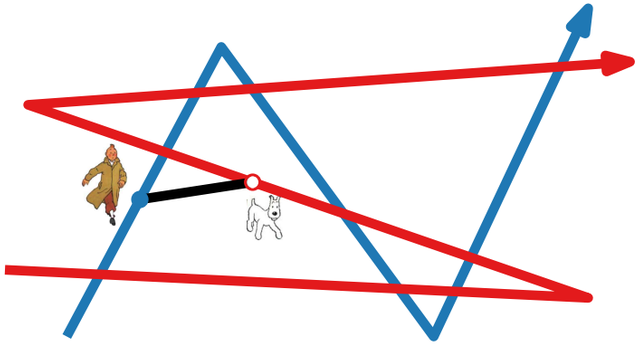
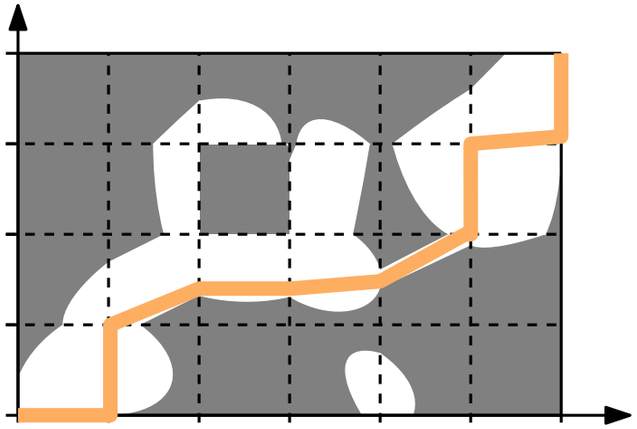
Universität Trier

Algorithmen für geographische Informationssysteme

3. Vorlesung
Map Matching



Teil I:
Problemstellung



Philipp Kindermann

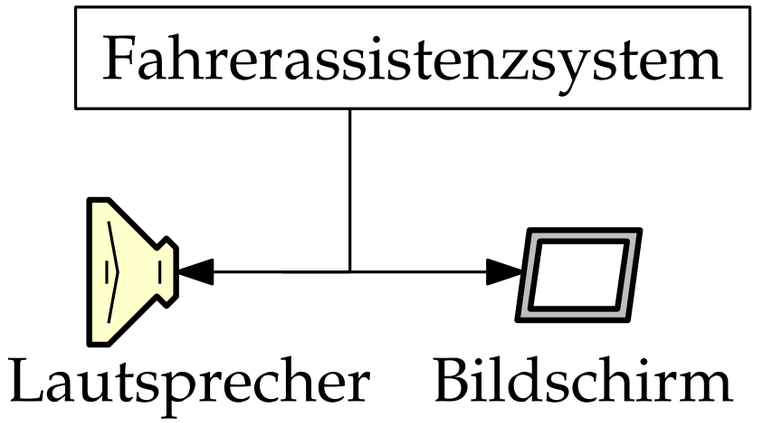
Wintersemester 2024/25

Navigationssysteme

Fahrerassistenzsystem

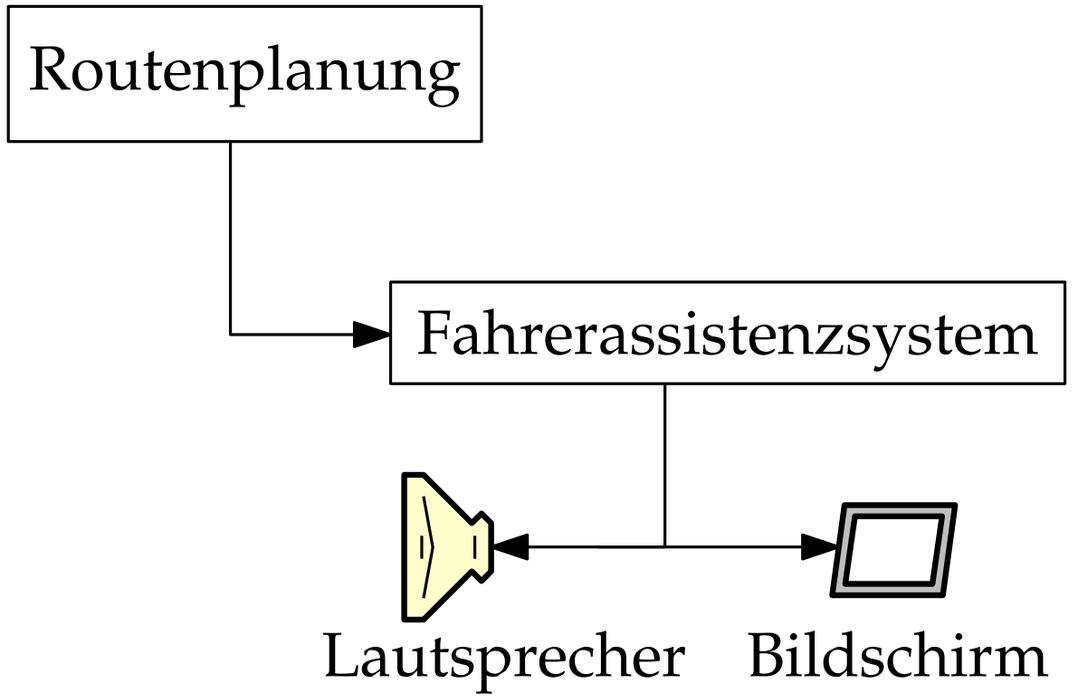
[Quddus, 2006]

Navigationssysteme



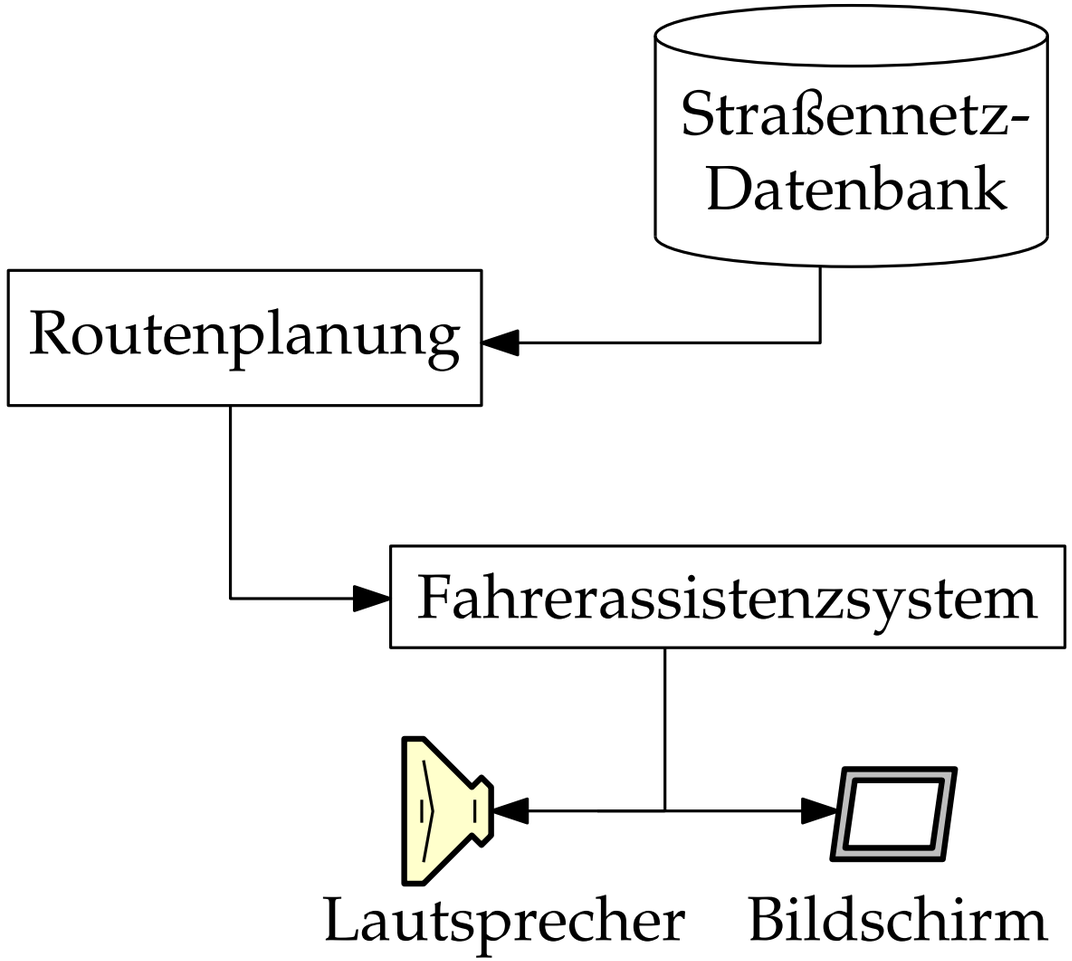
[Quddus, 2006]

Navigationssysteme



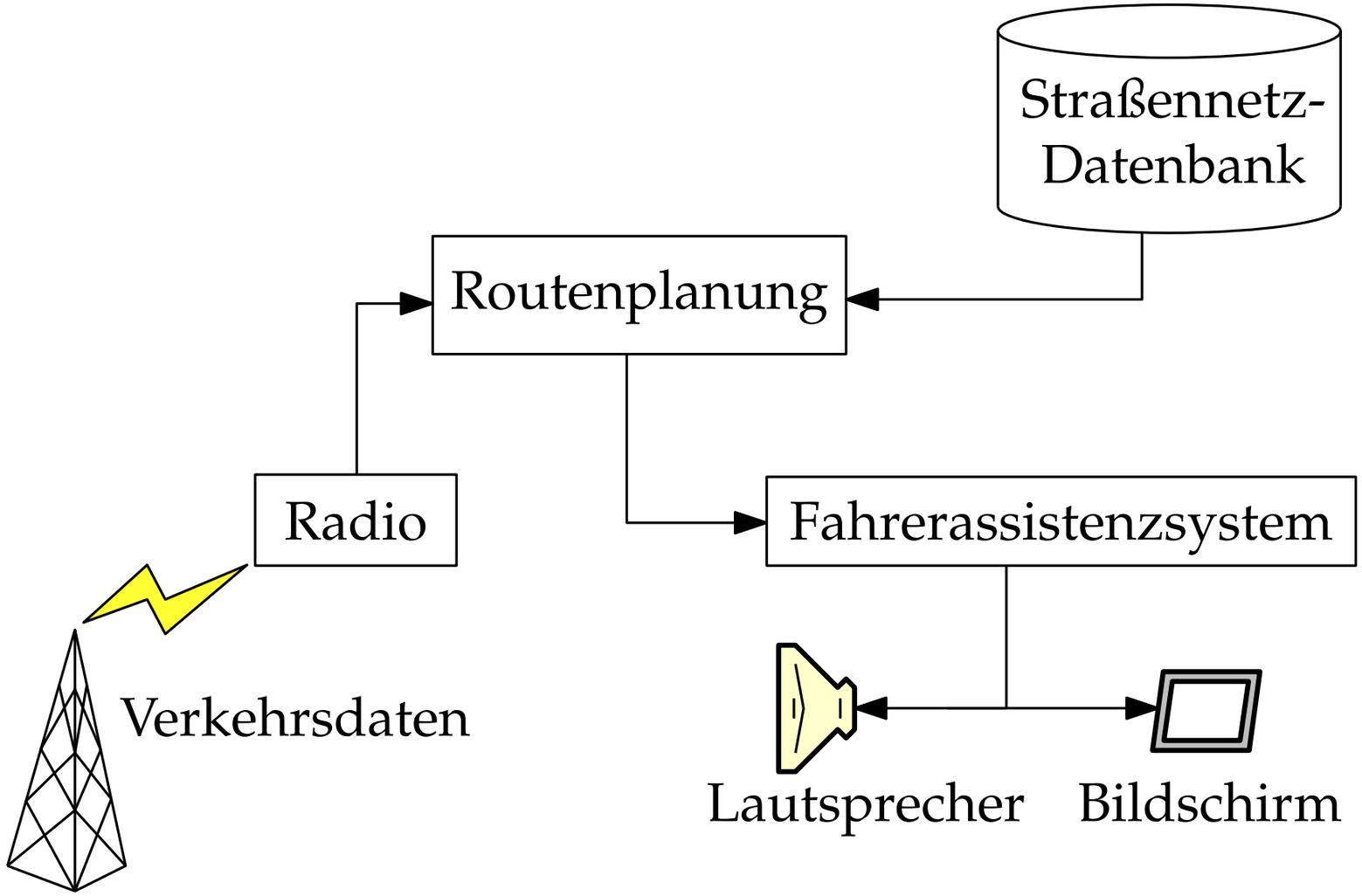
[Quddus, 2006]

Navigationssysteme



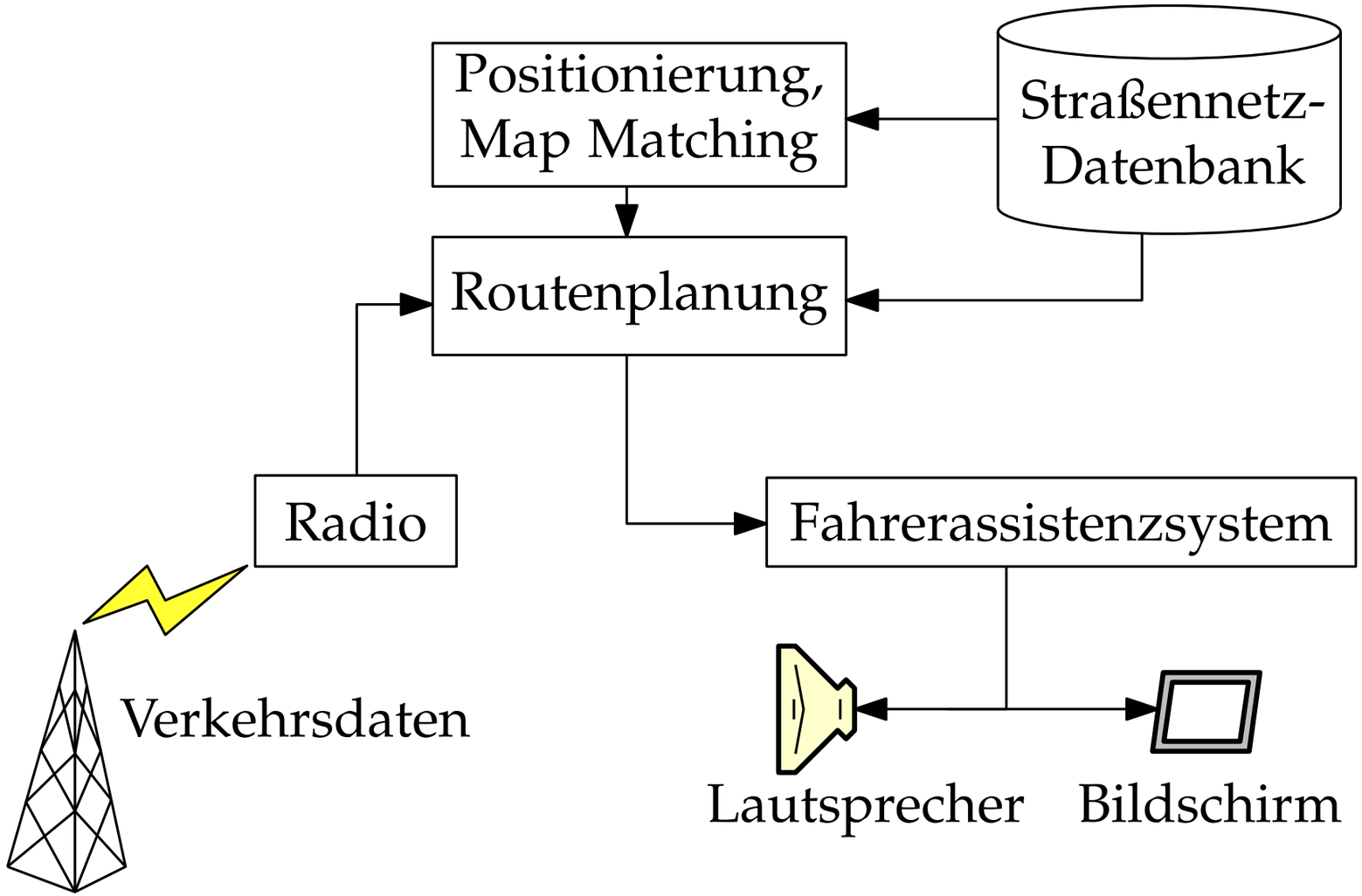
[Quddus, 2006]

Navigationssysteme



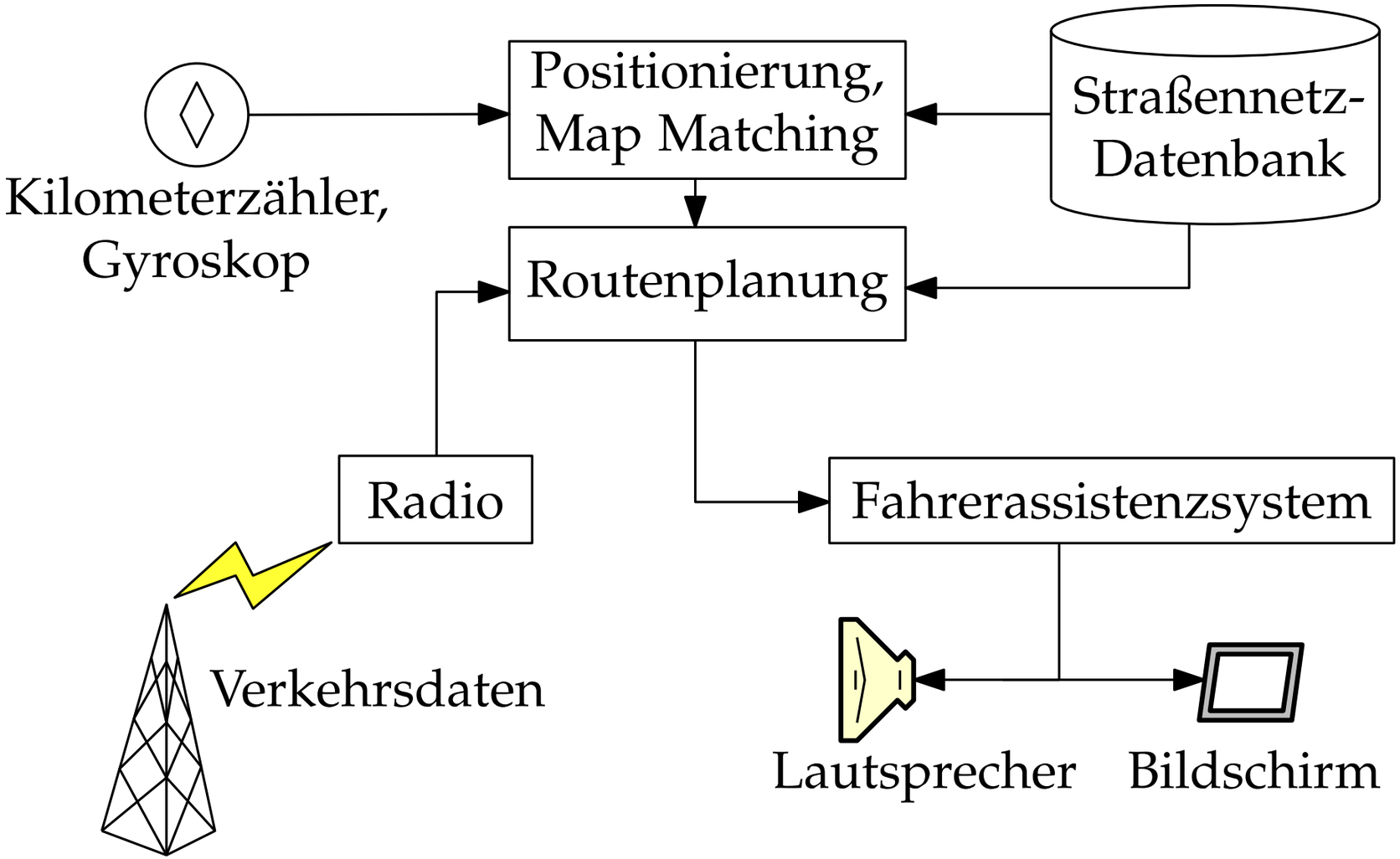
[Quddus, 2006]

Navigationssysteme



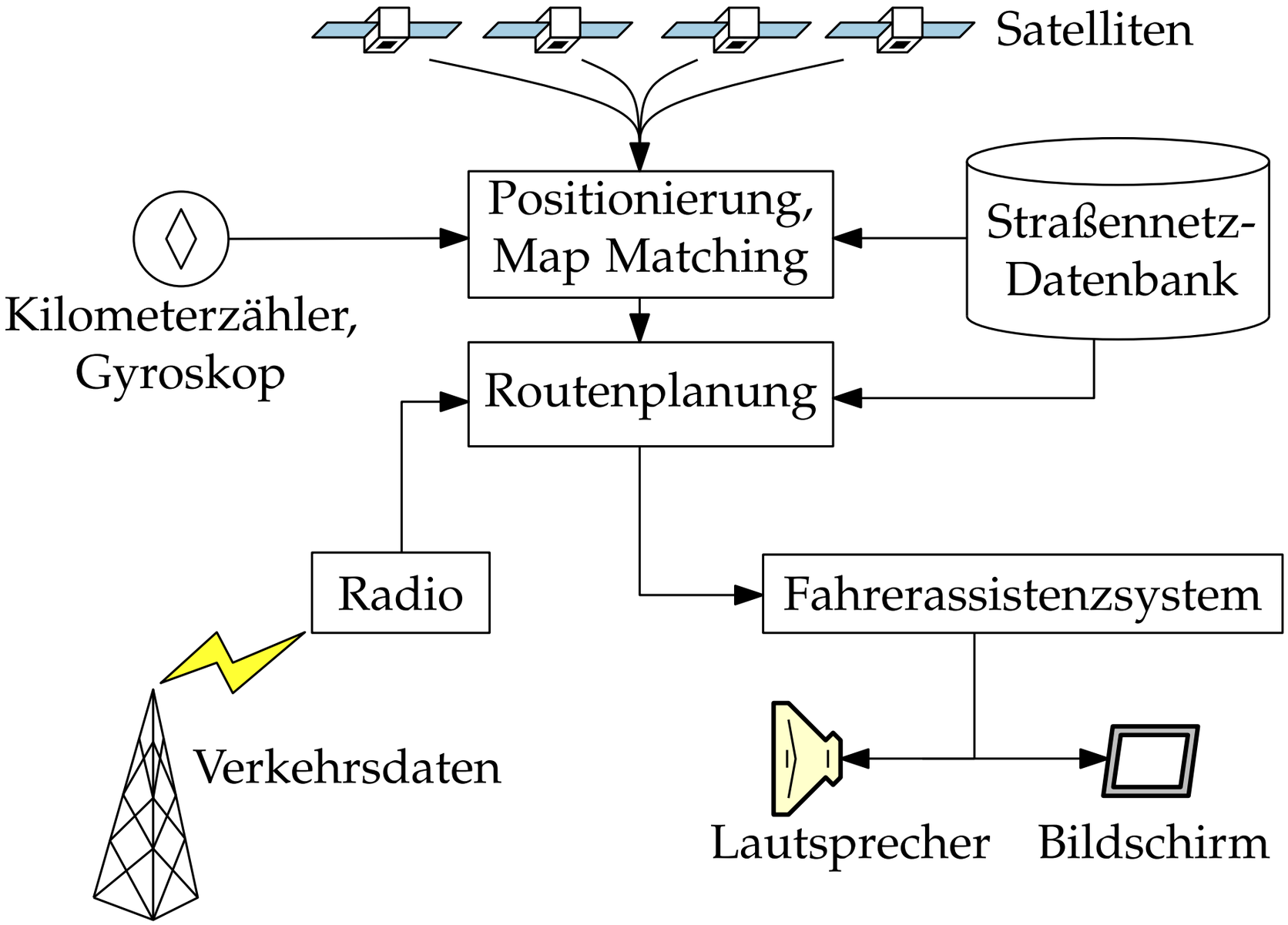
[Quddus, 2006]

Navigationssysteme



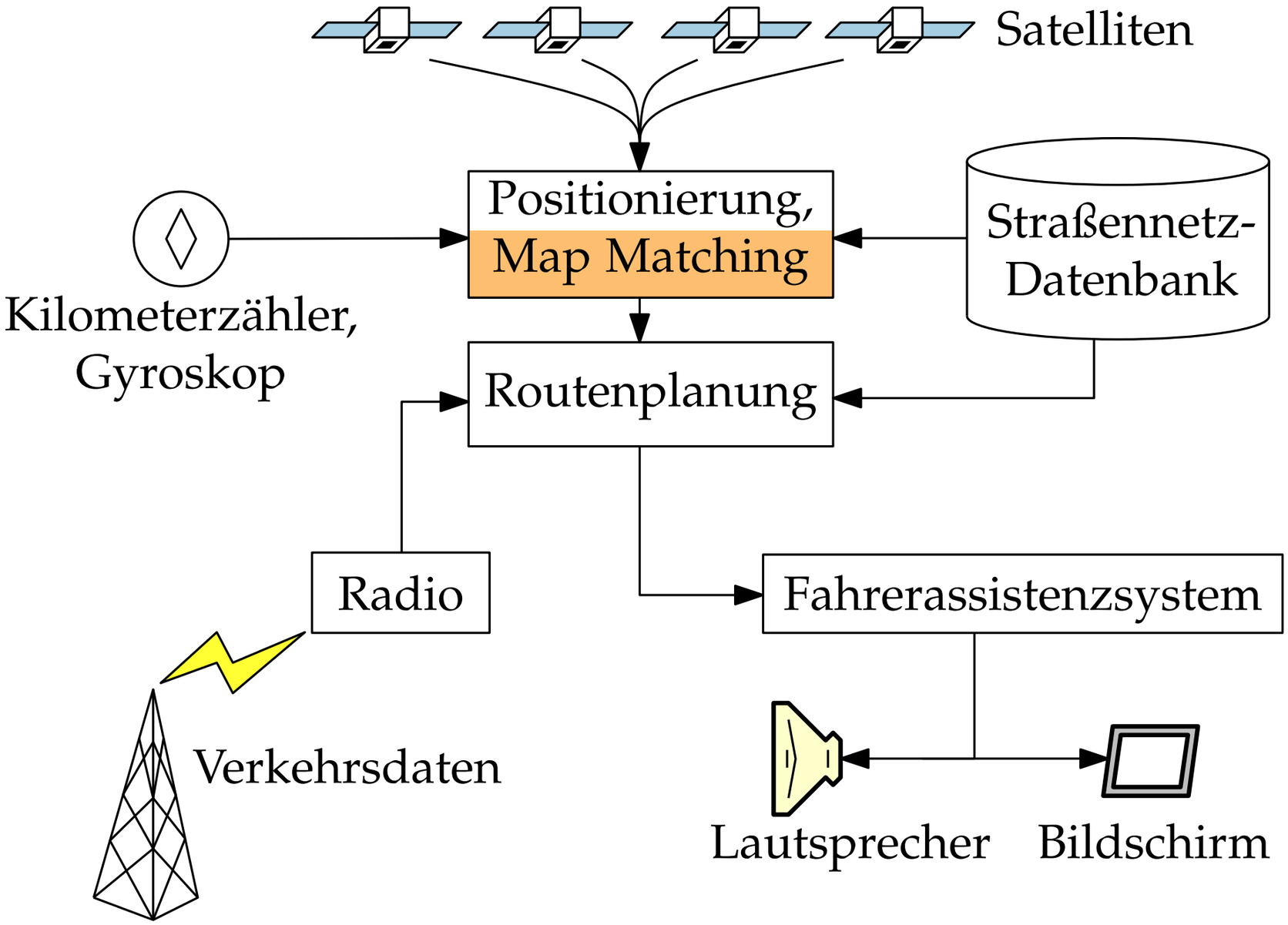
[Quddus, 2006]

Navigationssysteme



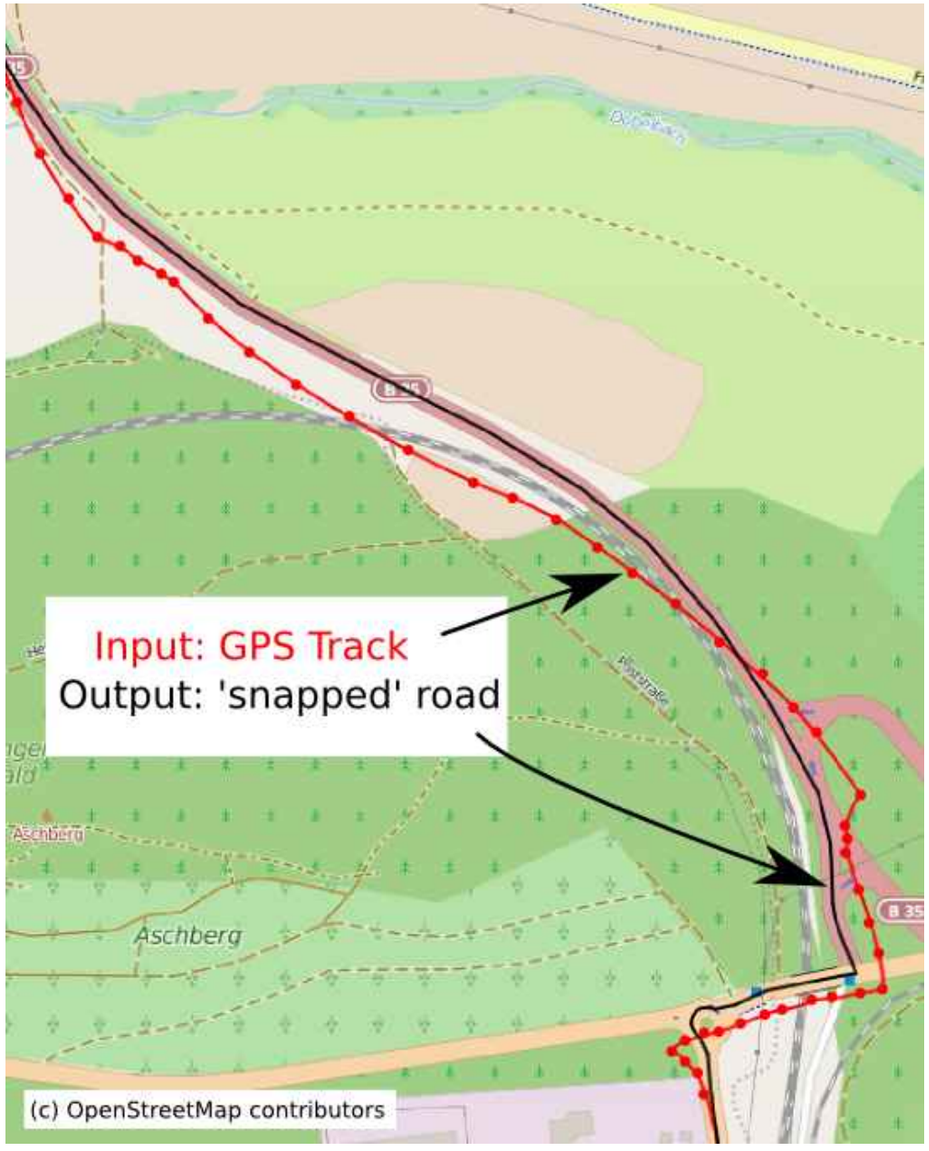
[Quddus, 2006]

Navigationssysteme



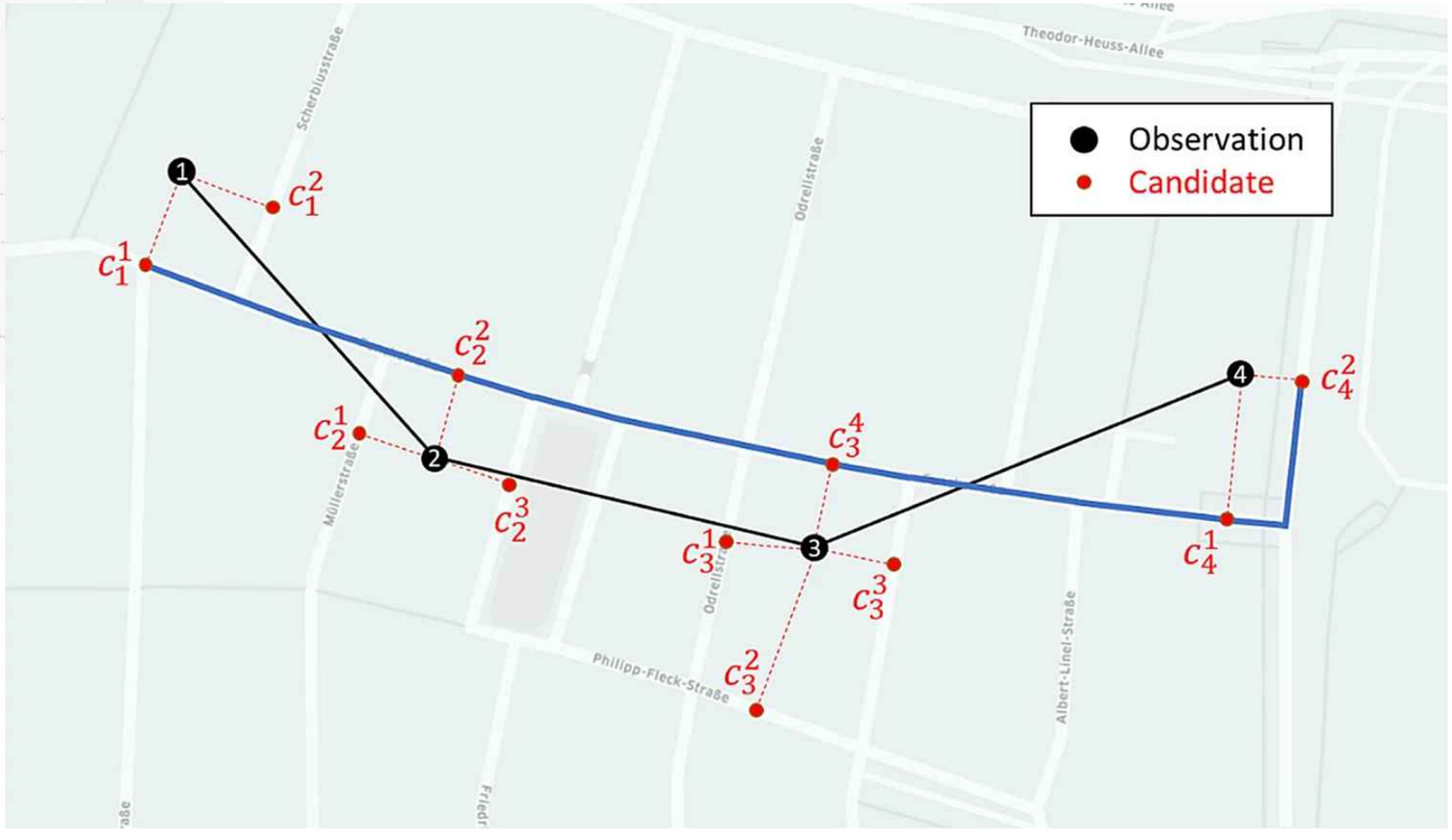
[Quddus, 2006]

Map Matching

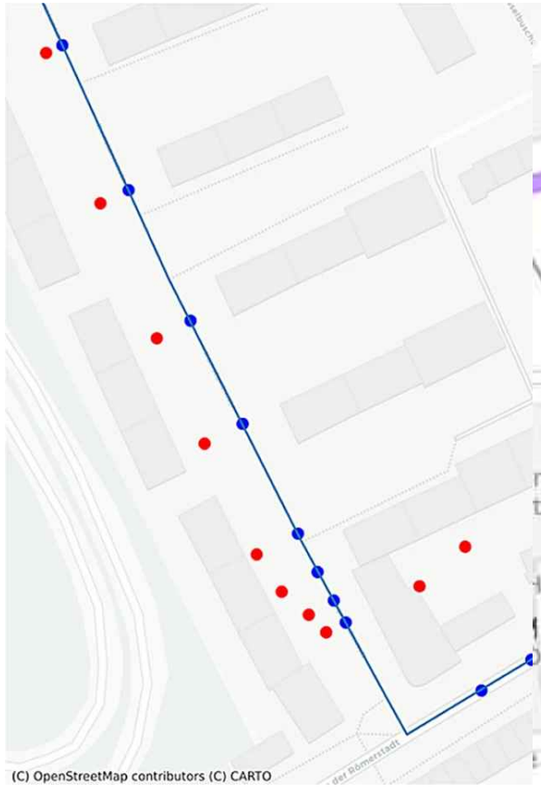


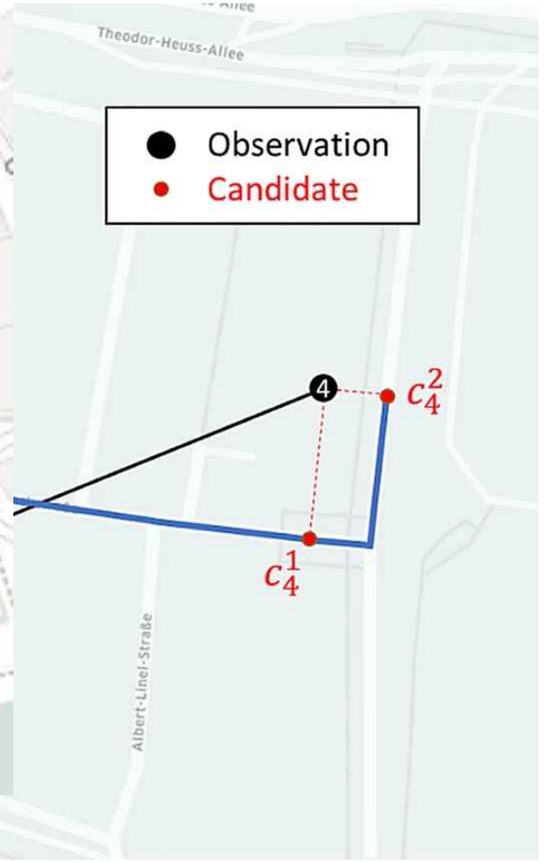
Map Matching

Map Matching

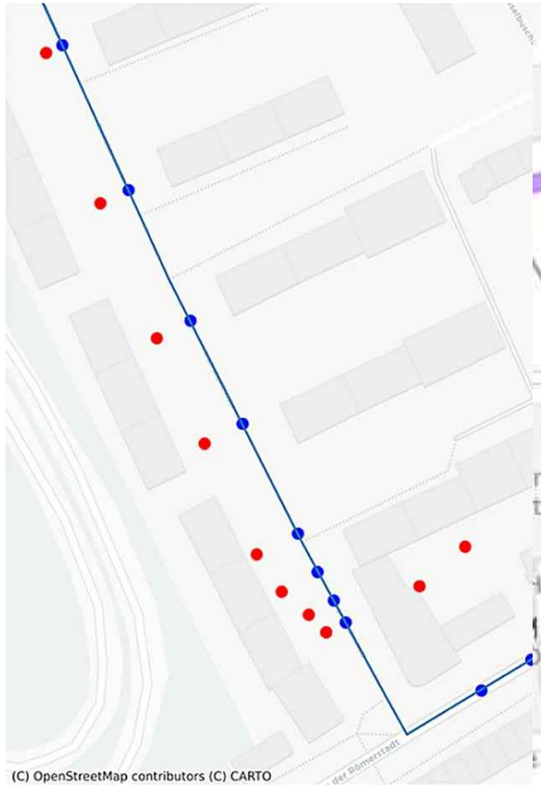


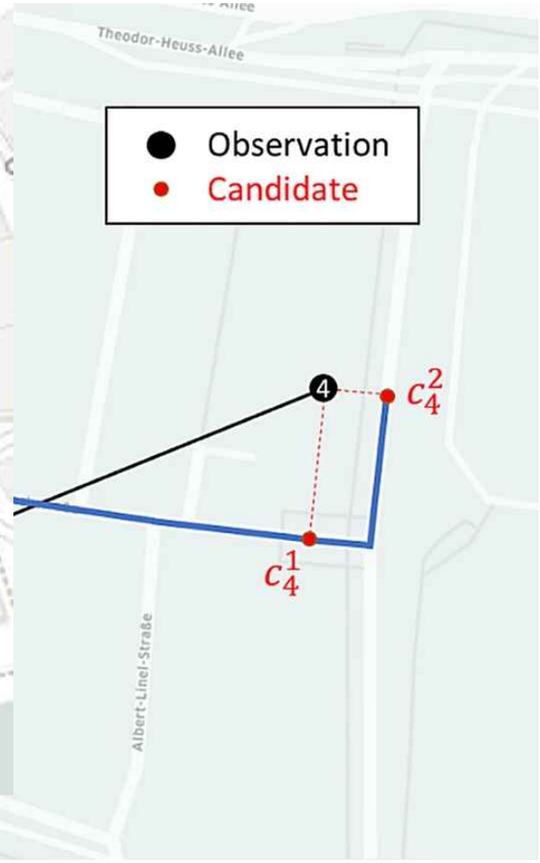
Map Matching





Map Matching





Demo. https://www.geoapify.com/map-matching-api

Schwierigkeiten

Schwierigkeiten

- Ungenaue Beobachtungen

Schwierigkeiten

- Ungenaue Beobachtungen

GPS: Positionsgenauigkeit ca. 20 m

[Wikipedia]

Schwierigkeiten

- Ungenaue Beobachtungen

GPS: Positionsgenauigkeit ca. 20 m

[Wikipedia]

- Abstrahierte/vereinfachte Repräsentation des Straßennetzes

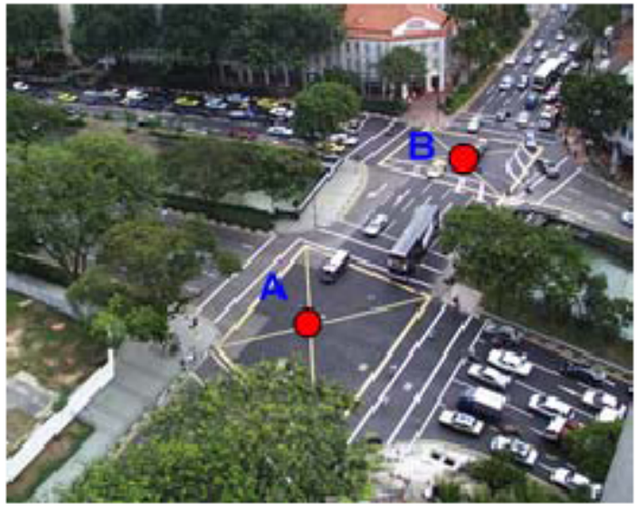
Schwierigkeiten

- Ungenaue Beobachtungen

GPS: Positionsgenauigkeit ca. 20 m

[Wikipedia]

- Abstrahierte/vereinfachte Repräsentation des Straßennetzes



tatsächliche Straßenkreuzung

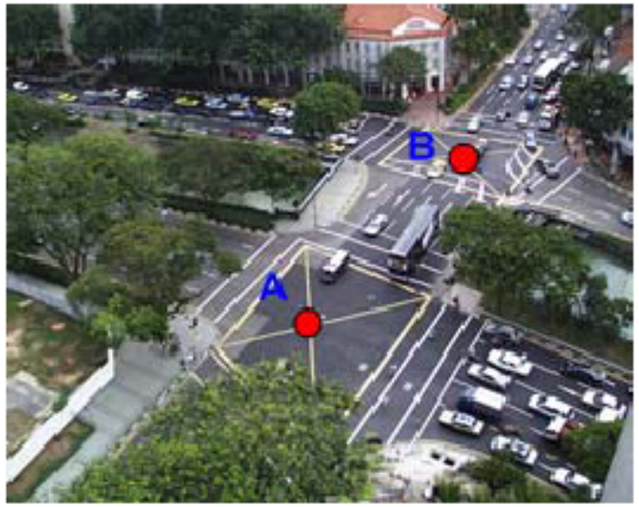
Schwierigkeiten

- Ungenaue Beobachtungen

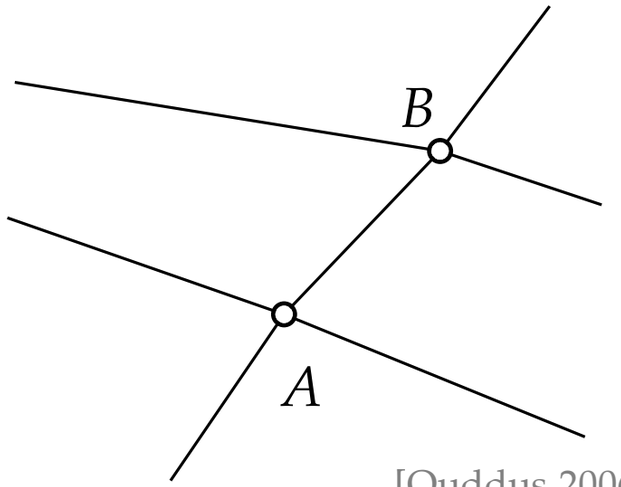
GPS: Positionsgenauigkeit ca. 20 m

[Wikipedia]

- Abstrahierte/vereinfachte Repräsentation des Straßennetzes



tatsächliche Straßenkreuzung



[Quddus 2006]

Repräsentation als Graph

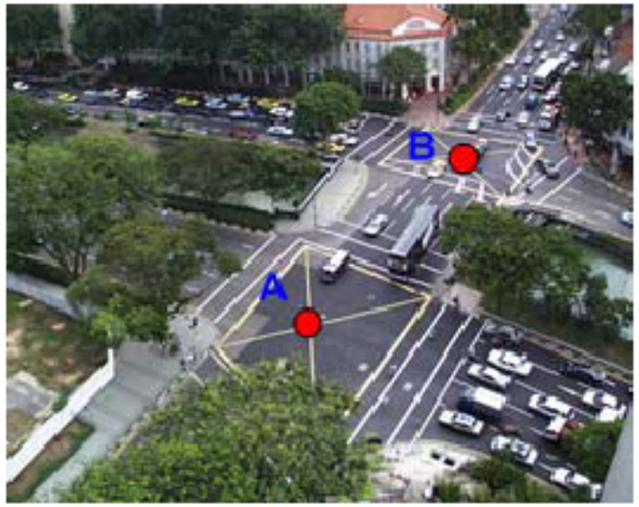
Schwierigkeiten

- Ungenaue Beobachtungen

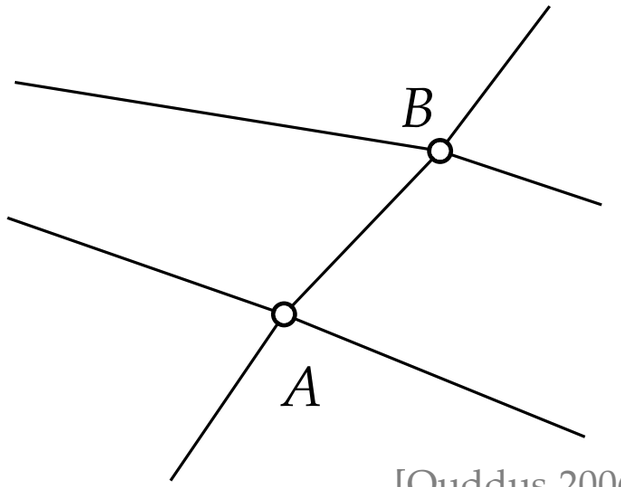
GPS: Positionsgenauigkeit ca. 20 m

[Wikipedia]

- Abstrahierte/vereinfachte Repräsentation des Straßennetzes

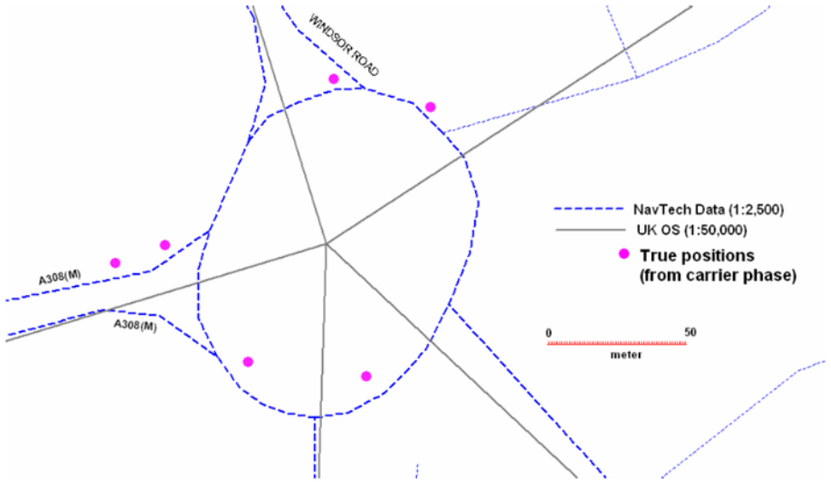


tatsächliche Straßenkreuzung



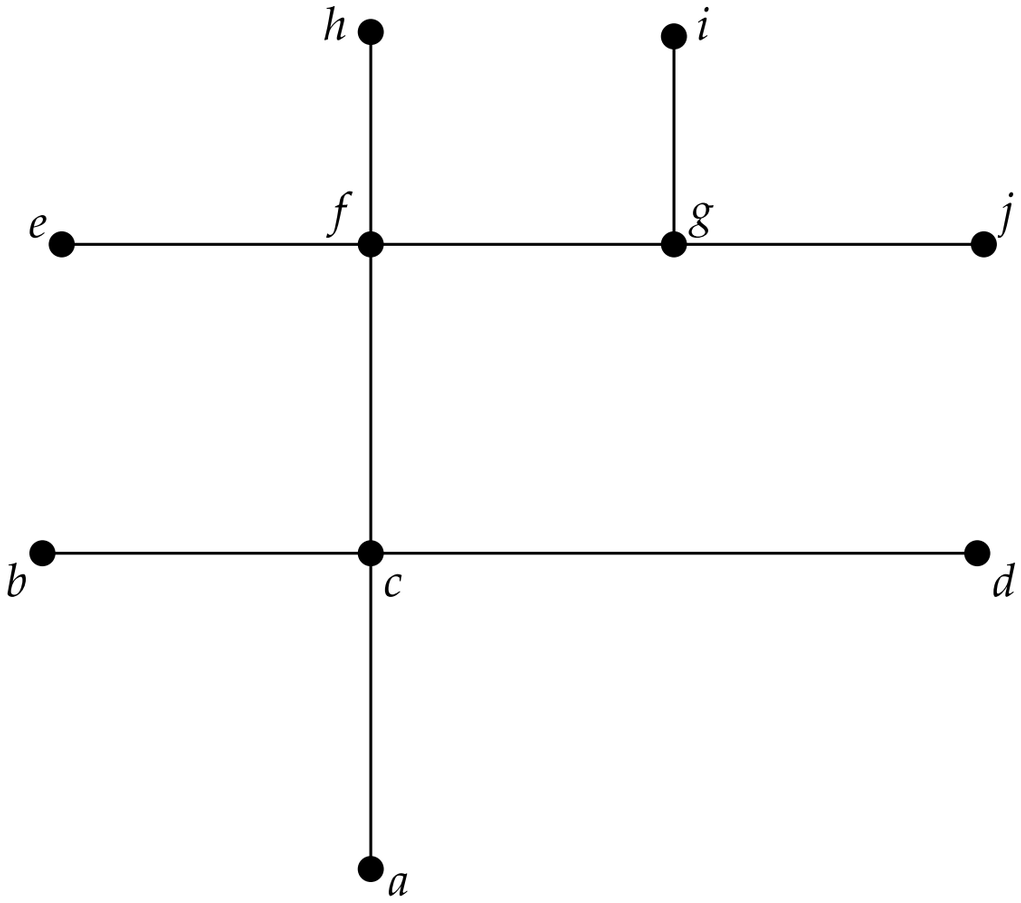
[Quddus 2006]

Repräsentation als Graph

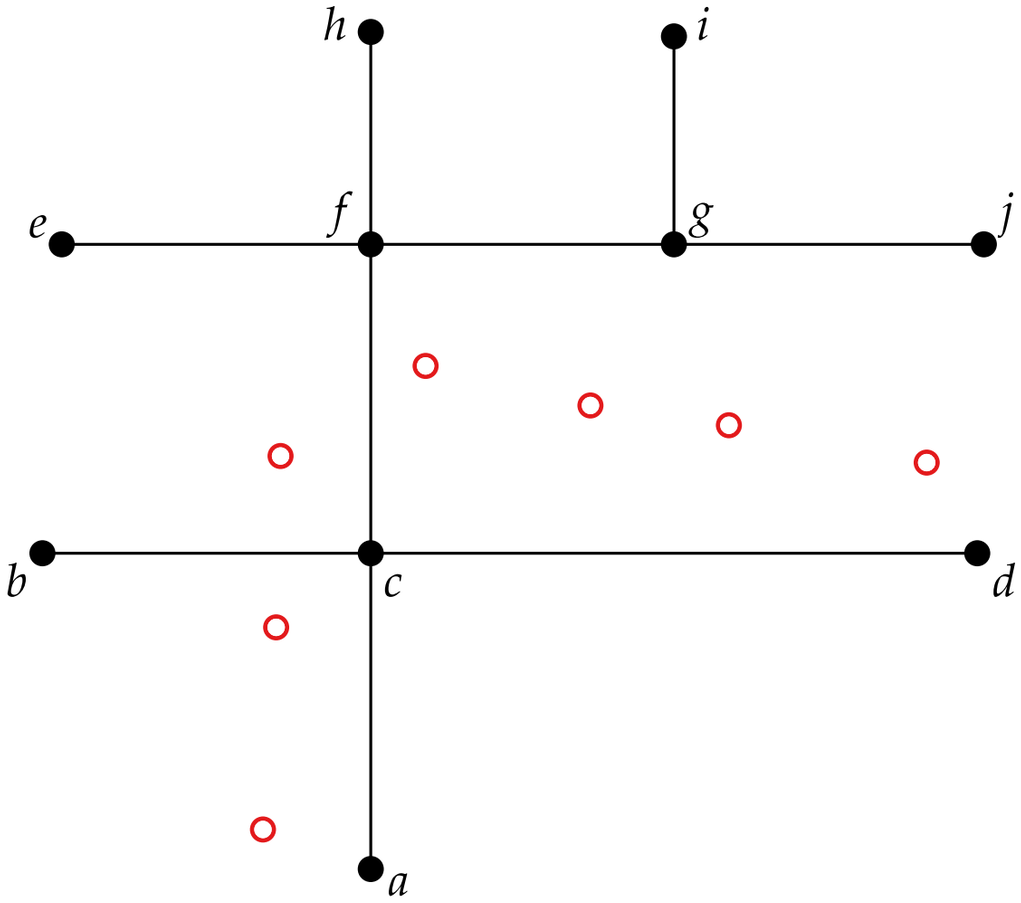


Unterschiede zweier Datensätze

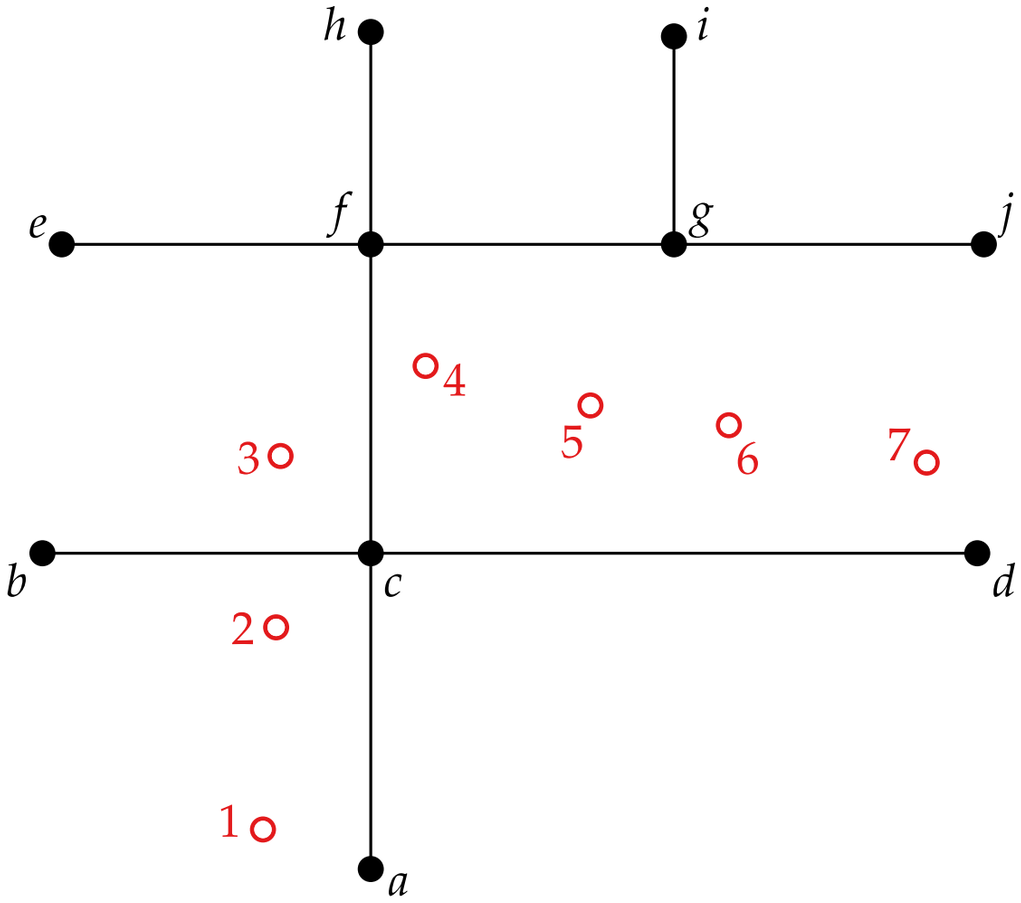
Naive Ansätze



Naive Ansätze

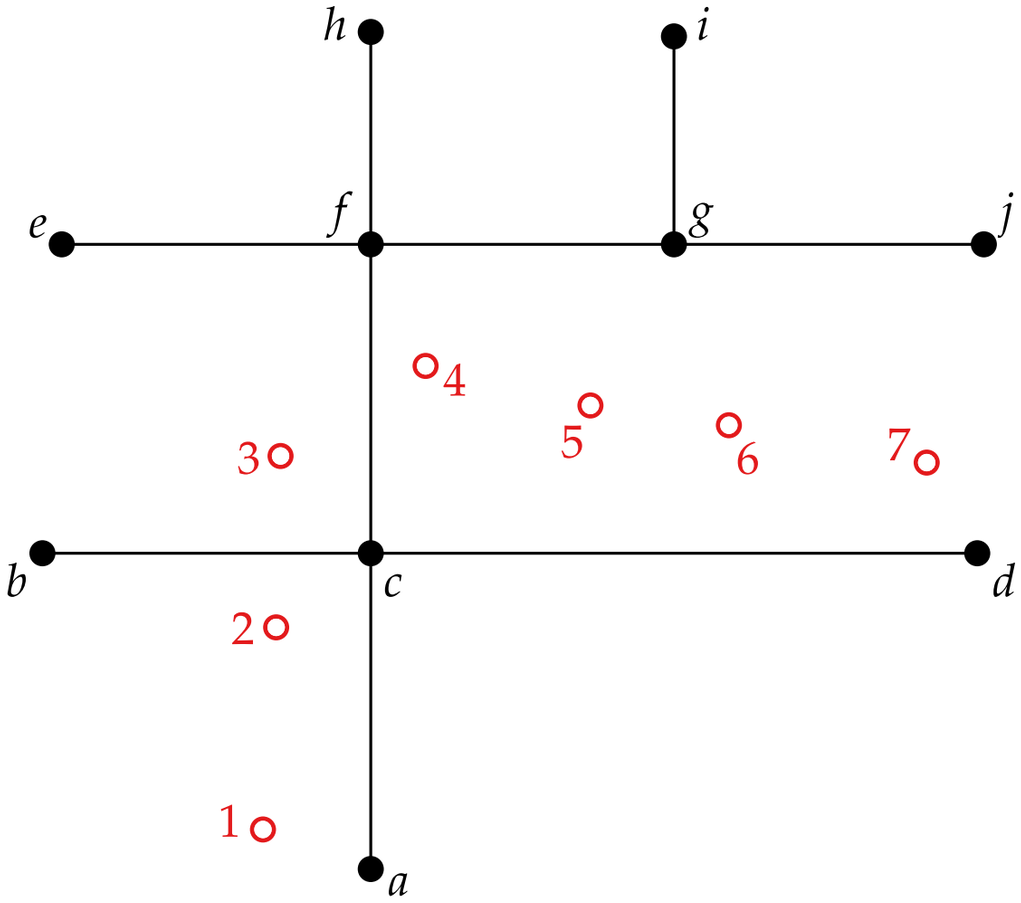


Naive Ansätze



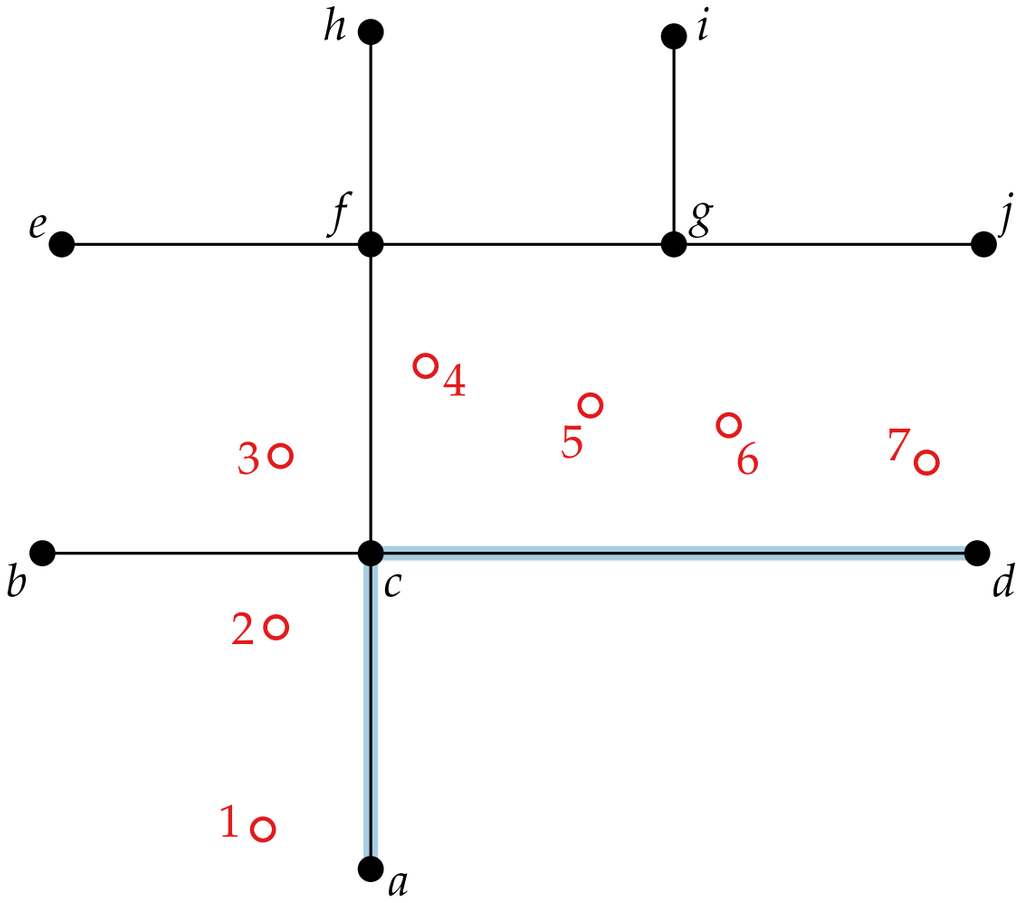
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?



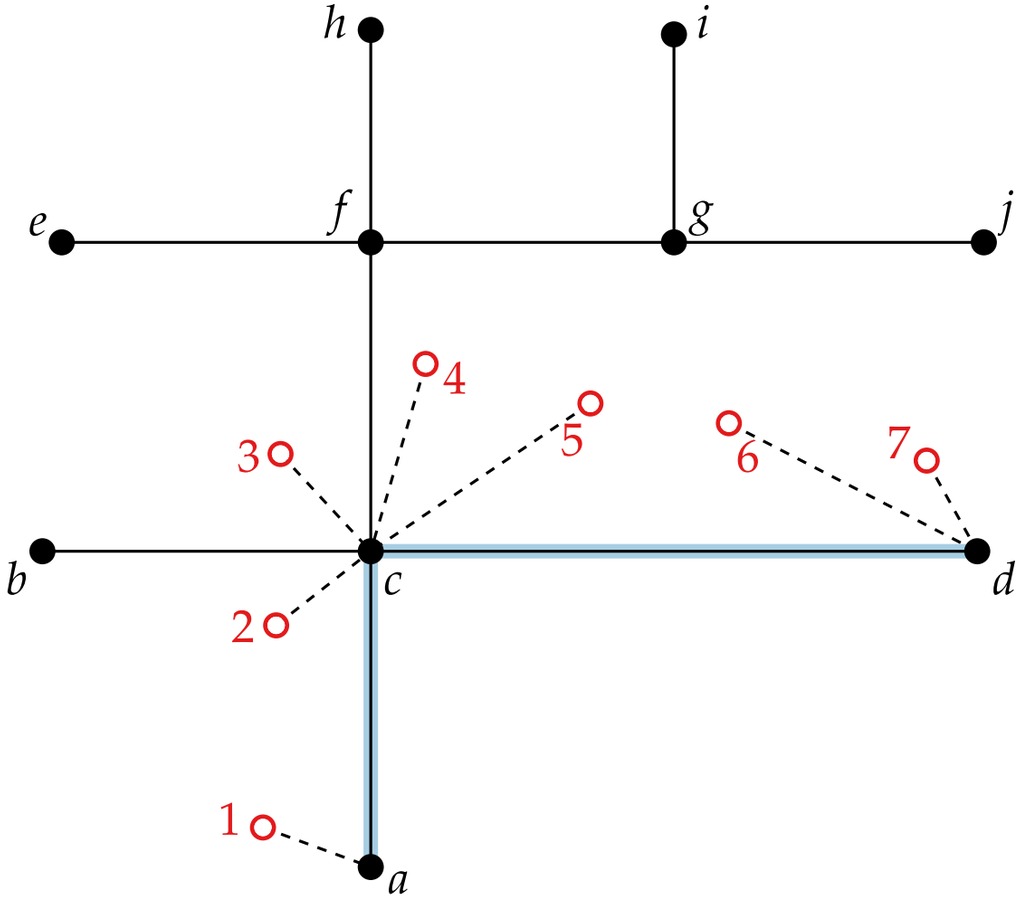
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?



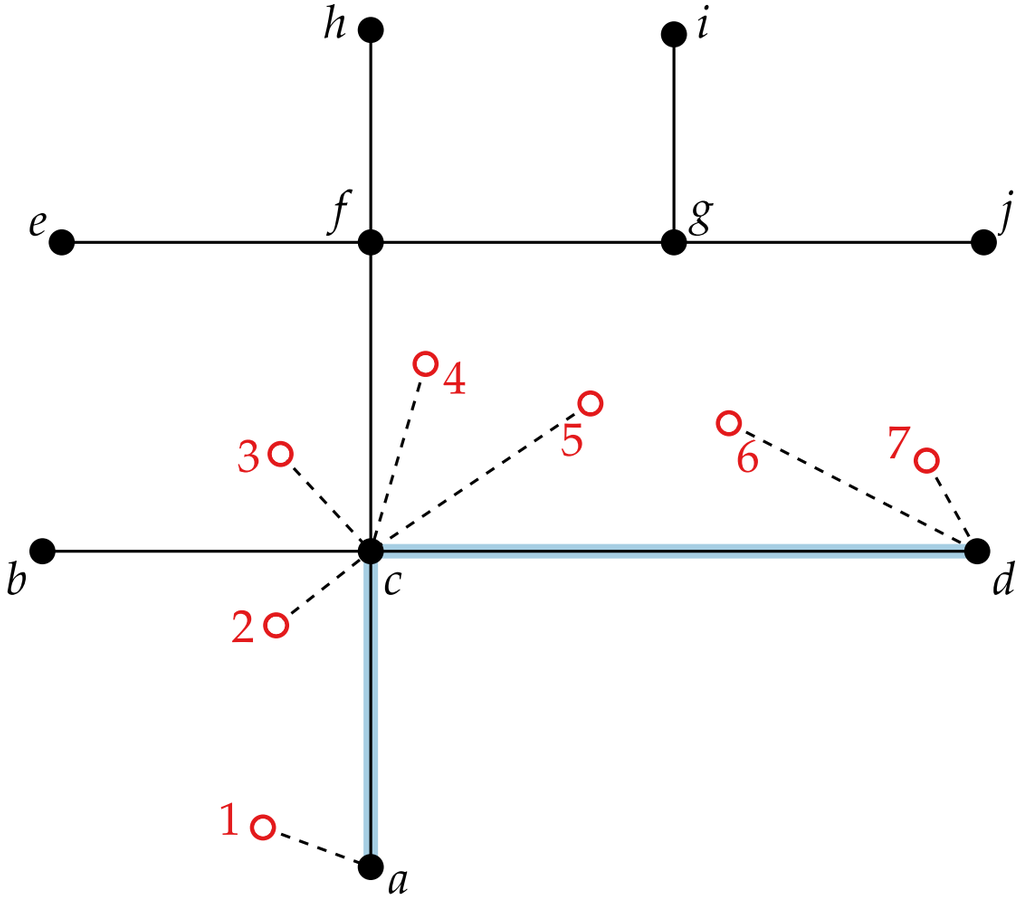
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?



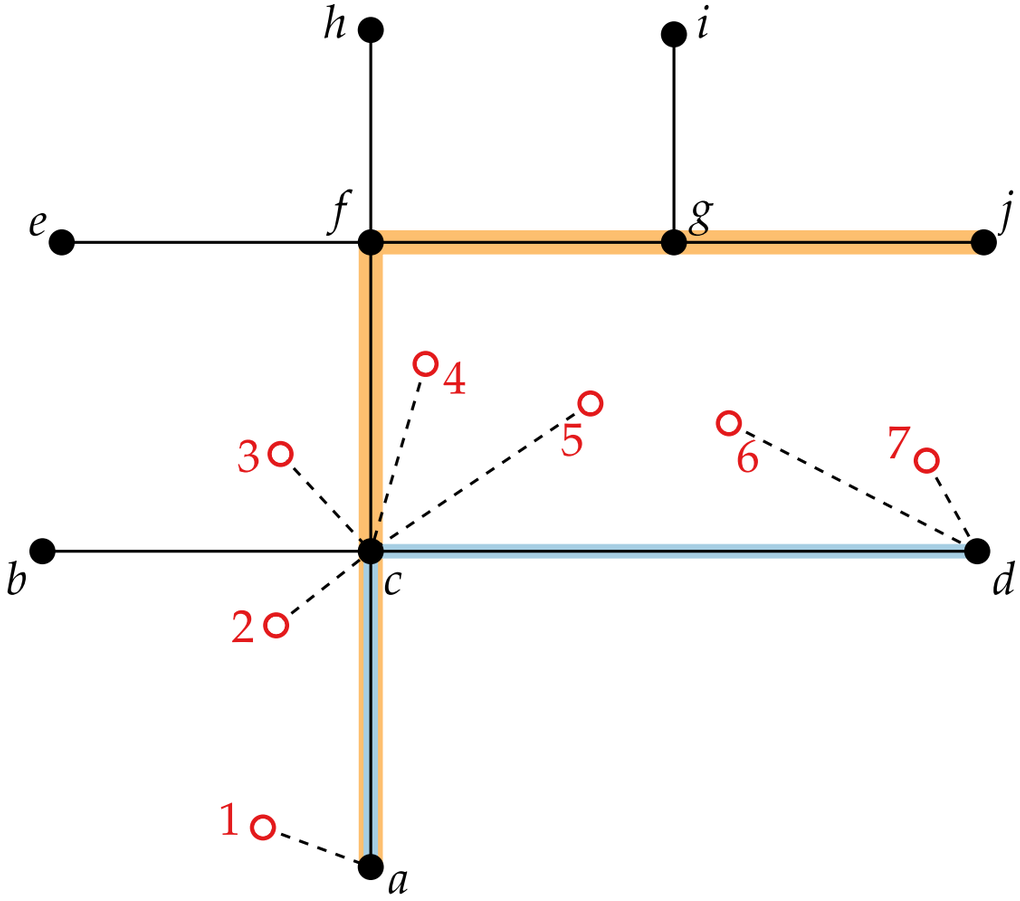
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?



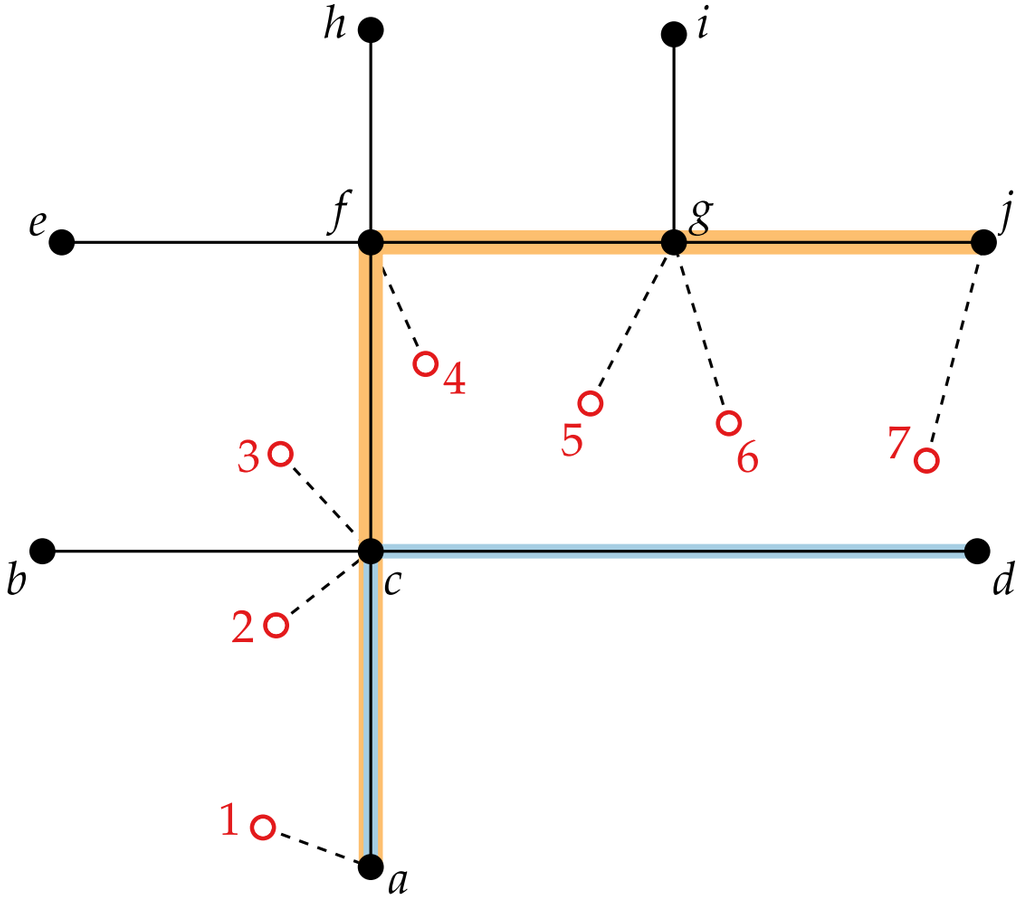
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?



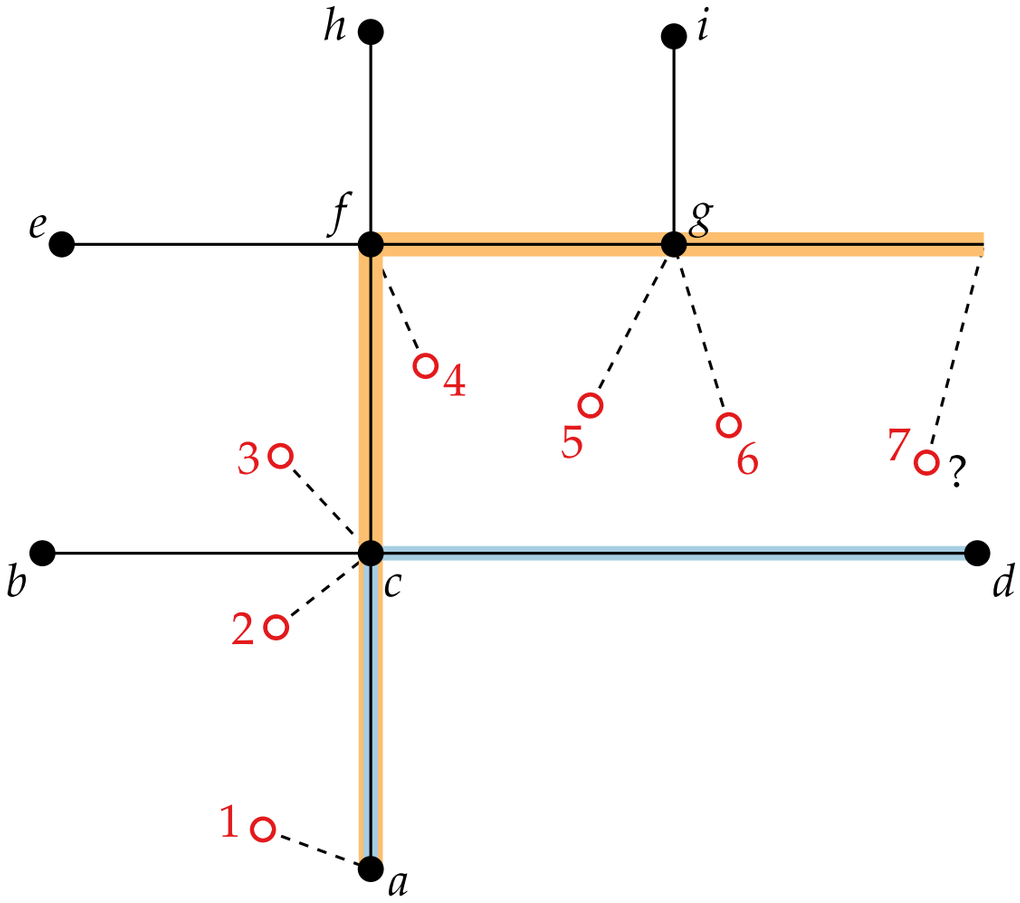
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?



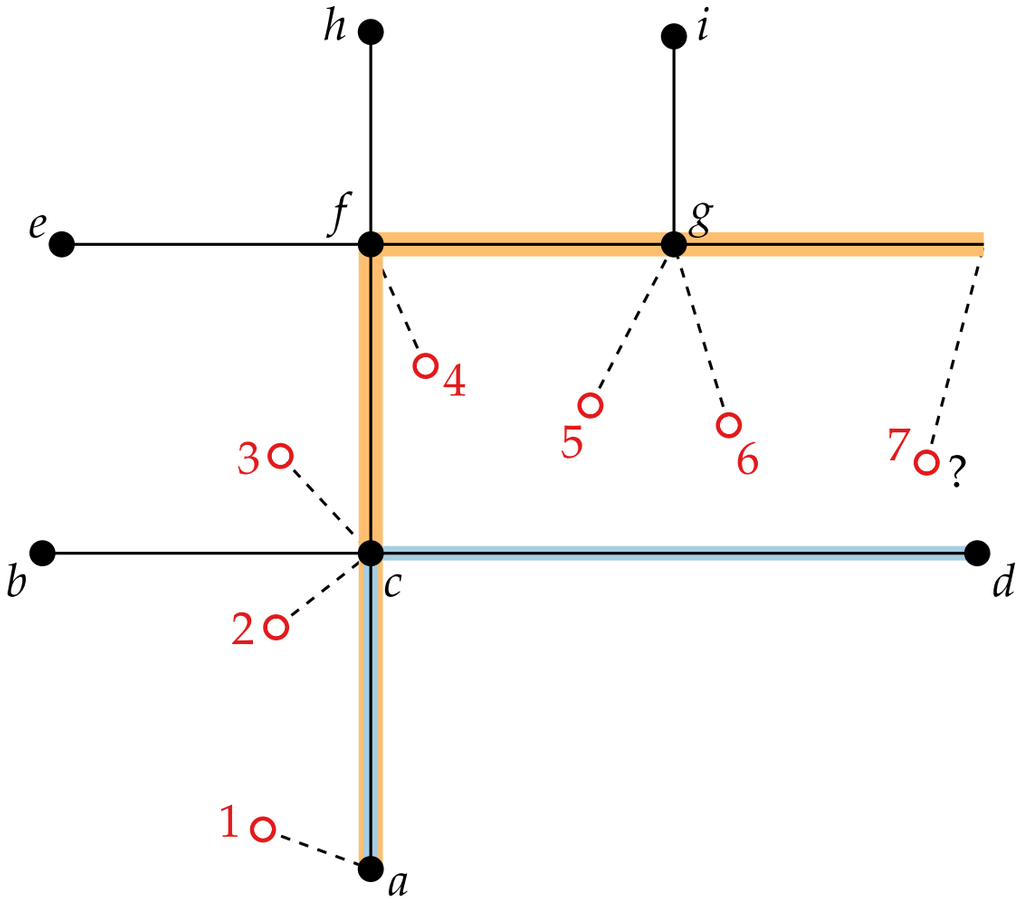
Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?



Naive Ansätze

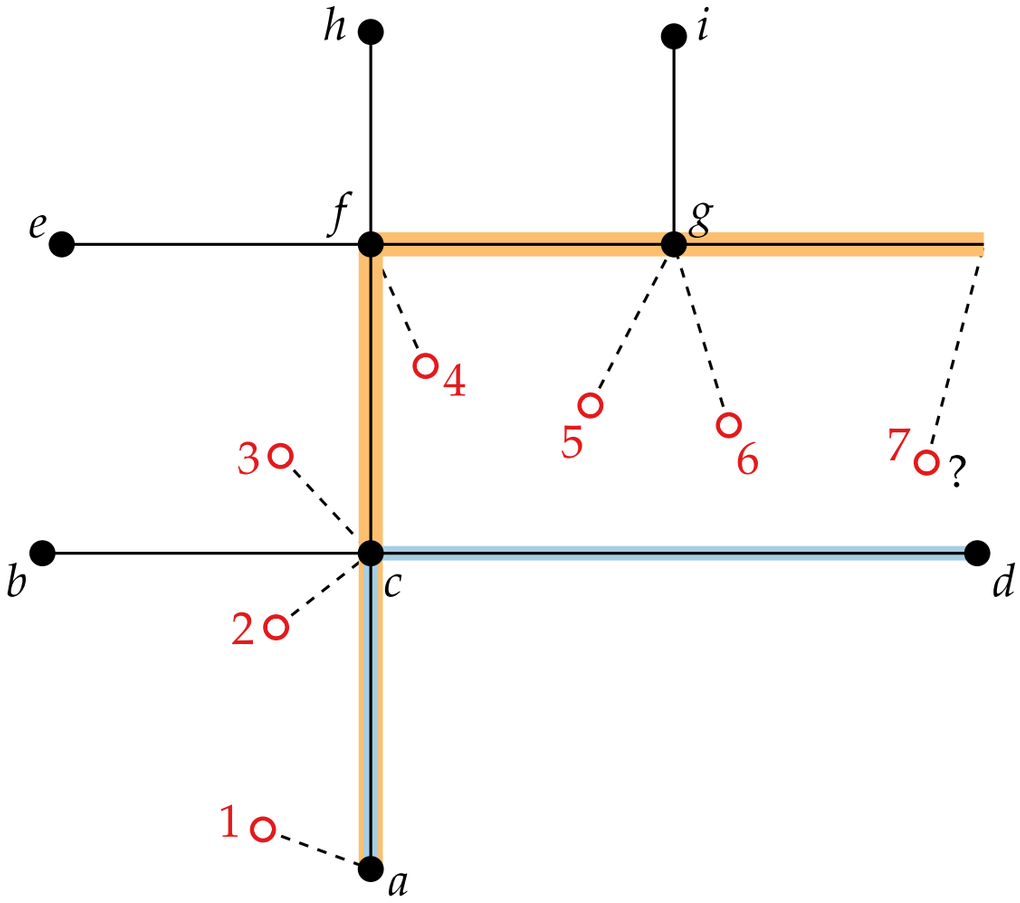
- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt



Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

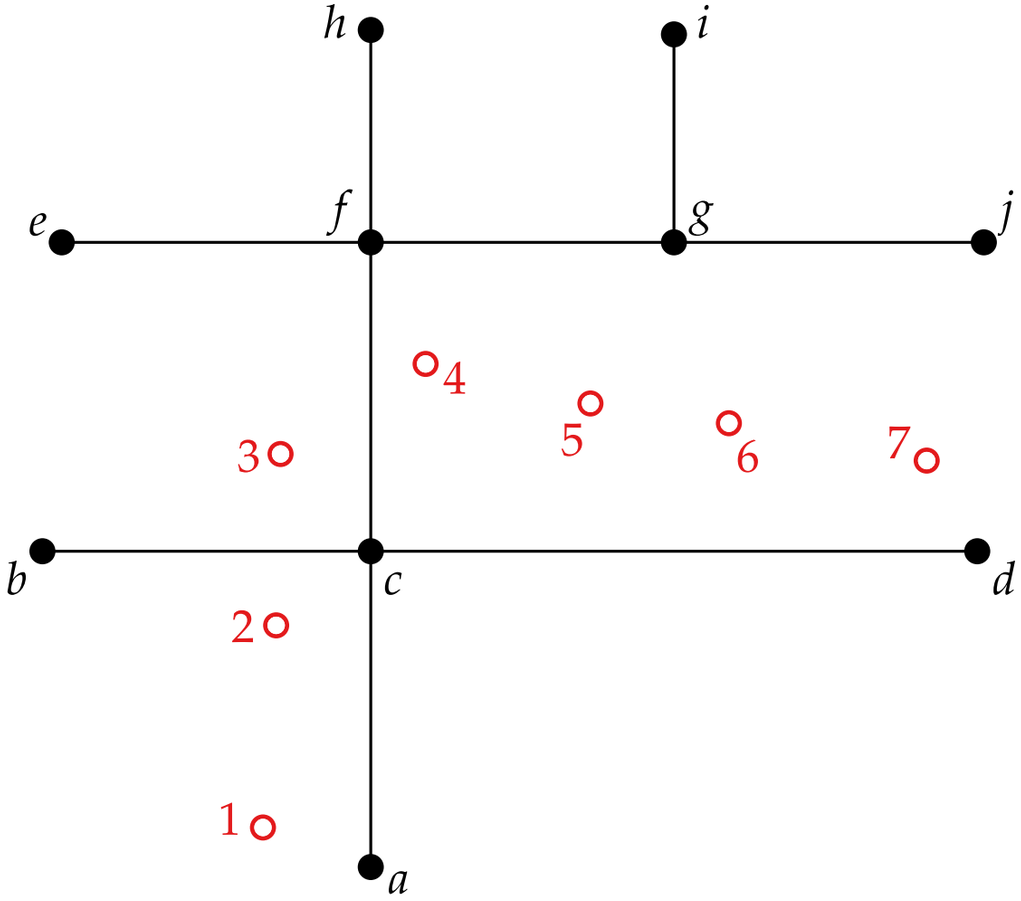
Ideen für Berechnung?



Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

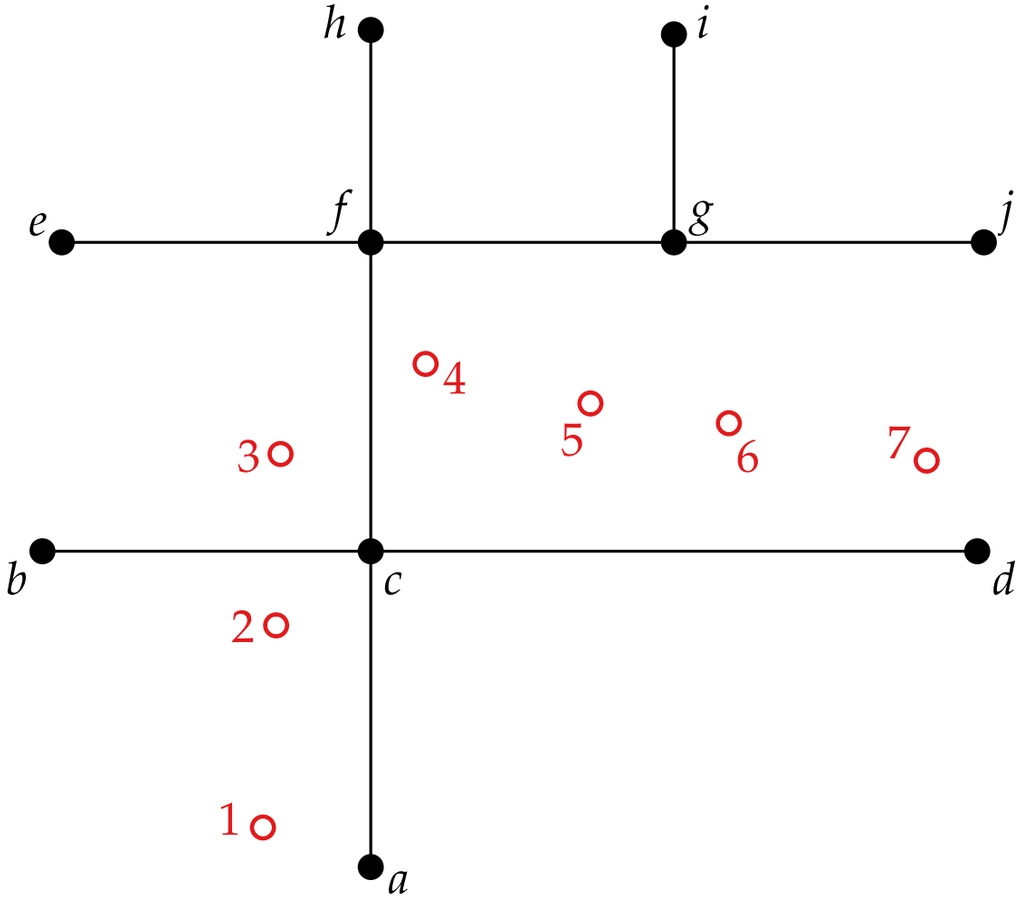


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung

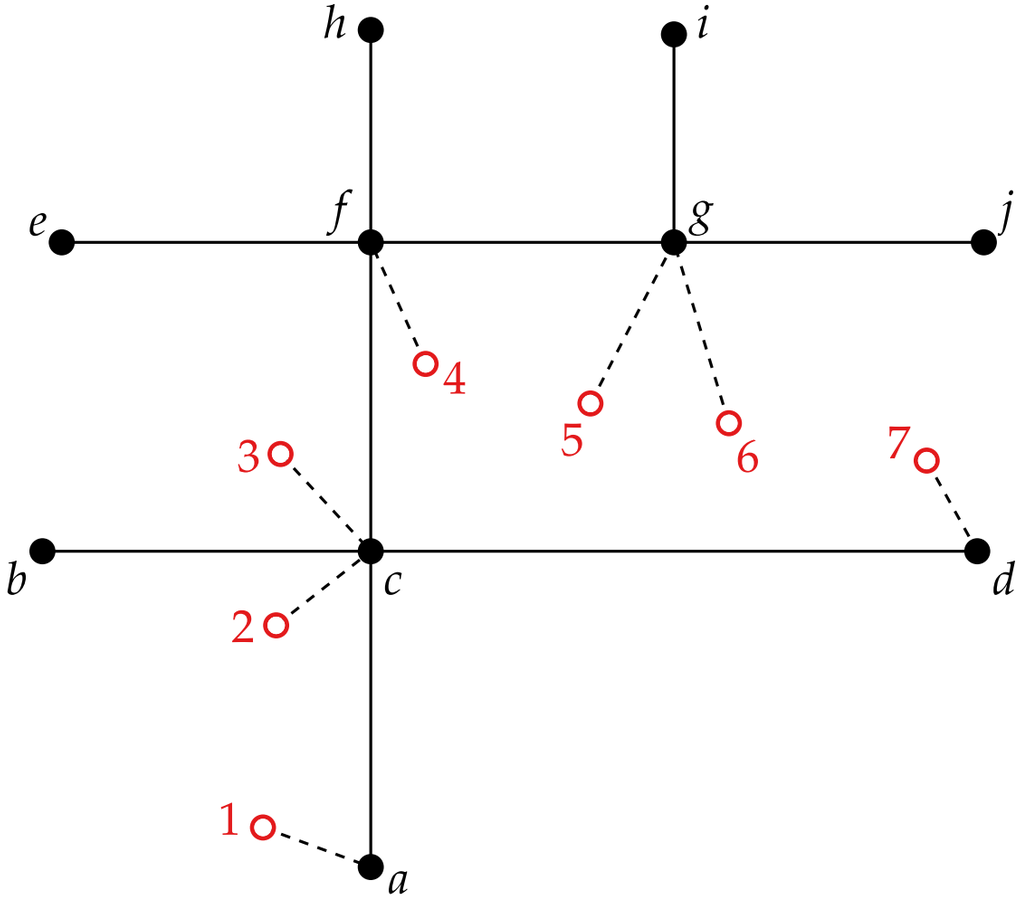


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung

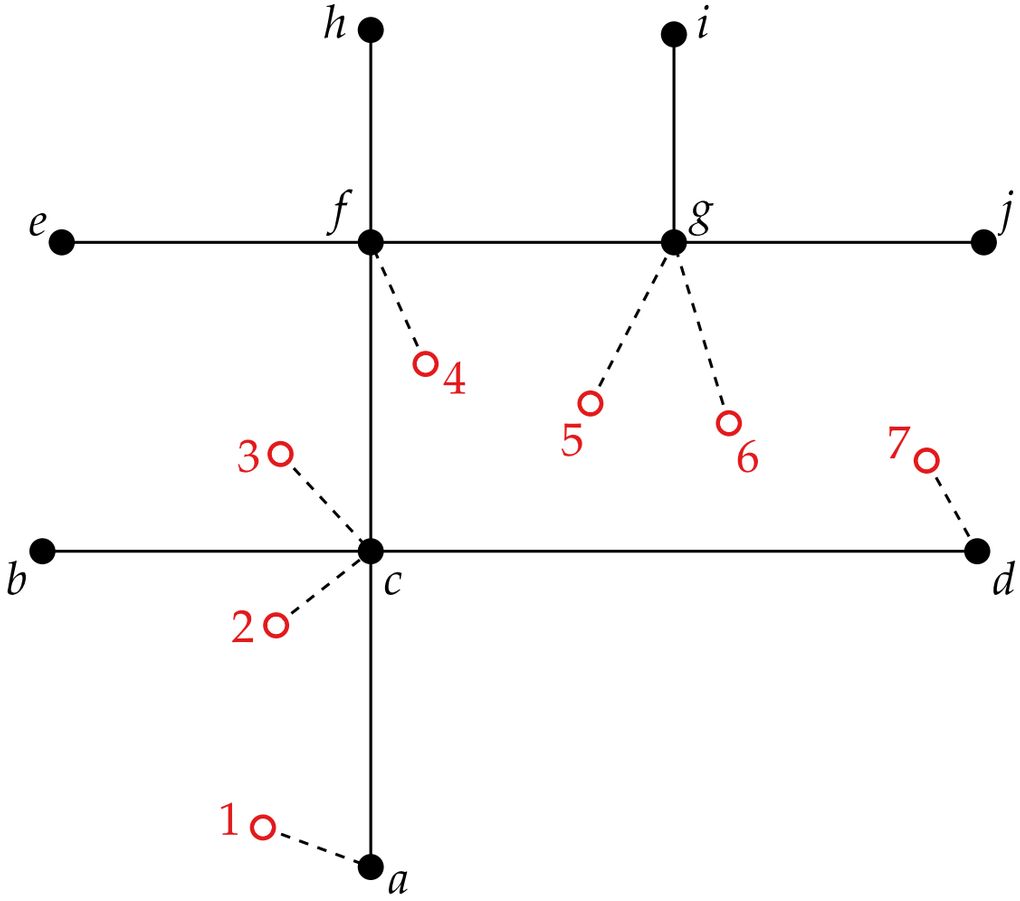


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?

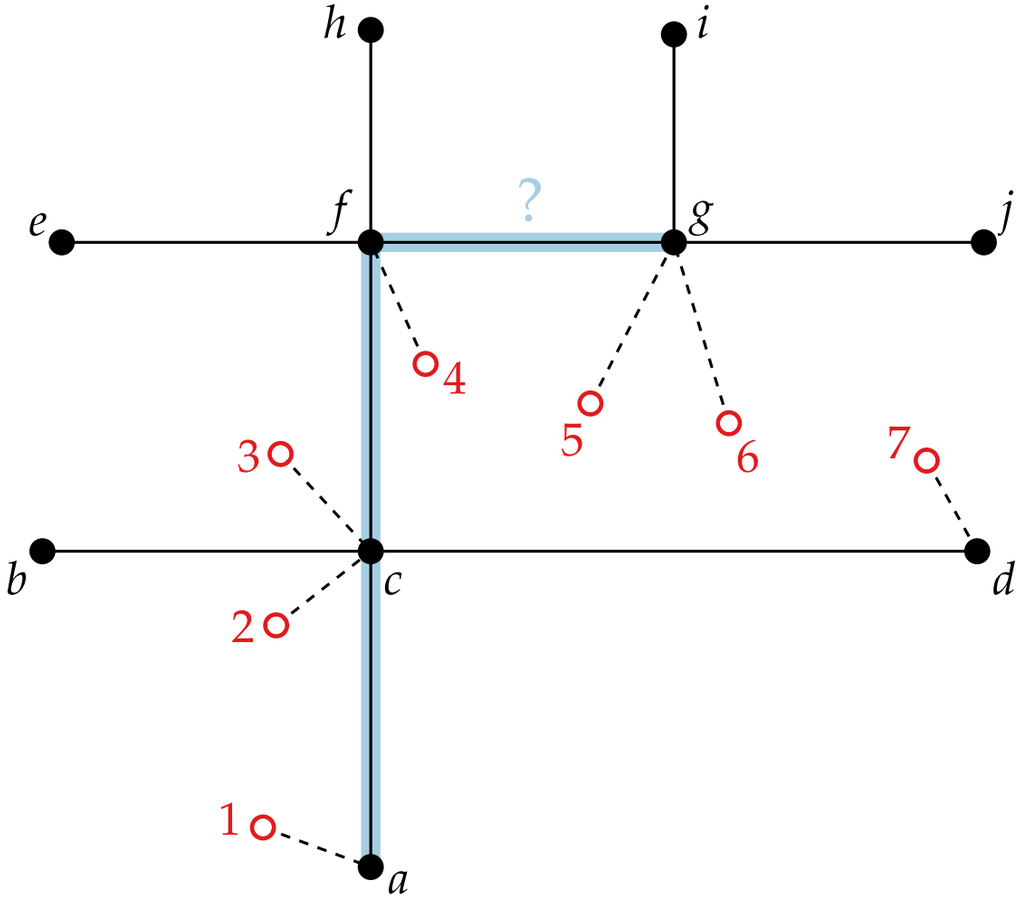


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?

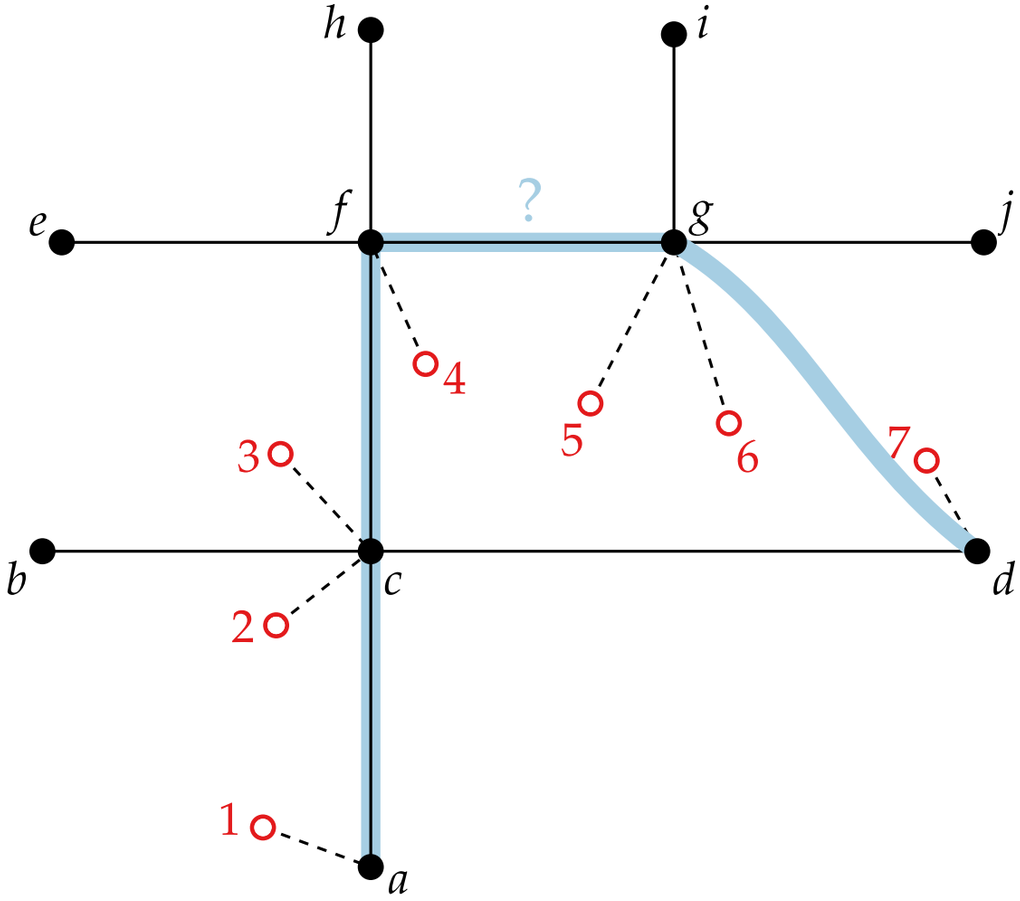


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?

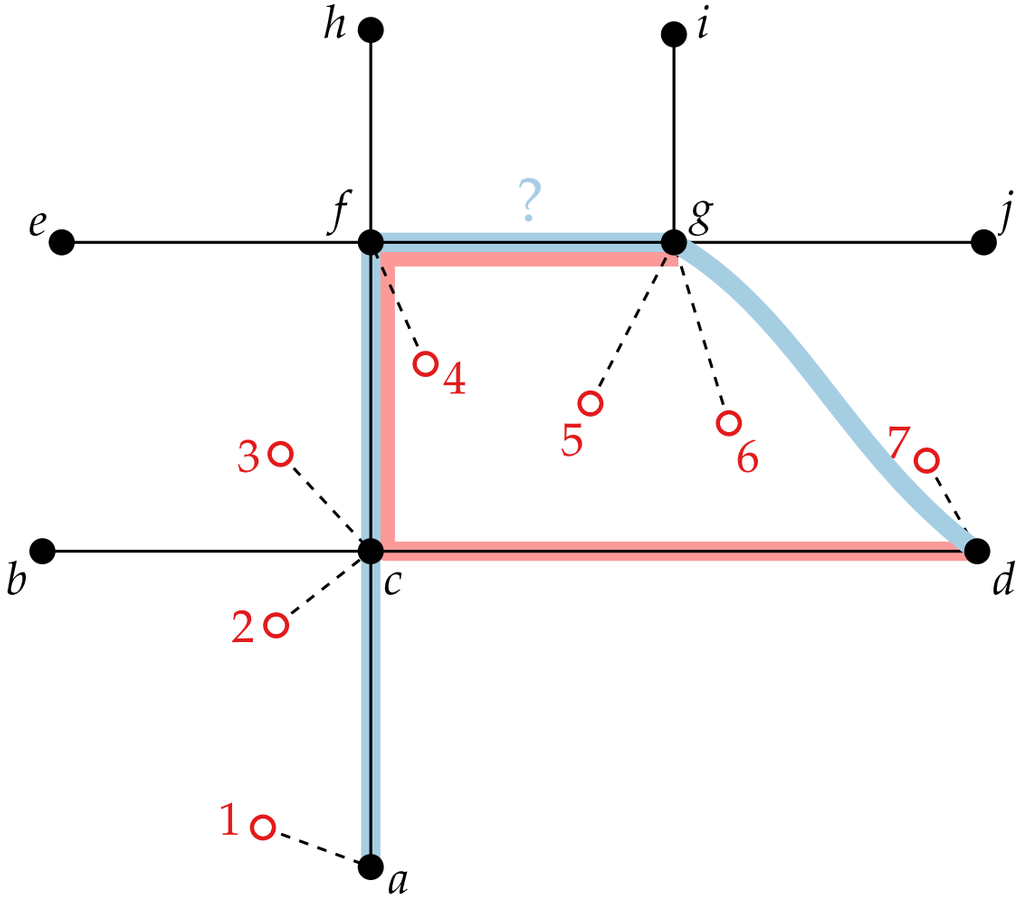


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?

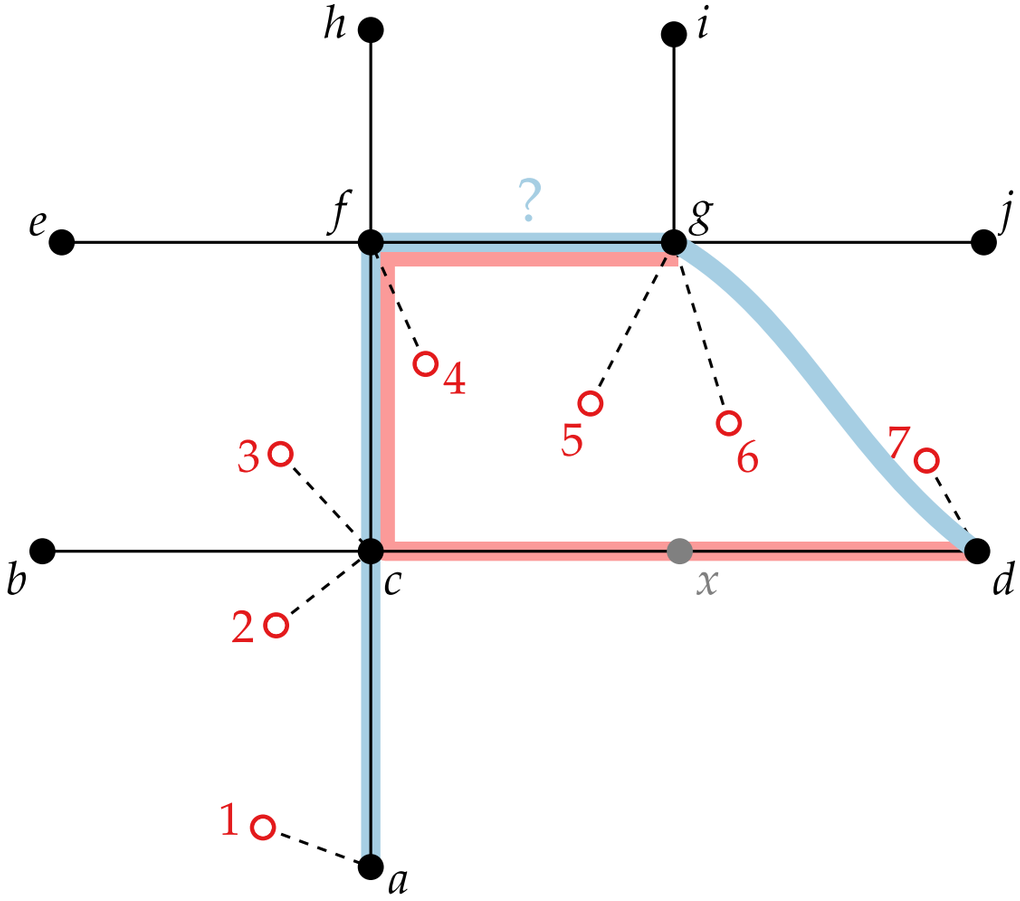


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?

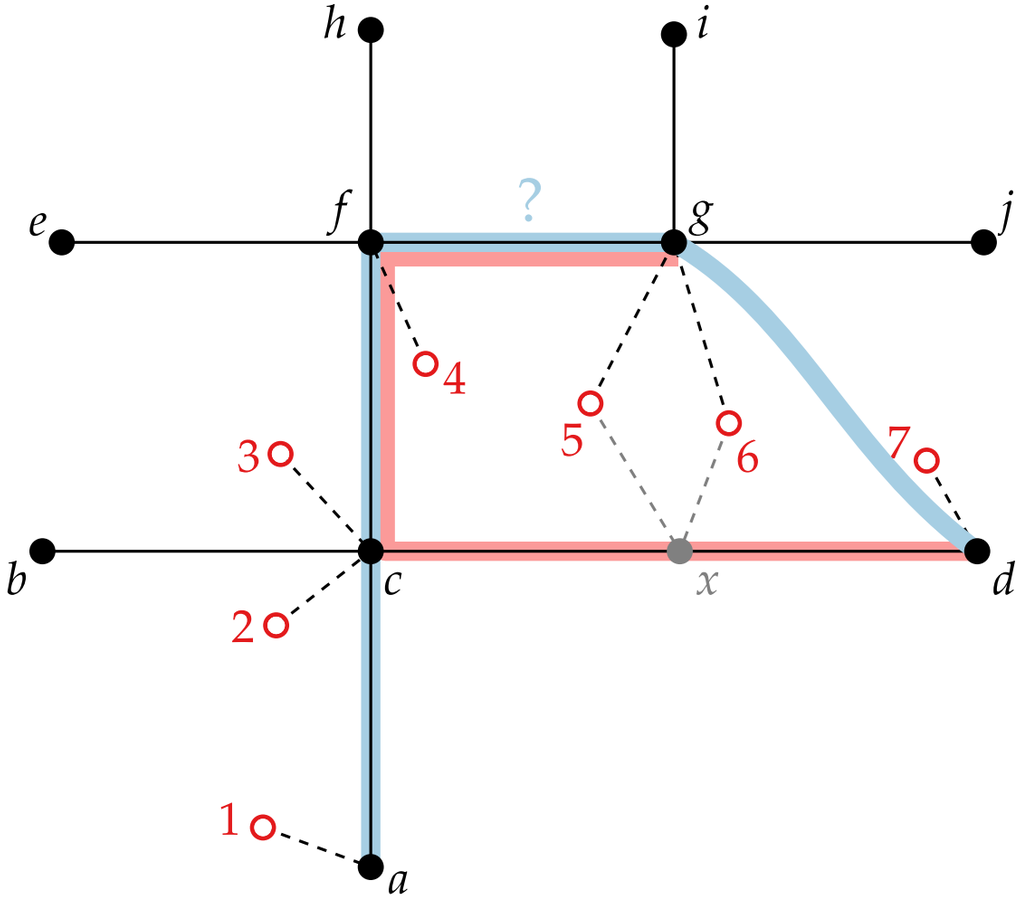


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?

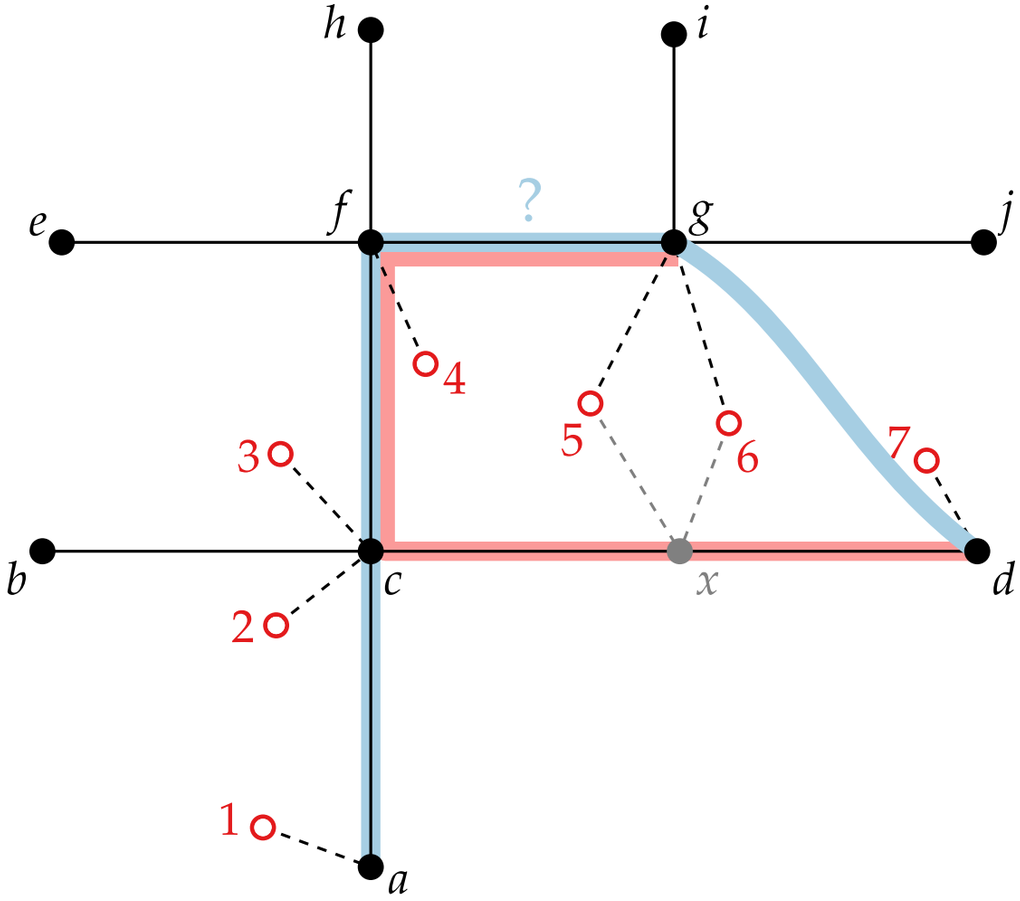


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?

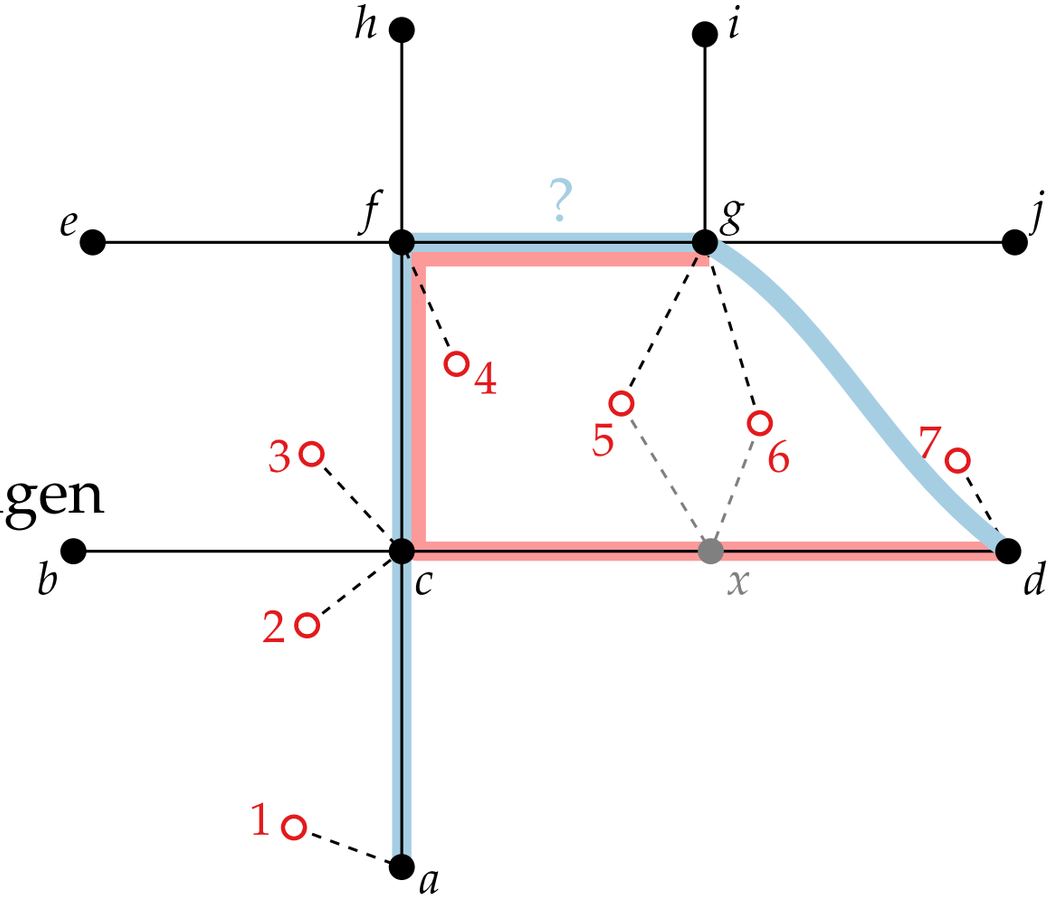


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen

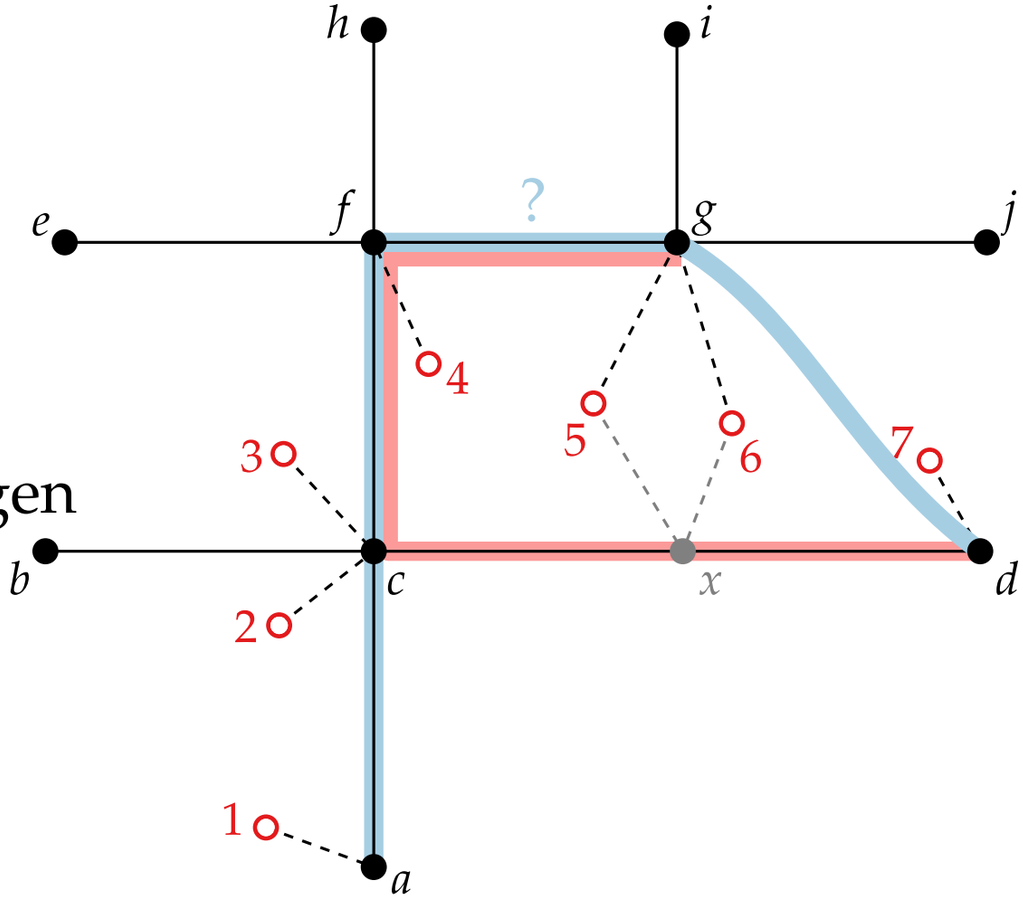


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees

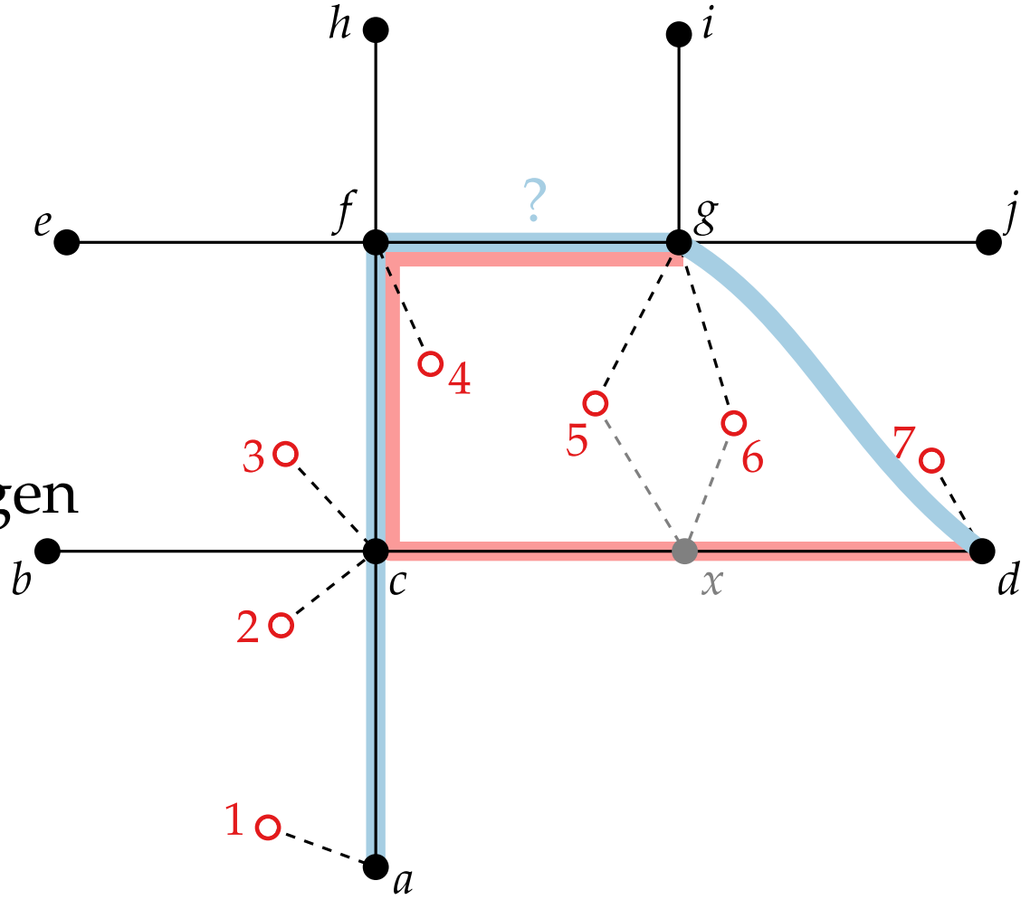


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees
- Punkt-zu-Kante-Zuordnung

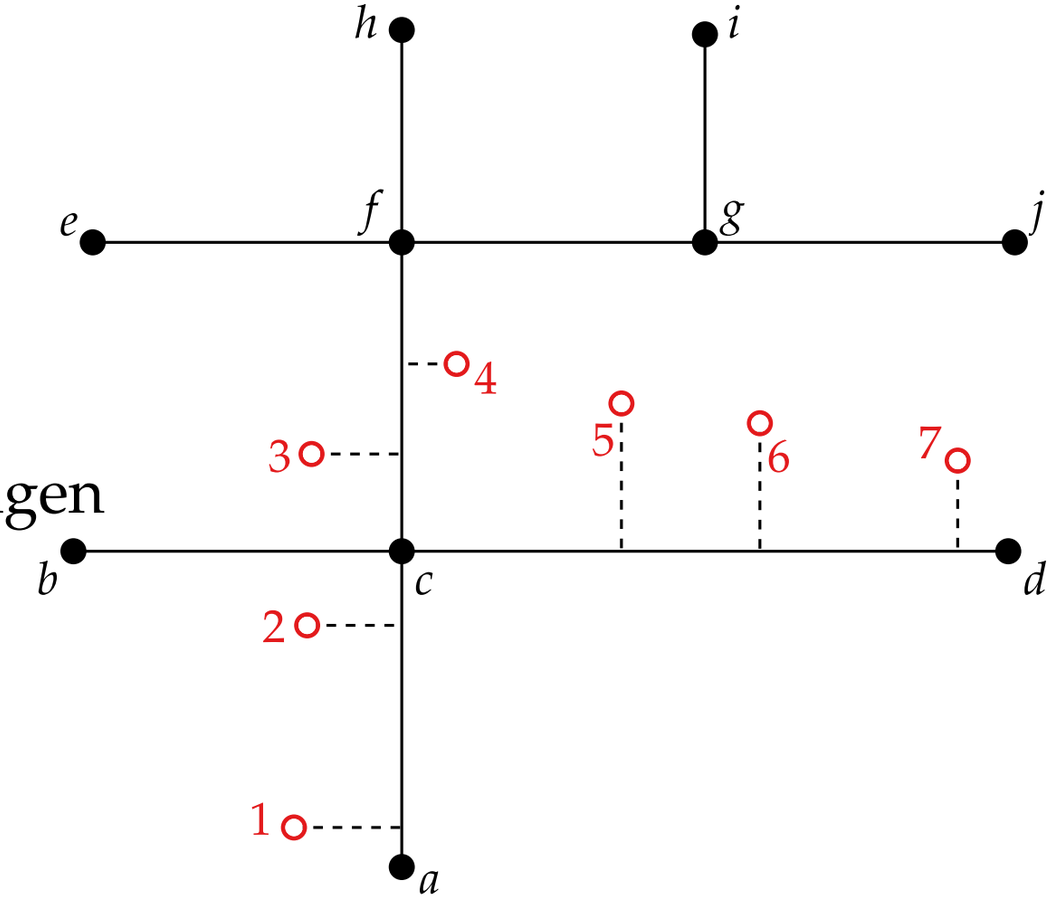


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees
- Punkt-zu-Kante-Zuordnung

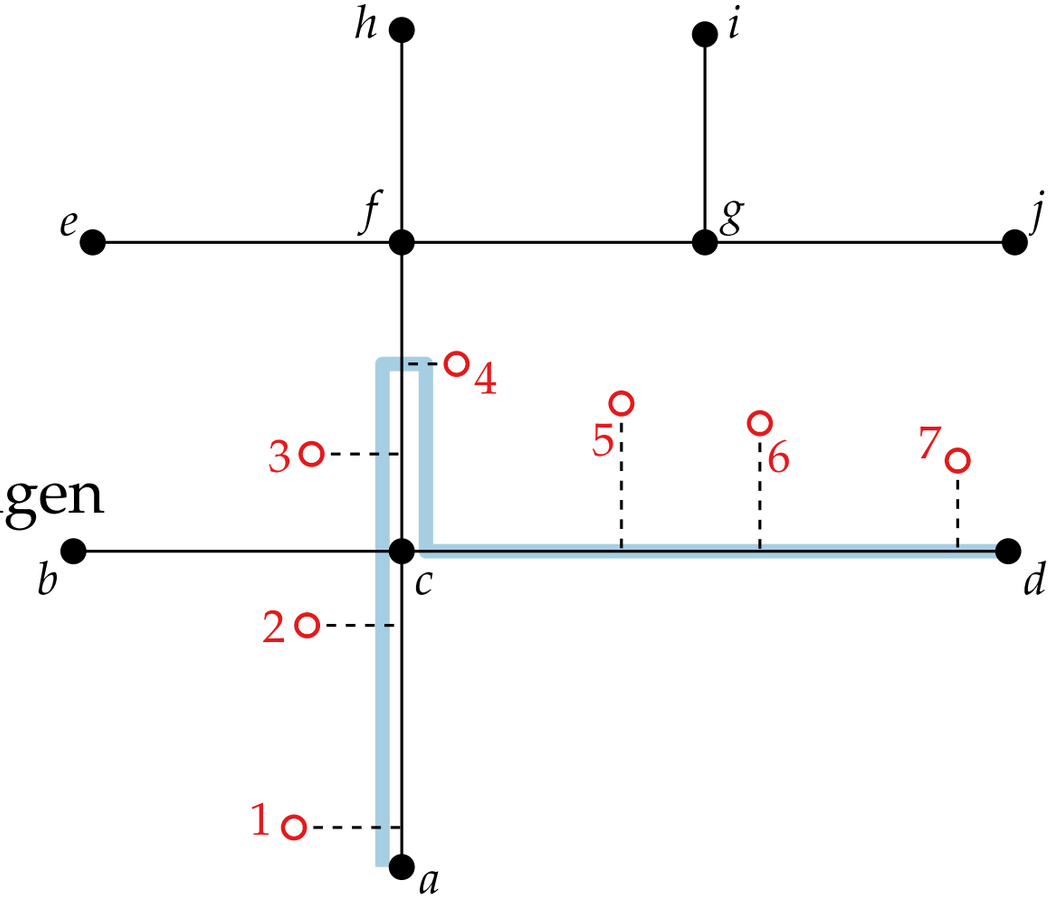


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees
- Punkt-zu-Kante-Zuordnung

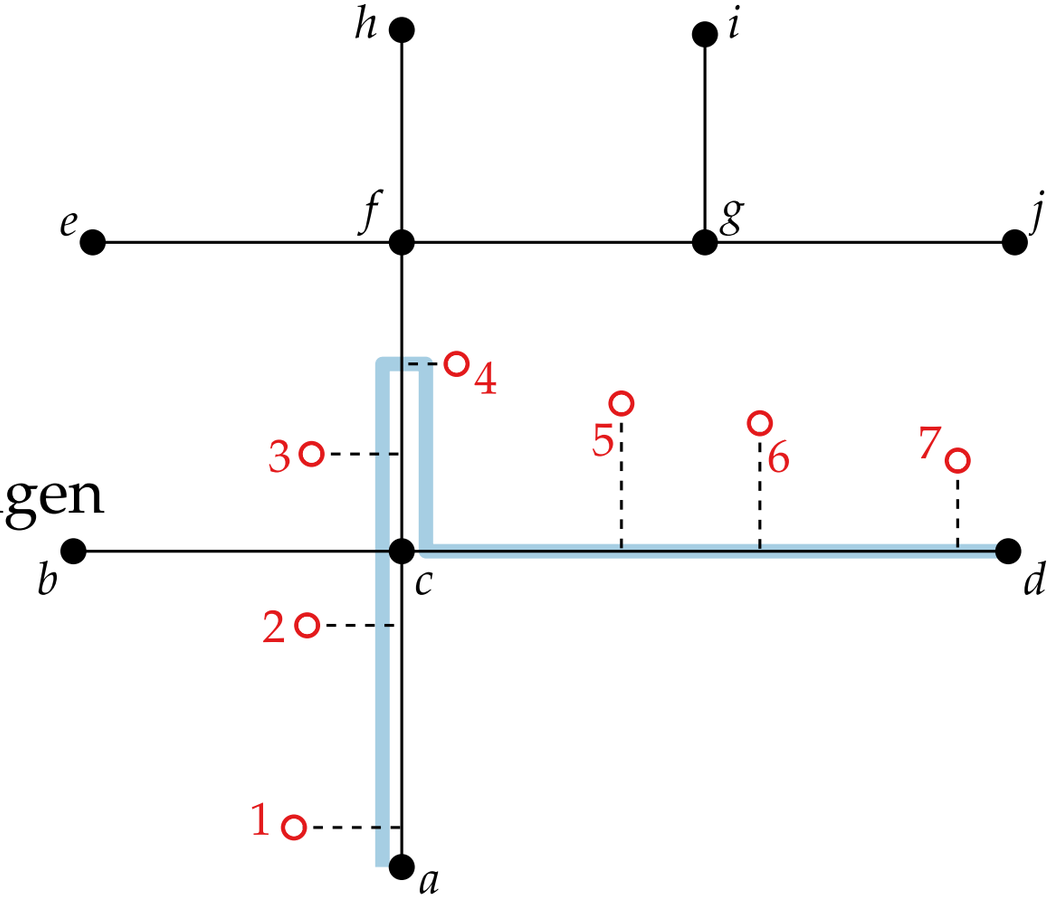


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees
- Punkt-zu-Kante-Zuordnung
  - Probleme?

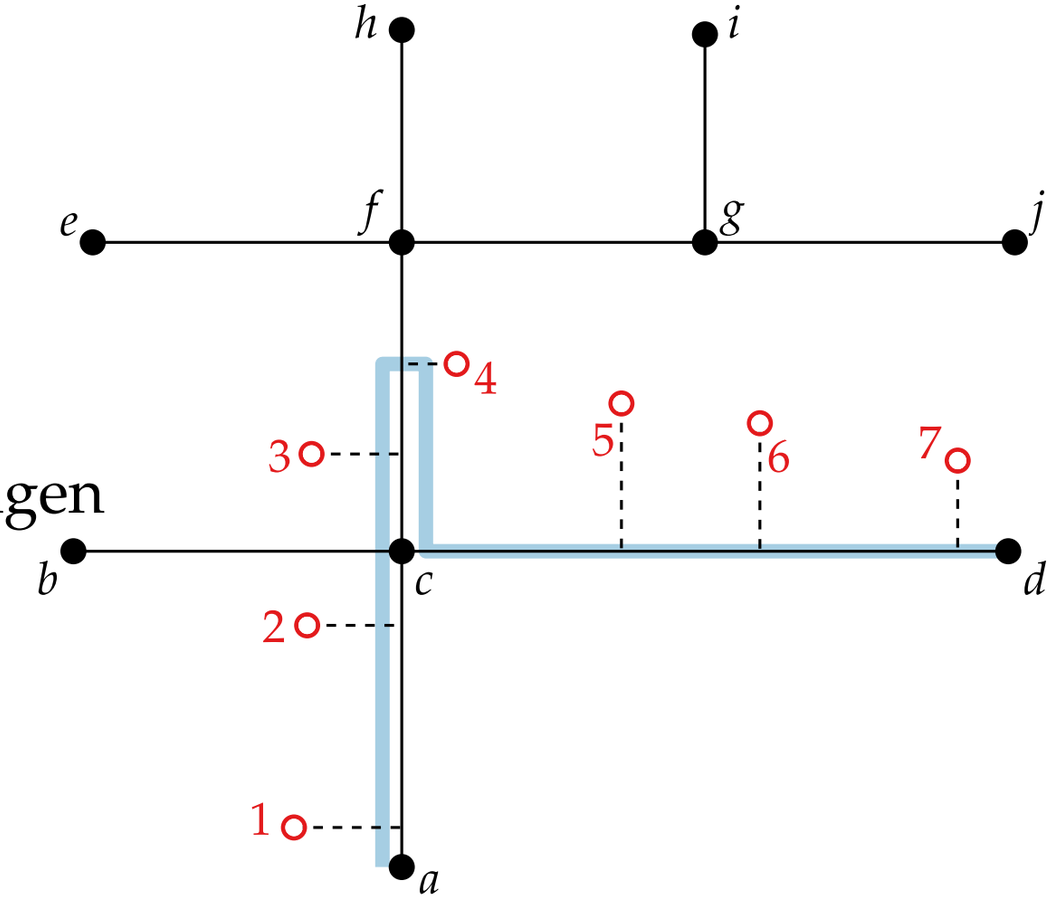


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees
- Punkt-zu-Kante-Zuordnung
  - Probleme?
  - Berechnung?

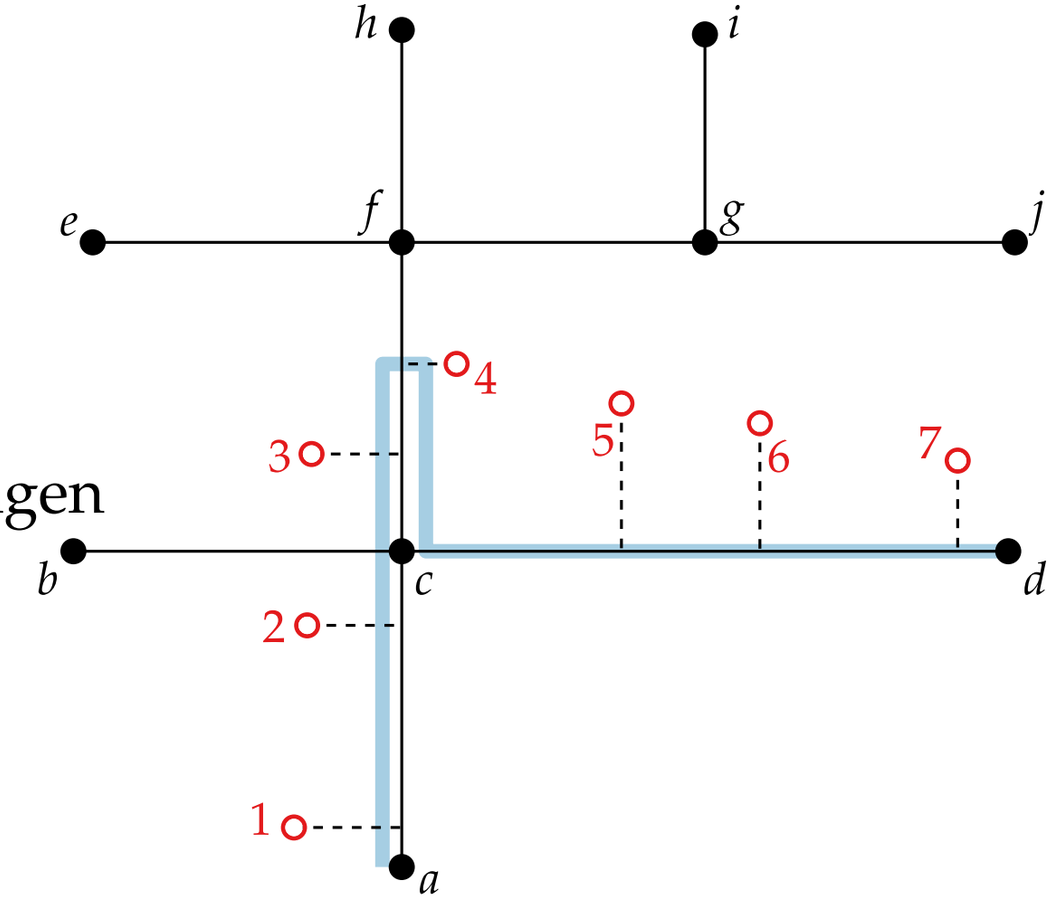


Naive Ansätze

- Welche tatsächliche Route ist wahrscheinlich?
  - $a - c - d$?
  - $a - c - f - g$?
- Benötigen ein Maß, das Ähnlichkeit bestimmt

Ideen für Berechnung?

- Punkt-zu-Punkt-Zuordnung
  - Probleme?
  - Berechnung?
    - ➡ Naiv: $\mathcal{O}(pn)$ für p Punkte und n Kreuzungen
    - ➡ $\mathcal{O}(p \log n)$ mit z.B. Range Trees
- Punkt-zu-Kante-Zuordnung
  - Probleme?
  - Berechnung?



Problemdefinition

MapMatching

Problemdefinition

MapMatching

Gegeben:

Problemdefinition

MapMatching

Gegeben:

- Das Straßennetz

Problemdefinition

MapMatching

Gegeben:

- Das Straßennetz
- Die GPS-Trajektorie

Problemdefinition

MapMatching

Gegeben:

- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie

Problemdefinition

MapMatching

Gegeben:

- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie als Folge P von p Punkten

Problemdefinition

MapMatching

Gegeben:
- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie als Folge P von p Punkten

Gesucht: Weg in G, der **minimale Distanz** zu P hat.

Problemdefinition

MapMatching

Gegeben:
- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie als Folge P von p Punkten

Gesucht: Weg in G, der **minimale Distanz** zu P hat.

Problemdefinition

MapMatching

Gegeben:
- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie als Folge P von p Punkten

Gesucht: Weg in G, der **minimale Distanz** zu P hat.

Betrachten 2 Maße:

Problemdefinition

MapMatching

Gegeben:
- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie als Folge P von p Punkten

Gesucht: Weg in G, der **minimale Distanz** zu P hat.

Betrachten 2 Maße:
- Hausdorff Distanz

Problemdefinition

MapMatching

Gegeben:
- Das Straßennetz als planar eingebetteter Graph $G = (V, E)$ mit n Knoten und m Kanten
- Die GPS-Trajektorie als Folge P von p Punkten

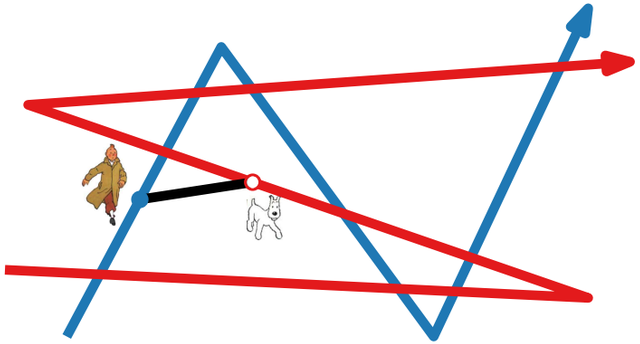
Gesucht: Weg in G, der **minimale Distanz** zu P hat.

Betrachten 2 Maße:
- Hausdorff Distanz
- Fréchet Distanz

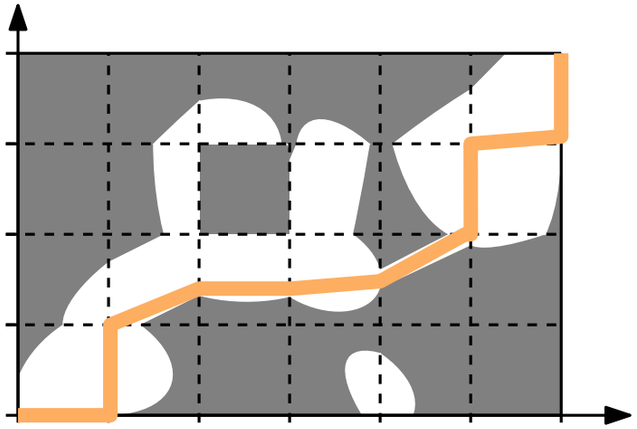
Universität Trier

Algorithmen für geographische Informationssysteme

3. Vorlesung
Map Matching



Teil II:
Hausdorff Distanz

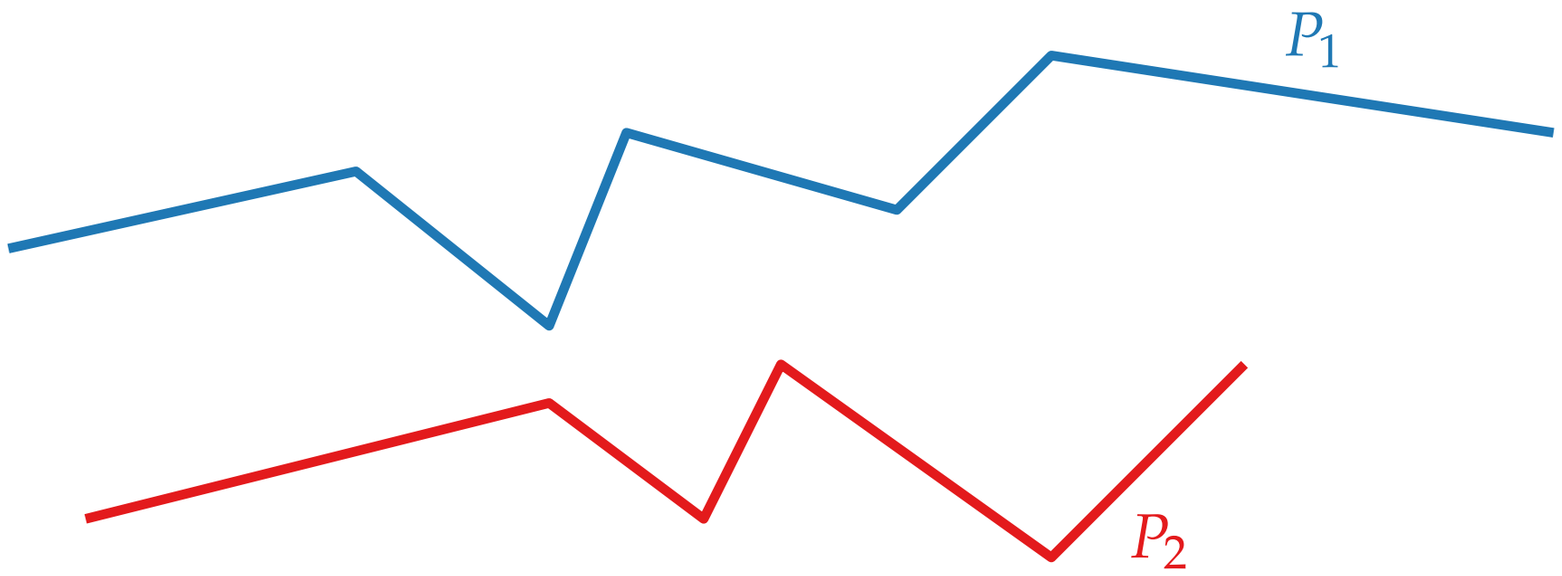


Philipp Kindermann

Wintersemester 2024/25

Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?

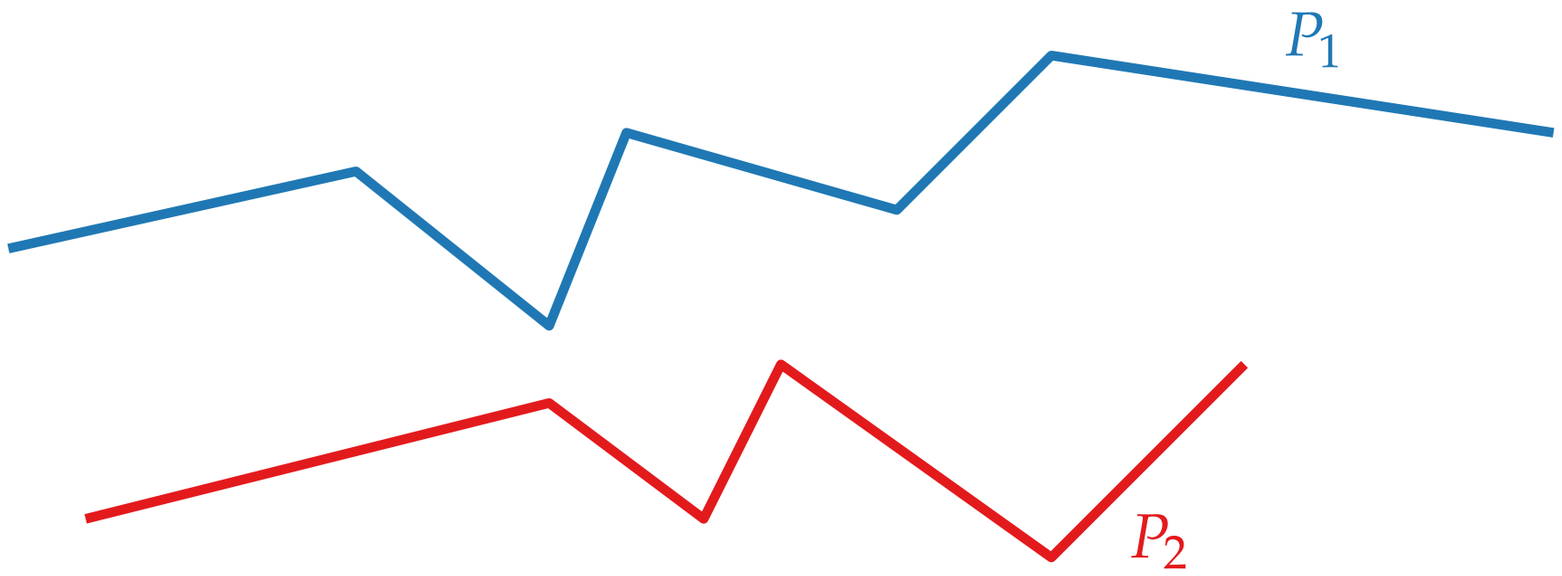


Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?

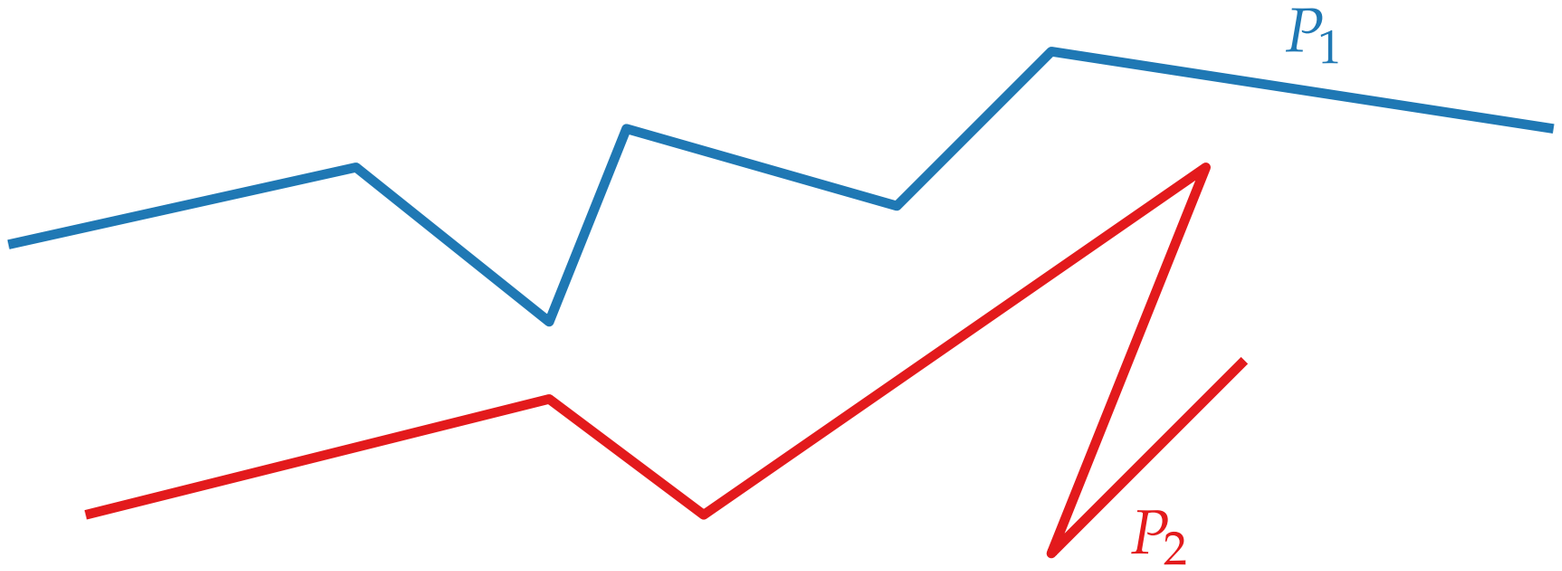
Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?



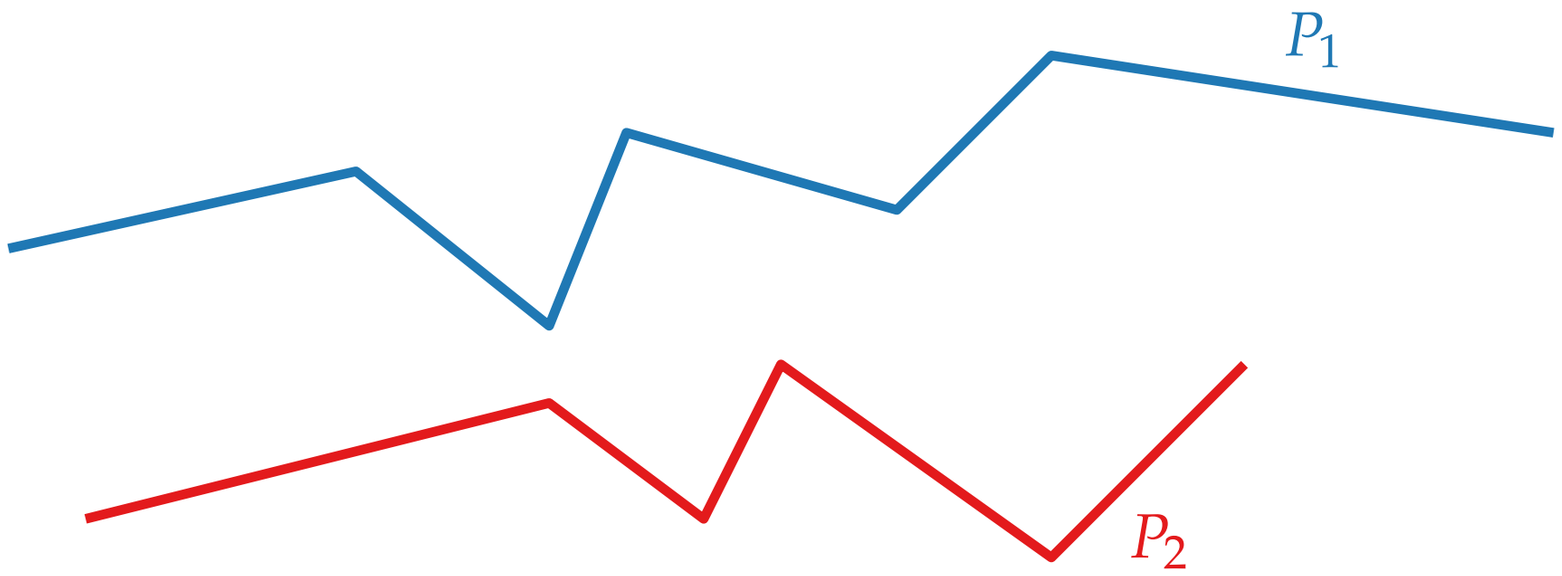
Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?



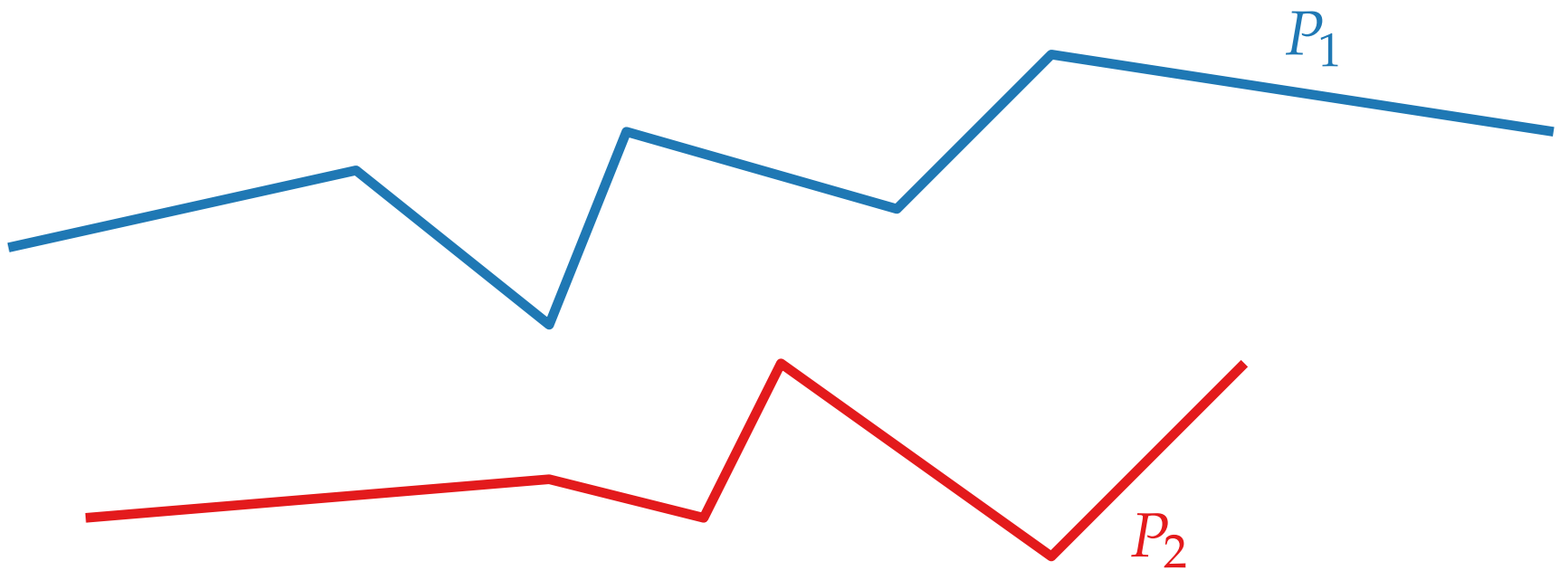
Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?



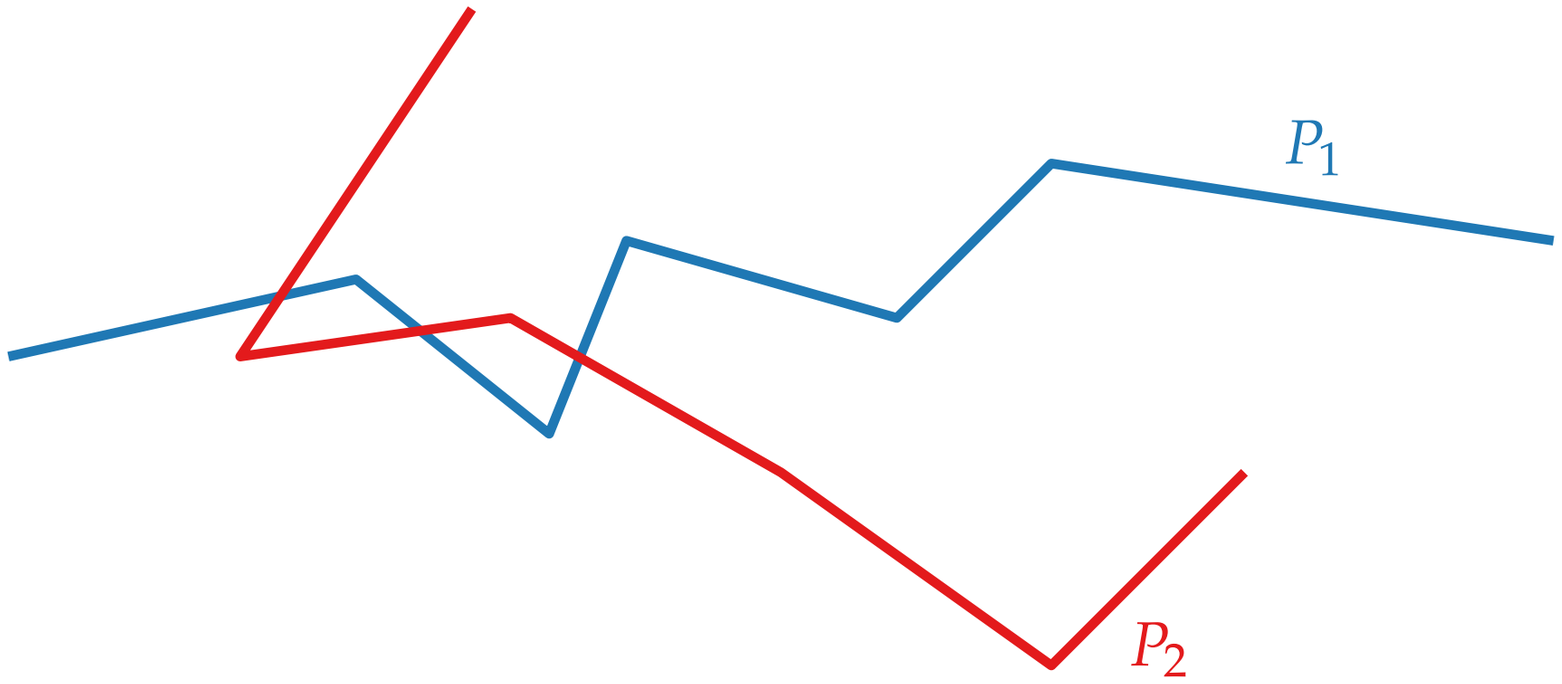
Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?



Abstände zwischen Linienzügen

Gegeben zwei Linienzüge P_1 und P_2, was ist der **Abstand** zwischen P_1 und P_2?



Euklidischer Abstand

$d(p_1, p_2)$

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

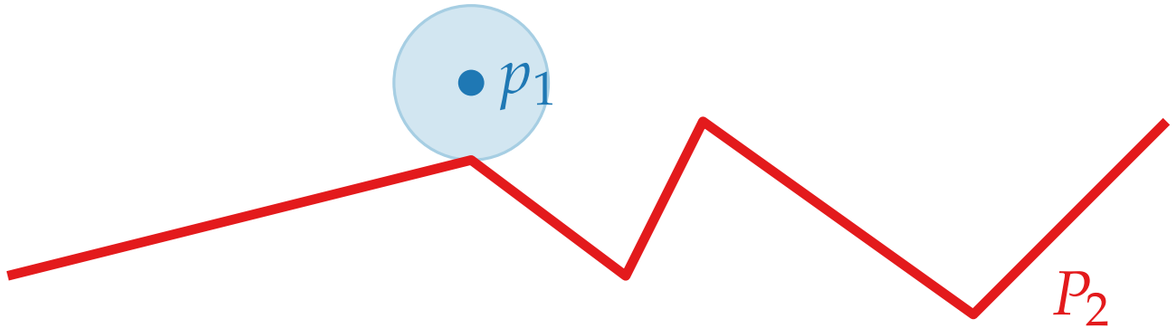
- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$

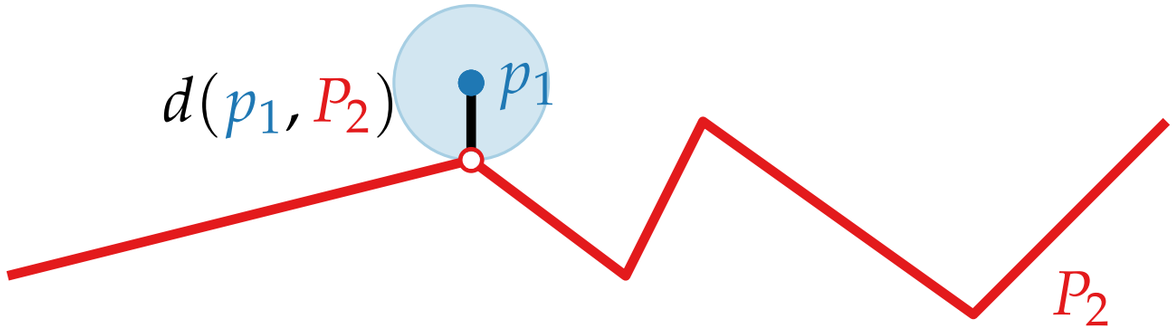


Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$

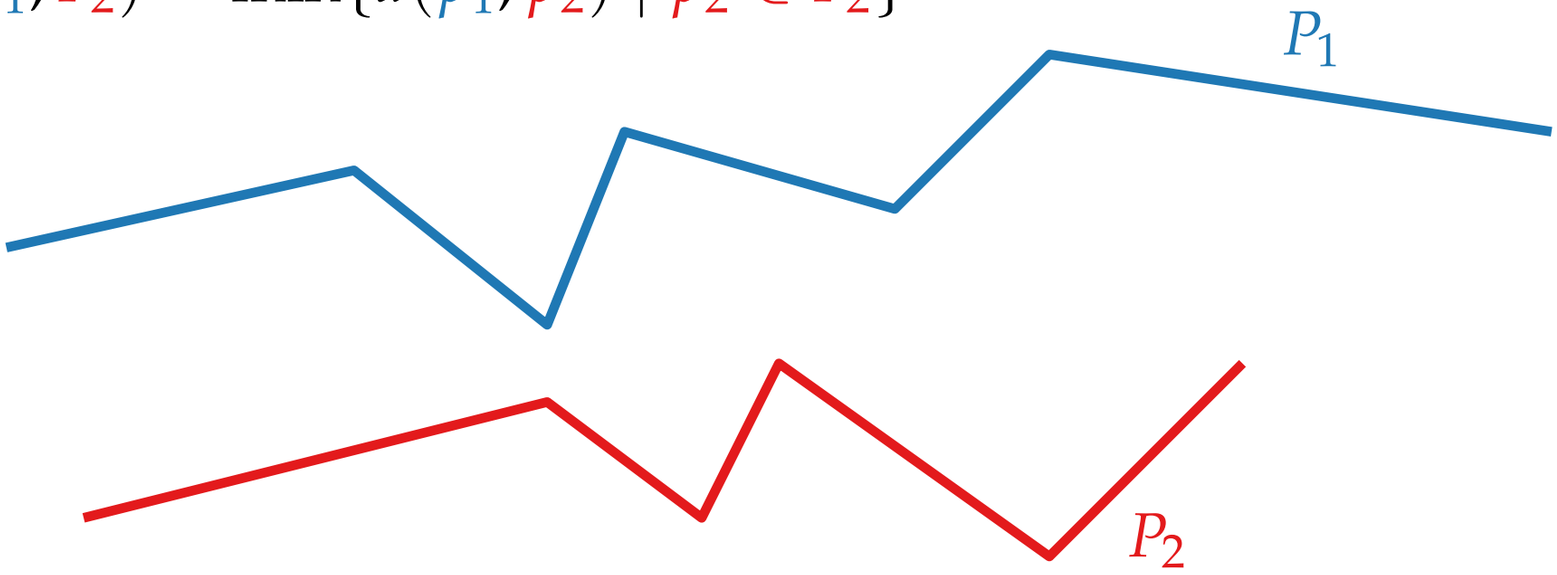


Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$

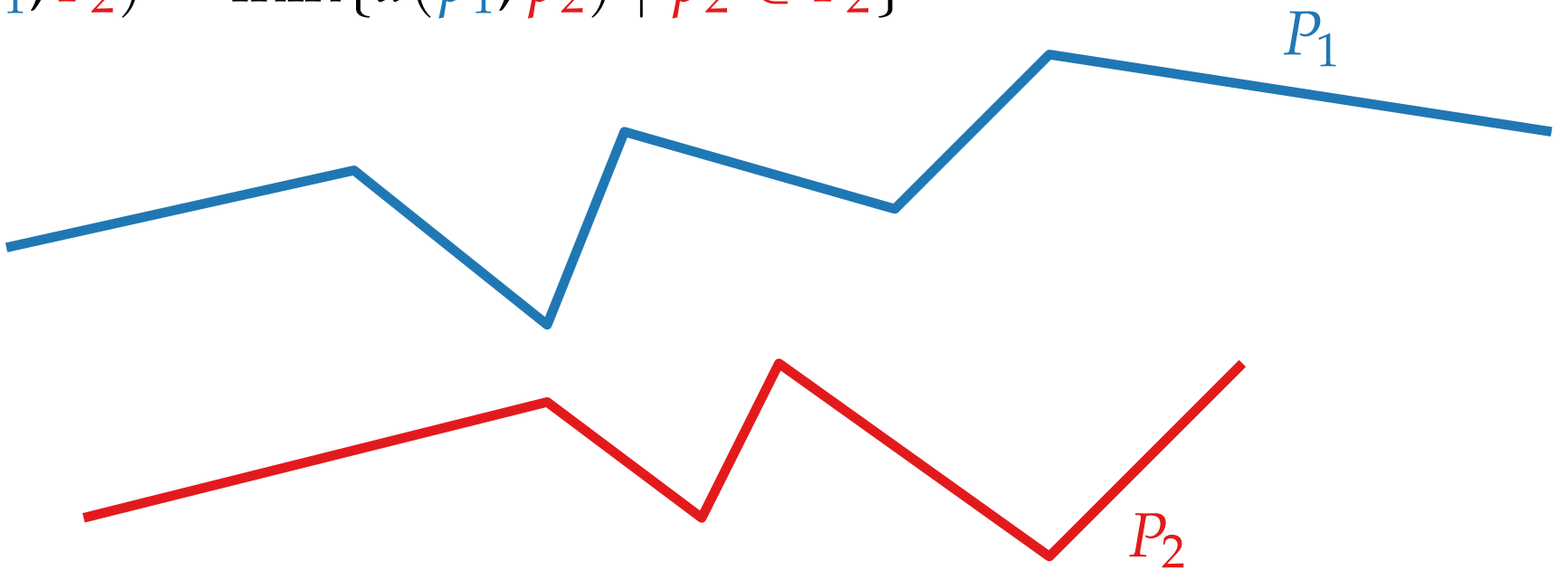


Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



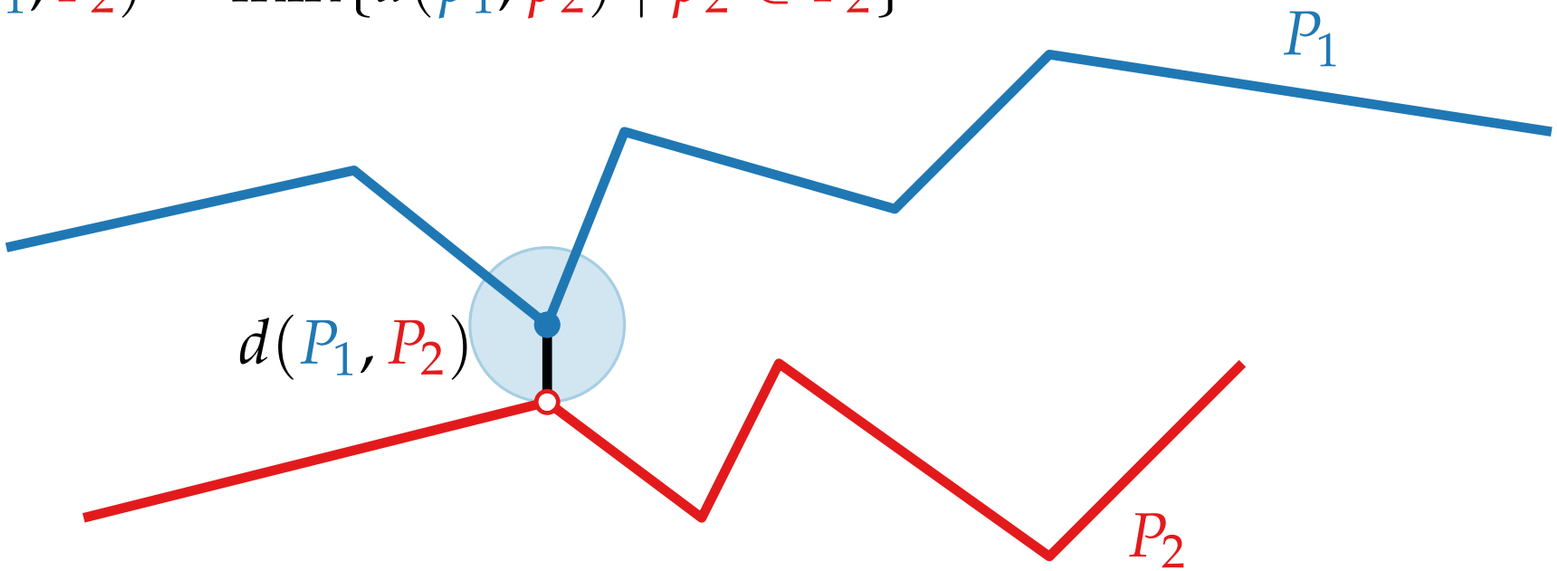
- Min. Euklidischer Abstand zwischen zwei Linienzügen P_1, P_2:
  $d(P_1, P_2) = \min\{d(p_1, p_2) \mid p_1 \in P_1, p_2 \in P_2\}$

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



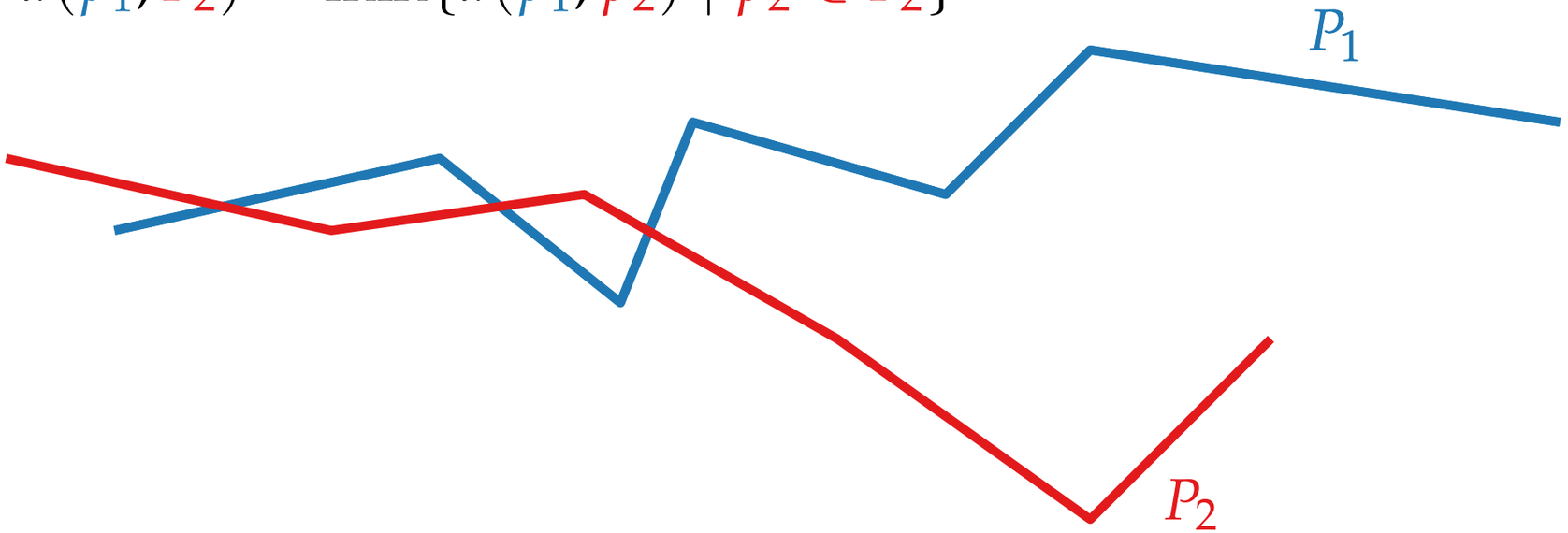
- Min. Euklidischer Abstand zwischen zwei Linienzügen P_1, P_2:
  $d(P_1, P_2) = \min\{d(p_1, p_2) \mid p_1 \in P_1, p_2 \in P_2\}$

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



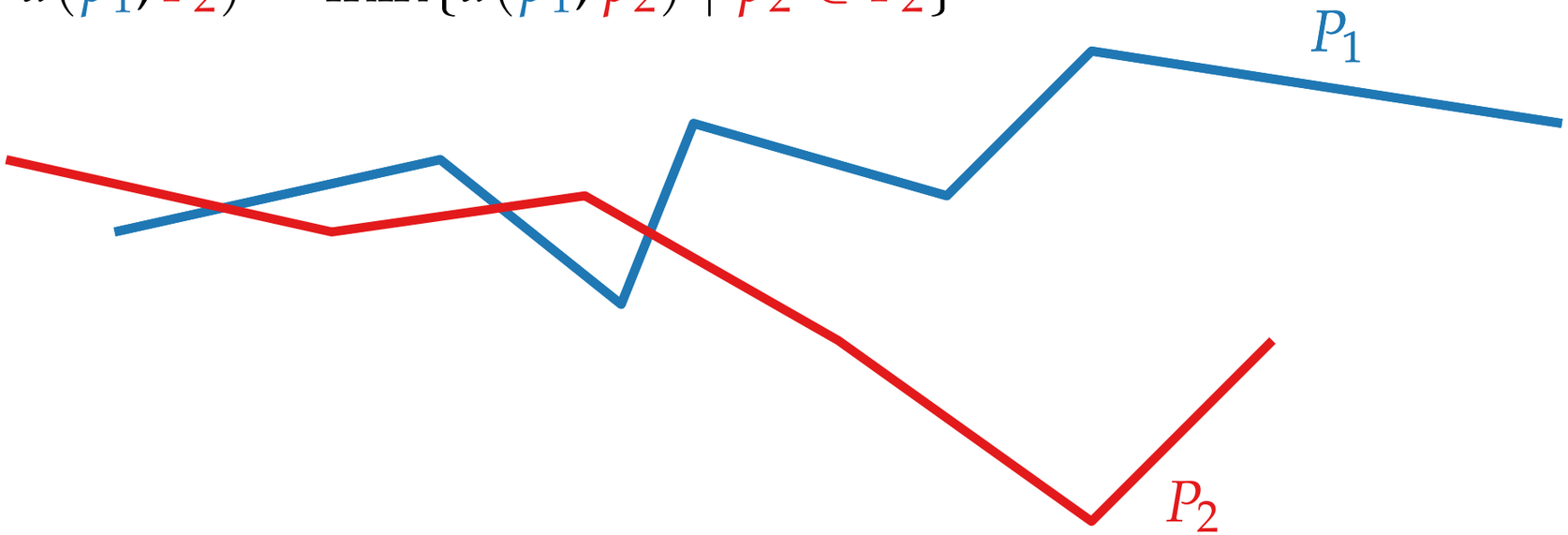
- Min. Euklidischer Abstand zwischen zwei Linienzügen P_1, P_2:
  $d(P_1, P_2) = \min\{d(p_1, p_2) \mid p_1 \in P_1, p_2 \in P_2\}$

Euklidischer Abstand

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



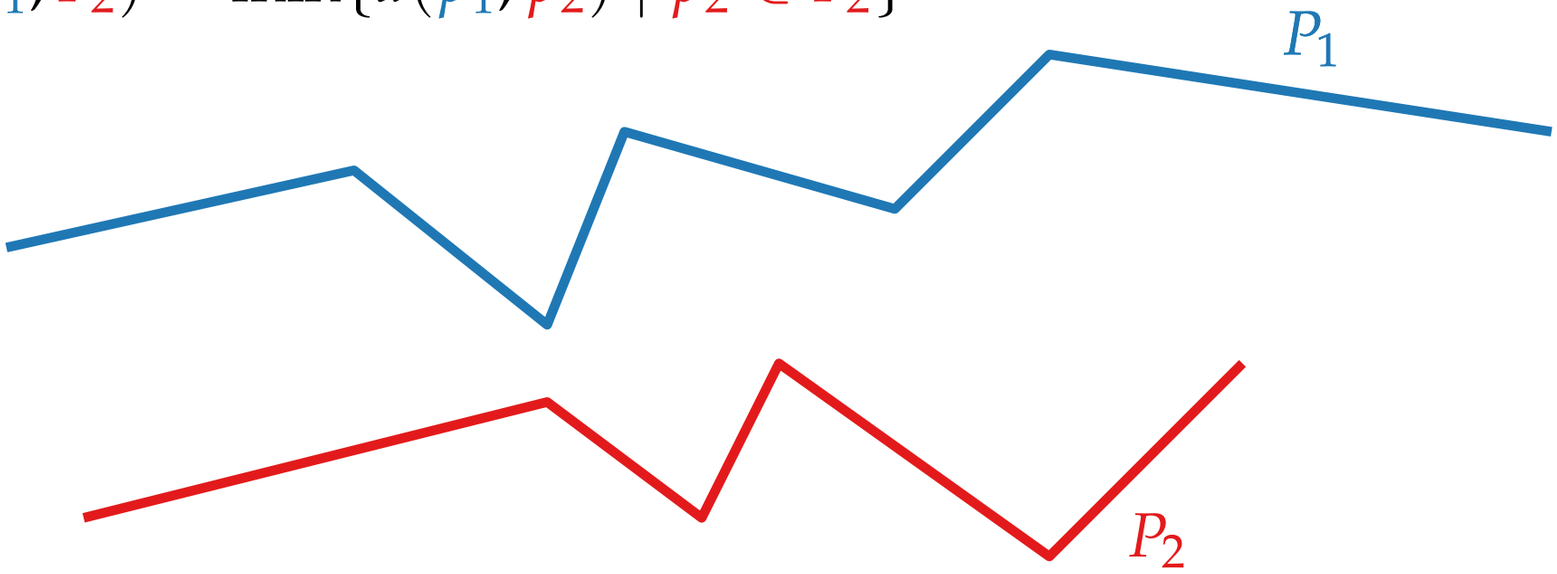
- Min. Euklidischer Abstand zwischen zwei Linienzügen P_1, P_2:
  $d(P_1, P_2) = \min\{d(p_1, p_2) \mid p_1 \in P_1, p_2 \in P_2\}$
- $d(P_1, P_2)$ als Distanzmaß ungeeignet, weil $d(P_1, P_2) = 0$ wenn sich P_1 und P_2 schneiden.

(Gerichtete) Hausdorff Distanz

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$

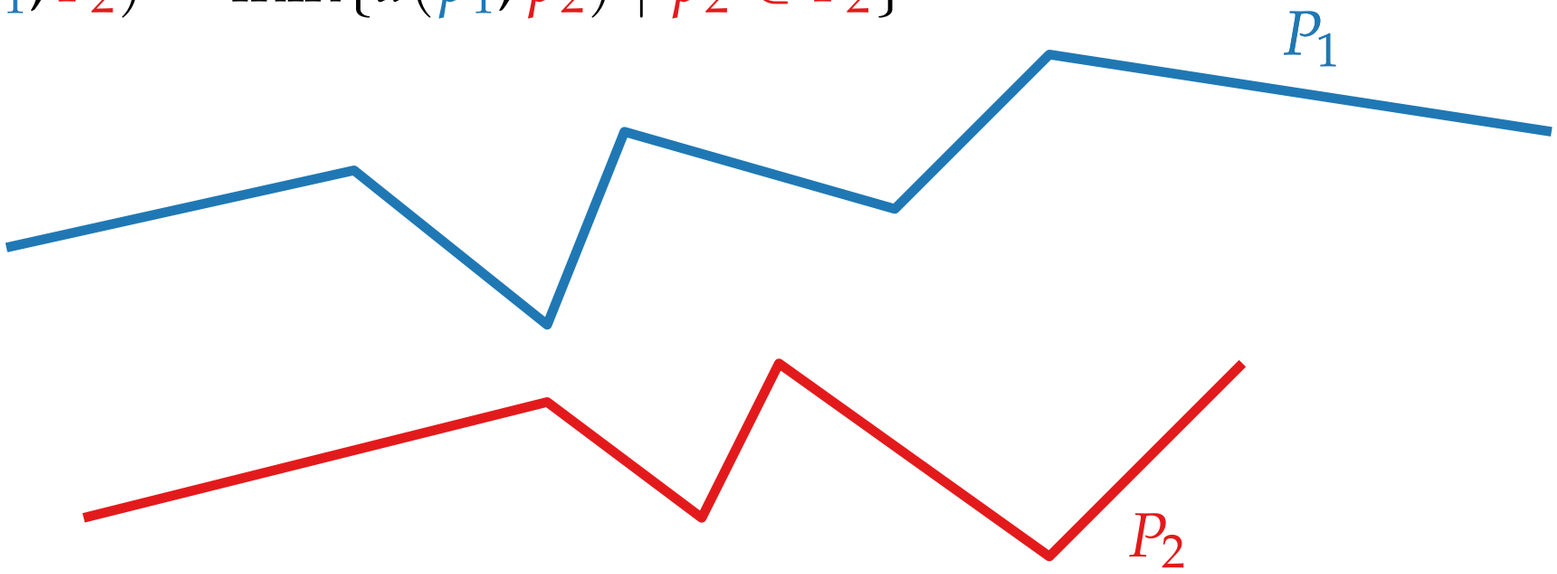


(Gerichtete) Hausdorff Distanz

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$ 

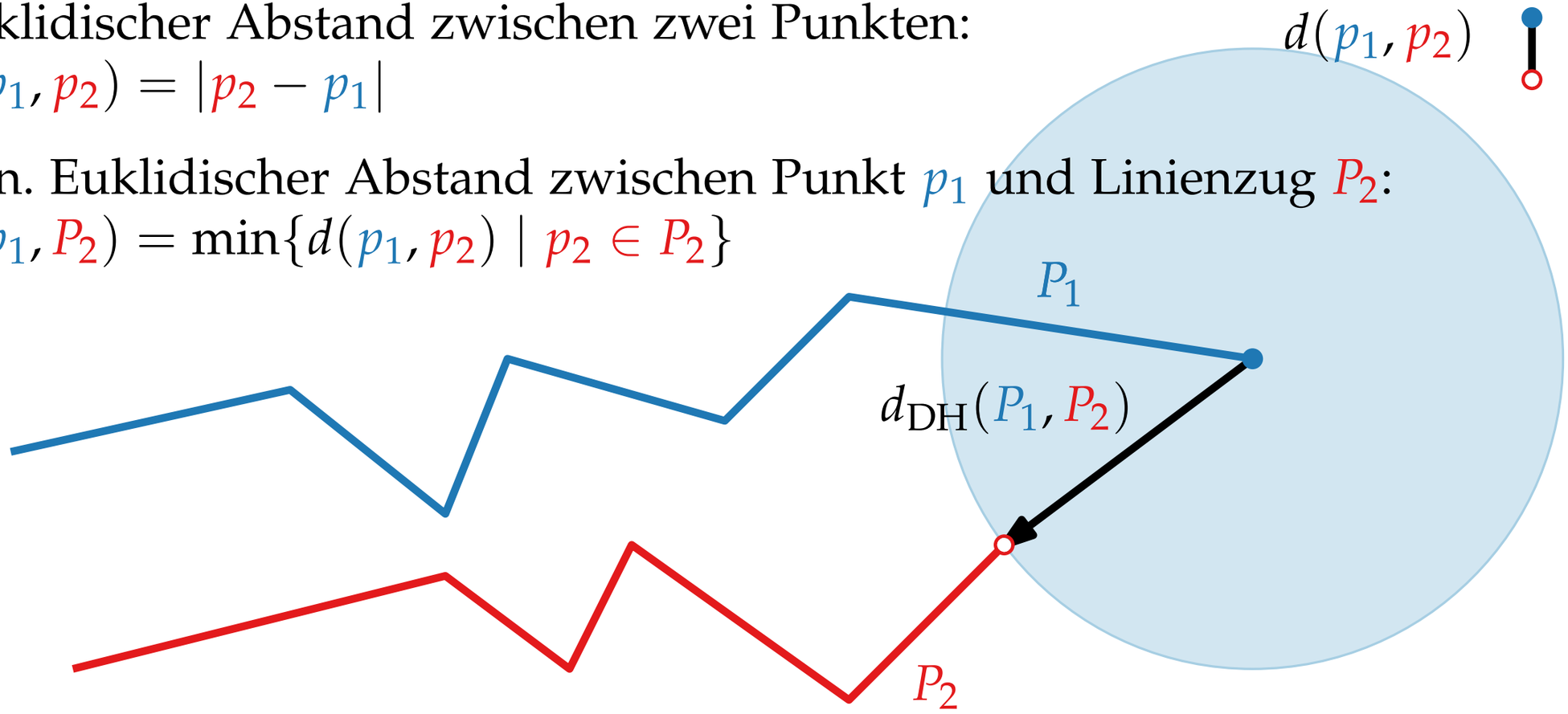
- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



- Gerichtete Hausdorff Distanz:
  $d_{\mathrm{DH}}(P_1, P_2) = \max\{d(p_1, P_2) \mid p_1 \in P_1\}$

(Gerichtete) Hausdorff Distanz

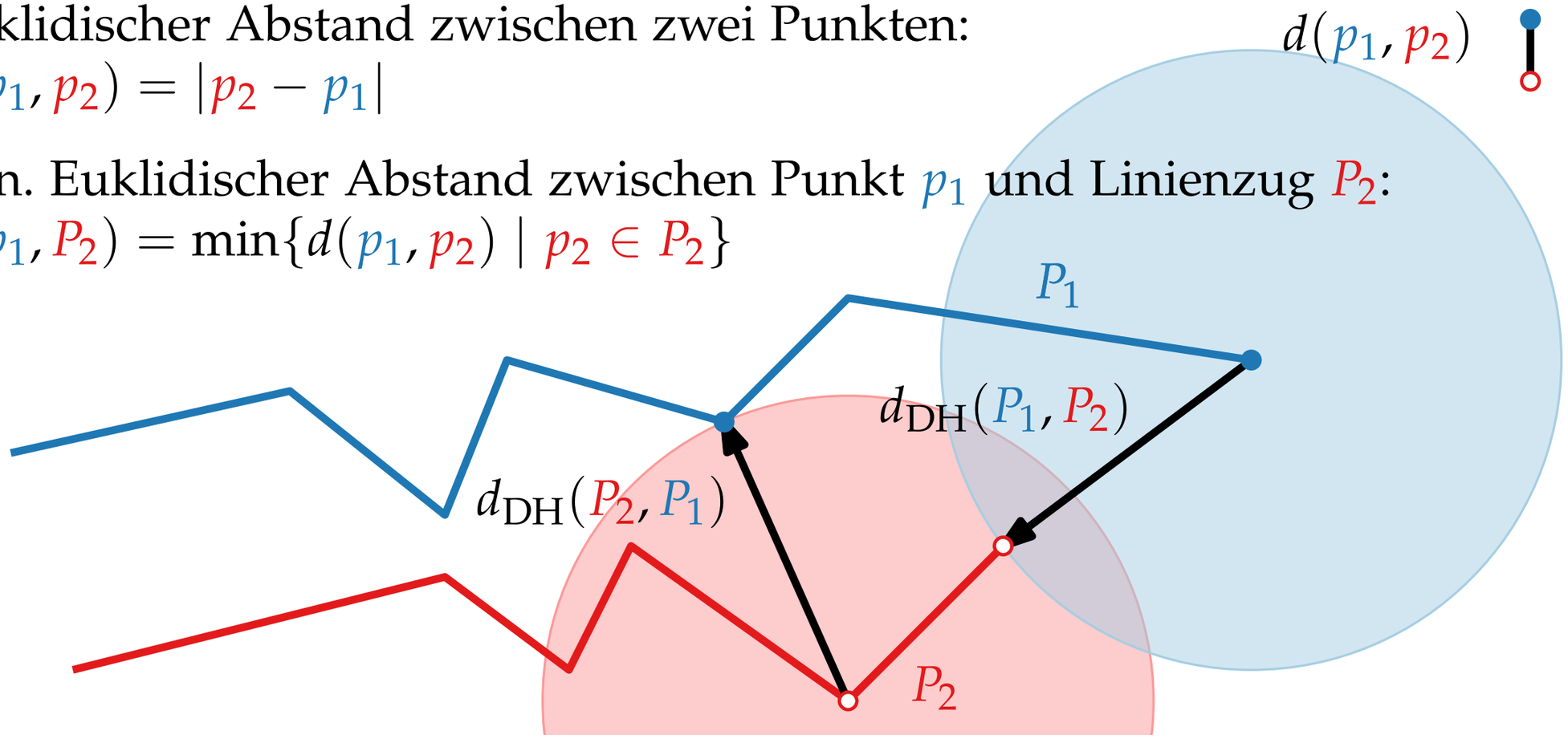
- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$
- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



- Gerichtete Hausdorff Distanz:
  $d_{\mathrm{DH}}(P_1, P_2) = \max\{d(p_1, P_2) \mid p_1 \in P_1\}$

(Gerichtete) Hausdorff Distanz

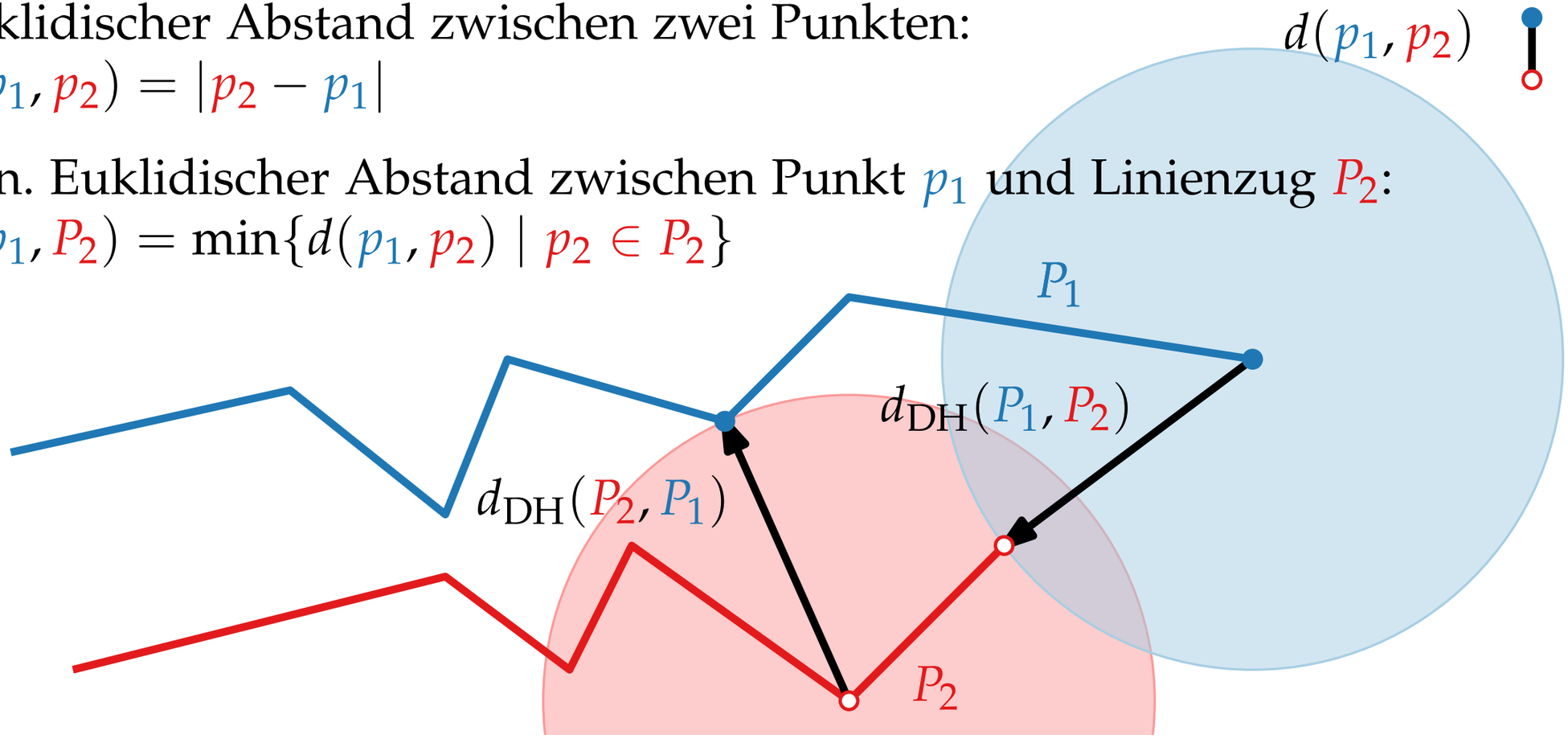
- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$
- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



- Gerichtete Hausdorff Distanz:
  $d_{\mathrm{DH}}(P_1, P_2) = \max\{d(p_1, P_2) \mid p_1 \in P_1\}$

(Gerichtete) Hausdorff Distanz

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$
- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



- Gerichtete Hausdorff Distanz:
  $d_{\mathrm{DH}}(P_1, P_2) = \max\{d(p_1, P_2) \mid p_1 \in P_1\}$

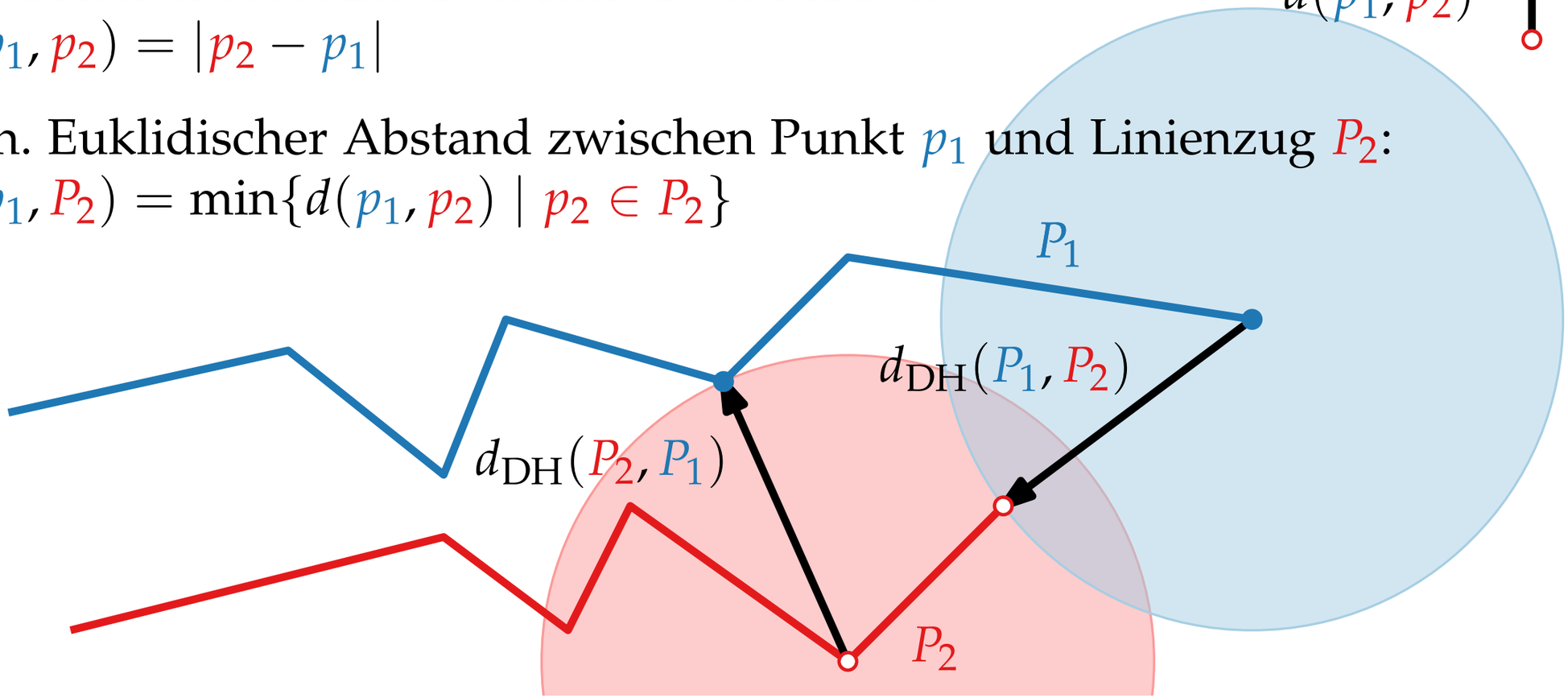
Achtung: Allgemein gilt
$d_{\mathrm{DH}}(P_1, P_2) \neq d_{\mathrm{DH}}(P_2, P_1)$

(Gerichtete) Hausdorff Distanz

- Euklidischer Abstand zwischen zwei Punkten:
  $d(p_1, p_2) = |p_2 - p_1|$

$d(p_1, p_2)$

- Min. Euklidischer Abstand zwischen Punkt p_1 und Linienzug P_2:
  $d(p_1, P_2) = \min\{d(p_1, p_2) \mid p_2 \in P_2\}$



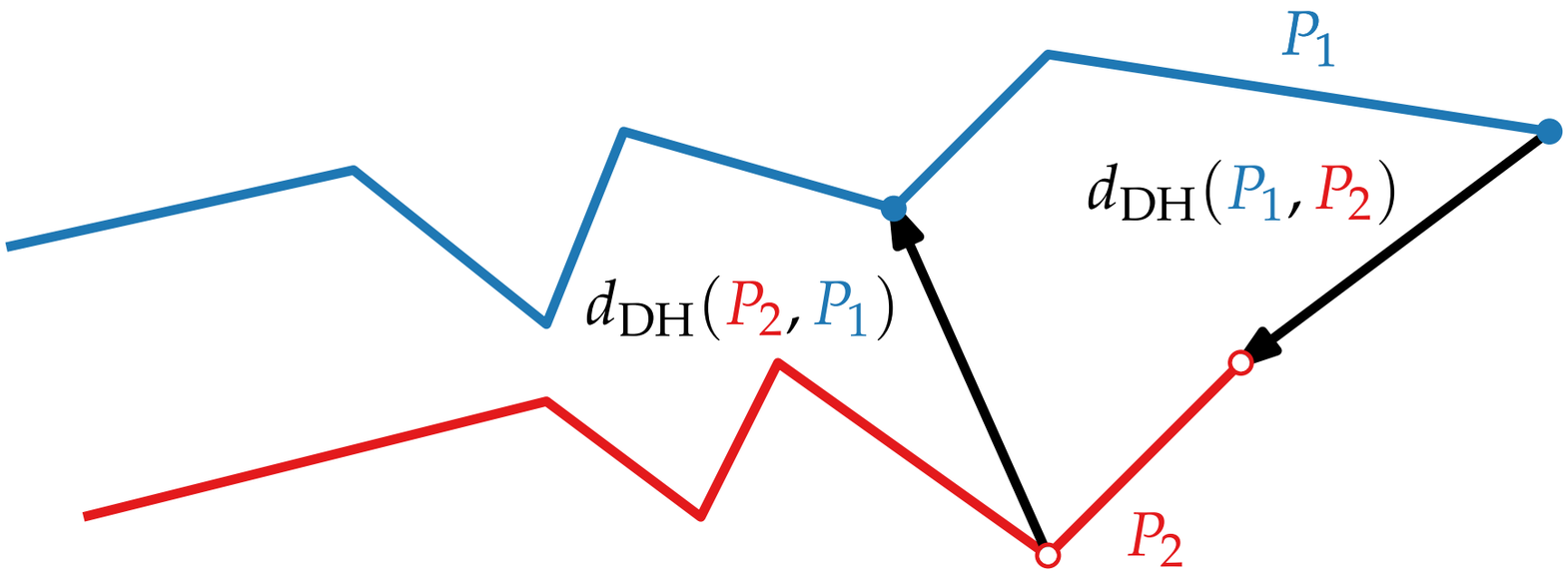
- Gerichtete Hausdorff Distanz:
  $d_{\mathrm{DH}}(P_1, P_2) = \max\{d(p_1, P_2) \mid p_1 \in P_1\}$

Achtung: Allgemein gilt
$d_{\mathrm{DH}}(P_1, P_2) \neq d_{\mathrm{DH}}(P_2, P_1)$

- Hausdorff Distanz: $d_{\mathrm{H}} = \max\{d_{\mathrm{DH}}(P_1, P_2), d_{\mathrm{DH}}(P_2, P_1)\}$

Hausdorff Distanz – Berechnung

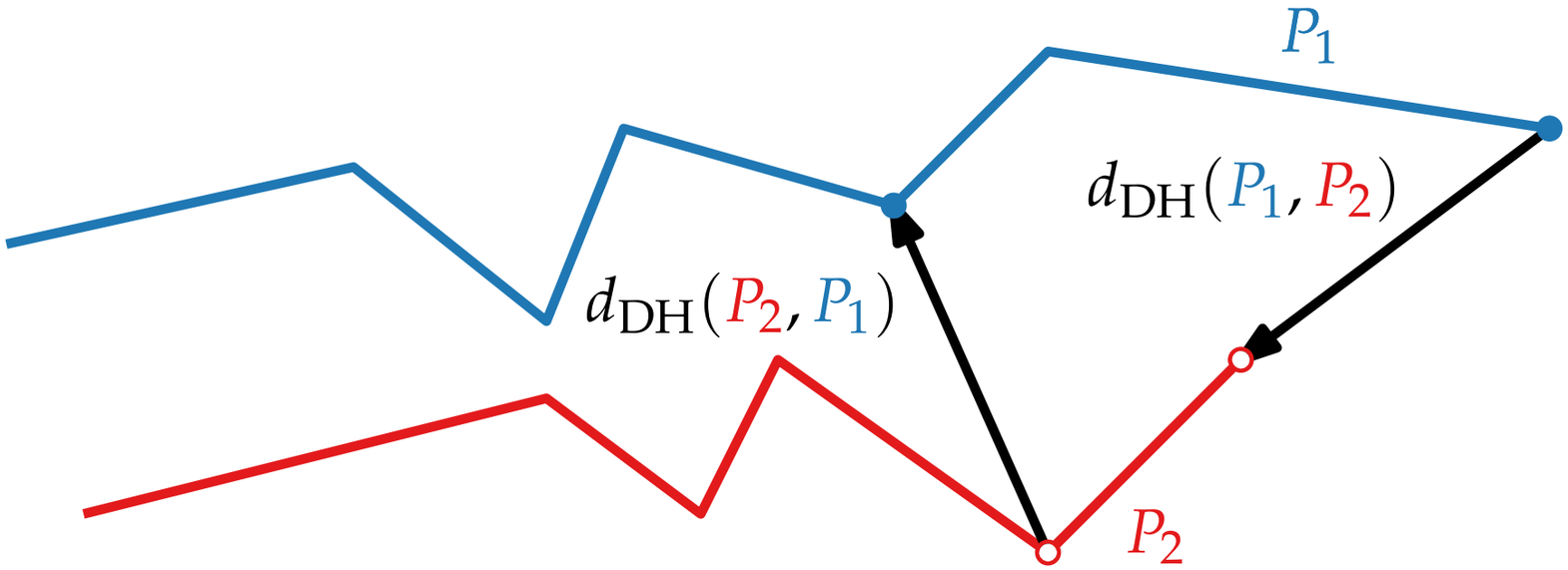
Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?



Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

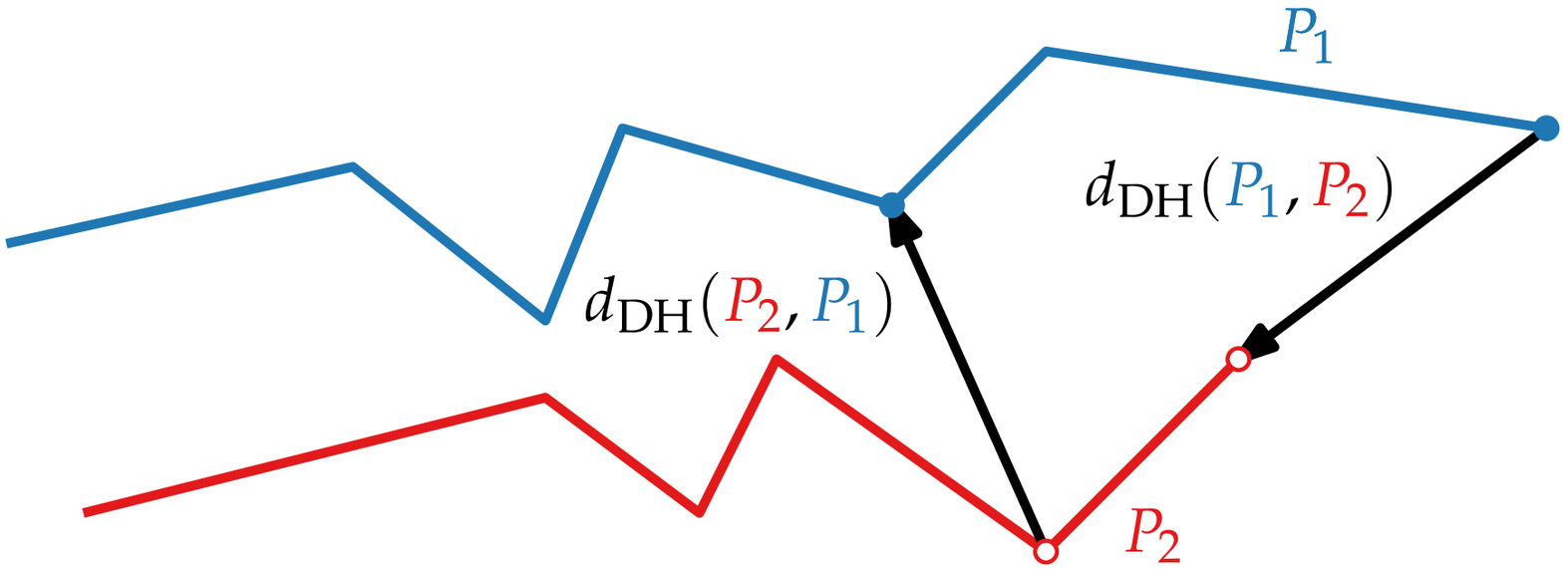
- Berechnen gerichteter Hausdorff Distanz reicht aus.



Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

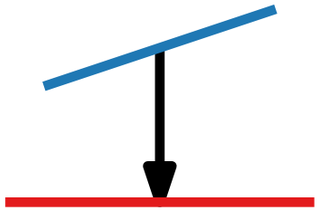
- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.



Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

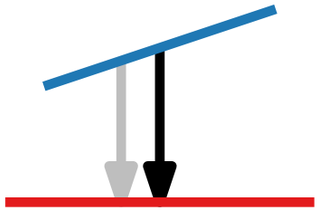
- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.



Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

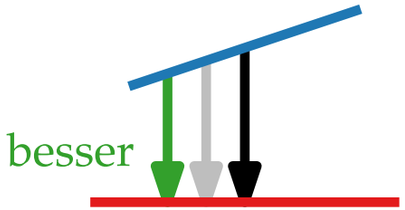
- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.



Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.



Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

➡ Naive Berechnung in $\mathcal{O}(n_1 n_2)$ Zeit.

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

➡ Naive Berechnung in $\mathcal{O}(n_1 n_2)$ Zeit.

➡ Berechnung mit Voronoi-Diagramm und Sweep-Line Algorithmus in $\mathcal{O}((n_1 + n_2)\log(n_1 + n_2))$ Zeit. [Alt et al. 1995]

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

➡ Naive Berechnung in $\mathcal{O}(n_1 n_2)$ Zeit.

➡ Berechnung mit Voronoi-Diagramm und Sweep-Line Algorithmus in $\mathcal{O}((n_1 + n_2)\log(n_1 + n_2))$ Zeit. [Alt et al. 1995]

Die Hausdorff Distanz stellt für jeden Punkt $p_1 \in P_1$
einen **Bezug** zu einem Punkt $p_2 \in P_2$ her.

Hausdorff Distanz – Berechnung

Es gibt unendlich viele Punkte auf P_1 und P_2.
Wie können wir die Berechnung vereinfachen?

- Berechnen gerichteter Hausdorff Distanz reicht aus.
- Hausdorff Distanz bezieht immer **mindestens eine Ecke** ein.

➡ Naive Berechnung in $\mathcal{O}(n_1 n_2)$ Zeit.

➡ Berechnung mit Voronoi-Diagramm und Sweep-Line Algorithmus in $\mathcal{O}((n_1 + n_2)\log(n_1 + n_2))$ Zeit. [Alt et al. 1995]

Die Hausdorff Distanz stellt für jeden Punkt $p_1 \in P_1$
einen **Bezug** zu einem Punkt $p_2 \in P_2$ her.

➡ Wichtig für Map Matching!

Beispiel – Point Pattern Matching

Typische Aufgabe:

Beispiel – Point Pattern Matching

Typische Aufgabe:

Gegeben: Zwei diskrete Punktmenge P_1, P_2

Beispiel – Point Pattern Matching

Typische Aufgabe:

Gegeben: Zwei diskrete Punktmenge P_1, P_2

Beispiel – Point Pattern Matching

Typische Aufgabe:

Gegeben: Zwei diskrete Punktmenge P_1, P_2

Gesucht: Finde Rotation/Verschiebung f, so dass $d_H(f(P_1), P_2)$ minimal ist.

Beispiel – Point Pattern Matching

Typische Aufgabe:

Gegeben: Zwei diskrete Punktmenge P_1, P_2

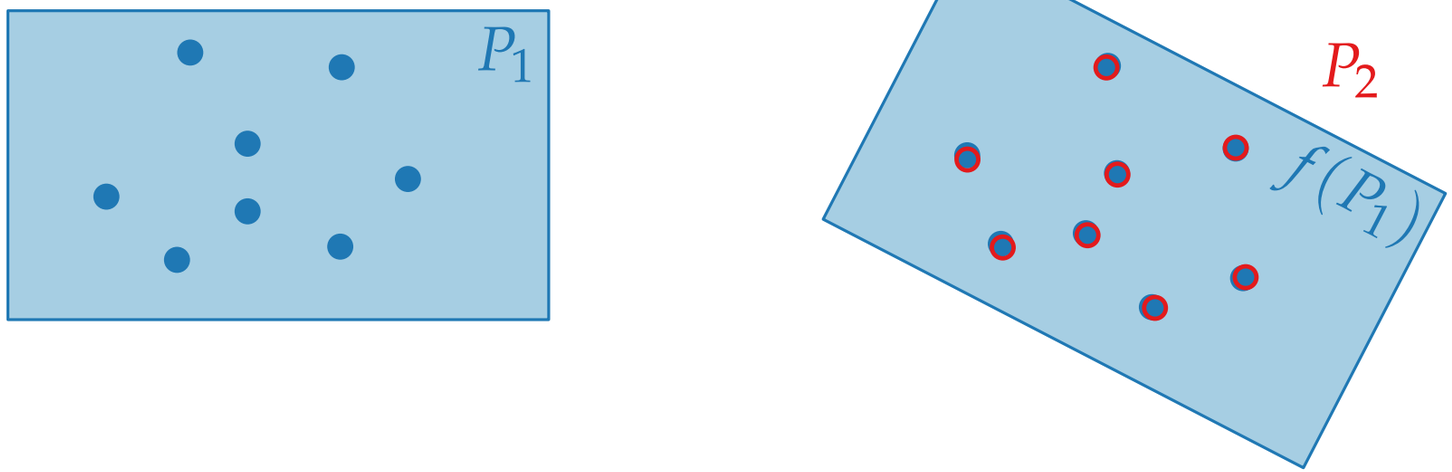
Gesucht: Finde Rotation/Verschiebung f, so dass $d_H(f(P_1), P_2)$ minimal ist.

Beispiel – Point Pattern Matching

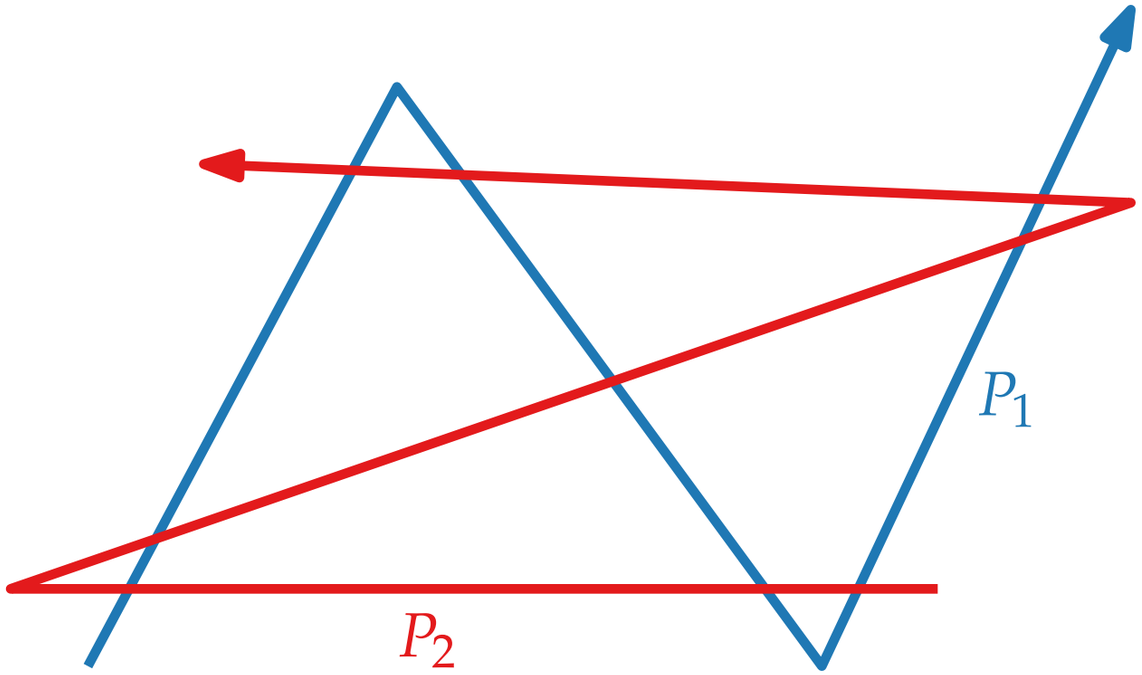
Typische Aufgabe:

Gegeben: Zwei diskrete Punktmenge P_1, P_2

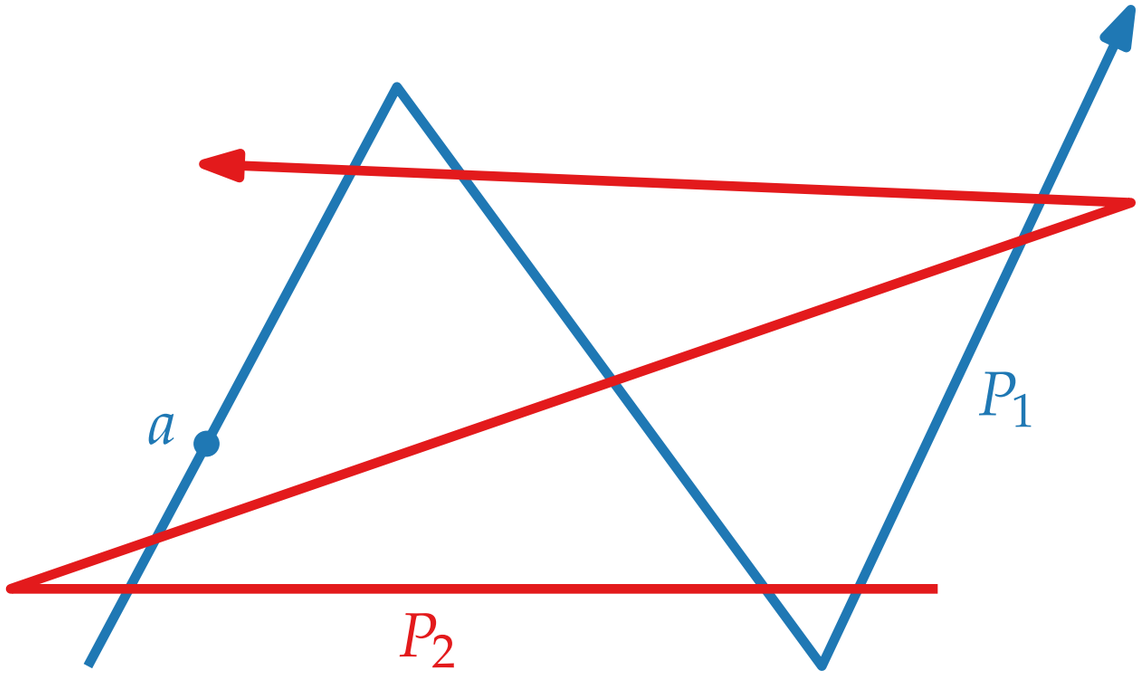
Gesucht: Finde Rotation/Verschiebung f, so dass $d_H(f(P_1), P_2)$ minimal ist.



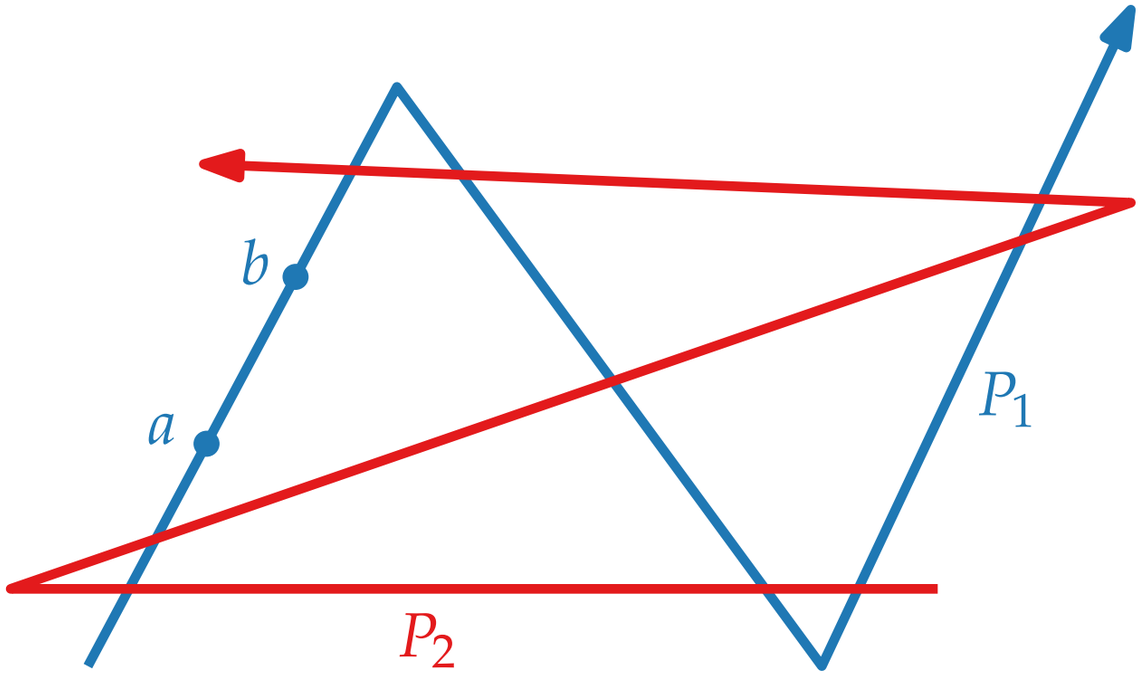
Hausdorff Distanz – Probleme?



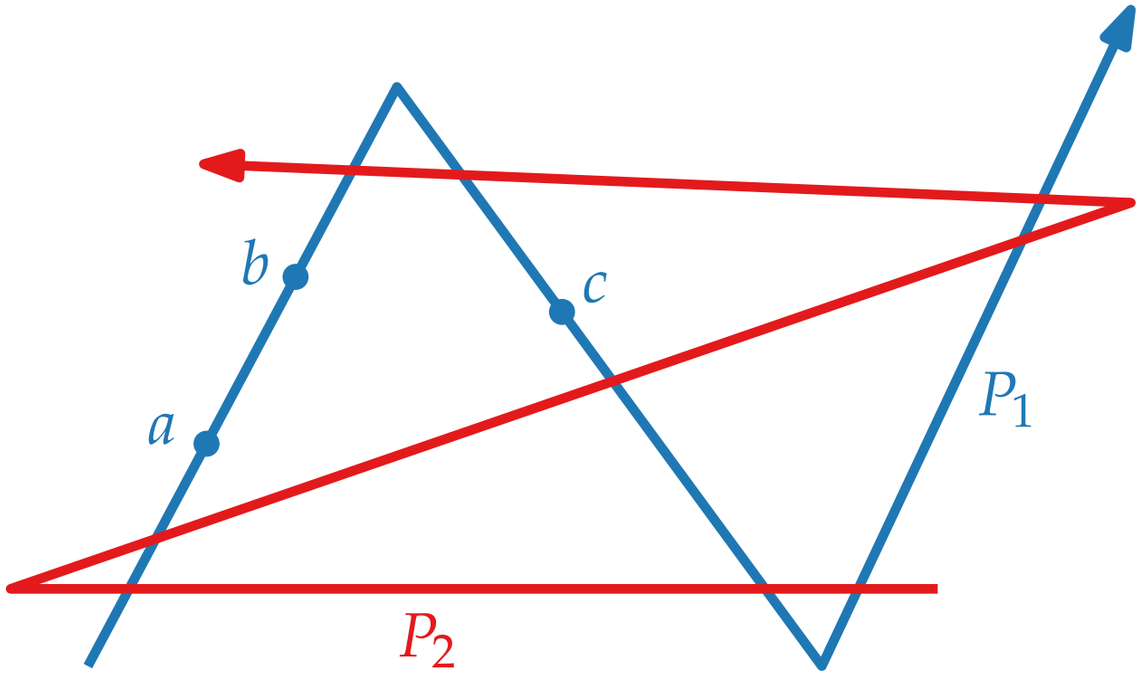
Hausdorff Distanz – Probleme?



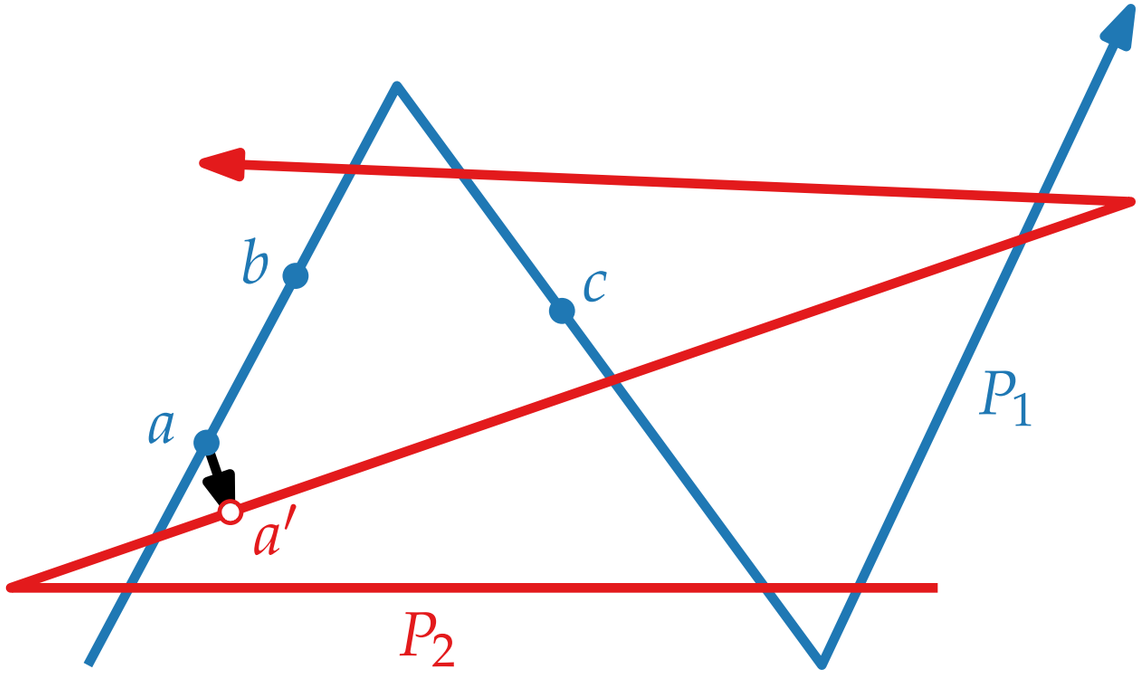
Hausdorff Distanz – Probleme?



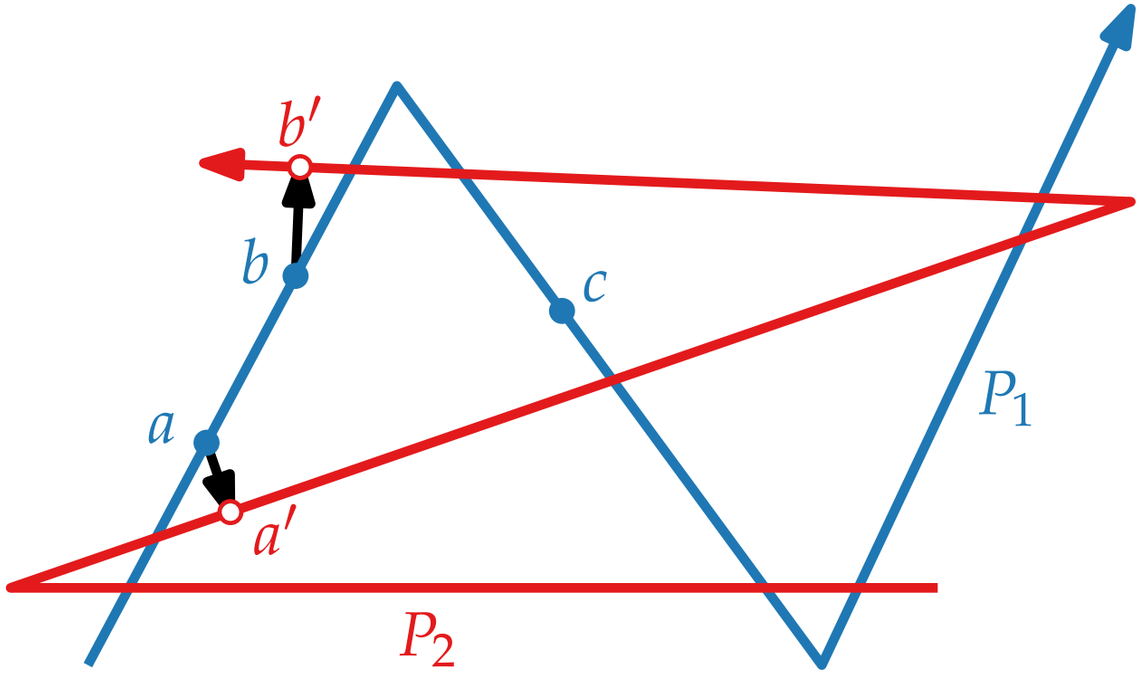
Hausdorff Distanz – Probleme?



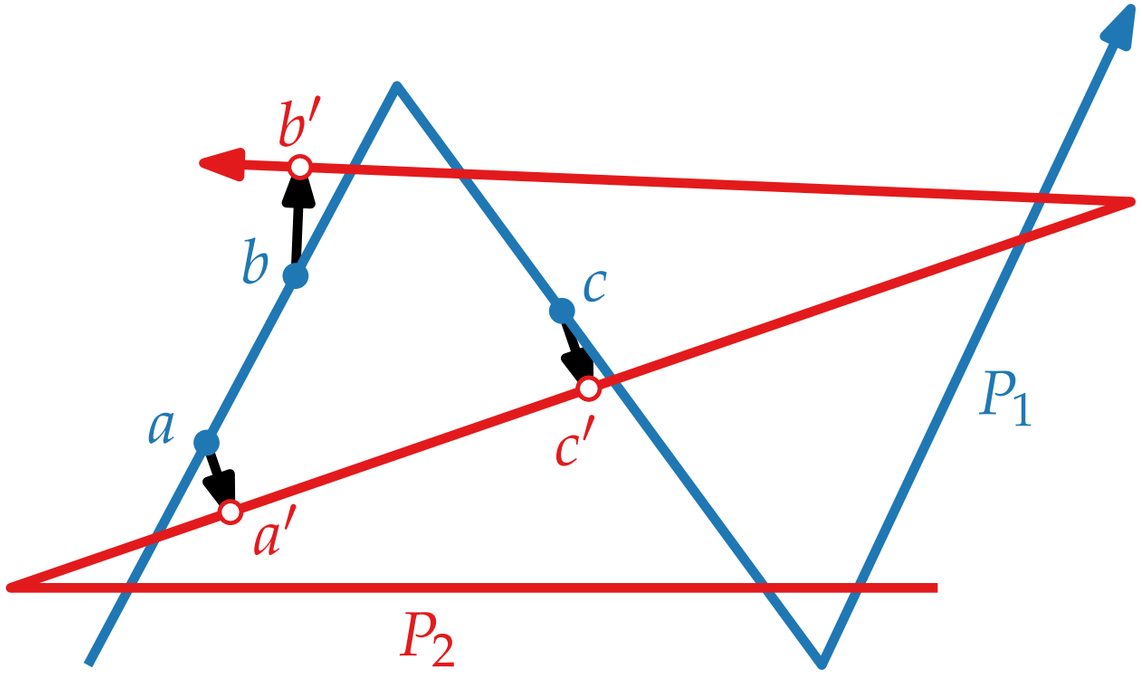
Hausdorff Distanz – Probleme?



Hausdorff Distanz – Probleme?

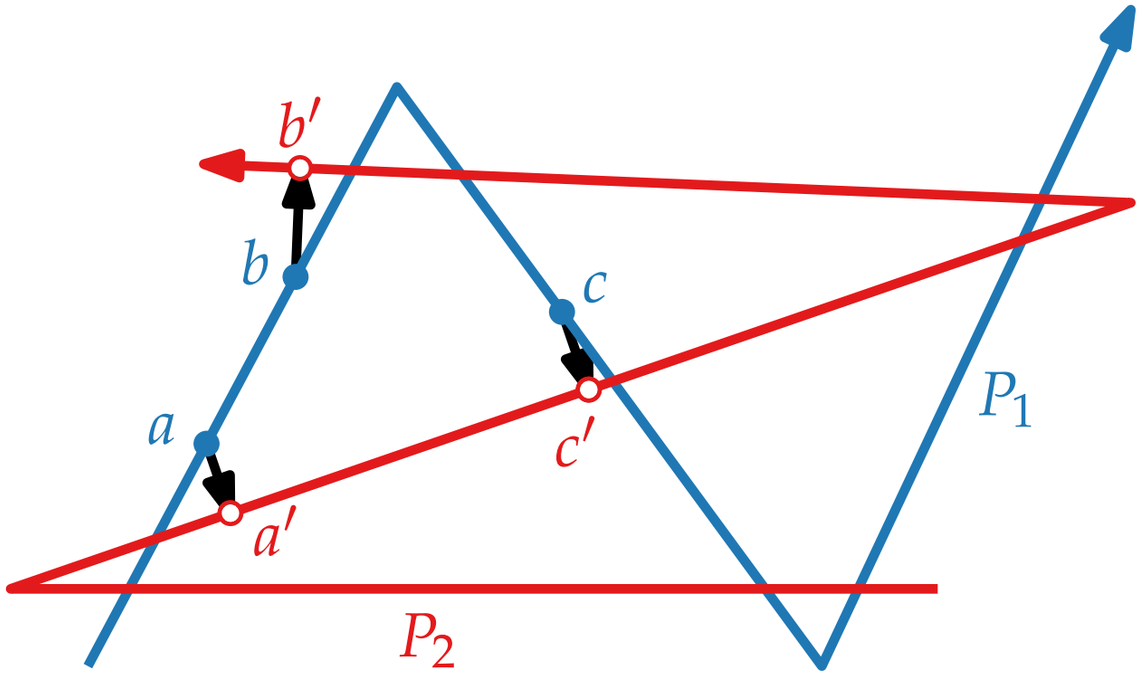


Hausdorff Distanz – Probleme?



Hausdorff Distanz – Probleme?

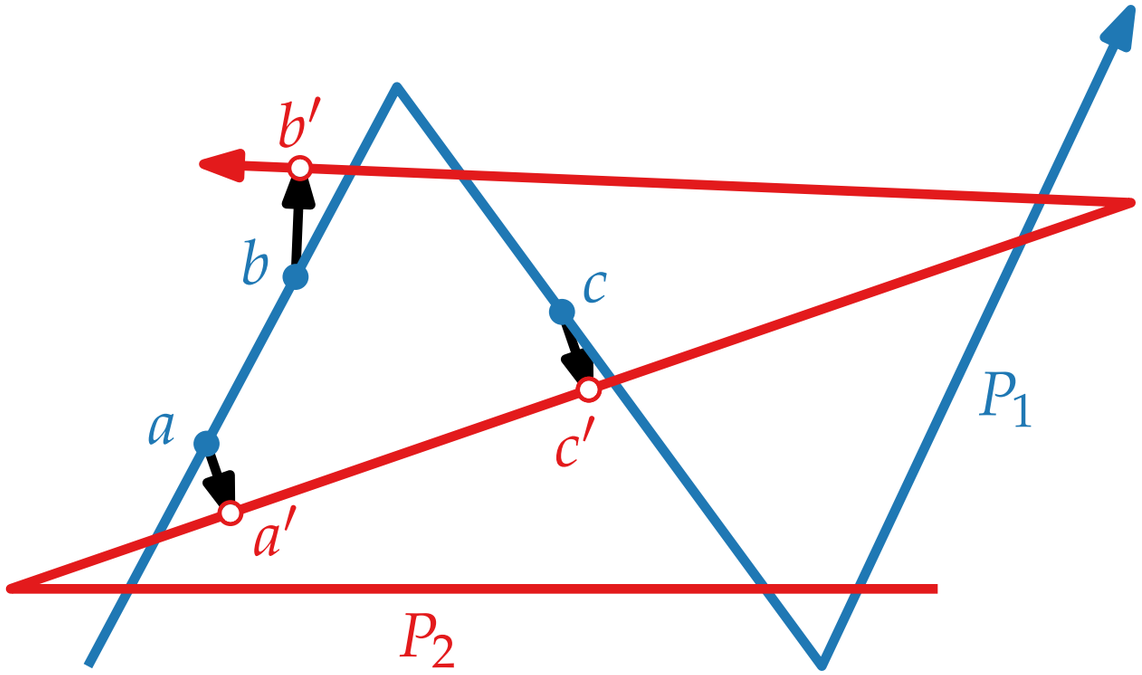
Bezugspunkte auf P_2 halten nicht die Reihenfolge der Punkte auf P_1 ein!



Hausdorff Distanz – Probleme?

Bezugspunkte auf P_2 halten nicht die Reihenfolge der Punkte auf P_1 ein!

$P_1 : a \rightarrow b \rightarrow c$

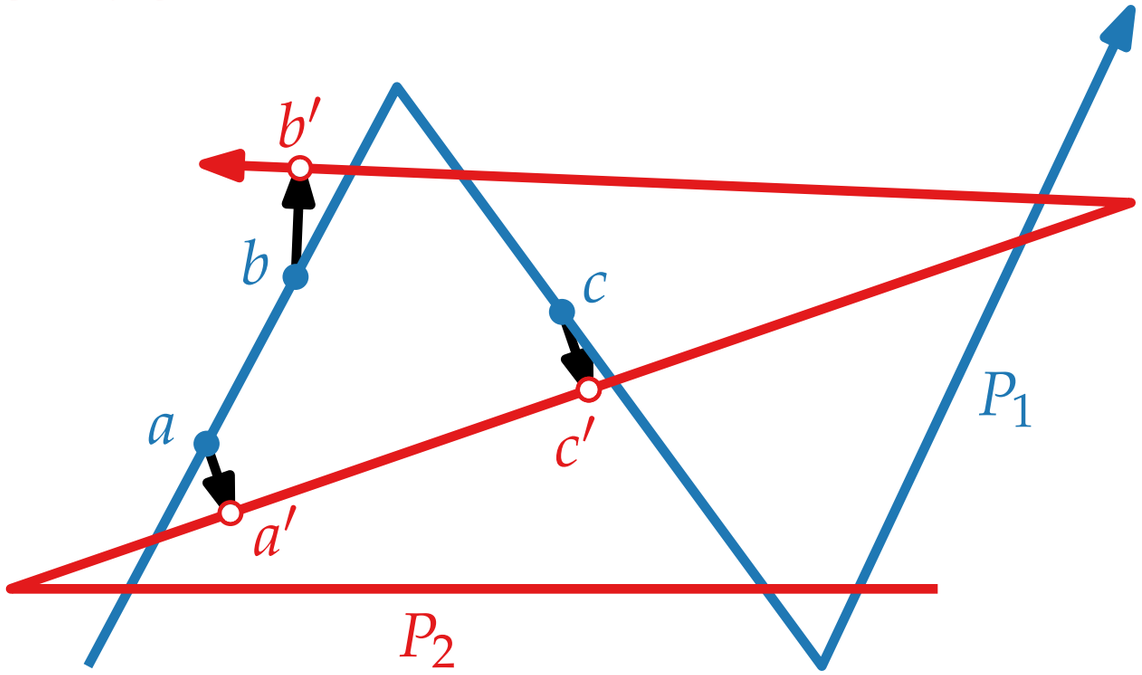


Hausdorff Distanz – Probleme?

Bezugspunkte auf P_2 halten nicht die Reihenfolge der Punkte auf P_1 ein!

$P_1 : a \rightarrow b \rightarrow c$

$P_2 : a' \rightarrow c' \rightarrow b'$

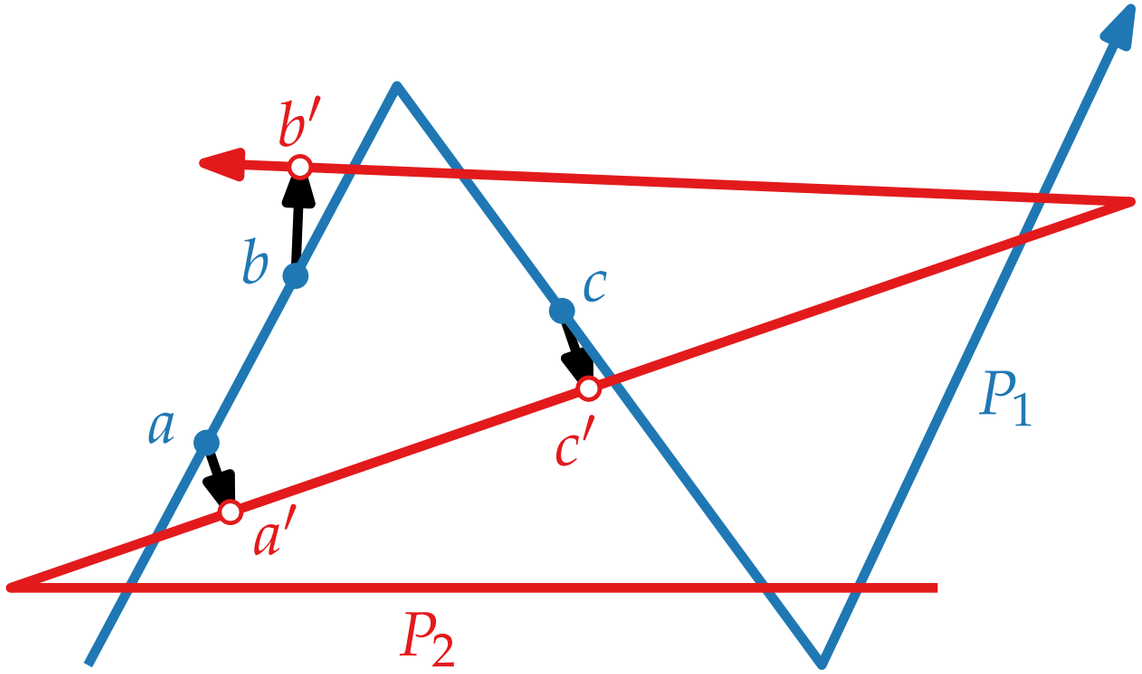


Hausdorff Distanz – Probleme?

Bezugspunkte auf P_2 halten nicht die Reihenfolge der Punkte auf P_1 ein!

$P_1 : a \rightarrow b \rightarrow c$

$P_2 : a' \rightarrow c' \rightarrow b'$

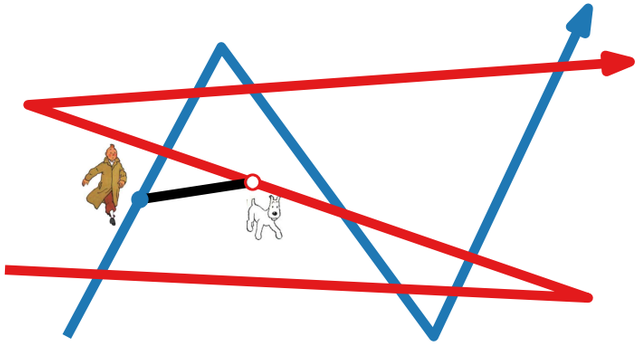


➡ Zuordnung der Punkte nicht plausibel.

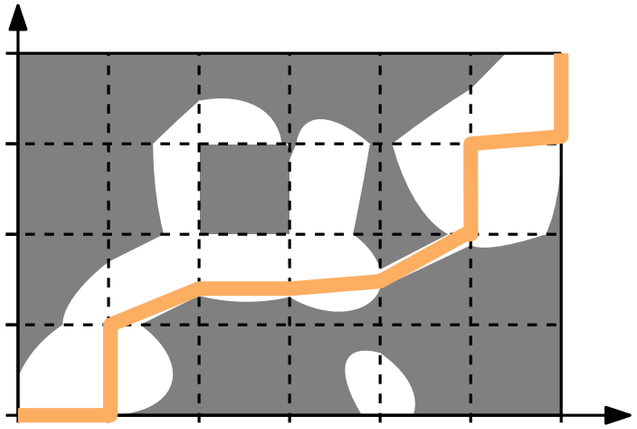
Universität Trier

Algorithmen für geographische Informationssysteme

3. Vorlesung
Map Matching



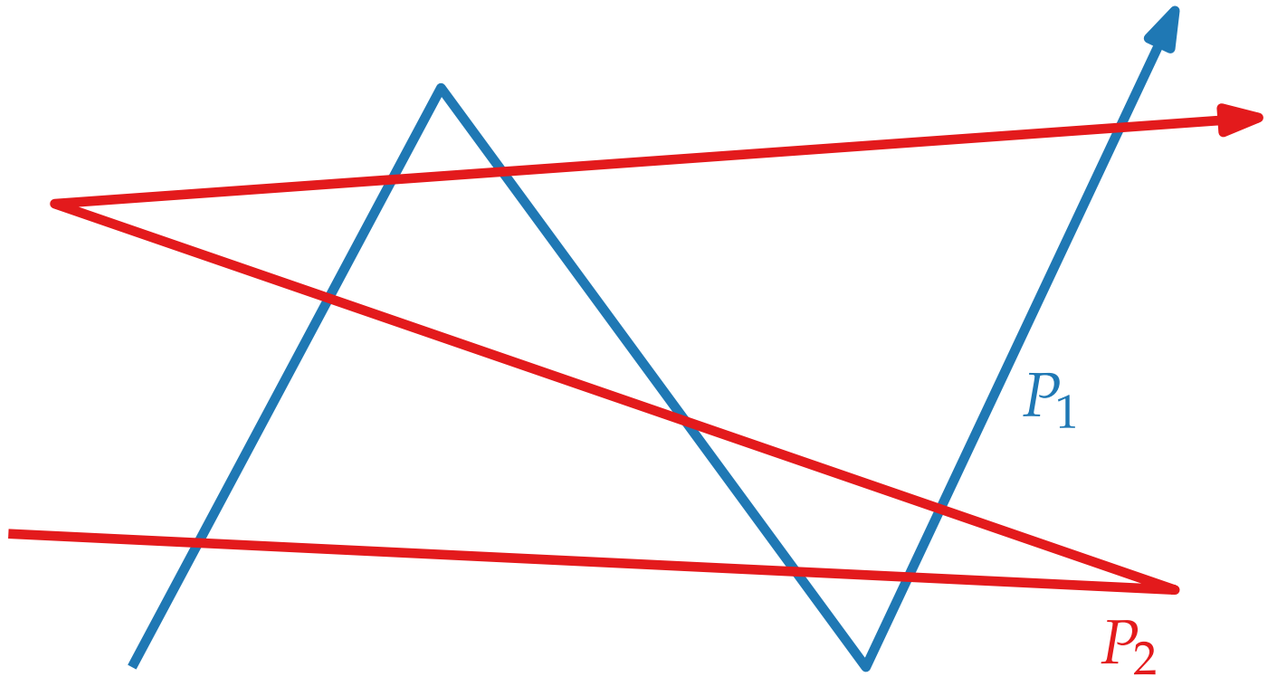
Teil III:
Fréchet Distanz



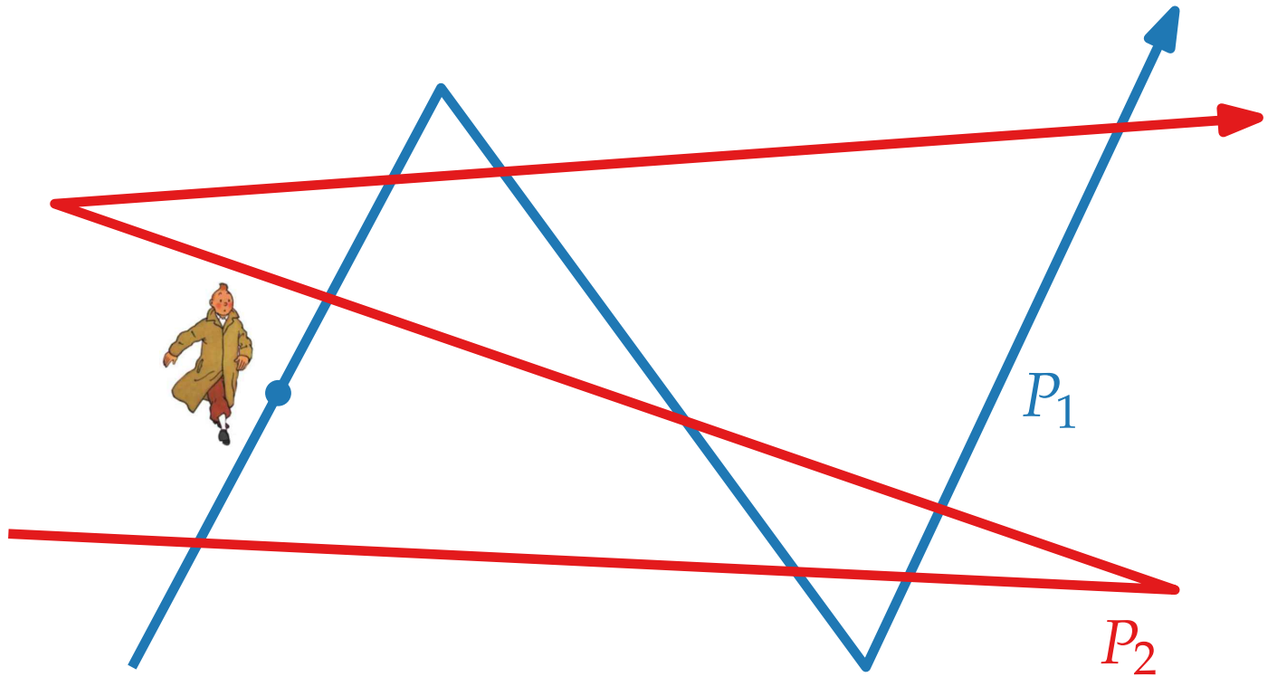
Philipp Kindermann

Wintersemester 2024/25

Fréchet Distanz – Idee

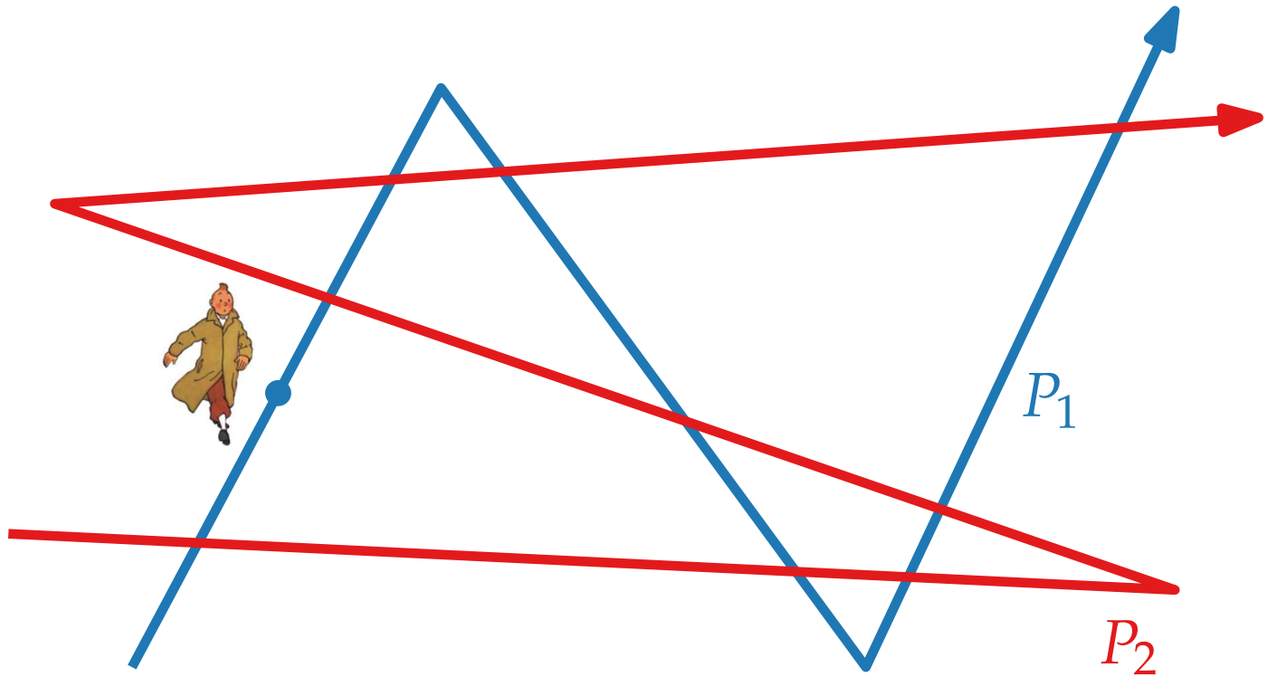


Fréchet Distanz – Idee



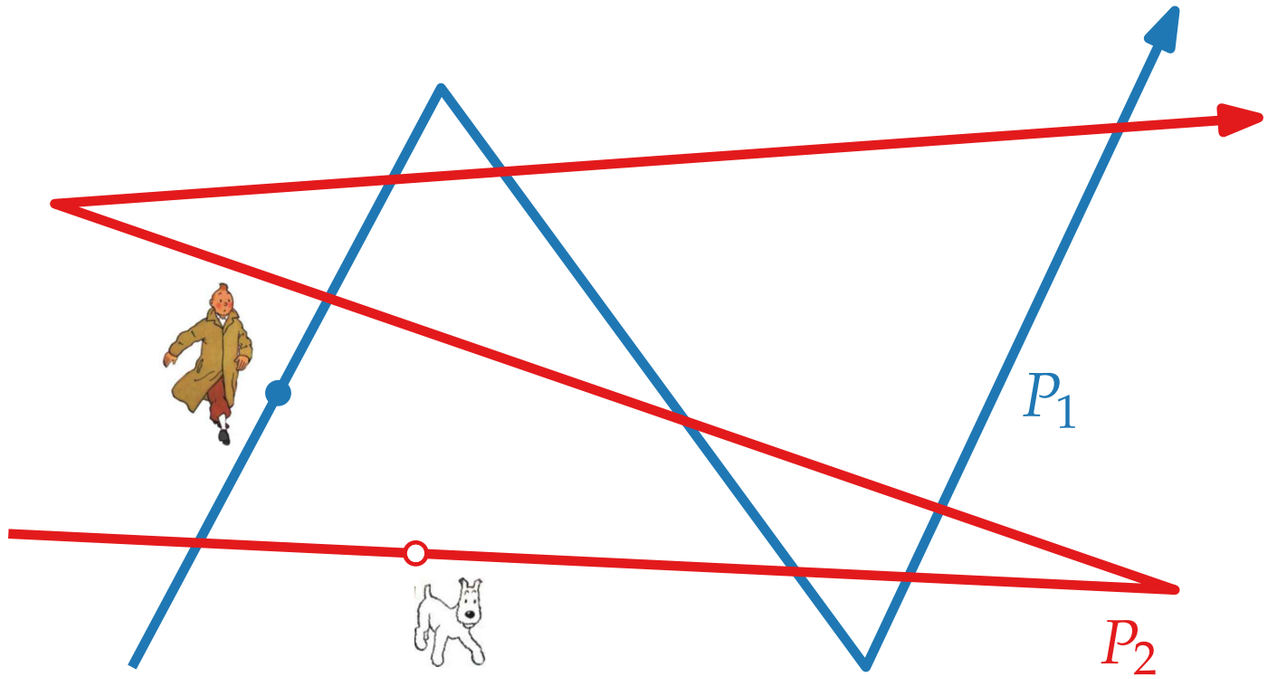
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.



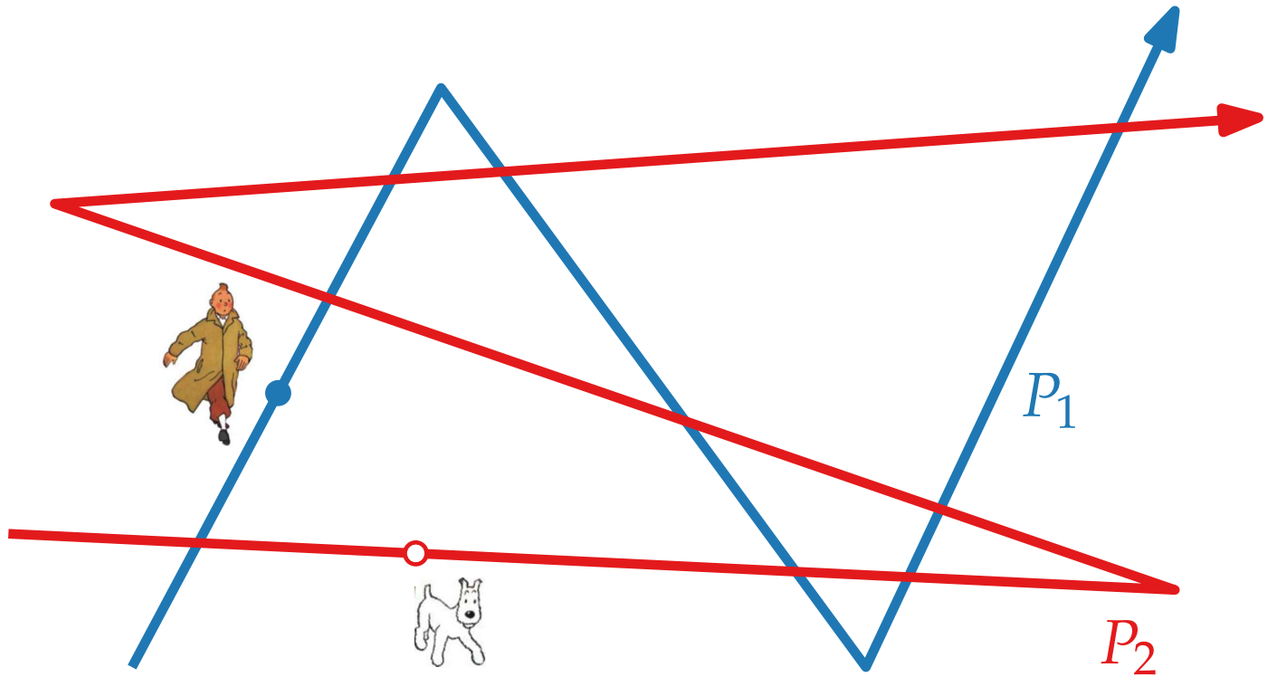
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.



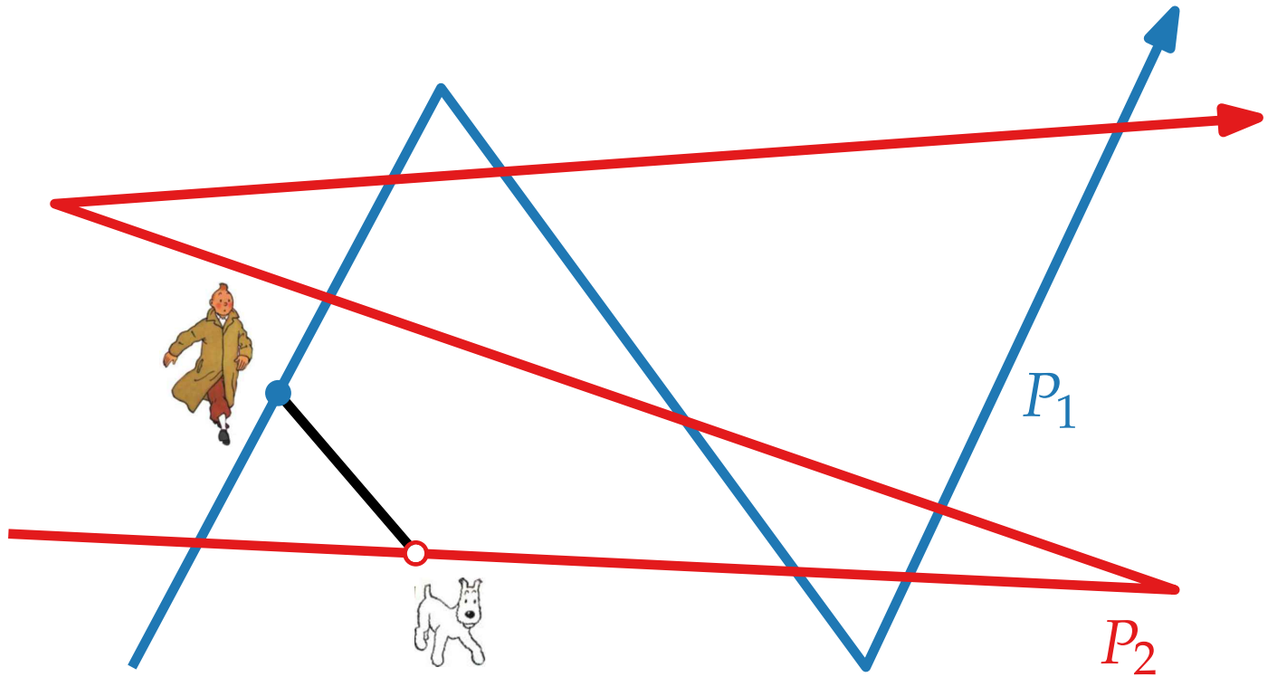
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.



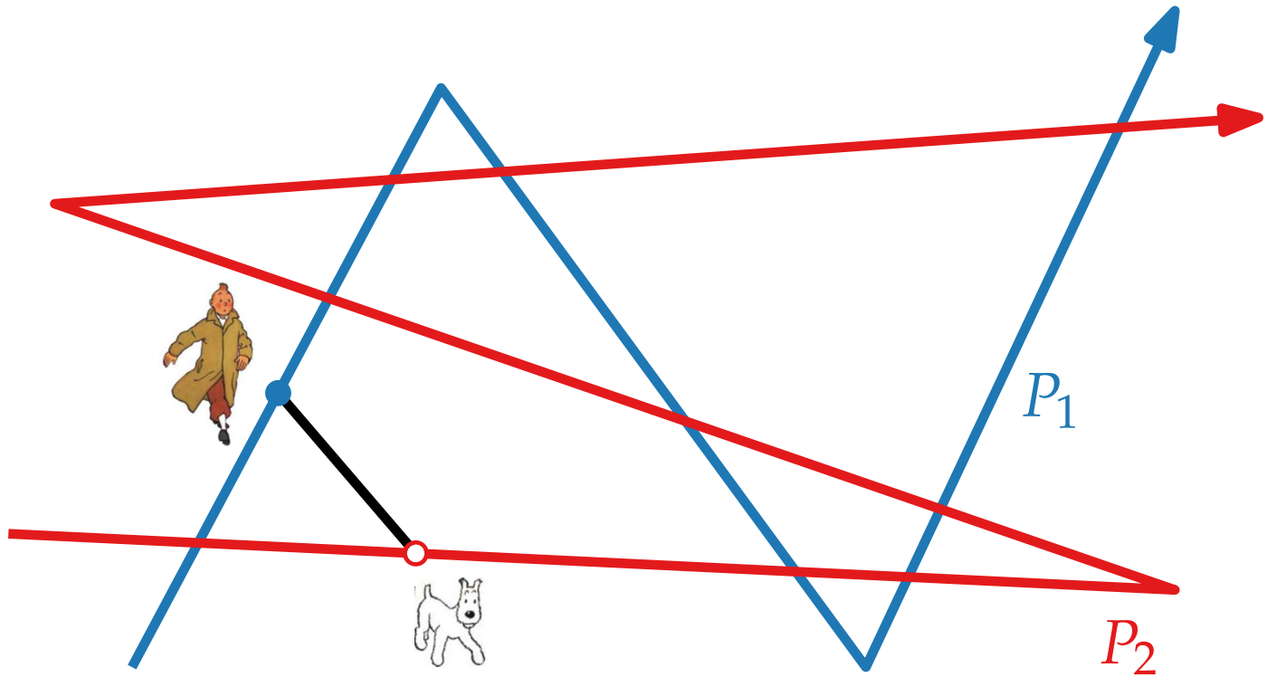
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.



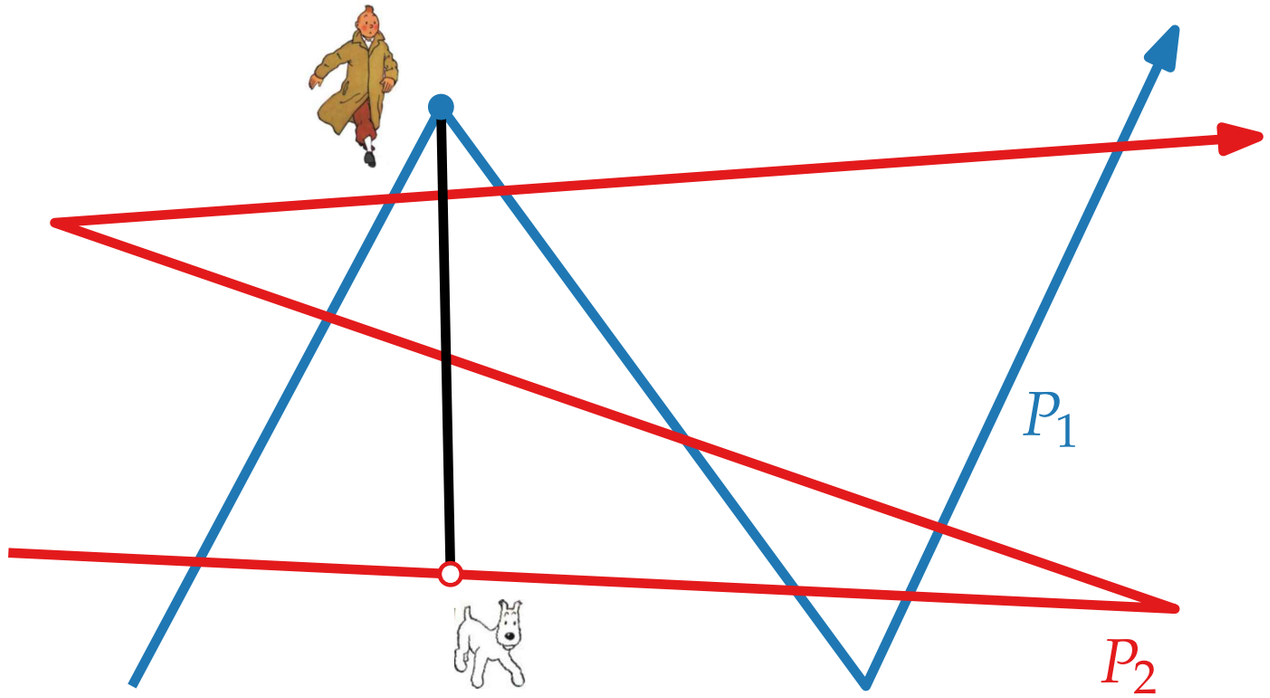
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



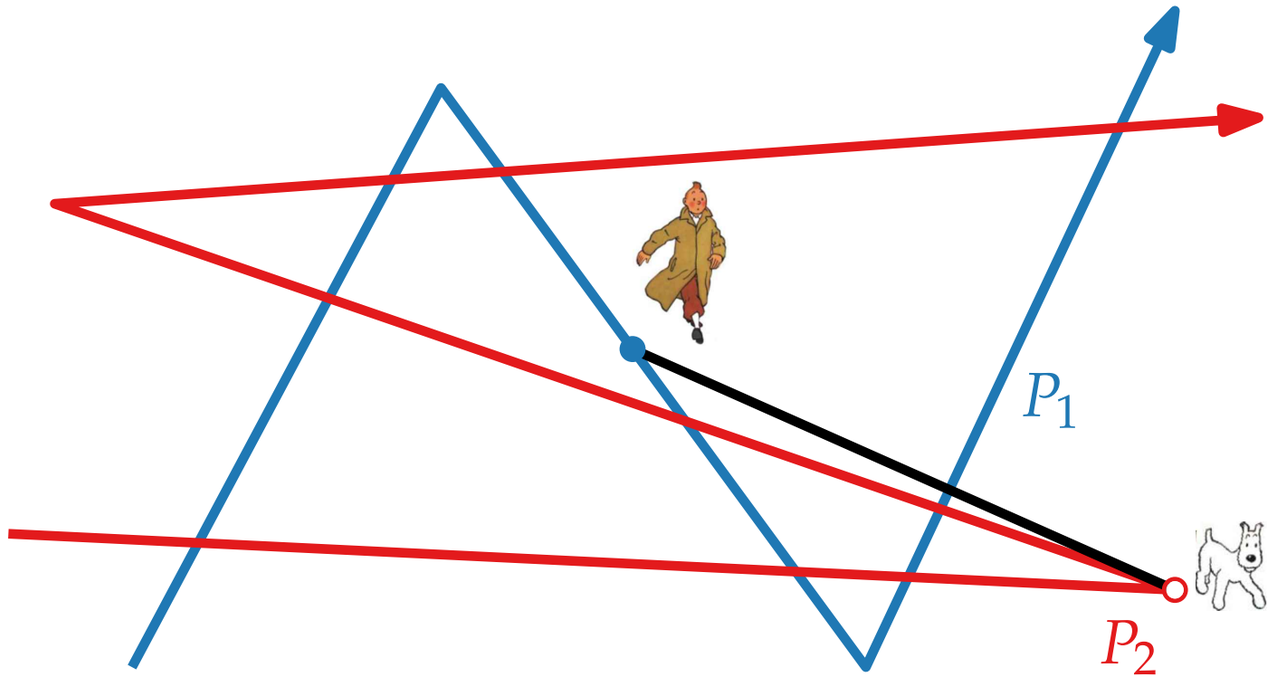
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



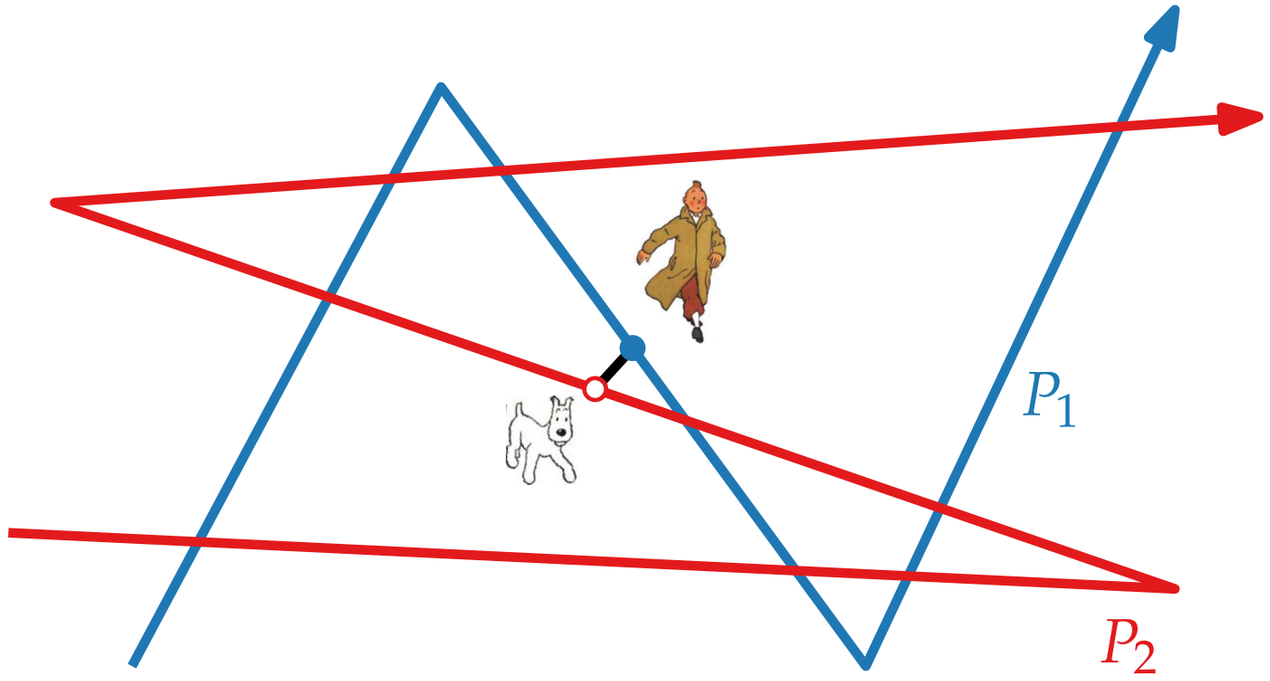
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



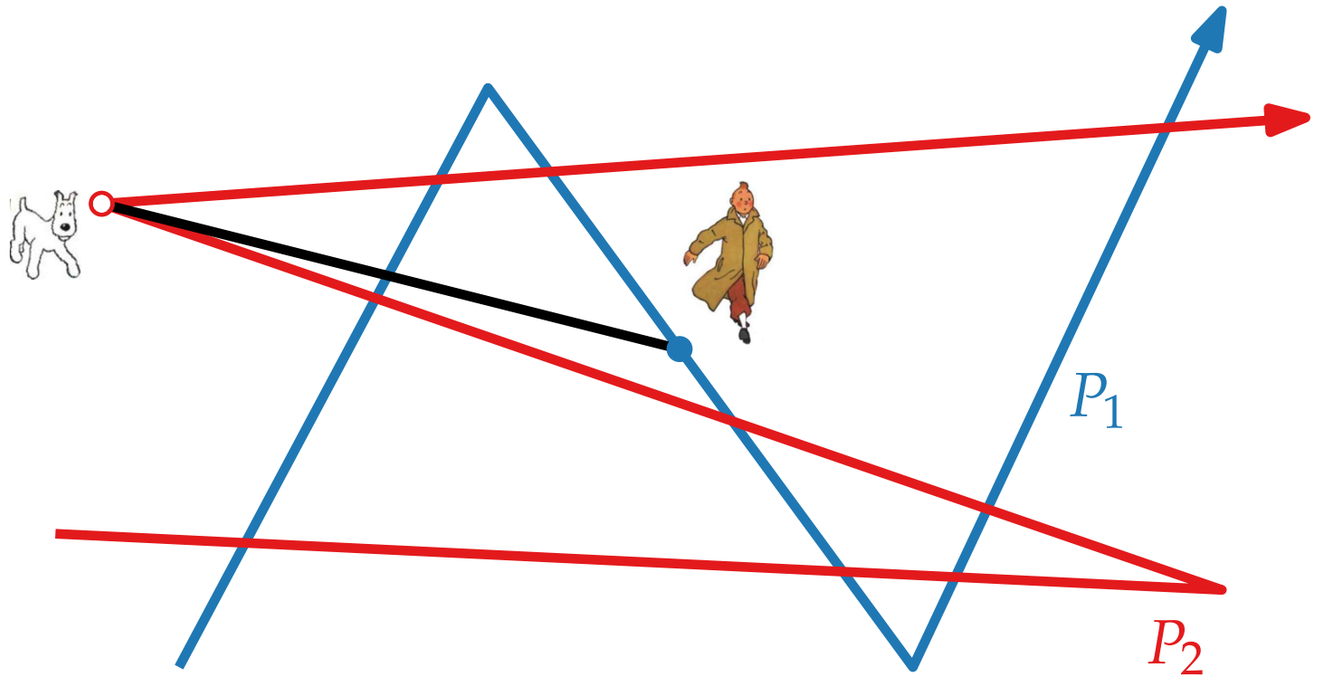
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



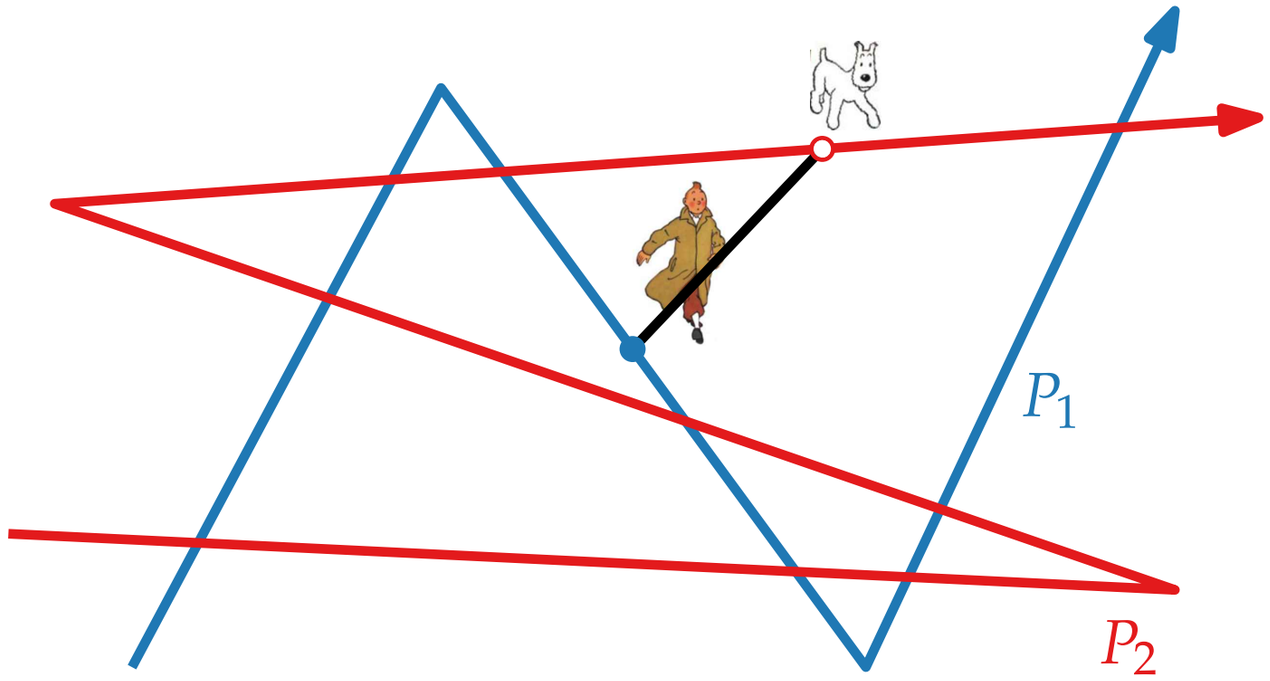
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



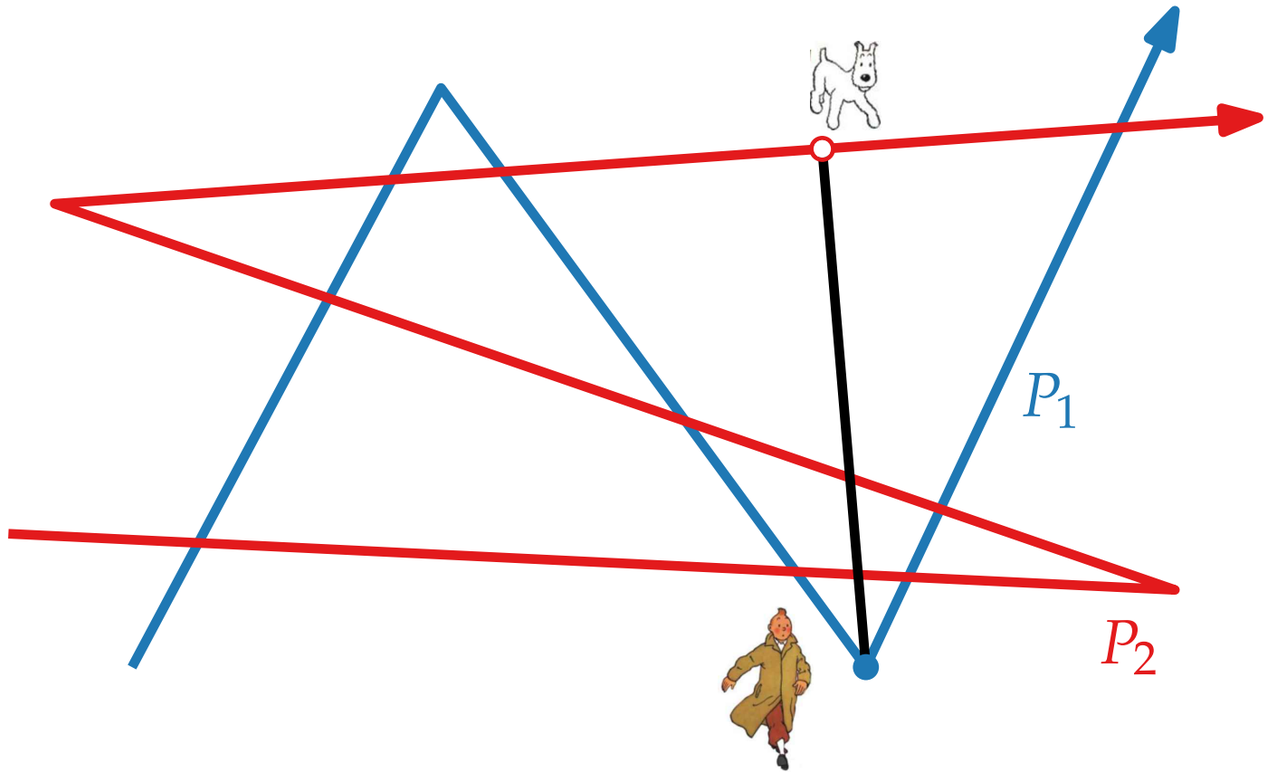
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



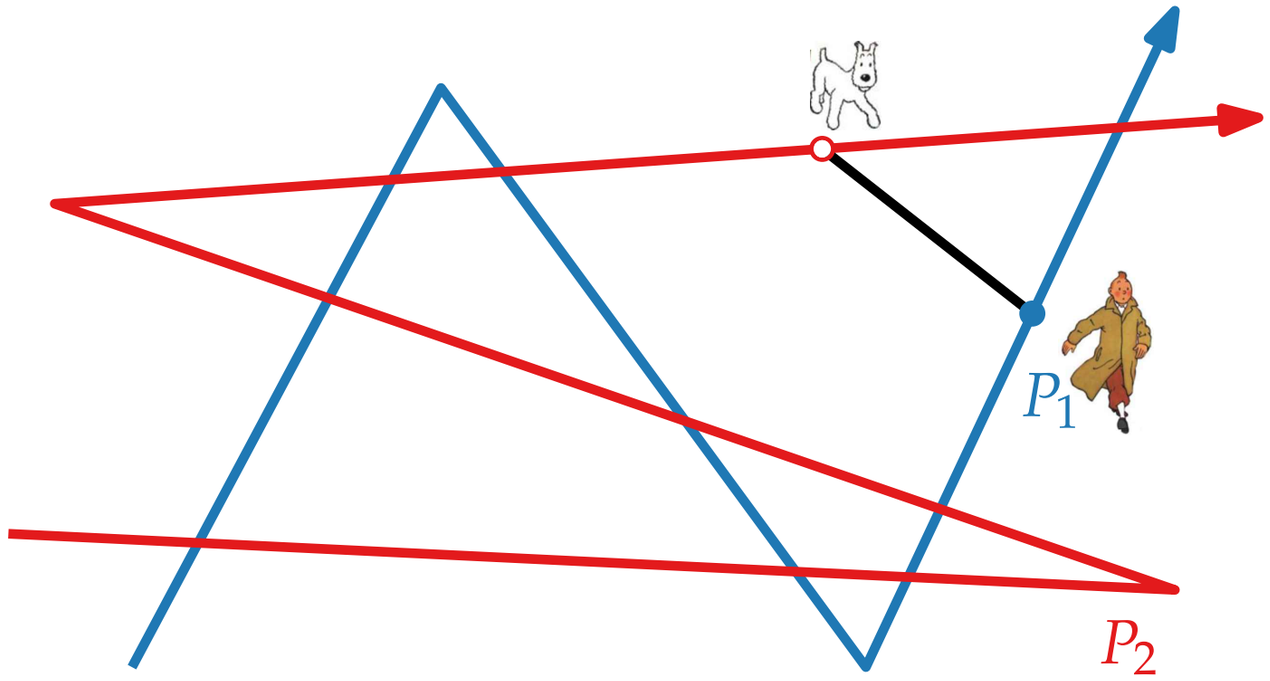
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



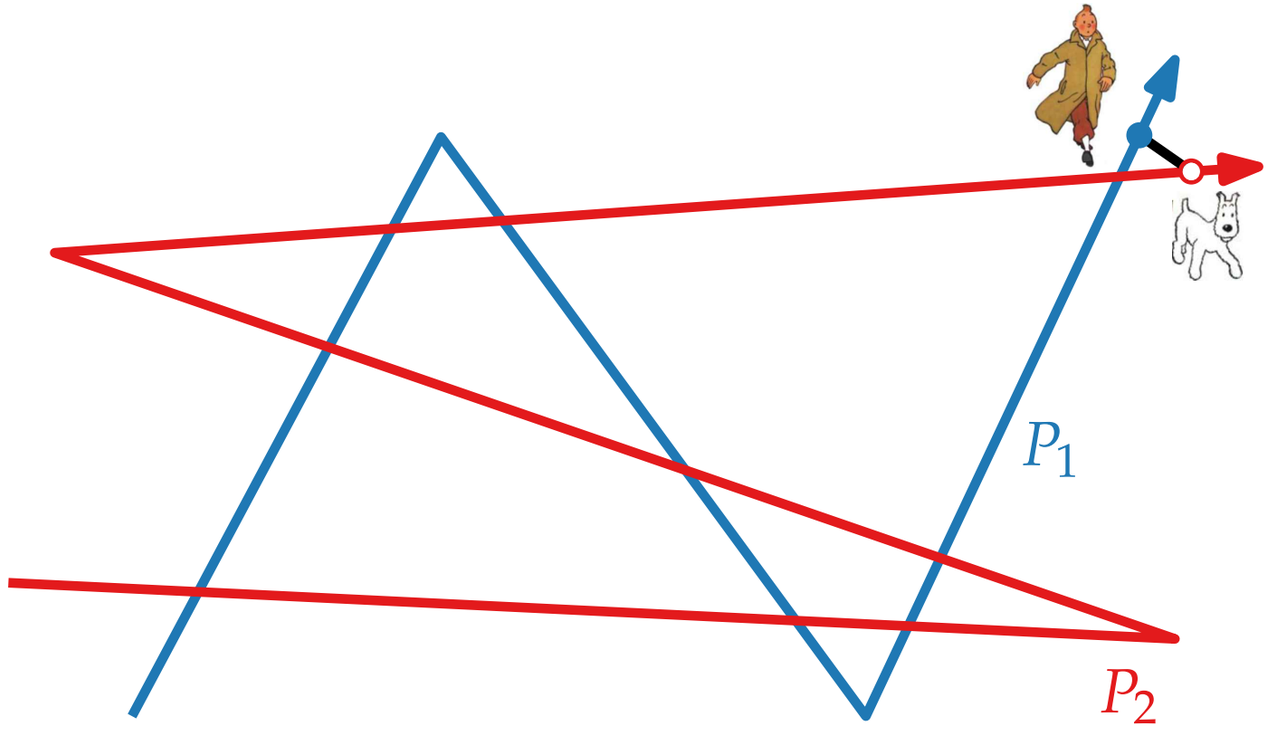
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



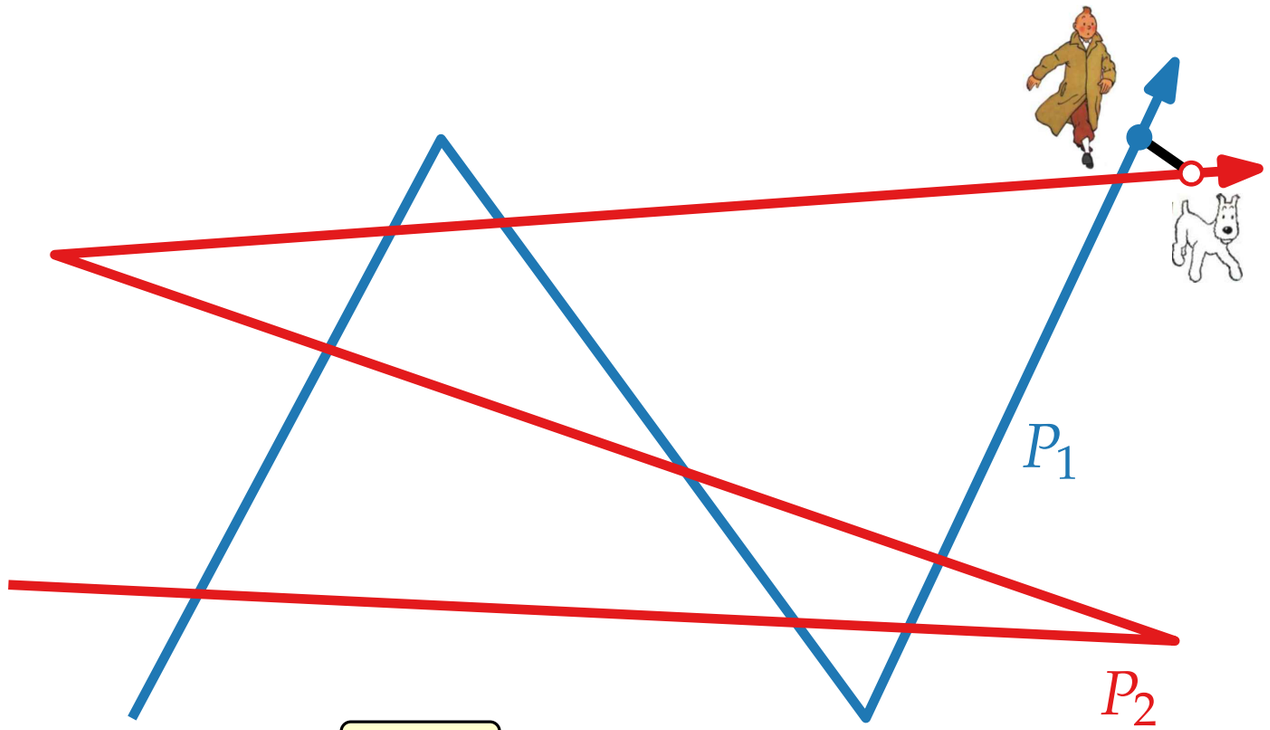
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



Fréchet Distanz – Idee

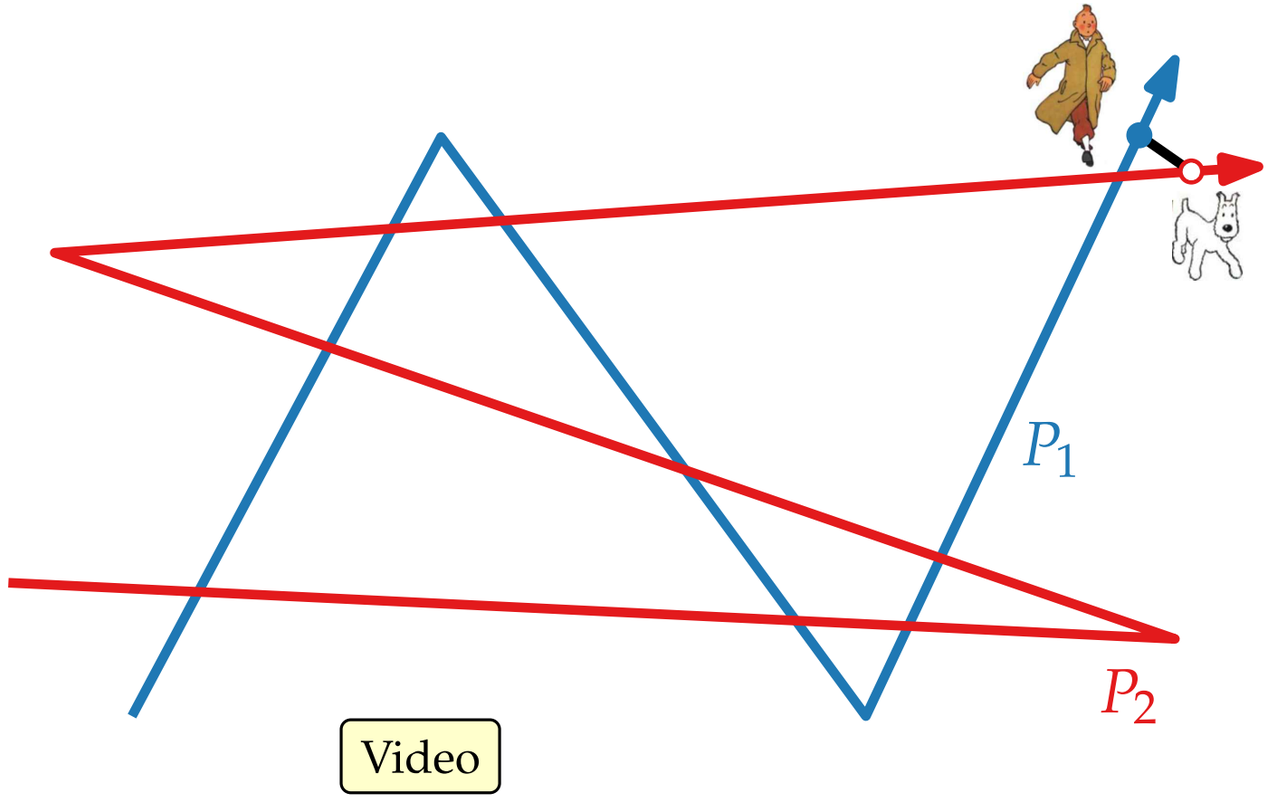
- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.



Video

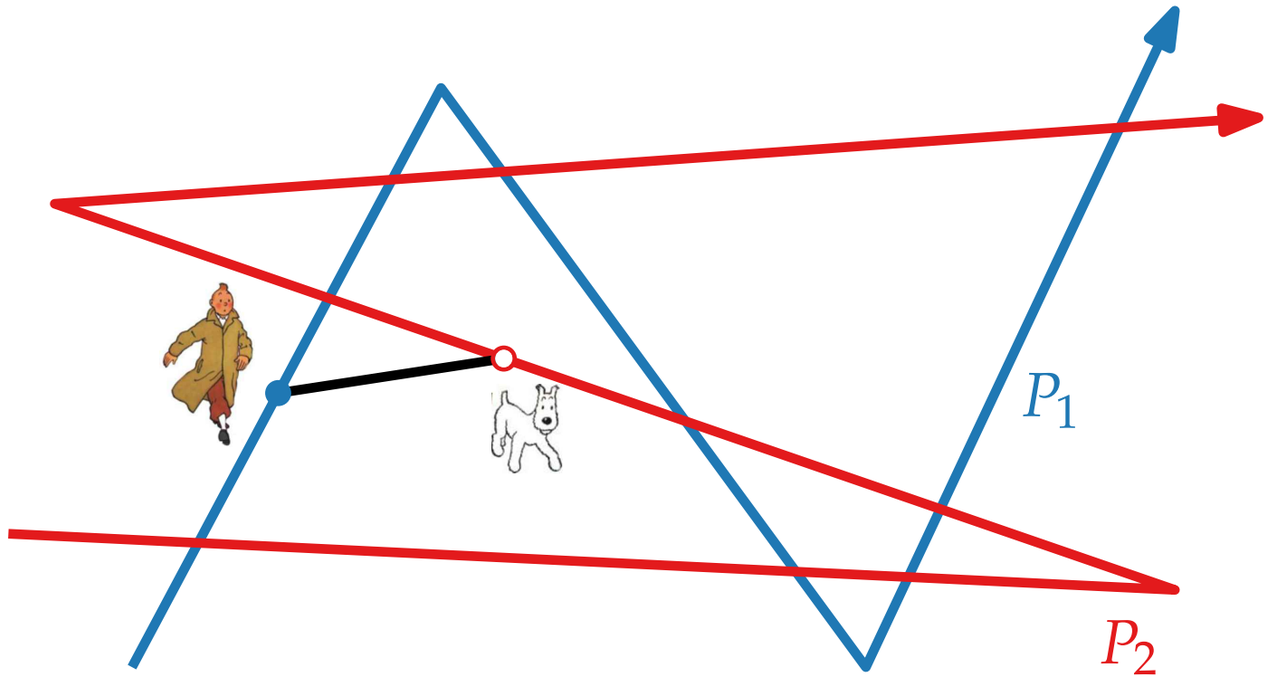
Fréchet Distanz – Idee

- **Herrchen** läuft entlang P_1 ohne jemals umzukehren.
- Hund läuft entlang P_2 ohne jemals umzukehren.
- Herrchen und Hund sind über **Hundeleine** verbunden.
- Wie lang muss die Hundeleine mindestens sein?



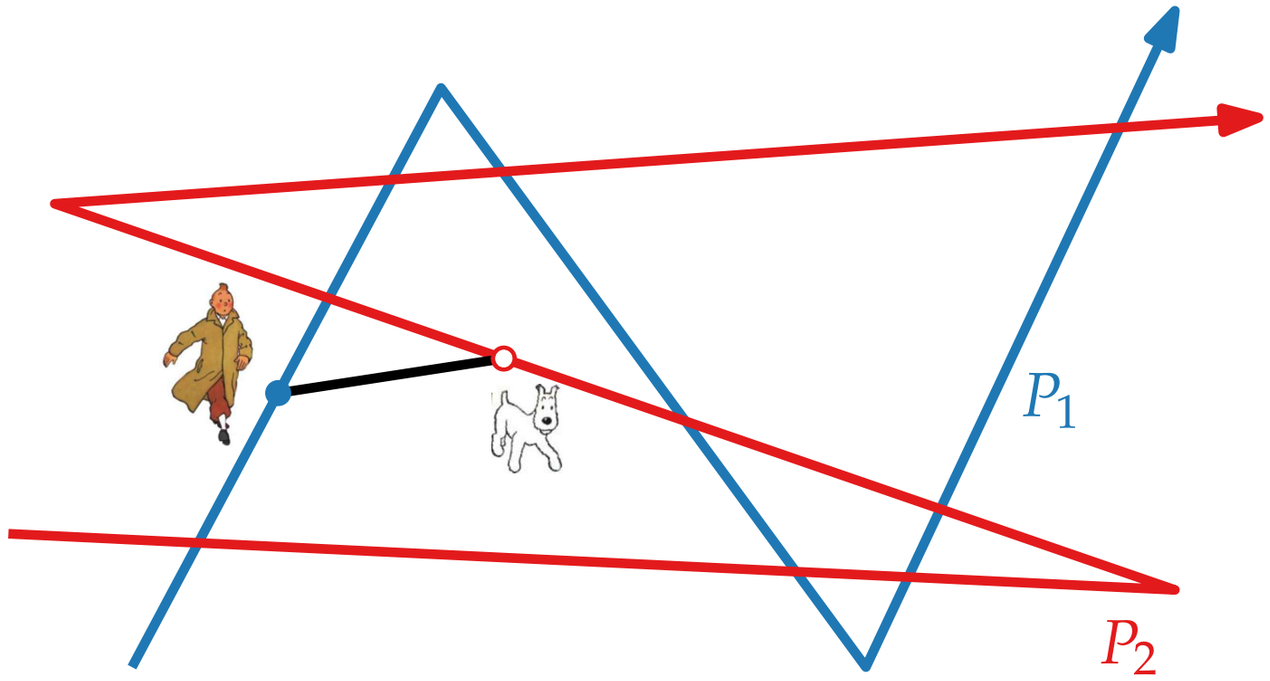
Video

Parametrisierte Kurve



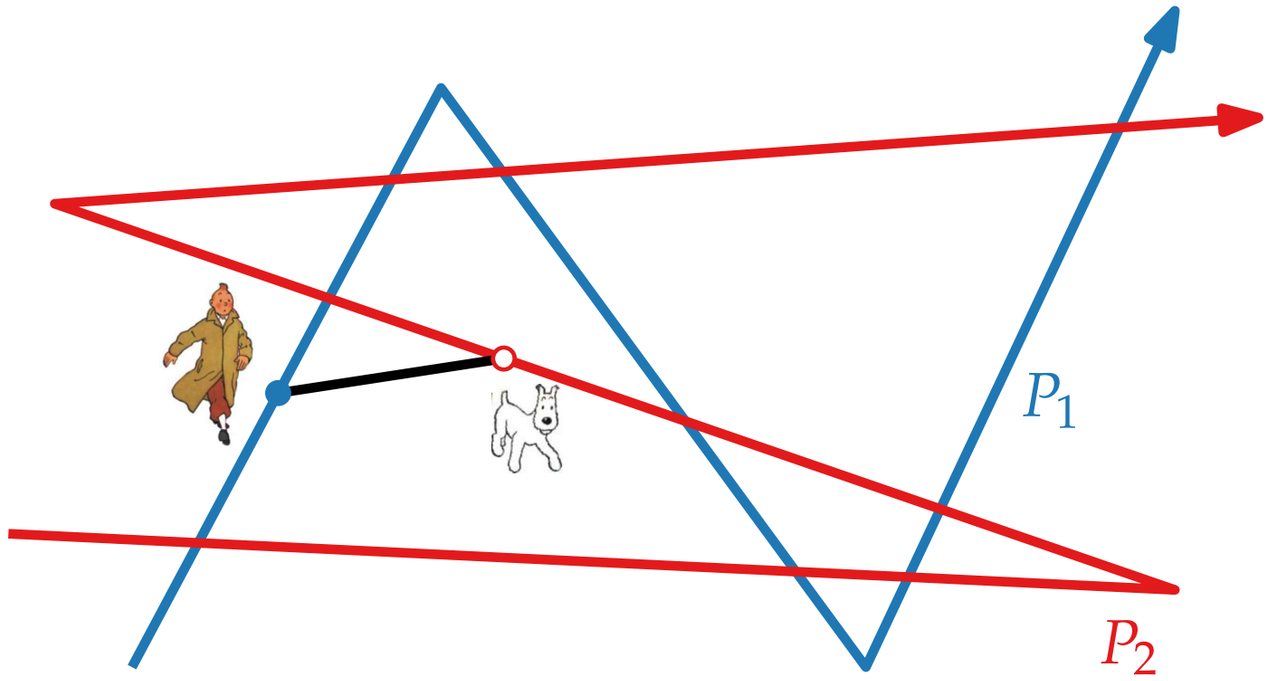
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.



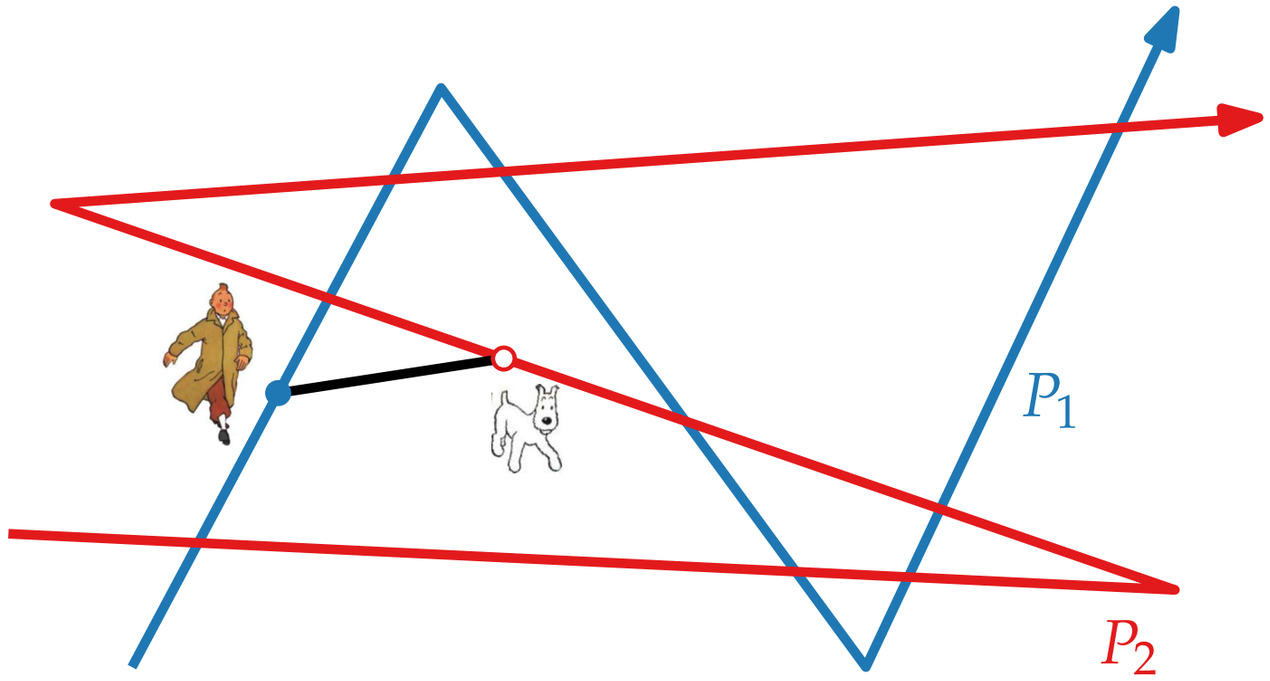
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.



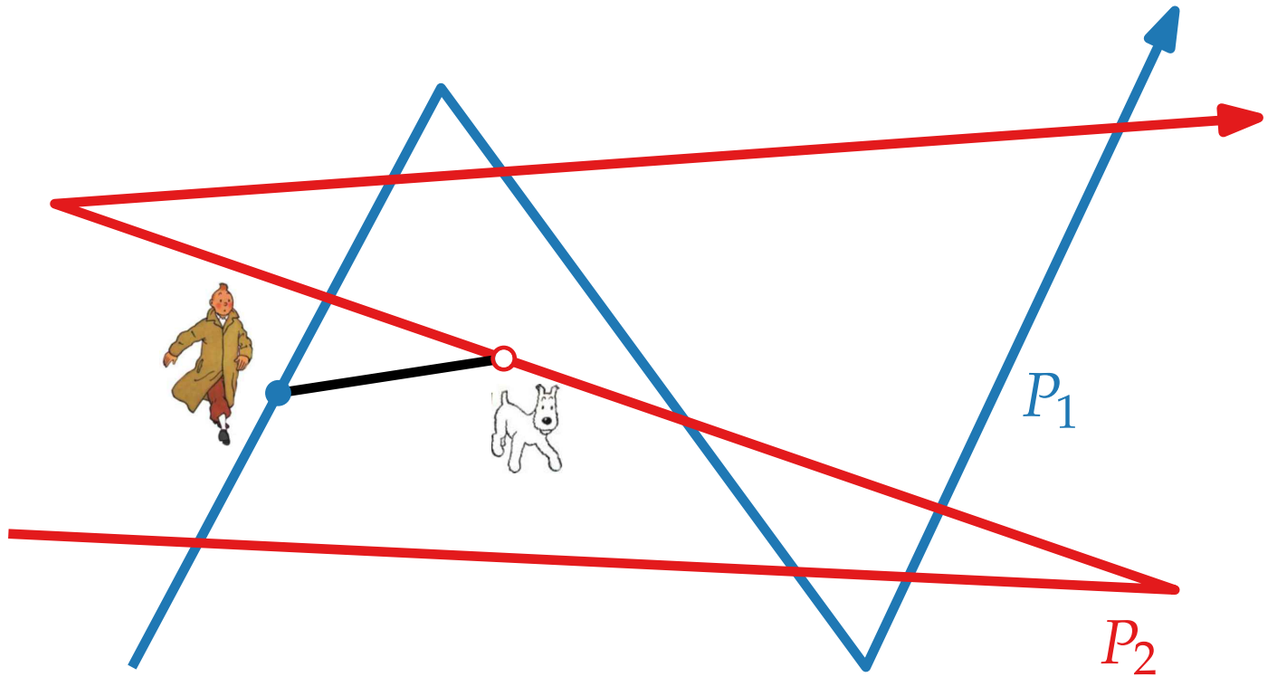
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.
- $P_1(t)$ mit $0 \leq t \leq n_1 - 1$ ist ein Punkt auf P_1.



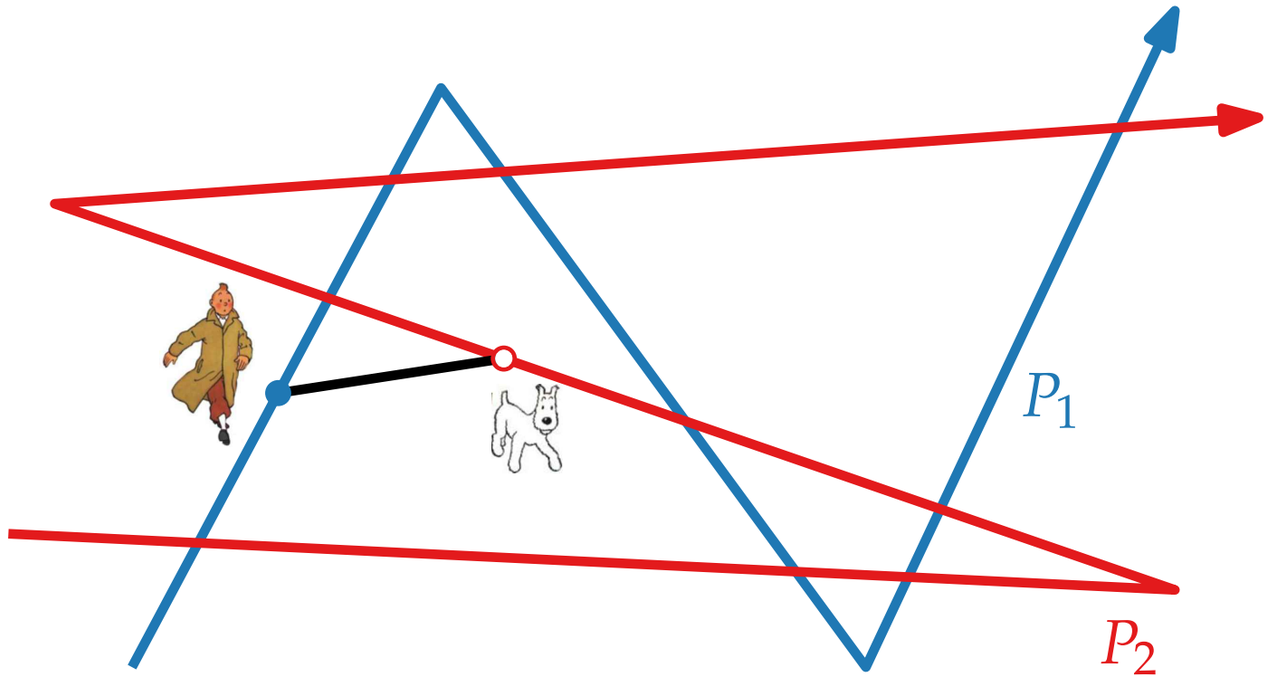
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.
- $P_1(t)$ mit $0 \leq t \leq n_1 - 1$ ist ein Punkt auf P_1.
- $P_2(t)$ mit $0 \leq t \leq n_2 - 1$ ist ein Punkt auf P_2.



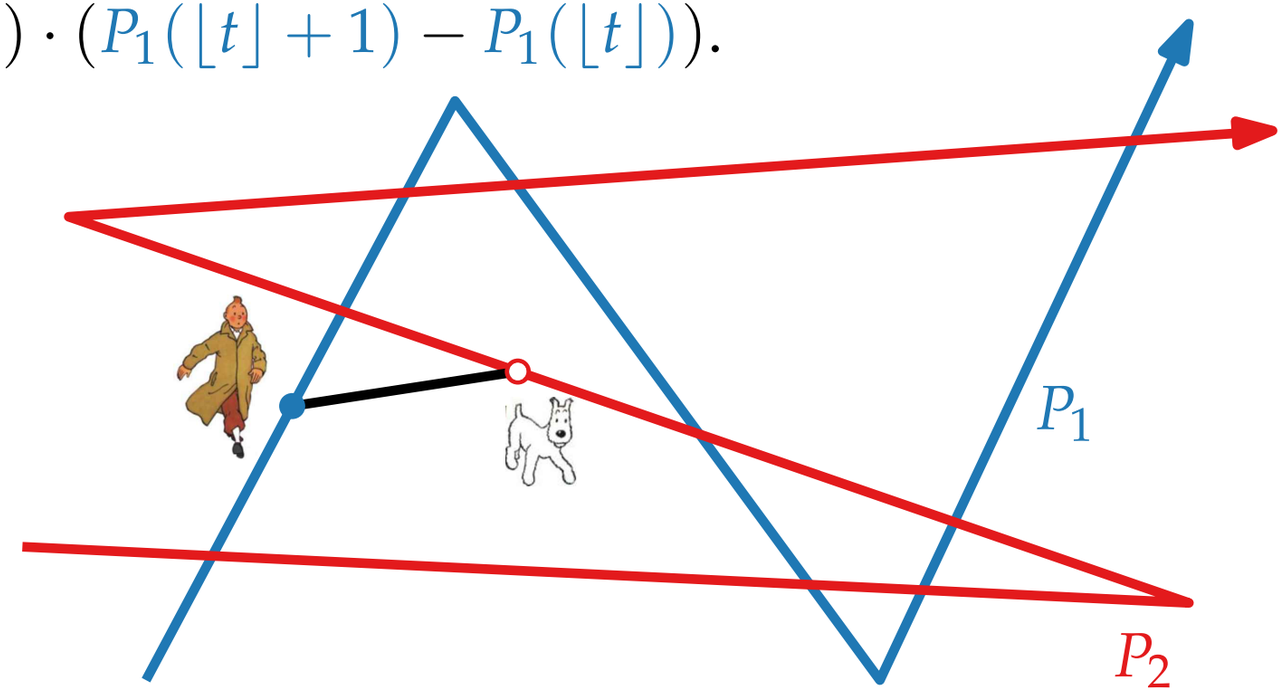
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.
- $P_1(t)$ mit $0 \leq t \leq n_1 - 1$ ist ein Punkt auf P_1.
- $P_2(t)$ mit $0 \leq t \leq n_2 - 1$ ist ein Punkt auf P_2.
- Für ganzzahlige s ist $P_1(t)$ ($P_2(t)$) die $(t+1)$-te Ecke von P_1 (P_2).



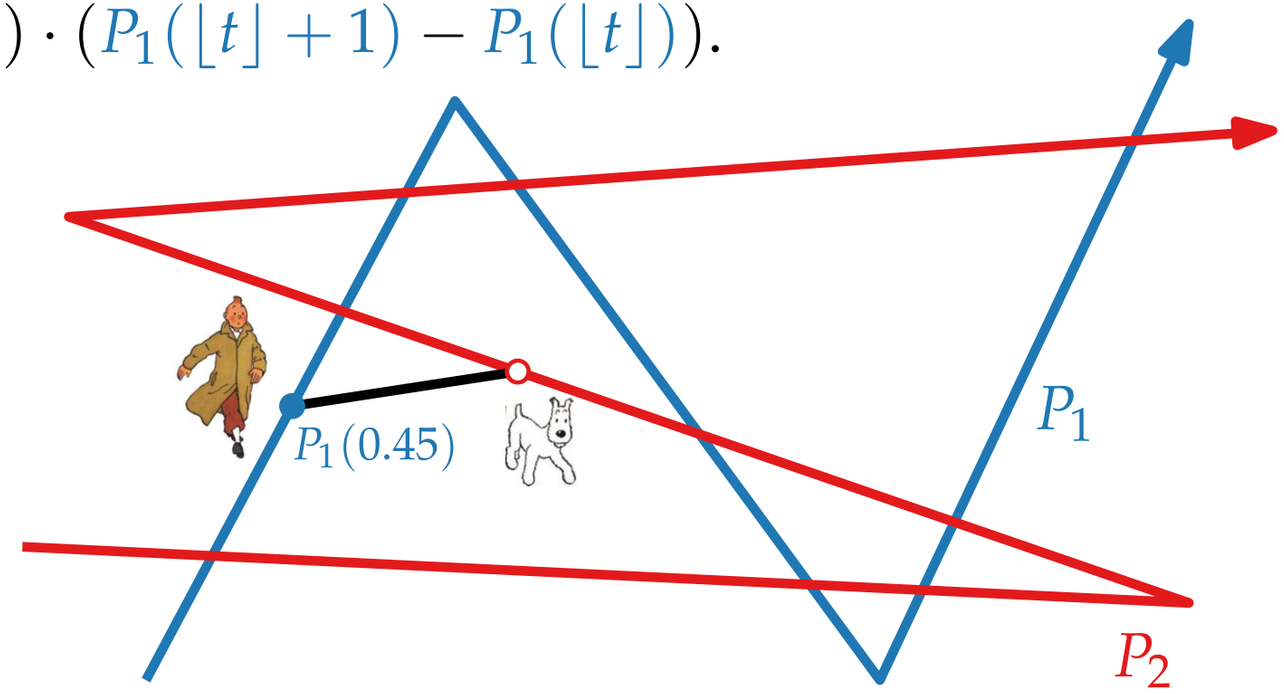
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.
- $P_1(t)$ mit $0 \leq t \leq n_1 - 1$ ist ein Punkt auf P_1.
- $P_2(t)$ mit $0 \leq t \leq n_2 - 1$ ist ein Punkt auf P_2.
- Für ganzzahlige s ist $P_1(t)$ ($P_2(t)$) die $(t+1)$-te Ecke von P_1 (P_2).
- Sonst ist $P_1(t) = P_1(\lfloor t \rfloor) + (t - \lfloor t \rfloor) \cdot (P_1(\lfloor t \rfloor + 1) - P_1(\lfloor t \rfloor))$.



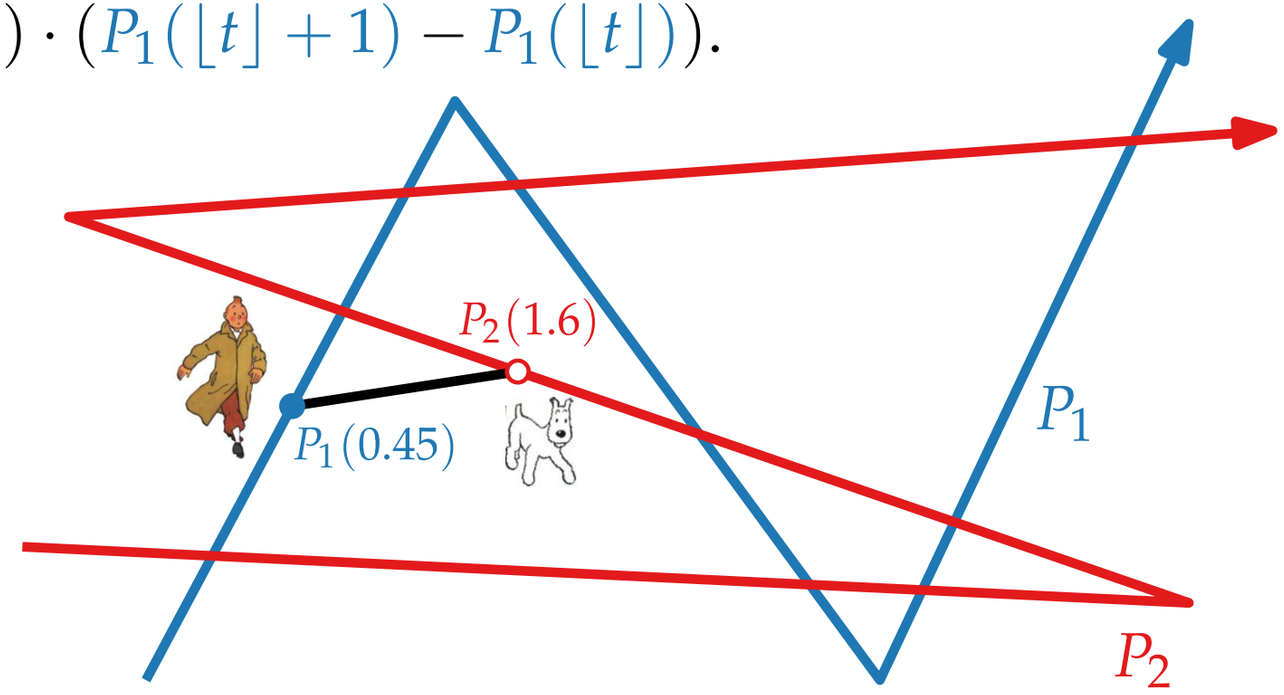
Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.
- $P_1(t)$ mit $0 \leq t \leq n_1 - 1$ ist ein Punkt auf P_1.
- $P_2(t)$ mit $0 \leq t \leq n_2 - 1$ ist ein Punkt auf P_2.
- Für ganzzahlige s ist $P_1(t)$ ($P_2(t)$) die $(t+1)$-te Ecke von P_1 (P_2).
- Sonst ist $P_1(t) = P_1(\lfloor t \rfloor) + (t - \lfloor t \rfloor) \cdot (P_1(\lfloor t \rfloor + 1) - P_1(\lfloor t \rfloor))$.

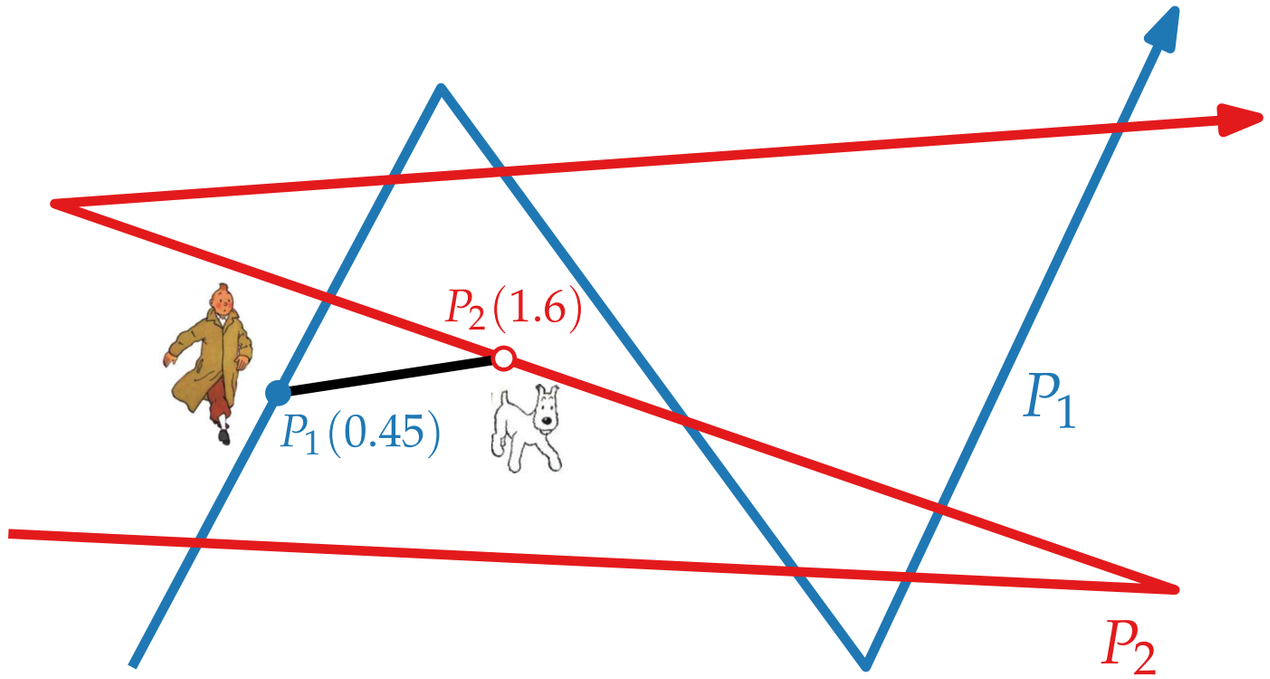


Parametrisierte Kurve

- Stelle Bezug zwischen **Zeitpunkt** und **Punkten auf Linienzügen** her.
- Beide bewegen sich von Zeitpunkt 0 bis Zeitpunkt 1.
- $P_1(t)$ mit $0 \leq t \leq n_1 - 1$ ist ein Punkt auf P_1.
- $P_2(t)$ mit $0 \leq t \leq n_2 - 1$ ist ein Punkt auf P_2.
- Für ganzzahlige s ist $P_1(t)$ ($P_2(t)$) die $(t+1)$-te Ecke von P_1 (P_2).
- Sonst ist $P_1(t) = P_1(\lfloor t \rfloor) + (t - \lfloor t \rfloor) \cdot (P_1(\lfloor t \rfloor + 1) - P_1(\lfloor t \rfloor))$.

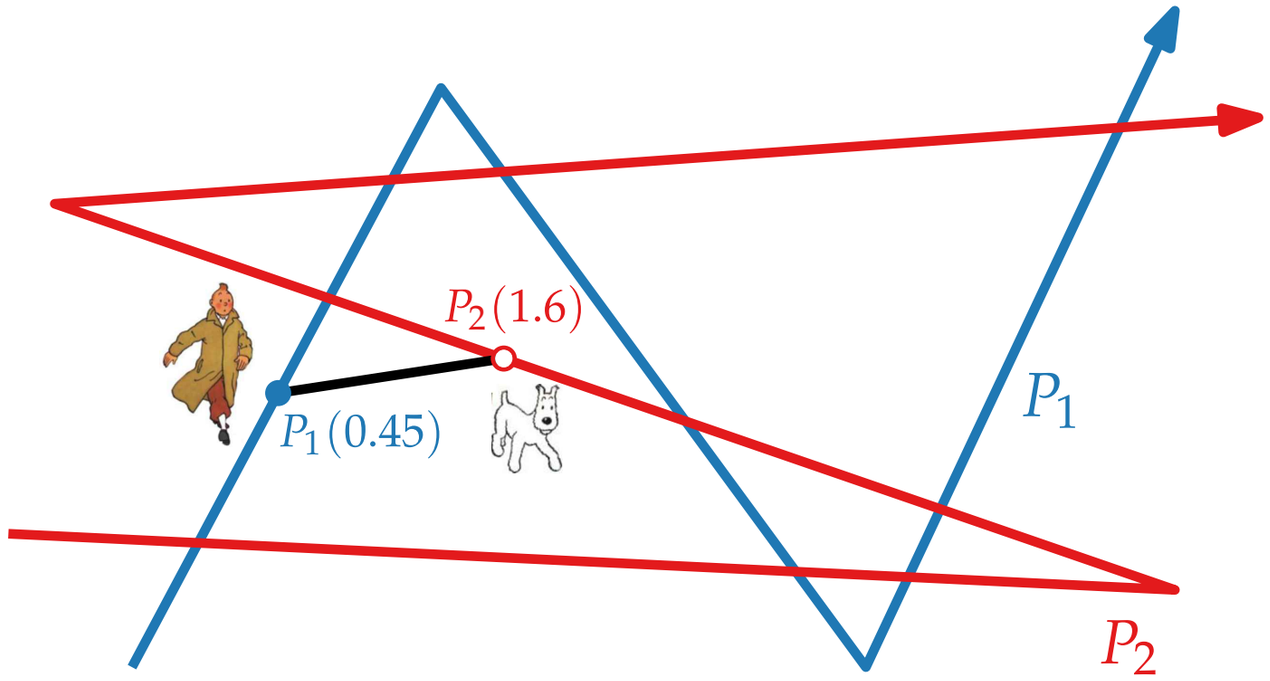


Fréchet Distance – Definition



Fréchet Distance – Definition

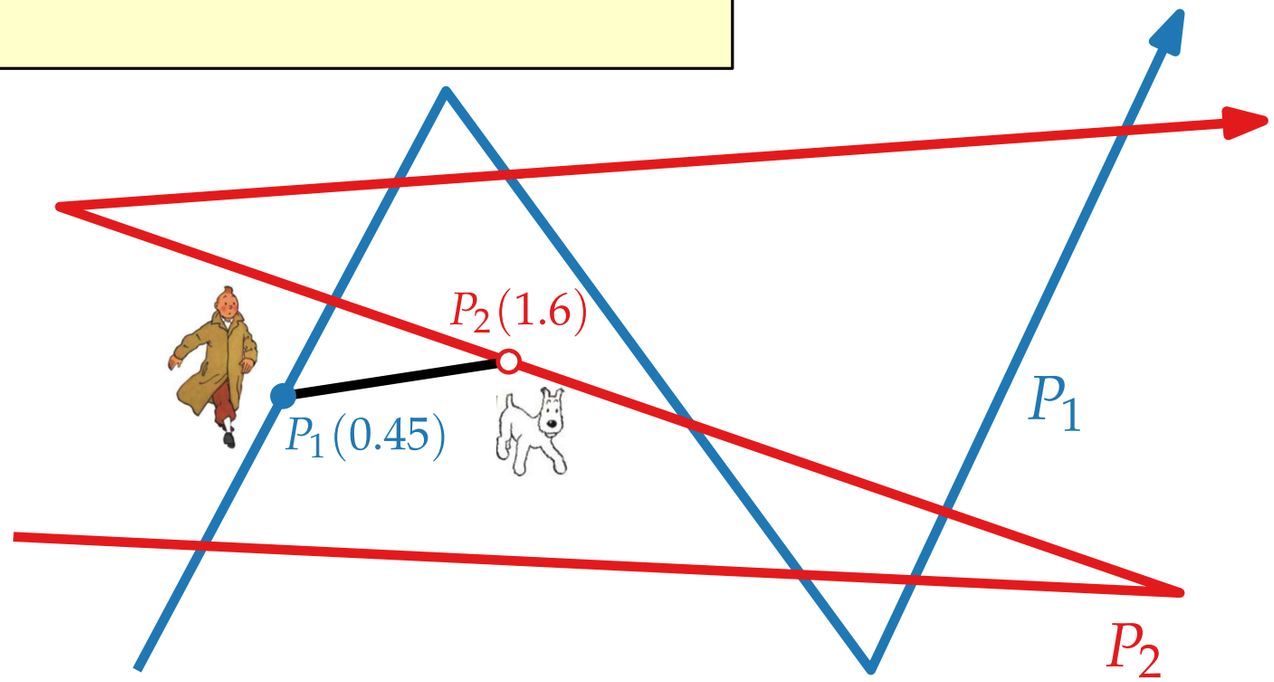
- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.



Fréchet Distance – Definition

- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.

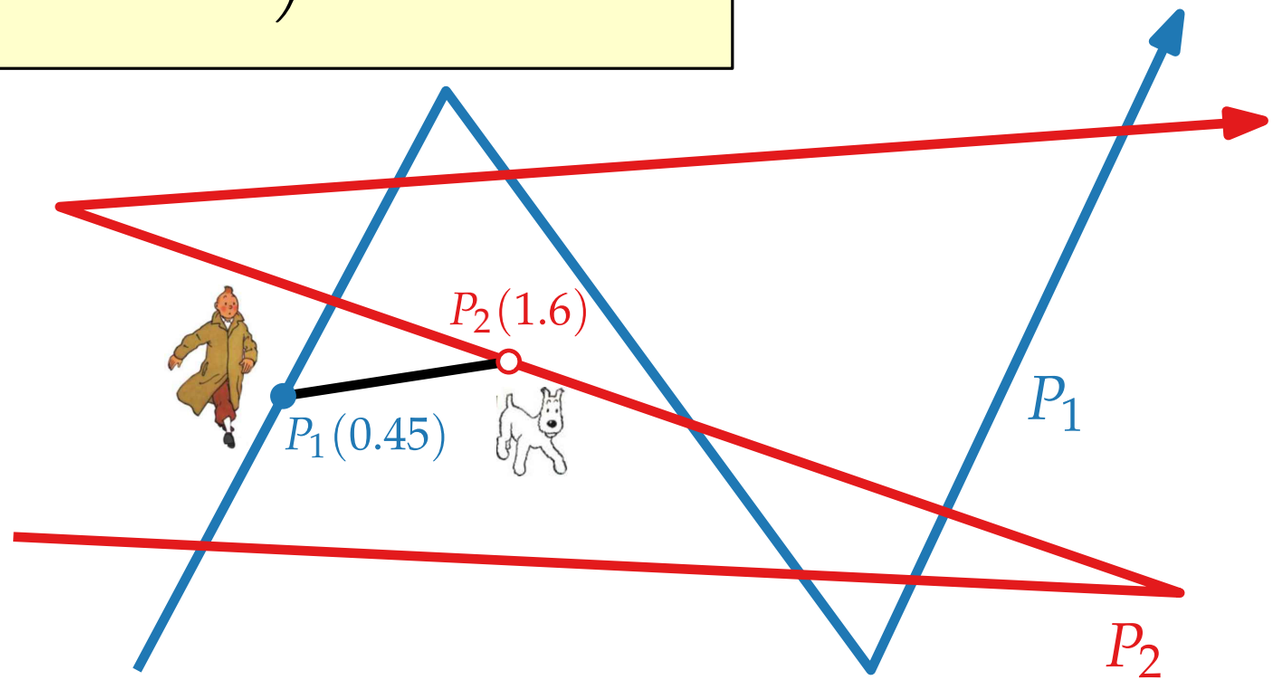
➡ $d_{\mathrm{F}}(P_1, P_2) =$



Fréchet Distance – Definition

- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.

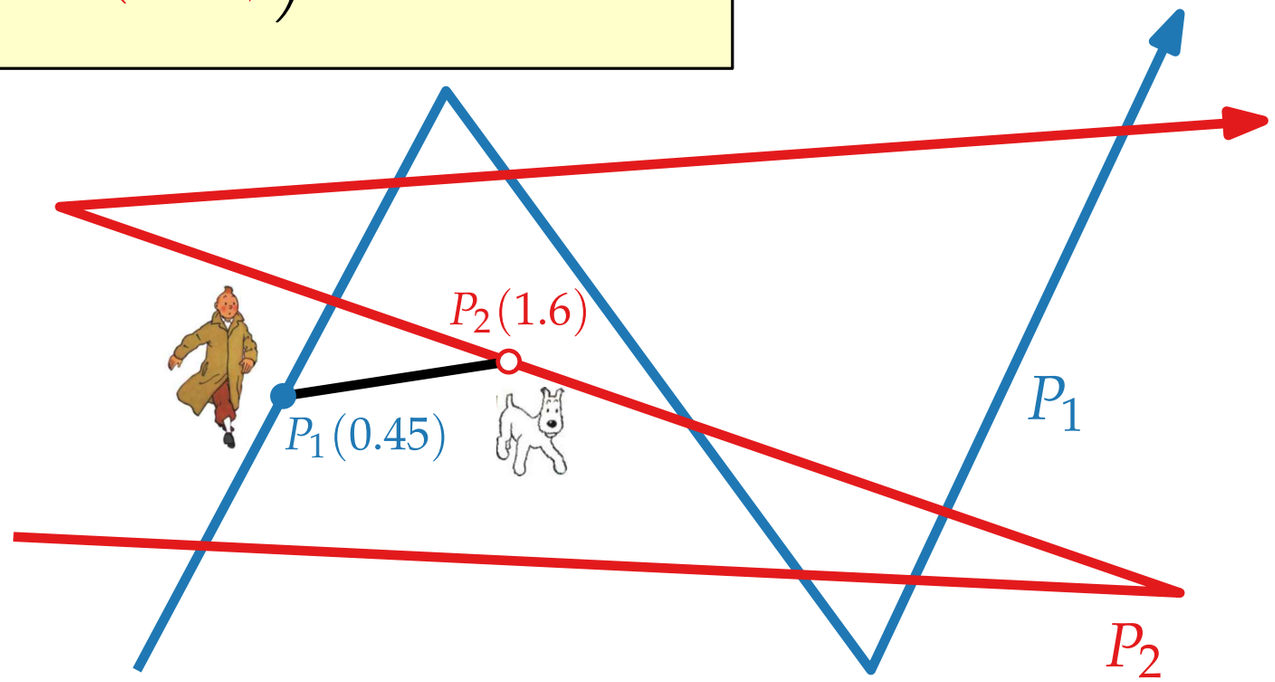
➡ $d_F(P_1, P_2) = d\Big($ $\Big)$



Fréchet Distance – Definition

- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.

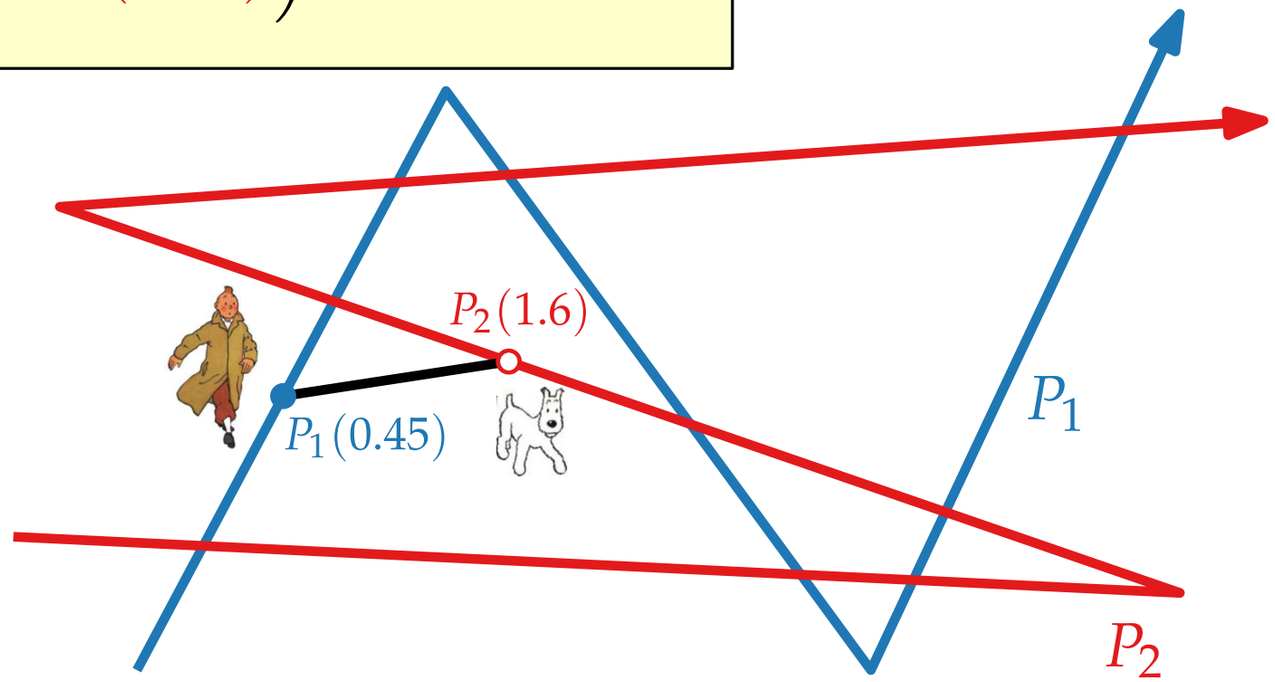
➡ $d_F(P_1, P_2) =$ $d\Big(P_1(\quad), P_2(\quad)\Big)$



Fréchet Distance – Definition

- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.

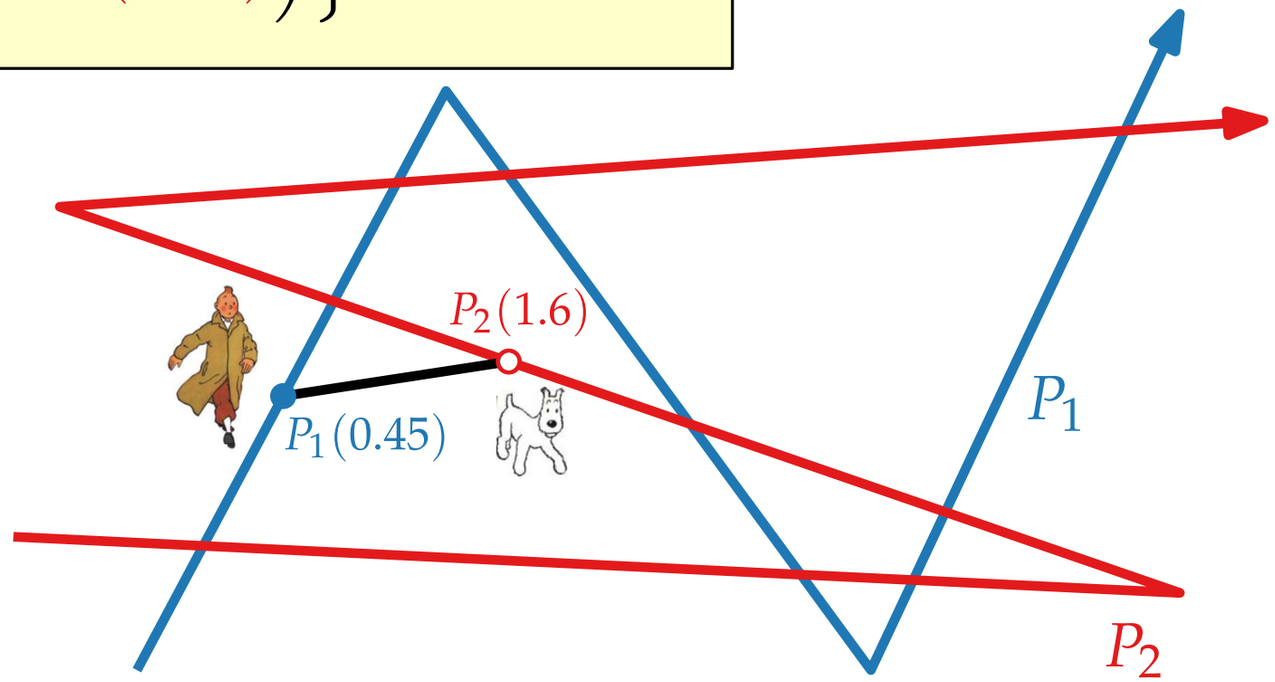
➡ $d_{\mathrm{F}}(P_1, P_2) = \qquad d\Big(P_1(\alpha(t)), P_2(\beta(t))\Big)$



Fréchet Distance – Definition

- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.

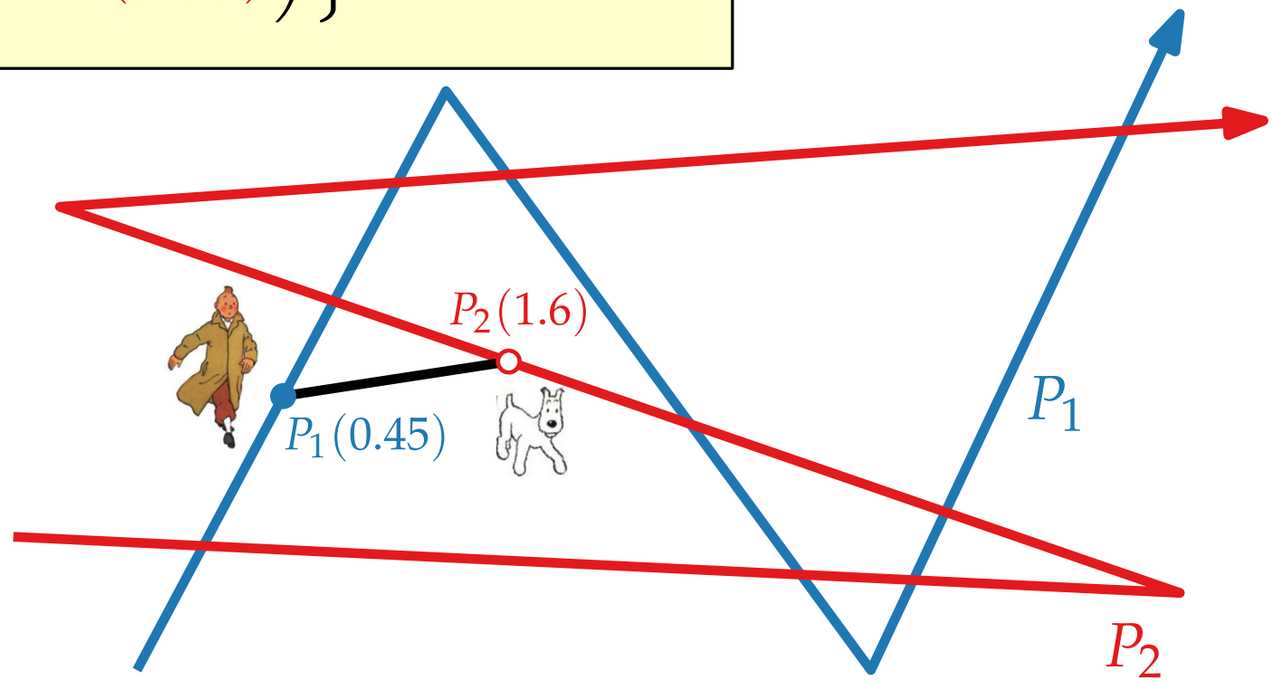
➡ $d_F(P_1, P_2) = \max_{t \in [0,1]} \left\{ d\left(P_1(\alpha(t)), P_2(\beta(t)) \right) \right\}$



Fréchet Distance – Definition

- Seien α : $\rightarrow$ und β : $\rightarrow$ Funktionen.

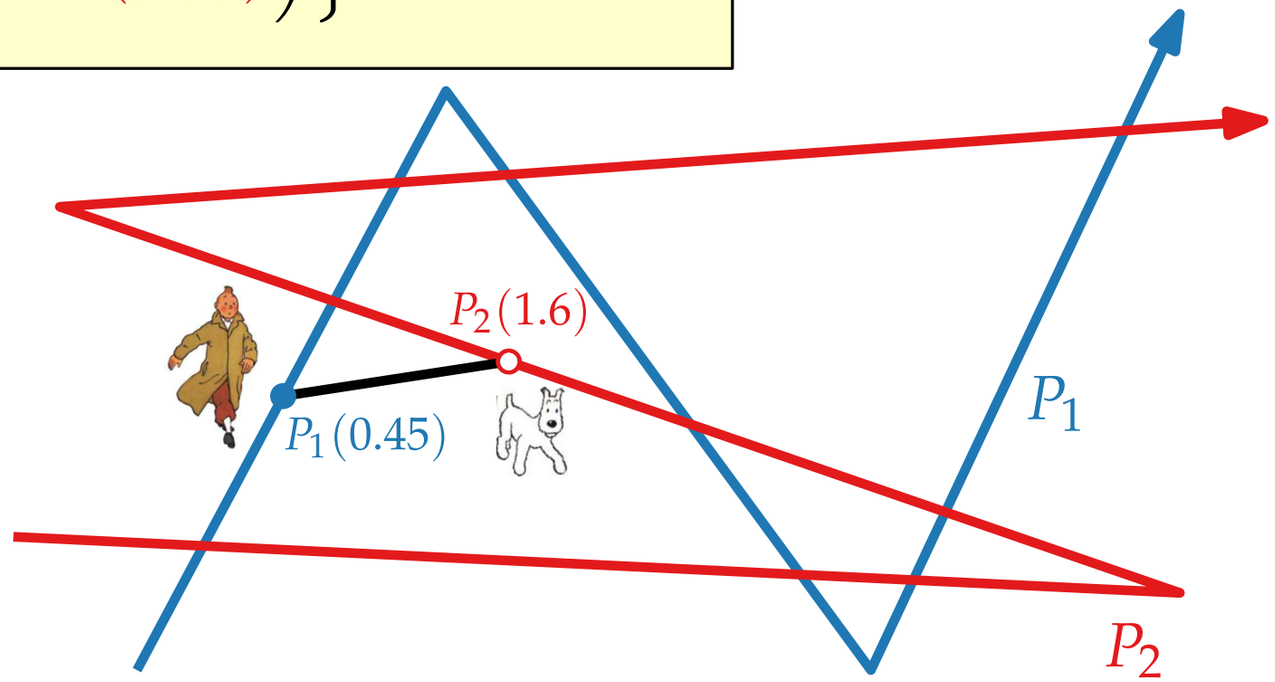
$$\Rightarrow d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \rightarrow$ und $\beta :$ $\rightarrow$ Funktionen.

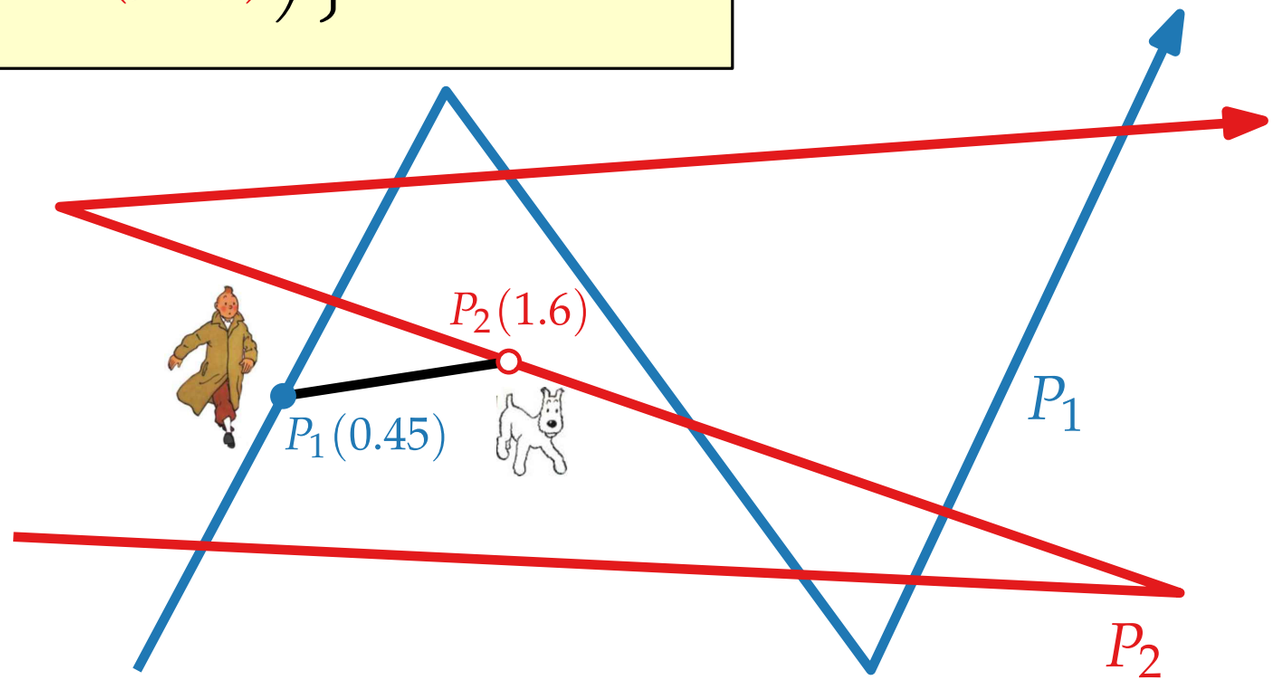
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \rightarrow$ und $\beta : [0,1] \rightarrow$ Funktionen.

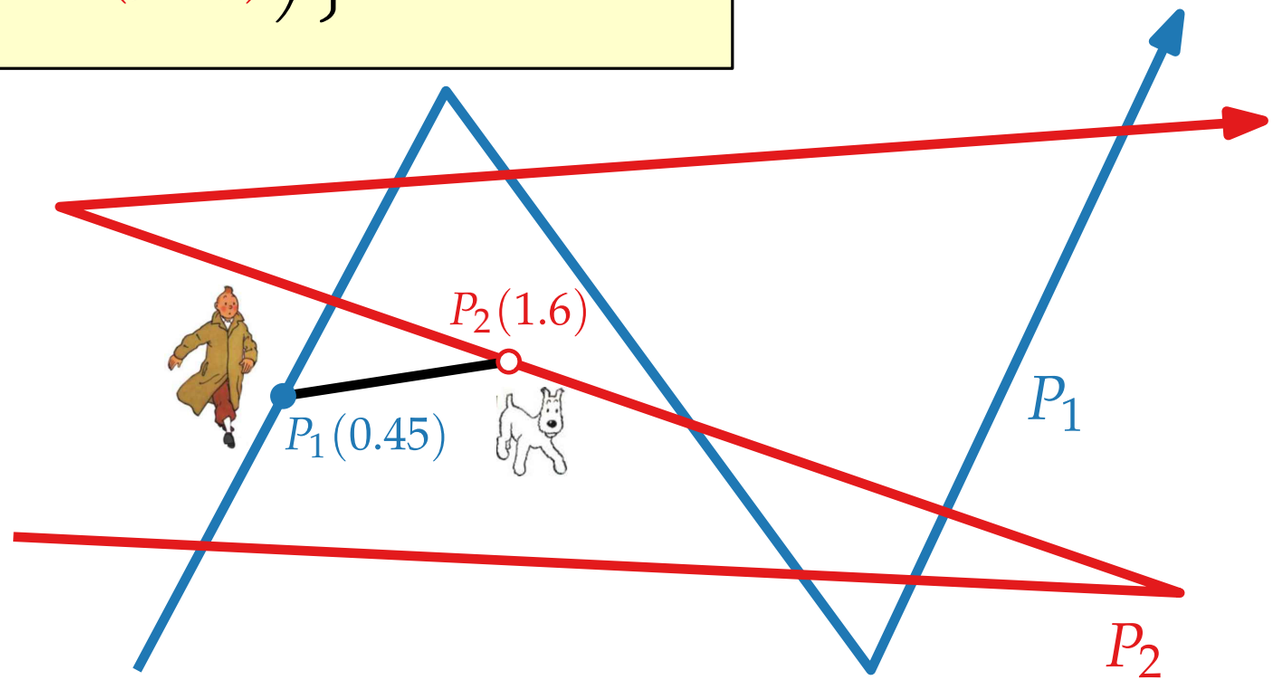
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to$ Funktionen.

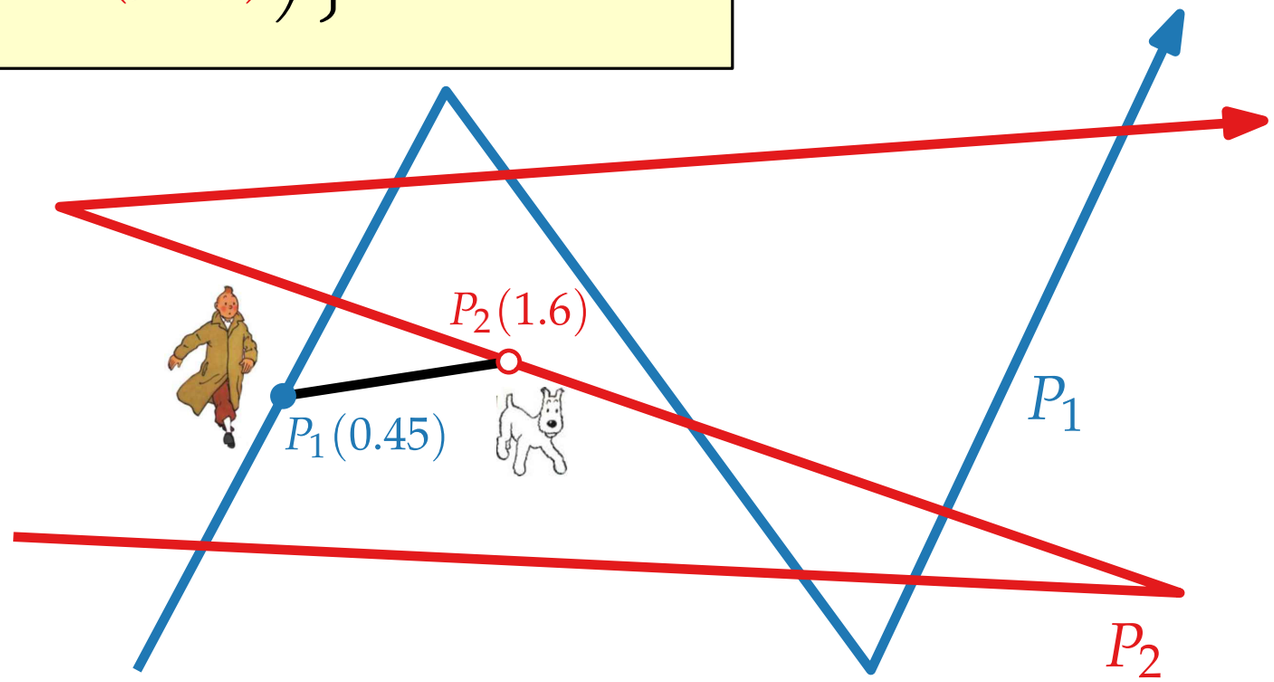
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ Funktionen.

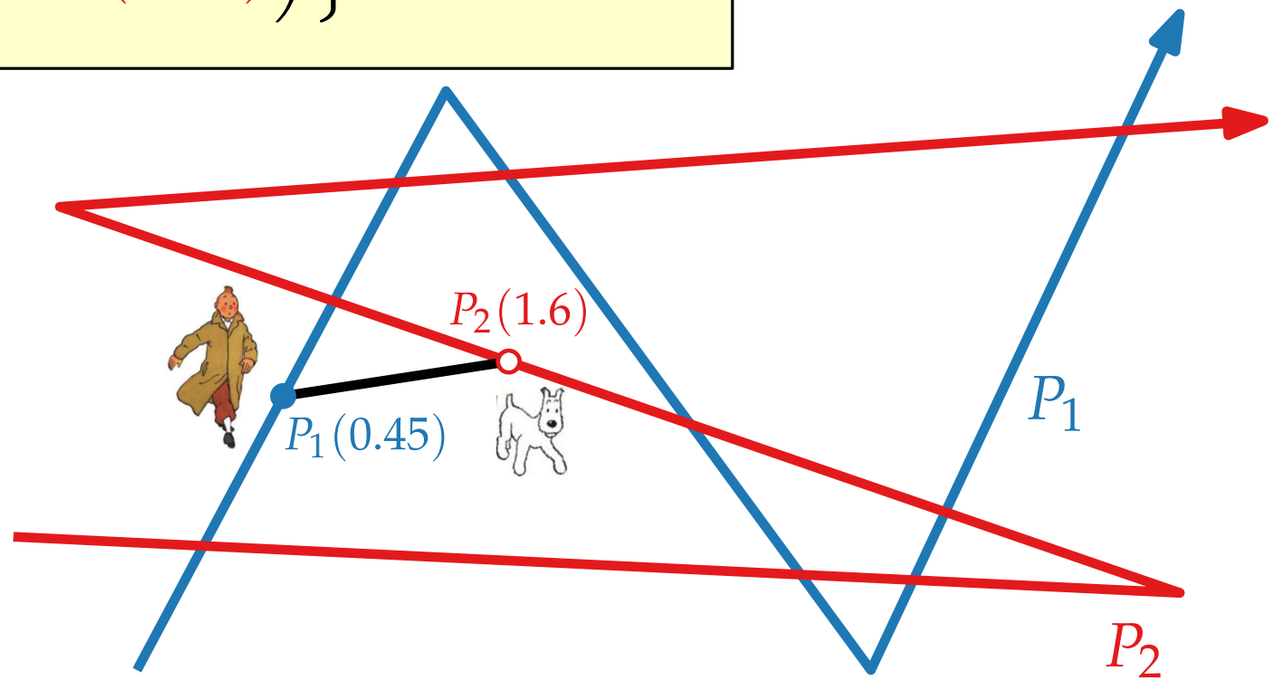
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ **surjektive**, Funktionen.

➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$

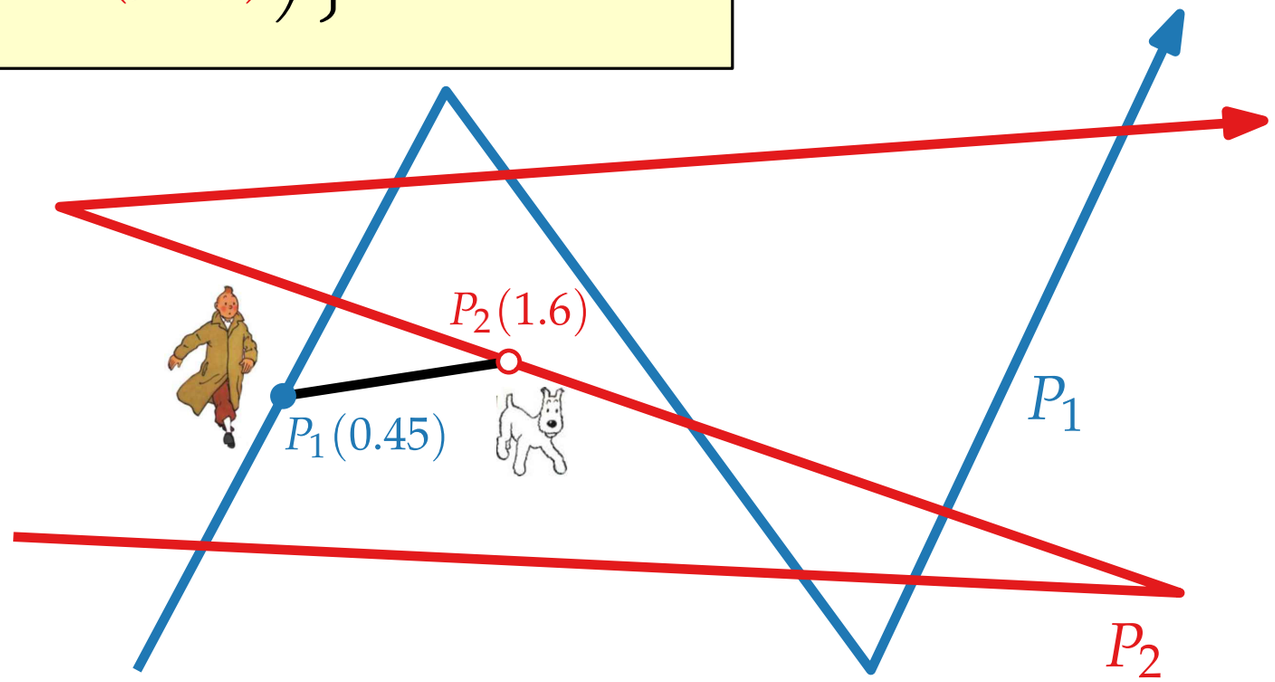


Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ **surjektive**, Funktionen.

  Jeder Zielwert wird angenommen

➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$

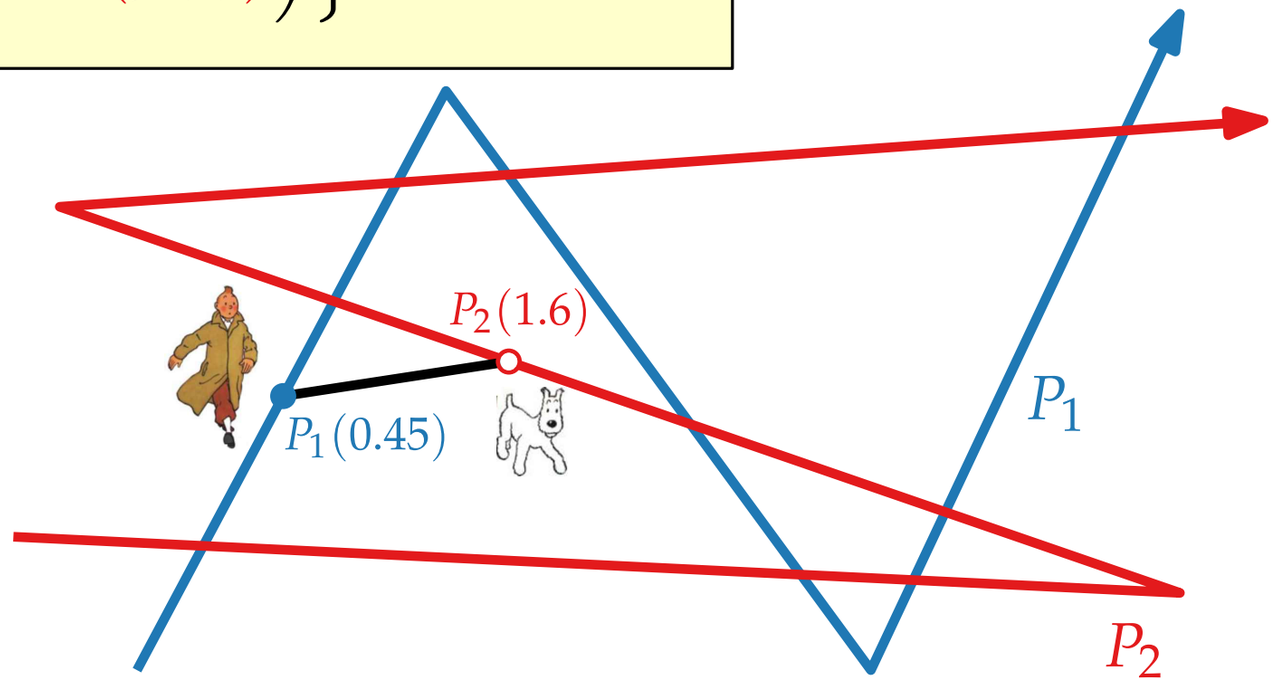


Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ **surjektive**, **monoton wachsende** und Funktionen.

  Jeder Zielwert wird angenommen

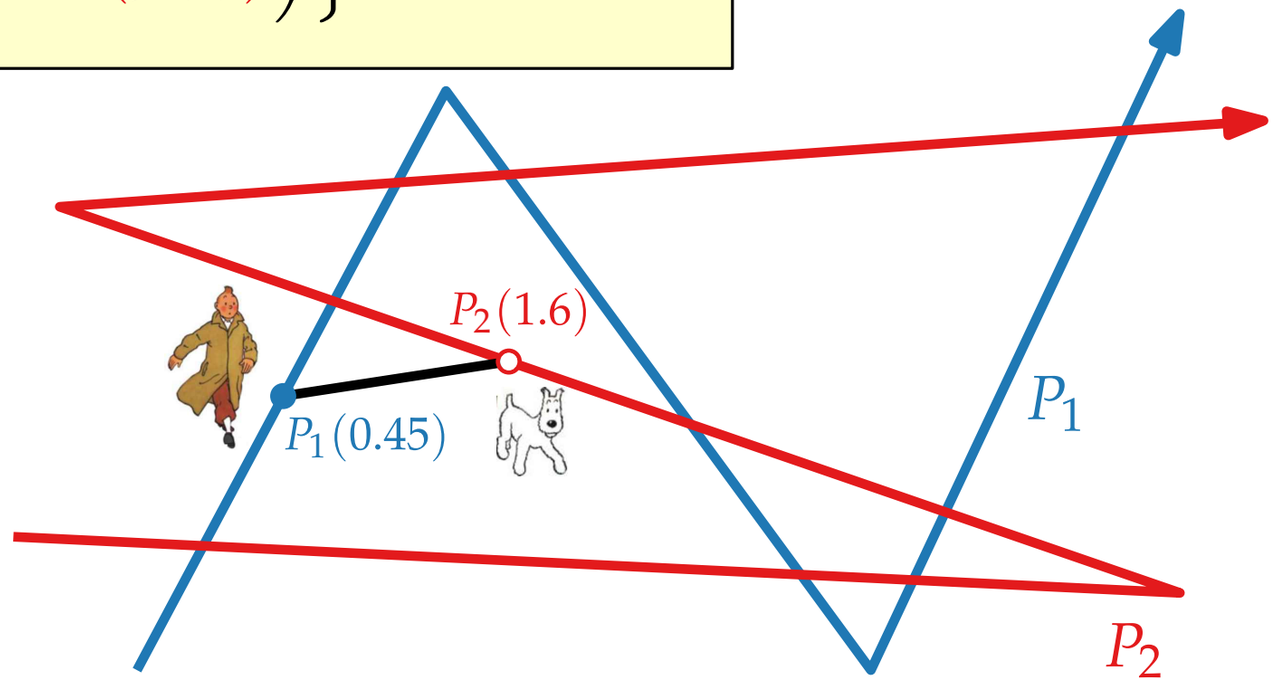
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ **surjektive**, **monoton wachsende** und Funktionen.
  - Jeder Zielwert wird angenommen
  - gehen nicht zurück

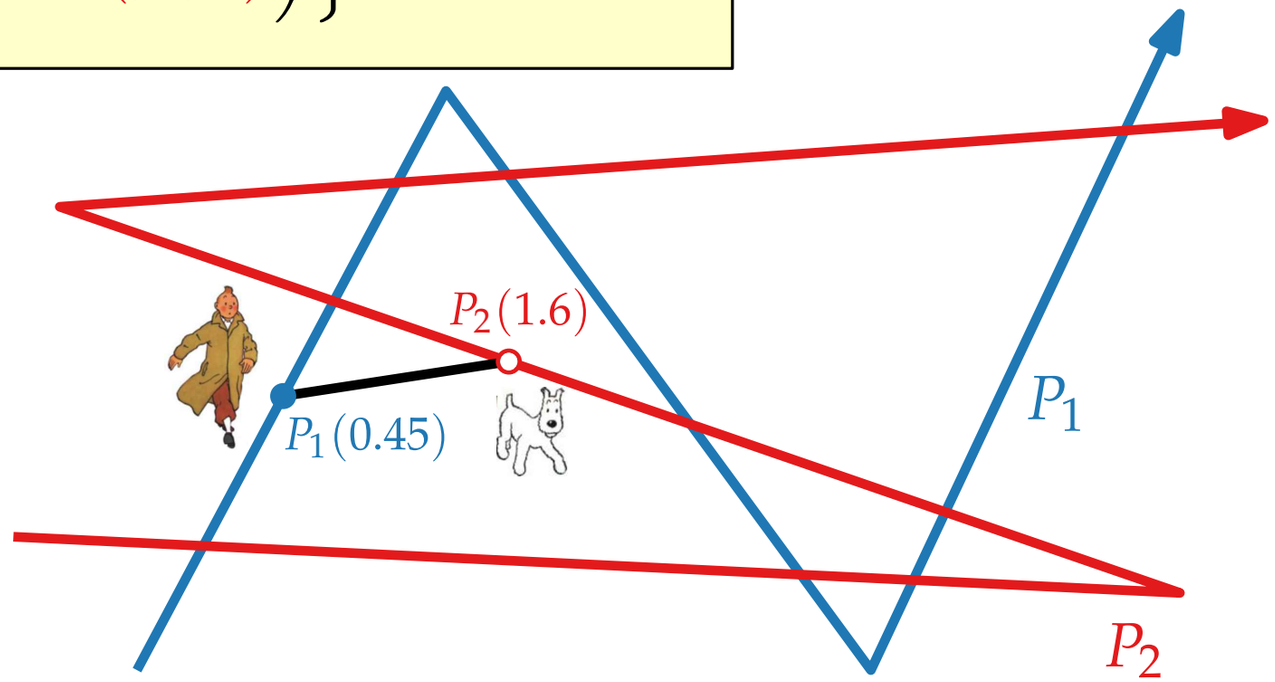
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ **surjektive**, **monoton wachsende** und **stetige** Funktionen.
  - Jeder Zielwert wird angenommen
  - gehen nicht zurück

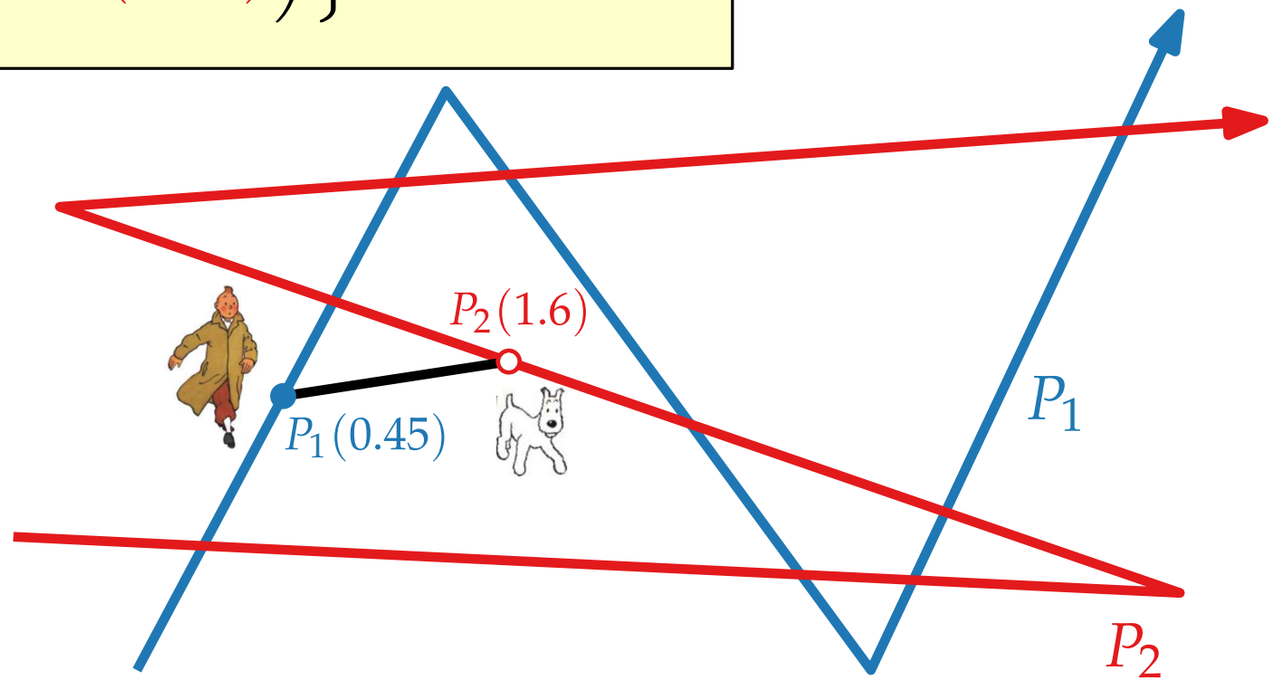
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distance – Definition

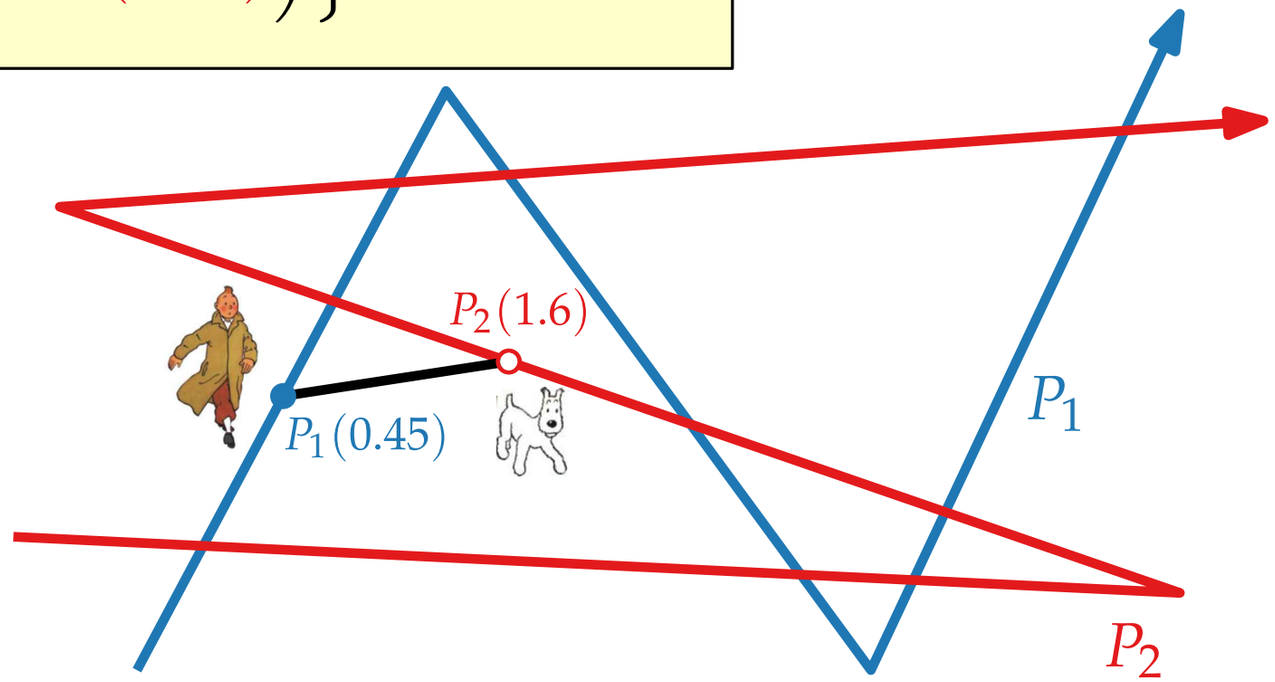
- Seien $\alpha : [0,1] \to [0, n_1 - 1]$ und $\beta : [0,1] \to [0, n_2 - 1]$ **surjektive**, **monoton wachsende** und **stetige** Funktionen.
  - Jeder Zielwert wird angenommen
  - gehen nicht zurück
  - teleportieren sich nicht

➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distanz – Berechnung

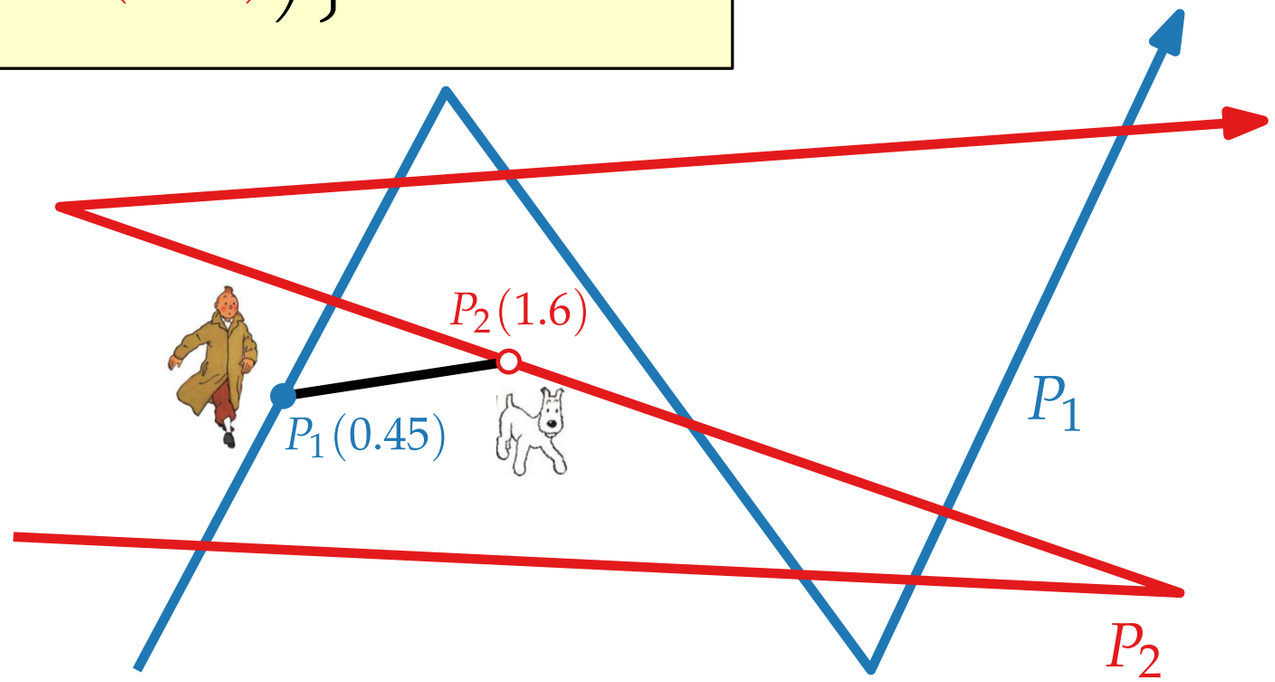
➡ $d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t \in [0,1]} \left\{ d\left(P_1(\alpha(t)), P_2(\beta(t)) \right) \right\}$



Fréchet Distanz – Berechnung

- Löse Entscheidungsproblem:

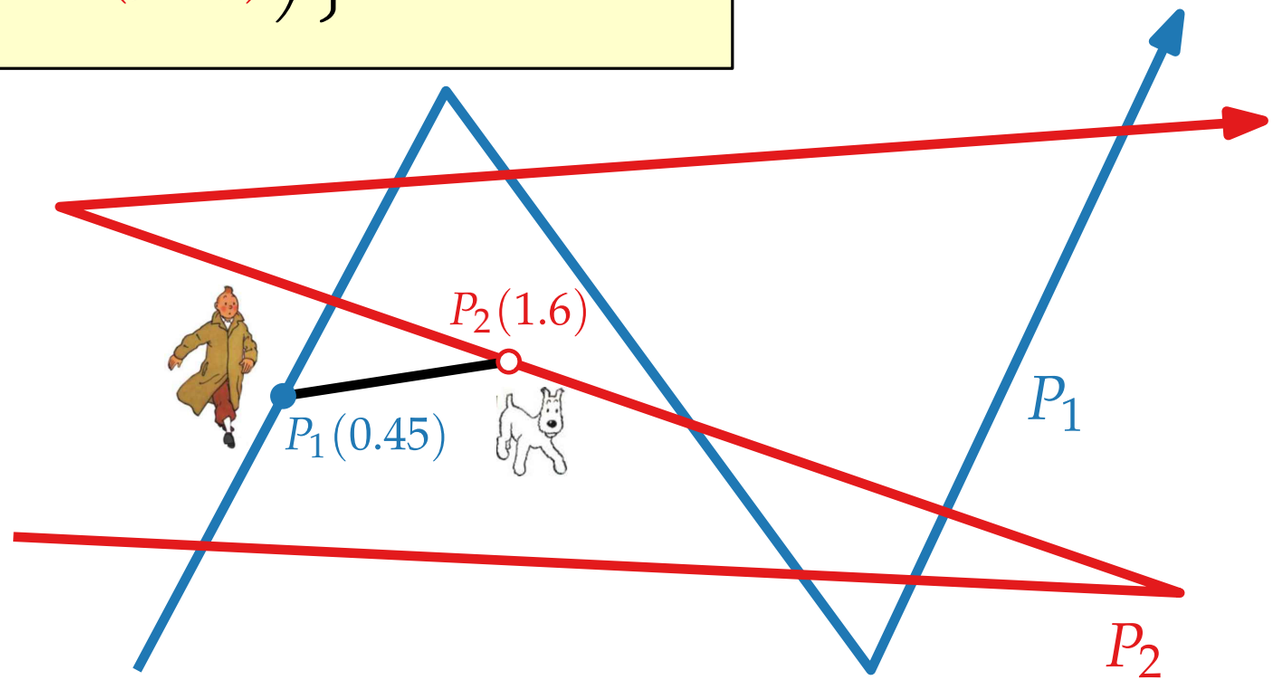
➡ $d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d\left(P_1(\alpha(t)), P_2(\beta(t)) \right) \right\}$



Fréchet Distanz – Berechnung

- Löse Entscheidungsproblem: Gegeben $\varepsilon \in \mathbb{R}$: Ist $d_F(P_1, P_2) \leq \varepsilon$?

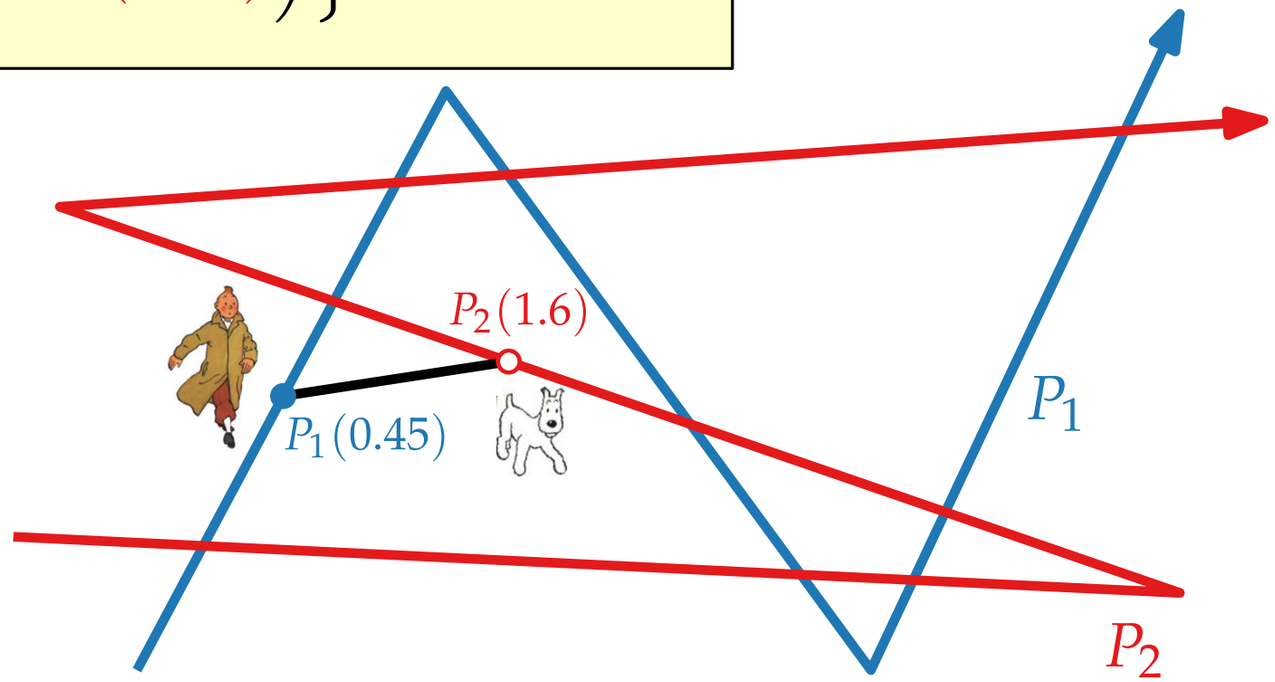
➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



Fréchet Distanz – Berechnung

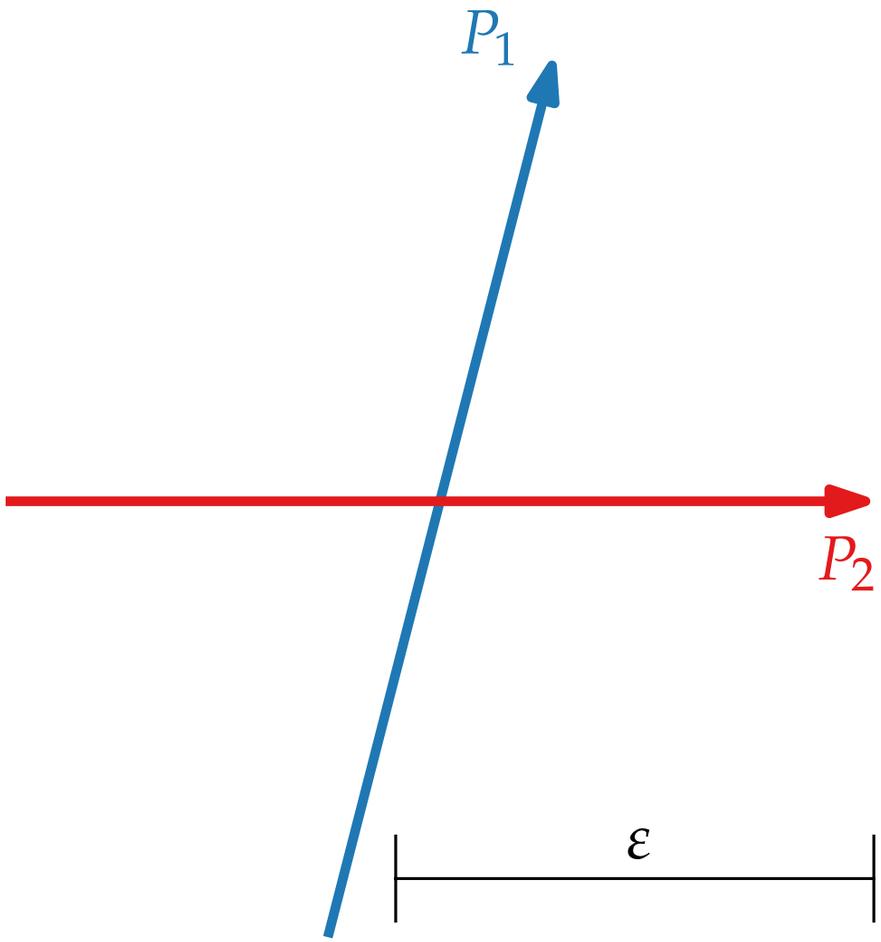
- Löse Entscheidungsproblem: Gegeben $\varepsilon \in \mathbb{R}$: Ist $d_F(P_1, P_2) \leq \varepsilon$?
- Verwende parametrisierte Suche (ähnlich wie binäre Suche) um kleinstes ε zu finden, sodass die Antwort **ja** ist.

➡ $$d_F(P_1, P_2) = \min_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d\Big(P_1(\alpha(t)), P_2(\beta(t)) \Big) \right\}$$



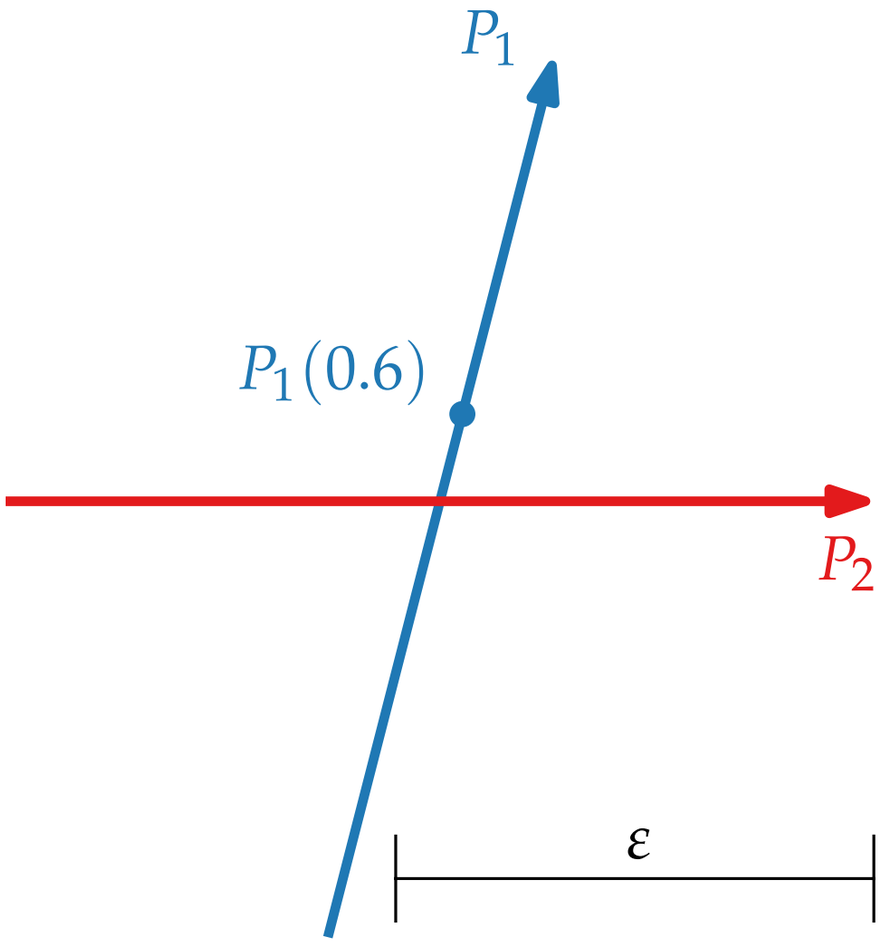
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.



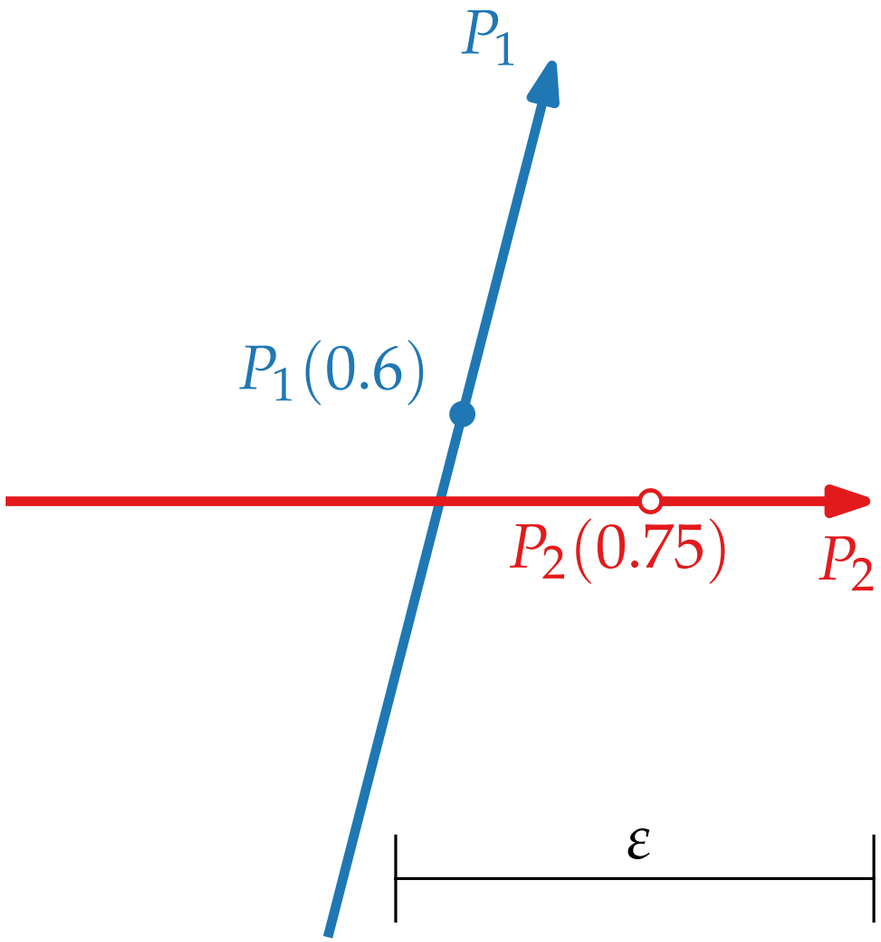
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.



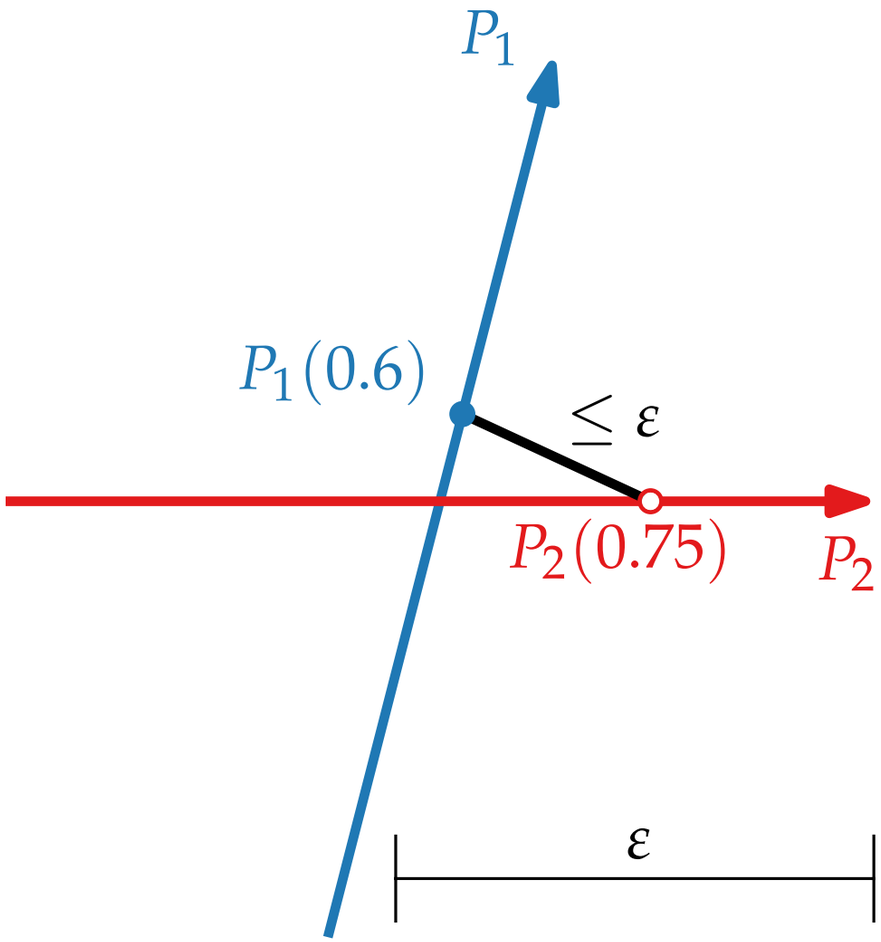
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.



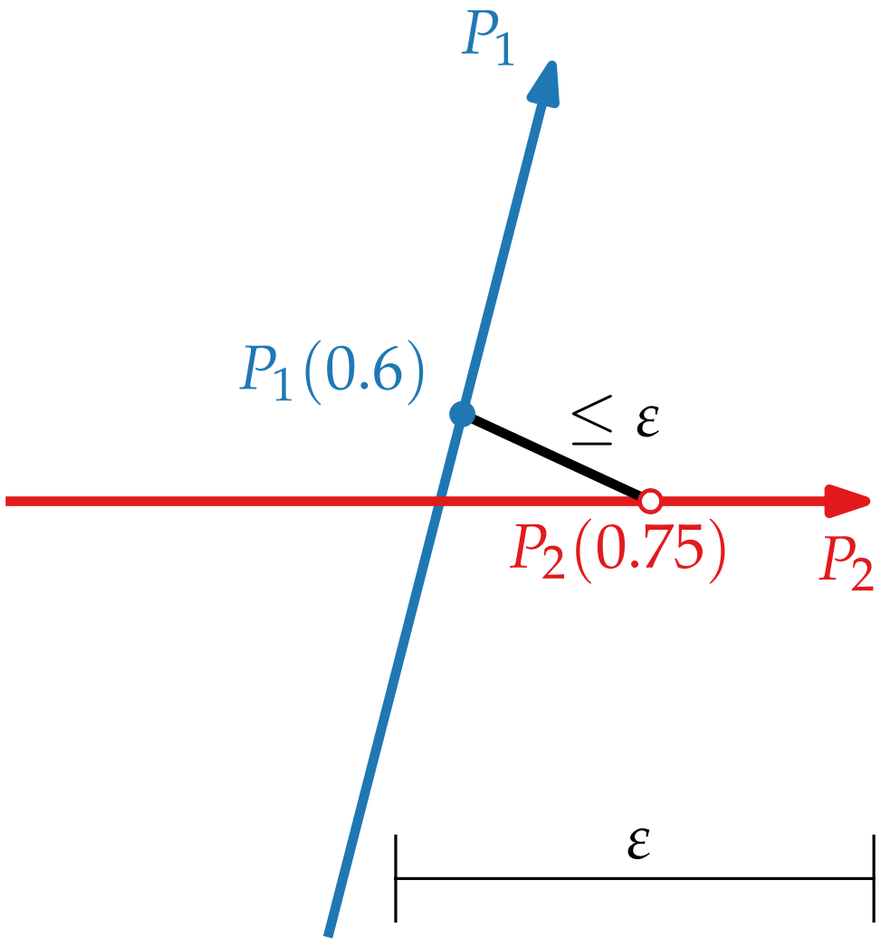
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.



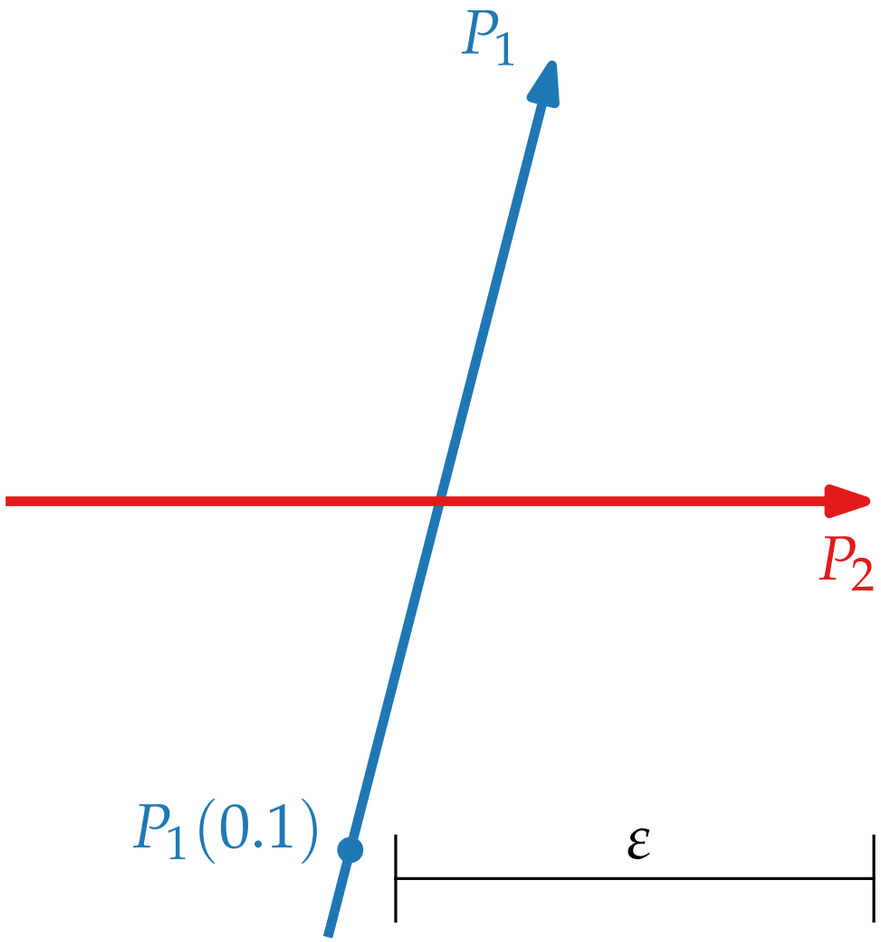
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.



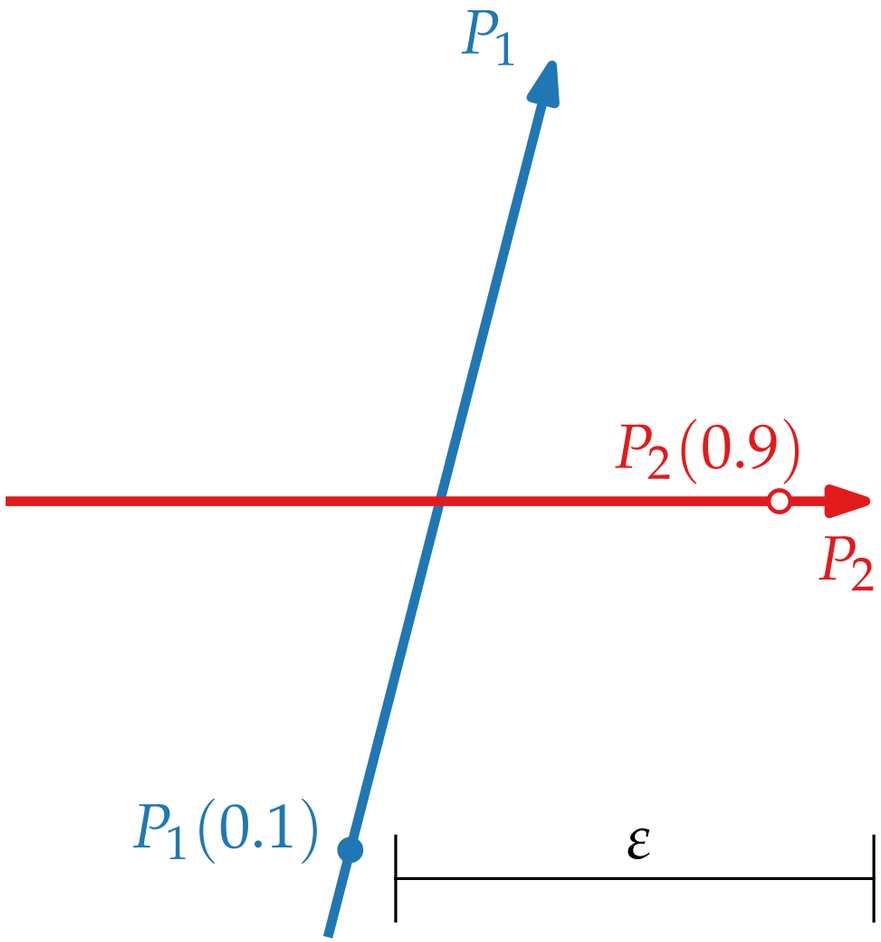
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.



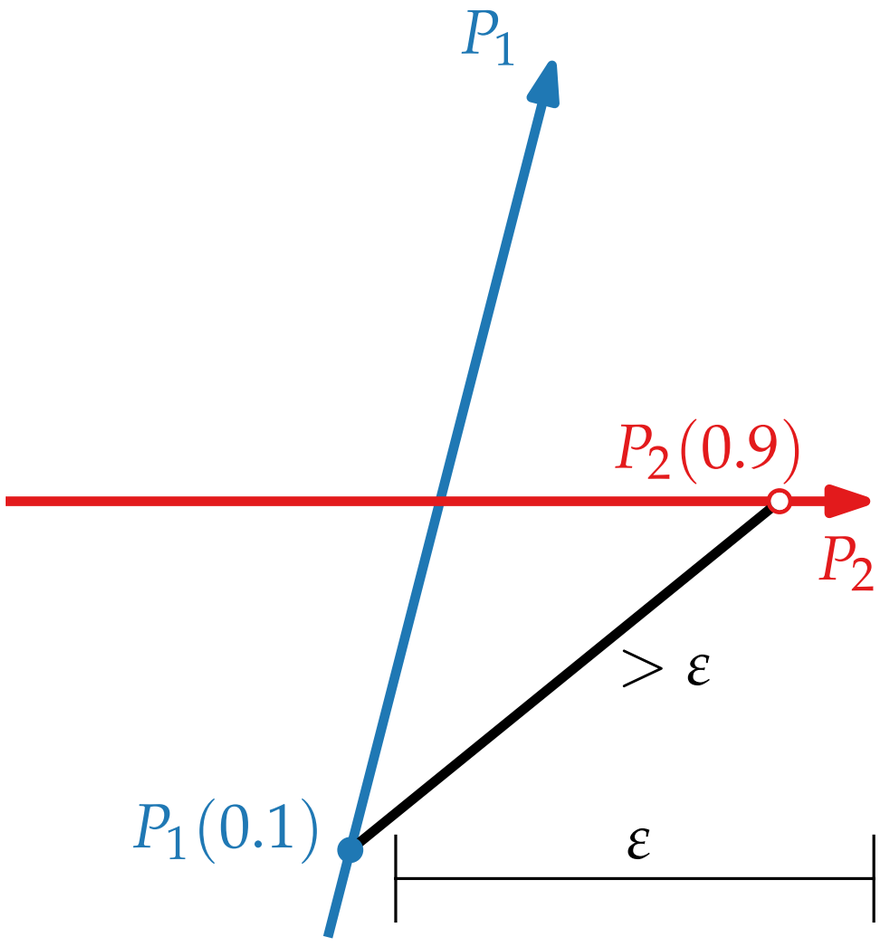
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.



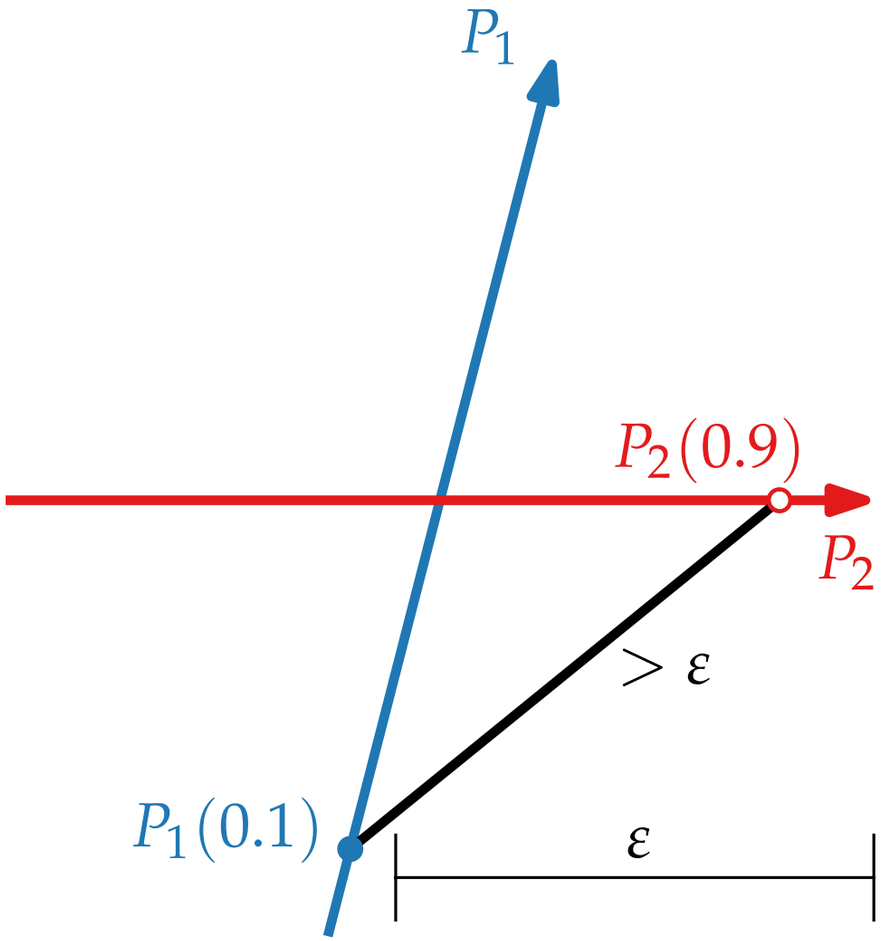
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.



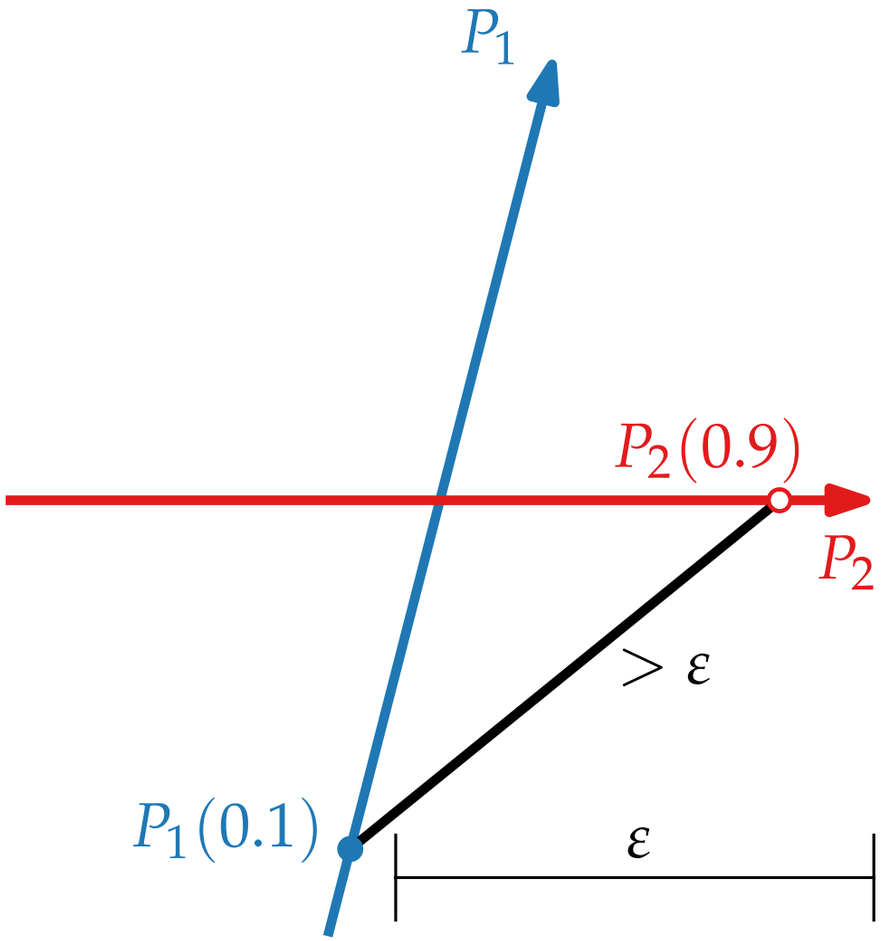
Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.
$P_1(0.1)$ und $P_2(0.9)$ können **nicht** verbunden werden.



Free-Space Diagram – 1 Segment

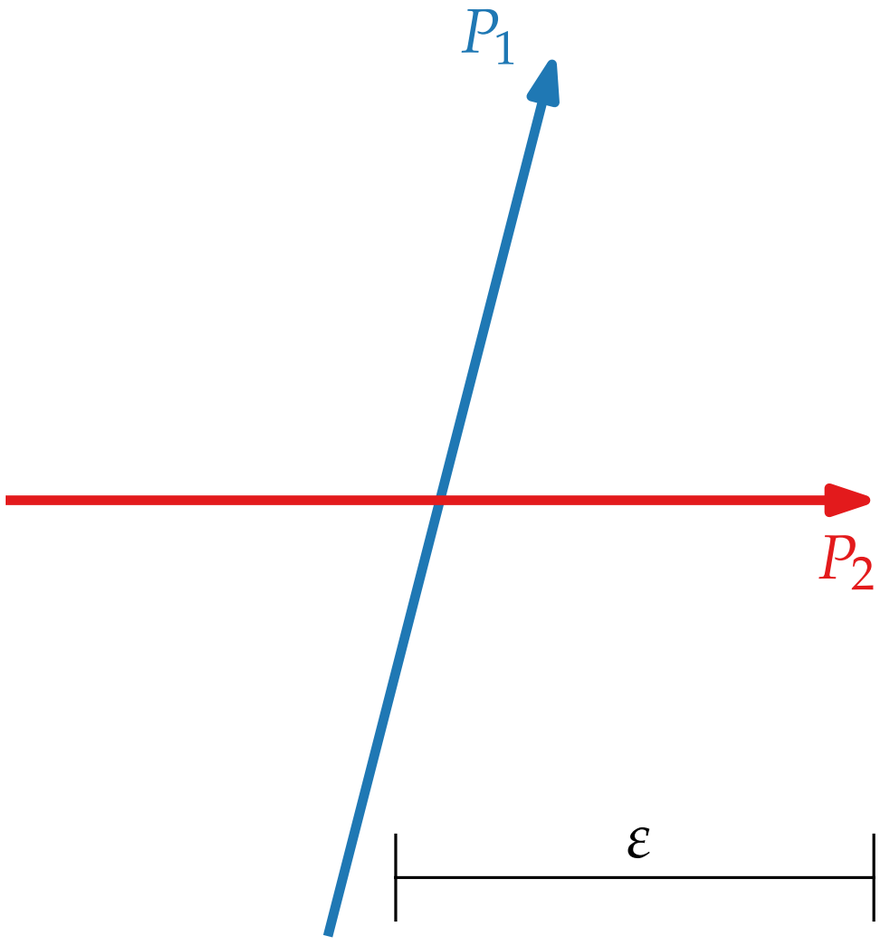
Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.
$P_1(0.1)$ und $P_2(0.9)$ können **nicht** verbunden werden.



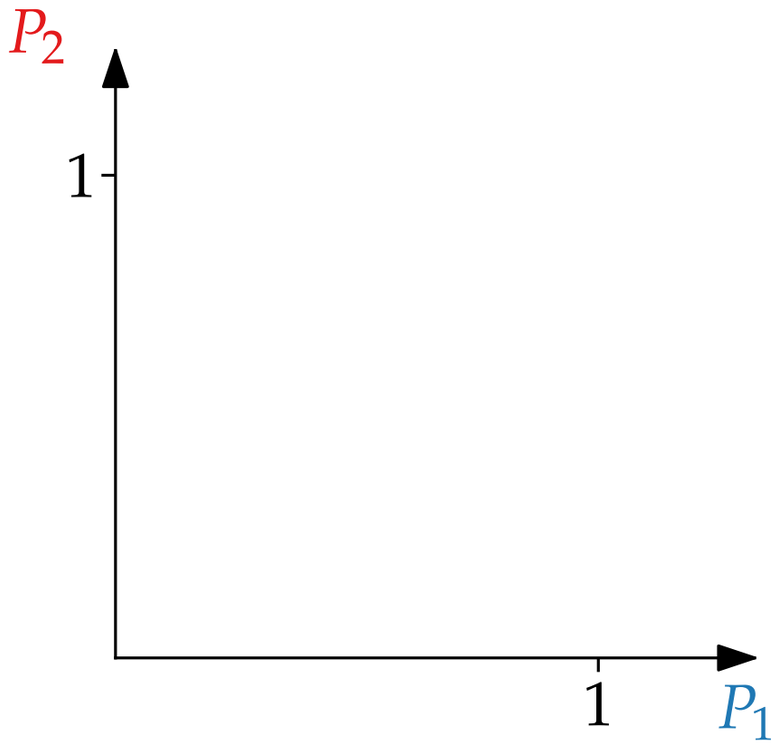
Das **Free-Space Diagram** F_ε zeigt an, welche Punktepaare mit einer Leine der Länge ε verbunden werden können.

Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.
$P_1(0.1)$ und $P_2(0.9)$ können **nicht** verbunden werden.

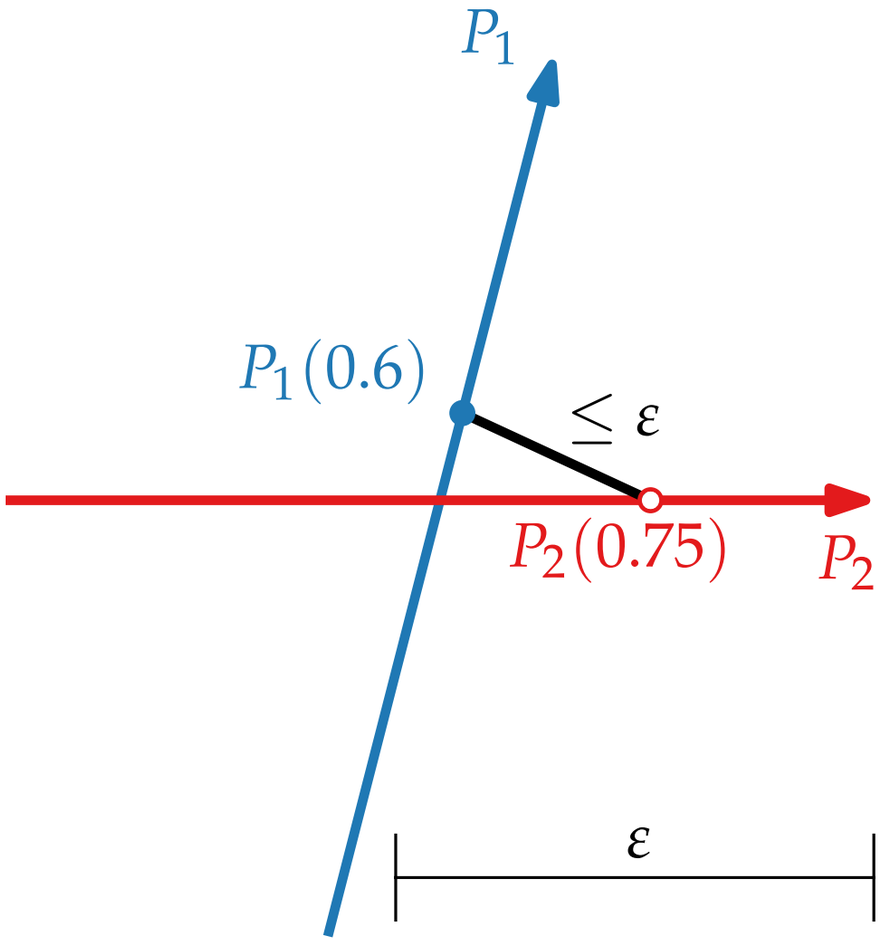


Das **Free-Space Diagram** F_ε zeigt an, welche Punktepaare mit einer Leine der Länge ε verbunden werden können.

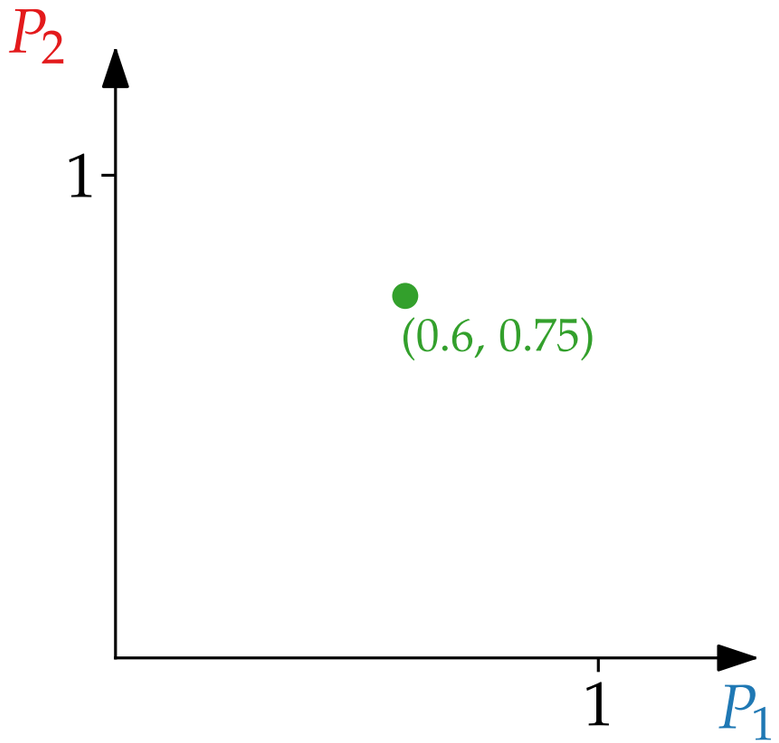


Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.
$P_1(0.1)$ und $P_2(0.9)$ können **nicht** verbunden werden.

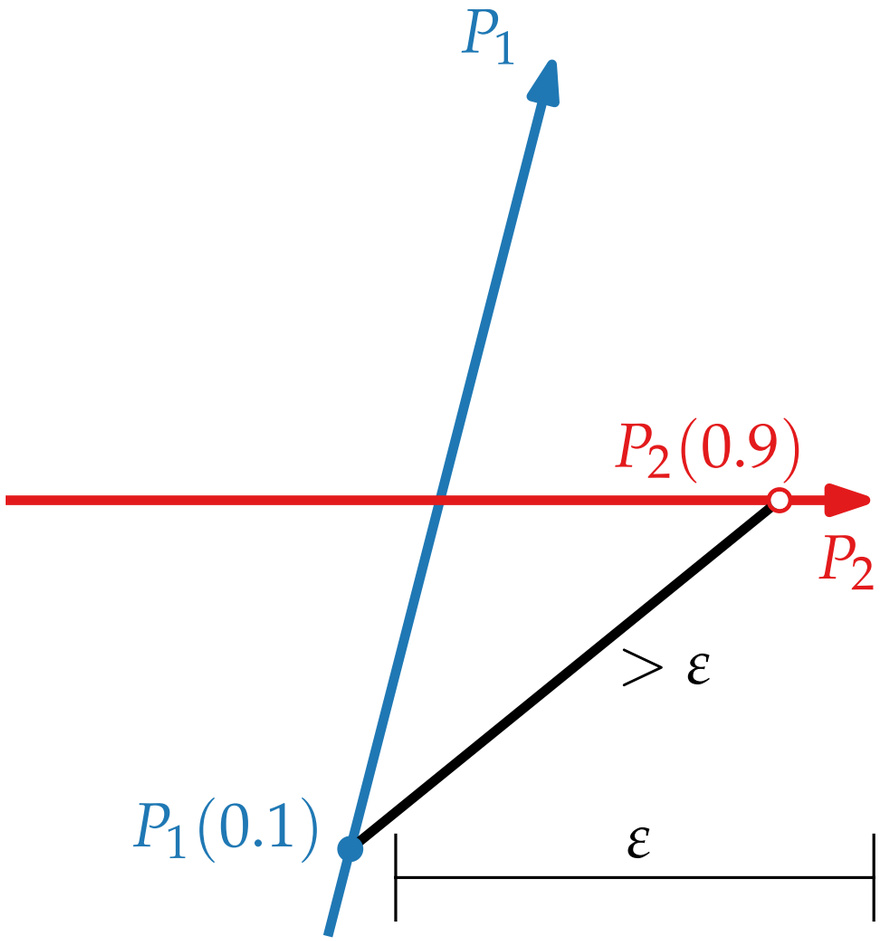


Das **Free-Space Diagram** F_ε zeigt an, welche Punktepaare mit einer Leine der Länge ε verbunden werden können.

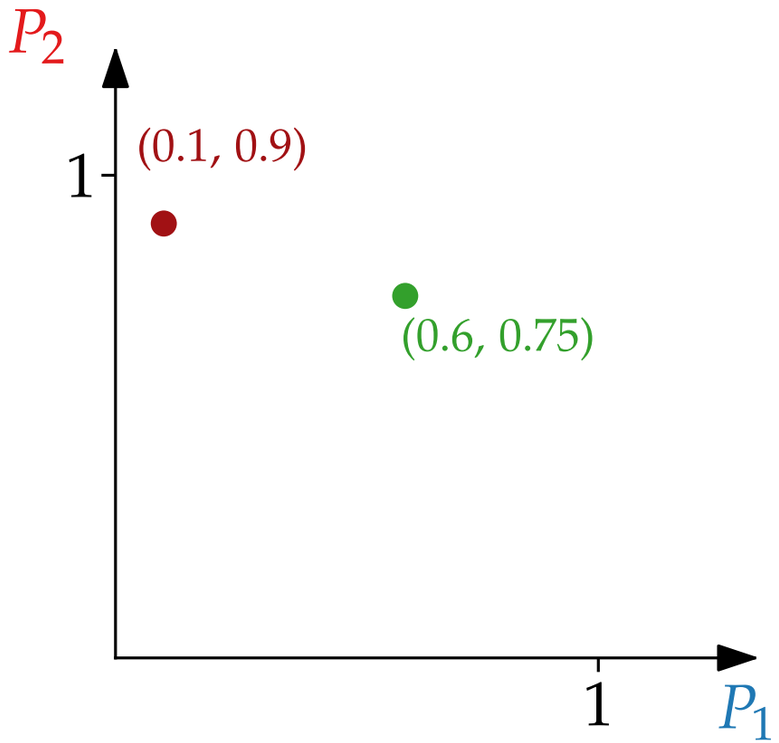


Free-Space Diagram – 1 Segment

Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.
$P_1(0.1)$ und $P_2(0.9)$ können **nicht** verbunden werden.

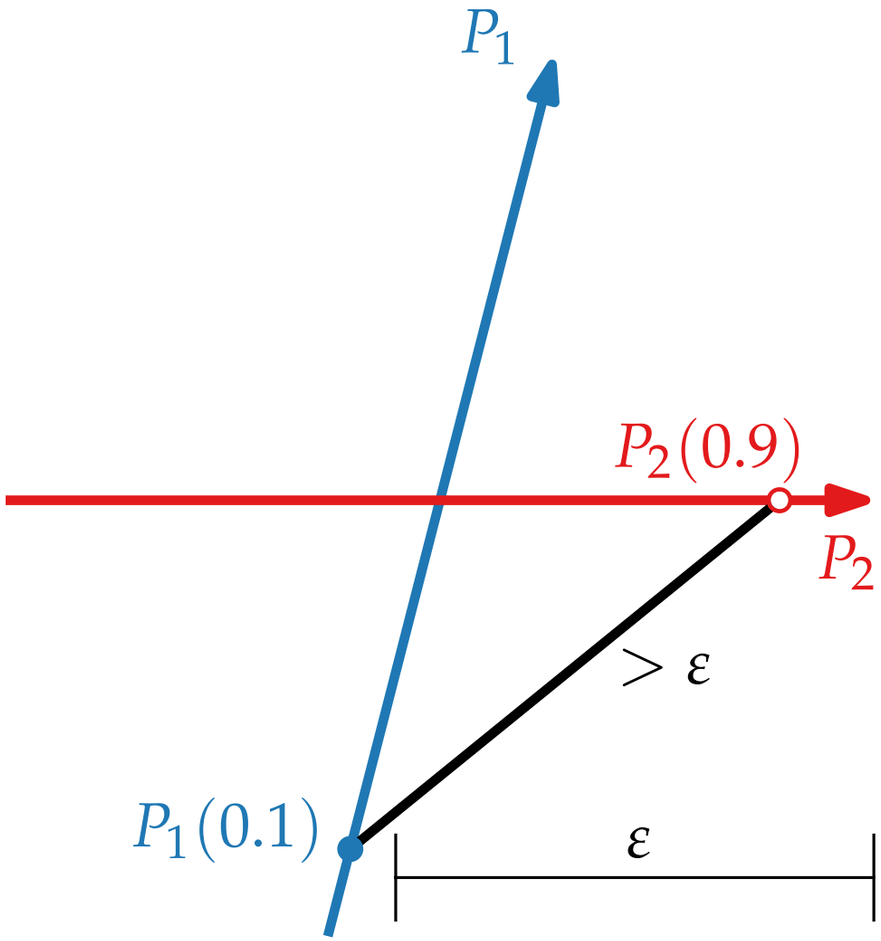


Das **Free-Space Diagram** F_ε zeigt an, welche Punktepaare mit einer Leine der Länge ε verbunden werden können.

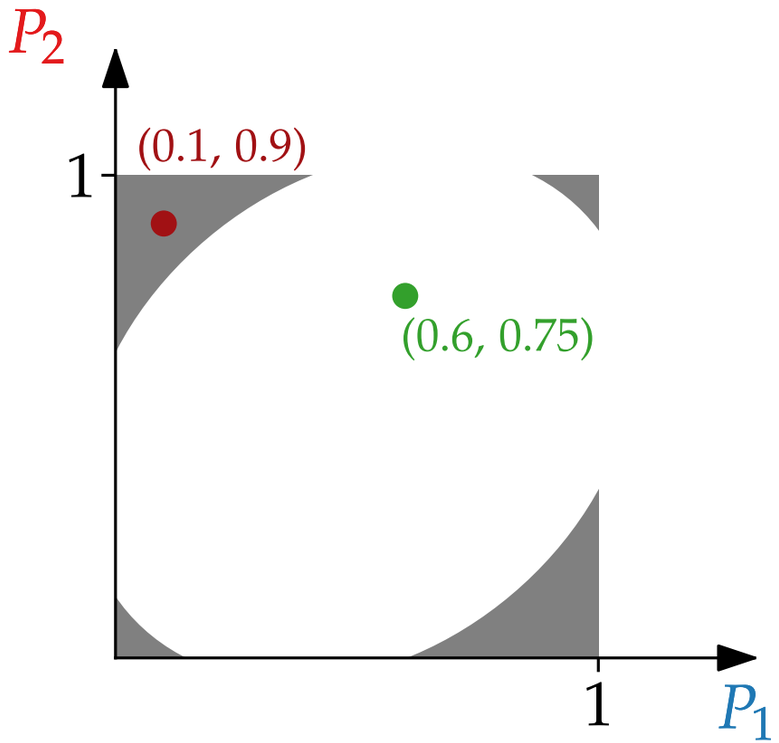


Free-Space Diagram – 1 Segment

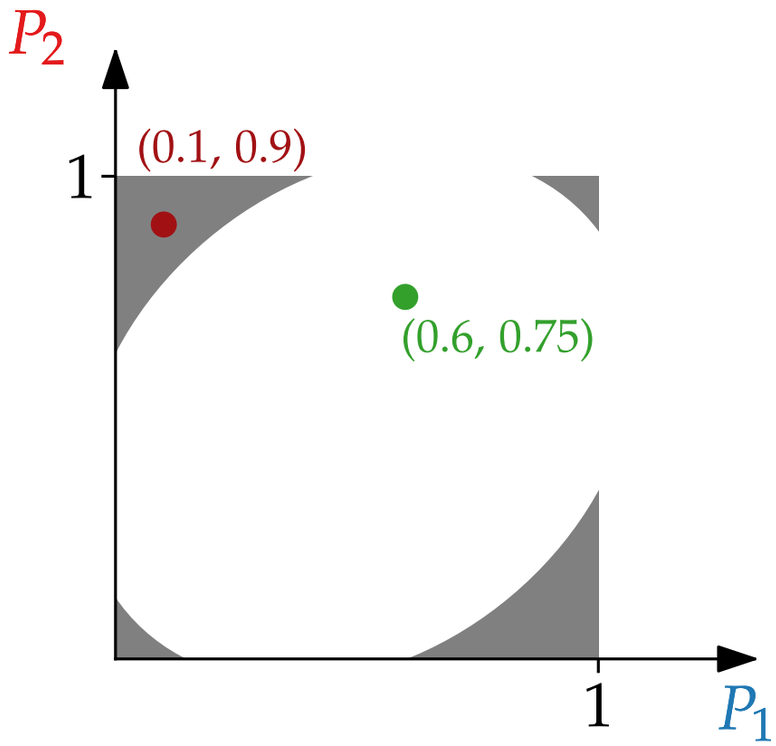
Spezialfall: P_1 und P_2 bestehen nur aus einem Segment.
$P_1(0.6)$ und $P_2(0.75)$ können verbunden werden.
$P_1(0.1)$ und $P_2(0.9)$ können **nicht** verbunden werden.



Das **Free-Space Diagram** F_ε zeigt an, welche Punktepaare mit einer Leine der Länge ε verbunden werden können.

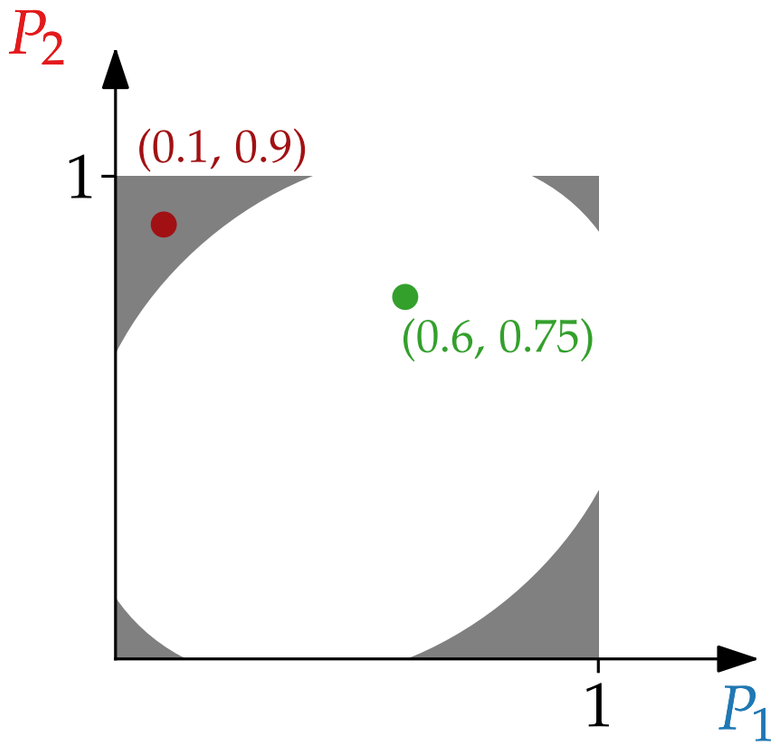


Free-Space Diagram – 1 Segment



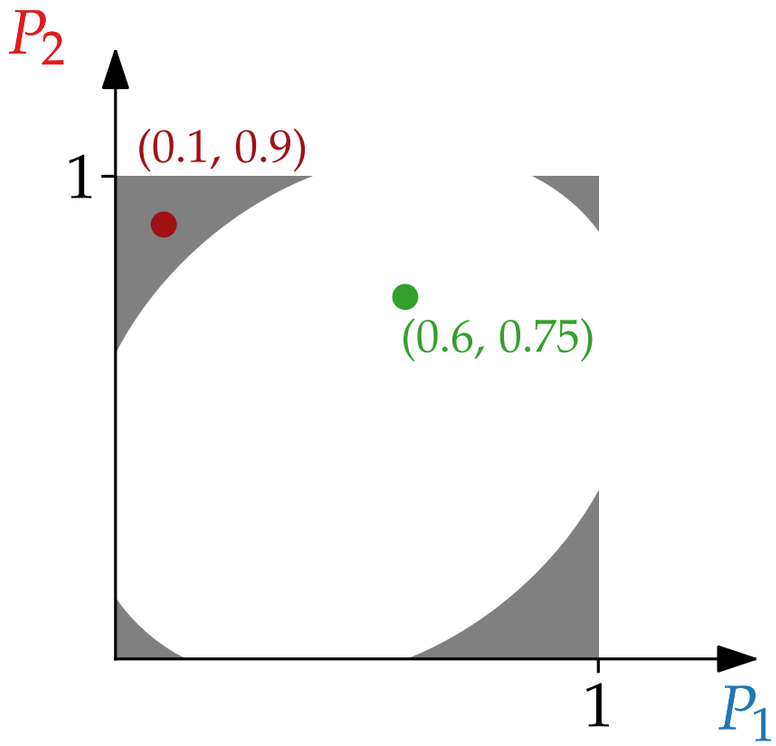
Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.



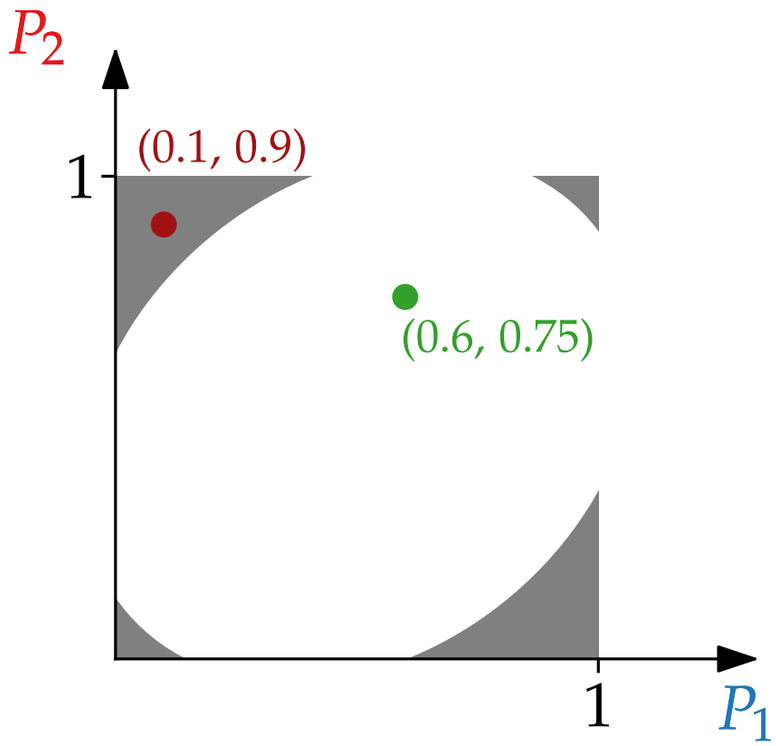
Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_{\mathrm{F}}(P_1, P_2) \leq \varepsilon \Leftrightarrow$



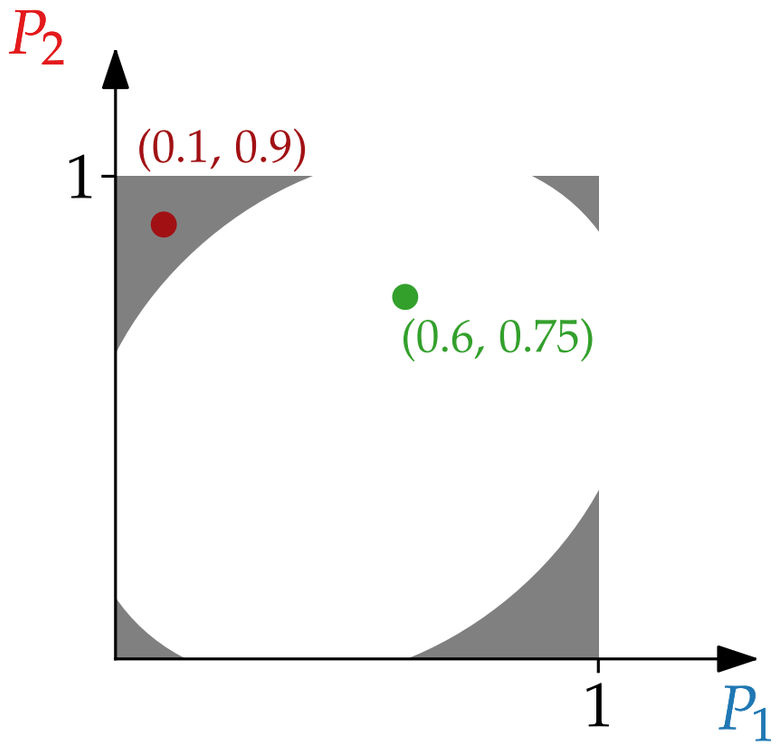
Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_\mathrm{F}(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$



Free-Space Diagram – 1 Segment

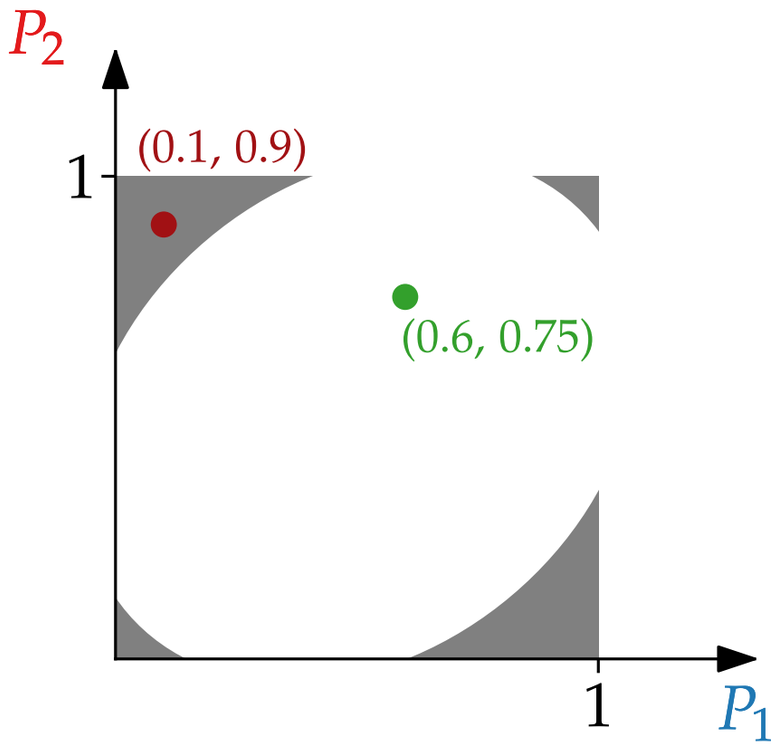
- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_{\mathrm{F}}(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.



Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_\mathrm{F}(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.

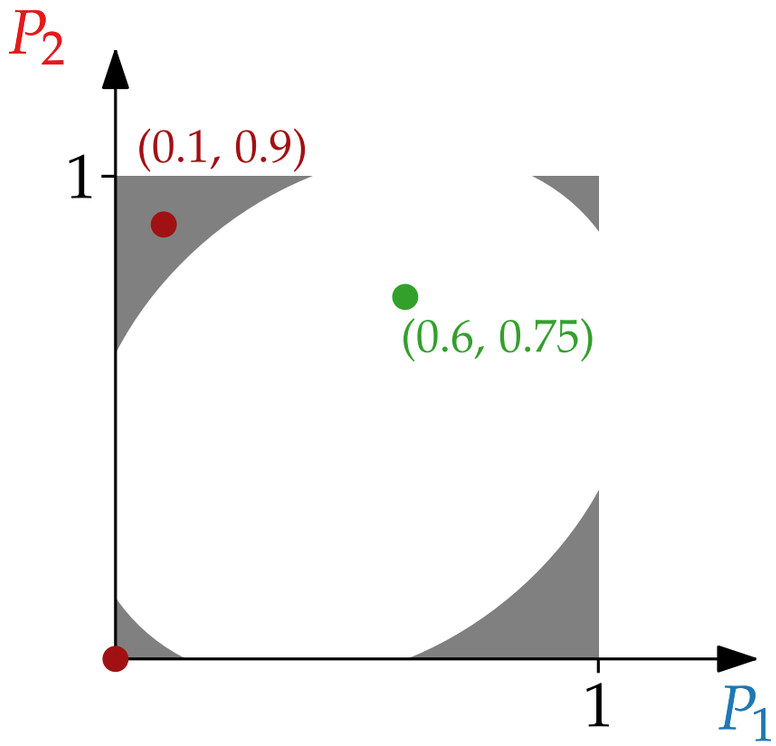
Unser Beispiel:



Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_\mathrm{F}(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.

Unser Beispiel:

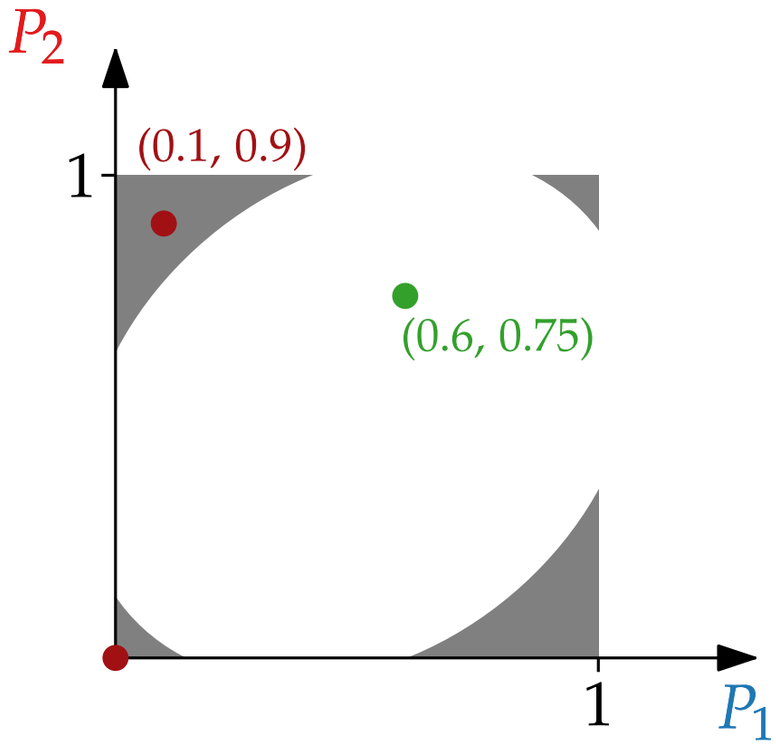


Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_F(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.

Unser Beispiel:

$(0,0) \notin F_\varepsilon \Rightarrow d_F(P_1, P_2) > \varepsilon$



Free-Space Diagram – 1 Segment

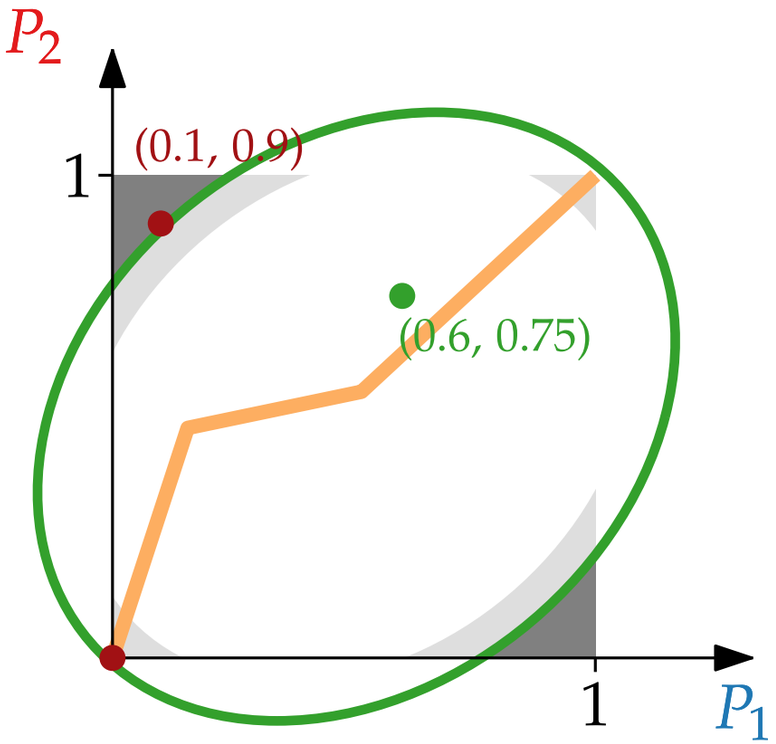
- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_F(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.

Unser Beispiel:

$(0,0) \notin F_\varepsilon \Rightarrow d_F(P_1, P_2) > \varepsilon$

Aber für etwas größeres ε:

$d_F(P_1, P_2) \leq \varepsilon$



Free-Space Diagram – 1 Segment

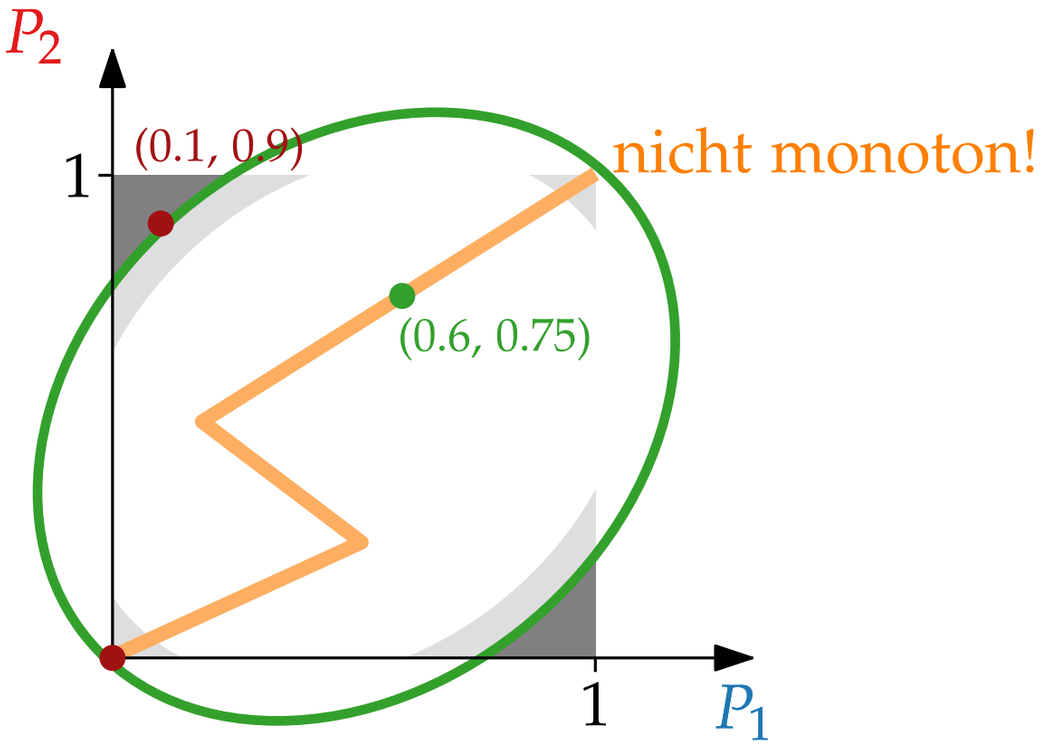
- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_F(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.

Unser Beispiel:

$(0,0) \notin F_\varepsilon \Rightarrow d_F(P_1, P_2) > \varepsilon$

Aber für etwas größeres ε:

$d_F(P_1, P_2) \leq \varepsilon$



Free-Space Diagram – 1 Segment

- $F_\varepsilon = \left\{ (s,t) \in [0,1]^2 \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ hat die Form einer Ellipse.
- $d_F(P_1, P_2) \leq \varepsilon \Leftrightarrow$ F_ε enthält einen Pfad von $(0,0)$ bis $(1,1)$, der in beide Richtungen monoton ist.

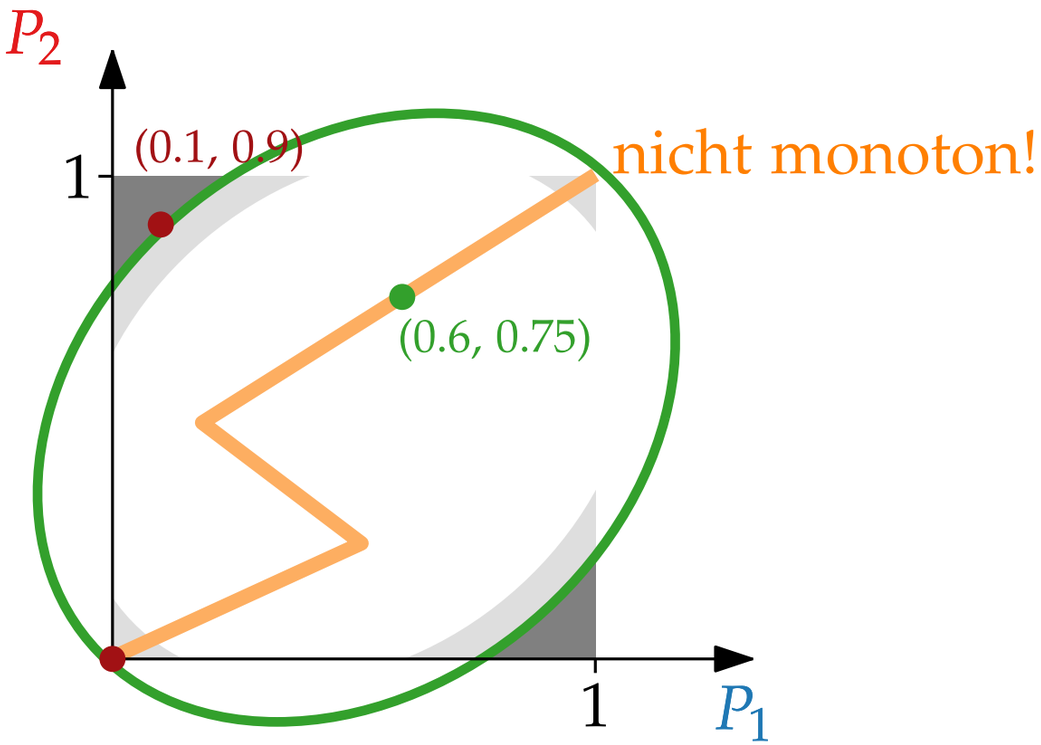
Unser Beispiel:

$(0,0) \notin F_\varepsilon \Rightarrow d_F(P_1, P_2) > \varepsilon$

Aber für etwas größeres ε:

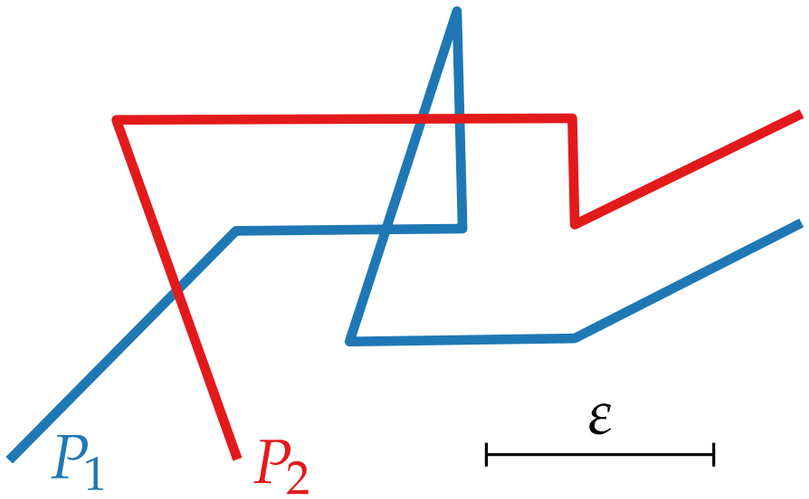
$d_F(P_1, P_2) \leq \varepsilon$

Mehr als ein Segment?



Free-Space Diagram – Allgemein

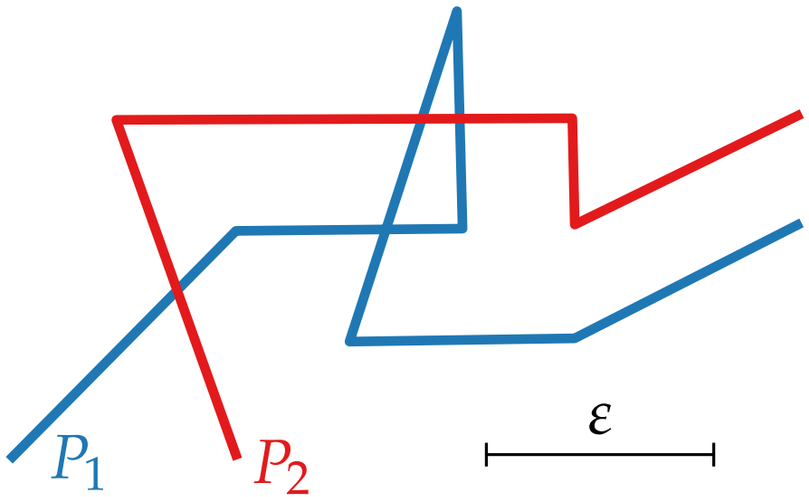
Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.



Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

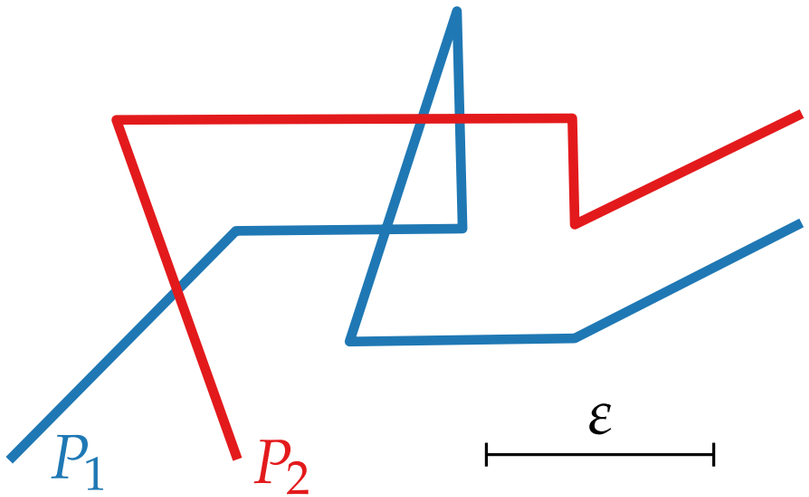
Free-Space Diagram $F_\varepsilon = \{$ $\}$



Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

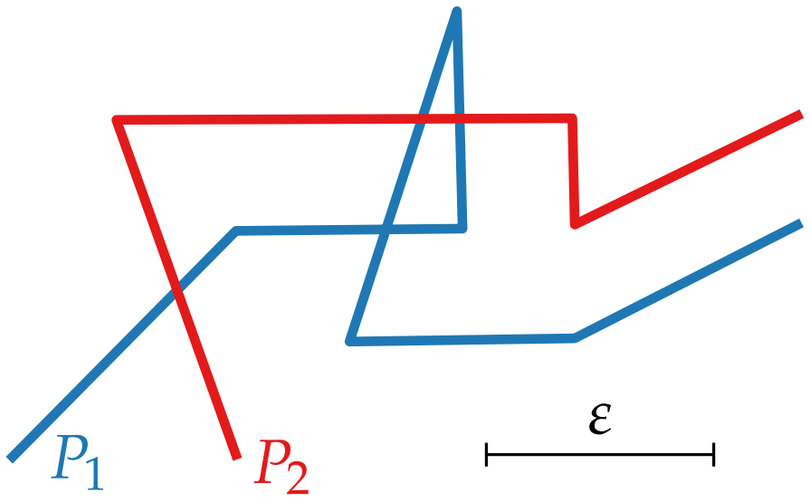
Free-Space Diagram $F_\varepsilon = \Big\{ \quad \Big| \; d(P_1(s), P_2(t)) \leq \varepsilon \Big\}$



Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

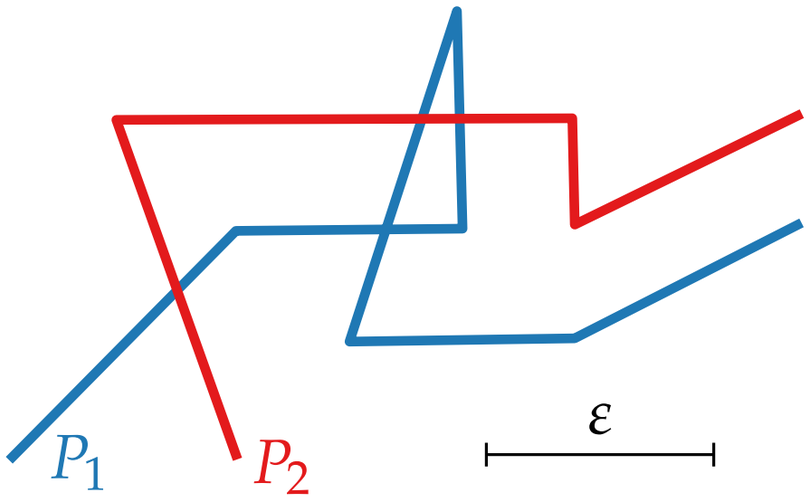
Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$

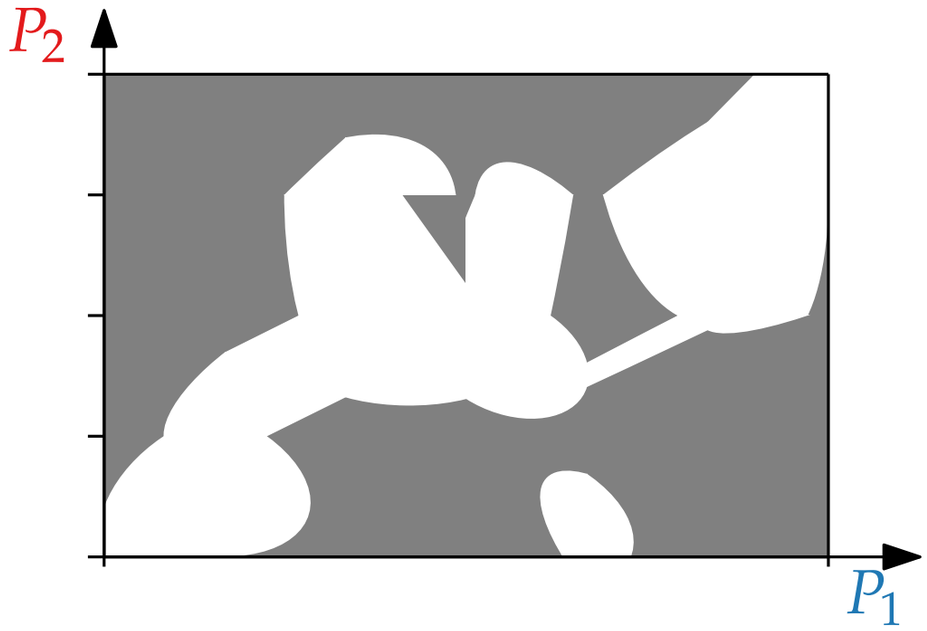


Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$

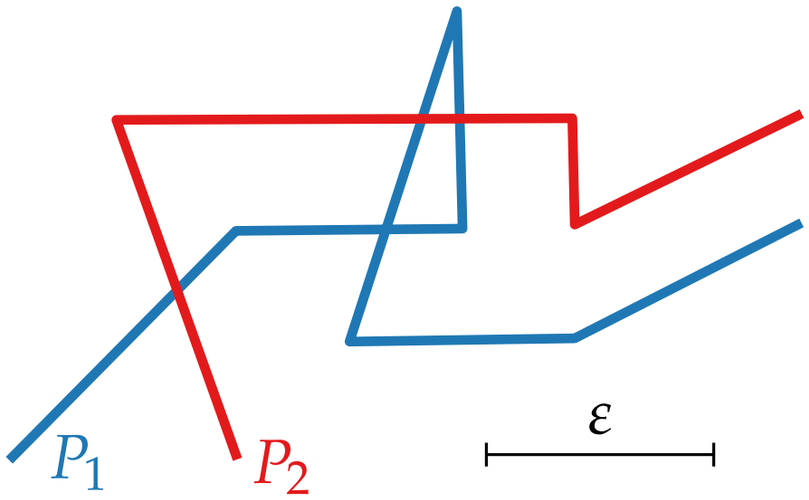


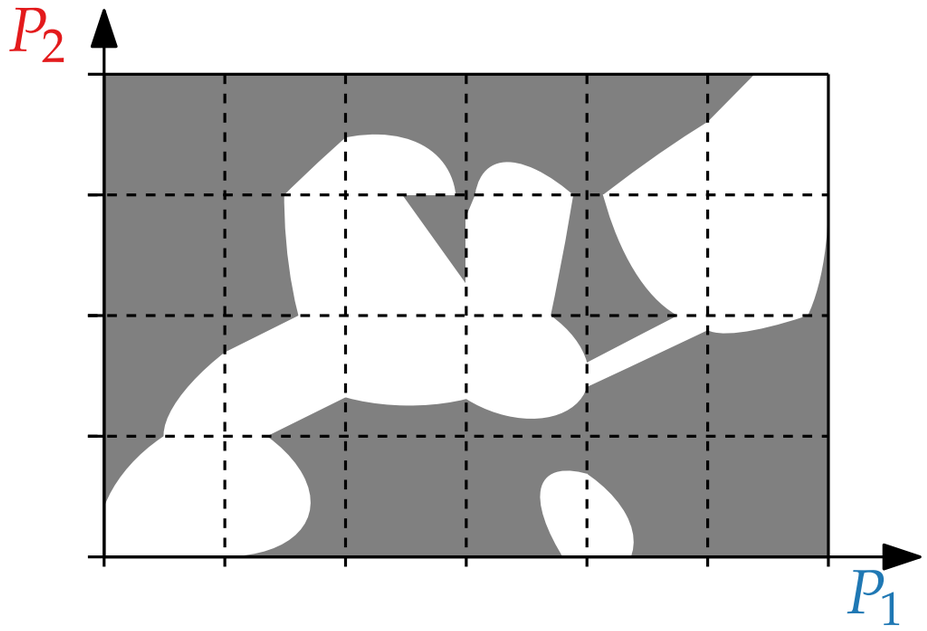


Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$

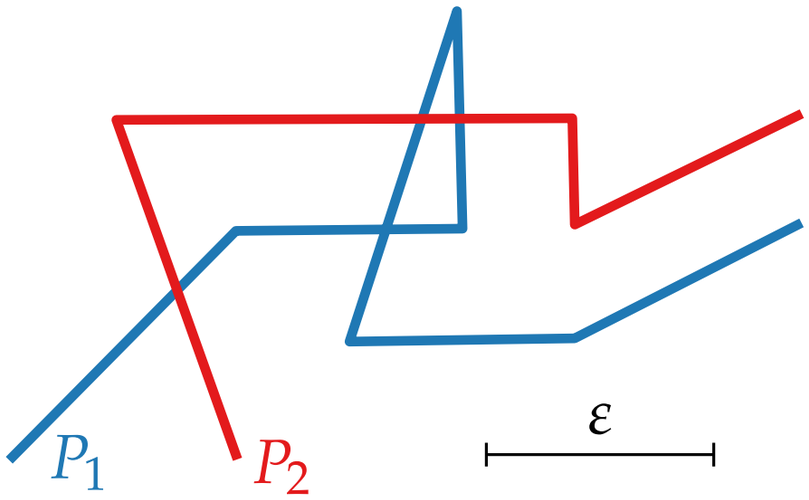


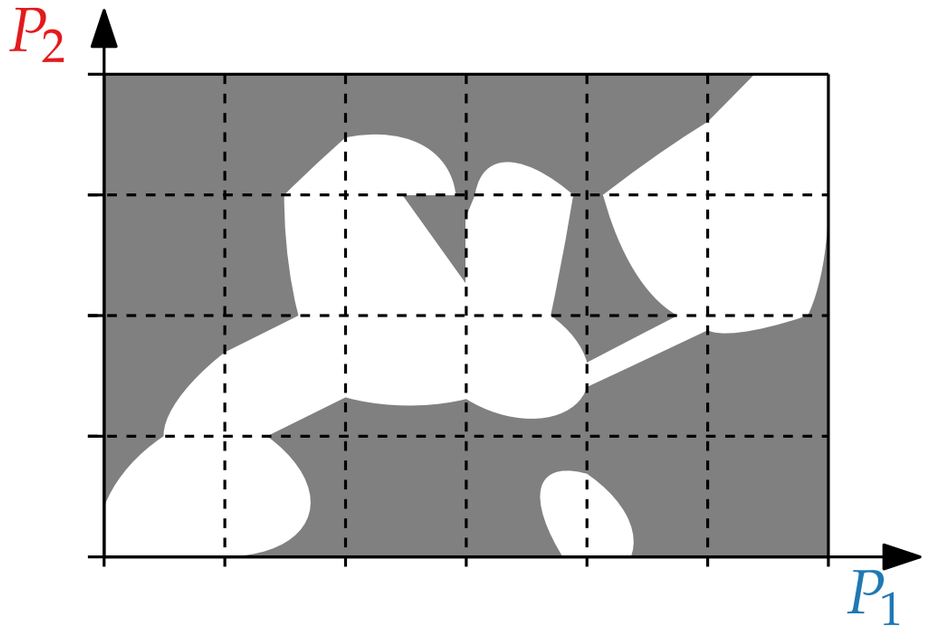


Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ besteht aus freien Bereichen für jedes Paar von Segmenten.

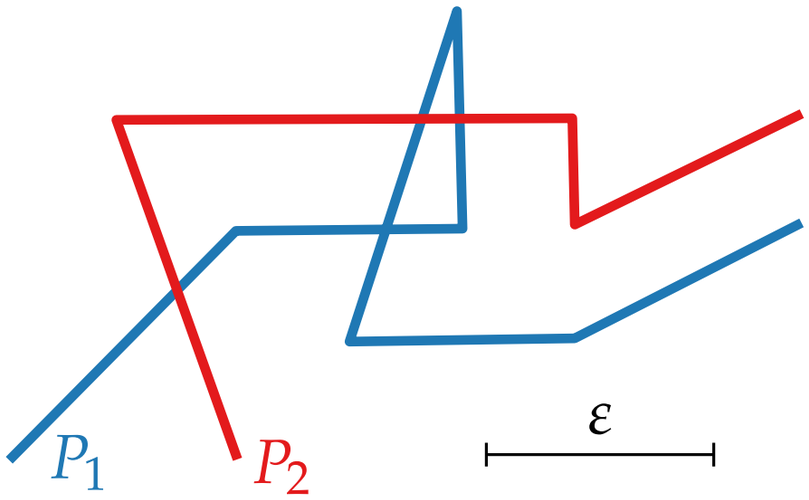


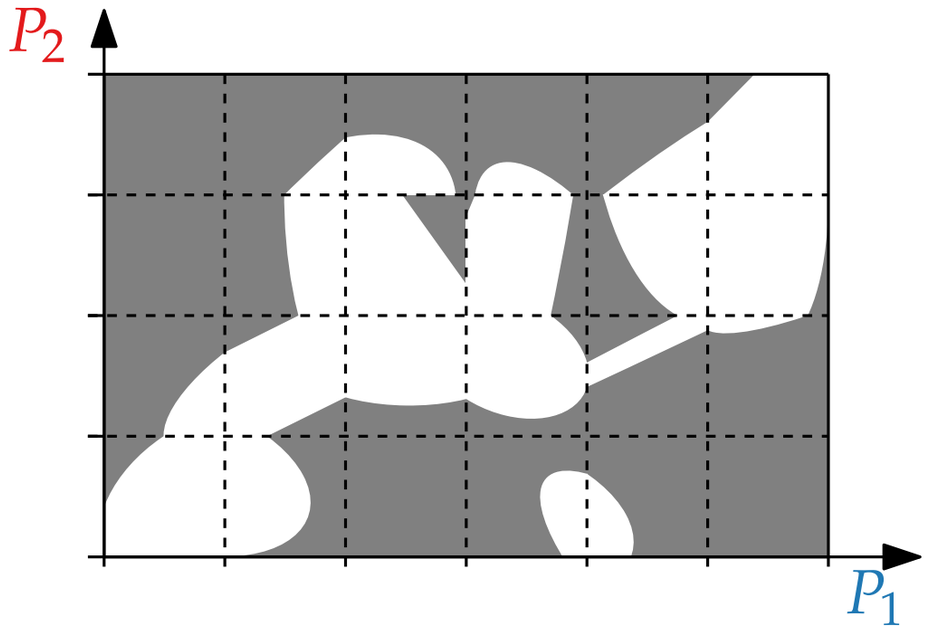


Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$ besteht aus freien Bereichen für jedes Paar von Segmenten.



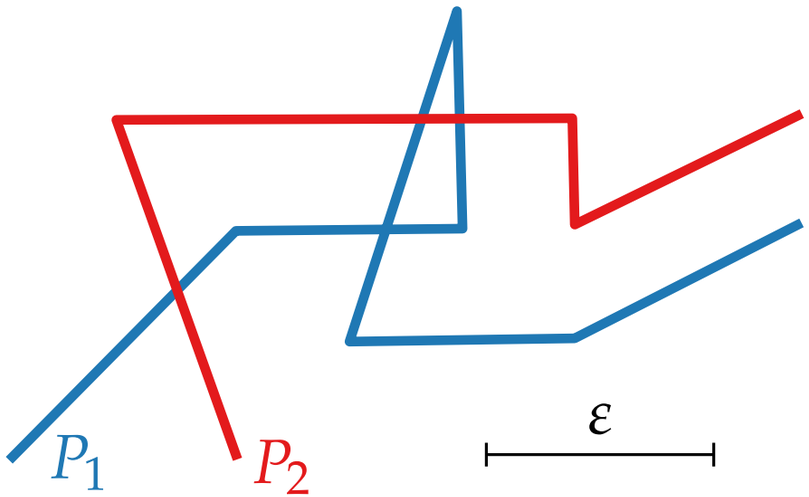


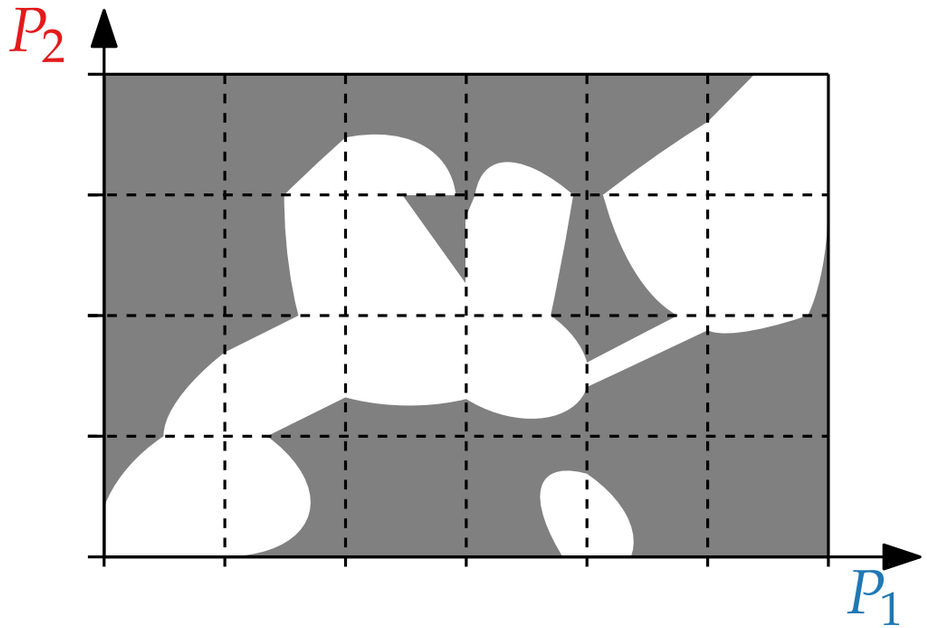
Unser Beispiel:

Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$
besteht aus freien Bereichen für jedes Paar von Segmenten.





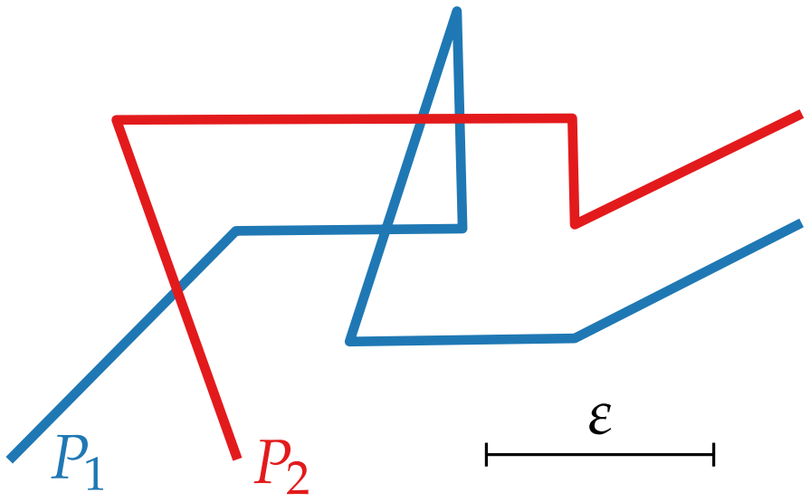
Unser Beispiel:

Demo. `https://algo.uni-trier.de/demos/frechet`

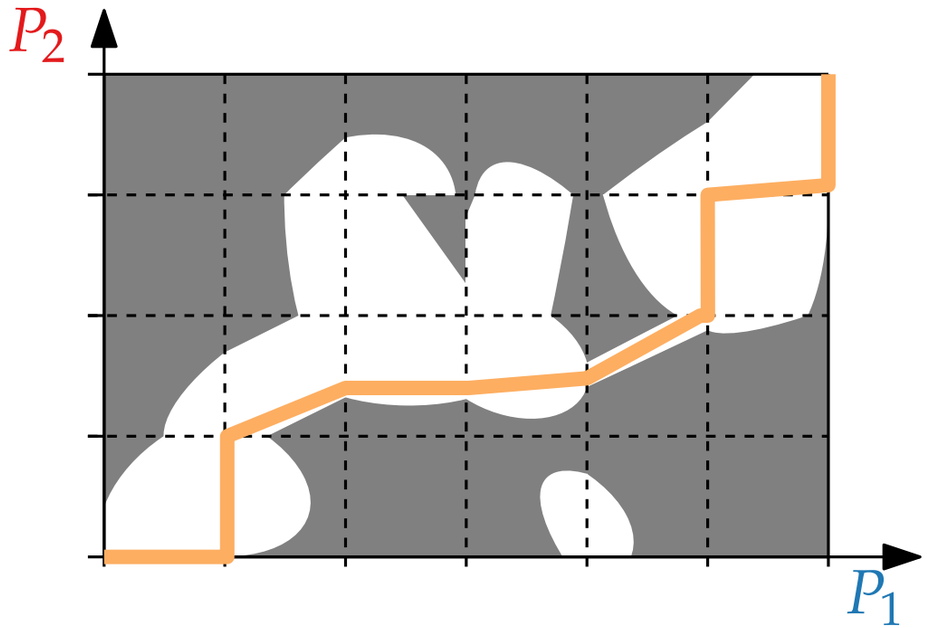
Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$
besteht aus freien Bereichen für jedes Paar von Segmenten.



Unser Beispiel:

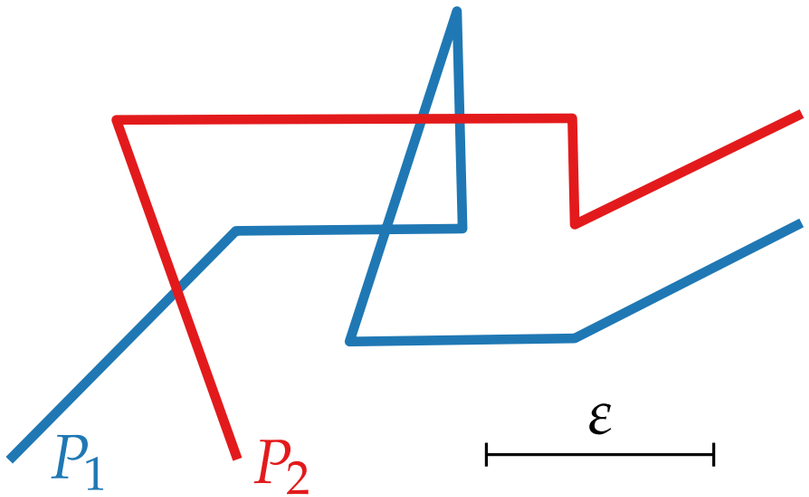


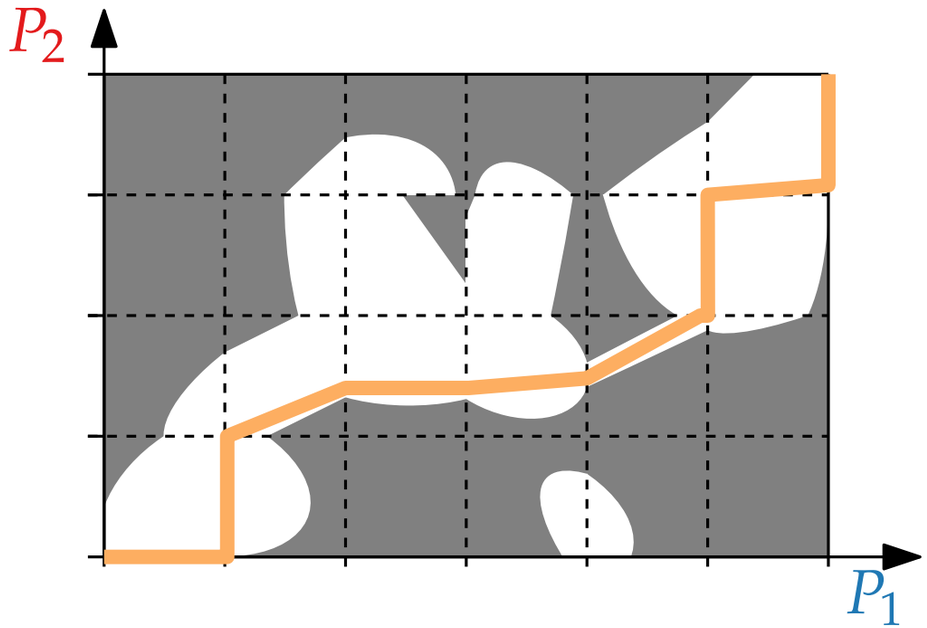
Demo. `https://algo.uni-trier.de/demos/frechet`

Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$
besteht aus freien Bereichen für jedes Paar von Segmenten.





Unser Beispiel:

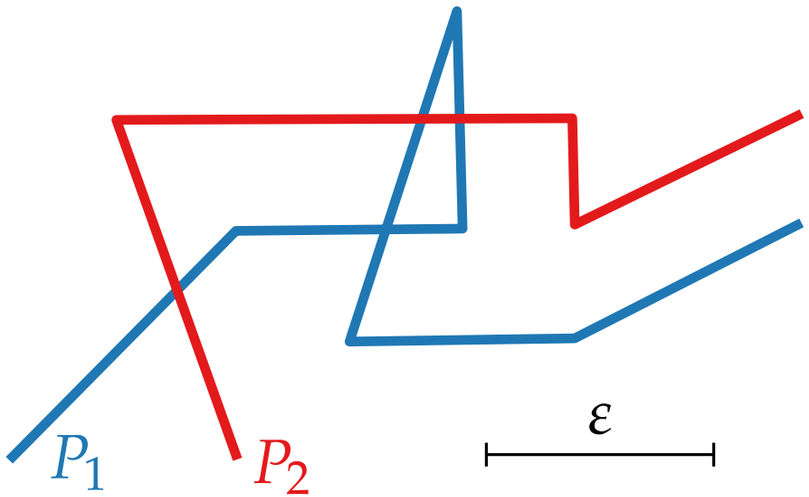
F_ε enthält monotonen Pfad von $(0,0)$ nach $(n_1 - 1, n_2 - 1)$
$\Rightarrow d_{\mathrm{F}}(P_1, P_2) \leq \varepsilon$

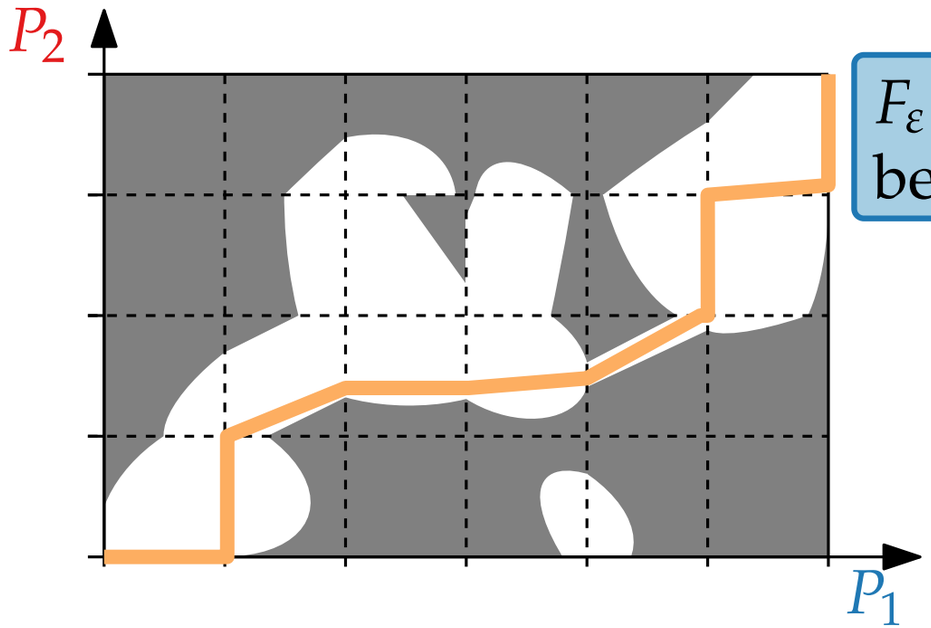
Demo. `https://algo.uni-trier.de/demos/frechet`

Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$
besteht aus freien Bereichen für jedes Paar von Segmenten.





F_ε kann in $\mathcal{O}(n_1 n_2)$ Zeit berechnet werden.

Unser Beispiel:

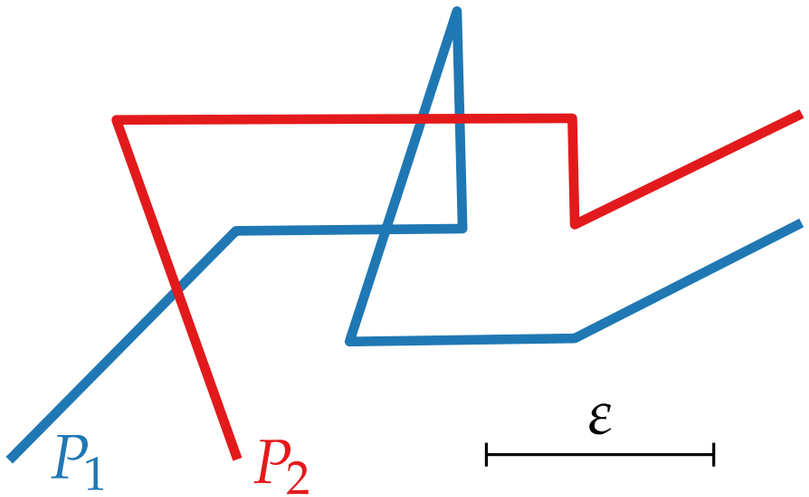
F_ε enthält monotonen Pfad von $(0,0)$ nach $(n_1 - 1, n_2 - 1)$
$\Rightarrow d_{\mathrm{F}}(P_1, P_2) \leq \varepsilon$

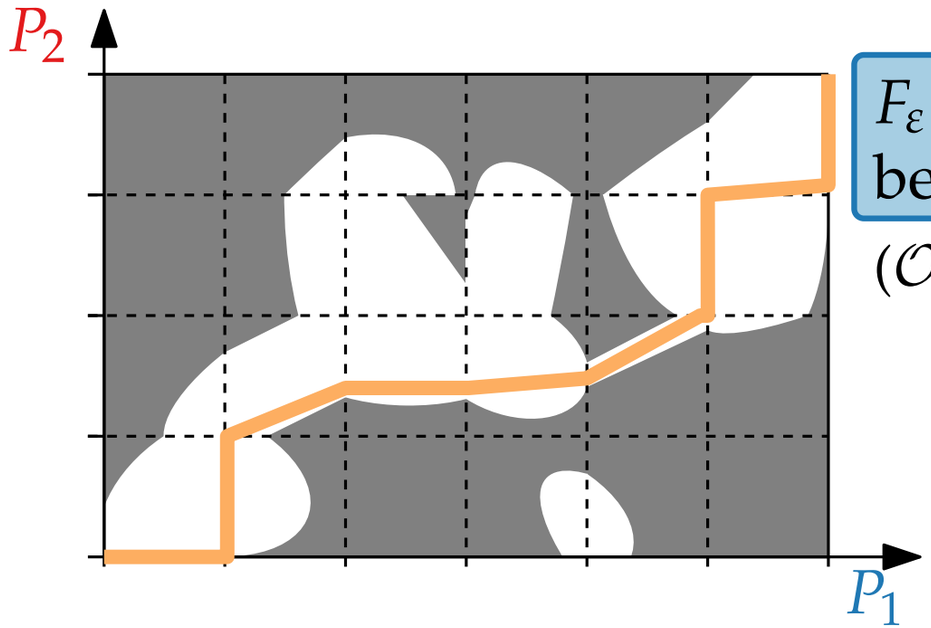
Demo. `https://algo.uni-trier.de/demos/frechet`

Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$
besteht aus freien Bereichen für jedes Paar von Segmenten.





F_ε kann in $\mathcal{O}(n_1 n_2)$ Zeit berechnet werden.

($\mathcal{O}(1)$ Zeit pro Segmentpaar).

Unser Beispiel:

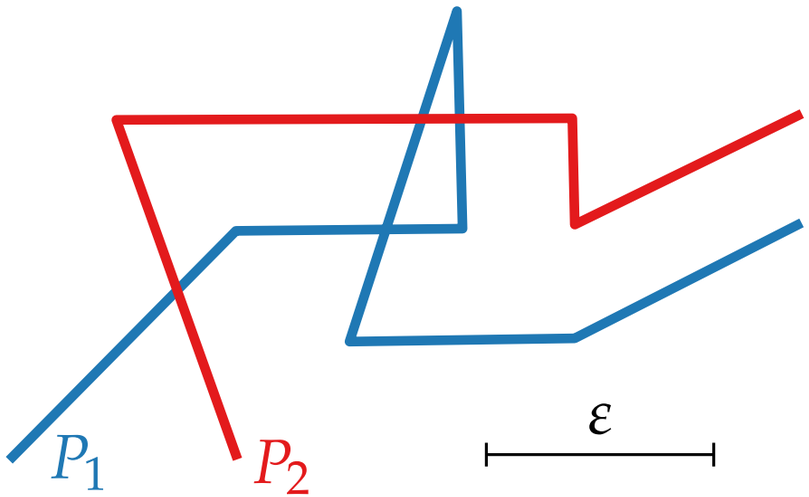
F_ε enthält monotonen Pfad von $(0,0)$ nach $(n_1 - 1, n_2 - 1)$
$\Rightarrow d_{\mathrm{F}}(P_1, P_2) \leq \varepsilon$

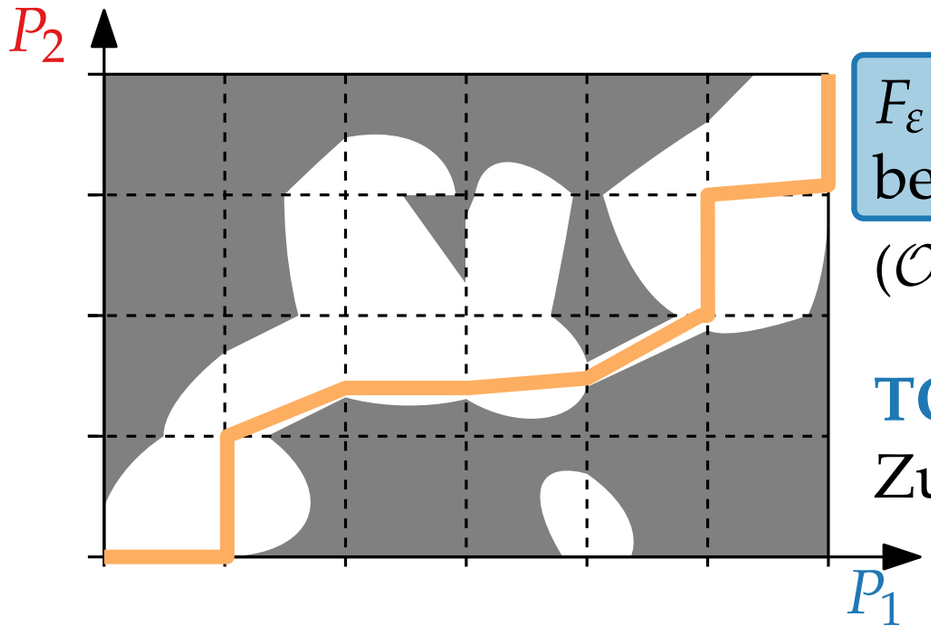
Demo. `https://algo.uni-trier.de/demos/frechet`

Free-Space Diagram – Allgemein

Allgemeiner Fall: P_1 und P_2 haben mehrere Segmente.

Free-Space Diagram $F_\varepsilon = \left\{ (s,t) \in [0, n_1 - 1] \times [0, n_2 - 1] \;\middle|\; d(P_1(s), P_2(t)) \leq \varepsilon \right\}$
besteht aus freien Bereichen für jedes Paar von Segmenten.





F_ε kann in $\mathcal{O}(n_1 n_2)$ Zeit berechnet werden.

($\mathcal{O}(1)$ Zeit pro Segmentpaar).

TODO:
Zulässigen Pfad finden

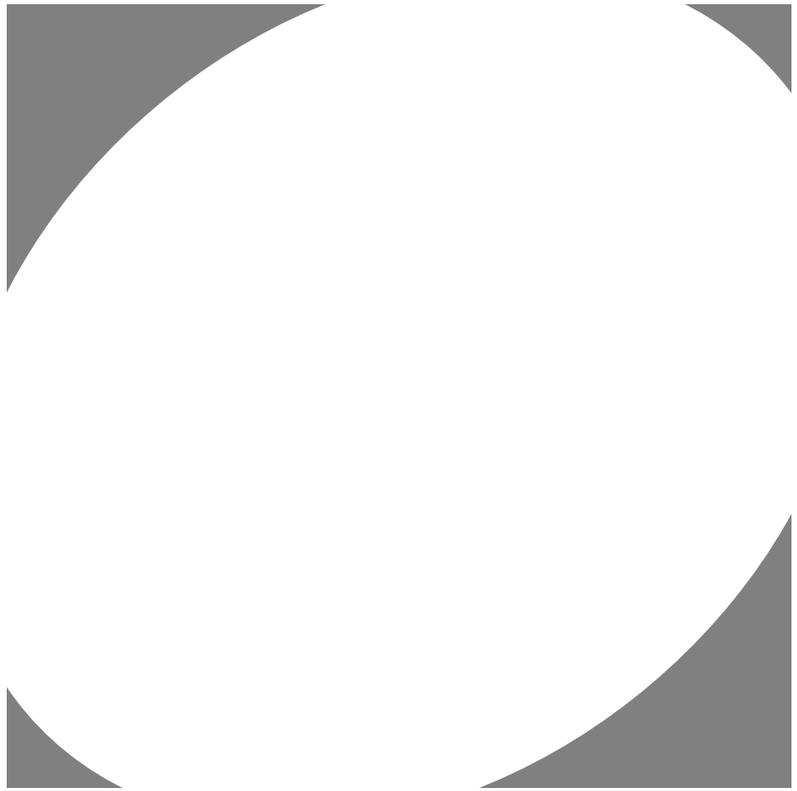
Unser Beispiel:

F_ε enthält monotonen Pfad von $(0,0)$ nach $(n_1 - 1, n_2 - 1)$
$\Rightarrow d_{\mathrm{F}}(P_1, P_2) \leq \varepsilon$

Demo. `https://algo.uni-trier.de/demos/frechet`

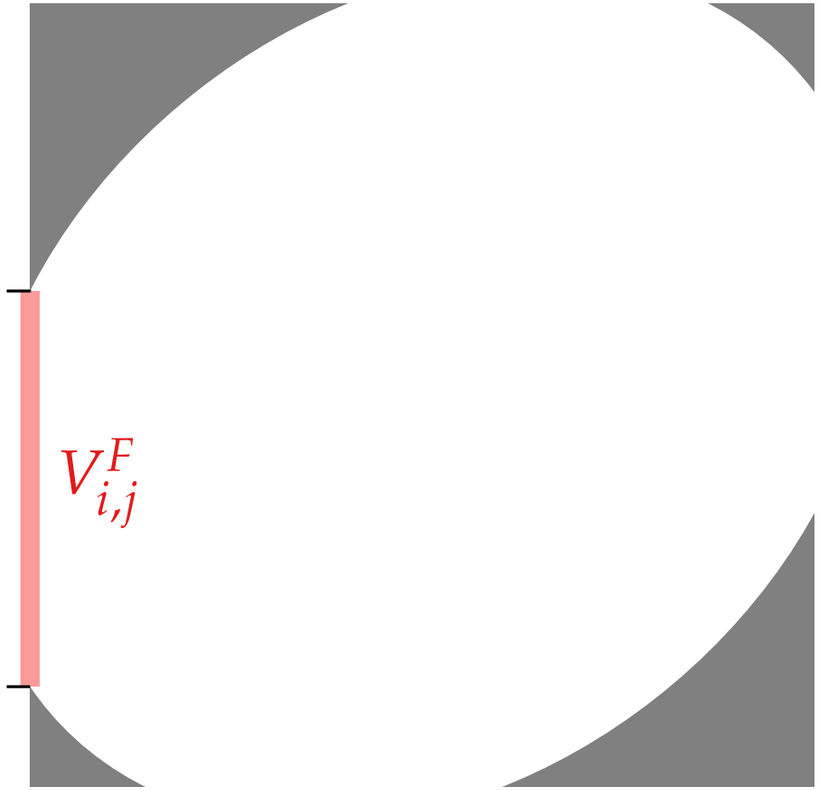
Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



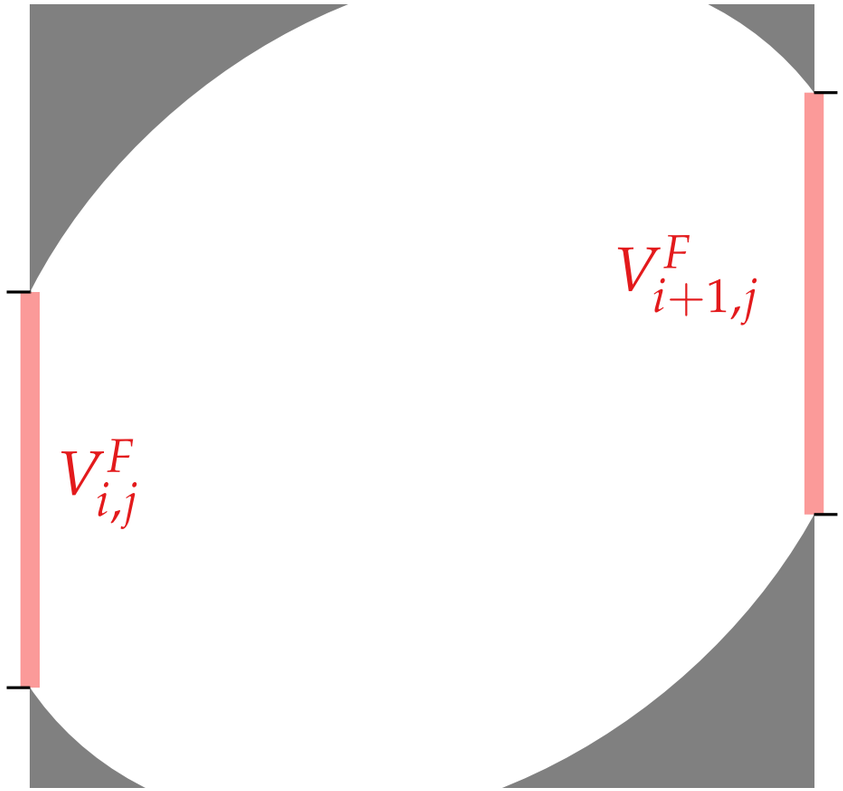
Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



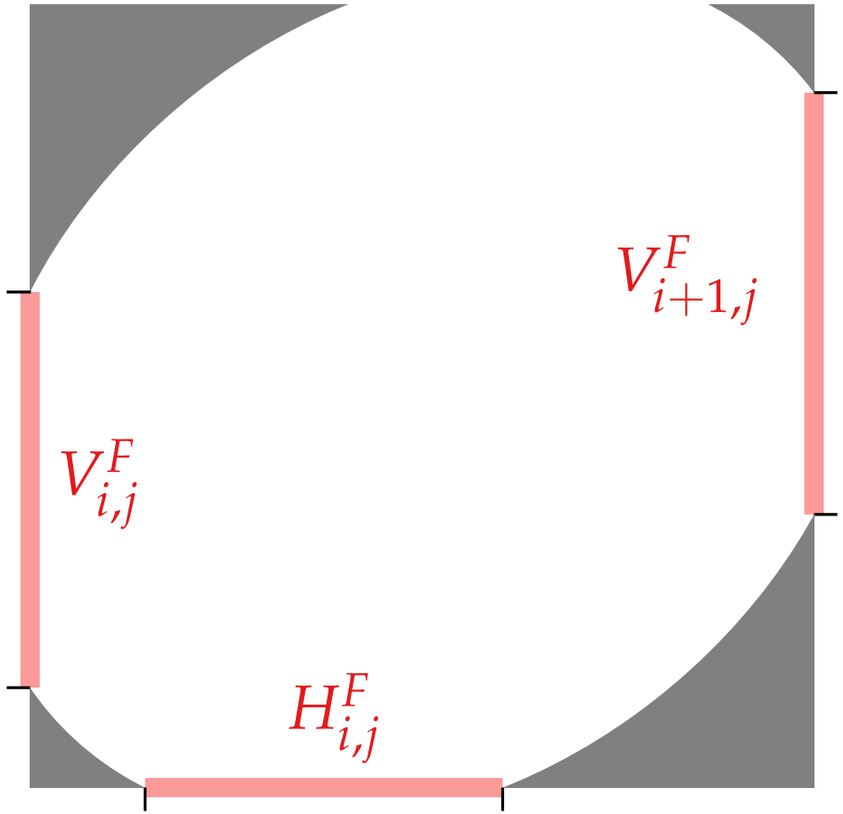
Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



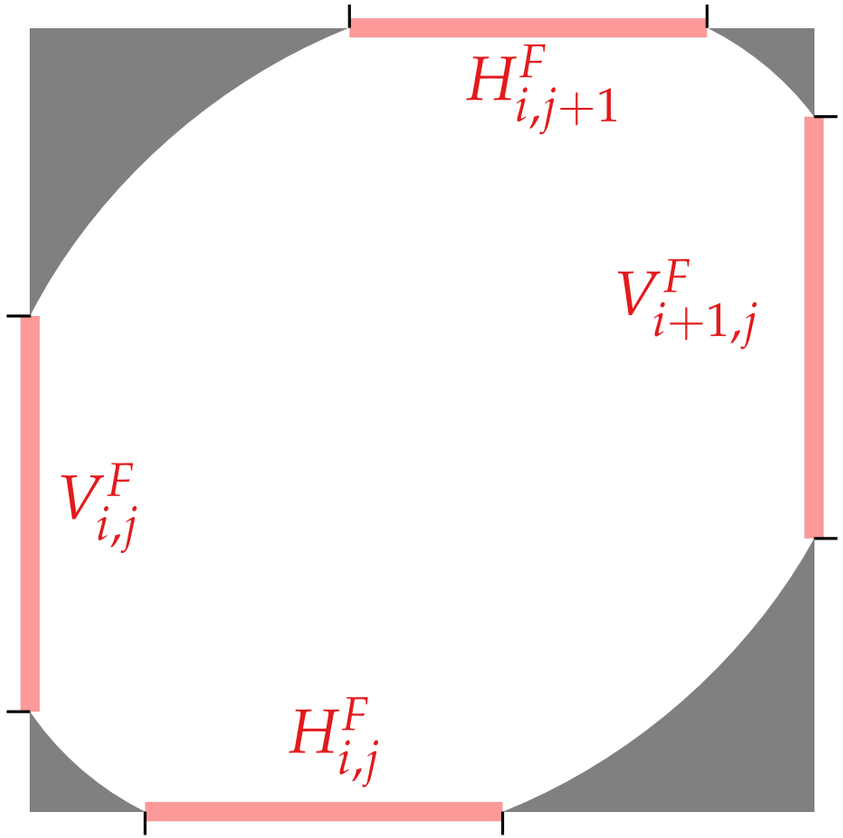
Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



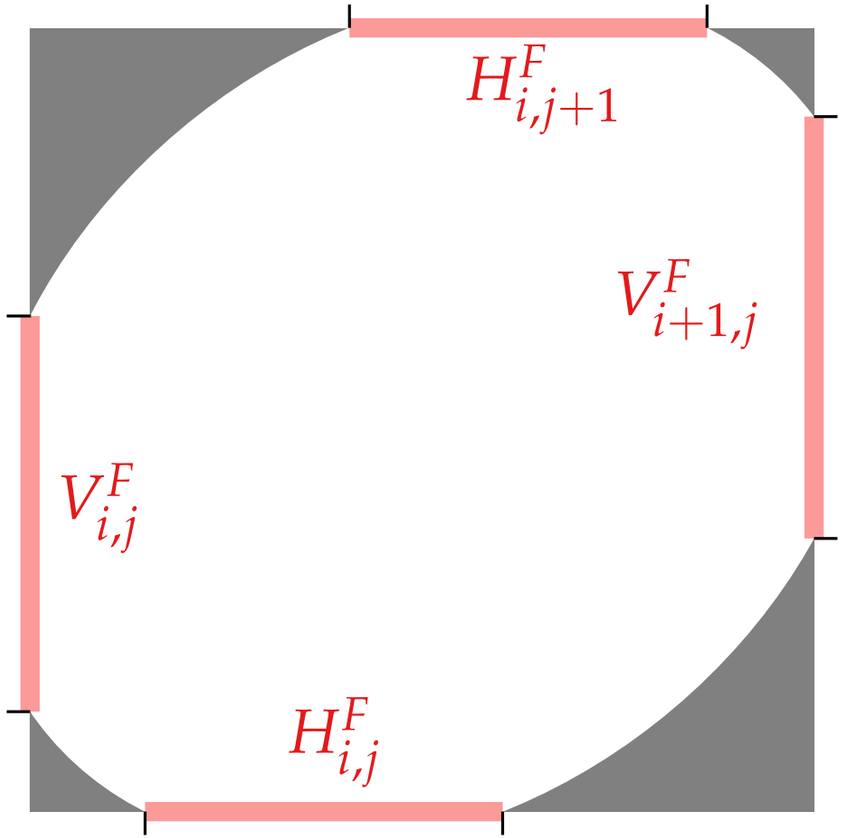
Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



Monotonen Pfad finden

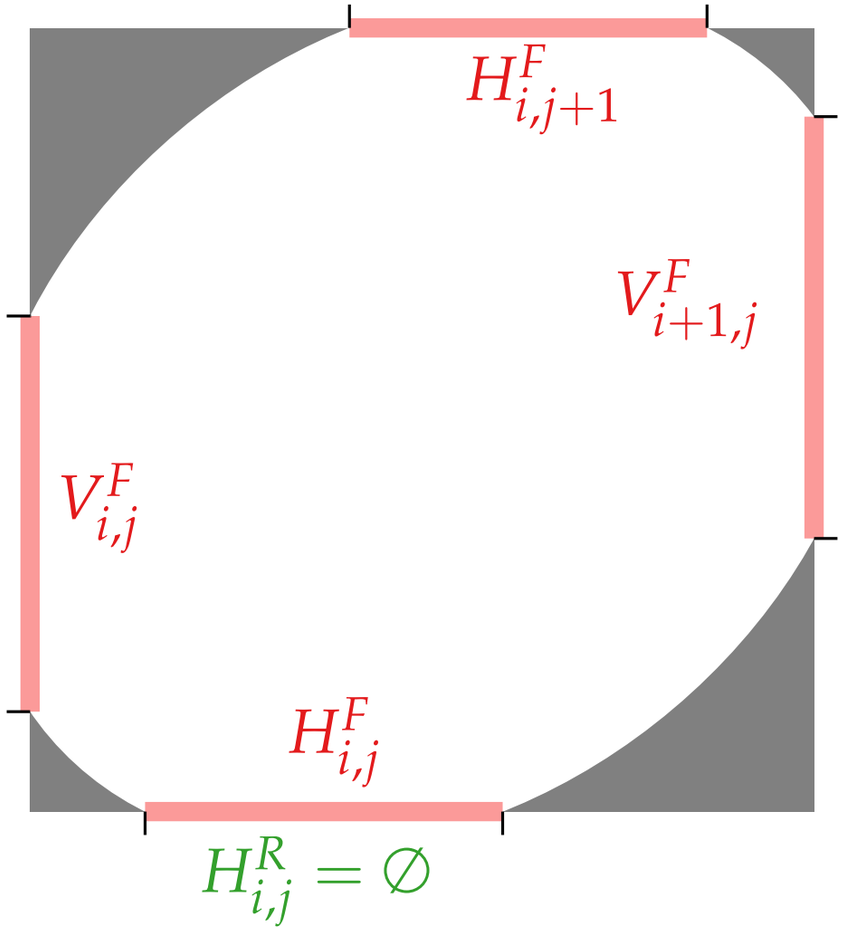
Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

Monotonen Pfad finden

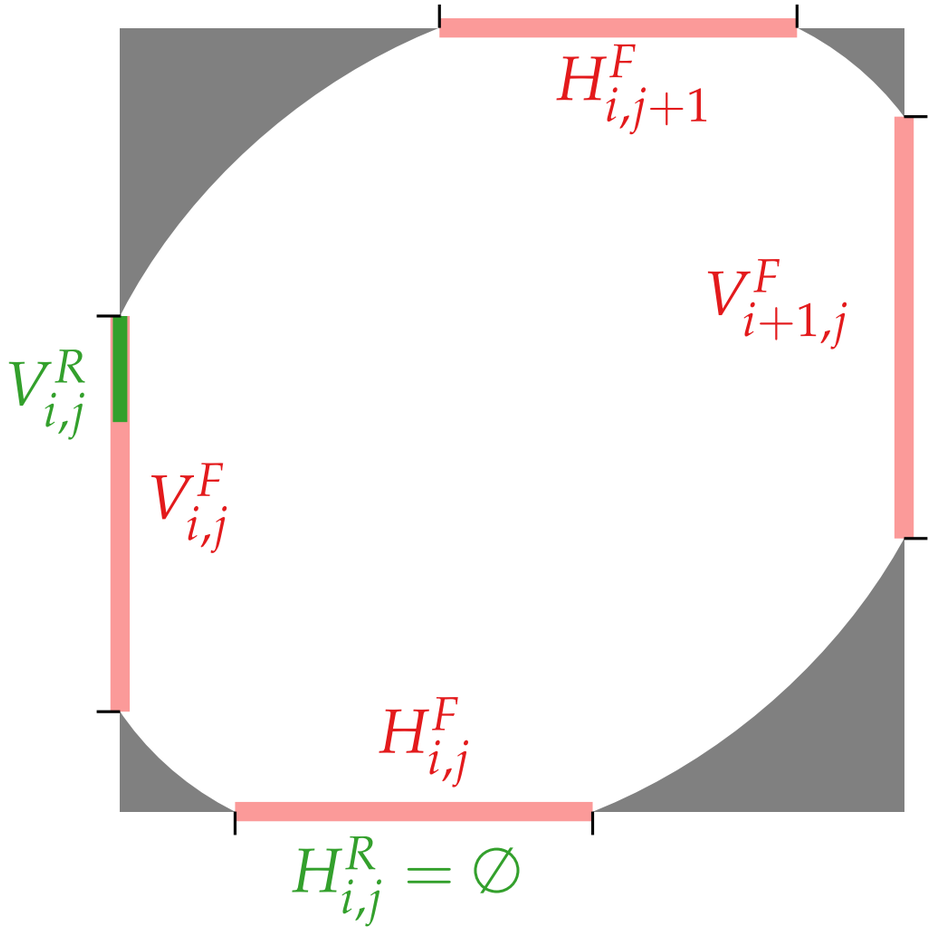
Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

Monotonen Pfad finden

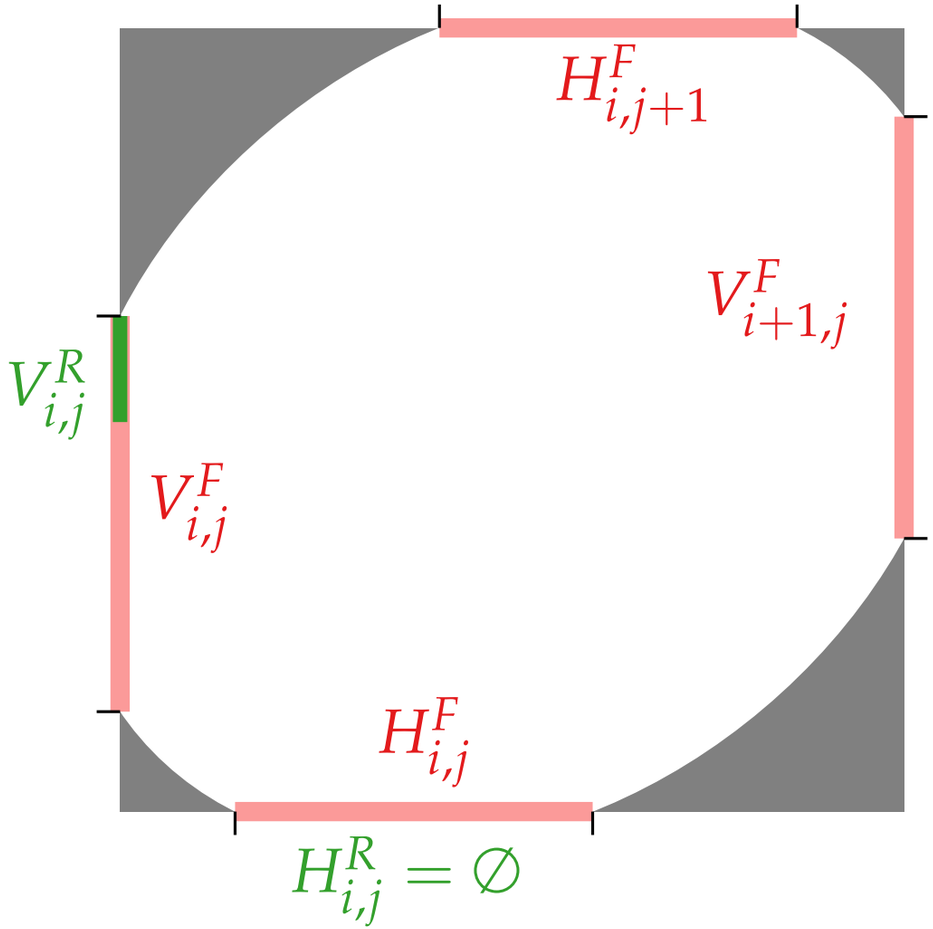
Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$



Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$

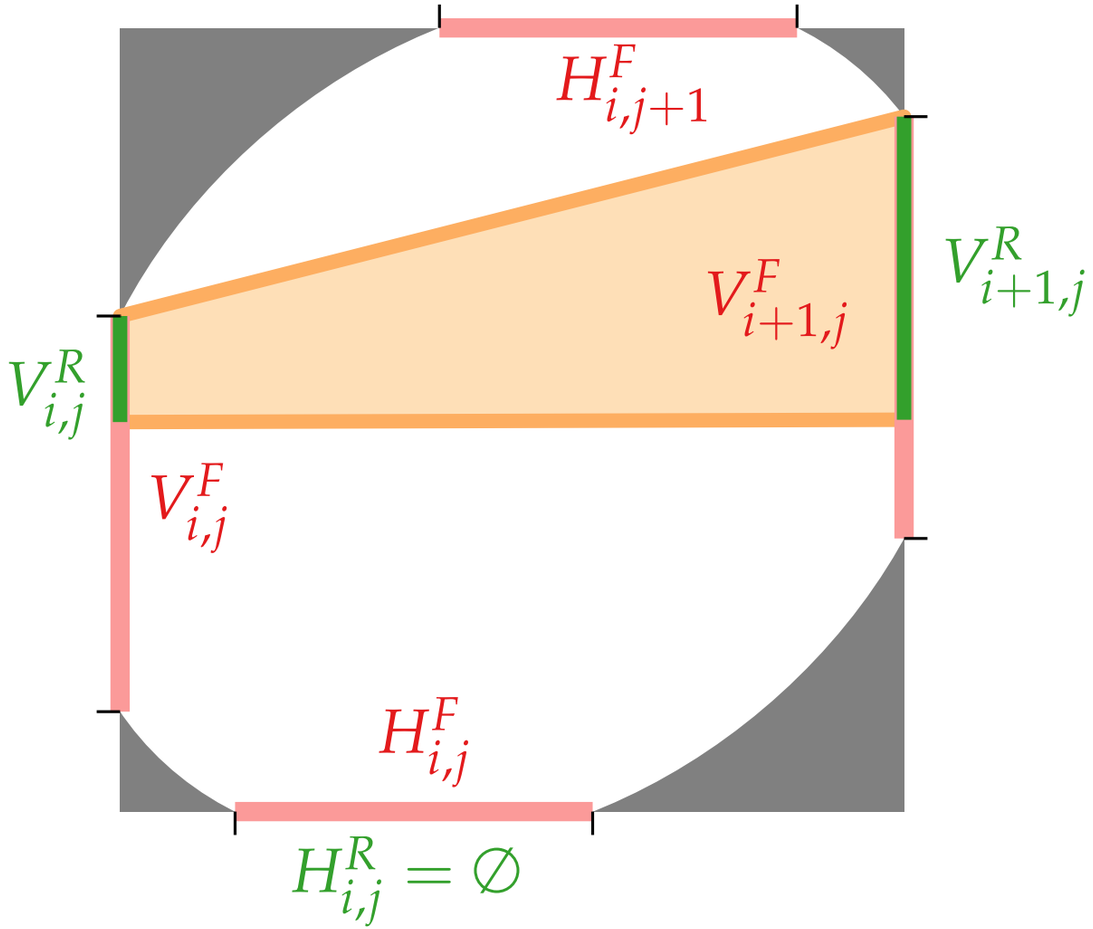


Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

$H^R_{i,j}$ und $V^R_{i,j}$ sind bekannt.

Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$

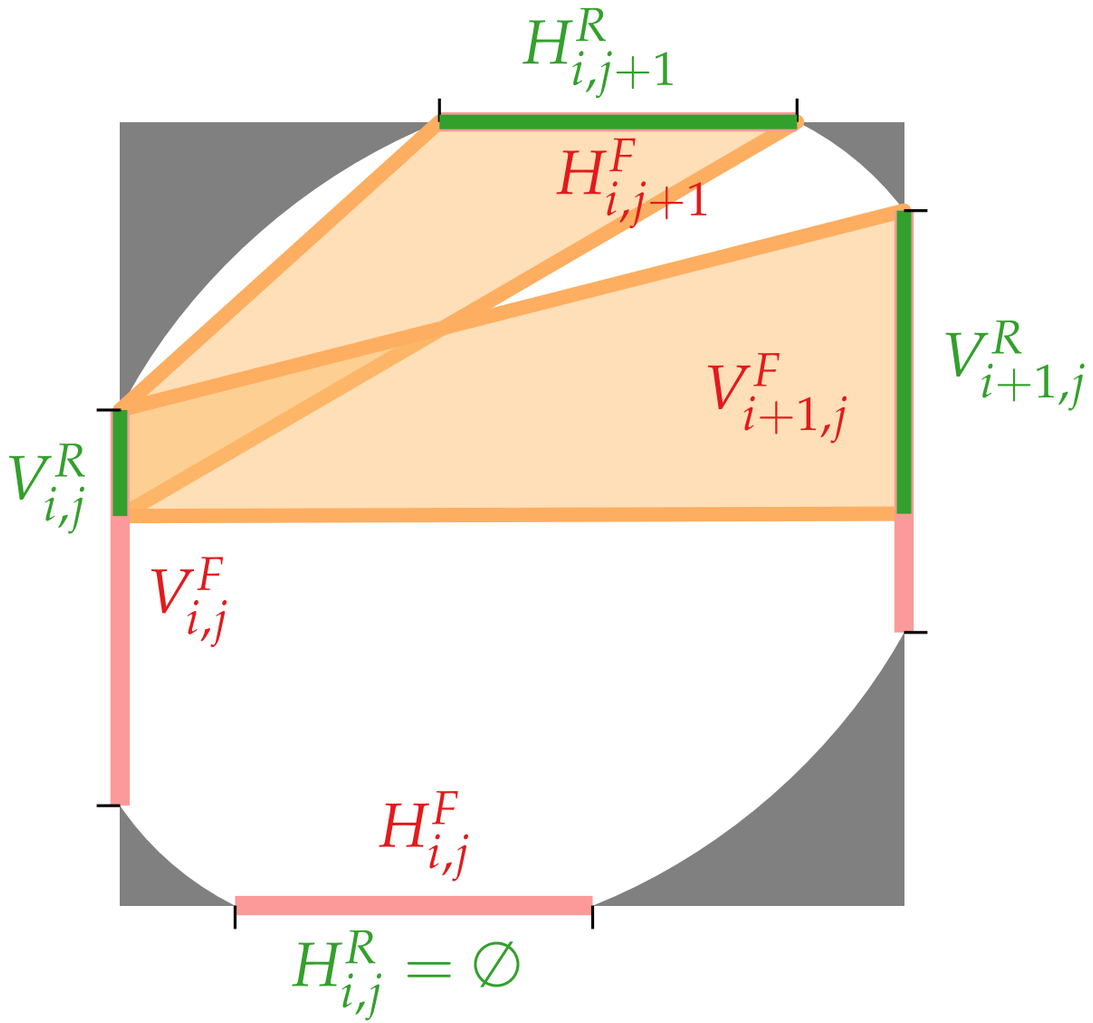


Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

$H^R_{i,j}$ und $V^R_{i,j}$ sind bekannt.

Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$

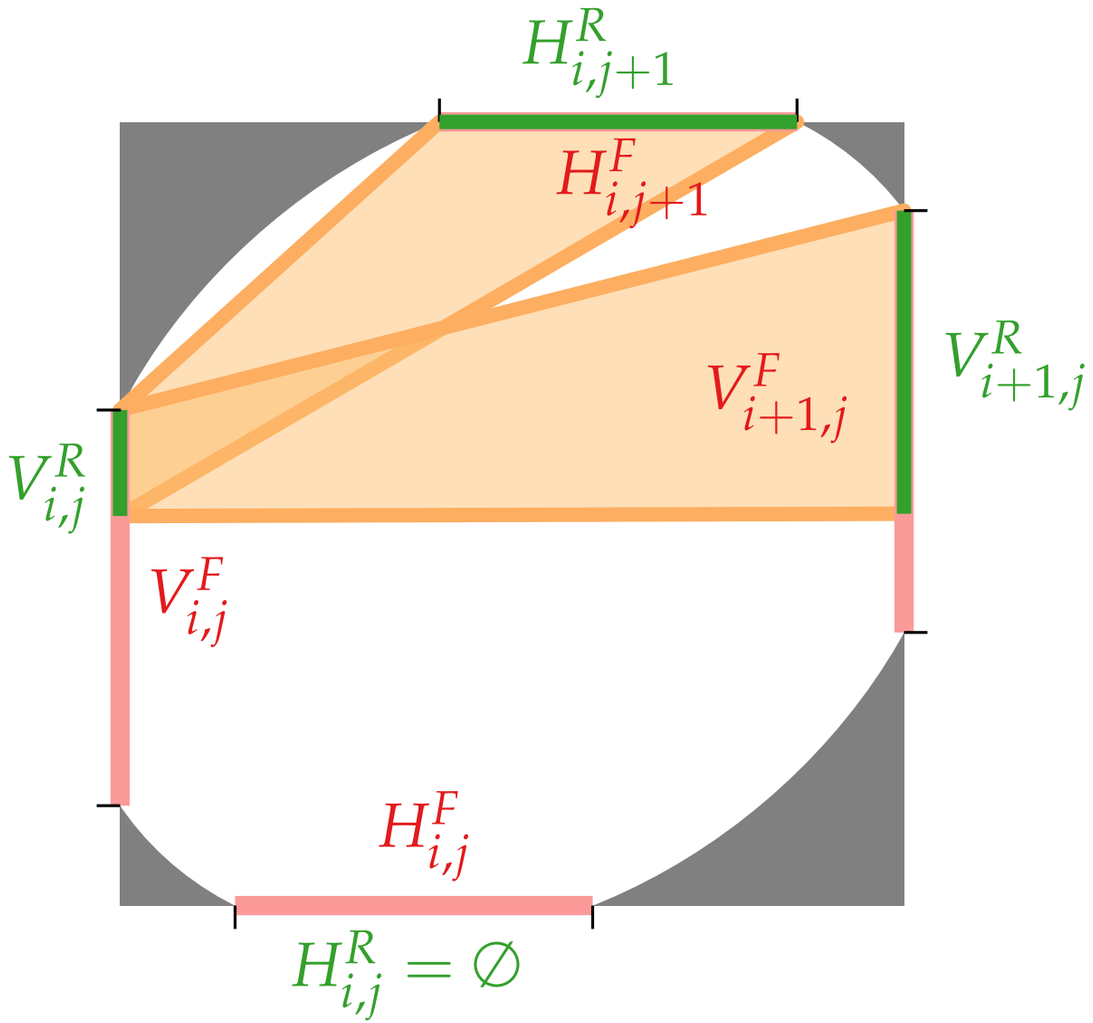


Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

$H^R_{i,j}$ und $V^R_{i,j}$ sind bekannt.

Monotonen Pfad finden

Kanten einer Zelle im Free-Space Diagram: $H^F_{i,j}$, $H^F_{i,j+1}$, $V^F_{i,j}$, $V^F_{i+1,j}$

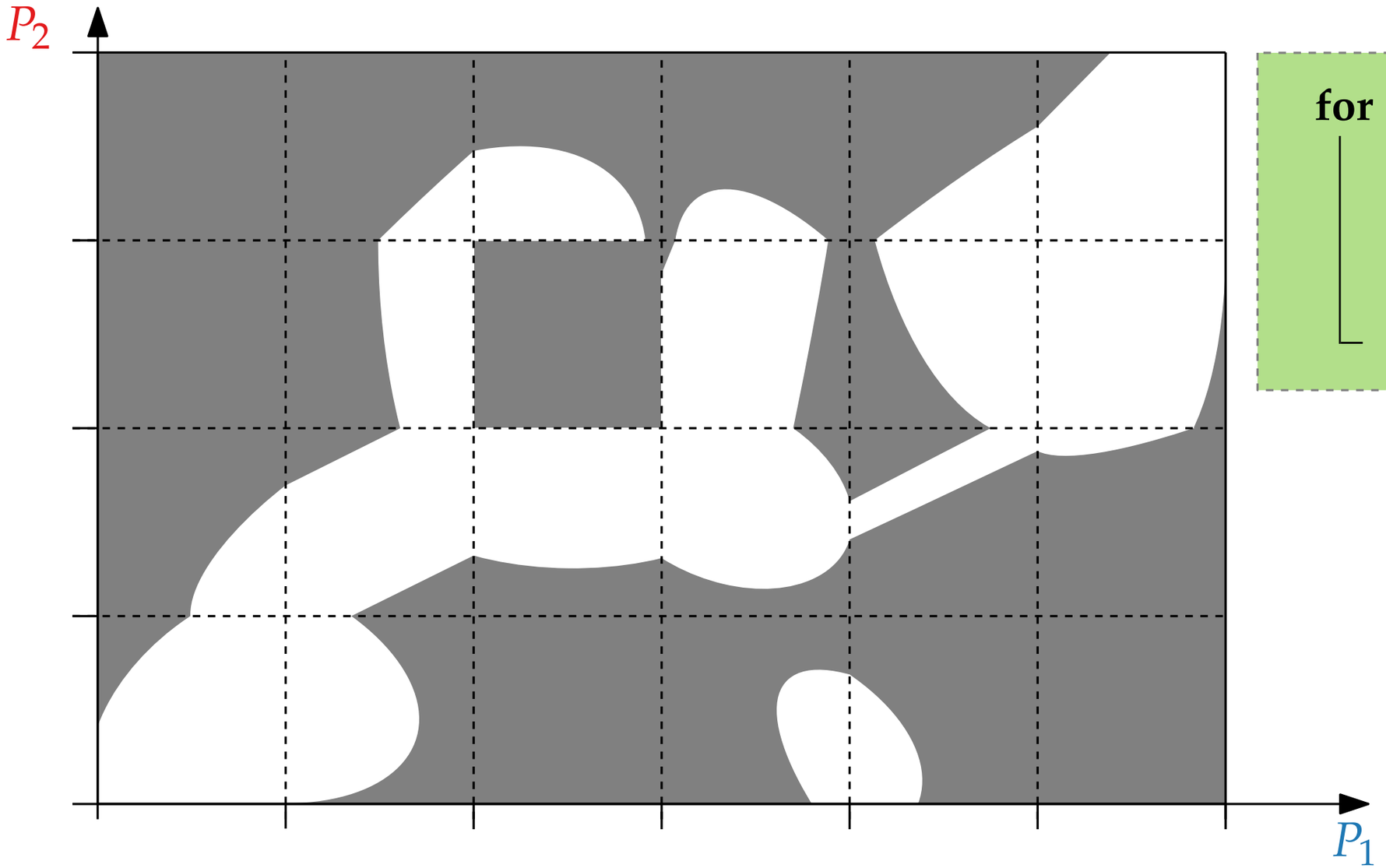


Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:
$H^R_{i,j}$, $H^R_{i,j+1}$, $V^R_{i,j}$, $V^R_{i+1,j}$

$H^R_{i,j}$ und $V^R_{i,j}$ sind bekannt.

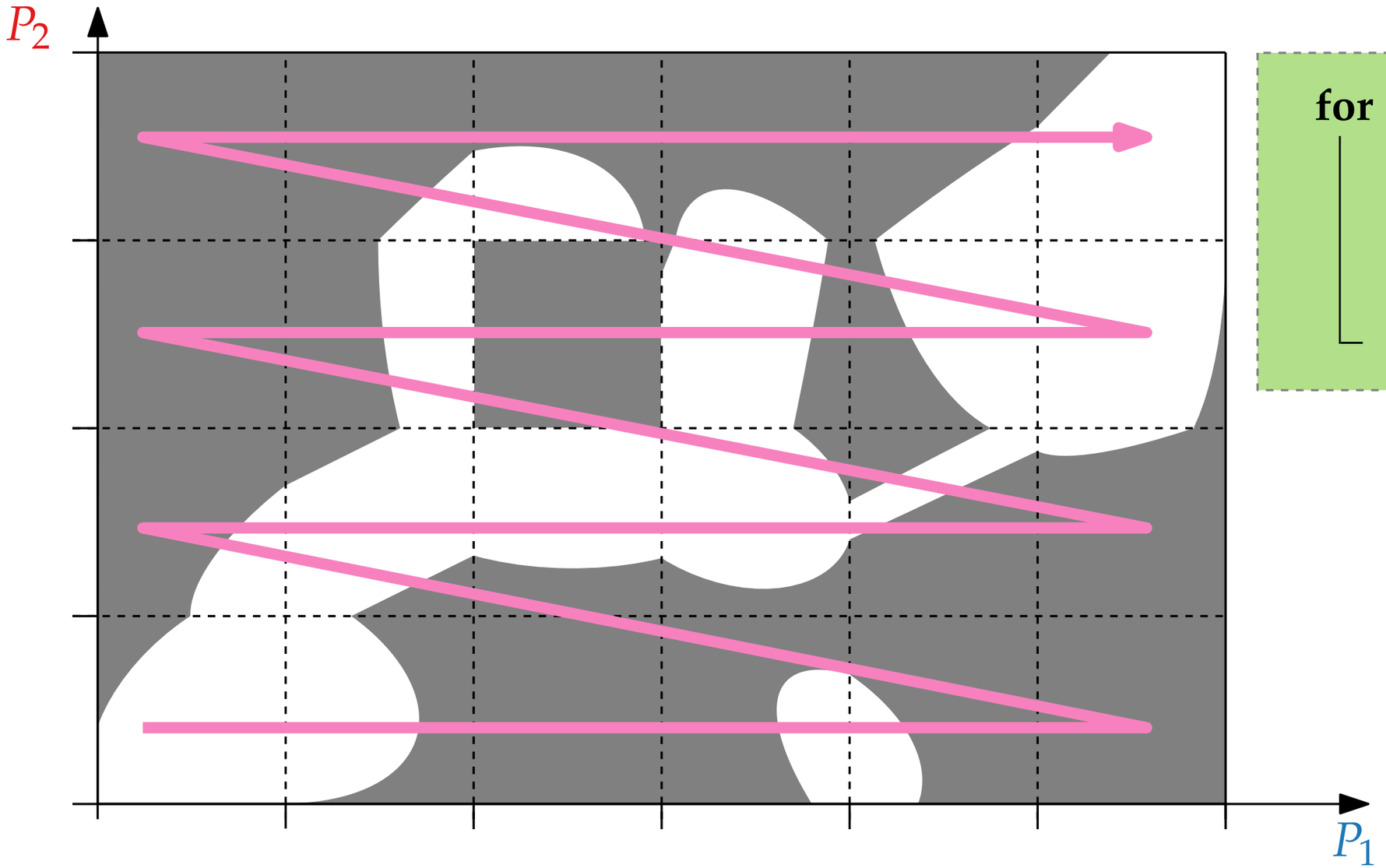
➡ $H^R_{i,j+1}$ und $V^R_{i+1,j}$ können berechnet werden.

Algorithmus



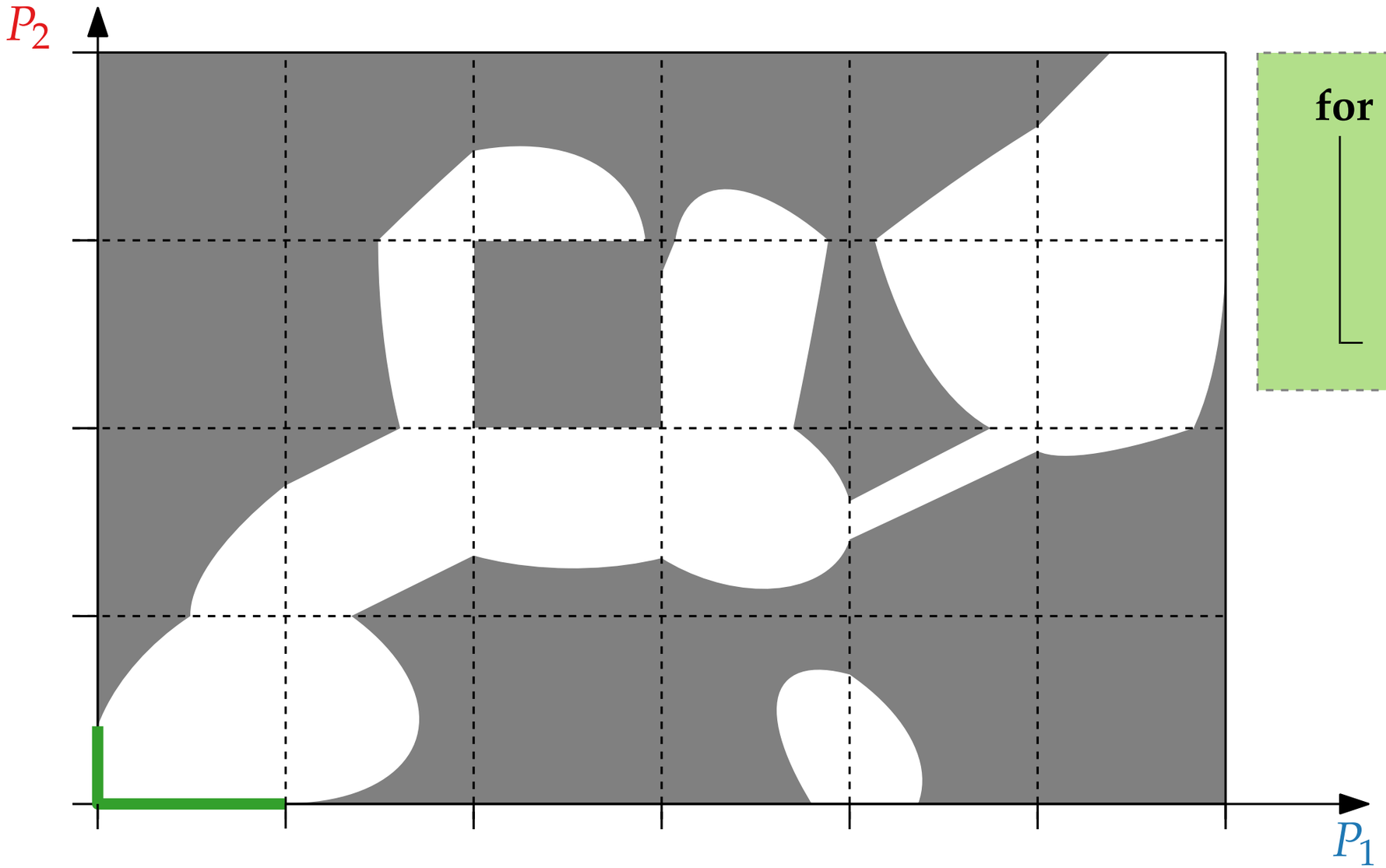
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



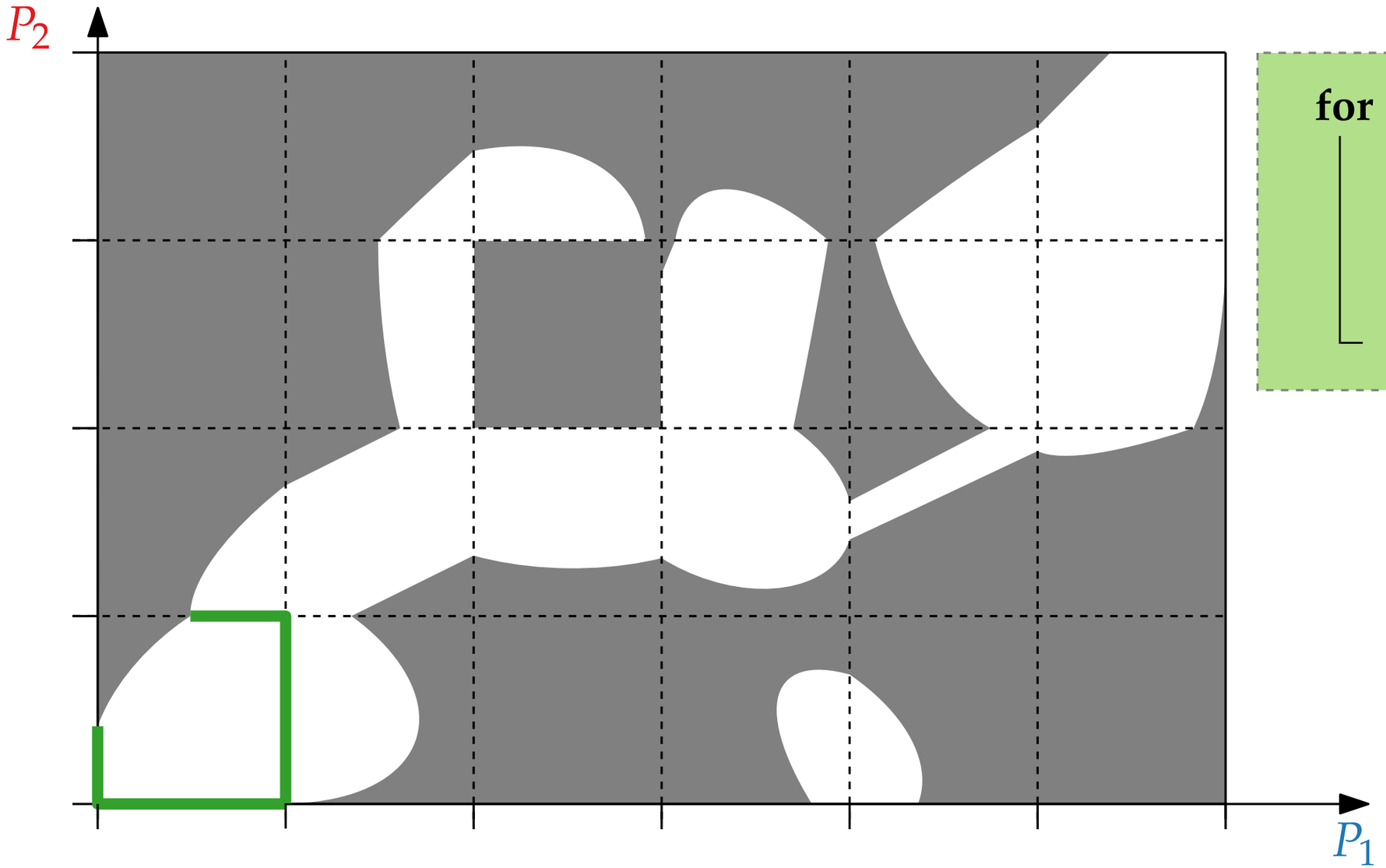
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



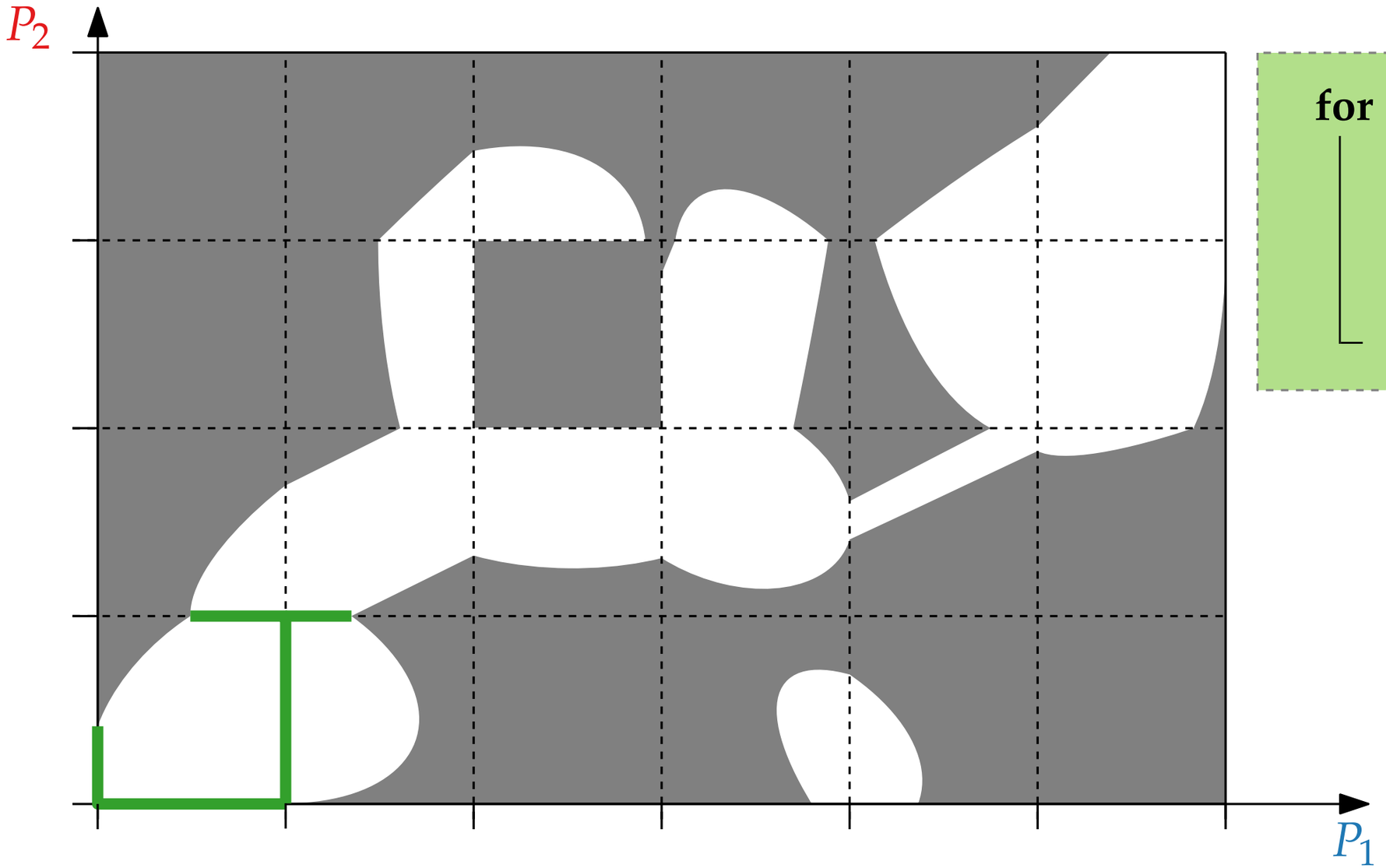
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



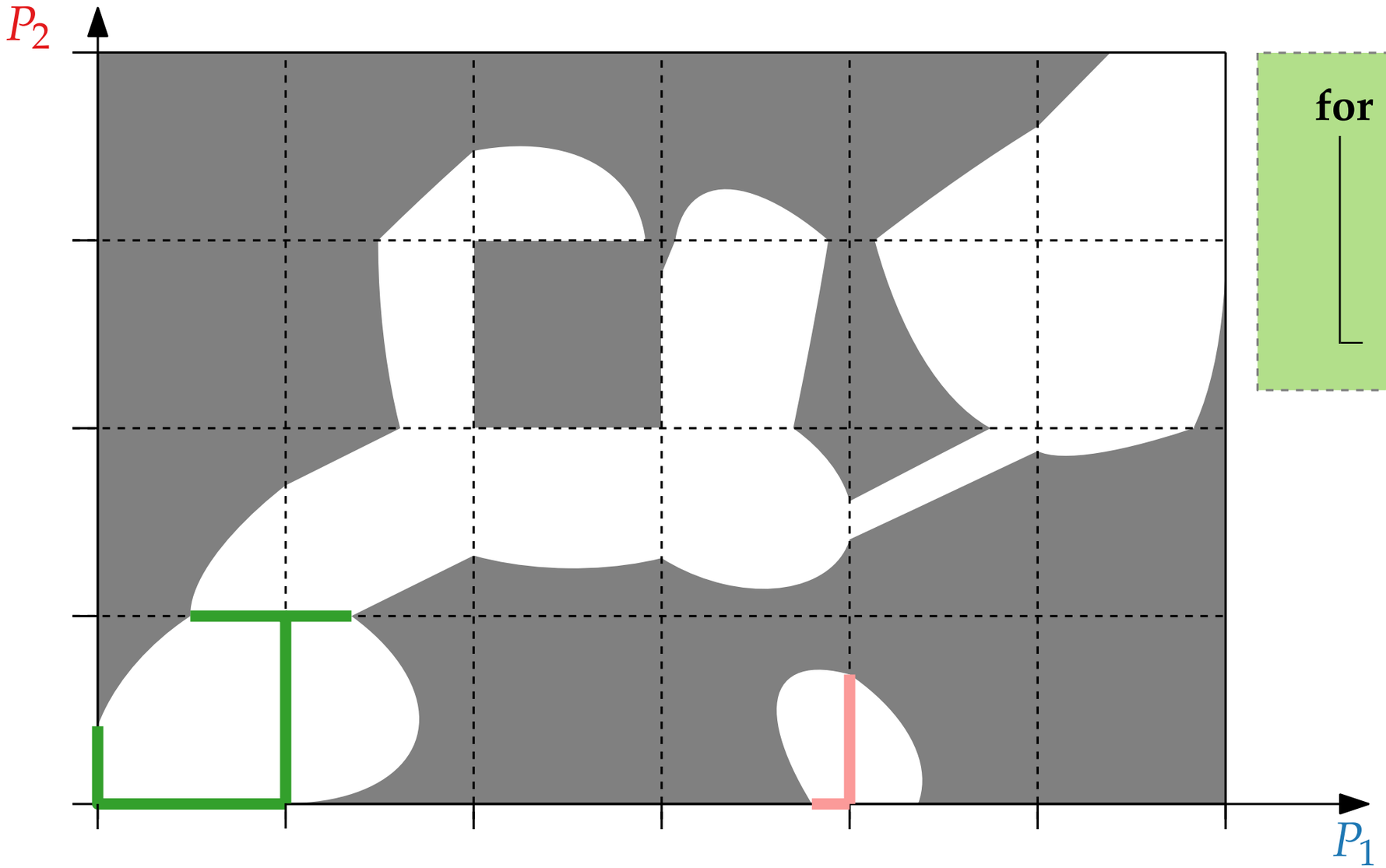
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



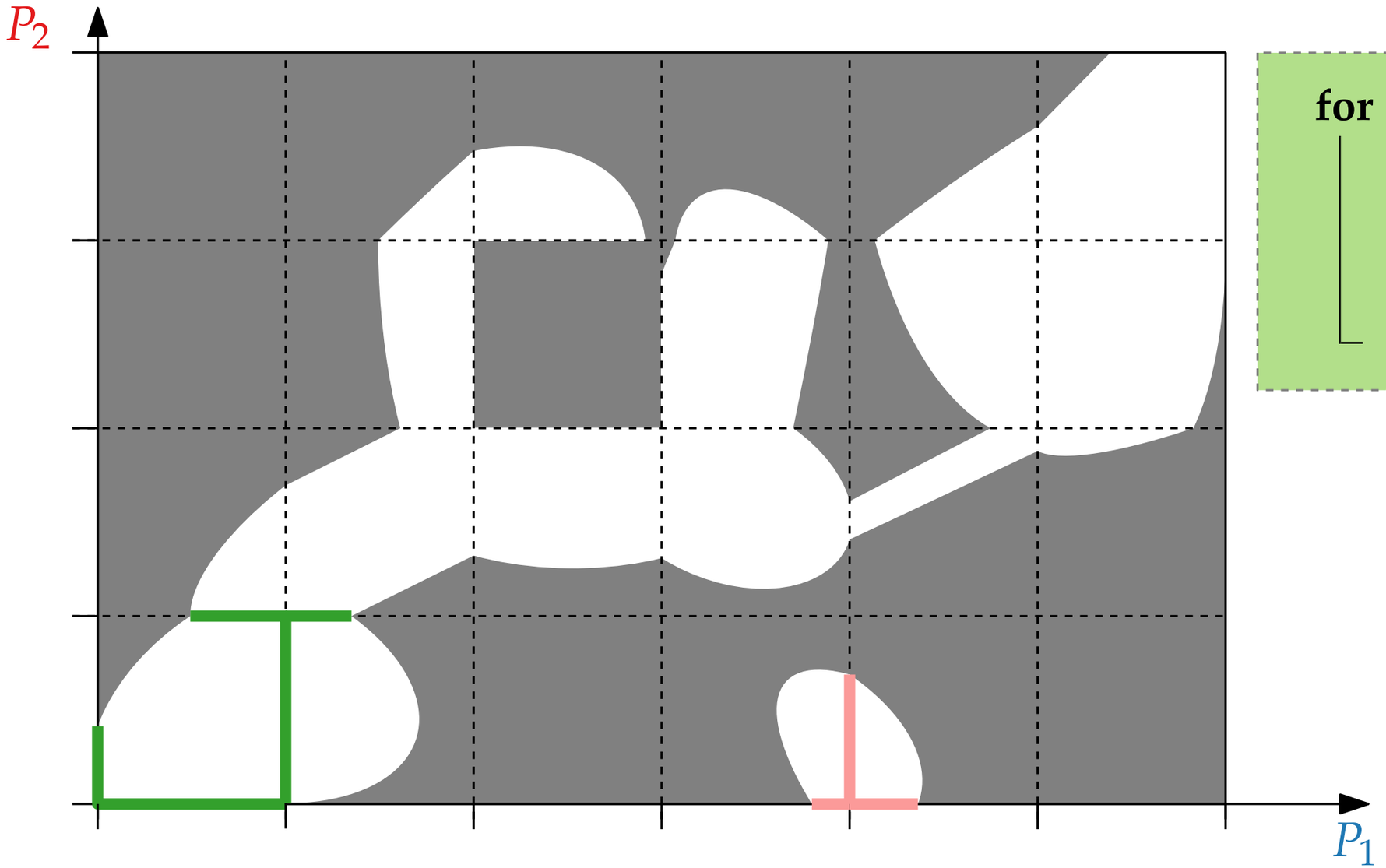
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



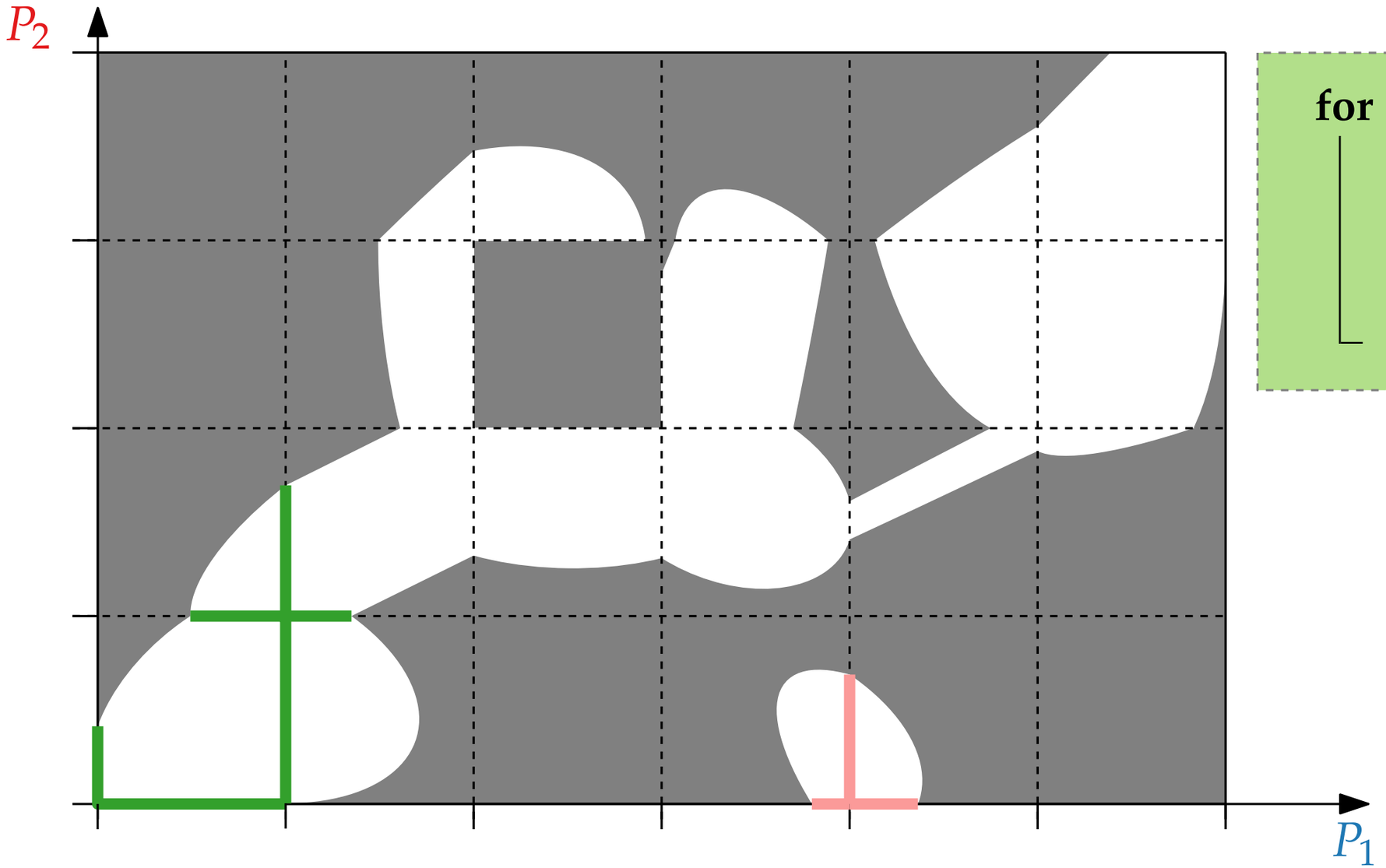
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



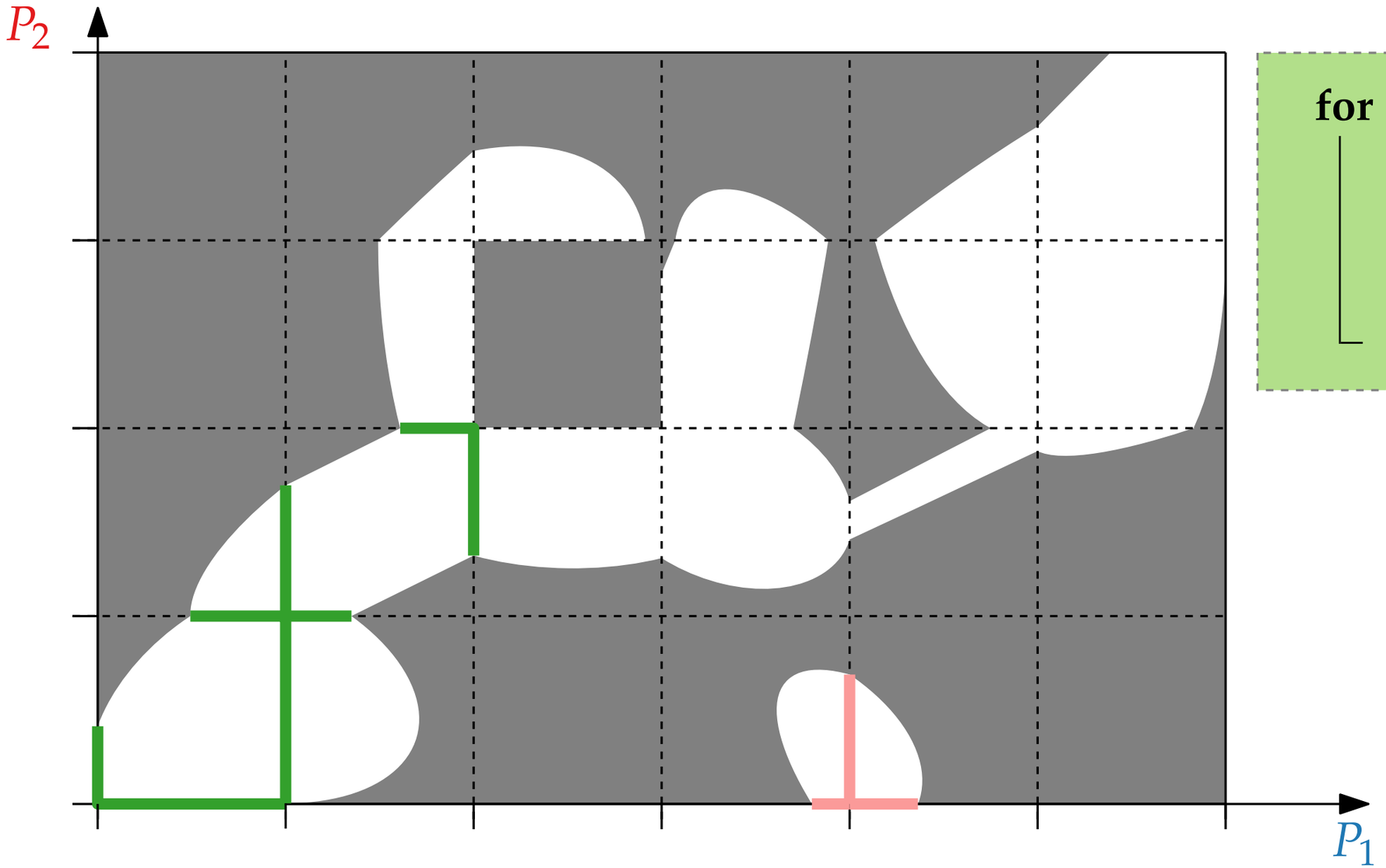
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



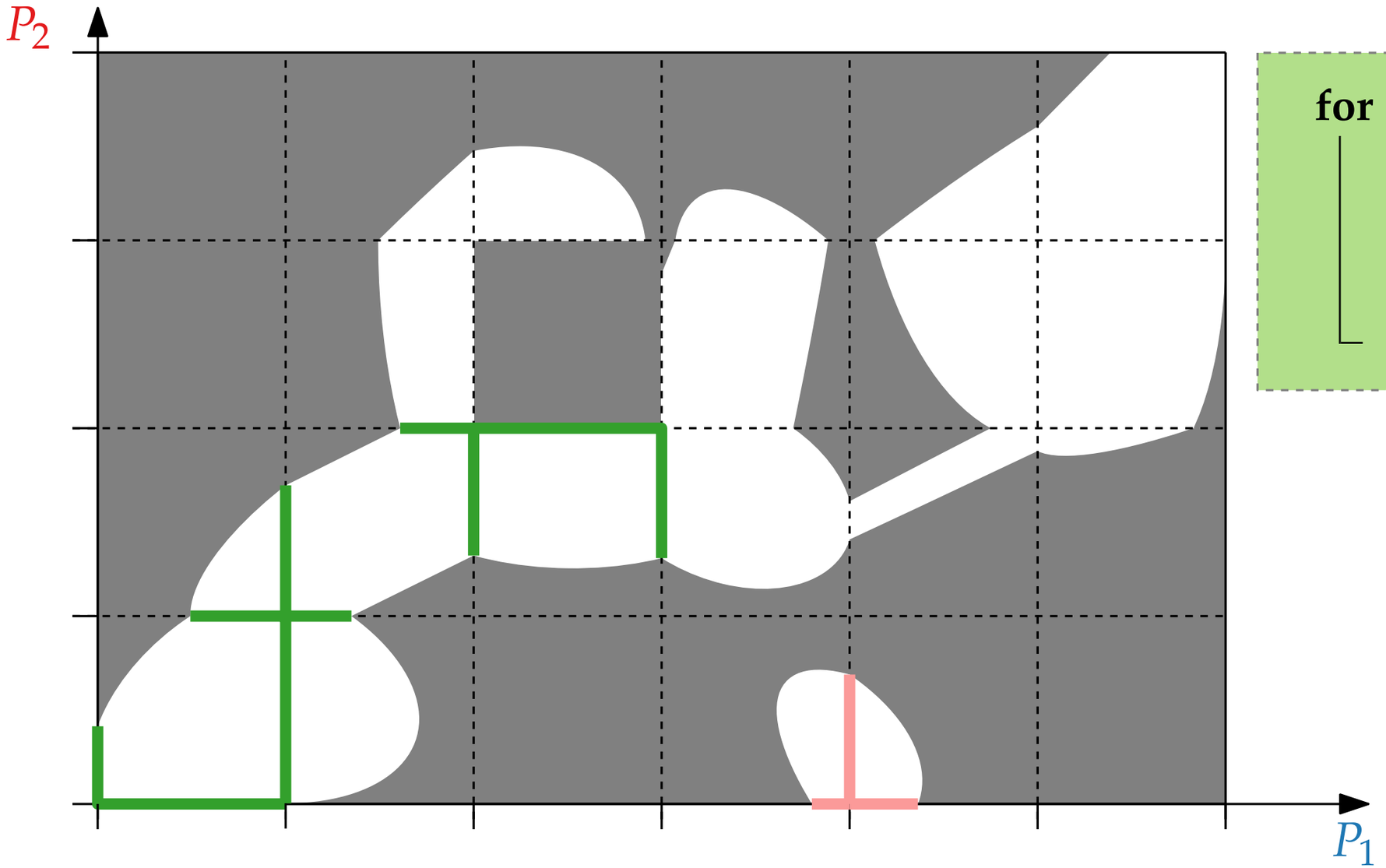
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



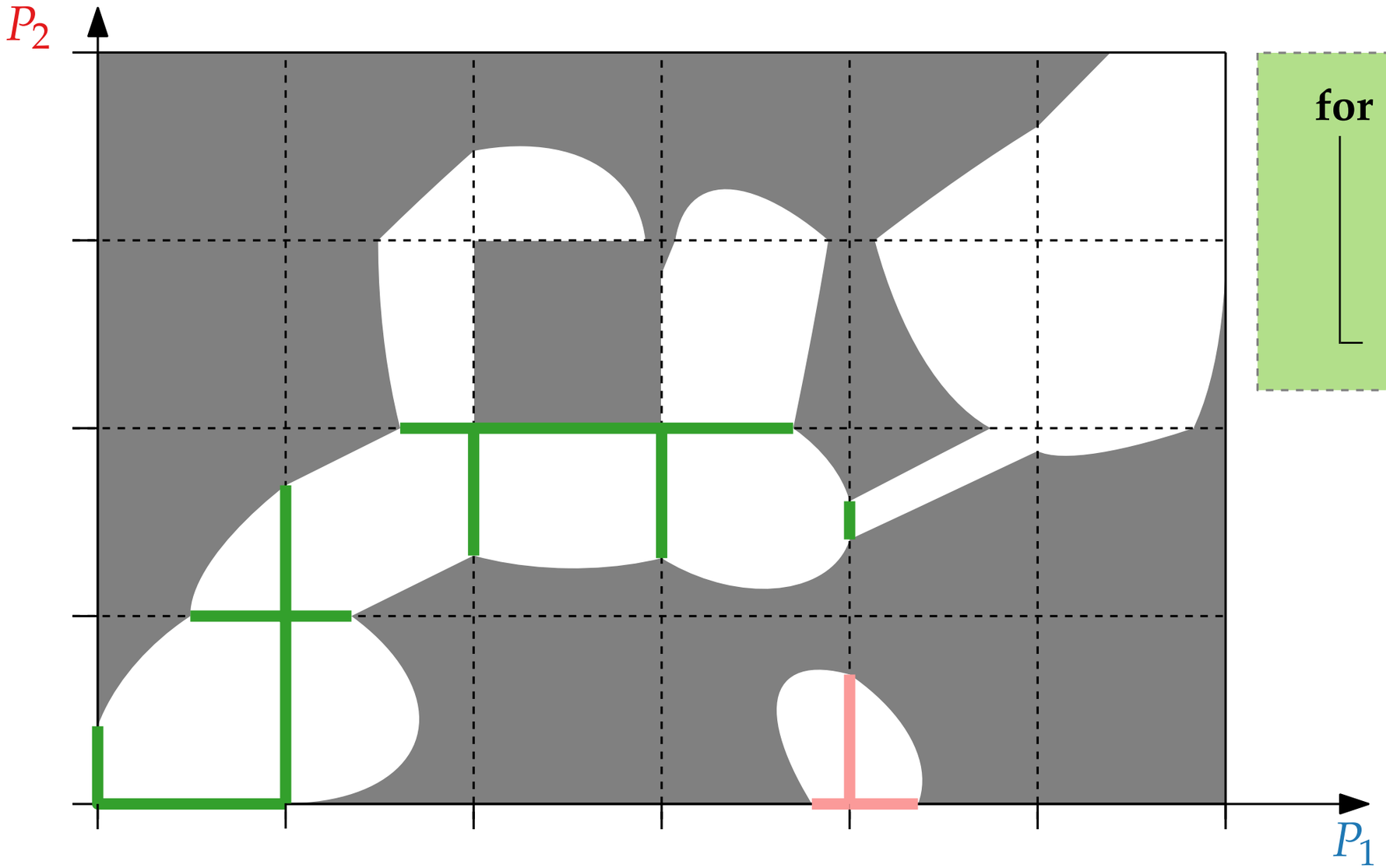
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



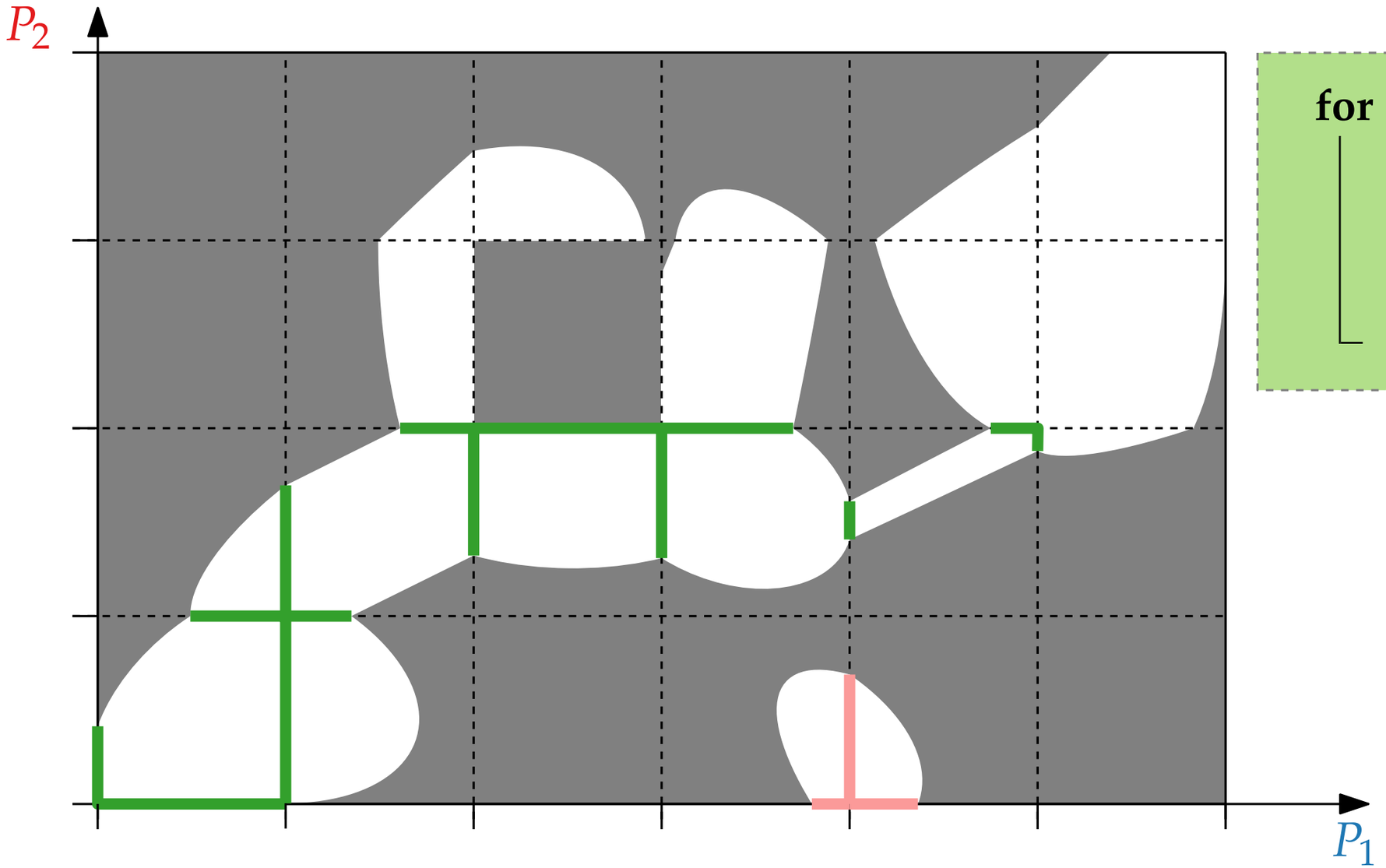
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



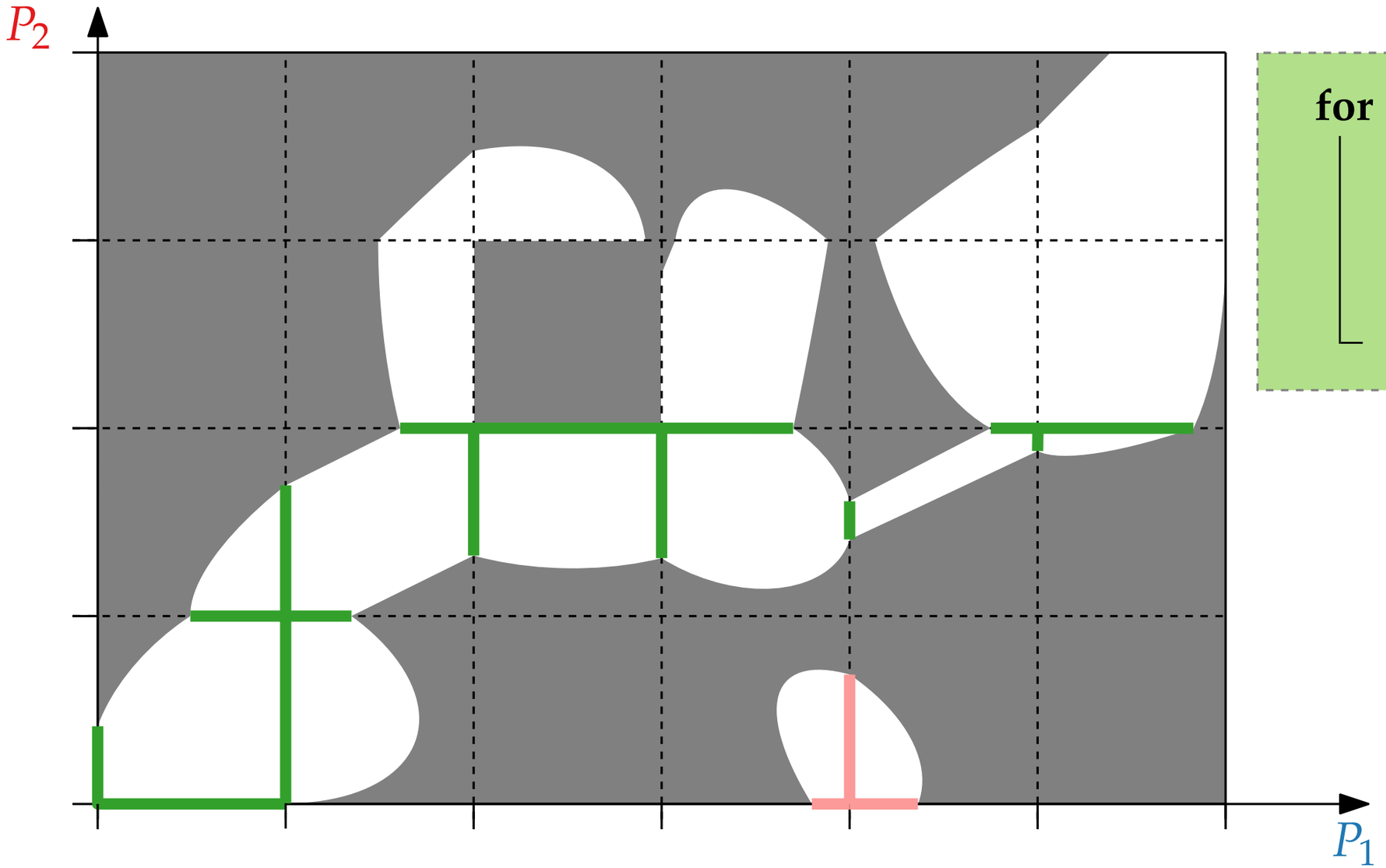
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



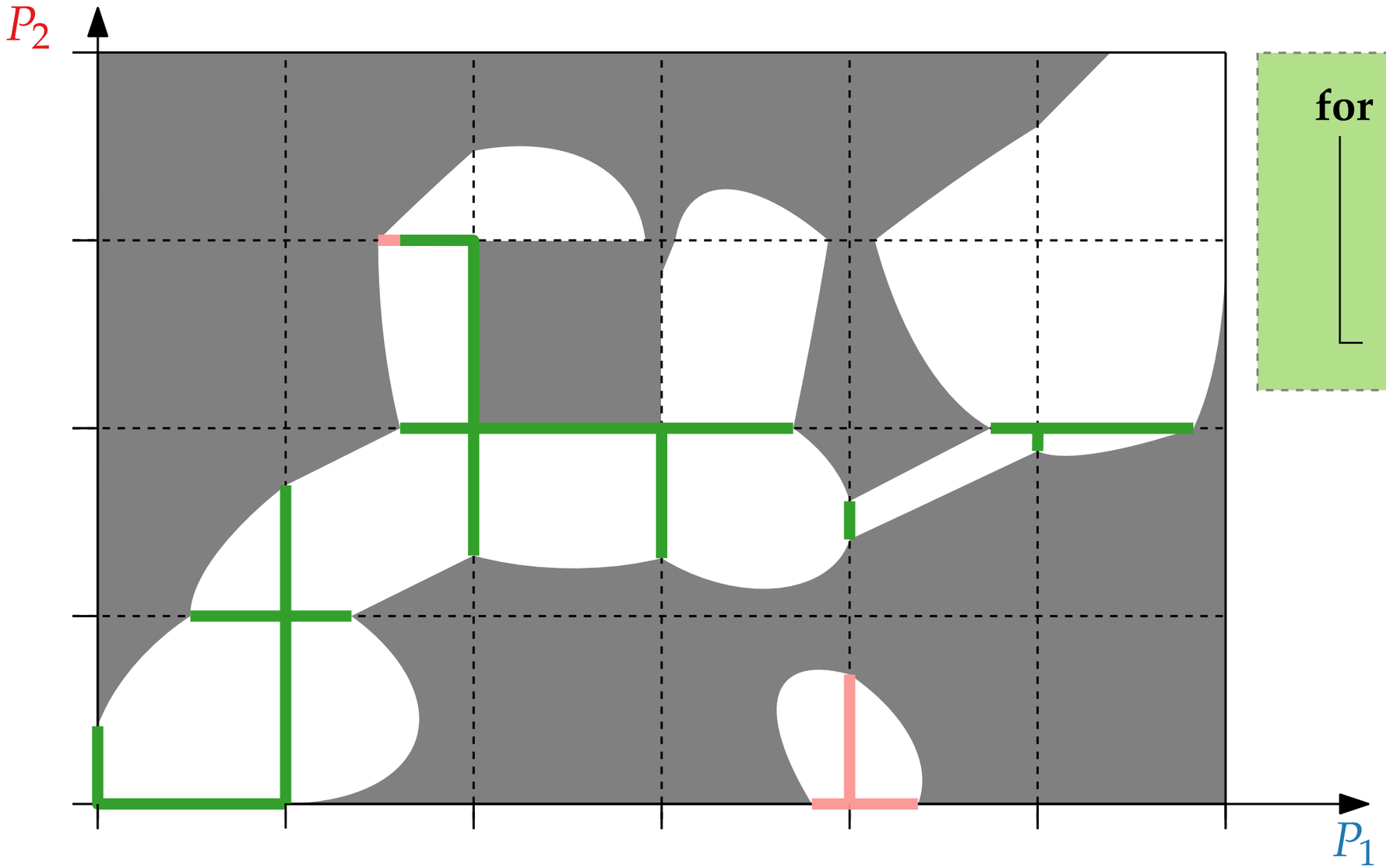
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



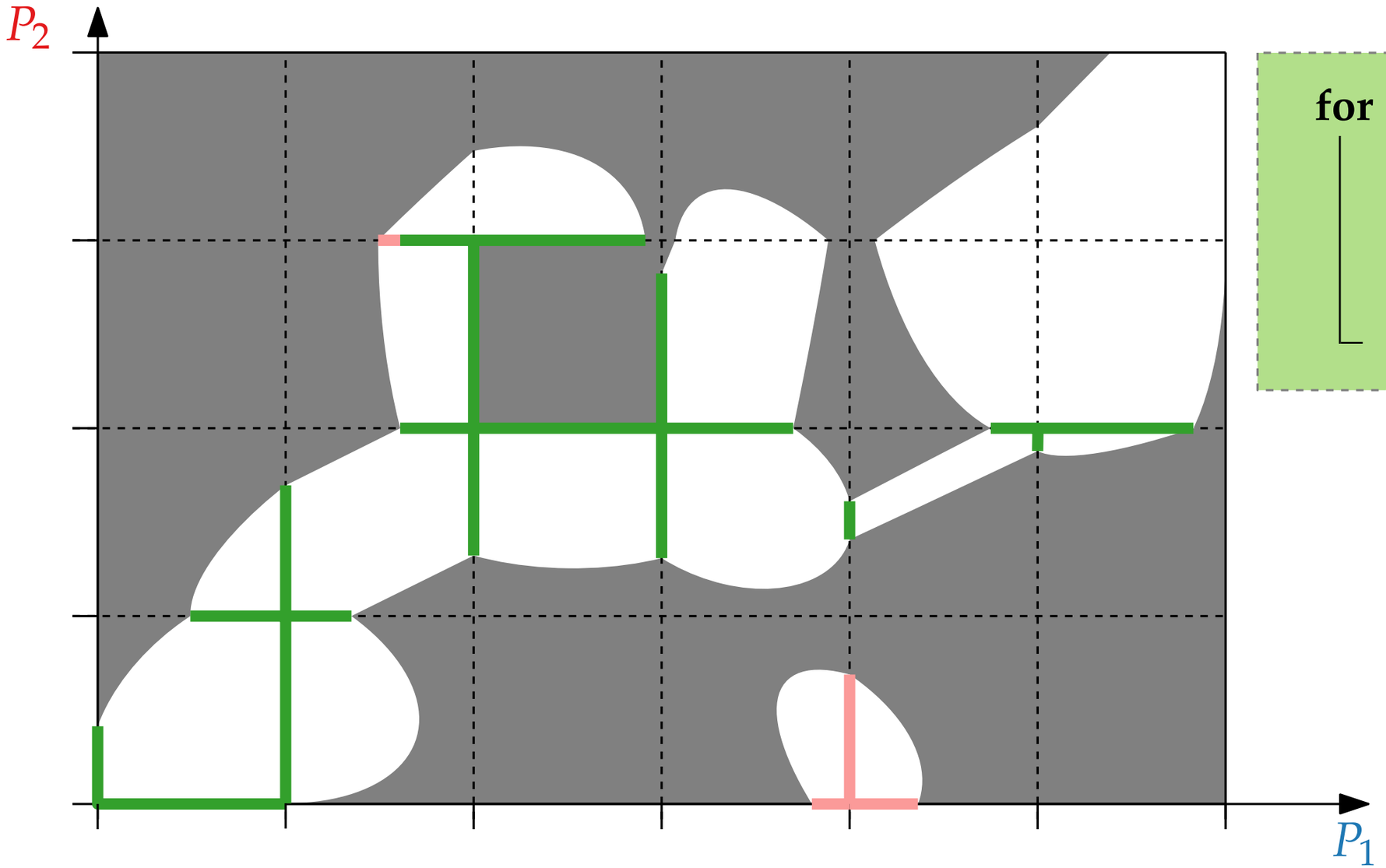
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



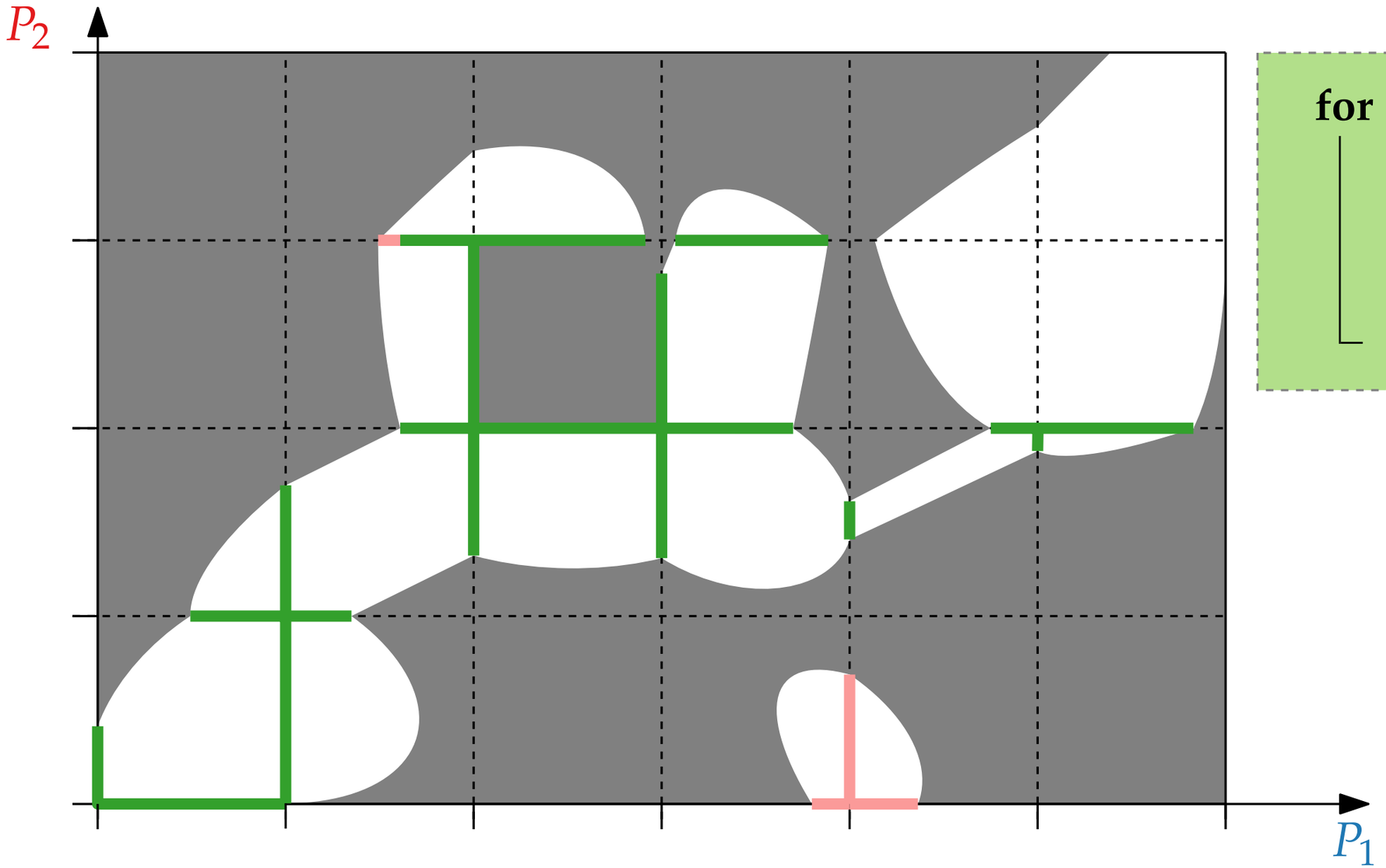
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



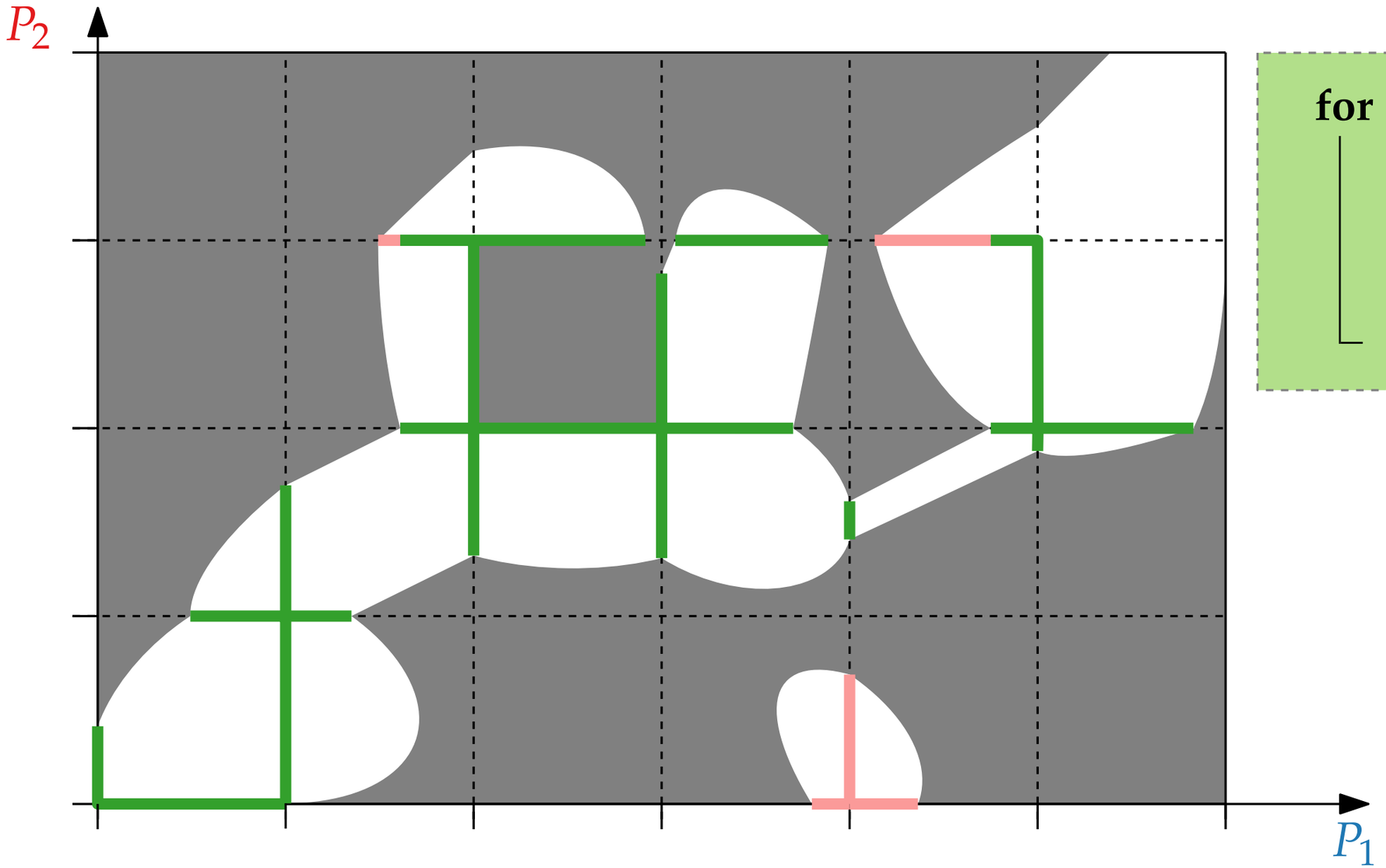
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



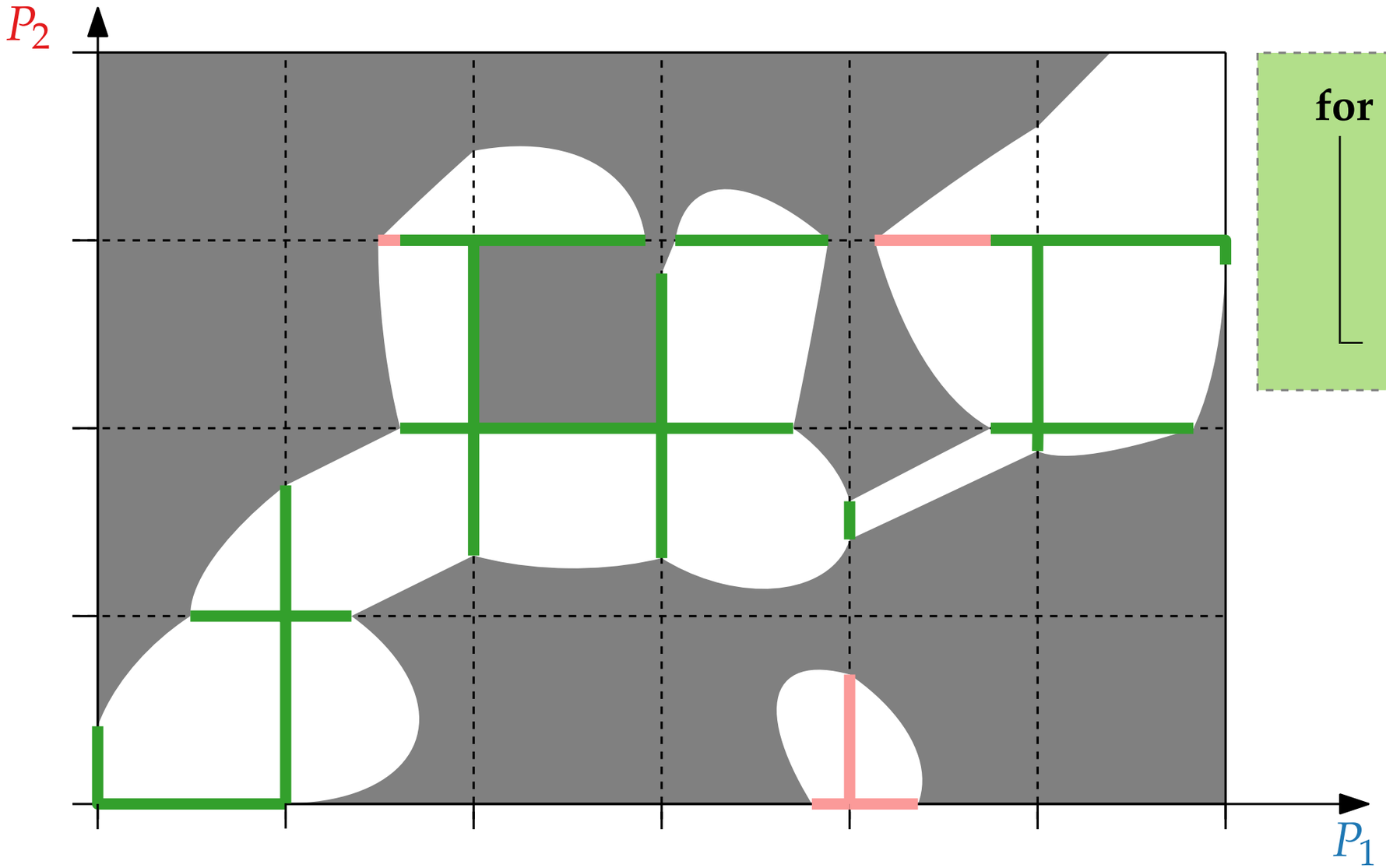
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



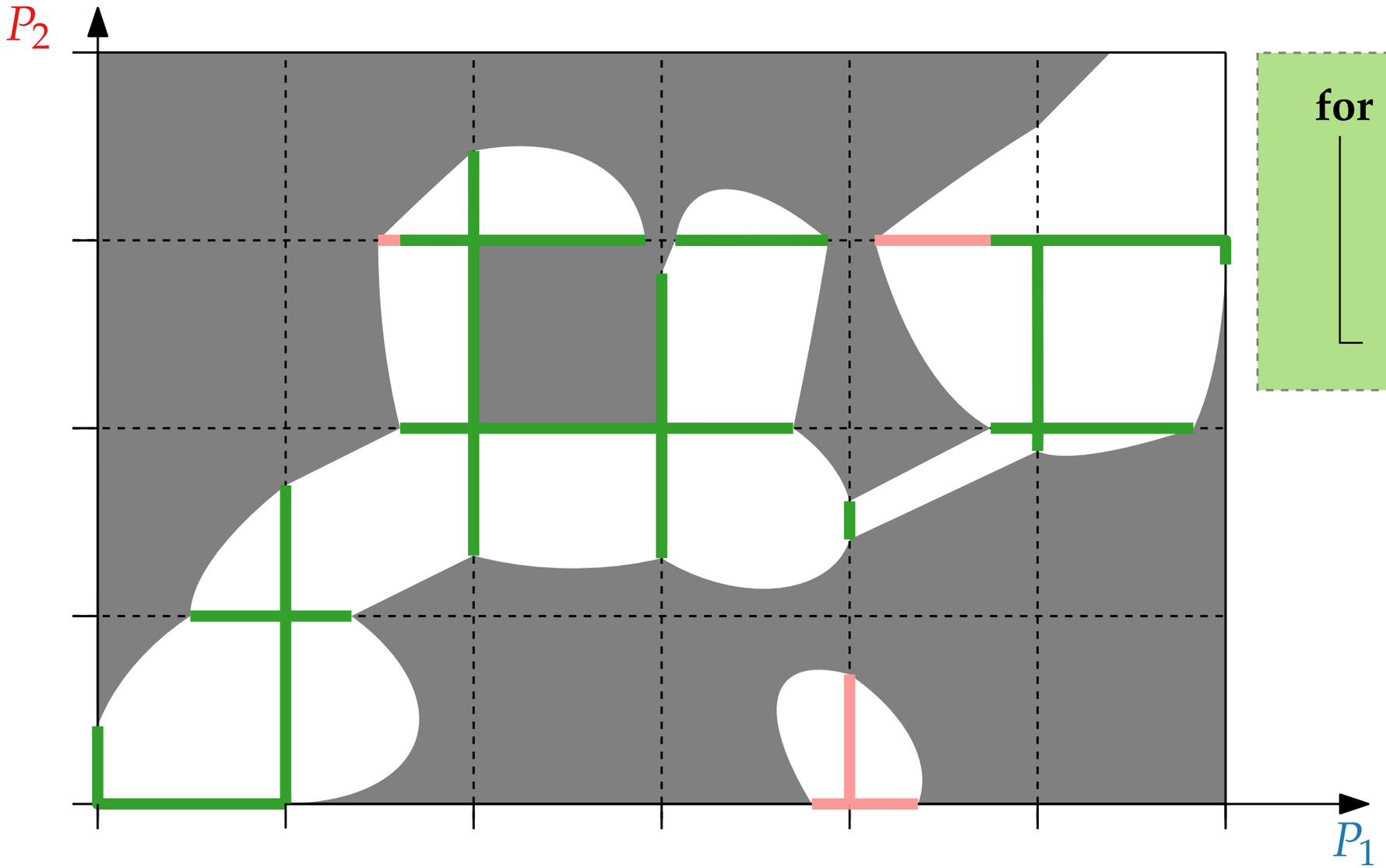
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



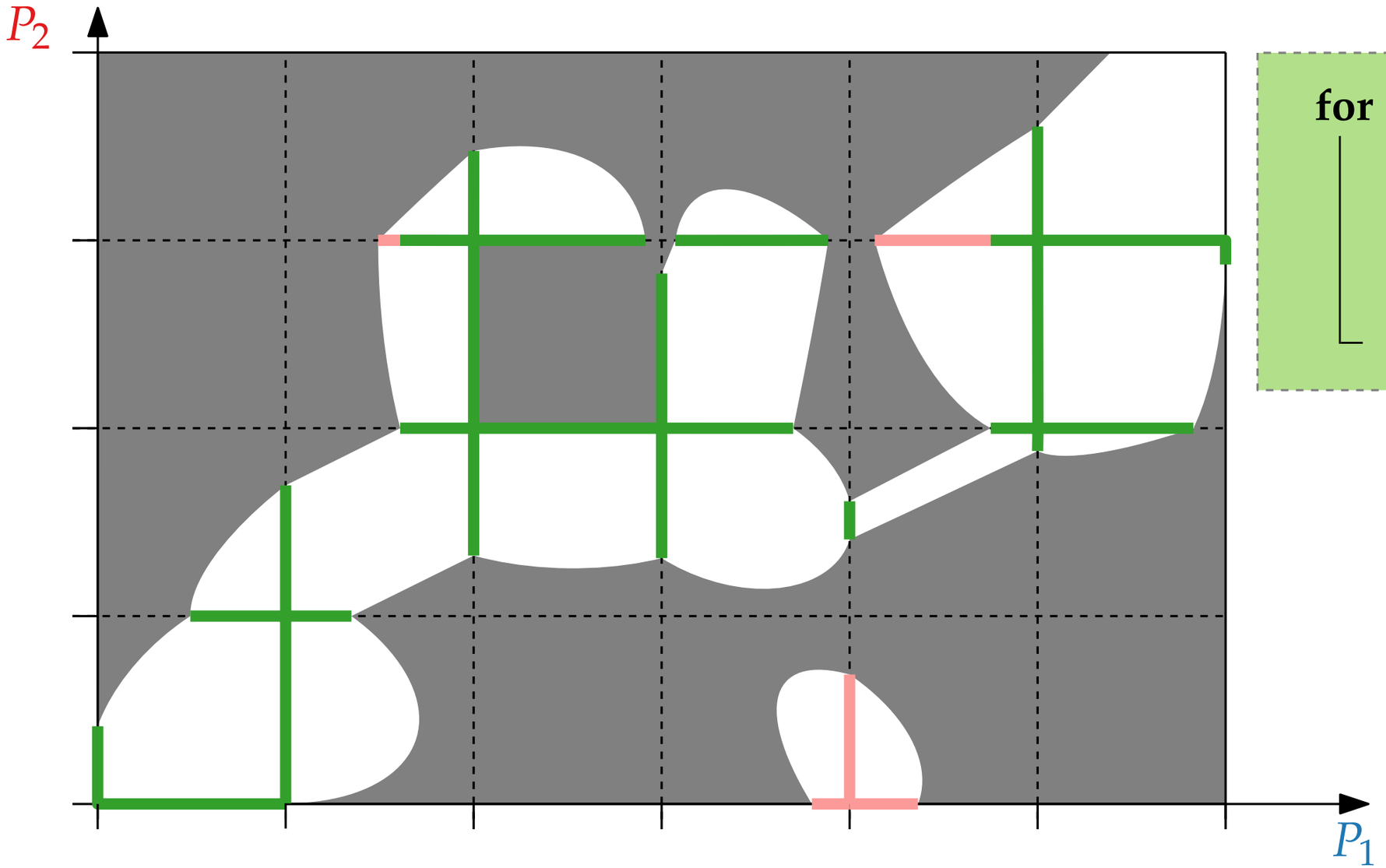
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



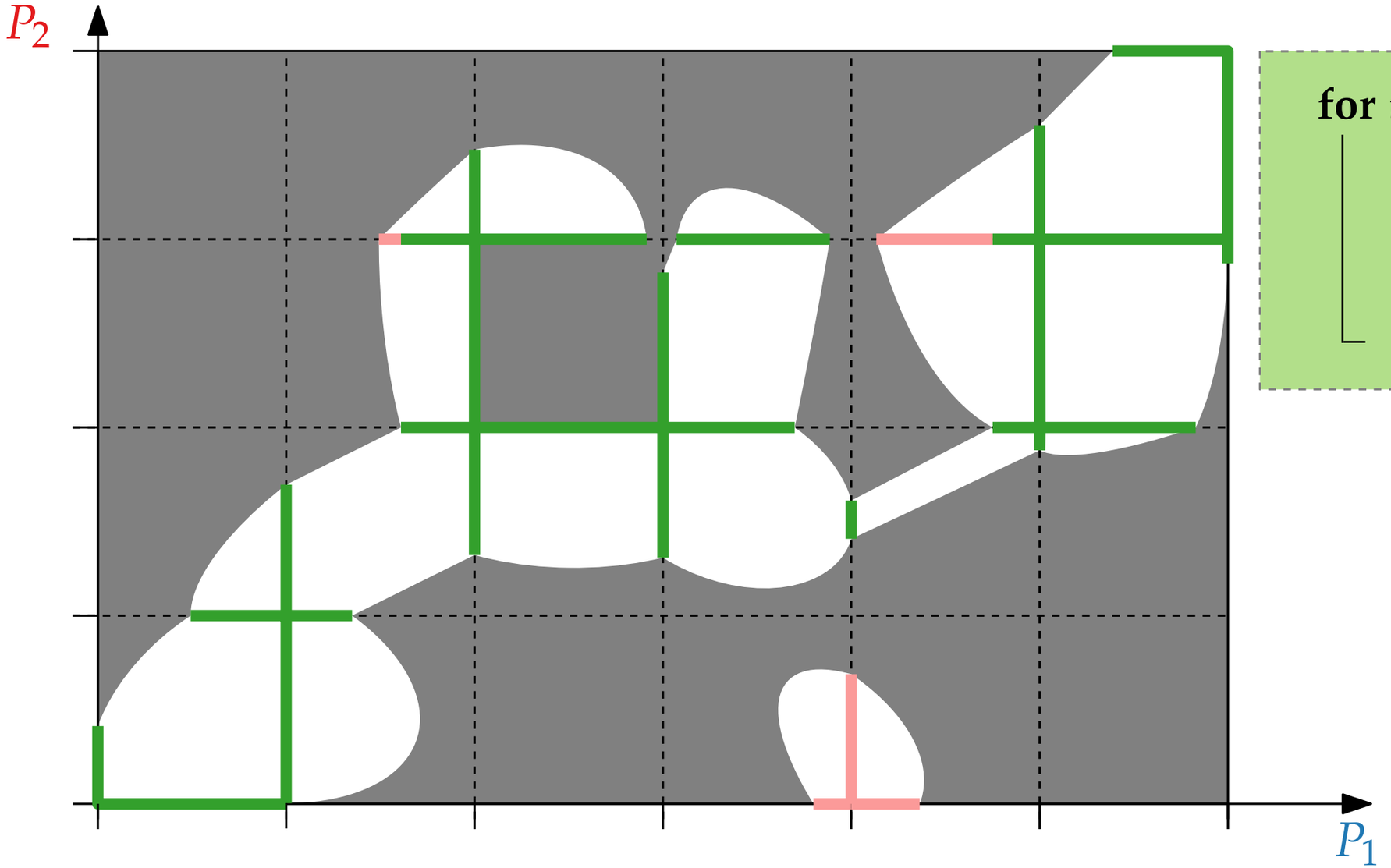
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



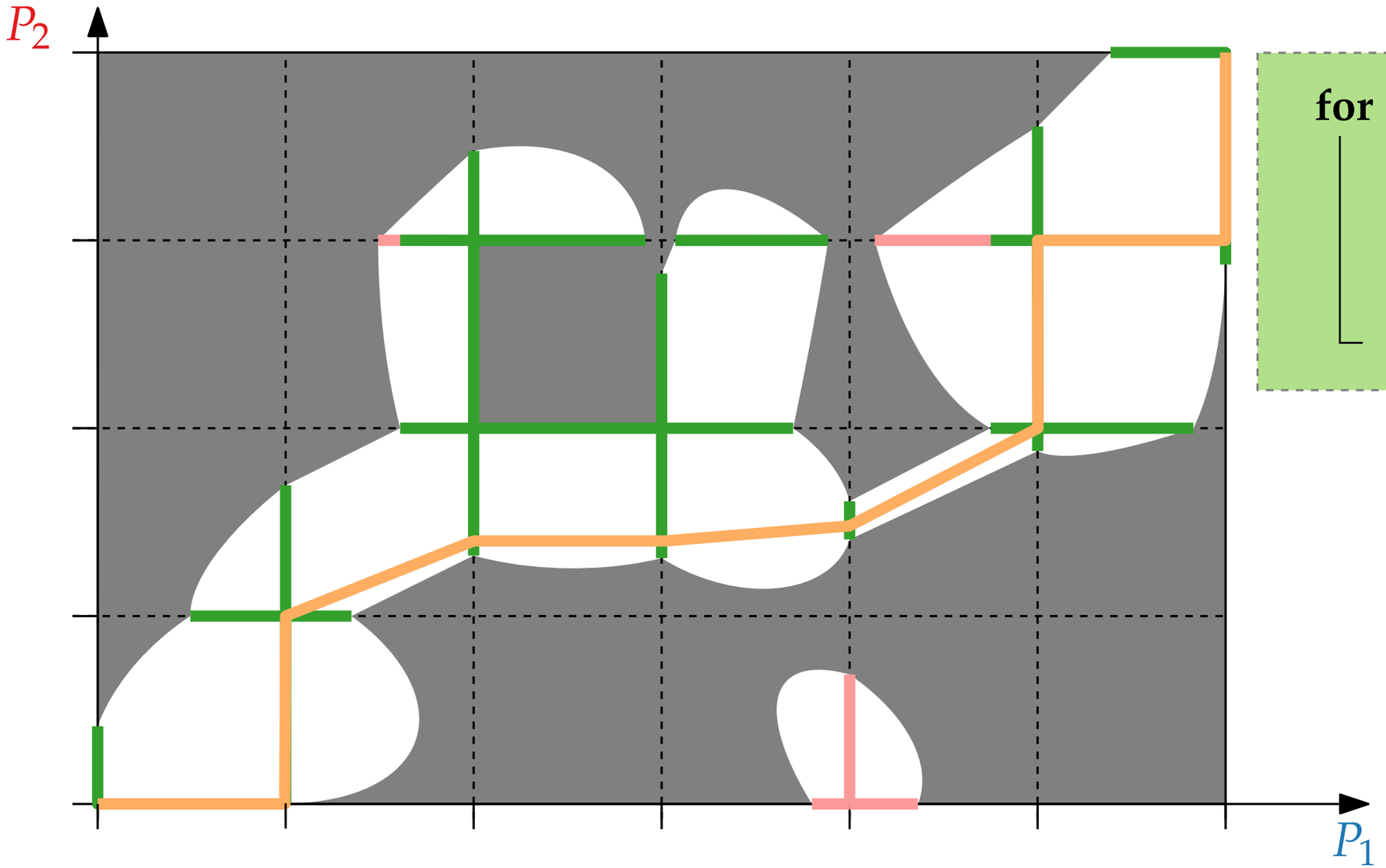
for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus



for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$
    und $H^R_{i,j+1}$

Algorithmus – Zusammenfassung

Für gegebenes ε:

Allgemein:

Algorithmus – Zusammenfassung

Für gegebenes ε:

- Berechne F_ε in $\mathcal{O}(n_1 n_2)$ Zeit. (Jedes Segmentpaar benötigt $\mathcal{O}(1)$ Zeit.)

Allgemein:

Algorithmus – Zusammenfassung

Für gegebenes ε:

- Berechne F_ε in $\mathcal{O}(n_1 n_2)$ Zeit. (Jedes Segmentpaar benötigt $\mathcal{O}(1)$ Zeit.)
- Berechne monotonen Pfad in F_ε in $\mathcal{O}(n_1 n_2)$ Zeit:

for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$ und $H^R_{i,j+1}$

Allgemein:

Algorithmus – Zusammenfassung

Für gegebenes ε:

- Berechne F_ε in $\mathcal{O}(n_1 n_2)$ Zeit. (Jedes Segmentpaar benötigt $\mathcal{O}(1)$ Zeit.)
- Berechne monotonen Pfad in F_ε in $\mathcal{O}(n_1 n_2)$ Zeit:

for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$ und $H^R_{i,j+1}$

Allgemein:

- Verwende parametrisierte Suche, um kleinstes gültiges ε zu finden

Algorithmus – Zusammenfassung

Für gegebenes ε:

- Berechne F_ε in $\mathcal{O}(n_1 n_2)$ Zeit. (Jedes Segmentpaar benötigt $\mathcal{O}(1)$ Zeit.)
- Berechne monotonen Pfad in F_ε in $\mathcal{O}(n_1 n_2)$ Zeit:

for $i = 0$ **to** $n_1 - 1$ **do**
  for $j = 0$ **to** $n_2 - 1$ **do**
    Berechne $V^R_{i+1,j}$ und $H^R_{i,j+1}$

Allgemein:

- Verwende parametrisierte Suche, um kleinstes gültiges ε zu finden
- Logarithmische Suche ➡ $\mathcal{O}(n_1 n_2 \log n_1 n_2)$ Zeit insgesamt.