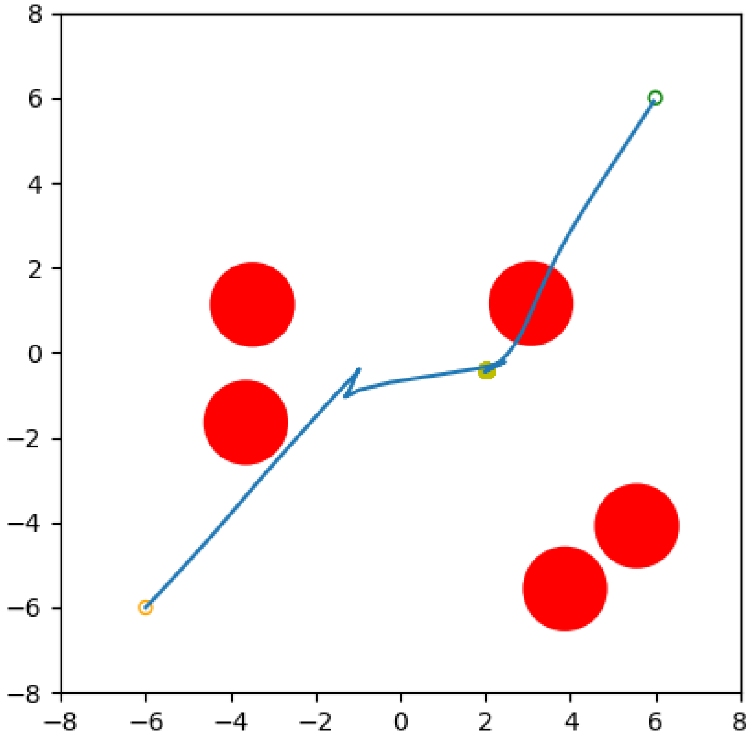
Just relax and add some slacks

Dynamic Collision Avoidance using RTI-MPC

Philipp Bordne, Ahmet Selim Canakci, Abdelhakim Khaled Amer

Problem description

Robot needs to cross square with circular obstacles moving around (perturbed by noise) without colliding with them:



General continuous time formulation

Adapted from *B. Brito, M. Everett, J. P. How and J. Alonso-Mora: “Where to go next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments”, 2021*

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \frac{1}{2}\int_0^{\mathrm{T}} \mathrm{L}\big(\mathbf{u}(\mathrm{t})\big)\mathrm{dt}$$

$$\begin{aligned}
\mathbf{s.t} \quad & \mathbf{x}(0) - \bar{\mathbf{x}}_{\mathbf{0}} = \mathbf{0} & \\
& \dot{\mathbf{x}}(\mathbf{t}) = \mathbf{f}\big(\mathbf{x}(\mathrm{t}), \mathbf{u}(\mathrm{t})\big) & \forall \mathrm{t} \in [0, \mathrm{T}] \\
& \|\mathbf{x}(\mathrm{T}) - \mathbf{g}\| - \boldsymbol{\delta} \leq \mathbf{0} & \\
& \boldsymbol{\mathcal{O}}^{\mathbf{R}}(\mathrm{t}) \cap \boldsymbol{\mathcal{O}}^{(\mathbf{j})}(\mathrm{t}) = \emptyset & \forall \mathrm{t} \in [0, \mathrm{T}], \mathrm{j} = 1, \ldots, \mathrm{M} \\
& \mathbf{u}(\mathrm{t}) \in \boldsymbol{\mathcal{U}}, \mathbf{x}(\mathrm{t}) \in \boldsymbol{\mathcal{X}} & \forall \mathrm{t} \in [0, \mathrm{T}]
\end{aligned}$$

Formulation as RTI problem

Again based on *B. Brito, M. Everett, J. P. How and J. Alonso-Mora: "Where to go next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments", 2021*

- Real Time Iteration (RTI) stopped as soon $\boldsymbol{x}(0)$ lies within some margin
- Least squares formulation as in Acados documentation

$$\min_{\substack{\mathbf{x}_0,\dots,\mathbf{x}_N\\ \mathbf{u}_0,\dots\mathbf{u}_{N-1}\\ \mathbf{s}_0,\dots,\mathbf{s}_N}} \frac{1}{2}\|V_x^e\mathbf{x}_N-\mathbf{g}\|_{W^e}^2+\mathbf{s}_N Z_N\mathbf{s}_N+\sum_{i=0}^{N-1}\|V_u\mathbf{u}_i\|_W^2+\mathbf{s}_i Z_i\mathbf{s}_i$$

$$\begin{aligned}
\text{s.t.}\quad && \mathbf{x}_0 &= \mathbf{x}(0) && \\
&& \mathbf{x}_{i+1} &= F(\mathbf{x}_i,\mathbf{u}_i) && i=0,\dots,N-1\\
&& \left(x_i^R-x_i^{(j)}\right)^2+\left(y_i^R-y_i^{(j)}\right)^2-\left(r^R+r^{(j)}+m\right)^2+s_{ij} &\geq 0 && i=0,\dots,N\,;j=1,\dots,M\\
&& -\mathbf{x}_{max}\leq J_{bx}\mathbf{x}_i &\leq \mathbf{x}_{max} && i=0,\dots,N\\
&& \mathbf{s}_i &\geq 0 && i=0,\dots,N
\end{aligned}$$

Detailed look on variables and dynamics model

Dynamics taken from *B. Brito, M. Everett, J. P. How and J. Alonso-Mora: "Where to go next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments", 2021*

$$\mathbf{x}_\mathrm{i} = \begin{bmatrix} x_i \\ y_i \\ \psi_i \\ v_i \\ \omega_i \end{bmatrix} \qquad \dot{\mathbf{x}}_\mathrm{i} = \begin{bmatrix} v_i \cos\psi_i \\ v_i \sin\psi_i \\ \omega_i \\ u_{i_1} \\ u_{i_2} \end{bmatrix} \qquad \mathbf{u}_\mathrm{i} \in \mathbb{R}^2$$

$\mathrm{F}: \mathbb{R}^7 \rightarrow \mathbb{R}^5$: implicit Runge-Kutta integrator

$$\mathbf{s}_\mathrm{i} \in \mathbb{R}^\mathrm{M} \qquad \mathrm{Z}_i = \alpha_\mathrm{i}^{\mathbf{x}(0)} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{\mathrm{M}\times\mathrm{M}}$$

with α_i discount for constraint violations at later stages

$$W^e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

no penalty on orientation of robot

$$\mathbf{x}_\mathrm{max} = \begin{bmatrix} x_\mathrm{max} \\ y_\mathrm{max} \\ v_\mathrm{max} \\ \omega_\mathrm{max} \end{bmatrix}$$

: no limit on angular velocity

$$\mathrm{J}_\mathrm{bx} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Implementation in Acados

- RTI with solving for N=20 prediction steps
- parameterized model for obstacle positions per iteration:
  ```
  h += [(model.x[0] - model.p[2*i])**2 + (model.x[1] - model.p[2*i+1])**2 - (R_OBST + R_ROBOT + MARGIN)**2]
  ```
- slack variables:
  even when running „full“ SQP solver hard to solve for N > 10
- scaling $\alpha_i^{\boldsymbol{x}(0)}$ according to current robot position $\boldsymbol{x}(0)$:
  compensate that L2 penalty on robot position will be higher when further away from the goal
- adding margin $m > 1$ on constraints:
  gets us into steep region of L2-loss as soon as we really would hit obstacle
- initialization strategy:
  set controls to 0 and state to current

Demo

Results

- For M=5 obstacles, radius=1, max_v=2
  vs robot radius=0.2, max_v=10
- Noisy obstacle movements (10% of current obstacle velocity):
  Scenario RANDOM: collision in 18% of cases
  Scenario EDGE: collision in 38% of cases
- No noise on obstacles:
  Scenario RANDOM: collision in 19% of cases
  Scenario EDGE: collision in 10% of cases
- *Further data to be evaluated (goals reached, leaving admissable square)*

Synergies high level RL could provide

RL agent proposes subgoals for the MPC controller
as proposed by B. Brito et al (current understanding, need to look in more detail):
⍰ better long term anticipation of unsolvable configurations
⍰ better dealing with sochasticity
⍰ shorter overall paths to goal

for our specific solution proposal:
⍰ alleviate need for adaptation of slack penalty

Acknowledgements

B. Brito, M. Everett, J. P. How and J. Alonso-Mora: “Where to go next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments”, 2021

R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen and Moritz Diehl: “acados -- a modular open-source framework for fast embedded optimal control”, 2021