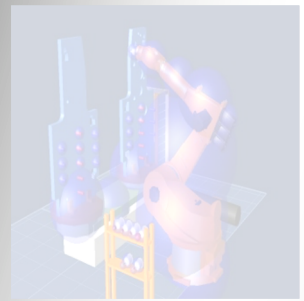
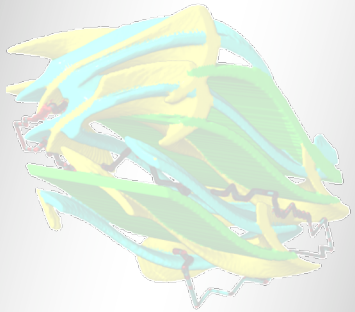
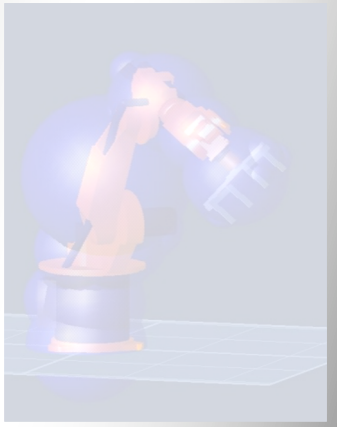
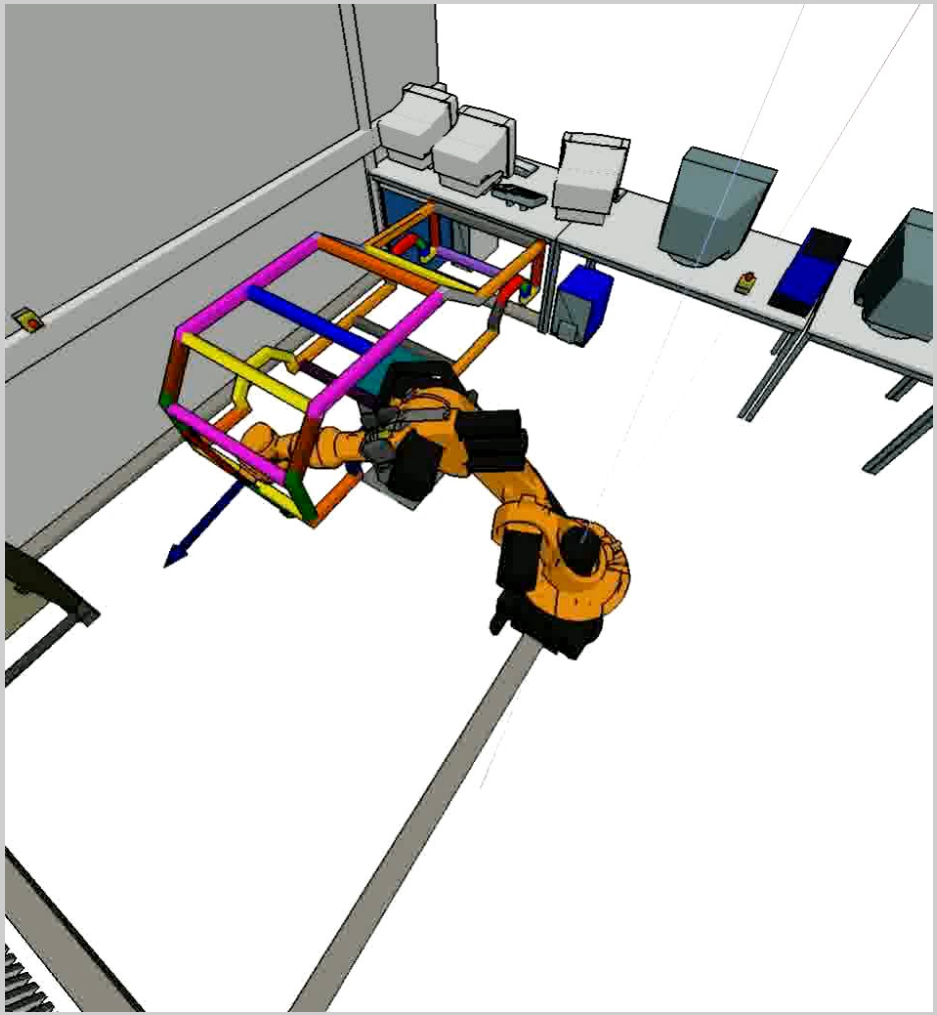
Smoothing

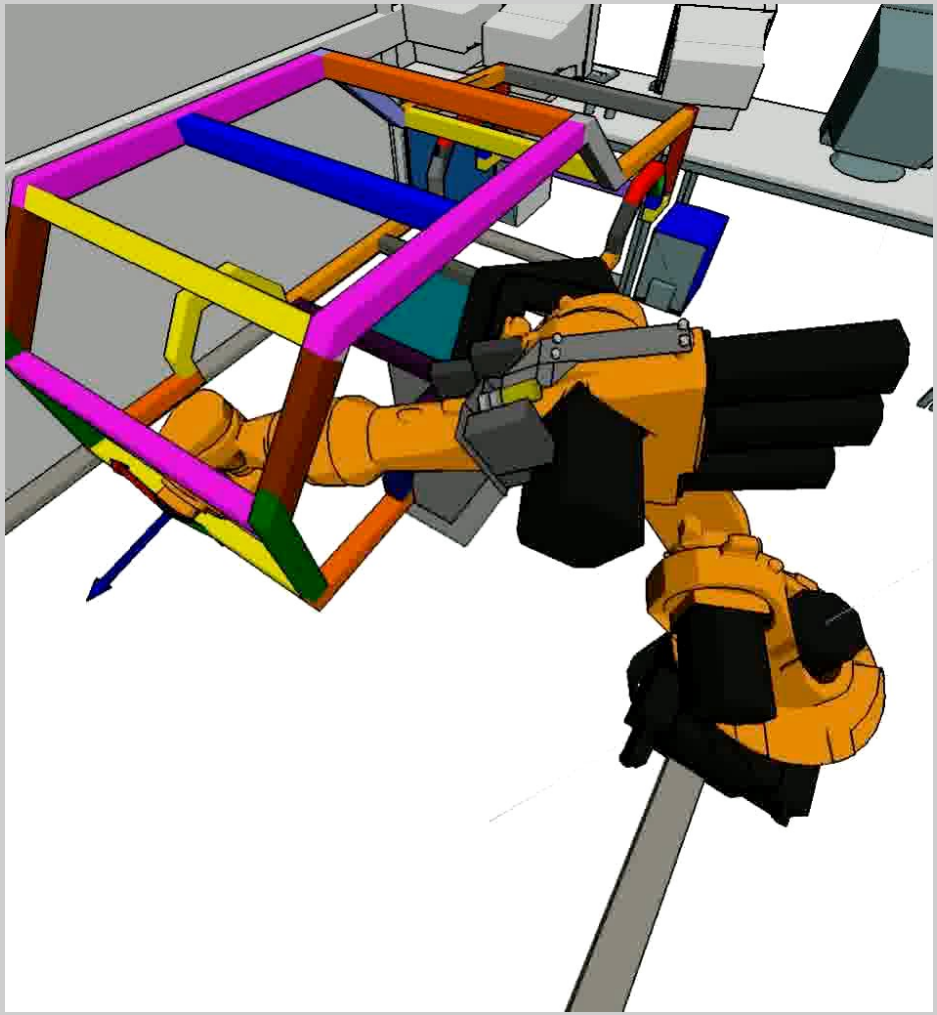






Why smoothing?





Why smoothing? ☺

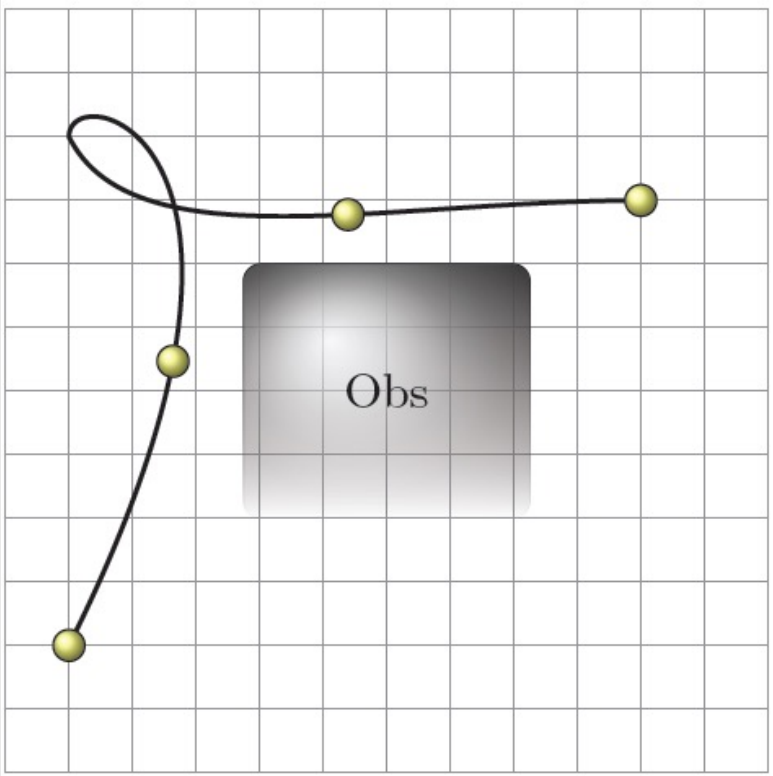
- It‘s obvious that solutions given by the pathplanning algorithms A*, PRM, … don‘t look very nice…..
- BB-Method looks quite good, but sometimes unnecessary detours are made.

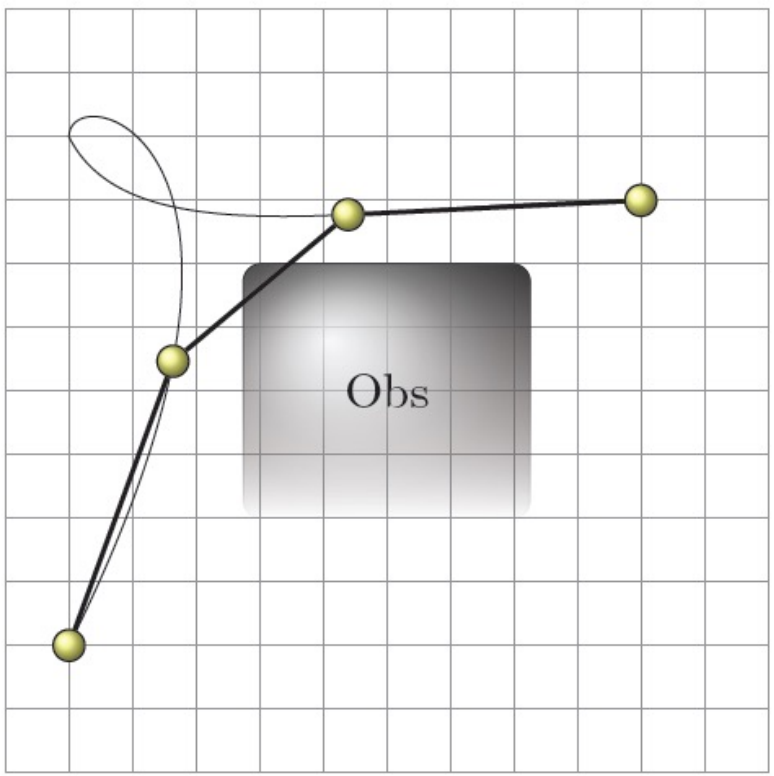
→Smoothing is always useful but it must be collision free

- Removing unnecessary points
- Reducing path length

Problem in smoothing

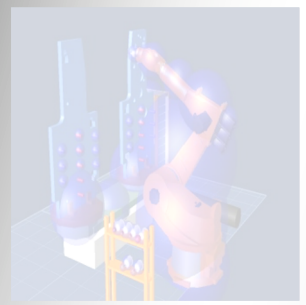
- Direct connection leads to collision

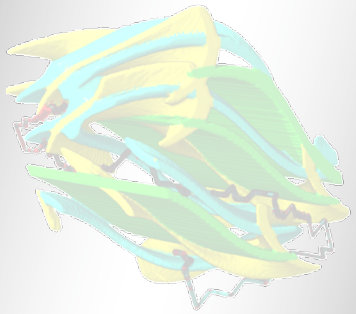


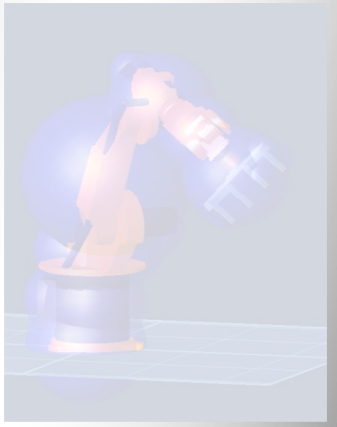


Smoothing Approach

Linear approximation examples

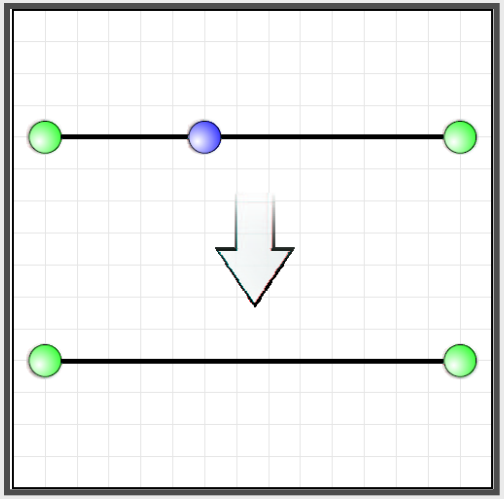




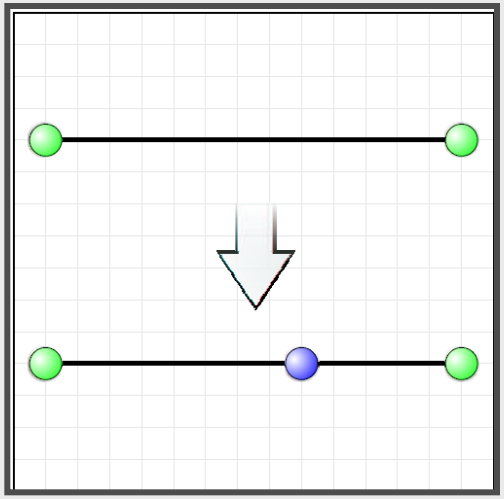


Necessary basic operations

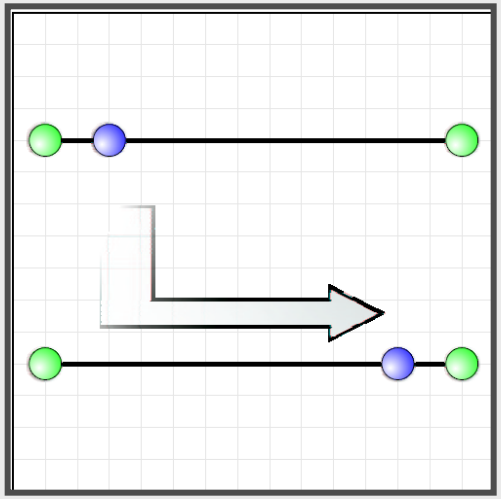
Removing collinear points



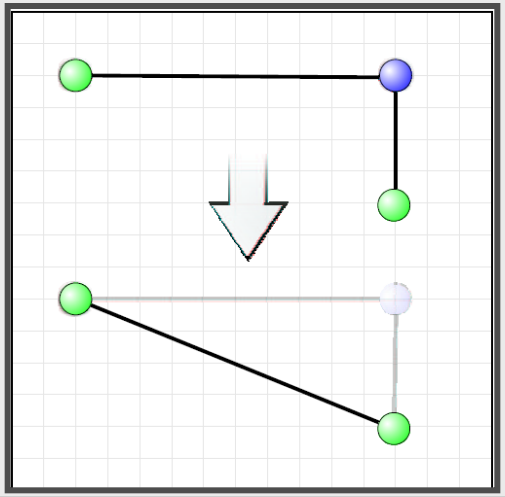
Adding collinear points



Shifting collinear points

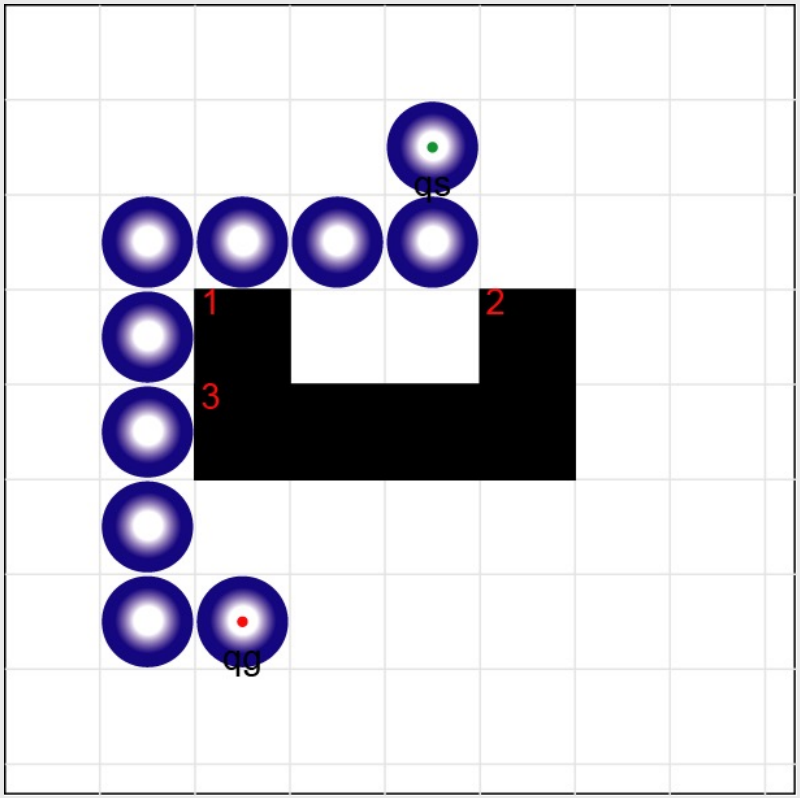


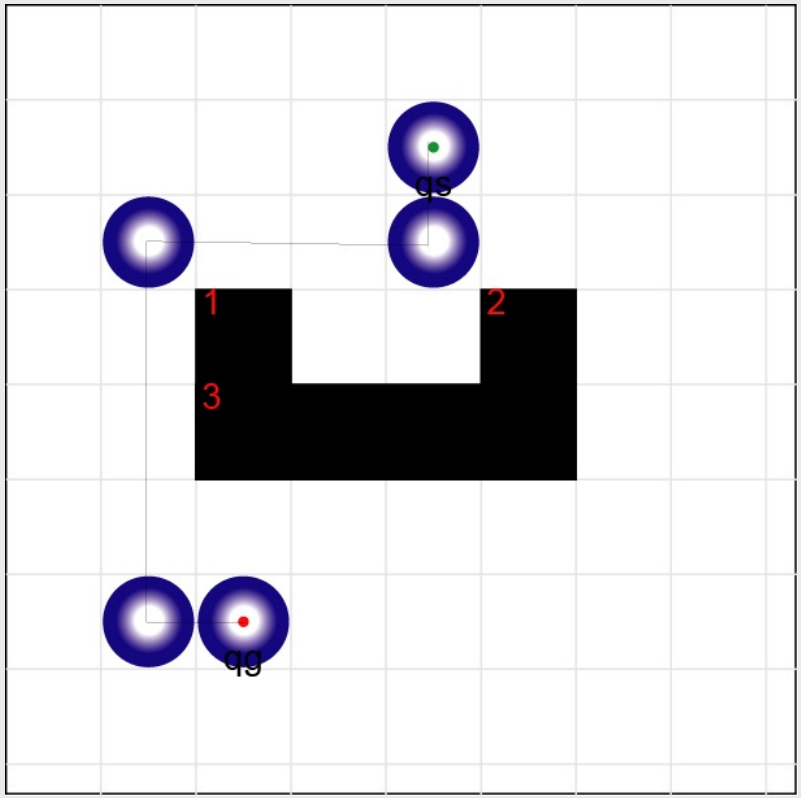
delete triangle points



First of all... remove collinear points

- Extremly necessary using A* and sometimes for RRT

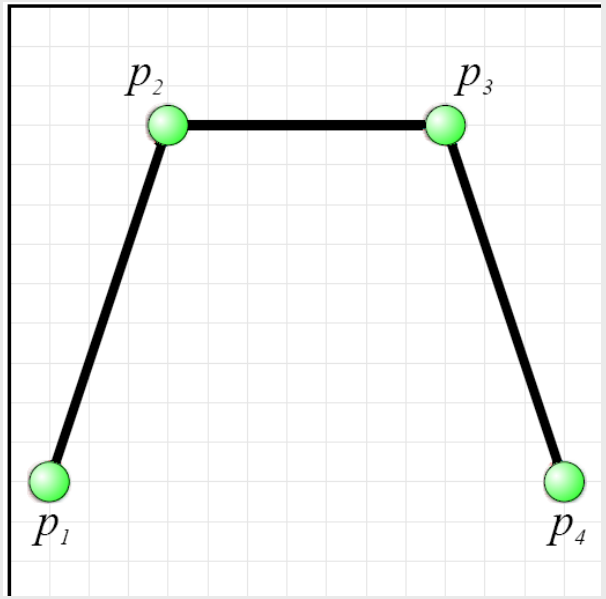


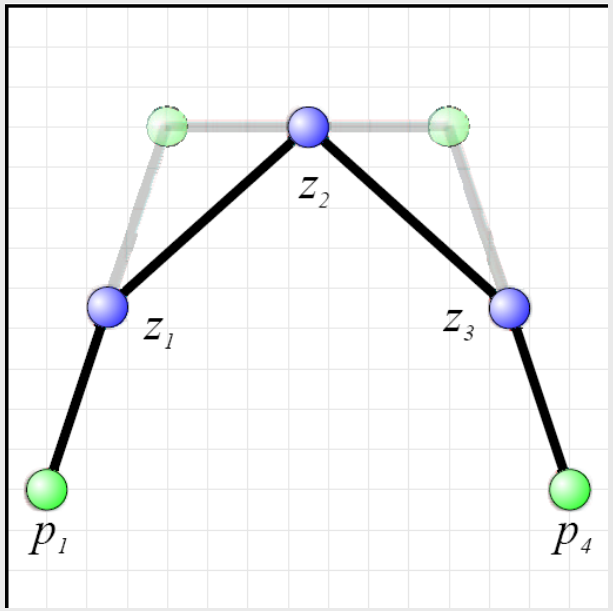


Mid-Point Approach

- For a triple of points, intermediat positions are generated

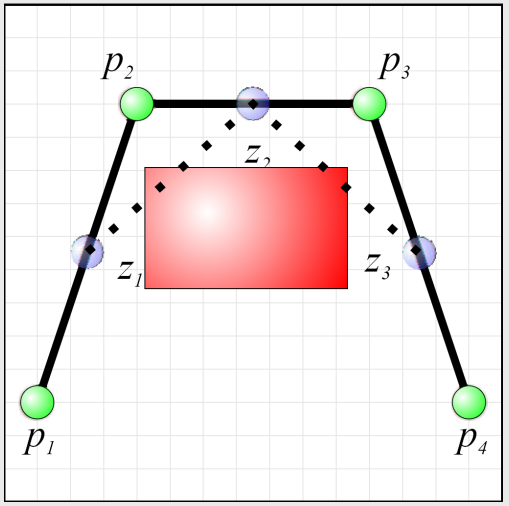
$$z_i = \frac{p_i + p_{i+1}}{2} \quad \text{and} \quad z_{i+1} = \frac{p_{i+1} + p_{i+2}}{2}$$





Mid-Point Approach (2)

- Algorithm ends
  - if no pathpoint can removed anymore



  - distance between p_{i+1}, z_i, z_{i+1} is smaller than a certain value ε:

$$\forall i : d(p_{i+1}, z_i) < \varepsilon \quad \text{and} \quad \forall i : d(p_{i+1}, z_{i+1}) < \varepsilon$$

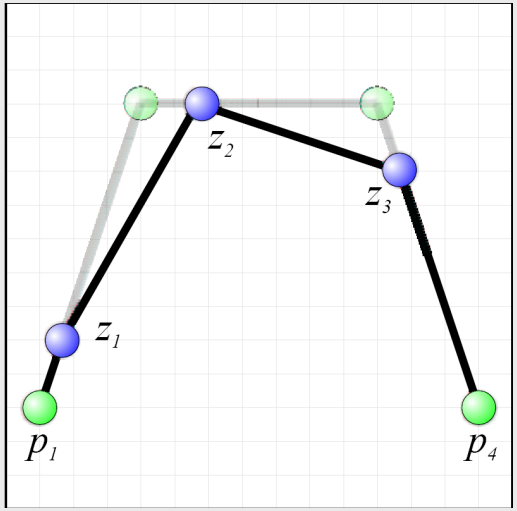
- This approach will end up in a Bezier-shaped curve

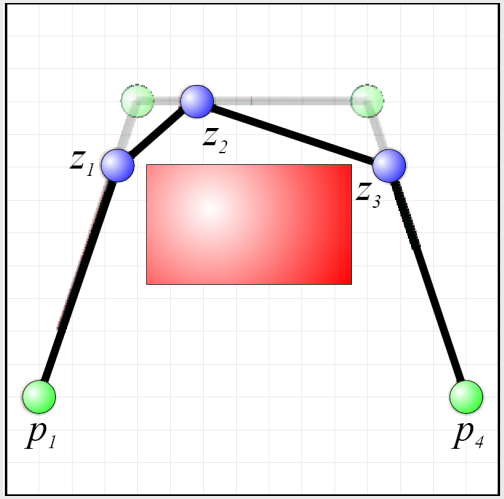
Mid-Point Approach (3)

- Possible extensions are
  - Dont use exactly mid of edges

$$z_i = \frac{u * p_i + (1-u)p_{i+1}}{2} \qquad z_{i+1} = \frac{u * p_{i+1} + (1-u)p_{i+2}}{2}$$

$0 < u < 1,\ u \in \Re$

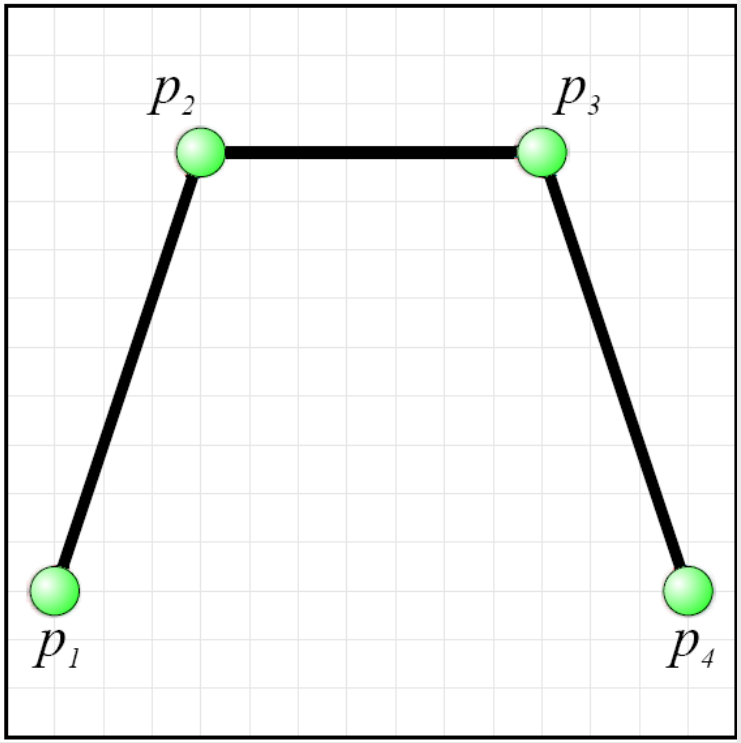


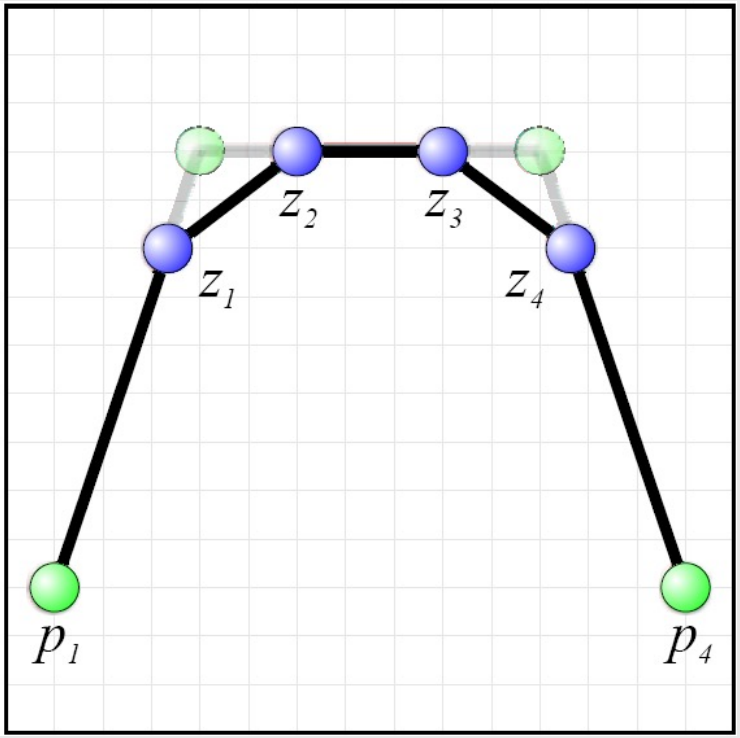


Or of course using any other kind of subdivision…

2-Point-Approach

- Try to replace a pathpoint with 2 new pathpoints and a shorter path segment
  - Also converging to a curve
  - Path is shorter but more path points in it

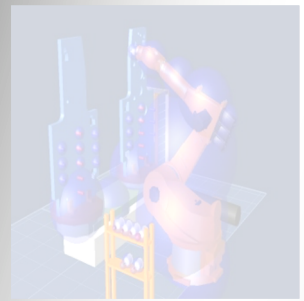


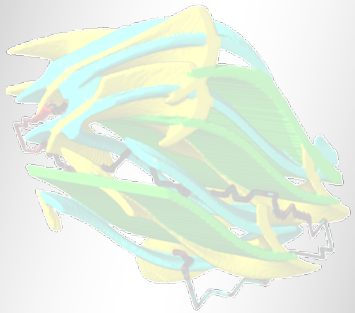


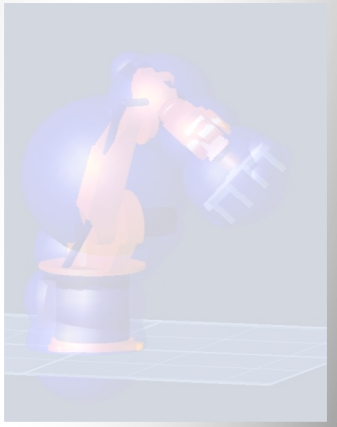
Summary

- Both approaches Mid-Point and 2-Point will shorten path length avoiding obstacles
- But: they will also increase number of points of the path
  - → bad, bad, bad for using it with an industrial robot controller! These controllers cannot execute fast movements as soon as many points are given in short distances.
  - → Need an approach which removes as many points as possible, but also add necessary points

Generic Approach for Path Smoothing



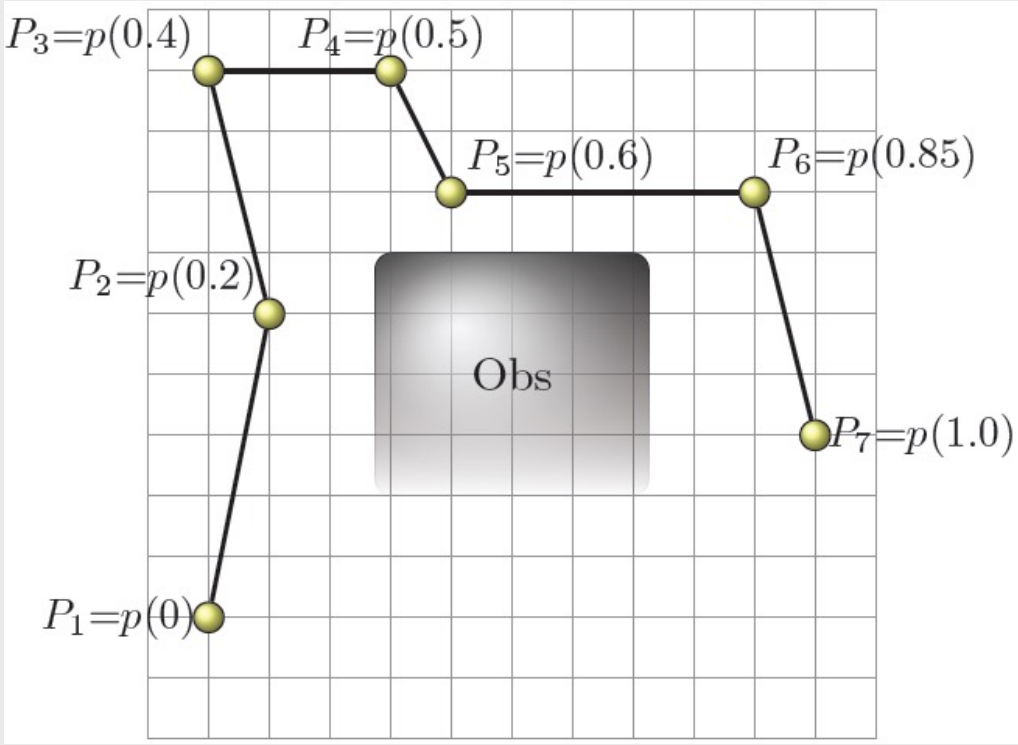


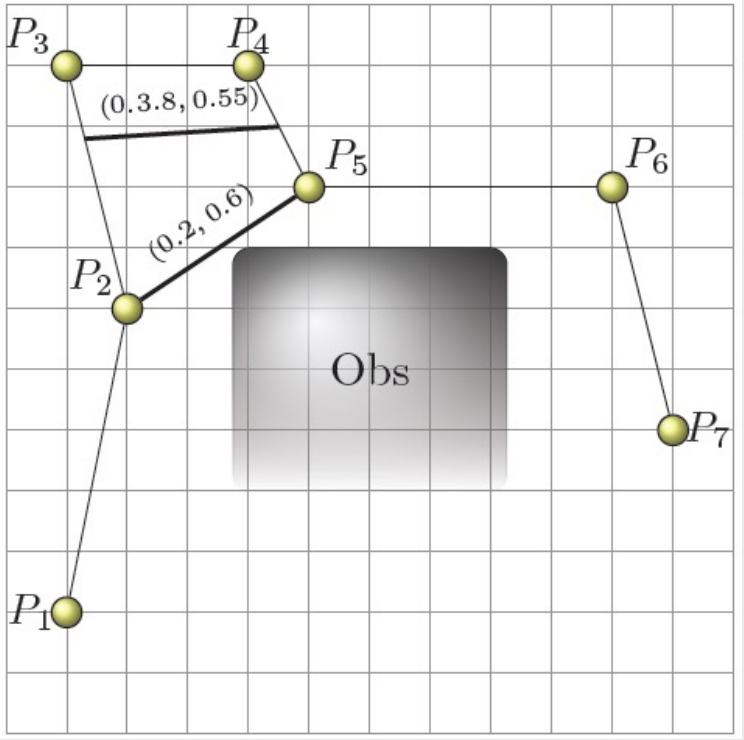


How to

Path definition

- P = p(t), t in [0,1]
- P_{start} = p(0), P_{end} = p(1)





- (l_A,l_B) → path segment

Basic smoothing algorithm

`Glätte`(p):

Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten

```
// Bestimme ein Pfadsegment
(l_A, l_B) ⟵ WählePositionen(p, wahr)

solange ? tue
    erfolg ⟵ OptimierStrategie(l_A, l_B, p)
    (l_A, l_B) ⟵ WähleAbkürzung(p, l_A, l_B, erfolg)
```

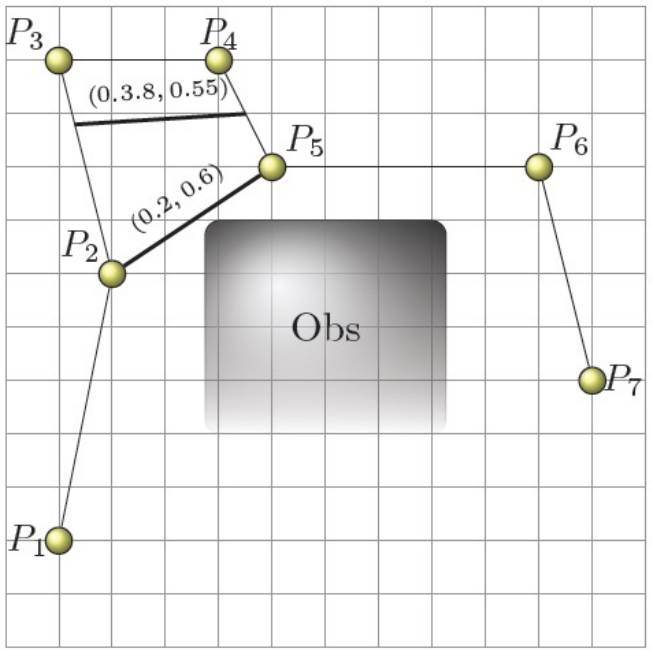
Auswahlfunktion: zwei **Stützstellen des Pfades** durch Auswahl der korrespondierenden Laufvariablen

Optimisation strategy

`OptimierStrategie`(l_A, l_B, p):

Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten
Eingabe : l_A, l_B aktuelle Positionen, zwischen denen getestet werden soll

```
// Wenn das Liniensegment frei ist
wenn FreieVerbindung(l_A, l_B) dann
    für alle Stützpunkte l ∈ (l_A, l_B) tue
        Entferne(p, l)

    // Füge Abkürzung ein
    FügeNeuesSegmentEin(p, l_A, l_B)

    zurück wahr
zurück falsch
```



Shortcut Strategies

- How to choose start and end point of shortcut?

Glätte(p):

Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten

```
// Bestimme ein Pfadsegment
(l_A, l_B) ⟵ WählePositionen(p, wahr)

solange ? tue
    erfolg ⟵ OptimierStrategie(l_A, l_B, p)
    (l_A, l_B) ⟵ WähleAbkürzung(p, l_A, l_B, erfolg)
```

- Pure probabilistic (Did work out for path planners!)
- „Extended“ Probabilistic (Latombe)
- Deterministic (Bechthold & Glavina)

Shortcut Strategy: Pure Probabilistic

`WähleAbkürzung`$(p, l_A, l_B, erfolg)$:

Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten
Eingabe : l_A,l_B aktuelle Position auf dem Pfad
Eingabe : erfolg, gibt an, ob der letzte Glättungsschritt erfolgreich war

```
// Bestimme ein Pfadsegment
```
$l_A \longleftarrow$ `zufallszahl`$(0, 1)$

$l_B \longleftarrow$ `zufallszahl`$(0, 1)$

zurück (l_A, l_B)

Shortcut Strategy: „Extended“ Probabilistic (Latombe)

`WähleAbkürzung`$(p, l_A, l_B, erfolg)$:

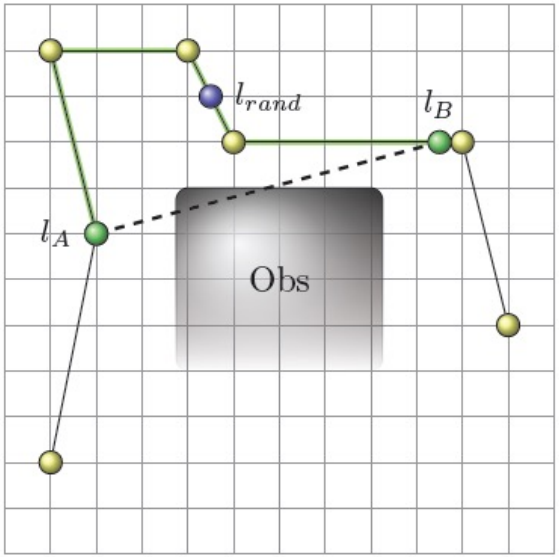
Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten
Eingabe : l_A,l_B aktuelle Position auf dem Pfad
Eingabe : erfolg, gibt an, ob der letzte Glättungsschritt erfolgreich war

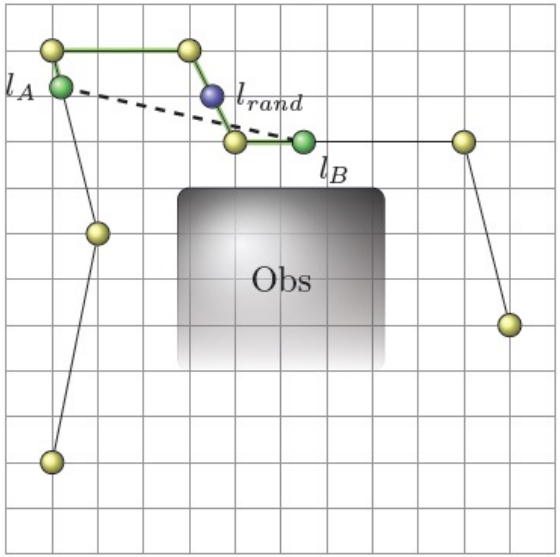
wenn *erfolg oder* $l_A - l_B < \Delta l_{min}$ **dann**
  $l_{rand} \longleftarrow$ `zufallszahl`$(0, 1)$
  $l_A \longleftarrow 0$
  $l_B \longleftarrow 1$

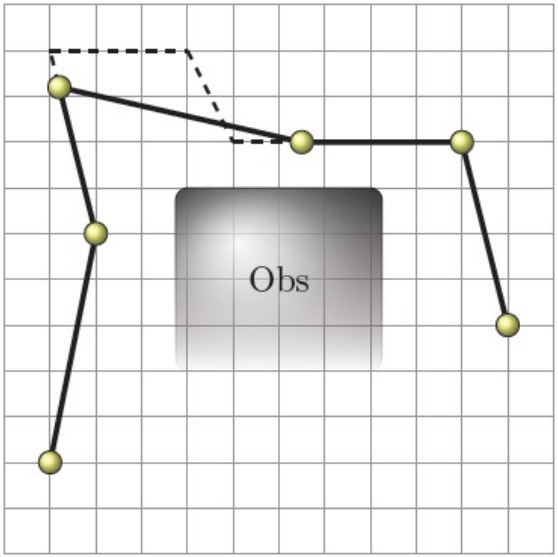
$l_A \longleftarrow \frac{1}{2} * (l_A + l_{rand})$
$l_B \longleftarrow \frac{1}{2} * (l_B + l_{rand})$
zurück (l_A, l_B)

l_{rand} „globale“ Variable, d.h. Wert bleibt bei weiterem Aufruf der Funktion erhalten.

Shortcut Strategy: „Extended“ Probabilistic (Latombe)







Shortcut Strategy: Deterministic (Bechthold & Glavina)

`WähleAbkürzung`$(p, l_A, l_B, erfolg)$:

Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten
Eingabe : l_A,l_B aktuelle Position auf dem Pfad
Eingabe : erfolg, gibt an, ob der letzte Glättungsschritt erfolgreich war

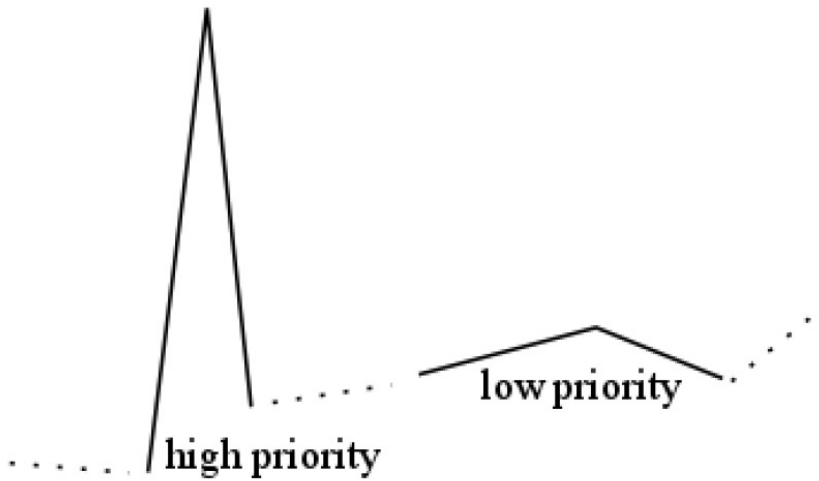
wenn *erfolg oder* $l_A - l_B < \Delta l_{min}$ **dann**
  $l_w \longleftarrow$ `LiefereSchlechtestenStüztpunkt`(p)
  $l_A \longleftarrow$ `Vorgänger`(l_w)
  $l_B \longleftarrow$ `Nachfolger`(l_w)
sonst
  $l_A \longleftarrow \frac{1}{2} * (l_A + l_w)$
  $l_B \longleftarrow \frac{1}{2} * (l_B + l_w)$
zurück (l_A, l_B)

Deterministic Shortcut: Evaluating points

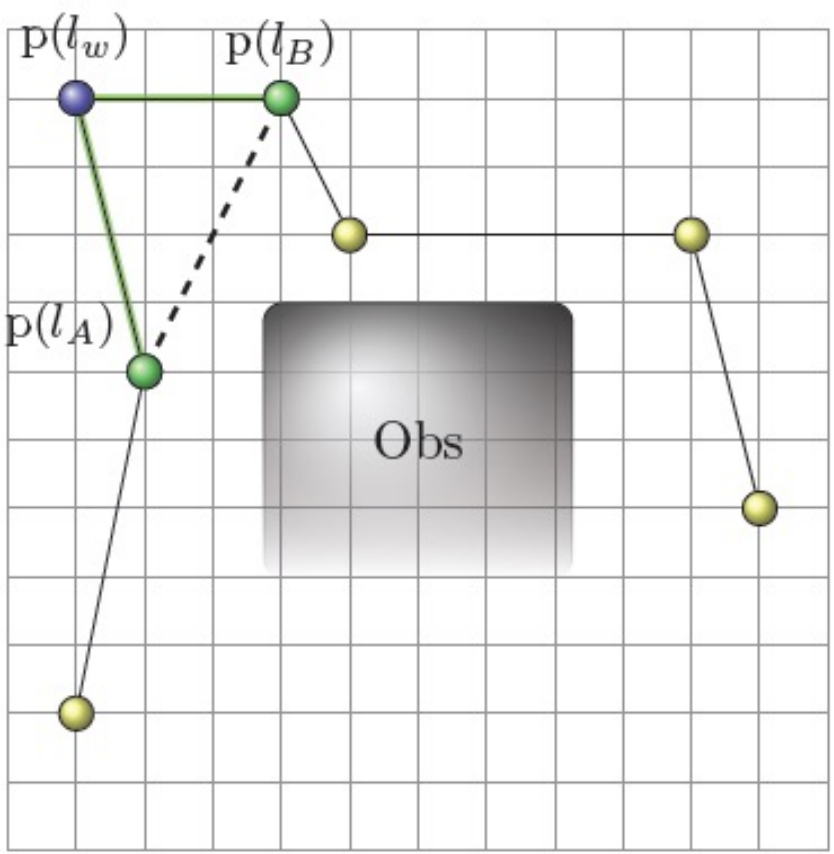
- Points are measured (evaluated) using:

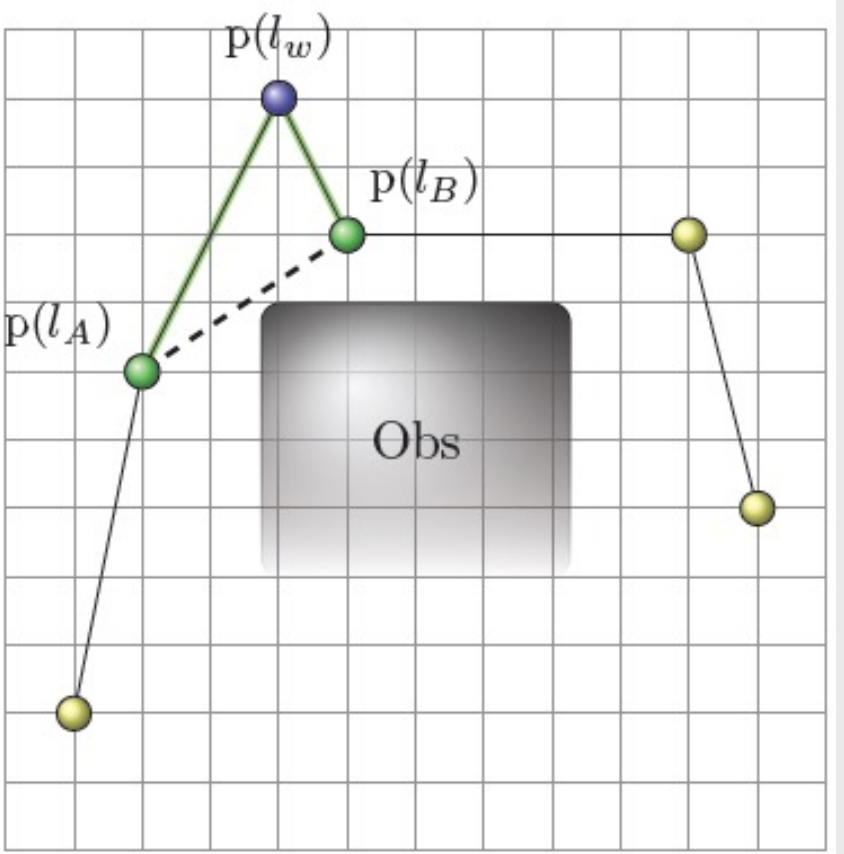
$$Q_{p_2} = \frac{|p_1p_2| + |p_2p_3|}{|p_1p_3|}$$

- Points are then inserted in a **priority list** (constant time getting „worst one“)



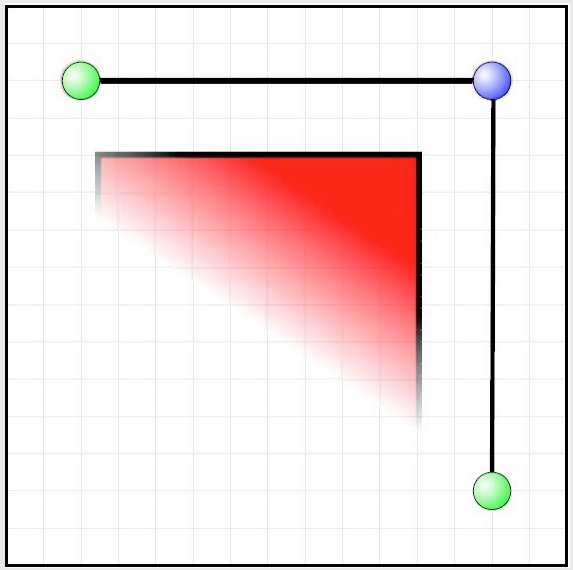
Shortcut: choosing „worst“ point of path

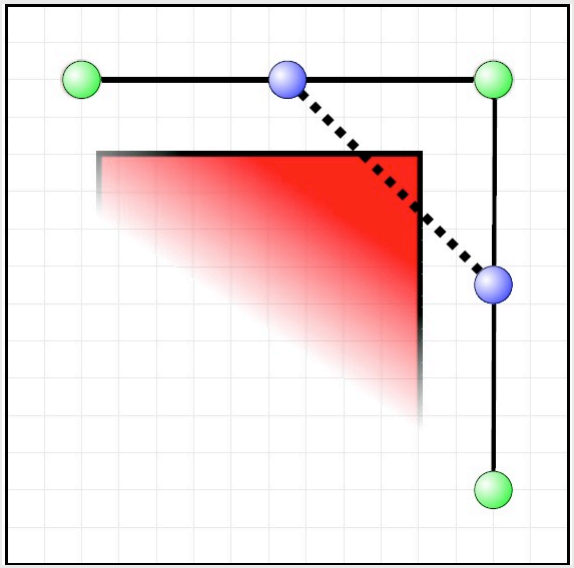


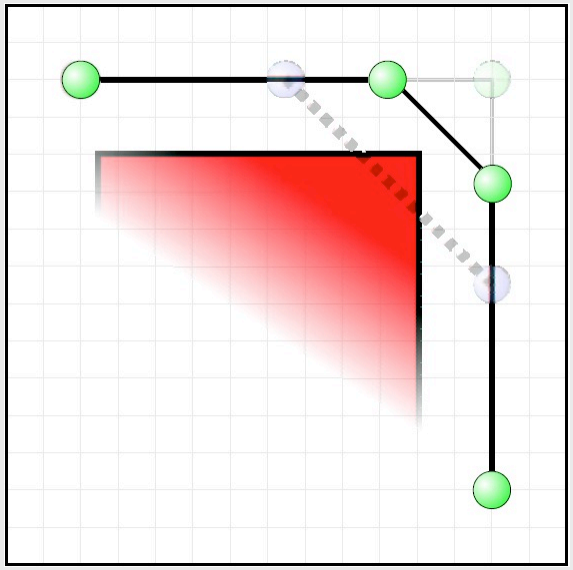


Operation DelTree

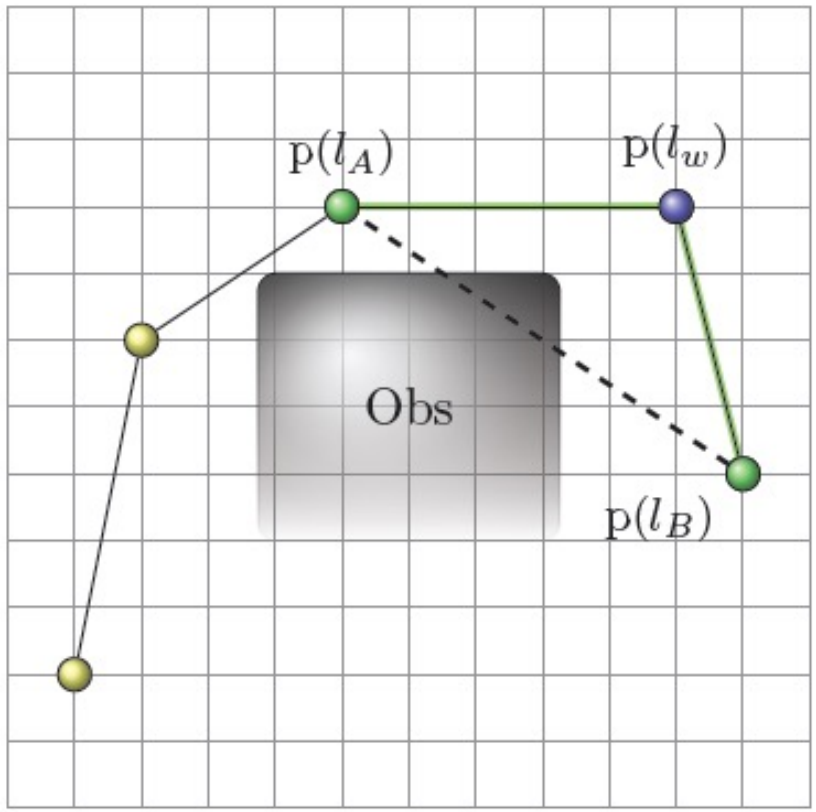
- DelTree (=Mid-Point):
  - Generating new points $p_{z1} = p_2 + \frac{\overrightarrow{p_2 p_1}}{2^t}$ $p_{z2} = p_2 + \frac{\overrightarrow{p_3 p_1}}{2^t}$
  - If connection fails → bisection (t = t+1)
  - Stopping if $\frac{|p_2 p_1|}{2^t} < \varepsilon$ $\frac{|p_2 p_3|}{2^t} < \varepsilon.$

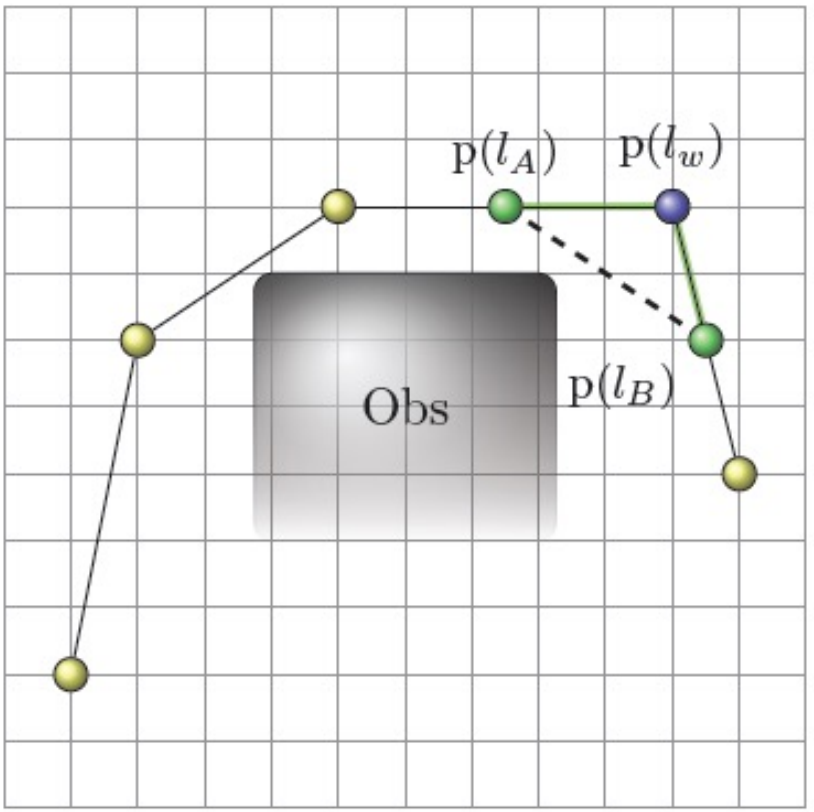






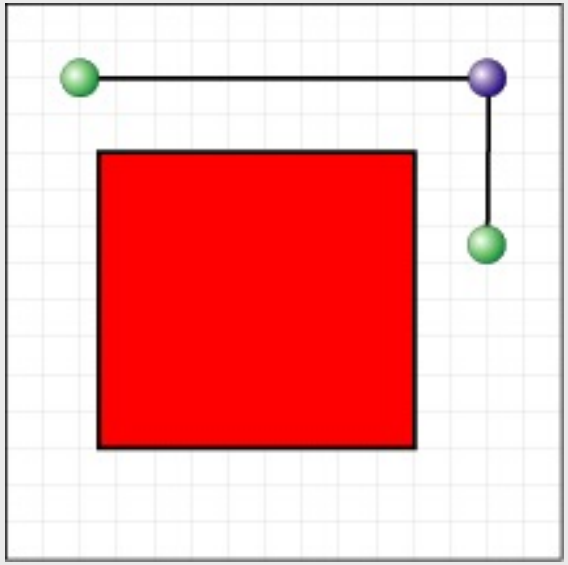
Operation DelTree

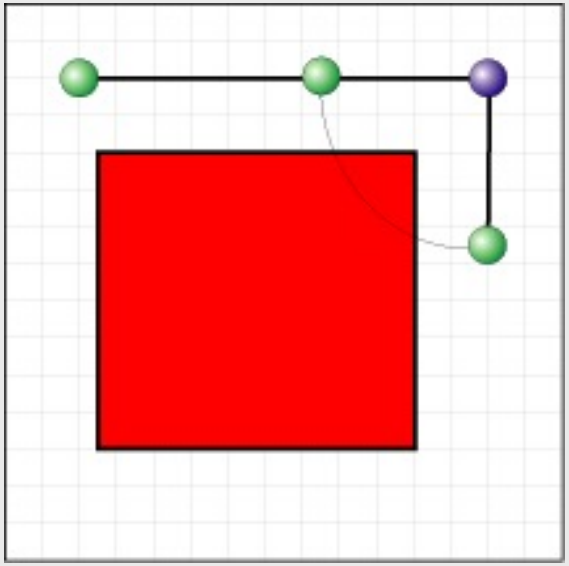


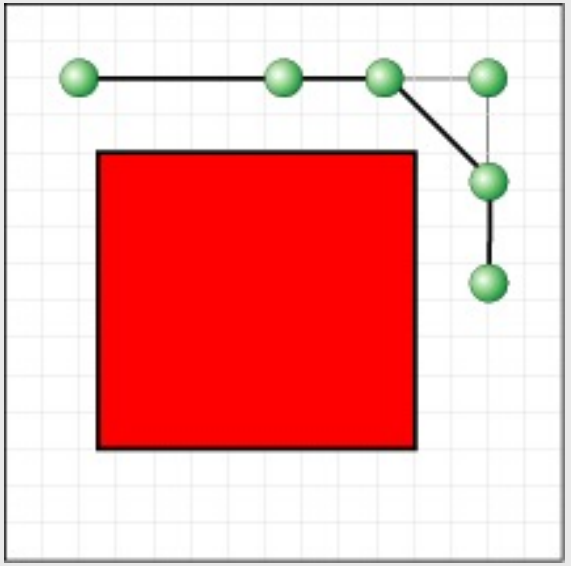


Operation DelEqual (alt. Strategy)

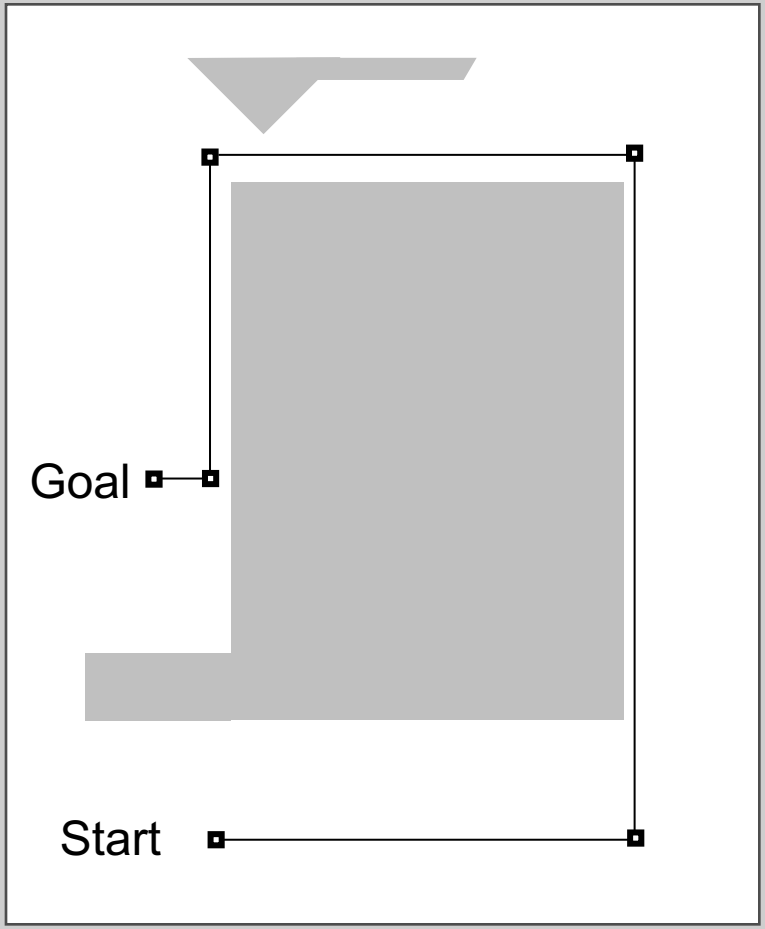
- DelEqual
  - Generate new point on longer segment, with length of shorter segment



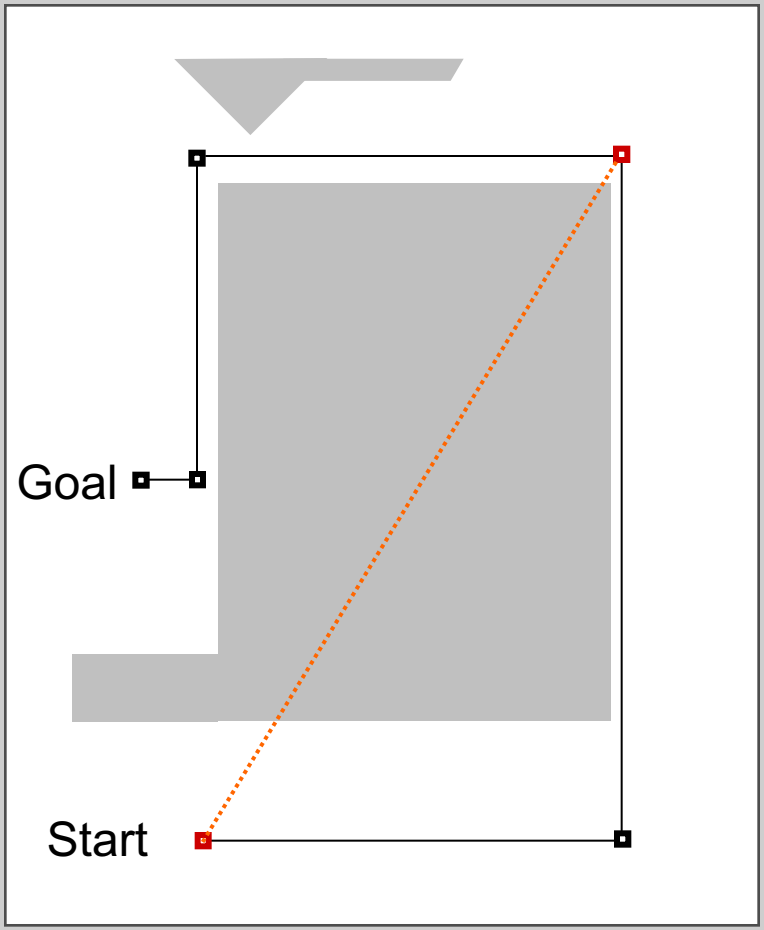




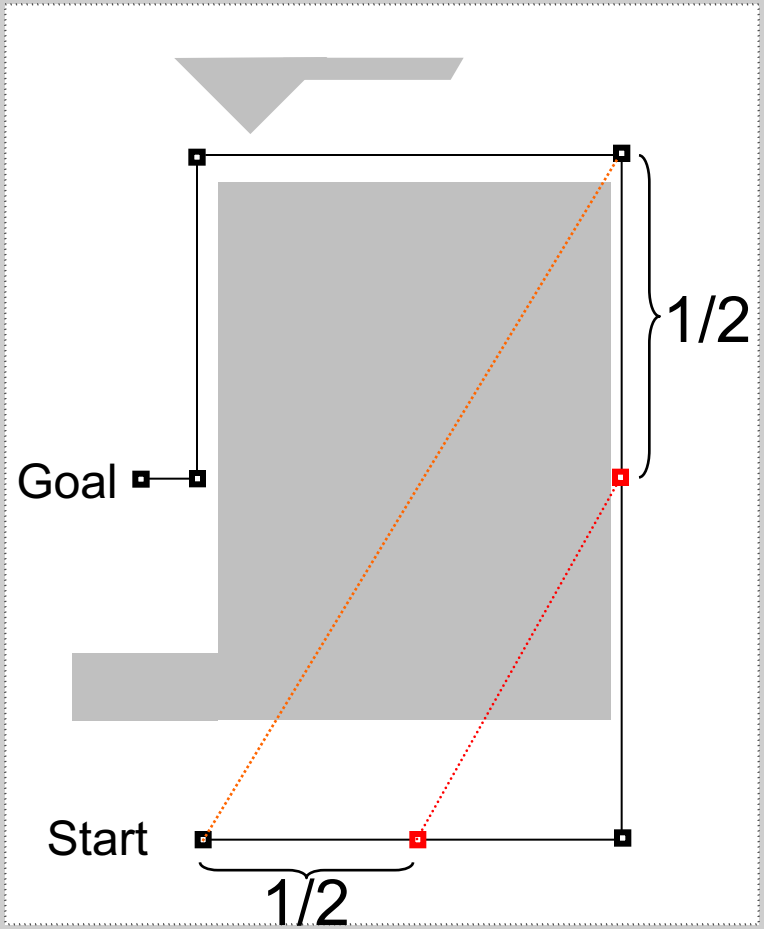
Example for a whole path



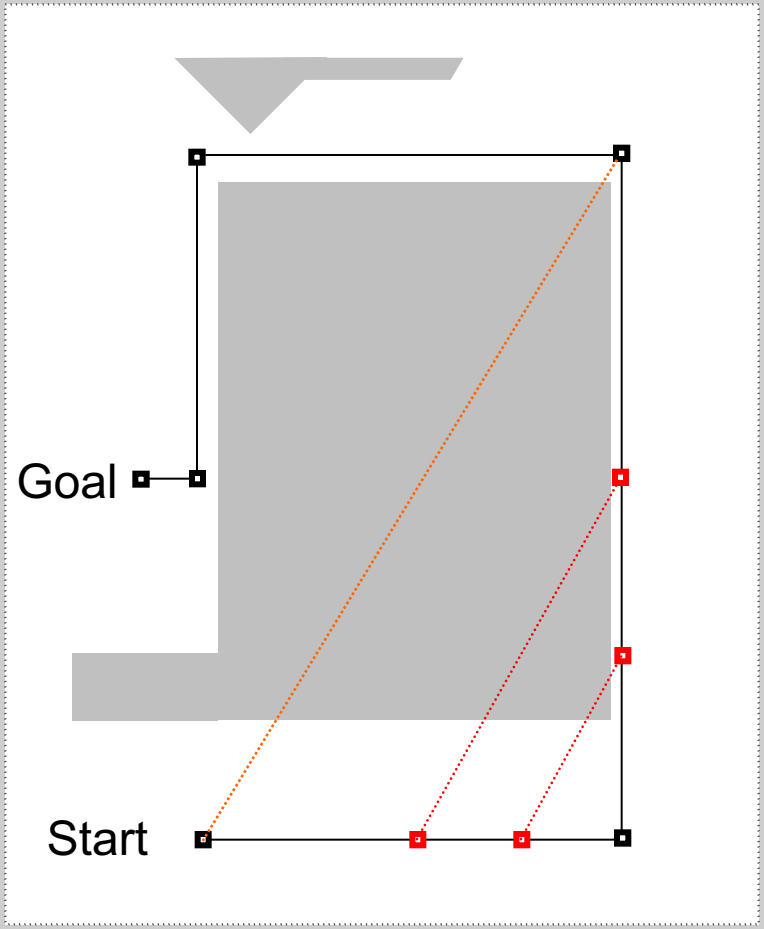
Example for a whole path



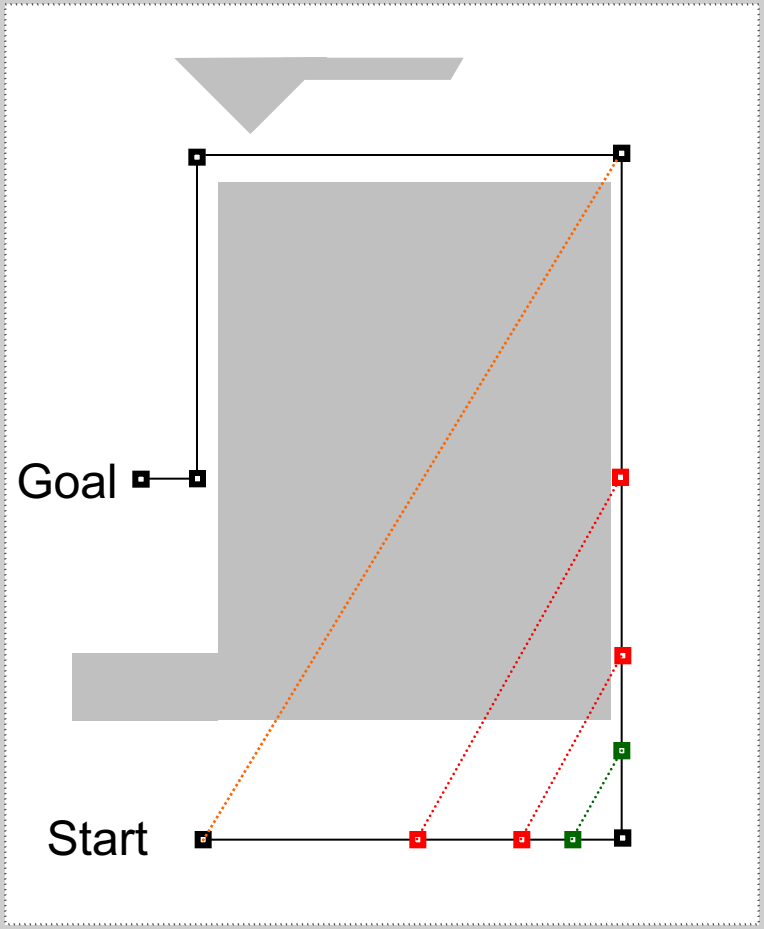
Example for a whole path



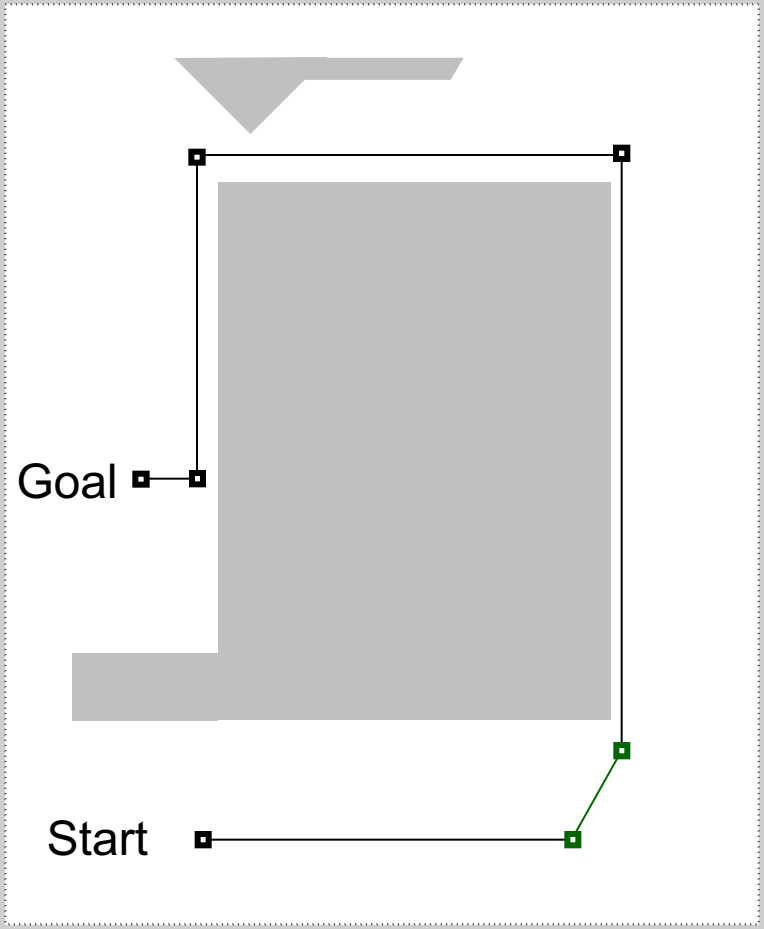
Example for a whole path



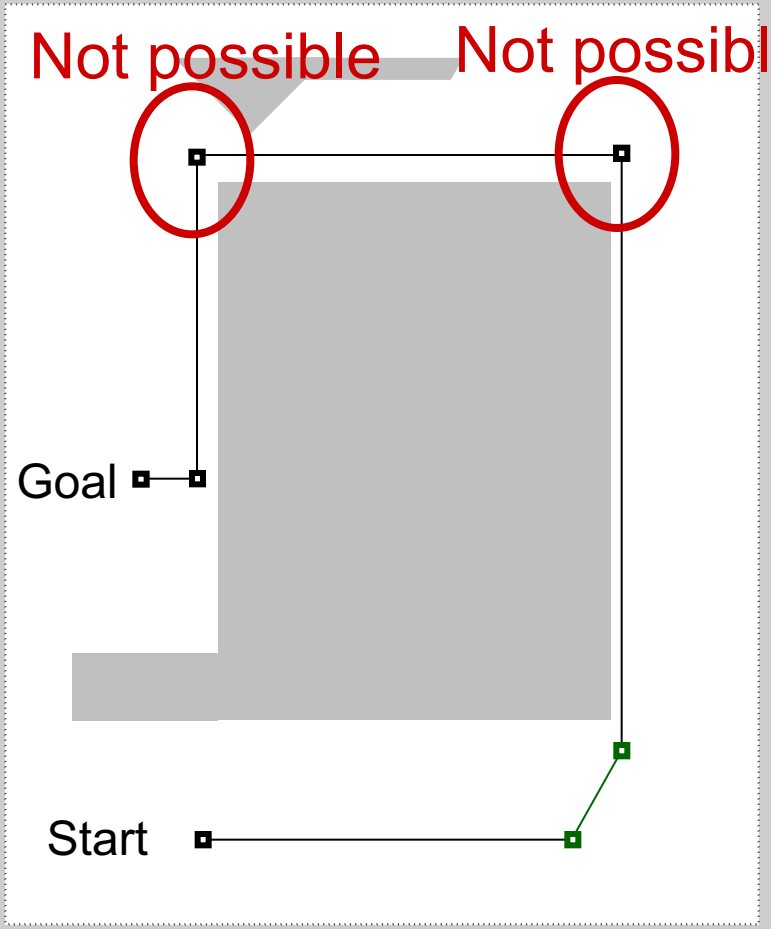
Example for a whole path



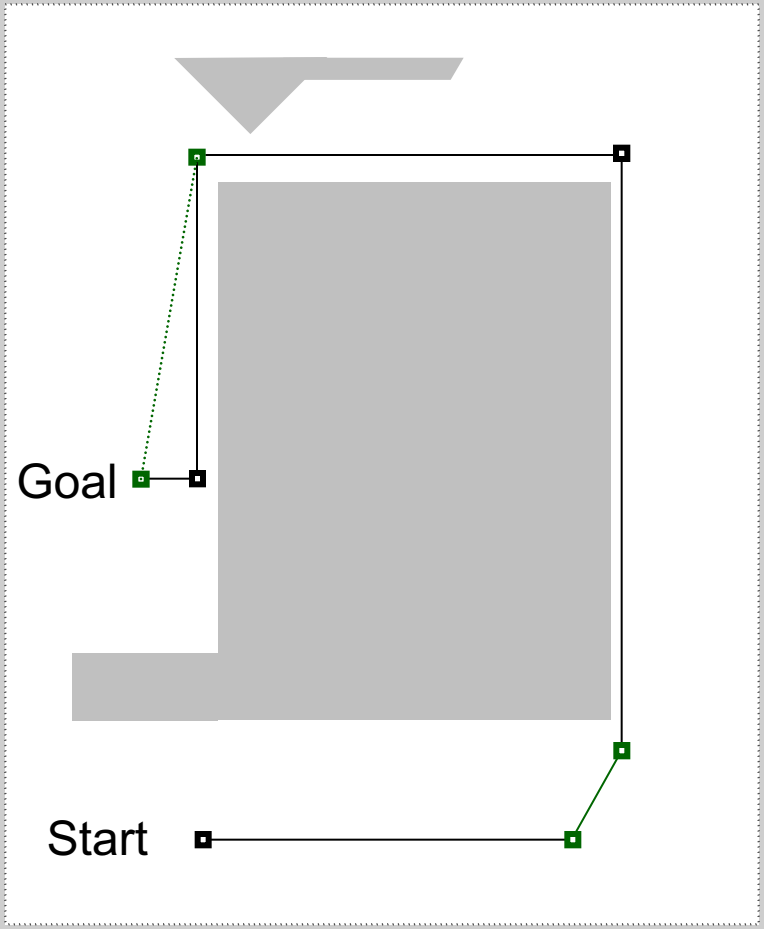
Example for a whole path



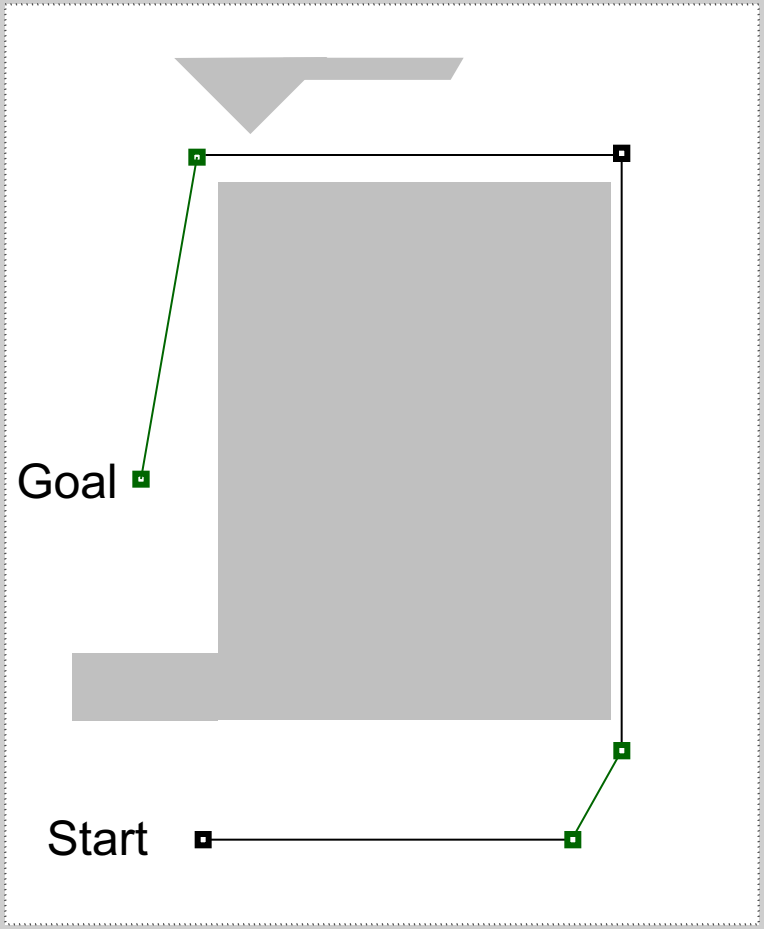
Example for a whole path



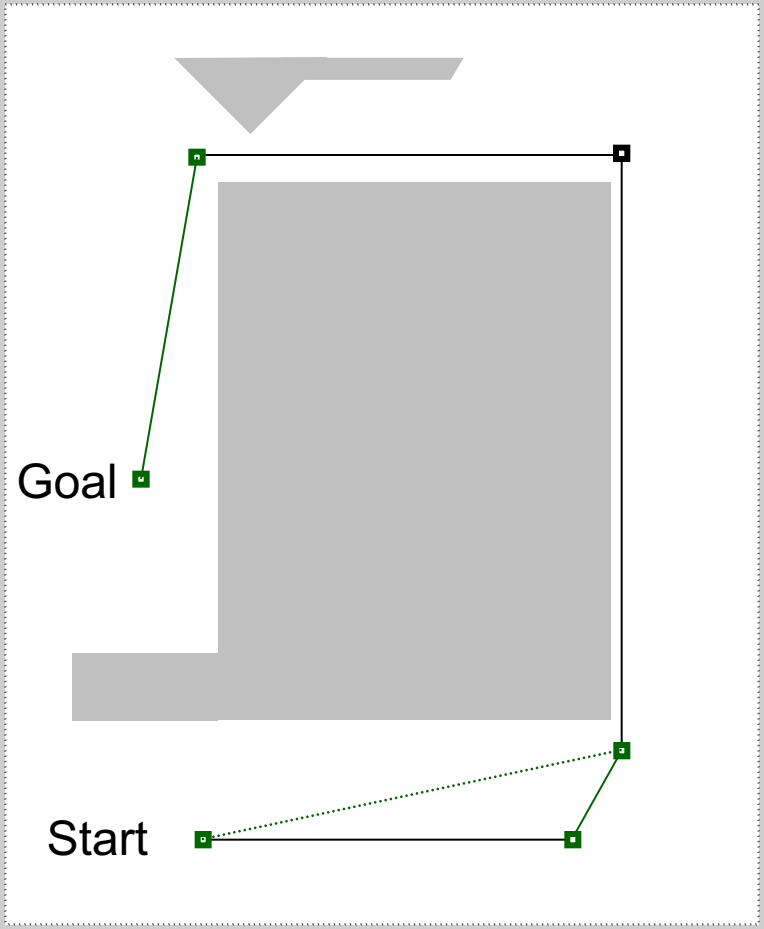
Example for a whole path



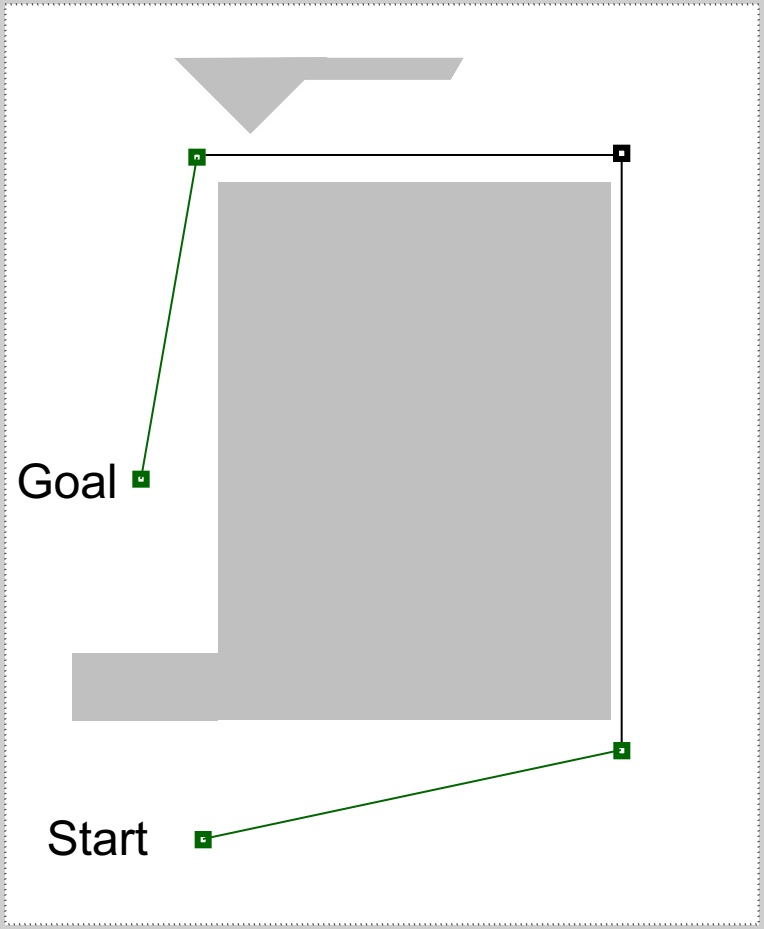
Example for a whole path



Example for a whole path

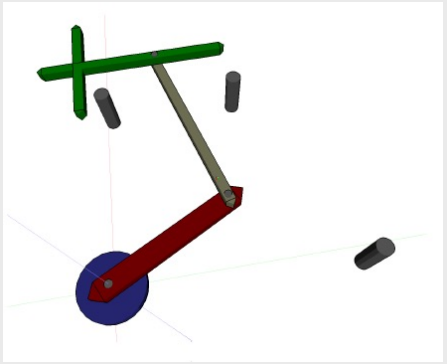


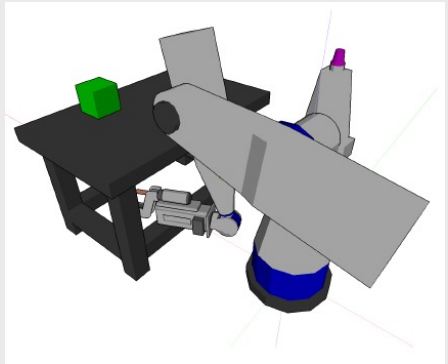
Example for a whole path : Result

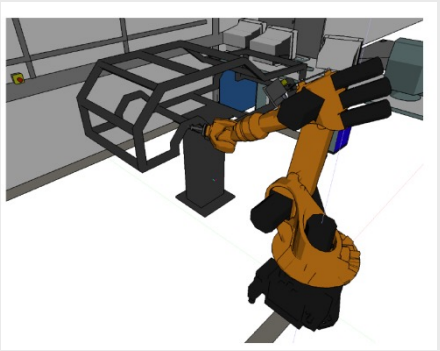


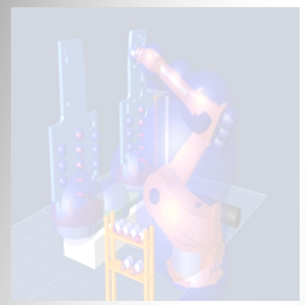
Comparison of Smoothing strategies

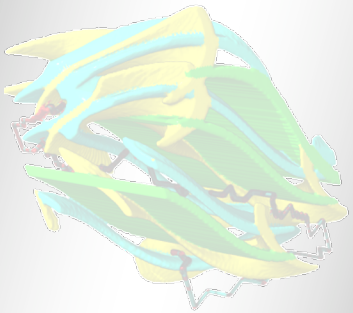
Test Environments

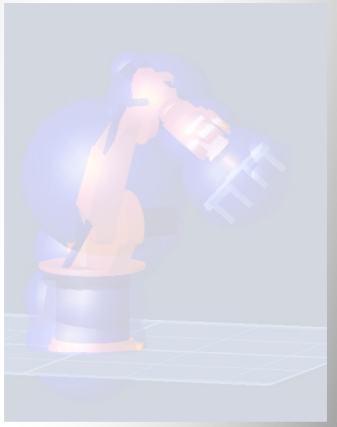








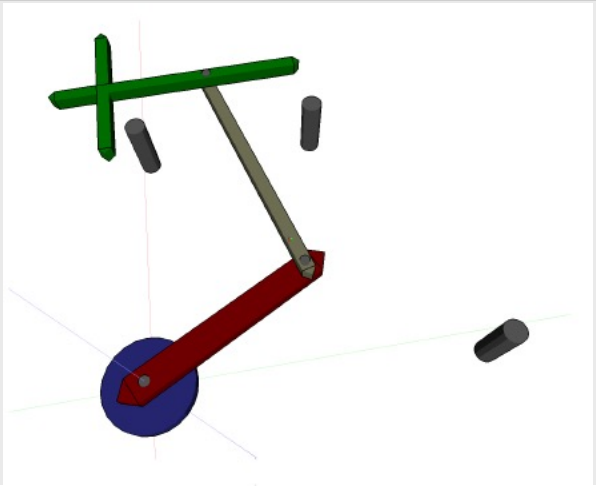


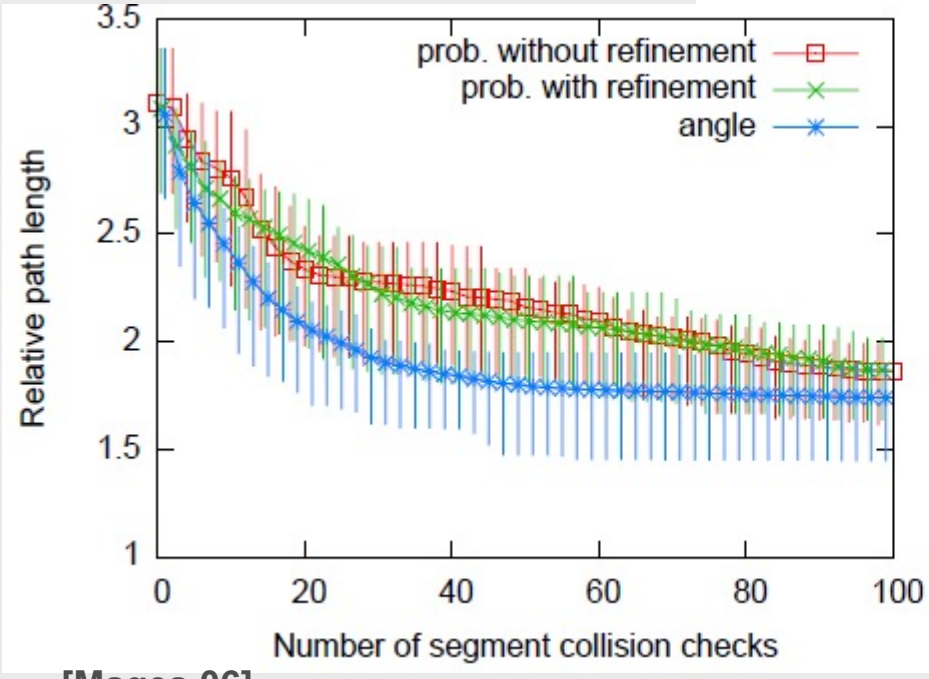


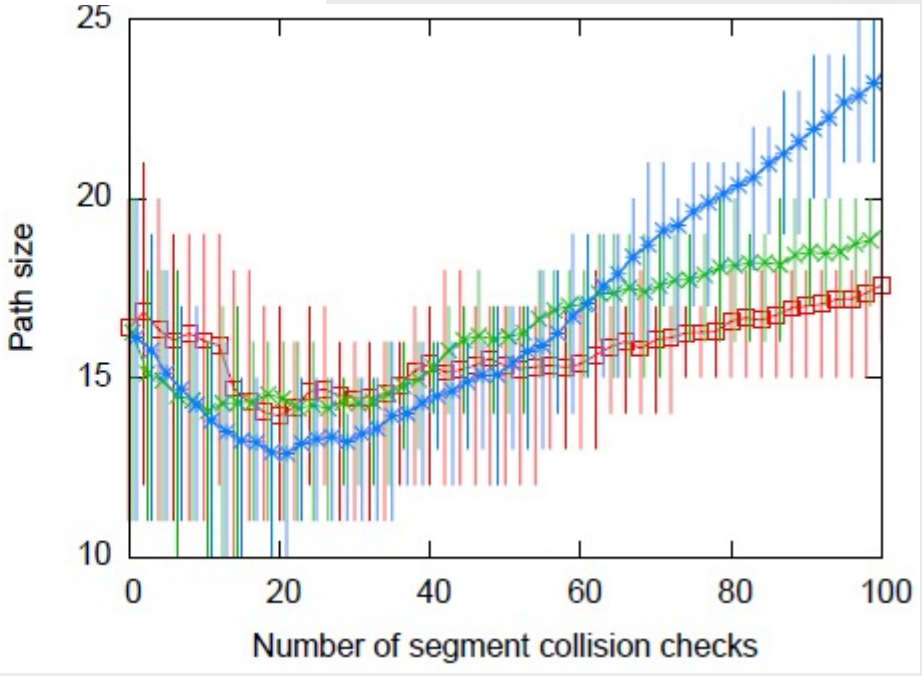
Comparison of different smoothing strategies

- Strategies:
  - Probabilistic
  - Probabilistic extended (Latombe)
  - Deterministic based on angle (Bechthold & Glavina)
- Tests for every test environment:
  - 3 different tasks executed 10 times
  - 3 different planer (k-Closest, Lazy, RRT)
- 90 experiments -> 90 executing pathplanner calls
- Calling 100 times smoothing algorithm
- Path quality: $\left(\frac{l_{\text{experiment}}}{\min(l)}\right)$

Smoothing Comparison: Benchmark I

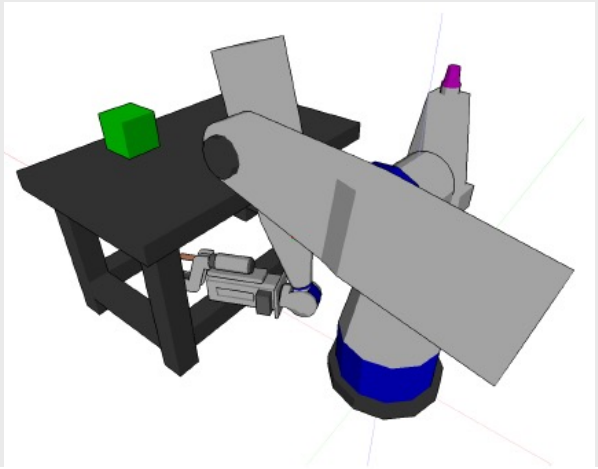


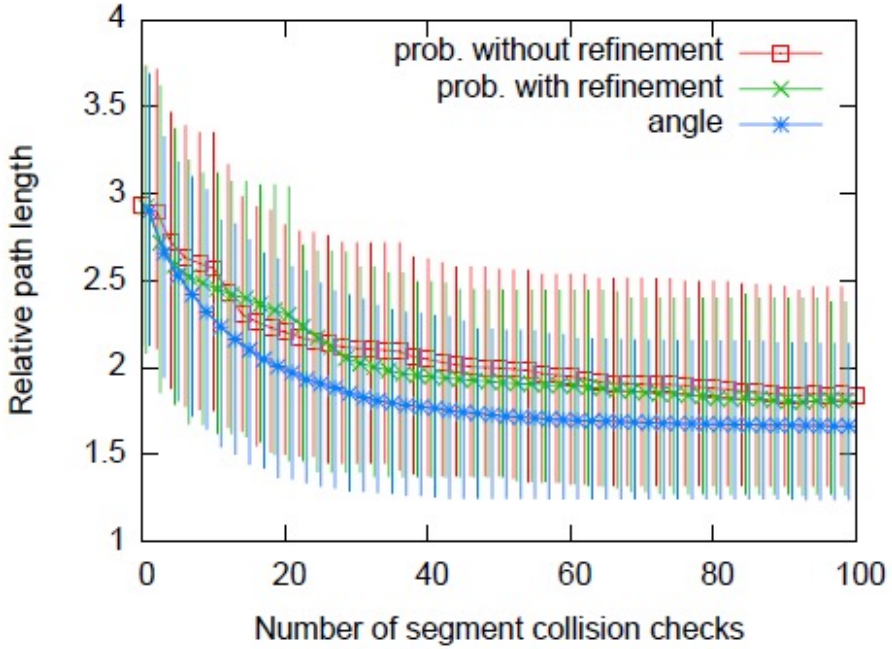


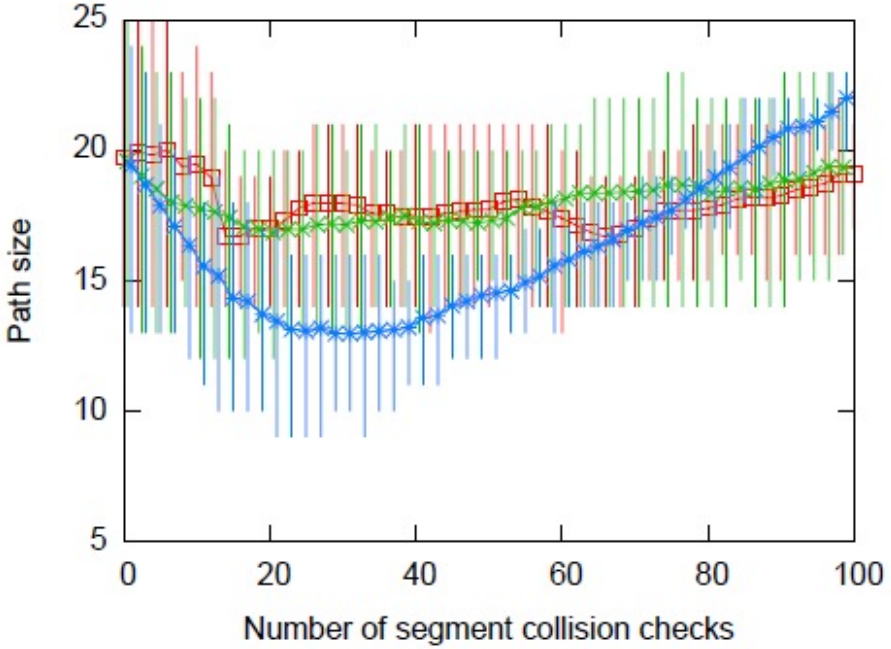


[Mages 06]

Smoothing Comparison: Benchmark II

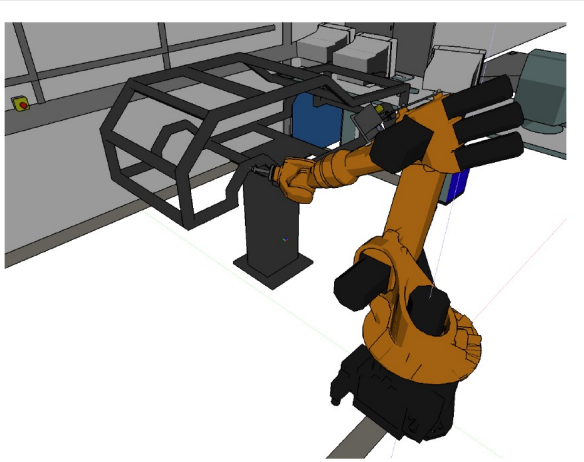


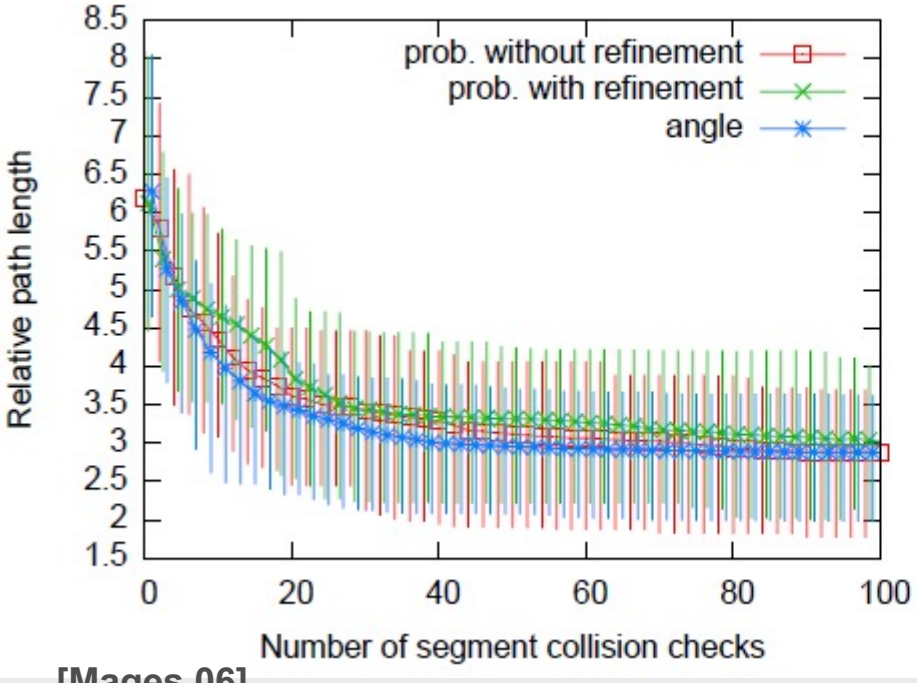


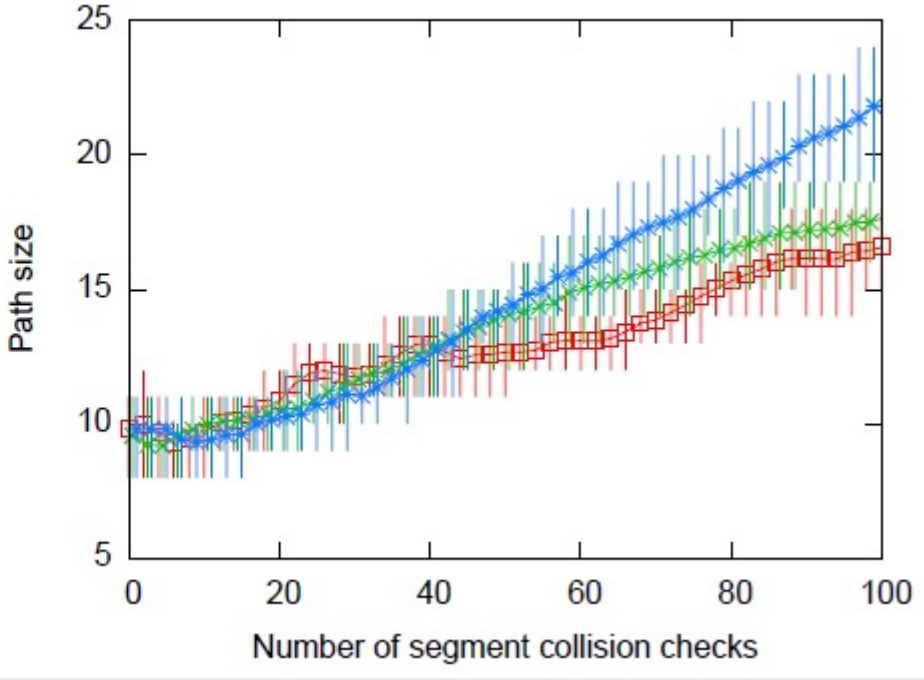


[Mages 06]

Smoothing Comparison: Benchmark III

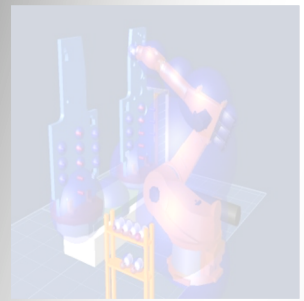


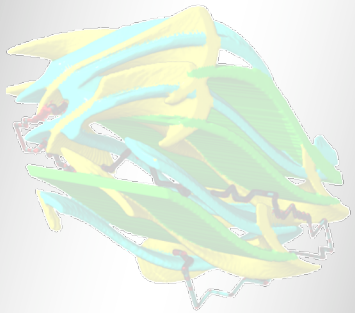


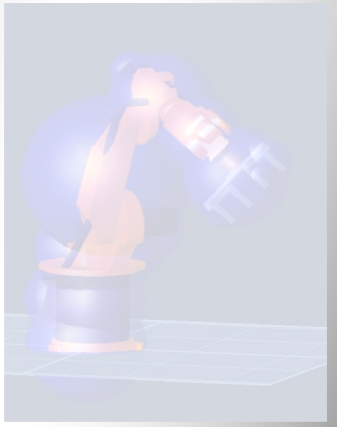


[Mages 06]

Termination of smoothing?







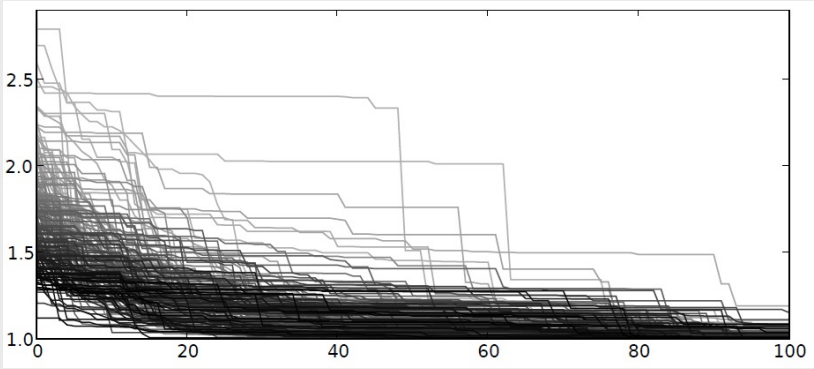
Basic smoothing Algorithm with extended termination

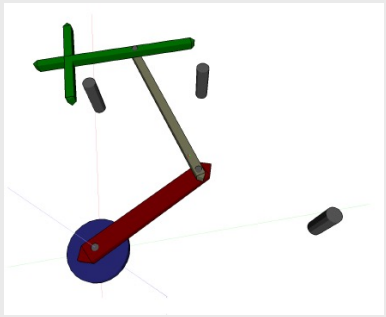
Glätte(p):

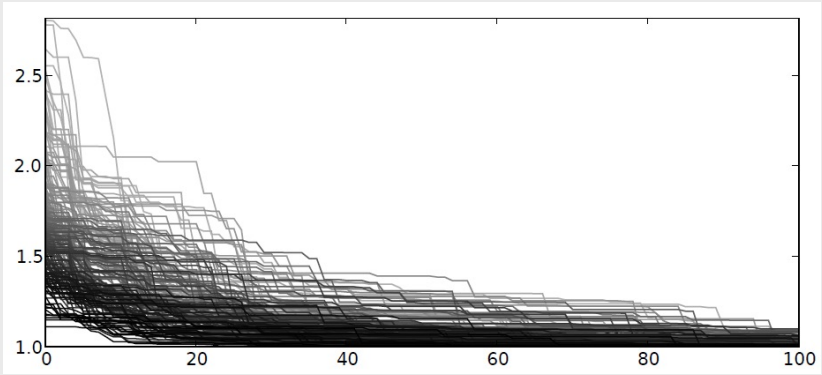
Eingabe : Pfad p bestehend aus einer geordneten Menge von Stützpunkten

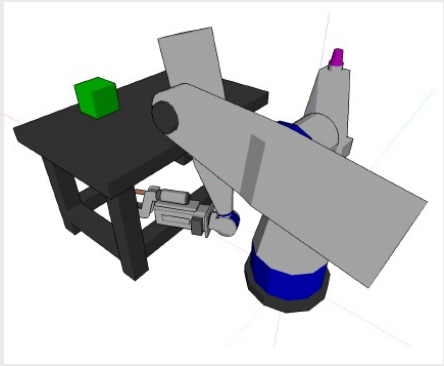
```
// Bestimme ein Pfadsegment
(l_A, l_B) ⟵ WählePositionen(p, wahr)

solange ? tue
    erfolg ⟵ OptimierStrategie(l_A, l_B, p)
    (l_A, l_B) ⟵ WähleAbkürzung(p, l_A, l_B, erfolg)
    wenn TERMINIERE(p) dann
        zurück ;
```

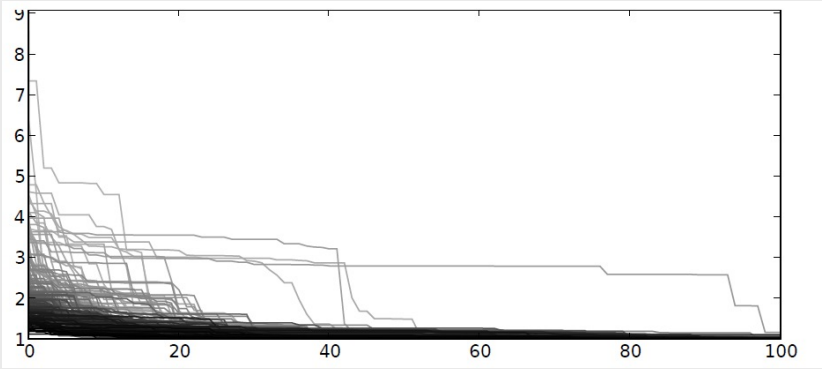
Approximation Behavior: 100 steps

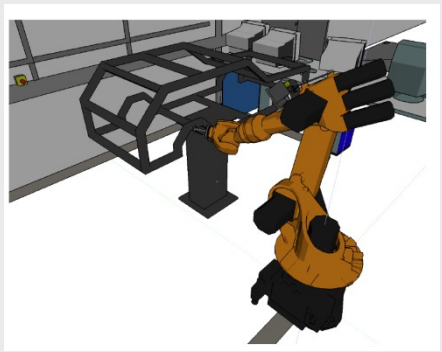












Termination strategy

Daten : Pfad p ist eine sortierte Menge von Stützpunkten
Daten : Schlange q

$l = p.\texttt{LÄNGE}();$
$q.\texttt{PUSH}(l);$
wenn $q.\mathit{GRÖSSE}() < s_{max}$ **dann zurück** $falsch;$

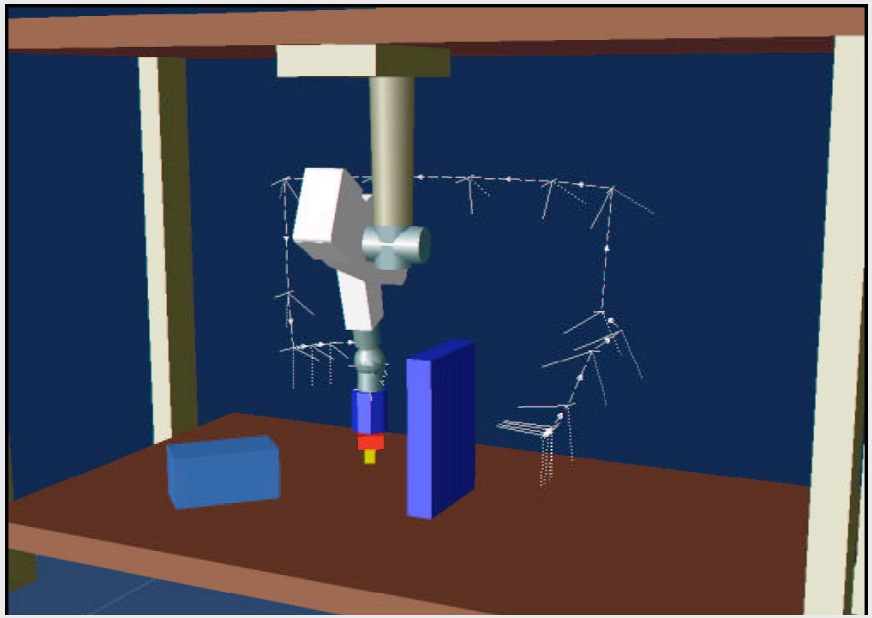
$q.\texttt{POP}();$ ⟵ Älteste Element entfernen
wenn $q.\mathit{VARIANZ}() < v_{min}$ **dann zurück** $wahr;$
zurück $falsch;$

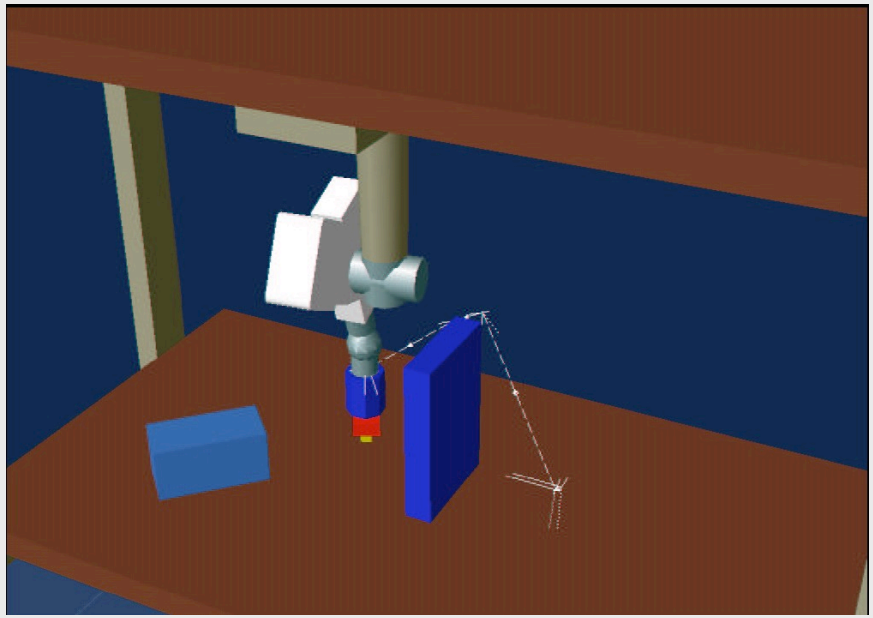
Conclusion

Observations:

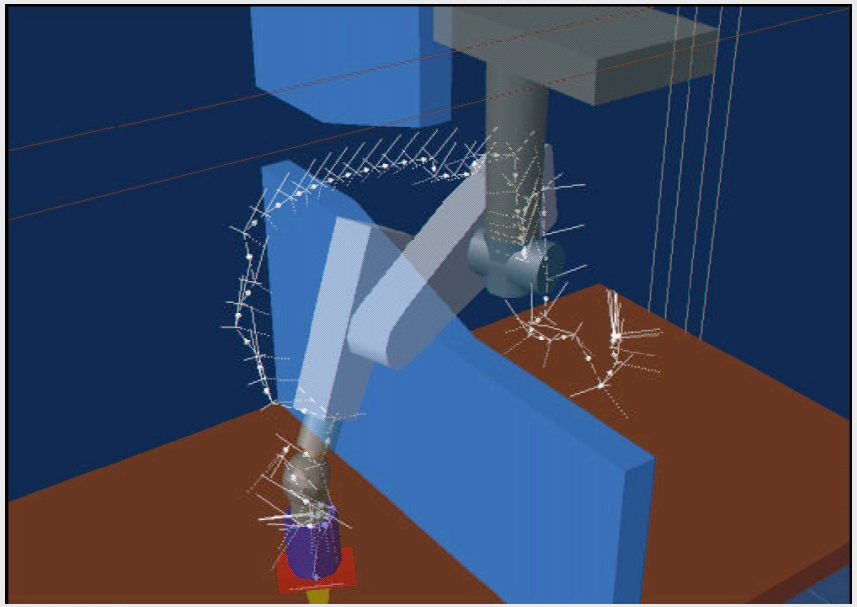
- All three strategies tend to same path quality
- Deterministic approach tends to be converge faster
- Variance of deterministic approach smaller during optimisation

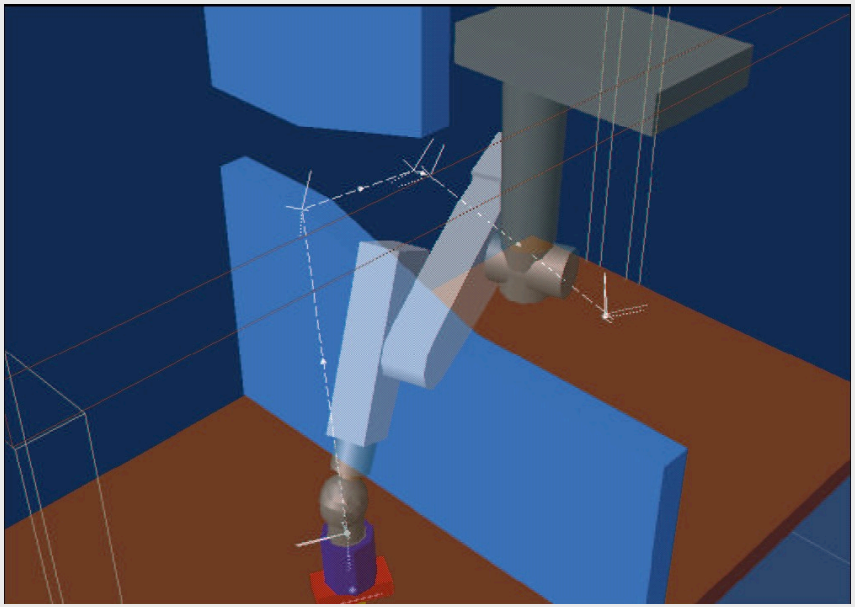
6-DOF-Example: Simple



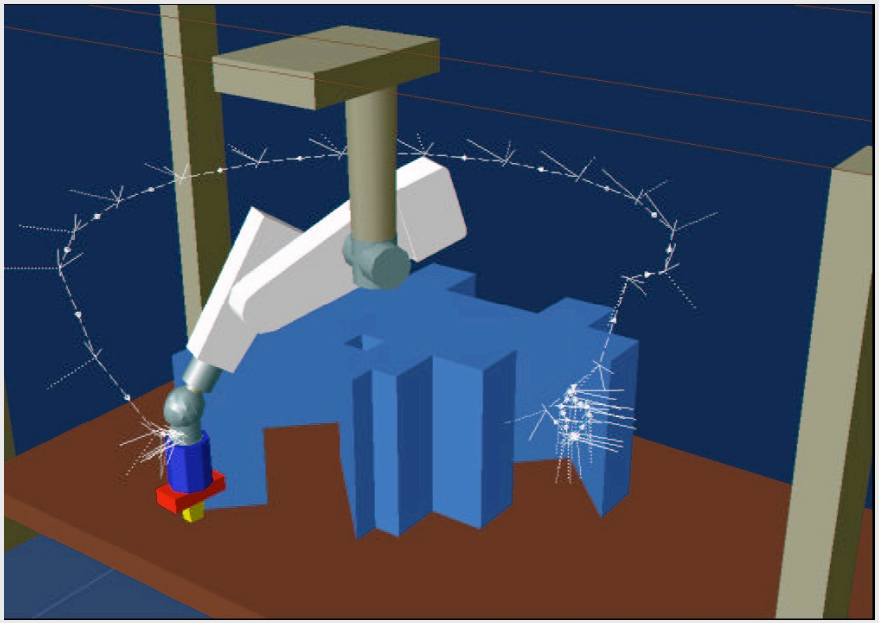


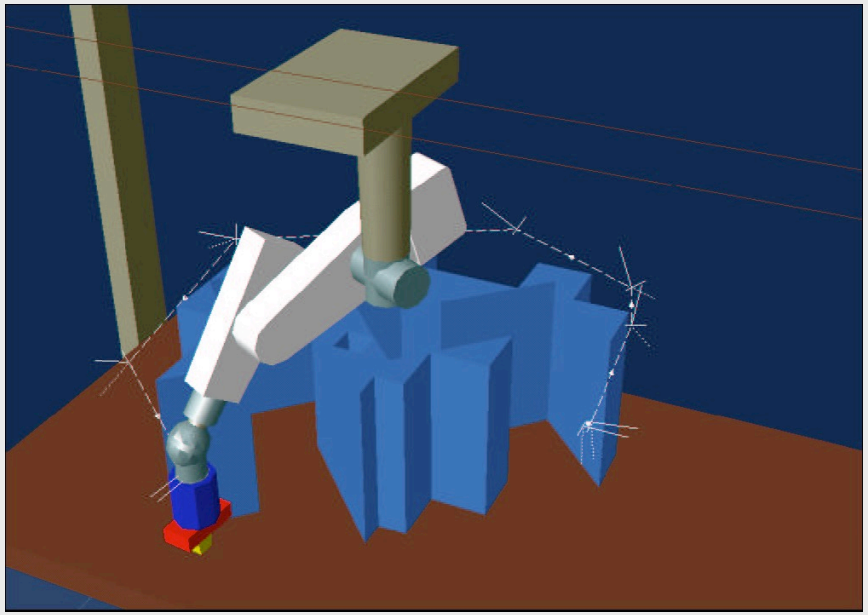
6-DOF-Example: Bottleneck



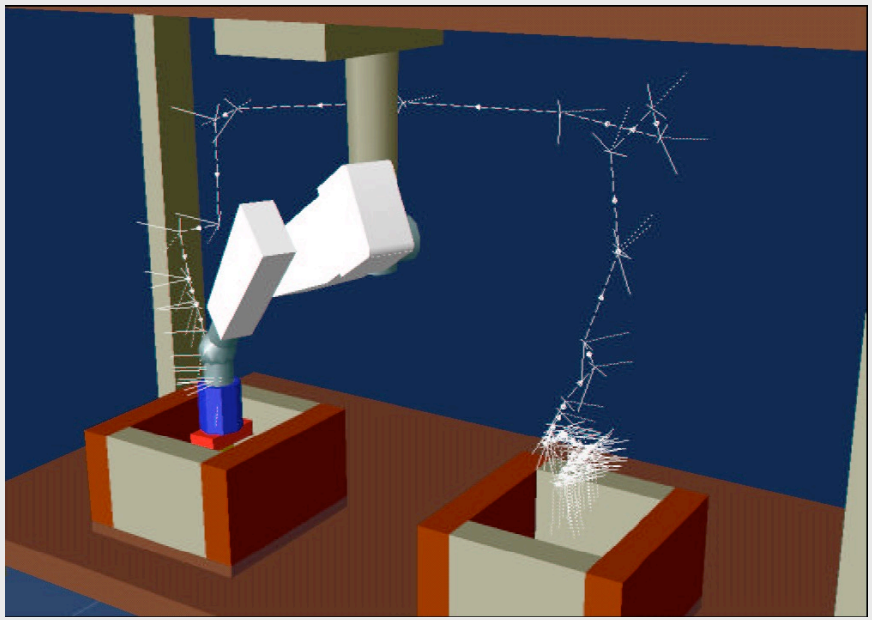


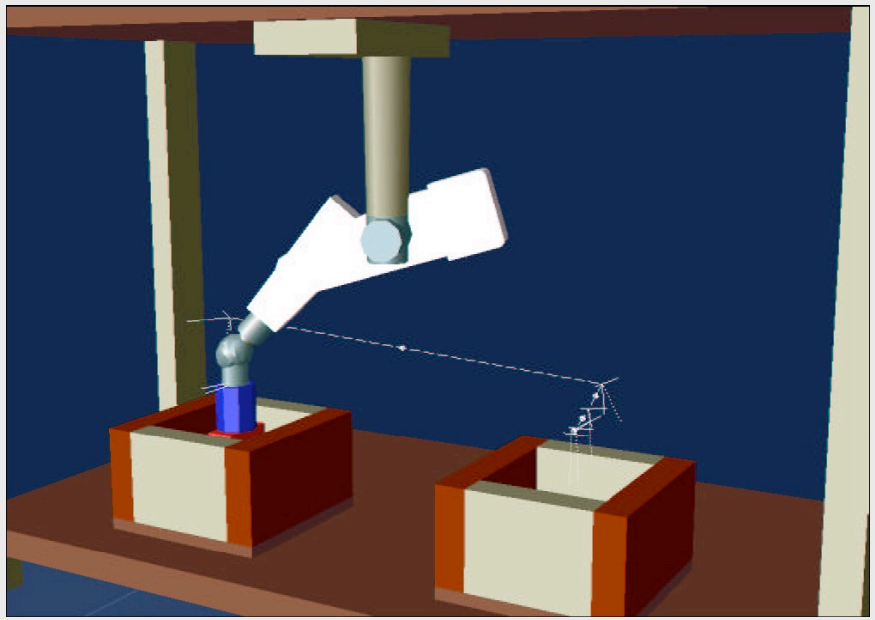
6-DOF-Example: Detour



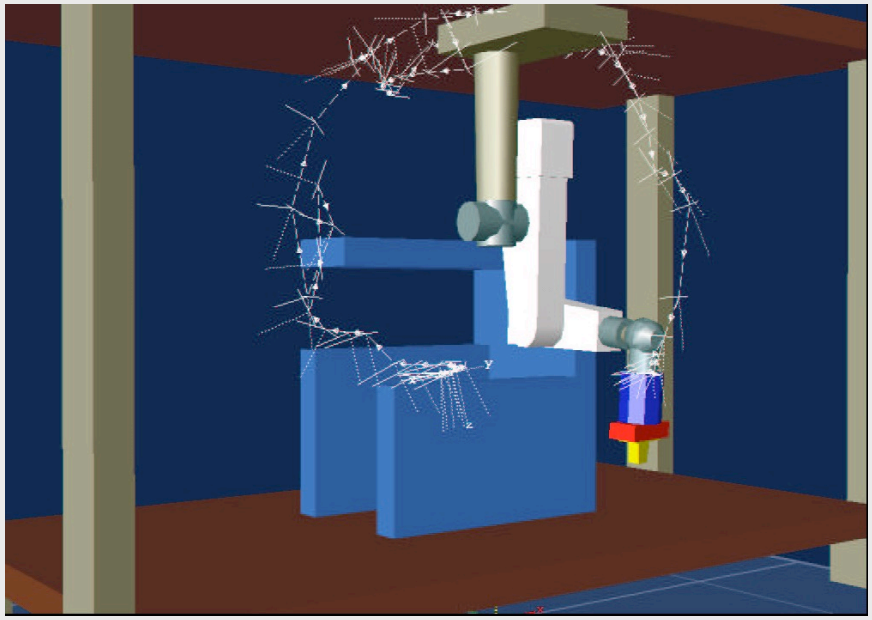


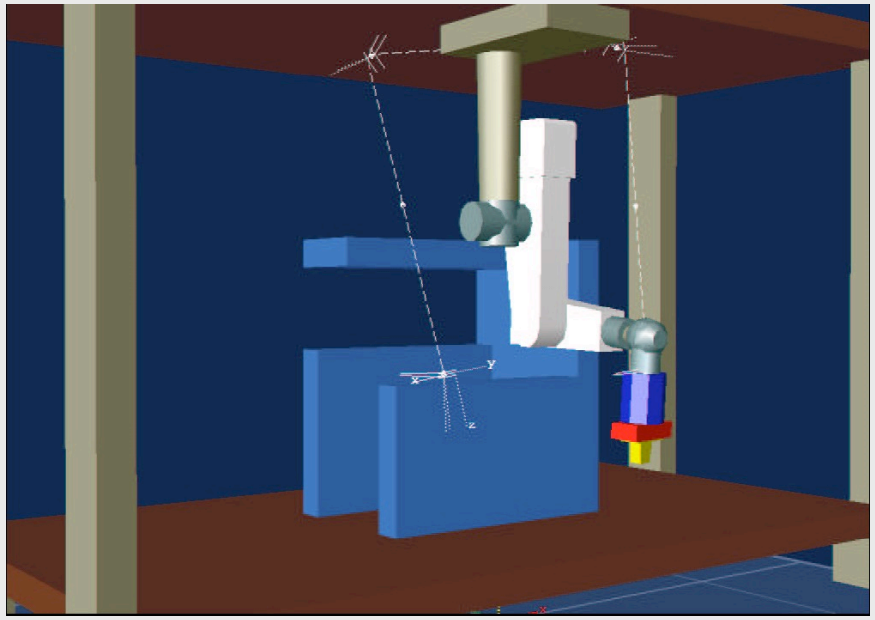
6-DOF-Example: Star



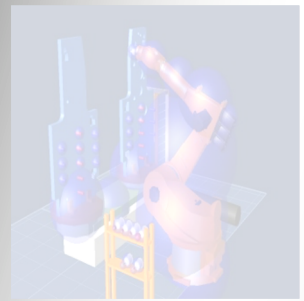


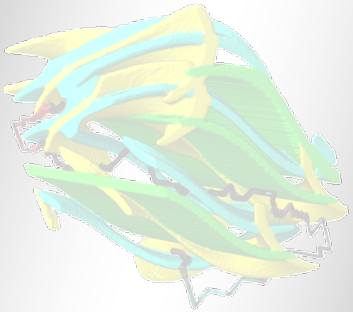
6-DOF-Example: Trap

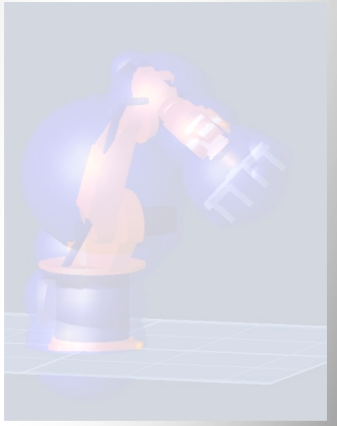




Optimisation







Result of Smoothing

Positive :-)

- movements are combined as much as possible
- path is shortest around obstacles (tightened like a rubber band)

"Not so good :-("

- robot must make a full stop at every path point.
  - Only then the path can guaranteed to be collision-free.

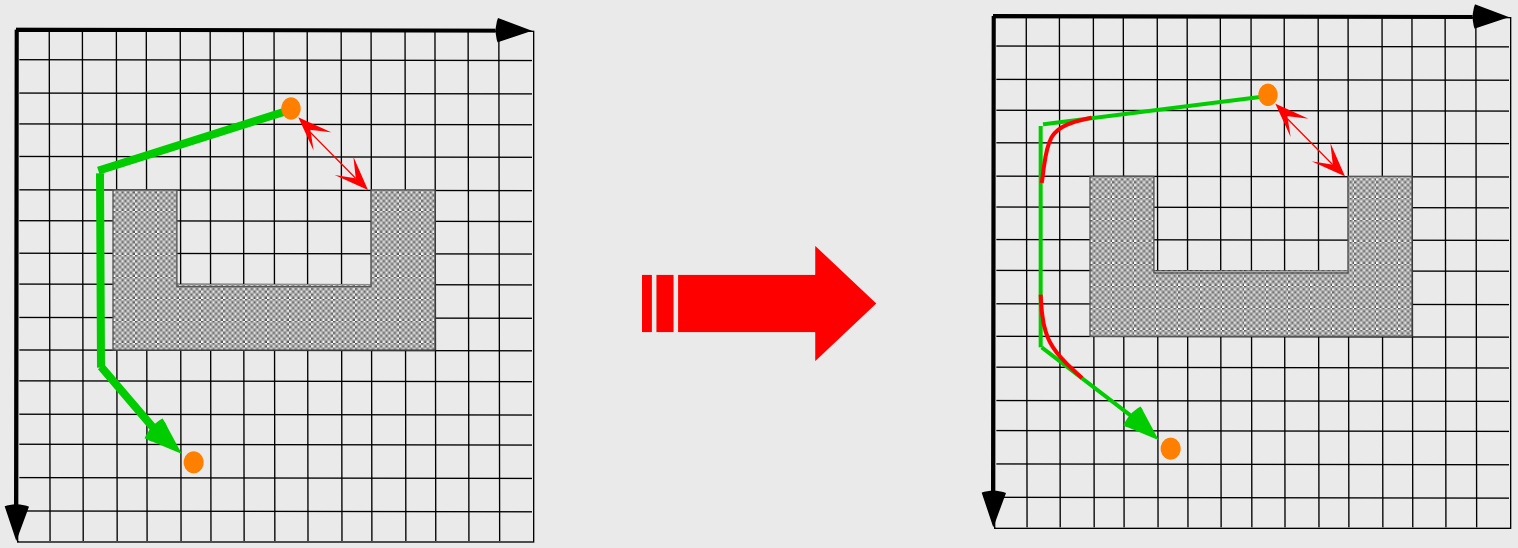
Optimisation

GOAL:

- Faster execution, without stopping
- more harmonious movement

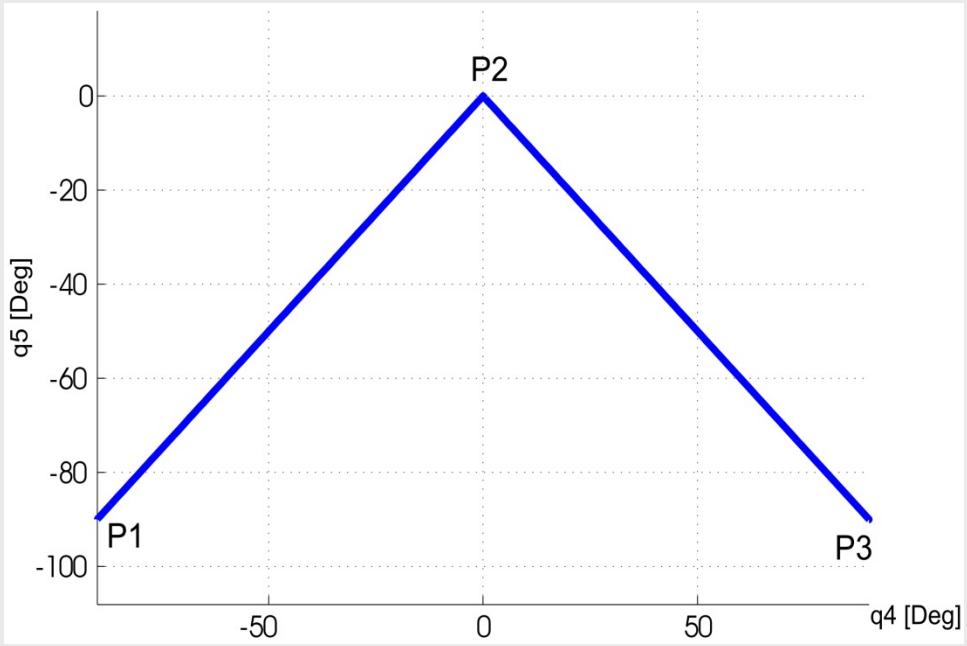
How?

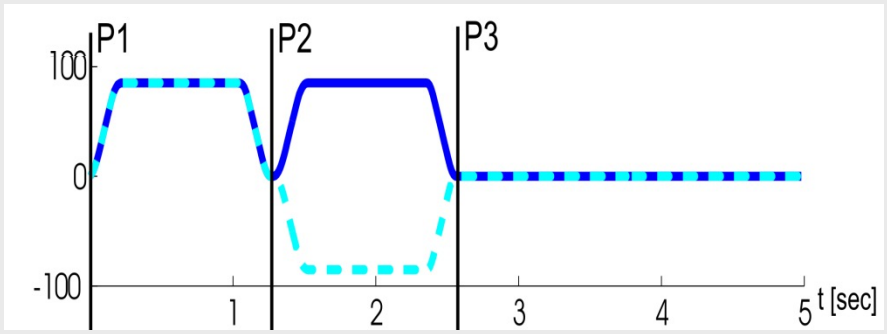
- using PTP motion command with rounding



Simple PTP-Command

- Movement stops at every point

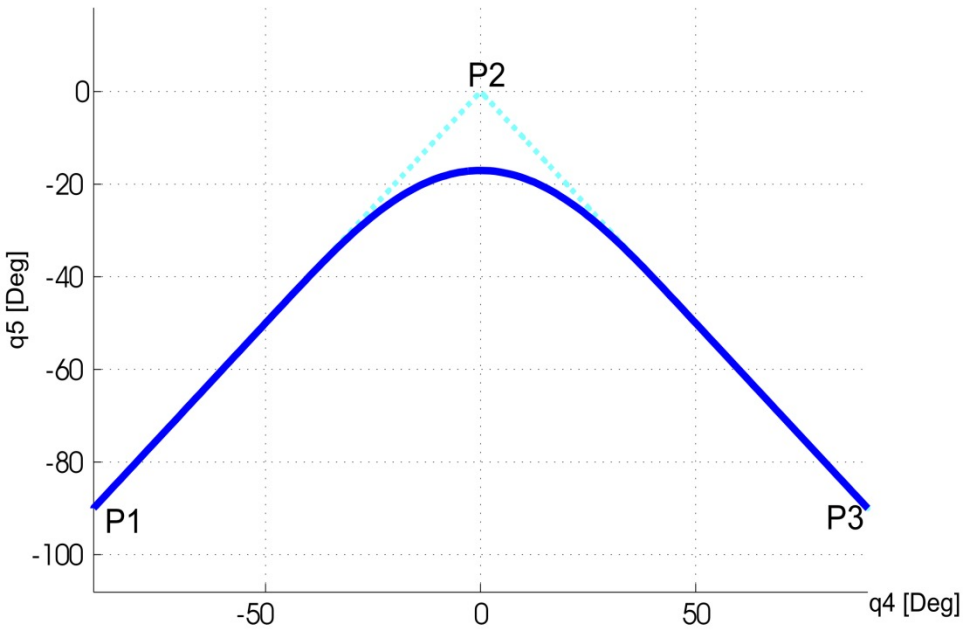


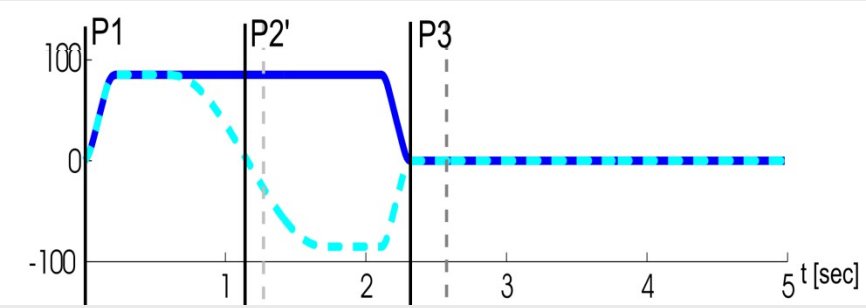



```
PTP(A1 0, A2 90, A3 -90 , A4 0, A5 -90, A6 -90) /* Start point */
PTP_REL(A5 90, A6 90) (Anmerkung: A5=q4)
PTP_REL(A5 90, A6 -90)
```

PTP-Command using Rounding

- Movement doesn‘t stop at point P2 (Rounding)






```
PTP(A1 0, A2 90, A3 -90 , A4 0, A5 -90, A6 -90) /* Start point */
PTP_REL(A5 90, A6 90) C_PTP /* Factor is given by e.g. APO.CPTP=0.1 */
PTP_REL(A5 90, A6 -90)
```

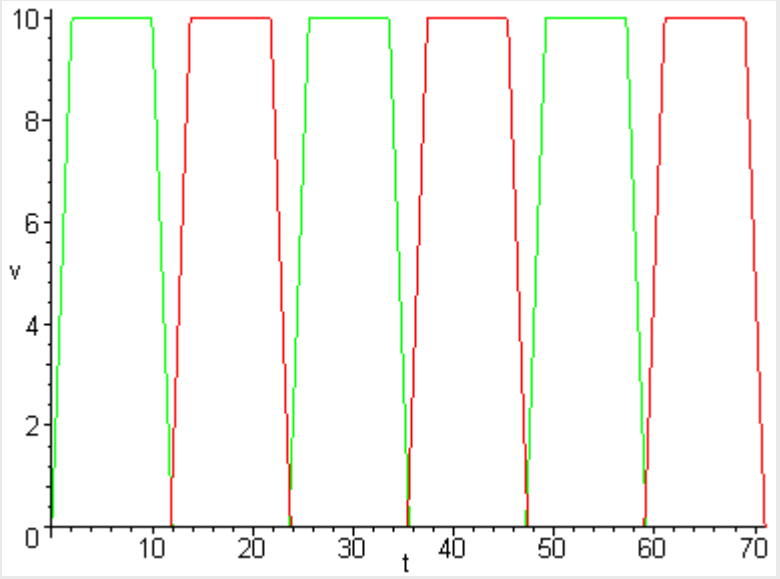
Rounding

Next movement begins, before the actual one is finished.

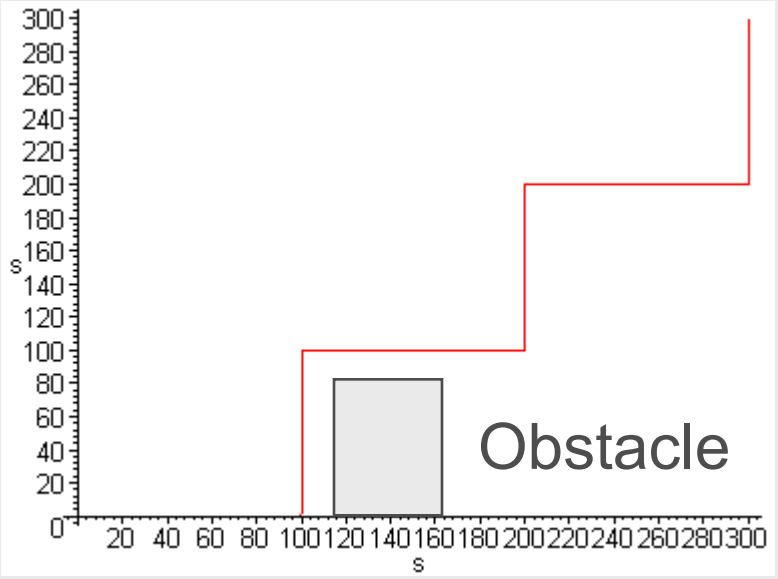
*R*ounding factor *r* (in percent) determines how much the two movements are overlapping.

Rounding examples

$r = 0\%$



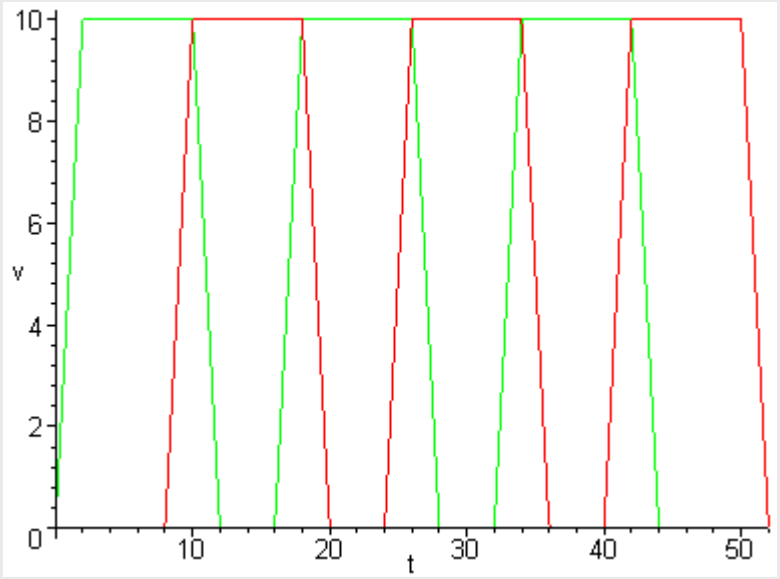
Velocity profiles

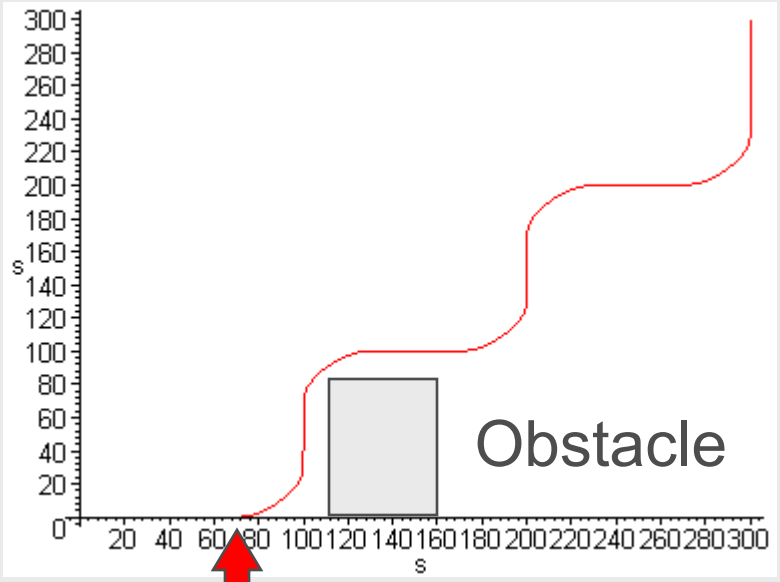


Movement in C-space

Rounding examples

r = 30%

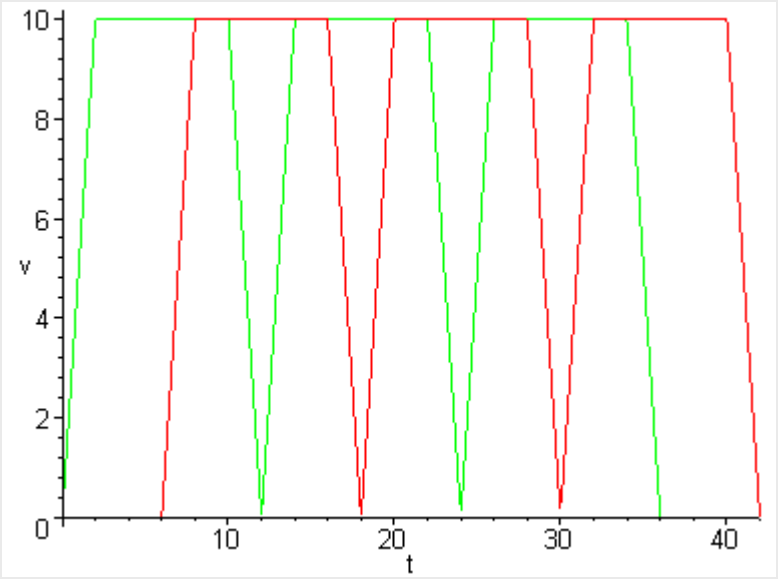


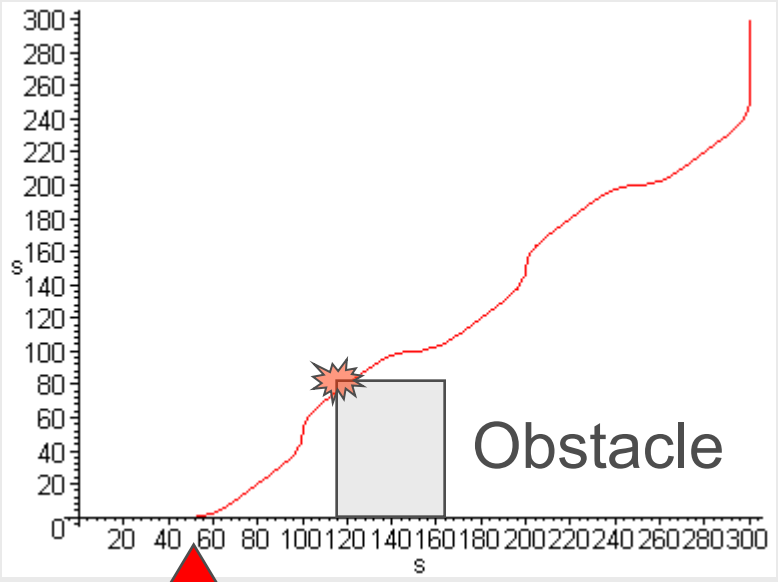


Start of rounding

Rounding examples

$r = 50\%$





Start of rounding

Rounding: Task

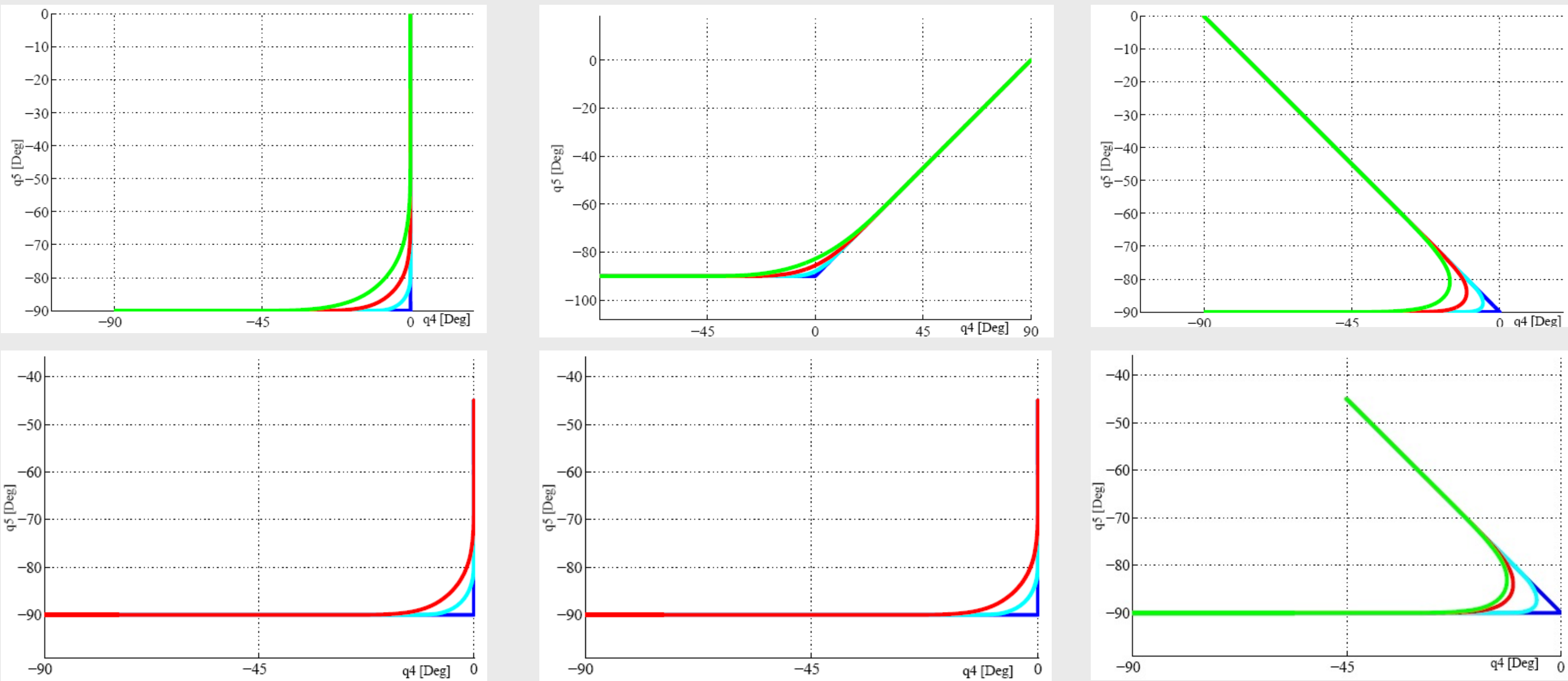
Modify path in such a way, that

in spite of using rounded movements

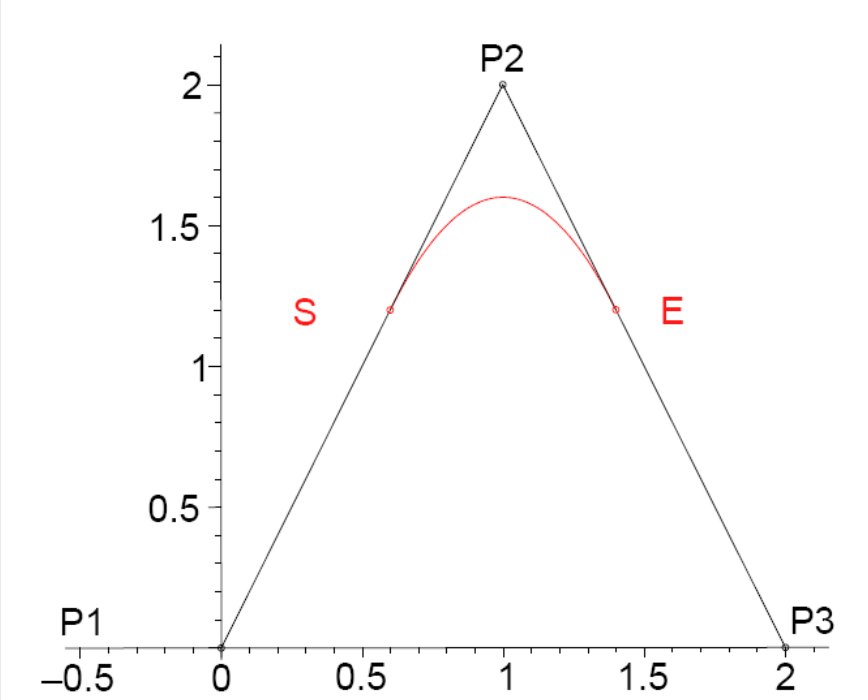
collisions are avoided

Rounding examples

- Using different paths and rounding factors

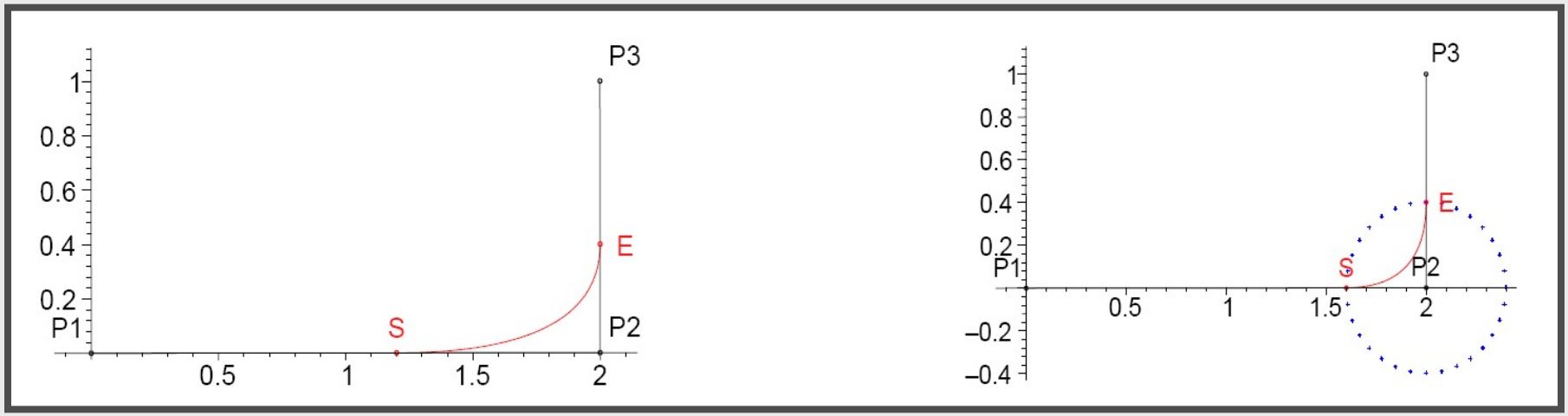


Model of rounding: first try

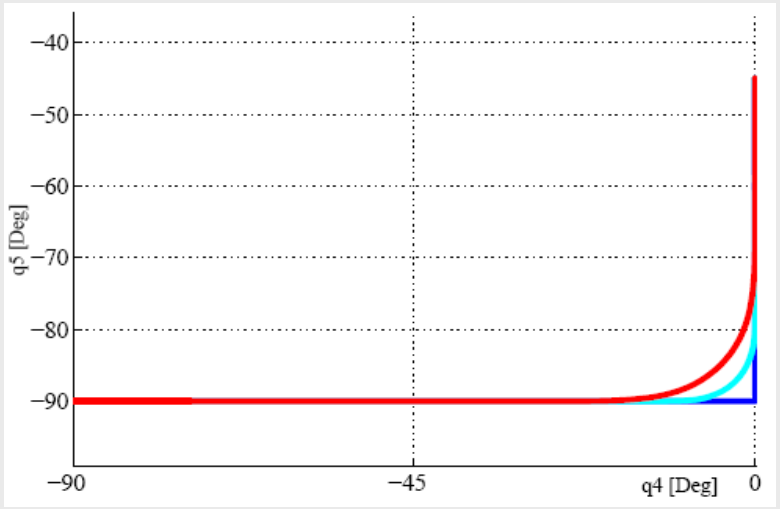


$$PA: \begin{cases} \text{Strecke}: & \overrightarrow{P_1S} \quad \text{mit} \quad S = r \cdot \overrightarrow{P_2P_1} + P_2 \\ \text{Parabel}: & r \cdot \overrightarrow{P_2P_1}\,(t-1)^2 + r \cdot \overrightarrow{P_2P_3}\,t^2 + P_2 \\ \text{Strecke}: & \overrightarrow{EP_3} \quad \text{mit} \quad E = r \cdot \overrightarrow{P_2P_3} + P_2 \end{cases}$$

First try doesn't match ☹



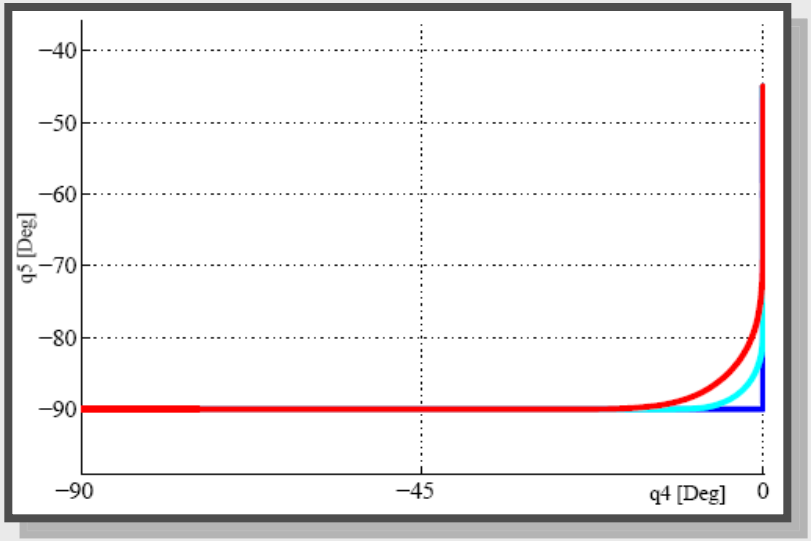
- On the left: „simple model" doesn't represent reality if segments differ in lengths

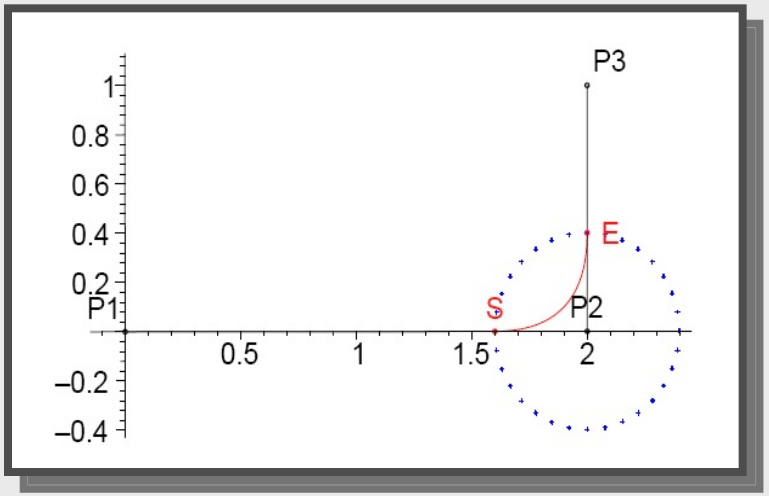


Second model: take care of shortest segment

- Rounding „parabol“:

$$\vec{z}(t) = \begin{cases} r \cdot \overrightarrow{P_2P_1}\,(t-1)^2 + r \cdot \dfrac{|\overrightarrow{P_2P_1}|}{|\overrightarrow{P_2P_3}|}\overrightarrow{P_2P_3}\,t^2 + P_2 & \text{wenn } |\overrightarrow{P_2P_3}| \geq |\overrightarrow{P_2P_1}|, \\ r \cdot \dfrac{|\overrightarrow{P_2P_3}|}{|\overrightarrow{P_2P_1}|}\overrightarrow{P_2P_1}\,(t-1)^2 + r \cdot \overrightarrow{P_2P_3}\,t^2 + P_2 & \text{wenn } |\overrightarrow{P_2P_1}| \geq |\overrightarrow{P_2P_3}| \end{cases}$$

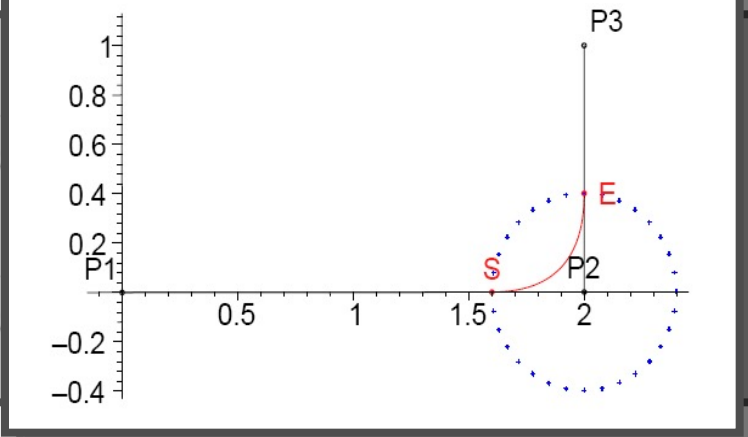




Second model: take care of shortest segment

- Rounding „parabol“:

$$\vec{z}(t) = \begin{cases} r \cdot \overrightarrow{P_2P_1}\,(t-1)^2 + r \cdot \dfrac{|\overrightarrow{P_2P_1}|}{|\overrightarrow{P_2P_3}|}\overrightarrow{P_2P_3} \\ r \cdot \dfrac{|\overrightarrow{P_2P_3}|}{|\overrightarrow{P_2P_1}|}\overrightarrow{P_2P_1}\,(t-1)^2 + r \cdot \overrightarrow{P_2P_3} \end{cases}$$

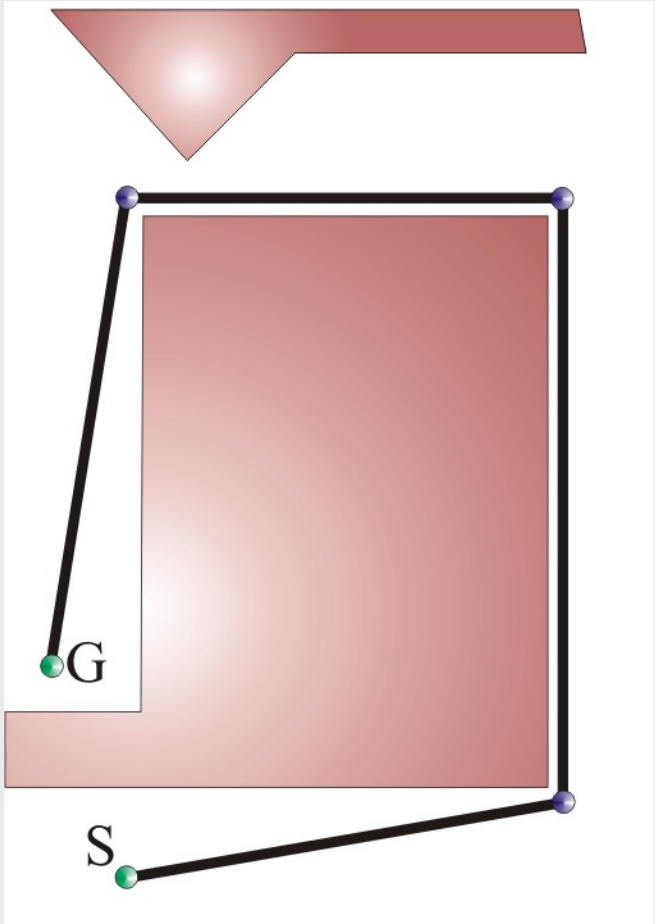


- Start and Endpoint:

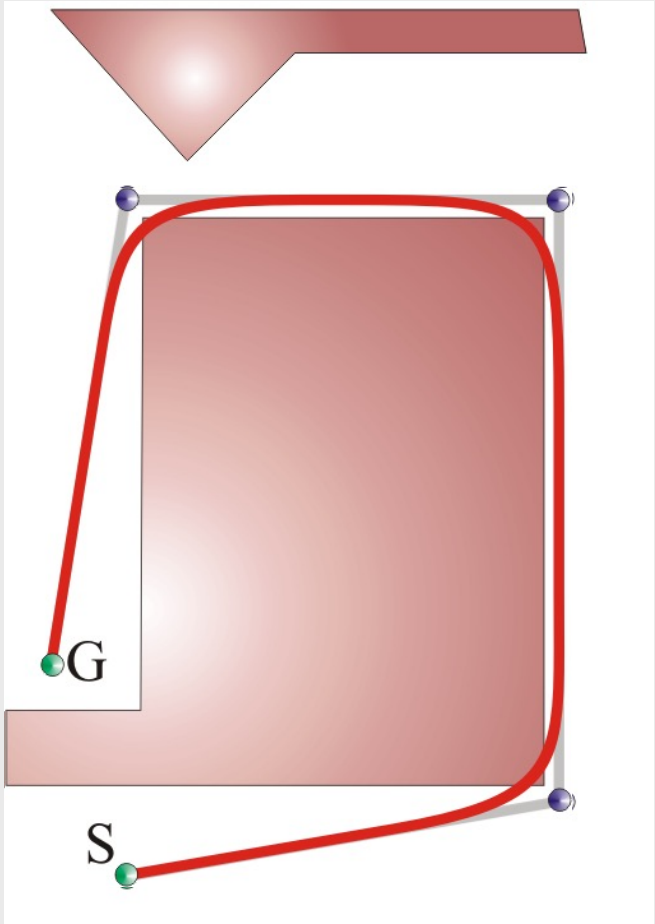
$$P_1S \quad \text{mit} \quad \begin{cases} S = r \cdot \overrightarrow{P_2P_1} + P_2 & \text{wenn } |\overrightarrow{P_2P_1}| < |\overrightarrow{P_2P_3}| \\ S = r \cdot \dfrac{\overrightarrow{P_1P_2}|\overrightarrow{P_2P_3}|}{|\overrightarrow{P_1P_2}|} + P_2 & \text{wenn } |\overrightarrow{P_2P_1}| \geq |\overrightarrow{P_2P_3}| \end{cases}$$

$$EP_3 \quad \text{mit} \quad \begin{cases} E = r \cdot \overrightarrow{P_2P_3} + P_2 & \text{wenn } |\overrightarrow{P_2P_3}| < |\overrightarrow{P_1P_2}| \\ E = r \cdot \dfrac{\overrightarrow{P_2P_3}|\overrightarrow{P_1P_2}|}{|\overrightarrow{P_2P_3}|} + P_2 & \text{wenn } |\overrightarrow{P_2P_3}| \geq |\overrightarrow{P_1P_2}| \end{cases}.$$

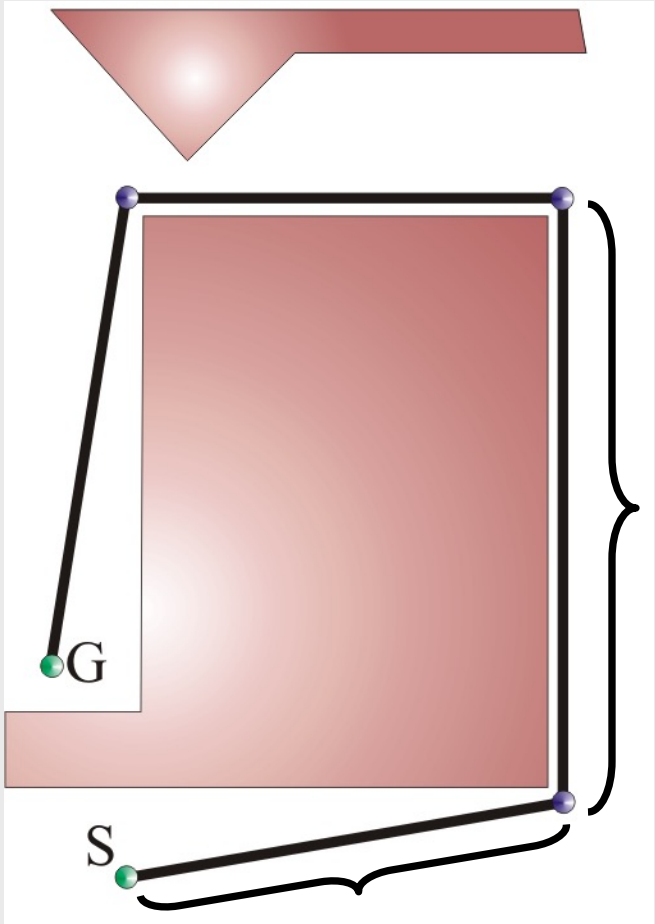
Situation after smoothing



Rounded movement collides

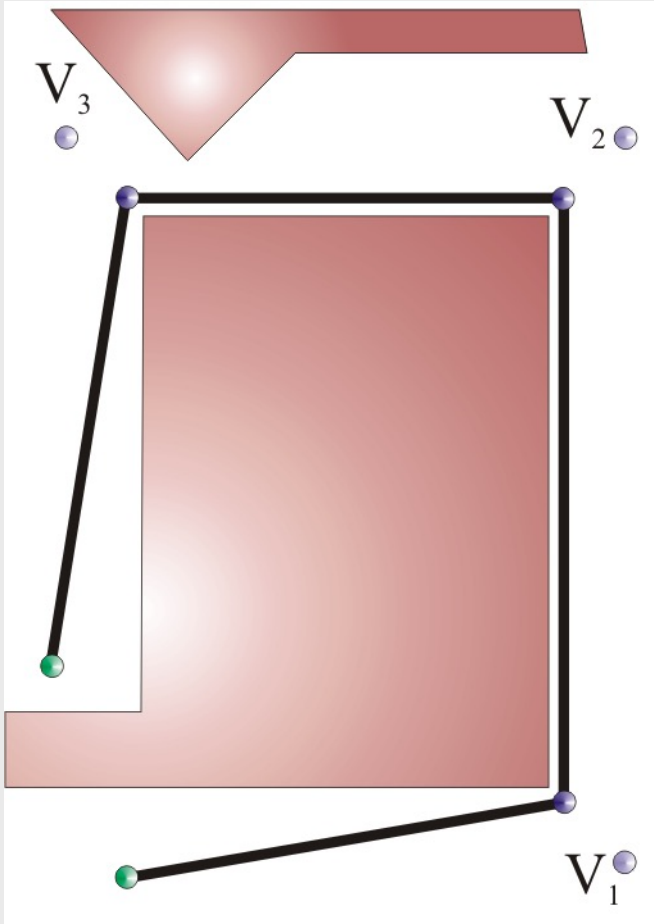


Situation after smoothing

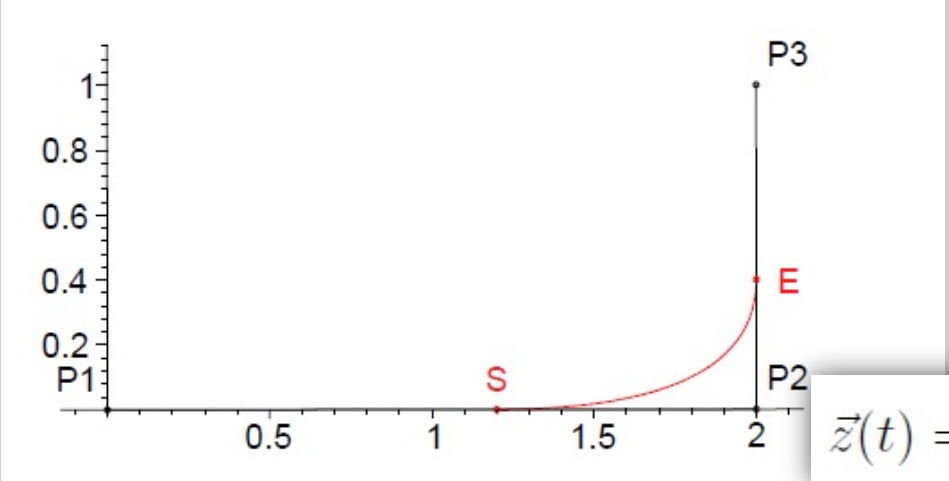


- Model of rounding Length of Segment
- Goal: hugest rounding factor as possible

Generation of virtual way points



„Inverse rounding“: first model (to get idea)

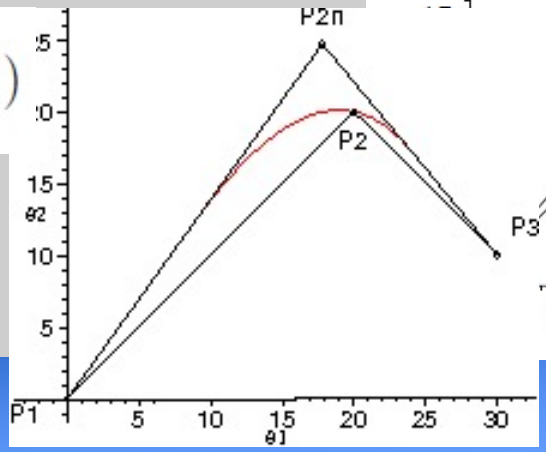


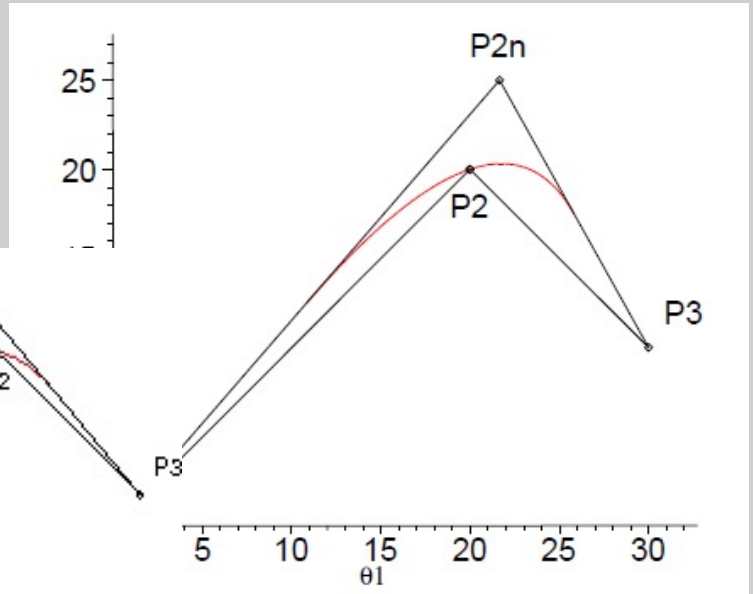
$$\vec{z}(t) = r \cdot \overrightarrow{P_2P_1}\,(t-1)^2 + r \cdot \overrightarrow{P_2P_3}\,t^2 + P_2\,,$$

$$\vec{z}_{neu}(0.5) = P_2$$

$$r \cdot \overrightarrow{P_{2_n}P_1} \cdot 0.25 + r \cdot \overrightarrow{P_{2_n}P_3} \cdot 0.25 + P_{2_n} = P_2$$

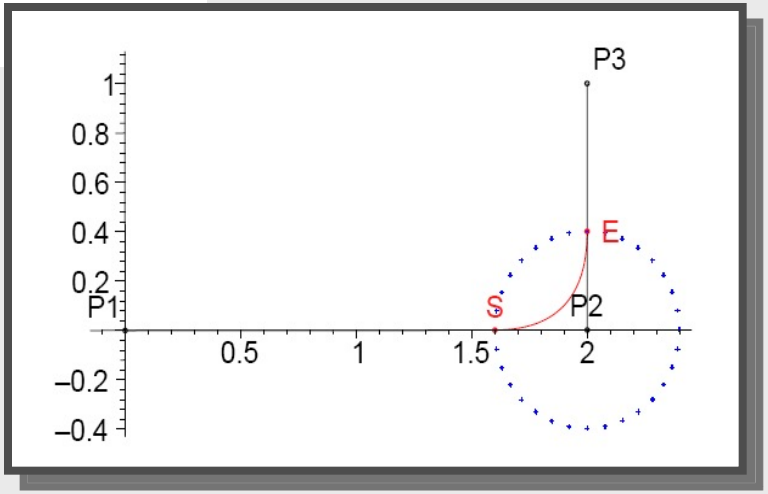
$$P_{2_n} = \frac{1}{2(r-2)}(rP_1 - 4P_2 + rP_3)$$





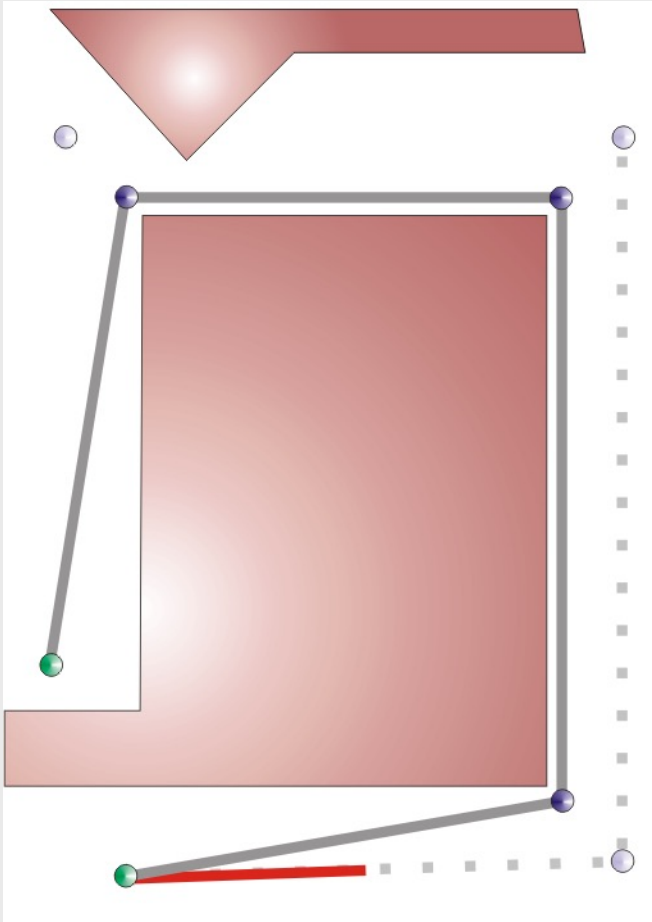
„inverse rounding“: second model

$$r \cdot \overrightarrow{P_{2_n}P_1}\,(t-1)^2 + r \cdot \frac{|\overrightarrow{P_{2_n}P_1}|}{|\overrightarrow{P_{2_n}P_3}|}\overrightarrow{P_{2_n}P_3}\,t^2 + P_{2_n} = P2$$

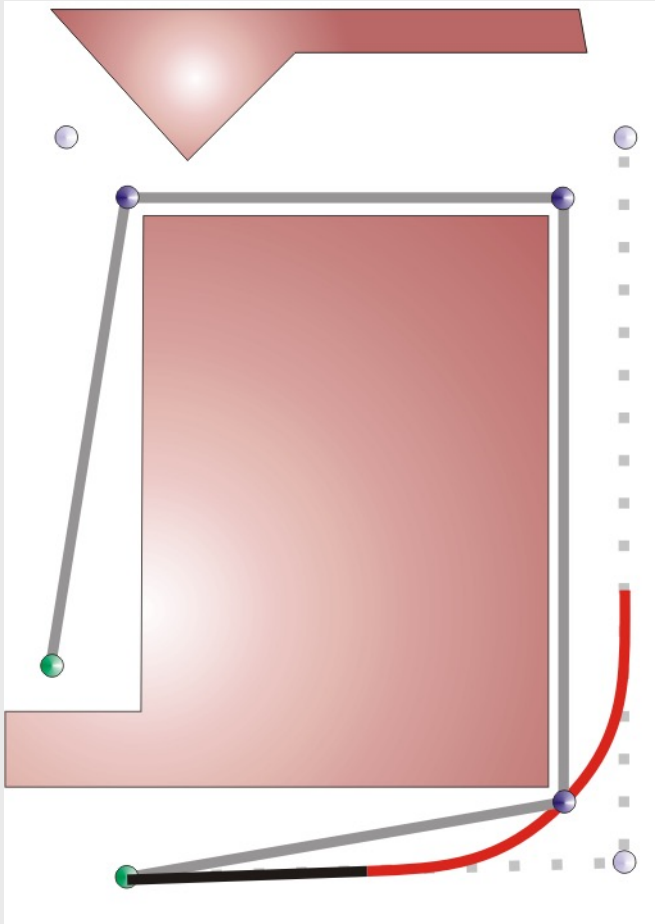


- Solution only solvable using numerical algorithms

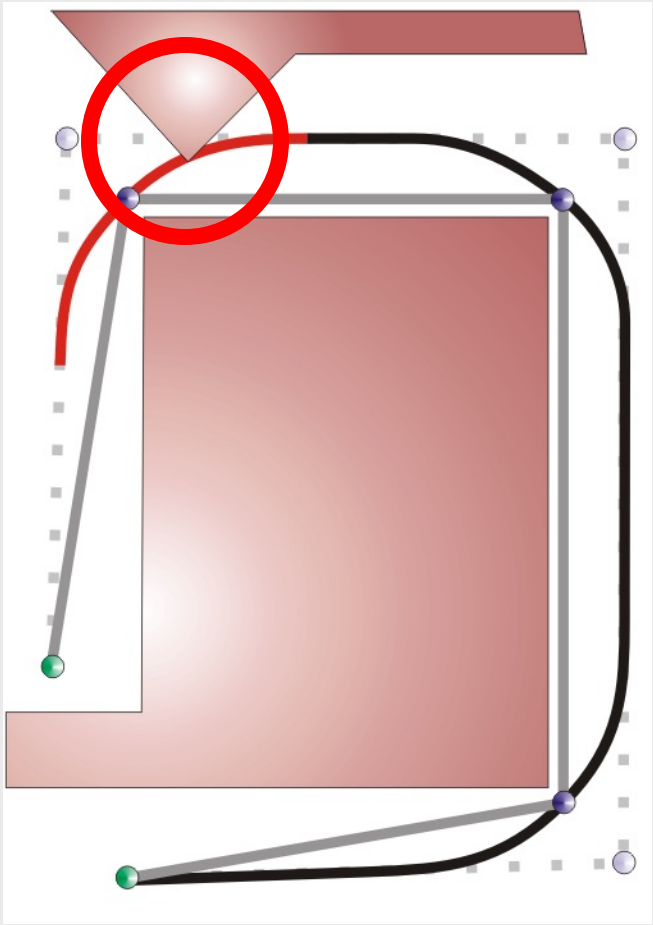
Collision testing of modified movement



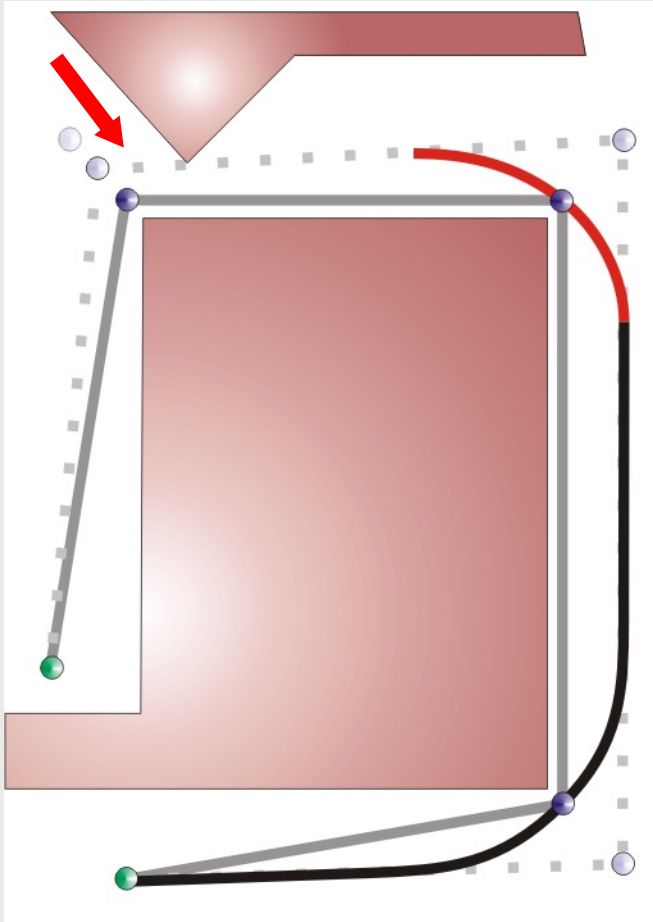
Collision testing of modified movement



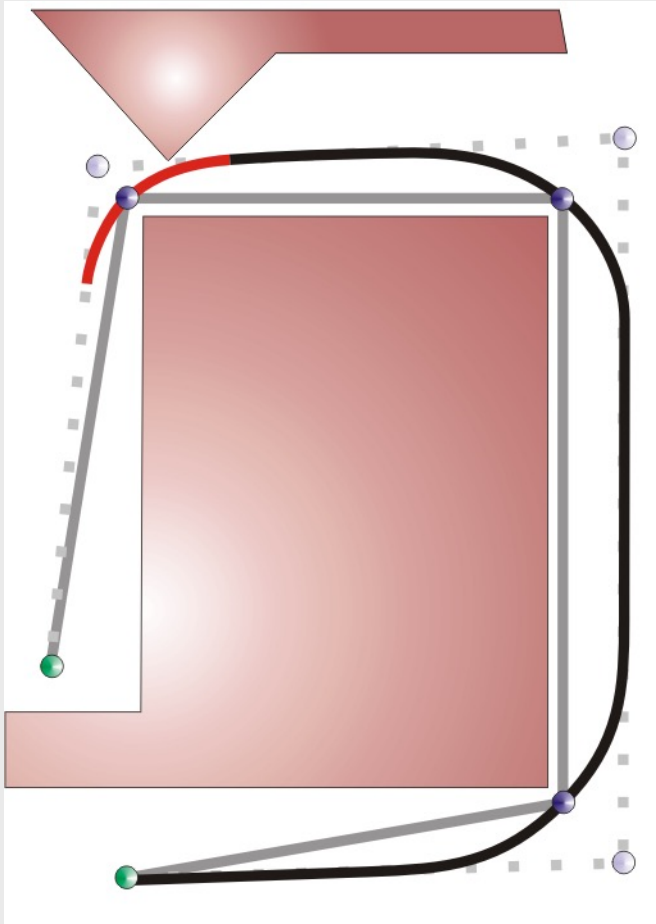
Collision testing of modified movement



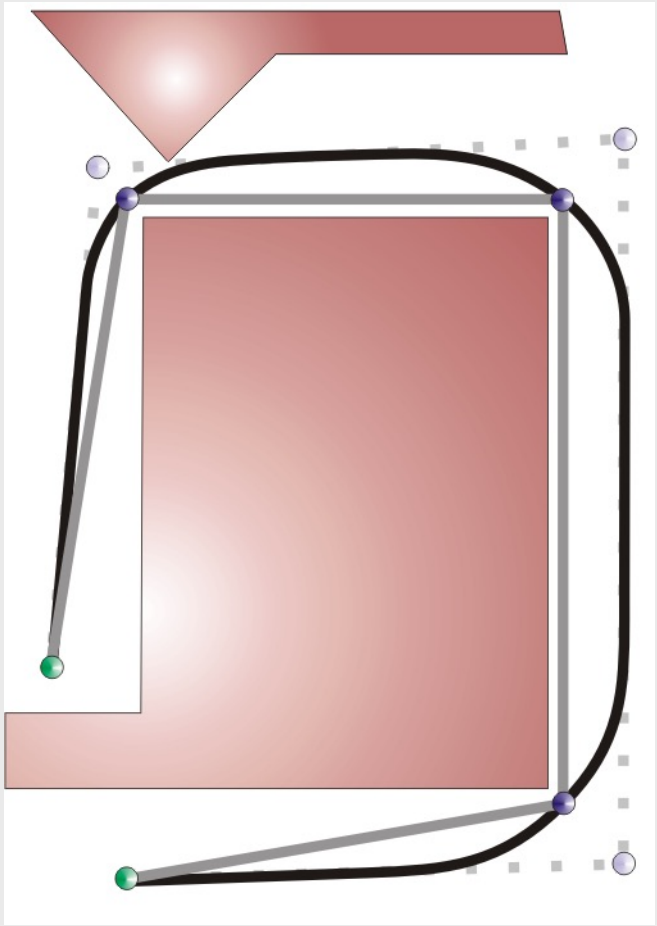
Collision testing of modified movement



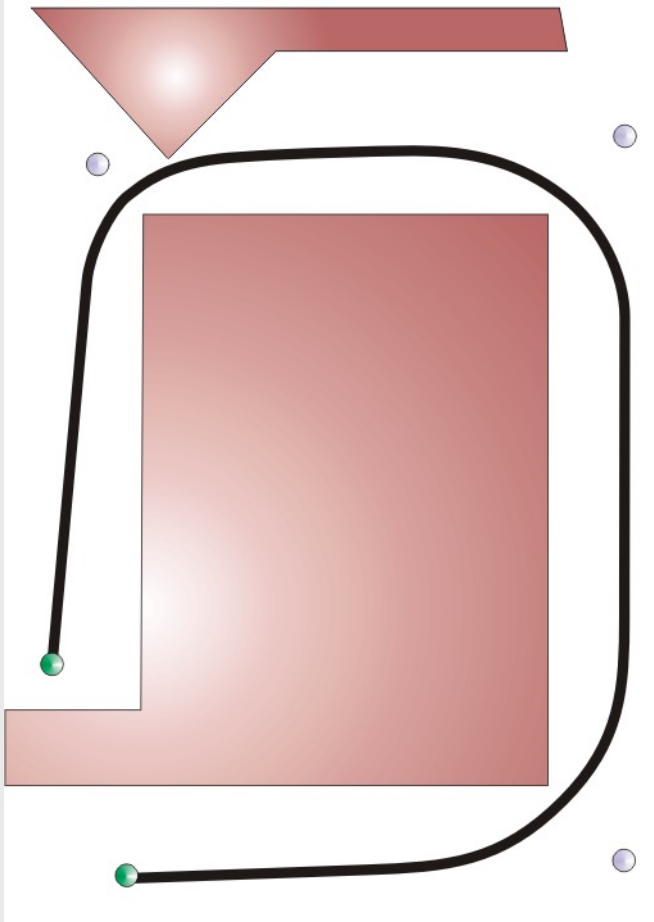
Collision testing of modified movement



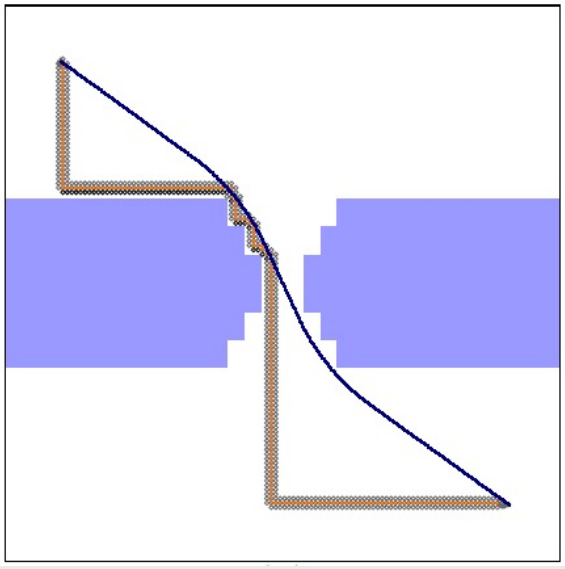
Collision testing of modified movement

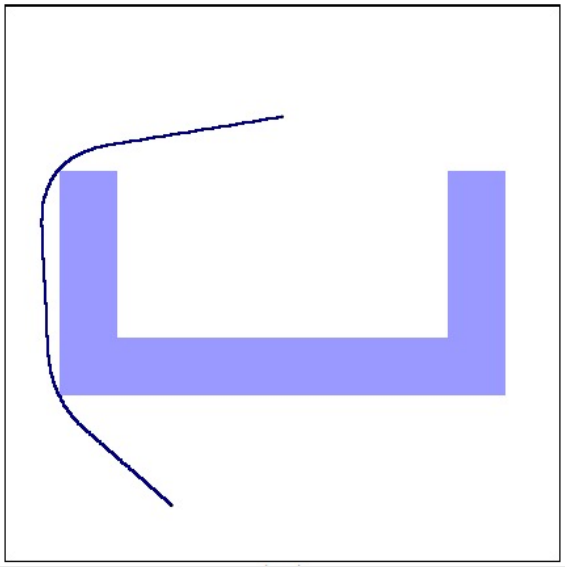


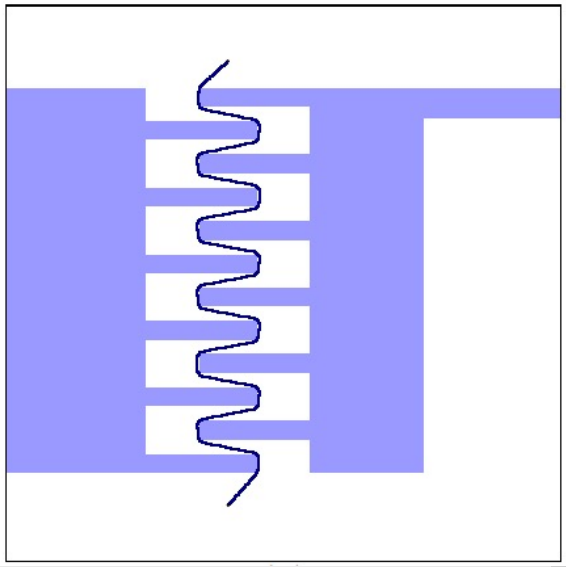
Result



Example: 2D







Example: SORT