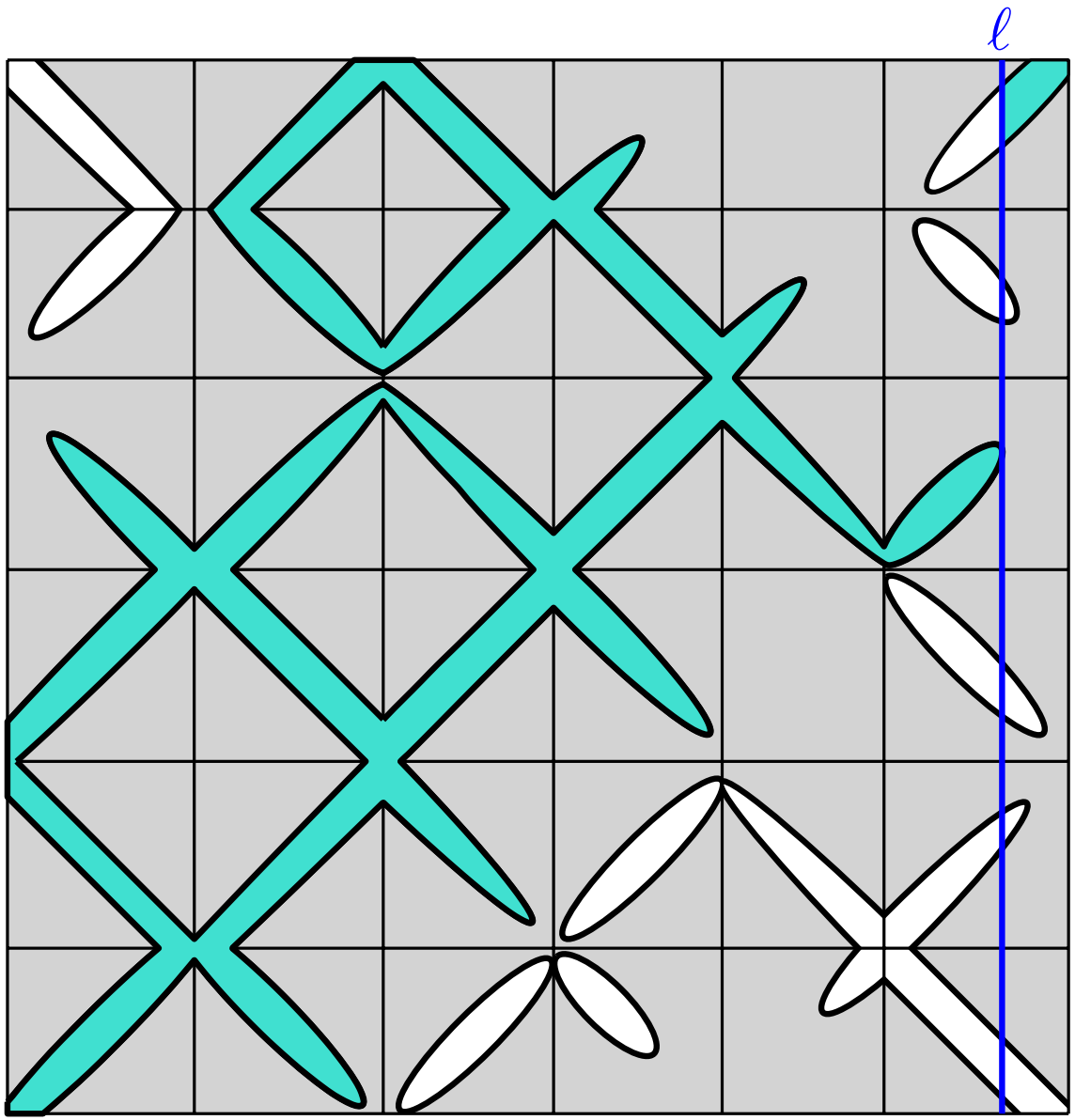


1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ

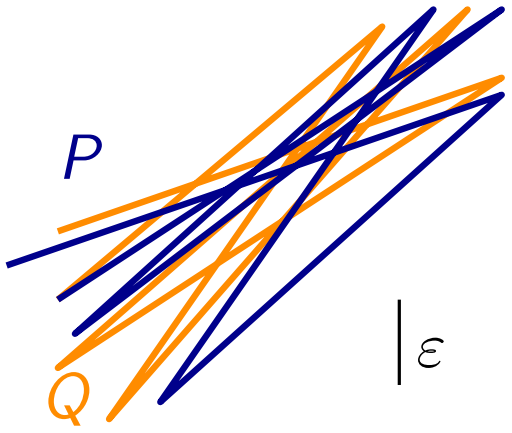


Algorithm for $k = 2$

RUB

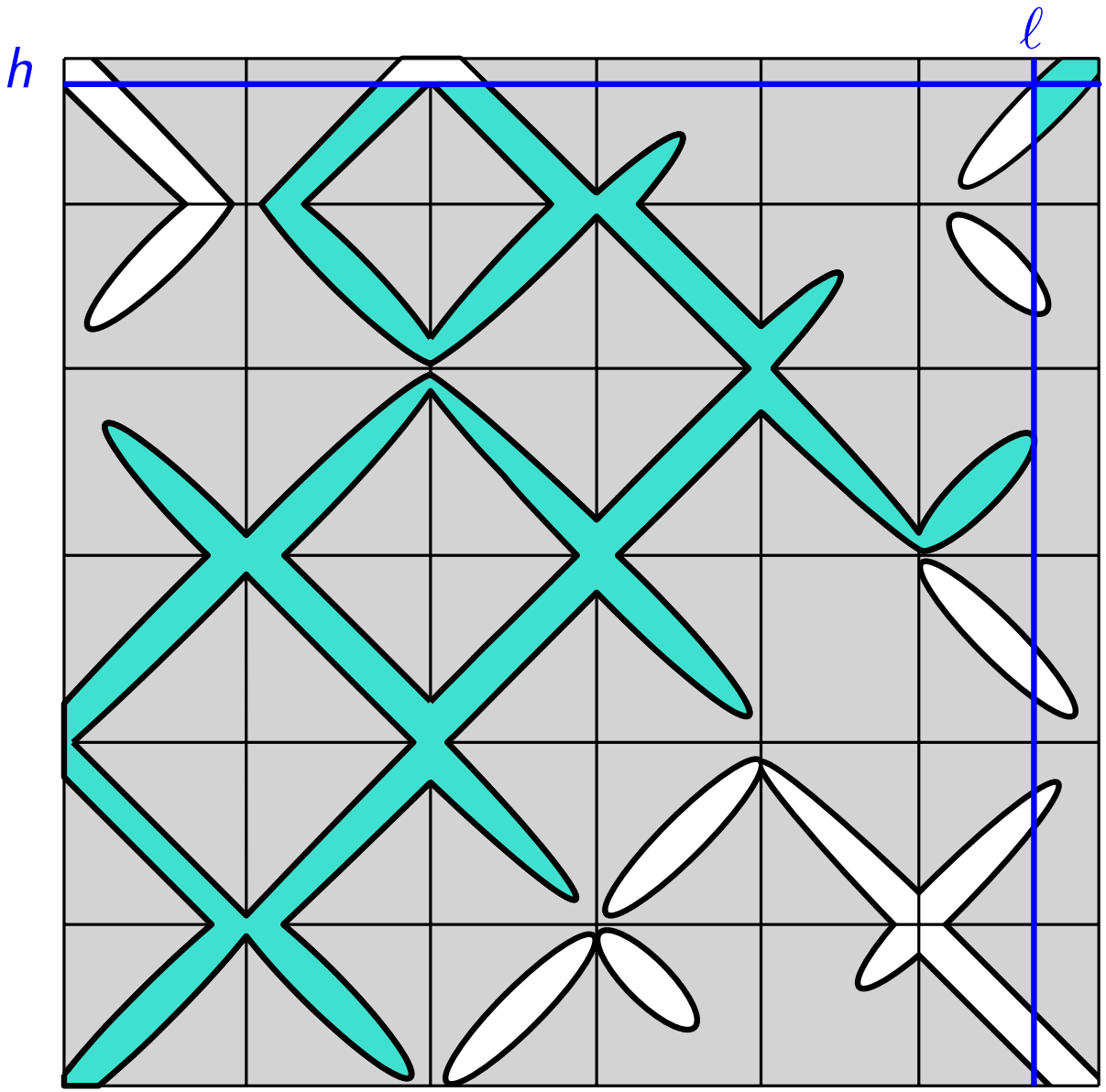


1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ



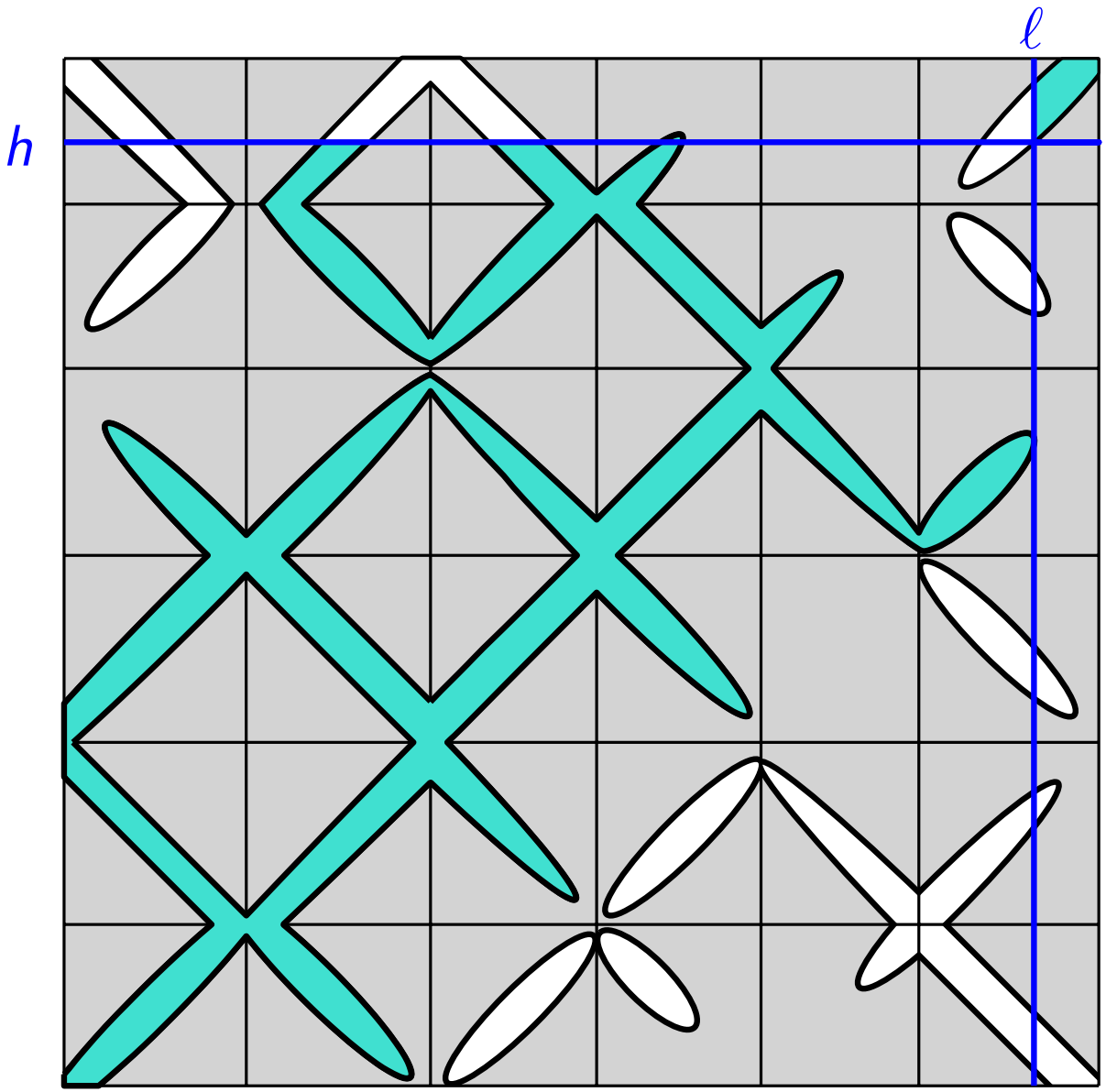
Algorithm for $k = 2$

RUB



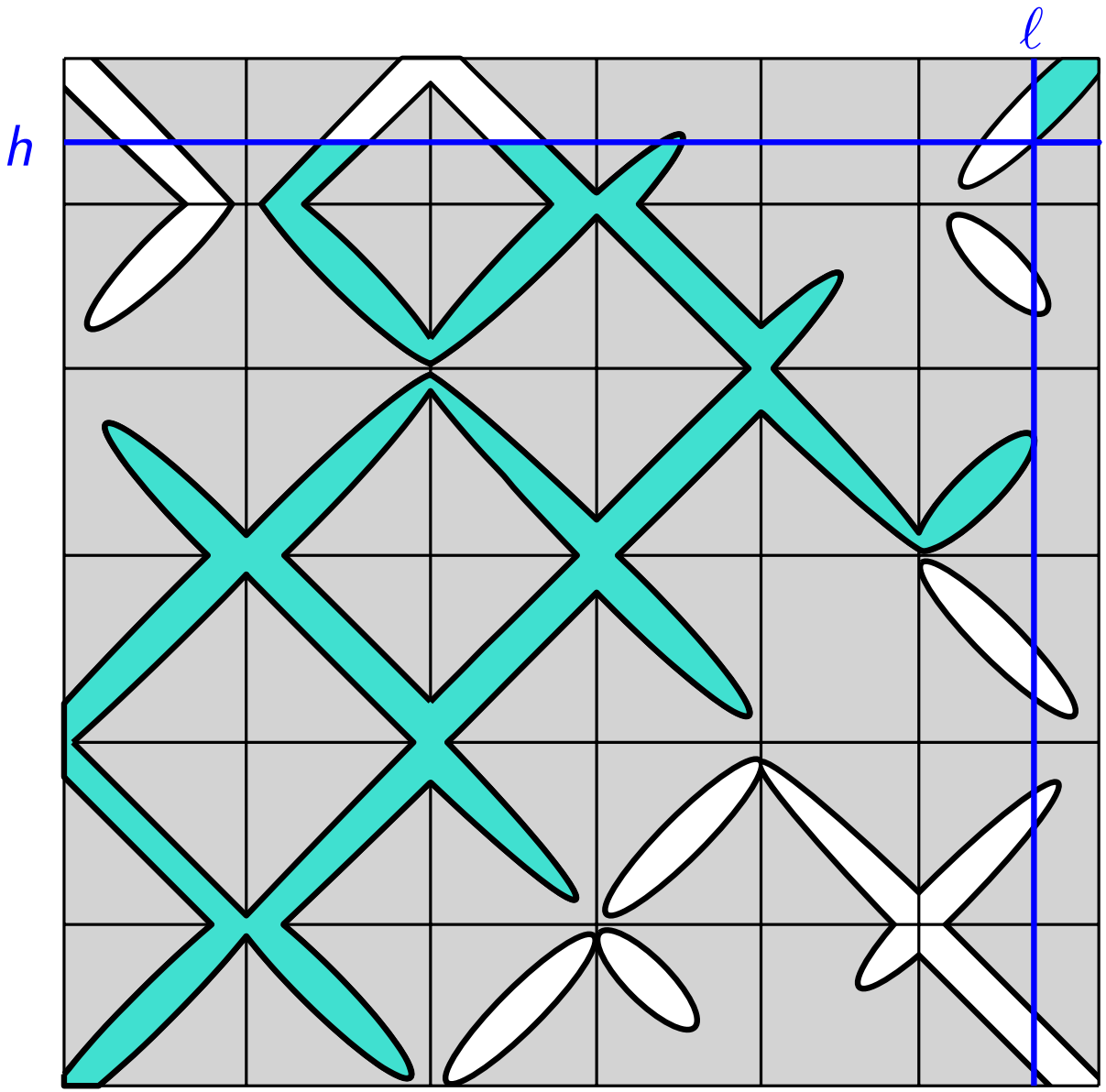
1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ
5. Determine horizontal cut h

Algorithm for $k = 2$



1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ
5. Determine horizontal cut h

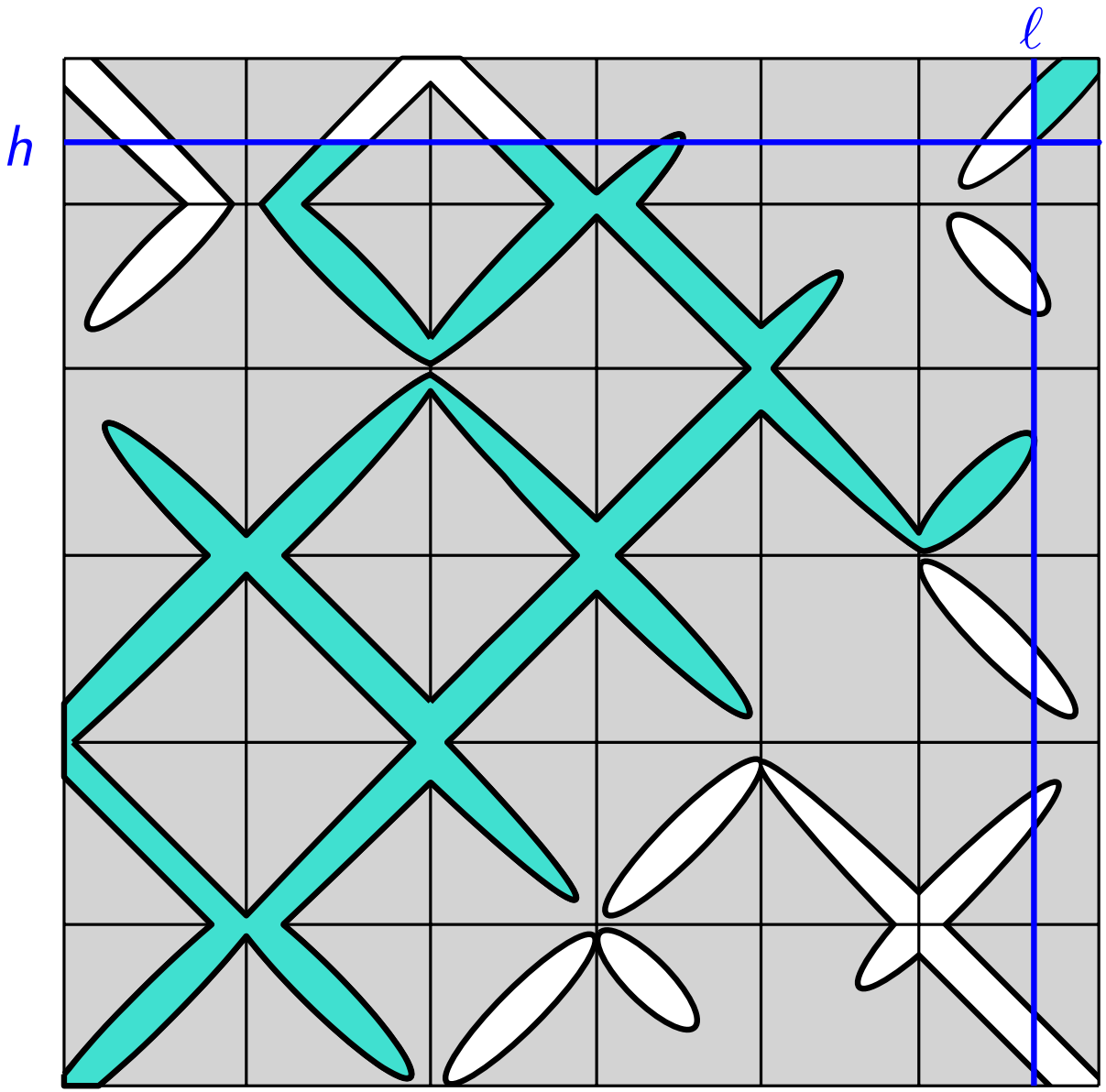
Algorithm for $k = 2$



1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ
5. Determine horizontal cut h

(6. Repeat for h at extremal points)

Algorithm for $k = 2$

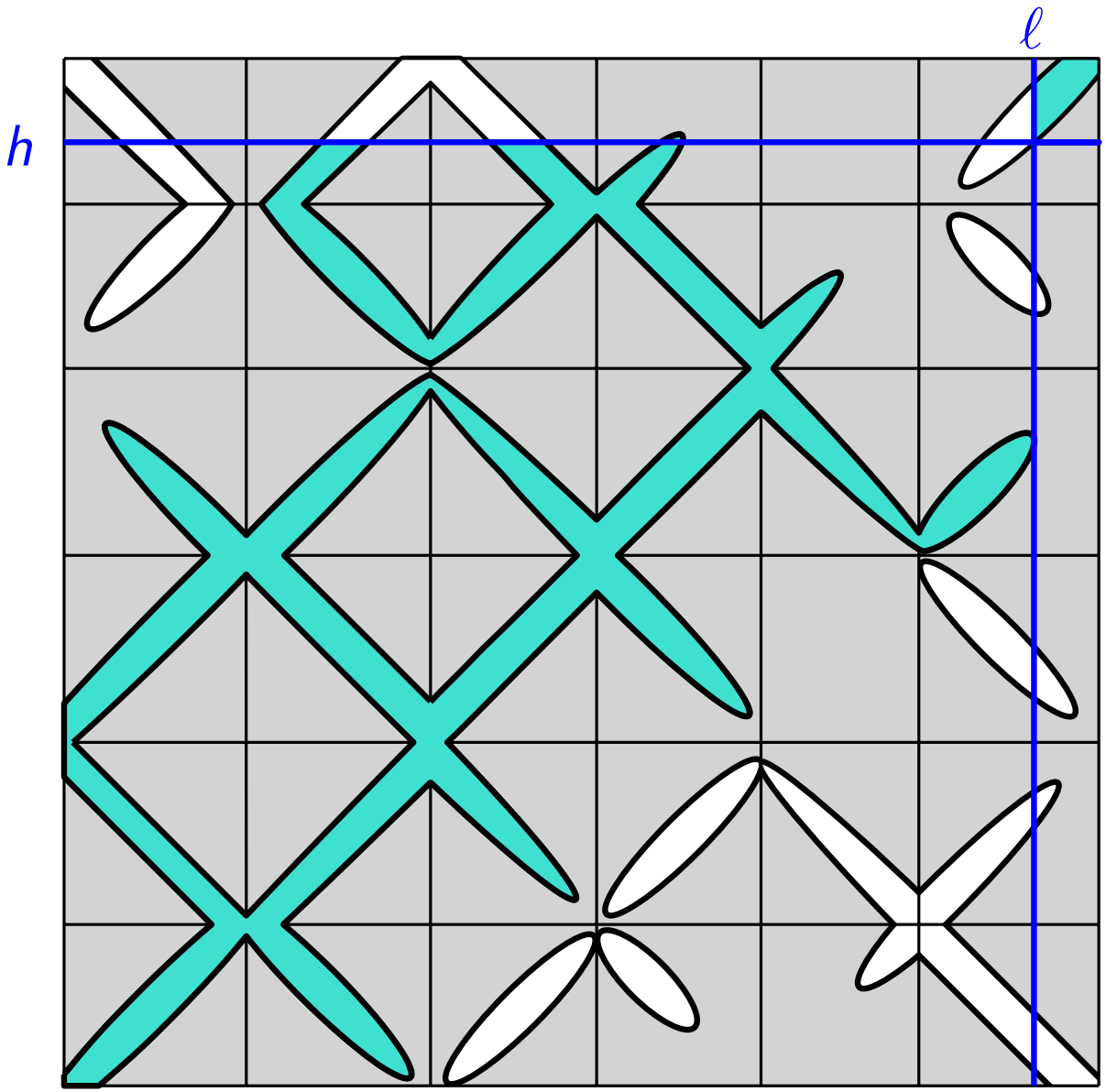


1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ
5. Determine horizontal cut h

(6. Repeat for h at extremal points)

(7. Repeat for opposite quadrants if necessary)

Algorithm for $k = 2$



1. Compute FSD
2. Identify *candidates* a_i, b_j
3. Compute cut lines ℓ at extremal points
4. Identify a_i^ℓ, b_j^ℓ
5. Determine horizontal cut h

(6. Repeat for h at extremal points)

(7. Repeat for opposite quadrants if necessary)

8. Return cut positions

RUB

Overview on the cut distance

RUB

Overview on the cut distance

- Deciding the cut distance is NP-hard in general

RUB

Overview on the cut distance

- Deciding the cut distance is NP-hard in general
- Optimizing k is APX-hard

RUB

Overview on the cut distance

- Deciding the cut distance is NP-hard in general
- Optimizing k is APX-hard
- $\mathcal{O}(n^2 \log n)$ time algorithm for $k = 2$

RUB

Overview on the cut distance

- Deciding the cut distance is NP-hard in general
- Optimizing k is APX-hard
- $\mathcal{O}(n^2 \log n)$ time algorithm for $k = 2$
- Conjecture: already NP-hard for $k = 3$

RUB

Overview on the cut distance

- Deciding the cut distance is NP-hard in general
- Optimizing k is APX-hard
- $\mathcal{O}(n^2 \log n)$ time algorithm for $k = 2$
- Conjecture: already NP-hard for $k = 3$

Thank you!