Constrained Path Optimization with Bézier Curve Primitives [ICRA 2015]

Ji-Wung Choi and Kalevi Huhtala

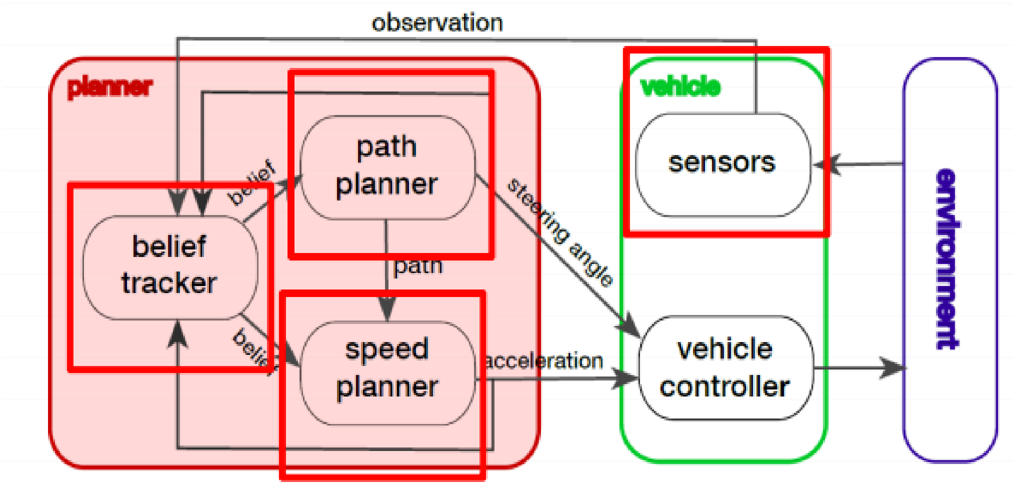
Heechan Shin

2017. 05. 30
CS686

KAIST

System Overview

- For every time step,
  - Belief tacking
  - Path planning
  - Speed planning



Bai, Haoyu, et al. "Intention-aware online POMDP planning for autonomous driving in a crowd." *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 2015.

Contents

- Problem of the paper
- Framework
- Methods
- Result
- Conclusion
- Quiz

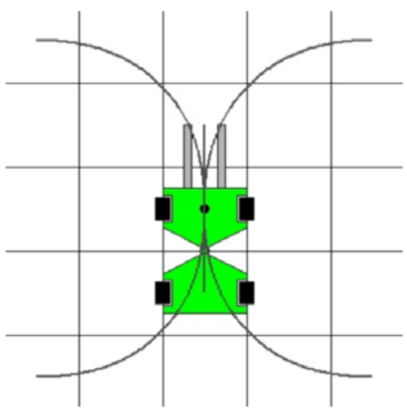
Problem of the paper

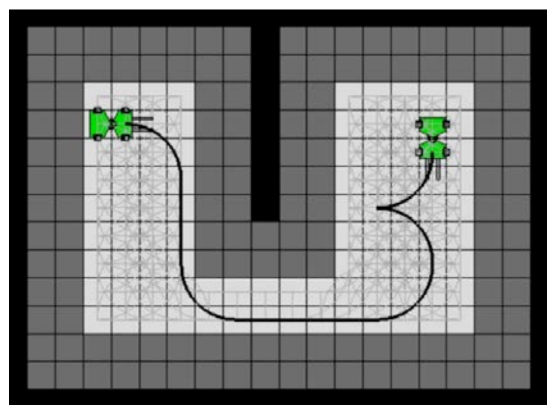
- Problem
  - path planning for non-holonomic wheeled vehicle moving on a plane
- Motivation
  - To make autonomous wheeled loaders pick up the pallet
  - Kalevi Huhtala
    - Professor of Tampere University at Finland
    - CEO of transportation company ‘Huhtala’

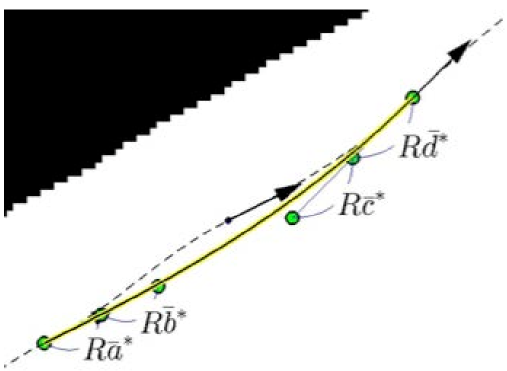
KAIST

Framework

- To compute finite set of segments of offline feasible motions
- Using A^*, Finding the path of sequence of segments
- Optimizing the global path and merging consecutive curves.





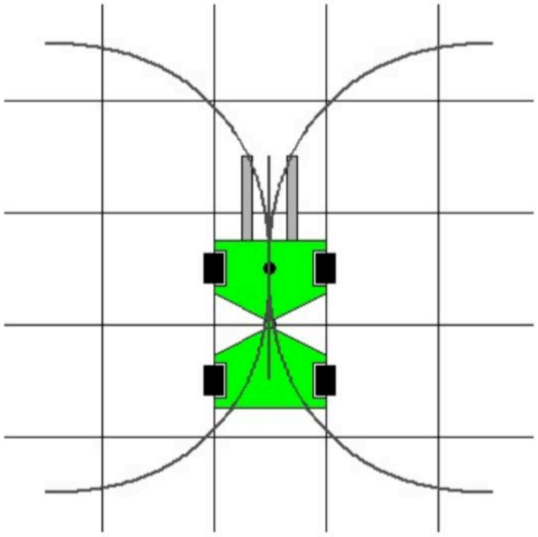


Methods – Computing segment

- Converting workspace to 2D state lattice
- Number of motion segments is arbitrary

- To generate smoothing segments, bezier cuvers are used

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{B}_i, \quad t \in [0,1].$$



Methods – Computing segment

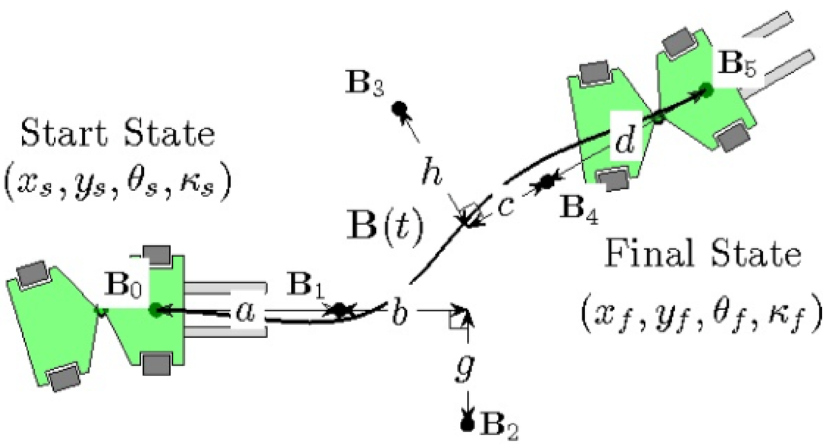
- To satisfy their minimum requirements, quintic Bezier curve is used
  - Minimum requirements
    - Passing through beginning and end point
    - Specified orientations θ and curvature κ

- Therefore, six control points are used

Methods – Computing segment

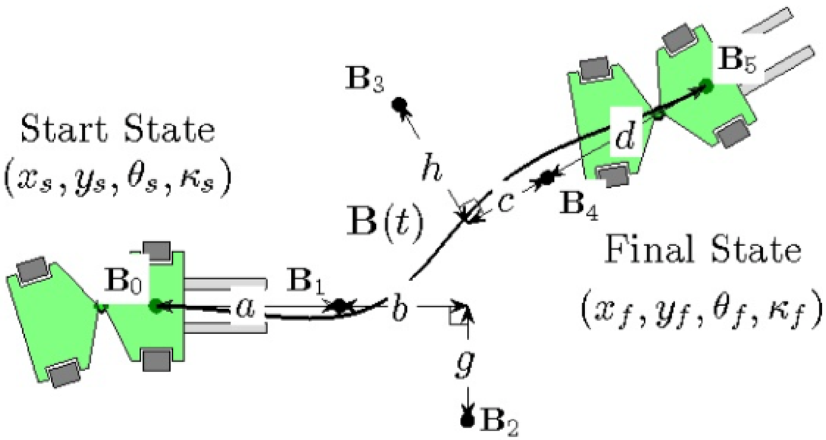
- $q_s = (x_s, y_s, \theta_s, \kappa_s)^T$ is beginning state
- $q_f = (x_f, y_f, \theta_f, \kappa_f)^T$ is end state

- $p_s = (x_s, y_s)^T$ is beginning position = B_0
- $p_f = (x_f, y_f)^T$ is beginning position = B_5



Methods – Computing segment

- **Bezier curve is tangent at the end points**
  - $\widehat{\theta_s} = \frac{1}{a}(B_1 - B_0)$ **therefore,** $B_1 = \boxed{a}\widehat{\theta_s} + p_s$
  - $\widehat{\theta_f} = \frac{1}{d}(B_4 - B_5)$ **therefore,** $B_4 = \boxed{d}\widehat{\theta_f} + p_f$



Methods – Computing segment

- **Curvatures at end of the bezier curve are**

$$\kappa(0) = \frac{(n-1)g}{n\|\mathbf{B}_1 - \mathbf{B}_0\|^2}, \quad \kappa(1) = \frac{(n-1)h}{n\|\mathbf{B}_n - \mathbf{B}_{n-1}\|^2}$$

- **Then, we can get**

$$\mathbf{B}_2 = \mathbf{p}_s + (a + \boxed{b})\hat{\theta}_s + \frac{5}{4}a^2\kappa_s\dot{\hat{\theta}}_s,$$

$$\mathbf{B}_3 = \mathbf{p}_f - (\boxed{c} + d)\hat{\theta}_f + \frac{5}{4}d^2\kappa_f\dot{\hat{\theta}}_f$$

- **The curve is spanned by 12-tuple parameters**

$$\boldsymbol{\lambda = (x_s, y_s, \theta_s, \kappa_s, a, b, c, d, x_f, y_f, \theta_f, \kappa_f)}$$

Methods – Computing segment

$$\lambda = (\underbrace{x_s, y_s, \theta_s, \kappa_s}_{\text{determined}}, a, b, c, d, \underbrace{x_f, y_f, \theta_f, \kappa_f}_{\text{determined}})$$

- **Four degree of freedom**
  - $h = (a, b, c, d)^T$

- **To calculate optimal control distance vector h^***

$$\text{Minimize: } J(\mathbf{h}) = \sum_{j=0}^{k-1} \left[w_s \cdot \underbrace{s_j(\mathbf{h})}_{\text{Path length}} + w_\kappa \cdot \underbrace{\kappa_j^2(\mathbf{h})}_{\text{Curvature}} \right]$$

$$\text{subject to: } \kappa_j^2 < \kappa_{max}^2.$$

Methods – Computing segment

$$\text{Minimize: } J(\mathbf{h}) = \sum_{j=0}^{k-1} \left[w_s \cdot s_j(\mathbf{h}) + w_\kappa \cdot \kappa_j^2(\mathbf{h}) \right]$$

$$\text{subject to: } \kappa_j^2 < \kappa_{max}^2.$$

- **Adapting Method-of-Moving Asymptotes(MMA)**
  - **Proximity of the initial guess to the optima**
  - **Computational efficiency**

- **Empirically, they found quarter of the curve length to be good**

$$\mathbf{h}_{guess} = \|\mathbf{p}_f - \mathbf{p}_s\| \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}^T$$

Methods – Collision avoidance

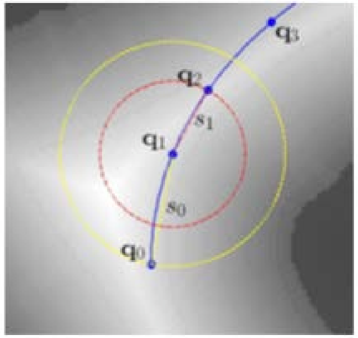
$$\text{Minimize: } J(\mathbf{h}) = \sum_{j=0}^{k-1} \left[w_s \cdot s_j(\mathbf{h}) + w_\kappa \cdot \kappa_j^2(\mathbf{h}) \right]$$

$$\text{subject to: } \kappa_j^2 < \kappa_{max}^2.$$

- Signed Distance Field(SDF) is used to avoiding collision

$$\gamma_j = \begin{cases} \|\mathbf{B}(t_j) - \mathbf{o}_j\|, & \text{if } \mathbf{B}(t_j) \in \mathcal{C}_{free}, \\ -\|\mathbf{B}(t_j) - \mathbf{o}_j\|, & \text{if } \mathbf{B}(t_j) \in \mathcal{C}_{obstacle} \end{cases}$$

- If $\gamma(q_j)$ is greater than $||q_j - q_{j-1}||$, no collision occurs along $q_j - q_{j-1}$



Methods – Global optimization

- **Global path**

$$\boldsymbol{P = B^{[0]}(t) \cup \cdots \cup B^{[m]}(t)}$$
$$\boldsymbol{= B(\lambda_0, t) \cup \cdots \cup B(\lambda_m, t)}$$

$$\boldsymbol{where\ \lambda_i = \left(q_i^T, h_i^T, q_{i+1}^T\right)^T}$$

- **Parameter variable of global path**

$$\boldsymbol{X = \left\{B_0^{[0]} \cdots B_5^{[0]}, \cdots, B_0^{[m]} \cdots B_5^{[m]}\right\}}$$

$$\boldsymbol{= (h_0^T\, q_1^T\, h_1^T\, q_2^T \cdots h_{m-1}^T\, q_m^T\, h_m^T)}$$

- **From 12(m+1) to 8m+4**

Methods – Global optimization

- **Global path**

$$\text{Minimize: } J(\mathbf{X}) = \sum_{i=0}^{m} L^{[i]}(\lambda_i),$$

$$\text{subject to: } \left(\kappa_j^{[i]}\right)^2 < \kappa_{max}^2,$$

$$\gamma_j^{[i]} > s_{j-1}^{[i]} \text{ and } \gamma_j^{[i]} > s_j^{[i]}$$

where

$$L^{[i]}(\lambda_i) = \sum_{j=0}^{k_i-1} \left[w_s \cdot s_j^{[i]} + w_\kappa \cdot \left(\kappa_j^{[i]}\right)^2 - \underbrace{w_\gamma \cdot \gamma_j^{[i]}}_{\text{SDF}} \right]$$

- **Also it can be normalized using maximum value of each term.**

KAIST

Methods – Merging

- **Using divide and conquer**
  - **Dividing global curve into segments**
  - **And then merging it again**

Algorithm 1 Merge Bézier Curves Comprising A Path

procedure MERGECURVES($\{\mathbf{q}_0, \mathbf{h}_0, \dots, \mathbf{h}_m, \mathbf{q}_{m+1}\}$)
    $\mathcal{K}$ is normalized motion segment set.
    if $m \leq 0$ **then**
        return $\{\mathbf{q}_0, \mathbf{h}_0, \dots, \mathbf{h}_m, \mathbf{q}_{m+1}\}$
    else
        $\tilde{c} \Leftarrow \lfloor \frac{m+2}{2} \rfloor$
        $left \Leftarrow MergeCurves(\{\mathbf{q}_0, \mathbf{h}_0, \dots, \mathbf{h}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}}\})$
        $right \Leftarrow MergeCurves(\{\mathbf{q}_{\tilde{c}}, \mathbf{h}_{\tilde{c}}, \dots, \mathbf{h}_m, \mathbf{q}_{m+1}\})$
        return $Merge(left, right)$
    end if
end procedure

Dividing

procedure MERGE($left = \{\mathbf{q}_{c_0}, \dots, \mathbf{h}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}}\}$, $right = \{\mathbf{q}_{\tilde{c}}, \mathbf{h}_{\tilde{c}}, \dots, \mathbf{q}_{c_f}\}$)
    $\mathbf{h}^* \Leftarrow LookUpClosestControl(\mathbf{q}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}+1}, \mathcal{C})$
    $\mathbf{B}^* \Leftarrow \mathbf{B}(\mathbf{q}_{\tilde{c}-1}, \mathbf{h}^*, \mathbf{q}_{\tilde{c}+1})$
    if $CollisionFree(\mathbf{B}^*) = true \wedge \kappa(\mathbf{B}^*) < \kappa_{max}$ **then**
        return $\{\mathbf{q}_{c_0}, \dots, \mathbf{q}_{\tilde{c}-1}, \mathbf{h}^*, \mathbf{q}_{\tilde{c}+1}, \dots, \mathbf{q}_{c_f}\}$
    else
        return $\{\mathbf{q}_{c_0}, \dots, \mathbf{h}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}}, \mathbf{h}_{\tilde{c}}, \dots, \mathbf{q}_{c_f}\}$
    end if
end procedure

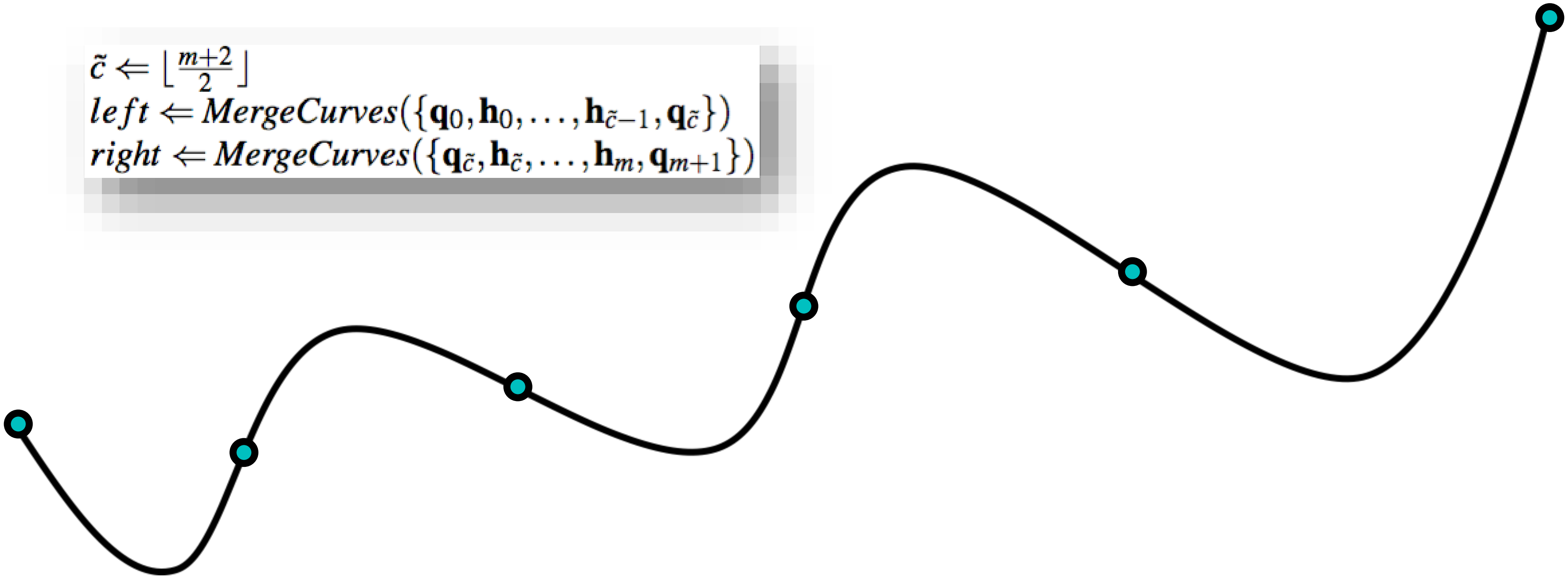
Merging

KAIST

Methods – Merging

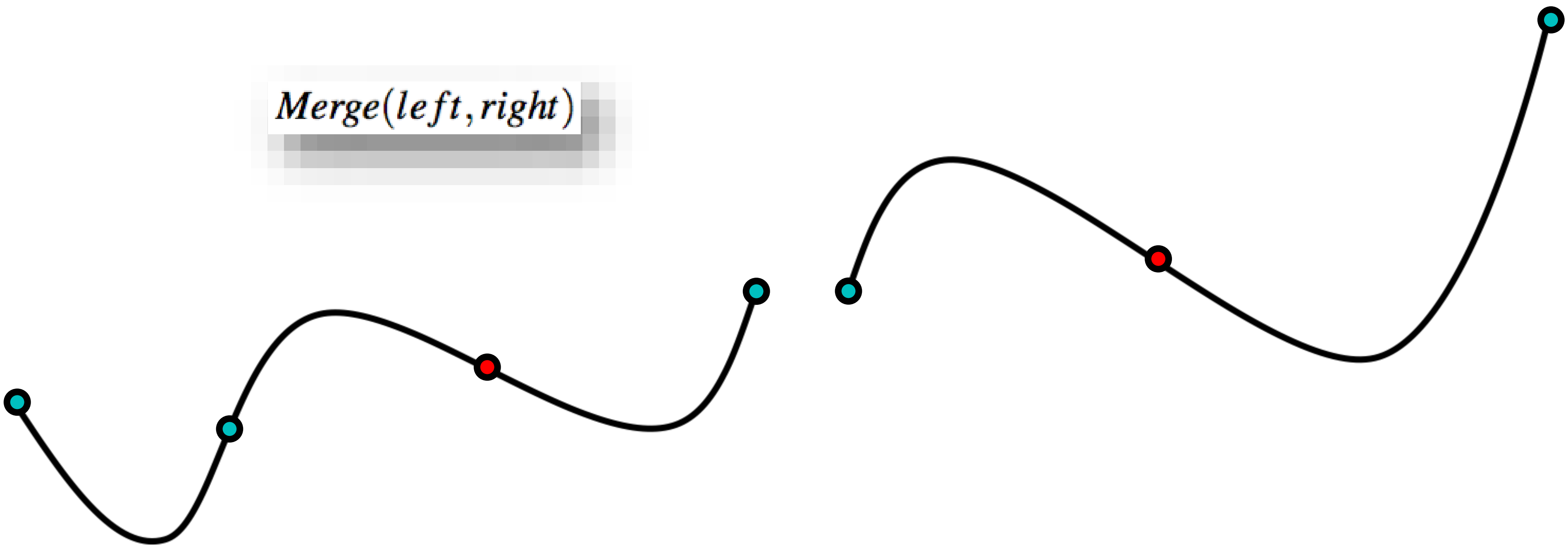
- **Using divide and conquer**
  - **Dividing global curve into segments**
  - **And then merging it again**

$$\tilde{c} \Leftarrow \left\lfloor \frac{m+2}{2} \right\rfloor$$
$$left \Leftarrow MergeCurves(\{\mathbf{q}_0, \mathbf{h}_0, \ldots, \mathbf{h}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}}\})$$
$$right \Leftarrow MergeCurves(\{\mathbf{q}_{\tilde{c}}, \mathbf{h}_{\tilde{c}}, \ldots, \mathbf{h}_m, \mathbf{q}_{m+1}\})$$



Methods – Merging

- **Using divide and conquer**
  - **Dividing global curve into segments**
  - **And then merging it again**



Methods – Merging

- **Using divide and conquer**
  - **Dividing global curve into segments**
  - **And then merging it again**

$\mathbf{h}^* \Leftarrow \mathit{LookUpClosestControl}(\mathbf{q}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}+1}, \mathcal{C})$
$\mathbf{B}^* \Leftarrow \mathbf{B}(\mathbf{q}_{\tilde{c}-1}, \mathbf{h}^*, \mathbf{q}_{\tilde{c}+1})$
if $\mathit{CollisionFree}(\mathbf{B}^*) = \mathit{true} \wedge \kappa(\mathbf{B}^*) < \kappa_{max}$ **then**
  return $\{\mathbf{q}_{c_0}, \ldots, \mathbf{q}_{\tilde{c}-1}, \mathbf{h}^*, \mathbf{q}_{\tilde{c}+1}, \ldots, \mathbf{q}_{c_f}\}$
else
  return $\{\mathbf{q}_{c_0}, \ldots, \mathbf{h}_{\tilde{c}-1}, \mathbf{q}_{\tilde{c}}, \mathbf{h}_{\tilde{c}}, \ldots, \mathbf{q}_{c_f}\}$
end if

Result

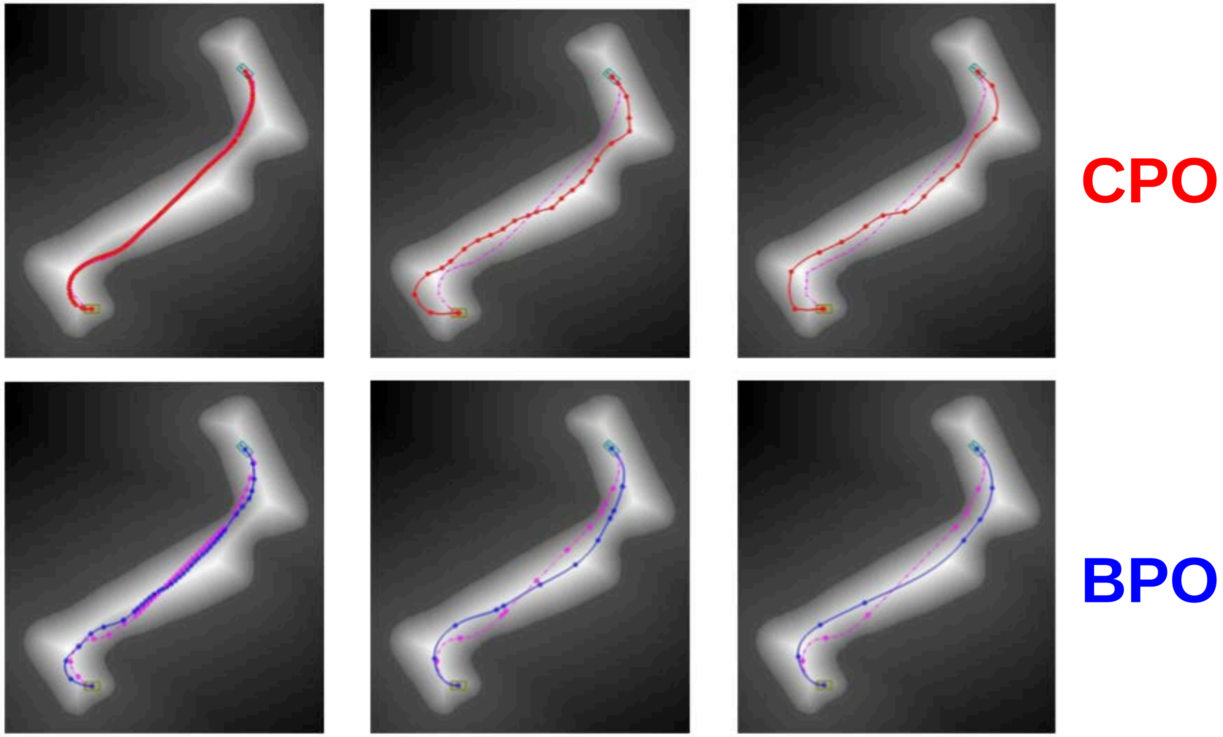
- **Experimental setting**
  - **Obstacle map : 150m x 80m**
  - **State lattice planner**
    - **Resolution : 1m**
    - **16 discrete headings**

- **Their method(Bezier curves parametric Path Optimization:BPO)**
  vs
  The previous method(Coordinates parametric Path Optimization:CPO)

KAIST

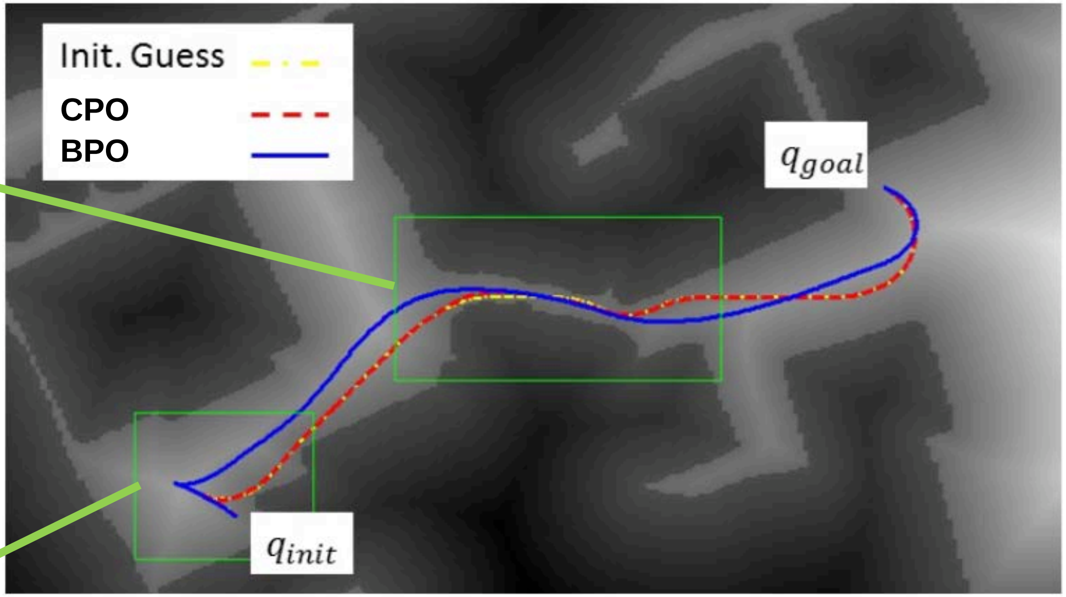
Result

- **Experimental result**
  - **Given an initial guess, both methods optimize the solutions**



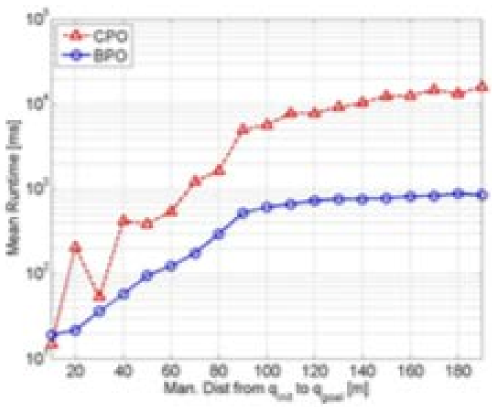
Result

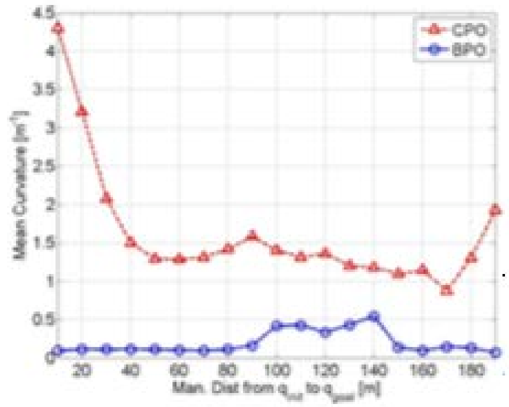
BPO is more like voronoi diagram

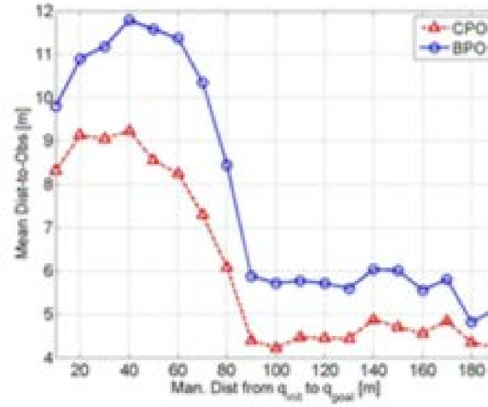


Cusp-point
CPO splits the path into forward/backward
But BPO optimizes the overall path at once

Result







- 4000 test cases in an identical environment
- Average path computing time
  - CPO : 3.28s
  - BPO : 300ms
- Improvement of curvature 8.9
- Improvement of dist.to.obstacle 1.295

Conclusion

- **Benefits of proposed algorithm**
  - **Can generate bezier curve with few parameters**
  - **Can efficiently compute optimization of path-length, smoothness, and collision clearance**
  - **Can achieve compactness with merging**

QnA

Thank you!