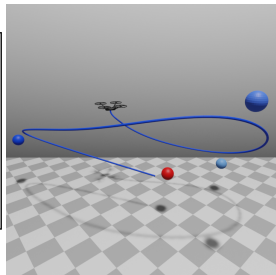
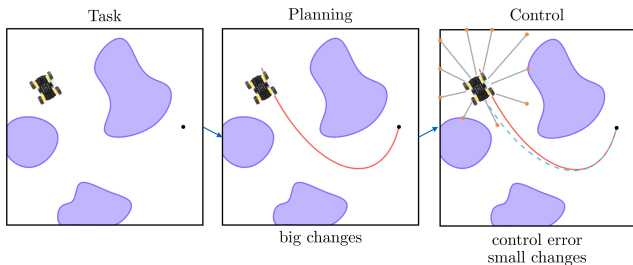


# Motion planning and control

**Vojtěch Vonásek**

Department of Cybernetics  
Faculty of Electrical Engineering  
Czech Technical University in Prague



## Classic decoupled planning-control approach

- A global plan is delivered by motion planning
- Robot executes the plan using a controller
  - controller uses fresh sensor data to handle control imprecision, disturbances
  - can be extended to cope with obstacles (obstacle avoidance)
- New global plan is replanned when a global change occurs

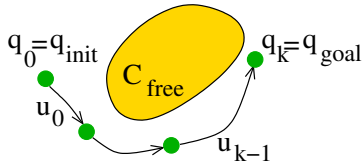
- Let  $L(\pi_k)$  is the cost of the sequence  $\pi_k = (u_0, \dots, u_{k-1})$

$$L(\pi_k) = l_f(q_k) + \sum_{i=0}^{k-1} l(q_i, u_i)$$

- the final term  $l_f(q_k) = 0$  if  $q_k = q_{\text{goal}}$ ; it is  $\infty$  otherwise

## Discrete optimal planning

$$\begin{aligned} & \underset{\pi_k=(u_0, \dots, u_{k-1})}{\text{minimize}} && L(\pi_k) \\ & \text{subject to} && q_{k+1} = f(q_k, u_k) \\ & && q_0 = q_{\text{init}} \\ & && q_k = q_{\text{goal}} \\ & && q_k \in \mathcal{C}_{\text{free}} \end{aligned}$$



- $L(\pi_k) = \infty$  means an infeasible solution
- $L(\pi_k) < \infty$  means a feasible solution with the cost  $L(\pi_k)$

- Optimal control for a discrete-time (and a finite horizon)
- initial state is  $x_i$ , goal state  $x_n$  may be given (or not)

$$\underset{u_i, \dots, u_{N-1}, (x_i), \dots, x_n}{\text{minimize}} \quad \left( \phi(x_n, N) + \sum_{k=i}^{N-1} L_k(x_k, u_k) \right)$$

$$\begin{aligned} \text{subject to} \quad & x_{k+1} = f_k(x_k, u_k) \\ & u_{lb} \leq u_k \leq u_{ub} \\ & x_{lb} \leq x_k \leq x_{ub} \end{aligned}$$

## Discrete optimal control (generally)

$$\underset{x \in \mathbf{R}^{n(N-i)}, u \in \mathbf{R}^{m(N-i)}}{\text{minimize}} \quad J(x, u)$$

$$\begin{aligned} \text{subject to} \quad & g(x, u) = 0 \\ & h(x, u) \leq 0 \end{aligned}$$

- Formulations of optimal planning and control are same
- Both can find path/trajectory from start to goal
- What is the practical difference?

## Motion planning

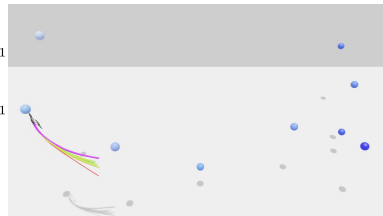
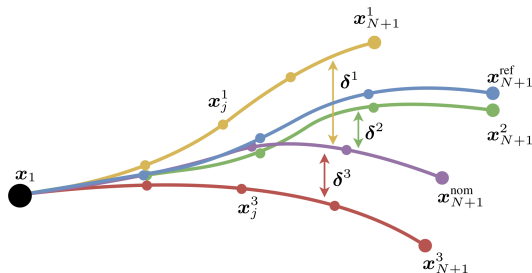
- Suitable for making long-term and complex plans
- Solution is (generally) achieved by searching  $\mathcal{C}$ -space
- Difficult to react on changes (robot control error, dynamic obstacles) → replanning
- Replanning requires to solve the problem from scratch → slow

## Control

- Achieved via mathematical optimization
- Difficult to find first (feasible) solution in large search space → needs a good initialization
- In robotics, usually used for reference tracking

**Does not make sense to solve motion plan by control-theory methods**  
**Does not make sense to control via planning!**

- Control on a finite prediction horizon
- Generate several trajectories (rollouts) using forward motion model
- Evaluate each rollout by a cost functions
- Compute new nominal trajectory as a weighted sum of the rollouts
- Control robot using the first step of the nominal trajectory, and repeat



- Compute  $K$  rollouts from the initial state  $\mathbf{x}_0$

$$\mathbf{x}^k = (\mathbf{x}_0^k, \dots, \mathbf{x}_j^k, \dots, \mathbf{x}_{N-1}^k, \mathbf{x}_N^k)$$

$$\mathbf{u}^k = (\mathbf{u}_0^k, \dots, \mathbf{u}_j^k, \dots, \mathbf{u}_{N-1}^k)$$

$$\delta^k = (\delta \mathbf{u}_0^k, \dots, \delta \mathbf{u}_j^k, \dots, \delta \mathbf{u}_{N-1}^k)$$

$$\mathbf{x}_{j+1}^k = \mathbf{x}_j^k + \mathbf{f}_{\text{RK4}}(\mathbf{x}_j^k, \mathbf{u}_j^k, \Delta t)$$

$$\delta \mathbf{u}_j^k \in \mathcal{N}(\mathbf{0}, \Sigma)$$

$$\mathbf{u}_j^k = \mathbf{u}_j^{\text{nom}} + \delta \mathbf{u}_j^k$$

$$k = 1, \dots, K$$

$$j = 0, \dots, N-1$$

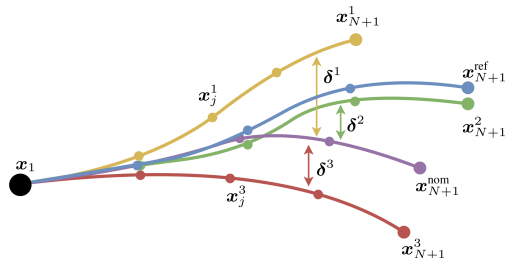
- Integration of forward motion model (e.g. using 4-th order Runge-Kutta method  $\mathbf{f}_{\text{RK4}}(\mathbf{x}_j^k, \mathbf{u}_j^k, \Delta t)$ )

- Rollout evaluation

$$S_k = \text{ComputeCost}(\mathbf{x}^k, \mathbf{u}^k)$$

- Compute new nominal trajectory

$$\mathbf{u}_j^{\text{nom}} := \mathbf{u}_j^{\text{nom}} + \sum_{k=1}^K \omega_k \cdot \delta \mathbf{u}_j^k$$



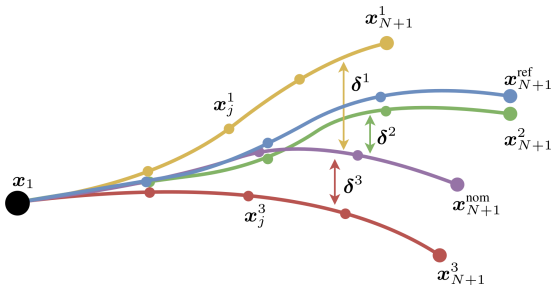
- Each rollout  $k$  is evaluated using  $S_k = \text{ComputeCost}(\mathbf{x}^k, \mathbf{u}^k)$
- Smaller cost is better
- Weights  $\omega_k$

$$\omega_k = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} (S_k - \rho)\right)$$

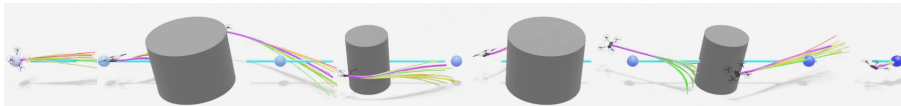
$$\eta = \sum_{k=1}^K \exp\left(-\frac{1}{\lambda} (S_k - \rho)\right)$$

$$\rho = \min\{S_1, \dots, S_K\}$$

- $\lambda$  scales contribution of the rollouts to final weights



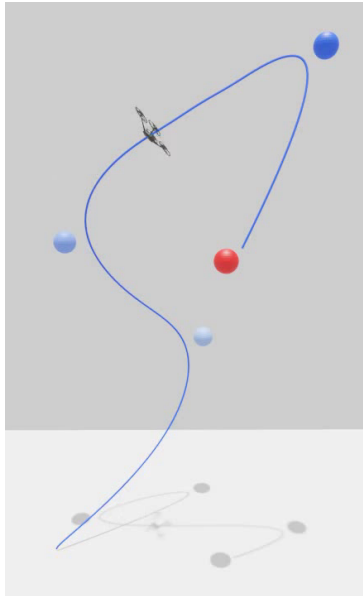
- Fast motion control
- Allows non-linear forward motion model
- Allows arbitrary cost functions (do not need to be convex)
  - useful e.g. to handle obstacles, formation coordination,
  - reference trajectory following
- When  $\lambda$  is low,  $\omega = (0, \dots, 0, 1, 0, \dots, 0)$ , where 1 is for the best rollout
- When  $\lambda$  is high,  $\omega = (\frac{1}{K}, \dots, \frac{1}{K})$



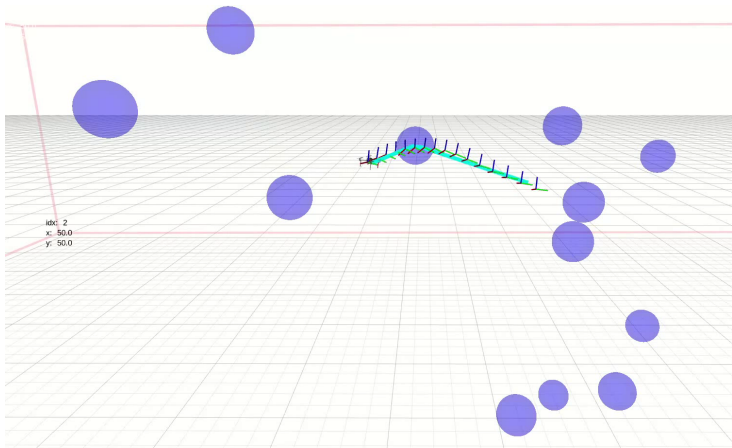
## Motion planning + MPPI

- Motion planning provides reference trajectory  $\mathbf{u}^{\text{nom}}$
- MPPI updates control inputs based on actual situation

# MPPI examples



# MPPI examples



# MPPI examples



# MPPI examples



