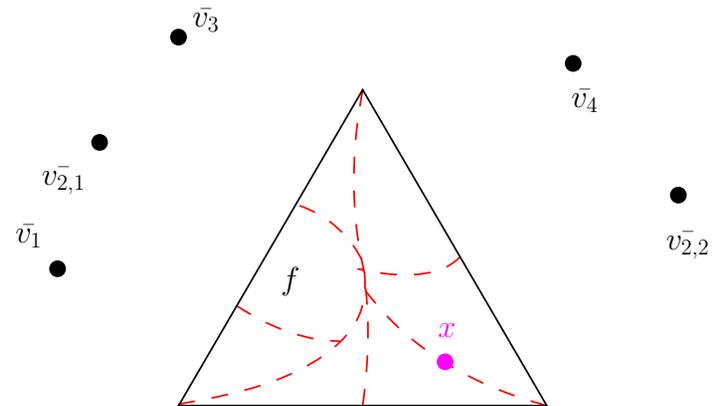


Offline Bewegungsplanung: Kürzeste Wege Polyeder, Objekte bewegen

Elmar Langetepe
University of Bonn

Aufteilen der Dreiecke!

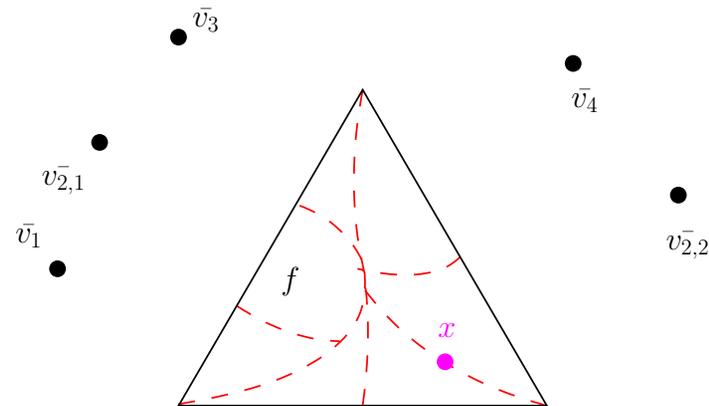
Aufteilen der Dreiecke!



$$|x - v_{2,2}| + d(s, v_{2,2}) = |x - v_5| + d(s, v_5)$$

- Gewichte: $d(s, v_i)$,

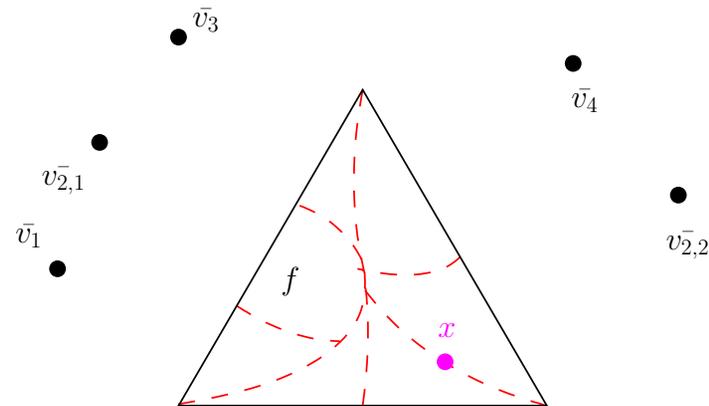
Aufteilen der Dreiecke!



$$|x - v_{2,2}| + d(s, v_{2,2}) = |x - \bar{v}_5| + d(s, v_5)$$

- Gewichte: $d(s, v_i)$, Regionen bezüglich Orte \bar{v}_i

Aufteilen der Dreiecke!

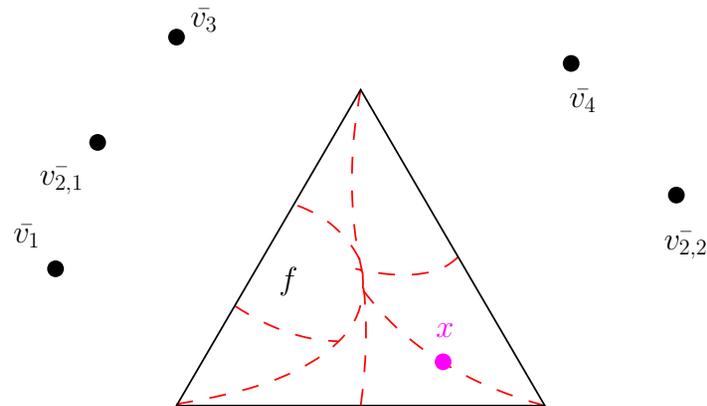


$$|x - v_{2,2}| + d(s, v_{2,2}) = |x - v_5| + d(s, v_5)$$



- Gewichte: $d(s, v_i)$, Regionen bezüglich Orte v_i
- Voronoi Diagramm mit additiven Gewichten $d(s, v_i)$
[voroAdd.html](#)

Aufteilen der Dreiecke!



$$|x - v_{2,2}| + d(s, v_{2,2}) = |x - \bar{v}_5| + d(s, v_5)$$



- Gewichte: $d(s, v_i)$, Regionen bezüglich Orte \bar{v}_i
- Voronoi Diagramm mit additiven Gewichten $d(s, v_i)$
`voroAdd.html`
- Lokalisationsmöglichkeit! (Separators/Seidel)

Alg. 1.13

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$,

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**,

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**,

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$
- Query:

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$
- Query:
 - **Voronoi Region von t in $VD(f)$**

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$
- Query:
 - **Voronoi Region von t in $VD(f)$** $O(\log n)$
 - Kürzesten Weg aus $I(v, E)$: Über E und in v abgesp. Kürz. Weg zu s , k Segmente,

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$
- Query:
 - **Voronoi Region von t in $VD(f)$** $O(\log n)$
 - Kürzesten Weg aus $I(v, E)$: Über E und in v abgesp. Kürz. Weg zu s , k Segmente, $O(k)$

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$
- Query:
 - **Voronoi Region von t in $VD(f)$** $O(\log n)$
 - Kürzesten Weg aus $I(v, E)$: Über E und in v abgesp. Kürz. Weg zu s , k Segmente, $O(k)$
 - Nur Länge,

Alg. 1.13

- Input: Triang. Polyder P , n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
 - Berechne Intervalle $I(v, E)$, $O(n^2 \log n)$
 - Für alle f **add. gew. VD**, $O(n^2 \log n)$
 - **Lokalisationstechnik!(Separators/Seidel)**, $O(n^2)$
- Query:
 - **Voronoi Region von t in $VD(f)$** $O(\log n)$
 - Kürzesten Weg aus $I(v, E)$: Über E und in v abgesp. Kürz. Weg zu s , k Segmente, $O(k)$
 - Nur Länge, $O(1)$

Ergebnis: **Theorem 1.45**

Ergebnis: Theorem 1.45

Sei s auf P fest, gegeben b auf P . Die Entfernung bzw. die Kürzeste von s nach b auf P läßt sich nach Vorbereitungszeit $O(n^2 \log n)$ mit Platz $O(n^2)$ in Zeit $O(\log n)$ bzw. $O(\log n + k)$ berechnen.

Ergebnis: **Theorem 1.45**

Sei s auf P fest, gegeben b auf P . Die Entfernung bzw. die Kürzeste von s nach b auf P läßt sich nach Vorbereitungszeit $O(n^2 \log n)$ mit Platz $O(n^2)$ in Zeit $O(\log n)$ bzw. $O(\log n + k)$ berechnen.

(Mount, Mitchell, Papdimitriou, 1986)

Berechnung aller $I(v, \mathbf{E})$

Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!

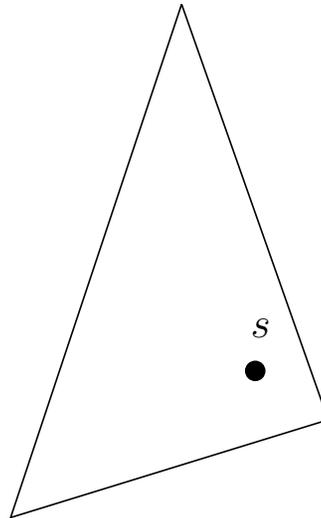
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s .

Berechnung aller $I(v, \mathbf{E})$

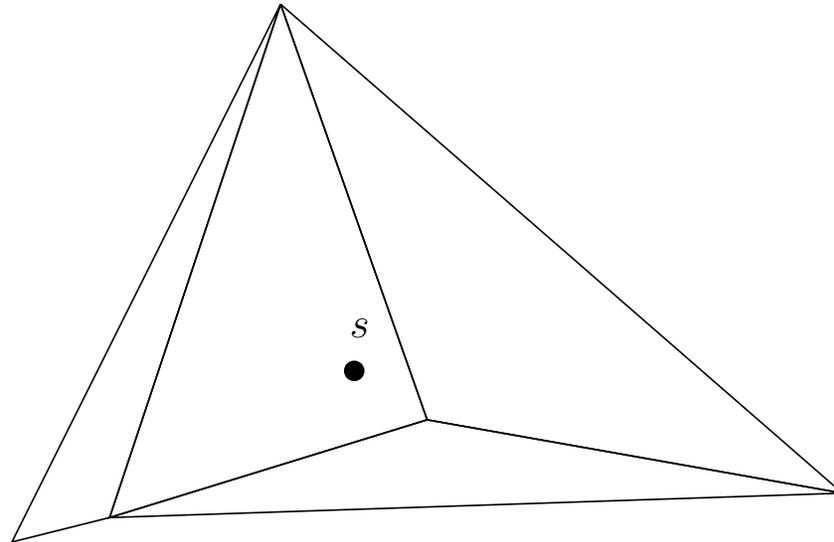
Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!

Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken!



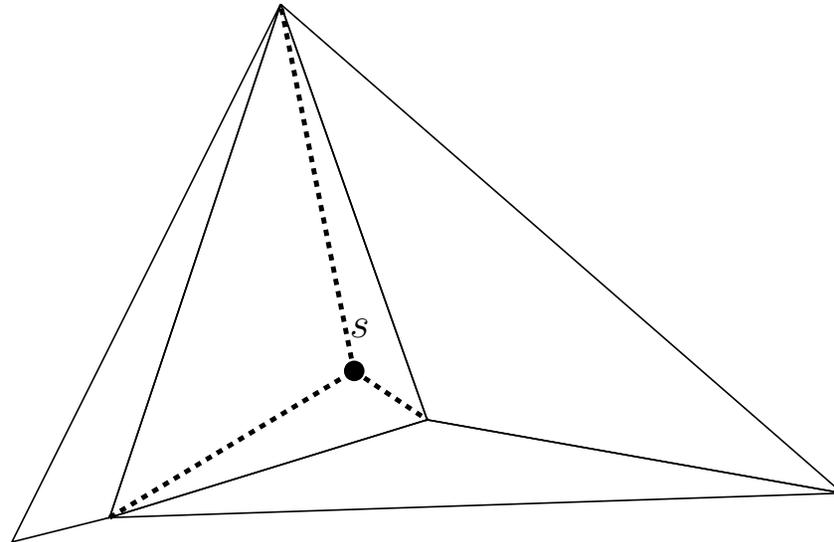
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



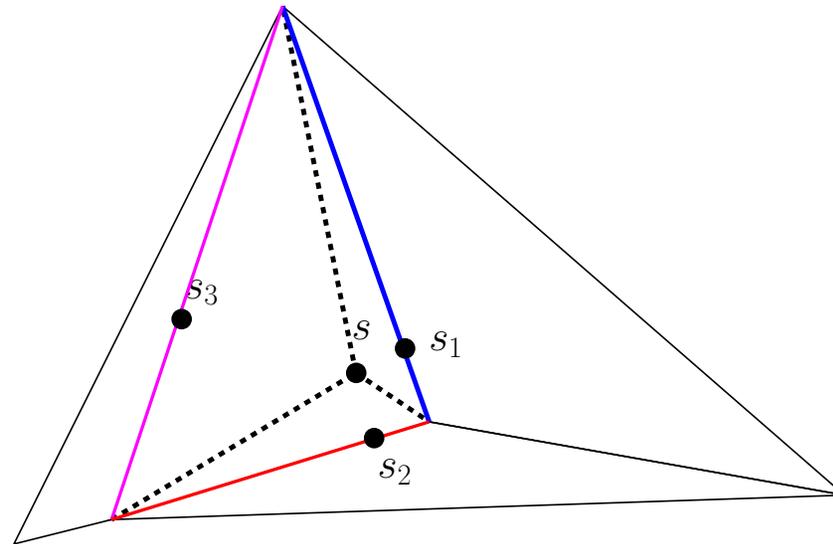
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



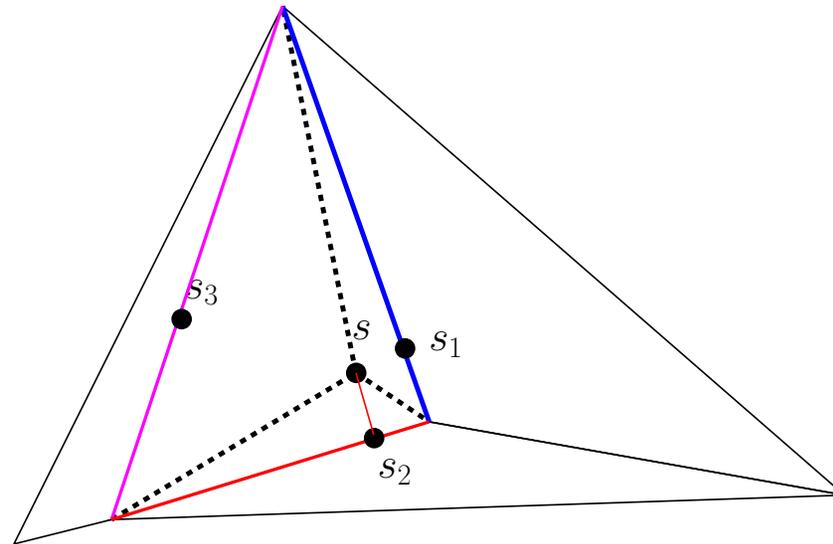
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



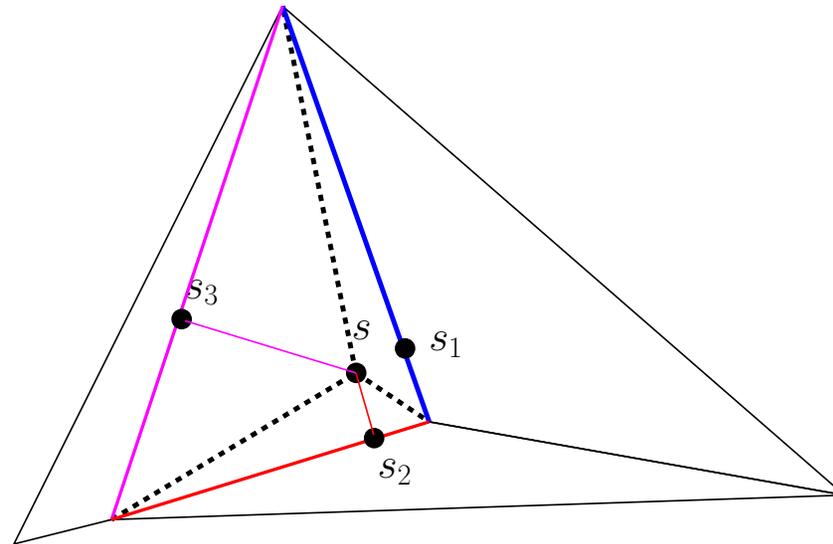
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



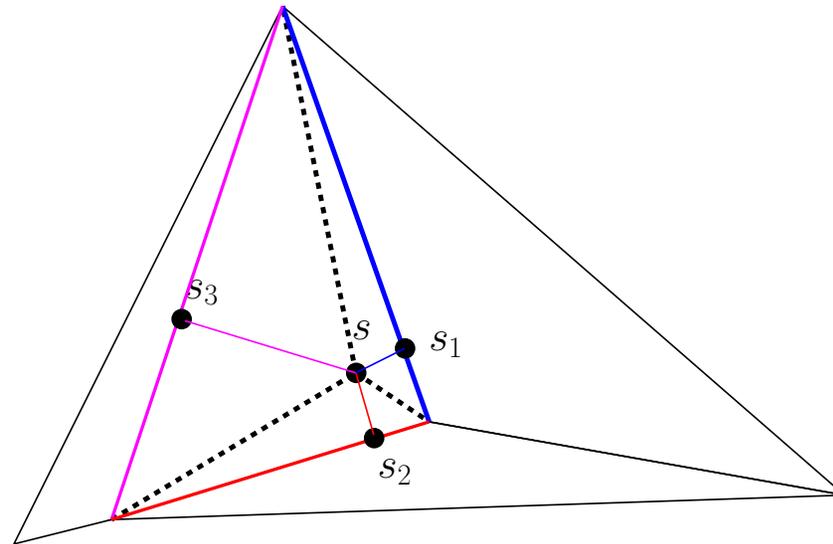
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



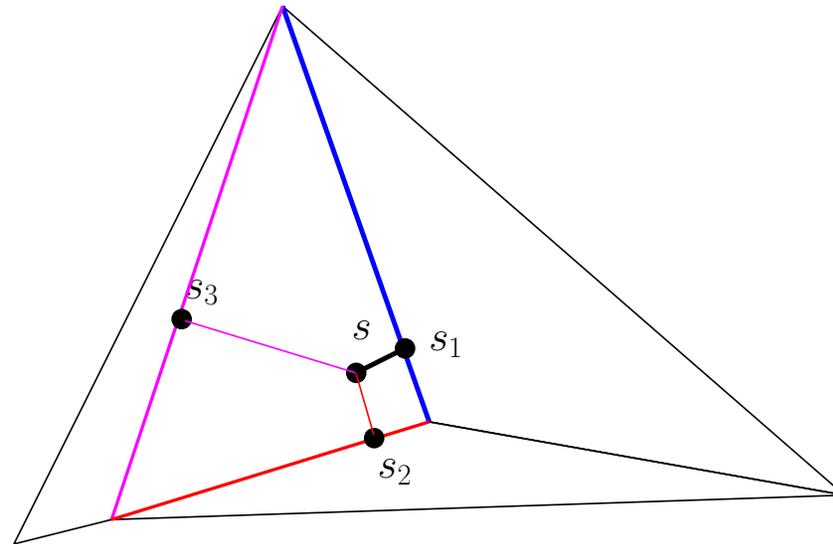
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



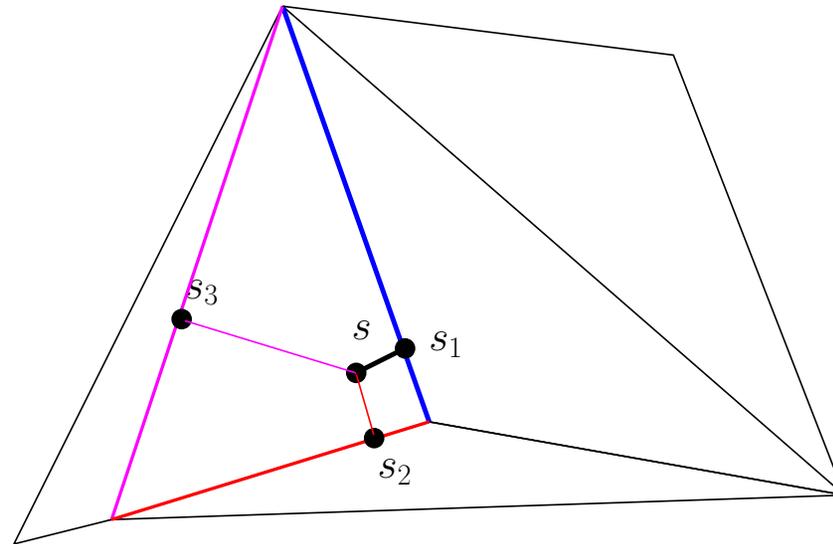
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



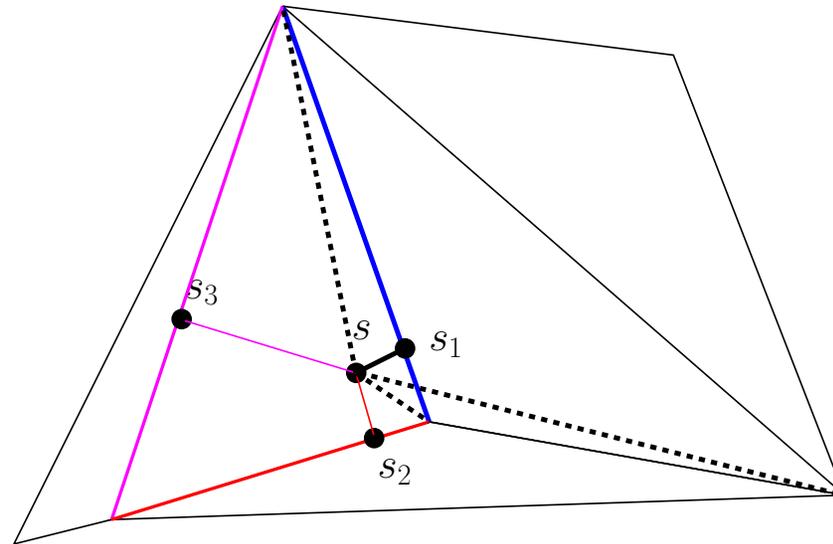
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



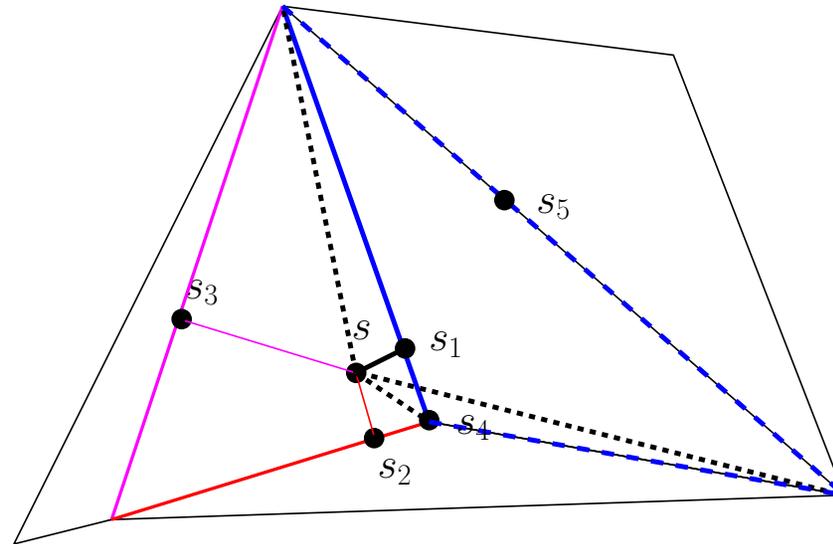
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



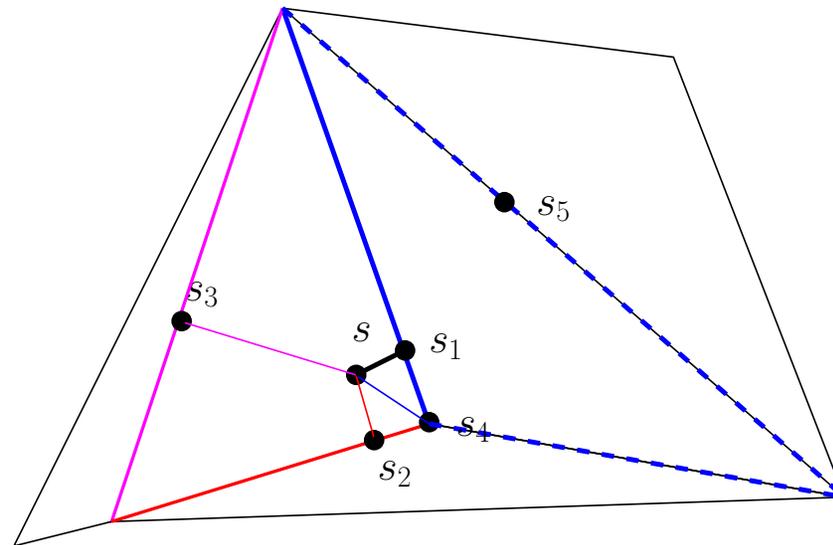
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



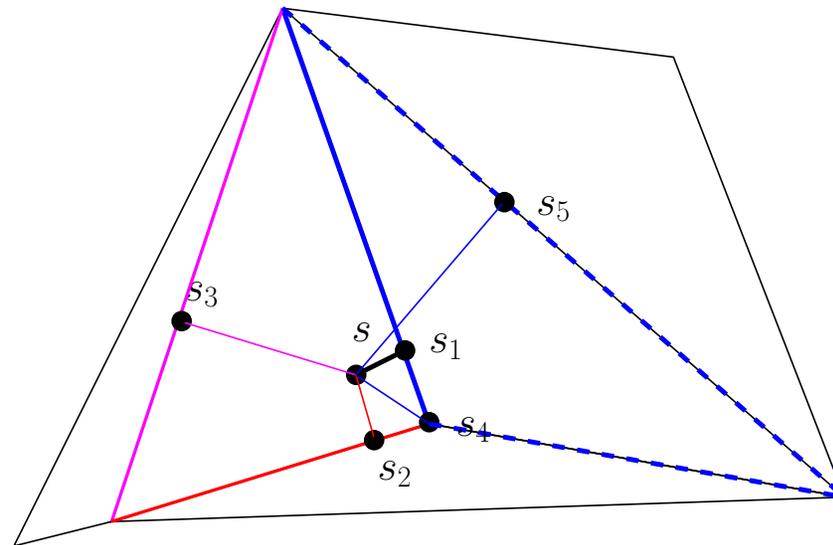
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



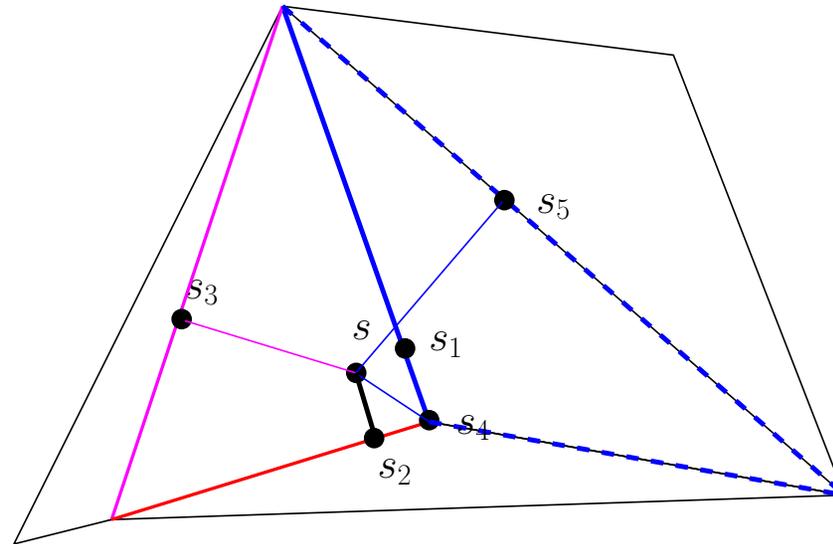
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



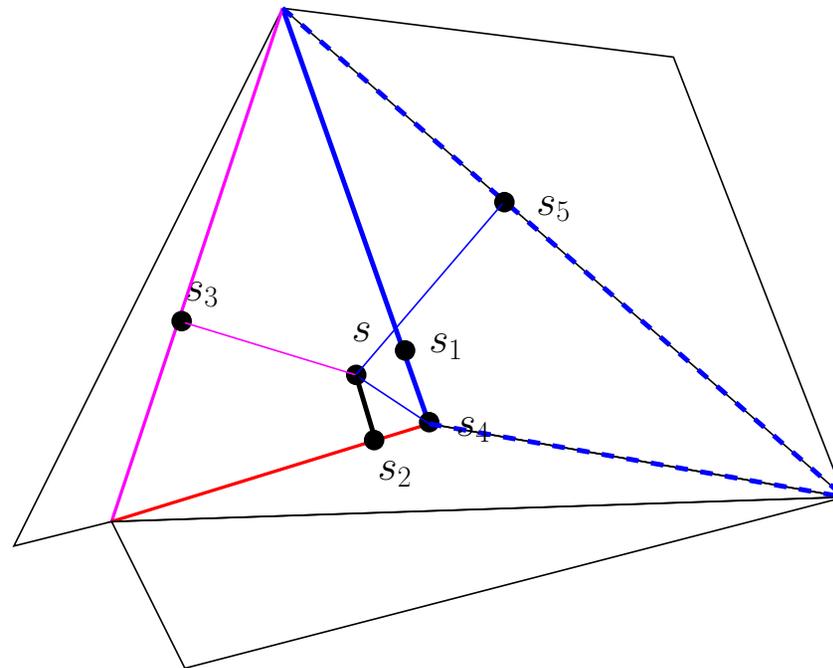
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



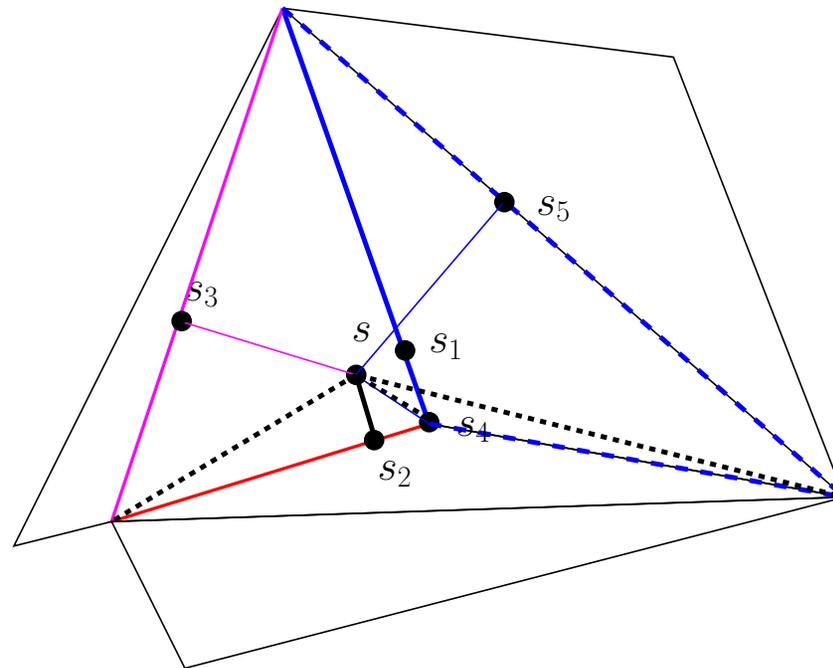
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



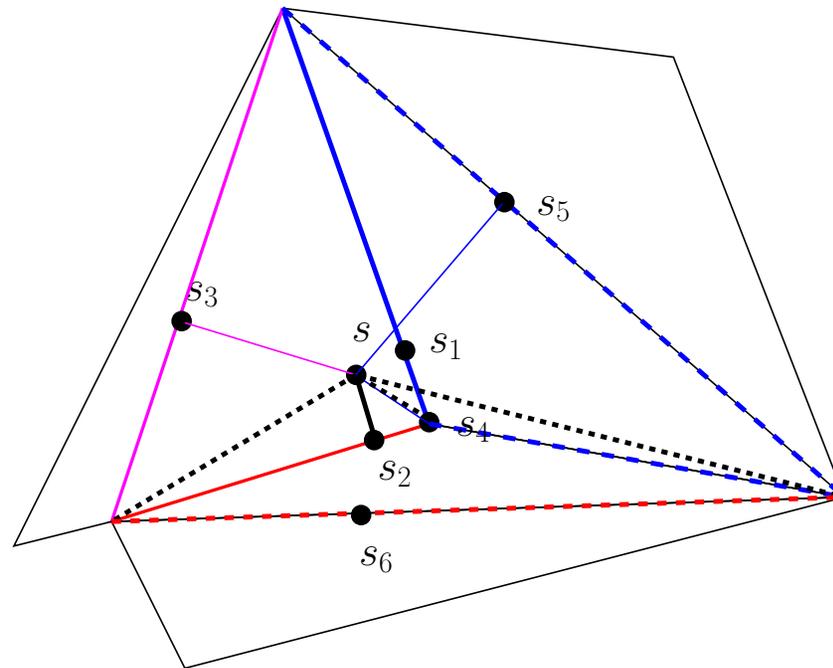
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



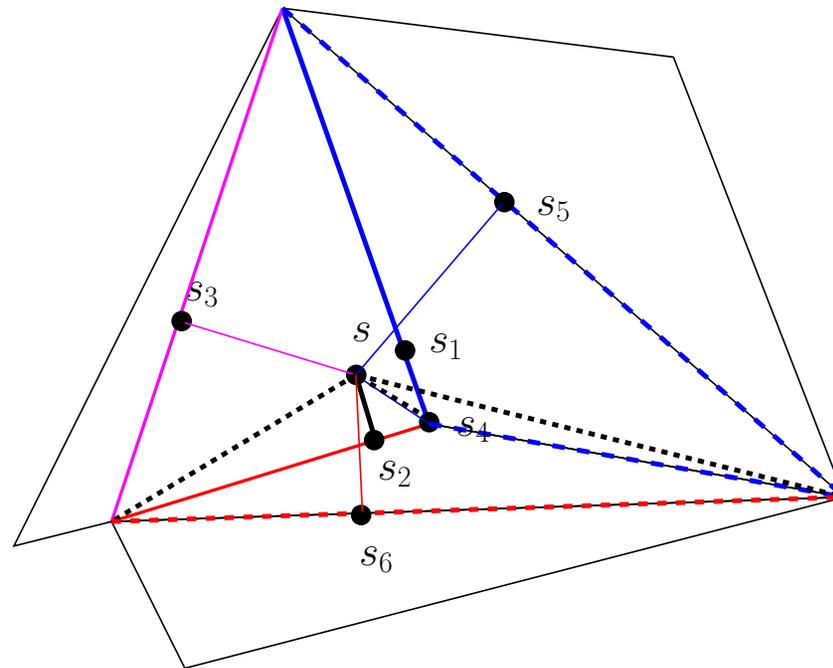
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



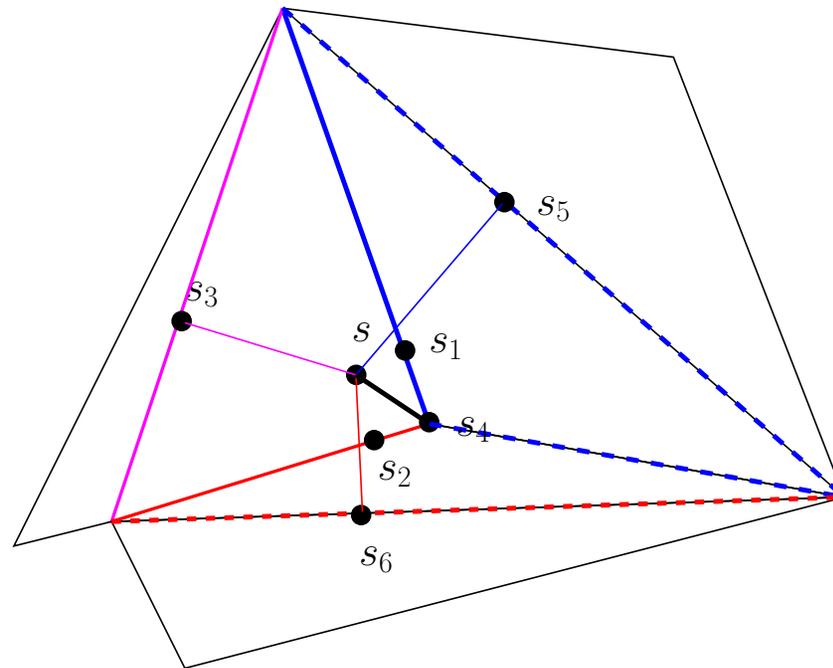
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



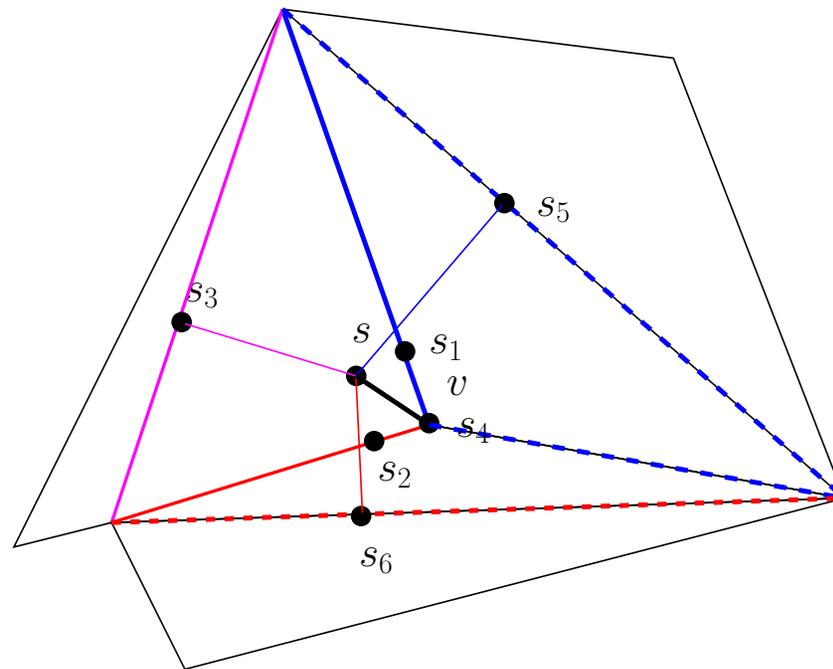
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



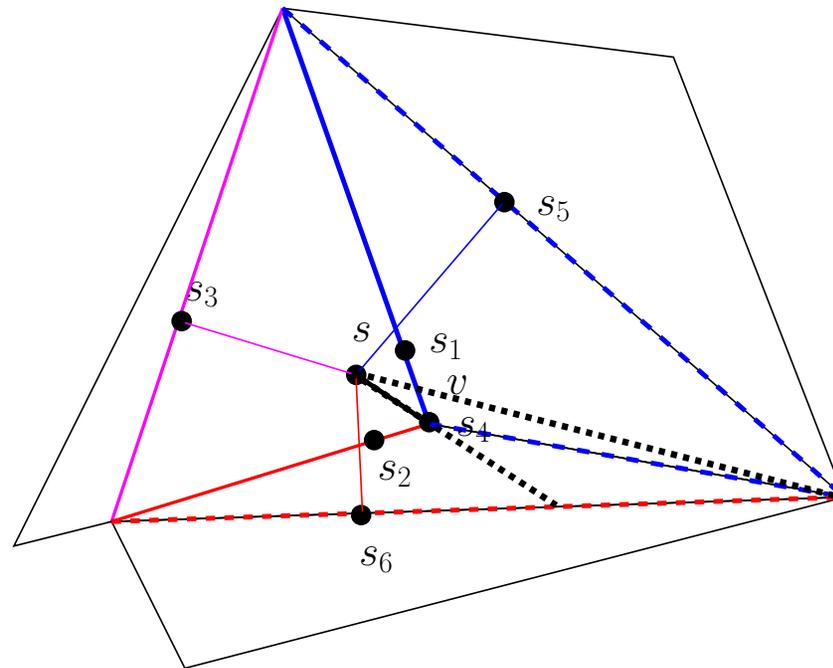
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



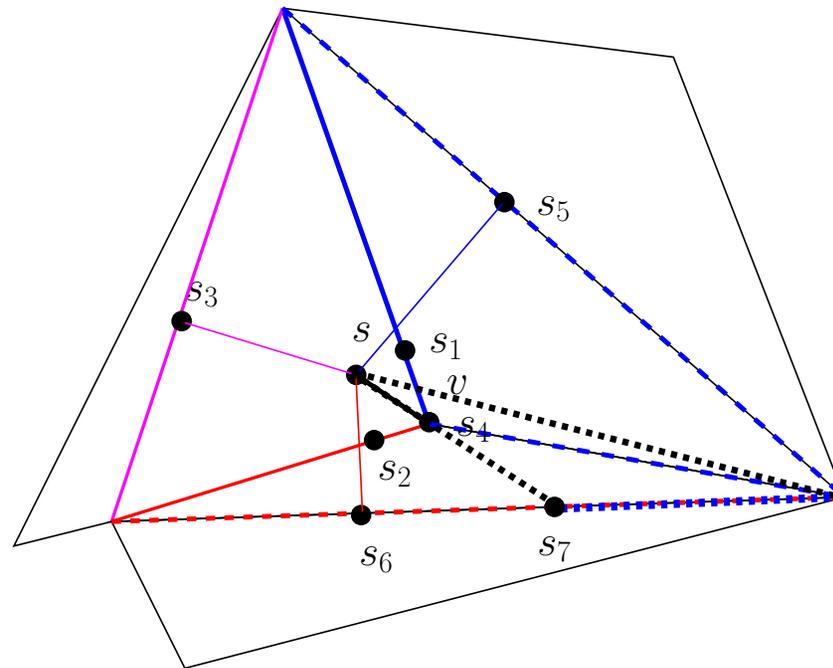
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



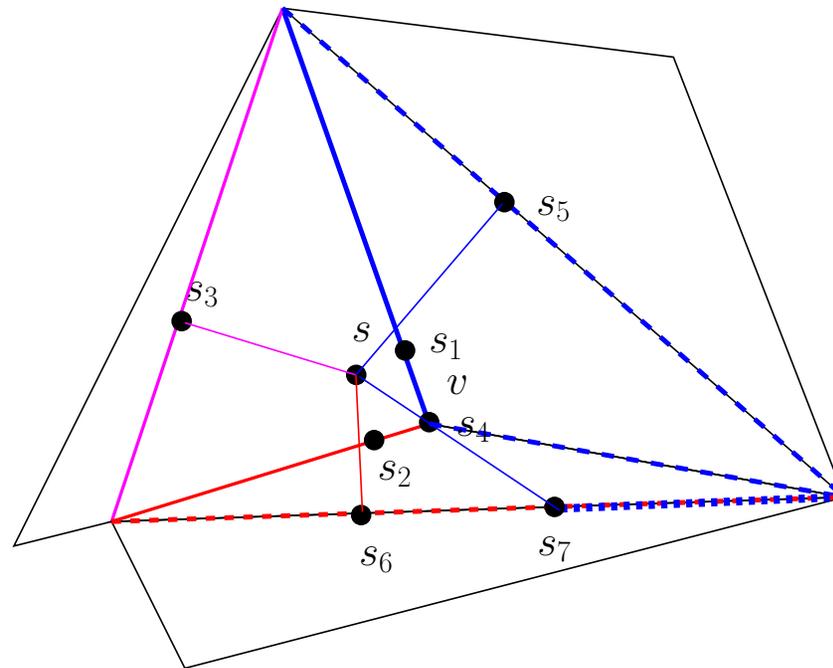
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



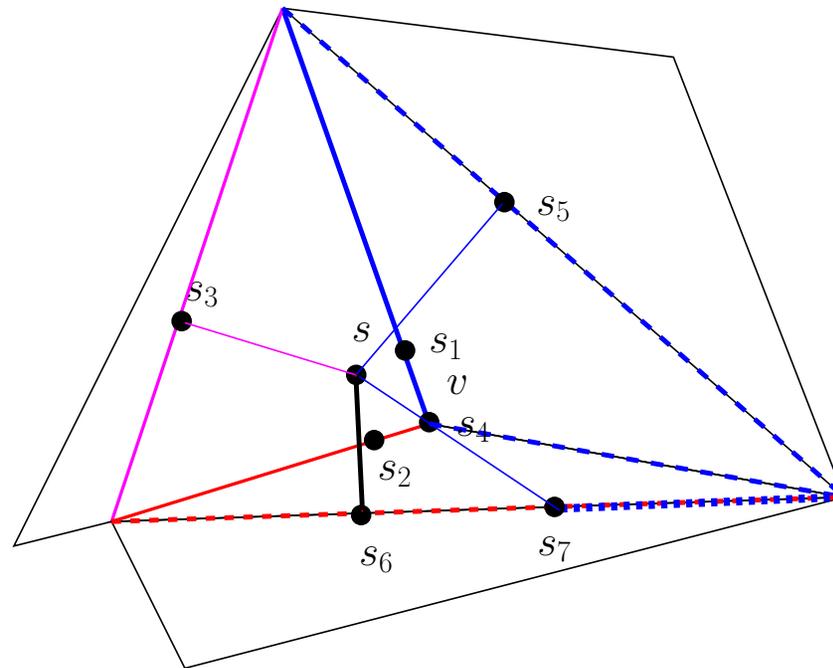
Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



Berechnung aller $I(v, \mathbf{E})$

Continuous Dijkstra: Sukzessive (Teil)Intervalle festlegen!
Garantiepunkt s_i auf dem Intervall, nächster Punkt zu s . Kürzeste Wege zu Knoten merken! Listen von Intervallen aufbauen!



Alg. 1.12 Continuous Dijkstra

- DS: **Alg. 1.12 Continuous Dijkstra**

Alg. 1.12 Continuous Dijkstra

- DS:

- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

Alg. 1.12 Continuous Dijkstra

- DS:

- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

- Iterationsschritt:

Alg. 1.12 Continuous Dijkstra

- DS:

- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

- Iterationsschritt:

- Wähle Intervall $I(v, E)$ mit kürzestem Abstand.

Alg. 1.12 Continuous Dijkstra

- DS:

- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

- Iterationsschritt:

- Wähle Intervall $I(v, E)$ mit kürzestem Abstand.
 $d(s, v) + |\bar{v} - s_i|$ ist am geringsten!

Alg. 1.12 Continuous Dijkstra

- DS:

- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

- Iterationsschritt:

- Wähle Intervall $I(v, E)$ mit kürzestem Abstand.
 $d(s, v) + |\bar{v} - s_i|$ ist am geringsten! Wird stets Intervall bleiben!
(Eventuell später gekürzt).

Alg. 1.12 Continuous Dijkstra

- DS:

- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

- Iterationsschritt:

- Wähle Intervall $I(v, E)$ mit kürzestem Abstand.
 $d(s, v) + |\bar{v} - s_i|$ ist am geringsten! Wird stets Intervall bleiben!
(Eventuell später gekürzt).
- Führe dieses Intervall auf die Kanten des Dreiecks fort, auf dem s_i lag und das jenseits der Folge E liegt.

Alg. 1.12 Continuous Dijkstra

- DS:

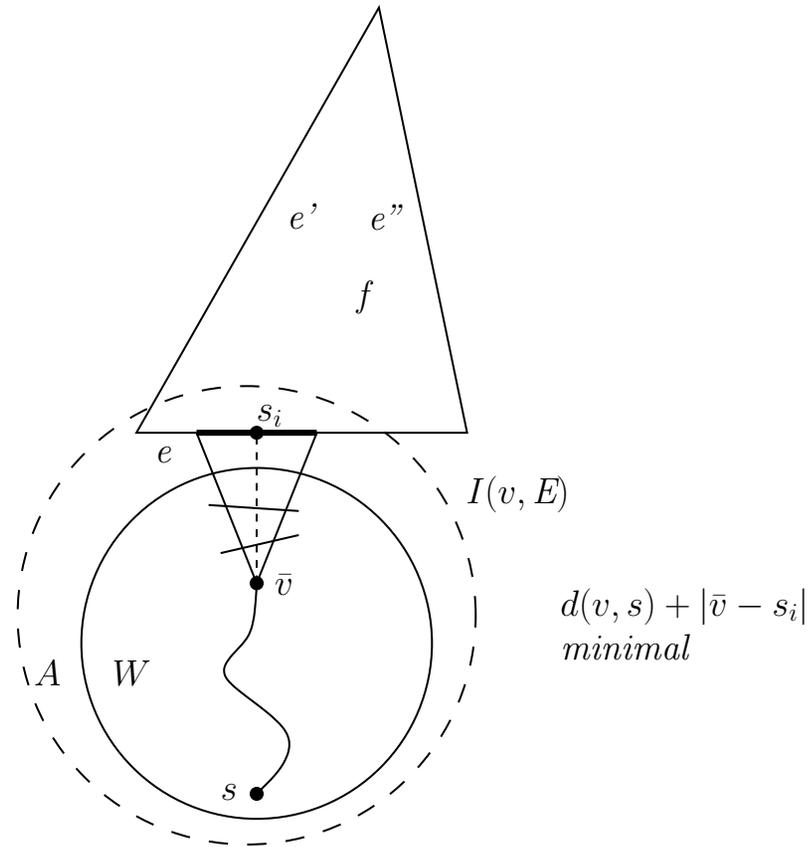
- Priority Queue W : Intervalle nach s_i -Abstände.
- Intervalllisten auf den Kanten in balancierten Baum vorhalten.

- Iterationsschritt:

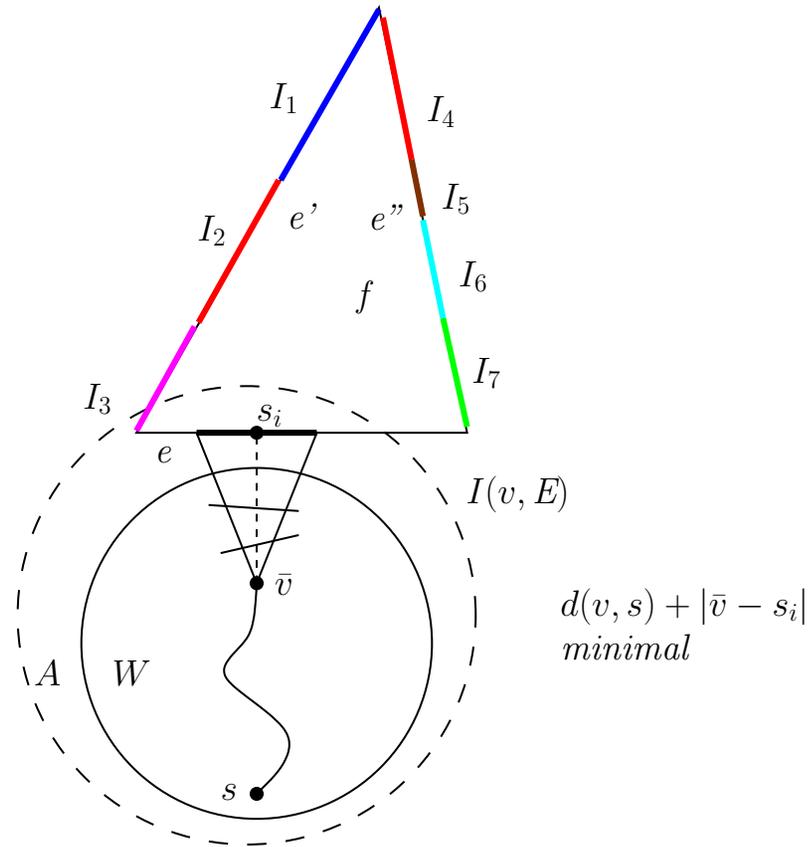
- Wähle Intervall $I(v, E)$ mit kürzestem Abstand.
 $d(s, v) + |\bar{v} - s_i|$ ist am geringsten! Wird stets Intervall bleiben!
(Eventuell später gekürzt).
- Führe dieses Intervall auf die Kanten des Dreiecks fort, auf dem s_i lag und das jenseits der Folge E liegt.
- Sortiere zwei neue Intervalle in die Liste der Intervalle der beiden Kanten ein. Bestimme jeweilige s_j und füge in W ein.

Schritt des Einsortierens!

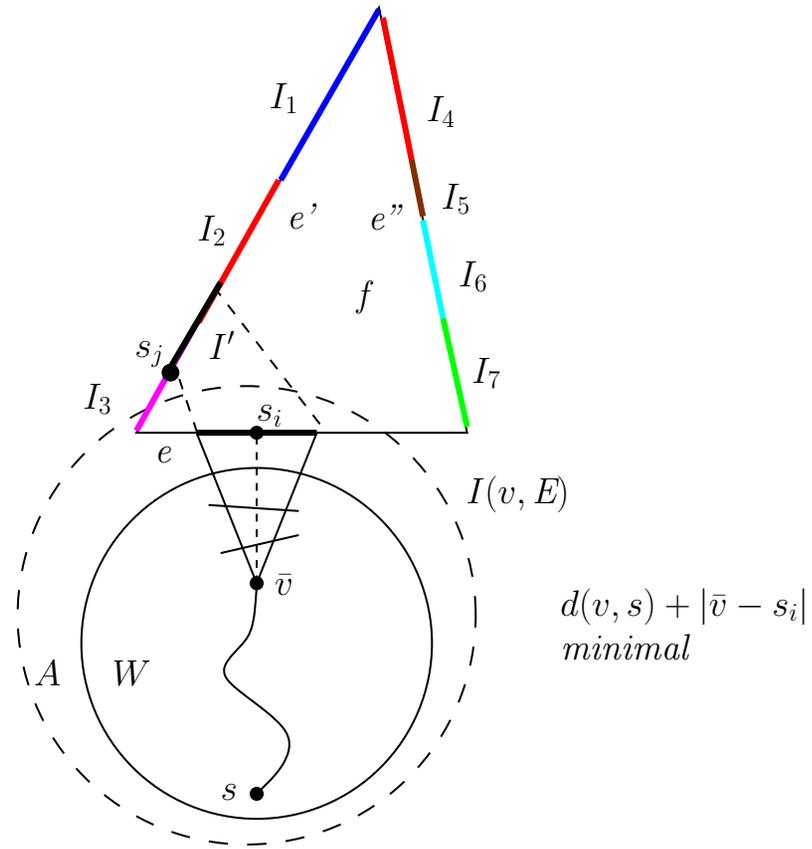
Schritt des Einsortierens!



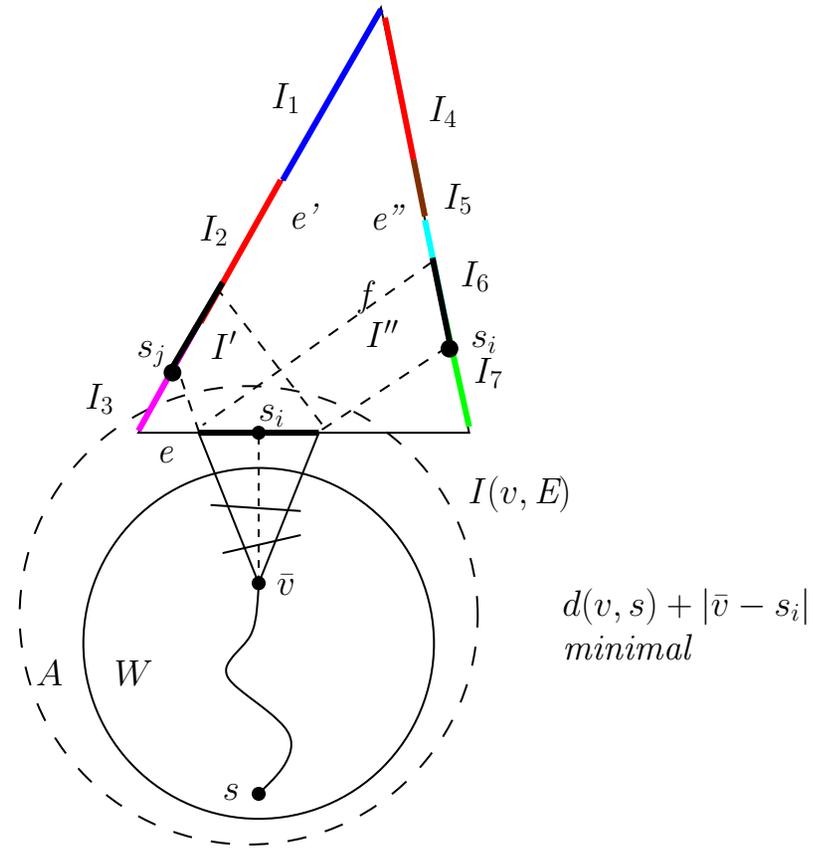
Schritt des Einsortierens!



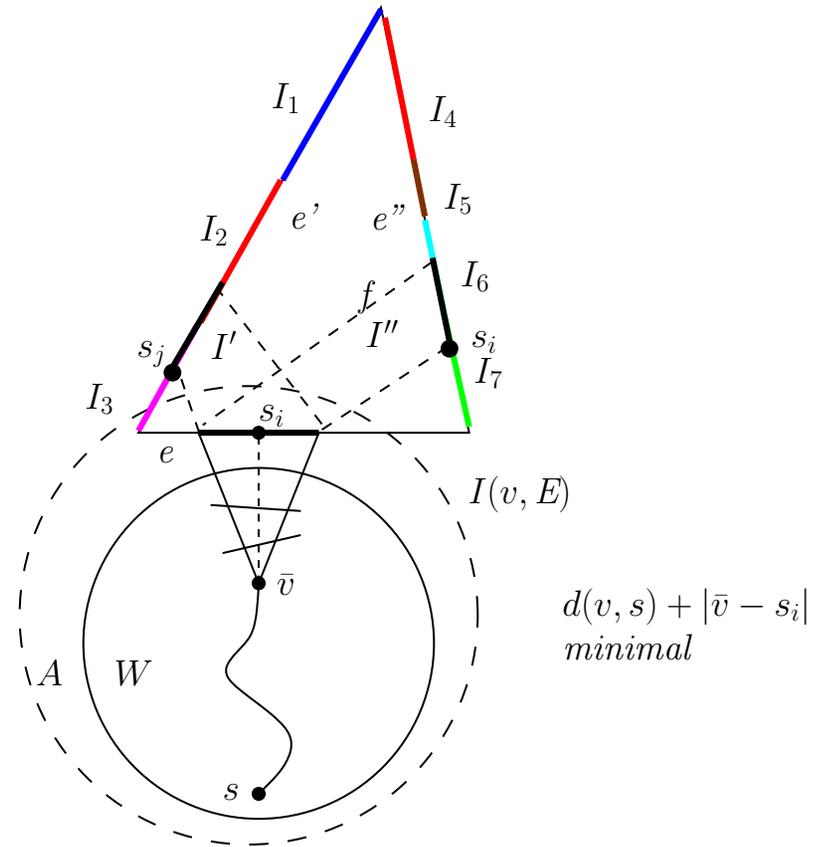
Schritt des Einsortierens!



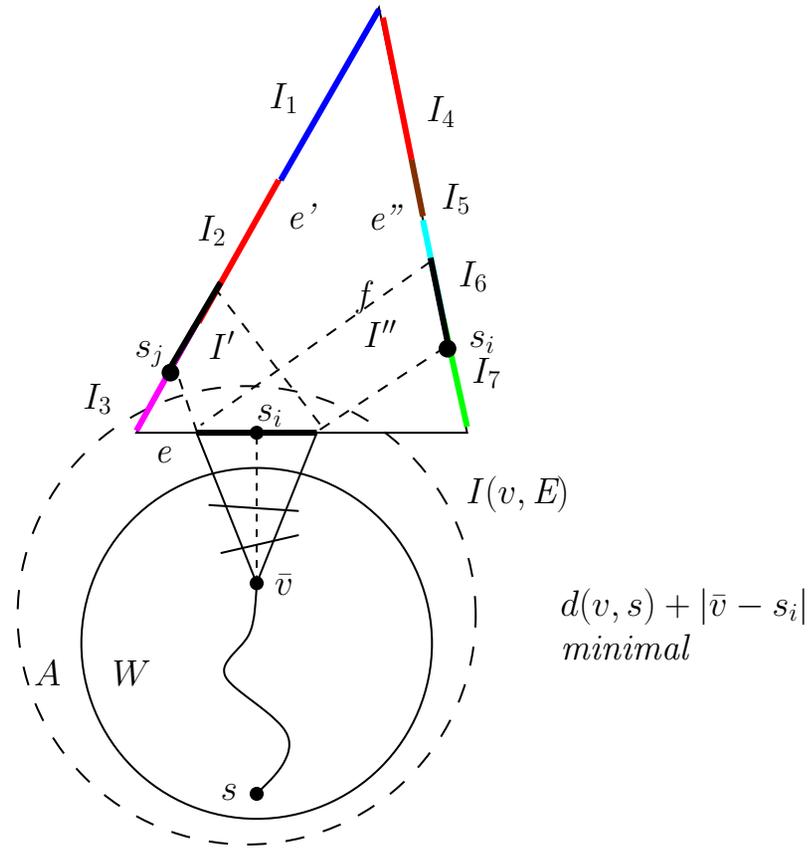
Schritt des Einsortierens!



Schritt des Einsortierens!

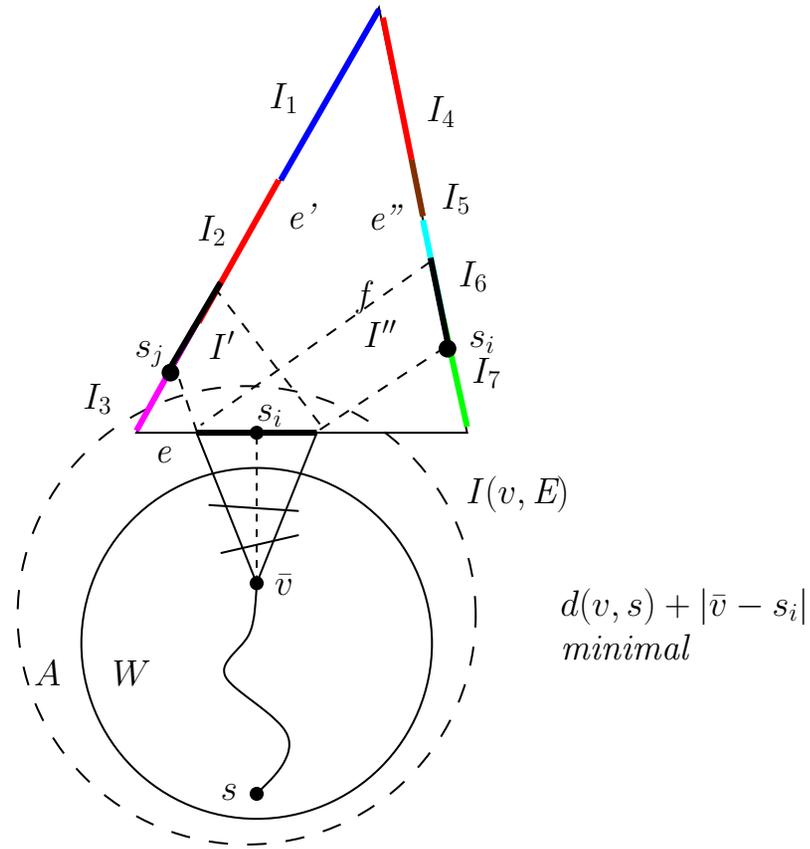


Schritt des Einsortierens!



Einsortieren in Baum der Intervalle!

Schritt des Einsortierens!



Einsortieren in Baum der Intervalle! $O(\log n)$!

Liniensegmente statt Punkte

Liniensegmente statt Punkte

- Liniensegment in der Ebene

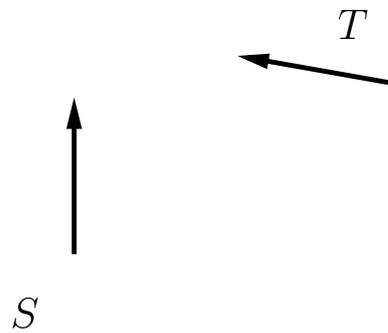
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S



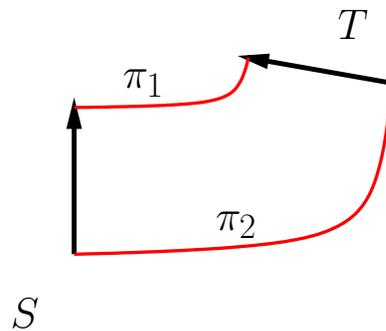
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T



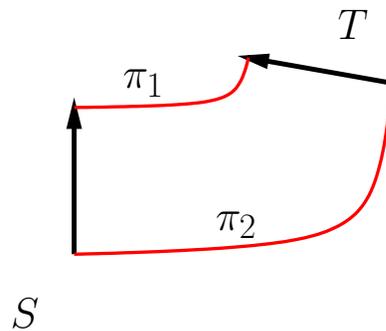
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T
- Stetige Bewegung von S nach T



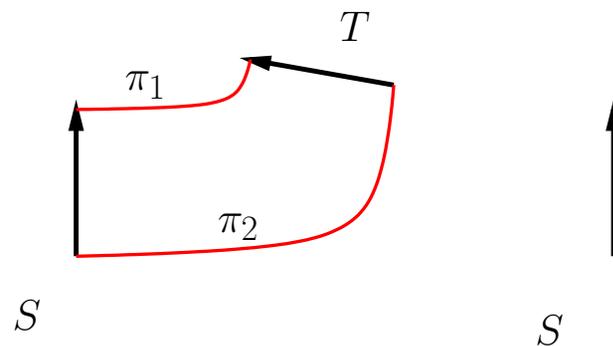
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T
- Stetige Bewegung von S nach T
- Möglichst kurz!



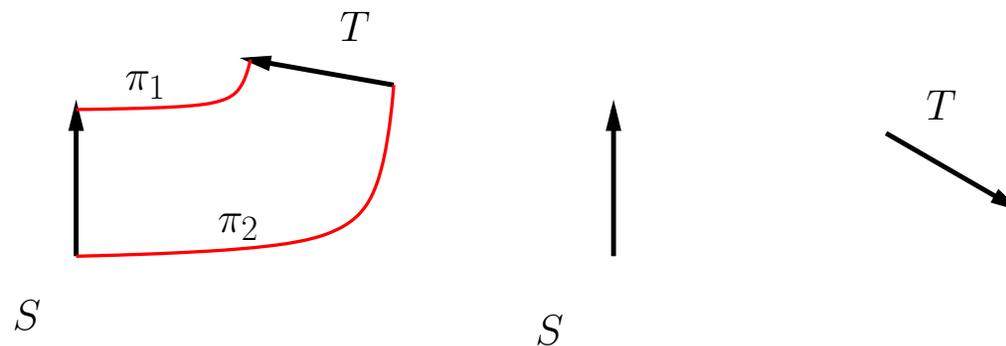
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T
- Stetige Bewegung von S nach T
- Möglichst kurz!



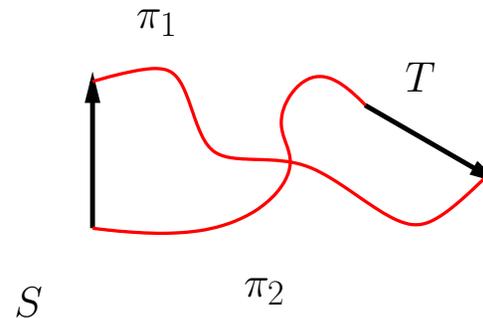
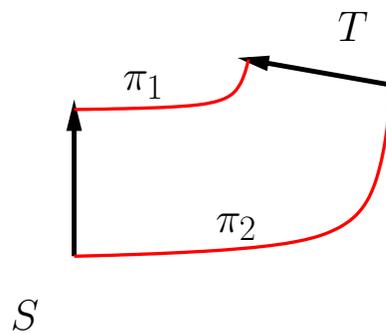
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T
- Stetige Bewegung von S nach T
- Möglichst kurz!



Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T
- Stetige Bewegung von S nach T
- Möglichst kurz!



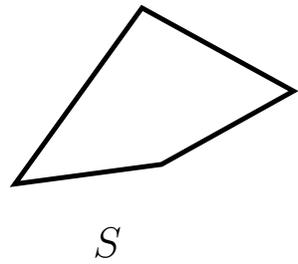
Was ist eine kurze Bewegung?

Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig

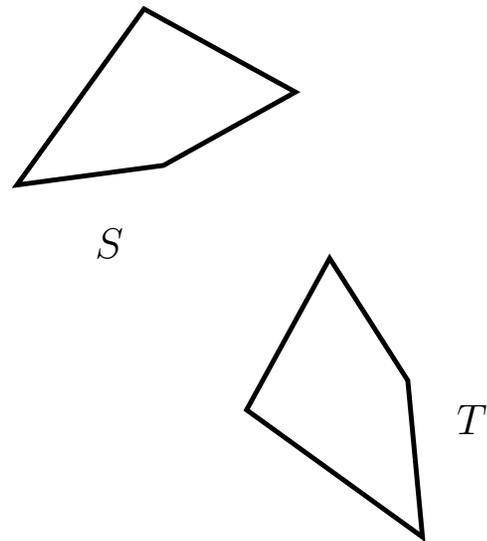
Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig



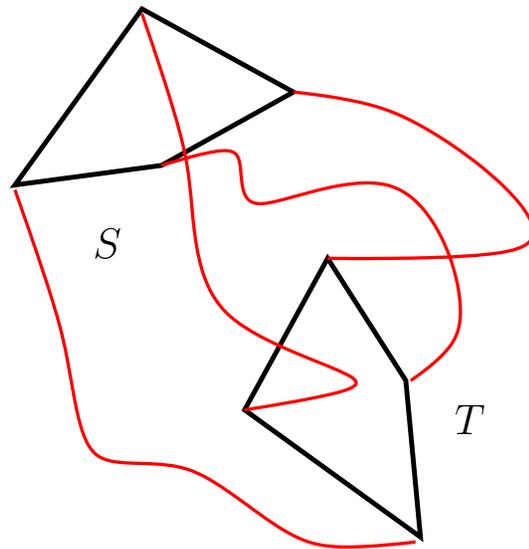
Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig



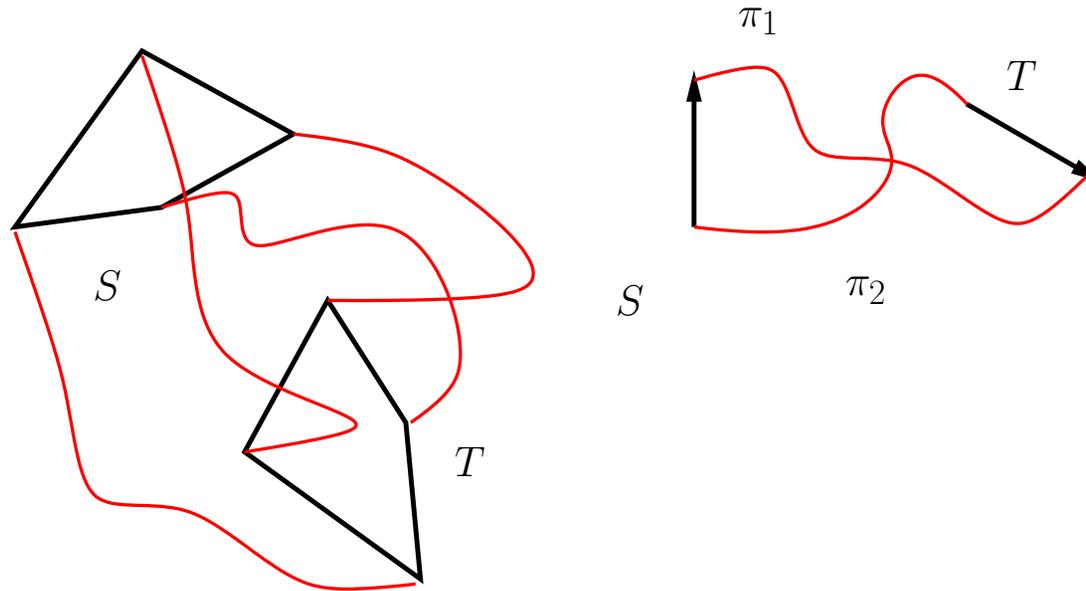
Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig



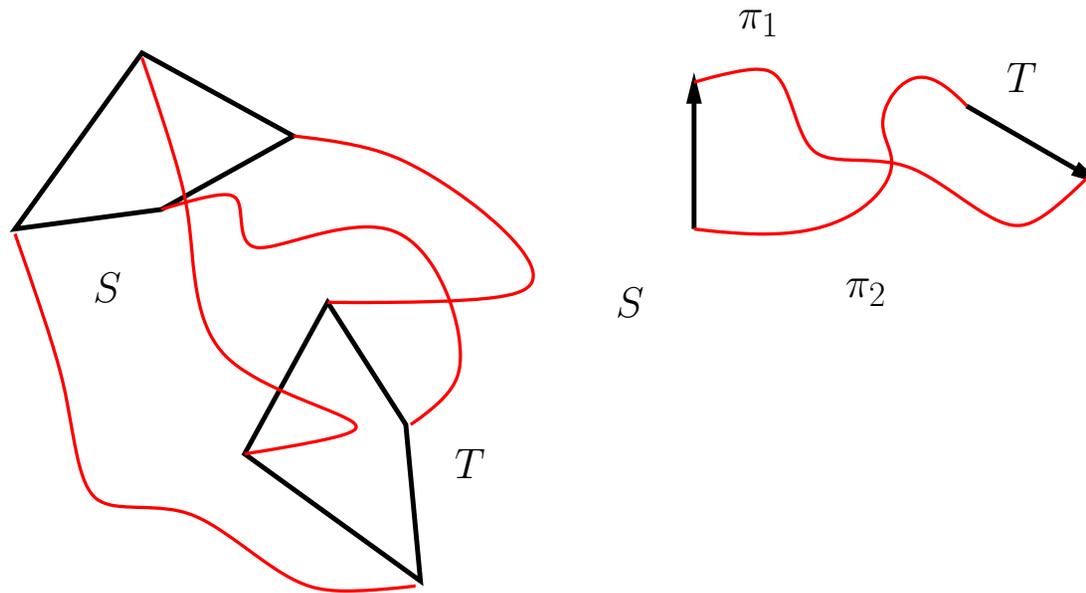
Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig
- Hier: Weglänge zweier(!) Träger: Summe



Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig
- Hier: Weglänge zweier(!) Träger: Summe
- Andere Maße: Mittlerer Weg, Min/Max eines Trägers



Formale Beschreibung!

Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)

Formale Beschreibung!

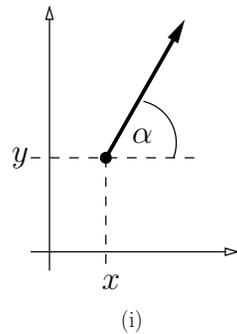
- Lage des Segmentes durch Tripel (x, y, α)



(i)

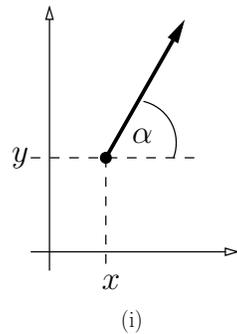
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!



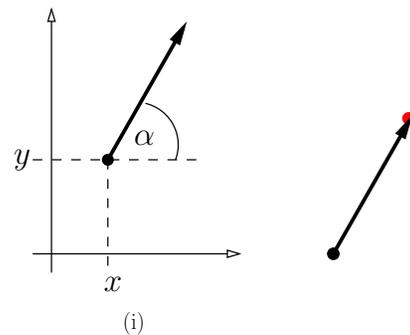
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



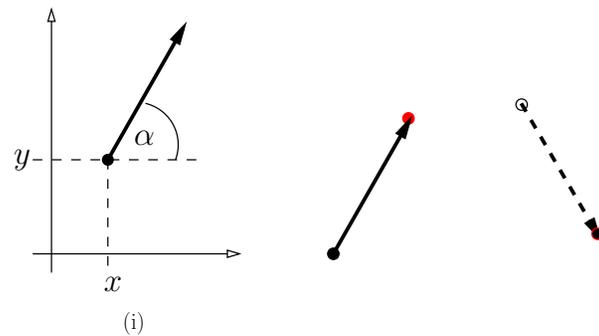
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



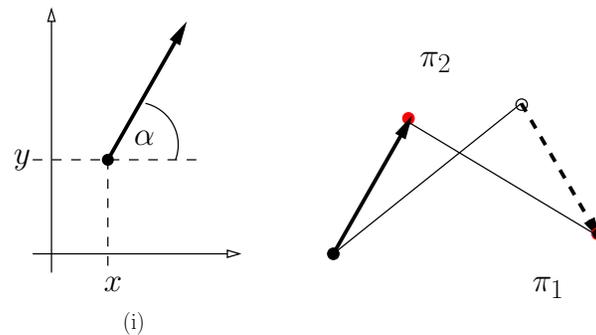
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



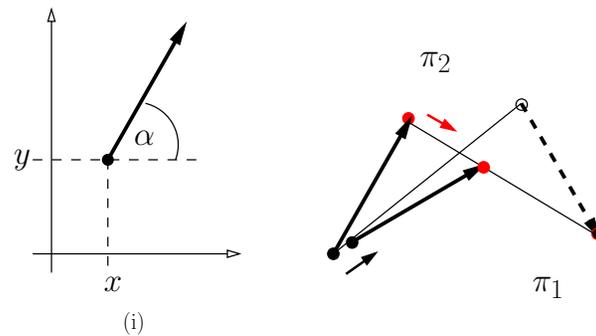
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



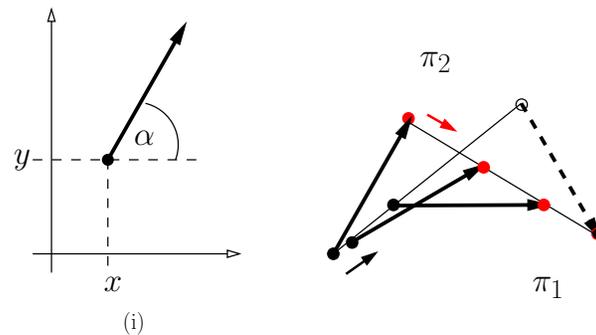
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



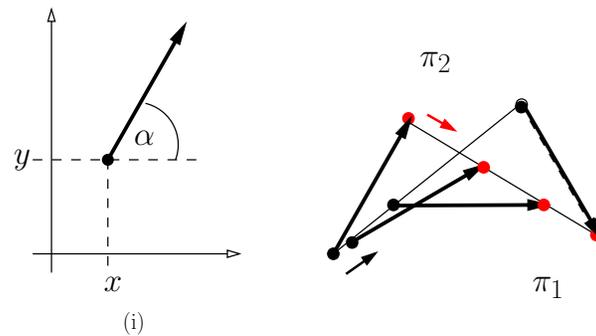
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



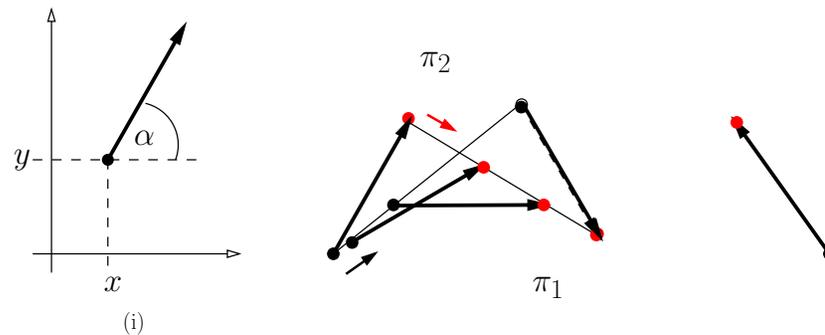
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



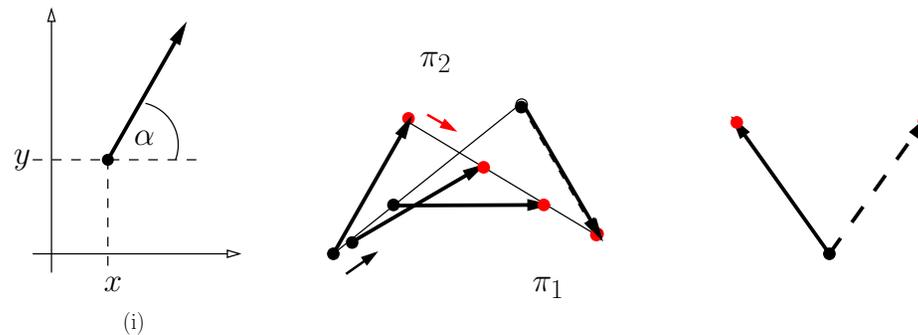
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



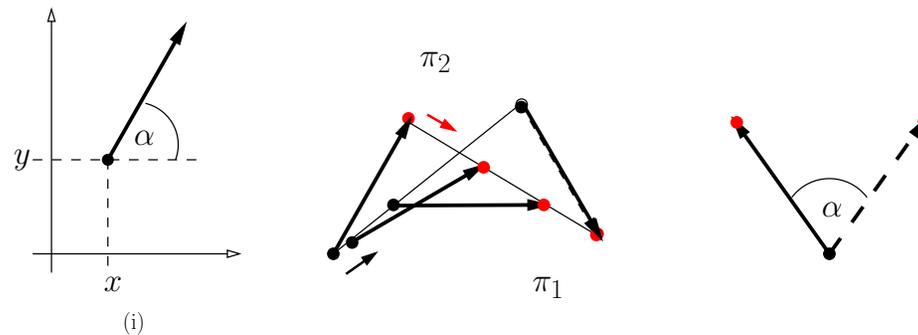
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



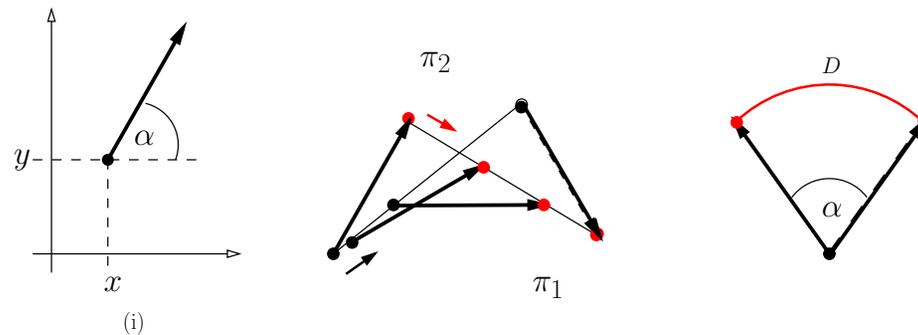
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



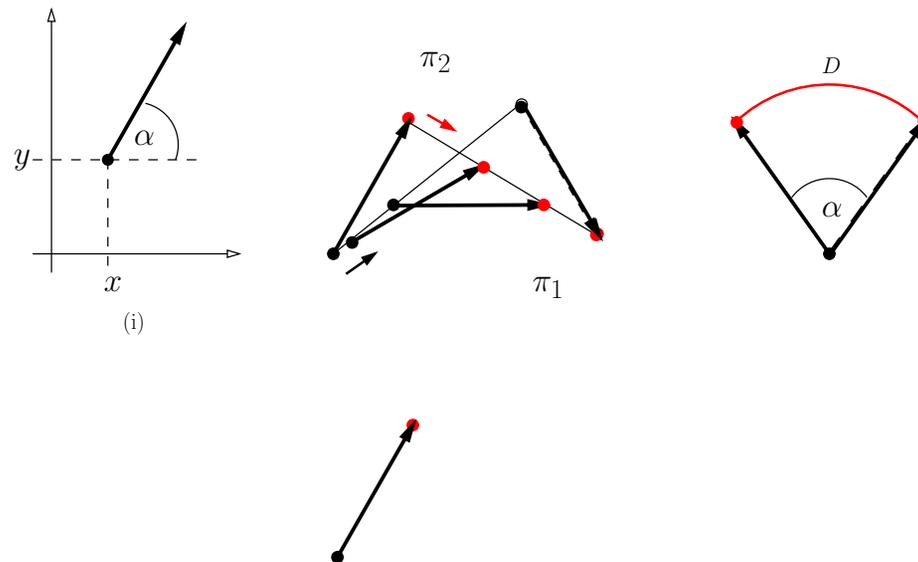
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)
- Translation geht nicht immer



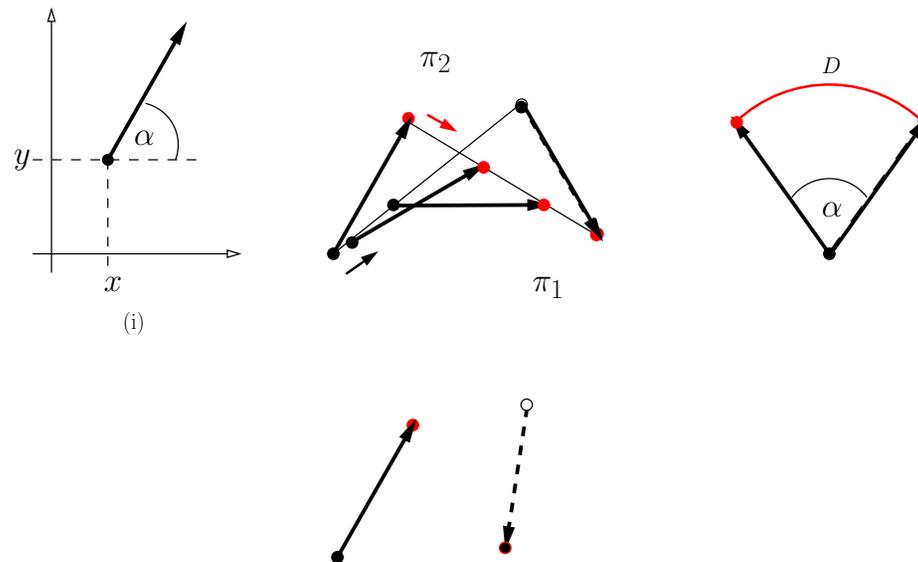
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)
- Translation geht nicht immer



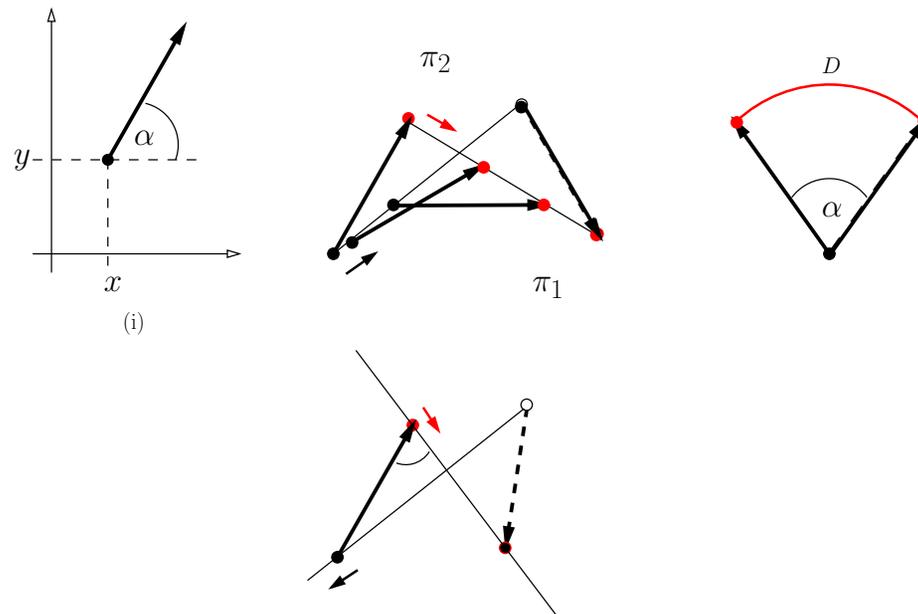
Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)
- Translation geht nicht immer



Formale Beschreibung!

- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)
- Translation geht nicht immer



Kombinationen **Th. 1.47**

Zwischen je zwei Positionen von Liniensegmenten gibt es eine optimale Bewegung von einem der folgenden Typen:

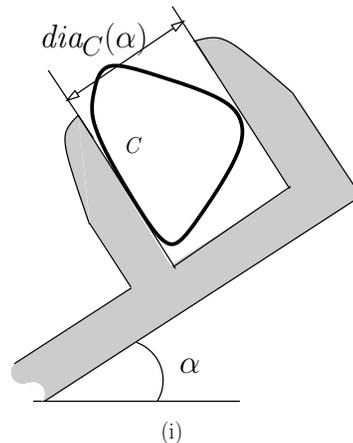
1. maximal drei Rotationen,
2. maximal zwei Rotationen und eine geradlinige Bewegung,
3. eine Rotation zwischen zwei geradlinigen Bewegungen.

Die Bewegungen lassen sich effizient berechnen.

Rotation? Hilfe: Surface-Area Th. 1.46!

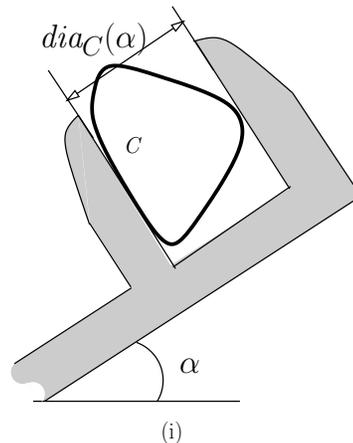
Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$



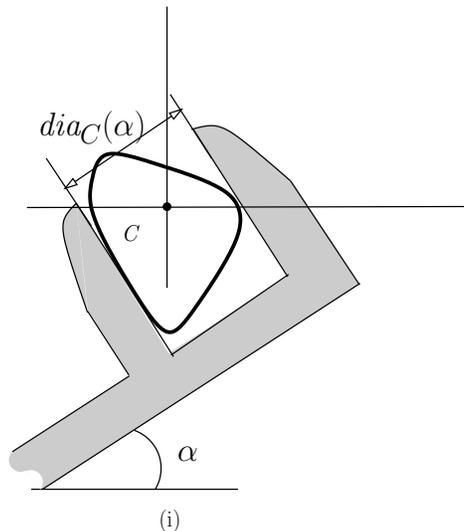
Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$
- Support Funktion: $h_C(\alpha) = \sup\{x \cos \alpha + y \sin \alpha \mid (x, y) \in C\}$
- X -Koordinate nach Rotation mit α



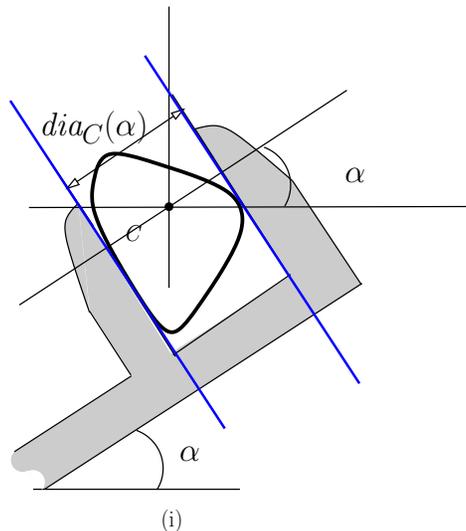
Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$
- Support Funktion: $h_C(\alpha) = \sup\{x \cos \alpha + y \sin \alpha \mid (x, y) \in C\}$
- X -Koordinate nach Rotation mit α



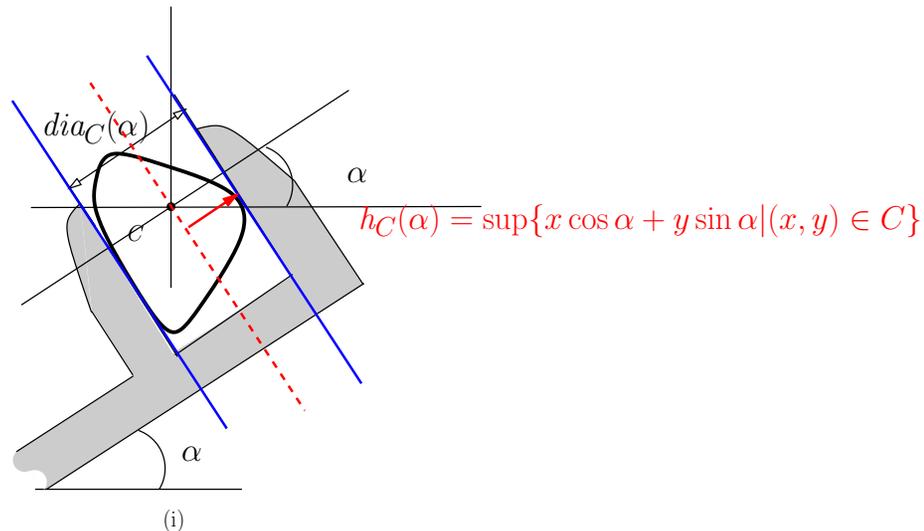
Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$
- Support Funktion: $h_C(\alpha) = \sup\{x \cos \alpha + y \sin \alpha \mid (x, y) \in C\}$
- X -Koordinate nach Rotation mit α



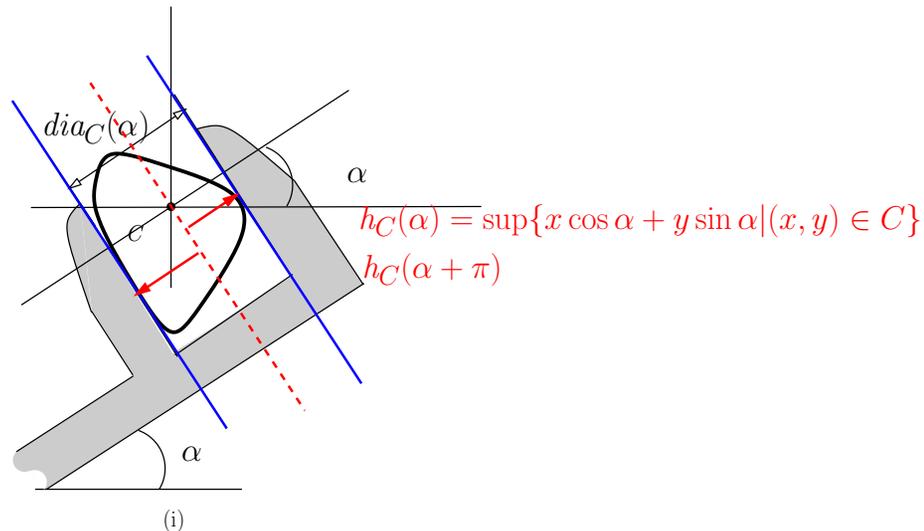
Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$
- Support Funktion: $h_C(\alpha) = \sup\{x \cos \alpha + y \sin \alpha \mid (x, y) \in C\}$
- X -Koordinate nach Rotation mit α



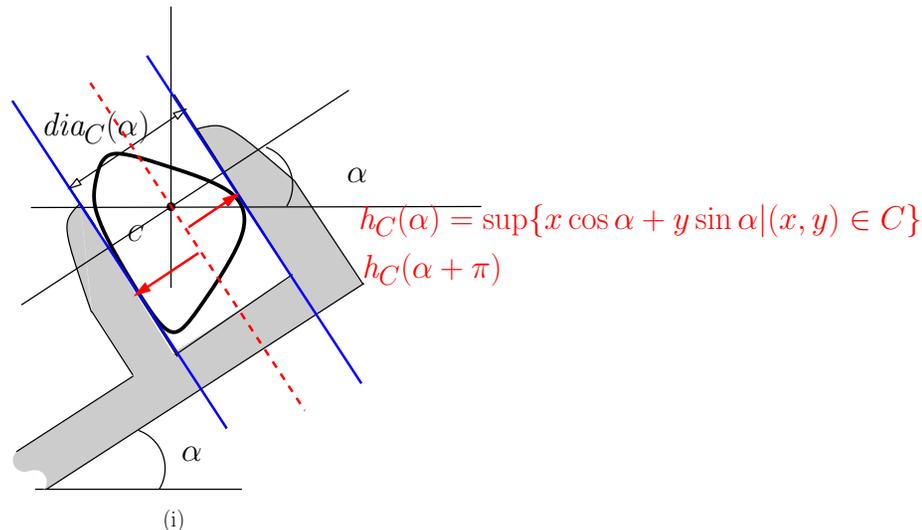
Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$
- Support Funktion: $h_C(\alpha) = \sup\{x \cos \alpha + y \sin \alpha \mid (x, y) \in C\}$
- X -Koordinate nach Rotation mit α
- Offensichtlich: $\text{dia}_C(\alpha) = h_C(\alpha) + h_C(\alpha + \pi)$



Rotation? Hilfe: Surface-Area **Th. 1.46!**

- Geschlossene konvexe Kurve C : Durchm. Funktion:
 $\text{dia}_C : [0, 2\pi) \rightarrow \mathbb{R}$
- Support Funktion: $h_C(\alpha) = \sup\{x \cos \alpha + y \sin \alpha \mid (x, y) \in C\}$
- X -Koordinate nach Rotation mit α
- Offensichtlich: $\text{dia}_C(\alpha) = h_C(\alpha) + h_C(\alpha + \pi)$
- Es gilt: $\text{Length}(C) = \int_0^\pi \text{dia}_C(\alpha) d\alpha$



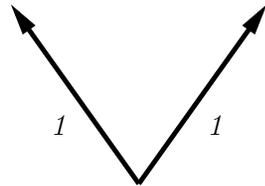
Beweis: Rotation ist optimal!

Beweis: Rotation ist optimal!

- Zwei Kurven C ,

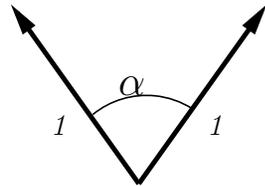
Beweis: Rotation ist optimal!

- Zwei Kurven C ,



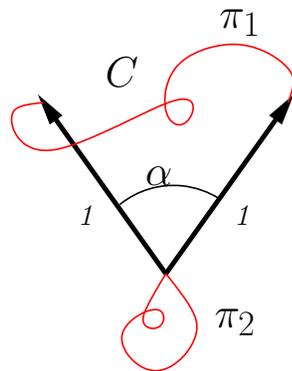
Beweis: Rotation ist optimal!

- Zwei Kurven C ,



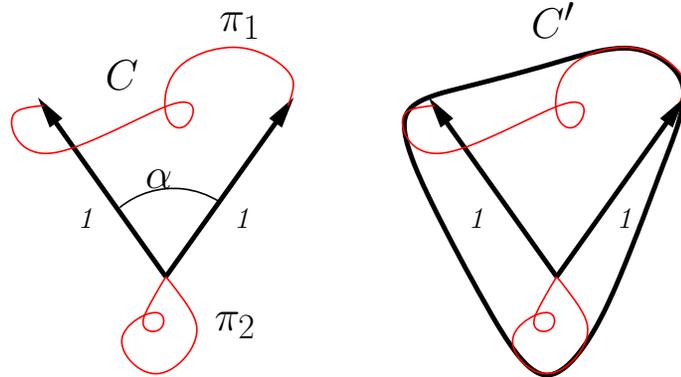
Beweis: Rotation ist optimal!

- Zwei Kurven C , konvexe Hülle C' ,



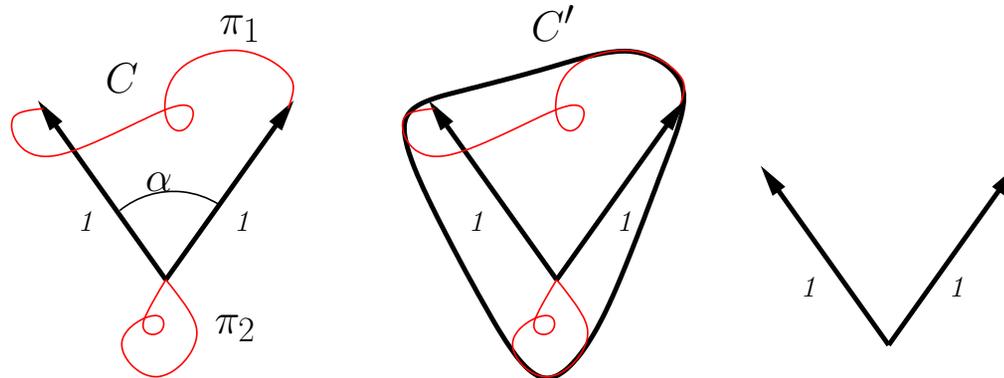
Beweis: Rotation ist optimal!

- Zwei Kurven C , konvexe Hülle C' , D einfache Rotation



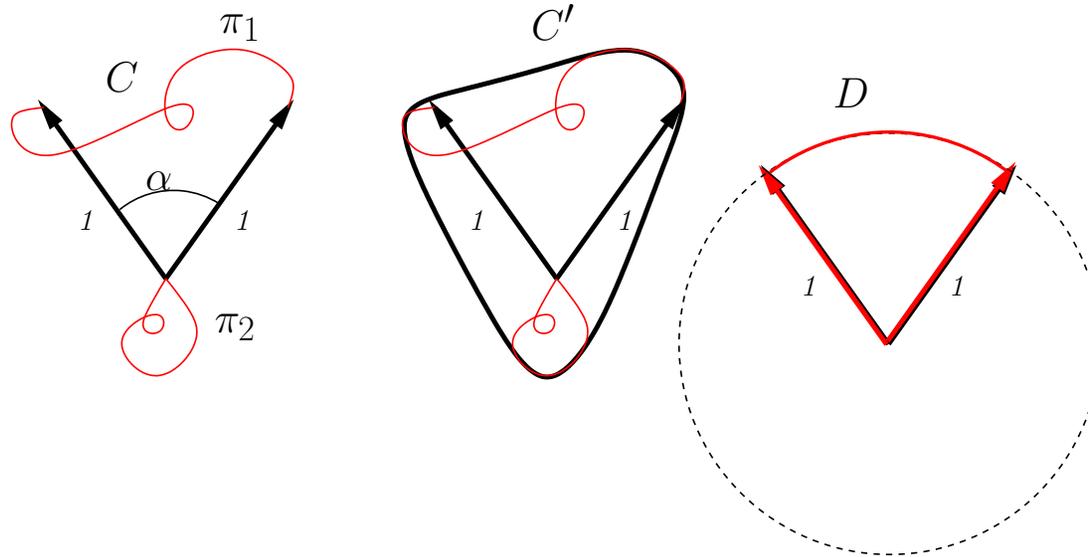
Beweis: Rotation ist optimal!

- Zwei Kurven C , konvexe Hülle C' , D einfache Rotation

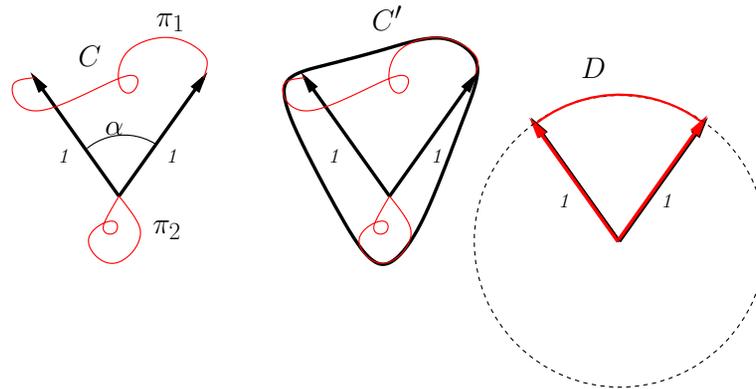


Beweis: Rotation ist optimal!

- Zwei Kurven C , konvexe Hülle C' , D einfache Rotation
- Rotation optimal?

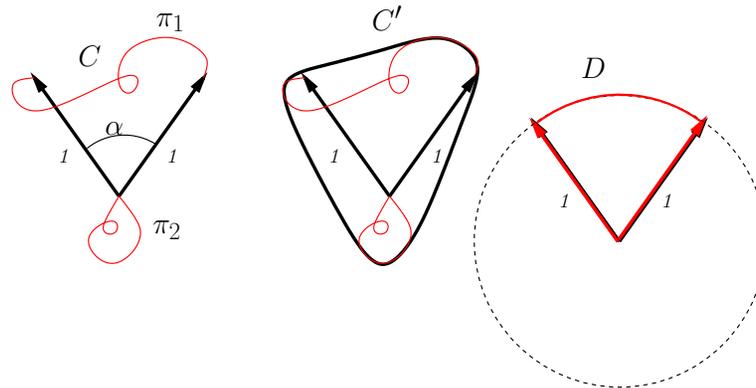


Beweis: Rotation ist optimal!



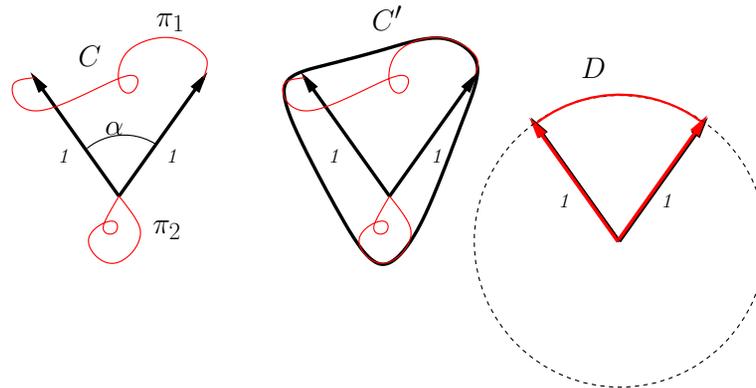
Beweis: Rotation ist optimal!

$2 + \text{Kosten}(C)$



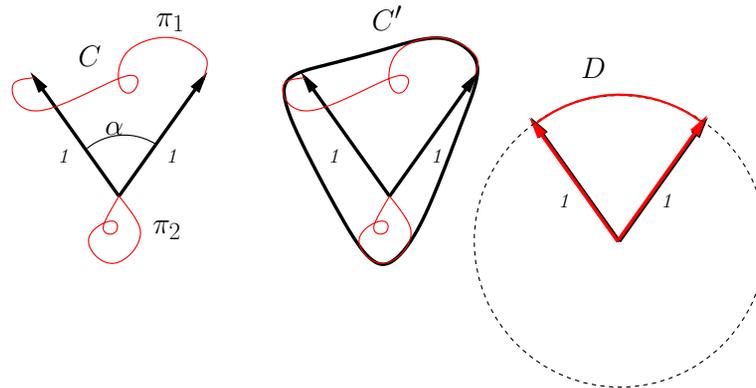
Beweis: Rotation ist optimal!

$$2 + \text{Kosten}(C) \geq \text{Länge}(C')$$



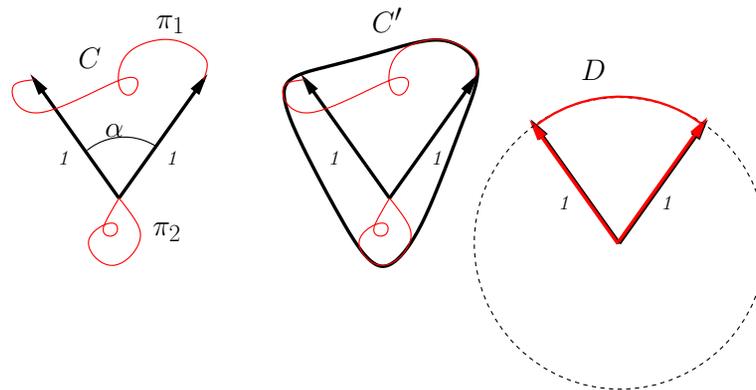
Beweis: Rotation ist optimal!

$$2 + \text{Kosten}(C) \geq \text{Länge}(C') = \int_0^\pi \text{dia}_{C'}(\alpha) d\alpha \quad (\text{Th. 1.46})$$



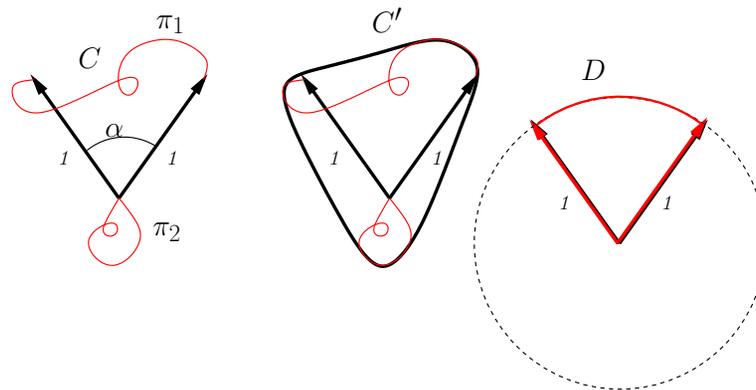
Beweis: Rotation ist optimal!

$$\begin{aligned}
 2 + \text{Kosten}(C) &\geq \text{Länge}(C') = \int_0^\pi \text{dia}_{C'}(\alpha) \, d\alpha \quad (\text{Th. 1.46}) \\
 &\geq \int_0^\pi \text{dia}_D(\alpha) \, d\alpha \quad (\text{wegen } \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha))
 \end{aligned}$$



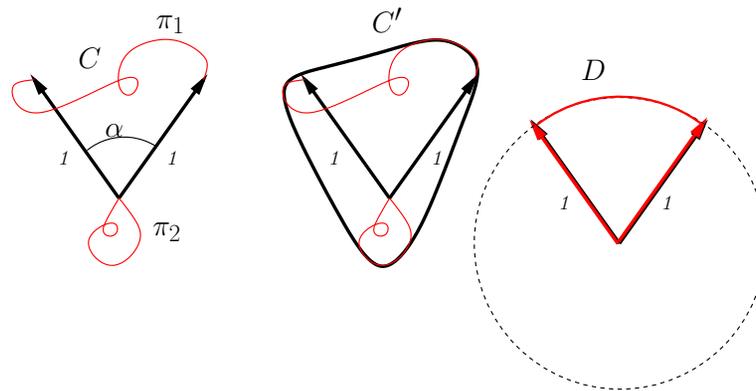
Beweis: Rotation ist optimal!

$$\begin{aligned}
 2 + \text{Kosten}(C) &\geq \text{Länge}(C') = \int_0^\pi \text{dia}_{C'}(\alpha) \, d\alpha \quad (\text{Th. 1.46}) \\
 &\geq \int_0^\pi \text{dia}_D(\alpha) \, d\alpha \quad (\text{wegen } \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)) \\
 &= \text{Länge}(D) \quad (\text{Th. 1.46})
 \end{aligned}$$



Beweis: Rotation ist optimal!

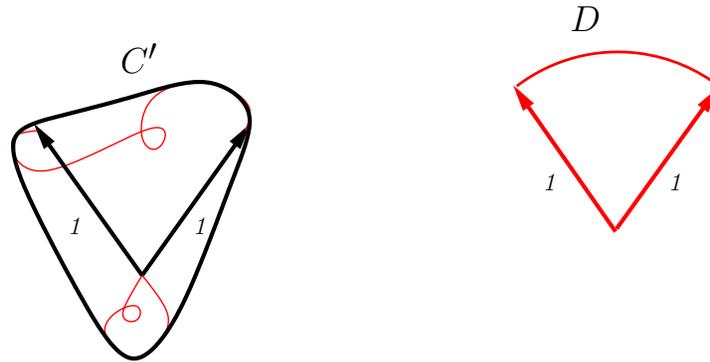
$$\begin{aligned}
 2 + \text{Kosten}(C) &\geq \text{Länge}(C') = \int_0^\pi \text{dia}_{C'}(\alpha) \, d\alpha \quad (\text{Th. 1.46}) \\
 &\geq \int_0^\pi \text{dia}_D(\alpha) \, d\alpha \quad (\text{wegen } \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)) \\
 &= \text{Länge}(D) \quad (\text{Th. 1.46}) = 2 + \text{Kosten}(\text{Rotation})
 \end{aligned}$$



Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha), \alpha \in [0, \pi]$

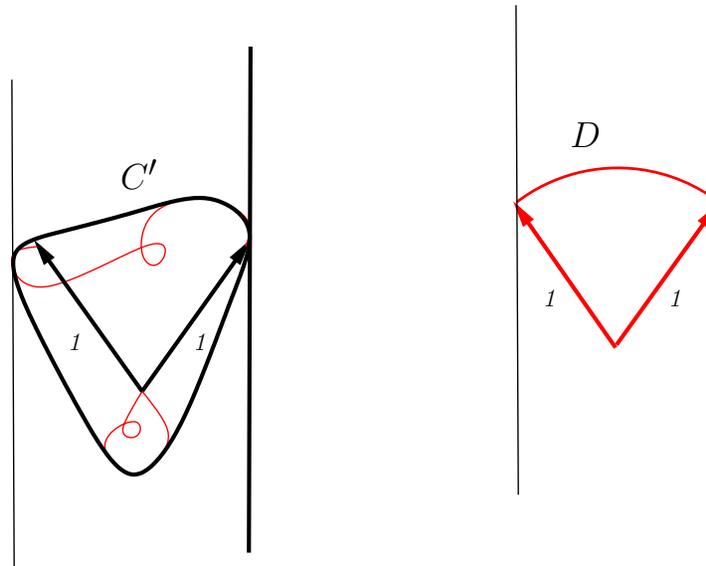
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



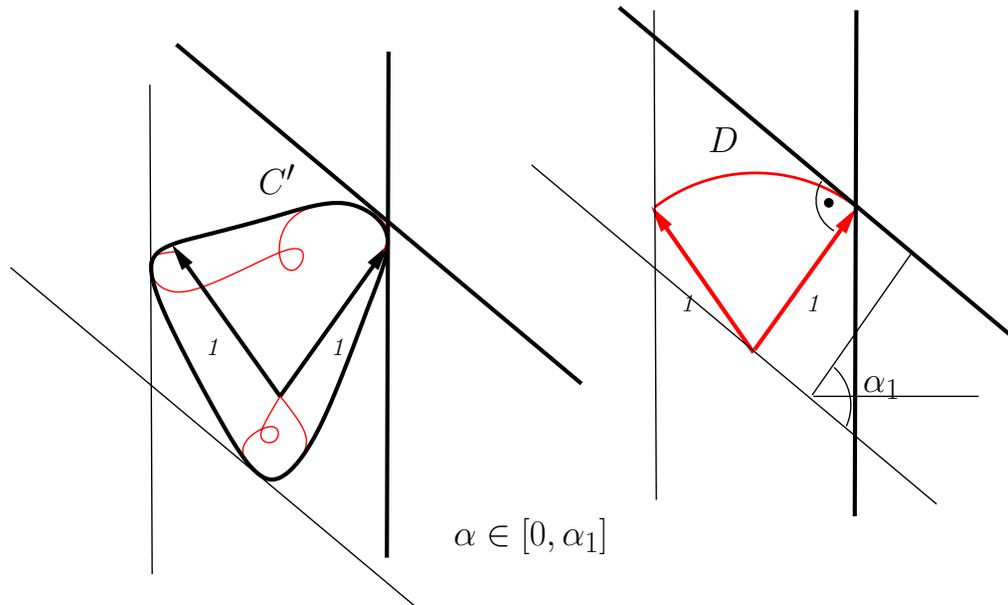
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



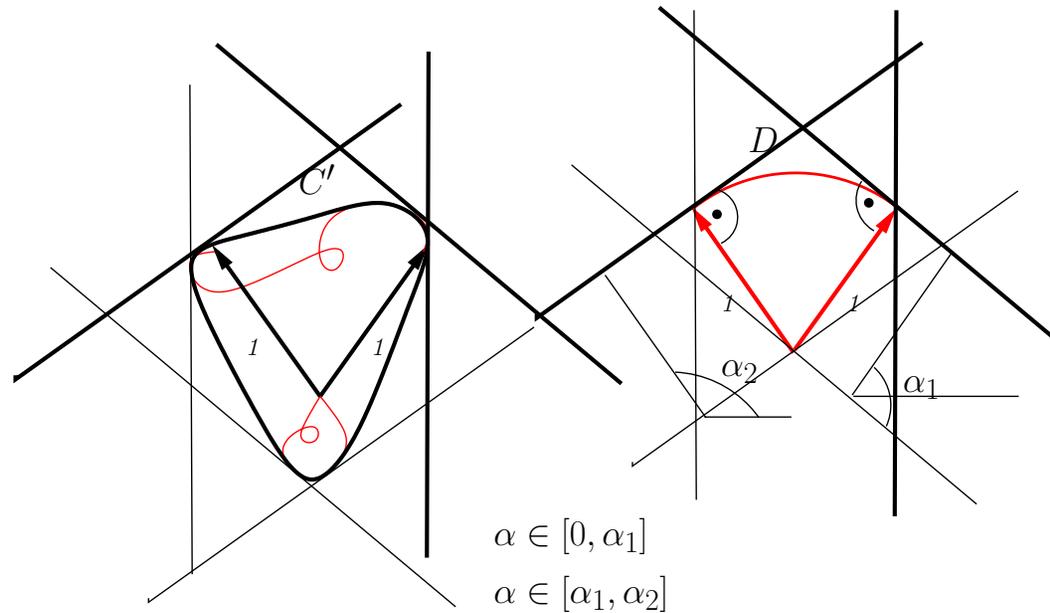
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



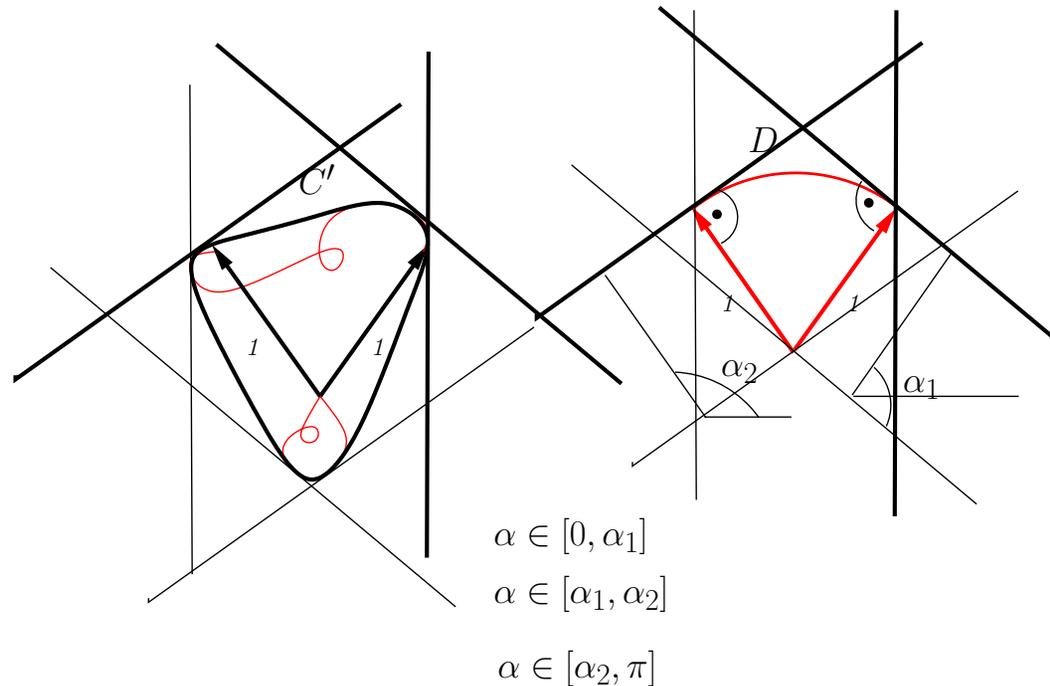
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



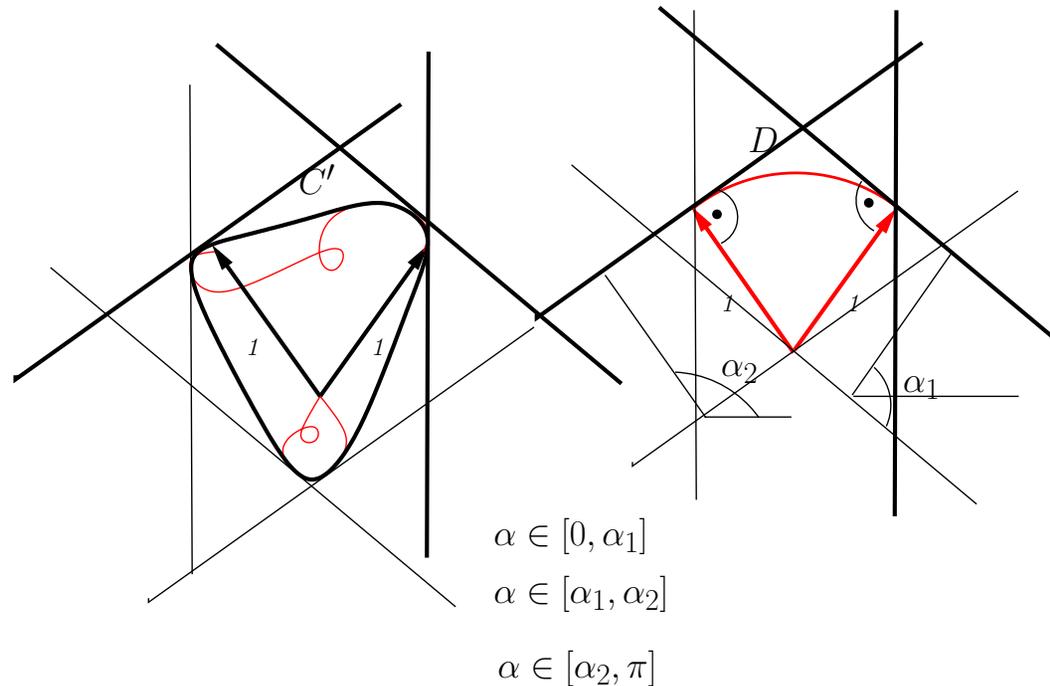
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



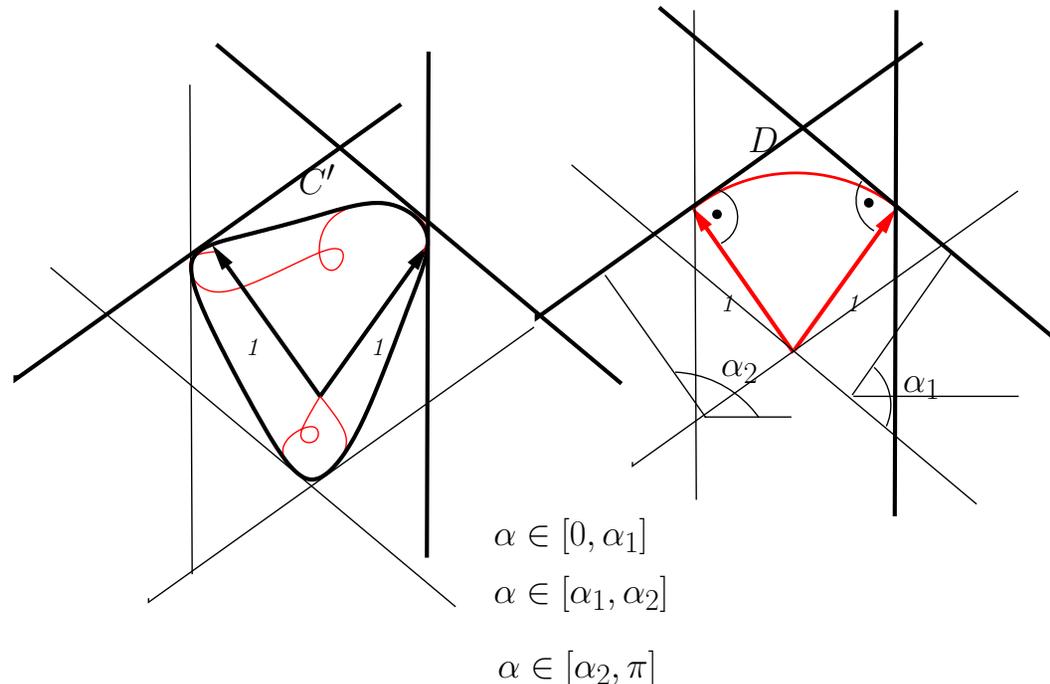
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!
- $\alpha \in [0, \alpha_1] \cup [\alpha_2, \pi] \Rightarrow \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$



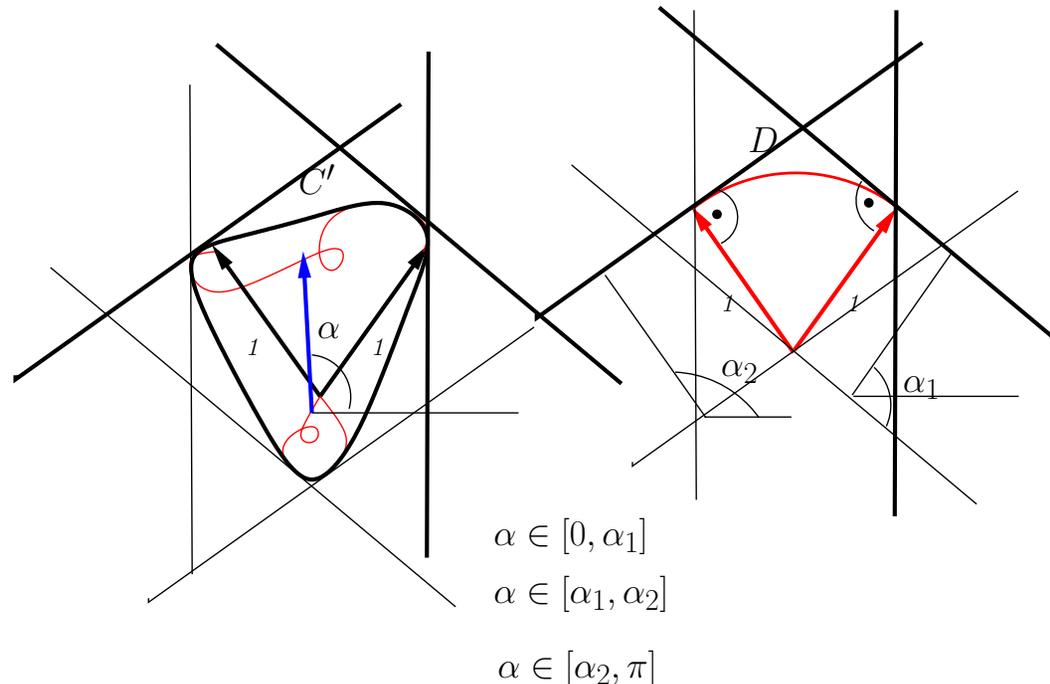
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!
- $\alpha \in [0, \alpha_1] \cup [\alpha_2, \pi] \Rightarrow \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$
- Für $\alpha \in [\alpha_1, \alpha_2]$ ex. stets Platzierung (x, y, α) in C'



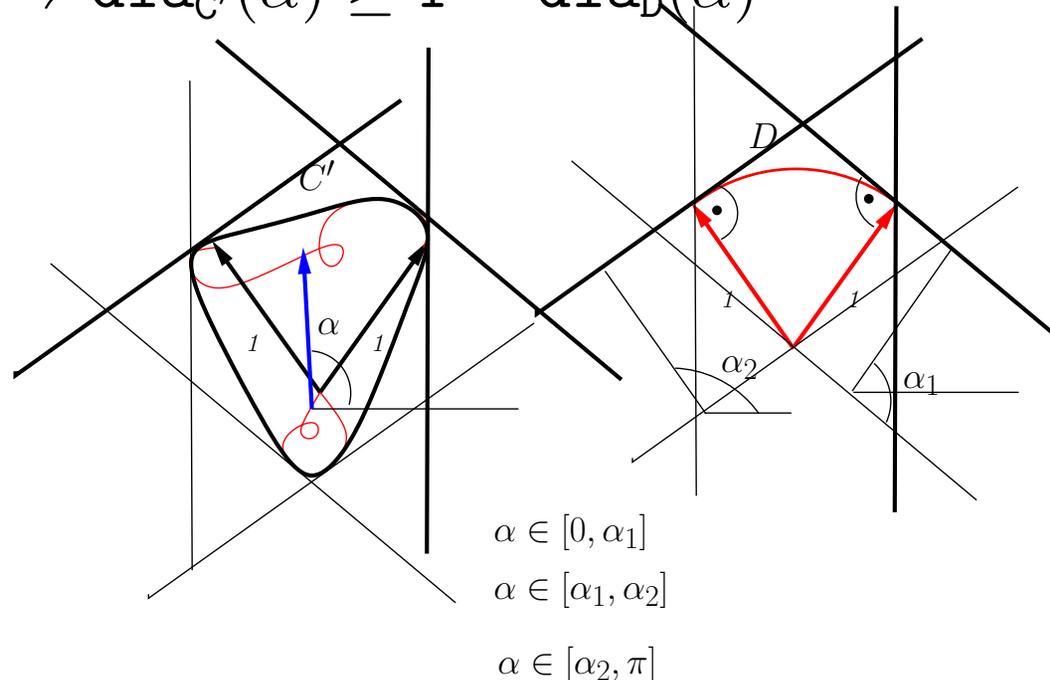
Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!
- $\alpha \in [0, \alpha_1] \cup [\alpha_2, \pi] \Rightarrow \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$
- Für $\alpha \in [\alpha_1, \alpha_2]$ ex. stets Platzierung (x, y, α) in C'



Beweis: $\text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!
- $\alpha \in [0, \alpha_1] \cup [\alpha_2, \pi] \Rightarrow \text{dia}_{C'}(\alpha) \geq \text{dia}_D(\alpha)$
- Für $\alpha \in [\alpha_1, \alpha_2]$ ex. stets Platzierung (x, y, α) in C'
- $\alpha \in [\alpha_1, \alpha_2] \Rightarrow \text{dia}_{C'}(\alpha) \geq 1 = \text{dia}_D(\alpha)$



Kombinationen **Th. 1.47**

Zwischen je zwei Positionen von Liniensegmenten gibt es eine optimale Bewegung von einem der folgenden Typen:

1. maximal drei Rotationen,
2. maximal zwei Rotationen und eine geradlinige Bewegung,
3. eine Rotation zwischen zwei geradlinigen Bewegungen.

Die Bewegungen lassen sich effizient berechnen.

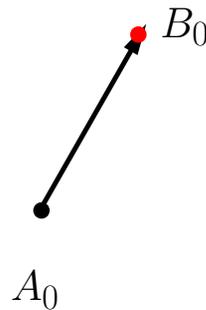
Beispiel: Nahe Placements **Th. 1.47**

Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1

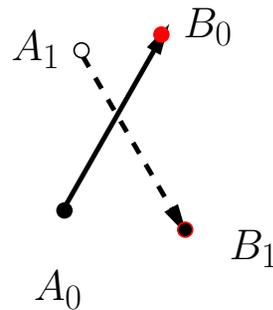
Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1
- Drei Rotationen!!



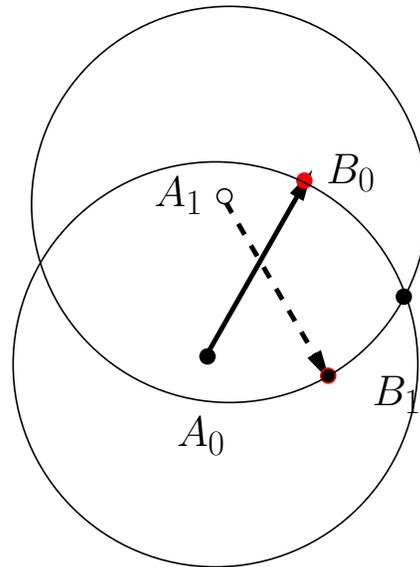
Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1
- Drei Rotationen!!



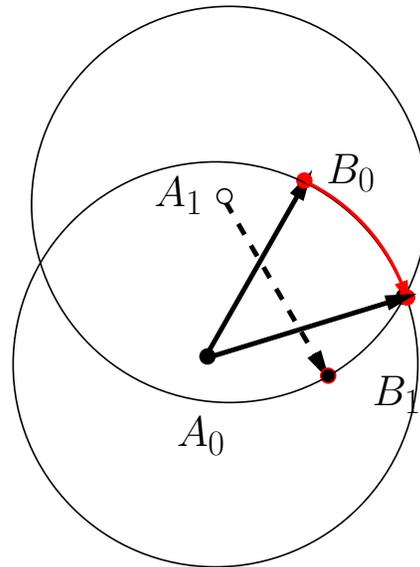
Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1
- Drei Rotationen!!



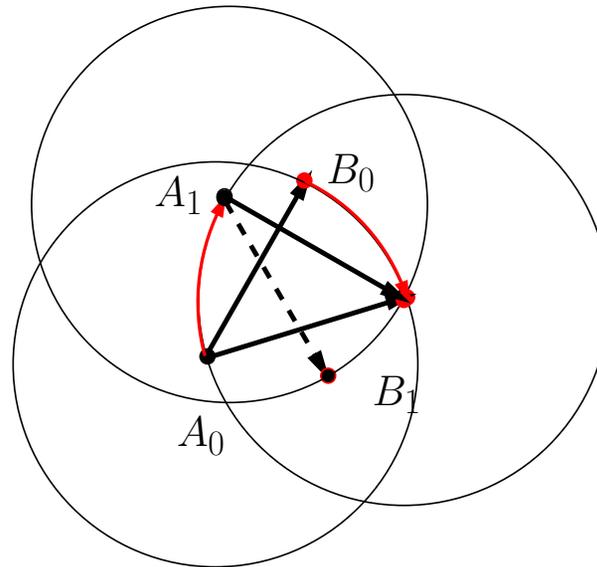
Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1
- Drei Rotationen!!



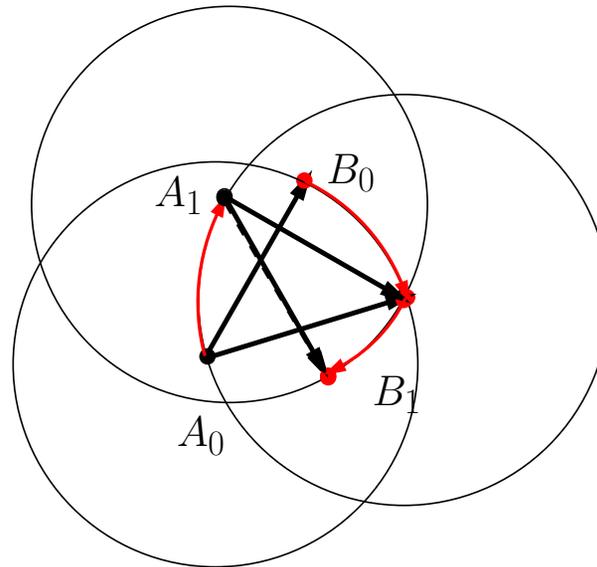
Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1
- Drei Rotationen!!



Beispiel: Nahe Placements **Th. 1.47**

- Abstände $|A_0 - B_1|$ und $|A_1 - B_0|$ kleiner gleich 1
- Drei Rotationen!!



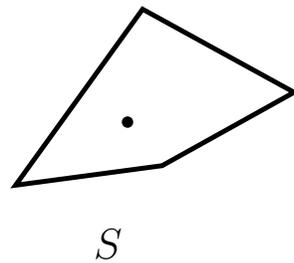
Bewegungen mit ausgedehnten Objekten

Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen

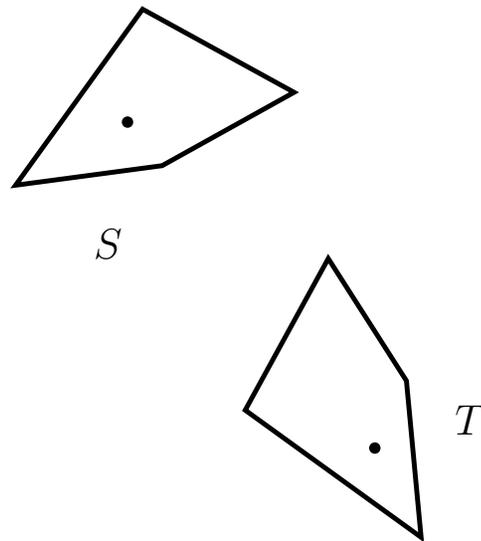
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen



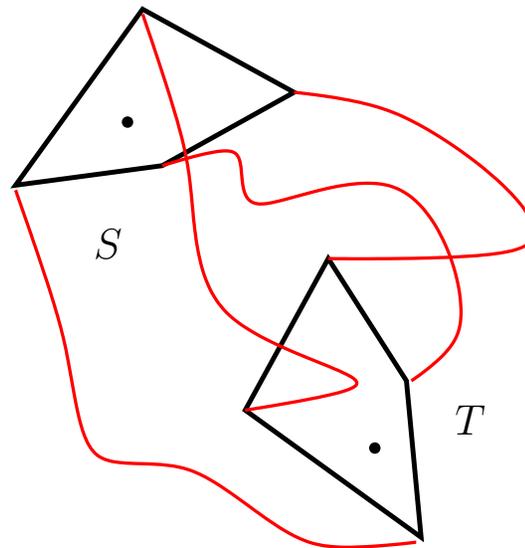
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen



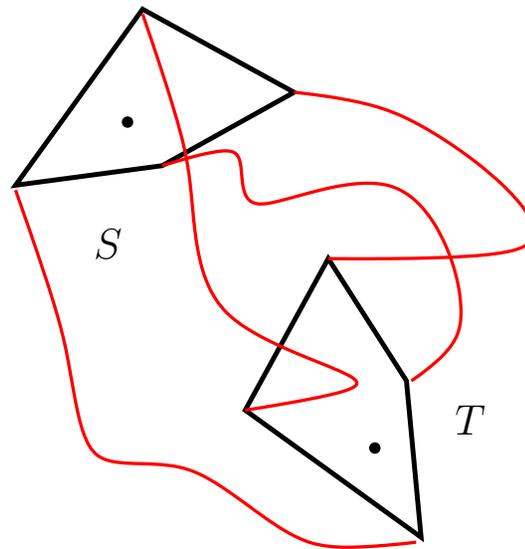
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt



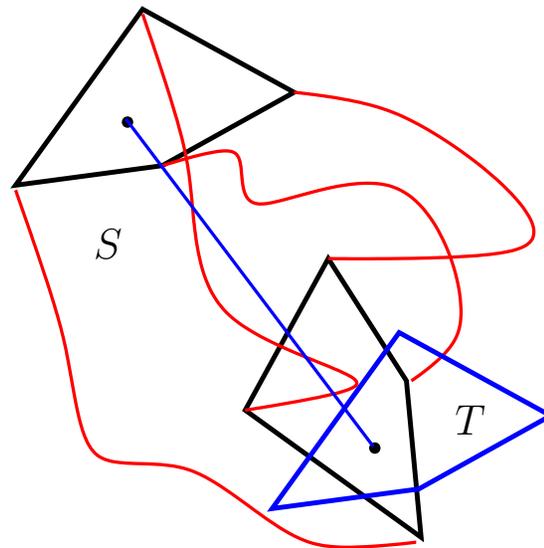
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt



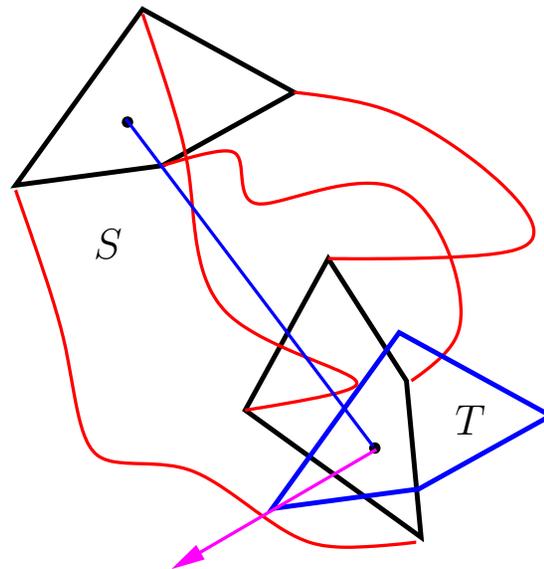
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt



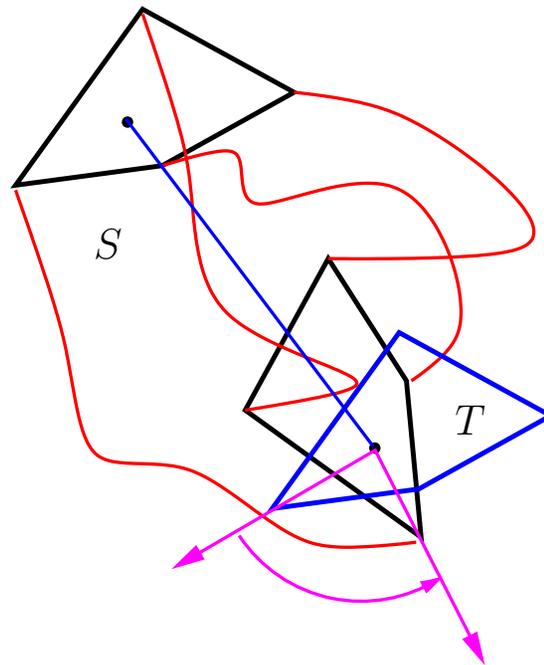
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt



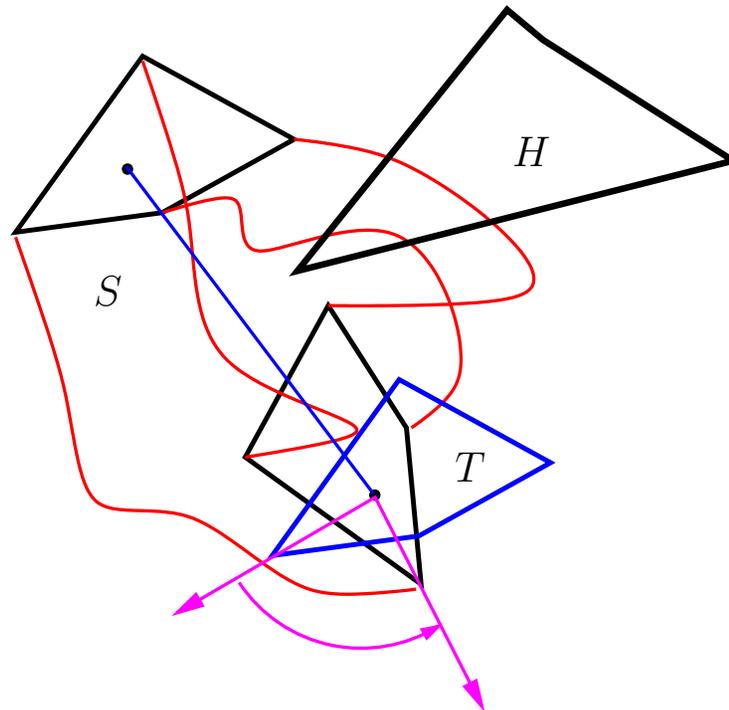
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt



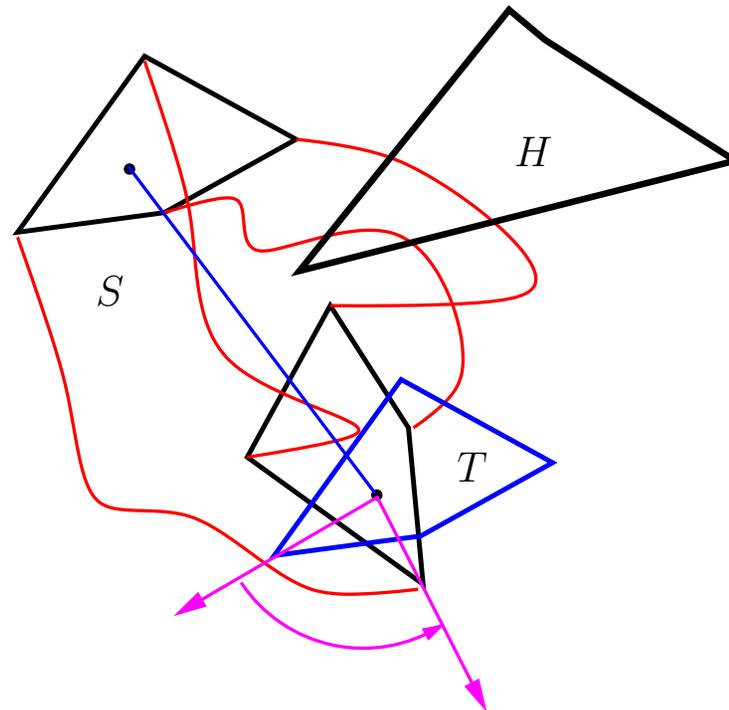
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt



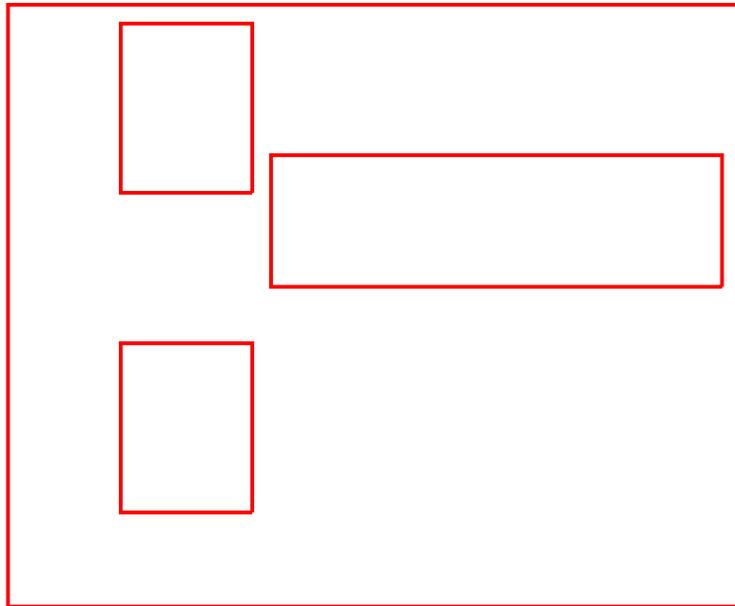
Bewegungen mit ausgedehnten Objekten

- Schwierig zu lösen
- Bewegungen über Referenzpunkt
- Kollisionsfrei??

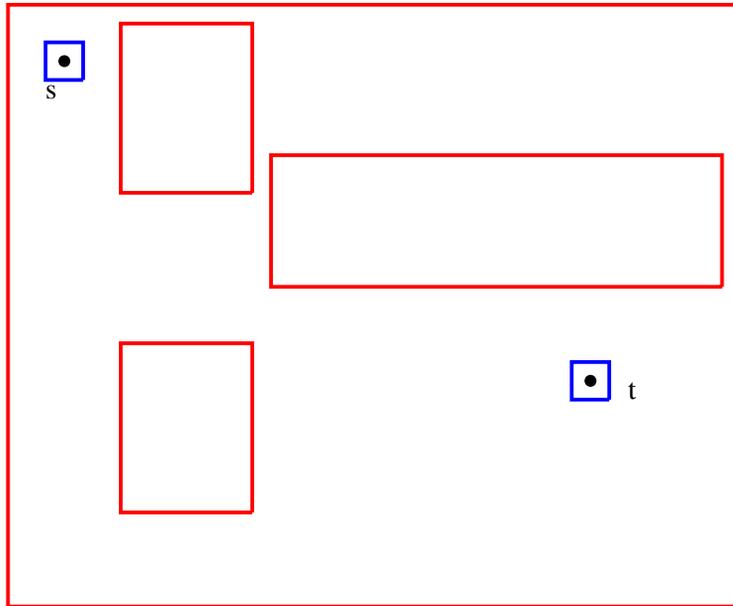


Kollisionsfreie Bahnen: Startbeispiel!!

Kollisionsfreie Bahnen: Startbeispiel!!

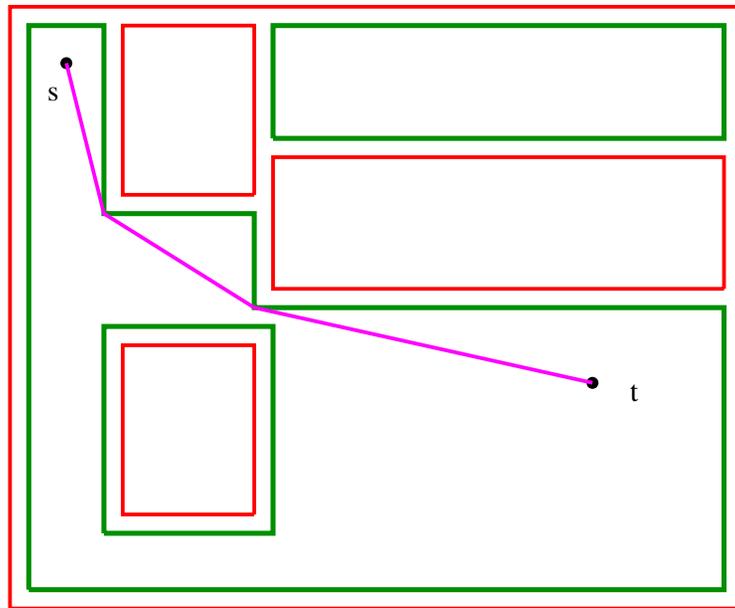


Kollisionsfreie Bahnen: Startbeispiel!!



Translation: Verschieben Referenzpunkt
Aufblasen der Hindernisse

Kollisionsfreie Bahnen: Startbeispiel!!



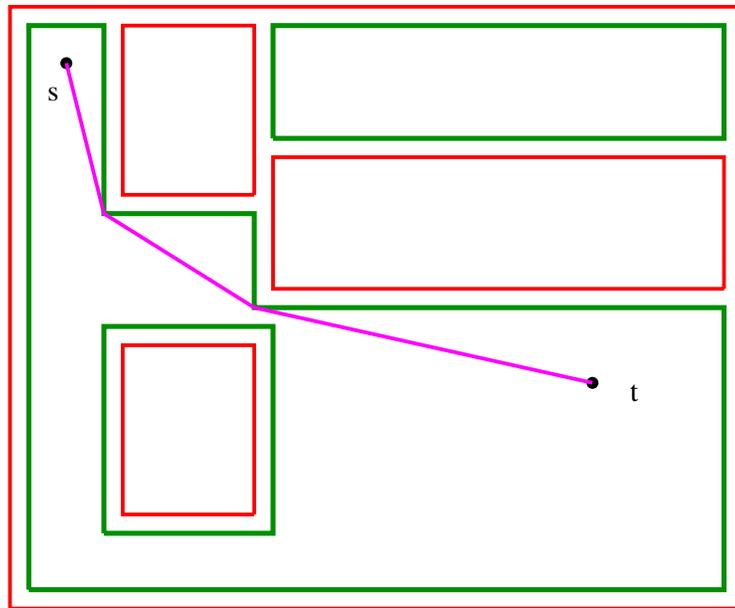
Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse

Vereinigung Sweep

Kürzester Weg????

Kollisionsfreie Bahnen: Startbeispiel!!



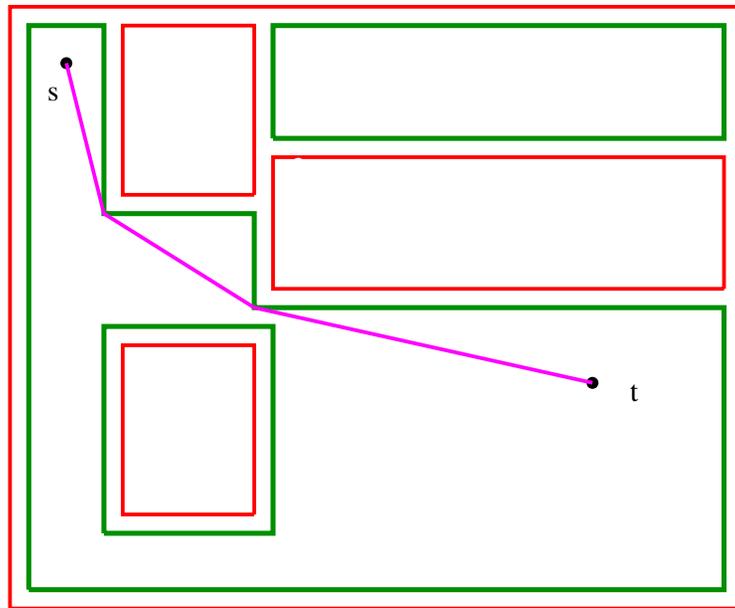
Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse

Vereinigung Sweep

Kürzester Weg????

Kollisionsfreie Bahnen: Startbeispiel!!



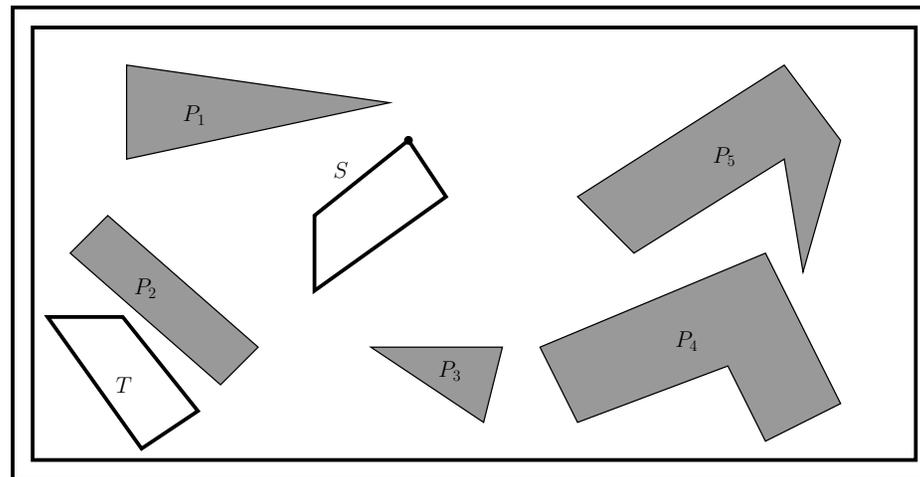
Translation: Verschieben Referenzpunkt

Aufblasen der Hindernisse

Vereinigung Sweep

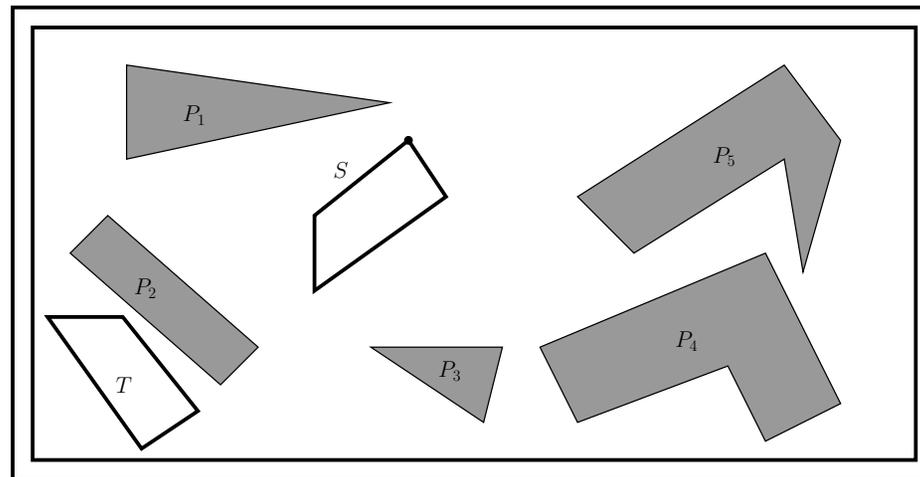
Kürzester Weg????

Kollisionsfreie Bahnen: Allgemeiner!!



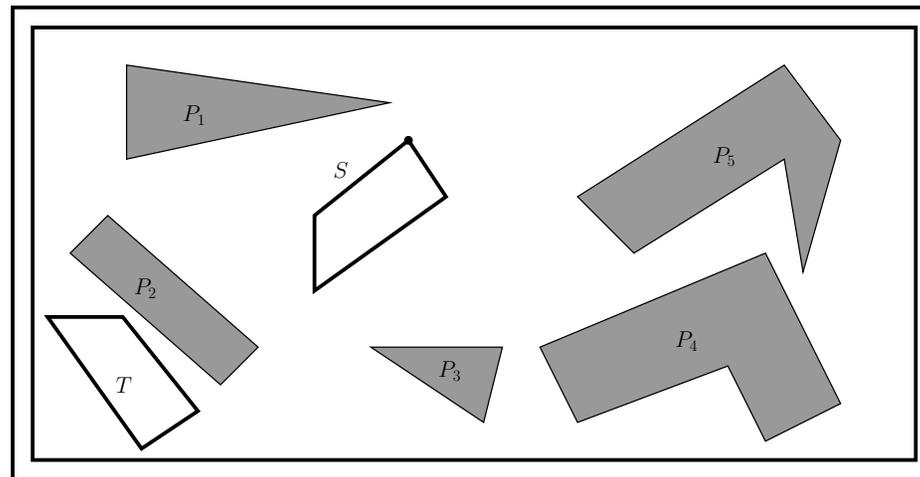
Kollisionsfreie Bahnen: Allgemeiner!!

- Existiert Weg von S nach T



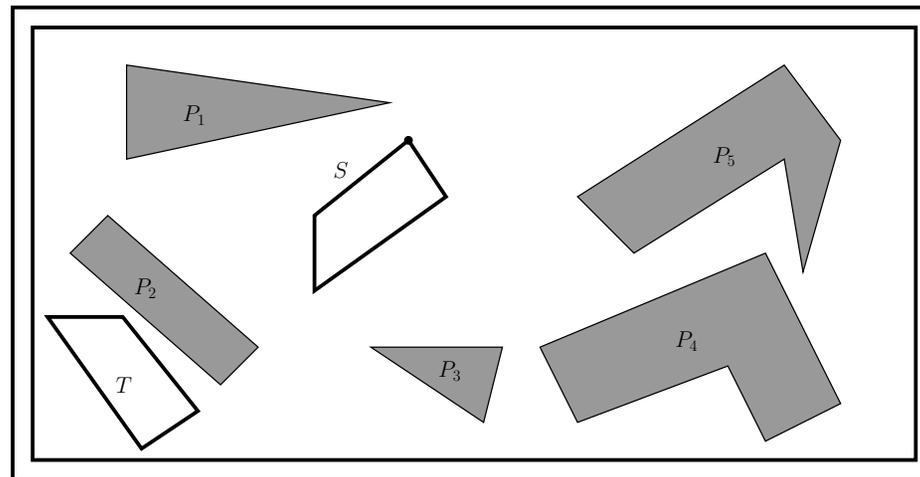
Kollisionsfreie Bahnen: Allgemeiner!!

- Existiert Weg von S nach T
- Berechne Weg von S nach T



Kollisionsfreie Bahnen: Allgemeiner!!

- Existiert Weg von S nach T
- Berechne Weg von S nach T
- Kürzester Weg

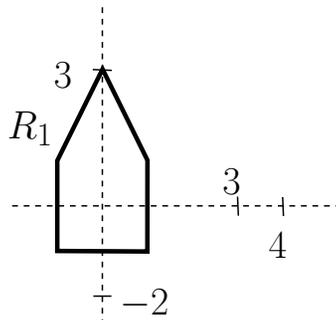


Formal: Rotation und Translation

Formal: Rotation und Translation

- Platzierung Endpunkte

$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

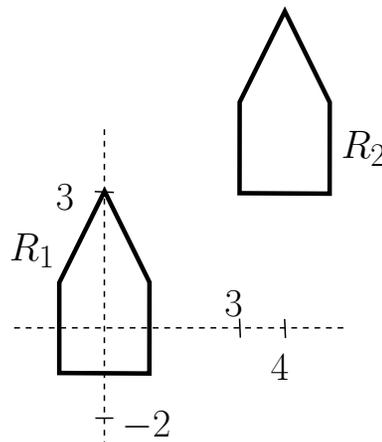


Formal: Rotation und Translation

- Platzierung Endpunkte

$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$



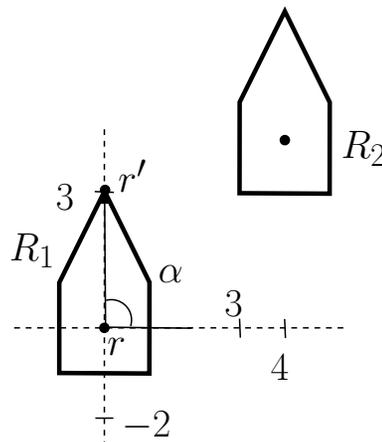
Formal: Rotation und Translation

- Platzierung Endpunkte

$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$

- Referenzpunkt r



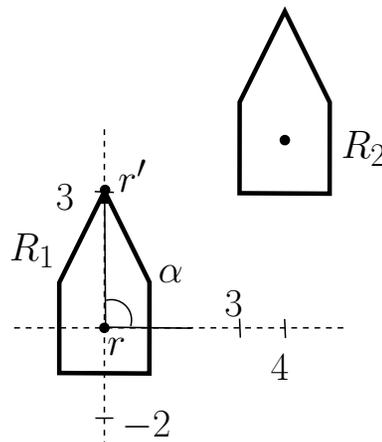
Formal: Rotation und Translation

- Platzierung Endpunkte

$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$

- Referenzpunkt r Winkelreferenzpunkt r'



Formal: Rotation und Translation

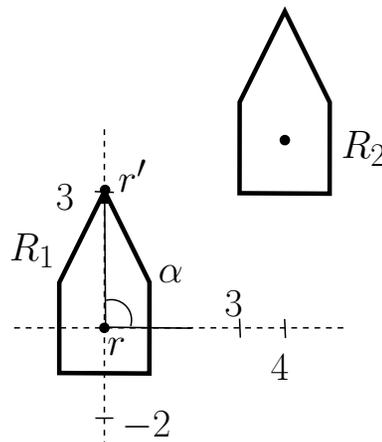
- Platzierung Endpunkte

$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$

- Referenzpunkt r Winkelreferenzpunkt r'

- $R_1 = R(0, 0, \pi/2) = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$



Formal: Rotation und Translation

- Platzierung Endpunkte

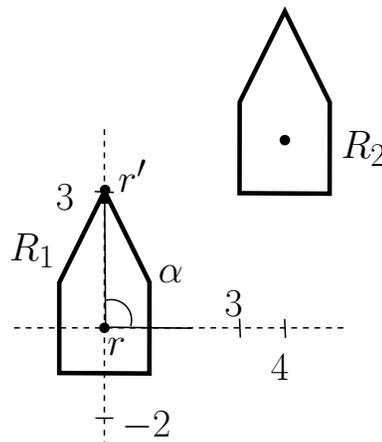
$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$

- Referenzpunkt r Winkelreferenzpunkt r'

- $R_1 = R(0, 0, \pi/2) = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$

- $R(x, y, \alpha)$,



Formal: Rotation und Translation

- Platzierung Endpunkte

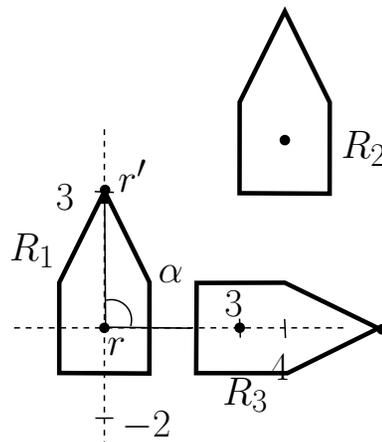
$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$

- Referenzpunkt r Winkelreferenzpunkt r'

- $R_1 = R(0, 0, \pi/2) = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$

- $R(x, y, \alpha)$, Translation/Rotation:



Formal: Rotation und Translation

- Platzierung Endpunkte

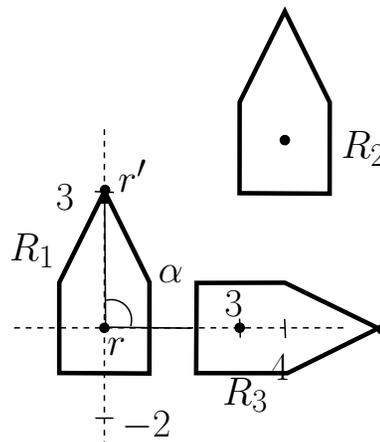
$$R_1 = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$$

- Verschiebung $R_2 = \{(5, 3), (5, 5), (4, 7), (3, 5), (3, 3)\}$

- Referenzpunkt r Winkelreferenzpunkt r'

- $R_1 = R(0, 0, \pi/2) = \{(1, -1), (1, 1), (0, 3), (-1, 1), (-1, -1)\}$

- $R(x, y, \alpha)$, Translation/Rotation: $R_3 = R(3, 0, 0)$



Formale Definitionen: Def. 2.1

Formale Definitionen: Def. 2.1

- Tripel $(x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ Platzierung, 3 DOF (Freiheitsgrade) Referenzpunkt+Winkel!!!

Formale Definitionen: Def. 2.1

- Tripel $(x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ Platzierung, 3 DOF (Freiheitsgrade) Referenzpunkt+Winkel!!!
- Konfigurationsraum $C = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$

Formale Definitionen: Def. 2.1

- Tripel $(x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ Platzierung, 3 DOF (Freiheitsgrade) Referenzpunkt+Winkel!!!
- Konfigurationsraum $C = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$
- Arbeitsraum: $R(x, y, \alpha), P_1, P_2, \dots, P_k$

Formale Definitionen: Def. 2.1

- Tripel $(x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ Platzierung, 3 DOF (Freiheitsgrade) Referenzpunkt+Winkel!!!
- Konfigurationsraum $C = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$
- Arbeitsraum: $R(x, y, \alpha), P_1, P_2, \dots, P_k$
- Verbotene Platzierungen $C_{\text{verb}} := \left\{ c \in C \mid R(c) \cap \bigcup \mathring{P}_i \neq \emptyset \right\}$

Formale Definitionen: Def. 2.1

- Tripel $(x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ Platzierung, 3 DOF (Freiheitsgrade) Referenzpunkt+Winkel!!!
- Konfigurationsraum $C = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$
- Arbeitsraum: $R(x, y, \alpha), P_1, P_2, \dots, P_k$
- Verbotene Platzierungen $C_{\text{verb}} := \left\{ c \in C \mid R(c) \cap \bigcup \mathring{P}_i \neq \emptyset \right\}$
- Freie Platzierungen $C_{\text{frei}} := C \setminus C_{\text{verb}}$

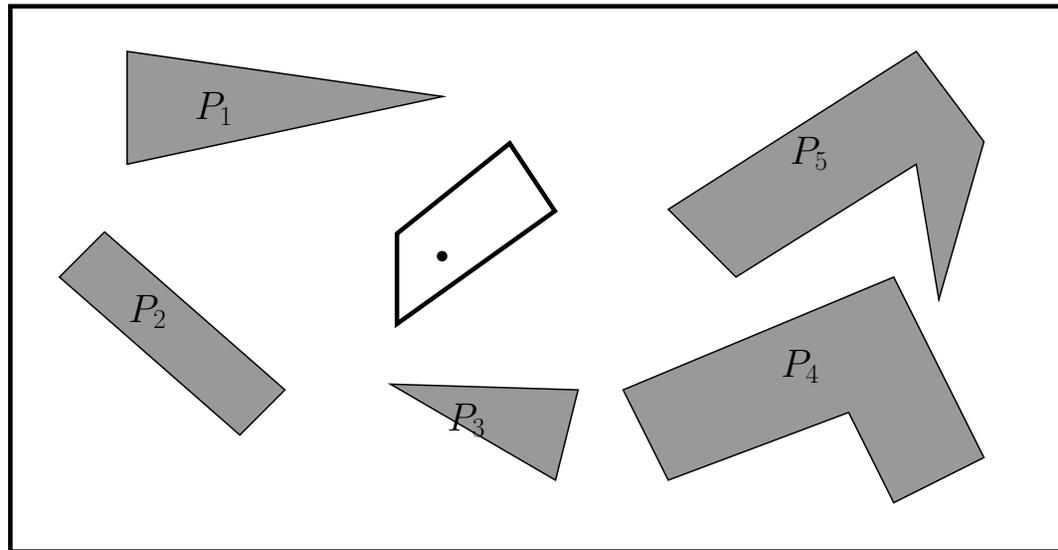
Formale Definitionen: Def. 2.1

- Tripel $(x, y, \alpha) \in \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$ Platzierung, 3 DOF (Freiheitsgrade) Referenzpunkt+Winkel!!!
- Konfigurationsraum $C = \mathbb{R} \times \mathbb{R} \times [0, 2\pi)$
- Arbeitsraum: $R(x, y, \alpha), P_1, P_2, \dots, P_k$
- Verbotene Platzierungen $C_{\text{verb}} := \left\{ c \in C \mid R(c) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Freie Platzierungen $C_{\text{frei}} := C \setminus C_{\text{verb}}$
- Halbfreie Platzierungen

Beispiel!!

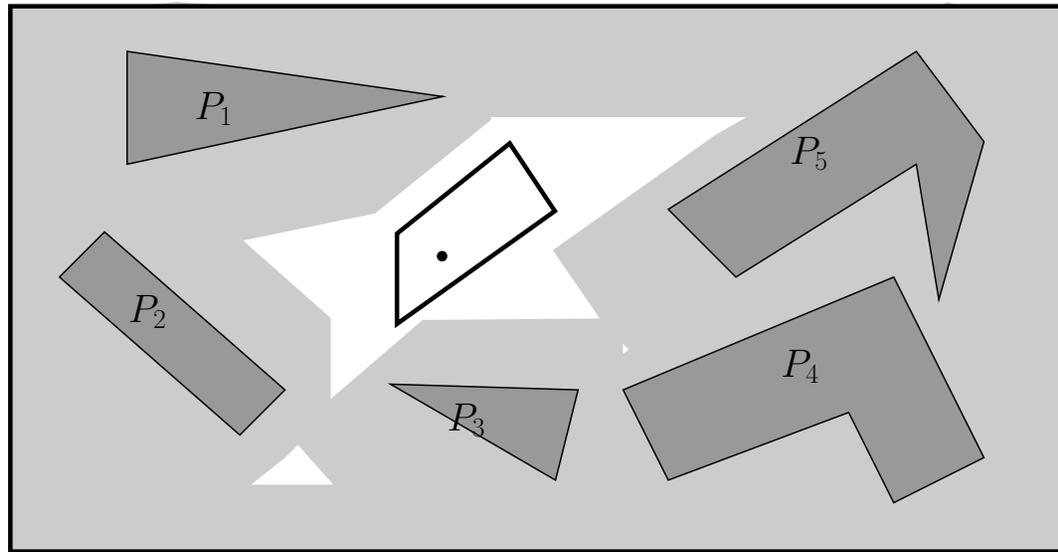
- Reine Translation
- Bereich der gültigen Platzierungen

Beispiel!!



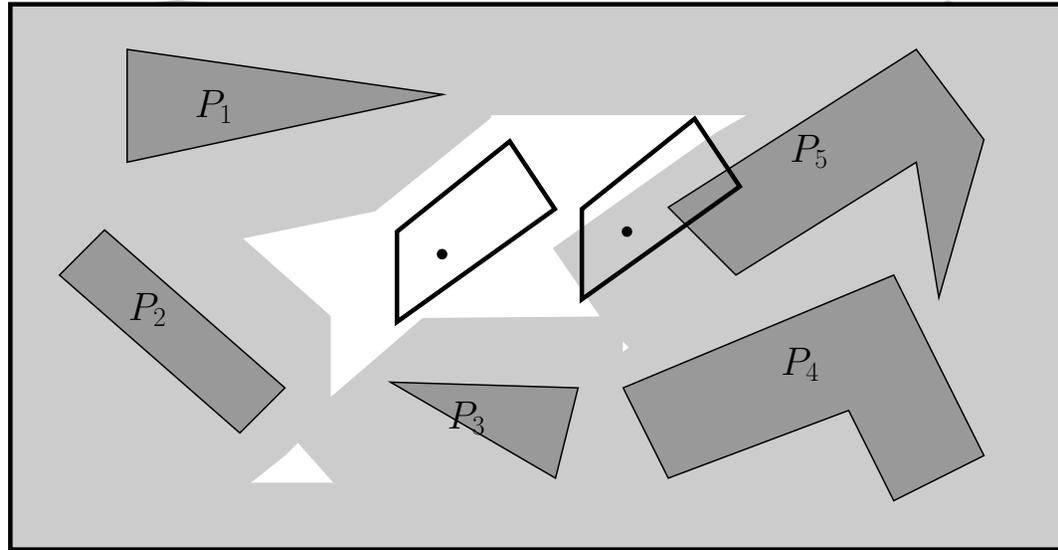
- Reine Translation
- Bereich der gültigen Platzierungen

Beispiel!!



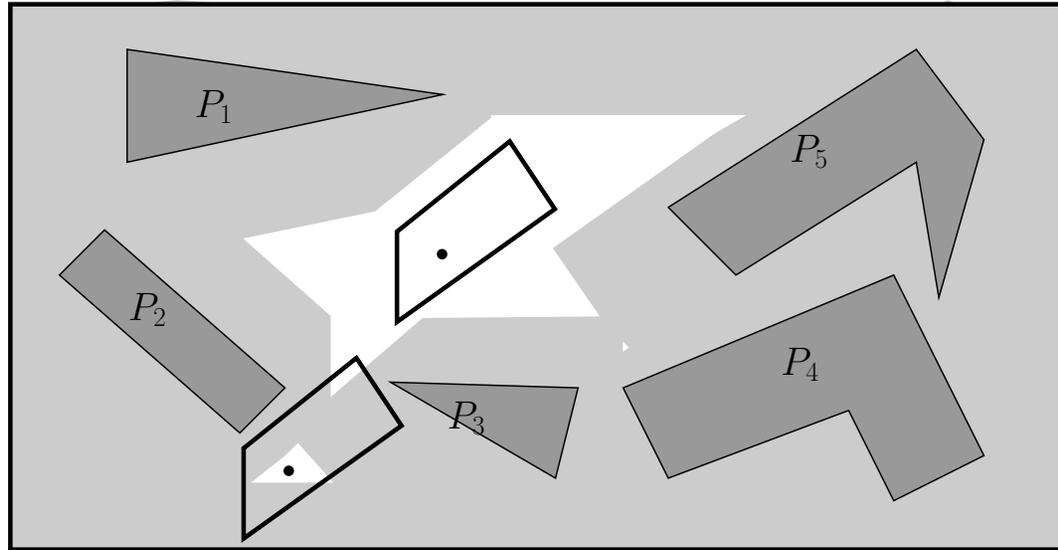
- Reine Translation
- Bereich der gültigen Platzierungen

Beispiel!!



- Reine Translation
- Bereich der gültigen Platzierungen

Beispiel!!



- Reine Translation
- Bereich der gültigen Platzierungen

Weg-zusammenhängend **Def. 2.2**

Weg-zusammenhängend **Def. 2.2**

- Menge Z ($Z \subseteq \mathbb{R}^d$)

Weg-zusammenhängend **Def. 2.2**

- Menge Z ($Z \subseteq \mathbb{R}^d$)
- Pfad π von a nach b :

Weg-zusammenhängend **Def. 2.2**

- Menge Z ($Z \subseteq \mathbb{R}^d$)
- Pfad π von a nach b :
 - Stetige Abbildung $\pi : [0, 1] \rightarrow Z$

Weg-zusammenhängend **Def. 2.2**

- Menge Z ($Z \subseteq \mathbb{R}^d$)
- Pfad π von a nach b :
 - Stetige Abbildung $\pi : [0, 1] \rightarrow Z$
 - $\pi(0) = a, \pi(1) = b$

Weg-zusammenhängend **Def. 2.2**

- Menge Z ($Z \subseteq \mathbb{R}^d$)
- Pfad π von a nach b :
 - Stetige Abbildung $\pi : [0, 1] \rightarrow Z$
 - $\pi(0) = a, \pi(1) = b$
- weg-zusammenhängend:

Weg-zusammenhängend **Def. 2.2**

- Menge Z ($Z \subseteq \mathbb{R}^d$)
- Pfad π von a nach b :
 - Stetige Abbildung $\pi : [0, 1] \rightarrow Z$
 - $\pi(0) = a, \pi(1) = b$
- weg-zusammenhängend:
 - $\forall a, b \in Z : \exists$ Pfad π von a nach b in Z .

Zusammenhangskomponenten **Lemma 2.3**

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$):

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:
 - Reflexiv: $a \sim a$

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:
 - Reflexiv: $a \sim a$
 - Symmetrisch: $a \sim b \Rightarrow b \sim a$

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:
 - Reflexiv: $a \sim a$
 - Symmetrisch: $a \sim b \Rightarrow b \sim a$
 - Transitiv: $a \sim b, b \sim c \Rightarrow a \sim c$

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:
 - Reflexiv: $a \sim a$
 - Symmetrisch: $a \sim b \Rightarrow b \sim a$
 - Transitiv: $a \sim b, b \sim c \Rightarrow a \sim c$
- Äquivalenzklassen (Zusammenhangskomponenten)
 - größte weg-zusammenhängende Teilmengen

Zusammenhangskomponenten **Lemma 2.3**

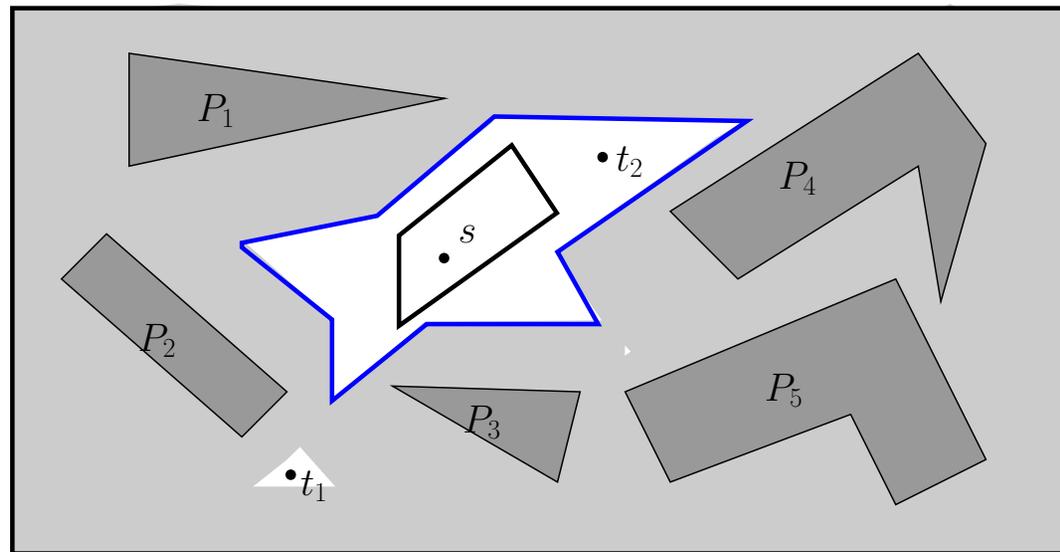
- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:
 - Reflexiv: $a \sim a$
 - Symmetrisch: $a \sim b \Rightarrow b \sim a$
 - Transitiv: $a \sim b, b \sim c \Rightarrow a \sim c$
- Äquivalenzklassen (Zusammenhangskomponenten)
 - größte weg-zusammenhängende Teilmengen
 - überdecken A disjunkt:

Zusammenhangskomponenten **Lemma 2.3**

- Relation auf $A \times A$ ($A \subseteq \mathbb{R}^d$): $a \sim b :\Leftrightarrow \exists$ Pfad $\pi \subset A$ von a nach b in A mit $\pi(0) = a$ und $\pi(1) = b$
- Äquivalenzrelation:
 - Reflexiv: $a \sim a$
 - Symmetrisch: $a \sim b \Rightarrow b \sim a$
 - Transitiv: $a \sim b, b \sim c \Rightarrow a \sim c$
- Äquivalenzklassen (Zusammenhangskomponenten)
 - größte weg-zusammenhängende Teilmengen
 - überdecken A disjunkt: $A = \dot{\bigcup} Z_i$

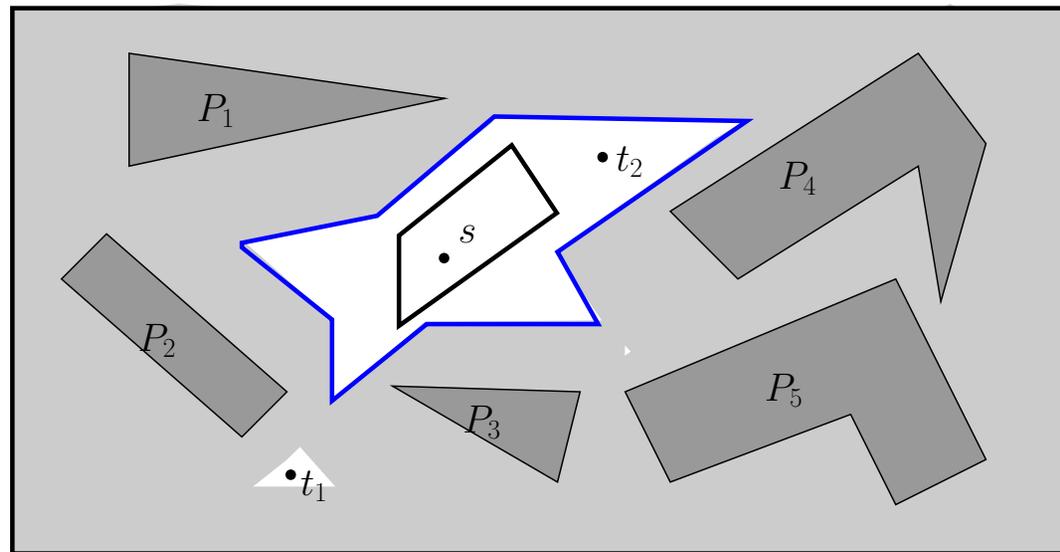
Aufgabe: Beispiel Translation!

- Berechne Zusammenhangskomponente von Z_s , s Startkoordinaten



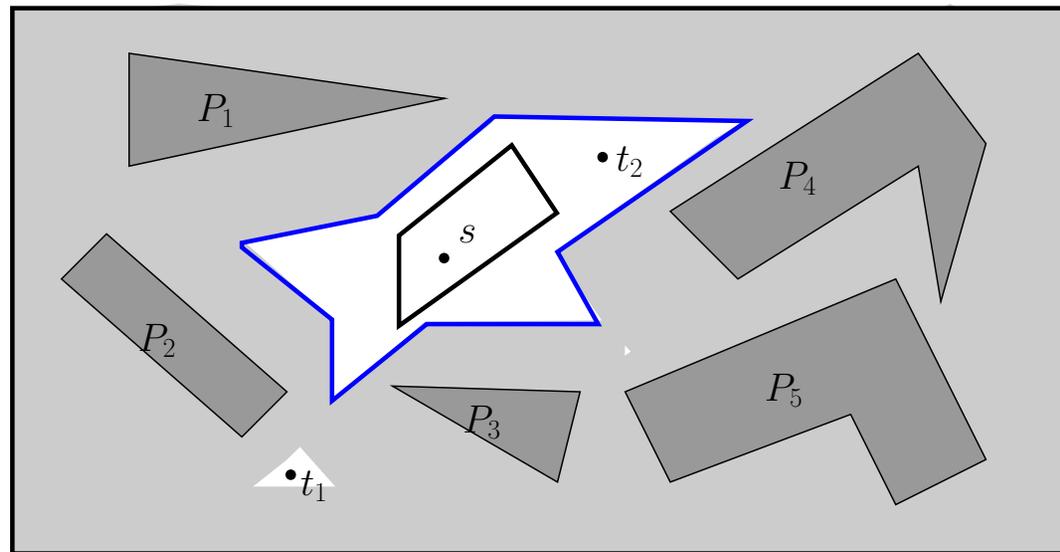
Aufgabe: Beispiel Translation!

- Berechne Zusammenhangskomponente von Z_s , s Startkoordinaten
- Liegt t auch in Z_s ?



Aufgabe: Beispiel Translation!

- Berechne Zusammenhangskomponente von Z_s , s Startkoordinaten
- Liegt t auch in Z_s ?
- Berechne ggf. Weg s nach t in Z_s



Reine Translationsbewegung, konvexe Roboter **2.1.2**

Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k

Reine Translationsbewegung, konvexe Roboter 2.1.2

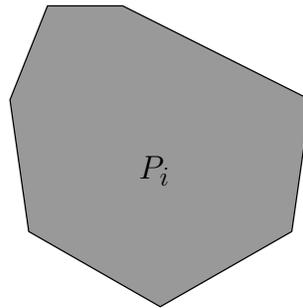
- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$

Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i

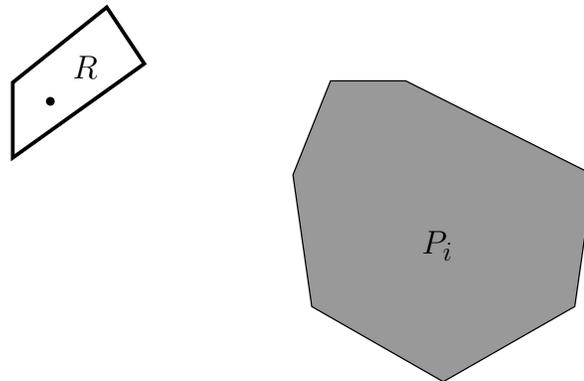
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



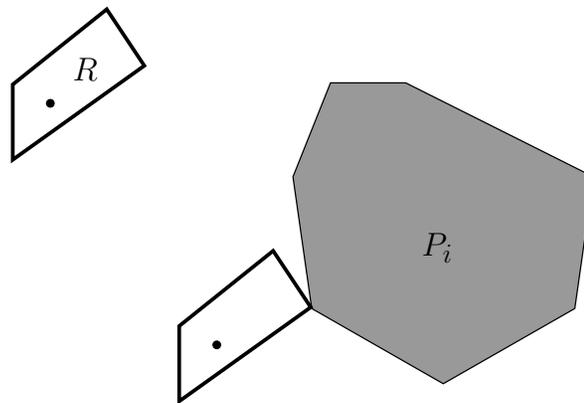
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



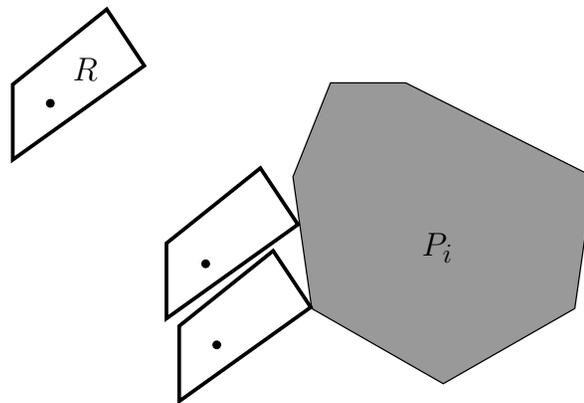
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



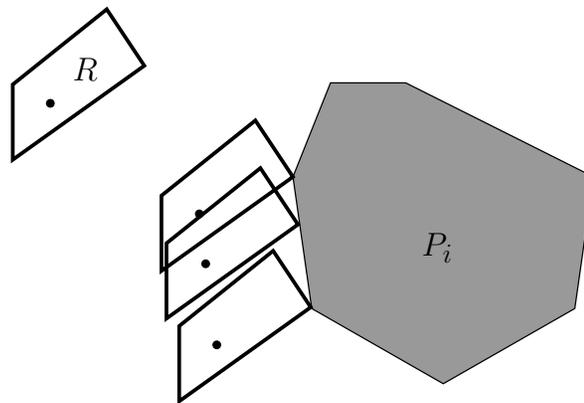
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



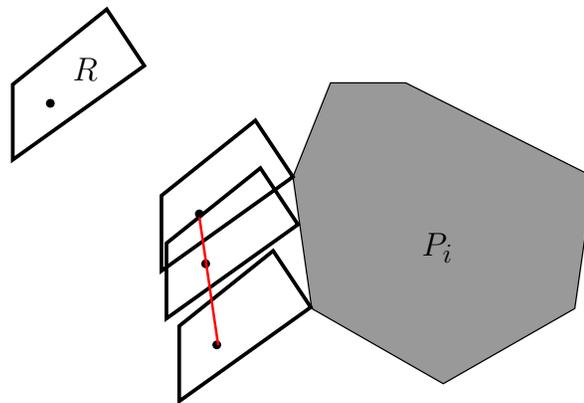
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



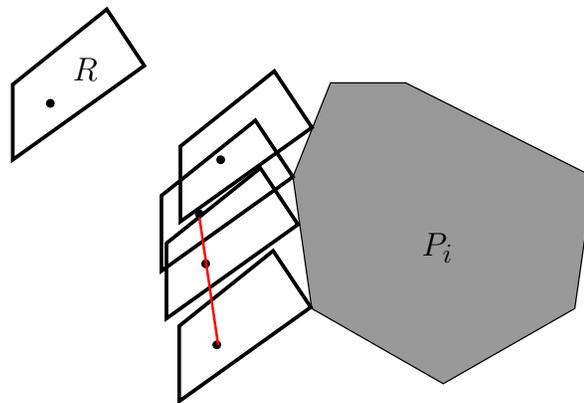
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



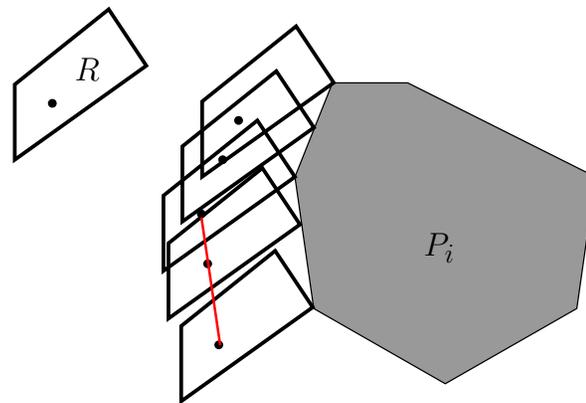
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



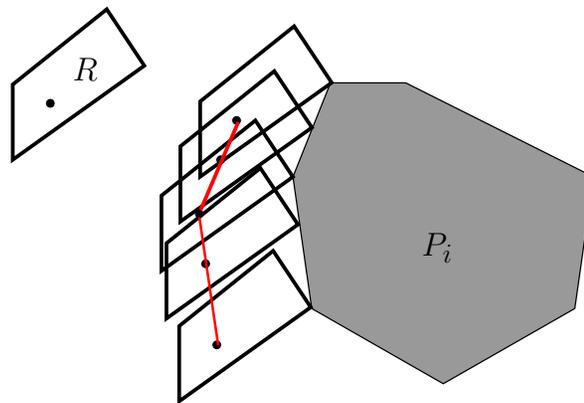
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



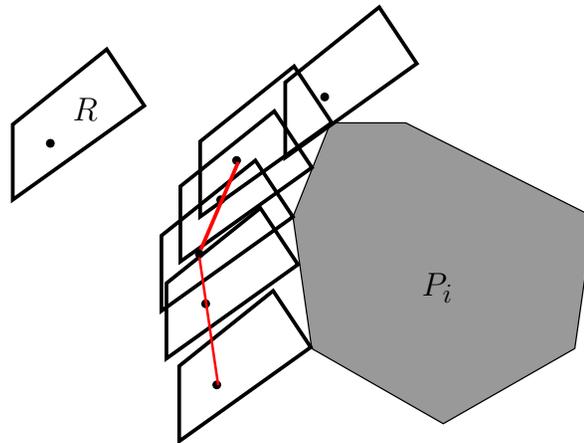
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



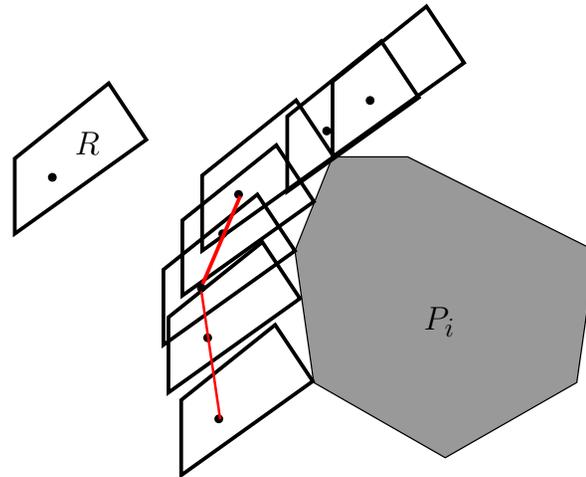
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



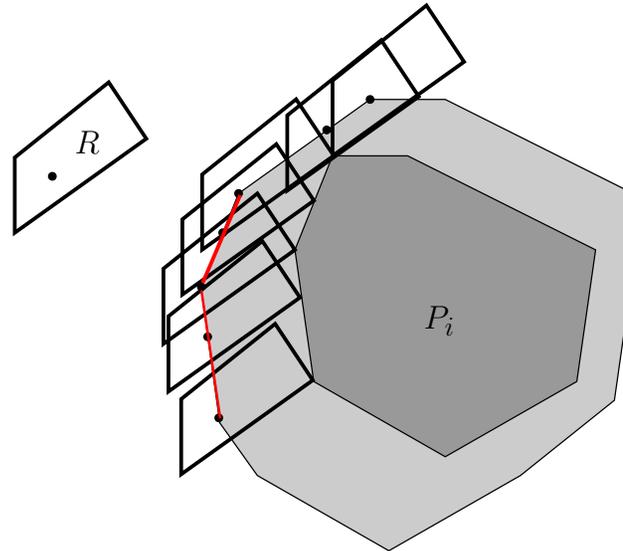
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i



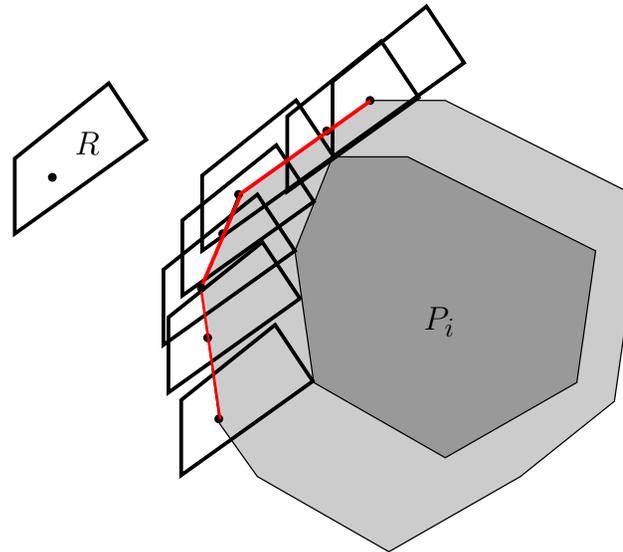
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



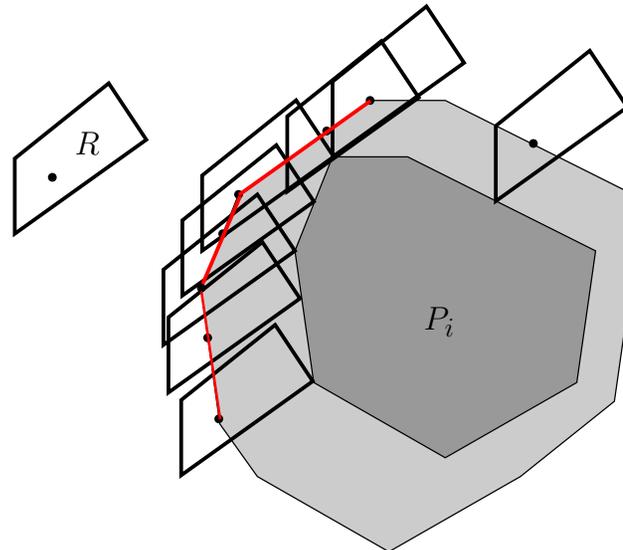
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



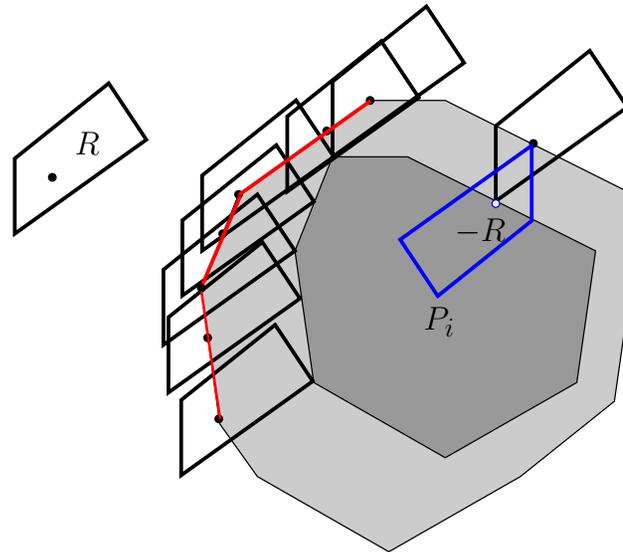
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



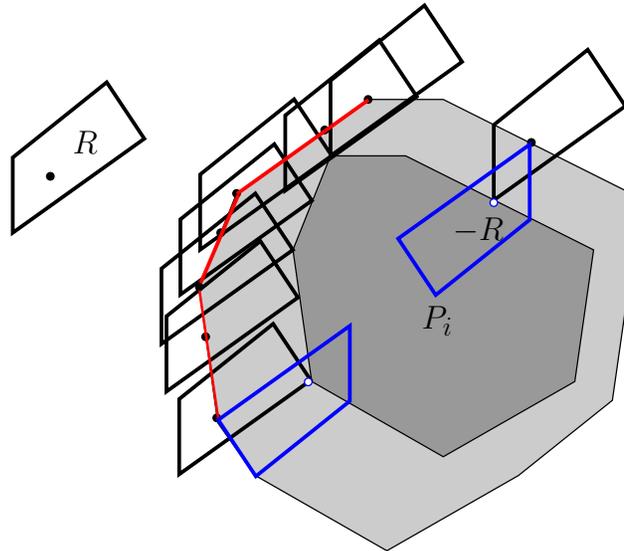
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



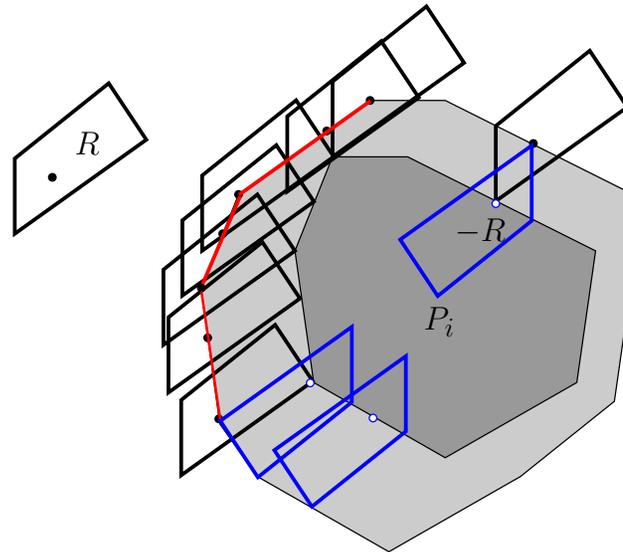
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



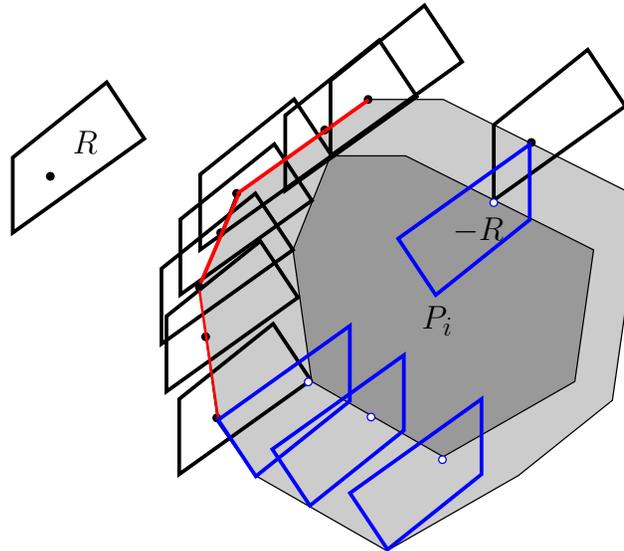
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



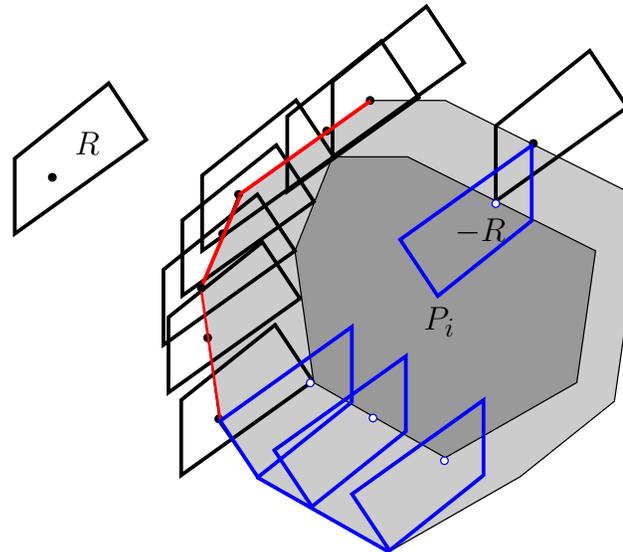
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



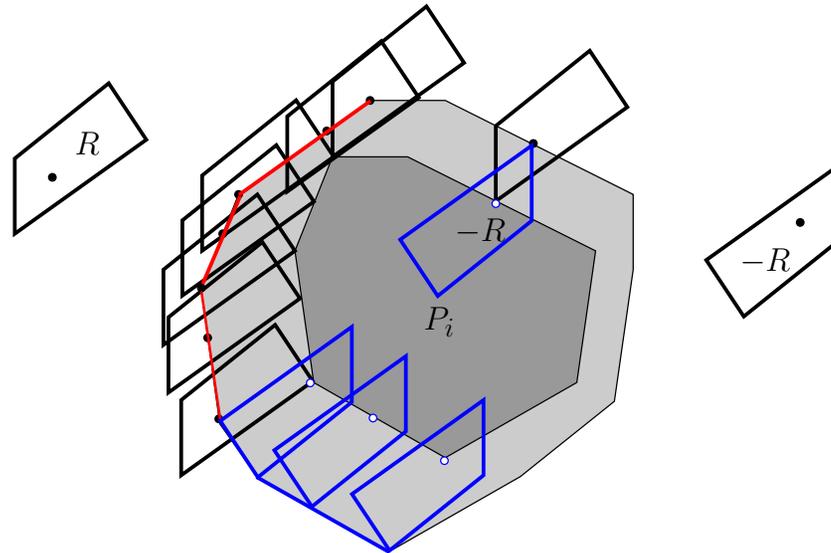
Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



Reine Translationsbewegung, konvexe Roboter 2.1.2

- Konvexer Roboter R , Hindernisse P_1, P_2, \dots, P_k
- Berechne $C_{\text{verb}} := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap \bigcup \overset{\circ}{P}_i \neq \emptyset \right\}$
- Bsp: Für konvexes Polygon P_i
- Konfigurationsraumhindernis
 $CP_i := \left\{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \right\}$



Minkowski-Summe **Def. 2.4**

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.
- **Bem. 2.4:**

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

- **Bem. 2.4:**

i) kommutativ: $A \oplus B = B \oplus A$

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

- **Bem. 2.4:**

i) kommutativ: $A \oplus B = B \oplus A$

ii) assoziativ: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

- **Bem. 2.4:**

i) kommutativ: $A \oplus B = B \oplus A$

ii) assoziativ: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

iii) distributiv bzgl. der Vereinigung:

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$$

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

- **Bem. 2.4:**

i) kommutativ: $A \oplus B = B \oplus A$

ii) assoziativ: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

iii) distributiv bzgl. der Vereinigung:

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$$

Anschaulich!!!

Minkowski-Summe **Def. 2.4**

- $A, B \subset \mathbb{R}^2$: $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

- **Bem. 2.4:**

i) kommutativ: $A \oplus B = B \oplus A$

ii) assoziativ: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

iii) distributiv bzgl. der Vereinigung:

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$$

Anschaulich!!! [MinkSum.html](#)

Lemma 2.6

Lemma 2.6

$$CP_i := \{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \}$$

Lemma 2.6

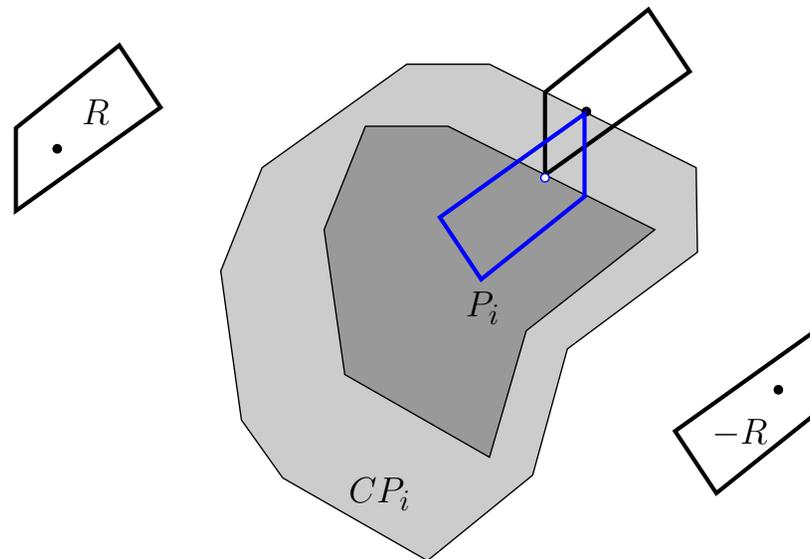
$$CP_i := \{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \}$$

$$CP_i = P_i \oplus (-R(0, 0))$$

Lemma 2.6

$$CP_i := \{ (x, y) \in \mathbb{R}^2 \mid R(x, y) \cap P_i \neq \emptyset \}$$

$$CP_i = P_i \oplus (-R(0, 0))$$



Beweis: Tafel!!!