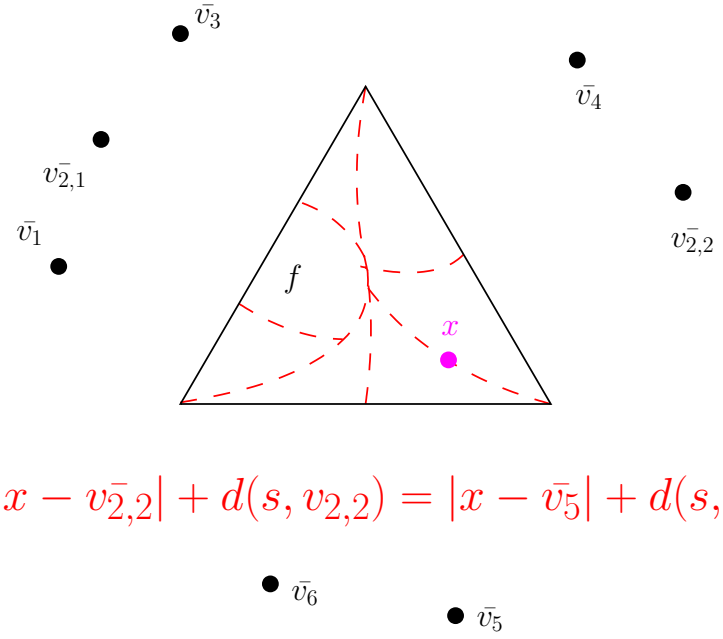
Offline Bewegungsplanung: Kürzeste Wege Polyeder, Objekte bewegen

Elmar Langetepe
University of Bonn

Aufteilen der Dreiecke!

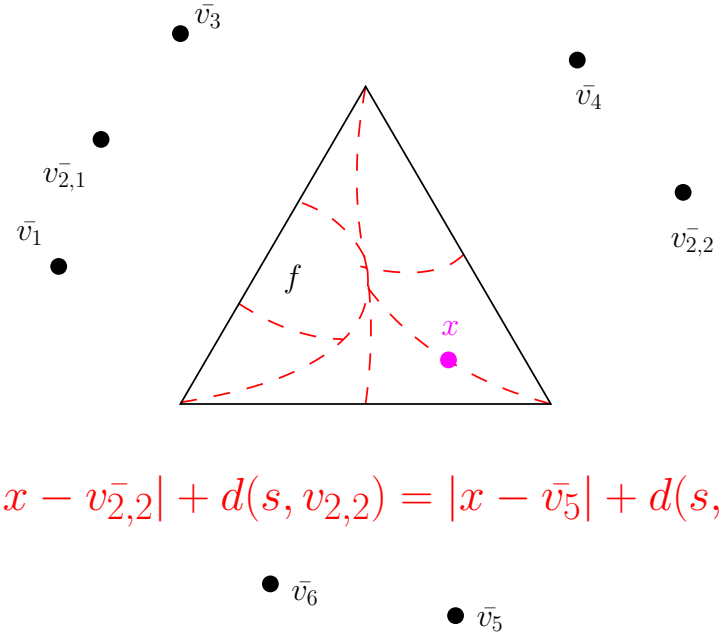
Aufteilen der Dreiecke!



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

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

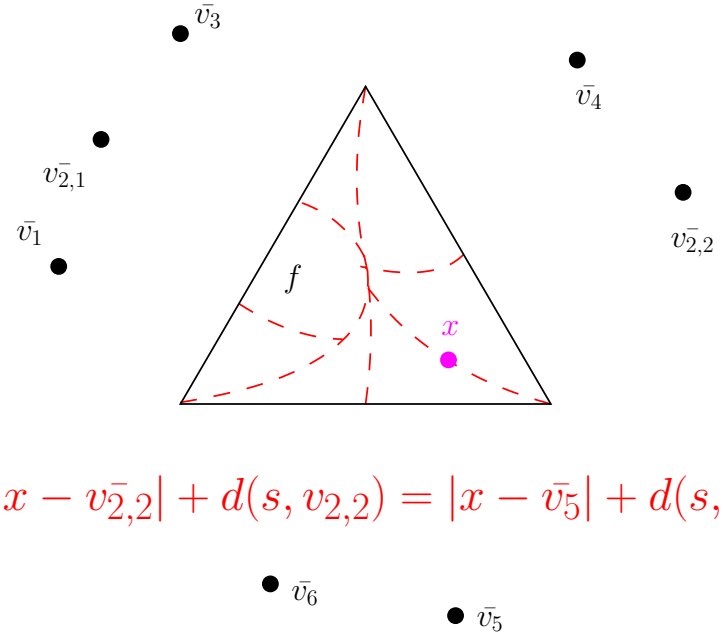
Aufteilen der Dreiecke!



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

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

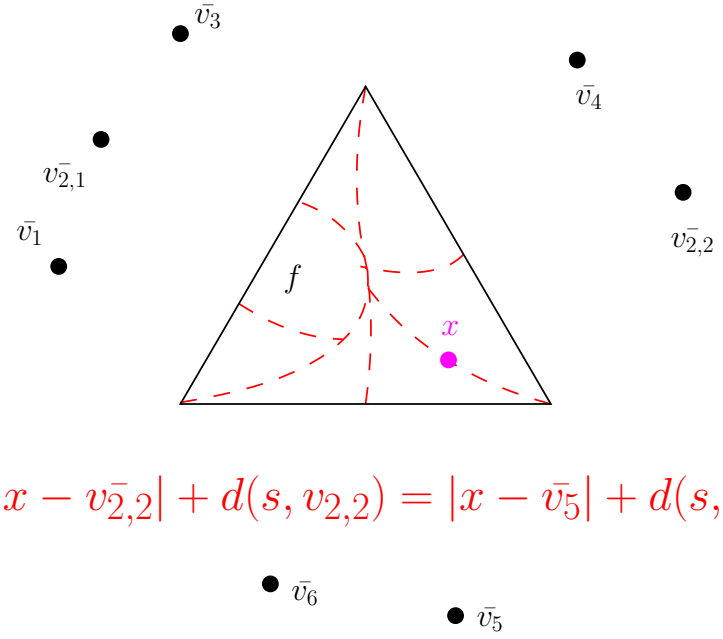
Aufteilen der Dreiecke!



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

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

Aufteilen der Dreiecke!



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

- 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 $I(v, \mathsf{E})$,

Alg. 1.13

- Input: Triang. Polyder P, n Ecken, s
- Output: Kürzester Weg für beliebigen Anfragepunkt $t \in f$
- Preprocessing:
  - Berechne Intervalle $I(v, \mathsf{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathsf{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathsf{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$: 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$: 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$: 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 $I(v, \mathrm{E})$, $O(n^2 \log n)$
  - 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 $I(v, \mathrm{E})$: 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 $I(v, \mathbf{E})$

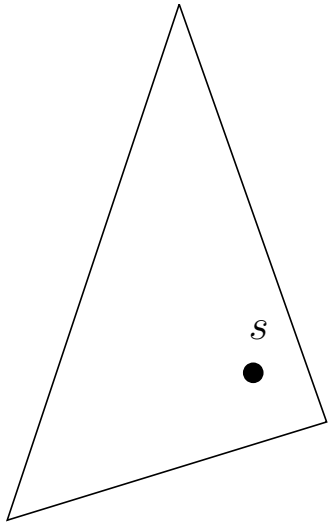
Continous Dijkstra: Sukzessive (Teil)Intervalle festlegen!

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

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

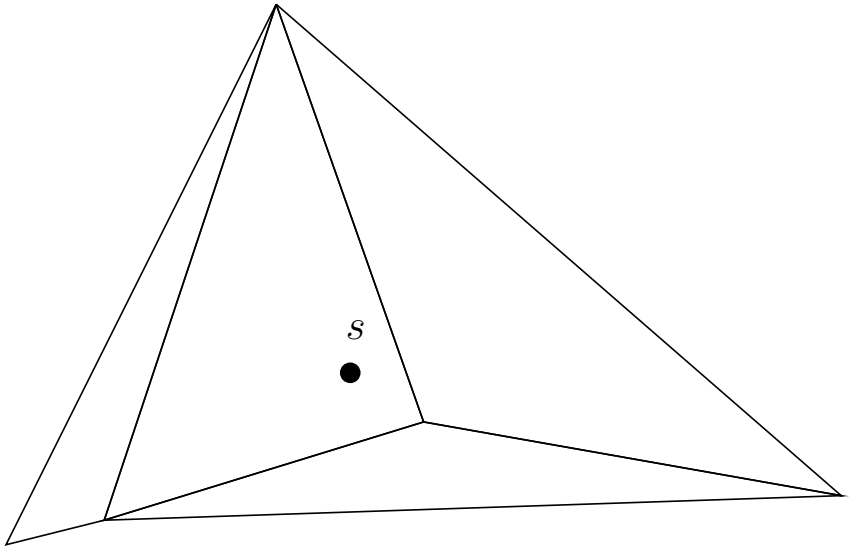
Berechnung aller $I(v, \mathbf{E})$

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



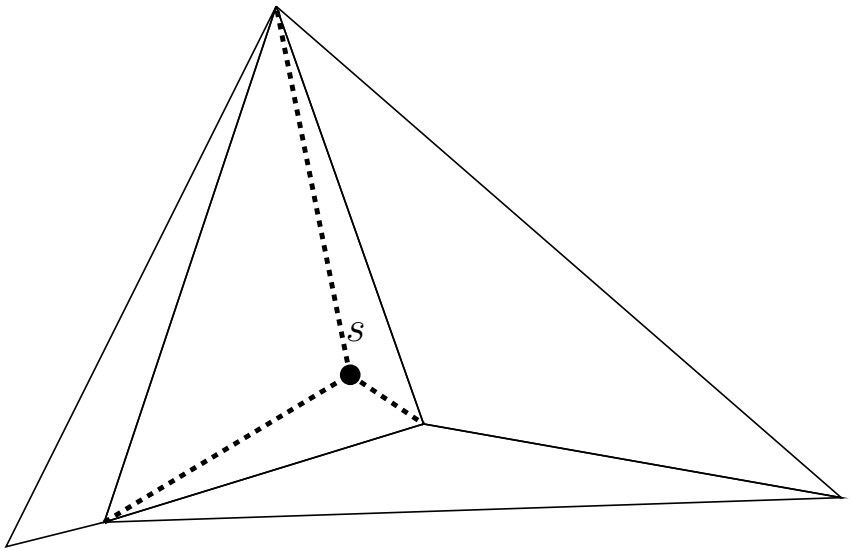
Berechnung aller $I(v, \mathbf{E})$

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



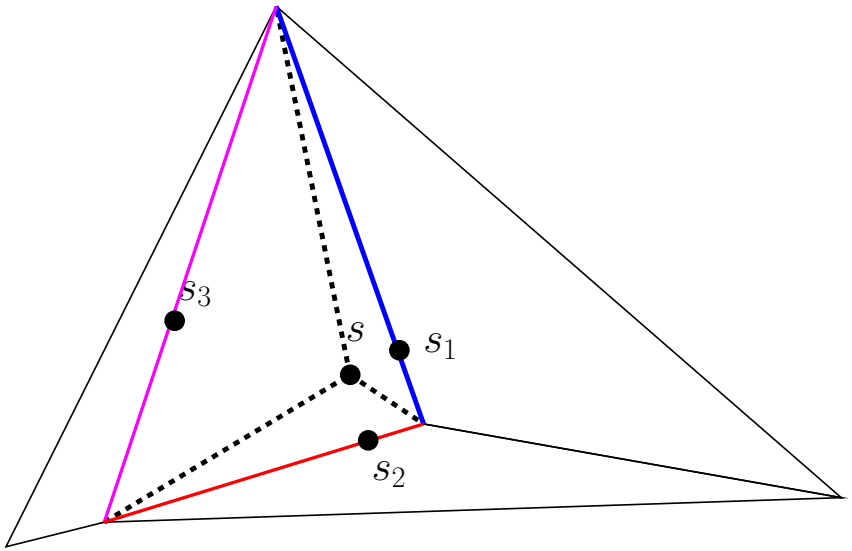
Berechnung aller $I(v, \mathbf{E})$

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



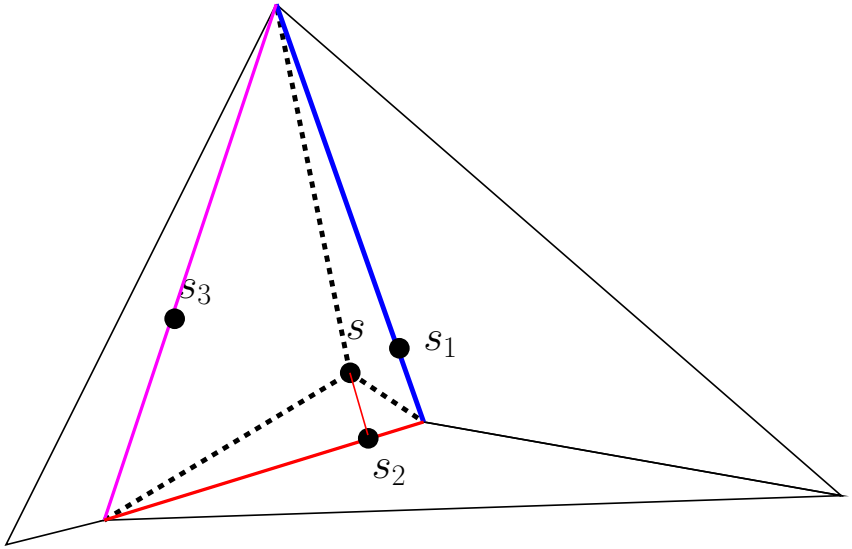
Berechnung aller $I(v, \mathbf{E})$

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



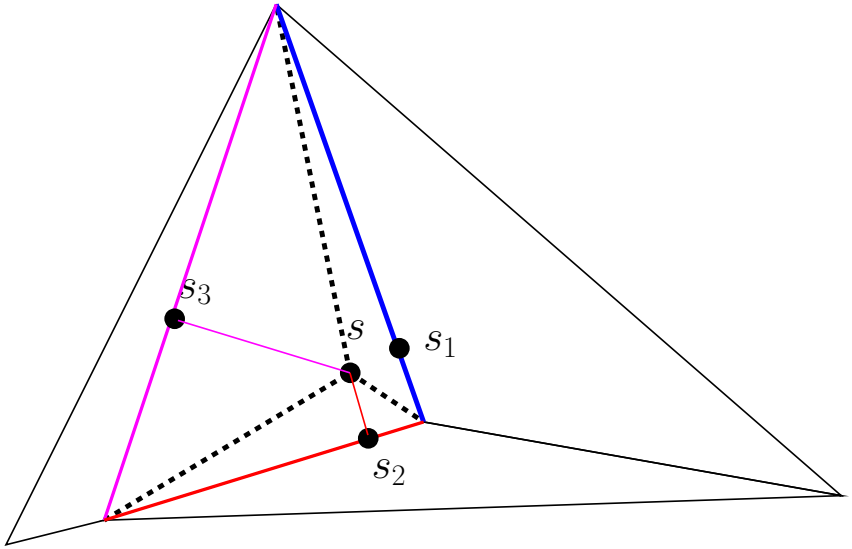
Berechnung aller $I(v, \mathbf{E})$

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



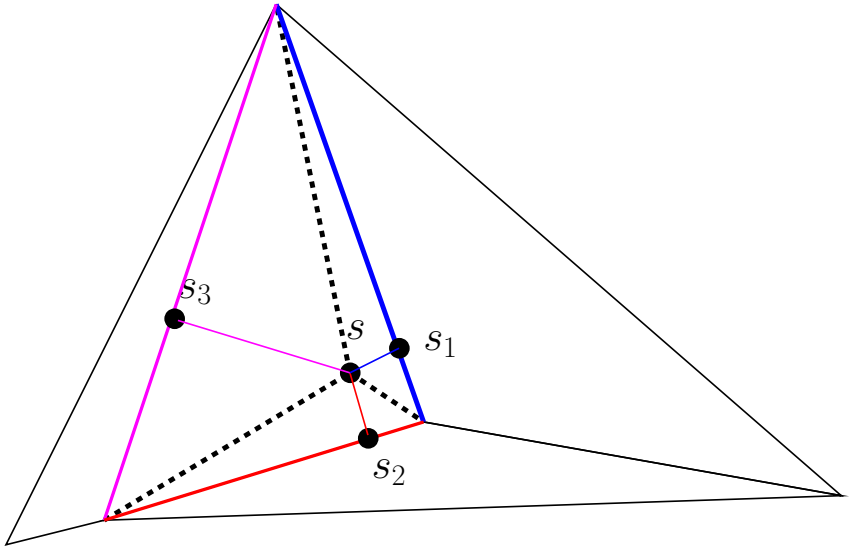
Berechnung aller $I(v, \mathbf{E})$

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



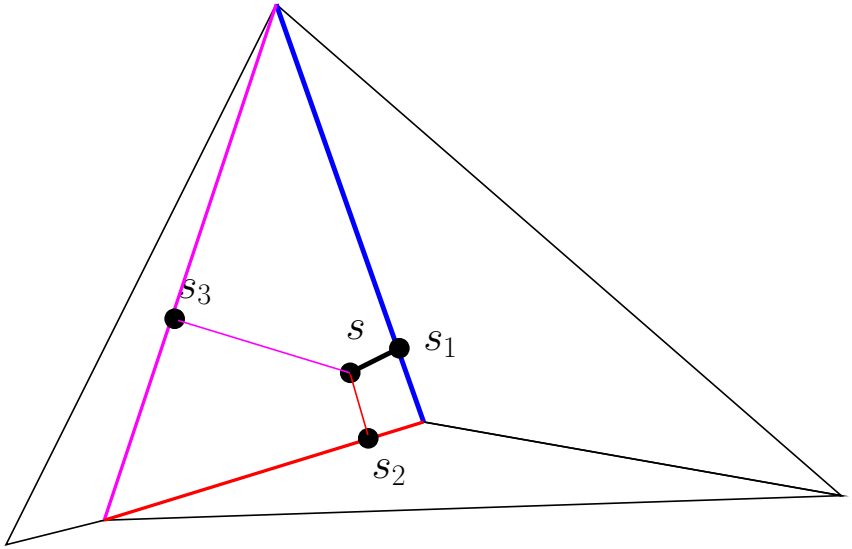
Berechnung aller $I(v, \mathbf{E})$

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



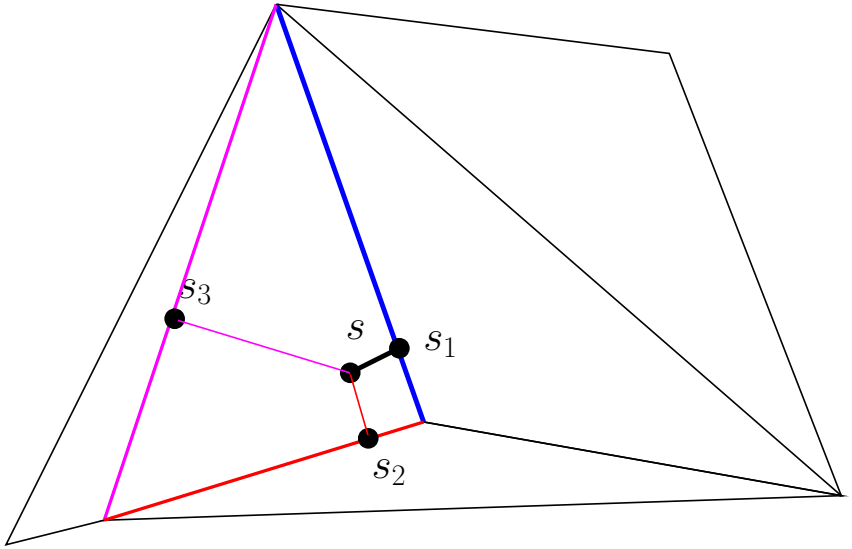
Berechnung aller $I(v, \mathbf{E})$

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



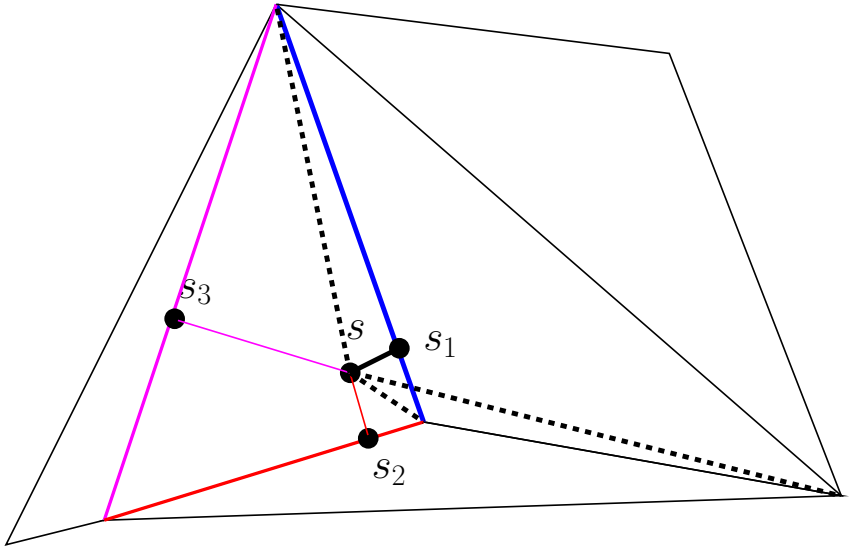
Berechnung aller $I(v, \mathbf{E})$

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



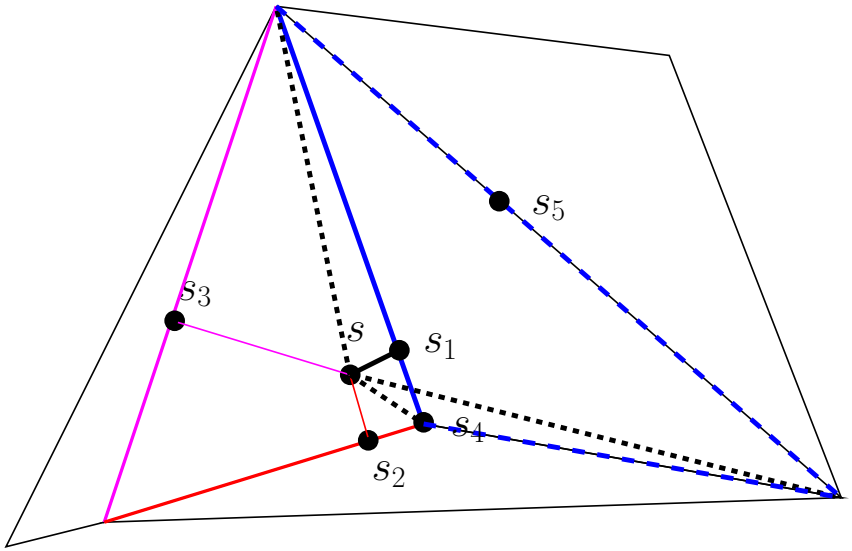
Berechnung aller $I(v, \mathbf{E})$

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



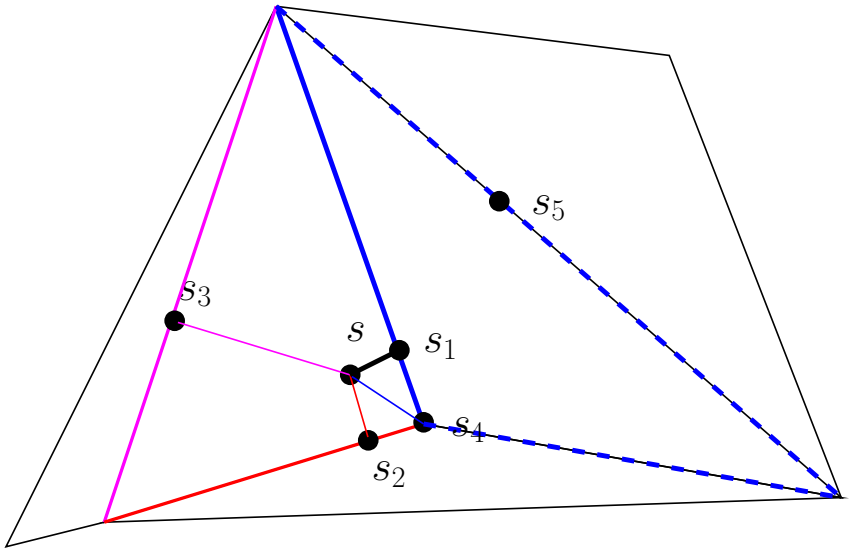
Berechnung aller $I(v, \mathbf{E})$

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



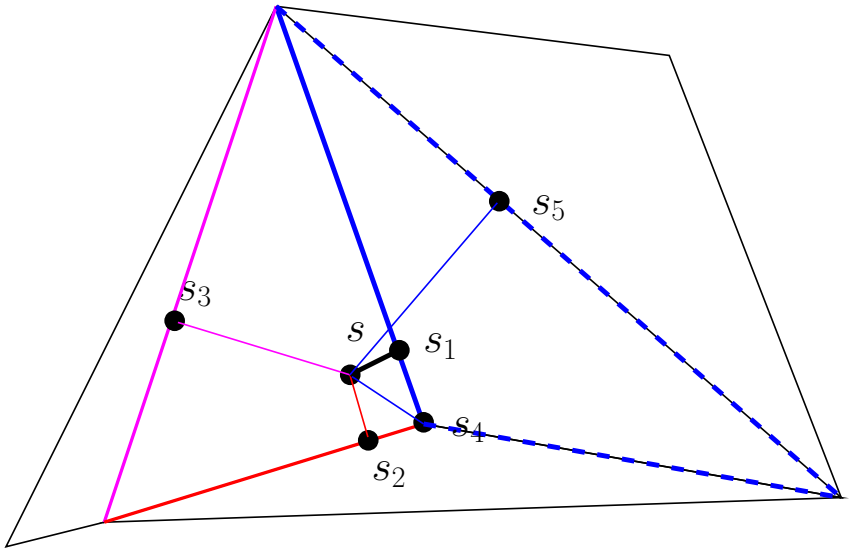
Berechnung aller $I(v, \mathbf{E})$

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



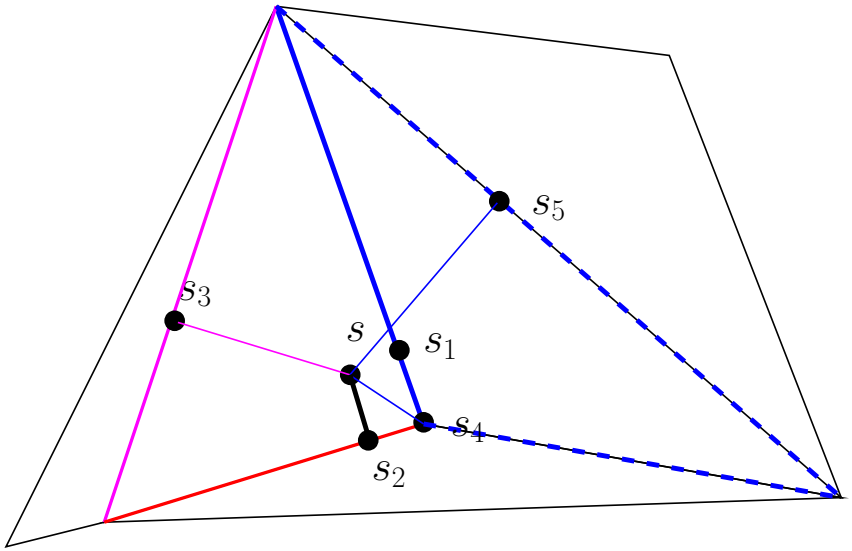
Berechnung aller $I(v, \mathbf{E})$

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



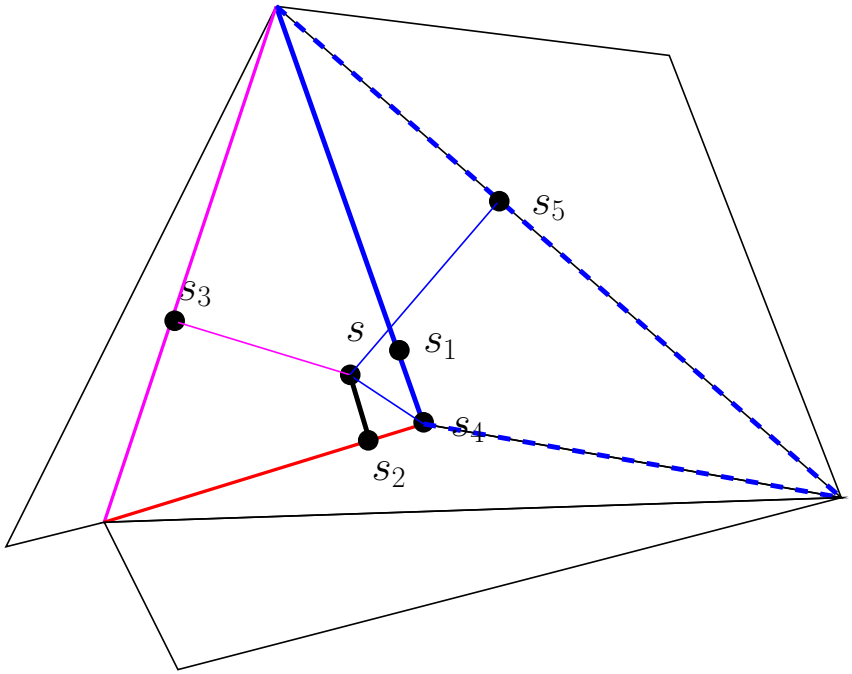
Berechnung aller $I(v, \mathbf{E})$

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



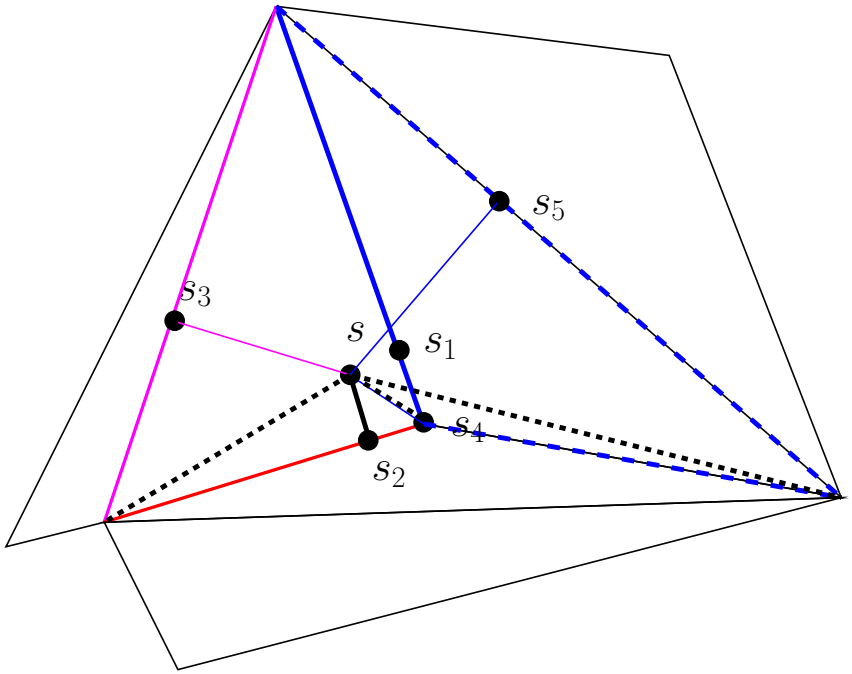
Berechnung aller $I(v, \mathbf{E})$

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



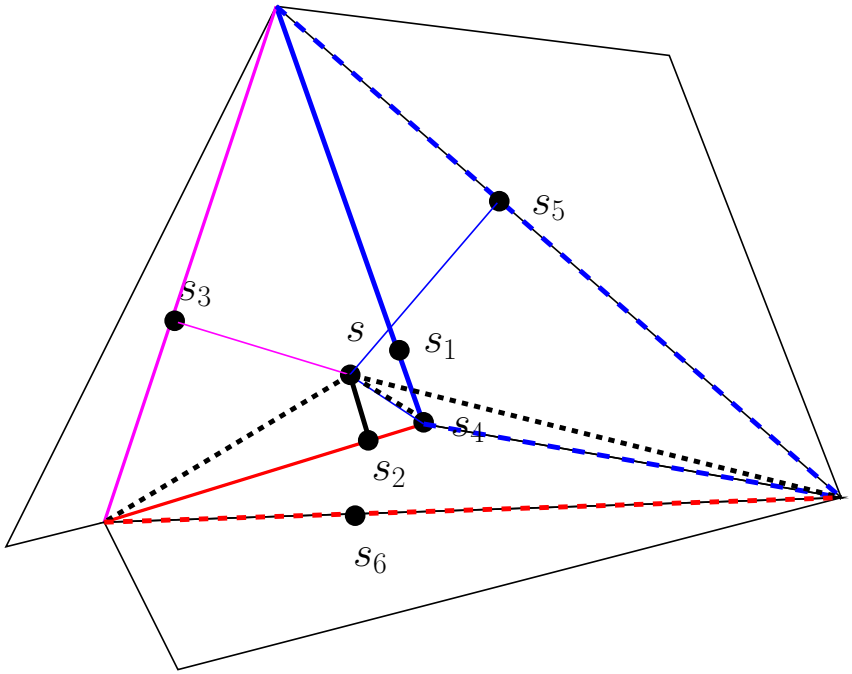
Berechnung aller $I(v, \mathbf{E})$

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



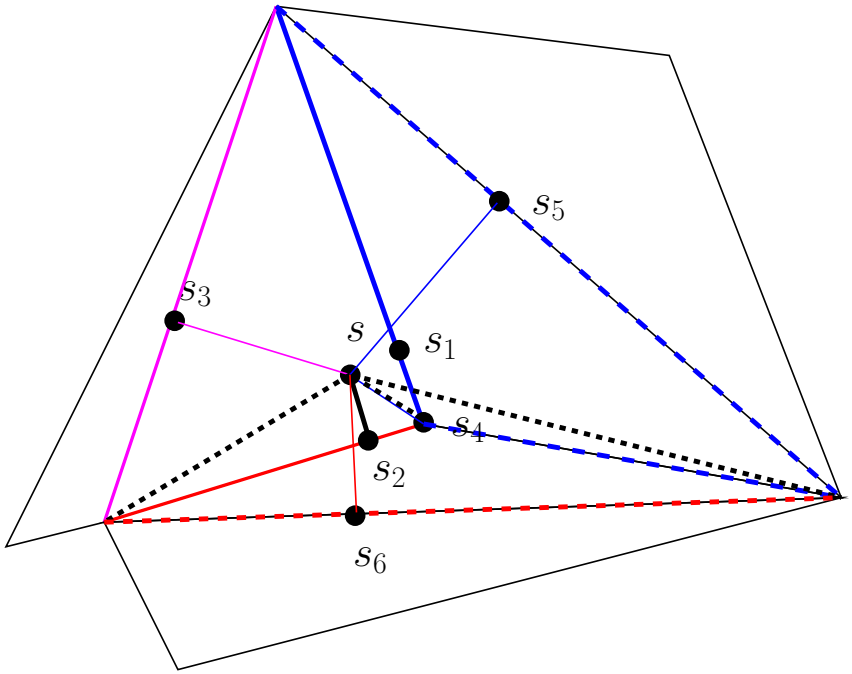
Berechnung aller $I(v, \mathbf{E})$

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



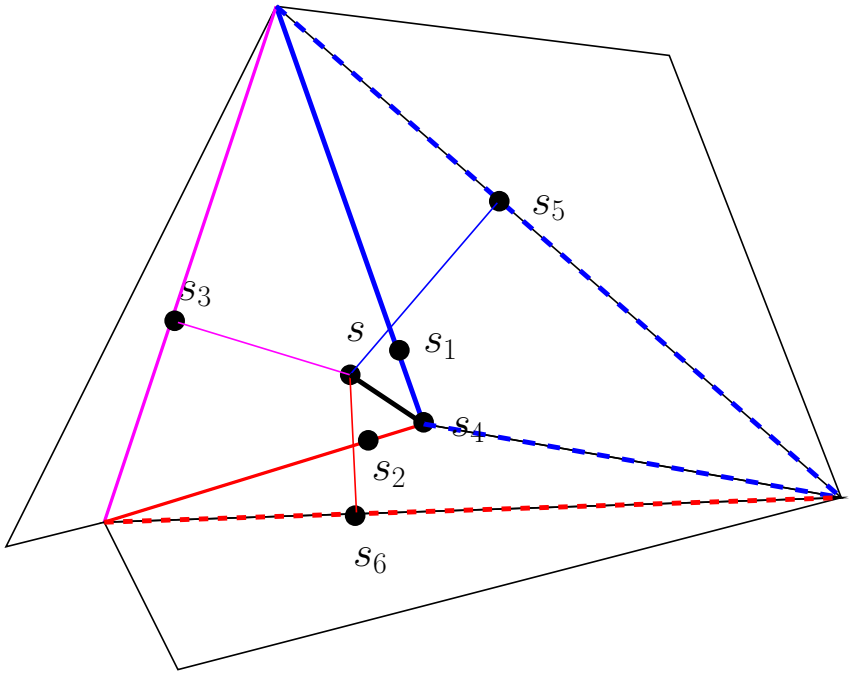
Berechnung aller $I(v, \mathbf{E})$

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



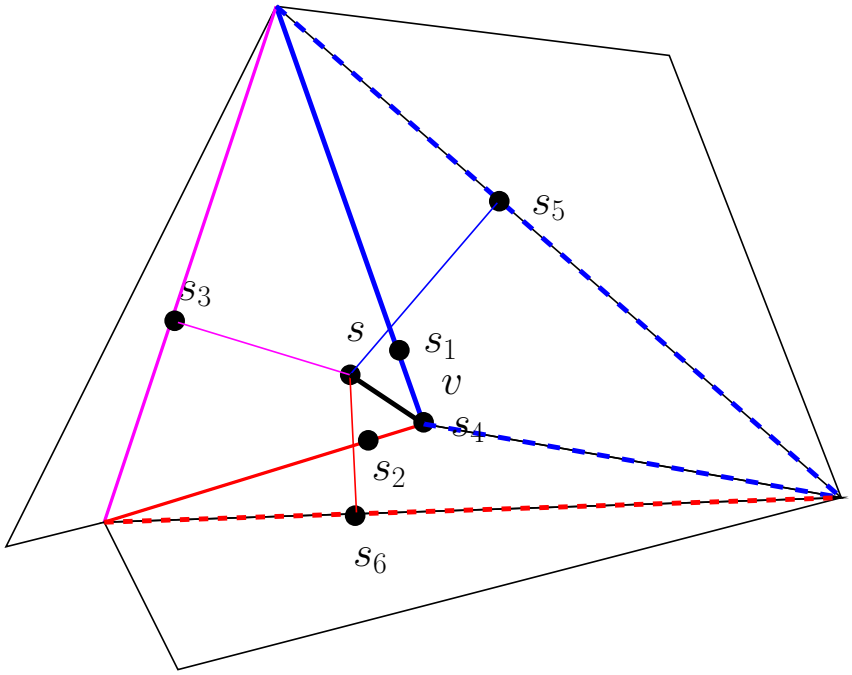
Berechnung aller $I(v, \mathbf{E})$

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



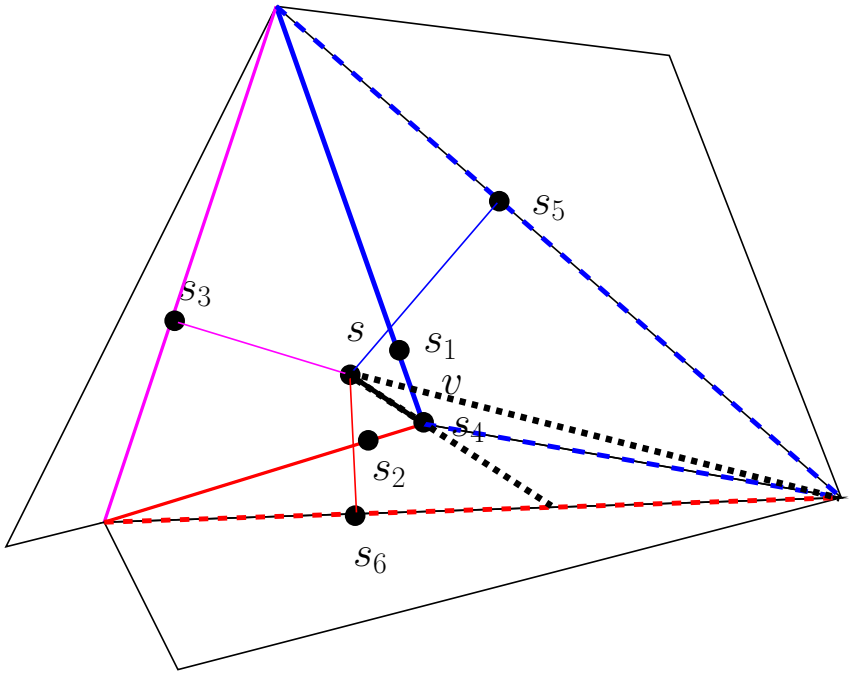
Berechnung aller $I(v, \mathbf{E})$

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



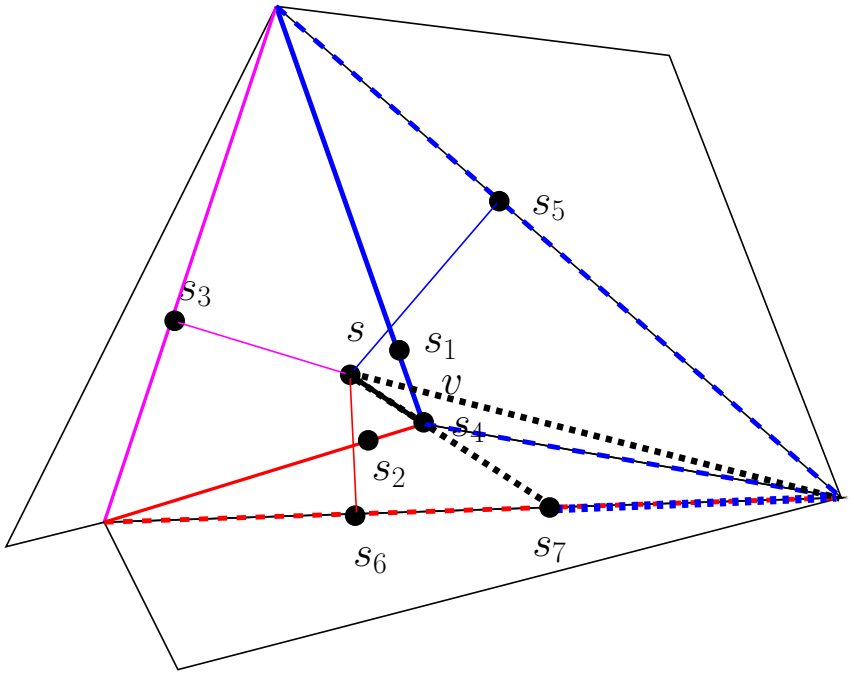
Berechnung aller $I(v, \mathbf{E})$

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



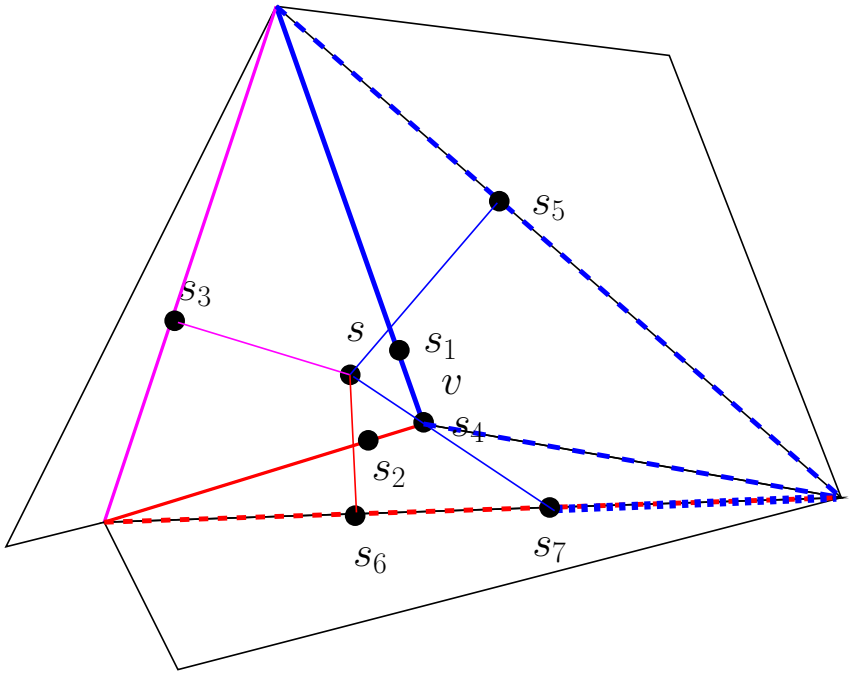
Berechnung aller $I(v, \mathbf{E})$

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



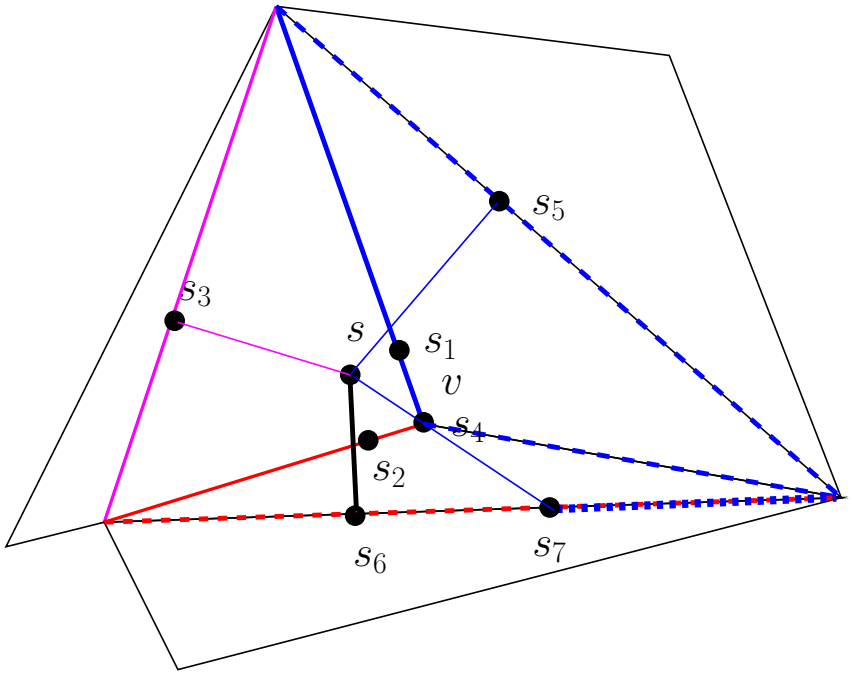
Berechnung aller $I(v, \mathbf{E})$

Continous 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 $I(v, \mathbf{E})$

Continous 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

Alg. 1.12 Continuous Dijkstra

- **DS:**

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 $I(v, \mathrm{E})$ 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 $I(v, \mathrm{E})$ 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 $I(v, \mathrm{E})$ 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 $I(v, \mathrm{E})$ 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.

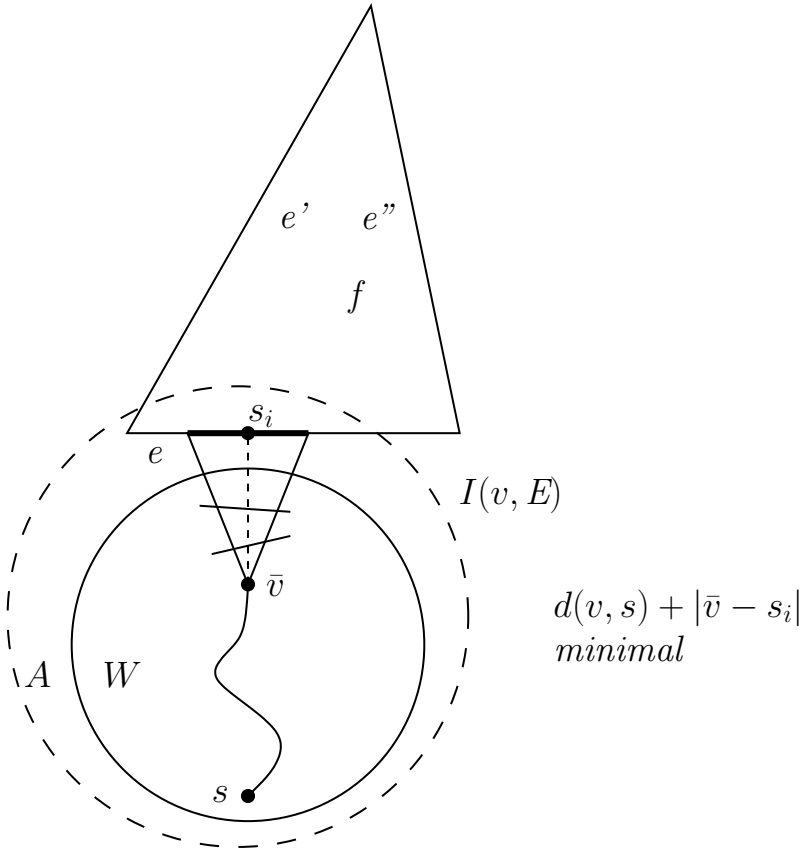
Alg. 1.12 Continuous Dijkstra

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

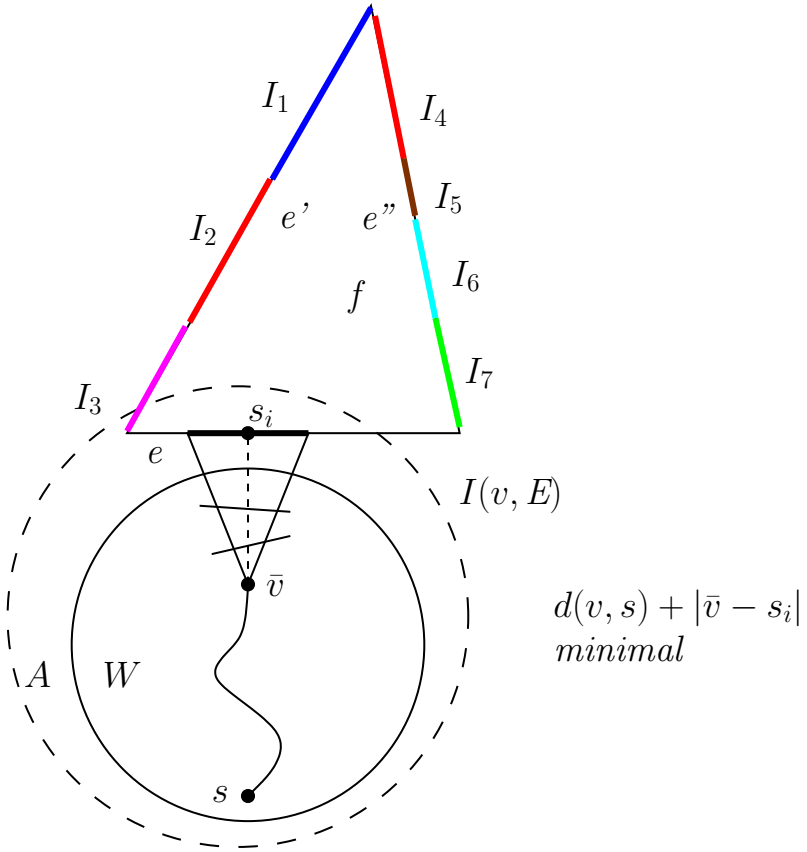
- **Iterationsschritt:**
  - Wähle Intervall $I(v, \mathrm{E})$ mit kürzestem Abstand. $d(s, v) + |\bar{v} - s_i|$ ist am geringsten! Wird stets Intervall bleiben! (Eventuell später gekürzt).
  - Führe dieses Intervall auf die Kanten des Dreiecks fort, auf dem s_i lag und das jenseits der Folge E liegt.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

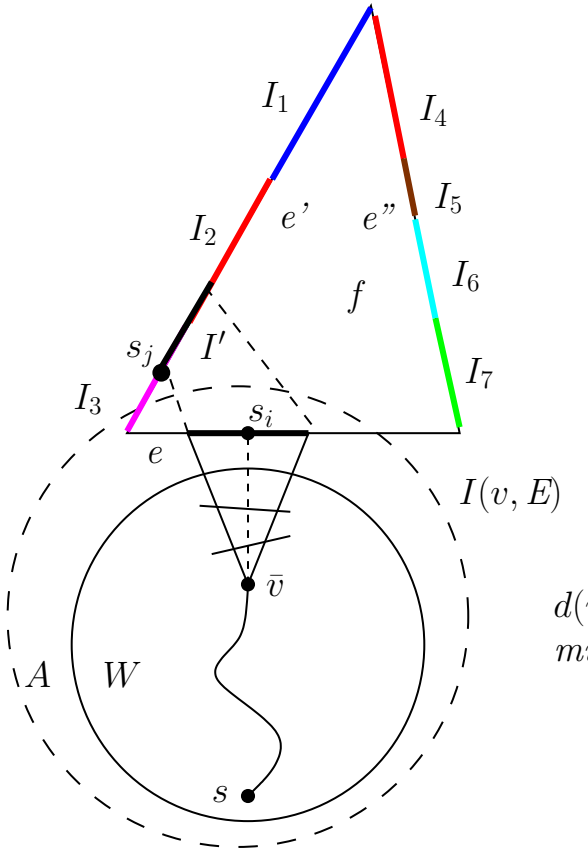
Schritt des Einsortierens!



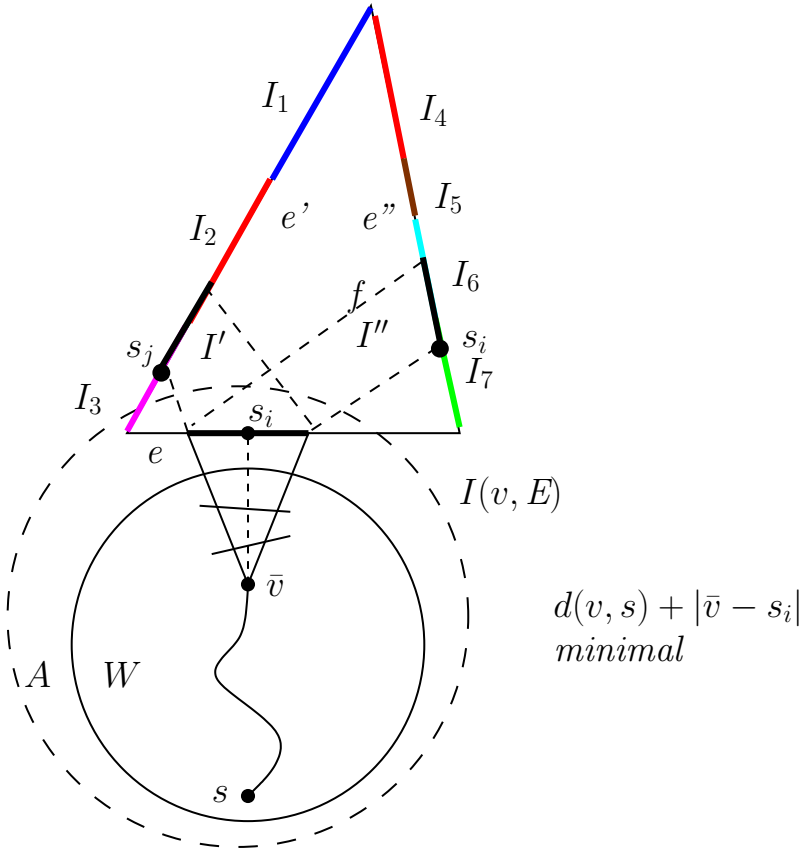
Schritt des Einsortierens!



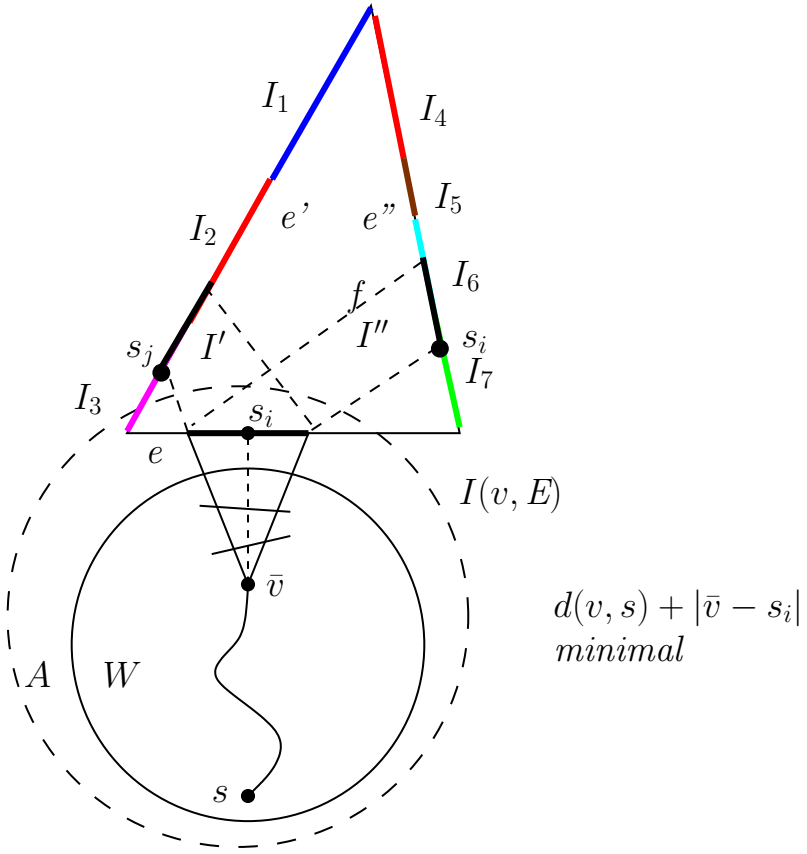
Schritt des Einsortierens!



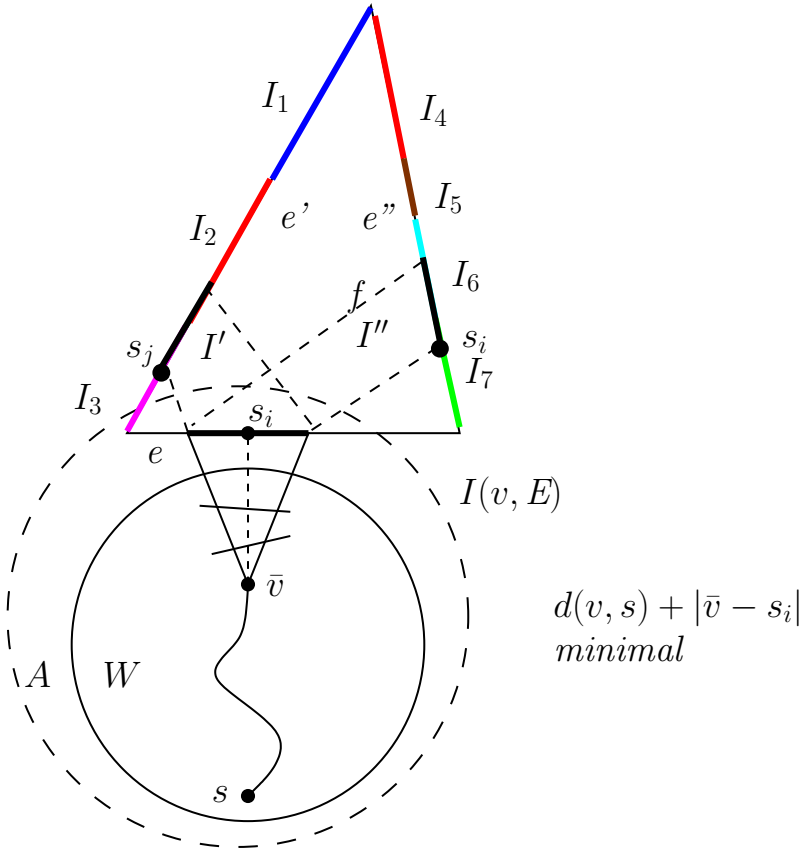
Schritt des Einsortierens!



Schritt des Einsortierens!

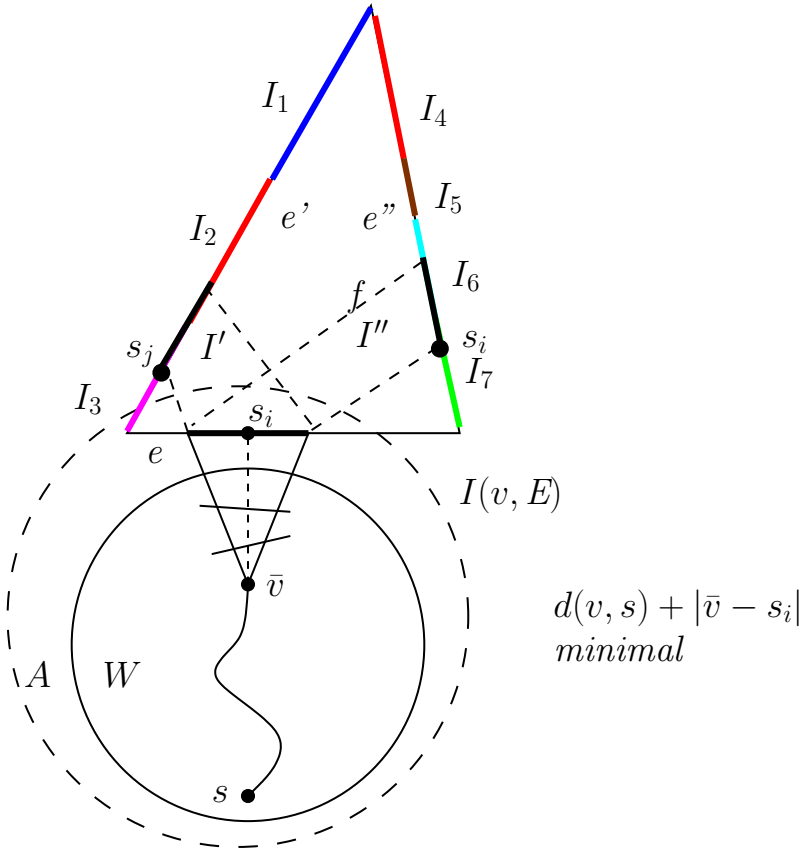


Schritt des Einsortierens!



Einsortieren in Baum der Intervalle!

Schritt des Einsortierens!



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

Liniensegmente statt Punkte

Liniensegmente statt Punkte

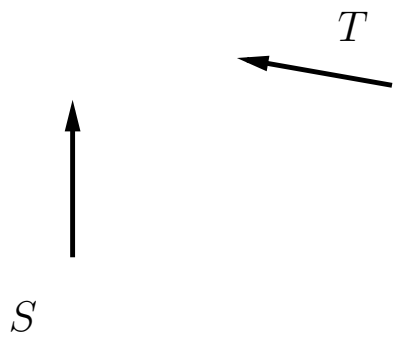
- Liniensegment in der Ebene

Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S

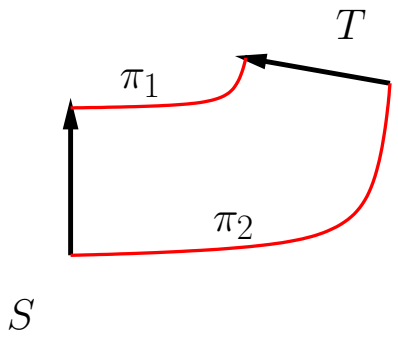
Liniensegmente statt Punkte

- Liniensegment in der Ebene
- Startlage S und Ziellage T



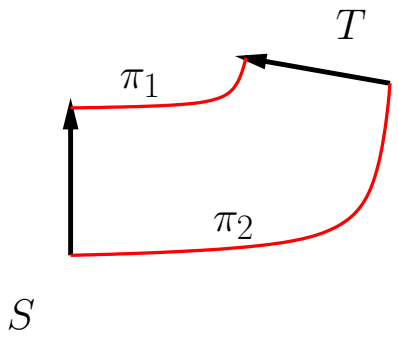
Liniensegmente statt Punkte

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



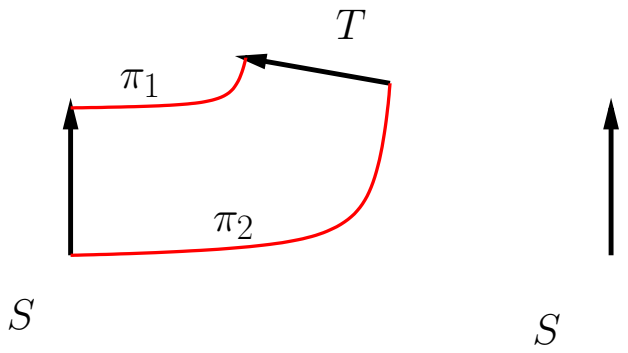
Liniensegmente statt Punkte

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



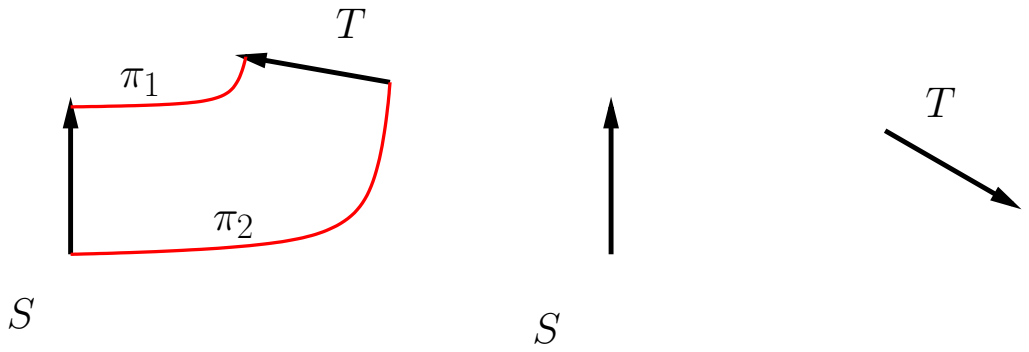
Liniensegmente statt Punkte

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



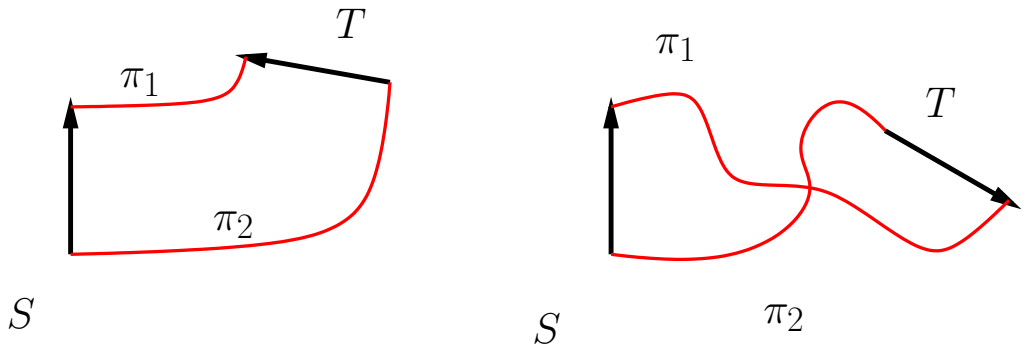
Liniensegmente statt Punkte

- 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!



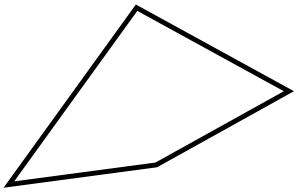
Was ist eine kurze Bewegung?

Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig

Was ist eine kurze Bewegung?

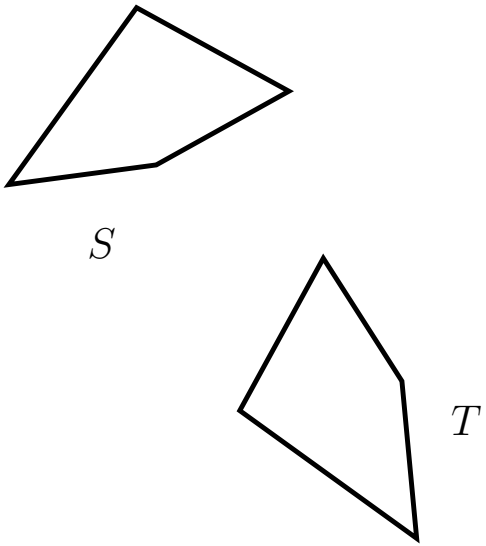
- Bei ausgedehnten Objekten schwierig



S

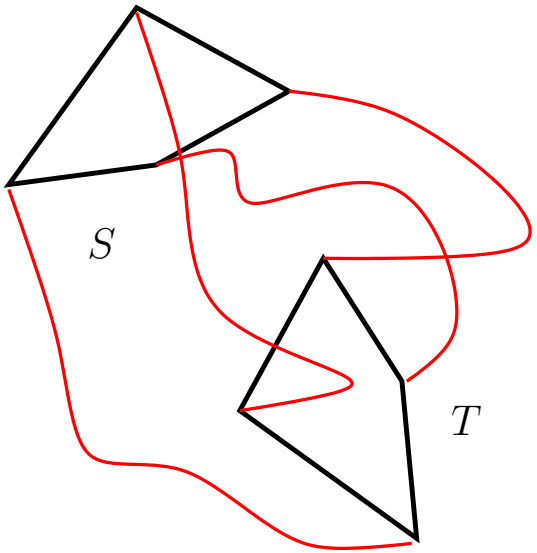
Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig



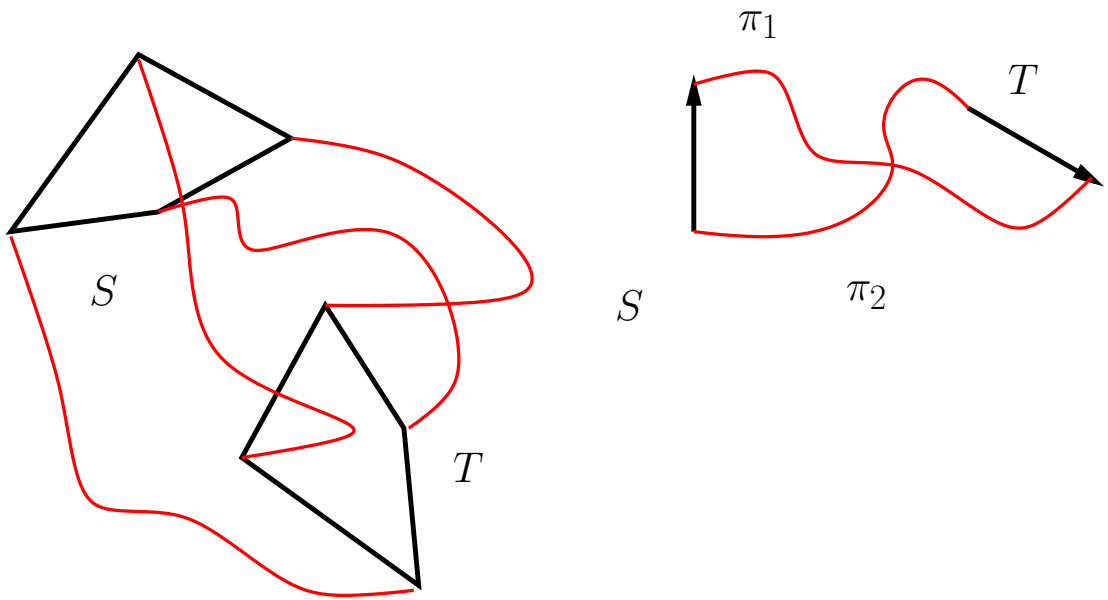
Was ist eine kurze Bewegung?

- Bei ausgedehnten Objekten schwierig



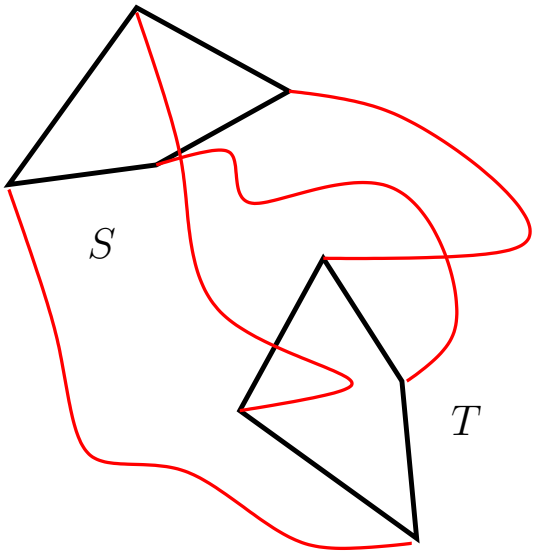
Was ist eine kurze Bewegung?

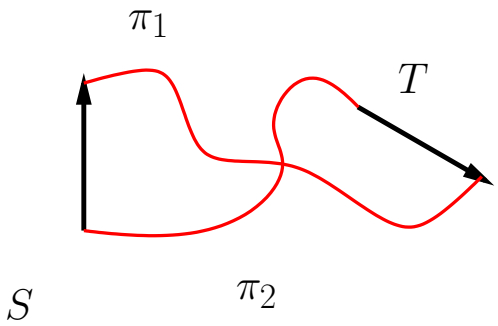
- Bei ausgedehnten Objekten schwierig
- Hier: Weglänge zweier(!) Träger: Summe



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)

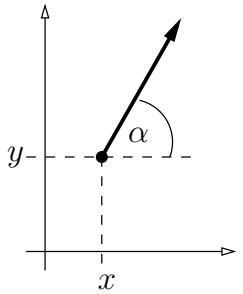
Formale Beschreibung!

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

(i)

Formale Beschreibung!

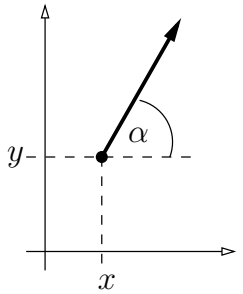
- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!



(i)

Formale Beschreibung!

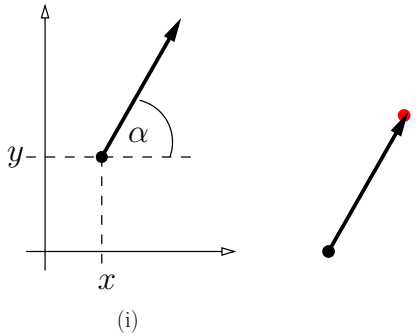
- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)



(i)

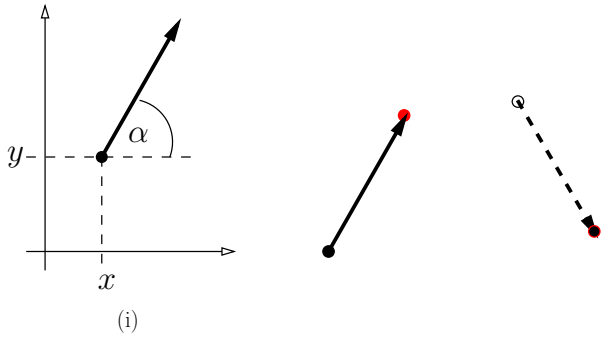
Formale Beschreibung!

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



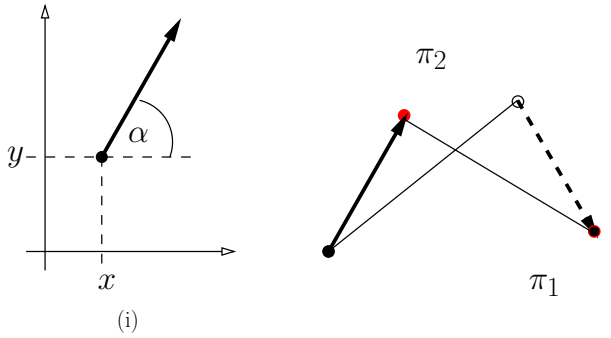
Formale Beschreibung!

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



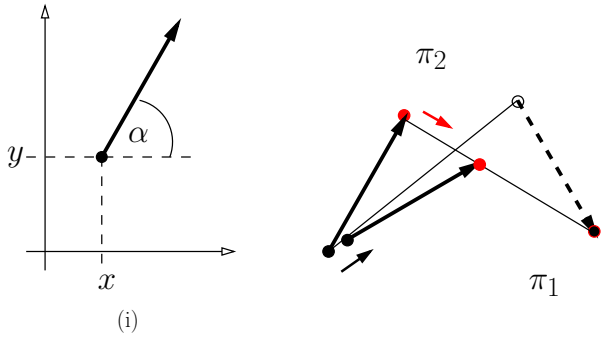
Formale Beschreibung!

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



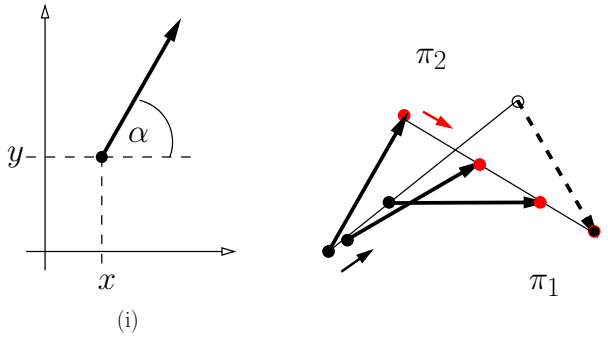
Formale Beschreibung!

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



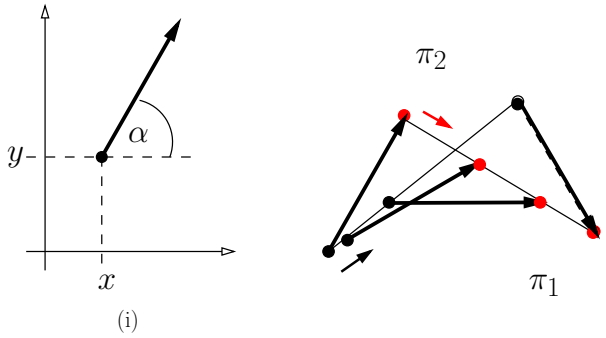
Formale Beschreibung!

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



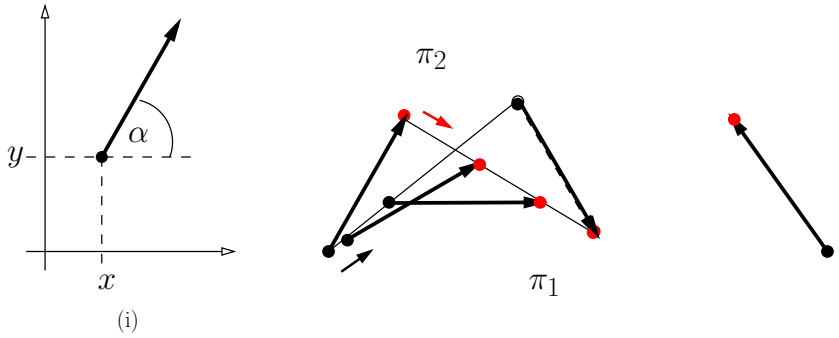
Formale Beschreibung!

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



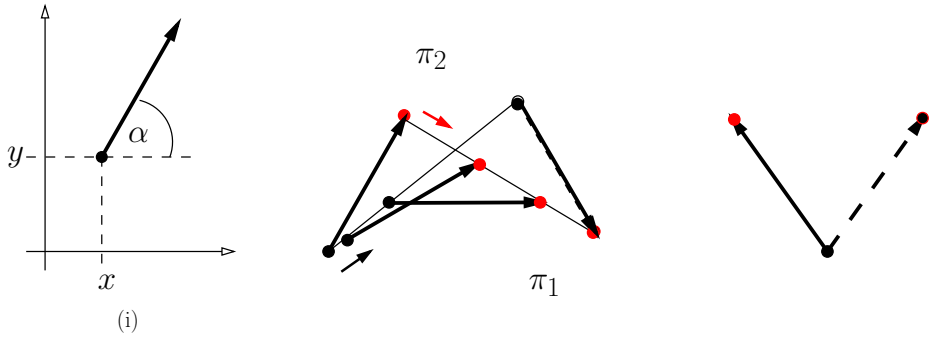
Formale Beschreibung!

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



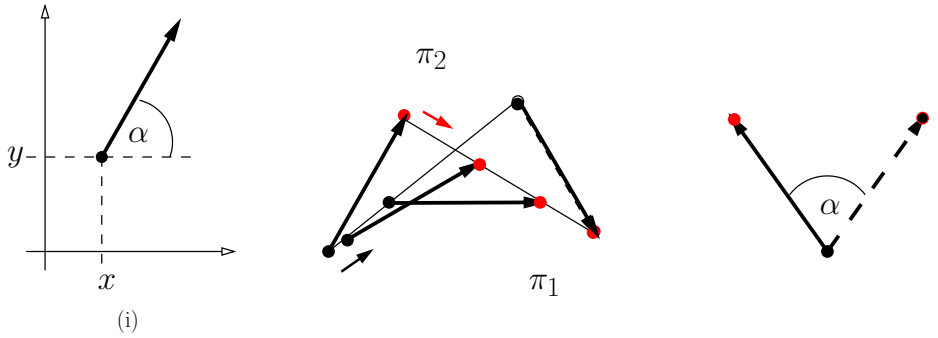
Formale Beschreibung!

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



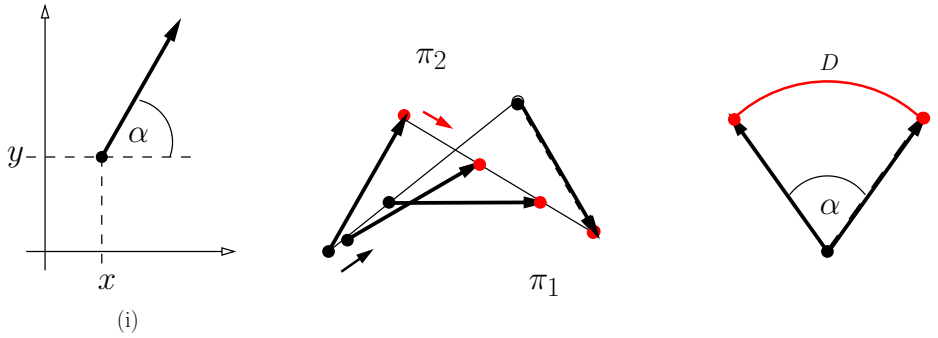
Formale Beschreibung!

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



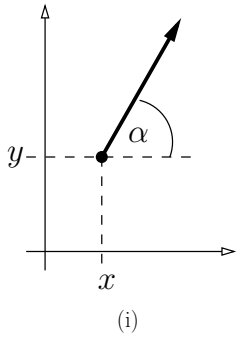
Formale Beschreibung!

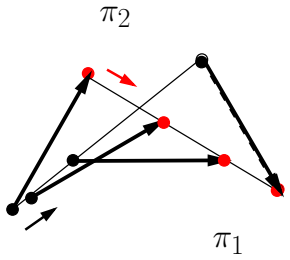
- Lage des Segmentes durch Tripel (x, y, α)
- Normiert!
- Optimale Bewegungen: Rotation/Translation(Klar!!)
- Translation geht nicht immer

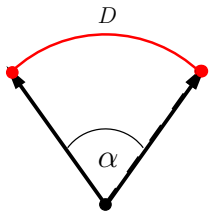


Formale Beschreibung!

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

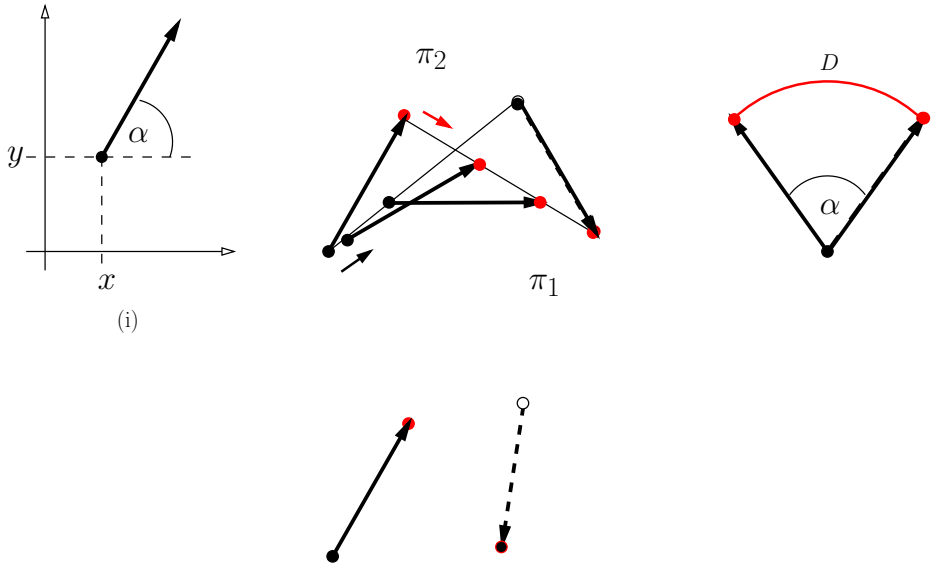






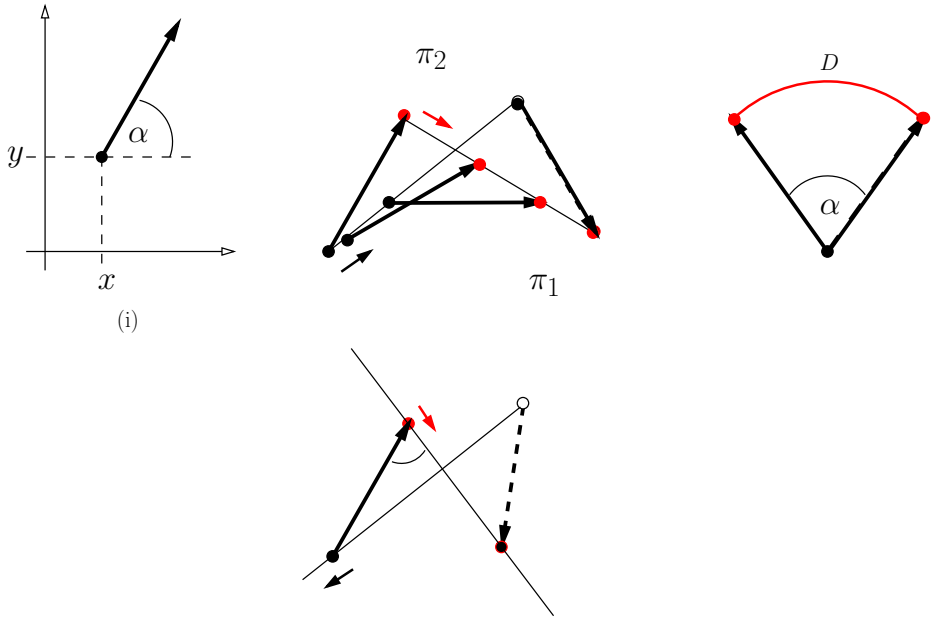
Formale Beschreibung!

- 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:

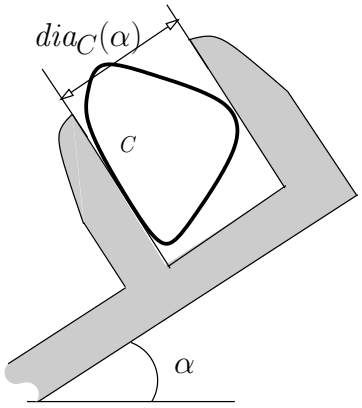
1. maximal drei Rotationen,
2. maximal zwei Rotationen und eine geradlinige Bewegung,
3. eine Rotation zwischen zwei geradlinigen Bewegungen.

Die Bewegungen lassen sich effizient berechnen.

Rotation? Hilfe: Surface-Area Th. 1.46!

Rotation? Hilfe: Surface-Area Th. 1.46!

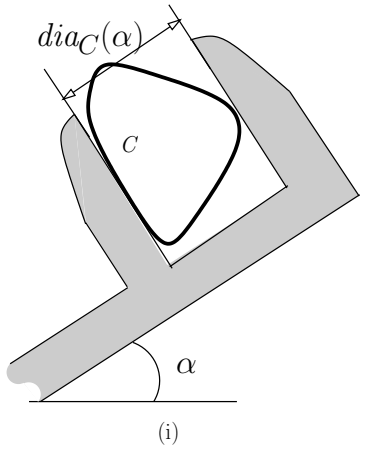
- Geschlossene konvexe Kurve C: Durchm. Funktion:
  $\mathtt{dia}_C : [0, 2\pi) \to \mathtt{R}$



(i)

Rotation? Hilfe: Surface-Area Th. 1.46!

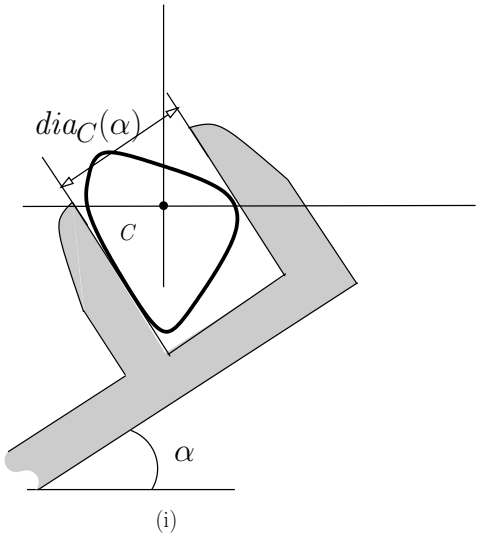
- Geschlossene konvexe Kurve C: Durchm. Funktion: $\mathtt{dia_C} : [0, 2\pi) \to \mathtt{R}$
- Support Funktion: $\mathtt{h_C}(\alpha) = \sup\{\mathtt{x}\cos\alpha + \mathtt{y}\sin\alpha | (\mathtt{x}, \mathtt{y}) \in \mathtt{C}\}$
- X-Koordinate nach Rotation mit α



(i)

Rotation? Hilfe: Surface-Area Th. 1.46!

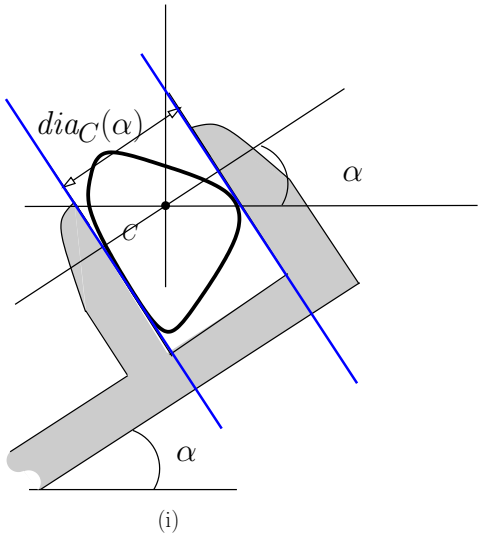
- Geschlossene konvexe Kurve C: Durchm. Funktion: $\mathtt{dia_C} : [0, 2\pi) \to \mathtt{R}$
- Support Funktion: $\mathtt{h_C}(\alpha) = \sup\{\mathtt{x}\cos\alpha + \mathtt{y}\sin\alpha | (\mathtt{x}, \mathtt{y}) \in \mathtt{C}\}$
- X-Koordinate nach Rotation mit α



(i)

Rotation? Hilfe: Surface-Area Th. 1.46!

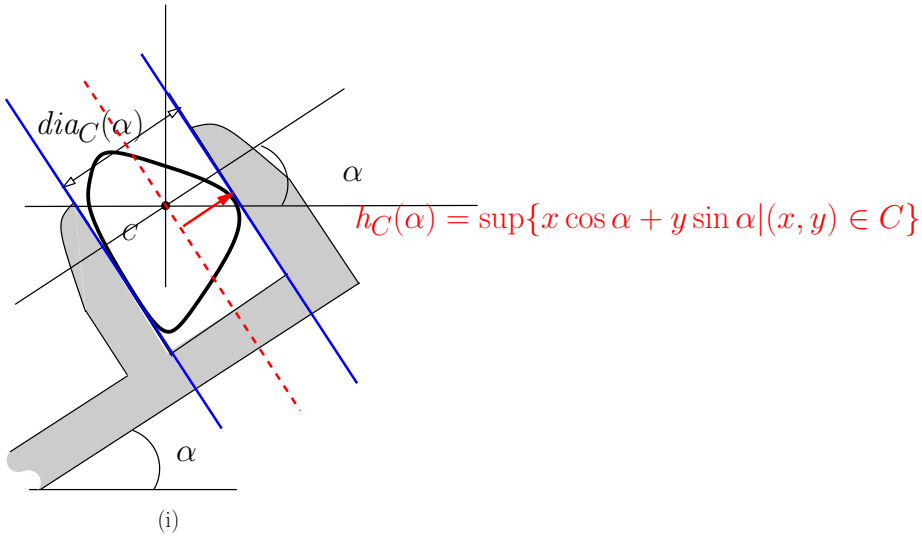
- Geschlossene konvexe Kurve C: Durchm. Funktion:
  $\mathtt{dia}_{\mathtt{C}} : [0, 2\pi) \to \mathtt{R}$
- Support Funktion: $\mathtt{h}_{\mathtt{C}}(\alpha) = \sup\{\mathtt{x}\cos\alpha + \mathtt{y}\sin\alpha | (\mathtt{x}, \mathtt{y}) \in \mathtt{C}\}$
- X-Koordinate nach Rotation mit α



(i)

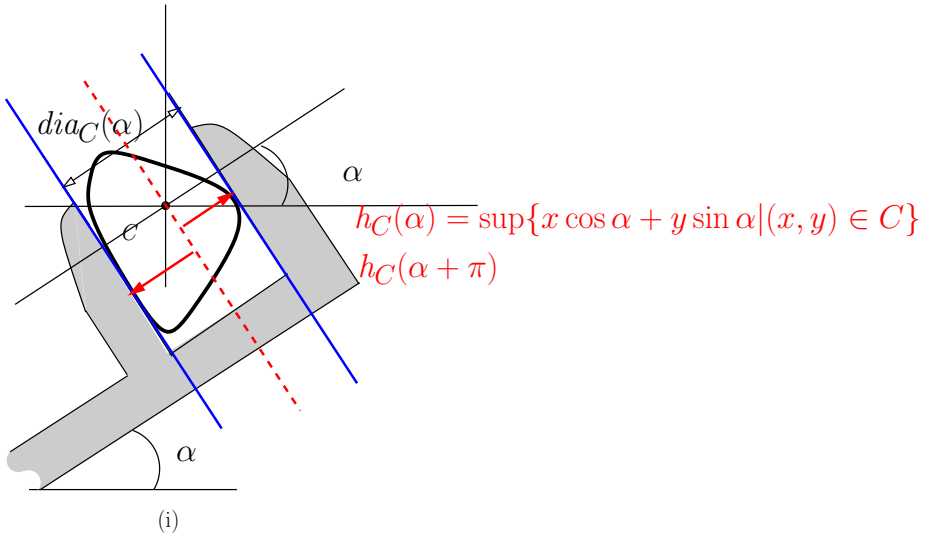
Rotation? Hilfe: Surface-Area Th. 1.46!

- Geschlossene konvexe Kurve C: Durchm. Funktion: $\mathtt{dia_C} : [0, 2\pi) \to \mathtt{R}$
- Support Funktion: $\mathtt{h_C}(\alpha) = \sup\{\mathtt{x}\cos\alpha + \mathtt{y}\sin\alpha | (\mathtt{x}, \mathtt{y}) \in \mathtt{C}\}$
- X-Koordinate nach Rotation mit α



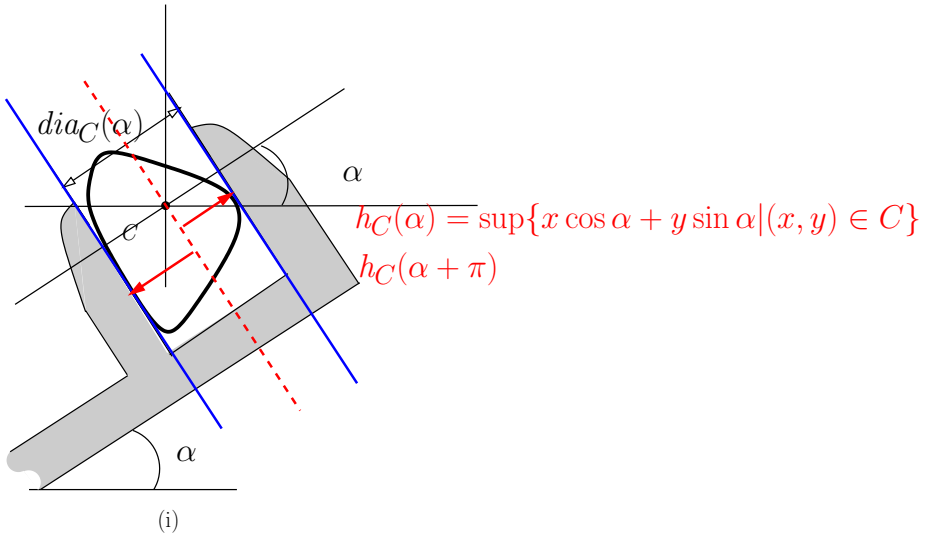
Rotation? Hilfe: Surface-Area Th. 1.46!

- Geschlossene konvexe Kurve C: Durchm. Funktion: $\mathtt{dia_C} : [0, 2\pi) \to \mathtt{R}$
- Support Funktion: $\mathtt{h_C}(\alpha) = \sup\{\mathtt{x}\cos\alpha + \mathtt{y}\sin\alpha | (\mathtt{x}, \mathtt{y}) \in \mathtt{C}\}$
- X-Koordinate nach Rotation mit α
- Offensichtlich: $\mathtt{dia_C}(\alpha) = \mathtt{h_C}(\alpha) + \mathtt{h_C}(\alpha + \pi)$



Rotation? Hilfe: Surface-Area Th. 1.46!

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



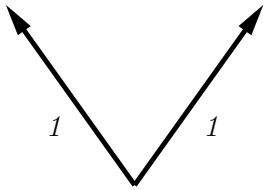
Beweis: Rotation ist optimal!

Beweis: Rotation ist optimal!

- Zwei Kurven C,

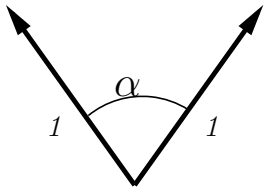
Beweis: Rotation ist optimal!

- Zwei Kurven C,



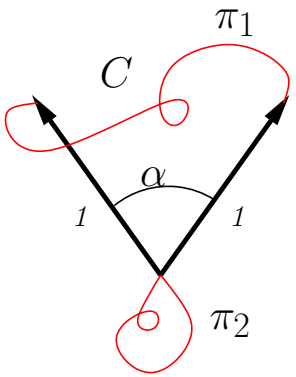
Beweis: Rotation ist optimal!

- Zwei Kurven C,



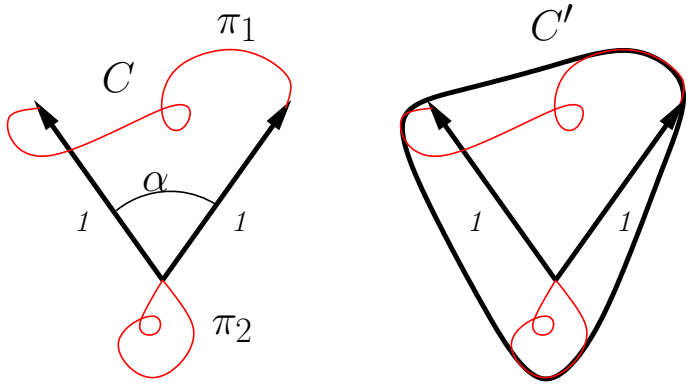
Beweis: Rotation ist optimal!

- Zwei Kurven C, konvexe Hülle C',



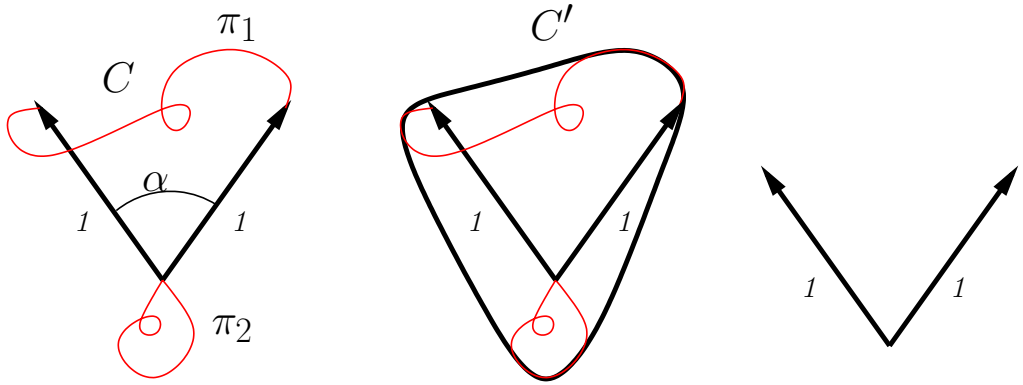
Beweis: Rotation ist optimal!

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



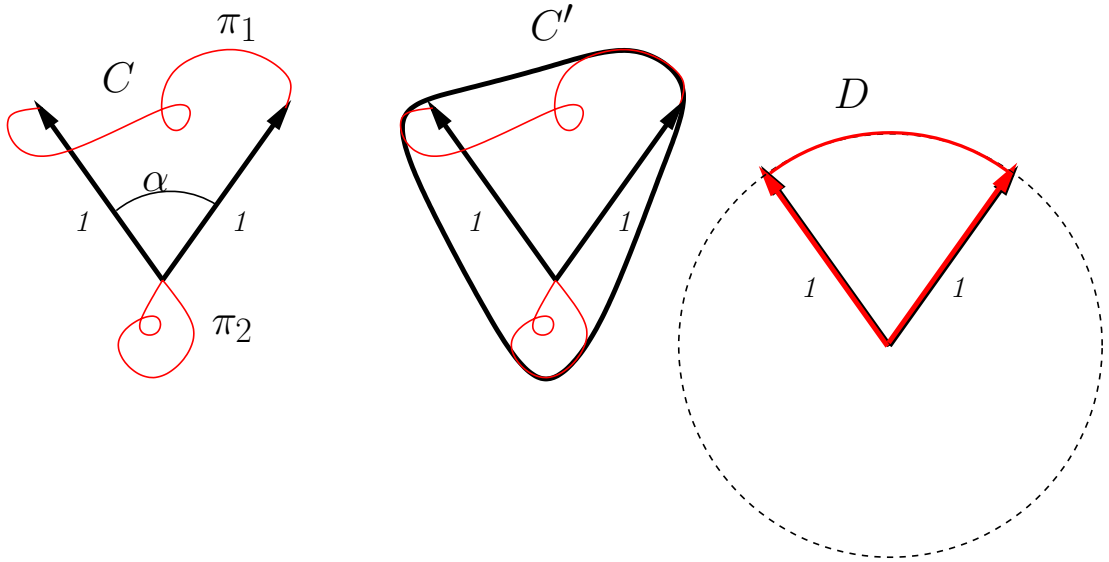
Beweis: Rotation ist optimal!

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

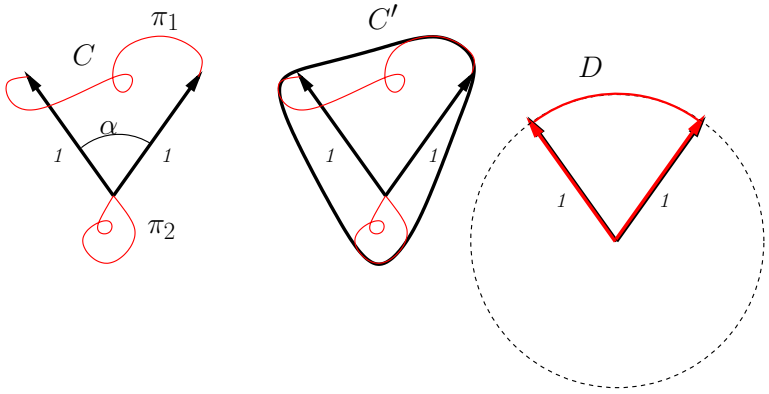


Beweis: Rotation ist optimal!

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

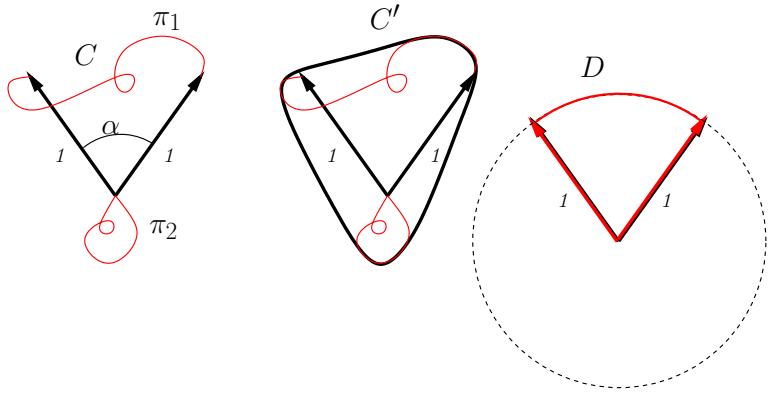


Beweis: Rotation ist optimal!



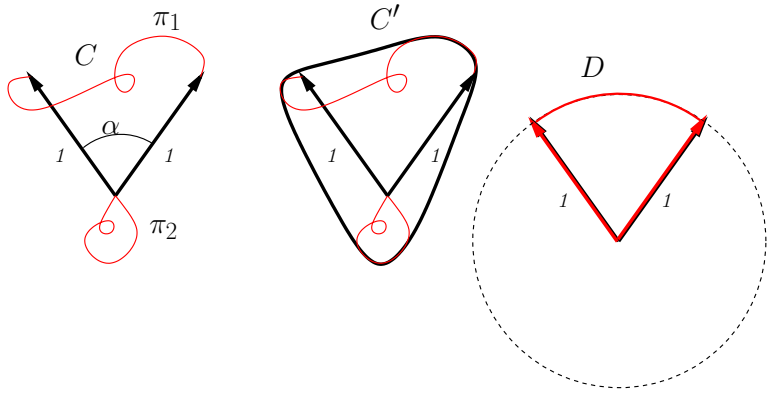
Beweis: Rotation ist optimal!

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



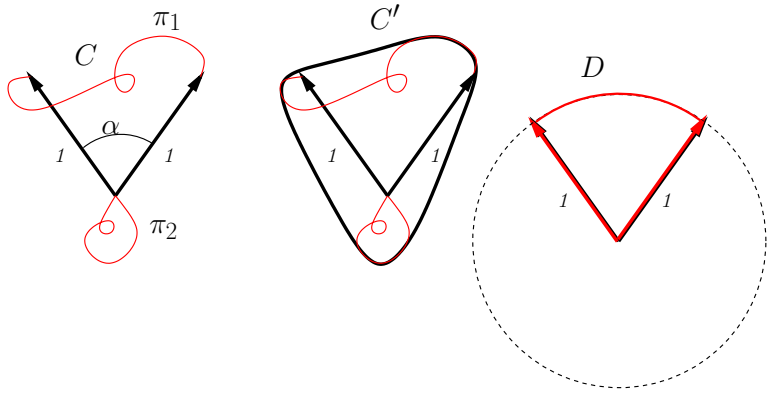
Beweis: Rotation ist optimal!

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



Beweis: Rotation ist optimal!

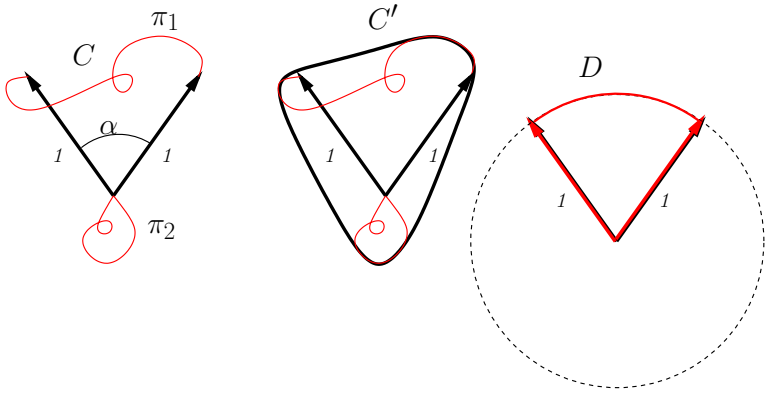
$$2 + \text{Kosten}(C) \quad \geq \quad \text{Länge}(C') = \int_0^{\pi} \mathtt{dia}_{C'}(\alpha)\, \mathrm{d}\alpha \qquad \text{(Th. 1.46)}$$



Beweis: Rotation ist optimal!

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

$$\geq \int_0^\pi \mathtt{dia}_D(\alpha)\,\mathrm{d}\alpha \quad \text{(wegen } \mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_D(\alpha)\text{)}$$

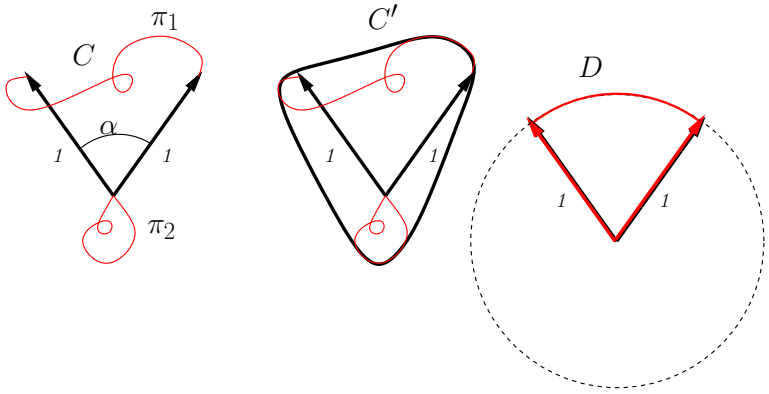


Beweis: Rotation ist optimal!

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

$$\geq \int_0^\pi \texttt{dia}_D(\alpha)\,\mathrm{d}\alpha \quad (\text{wegen } \texttt{dia}_{C'}(\alpha) \geq \texttt{dia}_D(\alpha))$$

$$= \text{Länge}(D) \quad \text{(Th. 1.46)}$$

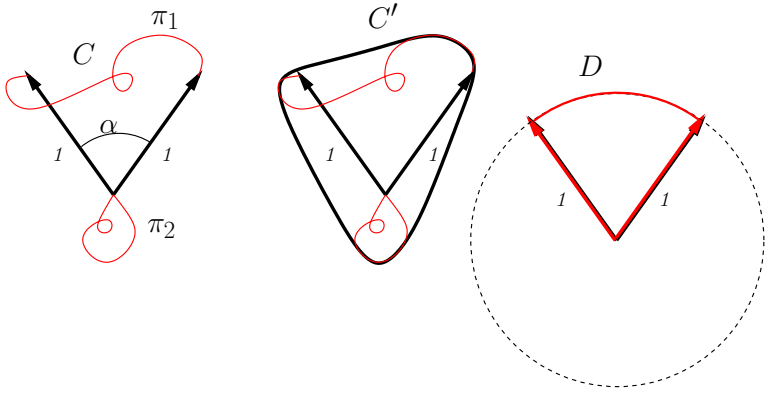


Beweis: Rotation ist optimal!

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

$$\geq \int_0^\pi \mathtt{dia}_D(\alpha)\,\mathrm{d}\alpha \quad (\text{wegen } \mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_D(\alpha))$$

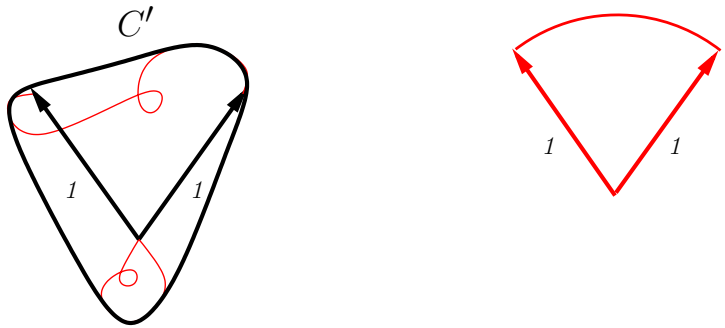
$$= \text{Länge}(D) \quad \text{(Th. 1.46)} = 2 + \text{Kosten(Rotation)}$$



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

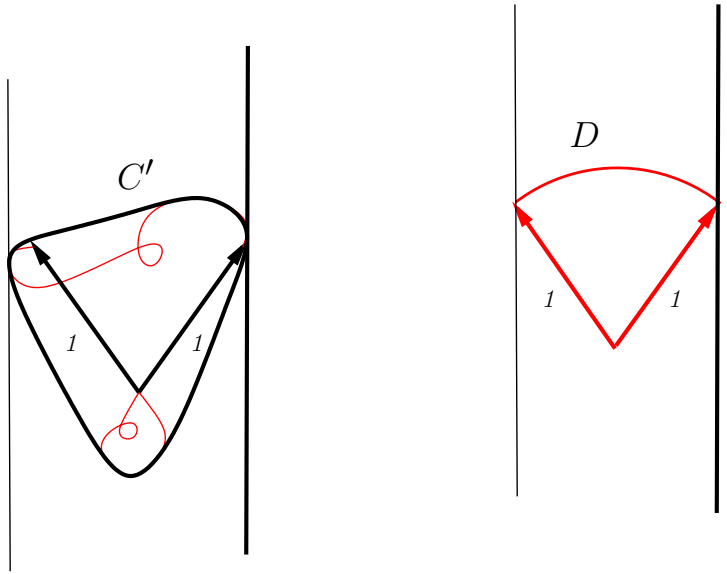
Beweis: $\texttt{dia}_{C'}(\alpha) \geq \texttt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



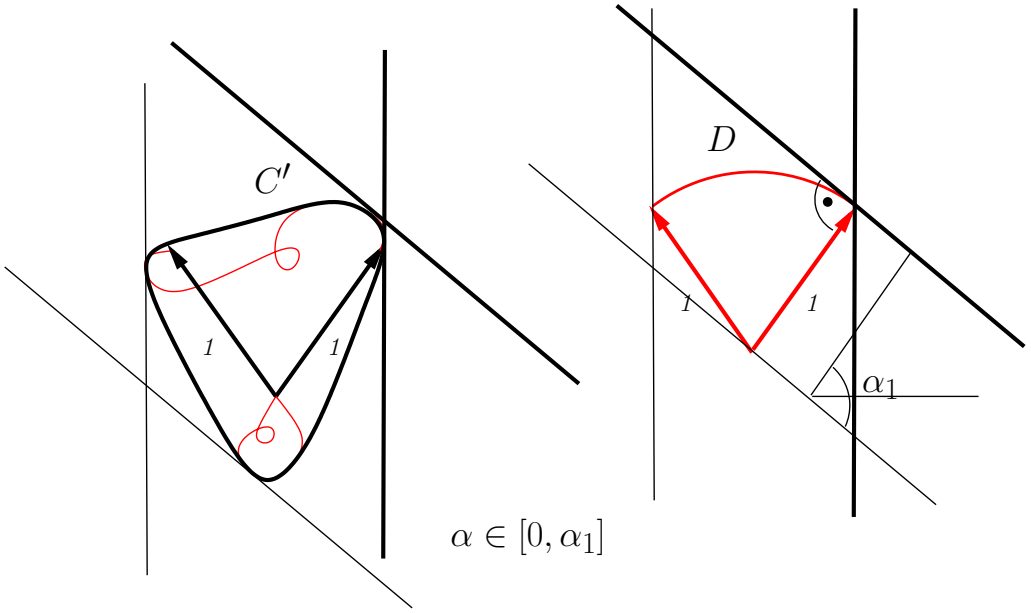
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



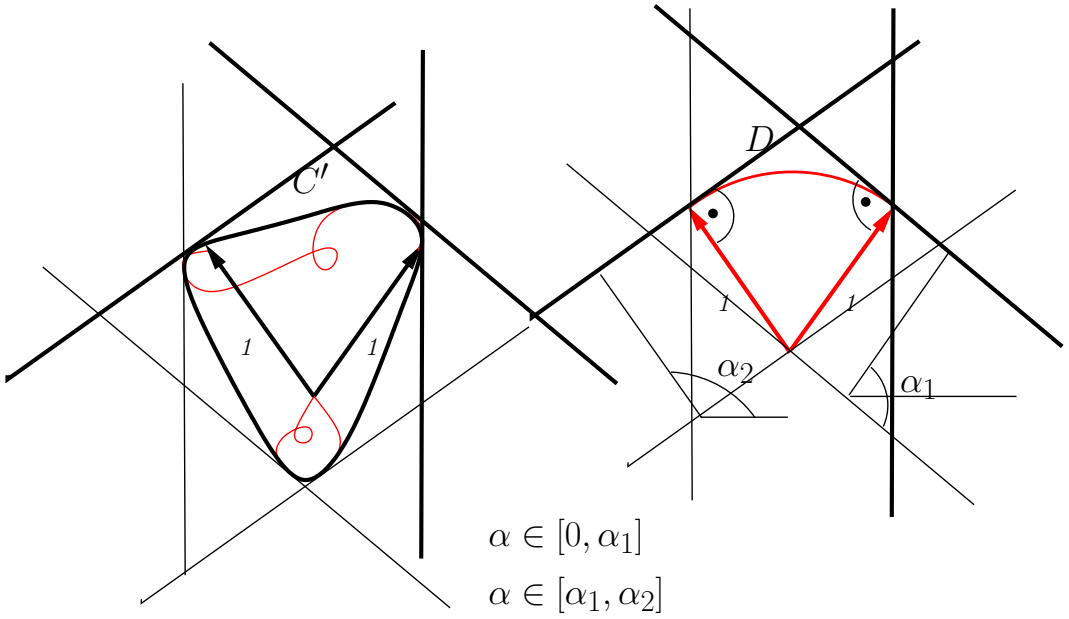
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



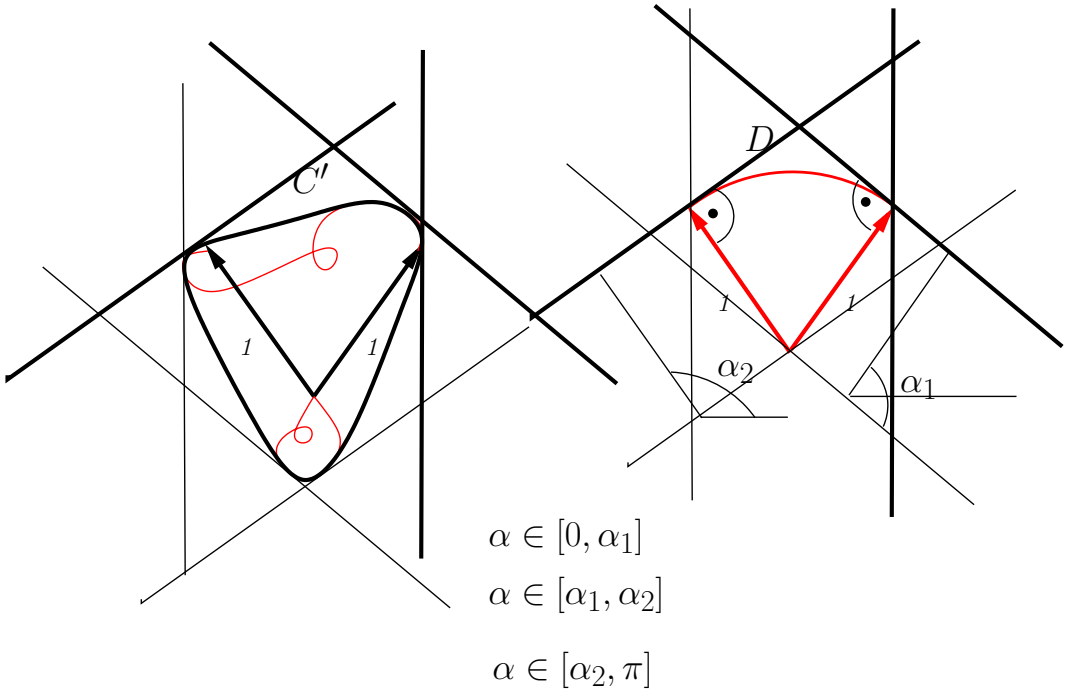
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



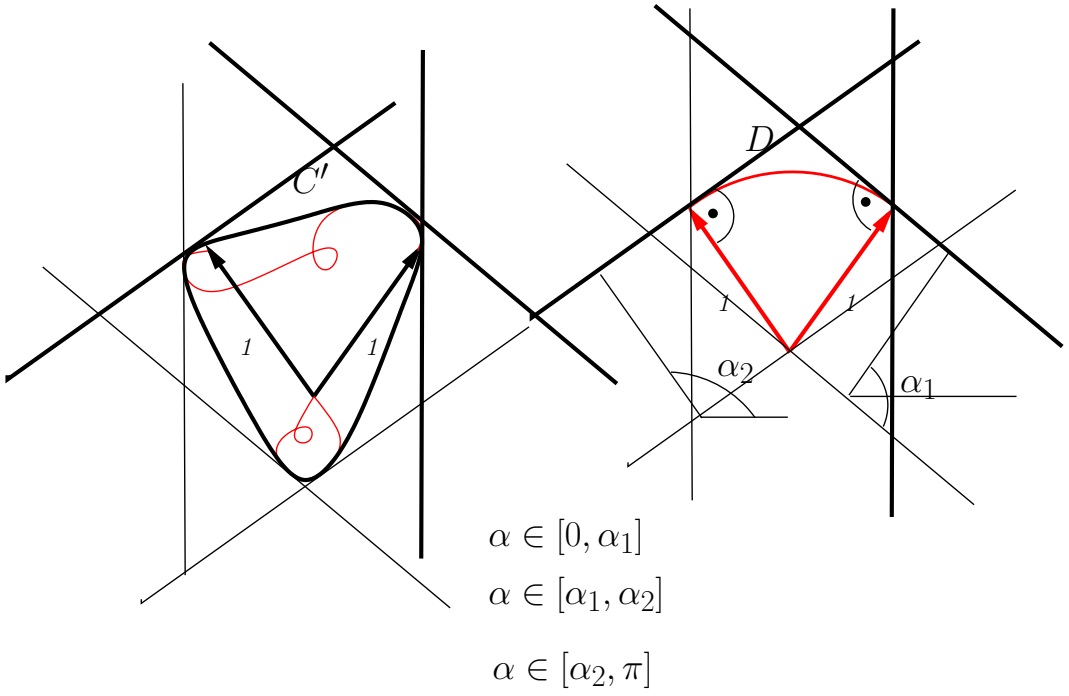
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

- Drei Winkelbereiche!!



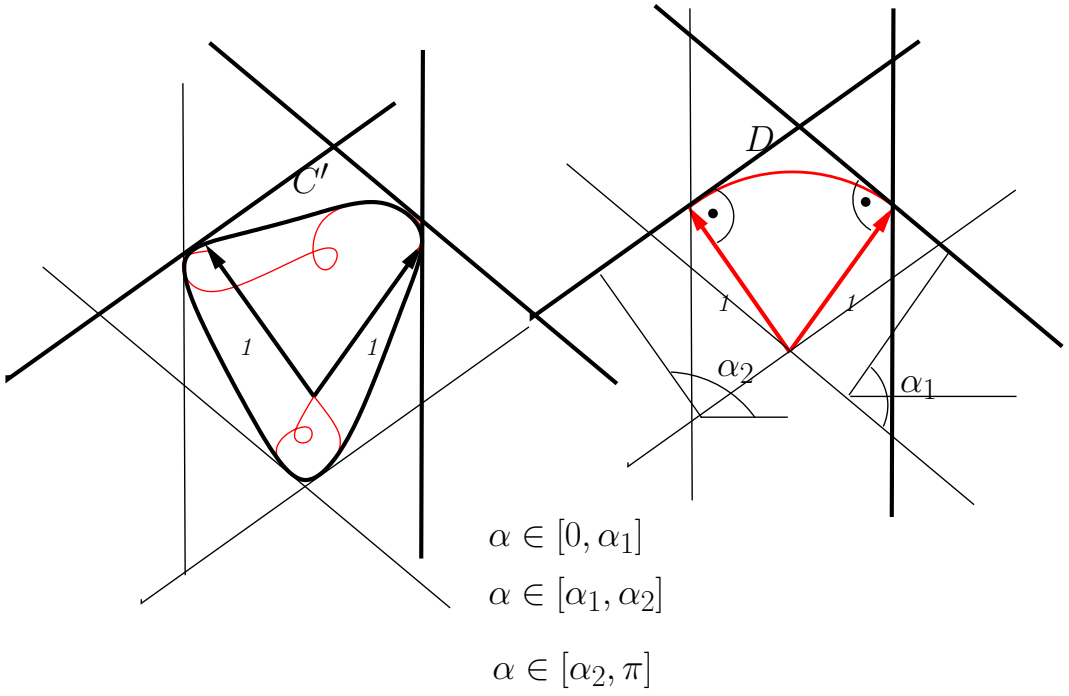
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

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



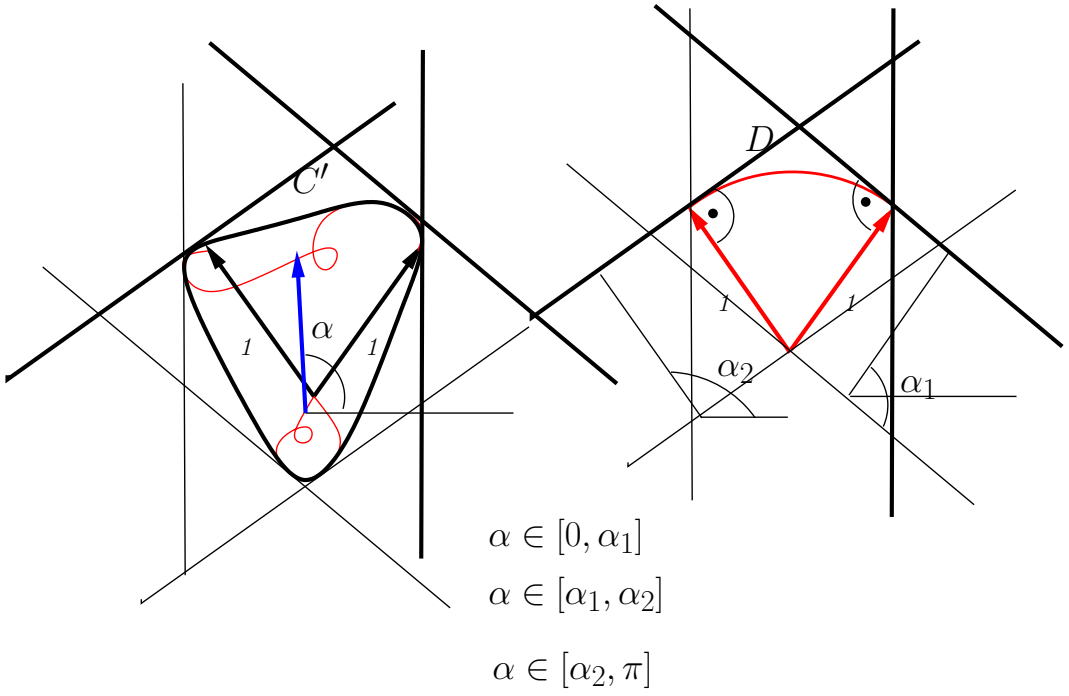
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

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



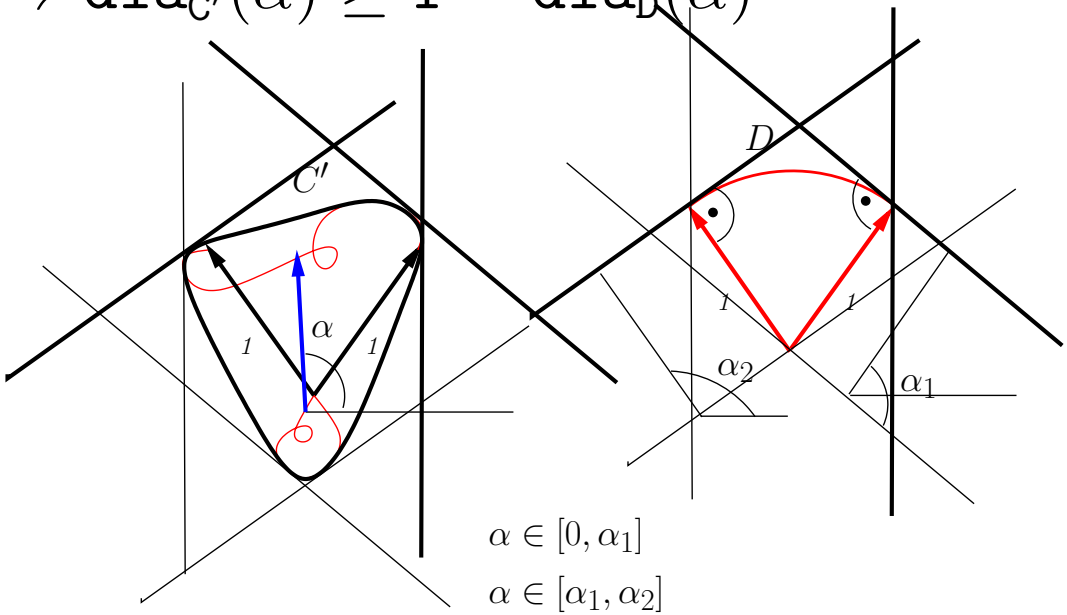
Beweis: $\mathtt{dia}_{C'}(\alpha) \geq \mathtt{dia}_{D}(\alpha)$, $\alpha \in [0, \pi]$

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



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

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



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.

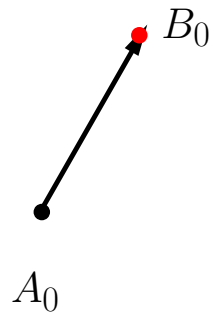
Beispiel: Nahe Placements Th. 1.47

Beispiel: Nahe Placements Th. 1.47

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

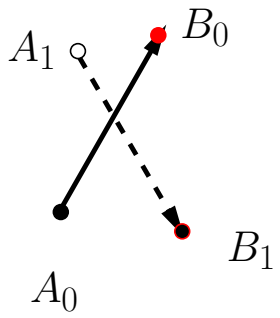
Beispiel: Nahe Placements Th. 1.47

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



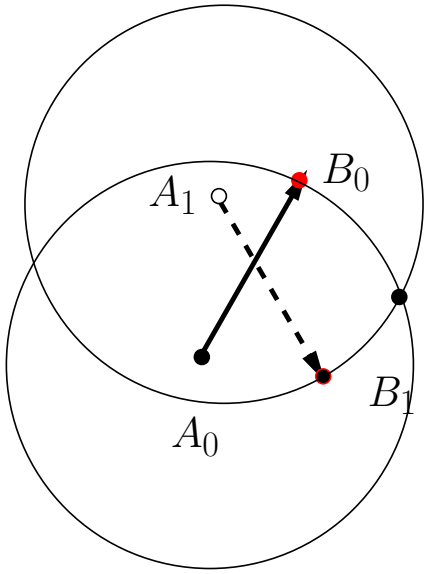
Beispiel: Nahe Placements Th. 1.47

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



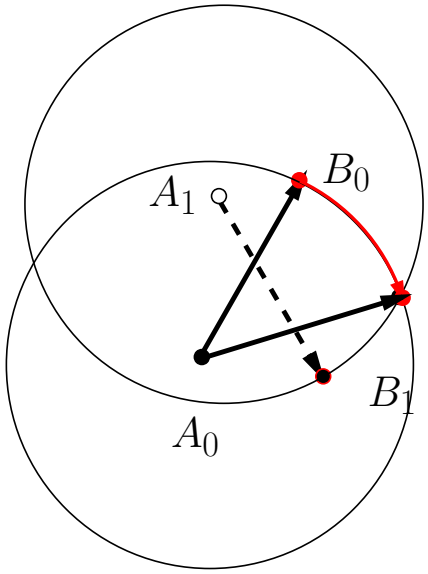
Beispiel: Nahe Placements Th. 1.47

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



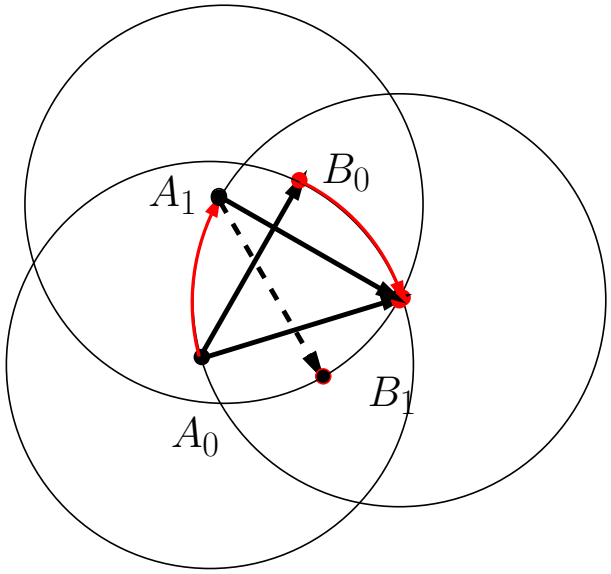
Beispiel: Nahe Placements Th. 1.47

- 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!!