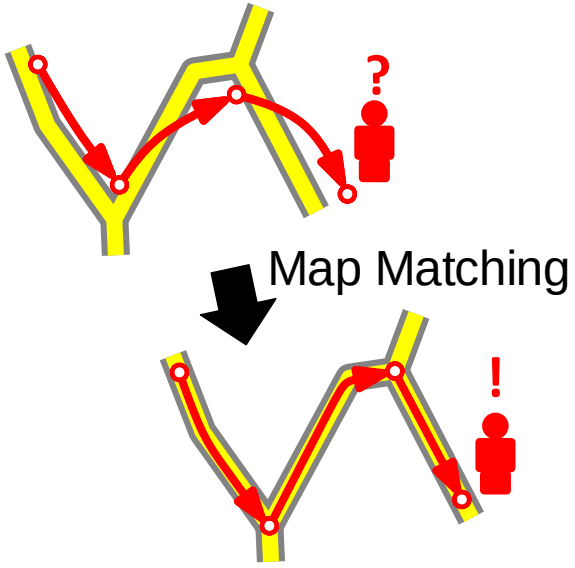
Algorithmen für Geographische Informationssysteme

Thomas van Dijk

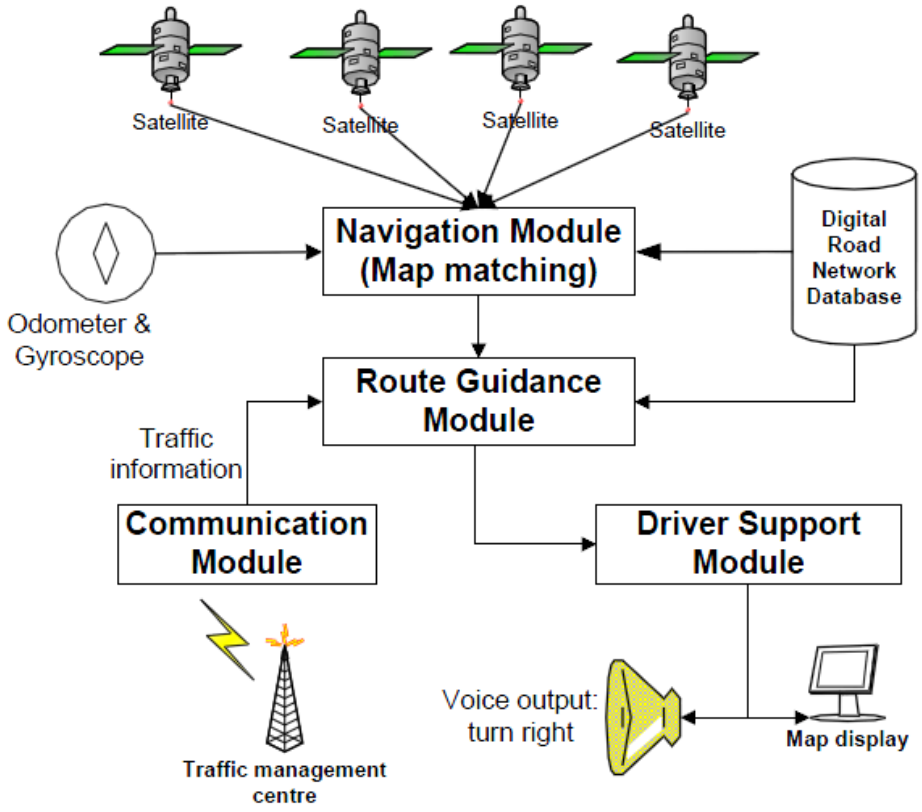
basiert auf Folien von Jan-Henrik Haunert

Map Matching



Map Matching

...als Teil von Fahrzeugnavigationssystemen



Bildquelle: Quddus 2006

Map Matching

Schwierigkeit:

- ungenaue Beobachtungen

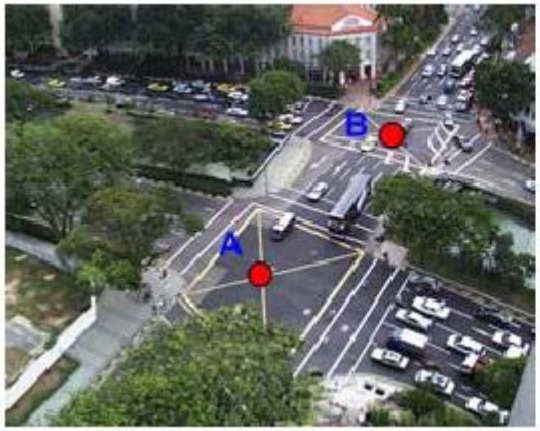
Bildquelle: Wikipedia

GPS: Positionsgenauigkeit ca. 20 m

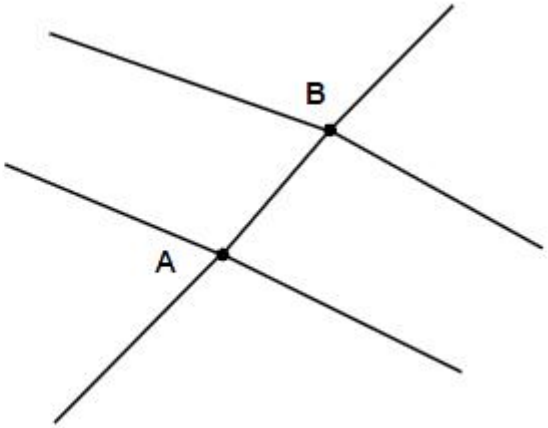
Map Matching

Schwierigkeit:

- ungenaue Beobachtungen
- abstrahierte/vereinfachte Repräsentation des Straßennetzes



tatsächliche Straßenkreuzung



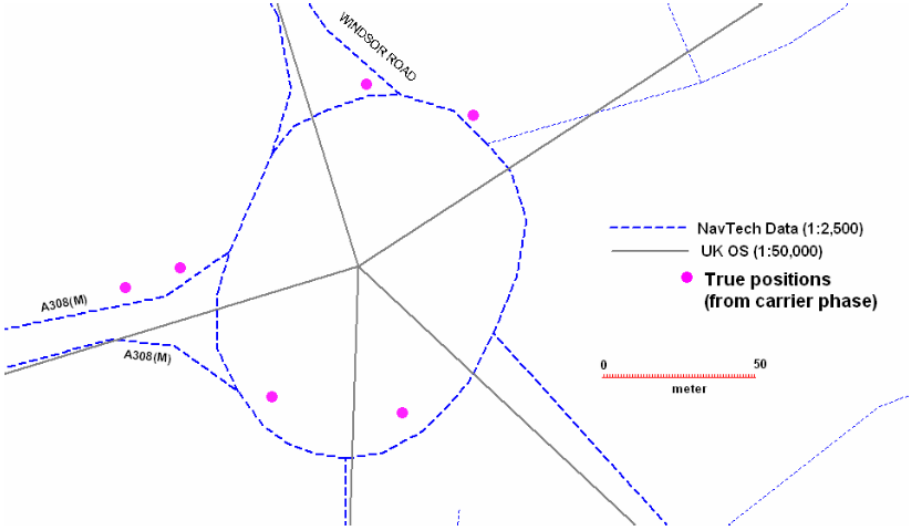
Repräsentation als Graph

Bildquelle: Quddus 2006

Map Matching

Schwierigkeit:

- ungenaue Beobachtungen
- abstrahierte/vereinfachte Repräsentation des Straßennetzes



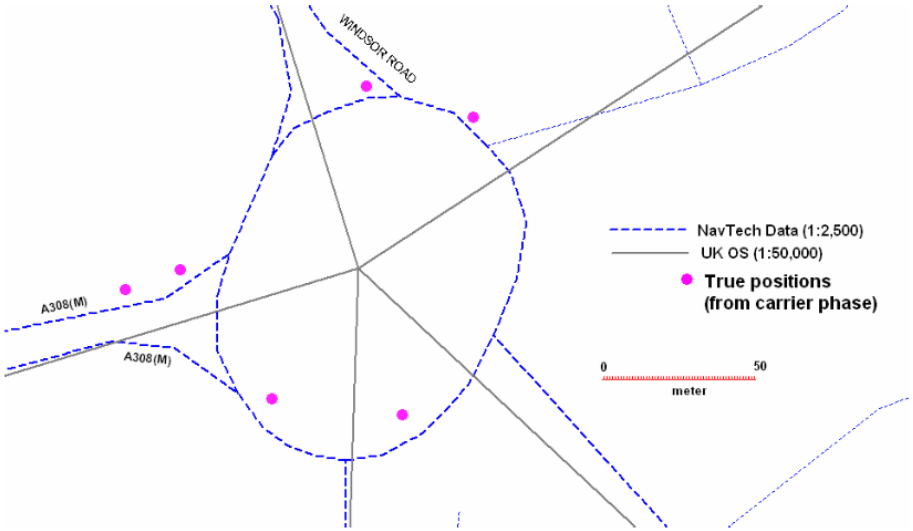
Unterschiede zweier Datensätze

Bildquelle: Quddus 2006

Map Matching

Aufgabenteile:

- Auf welcher Kante befindet sich das Fahrzeug?
- Wo auf der Kante befindet sich das Fahrzeug?

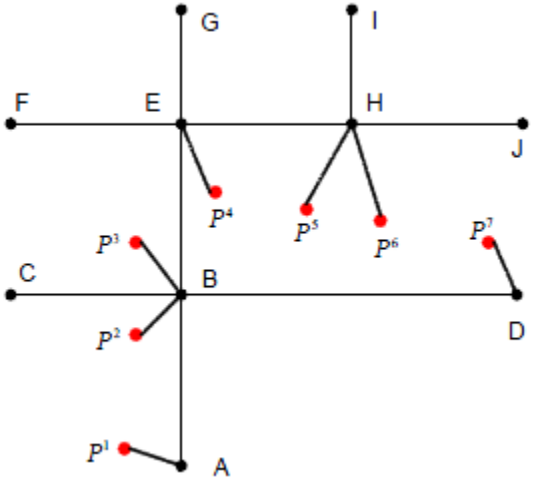


Unterschiede zweier Datensätze

Bildquelle: Quddus 2006

Map Matching

getrennt für jede Positionierung



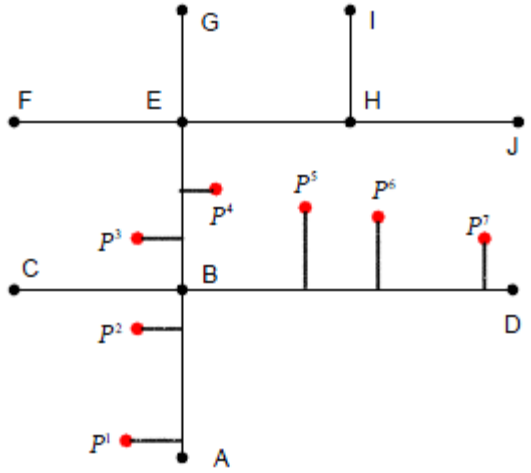
Punkt-zu-Punkt-Zuordnung

Algorithmus?

Bildquelle: Quddus 2006

Map Matching

getrennt für jede Positionierung



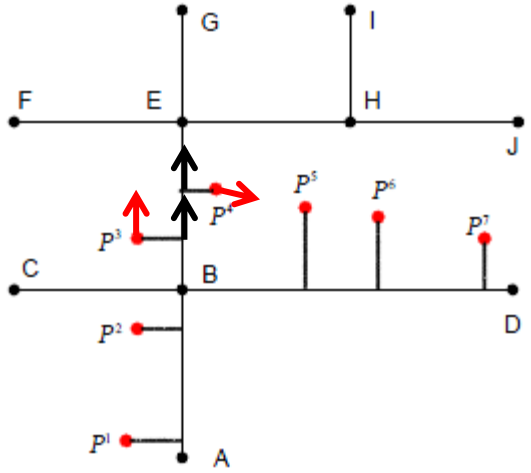
Punkt-zu-Kante-Zuordnung

Bildquelle: Quddus 2006

Map Matching

getrennt für jede Positionierung

weitere Verbesserung:
Vergleiche Fahrtrichtung
mit Kantenrichtung



Punkt-zu-Kante-Zuordnung

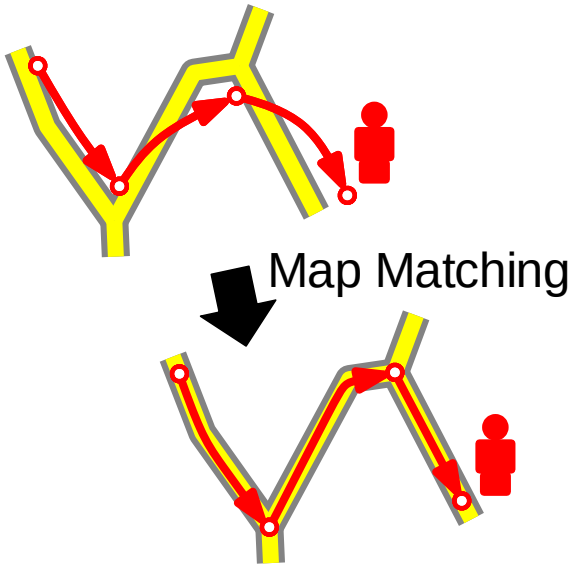
Bildquelle: Quddus 2006

Map Matching

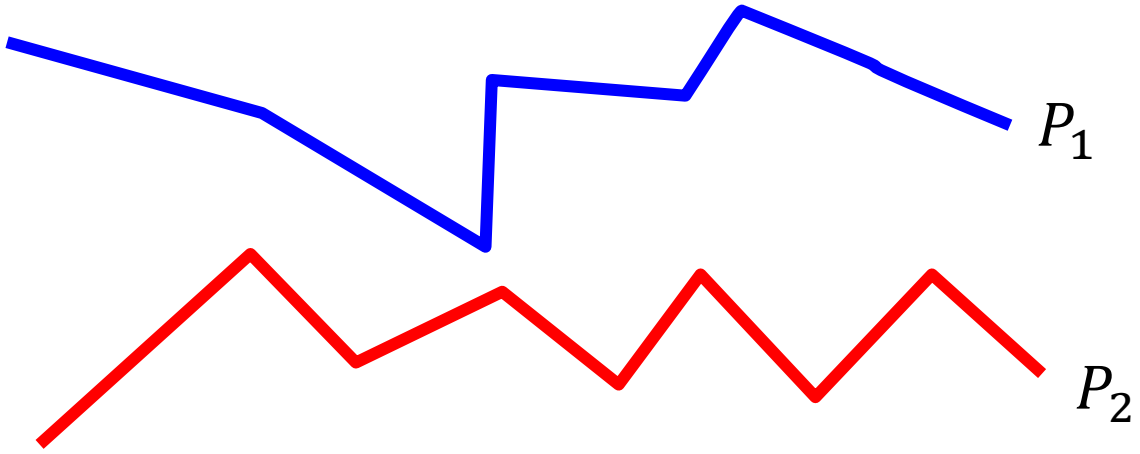
Problemformulierung

Gegeben:
Das Straßennetz als planar eingebetteter Graph $G = (V, E)$
Die GPS-Trajektorie als Folge $P = (p_1, p_2, \dots, p_m)$ von Punkten

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



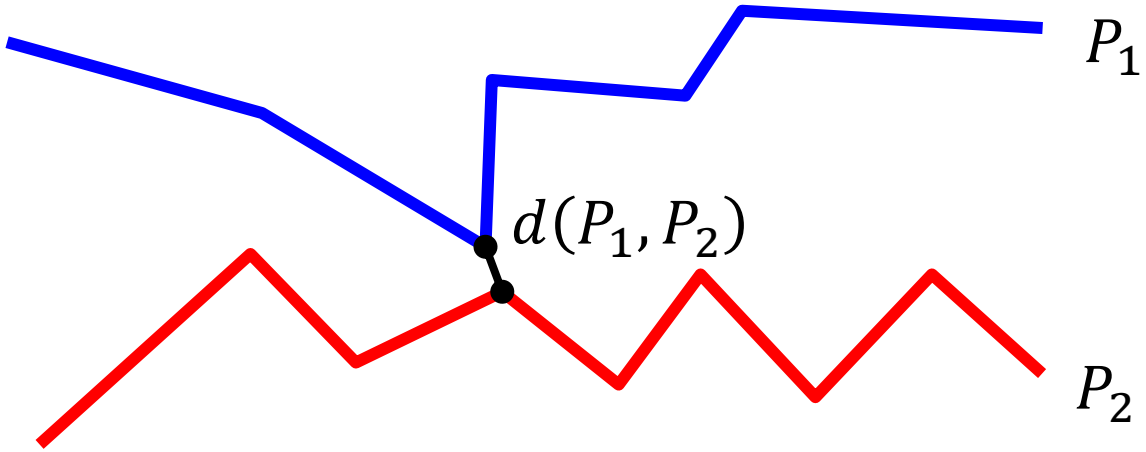
Map Matching



Map Matching

Minimaler euklidischer Abstand zwischen zwei beliebigen Punkten $p_1 \in P_1, p_2 \in P_2$

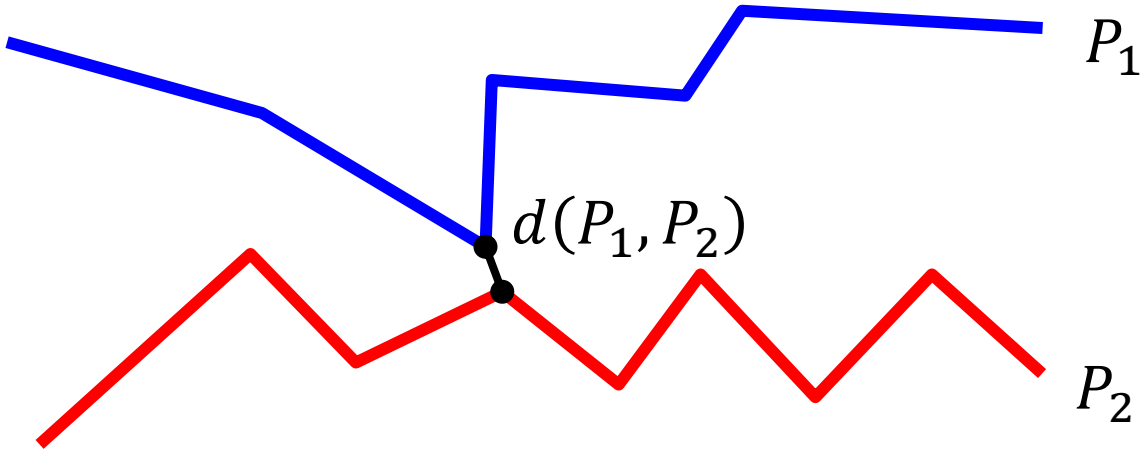
$$d(P_1, P_2) = \min\{d(p_1, p_2) | p_1 \in P_1, p_2 \in P_2\}$$



Map Matching

Minimaler euklidischer Abstand zwischen zwei beliebigen Punkten $p_1 \in P_1, p_2 \in P_2$

$$d(P_1, P_2) = \min\{d(p_1, p_2) | p_1 \in P_1, p_2 \in P_2\}$$

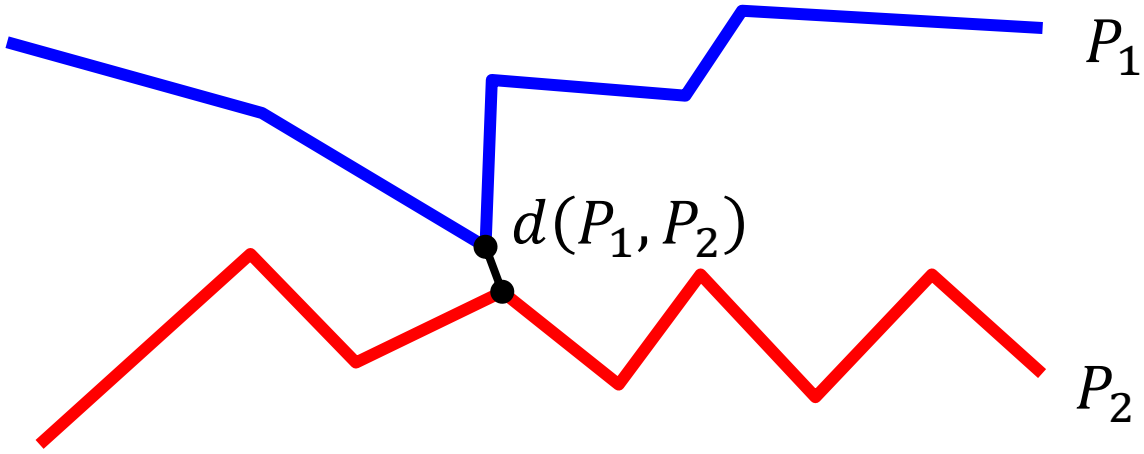


P_1 und P_2 hier: Polygonzüge
allgemeiner: Punktmengen

Map Matching

Minimaler euklidischer Abstand zwischen zwei beliebigen Punkten $p_1 \in P_1, p_2 \in P_2$

$$d(P_1, P_2) = \min\{d(p_1, p_2) | p_1 \in P_1, p_2 \in P_2\}$$

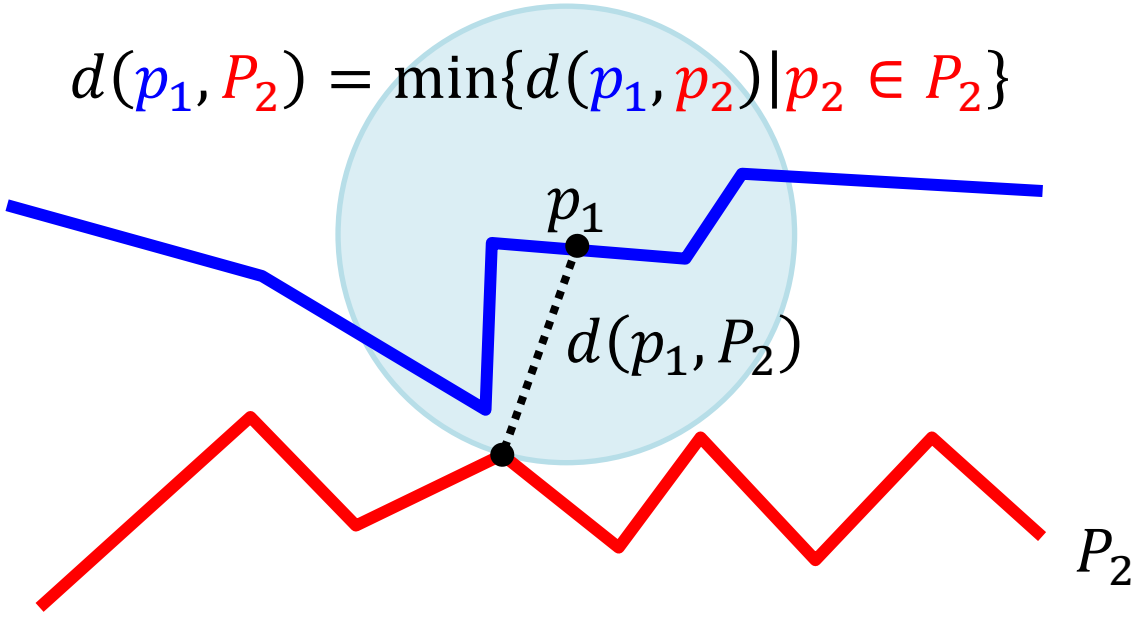


Wird zu Null, wenn sich P_1 und P_2 schneiden!

Als Distanzmaß ungeeignet!

Map Matching

Minimaler euklidischer Abstand zwischen einem festen Punkt $p_1 \in P_1$ und einem beliebigen Punkt $p_2 \in P_2$

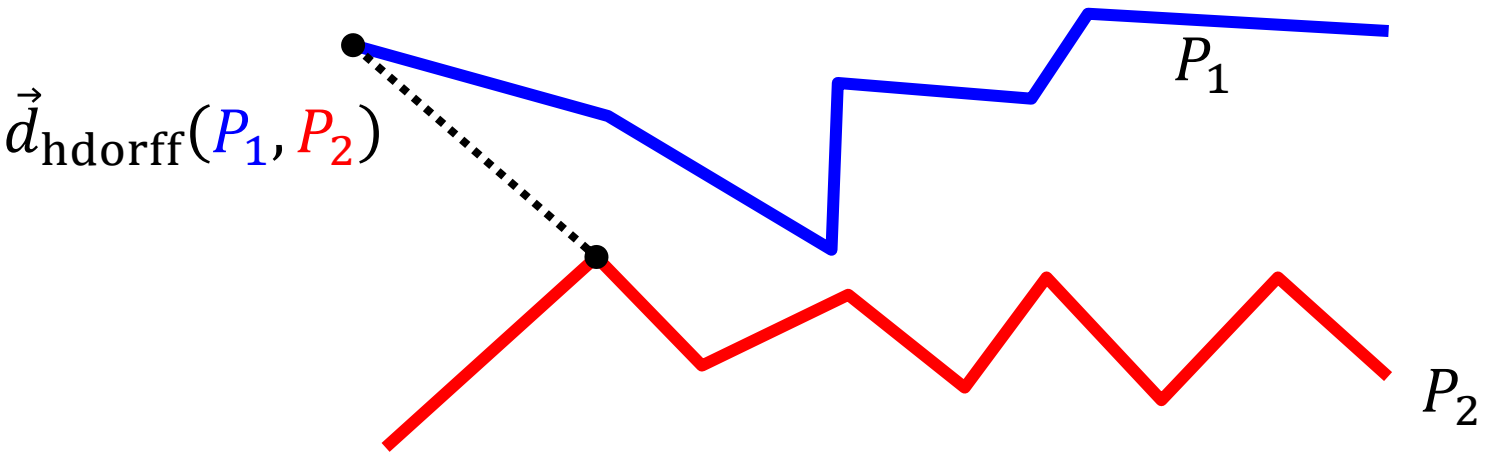


Map Matching

gerichtete Hausdorff-Distanz

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

mit $$d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$$

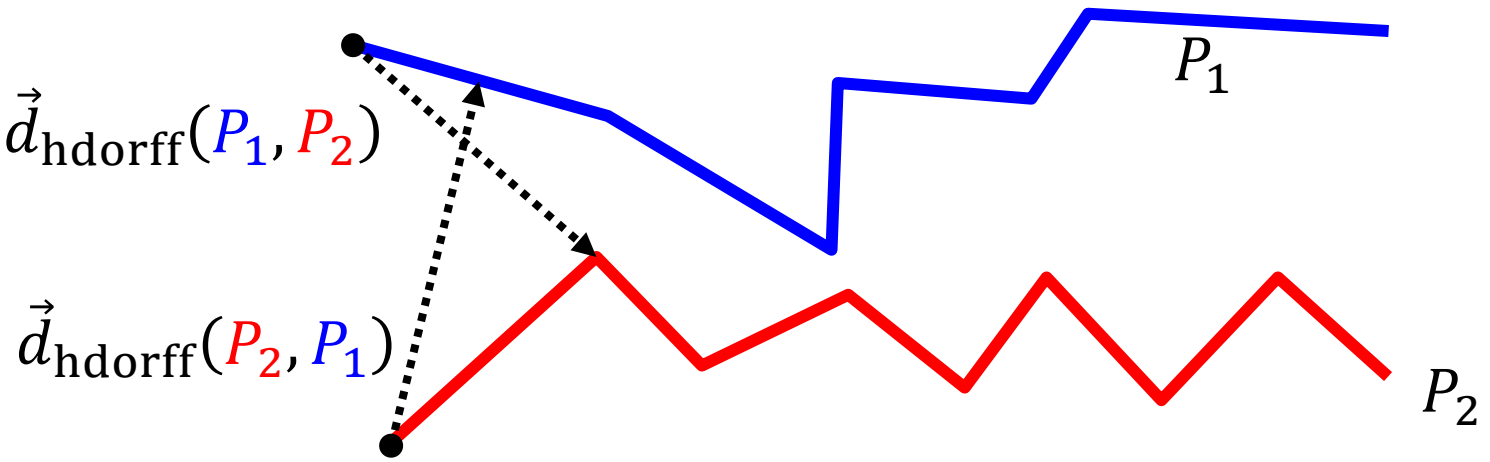


Map Matching

gerichtete Hausdorff-Distanz

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

mit $$d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$$



Achtung: Allgemein gilt $\vec{d}_{\text{hdorff}}(P_1, P_2) \neq \vec{d}_{\text{hdorff}}(P_2, P_1)$.

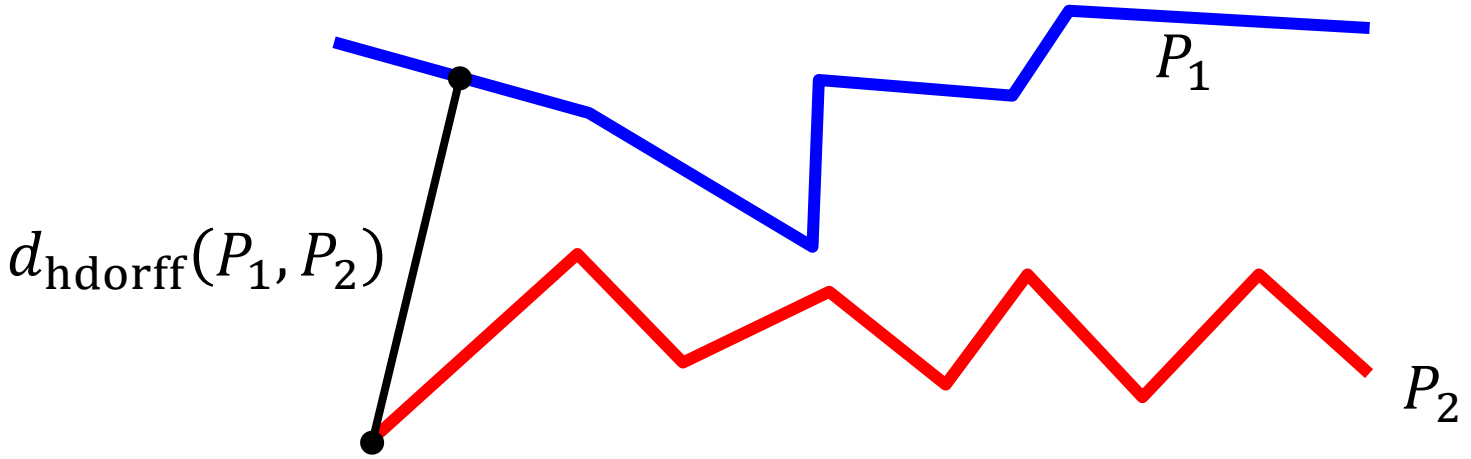
Map Matching

Hausdorff-Distanz

$$d_{\text{hdorff}}(P_1, P_2) = \max\{\vec{d}_{\text{hdorff}}(P_1, P_2), \vec{d}_{\text{hdorff}}(P_2, P_1)\}$$

mit $\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$

und $d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$

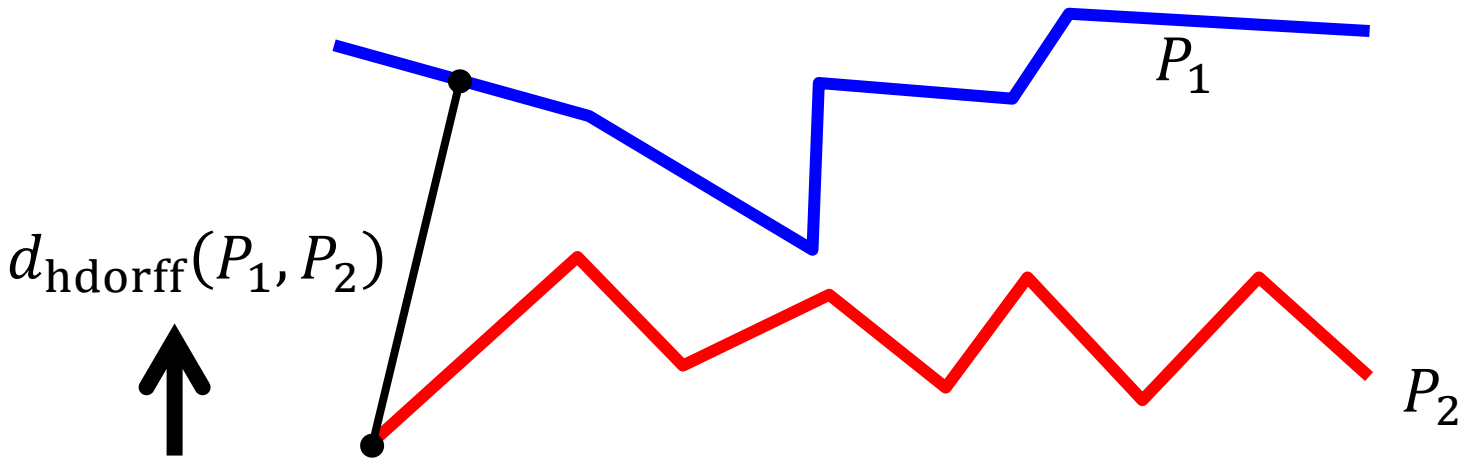


Algorithmus?

Map Matching

Hausdorff-Distanz

Berechnung für zwei Polygonzüge:

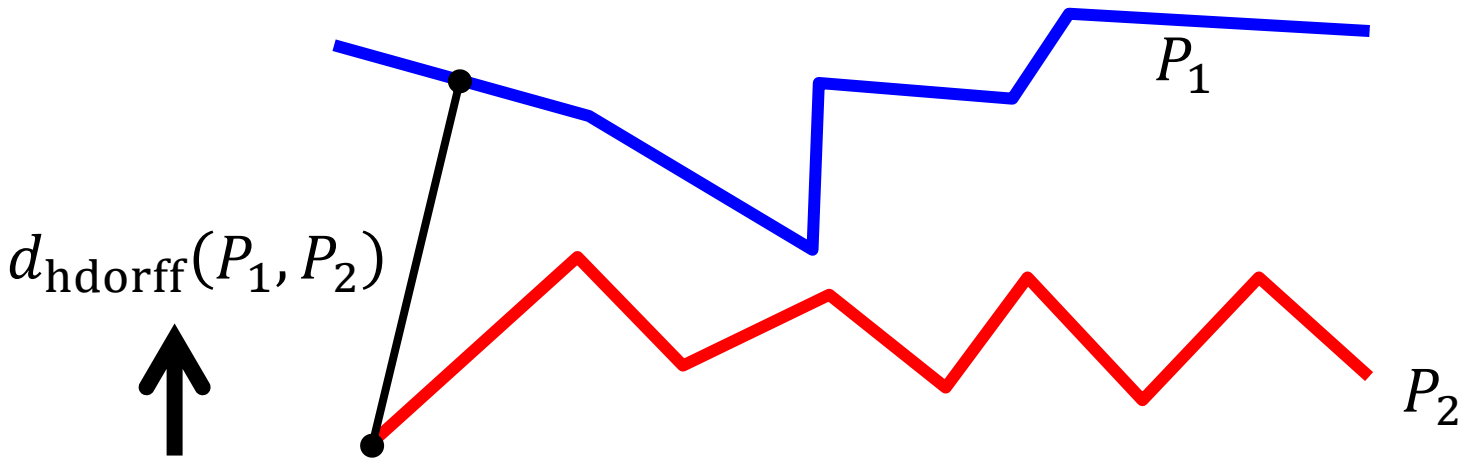


Bezieht immer eine **Ecke und eine Kante *oder* zwei Ecken** ein.

Map Matching

Hausdorff-Distanz

Berechnung für zwei Polygonzüge:



Bezieht immer
eine **Ecke und eine Kante** ***oder*** **zwei Ecken** ein.

Naiver Algorithmus
braucht $O(mn)$ Zeit.

Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

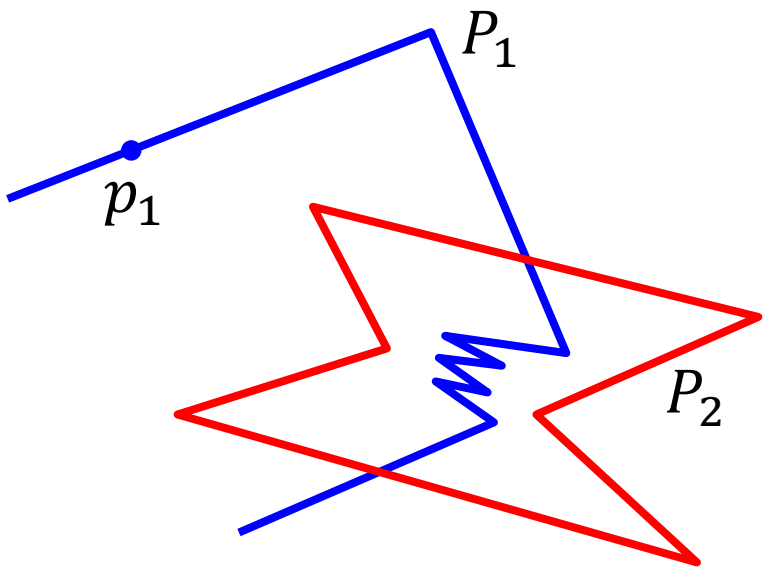
Alt et al. 1995: Approximate Matching of Polygonal Shapes. Annals of Mathematics and Artificial Intelligence, 13(1995)251-265.

Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

$$\vec{d}_{\mathrm{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$



Map Matching

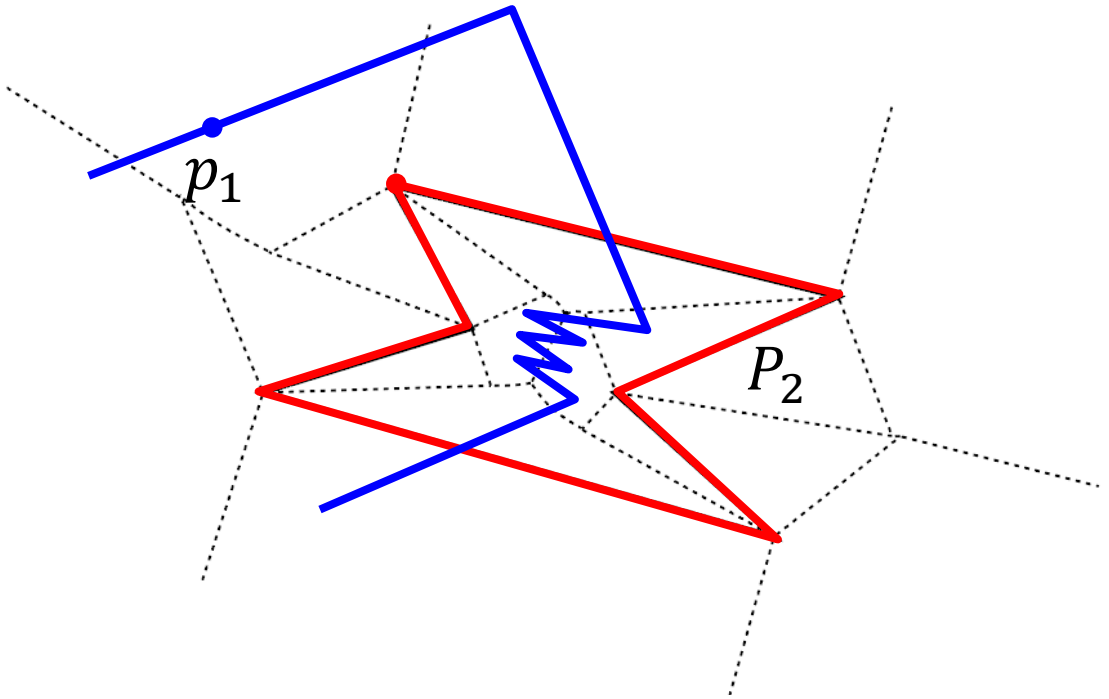
Hausdorff-Distanz

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

Voronoi-Diagramm für P_2

- kann in $O(n \log n)$ Zeit berechnet werden
- hat $O(n)$ Kanten

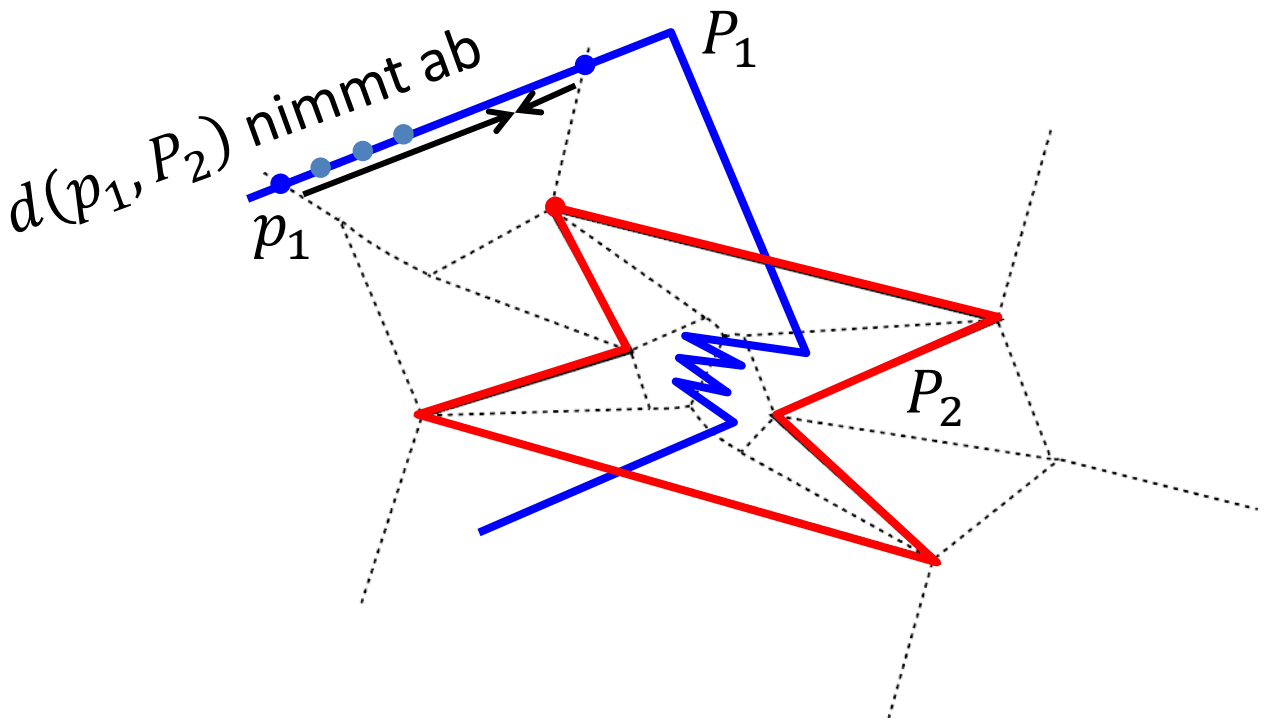


Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

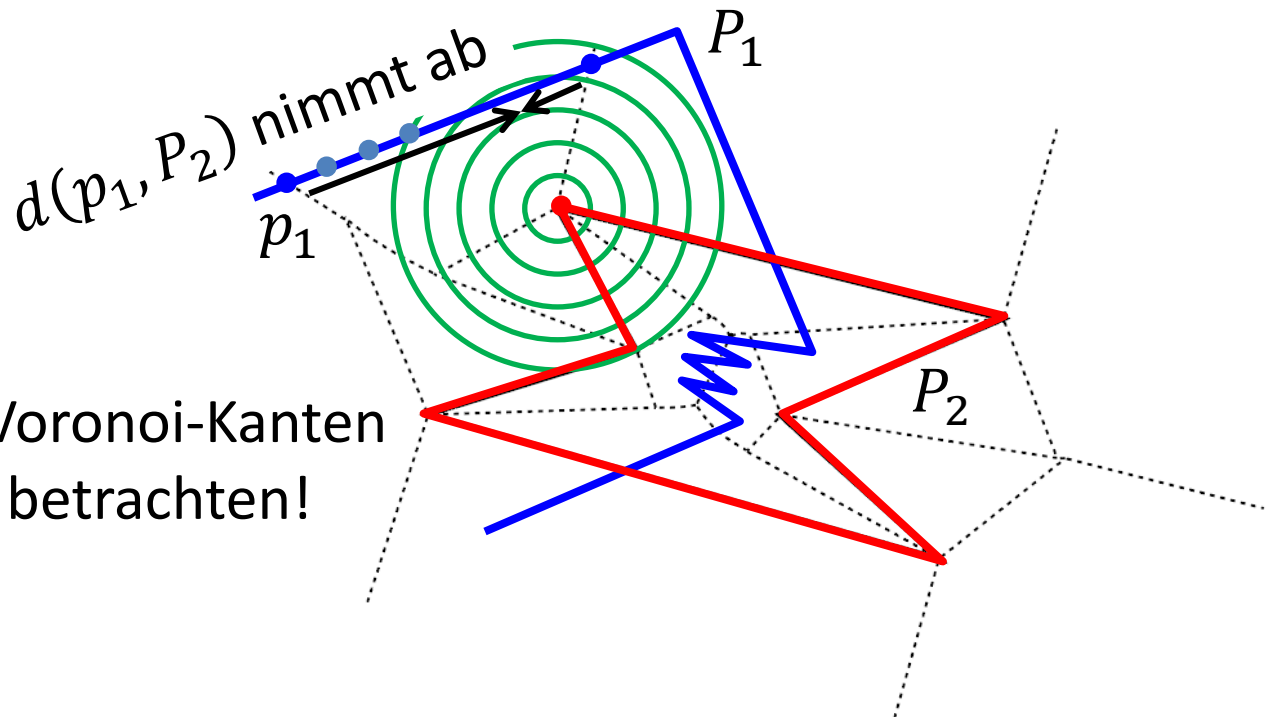


Map Matching

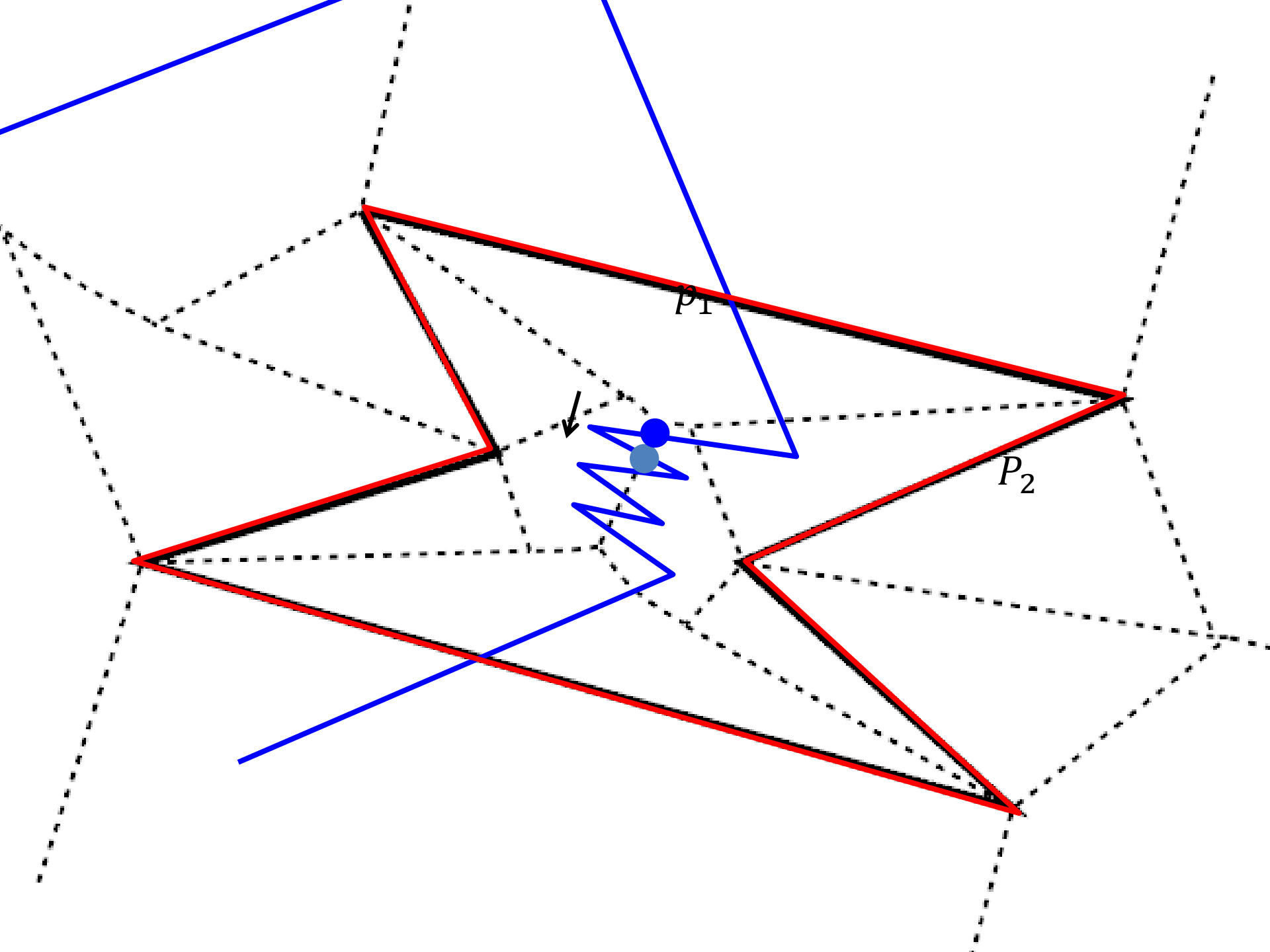
Hausdorff-Distanz

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$



Es reicht aus, für p_1
Schnitte von P_1 mit Voronoi-Kanten
und Ecken von P_1 zu betrachten!

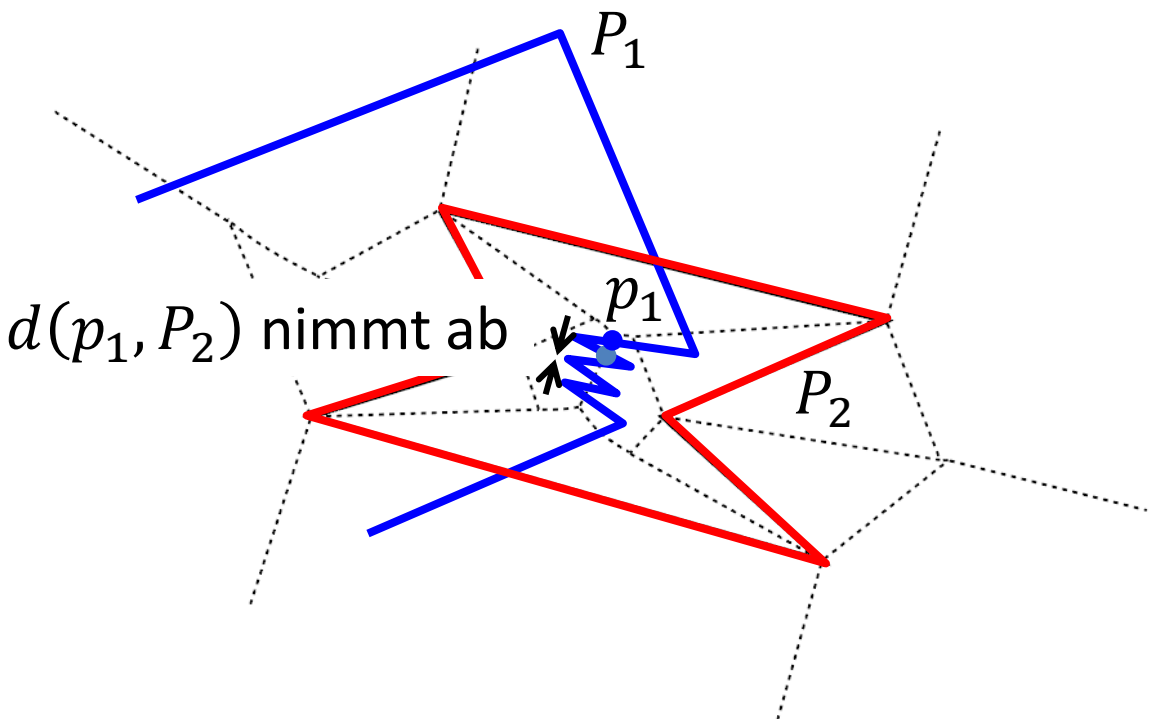

P_1
P_2

Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

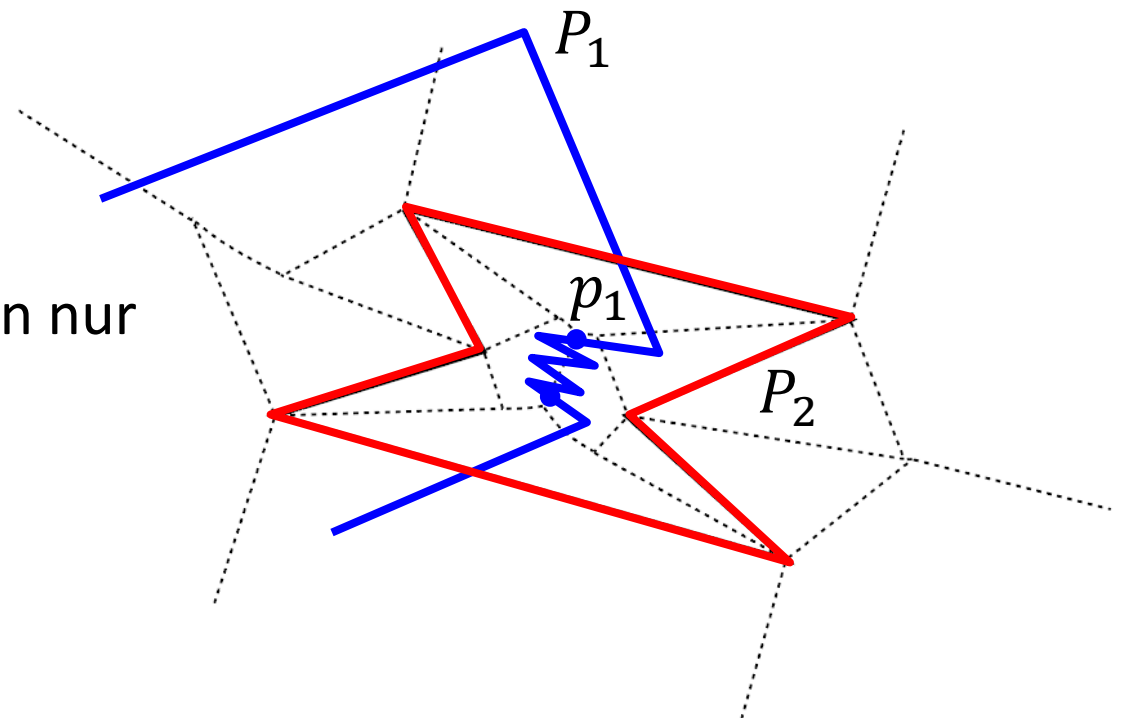


Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$



Pro Voronoi-Kante kommen nur zwei Schnitte für p_1 in Betracht!

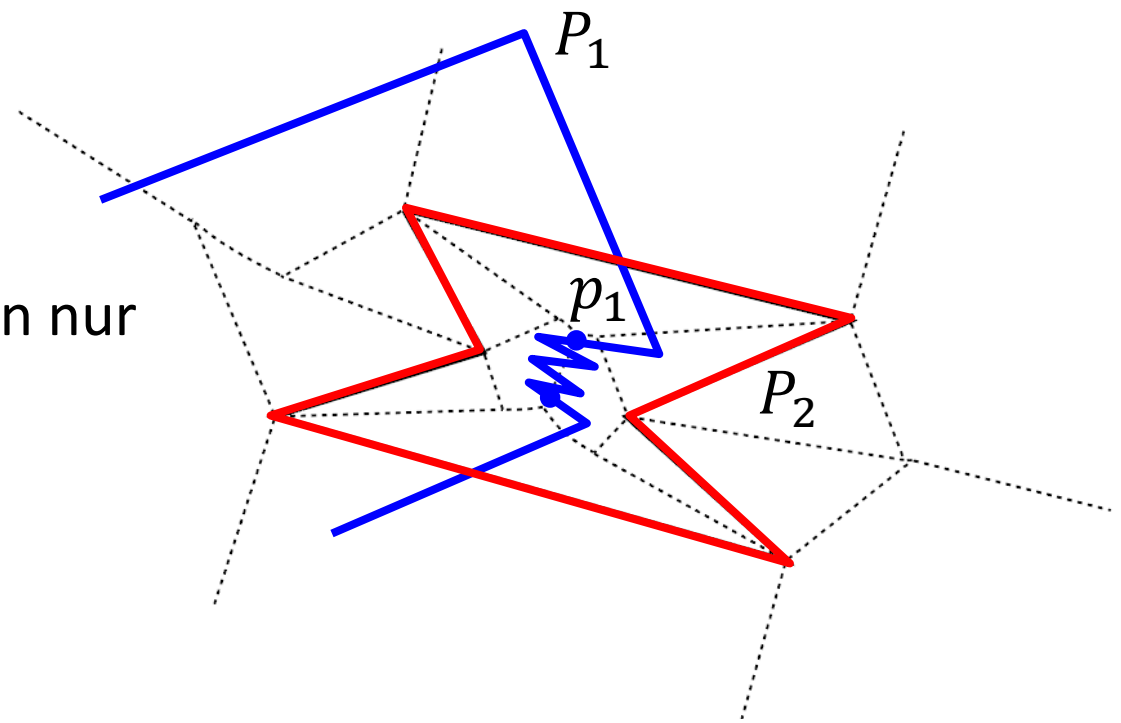
Map Matching

Hausdorff-Distanz

diskrete Menge mit $O(m+n)$ Elementen

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in S\}$$



Pro Voronoi-Kante kommen nur zwei Schnitte für p_1 in Betracht!

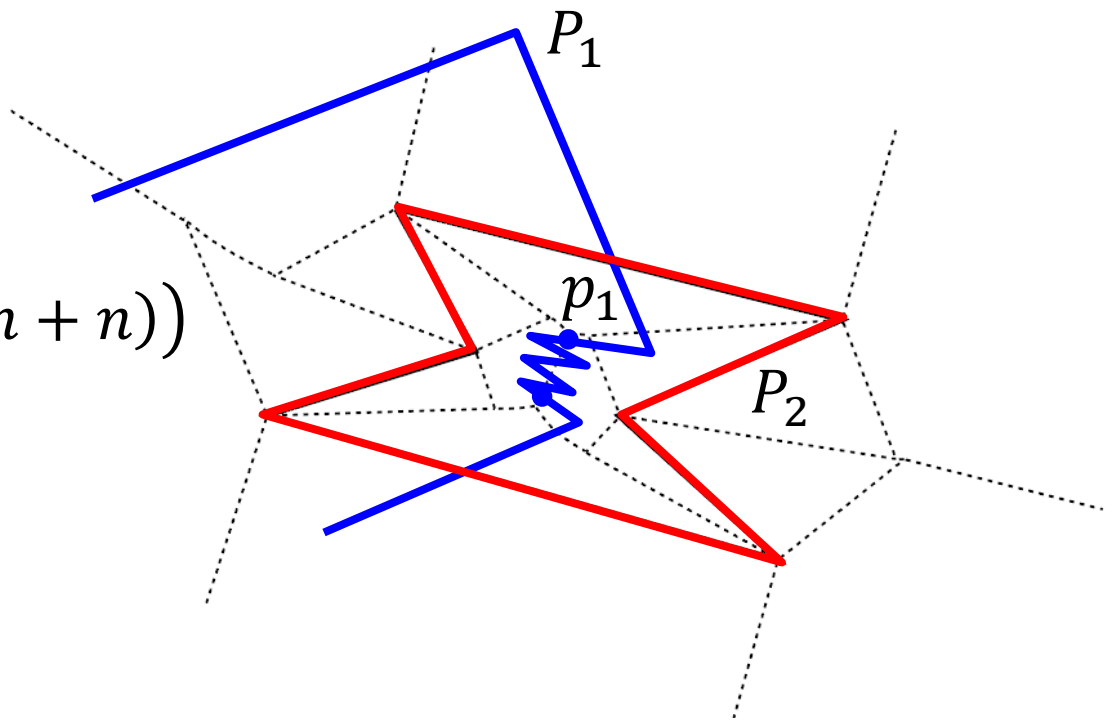
Map Matching

Hausdorff-Distanz

diskrete Menge mit $O(m+n)$ Elementen

Besserer Algorithmus:

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in S\}$$



S kann in $O\big((m+n)\log(m+n)\big)$ gefunden werden.
(Sweep Line Algorithmus)

Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

Laufzeit:	
Voronoi-Diagramm	$O(n \log n)$
Menge S	$O\big((m+n)\log(m+n)\big)$
Maximum finden	$O(m+n)$

Map Matching

Hausdorff-Distanz

Besserer Algorithmus:

Laufzeit:	
Voronoi-Diagramm	$O(n \log n)$
Menge S	$O\big((m+n)\log(m+n)\big)$
Maximum finden	$O(m+n)$
Gesamt	$\boldsymbol{O\big((m+n)\log(m+n)\big)}$

Map Matching

gerichtete Hausdorff-Distanz

$$\vec{d}_{\mathrm{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

mit $$d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$$

stellt für $p_1 \in P_1$ einen Bezug zu einem Punkt $p_2 \in P_2$ her

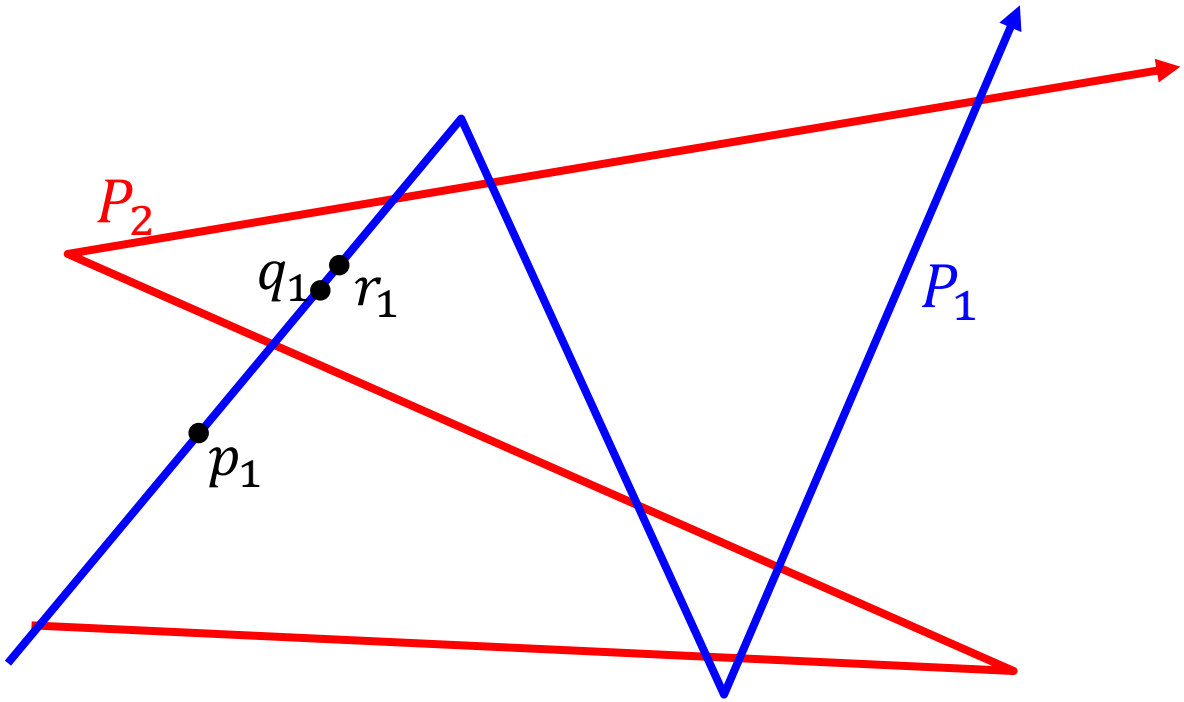
für Map Matching wichtig!

Map Matching

gerichtete Hausdorff-Distanz

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

mit $$d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$$

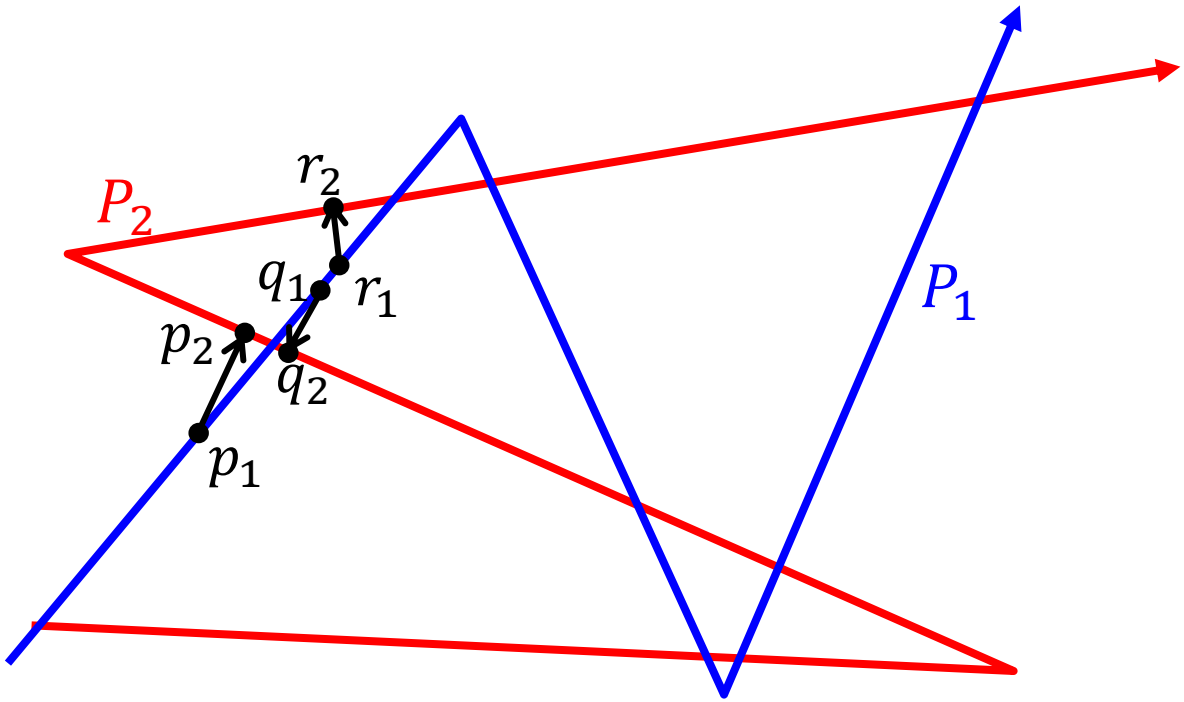


Map Matching

gerichtete Hausdorff-Distanz

$$\vec{d}_{\text{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

mit $$d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$$

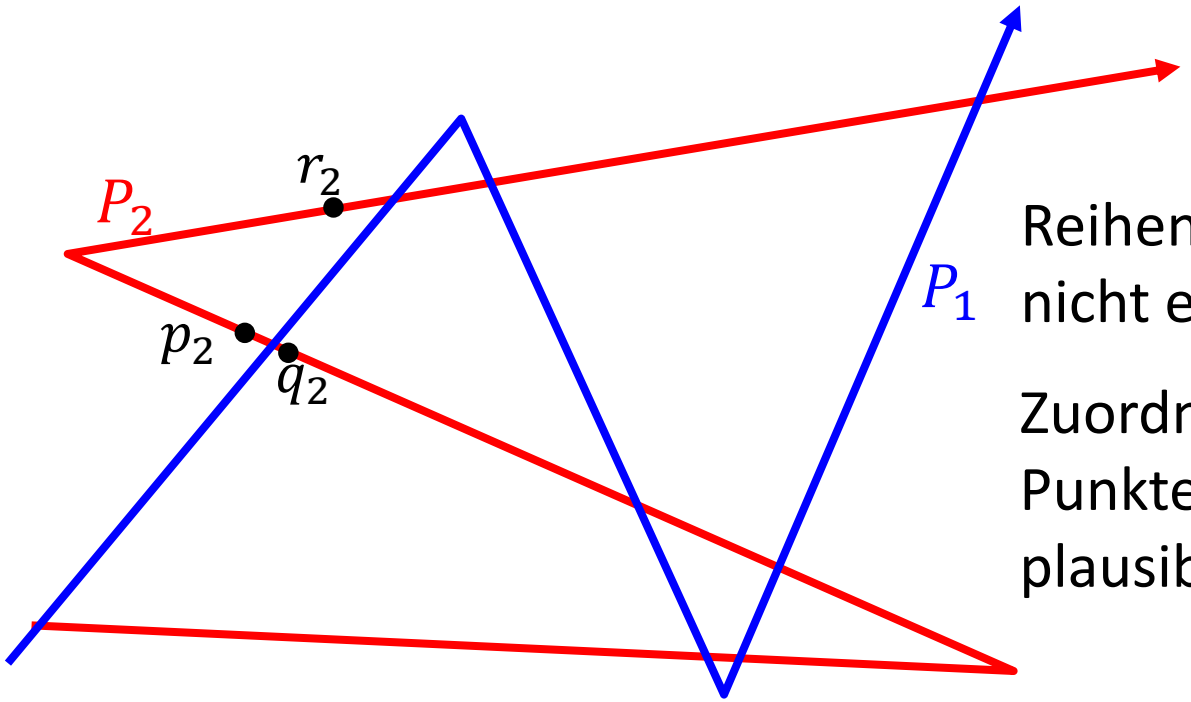


Map Matching

gerichtete Hausdorff-Distanz

$$\vec{d}_{\mathrm{hdorff}}(P_1, P_2) = \max\{d(p_1, P_2) | p_1 \in P_1\}$$

mit $$d(p_1, P_2) = \min\{d(p_1, p_2) | p_2 \in P_2\}$$



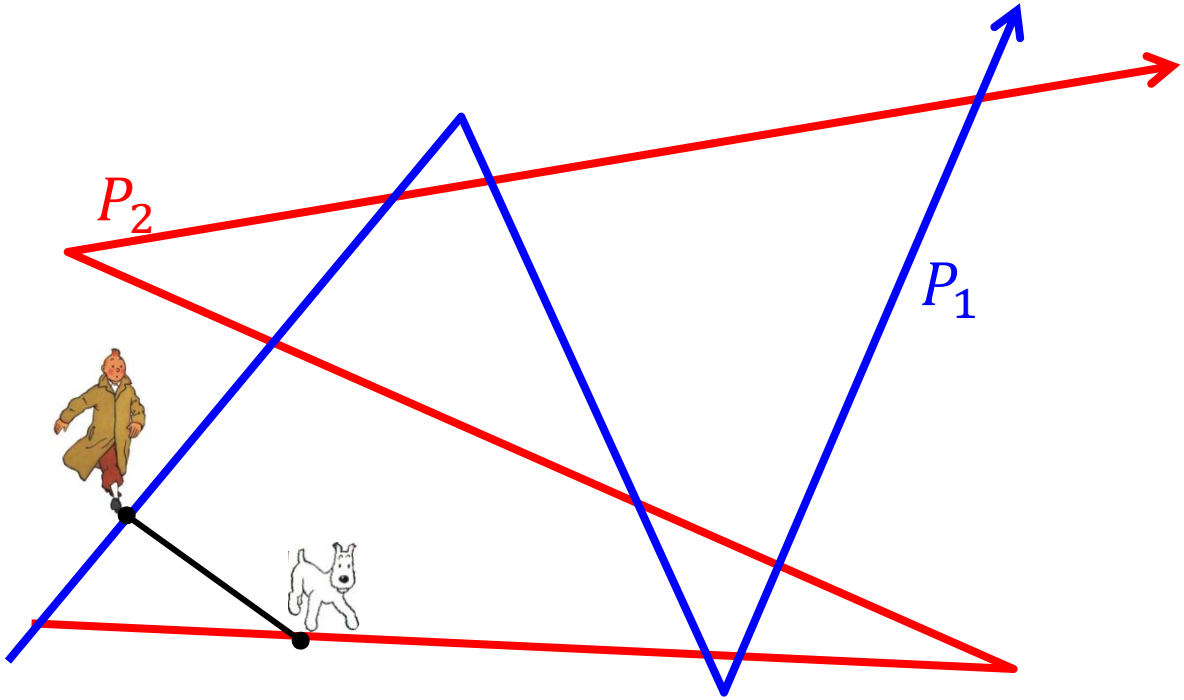
Reihenfolge pqr wird nicht eingehalten!

Zuordnung der Punkte nicht plausibel!

Fréchet-Distanz

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 ist die Hundeleine mindestens?



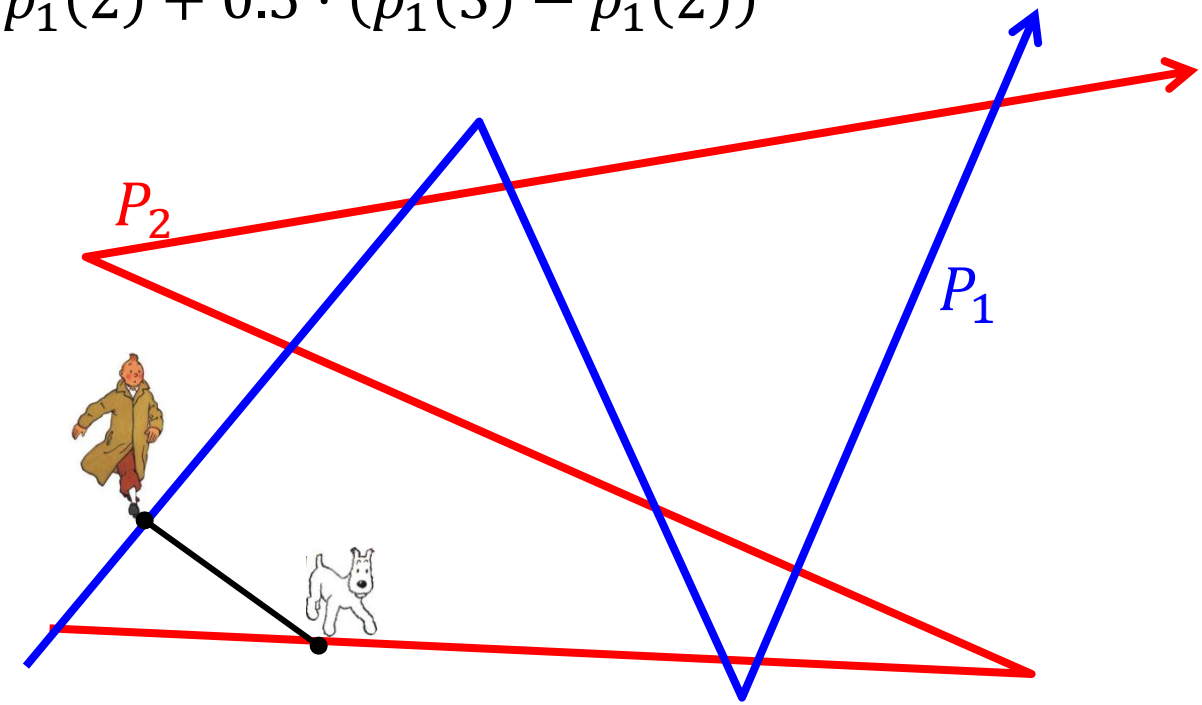
Fréchet-Distanz

Polygonzug als parametrisierte Kurve:

$p_1(t)$ mit $0 \leq t \leq m-1$ ist ein Punkt auf P_1.
Für ganzzahlige t ist $p_1(t)$ die $(t+1)$-te Ecke von P_1.
Ansonsten ist $p_1(t) = p_1(\lfloor t \rfloor) + (t - \lfloor t \rfloor) \cdot (\, p_1(\lfloor t \rfloor + 1) - p_1(\lfloor t \rfloor))$

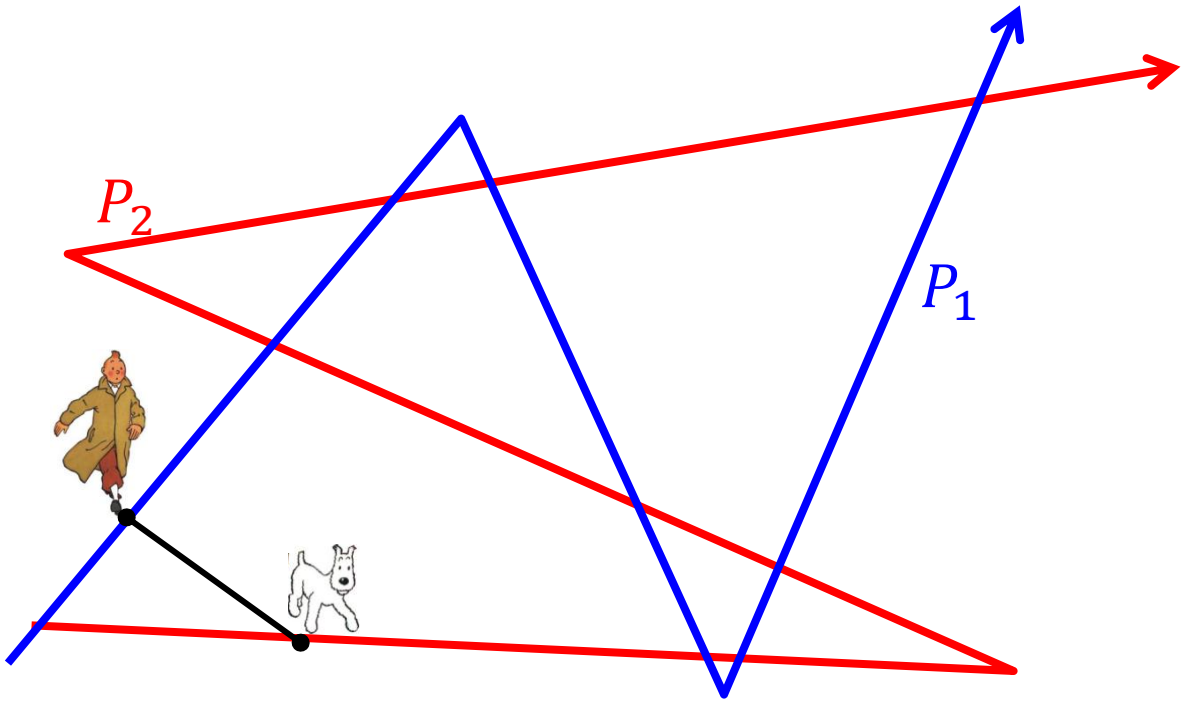
Beispiel:

$p_1(2.5) = p_1(2) + 0.5 \cdot (p_1(3) - p_1(2))$



Fréchet-Distanz

Definition: $d_{\text{fréchet}} =$ $\max_{t\in[0,1]}\{d(p_1(\alpha(t)), p_2(\beta(t)))\}$



Fréchet-Distanz

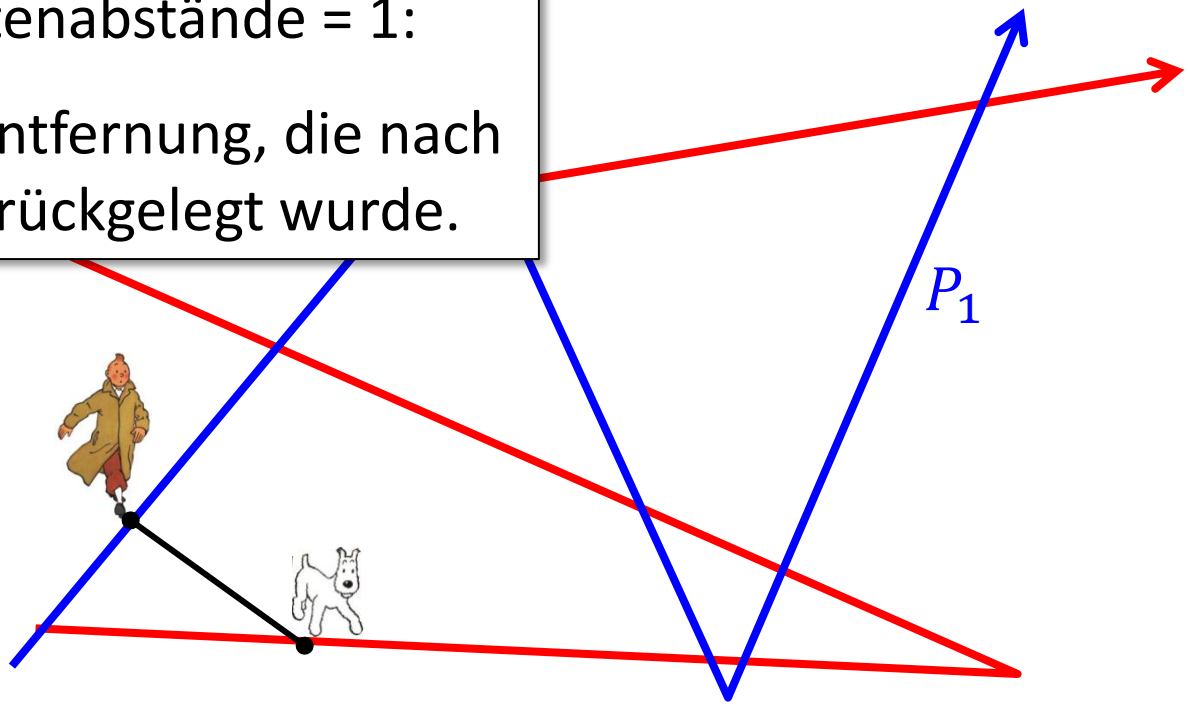
Definition:

$$d_{\text{fréchet}} = \min_{\substack{\alpha:[0,1]\to[0,m-1] \\ \beta:[0,1]\to[0,n-1]}} \max_{t\in[0,1]} \{d(p_1(\alpha(t)), p_2(\beta(t)))\}$$

wobei α und β **monoton wachsende** und **stetige Funktionen** sind und $\alpha(0) = \beta(0) = 0, \alpha(1) = m - 1, \beta(1) = n - 1.$

Für Knotenabstände = 1:

$\alpha(t)$ = Entfernung, die nach Zeit t zurückgelegt wurde.

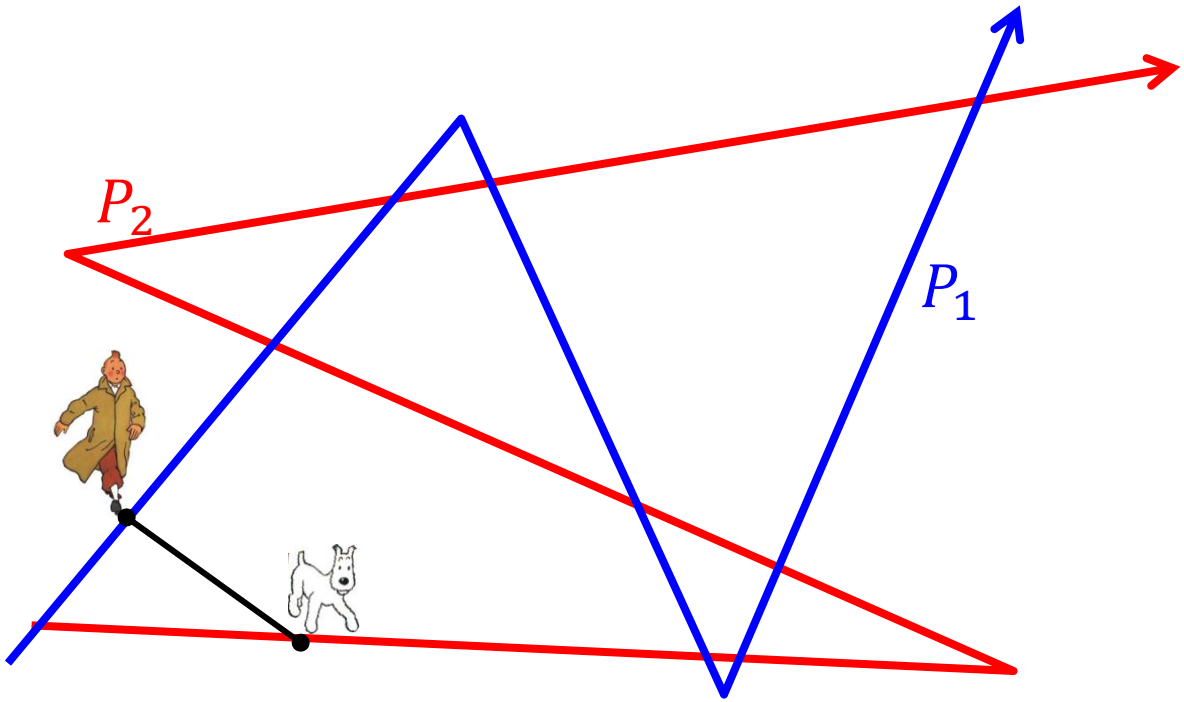


Fréchet-Distanz

Berechnung:

Löse Entscheidungsproblem: Ist $d_{\text{fréchet}} \leq \epsilon$?

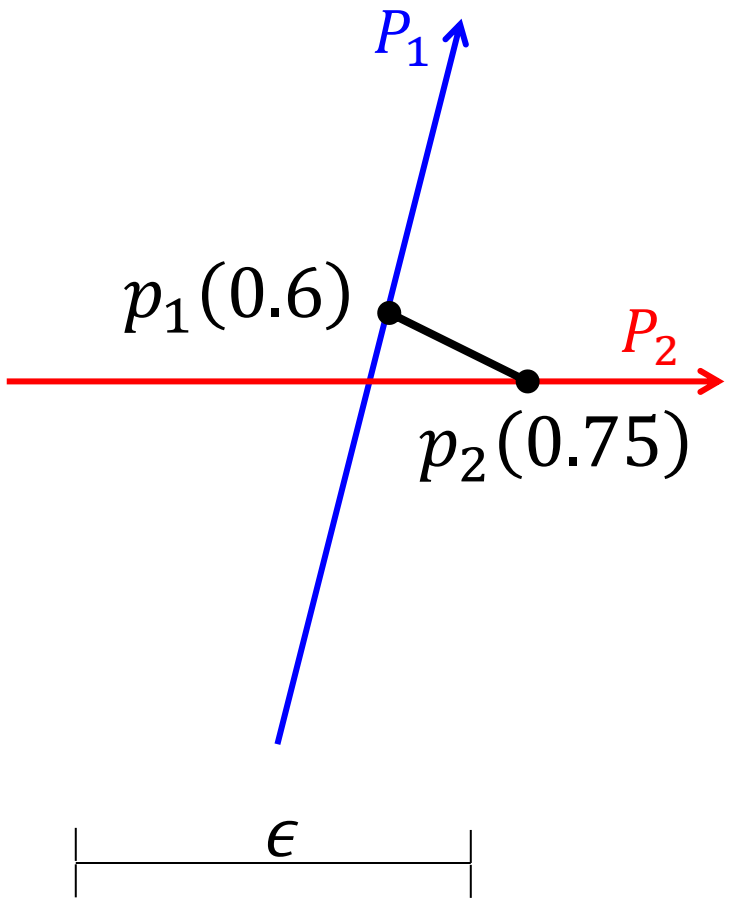
dann parametrische Suche...



Fréchet-Distanz

Ist $d_{\text{fréchet}} \leq \epsilon$?

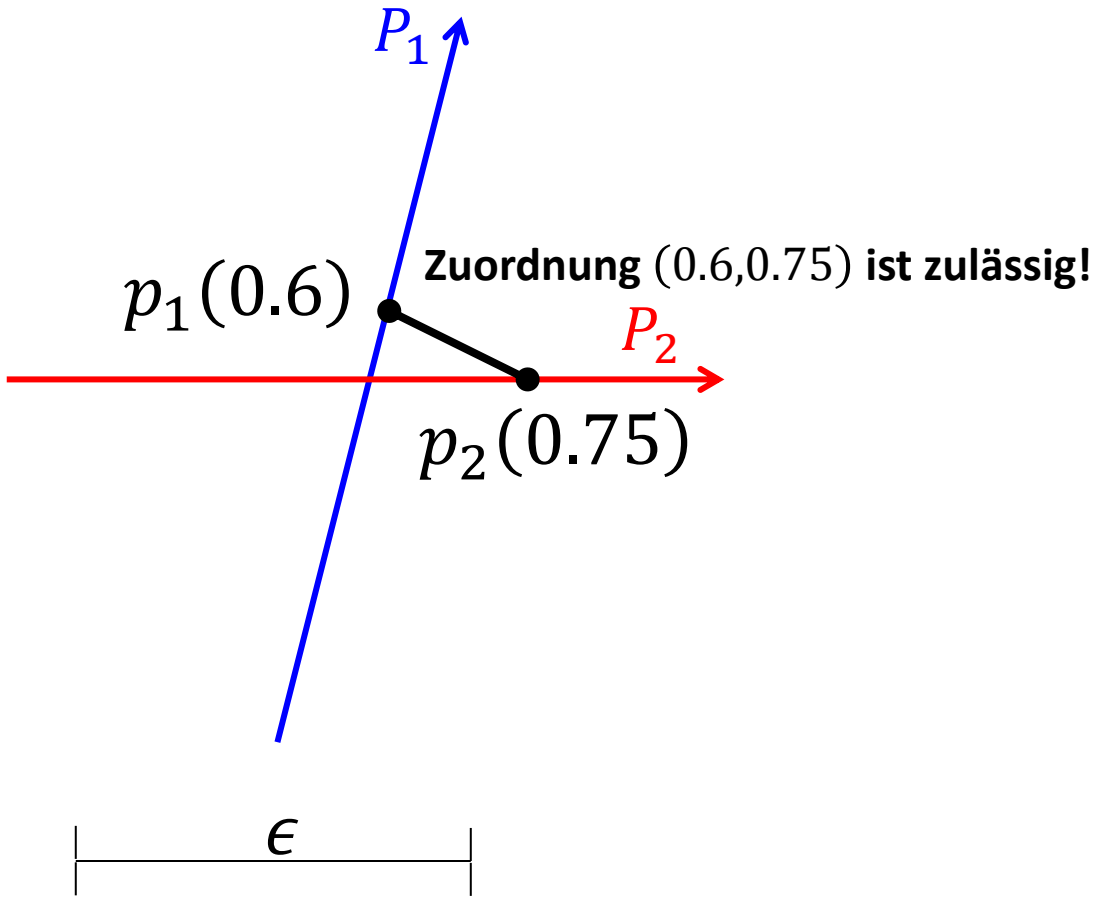
Spezialfall:



Fréchet-Distanz

Ist $d_{\text{fréchet}} \leq \epsilon$?

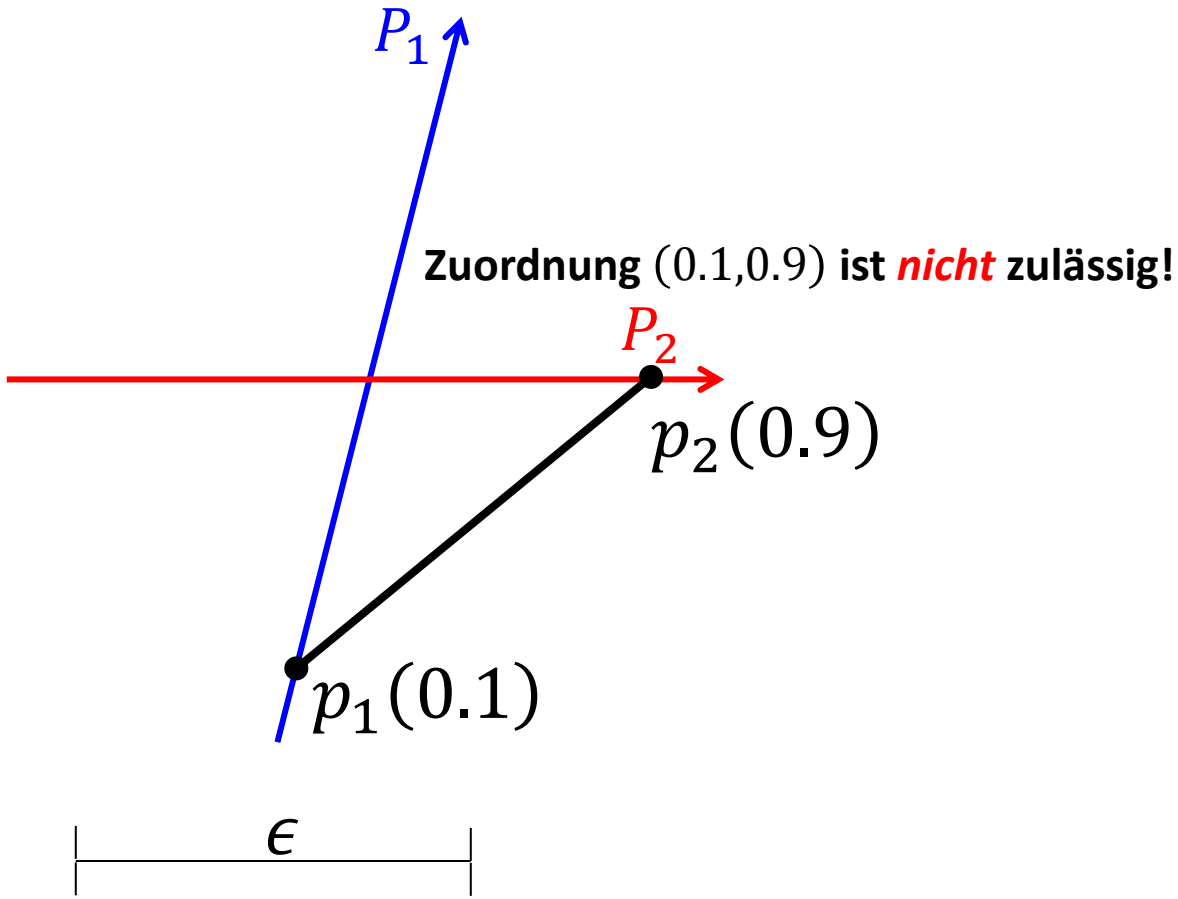
Spezialfall:



Fréchet-Distanz

Ist $d_{\text{fréchet}} \leq \epsilon$?

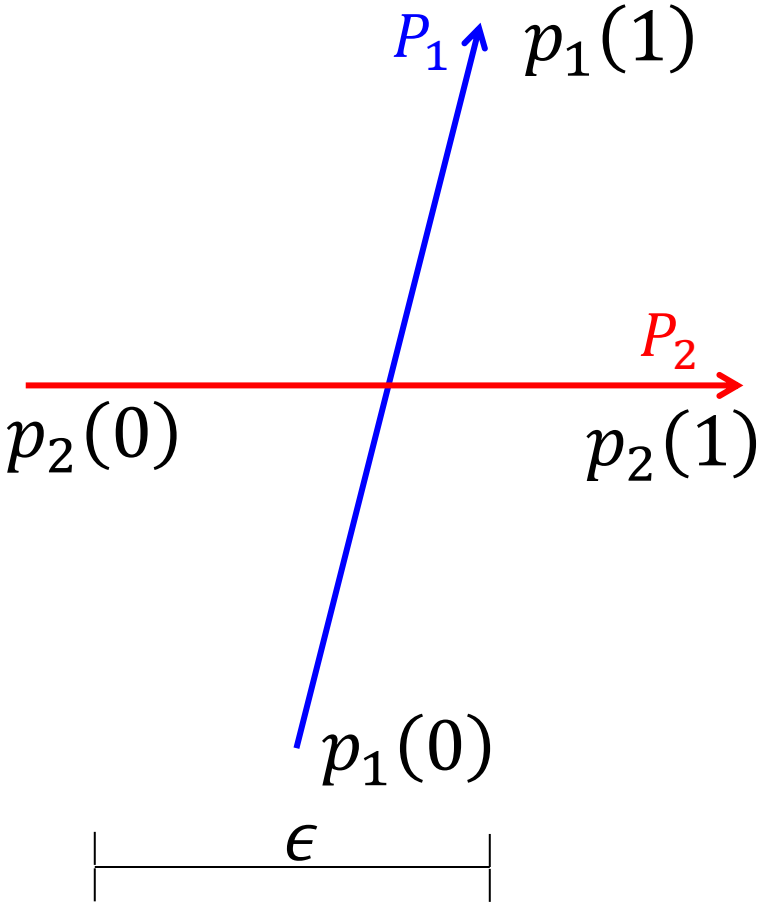
Spezialfall:

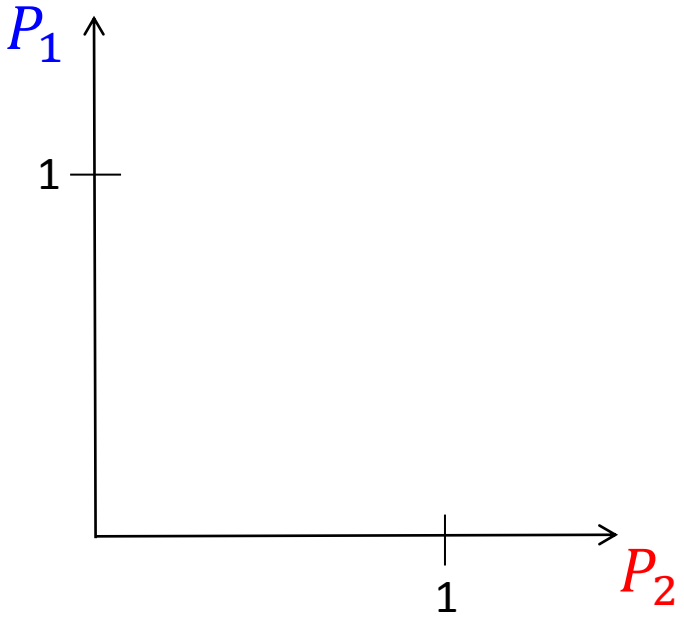


Fréchet-Distanz

Ist $d_{\text{fréchet}} \leq \epsilon$?

Spezialfall:

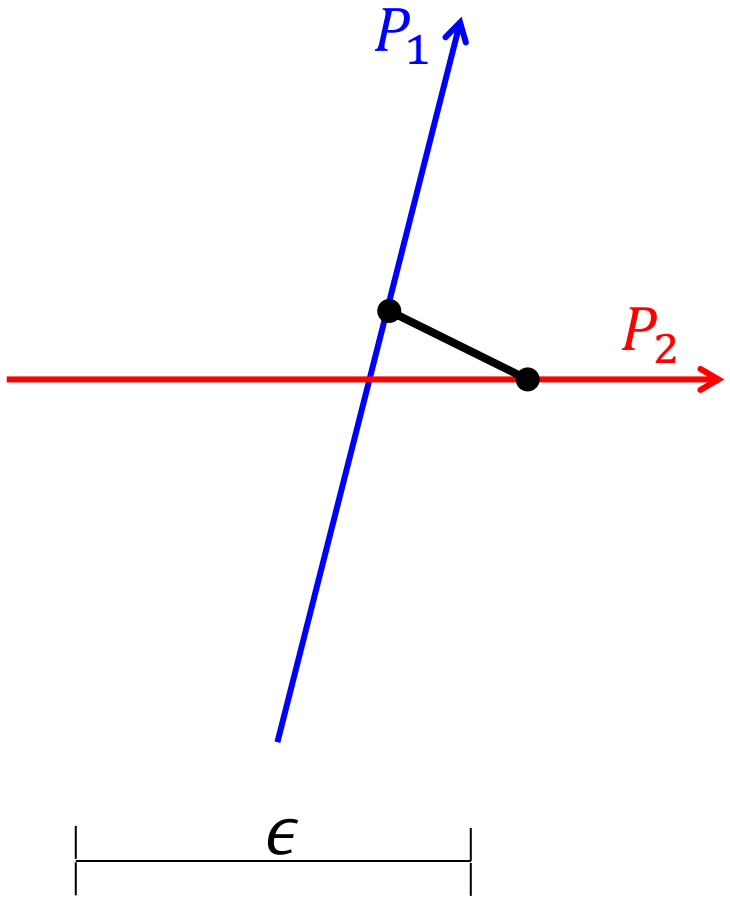




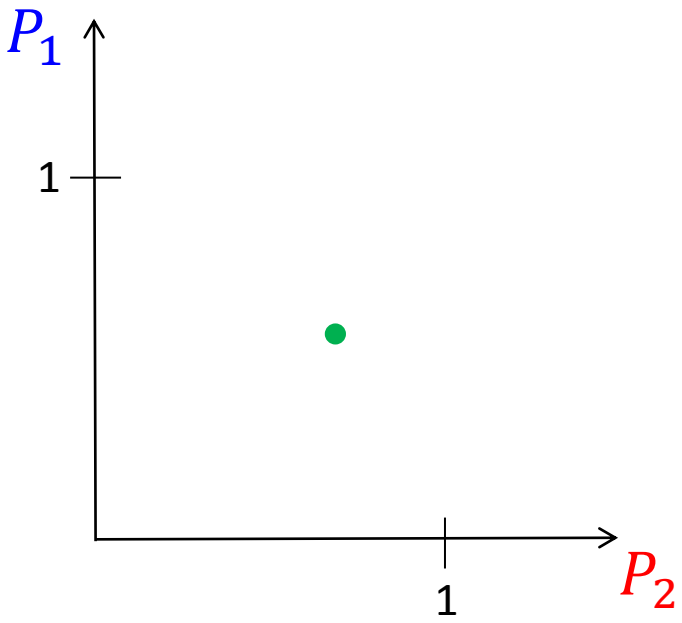
Fréchet-Distanz

Ist $d_{\text{fréchet}} \leq \epsilon$?

Spezialfall:



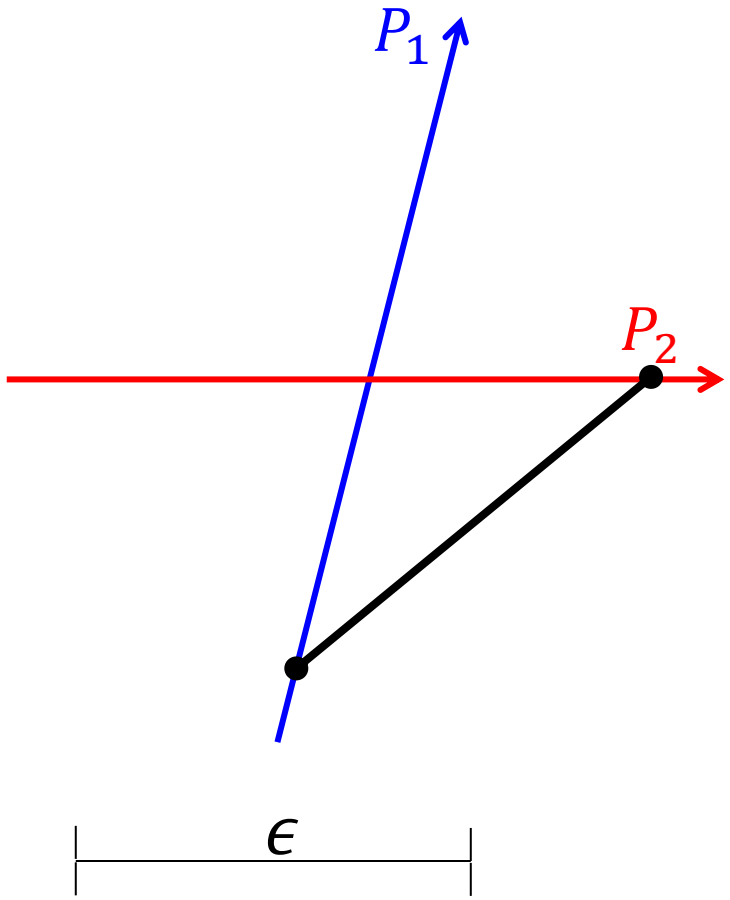
Zuordnung $(0.6,0.75)$ **ist zulässig!**



Fréchet-Distanz

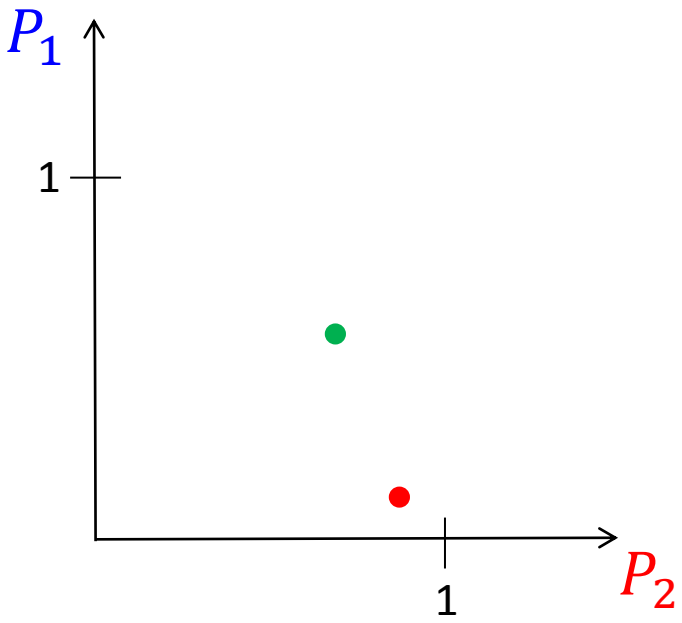
Ist $d_{\text{fréchet}} \leq \epsilon$?

Spezialfall:



Zuordnung $(0.6,0.75)$ **ist zulässig!**

Zuordnung $(0.1,0.9)$ **ist *nicht* zulässig!**



Fréchet-Distanz

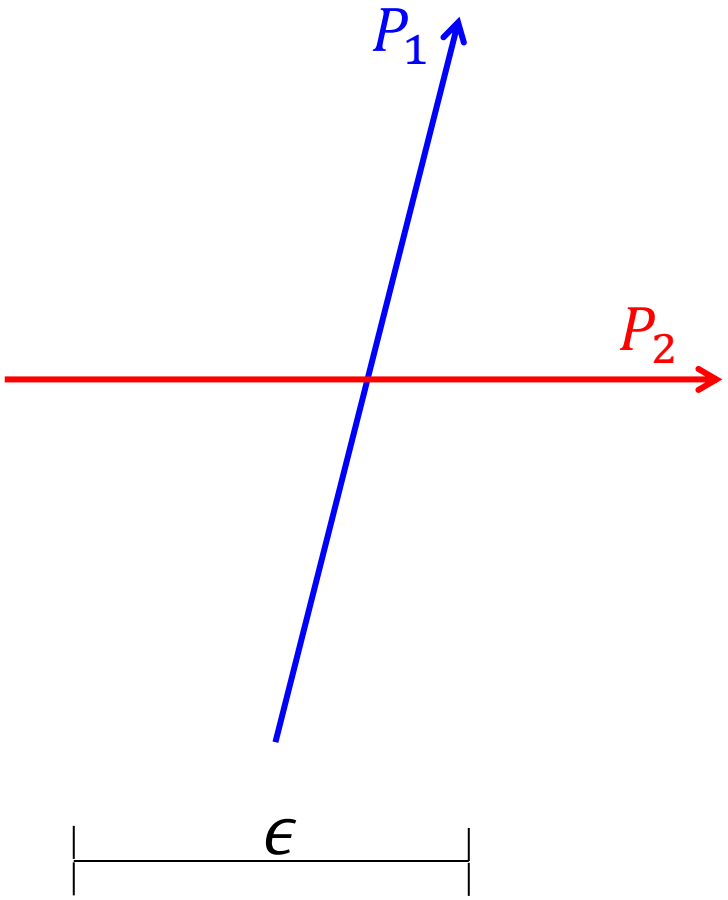
Ist $d_{\text{fréchet}} \leq \epsilon$?

Spezialfall:

Freiraum

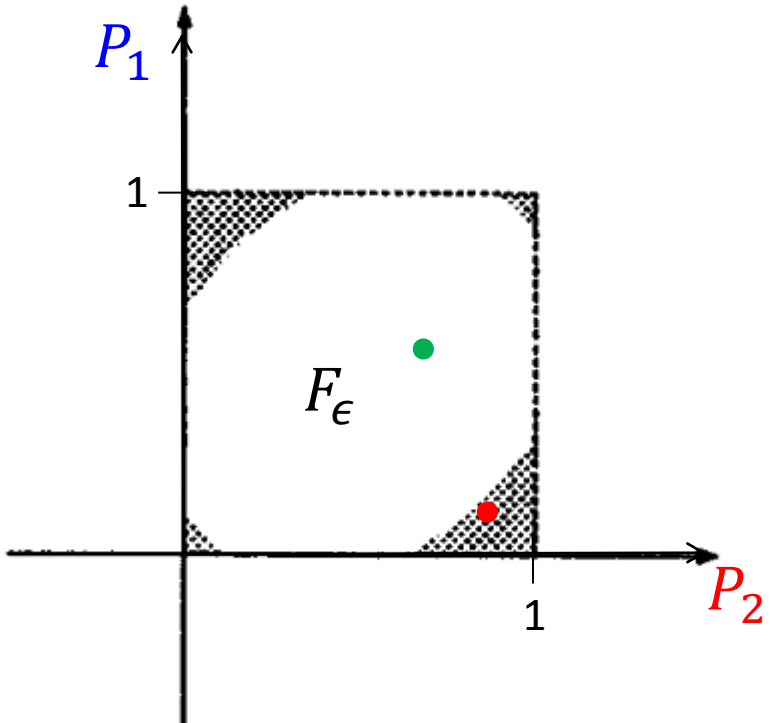
$F_\epsilon = \{(s,t) \in [0,1]^2 \mid$ [redacted] $\}$

hat die Form [redacted]



Zuordnung $(0.6,0.75)$ **ist zulässig!**

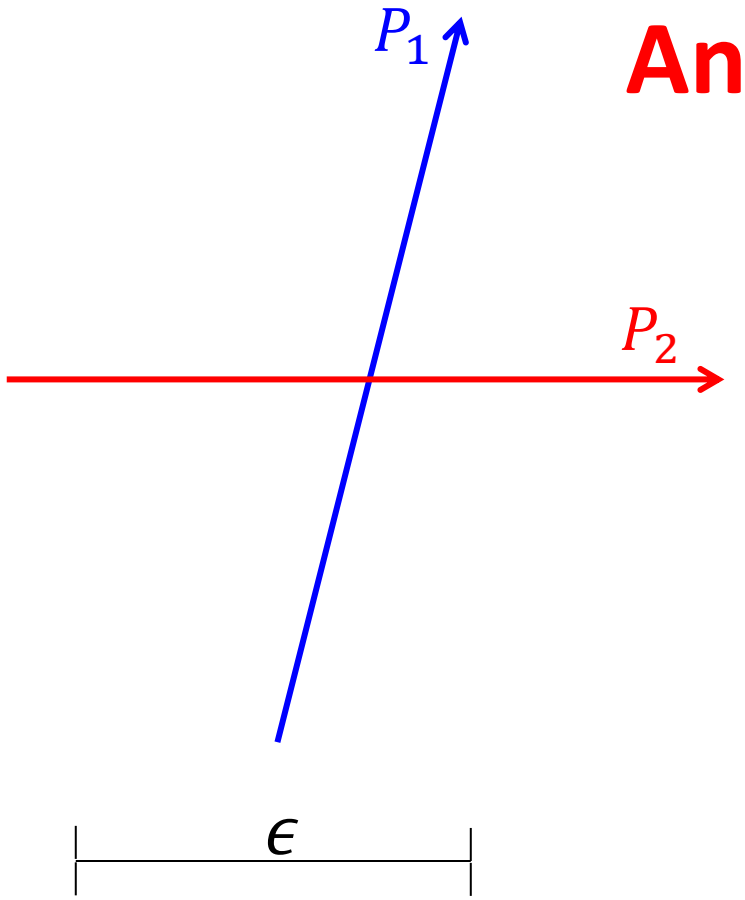
Zuordnung $(0.1,0.9)$ **ist *nicht* zulässig!**



Fréchet-Distanz

Ist $d_{\text{fréchet}} \leq \epsilon$?

Spezialfall:

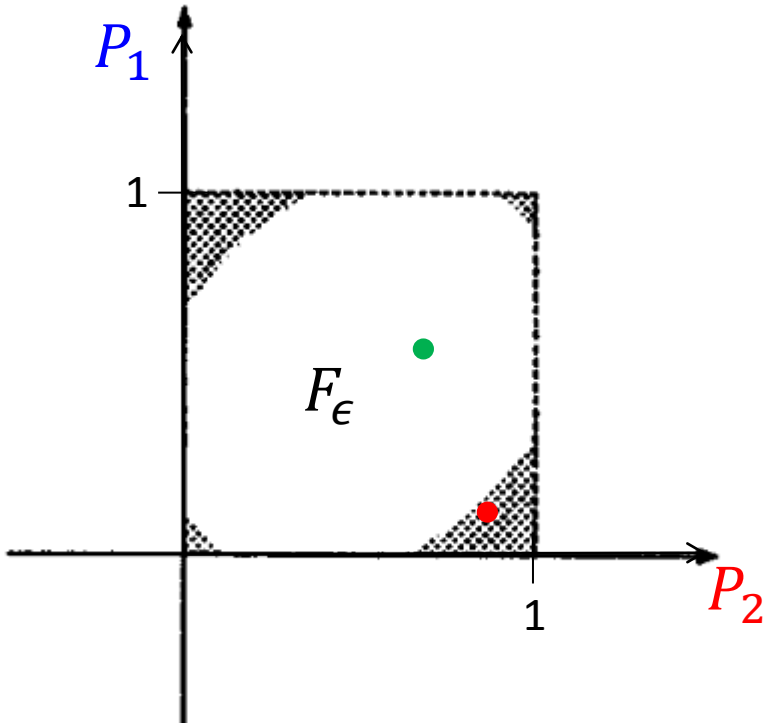


Freiraum

$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(p_1(s), p_2(t)) \leq \epsilon\}$

hat die Form einer Ellipse!

Anhand F_ϵ entscheiden!



Fréchet-Distanz

Freiraum

$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(p_1(s), p_2(t)) \leq \epsilon\}$

hat die Form einer Ellipse!

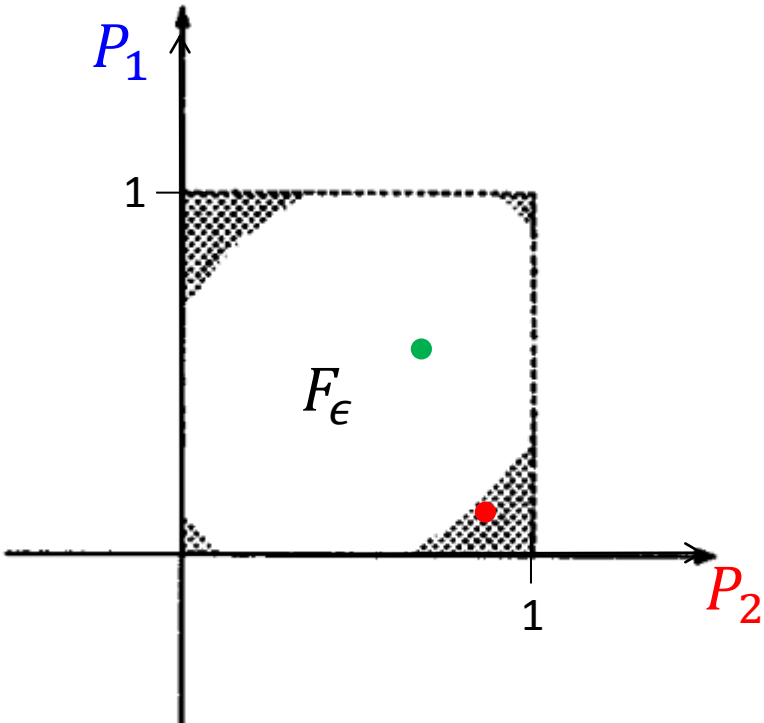
$d_{\text{fréchet}} \leq \epsilon$

$\Leftrightarrow$ Es gibt einen Pfad

von [ ] nach [ ]

im inneren von F_ϵ,

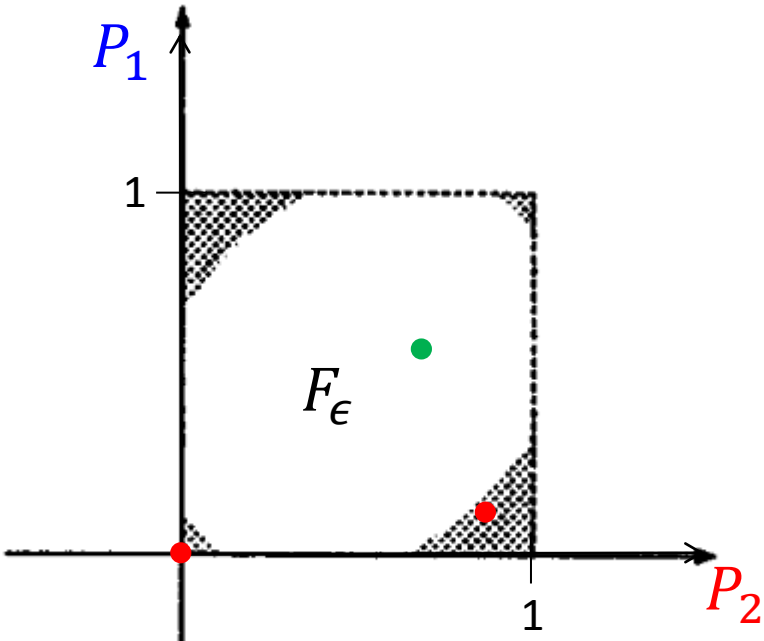
[ ]



Fréchet-Distanz

Freiraum
$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(p_1(s), p_2(t)) \leq \epsilon\}$
hat die Form einer Ellipse!

$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach $(1,1)$
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.

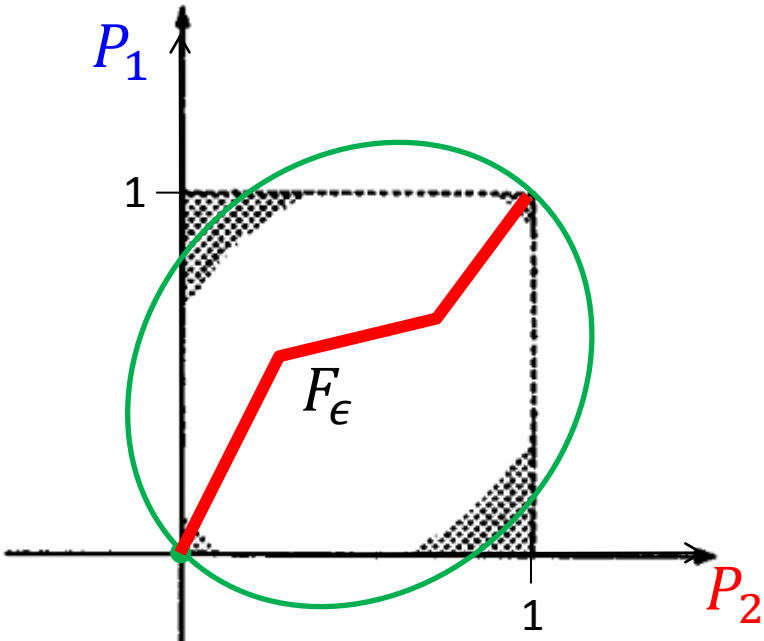


Hier gibt es ihn nicht, da $(0,0) \notin F_\epsilon$!

Fréchet-Distanz

Freiraum
$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(p_1(s), p_2(t)) \leq \epsilon\}$
hat die Form einer Ellipse!

$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach $(1,1)$
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.

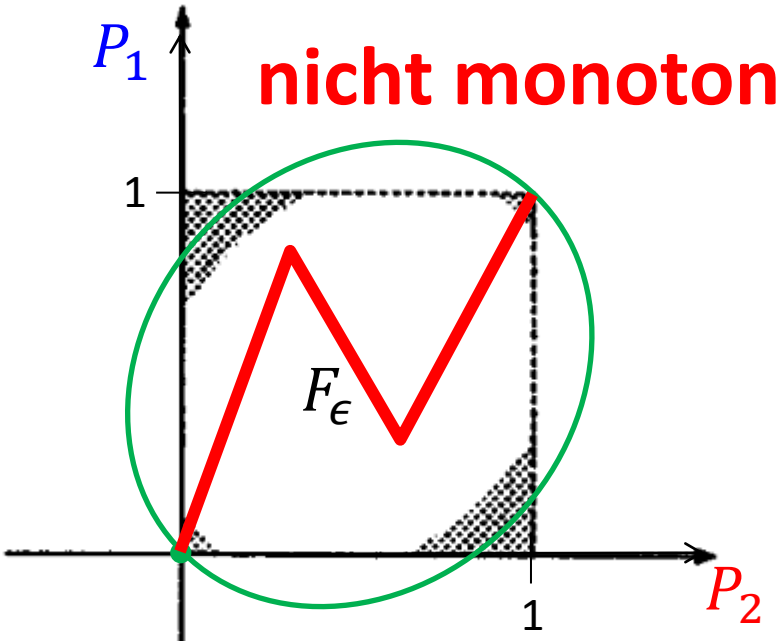


Für größeres ϵ gibt es einen Pfad!

Fréchet-Distanz

Freiraum
$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(p_1(s), p_2(t)) \leq \epsilon\}$
hat die Form einer Ellipse!

$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach $(1,1)$
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.



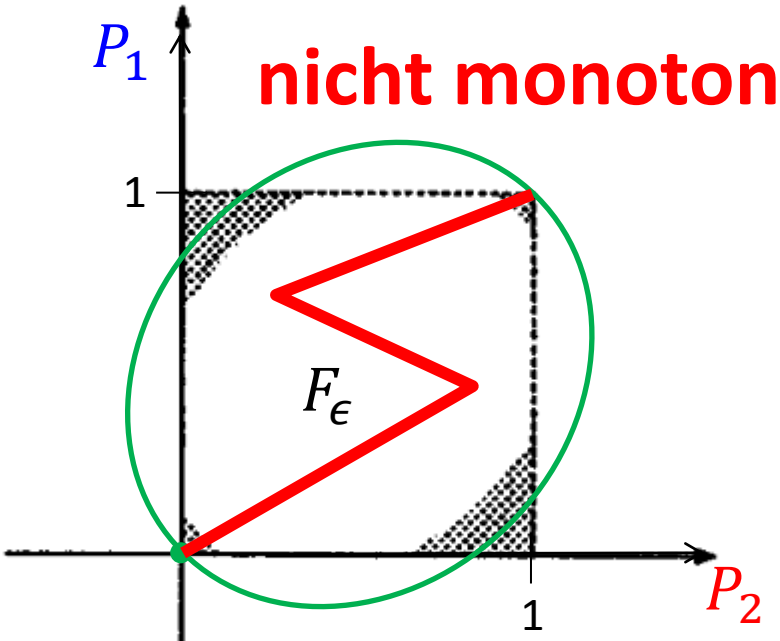
Für größeres ϵ gibt es einen Pfad!

Fréchet-Distanz

Freiraum
$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(p_1(s), p_2(t)) \leq \epsilon\}$
hat die Form einer Ellipse!

mehr als 1 Segment?

$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach $(1,1)$
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.



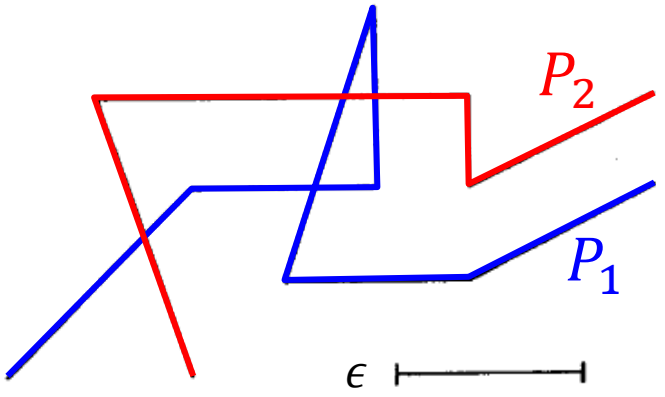
nicht monoton!

Für größeres ϵ gibt es einen Pfad!

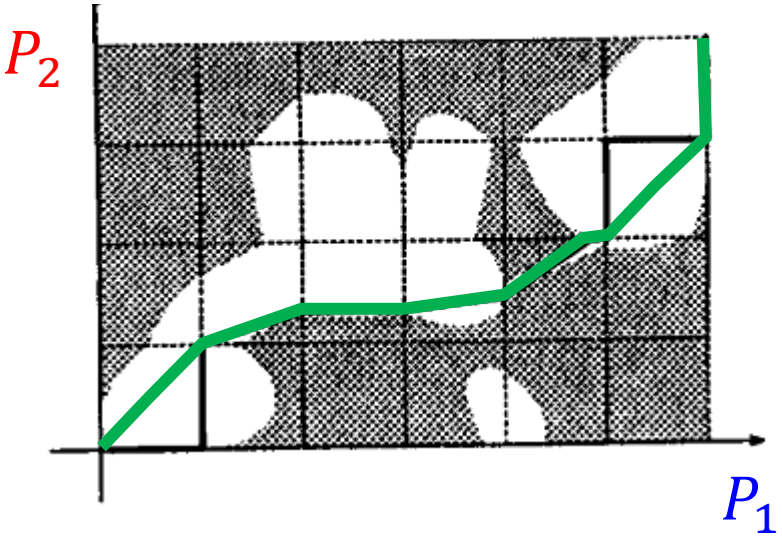
Fréchet-Distanz

Freiraum

$$F_\epsilon = \{(s,t) \in [0, m-1] \times [0, n-1] \mid d(p_1(s), p_2(t)) \leq \epsilon\}$$



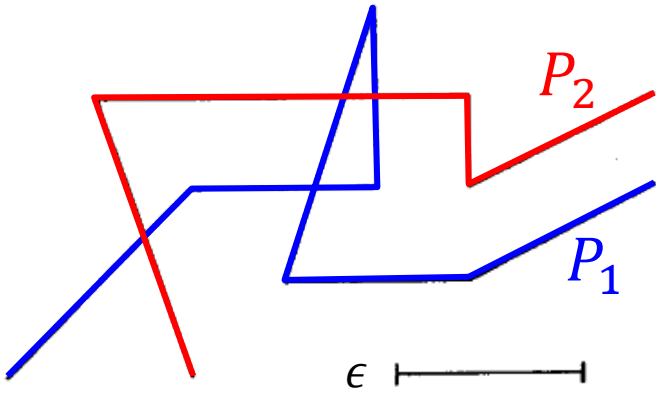
$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach (m,n)
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.



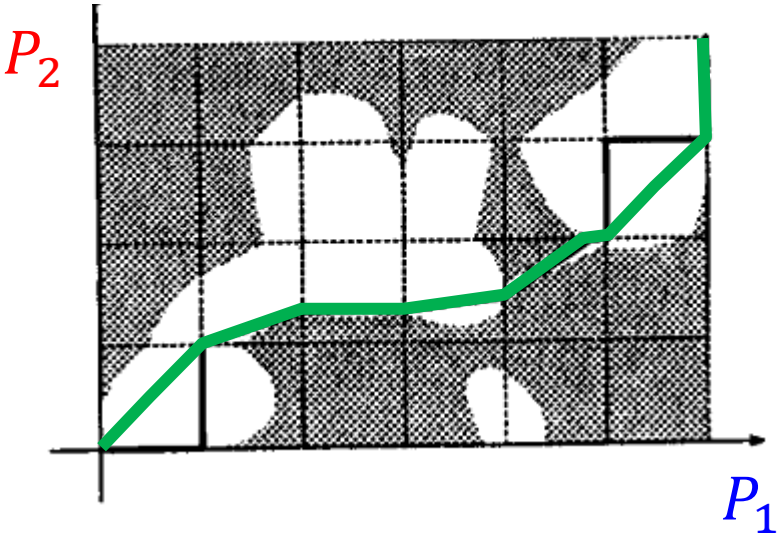
Fréchet-Distanz

Freiraum

$$F_\epsilon = \{(s,t) \in [0, m-1] \times [0, n-1] \mid d(p_1(s), p_2(t)) \leq \epsilon\}$$



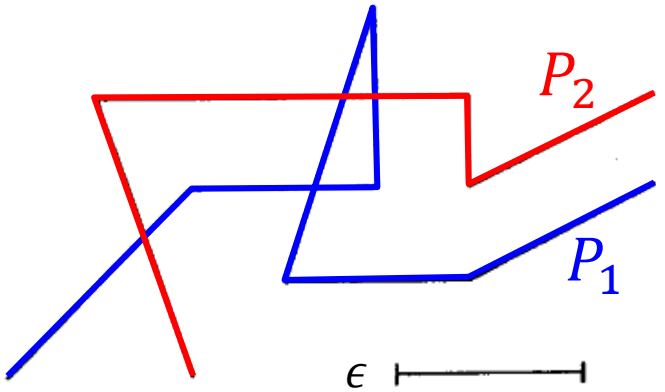
$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach (m,n)
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.



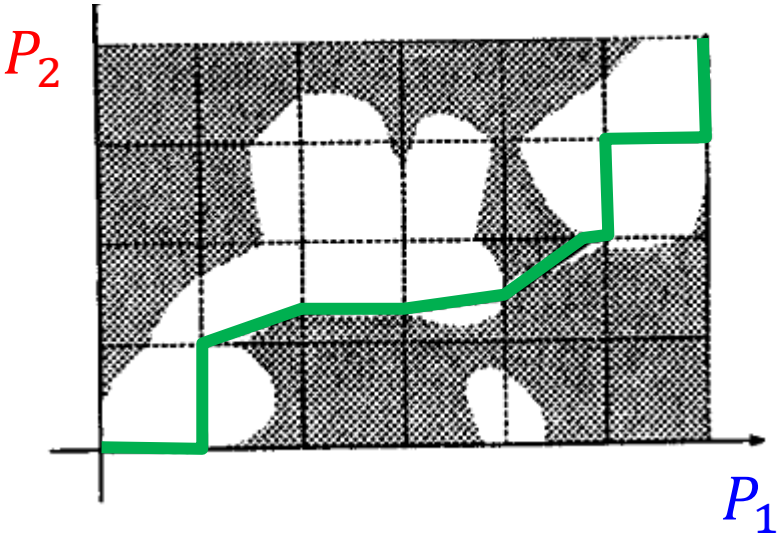
Fréchet-Distanz

Freiraum

$$F_\epsilon = \{(s,t) \in [0, m-1] \times [0, n-1] \mid d(p_1(s), p_2(t)) \leq \epsilon\}$$



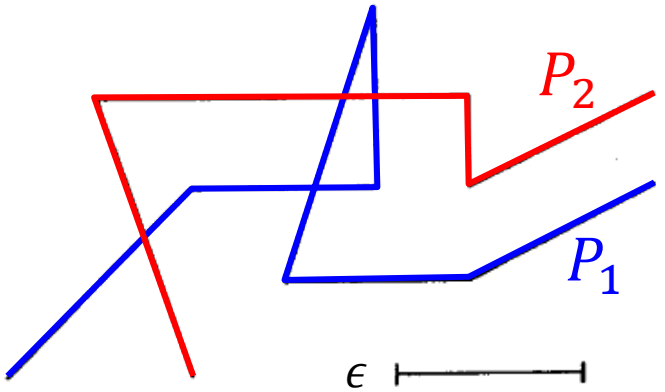
$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach (m,n)
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.



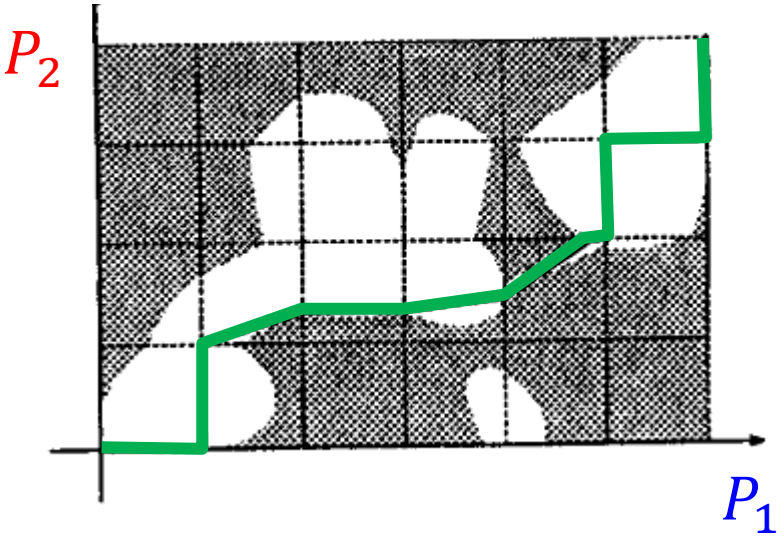
Fréchet-Distanz

Freiraum

$$F_\epsilon = \{(s,t) \in [0, m-1] \times [0, n-1] \mid d(p_1(s), p_2(t)) \leq \epsilon\}$$



$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach (m,n)
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.

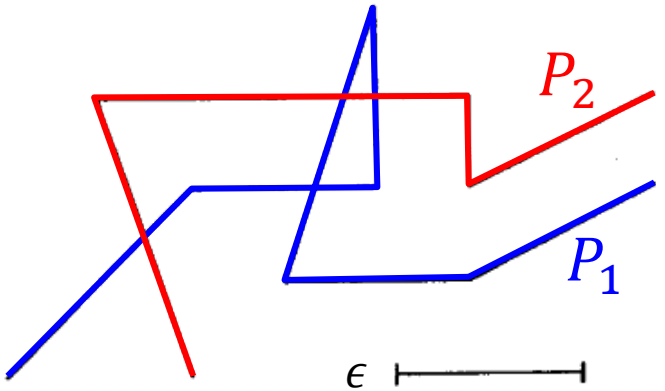


F_ϵ kann in $O(mn)$ Zeit
berechnet werden.

Fréchet-Distanz

Freiraum

$$F_\epsilon = \{(s,t) \in [0, m-1] \times [0, n-1] \mid d(p_1(s), p_2(t)) \leq \epsilon\}$$



$d_{\text{fréchet}} \leq \epsilon$
$\Leftrightarrow$ Es gibt einen Pfad
von $(0,0)$ nach (m,n)
im inneren von F_ϵ,
der in beide Richtungen
monoton ist.

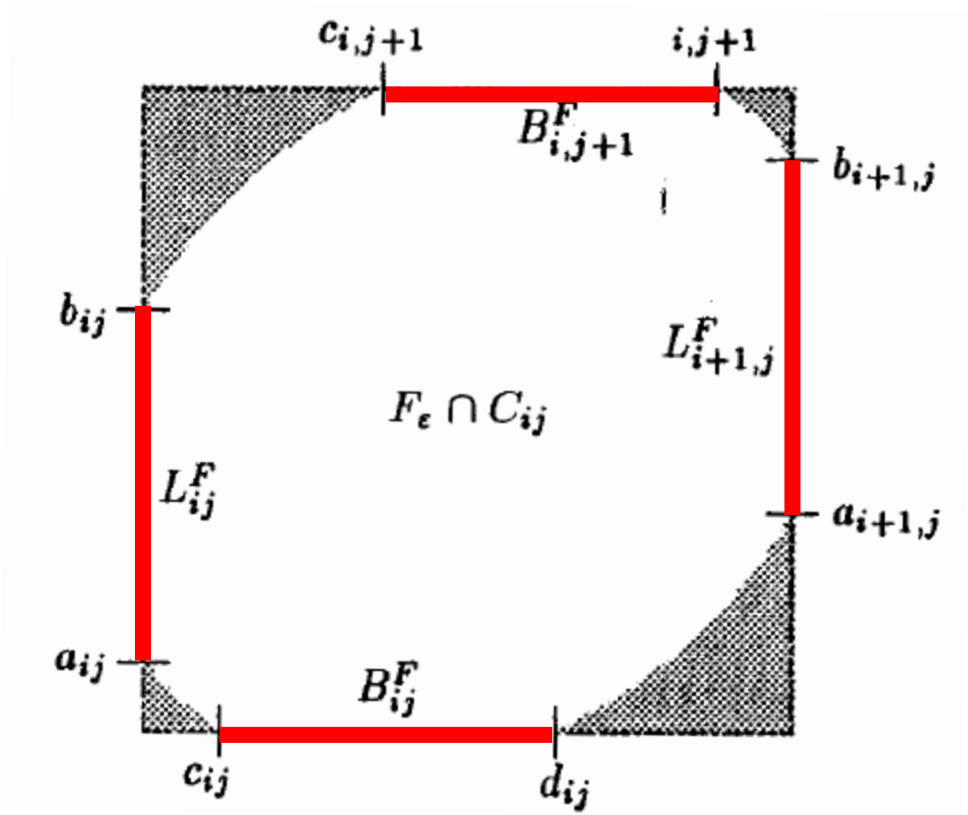
F_ϵ kann in $O(mn)$ Zeit
berechnet werden.

TODO:
Finde zulässigen Pfad in F_ϵ.

Fréchet-Distanz

Kanten einer Freiraumzelle:

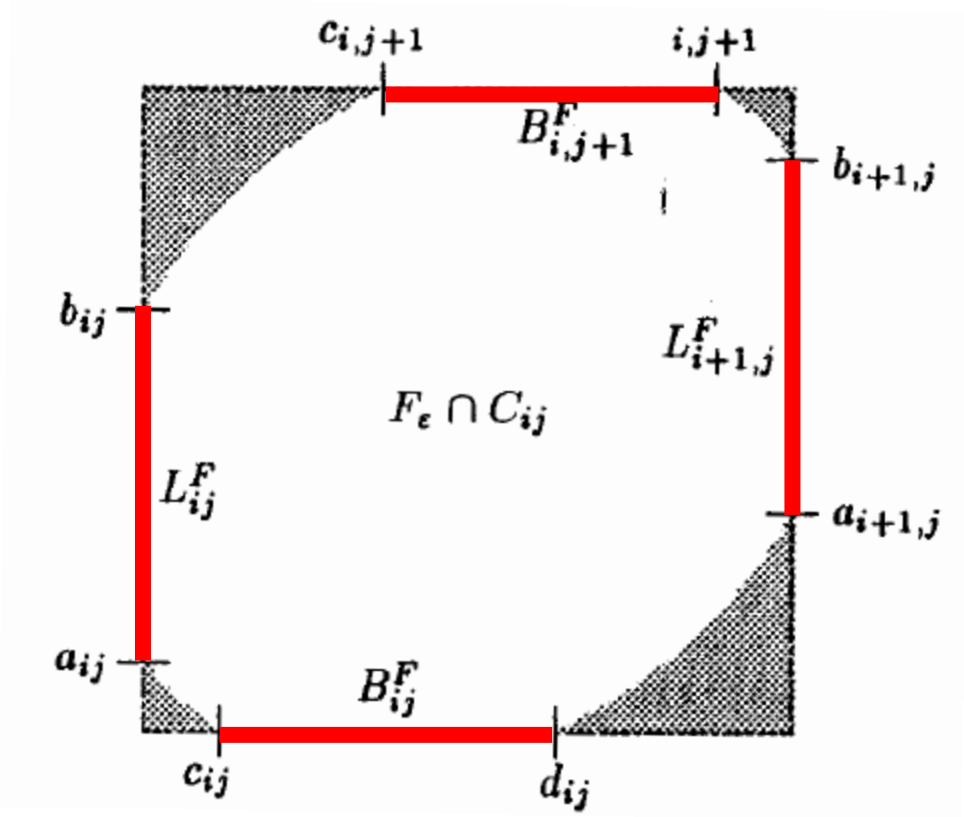
$B^F_{i,j}$, $B^F_{i,j+1}$, $L^F_{i,j}$, $L^F_{i+1,j}$



Fréchet-Distanz

Kanten einer Freiraumzelle:

$B^F_{i,j}$, $B^F_{i,j+1}$, $L^F_{i,j}$, $L^F_{i+1,j}$



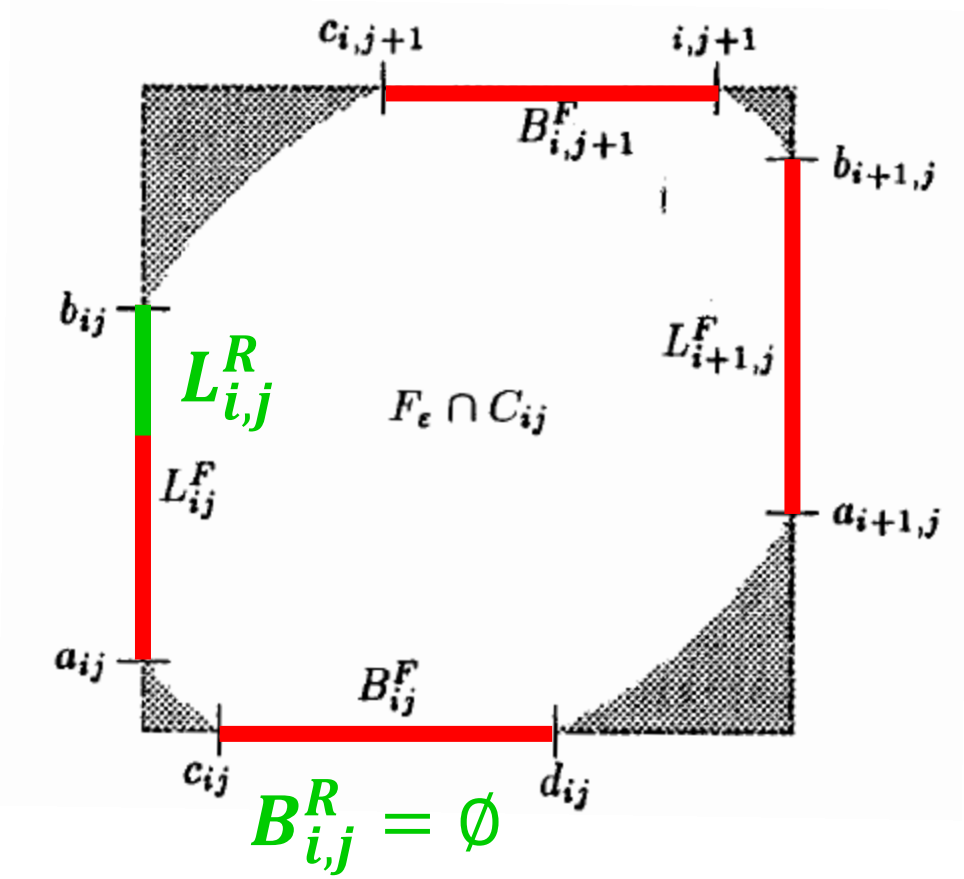
Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:

$B^R_{i,j}$, $B^R_{i,j+1}$, $L^R_{i,j}$, $L^R_{i+1,j}$

Fréchet-Distanz

Kanten einer Freiraumzelle:

$B^F_{i,j}$, $B^F_{i,j+1}$, $L^F_{i,j}$, $L^F_{i+1,j}$



Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:

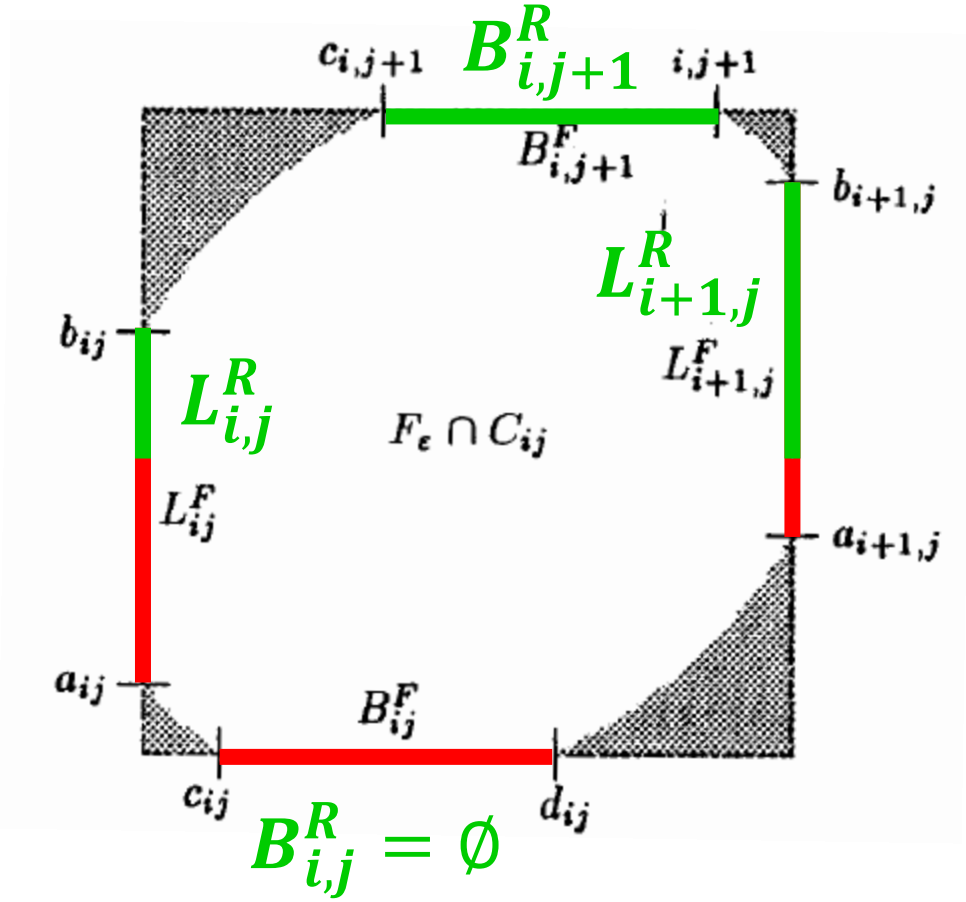
$B^R_{i,j}$, $B^R_{i,j+1}$, $L^R_{i,j}$, $L^R_{i+1,j}$

$B^R_{i,j}$, $L^R_{i,j}$ bekannt

Fréchet-Distanz

Kanten einer Freiraumzelle:

$B^F_{i,j}$, $B^F_{i,j+1}$, $L^F_{i,j}$, $L^F_{i+1,j}$



Teile dieser Kanten, die von $(0,0)$ aus erreichbar sind:

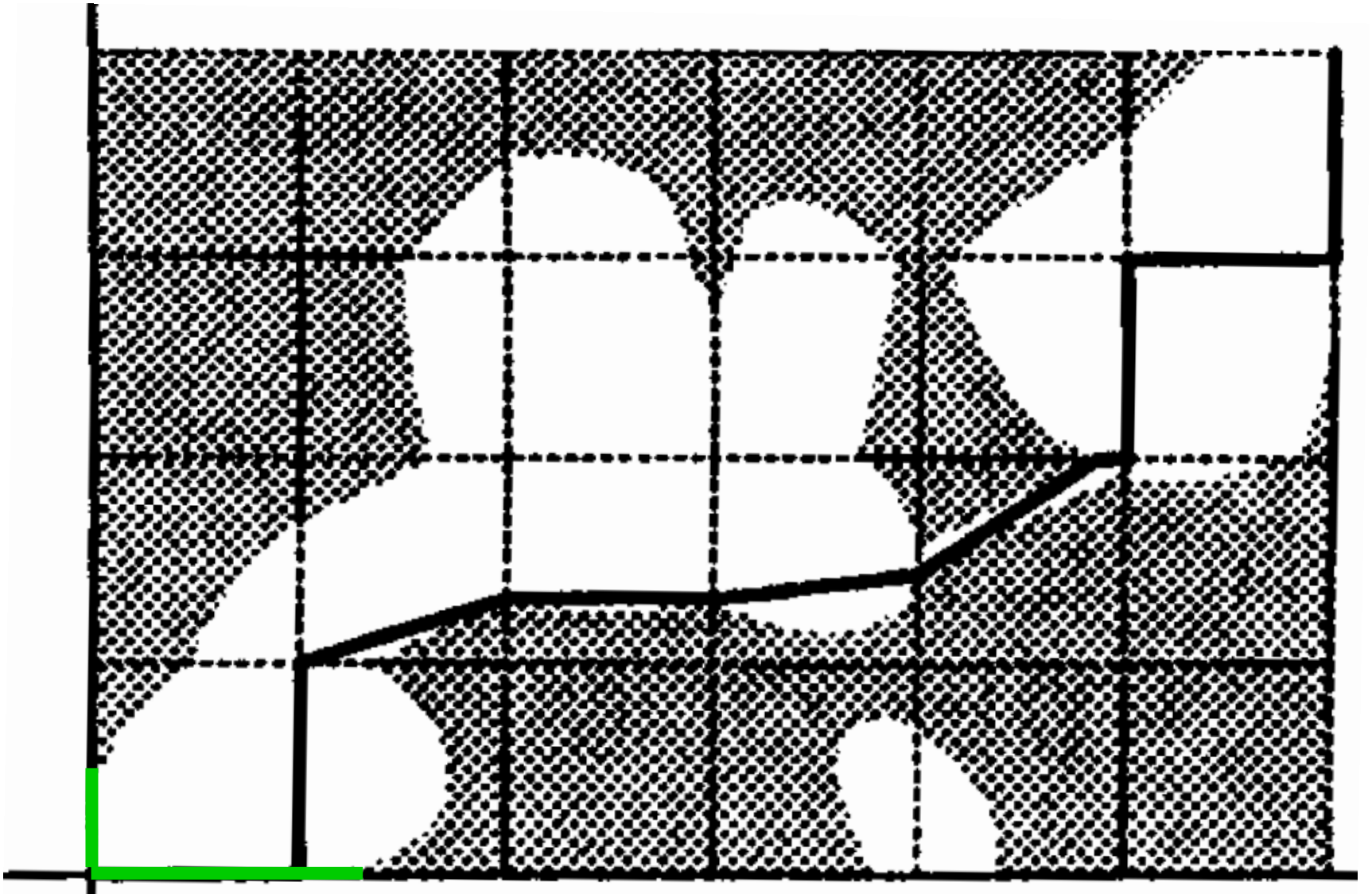
$B^R_{i,j}$, $B^R_{i,j+1}$, $L^R_{i,j}$, $L^R_{i+1,j}$

$B^R_{i,j}$, $L^R_{i,j}$ bekannt

$\Rightarrow B^R_{i,j+1}$, $L^R_{i+1,j}$

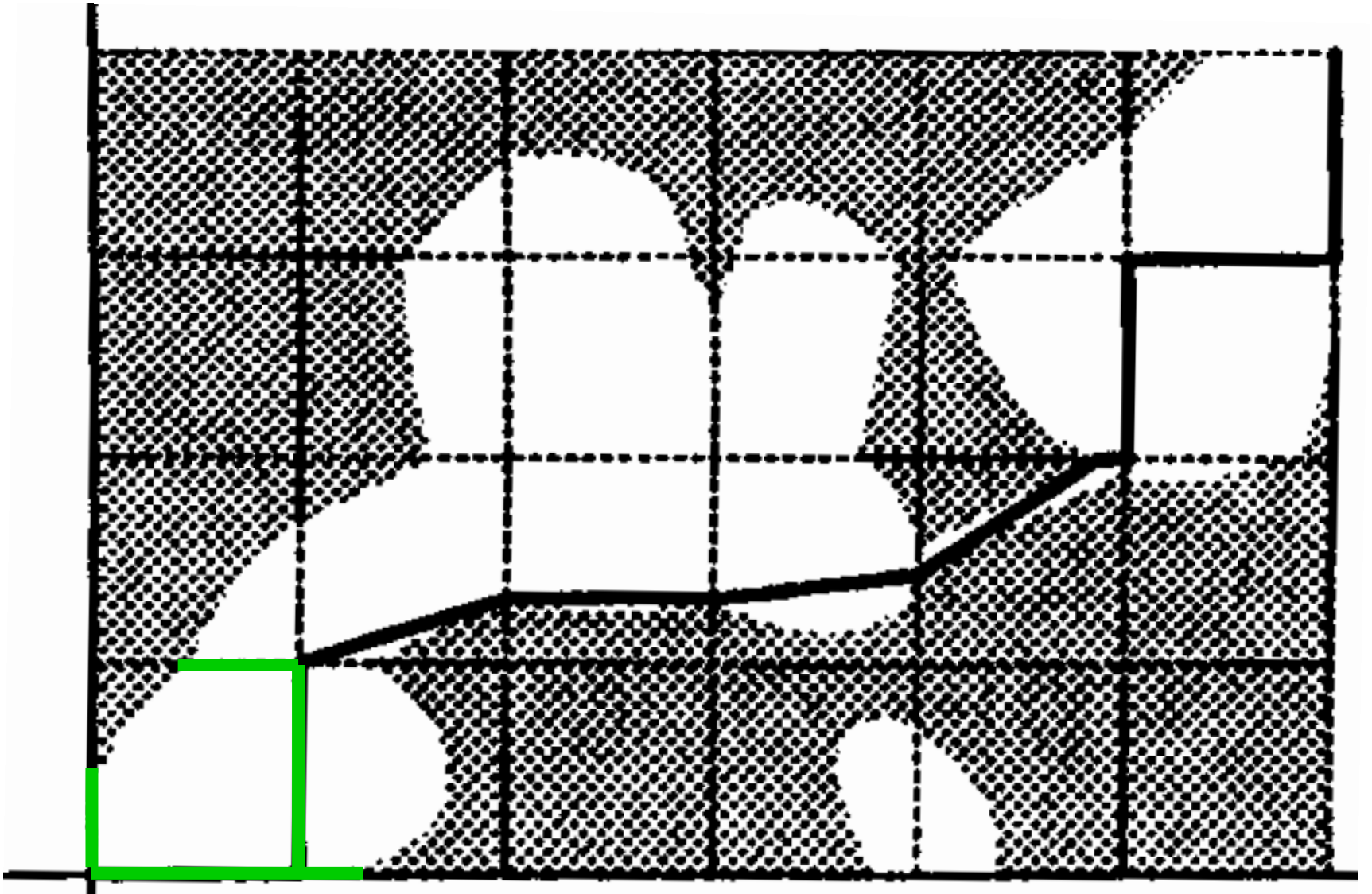
Fréchet-Distanz

Algorithmus



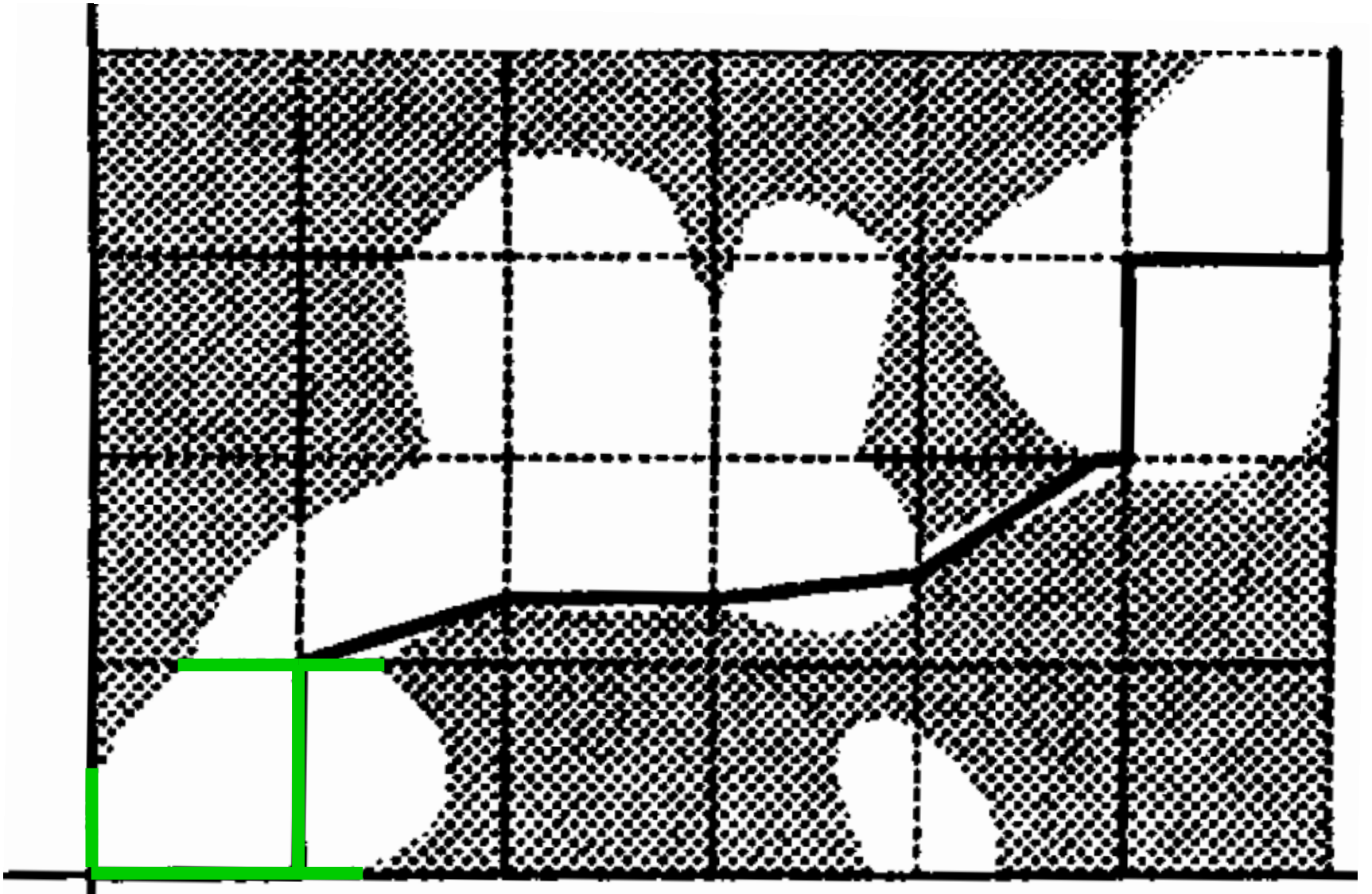
Fréchet-Distanz

Algorithmus



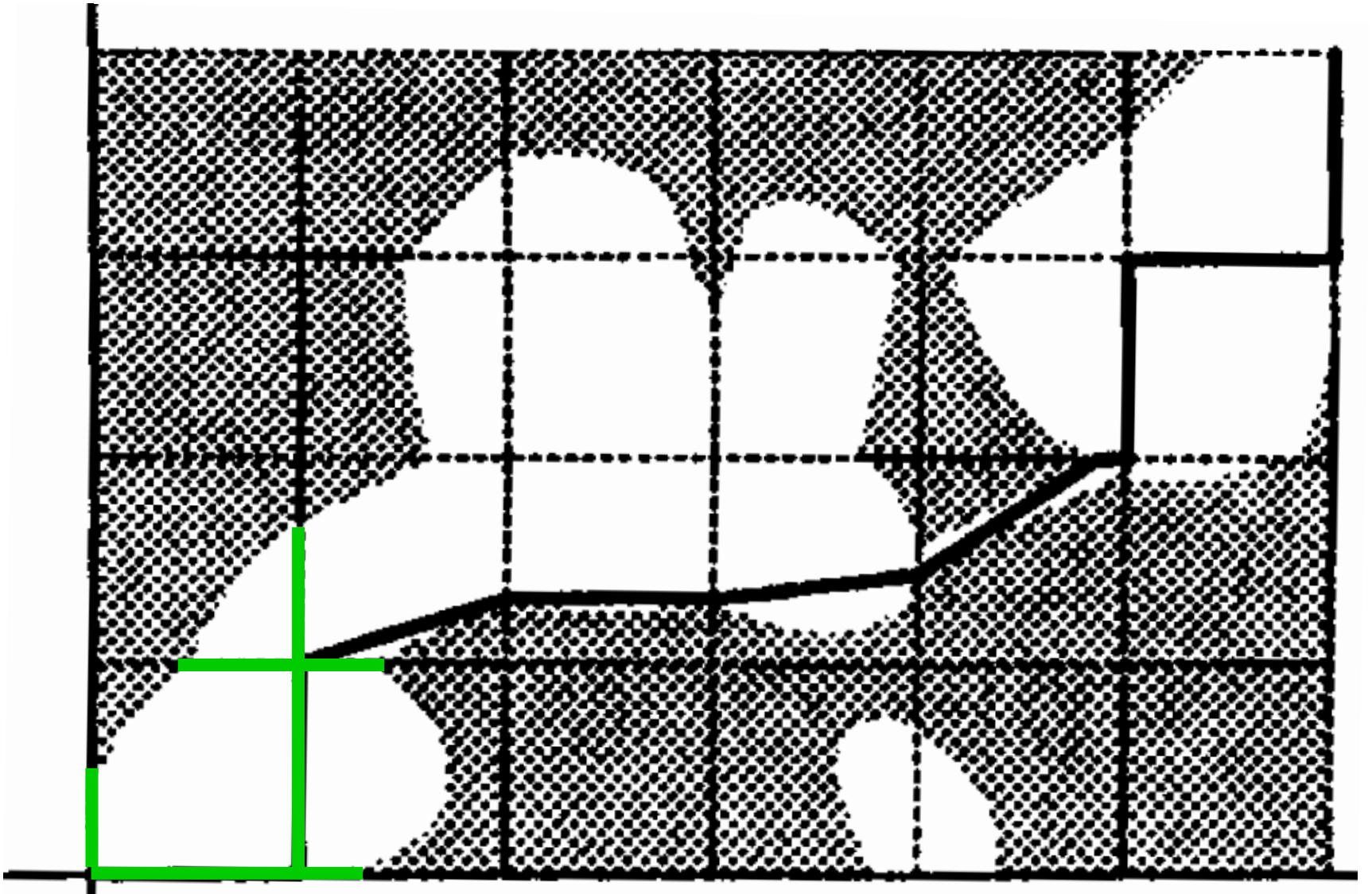
Fréchet-Distanz

Algorithmus



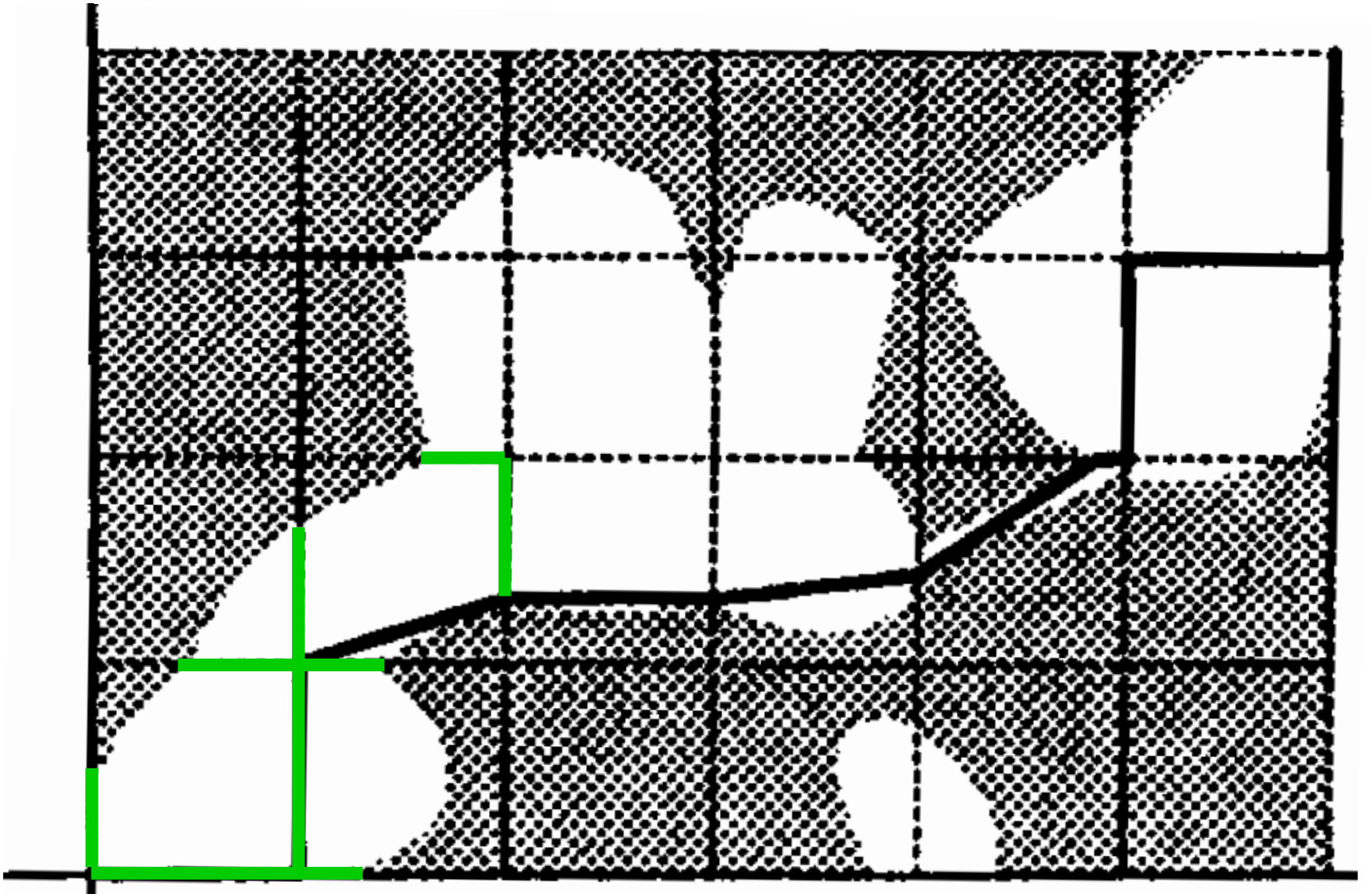
Fréchet-Distanz

Algorithmus



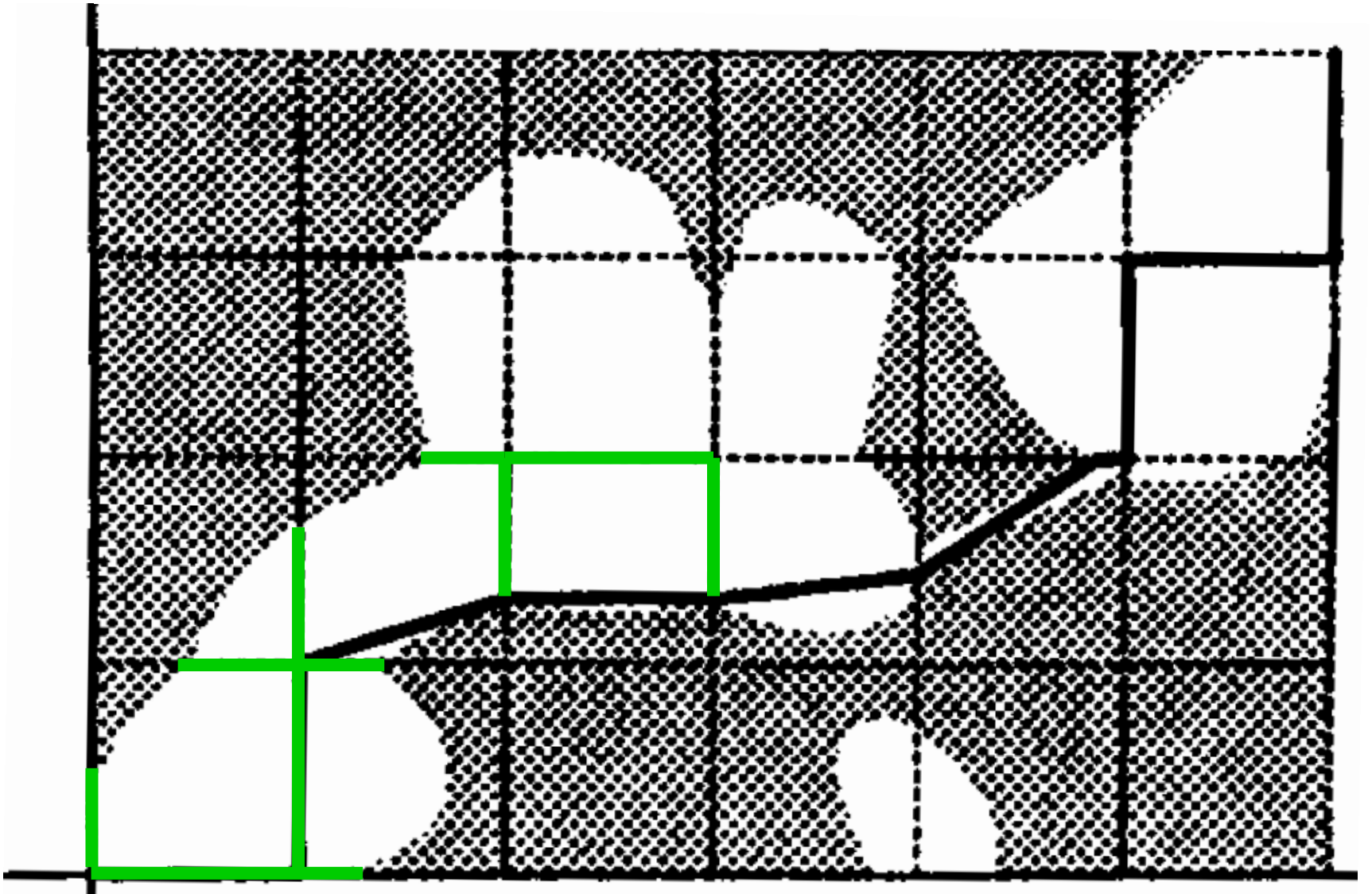
Fréchet-Distanz

Algorithmus



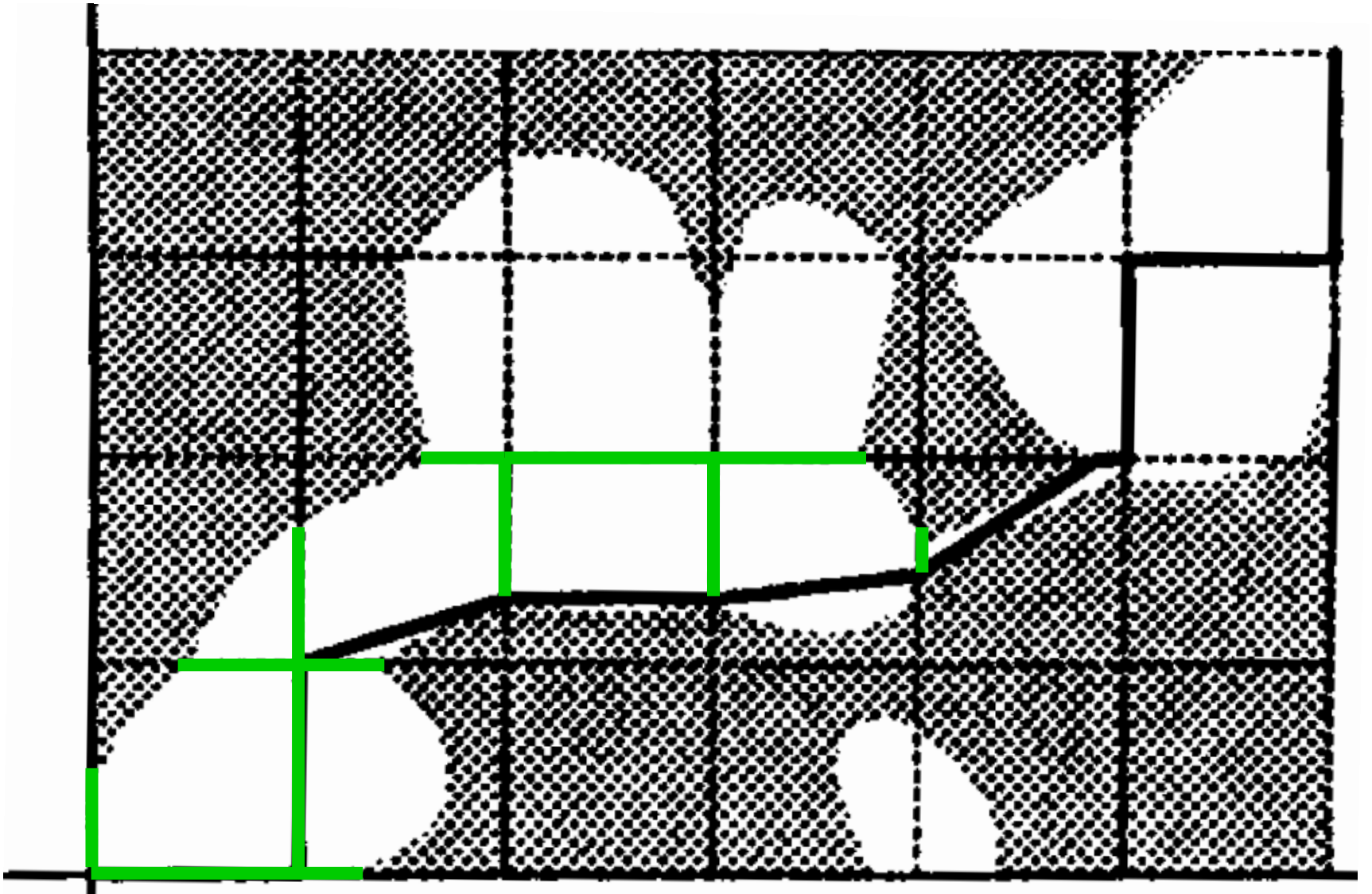
Fréchet-Distanz

Algorithmus



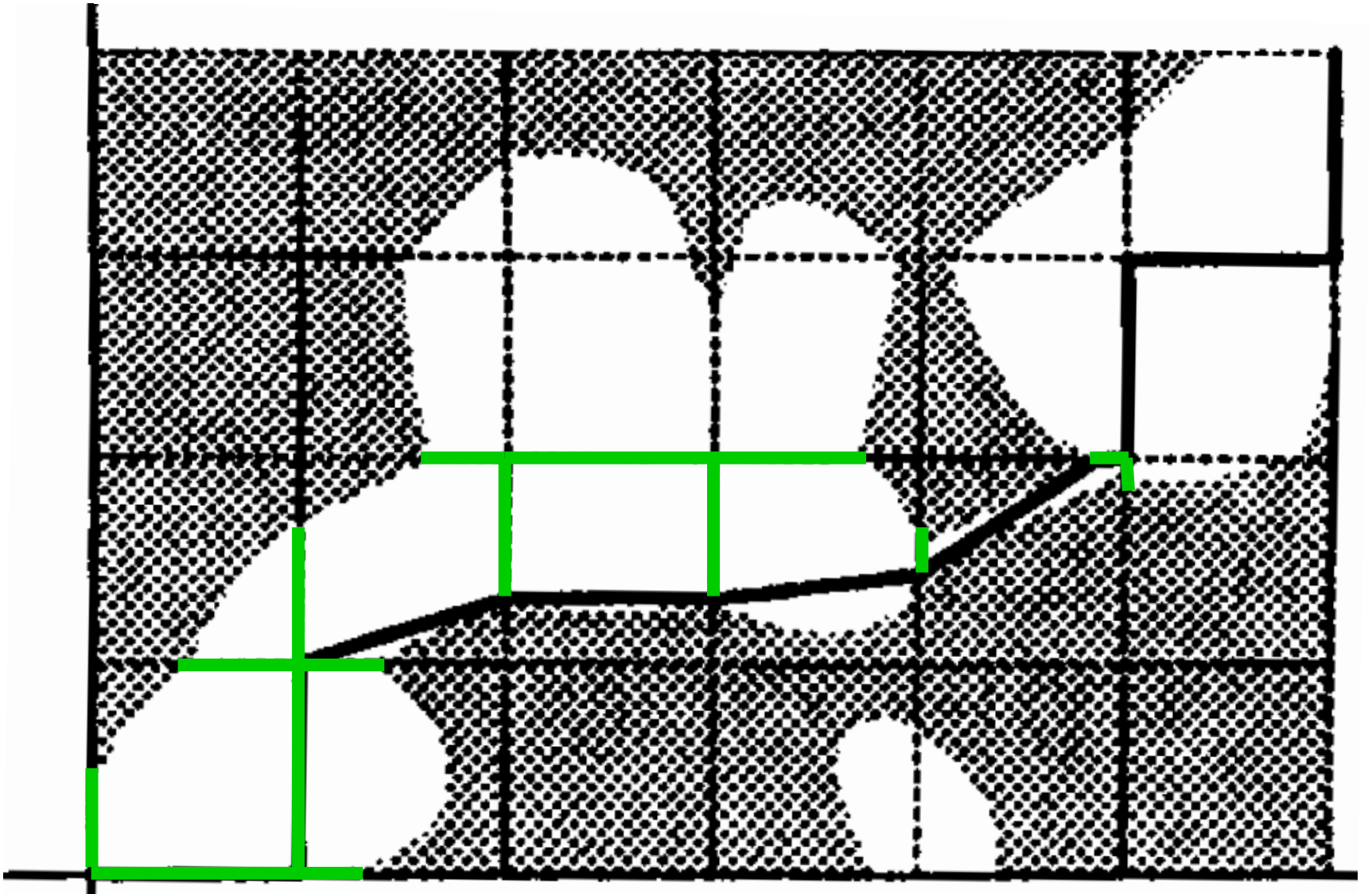
Fréchet-Distanz

Algorithmus



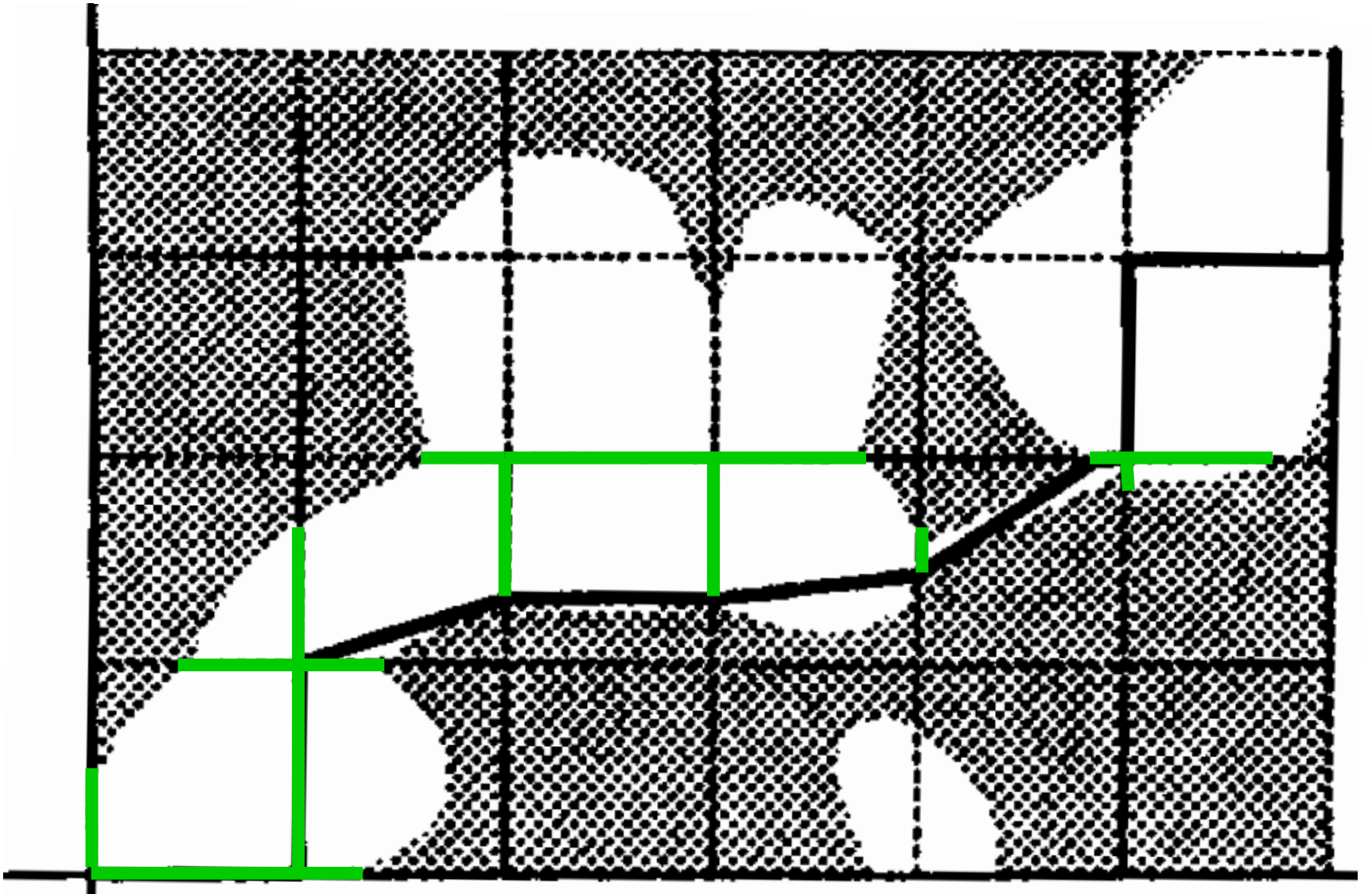
Fréchet-Distanz

Algorithmus



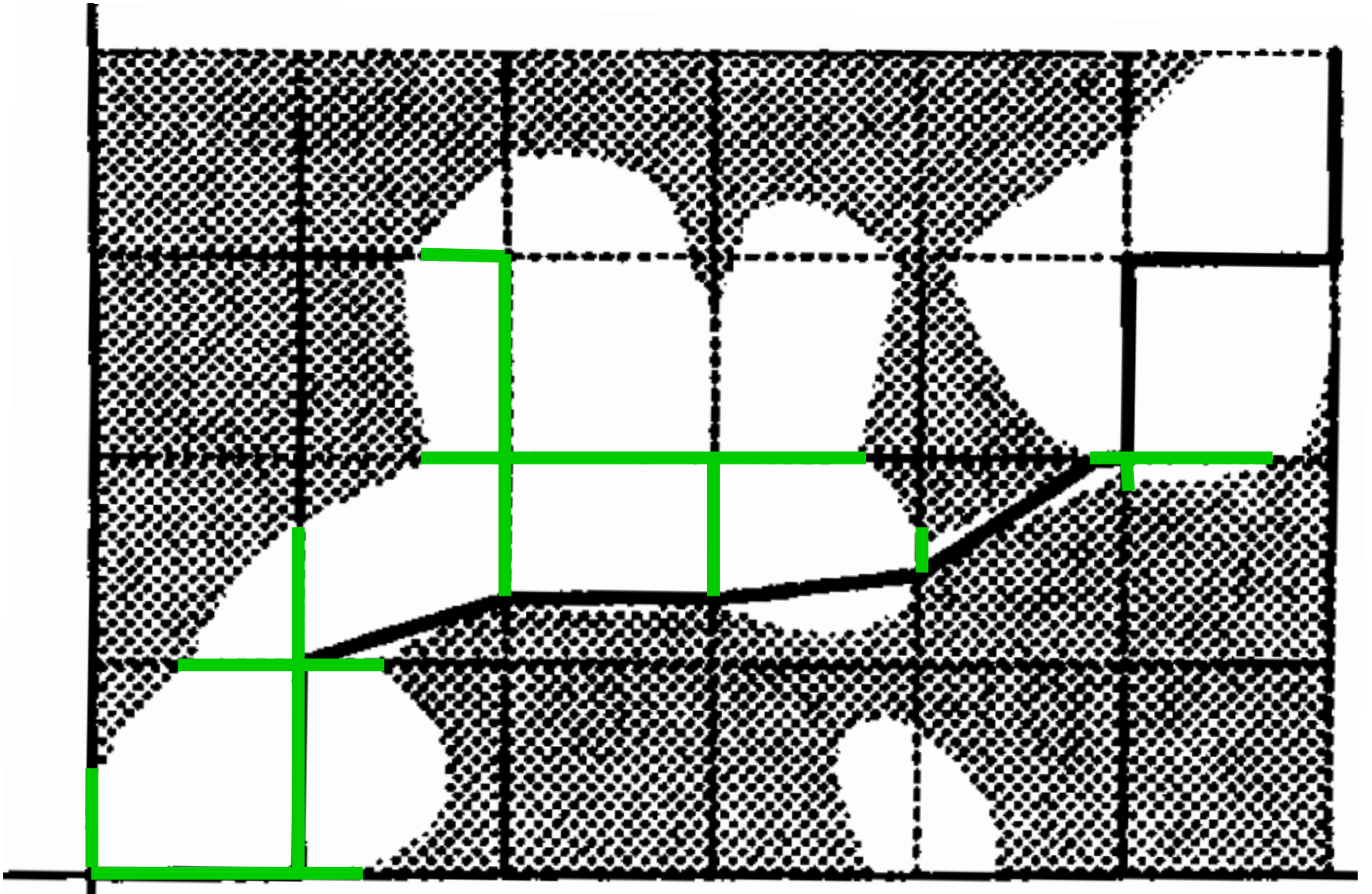
Fréchet-Distanz

Algorithmus



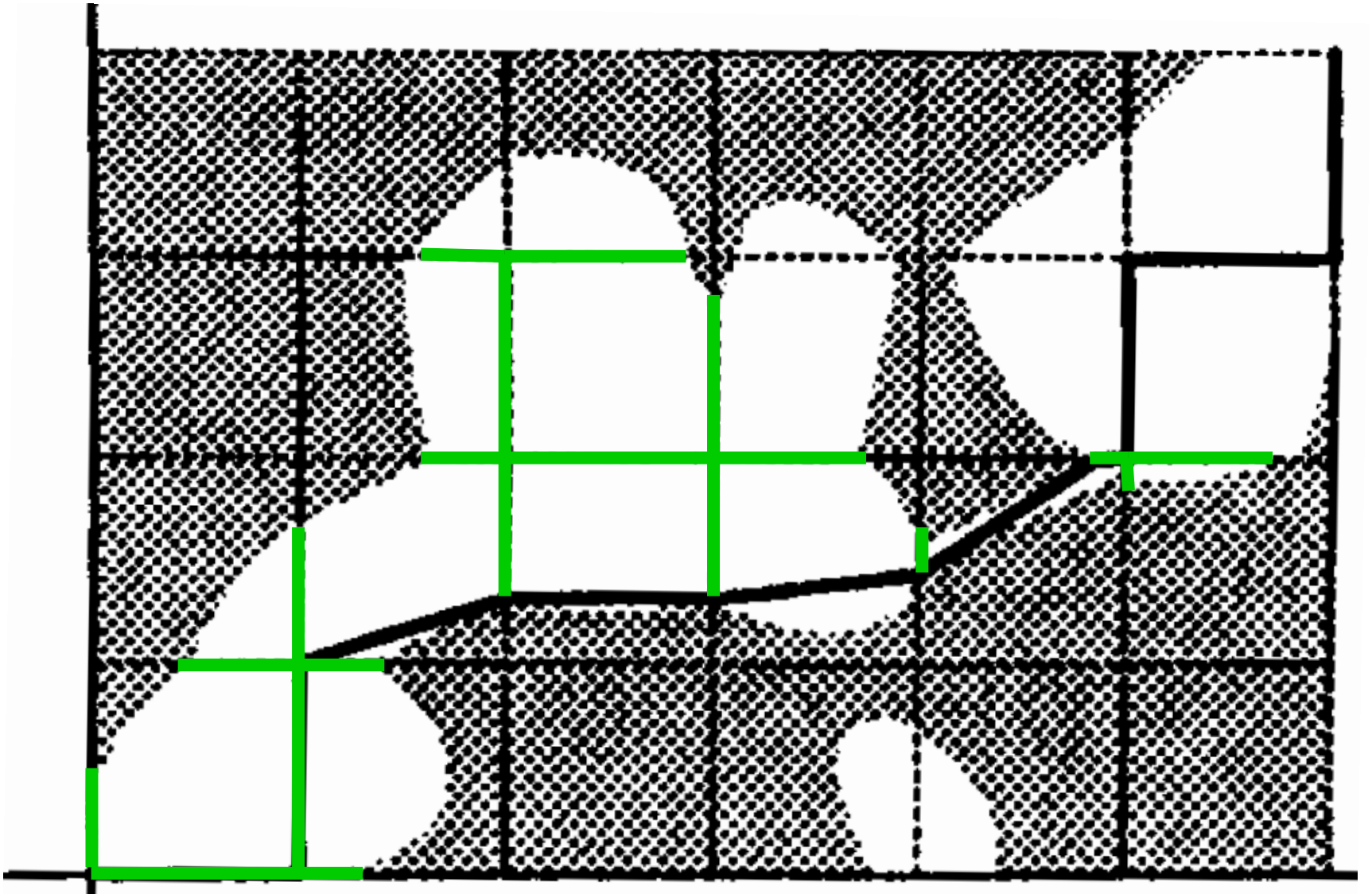
Fréchet-Distanz

Algorithmus



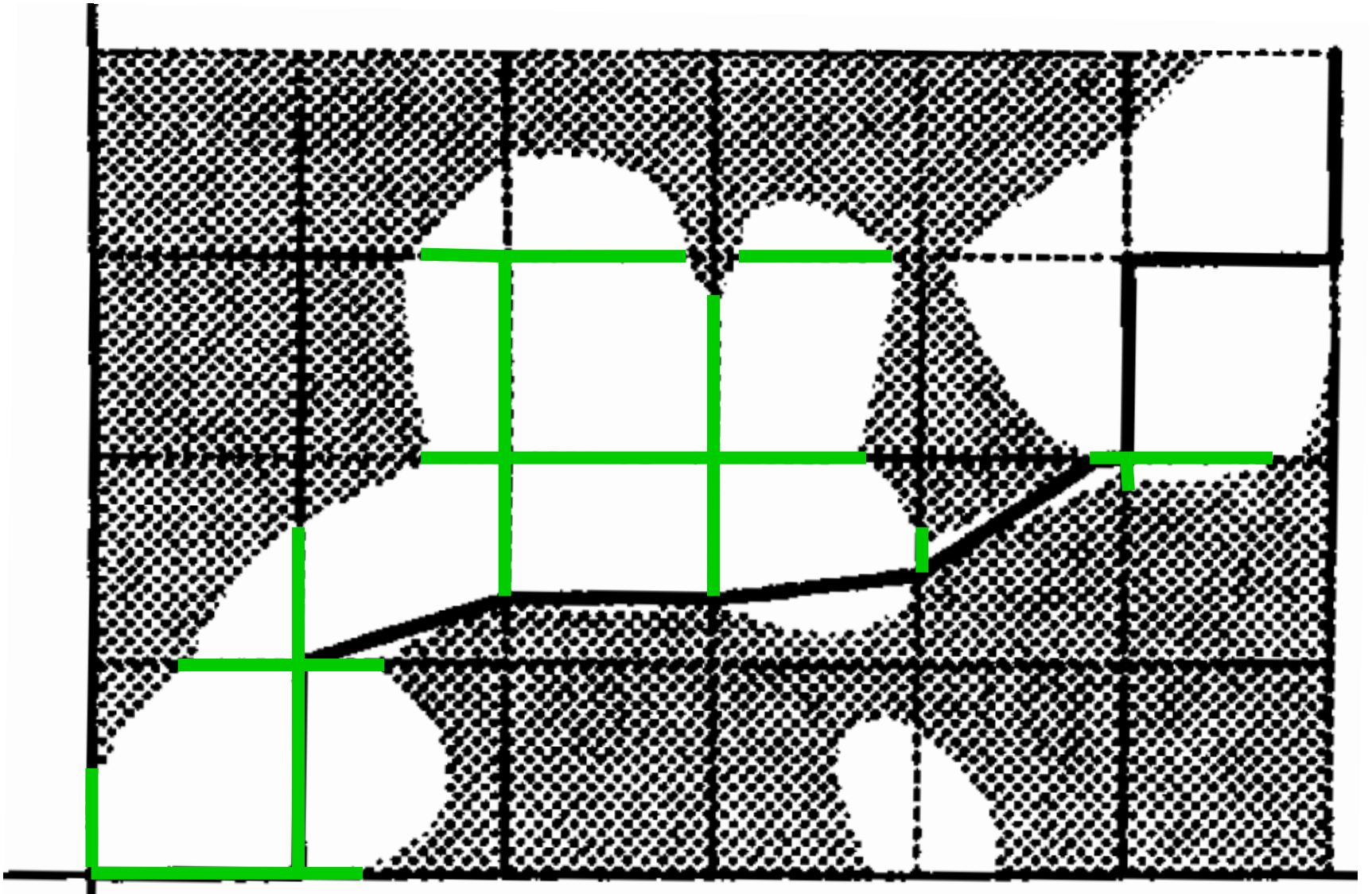
Fréchet-Distanz

Algorithmus



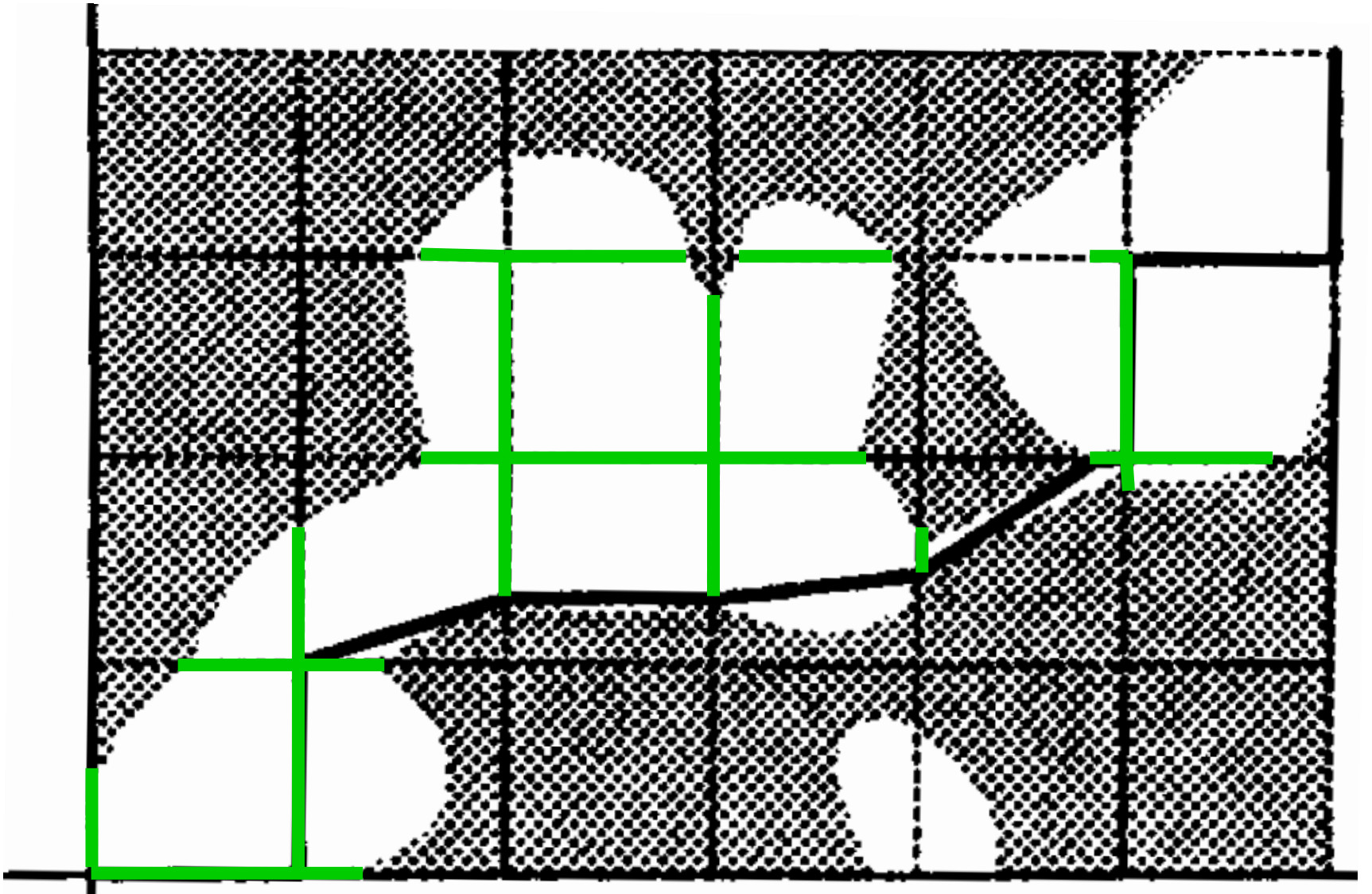
Fréchet-Distanz

Algorithmus



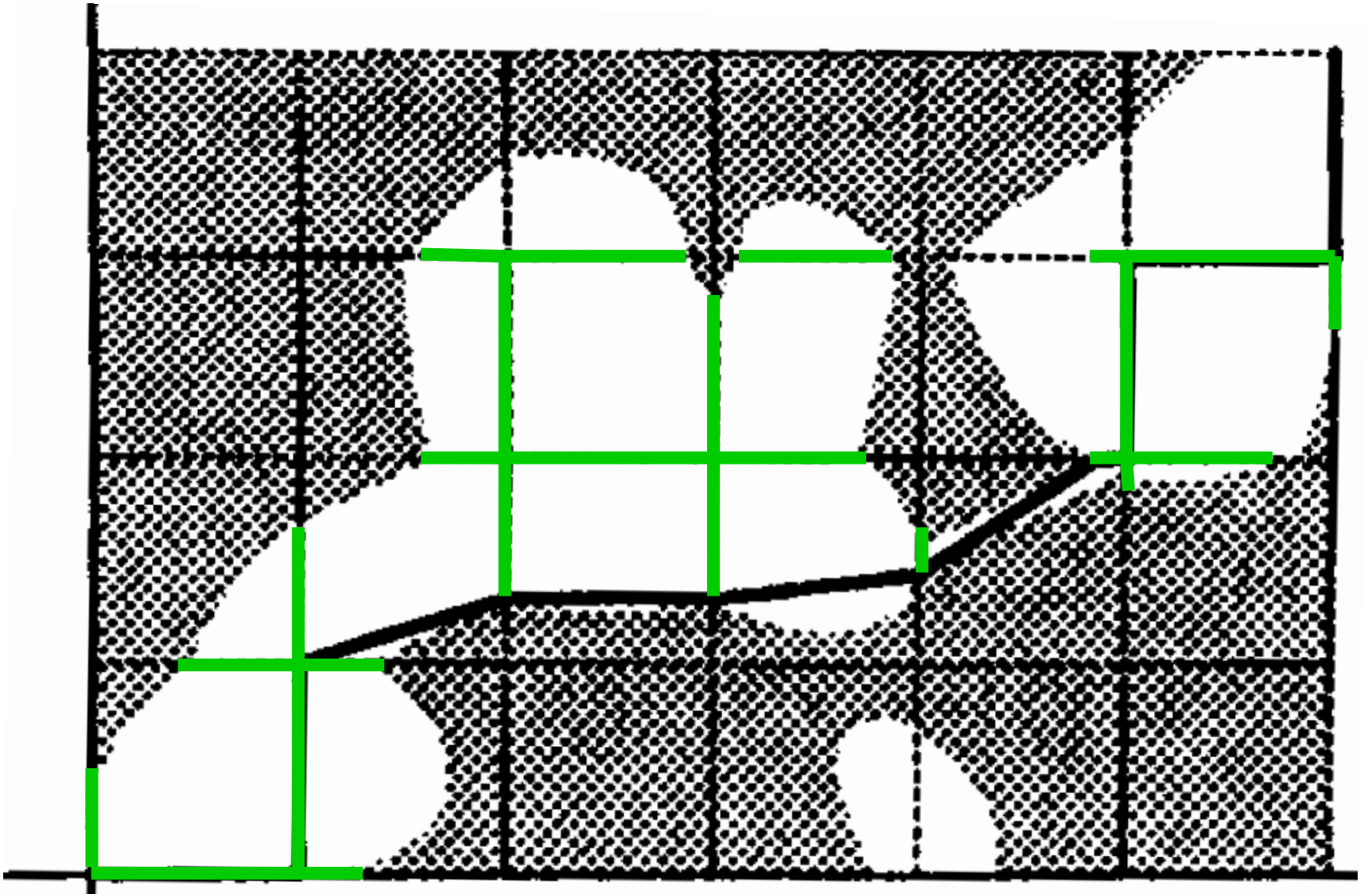
Fréchet-Distanz

Algorithmus



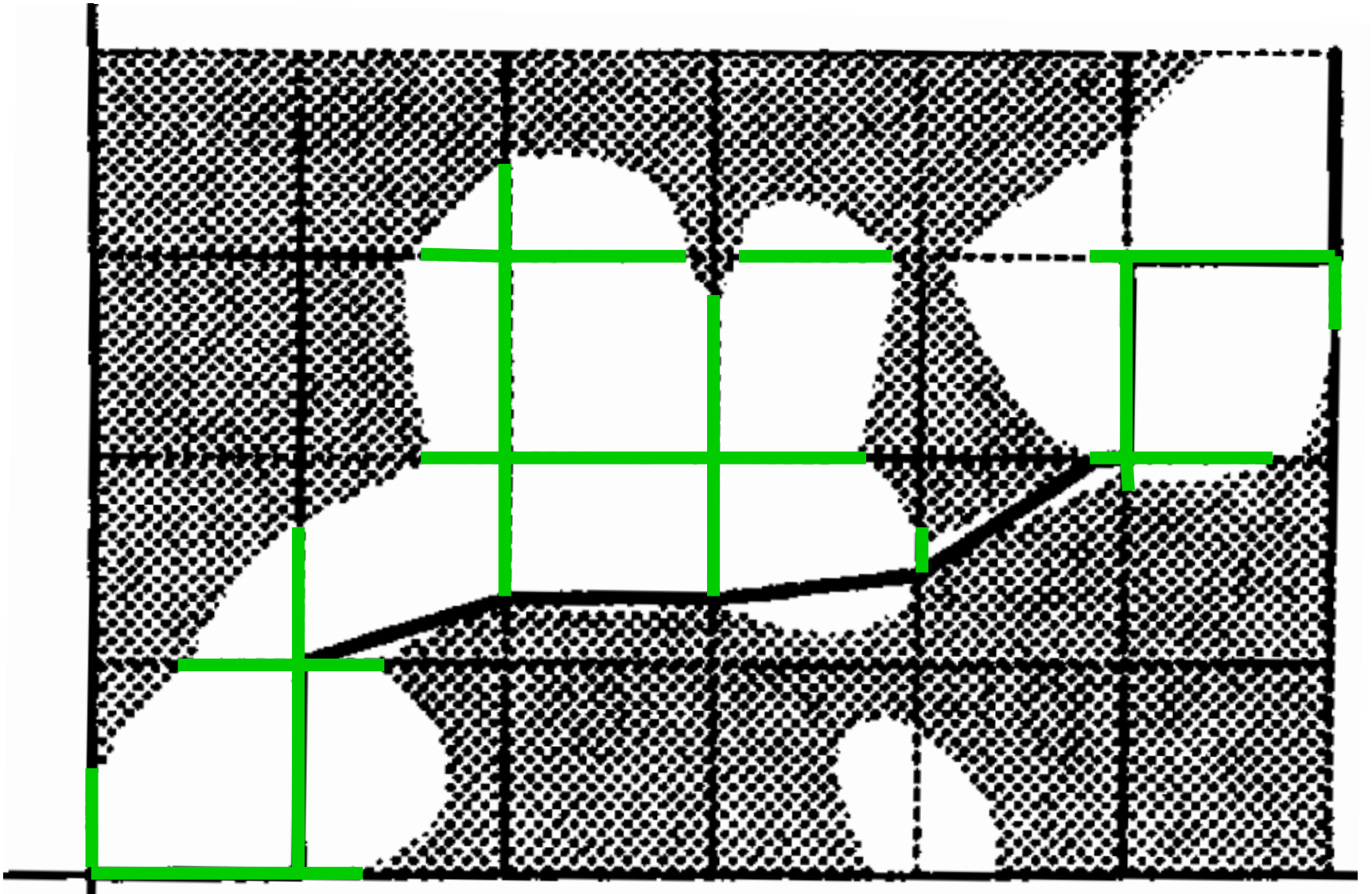
Fréchet-Distanz

Algorithmus



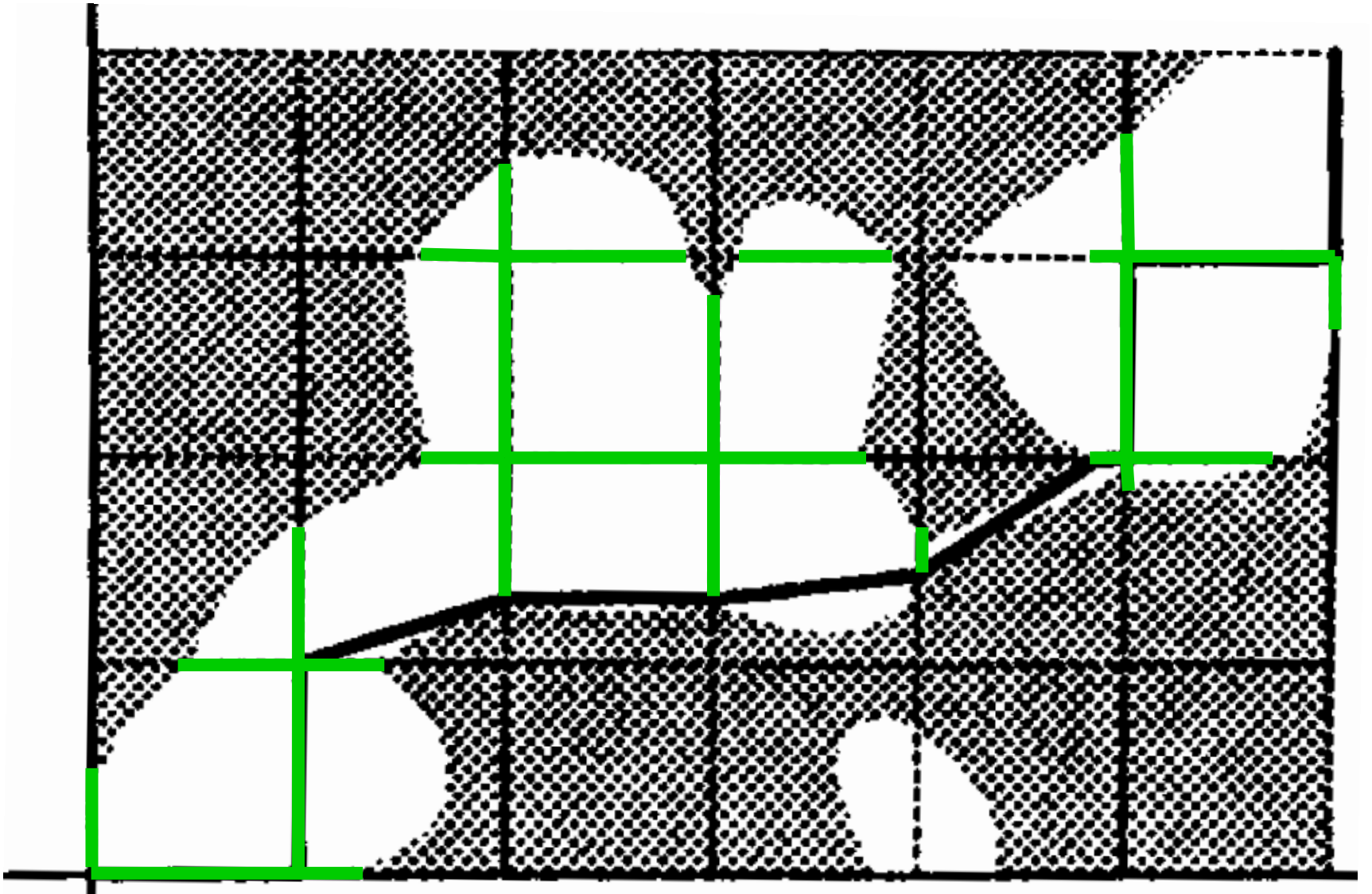
Fréchet-Distanz

Algorithmus



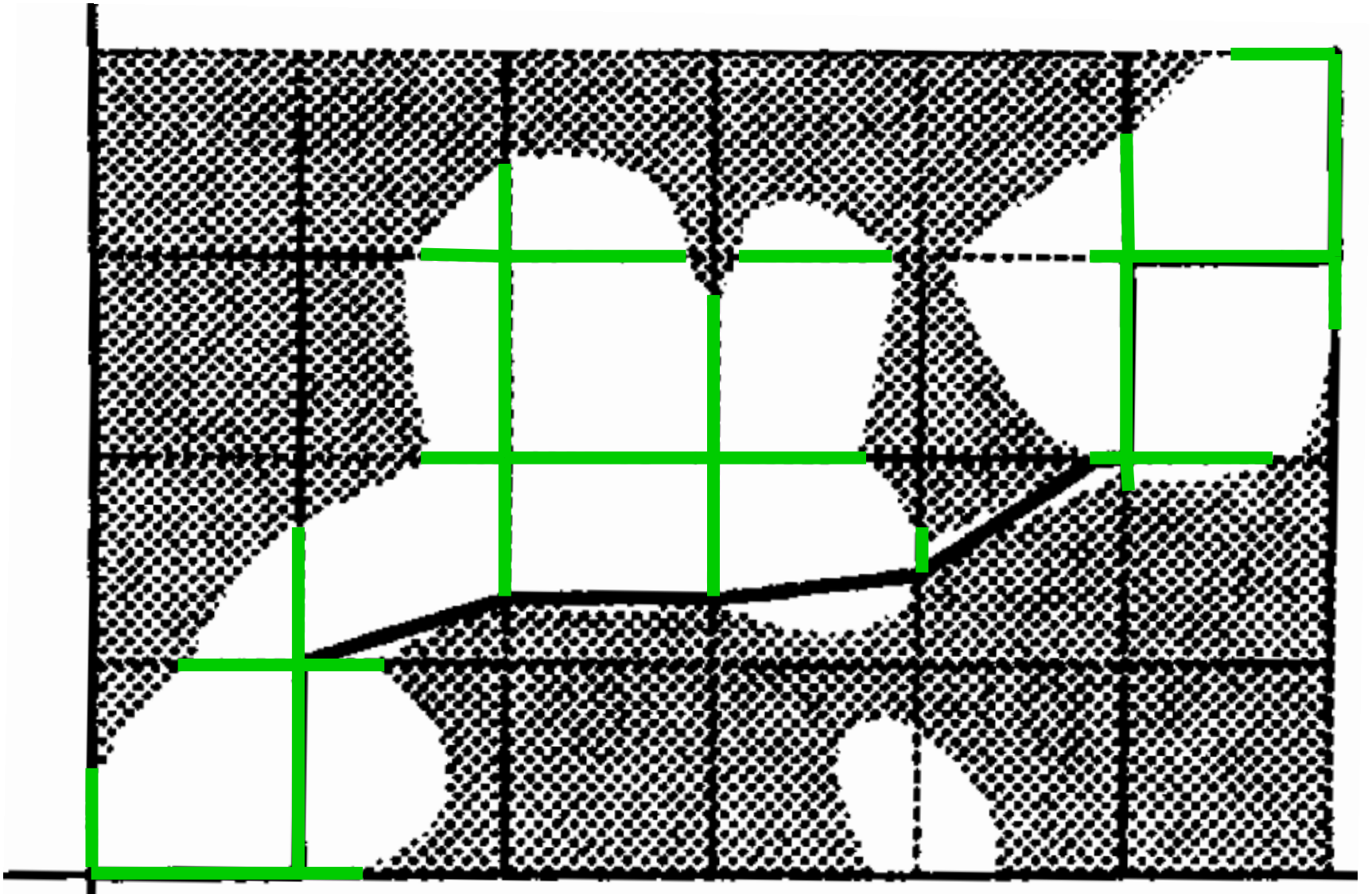
Fréchet-Distanz

Algorithmus



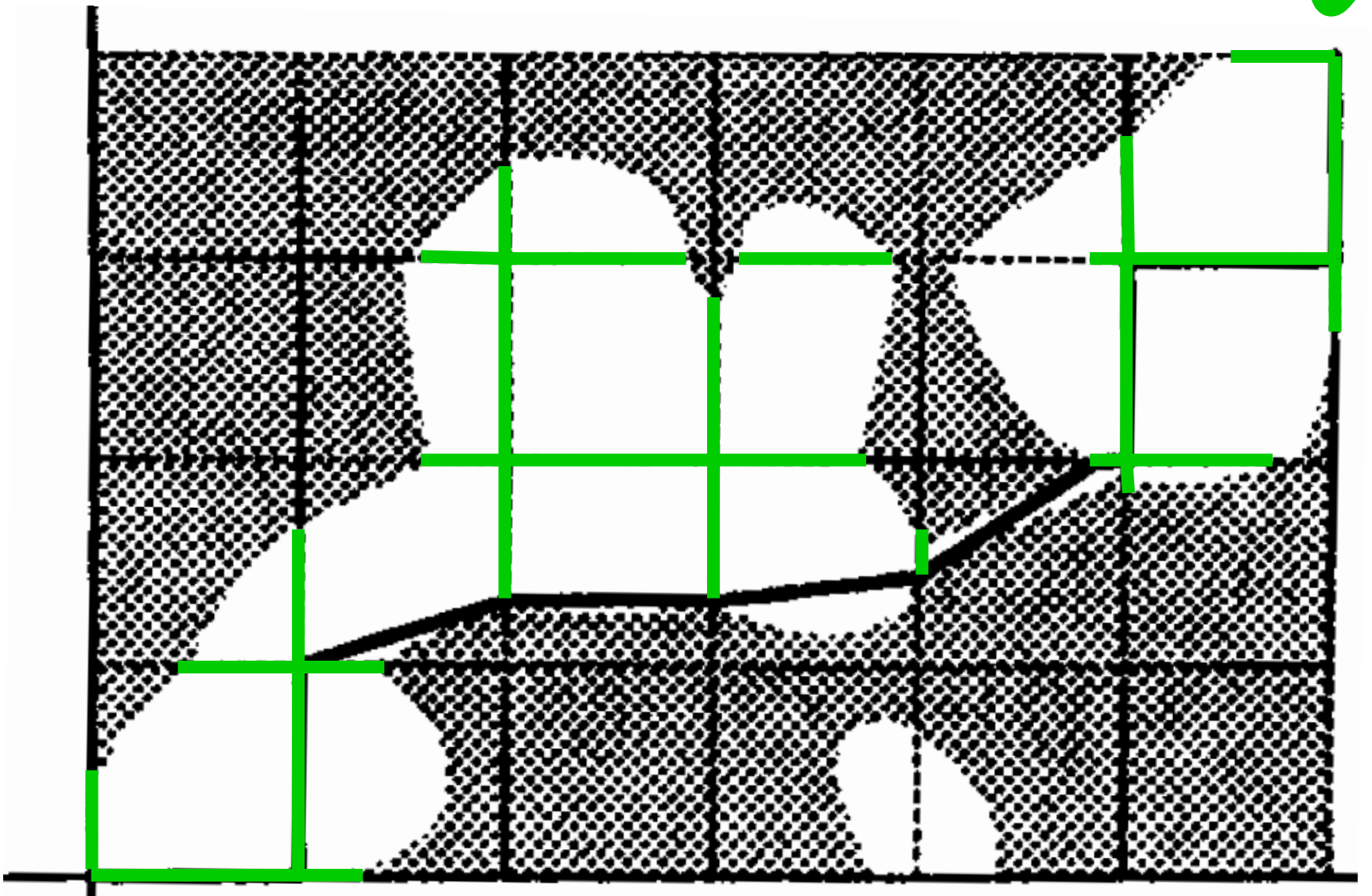
Fréchet-Distanz

Algorithmus



Fréchet-Distanz

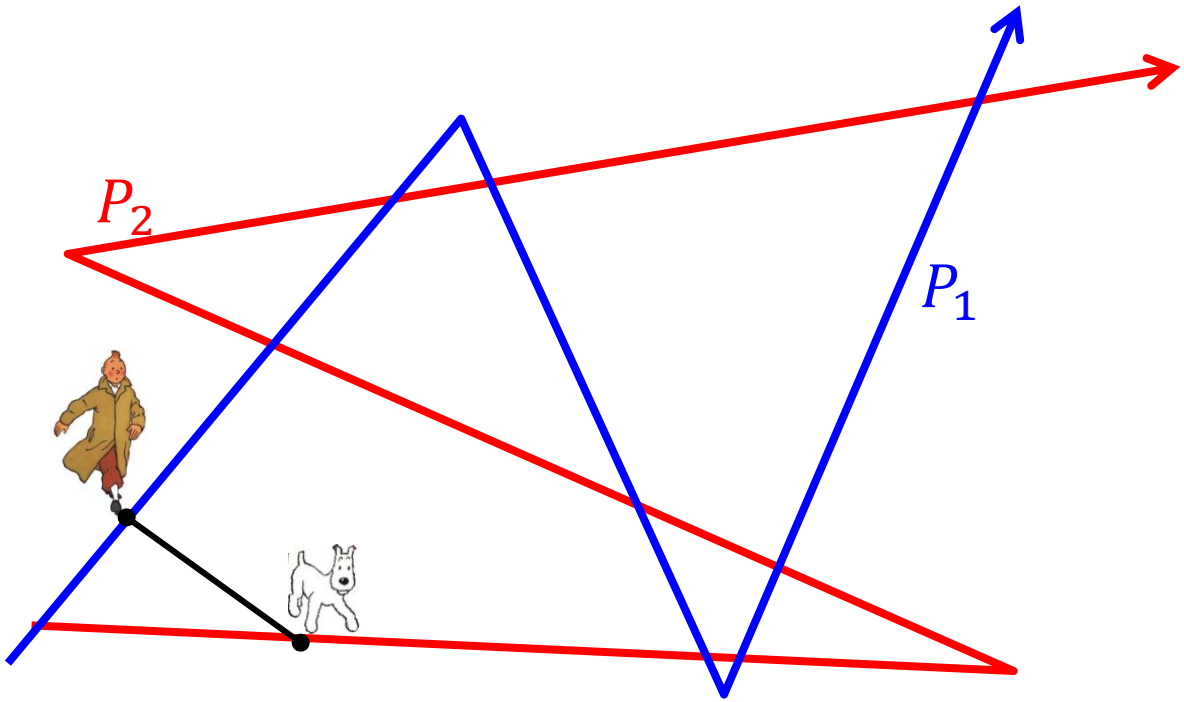
Algorithmus



Fréchet-Distanz

Berechnung:

Löse Entscheidungsproblem: Ist $d_{\text{fréchet}} \leq \epsilon$?



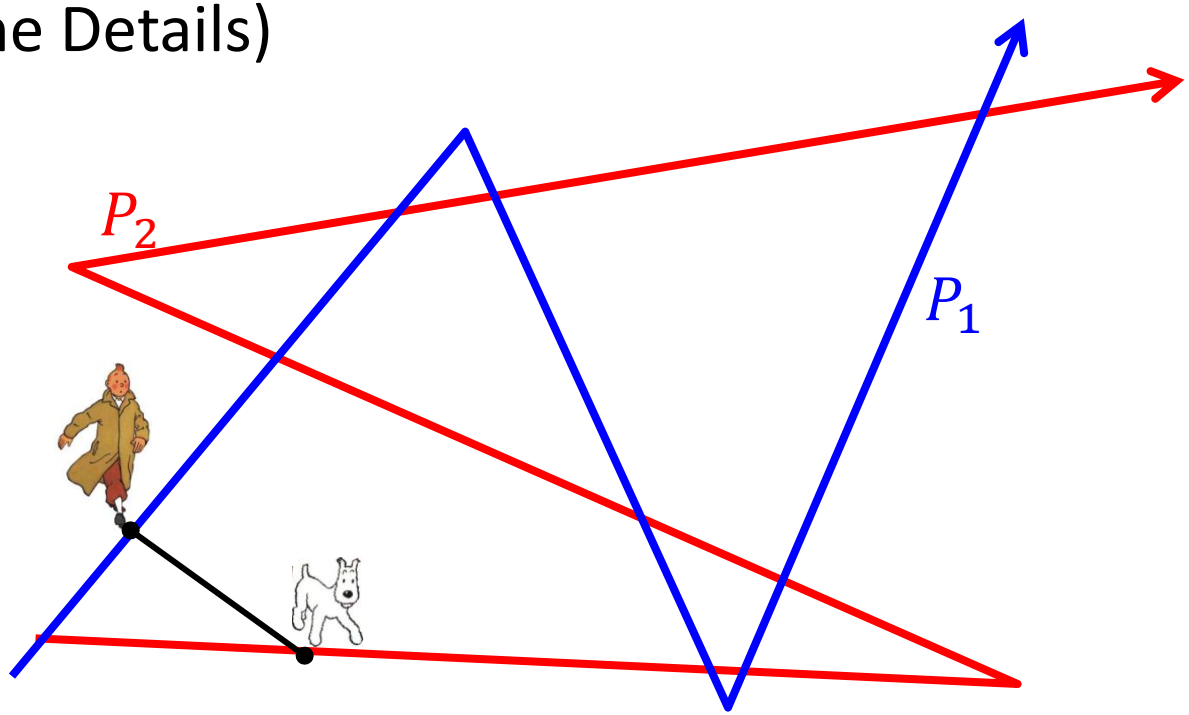
Fréchet-Distanz

Berechnung:

Löse Entscheidungsproblem: Ist $d_{\text{fréchet}} \leq \epsilon$?

dann parametrische Suche... $O(mn \log mn)$ Zeit

(hier ohne Details)

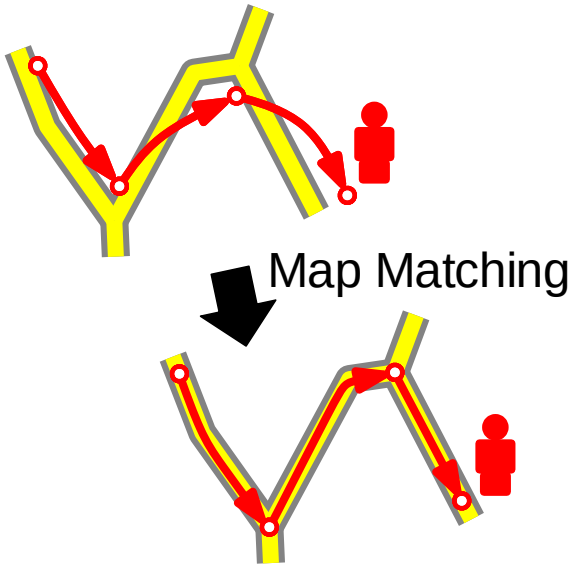


Map Matching

Problemformulierung

Gegeben:
Das Straßennetz als planar eingebetteter Graph $G = (V, E)$
Die GPS-Trajektorie als Folge $P = (p_1, p_2, \ldots, p_n)$ von Punkten

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

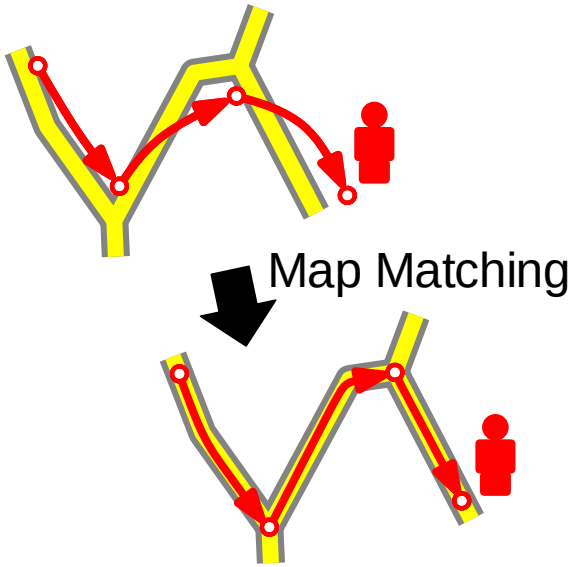


Map Matching

Entscheidungsproblem

Gegeben:
Das Straßennetz als planar eingebetteter Graph $G = (V, E)$
Die GPS-Trajektorie als Folge $P = (p_1, p_2, \dots, p_n)$ von Punkten
Zulässige Distanz ϵ

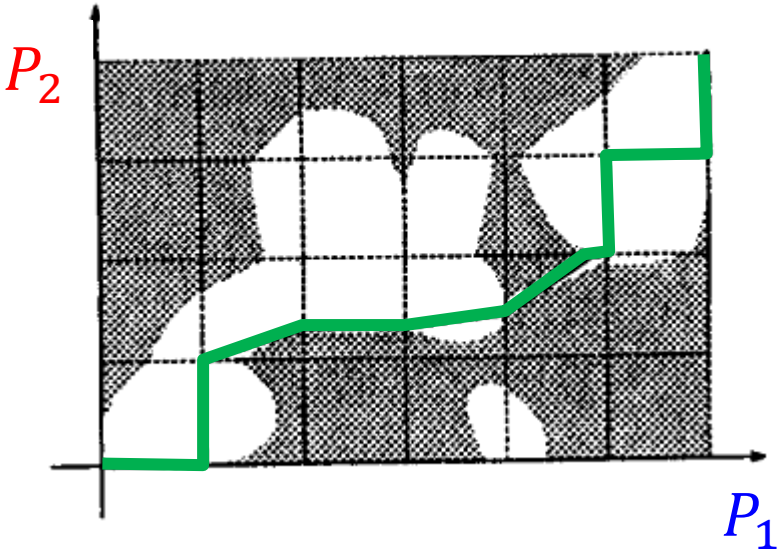
Gibt es einen Weg in G, dessen
Fréchet-Distanz
zu P kleiner ist als ϵ?



Map Matching

Entscheidungsproblem

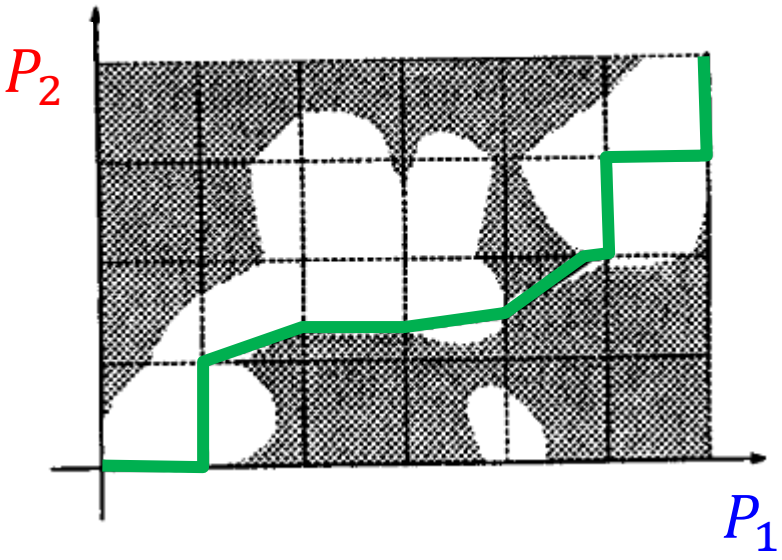
Freiraumdiagramm für zwei Polygonzüge



Map Matching

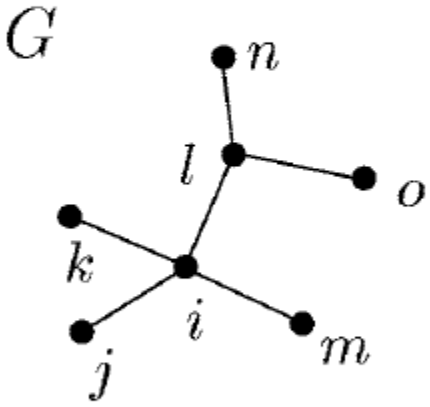
Entscheidungsproblem

Freiraumdiagramm für zwei Polygonzüge



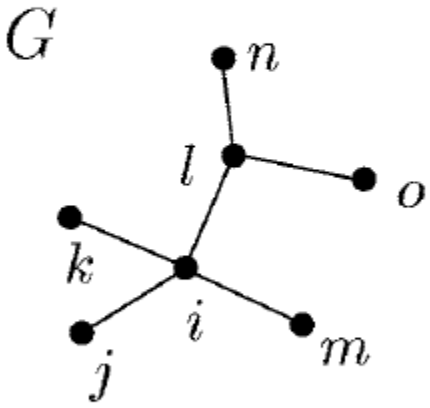
Idee: Freiraumdiagramm für Polygonzug und Graph

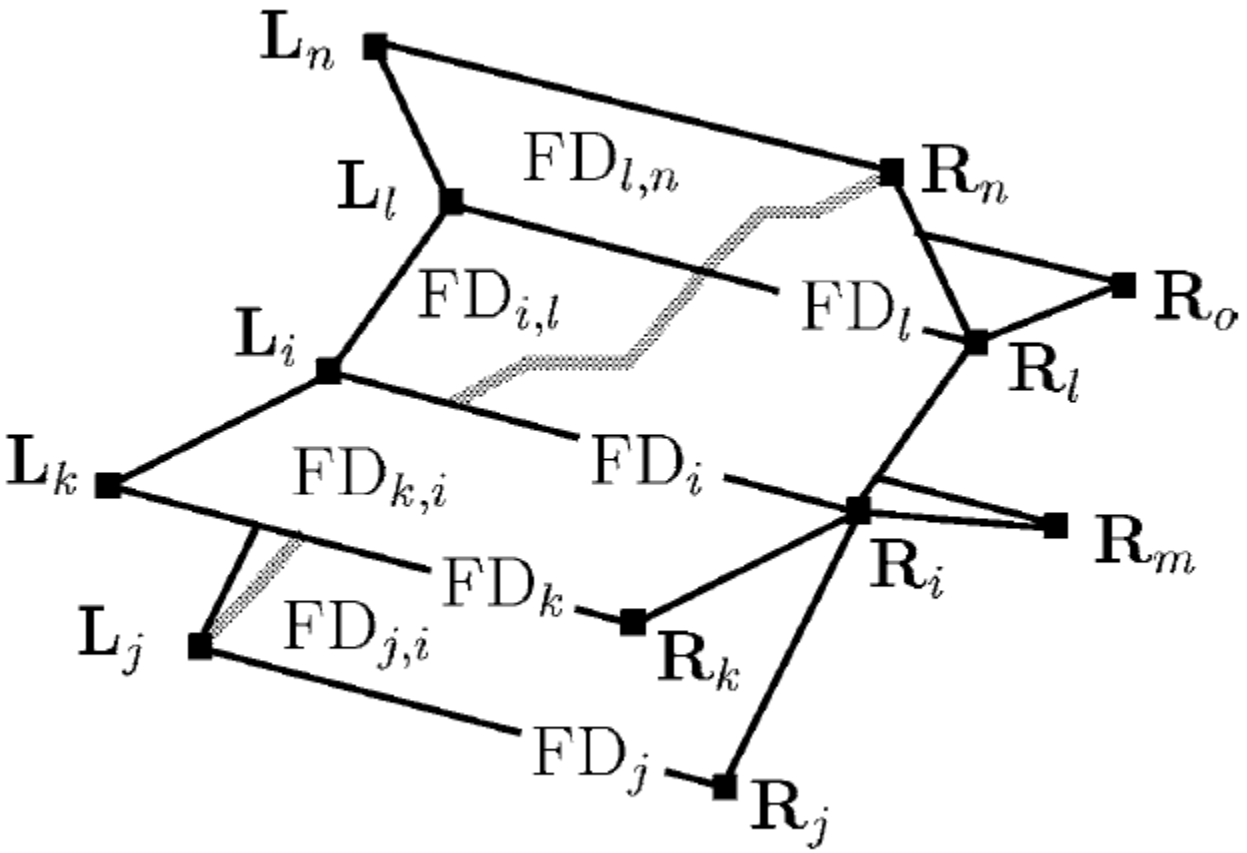
Map Matching



Idee: Freiraumdiagramm für Polygonzug und Graph

Map Matching

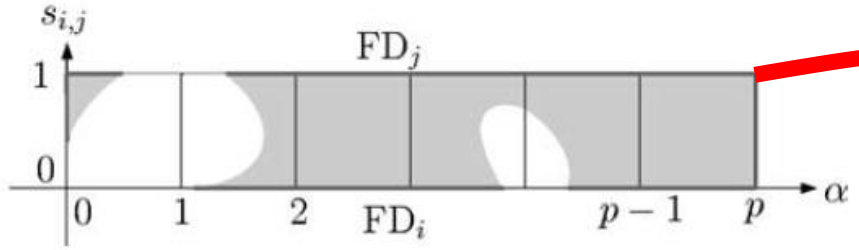


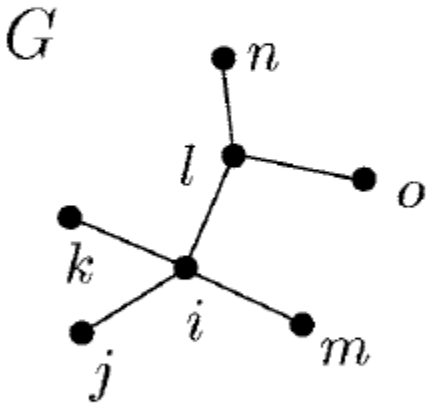


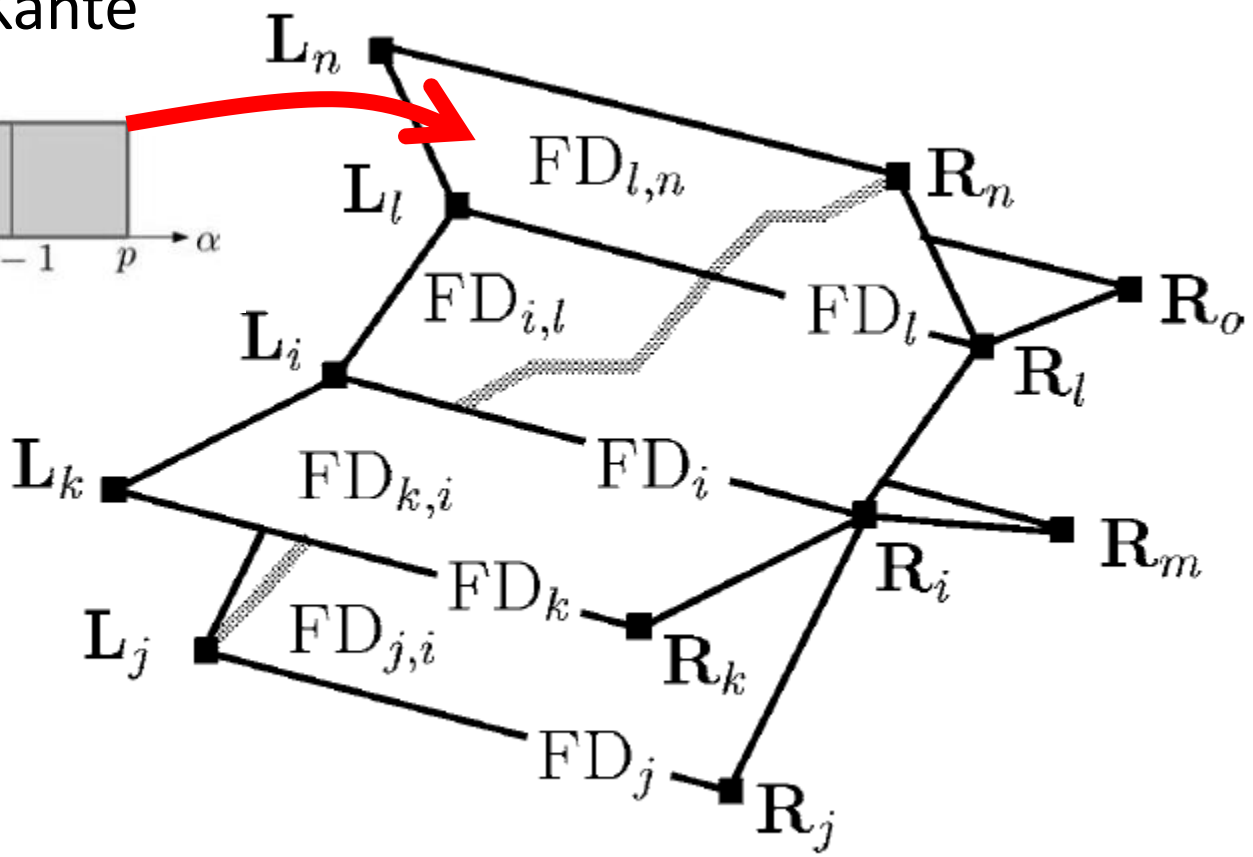
Idee: Freiraumdiagramm für Polygonzug und Graph

Map Matching

Freiraumdiagramm für
Polygonzug und eine Kante



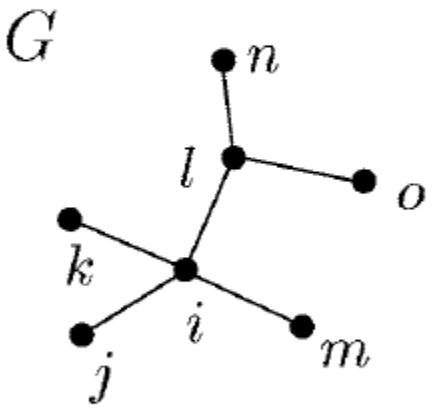


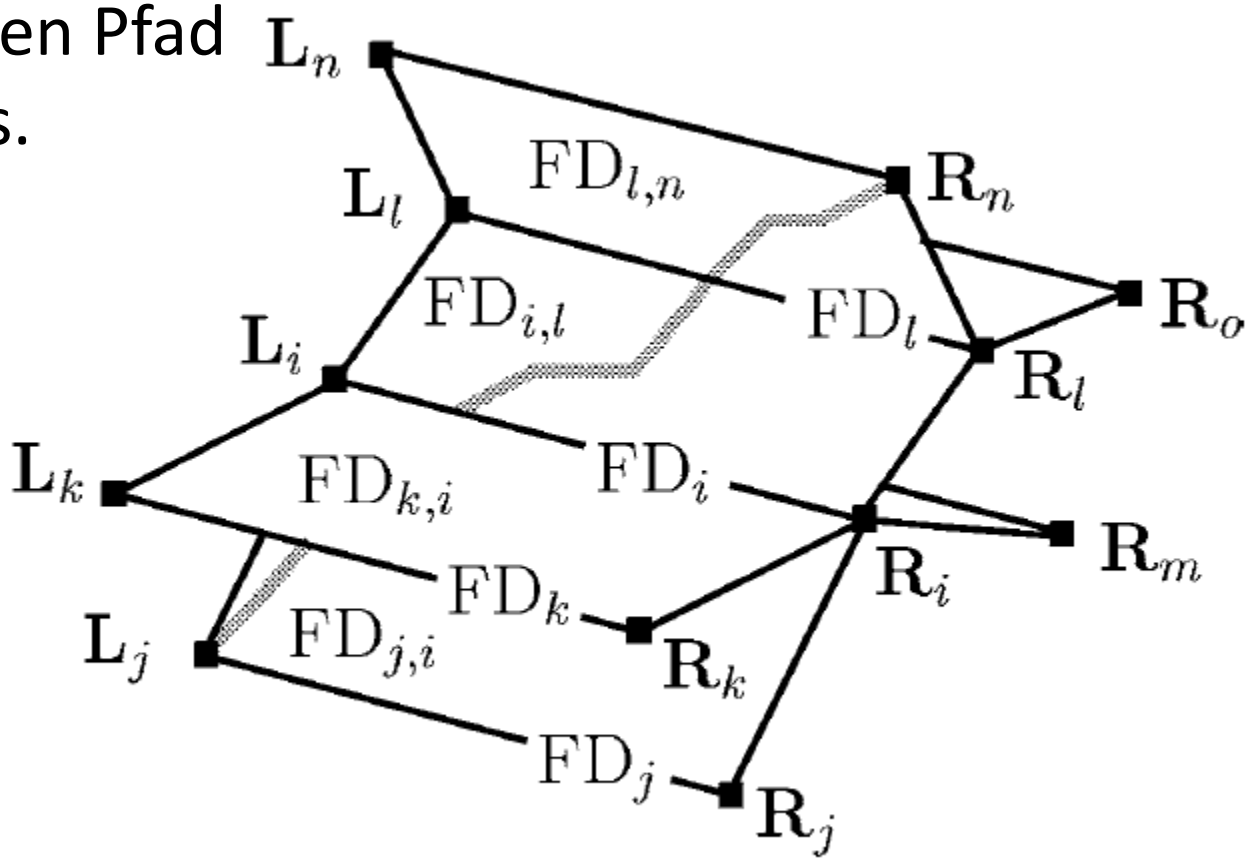


Idee: Freiraumdiagramm für Polygonzug und Graph

Map Matching

Suche einen zulässigen Pfad von links nach rechts.



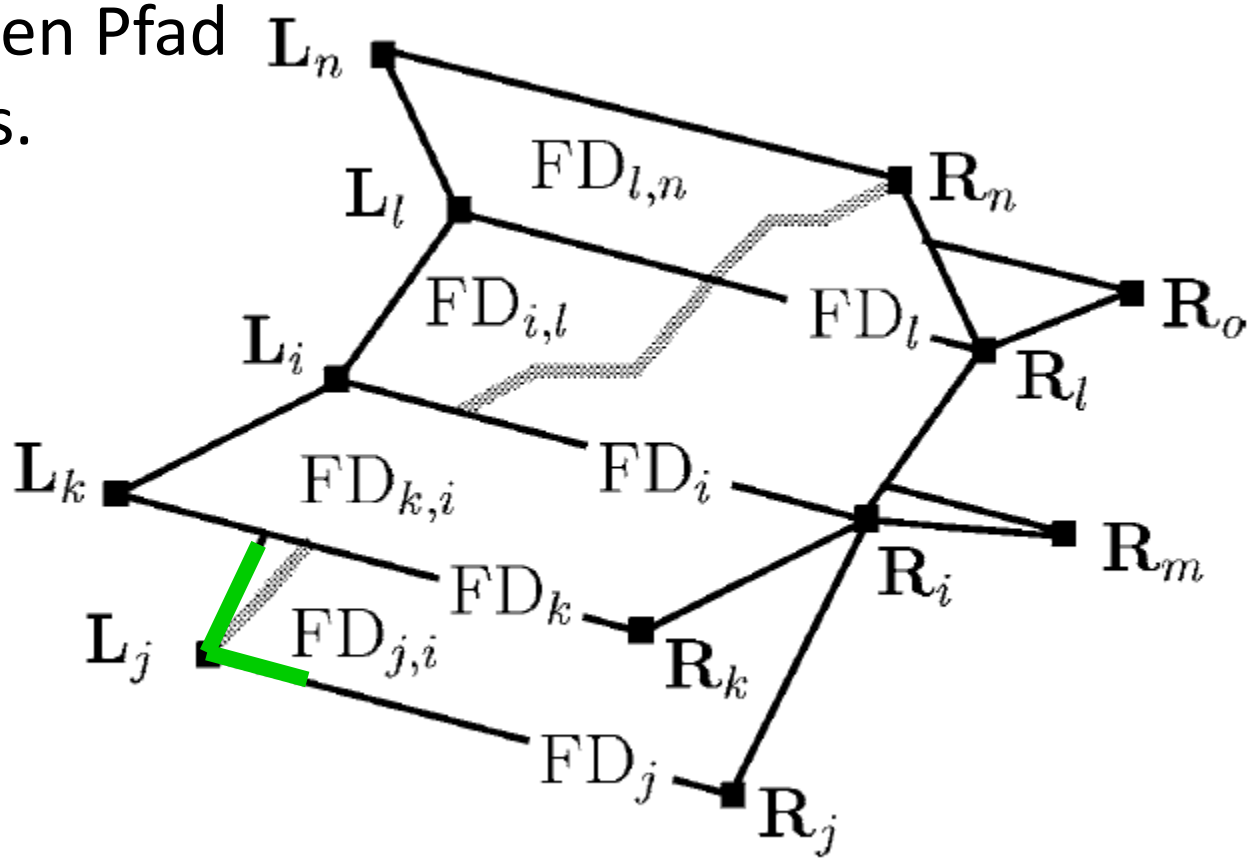


Idee: Freiraumdiagramm für Polygonzug und Graph

Map Matching

Entscheidbar in $O(|E|\,|P|\log|E|)$ Zeit.

Suche einen zulässigen Pfad von links nach rechts.

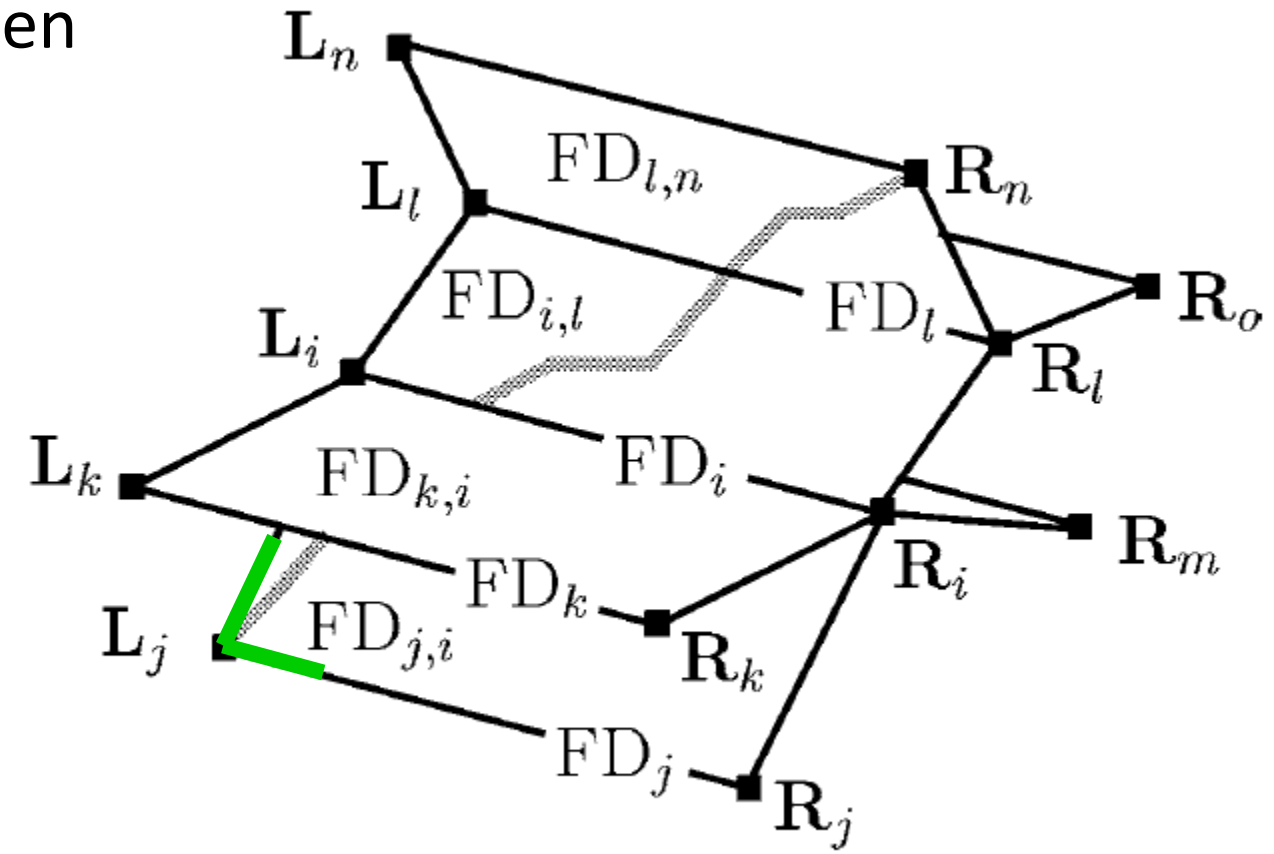


Alt et al. 2003:
Matching Planar Maps
J. Algorithm.

Map Matching

Lösbar in $O(|E|\,|P|\,\log|E|\,\log|E|\,|P|)$ Zeit.

Finde einen zulässigen Pfad mit **minimaler** Fréchet-Distanz



Alt et al. 2003:
Matching Planar Maps
J. Algorithm.