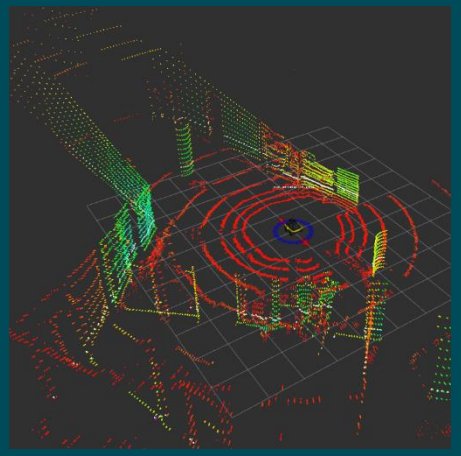
UNIVERSITÄT ZU LÜBECK

Institute for Robotics and Cognitive Systems

Dr. Ngoc Thinh Nguyen. Email: nguyen@rob.uni-luebeck.de

Advanced topics in robotics: Safe navigation in polytopic environment



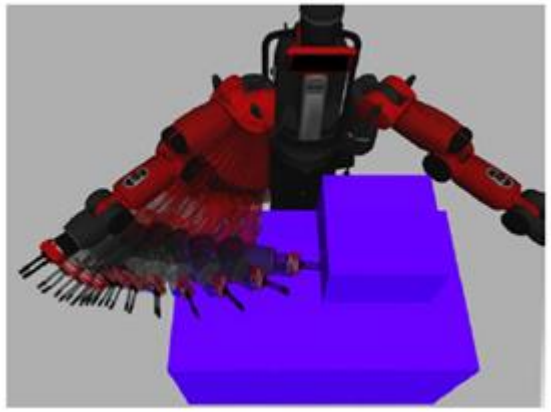
Lecture WS 2021/2022
(RO5800/ KP 04/ KP 08/ KP 12)
Dr. Ngoc Thinh Nguyen

Content

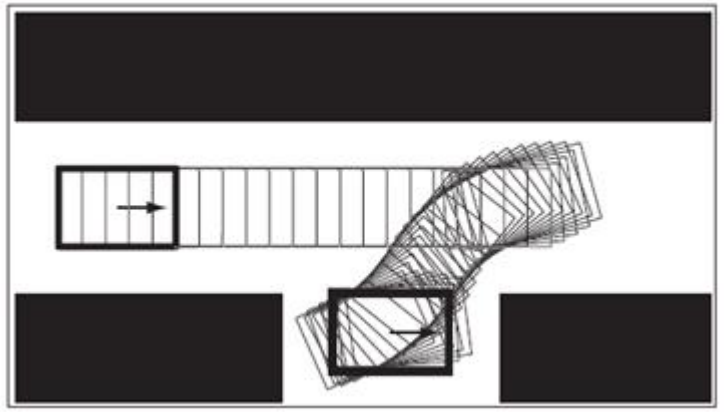
1. Introduction
2. From real world to polytope map
3. B-spline path planning in polytope map
4. MPC designs with tightening polytopic constraint
5. Conclusions

Introduction

Problem of finding a robot motion from a start state to a goal state that avoids obstacles and satisfies other constraints (velocities, accelerations, and so on).



A robot arm executing an obstacle-avoiding motion plan



A mobile robot executing parallel parking

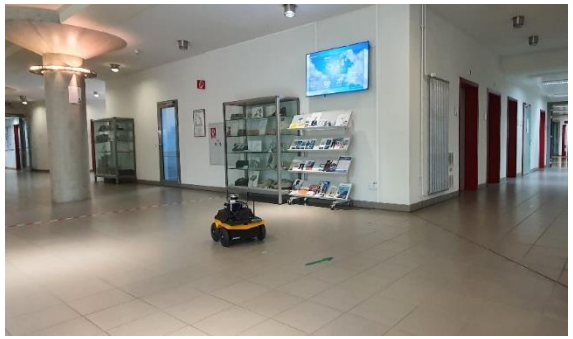
[1] Kevin M. Lynch, and Frank C. Park. Modern Robotics: Mechanics, Planning and Control. Cambridge University Press, May 2017.

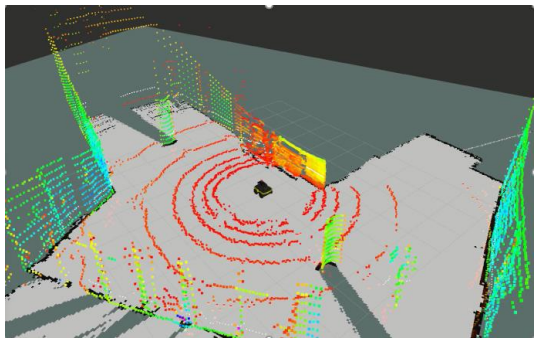
Introduction

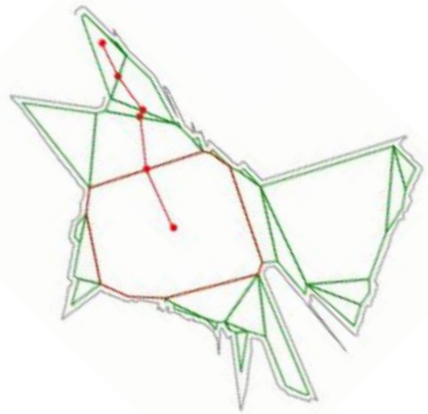
Process can be split into three major steps:

1. Mapping: capture of surrounding environment by using necessary sensors, e.g., proximity sensor, range sensor, (depth) camera, and provide a map.

→ Study the creation of polytope map for easily cope with optimal motion planning and optimal control designs.







A mobile robot mapping at University of Luebeck, using 3D LIDAR sensor

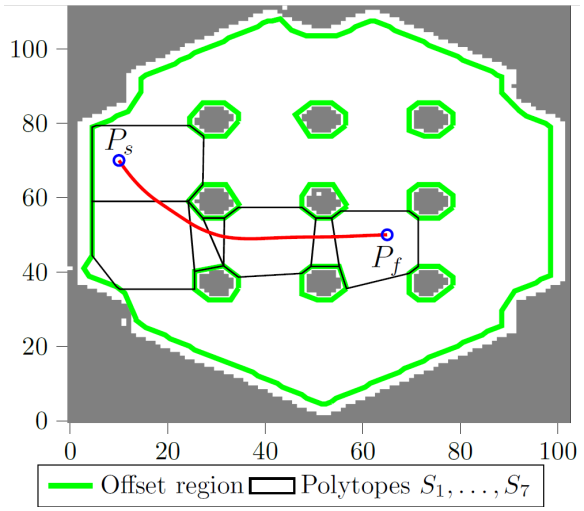
[3] Johann Dichtl et al. "PolySLAM: A 2D Polygon-based SLAM algorithm", in Proc. of IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 1-6, 2019.
[4] Christian Meerpohl et al. "Free-space Polygon Creation based on Occupancy Grid Maps for Trajectory Optimization Methods", in IFAC-PapersOnline, 52(8): 368-374, 2019.

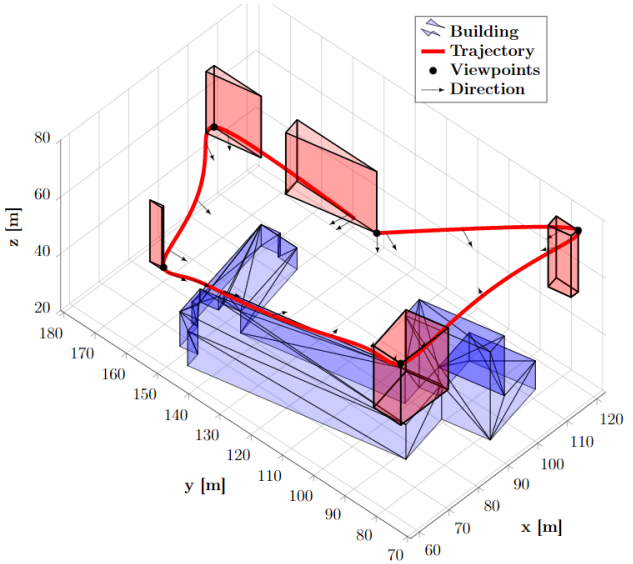
Introduction

Process can be split into three major steps:

2. Reference generation: planning desired path or trajectory in the given map of environment.

→ study the use of B-spline curve in path planning optimization problems.





B-spline path/ trajectory planning in 2D/ 3D polytopic environment

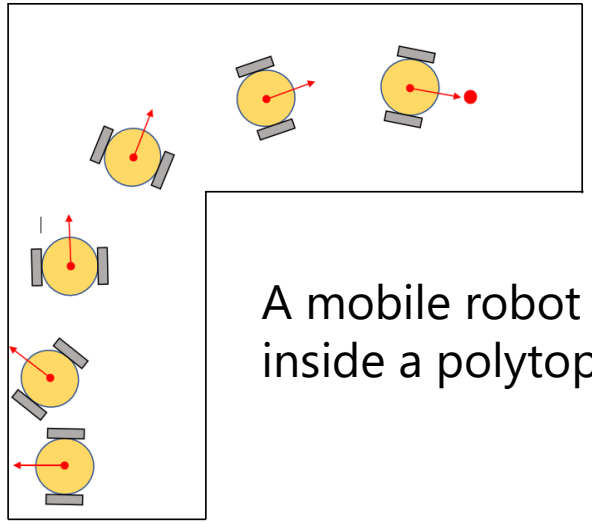
[8] Ionela Prodan et al. "Necessary and sufficient LMI conditions for constraints satisfaction within a B-spline framework", in Proc. of 58th Conference on Decision and Control (CDC), 8061-8066, 2019
[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021

Introduction

Process can be split into three major steps:

3. Navigation: track the desired reference or directly move to the desired goal within the map by using a controller with careful consideration of constraints.

→ study the use of a new formulation, so called tightening polytopic constraint, in Model Predictive Control (MPC) designs for 2D navigation.



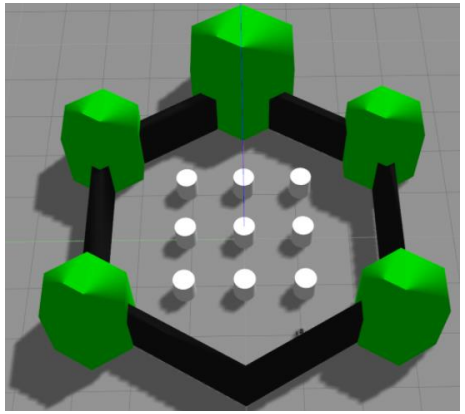
A mobile robot navigating inside a polytopic corridor

[13] Ngoc Thinh Nguyen, and Georg Schildbach. "Tightening polytopic constraint in MPC designs for mobile robot navigation", in Proc. of 25th International Conference on System Theory, Control and Computing (ICSTCC), 407-412, 2021. **Best Paper Award**.

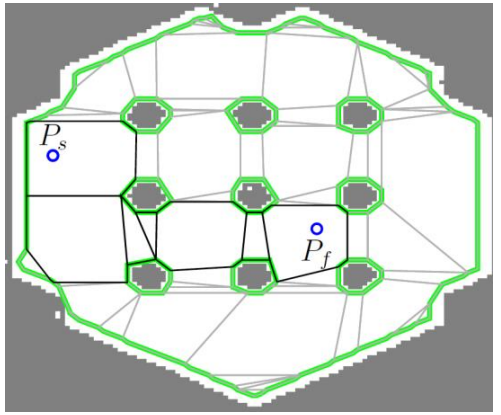
[14] Jong Jin Park, Collin Johnson, and Benjamin Kuipers. "Robot navigation with model predictive equilibrium point control". In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4945-4952, 2012.

2. From real world to polytope map

a) Sensors
b) Problems when creating polytope map
c) Existing approaches on creating polytope map
d) Exercises on creating polytope map



?



Sensors

- Sensors mostly used for capturing the surrounding environment:
  Lidar (Light detection and ranging) and stereo camera.

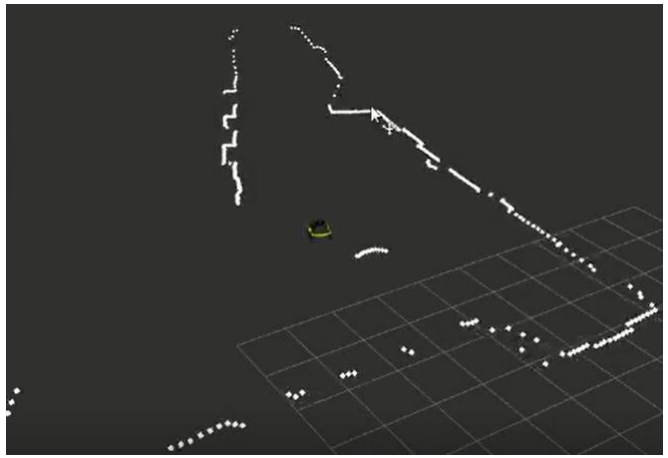
Ouster 3D Lidar and ZED stereo camera equipped in the Jackal mobile robot

Velodyne Lidar and stereo camera equipped in KITTI Vision Benchmark Suite vehicle [2]

[2] Andreas Geiger et al, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", in Proc. of Conference on Computer Vision and Pattern Recognition, 2012.

Lidar (Light Detection and Ranging)

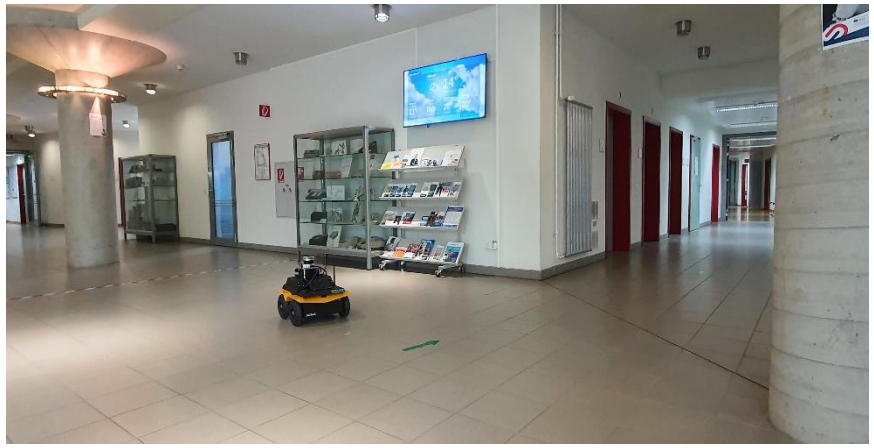
- Lidar emits pulsed light waves into the surrounding environment and receives the reflected waves.
- Based on the difference on time of transmitting and receiving, the distance to an obstacle is calculated. Repeating the process while rotating the laser source creates a map of the environment.

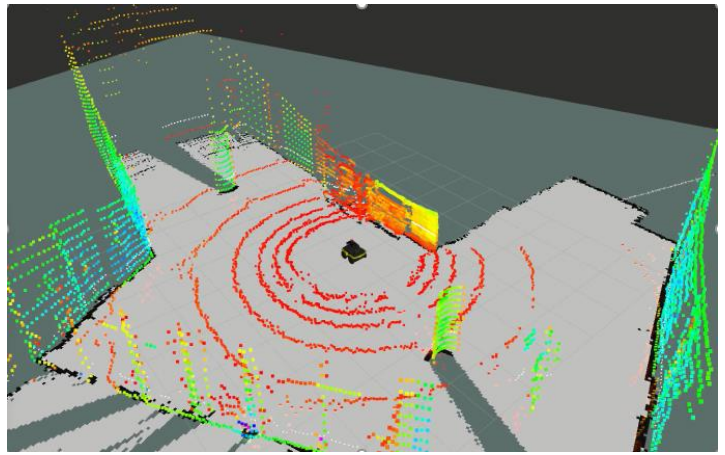


Results obtained from a 2D Lidar

Lidar (Light Detection and Ranging)

- Lidar emits pulsed light waves into the surrounding environment and receives the reflected waves.
- Based on the difference on time of transmitting and receiving, the distance to an obstacle is calculated. Repeating the process while rotating the laser source creates a map of the environment.

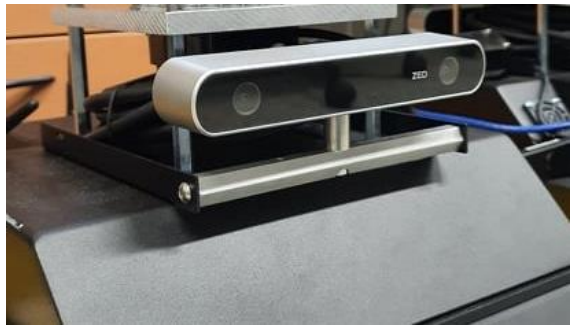




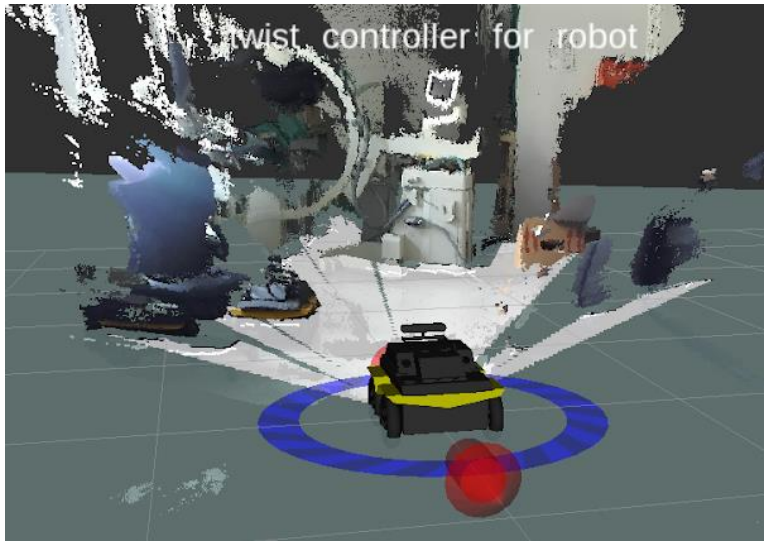
Results obtained from a 3D Lidar

Stereo camera

- Stereo camera reproduces the way human vision works. It has two “eyes”.
- By comparing the displacement of pixels between the left and right images, it can create a 3D map of the scene.



ZED stereo camera with two optical lens

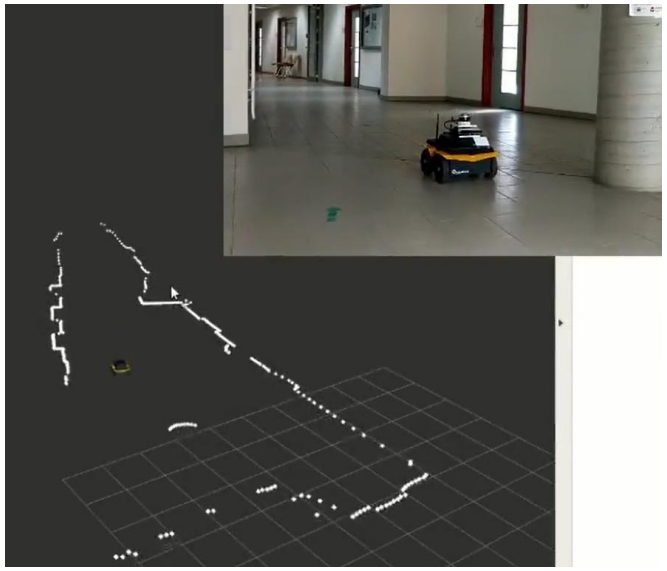


Results obtained from stereo camera

Problems when creating a polytope map

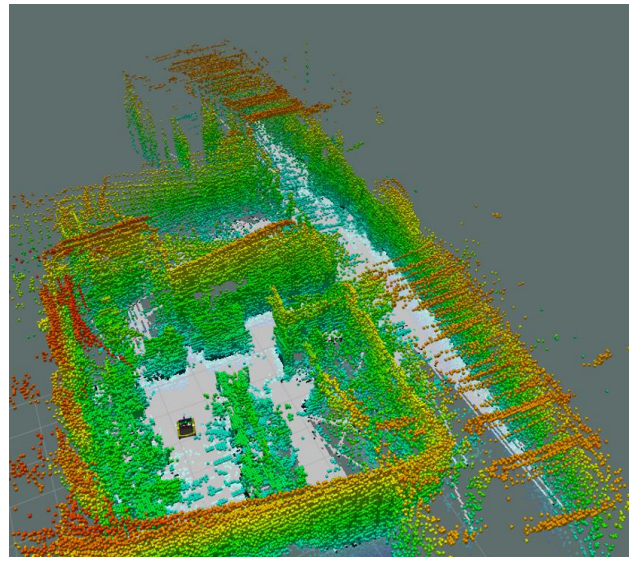
- Surrounding environment appears as discrete points, meshes, grids with noises .
- Obstacles and free space can be extremely non-convex, non-connected.

→ difficulties on finding a polygon representation of the surrounding environment.



A 2D laser scan of surrounding environment

A grid map in which, a robot is localizing itself using Particle filter

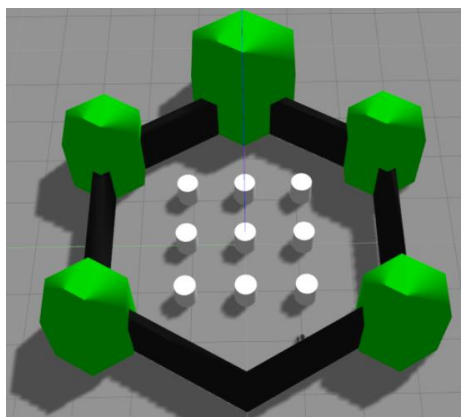


A 3D OctoMap

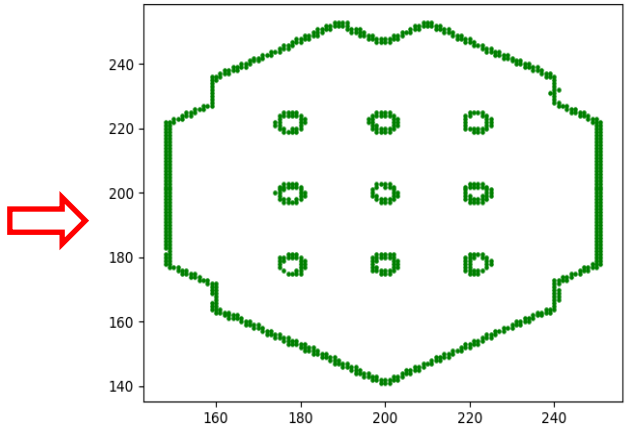
Existing approaches on creating polytope map

1. Polytope map

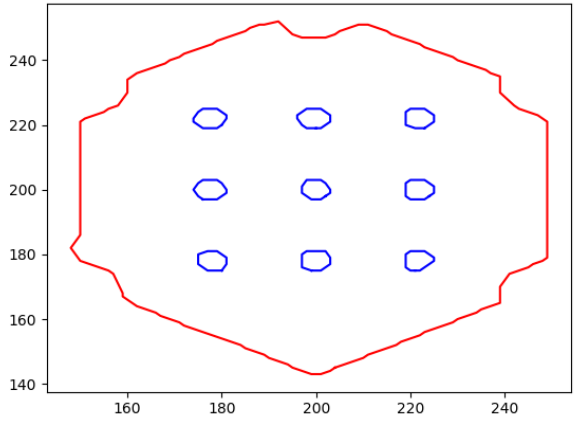
- Transformation from a standard occupancy grid map to a polytope map [12]: approximate the outer boundary of the free space and the obstacles by polygons, make a subtraction and decompose the polygonal free space into polytopes.



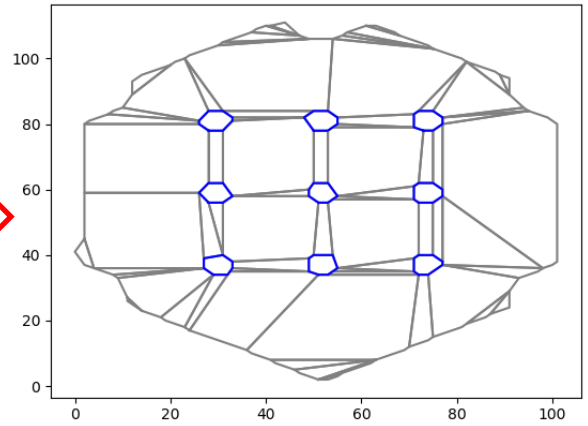
Actual world



Occupancy grid map



Polyhedron approximation



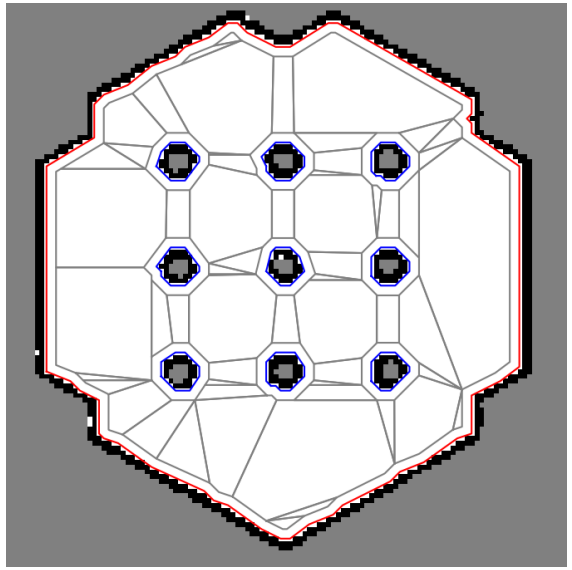
Polytope map

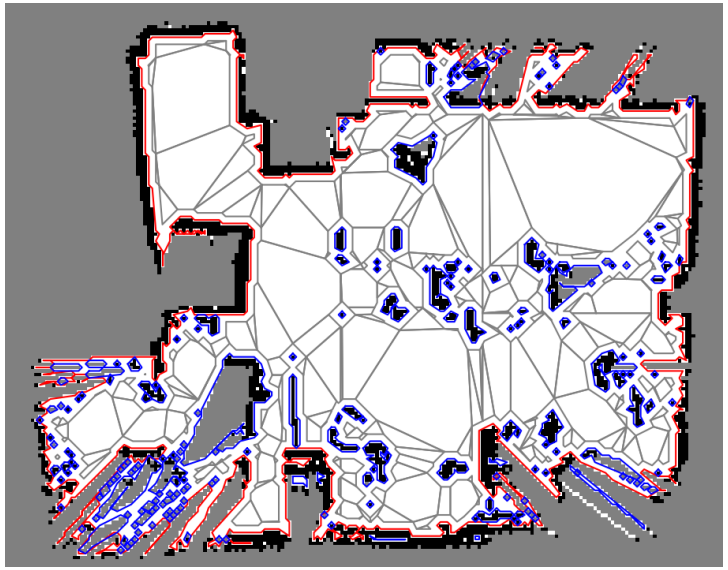
[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

Existing approaches on creating polytope map

1. Polytope map

- Transformation from a standard occupancy grid map to a polytope map [12]: approximate the outer boundary of the free space and the obstacles by polygons, make a subtraction and decompose the polygonal free space into polytopes.



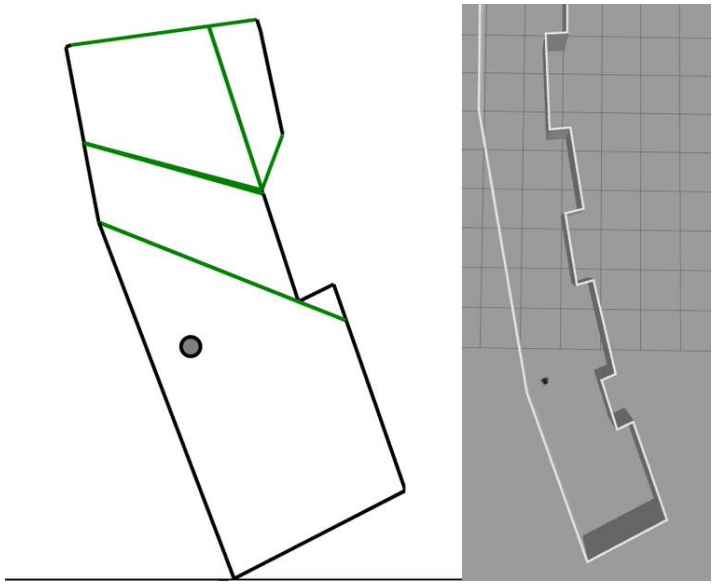


[12] Ngoc Thinh Nguyen et al. “B-spline path planner for safe navigation of mobile robots”, in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

Existing approaches on creating polytope map

1. Polytope map

- Polygon-based SLAM (Simultaneous Localization and Mapping) – PolySLAM [3]: directly creating a polytope map from laser scan while exploring the surroundings.



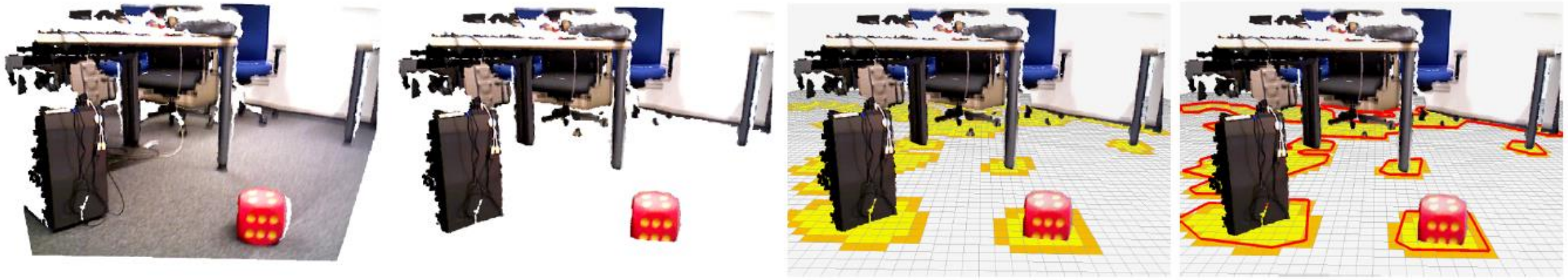
Implementation of PolySLAM by Killian Schweppe (Uni. of Lubeck) in his bachelor thesis, 2021

[3] Johann Dichtl et al. "PolySLAM: A 2D Polygon-based SLAM algorithm", in Proc. of IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 1-6, 2019.

Existing approaches on creating polytope map

1. Polytope map

- Polygonal perception for mobile robots [15]: creating polygonal boundaries of the obstacles using the 3D point cloud obtained from a stereo camera in real time.



(a) Rectified Point Cloud (b) Floor Removal (c) Dilated Occupancy Grid (d) Smoothed Polygons

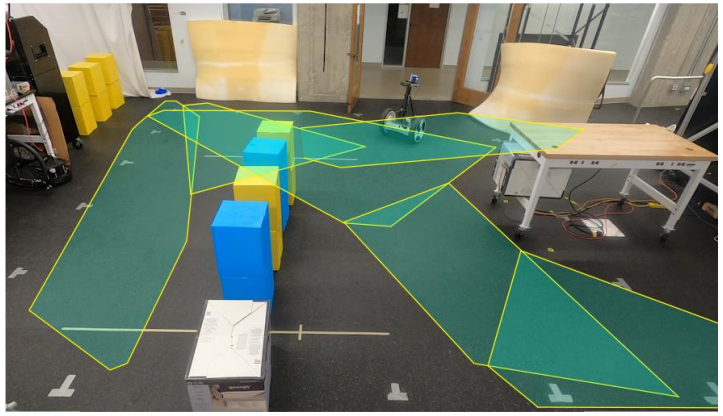
Construction of a two-dimensional polygonal environment model: from raw point cloud to polygonal obstacles [15].

[15] Marcell Missuar, et al. Polygonal Perception for Mobile Robots, in Proc. of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 10476-10482, 2020.

Existing approaches on creating polytope map

1. Polytope map
2. Polytope corridor

- Obtained directly from Lidar data [5]: at first find several way-points leading to the goal, next, apply IRIS (Iterative Regional Inflation by SDP) algorithm to find free polytopes containing these way-points to form the corridor.



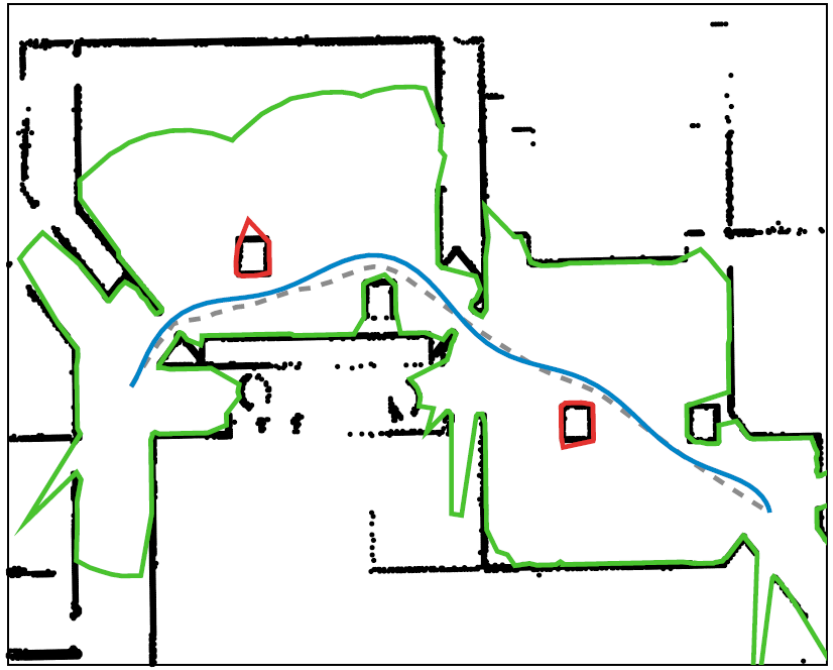
Safe corridor as union of free polytopes generated from Lidar data [5]

[5] Yuxiao Chen et al. “Lidar-based exploration and discretization for mobile robot planning”, inarXiv:2011.10066v1 [cs.RO], 19 November 2020.

Existing approaches on creating polytope map

1. Polytope map
2. Polytope corridor

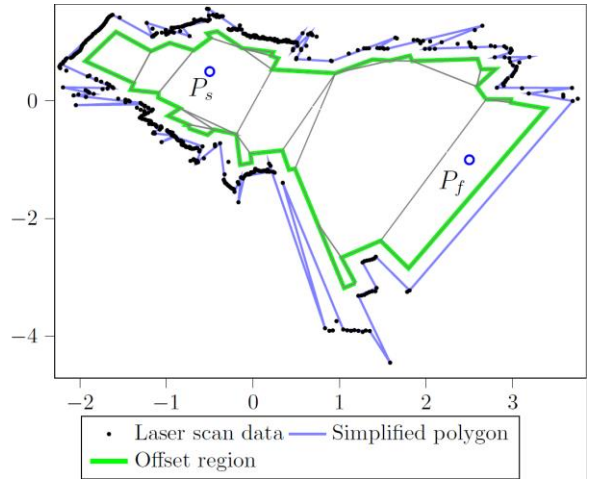
- Generated from an existing occupancy grid map [4]: apply ray-casting techniques to create polygons from different viewpoints and then, unify them.



The corridor given by union of green polygons within a grid map [4]

[4] Christian Meerpohl et al. “Free-space Polygon Creation based on Occupancy Grid Maps for Trajectory Optimization Methods”, in IFAC-PapersOnline, 52(8): 368-374, 2019.

Exercises on creating polytope map



E1: Creating a polytope map from laser scan data

Implement the following procedure [12]:

1) Approximate the obstacle-free space by a polygon region: using Ramer-Douglas-Peucker algorithm, available at https://github.com/fhirschmann/rdp.
2) Shrink the polygon region by a safety offset in order to account for the vehicle's size and possible detection noises: using Gdspy toolbox, available at https://github.com/heitzmann/gdspy.
3) Partition the shrunk polygon into connected polytopes: using Mark Bayazit's algorithm, available at https://github.com/wsilva32/poly_decomp.py

[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.

Exercises on creating polytope map

Related files: laserscan_reading.py, map_reading.py

E1: Creating a polytope map from laser scan data

Implement the following procedure [12]:

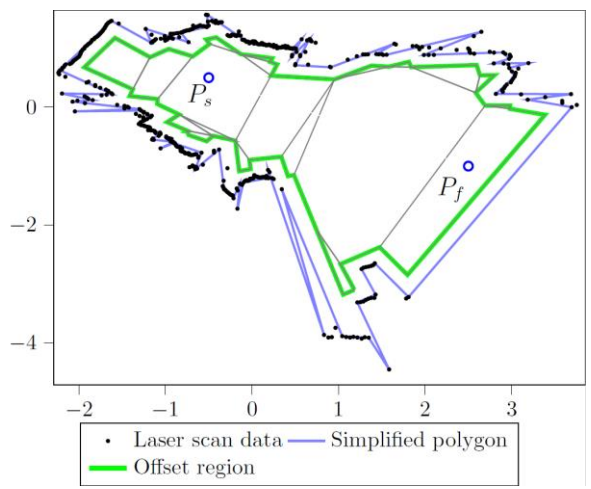
1) Approximate the obstacle-free space by a polygon region: using Ramer-Douglas-Peucker algorithm, available at https://github.com/fhirschmann/rdp.
2) Shrink the polygon region by a safety offset in order to account for the vehicle's size and possible detection noises: using Gdspy toolbox, available at https://github.com/heitzmann/gdspy.
3) Partition the shrunk polygon into connected polytopes: using Mark Bayazit's algorithm, available at https://github.com/wsilva32/poly_decomp.py

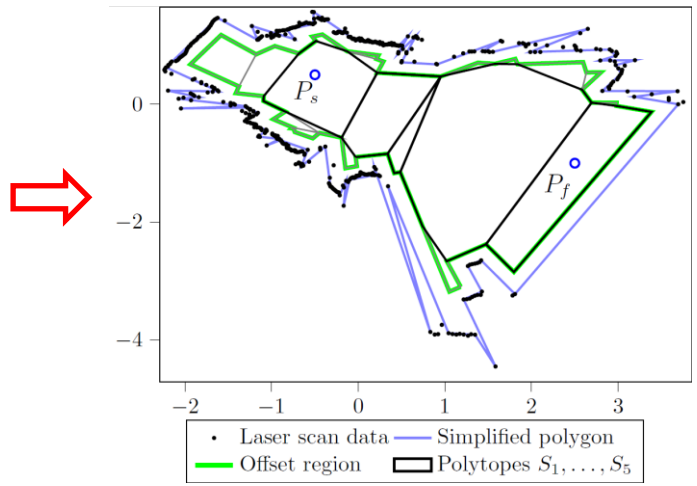
[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

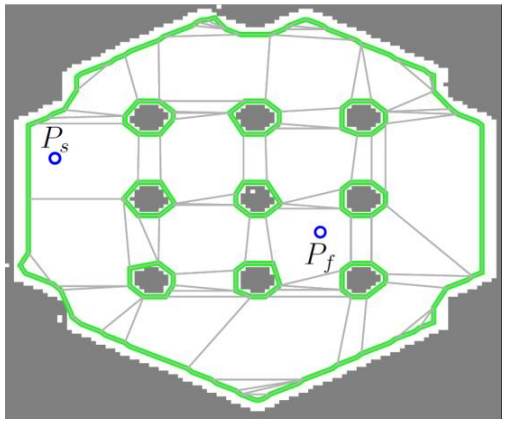
Exercises on creating polytope map

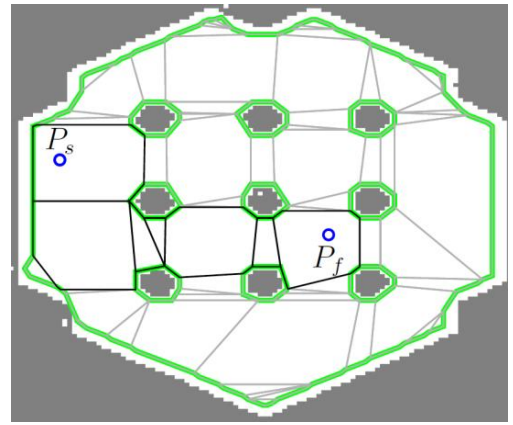
E2: Find a sequence of polytopes connecting two points.

Given a set of polytopes (defined by their vertices) in an arbitrary order and two points within the map, find a sequence of polytopes which connects these two points.







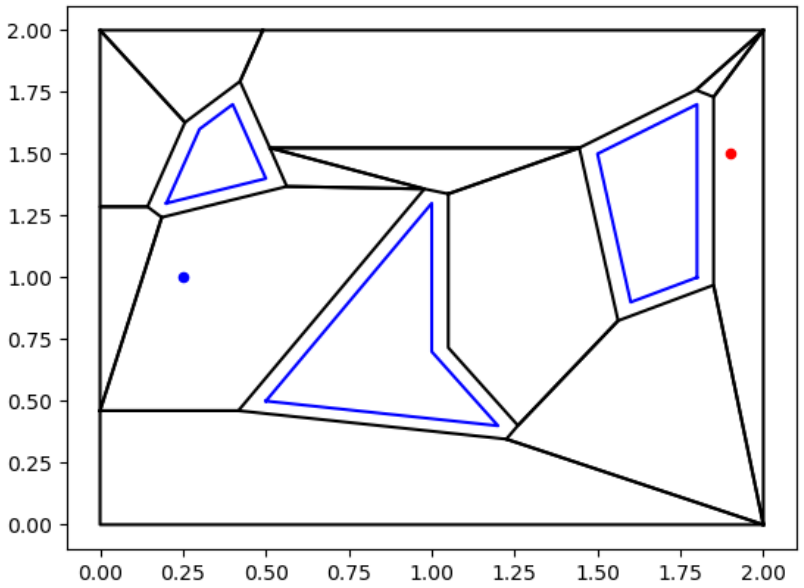


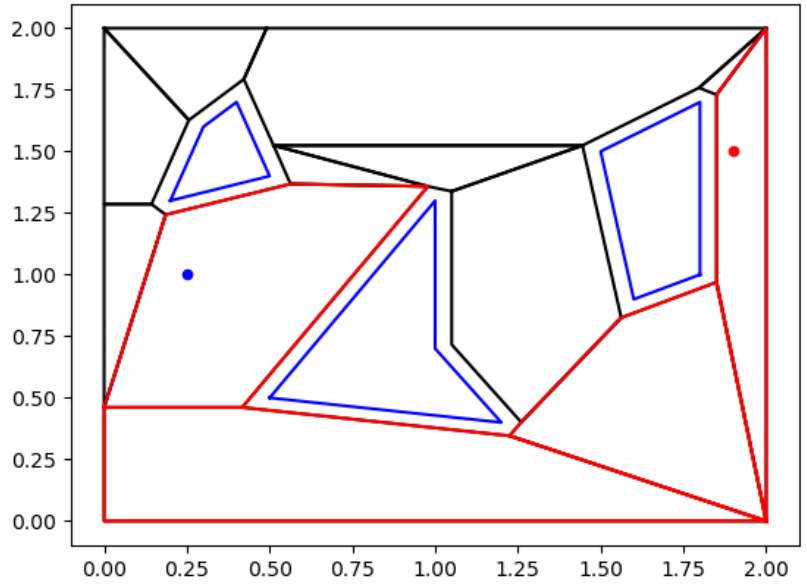
Exercises on creating polytope map

E2: Find a sequence of polytopes connecting two points.

Given a set of polytopes (defined by their vertices) in an arbitrary order and two points within the map, find a sequence of polytopes which connects these two points. Related file: polytope_map_creation.py

Input

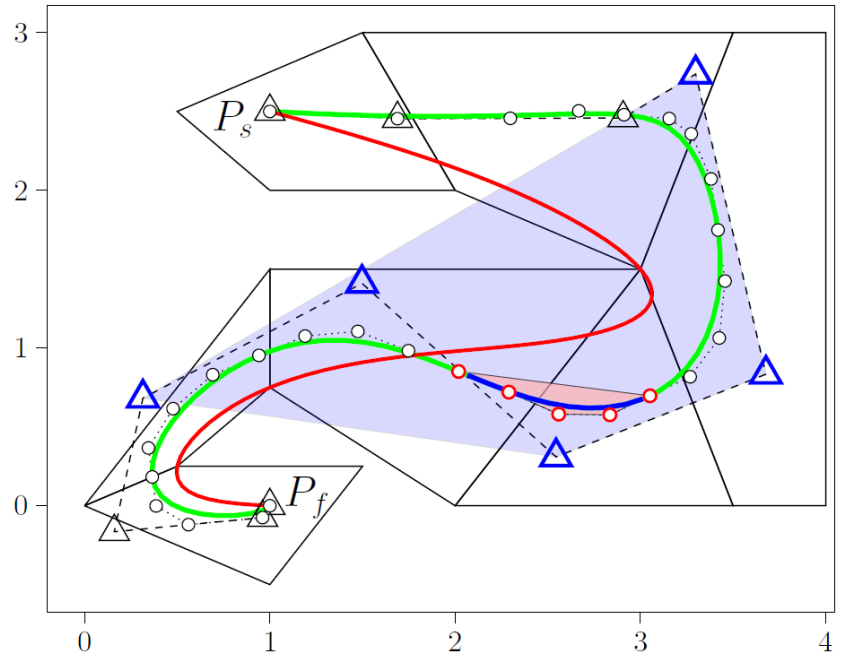




Output

3. B-spline path planner in polytope map

a) B-spline curve and its geometrical properties
b) Path planner in corridor
c) Simulation and experiment results
d) Exercises on B-spline path planner



[8] Ionela Prodan et al. "Necessary and sufficient LMI conditions for constraints satisfaction within a B-spline framework", in Proc. of 58th Conference on Decision and Control (CDC), 8061-8066, 2019
[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021

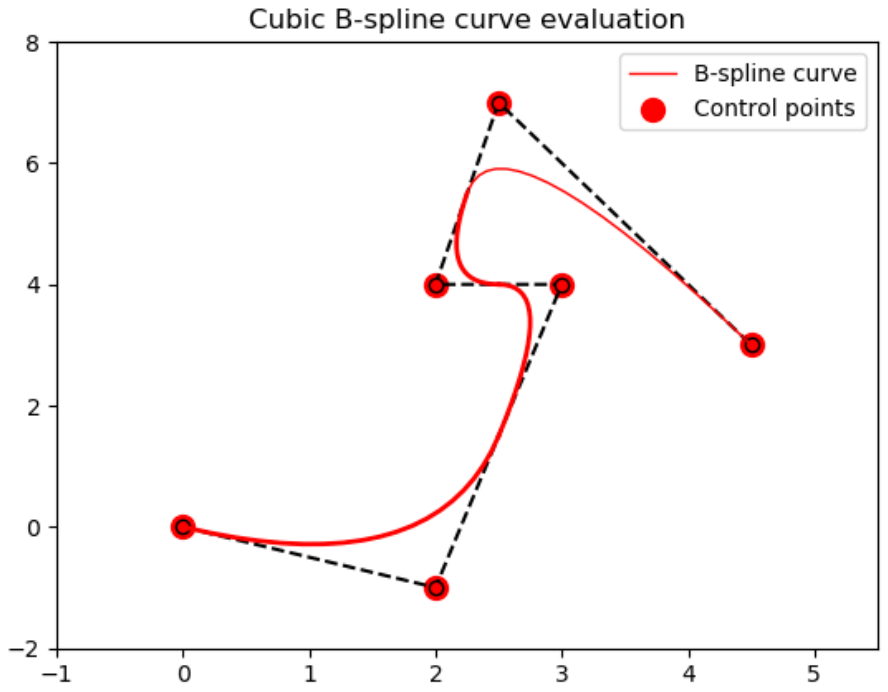
B-spline characterization

- A B-spline curve of degree d are defined by n control points P_i and the basis functions $B_{i,d}(t)$:

$$p(t) = \sum_{i=1}^{n} P_i B_{i,d}(t), t \in [t_s, t_f]$$

- Example of second-order B-spline curve with 6 control points:

$$P_1 = (0,0), \quad P_2 = (2,-1), \quad P_3 = (3,4),$$
$$P_4 = (2,4), \quad P_5 = (2.5,7), \quad P_6 = (4.5,3)$$



B-spline characterization

- A B-spline curve of degree d are defined by n control points P_i and the basis functions $B_{i,d}(t)$:

$$p(t) = \sum_{i=1}^{n} P_i B_{i,d}(t), t \in [t_s, t_f]$$

- The basis functions $B_{i,d}(t)$ are defined based on the knot vector ξ consisting of $(n + d + 1)$ time instants. A clamped and uniform knot vector is given by:

$$\xi = \{\underbrace{\tau_1, \dots, \tau_d}_{t_s}, \underbrace{\tau_{d+1}, \dots, \tau_{n+1}}_{increasing\ equally}, \underbrace{\tau_{n+2}, \dots, \tau_{n+d+1}}_{t_f}\}$$

$$\tau_j = t_s + (j - d - 1)\left(\frac{t_f - t_s}{n - d}\right), d + 1 \le j \le n + 1$$

- E.g. $d = 2, n = 6, t_s = 0, t_f = 4$: $\xi = \{0,0,\ 0,1,2,3,4,\ 4,4\}$.

[6] F. Suryawan, et al, "Splines and polynomial toolsfor flatness-based constrained motion planning". International Journalof Systems Science, 43(8):1396–1411, 2012.
[11] N.T. Nguyen et al, "Flat trajectory design and tracking with saturation guarantees: a nano-drone application". International Journal of Control, 93(6):1266-1279, 2020.

B-splines characterization

- A B-spline curve of degree d are defined by n control points P_i and the basis functions $B_{i,d}(t)$:

$$p(t) = \sum_{i=1}^{n} P_i B_{i,d}(t), t \in [t_s, t_f]$$

- The basis functions $B_{i,d}(t)$ are defined based on the knot vector ξ consisting of $(n + d + 1)$ time instants. A clamped and uniform knot vector is given by:

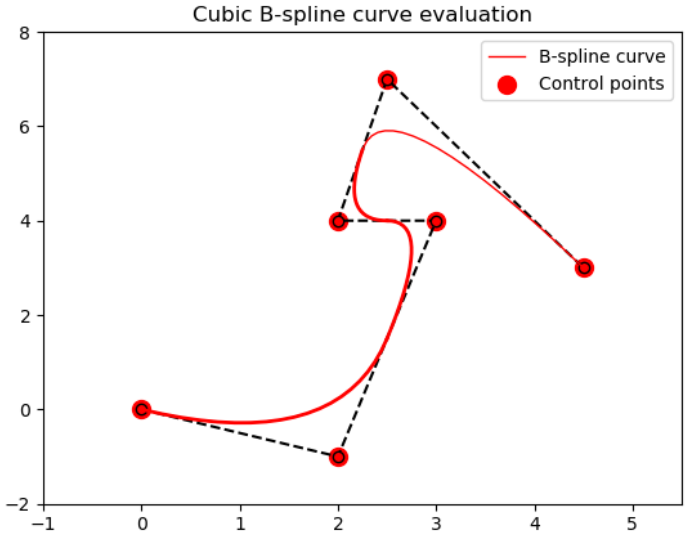
$$\xi = \{\underbrace{\tau_1, \dots, \tau_d}_{t_s}, \underbrace{\tau_{d+1}, \dots, \tau_{n+1}}_{increasing\ equally}, \underbrace{\tau_{n+2}, \dots, \tau_{n+d+1}}_{t_f}\}$$

$$B_{i,0}(t) = \begin{cases} 1, & \forall t \in [\tau_i, \tau_{i+1}) \\ 0, & otherwise \end{cases}$$

$$B_{i,d}(t) = \frac{t - \tau_i}{\tau_{i+d} - \tau_i} B_{i,d-1}(t) + \frac{\tau_{i+d+1} - t}{\tau_{i+d+1} - \tau_{i+1}} B_{i+1,d-1}(t)$$

B-spline curve implementation

- For Matlab, we have to implement the B-spline basis functions. There exists a plotting tool but not very straightforward to use. https://www.mathworks.com/help/curvefit/bspline.html
- For Python, we can use *scipy* toolbox. E.g. plot the second-order B-spline curve with 6 control points: $P_1 = (0,0)$, $P_2 = (2,-1)$, $P_3 = (3,4)$, $P_4 = (2,4)$, $P_5 = (2.5,7)$, $P_6 = (4.5,3)$, $[t_s, t_f] = [0,4]$.



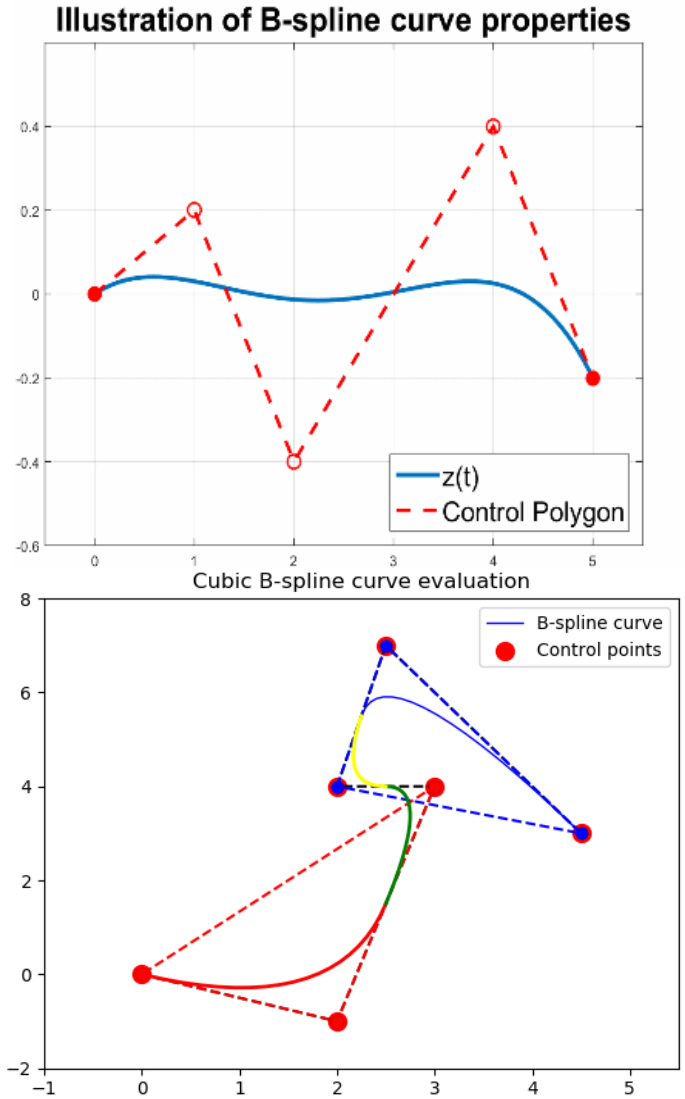

```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
ctr_points = np.array( [(0 , 0), (2, -1), (3, 4), (2, 4), (2.5, 7), (4.5, 3)])
d = 2
knot_vector = [0, 0, 0, 1, 2, 3, 4, 4, 4]
t = np.linspace(0, 4, 100, endpoint=True)
out = interpolate.splev(t, [knot_vector, [ctr_points[:,0], ctr_points[:,1]], d])
plt.figure()
plt.plot(out[0], out[1], label='B-spline curve')
```

Properties of B-spline curve

Geometrical properties: the curve's shape is controlled by the control points:

- The whole curve is contained in the convex hull of all the control points.
- First and last control points are also starting and ending points of the curve.
- Partition of unity: a curve has $n - d$ partitions. The j^{th} partition lies inside the convex hull of $(d + 1)$ control points $\{P_j \cdots P_{j+d}\}$.

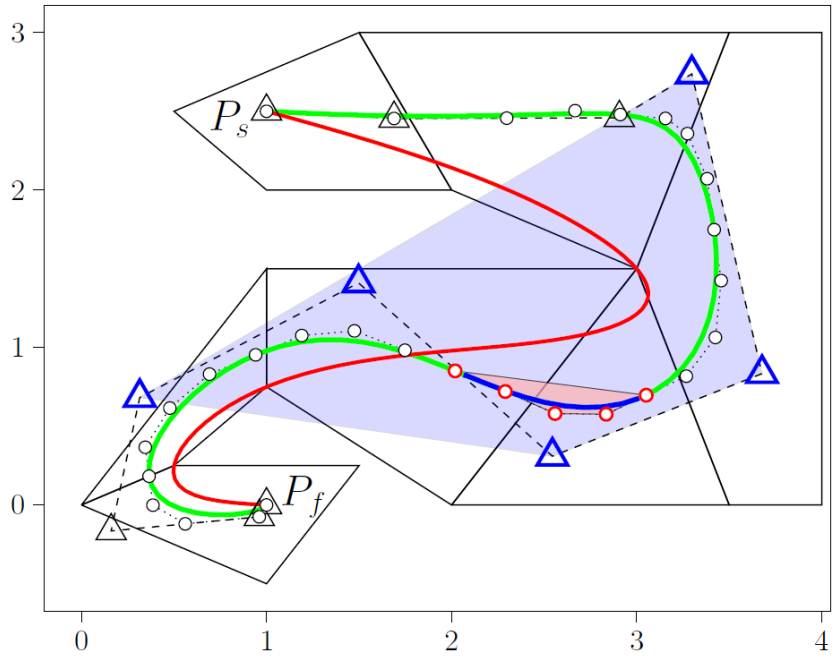
E.g. for the second-order B-spline curve with 6 control points: $P_1 = (0,0)$, $P_2 = (2,-1), P_3 = (3,4), P_4 = (2,4), P_5 = (2.5,7), P_6 = (4.5,3), [t_s, t_f] = [0,4]$.
The 1st partition $t \in [0,1)$ stays inside the triangle $\Delta P_1 P_2 P_3$.
The 4th partition $t \in [3,4]$ stays inside the triangle $\Delta P_4 P_5 P_6$.



The 1st and 4th partitions w.r.t. their control boundaries of a second-order B-spline curve.

3. B-spline path planner in polytope map

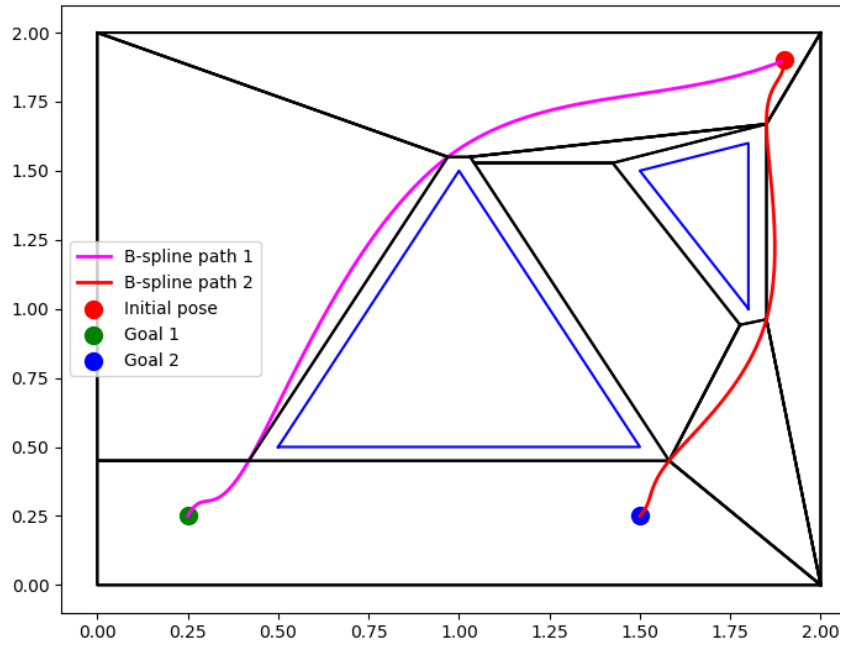
a) B-spline curve and its geometrical properties
b) B-spline path planner in corridor:
   i. using B-spline representation
   ii. using equivalent Bézier representation
c) Simulation and experiment results
d) Exercises on B-spline path planner

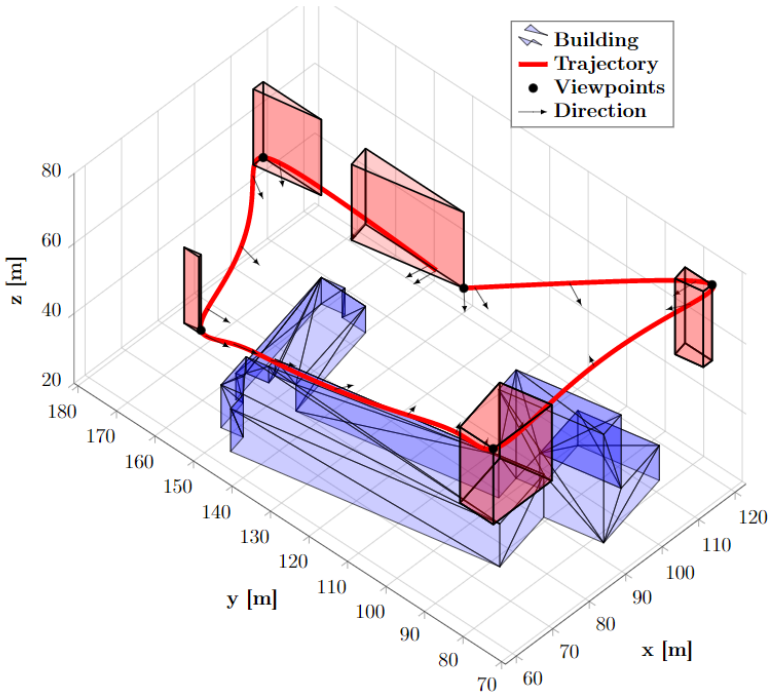


[8] Ionela Prodan et al. "Necessary and sufficient LMI conditions for constraints satisfaction within a B-spline framework", in Proc. of 58th Conference on Decision and Control (CDC), 8061-8066, 2019
[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021

B-spline curve and obstacle avoidance problems

- B-spline parametrization is widely applied for path/trajectory planning thanks to its geometrical properties.
- Can be used in 2D and 3D.
- Being an active research topic in motion planning [6]-[12].





2D and 3D trajectories using B-spline parametrization: avoiding obstacles (blue zones).

[7] Florin Stoican et al. "Flat Trajectory generation for way-points relaxations and obstacle avoidance". In Proc. of the 23rd IEEE Conference Mediterranean Conference on Control and Automation (MED), 695-700, 2015

B-spline path planner in corridor

Problem formulation:

- Path planning from the starting point P_s to the goal point P_f in 2D plane.

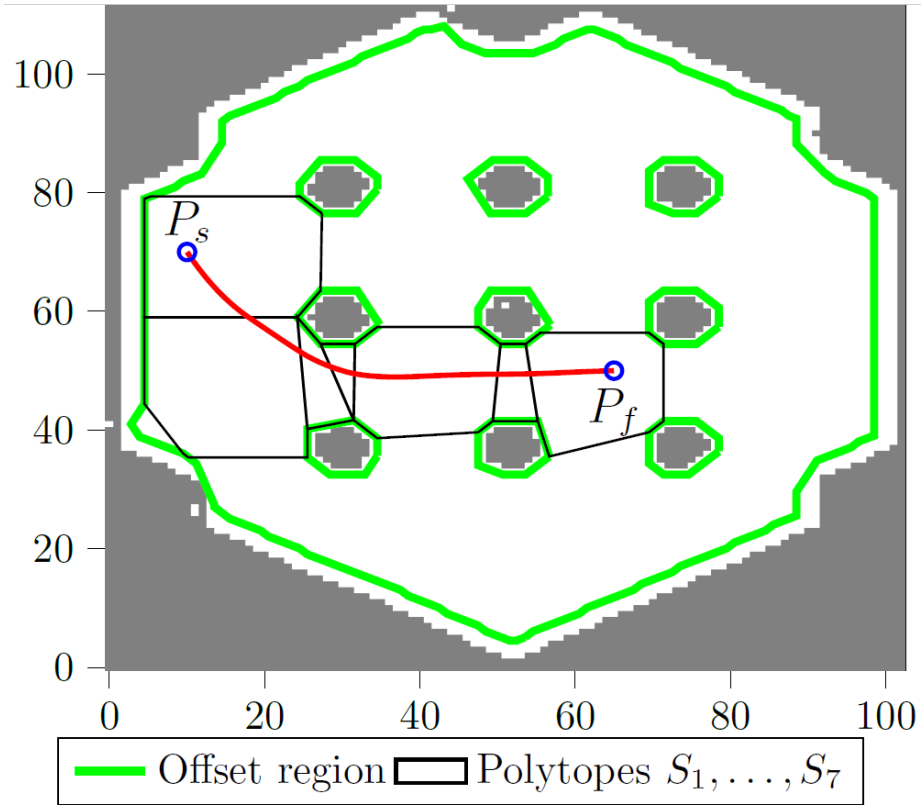
$$p(t): [t_s, t_f] \rightarrow \mathbb{R}^2, \qquad p(t_s) = P_s, \qquad p(t_f) = P_f,$$

with t representing path length, pseudo-time increment, etc. depending on specific circumstances.

- Requirement: the path stay within the non-convex, connected safe corridor:

$$p(t) \in S \triangleq S_1 \cup S_2 \ldots \cup S_q,$$

with the safe corridor defined as union of an ordered list of q connected polytopes, $S \triangleq S_1 \cup S_2 \ldots \cup S_q$.



[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

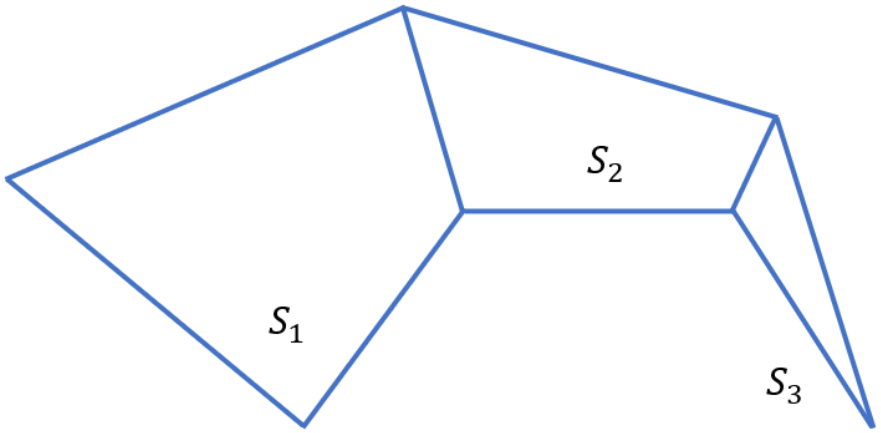
B-spline path planner in corridor

Definitions:

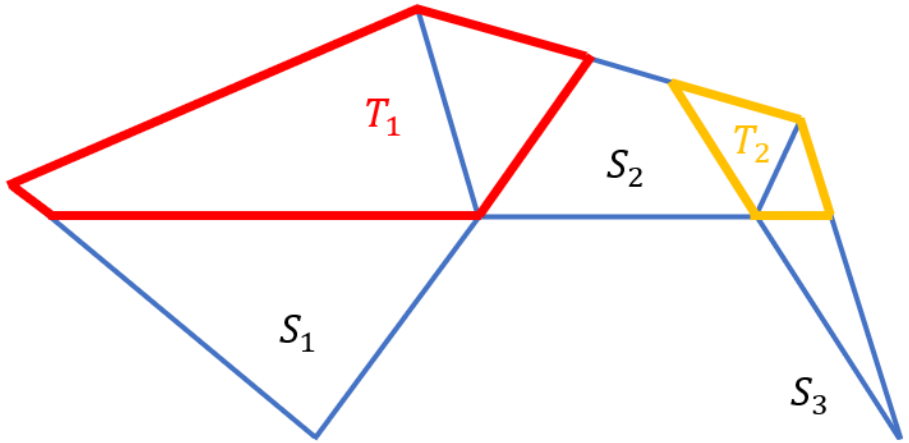
- Transition zone T_i of two consecutive polytopes S_i and S_{i+1}.

  Properties: $T_i \subset (S_i \cup S_{i+1})$.

  $T_i \cup S_i$, $T_i \cup S_{i+1}$ and $T_i \cup S_{i+1} \cup T_{i+1}$ are convex.



Three connected polytopes S_1, S_2, S_3



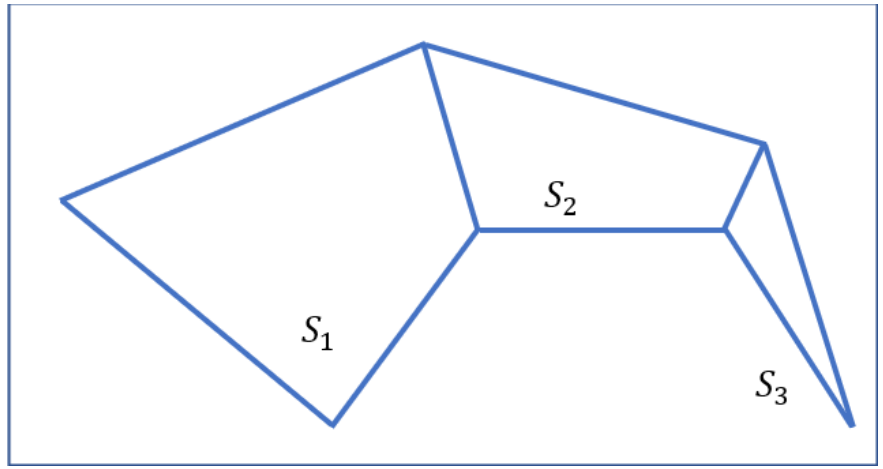
Two transition zones T_1, T_2

B-spline path planner in corridor

Definitions:

- Extended polytope $S_{i,i+1}$ is the extension of S_i towards S_{i+1}

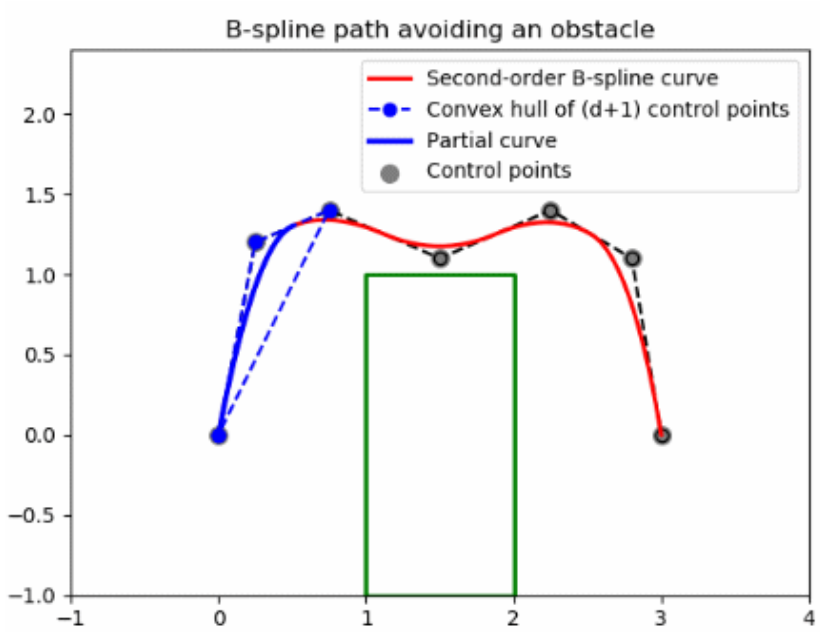
$$S_{i,i+1} = S_i \cup T_i$$



Extended polytopes

General solution towards obstacle avoidance problem

- **Key concept:**
  place the control points such that the union of all the local control boundaries does not collide with any obstacles
  ↔ the union of all the local control boundaries is inside the free space.

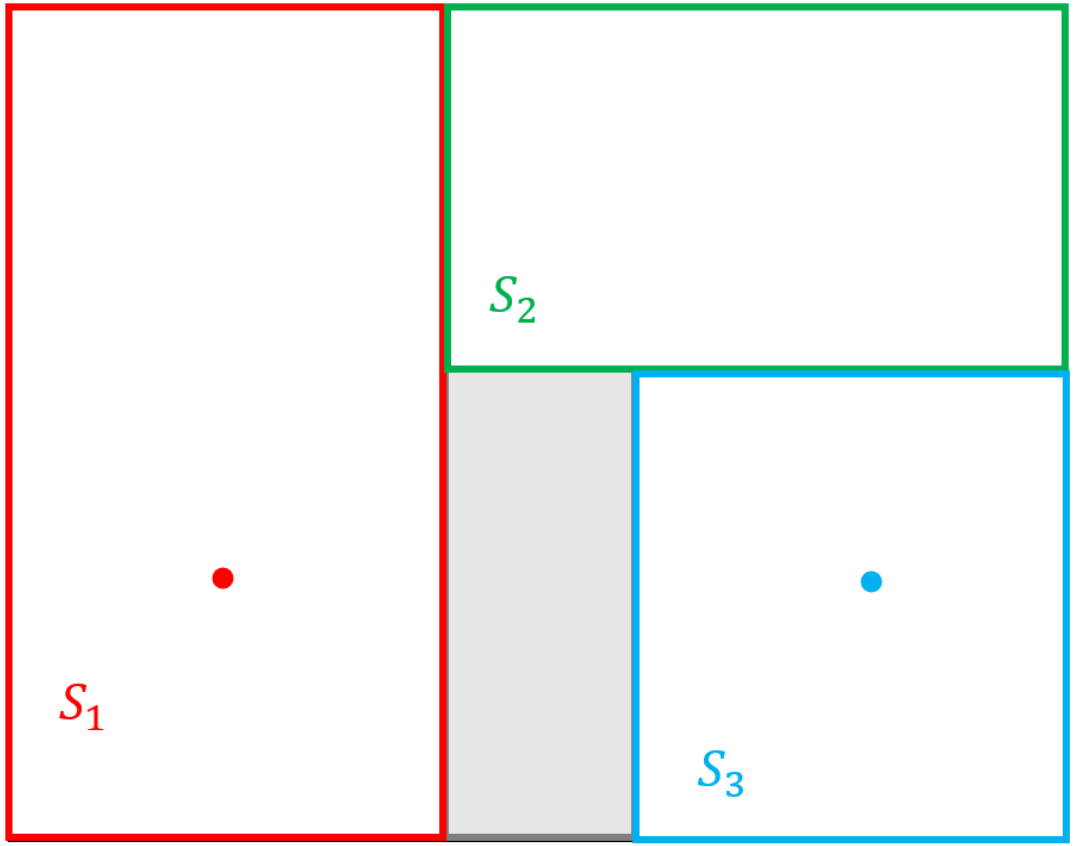


Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding the green rectangle obstacle.

[7] Florin Stoican et al. "Flat Trajectory generation for way-points relaxations and obstacle avoidance". In Proc. of the 23rd IEEE Conference Mediterranean Conference on Control and Automation (MED), 695-700, 2015

Path planner using B-spline representation

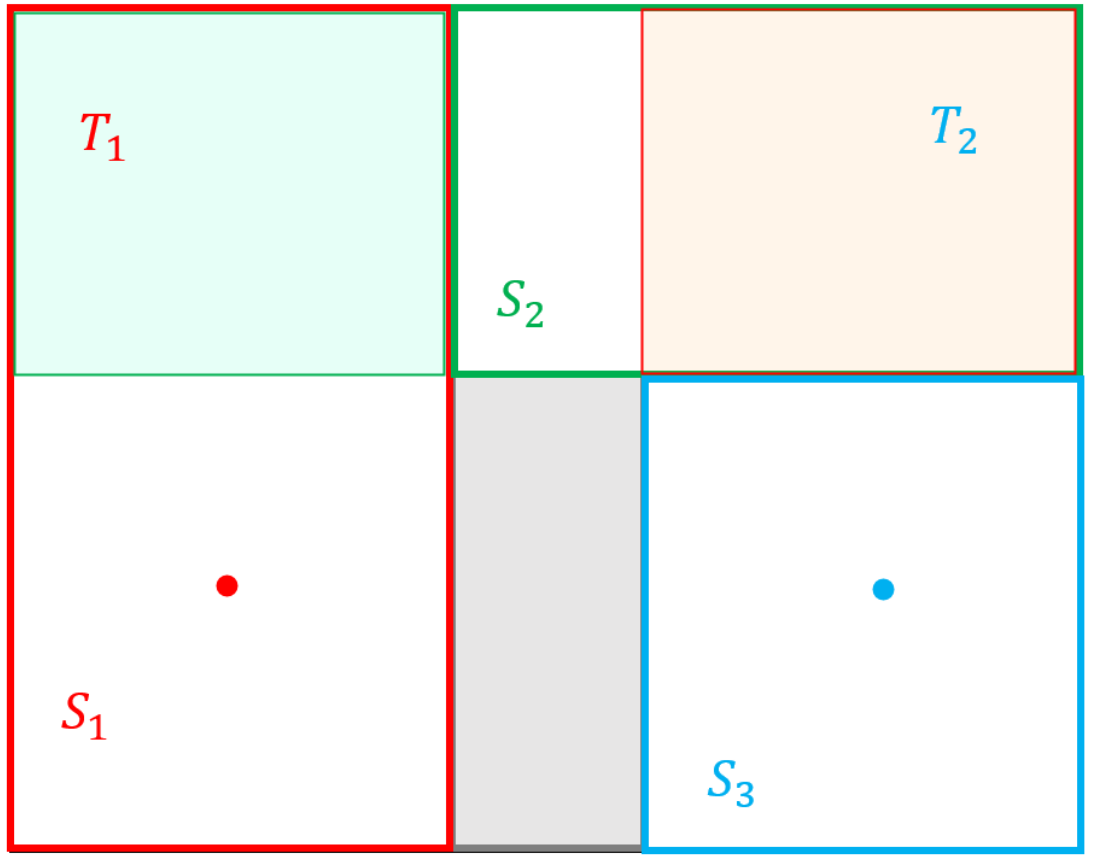
- Find a sequence of connected polytopes leading from the start point to the goal point.



Three connected polytopes $\{S_1, S_2, S_3\}$ forming a safe corridor leading to the goal

Path planner using B-spline representation

- Find a sequence of connected polytopes leading from the start point to the goal point.
- Find the transition zones of each pair of two consecutive polytopes.

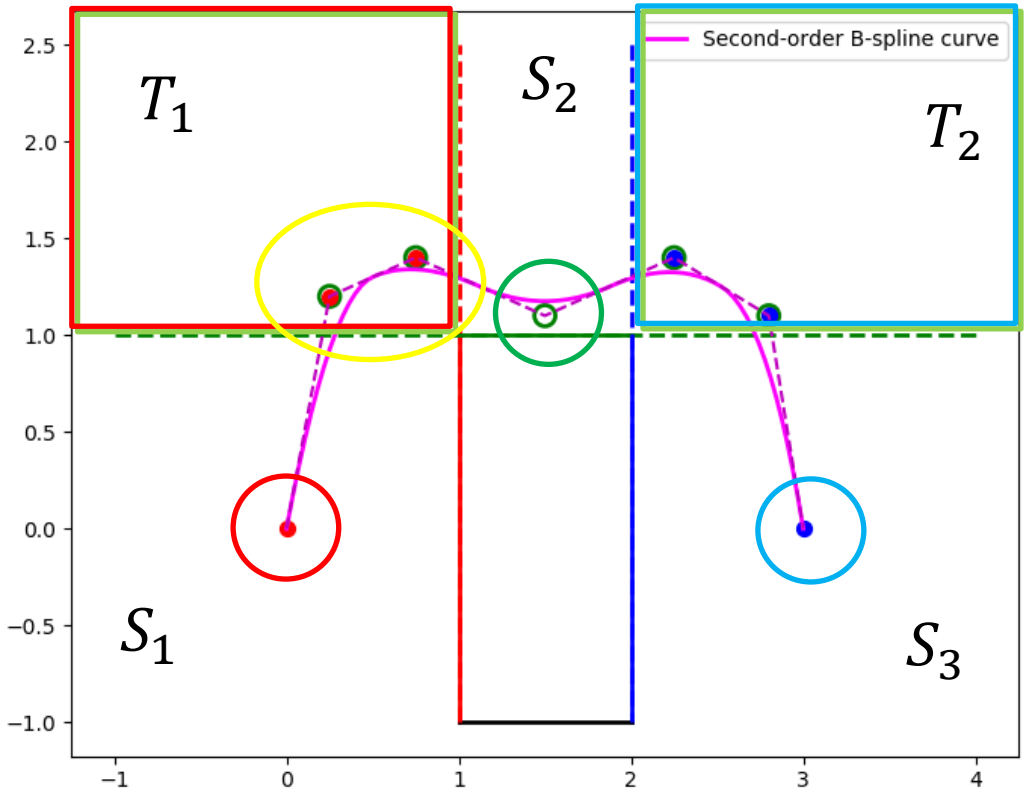


Three connected polytopes $\{S_1, S_2, S_3\}$ and two transition zones $\{T_1, T_2\}$

Path planner using B-spline representation

Having the convex obstacle-free zones $\{S_1, \dots, S_m\}$ and their transition zones $\{T_1, \dots, T_{m-1}\}$:

- Use $n = m + (m-1)d$ control points for a B-spline curve of degree d.
  E.g., $n = 3 + 2 * 2 = 7$
- Place the first control point at the starting pose.
- Place the next d control points inside T_1.
- Place one next control points inside S_2 (not essential but recommended for smoothness and further waypoint-passing problems).
- Continue until finishing the transition zone T_{m-1}.
- Place the final control point at the ending pose.



Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding an obstacle.

Path planner using B-spline representation

Having the convex obstacle-free zones $\{S_1, \dots, S_m\}$ and their transition zones $\{T_1, \dots, T_{m-1}\}$:

- Use $n = m + (m-1)d$ control points for a B-spline curve of degree d.
- Place the first and last control point at the starting and ending poses:

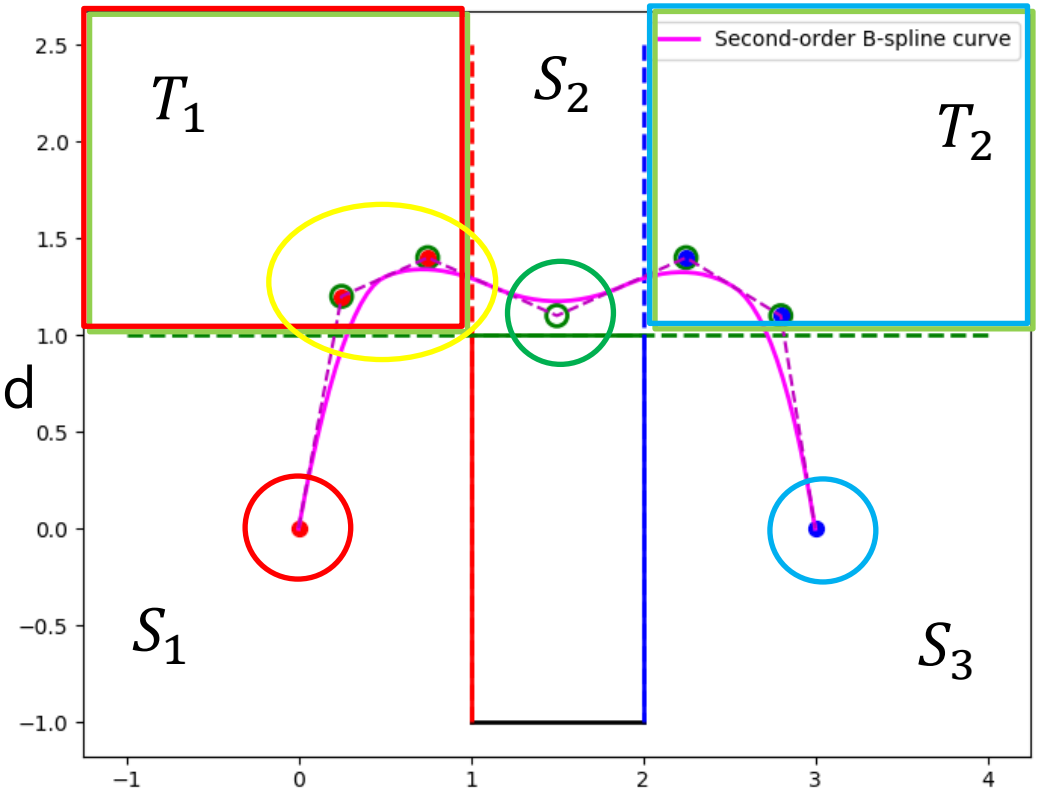
$$P_1 \equiv P_s, \qquad P_n \equiv P_f$$

- Place one control point inside each S_i:

$$P_{(i-1)(d+1)+1} \in S_i, \qquad i \in \{2, \dots, m-1\}$$

- Place d control points inside each transition zone T_j:

$$\{P_{(j-1)(d+1)+2}, \dots, P_{j(d+1)}\} \in T_j, \qquad j \in \{1, \dots, m-1\}$$

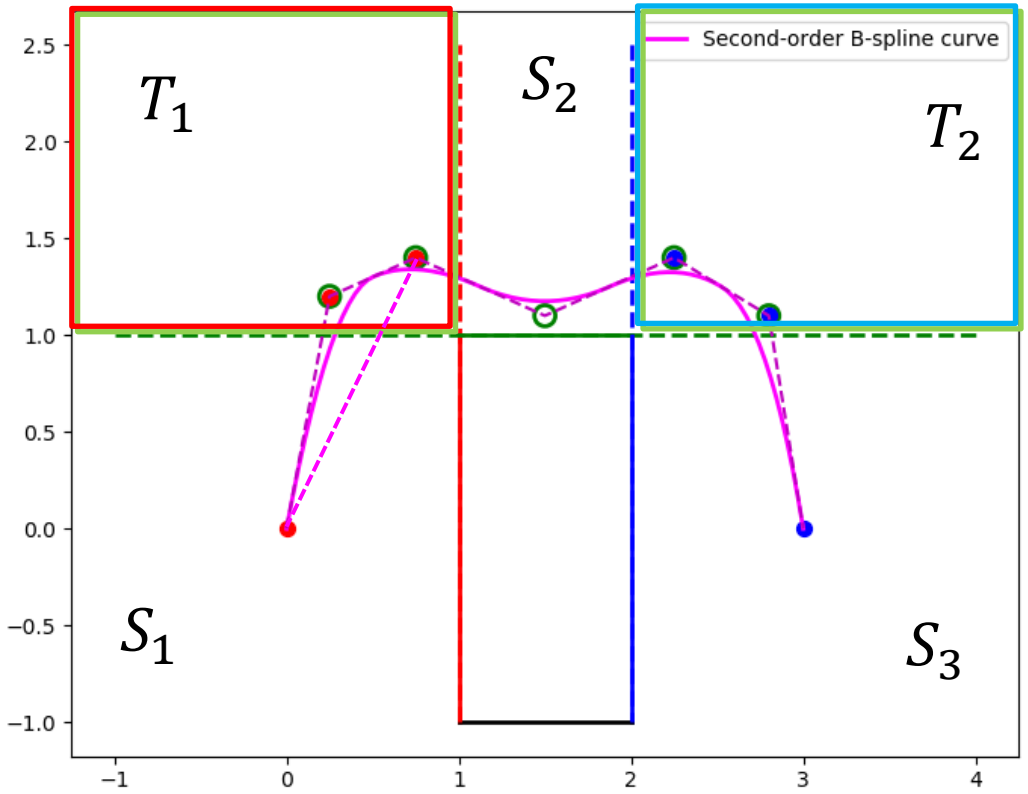


Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding an obstacle.

Path planner using B-spline representation

Proof for the obstacle-free property:

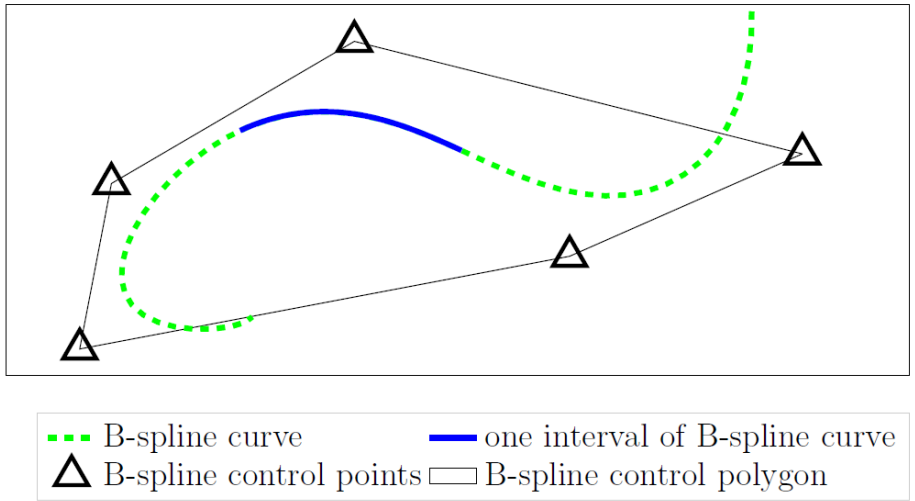
- B-spline property: The j^{th} partition ($j \in [1, n-d]$) lies inside the convex hull of $(d+1)$ control points $\{P_j \cdots P_{j+d}\}$.
- $P_1 \in S_1$, next d control points $\{P_2, \dots, P_{d+1}\} \in T_1$.
- $S_1 \cup T_1$ is convex as definition.
- $Conv(P_1, \dots, P_{d+1})$ is obstacle-free => the 1st partition of the curve is obstacle-free.
- Similar arguments are applied for the rest.



Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding an obstacle.

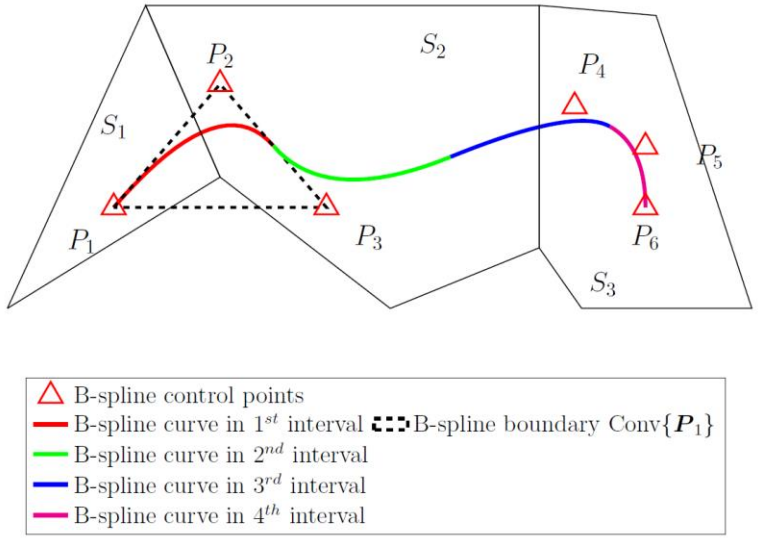
B-spline path planner using Bézier representation

- Problem with B-spline representation:
  The local control polygon formed by the B-spline control points is relatively large w.r.t. the corresponding B-spline section.



B-spline control boundary of the blue section of a fourth-order B-spline curve

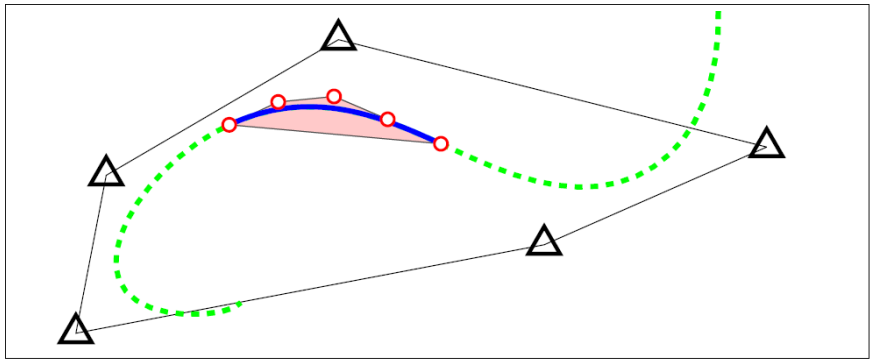
Conservativeness:
The curve is safe while its control boundary reports unsafe.

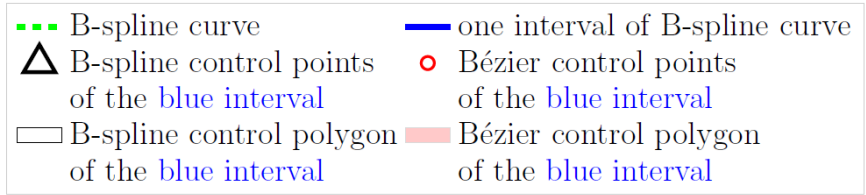


B-spline control boundary of the red section of a second-order B-spline curve

B-spline path planner using Bézier representation

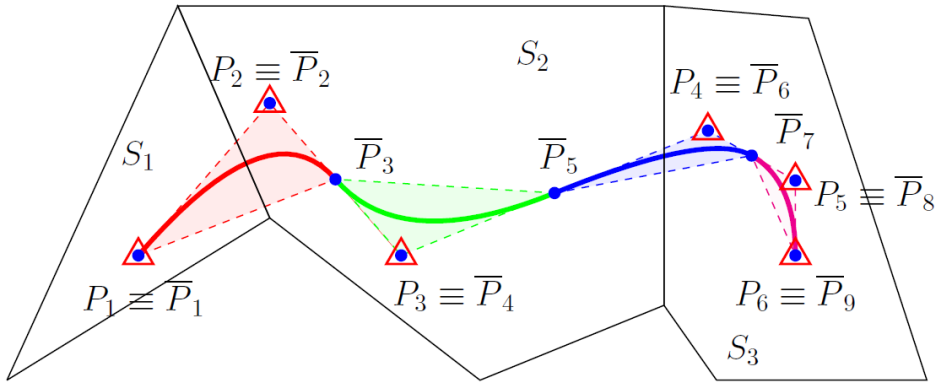
- Solution from Computer-Aid-Design (CAD) community [9]:
  Each B-spline section is a Bézier curve of a same degree, and the equivalent Bézier control boundary is much tighter.

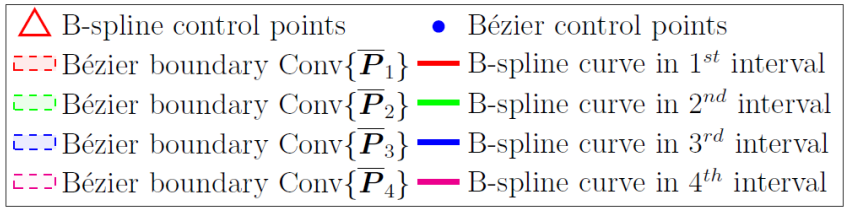




Bézier vs B-spline control boundaries of the blue section of a fourth-order B-spline curve

Bézier control boundaries confirm the safety of the B-spline curve.





Bézier control boundaries of a second-order B-spline curve, fitting inside the corridor

[9] L. Romani and M. A. Sabin: The conversion matrix between uniform B-spline and Bézier representations, Computer Aided Geometric Design, 21(6), 549-560, 2014.

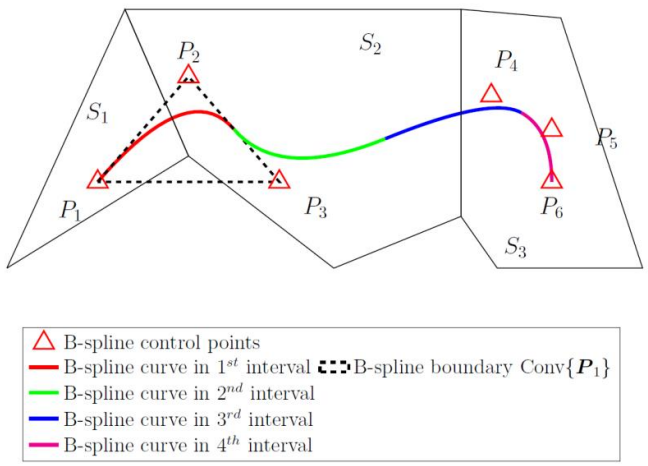
From B-spline to Bézier representation

- The Bézier control points can be calculated using the original B-spline control points. For the j^{th} section of the d-order B-spline curve with n control points:

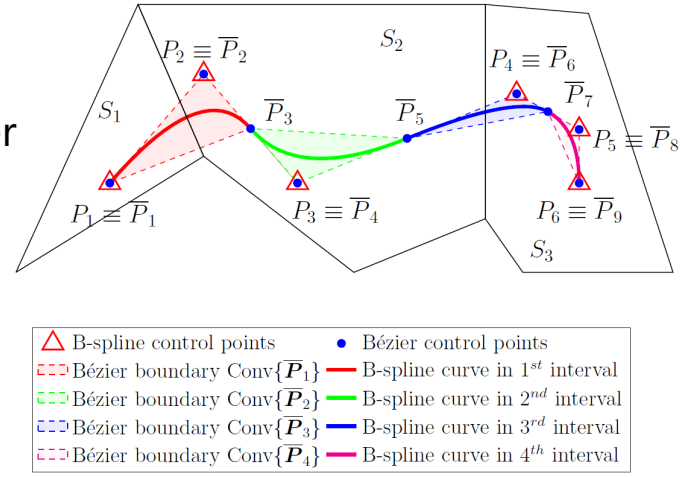
$$\overline{\boldsymbol{P}}_j = \boldsymbol{P}_j A(d,n,j),$$

with $\overline{\boldsymbol{P}}_j \triangleq [\overline{P}_{(j-1)d+1} \dots \overline{P}_{jd+1}]$ and $\boldsymbol{P}_j = [P_j \dots P_{j+d}]$ consisting of (d+1) Bézier and B-spline control points corresponding to the j^{th} section, respectively.

- The matrix $A(d,n,j) \in \mathbb{R}^{(d+1)} \times \mathbb{R}^{(d+1)}$is calculated as in Ref [9].



Equivalent Bézier representation



For the 1st partition:

$$\begin{bmatrix}\bar{P}_1\\ \bar{P}_2\\ \bar{P}_3\end{bmatrix} = \underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 1/2 & 1/2\end{bmatrix}}_{A(2,6,1)} \begin{bmatrix}P_1\\ P_2\\ P_3\end{bmatrix}$$

[9] L. Romani and M. A. Sabin: The conversion matrix between uniform B-spline and Bézier representations, Computer Aided Geometric Design, 21(6), 549-560, 2014.

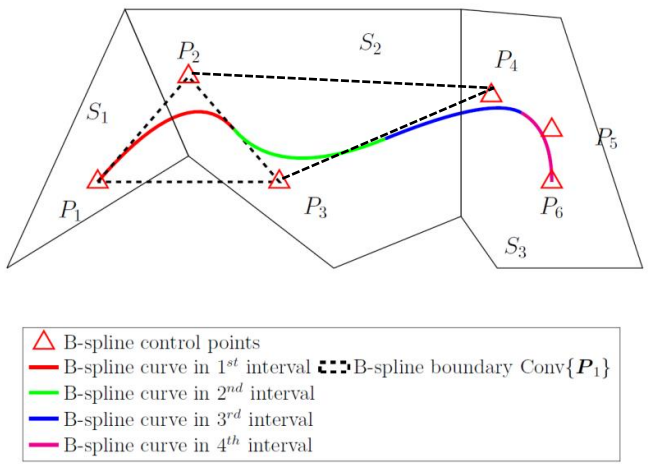
From B-spline to Bézier representation

- The Bézier control points can be calculated using the original B-spline control points. For the j^{th} section of the d-order B-spline curve with n control points:

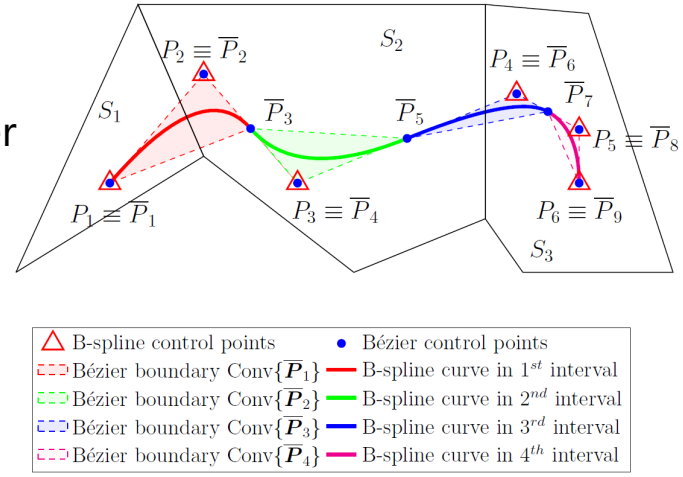
$$\overline{\boldsymbol{P}}_j = \boldsymbol{P}_j A(d,n,j),$$

with $\overline{\boldsymbol{P}}_j \triangleq [\overline{P}_{(j-1)d+1} \dots \overline{P}_{jd+1}]$ and $\boldsymbol{P}_j = [P_j \dots P_{j+d}]$ consisting of (d+1) Bézier and B-spline control points corresponding to the j^{th} section, respectively.

- The matrix $A(d,n,j) \in \mathbb{R}^{(d+1)} \times \mathbb{R}^{(d+1)}$is calculated as in Ref [9].



Equivalent Bézier representation



For the 2nd partition:

$$\begin{bmatrix}\overline{P}_3\\ \overline{P}_4\\ \overline{P}_5\end{bmatrix} = \underbrace{\begin{bmatrix}1/2 & 1/2 & 0\\ 0 & 1 & 0\\ 0 & 1/2 & 1/2\end{bmatrix}}_{A(2,6,2)=A(2,6,3)} \begin{bmatrix}P_2\\ P_3\\ P_4\end{bmatrix}$$

[9] L. Romani and M. A. Sabin: The conversion matrix between uniform B-spline and Bézier representations, Computer Aided Geometric Design, 21(6), 549-560, 2014.

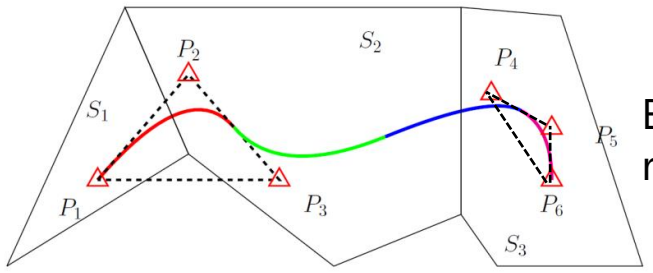
From B-spline to Bézier representation

- The Bézier control points can be calculated using the original B-spline control points. For the j^{th} section of the d-order B-spline curve with n control points:

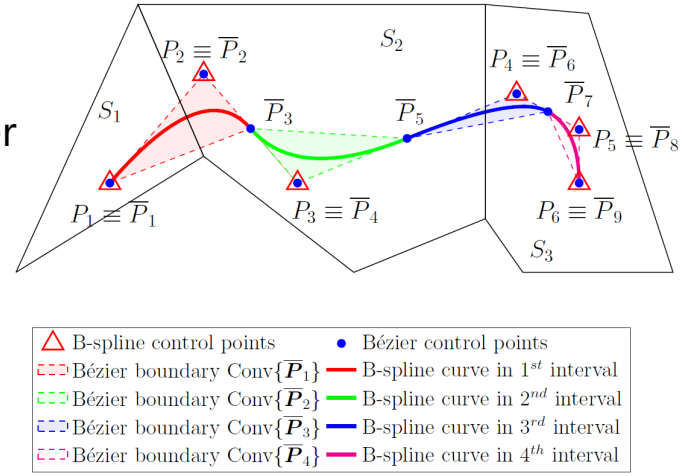
$$\overline{\boldsymbol{P}}_j = \boldsymbol{P}_j A(d,n,j),$$

with $\overline{\boldsymbol{P}}_j \triangleq [\overline{P}_{(j-1)d+1} \dots \overline{P}_{jd+1}]$ and $\boldsymbol{P}_j = [P_j \dots P_{j+d}]$ consisting of (d+1) Bézier and B-spline control points corresponding to the j^{th} section, respectively.

- The matrix $A(d,n,j) \in \mathbb{R}^{(d+1)} \times \mathbb{R}^{(d+1)}$is calculated as in Ref [9].



Equivalent Bézier representation



For the 4th partition:

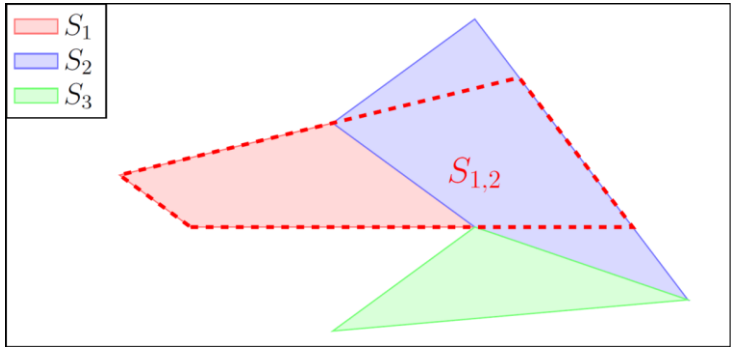
$$\begin{bmatrix} \bar{P}_7 \\ \bar{P}_8 \\ \bar{P}_9 \end{bmatrix} = \underbrace{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{A(2,6,4)} \begin{bmatrix} P_4 \\ P_5 \\ P_6 \end{bmatrix}$$

[9] L. Romani and M. A. Sabin: The conversion matrix between uniform B-spline and Bézier representations, Computer Aided Geometric Design, 21(6), 549-560, 2014.

B-spline path planner using Bézier representation

Conditions for a B-spline curve of degree d to stay inside the safe corridor and to satisfy the endpoint constraints:

Given the sequence of extended polytopes $S_{j,j+1}$ (j $\in \{1, \dots, m\}$, $S_{m,m+1} \equiv S_m$):



Extended polytopes

- Use $n = m + d$ control points.
- Place the first and last control point at the starting and ending poses:

$$P_1 \equiv P_s, \qquad P_n \equiv P_f$$

- Enforce each interval inside one extended polytope by constraining its Bézier control boundary:

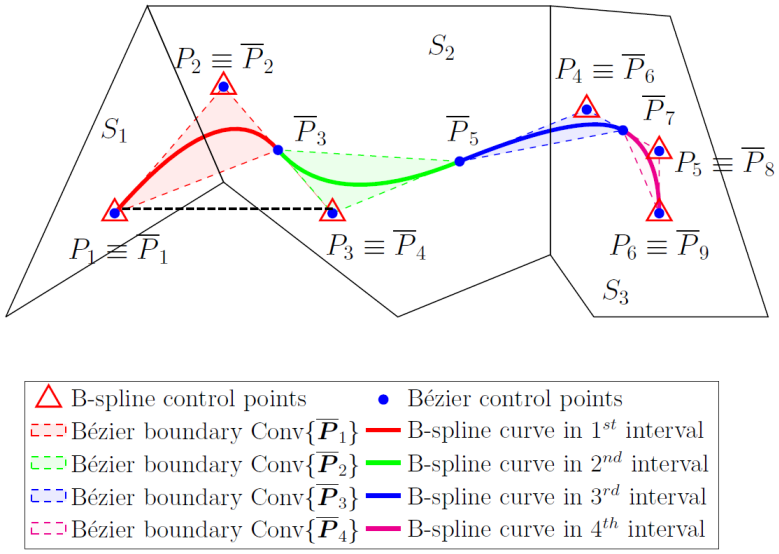
$$\bar{P}_k \in S_{j,j+1}, \qquad \forall \bar{P}_k \in \bar{\boldsymbol{P}}_j, \qquad j \in \{1, \dots, m\},$$

With $\bar{\boldsymbol{P}}_j = \boldsymbol{P}_j A(d, n, j)$, consisting of $(d+1)$ Bézier control points given in terms of $(d+1)$ B-spline control points $\boldsymbol{P}_j$, both corresponding to the j^{th} section of the curve.

[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

B-spline path planner using Bézier representation

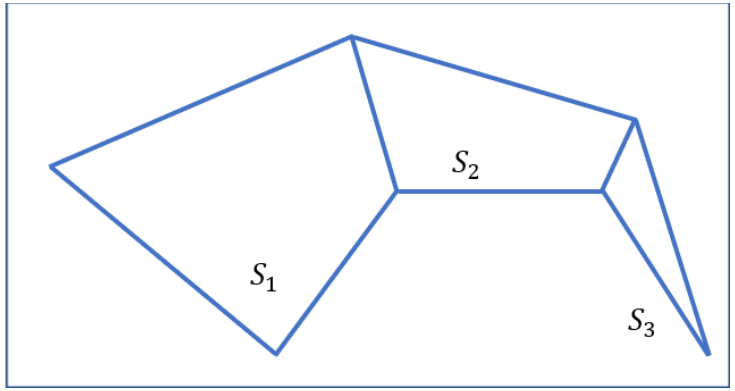
- Advantages w.r.t. using B-spline representation:
  - Less conservative, more freedom to place the control points.



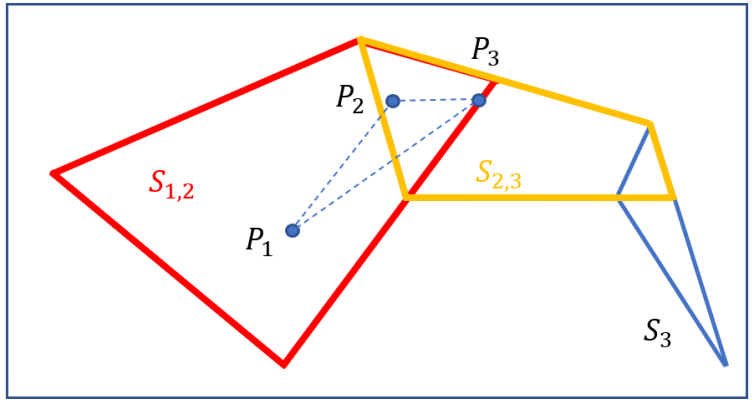
Bézier control boundaries of a second-order B-spline curve, fitting inside the corridor which allow the usage of the chosen B-spline control points.

B-spline path planner using Bézier representation

- Advantages w.r.t. using B-spline representation:
  - Less conservative, more freedom to place the control points.
  - Constraints defined with clear interpretation, “each section inside each extended polytope”, which is unable to enforce with B-spline representation.



Extended polytopes

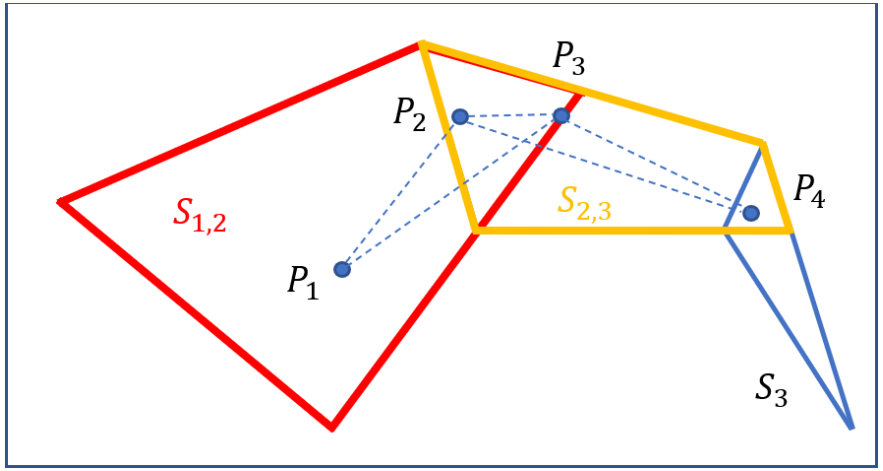


Constraints:

1) $\Delta P_1 P_2 P_3 \subseteq S_{1,2}$

B-spline path planner using Bézier representation

- Advantages w.r.t. using B-spline representation:
  - Less conservative, more freedom to place the control points.
  - Constraints defined with clear interpretation, “each section inside each extended polytope”, which is unable to enforce with B-spline representation.



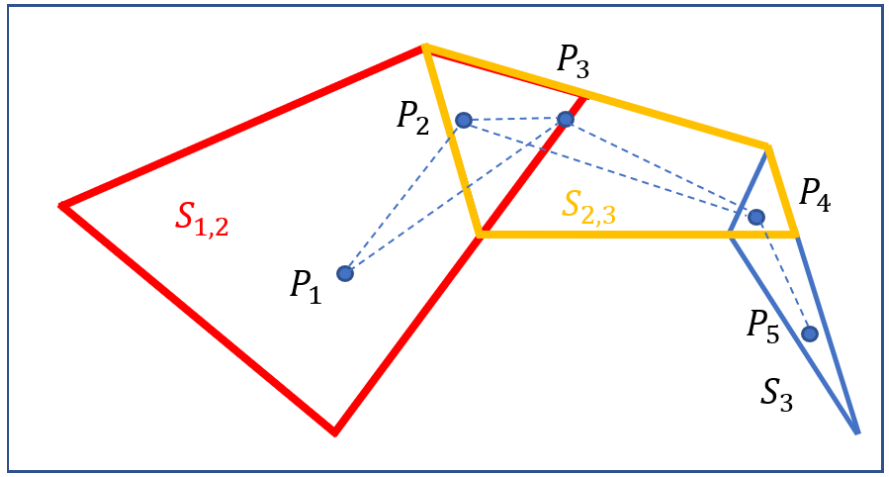
Constraints:

1) $\Delta P_1 P_2 P_3 \subseteq S_{1,2}$

2) $\Delta P_2 P_3 P_4 \subseteq S_{2,3}$

B-spline path planner using Bézier representation

- Advantages w.r.t. using B-spline representation:
  - Less conservative, more freedom to place the control points.
  - Constraints defined with clear interpretation, "each section inside each extended polytope", which is unable to enforce with B-spline representation.



Constraints:

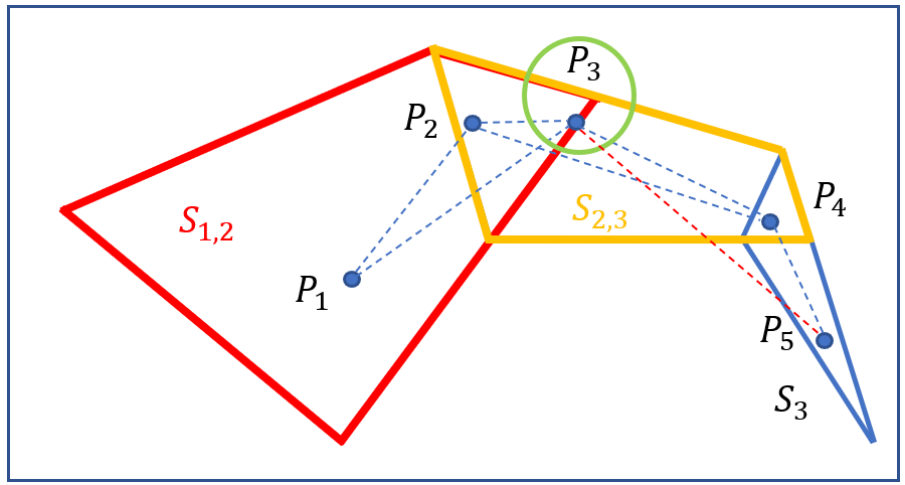
1) $\Delta P_1 P_2 P_3 \subseteq S_{1,2}$

2) $\Delta P_2 P_3 P_4 \subseteq S_{2,3}$

3) $\Delta P_3 P_4 P_5 \subseteq S_3$

B-spline path planner using Bézier representation

- Advantages w.r.t. using B-spline representation:
  - Less conservative, more freedom to place the control points.
  - Constraints defined with clear interpretation, "each section inside each extended polytope", which is unable to enforce with B-spline representation.



Constraints:

1) $\Delta P_1 P_2 P_3 \subseteq S_{1,2}$

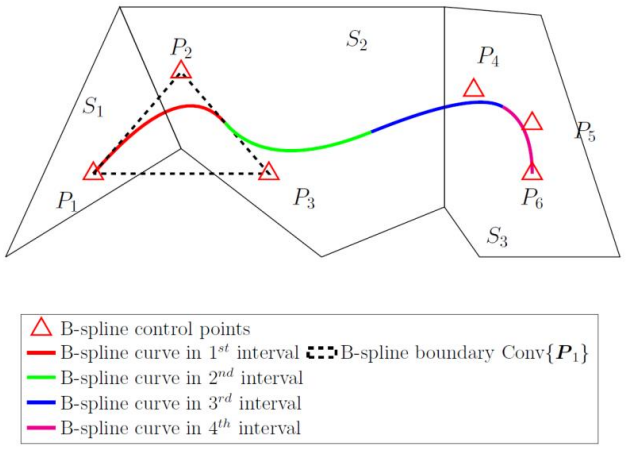
2) $\Delta P_2 P_3 P_4 \subseteq S_{2,3}$

3) $\Delta P_3 P_4 P_5 \subseteq S_3$

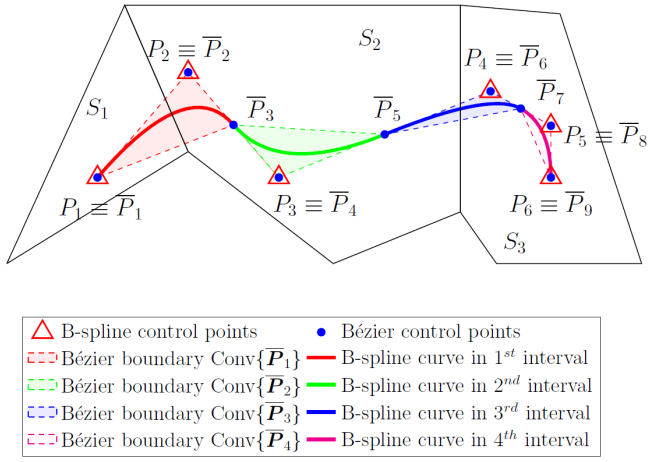
$\Rightarrow P_3 \in (S_{1,2} \cap S_{2,3} \cap S_3)$ *(empty)*

B-spline path planner using Bézier representation

- Advantages w.r.t. using B-spline representation:
  - Less conservative, more freedom to place the control points.
  - Constraints defined with clear interpretation, “each section inside each extended polytope”, which is unable to enforce with B-spline representation.
  - Exact position calculation of starting and ending points of each section along the B-spline curve.



With B-spline representation, we only know that the first and last control points are the starting and ending points of the B-spline curve.



With Bézier representation, we know that the first and last control points as well as midpoints of any pairs of two consecutive control points (!!) are also way-points of the 2-order B-spline curve.

3. B-spline path planner in polytope map

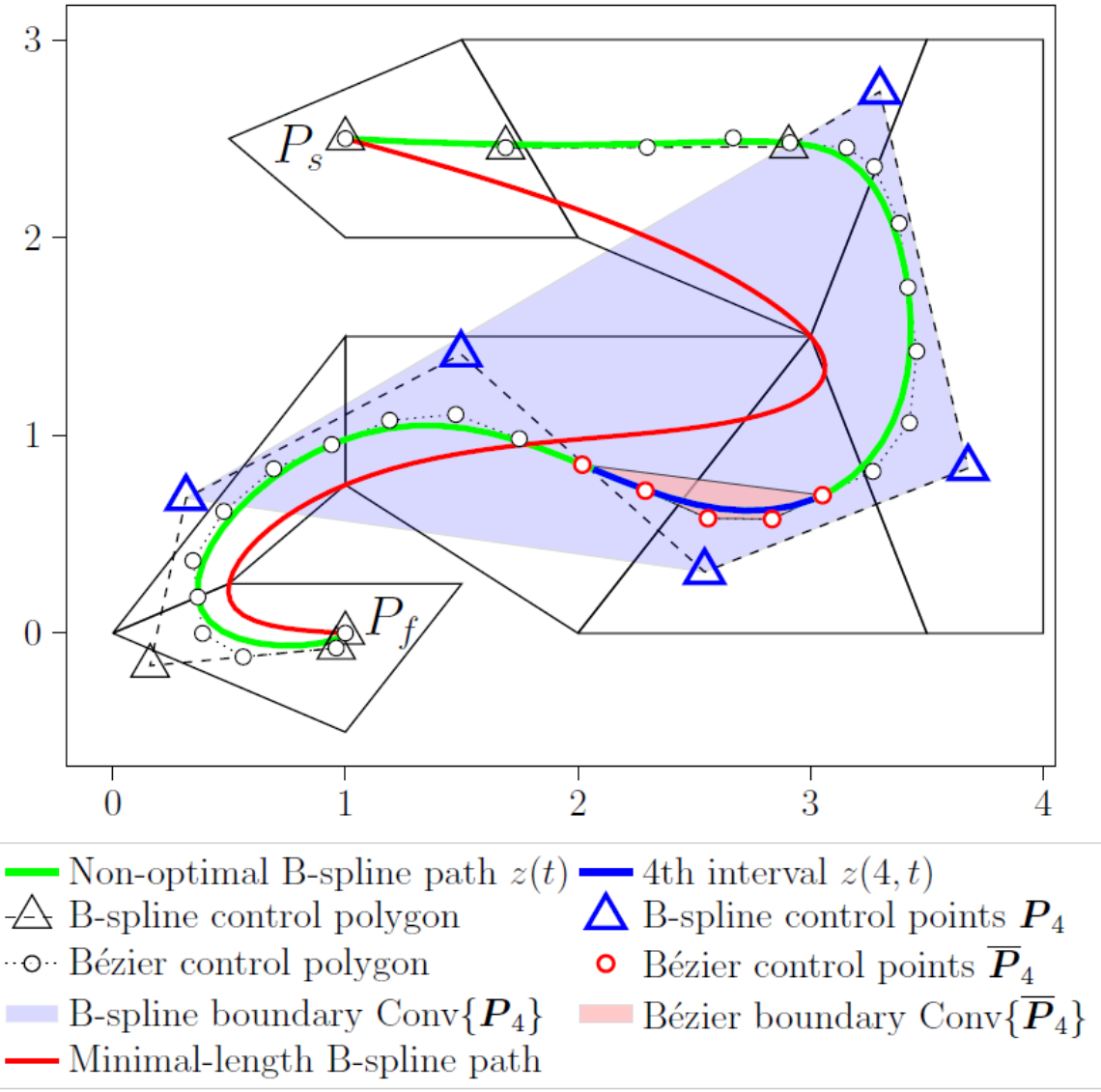
a) B-spline curve and its geometrical properties
b) B-spline path planner in corridor:
   i. using B-spline representation
   ii. using equivalent Bézier representation
c) Simulation and experiment results
d) Exercises on B-spline path planner

[12] Ngoc Thinh Nguyen et al. “B-spline path planner for safe navigation of mobile robots”, in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

Simulation results

Parameters	Value
Degree d	4
Number of B-spline control points n	11
Number of equivalent Bézier control points	29

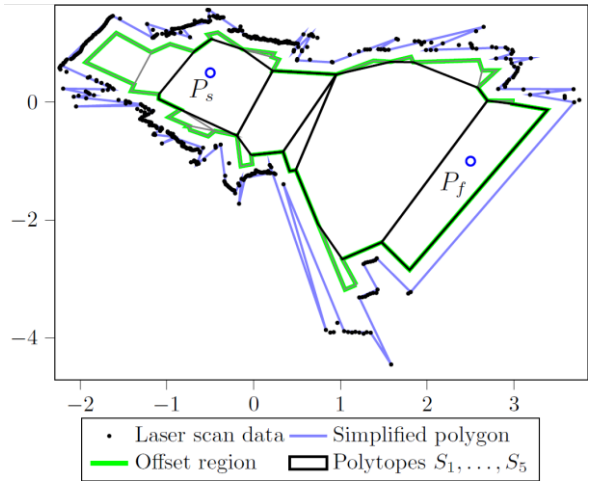
- The green curve is one feasible solution (manually chosen) while the red one shows the minimal-length path.
- Regarding the blue section, the Bézier boundary (red polytope) is significantly tighter than the B-spline control boundary (blue polytope).
- The average computation time is 45 ms for 100 samples using solver IPOPT with Pyomo in Python 3.

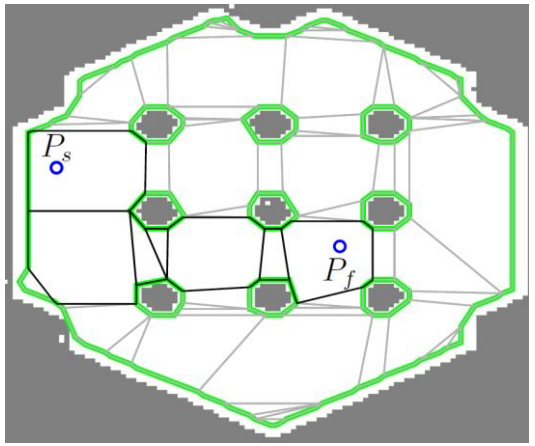


Experiment results

Implementation procedure:

1) Approximate the obstacle-free space by a polygon region: using Ramer-Douglas-Peucker algorithm, available at https://github.com/fhirschmann/rdp.
2) Shrink the polygon region by a safety offset in order to account for the vehicle's size and possible detection noises: using Gdspy toolbox, available at https://github.com/heitzmann/gdspy.
3) Partition the shrunk polygon into connected polytopes: using Mark Bayazit's algorithm, available at https://github.com/wsilva32/poly_decomp.py
4) Perform a graph search in order to obtain the sequence of polytopes leading from the initial pose to the desired goal.



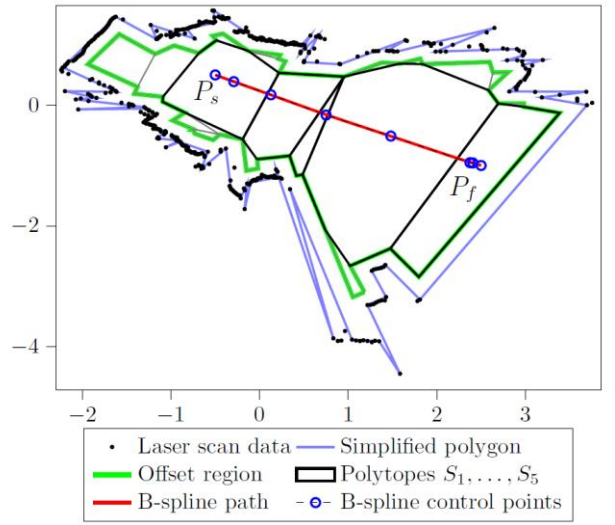


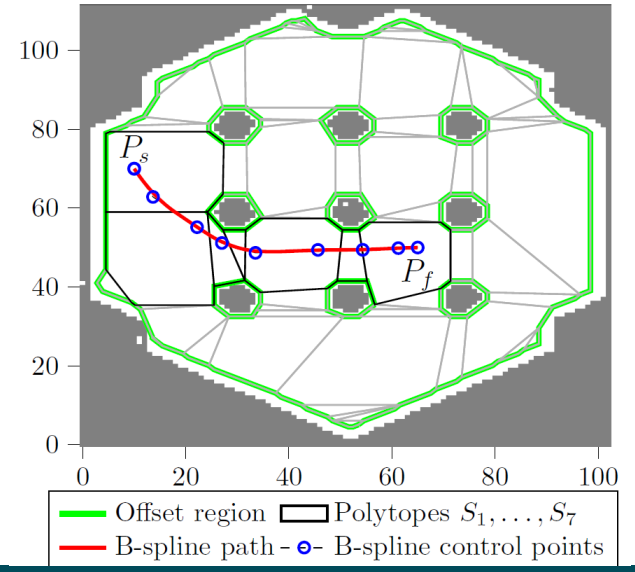
Exercises from Section I

Experiment results

Implementation procedure:

1) Approximate the obstacle-free space by a polygon region: using Ramer-Douglas-Peucker algorithm, available at https://github.com/fhirschmann/rdp.
2) Shrink the polygon region by a safety offset in order to account for the vehicle's size and possible detection noises: using Gdspy toolbox, available at https://github.com/heitzmann/gdspy.
3) Partition the shrunk polygon into connected polytopes: using Mark Bayazit's algorithm, available at https://github.com/wsilva32/poly_decomp.py
4) Perform a graph search in order to obtain the sequence of polytopes leading from the initial pose to the desired goal.
5) Solve the path planning problem and return the optimal path.





Experiment results

Specifications:
- Differential wheeled Jackal robot from Clearpath Robotics.
- Ouster OS1-16 LIDAR.
- On-board computer using Intel Core i5-4750 processor

[12] Ngoc Thinh Nguyen et al. “B-spline path planner for safe navigation of mobile robots”, in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

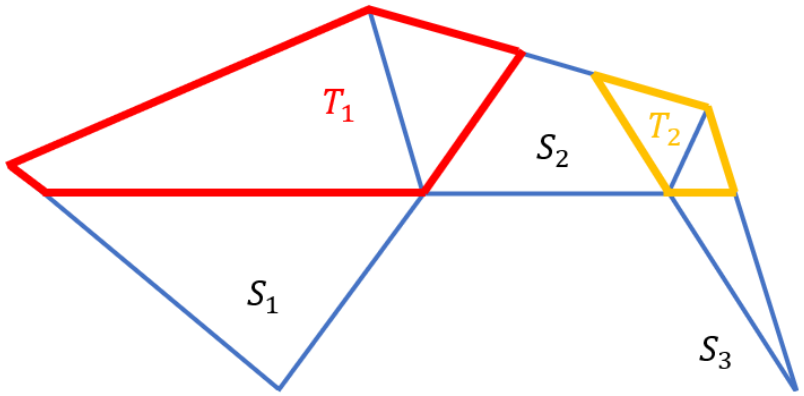
Exercises on planning B-spline path in polytope corridor

E3: Geometric computation

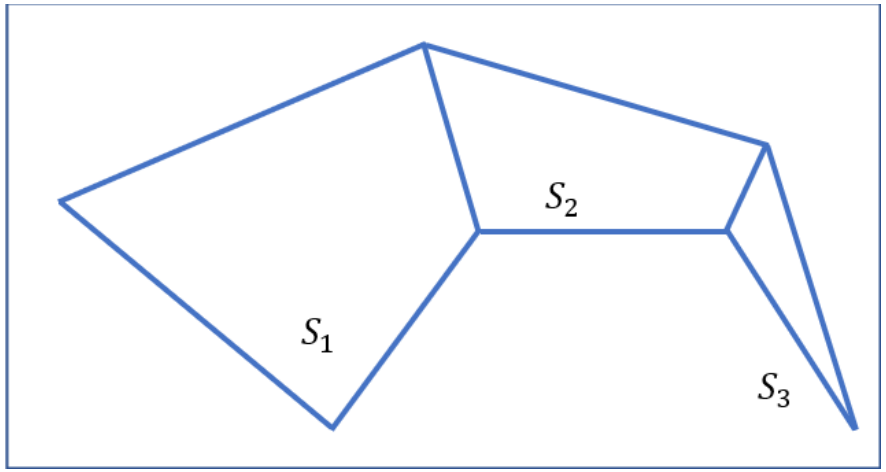
Given two connected polytopes defined by their vertices, implement functions which return:

a) Transition zones

b) Extended polytopes



Two transition zones T_1, T_2



Extended polytopes

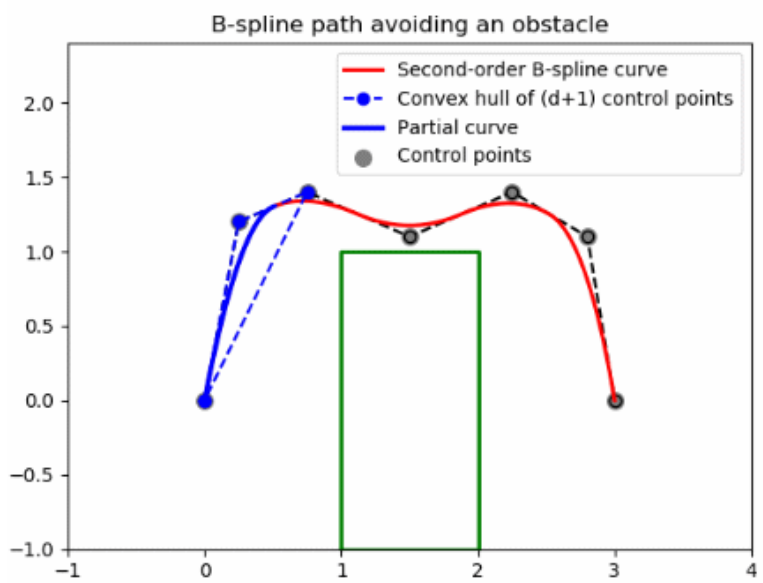
Exercises on planning B-spline path in polytope corridor

E4: B-spline path planner using B-spline representation

Study the code given in Appendix 2, complete it for the given scenario.

Choose several different optimization goals and implement them.
For example, minimal-length curve [7].

Related file: bspline_path_planner_minimal_length.py



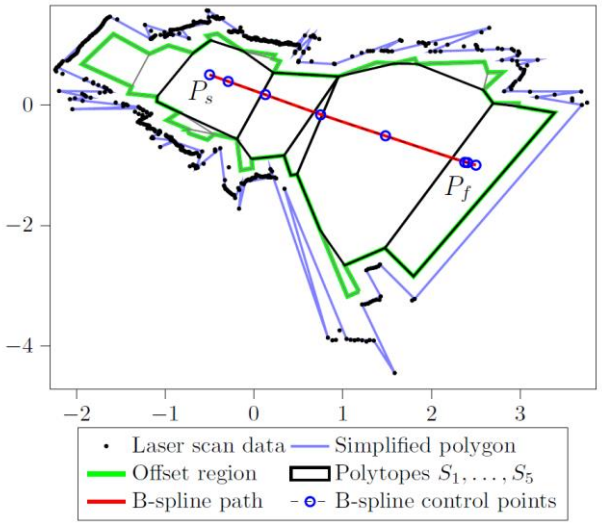
Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding the green rectangle obstacle.

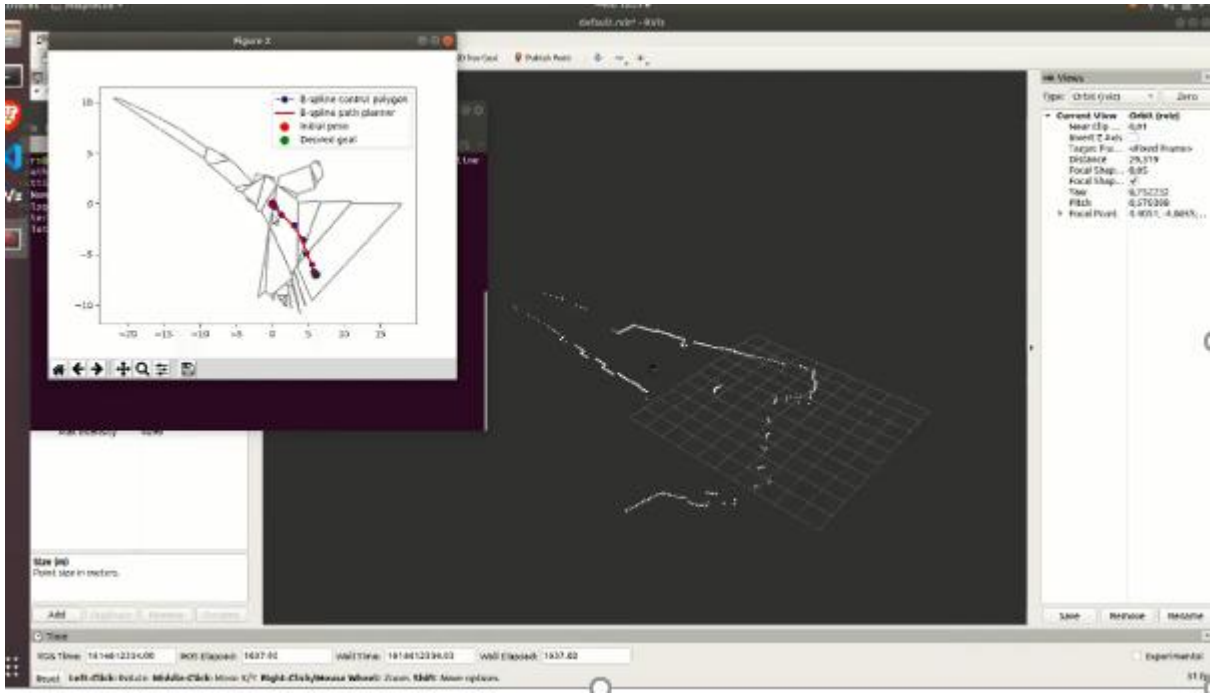
[7] Florin Stoican et al. "Flat Trajectory generation for way-points relaxations and obstacle avoidance". In Proc. of the 23rd IEEE Conference Mediterranean Conference on Control and Automation (MED), 695-700, 2015

Exercises on planning B-spline path in polytope corridor

E5: B-spline path planner from laser scan data

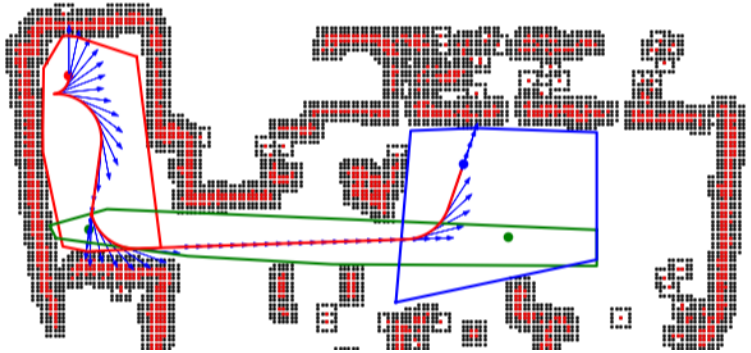
Combine exercises E1-E4 into one complete project in order to plan a B-spline path within a free space determined by a single laser scan data.





4. MPC design with tightening polytopic constraints

a) Introduction to tightening polytopic constraints
b) Problem formulation
c) MPC designs for DWR navigation
d) Simulation results and comparisons
e) Exercises on MPC design with tightening polytopic constraints



[13] Ngoc Thinh Nguyen, and Georg Schildbach. "Tightening polytopic constraint in MPC designs for mobile robot navigation", in Proc. of 25th International Conference on System Theory, Control and Computing (ICSTCC), 407-412, 2021. **Best Paper Award**.

Introduction to tightening polytopic constraints

- How to not only avoid obstacles but also keep a certain distance from them?
- Existing solution:
  Repulsive potential field [10] can provide smooth paths that keep a distance to obstacle whenever possible, without unnecessarily restricting the free-space.
- Drawbacks:
  Usage of strongly nonlinear functions.
  Local minima.
  → cause difficulties (solvability, local optimum, computing time) for solving planning and control problems.

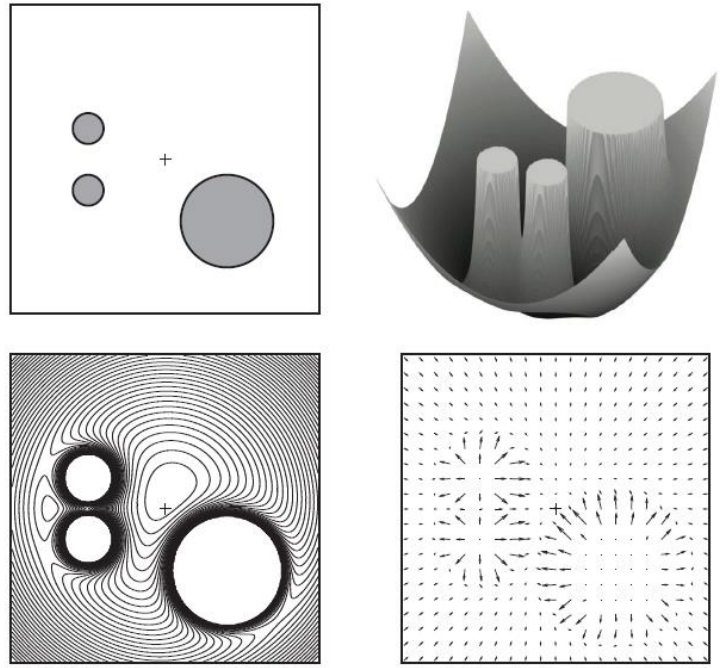


Illustration of potential field functions built for 3 circular obstacles [1].

[10] Ngo-Quoc-Huy Tran et al. Potential-field construction in an MPC framework: application for safe navigation in a variable coastal environment, in IFAC-PapersOnline, 51(20): 307-312, 2018.
[1] Kevin M. Lynch, and Frank C. Park. Modern Robotics: Mechanics, Planning and Control. Cambridge University Press, May 2017.

Introduction to tightening polytopic constraints

Novelties:

- A new linear constraint formulation, so-called, tightening polytopic constraint, which allows to simultaneously validate the polytopic constraint and perform an inward offset to the polytopic region.
- Two MPC designs for mobile robot navigation to the desired goal using the tightening polytopic constraint:
  1. The first one aims to maximize the offset value;
  2. The second one is to maintain a desired offset value.

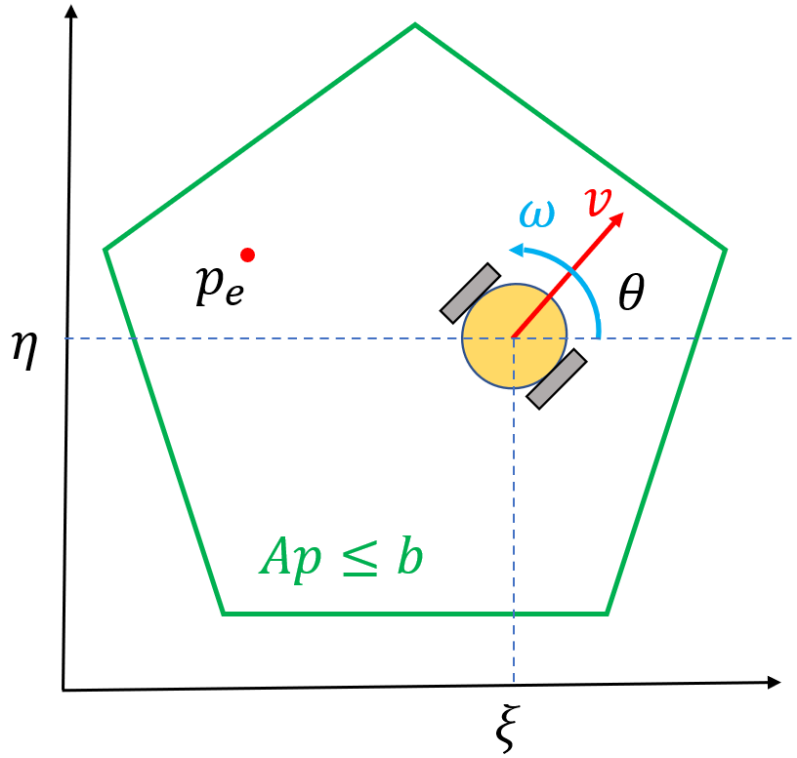
Problem formulation

- DWR (Differential Wheeled Robot) kinematics model:

$$\begin{bmatrix} \dot{\xi} \\ \dot{\eta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \Leftrightarrow \dot{x} = f(x,u)$$

- The state vector $x \triangleq (\xi,\ \eta,\ \theta)^{\mathsf{T}}$ consisting of the 2D position $(\xi,\ \eta)$ and the direction angle θ.
- The input vector is $u \triangleq (v,\ \omega)^{\mathsf{T}}$ with v the linear, and ω the angular speeds of the robot, respectively.
- The output to be controlled is the 2D position $p = (\xi,\ \eta)^{\mathsf{T}}$:

$$p = Mx, \qquad M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$



DWR and its workspace

Problem formulation

- DWR (Differential Wheeled Robot) kinematics model:

$$\begin{bmatrix} \dot{\xi} \\ \dot{\eta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \Leftrightarrow \dot{x} = f(x,u)$$

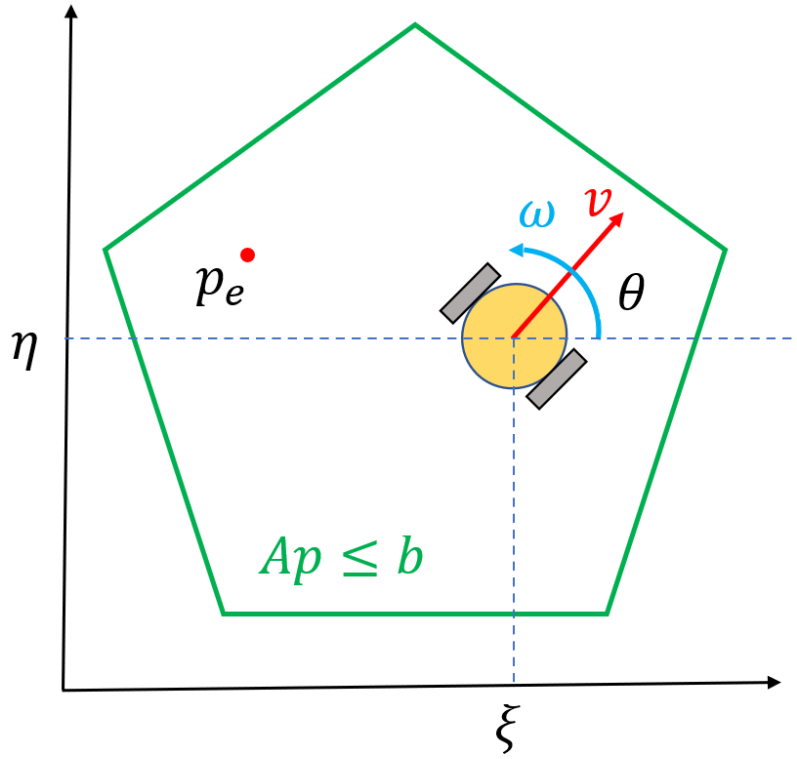
- The input u is bounded by:

$$u_{\min} \le u \le u_{\max}, \qquad \text{with } u_{\min} \le 0 \le u_{\max}$$

- The robot is allowed to move inside a free zone, represented by a 2D polytope:

$$p \in \mathcal{P}(A,b) \Leftrightarrow Ap \le b,$$

with $A \in \mathbb{R}^{m\times 2}, b \in \mathbb{R}^m$.



DWR and its workspace

Problem formulation

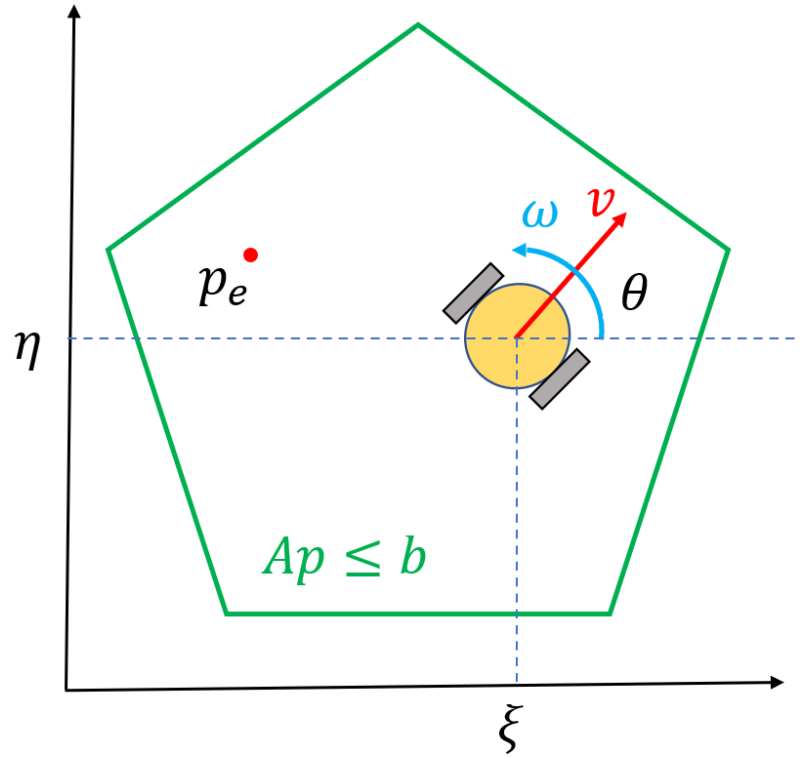
- DWR (Differential Wheeled Robot) kinematics model:

$$\begin{bmatrix} \dot{\xi} \\ \dot{\eta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \Leftrightarrow \dot{x} = f(x,u),$$

$$u_{\min} \leq u \leq u_{\max}$$

$$p \in \mathcal{P}(A,b)$$

- Problem: the robot navigates to the desired position p_e while satisfying the state and input constraints. Also, the robot should try to stay away from the edges of the polytope $\mathcal{P}(A,b)$.



DWR and its workspace

4. MPC design with tightening polytopic constraints

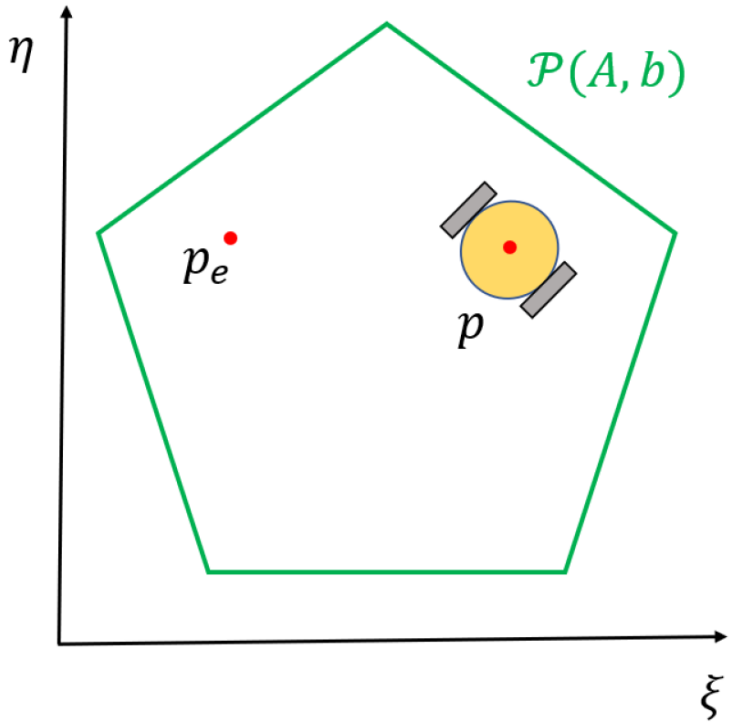
a) Introduction to tightening polytopic constraints
b) Problem formulation
c) MPC designs for DWR navigation
   1) Tightening polytopic constraint
   2) Normalization of the half-space representation
   3) MPC design with tightening polytopic constraint
d) Simulation results and comparisons
e) Exercises on tightening polytopic constraints

[13] Ngoc Thinh Nguyen, and Georg Schildbach. "Tightening polytopic constraint in MPC designs for mobile robot navigation", in Proc. of 25th International Conference on System Theory, Control and Computing (ICSTCC), 407-412, 2021. **Best Paper Award**.

Tightening polytopic constraint

- Problem of constraining a point p inside a polytope $\mathcal{P}(A, b)$.
- Original polytopic constraint:

$$p \in \mathcal{P}(A, b) \Leftrightarrow Ap \leq b$$



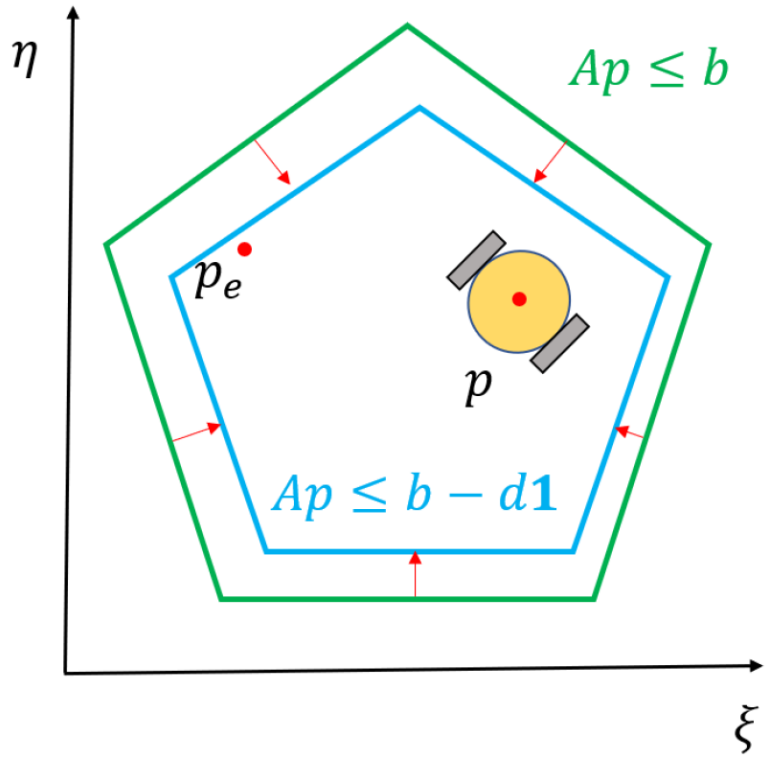
Original polytopic constraint

Tightening polytopic constraint

- Problem of constraining a point p inside a polytope $\mathcal{P}(A, b)$.
- Original polytopic constraint:

$$p \in \mathcal{P}(A, b) \Leftrightarrow Ap \leq b$$

- In order to encourage p to stay away from the edges of $\mathcal{P}(A, b)$, we rate the distance between p and the closest edge:
  1. Represent the polytope with an inward offset d, i.e., $\mathcal{P}(A, b - d\mathbf{1})$.
  2. Check if there exists any value of $d \geq 0$ such that $p \in \mathcal{P}(A, b - d\mathbf{1})$.



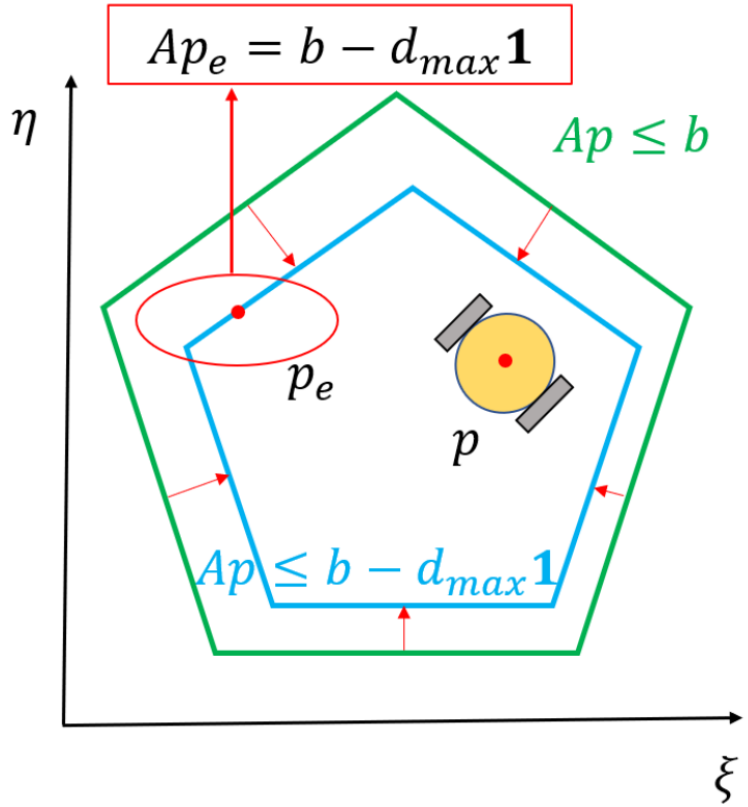
Tightening polytopic constraint

Tightening polytopic constraint

- Problem of constraining a point p inside a polytope $\mathcal{P}(A, b)$.
- Original polytopic constraint:

$$p \in \mathcal{P}(A, b) \Leftrightarrow Ap \leq b$$

- In order to encourage p to stay away from the edges of $\mathcal{P}(A, b)$, we rate the distance between p and the closest edge:
  1. Represent the polytope with an inward offset d, i.e., $\mathcal{P}(A, b - d\mathbf{1})$.
  2. Check if there exists any value of $d \geq 0$ such that $p \in \mathcal{P}(A, b - d\mathbf{1})$.
  3. Encourage a big value of the offset d but limit its maximum value $d_{\max}$ such that the robot can still reach the desired goal.



Tightening polytopic constraint with maximum offset value

Tightening polytopic constraint

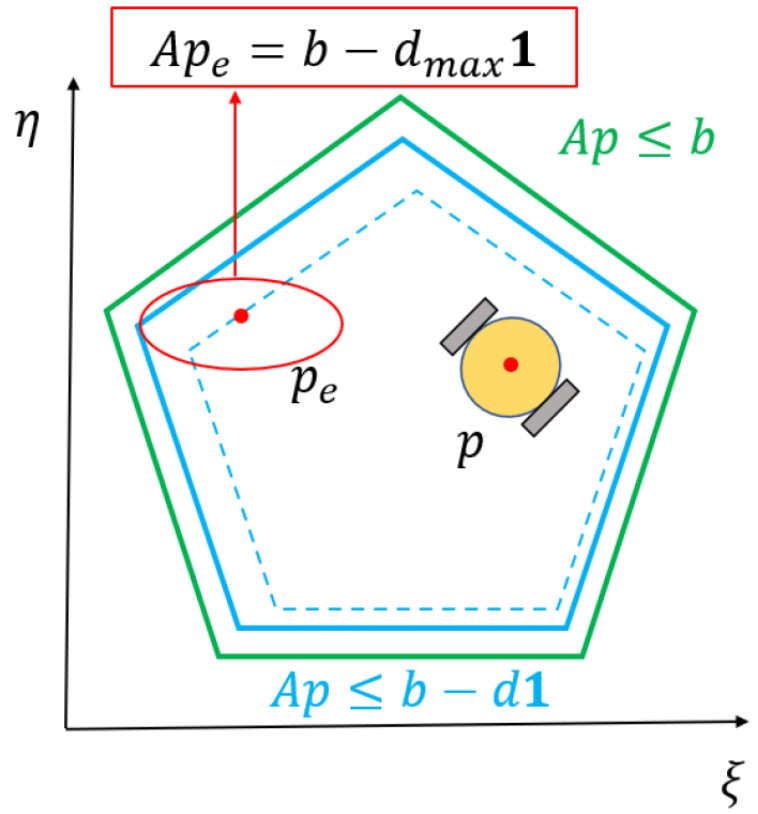
- Problem of constraining a point p inside a polytope $\mathcal{P}(A, b)$.
- Original polytopic constraint:

$$p \in \mathcal{P}(A, b) \Leftrightarrow Ap \leq b$$

- In order to encourage p to stay away from the edges of $\mathcal{P}(A, b)$, we rate the distance between p and the closest edge:
  1. Represent the polytope with an inward offset d, i.e., $\mathcal{P}(A, b - d\mathbf{1})$.
  2. Check if there exists any value of $d \geq 0$ such that $p \in \mathcal{P}(A, b - d\mathbf{1})$.
  3. Encourage a big value of the offset d but limit its maximum value $d_{\max}$ such that the robot can still reach the desired goal.
- Tightening polytopic constraint:

$$\begin{cases} Ap \leq b - d\mathbf{1}, \\ 0 \leq d \leq d_{\max}, \end{cases}$$

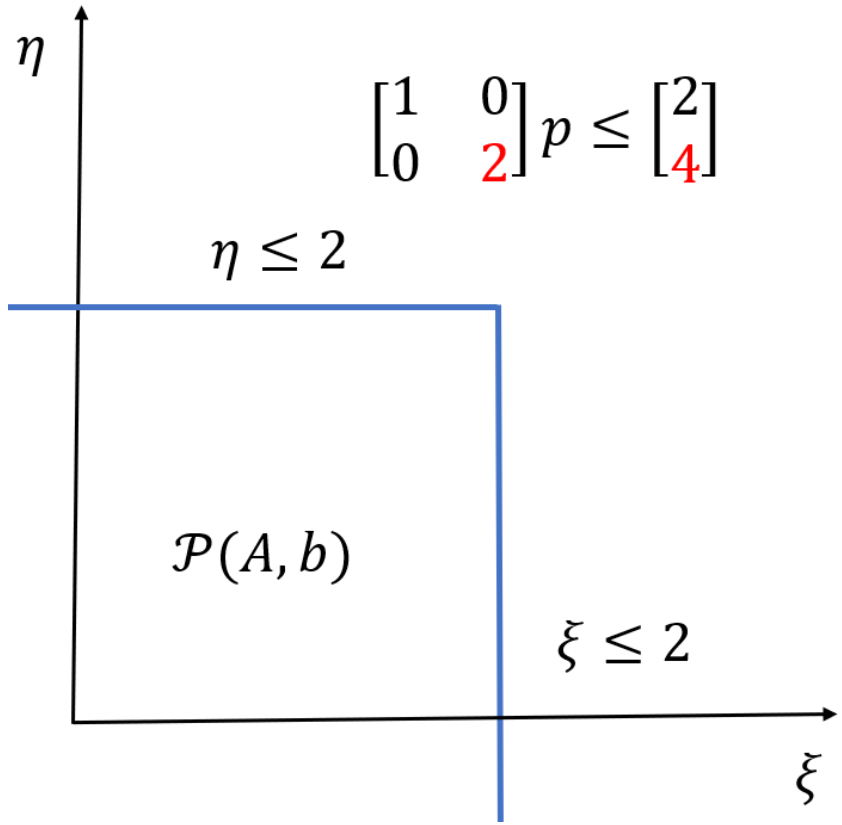
with d, the inward offset value, defined as a scalar decision variable.



Tightening polytopic constraint with constrained offset value

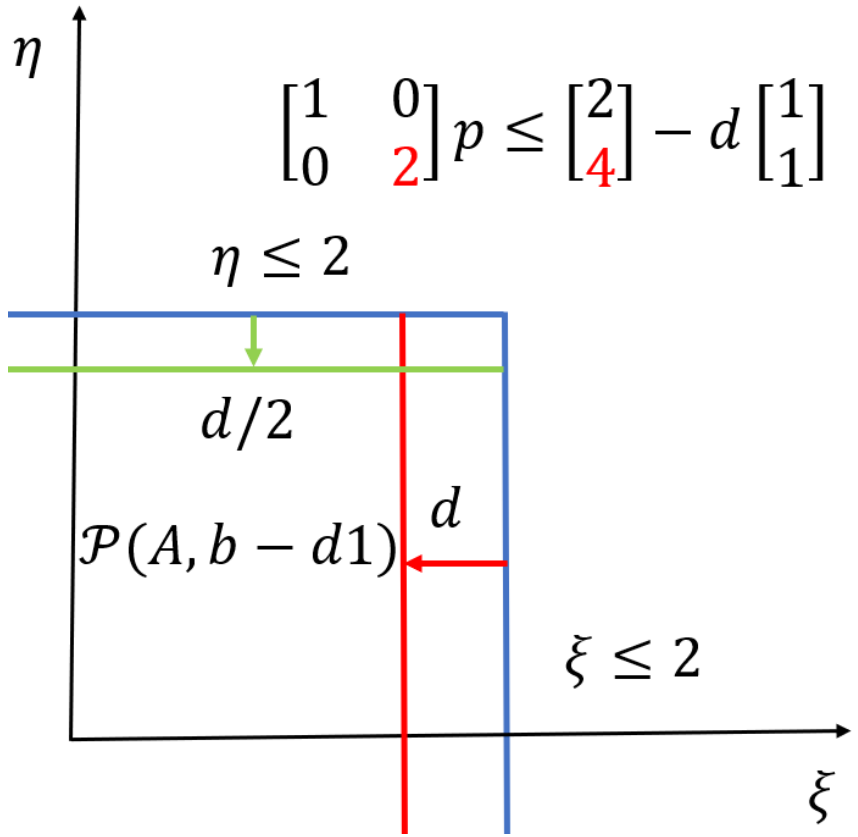
Normalization of the half-space representation

- Tightening polytopic constraint: $Ap \leq b - d\mathbf{1}$



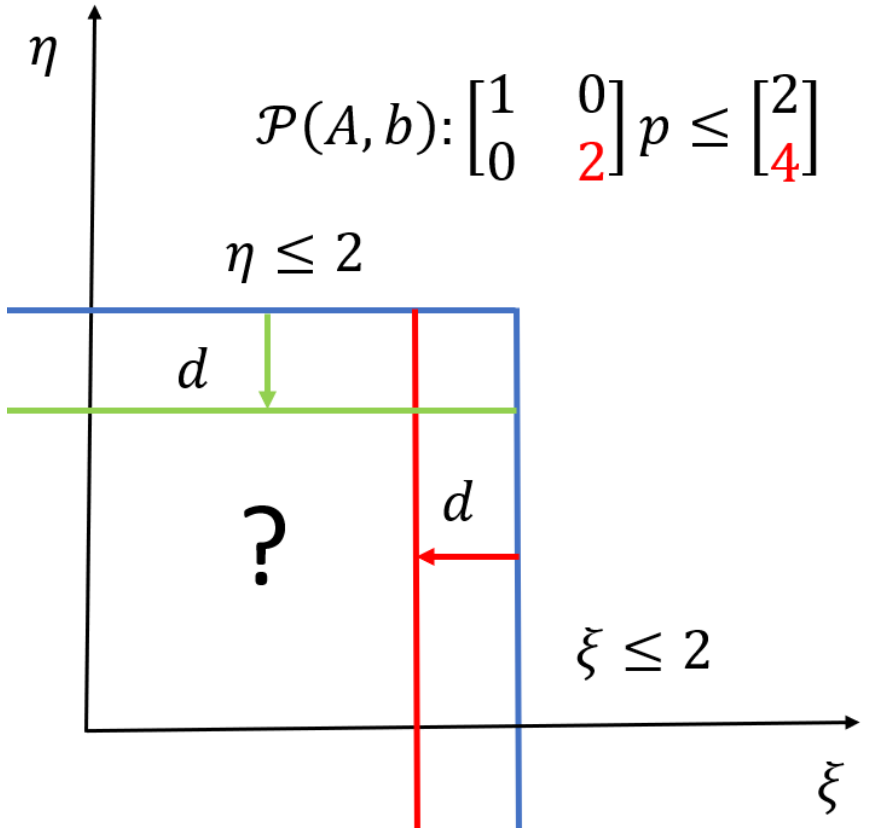
Normalization of the half-space representation

- Tightening polytopic constraint: $Ap \leq b - d\mathbf{1}$
- Problem: the distances between the pairs of corresponding edges of $\mathcal{P}(A, b)$ and $\mathcal{P}(A, b - d\mathbf{1})$ may not equal to the chosen value d.



Normalization of the half-space representation

- Tightening polytopic constraint: $Ap \leq b - d\mathbf{1}$
- Problem: the distances between the pairs of corresponding edges of $\mathcal{P}(A, b)$ and $\mathcal{P}(A, b - d\mathbf{1})$ may not equal to the chosen value d.
- Normalization of half-space representation: to ensure that the same offset distance d will be applied to all the edges of the polytope.



Normalization of the half-space representation

- Tightening polytopic constraint: $Ap \leq b - d\mathbf{1}$
- Problem: the distances between the pairs of corresponding edges of $\mathcal{P}(A, b)$ and $\mathcal{P}(A, b - d\mathbf{1})$ may not equal to the chosen value d.
- Normalization of half-space representation: to ensure that the same offset distance d will be applied to all the edges of the polytope. The polytope $\mathcal{P}(A, b)$ has its equivalent but normalized representation $\mathcal{P}(\bar{A}, \bar{b})$ with:

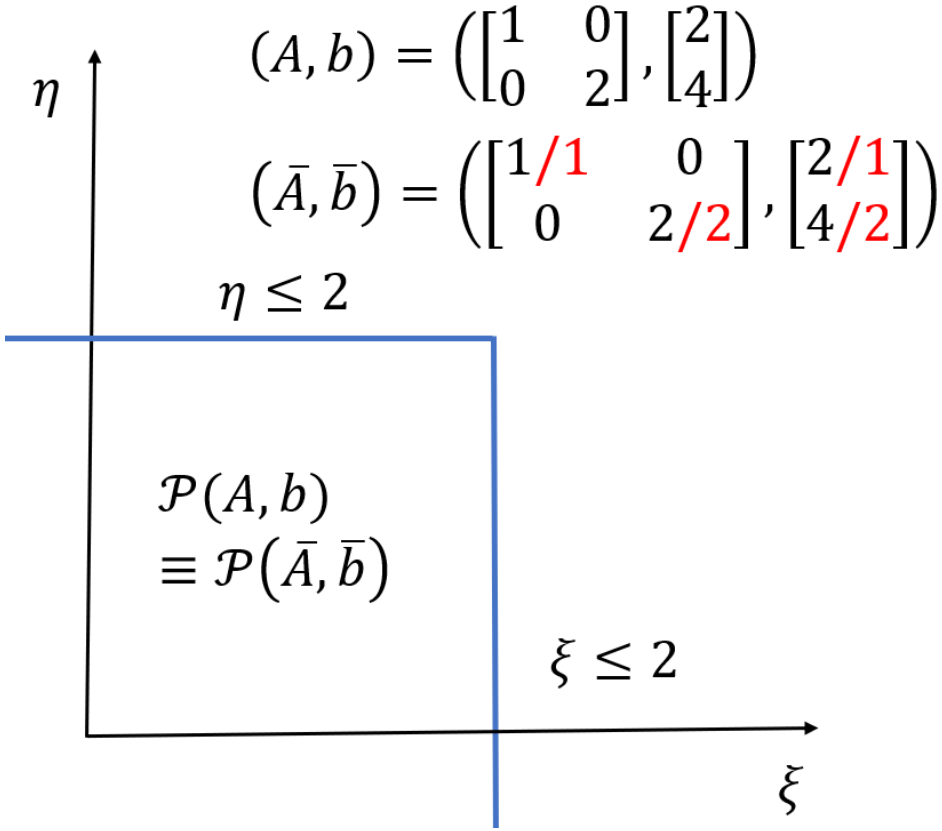
$$(\bar{A}, \bar{b}) = \overline{(A, b)},$$

or in explicit form:

$$\bar{A} = \begin{bmatrix} a_{11}/\|A_1\| & a_{12}/\|A_1\| & \dots & a_{1n}/\|A_1\| \\ \dots & \dots & \dots & \dots \\ a_{m1}/\|A_m\| & a_{m2}/\|A_m\| & \dots & a_{mn}/\|A_m\| \end{bmatrix},$$

$$\bar{b} = \left[b_1/\|A_1\|, \ \dots, \ b_m/\|A_m\|\right]^\top,$$

with $\|A_i\| = \sqrt{a_{i1}^2 + a_{i2}^2 + \cdots + a_{in}^2}$, the 2-norm of the i^{th} row of A.

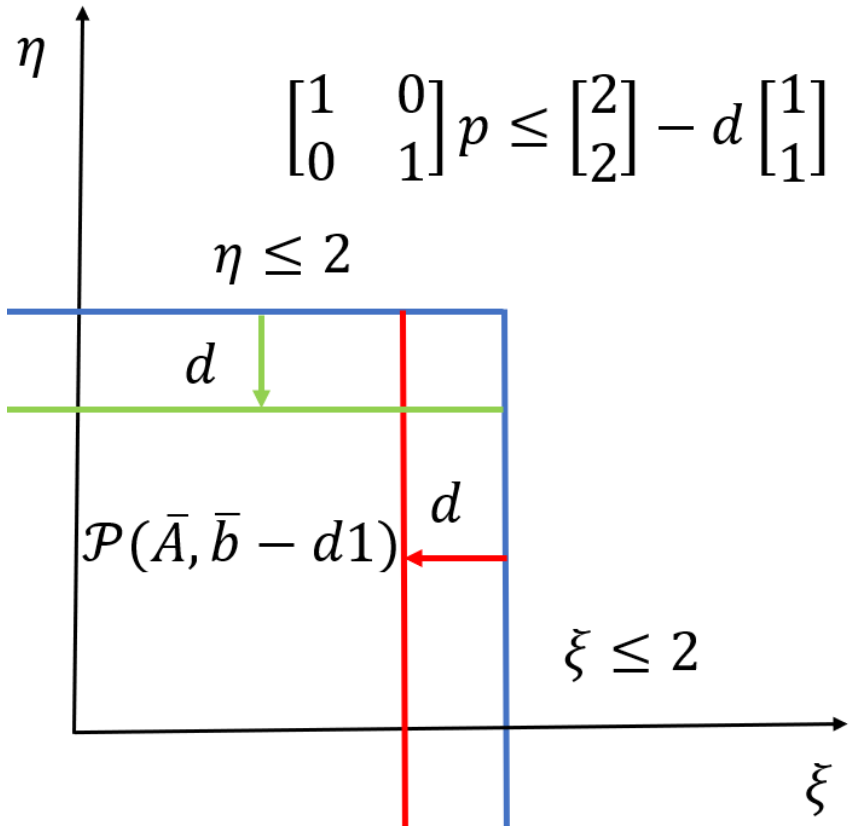


Normalization of the half-space representation

- Tightening polytopic constraint: $Ap \leq b - d\mathbf{1}$
- Problem: the distances between the pairs of corresponding edges of $\mathcal{P}(A, b)$ and $\mathcal{P}(A, b - d\mathbf{1})$ may not equal to the chosen value d.
- Normalization of half-space representation: to ensure that the same offset distance d will be applied to all the edges of the polytope. The polytope $\mathcal{P}(A, b)$ has its equivalent but normalized representation $\mathcal{P}(\bar{A}, \bar{b})$ with:

$$(\bar{A}, \bar{b}) = \overline{(A, b)}$$

- Proposition: the offset polytope $\mathcal{P}(\bar{A}, \bar{b} - d\mathbf{1})$ has all of its edges being inwardly translated for a distance d with respect to the corresponding edges of the original polytope $\mathcal{P}(A, b)$.



MPC design for maximizing offset value

MPC framework for a DWR system to navigate to the desired goal p_e using the tightening polytopic constraint:

$$\{u_0^*, \dots\} = \arg\min \left(\sum_{i=0}^{N_p} \left(\|\hat{p}_i - p_e\|_Q^2 + \|\hat{u}_i\|_R^2 \right) + \sum_{i=0}^{N_s} K(-d_i) \right),$$

subject to:

$$\begin{aligned}
&\hat{x}_{i+1} = f_d(\hat{x}_i, \hat{u}_i), \forall i \in [0, N_p - 1], && \text{system model,}\\
&u_{\min} \le \hat{u}_i \le u_{\max}, \forall i \in [0, N_p - 1], && \text{input constraint,}\\
&\hat{p}_i = M\hat{x}_i, \forall i \in [0, N_p], && \text{position calculation,}\\
&\bar{A}\hat{p}_i \le \bar{b} - d_i \mathbf{1}, \forall i \in [0, N_p], && \text{tightening polytopic constraint,}\\
&0 \le d_i \le \bar{d}_{\max}, \forall i \in [0, N_p], && \text{offset constraint,}\\
&\hat{x}_0 = x(k), && \text{initial state,}
\end{aligned}$$

In which, $(\bar{A}, \bar{b}) = \overline{(A, b)}$ and $\bar{d}_{\max}$ is the maximum offset value chosen w.r.t. $(\bar{A}, \bar{b})$. N_p is the prediction horizon while N_s is the shrinking horizon, $N_s \le N_p$. Appropriate values of N_s are 3 or 4.

Maximizing offset value: by adding a linear term of $K(-d_i)$ into the cost function, with $K > 0$ a positive coefficient.

MPC design for maintaining a desired offset value

MPC framework for a DWR system to navigate to the desired goal p_e using the tightening polytopic constraint:

$$\{u_0^*, \dots\} = \arg\min \left(\sum_{i=0}^{N_p} \left(\|\hat{p}_i - p_e\|_Q^2 + \|\hat{u}_i\|_R^2 \right) + \sum_{i=0}^{N_s} K(d_i - d_r)^2 \right),$$

subject to:

$$\begin{aligned}
&\hat{x}_{i+1} = f_d(\hat{x}_i, \hat{u}_i), \forall i \in [0, N_p - 1], && \text{system model,}\\
&u_{\min} \le \hat{u}_i \le u_{\max}, \forall i \in [0, N_p - 1], && \text{input constraint,}\\
&\hat{p}_i = M\hat{x}_i, \forall i \in [0, N_p], && \text{position calculation,}\\
&\bar{A}\hat{p}_i \le \bar{b} - d_i \mathbf{1}, \forall i \in [0, N_p], && \text{tightening polytopic constraint,}\\
&0 \le d_i \le \bar{d}_{\max}, \forall i \in [0, N_p], && \text{offset constraint,}\\
&\hat{x}_0 = x(k), && \text{initial state,}
\end{aligned}$$

In which, $(\bar{A}, \bar{b}) = \overline{(A, b)}$ and $\bar{d}_{\max}$ is the maximum offset value chosen w.r.t. $(\bar{A}, \bar{b})$. N_p is the prediction horizon while N_s is the shrinking horizon, $N_s \le N_p$. Appropriate values of N_s are 3 or 4.

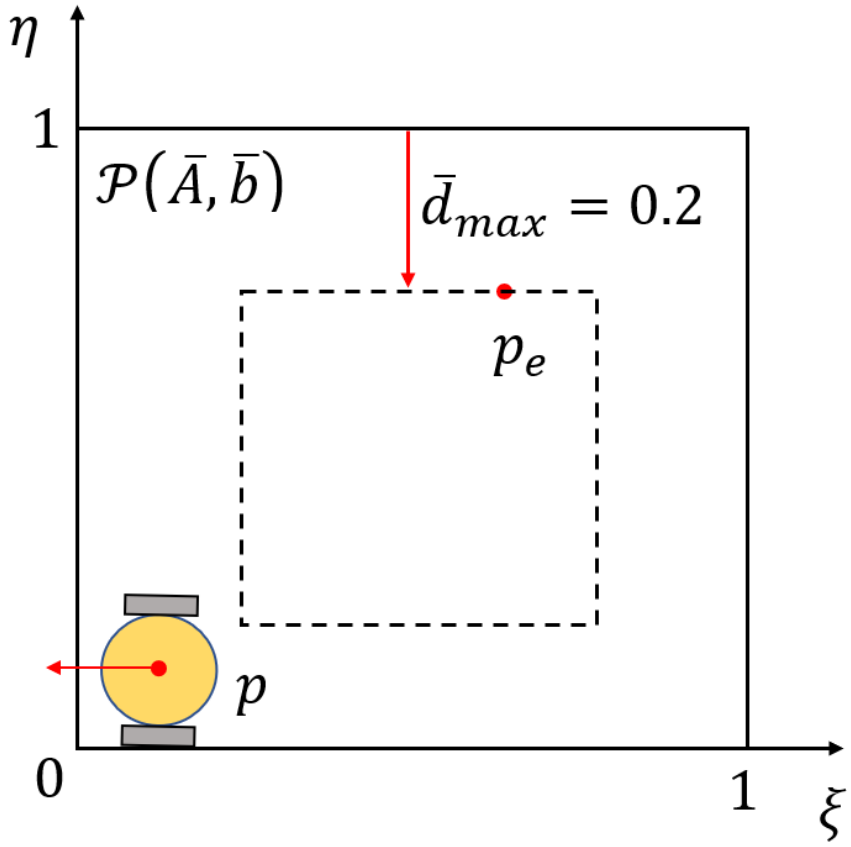
Maintaining a desired offset value d_r: by adding a quadratic term of $K(d_i - d_r)^2$ into the cost function, with $K > 0$ a positive coefficient.

Simulation setup and scenarios

The simulation model of a DWR system is characterized by the following numerical values:

- Linear velocity: $v \in [0, 0.26]\ \mathrm{m/s}$
- Angular velocity: $\omega \in [-0.5, 0.5]\ \mathrm{rad}/s$
- Working zone: $0 \leq \xi \leq 1,\ 0 \leq \eta \leq 1\ (\mathrm{meter})$

$$\bar{A} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \bar{B} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



- Initial state $x_0 = (0.1, 0, 1, \pi)^\top$
- Desired goal $p_e = (0.6, 0.8)^\top$ → maximum offset $\bar{d}_{\max} = 0.2$

Simulation setup and scenarios

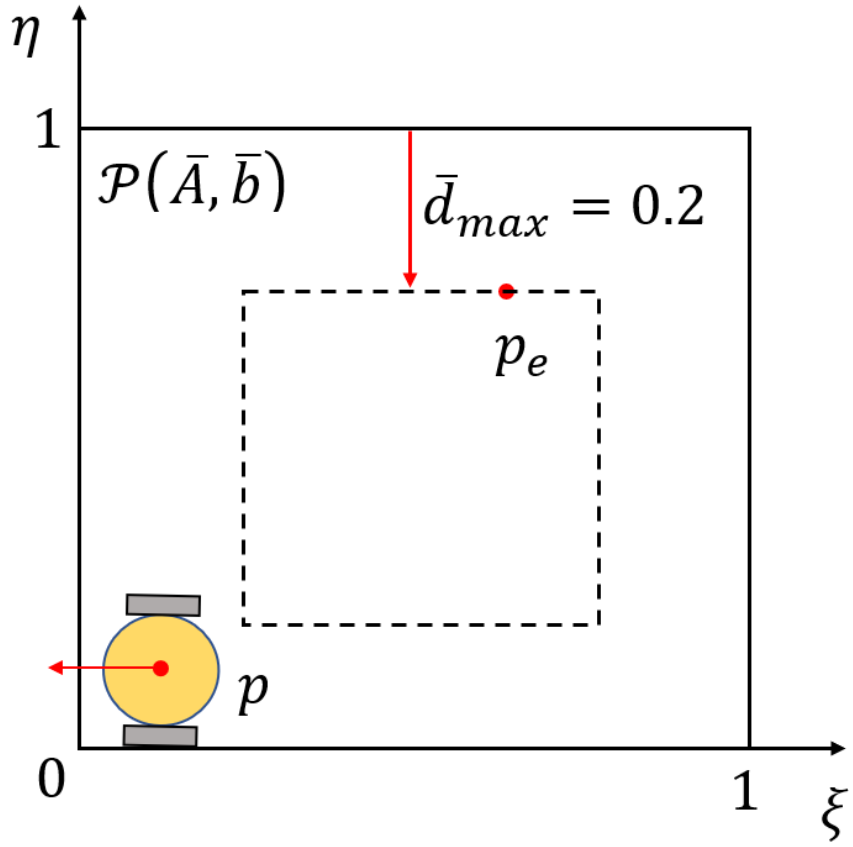
Four different controllers are simulated:

- Sce. 1: Standard MPC controller
- Sce. 2: MPC controller with TPC, maximizing offset.
- Sce. 3: MPC controller with TPC, maintaining desired offset.
- Sce. 4: MPC with repulsive potential field [10]

$$K \sum_{i=0}^{N_p} \left(\sum_{j=1}^{m} \frac{d_{\text{effect}}^2}{\left(\bar{A}_j \hat{p}_j - \bar{b}_j\right)^2 + d_{\text{effect}}^2} \right),$$

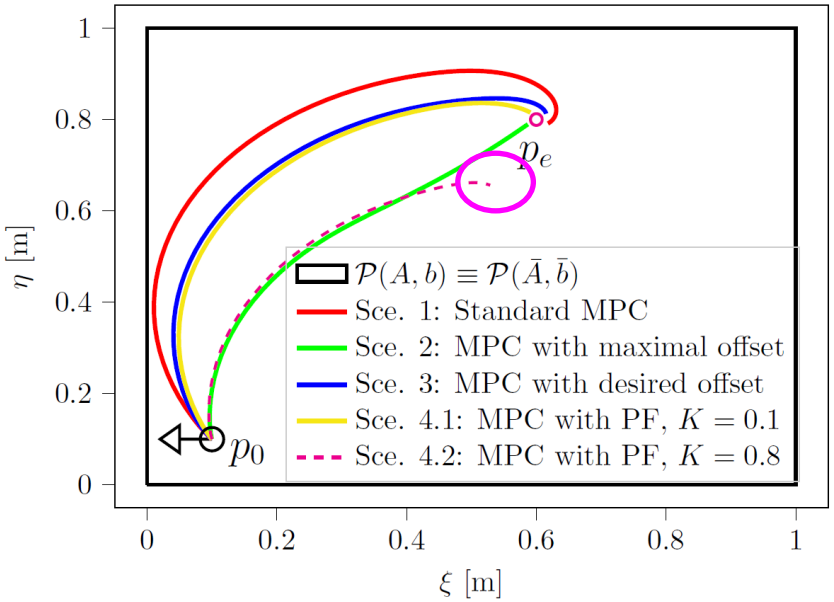
in which, $d_{\text{effect}} = 0.2$ stands for the effective range and $K > 0$ is the coefficient:

  - Sce. 4.1: weak PF, $K = 0.1$
  - Sce. 4.2: strong PF, $K = 0.8$



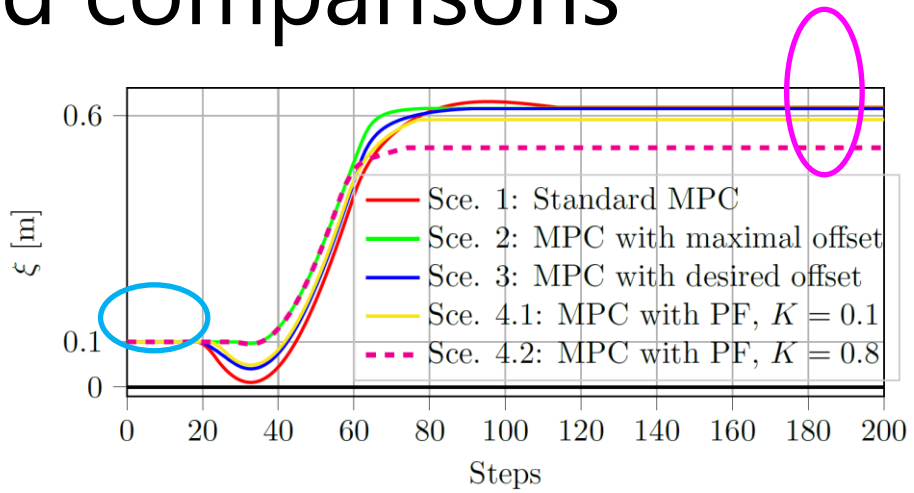
[10] Ngo-Quoc-Huy Tran et al. Potential-field construction in an MPC framework: application for safe navigation in a variable coastal environment, in IFAC-PapersOnline, 51(20): 307-312, 2018.

Simulation results and comparisons

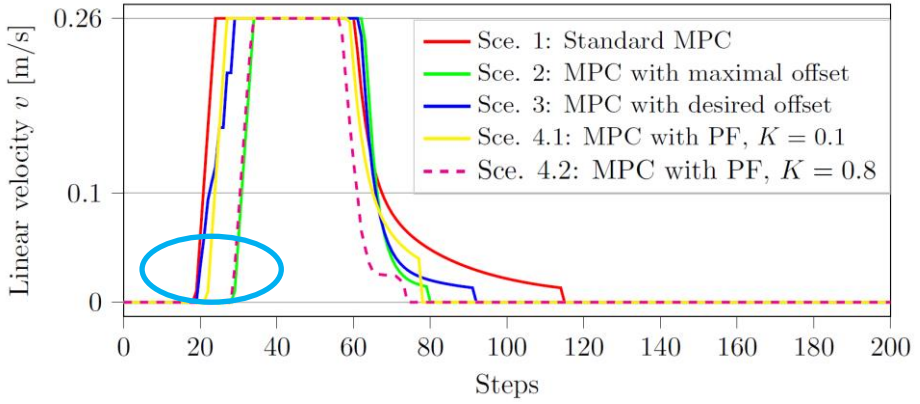


Simulation trajectories

- Standard MPC (red line) drives the DWR closest to the boundary.
- MPC with strong PF (magenta line) gets stuck in its local minima.
- Similar effects: Sce. 2 vs Sce. 4.2, Sce. 3 vs Sce. 4.1.



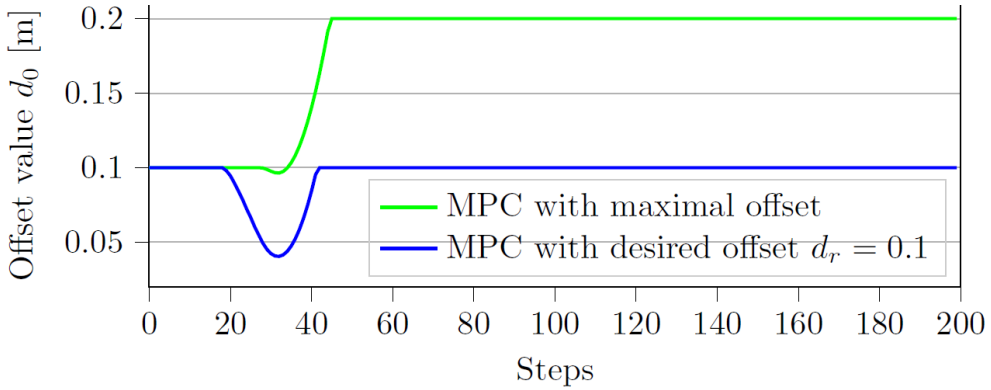
Convergences of ξ



Linear velocity input v

- Similar responses of all the controllers at the beginning phase, i.e., rotating on spot in counter-clockwise within twenty first steps.
- The robot starts to move at different moments under different scenarios.
- The MPC with maximal offset and the MPC with strong PF keep rotating on spot and only move forward after step 27 when they already obtain clearer heading angles.

Simulation results and comparisons



Offset distances of the first prediction step.

Name	Mean	Std	Min	Max
Standard MPC	66.76	8.20	57.88	86.71
MPC with maximal offset	68.49	4.60	61.83	80.73
MPC with desired offset	72.33	6.18	61.52	90.63
MPC with PF ($K = 0.1$)	82.85	4.43	74.82	99.78
MPC with PF ($K = 0.8$)	87.90	3.58	78.77	103.86

Computing time of five MPC controllers
Unit: millisecond. Std: standard deviation.

- The offset distances resulted from the MPC with maximal offset reaches the maximal offset value $d_{\max}$ while those resulted from the MPC with desired offset stay at $d_r = 0.1$ after a reduction happening from step 20 to step 40.
- The computing times increase from Sce. 1 to Sce. 5 which implies order of computation burden:
  Standard MPC < MPC with maximal offset (linear additional term) < MPC with desired offset (quadratic additional term) < MPC with medium PF < MPC with strong PF.

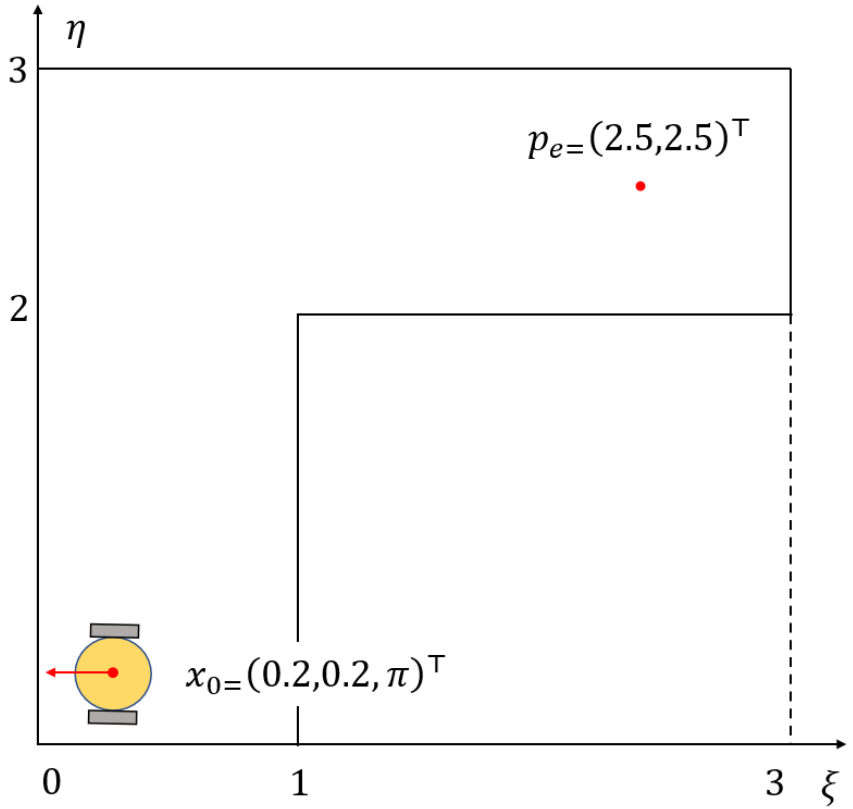
Exercises on tightening polytopic constraints

Consider the discrete model of a DWR given as:

$$\xi_{k+1} = \xi_k + \Delta_t \, v \cos\theta_k$$
$$\eta_{k+1} = \eta_k + \Delta_t \, v \sin\theta_k$$
$$\theta_{k+1} = \theta_k + \Delta_t \, \omega_k$$

with $v \in [0, 0.26]$ m/s and $\omega \in [-0.5, 0.5]$ rad/s.

The task is to navigate inside the corridor to the goal.



DWR navigating inside a non-convex corridor

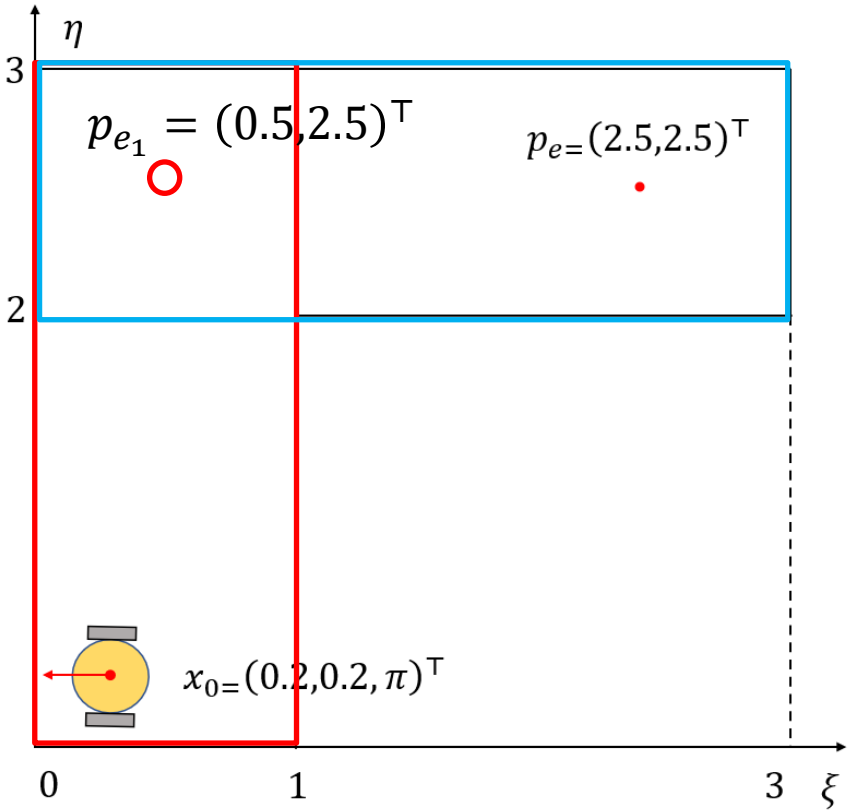
Exercises on tightening polytopic constraints

E6: MPC design for navigation within one single polytope

Implement two MPC controllers using tightening polytopic constraint maximizing the offset for the DWR:

1) $\mathcal{C}_1$ to navigate to a pseudo goal $p_{e_1} = (0.5,2.5)^\mathsf{T}$. The DWR is constrained to stay within the red rectangle.

2) $\mathcal{C}_2$ to navigate to the goal $p_e = (2.5,2.5)^\mathsf{T}$. The DWR is constrained to stay within the blue rectangle.



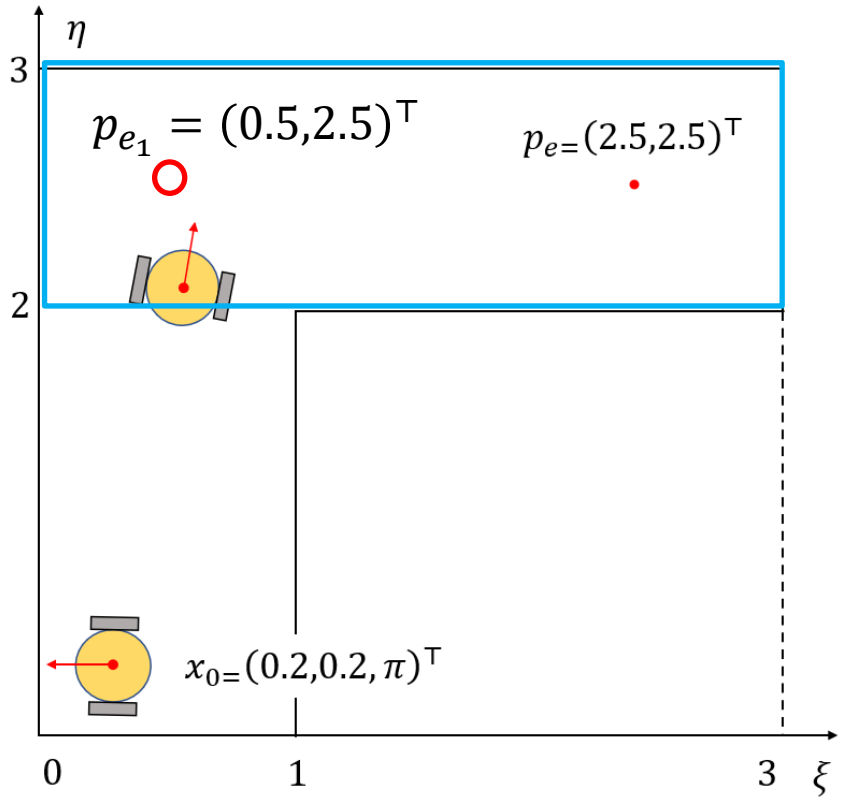
DWR navigating inside a non-convex corridor

Exercises on tightening polytopic constraints

E7: Switching MPC controllers for navigation within a given corridor:

1) Using $\mathcal{C}_1$ to navigate from $x_0 = (0.1,0,1,\pi)^\top$ to p_{e_1}.
2) Whenever the DWR enters the blue rectangle, switch to $\mathcal{C}_2$ to navigate to the goal $p_e = (2.5,2.5)^\top$.

Simulate the whole scenario and present the results.

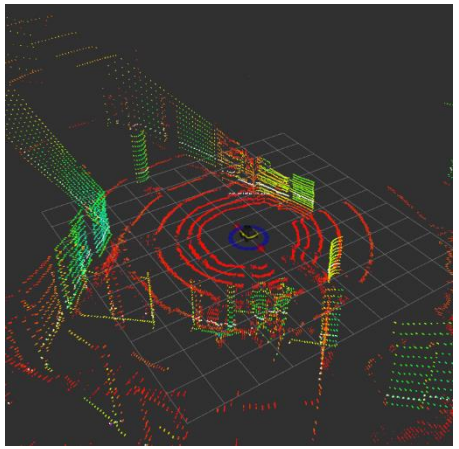


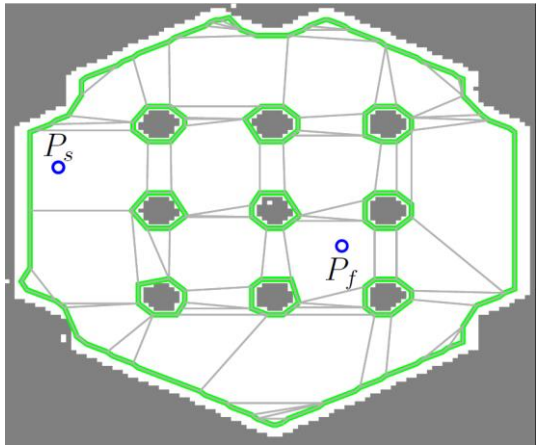
DWR navigating inside a non-convex corridor

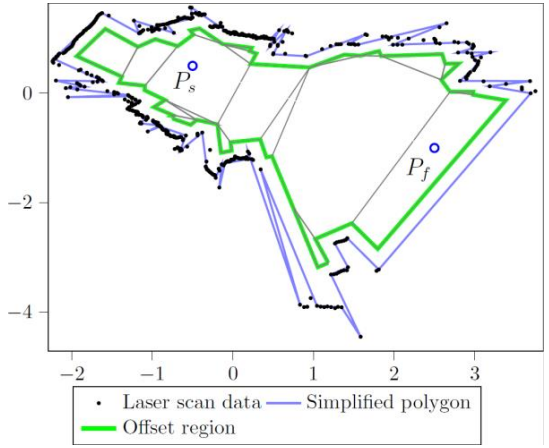
1. Introduction
2. From real world to polytope map
3. B-spline path planning in polytope map
4. MPC designs with tightening polytopic constraint
5. Conclusions

Conclusions

- Creation of polytope maps from different sources: laser scan data, depth image, occupancy grid map.

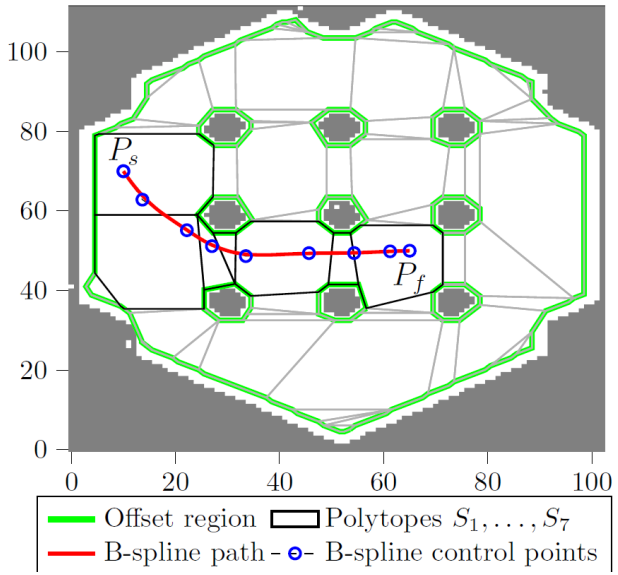


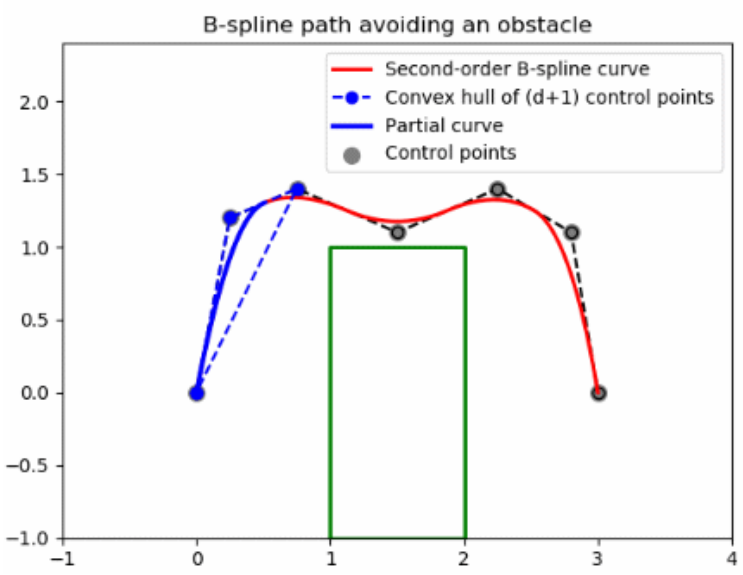


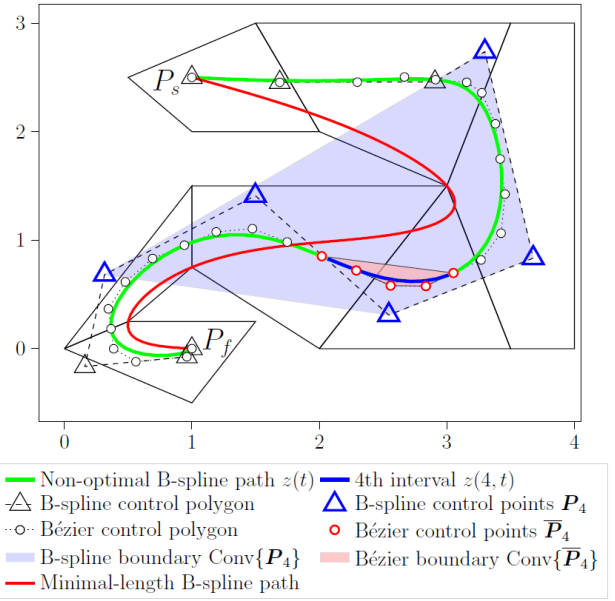


Conclusions

- Creation of polytope maps from different sources: laser scan data, depth image, occupancy grid map.
- Constraints for path planning in a polytopic corridor using B-spline curve as well as its equivalent Bézier representation.

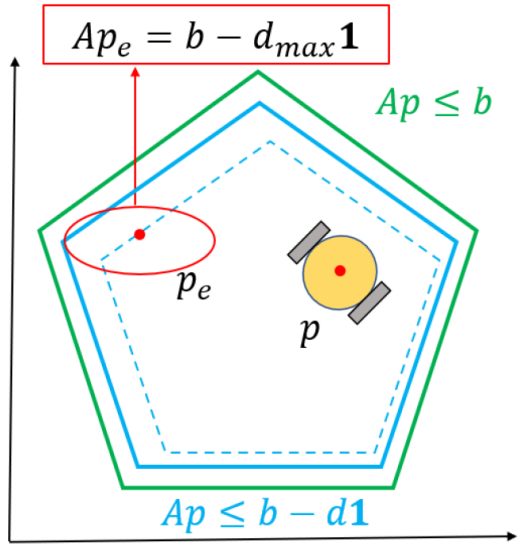


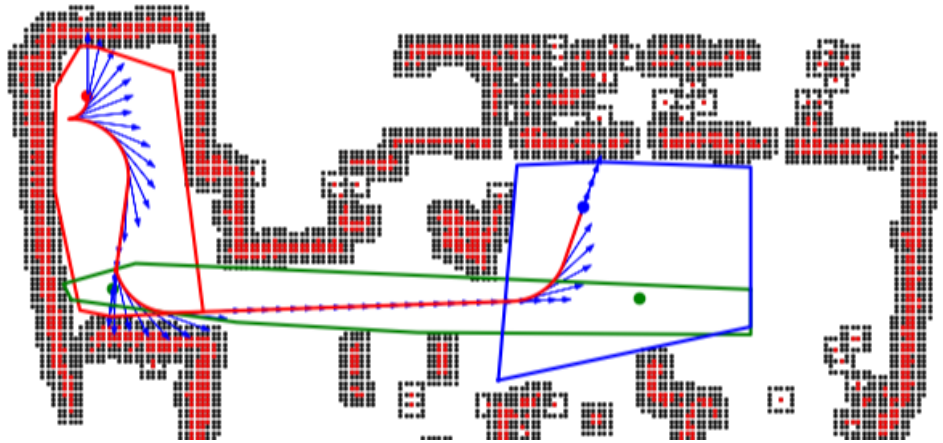




Conclusions

- Creation of polytope maps from different sources: laser scan data, depth image, occupancy grid map.
- Path planning in a polytopic corridor using B-spline curve as well as its equivalent Bézier representation.
- Tightening polytopic constraint and its use in MPC designs for navigation in 2D space.





References

[1] Kevin M. Lynch, and Frank C. Park. Modern Robotics: Mechanics, Planning and Control. Cambridge University Press, May 2017.

[2] Andreas Geiger et al, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", in Proc. of Conference on Computer Vision and Pattern Recognition, 2012.

[3] Johann Dichtl et al. "PolySLAM: A 2D Polygon-based SLAM algorithm", in Proc. of IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 1-6, 2019.

[4] Christian Meerpohl et al. "Free-space Polygon Creation based on Occupancy Grid Maps for Trajectory Optimization Methods", in IFAC-PapersOnline, 52(8): 368-374, 2019.

[5] Yuxiao Chen et al. "Lidar-based exploration and discretization for mobile robot planning", inarXiv:2011.10066v1 [cs.RO], 19 November 2020.

[6] Fajar Suryawan, et al, "Splines and polynomial tools for flatness-based constrained motion planning". International Journalof Systems Science, 43(8):1396–1411, 2012.

[7] Florin Stoican et al. "Flat Trajectory generation for way-points relaxations and obstacle avoidance". In Proc. of the 23rd IEEE Conference Mediterranean Conference on Control and Automation (MED), 695-700, 2015

[8] Ionela Prodan et al. "Necessary and sufficient LMI conditions for constraints satisfaction within a B-spline framework", in Proc. of 58th Conference on Decision and Control (CDC), 8061-8066, 2019.

[9] L. Romani and M. A. Sabin: The conversion matrix between uniform B-spline and Bézier representations, Computer Aided Geometric Design, 21(6), 549-560, 2014.

[10] Ngo-Quoc-Huy Tran et al. Potential-field construction in an MPC framework: application for safe navigation in a variable coastal environment, in IFAC-PapersOnline, 51(20): 307-312, 2018.

[11] Ngoc Thinh Nguyen et al, "Flat trajectory design and tracking with saturation guarantees: a nano-drone application". International Journal of Control, 93(6):1266-1279, 2020.

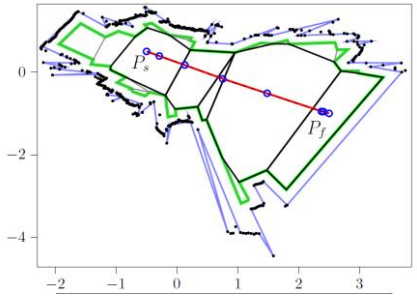
[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.

[13] Ngoc Thinh Nguyen, and Georg Schildbach. "Tightening polytopic constraint in MPC designs for mobile robot navigation", in Proc. of 25th International Conference on System Theory, Control and Computing (ICSTCC), 407-412, 2021. Best Paper Award.

[14] Jong Jin Park, Collin Johnson, and Benjamin Kuipers. Robot navigation with model predictive equilibrium point control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4945-4952, 2012.

[15] Marcell Missuar, et al. Polygonal Perception for Mobile Robots, in Proc. of 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 10476-10482, 2020.

Appendix 1: Exercises as a story



E1: Creating a polytope map from laser scan data

E2: Find a sequence of polytopes connecting two points.

E3: Geometric computation: finding transition zones and extended polytopes

E4: B-spline path planner using B-spline representation in a given corridor

E5: B-spline path planner from laser scan data (E1-E4)

E6: MPC design using tightening polytopic constraint for navigation within one single polytope

E7: Switching MPC controllers for navigation within a given corridor

[12] Ngoc Thinh Nguyen et al. "B-spline path planner for safe navigation of mobile robots", in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 316-322, 2021.
[13] Ngoc Thinh Nguyen, and Georg Schildbach. "Tightening polytopic constraint in MPC designs for mobile robot navigation", in Proc. of 25th International Conference on System Theory, Control and Computing (ICSTCC), 407-412, 2021. **Best Paper Award**.

Appendix 2: Python implementation

- Recommended setup:
  1. Anaconda environment: https://www.anaconda.com/
  2. Ipopt solver: **conda install -c conda-forge ipopt=3.11.1**
  3. Pyomo optimization interface: **conda install** -c **conda**-forge **pyomo**
  4. Install some essential Python packages:
     pip install numpy
     pip install scipy
     pip install matplotlib
     pip install rdp
     pip install gdspy
  5. Use Spyder (already available with Anaconda)

B-spline path planner: implementation

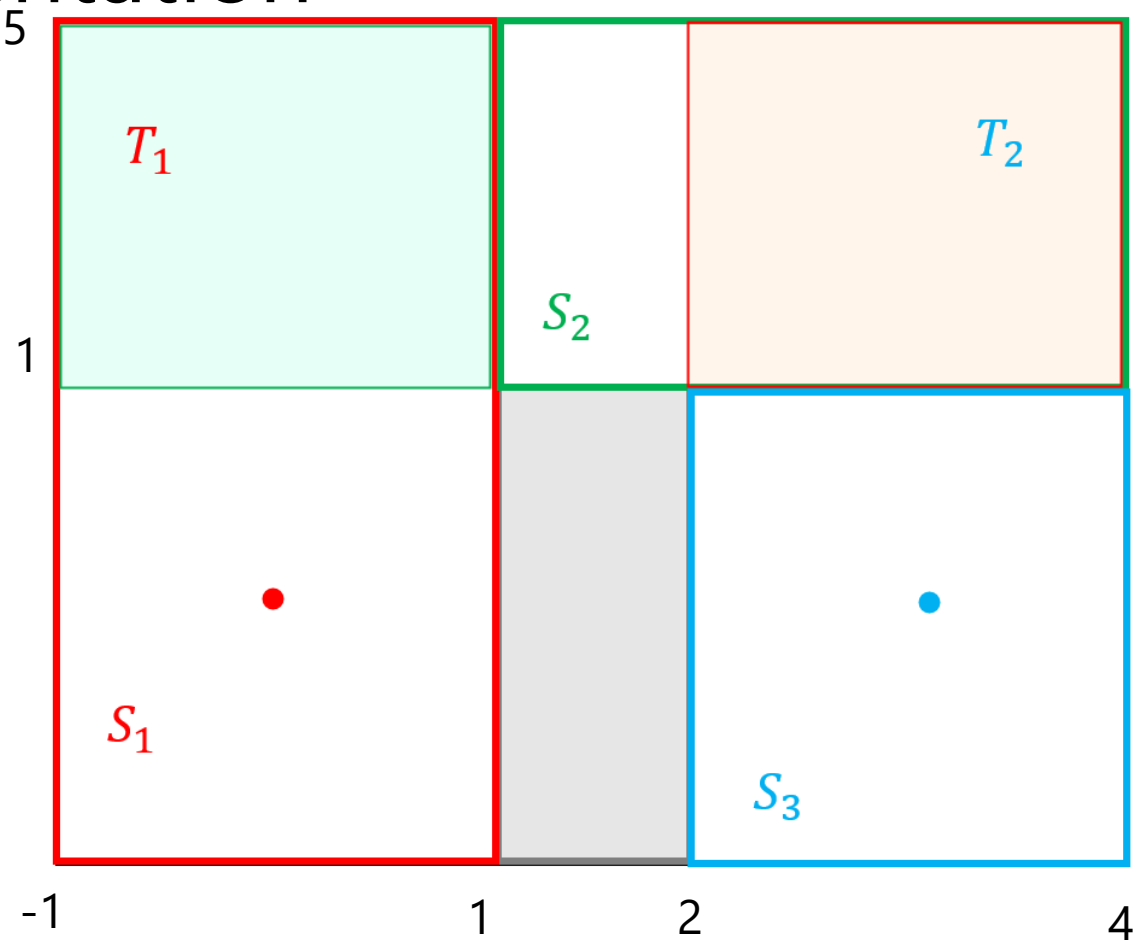
```
import numpy as np
import matplotlib.pyplot as plt
import pyomo.environ as pyo
from pyomo.opt import SolverFactory
from scipy import interpolate

# starting and ending:
start_point = [0, 0]
end_point = [3, 0]
# connected polytopes:
m = 3
A1=[[1, 0], [-1, 0], [0, 1], [0, -1]]
B1 = [1, 1, 2.5, 1]
A2=.....        B2=.....
A2=.....        B3=.....
# Transition zones:
A_T1=.....      B_T1=.....
A_T2=.....      B_T2=.....
```

$$S_1: -1 \le \xi \le 1, \quad -1 \le \eta \le 2.5,$$
$$\Rightarrow \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} \le \begin{bmatrix} 1 \\ 1 \\ 2.5 \\ 1 \end{bmatrix}$$

$$S_2: 1 \le \xi \le 4, \quad 1 \le \eta \le 2.5,$$
$$S_3: 2 \le \xi \le 4, \quad -1 \le \eta \le 1,$$

$$T_1: -1 \le \xi \le 1, \quad 1 \le \eta \le 2.5,$$
$$T_2: 2 \le \xi \le 4, \quad 1 \le \eta \le 2.5,$$

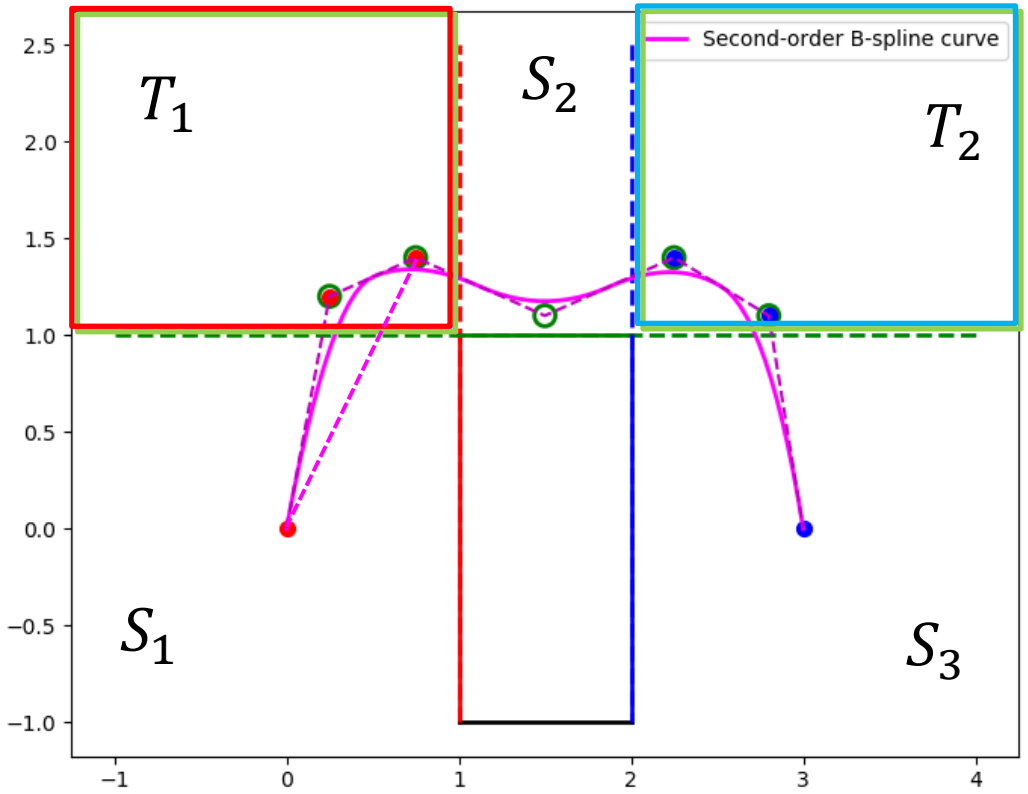


Connected polytopes and their transition zones

B-spline path planner: implementation

```
# degree of Bspline
d = 2
# number of control points
npoint = m + (m-1) * d

model = pyo.ConcreteModel()
model.d = pyo.Param(initialize=d)
model.constraints = pyo.ConstraintList()
model.x = pyo.Var(range(npoint)) # 0, 1, ..., npoint-1
model.y = pyo.Var(range(npoint)) # 0, 1, ..., npoint-1
# first point- initial condition
model.constraints.add(model.x[0] == start_point[0])
model.constraints.add(model.y[0] == start_point[1])
# end point - reference
model.constraints.add(model.x[npoint-1] == end_point[0])
model.constraints.add(model.y[npoint-1] == end_point[1])
```

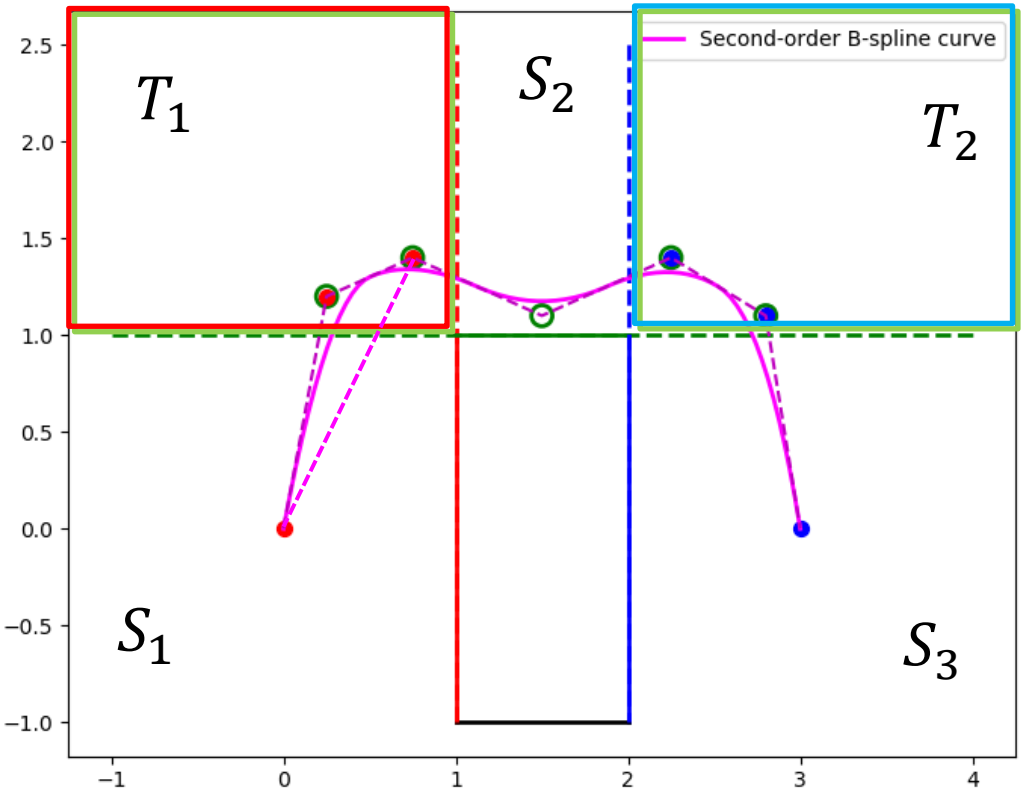


Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding an obstacle.

B-spline path planner: implementation

```
# d point in transition zone T1:
for i in range(d):
    for j in range(len(B_T1)):
        lhs = A_T1[j][0] * model.x[i+1] + A_T1[j][1] * model.y[i+1]
        model.constraints.add(lhs <= B_T1[j])

# 1 point in 2nd polytope:
for j in range(len(B2)):
    lhs = A2[j][0] * model.x[d+1] + A2[j][1] * model.y[d+1]
    model.constraints.add(lhs <= B2[j])
# d point in transition zone T2:
for i in range(d):
    for j in range(len(B_T2)):
        lhs = A_T2[j][0] * model.x[i+d+2] + A_T2[j][1] * model.y[i+d+2]
        model.constraints.add(lhs <= B_T2[j])
```



Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding an obstacle.

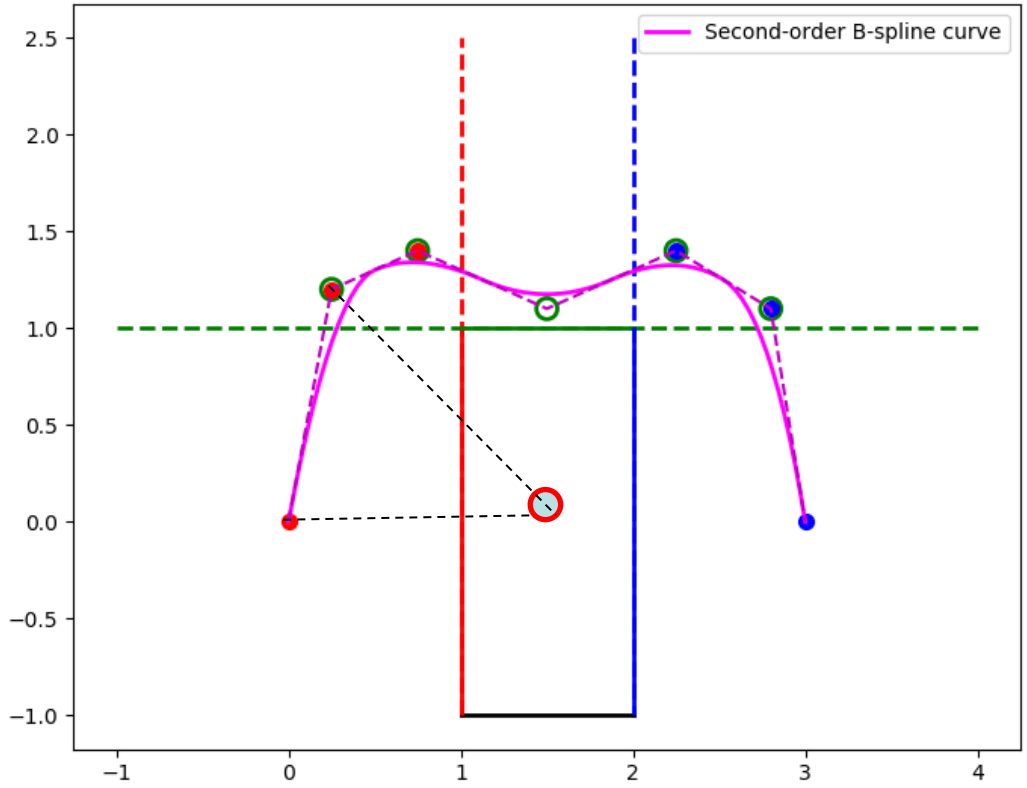
B-spline path planner: implementation

Minimize the distances between all control points and center of the obstacle (1.5, 0)

```
def _obj(model):
    cost= 0
    n = len(model.x)
    for i in range(n):
        cost += (model.x[i]-1.5)**2 + (model.y[i])**2
    return cost

model.obj = pyo.Objective(rule=_obj, sense=pyo.minimize)
SolverFactory('ipopt').solve(model)

ctrl_x=[]
ctrl_y=[]
for i in range(npoint):
    ctrl_x.append(model.x[i].value)
    ctrl_y.append(model.y[i].value)
```

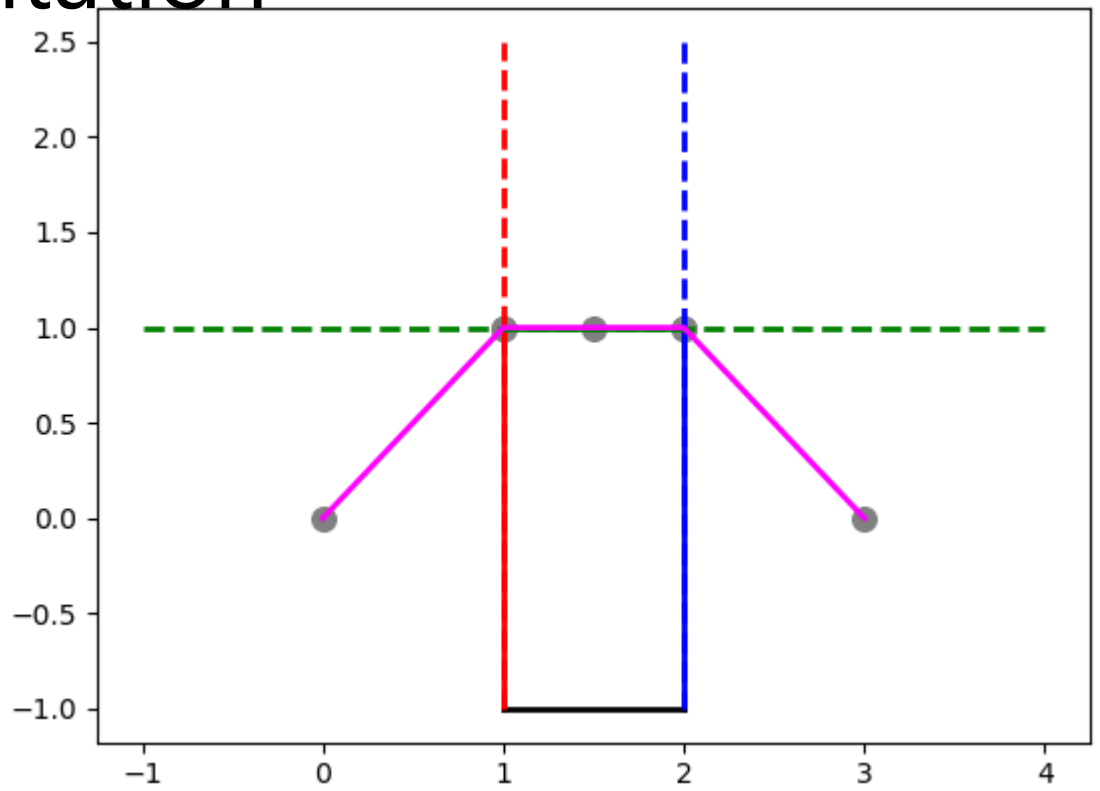


Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding a rectangle obstacle.

B-spline path planner: implementation

```
knot_vector=np.linspace(0,1,npoint+1-d,endpoint=True)
knot_vector=np.append([0]*d,knot_vector)
knot_vector=np.append(knot_vector,[1]*d)
tck=[knot_vector,[ctrl_x, ctrl_y], d] # define a Bspline function as required by scipy
# calculate Bspline
t=np.linspace(0,1,(max(npoint*2,100)),endpoint=True)
out = interpolate.splev(t,tck)

# plotting
obstacle1 = np.array([[1, -1], [1, 1], [2, 1], [2, -1], [1, -1]])
plt.figure()
plt.plot(obstacle1[0:2,0],obstacle1[0:2,1],'red',linewidth=2.0)
plt.plot(obstacle1[1:3,0],obstacle1[1:3,1],'green',linewidth=2.0)
plt.plot(obstacle1[2:4,0],obstacle1[2:4,1],'blue',linewidth=2.0)
plt.plot(obstacle1[3:5,0],obstacle1[3:5,1],'black',linewidth=2.0)
plt.scatter(ctrl_x, ctrl_y, color='gray', s= 70, label='Control points')
plt.plot(out[0],out[1],'magenta',linewidth=2.0,label='Second-order B-spline curve')
plt.legend(loc='best')
```



Second-order B-spline curve starting from (0,0) and ending at (3,0) avoiding a rectangle obstacle.