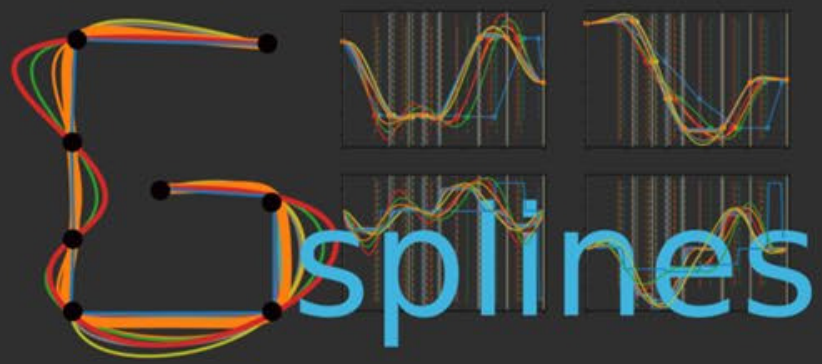
ImFusion

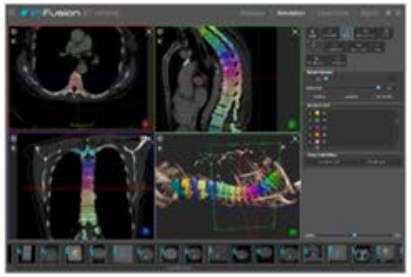


GSplines: Generalized Splines for Motion Optimization and Smooth Collision Avoidance

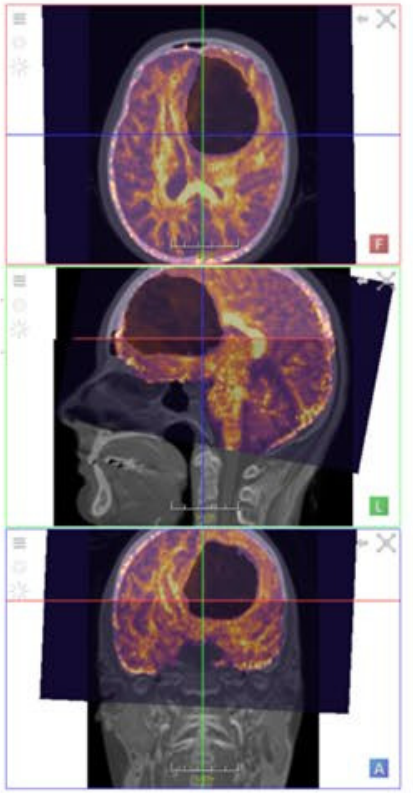
Rafael A. Rojas

ROSCon Odense: Day One, October 22nd, 2024

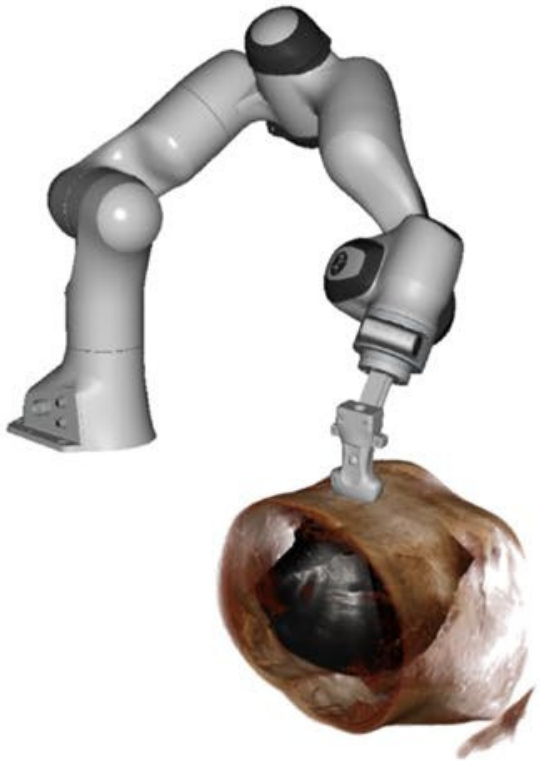
Who we are?

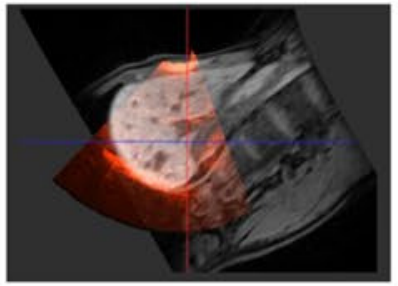


- Founded: 2012 | Location: Munich, Germany
- Private & Independent R&D Lab
- Software framework: trusted by industry leaders, start-ups, and research labs
- Academic engagement: Actively contributing to the research community
- Our team:
  - 50+ dedicated employees and several students
  - Global Diversity: 10+ Nationalities
  - Excellence in Expertise: 50% of technical staff hold a PhD
  - Groups: Computer Vision, Machine Learning, Ultrasound, CT & Interventional X-Ray, Robotics.



Robotics





- Deep integration with our framework in Robotics
- ROS integration
- Native support for popular robots: Franka Panda, Universal Robot UR-e series, Kuka
- Advanced utilities for key use cases, e.g. hand-eye calibration
- The ImFusion Robotics Team:





Table of Contents

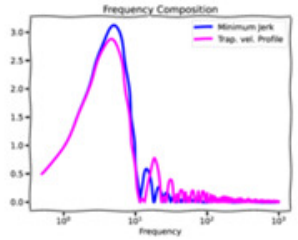
Motivation: Academic Interest

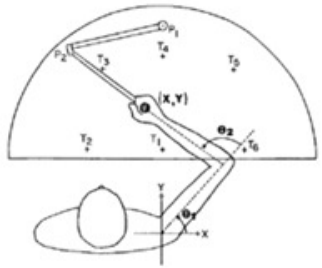
$$\min \int_0^T \left\| \frac{\mathrm{d}^3 \mathbf{q}}{\mathrm{d}t^3} \right\| \mathrm{d}t$$

- **Minimum Jerk** seems to be a thing
  - Low frequency composition: low wear
  - Human-like motions: Flash and Hogan
  - Psychological safety

The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model[1]

TAMAR FLASH*,[2] AND NEVILLE HOGAN‡[3]





Reduction of Heart Rate by Robot Trajectory Profiles in Cooperative HRI

Barbara Kühnlenz and Kolja Kühnlenz

Motivation: Academic Interest

- Minimum Snap is of interest for drone motion planning
- This concept may be generalized to Minimum-X trajectories

$$\min \int_0^T \left\| \frac{\mathrm{d}^X \mathbf{q}}{\mathrm{d}t^X} \right\| \mathrm{d}t$$

$$\min \int_0^T \left\| \frac{\mathrm{d}^4 \mathbf{q}}{\mathrm{d}t^4} \right\| \mathrm{d}t$$

Minimum Snap Trajectory Generation and Control for Quadrotors

Daniel Mellinger and Vijay Kumar

***Abstract*— We address the controller design and the trajectory generation for a quadrotor maneuvering in three dimensions in a tightly constrained setting typical of indoor environments. In such settings, it is necessary to allow for significant excursions of the attitude from the hover state and small angle approximations cannot be justified for the roll and pitch. We develop an algorithm that enables the real-time generation of optimal trajectories through a sequence of 3-D positions and yaw angles, while ensuring safe passage through specified corridors and satisfying constraints on velocities, accelerations and inputs. A nonlinear controller ensures the faithful tracking of these trajectories. Experimental results illustrate the application of the method to fast motion (5-10 body lengths/second) in three-dimensional slalom courses.**

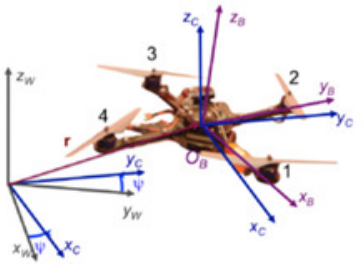
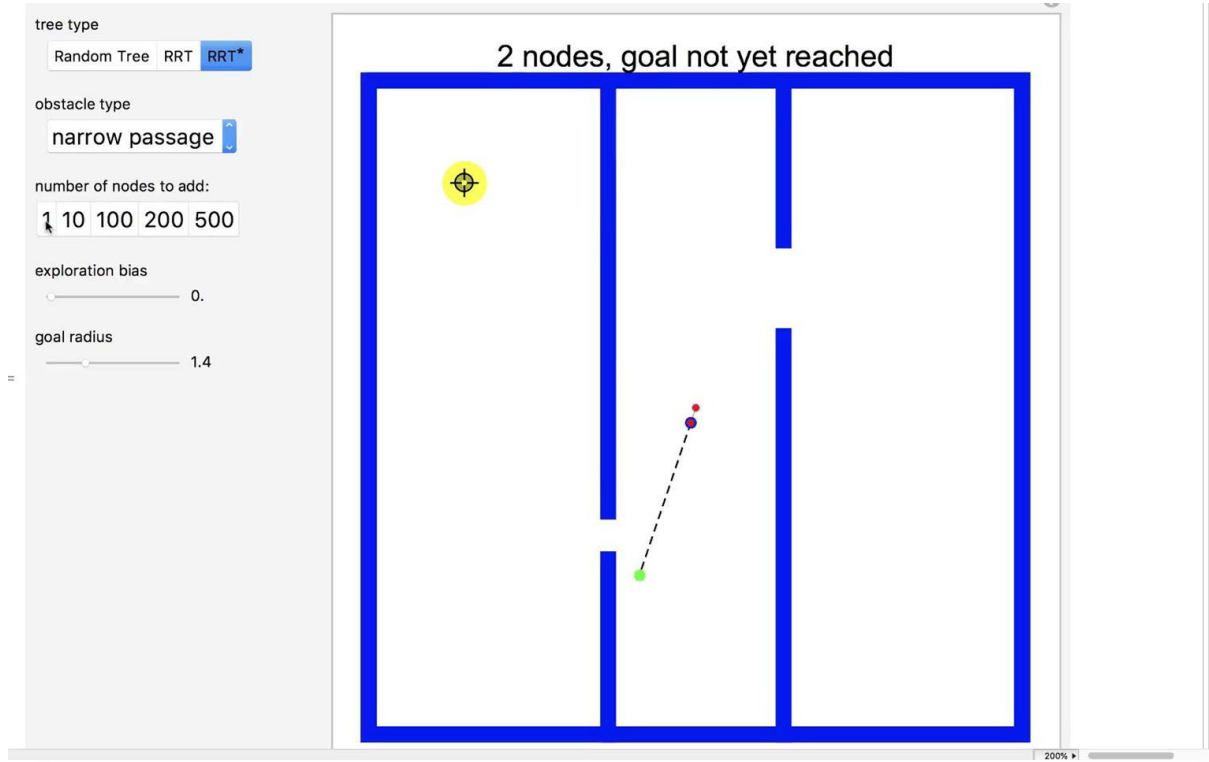


Fig. 1. The flat outputs and the reference frames.

Motivation: Output of Motion Planners

The output of of sampled based motion players are waypoints



RRT, RRT* & Random Trees

Aaron Becker
4.59K subscribers
Subscribed
1.3K
Share

Motivation: Output of Motion Planners

- Lead-Through/Hand-Guiding programming of cobots

Motivation: Current Classes Used in ROS Ecosystem

- Default ROS messages types for are defined as nested structrures. The array-of-structures format leads to inefficient memory layout and complicates the translation into the flat vector format that optimizers require.
- Other types are thighty coupled with the robot model
- do not provide clear interface for
  - Evaluation at arbitrary time instantes
  - Computation of the derivatives
  - Time scaling

Trajectory Message

```
std_msgs/msg/Header header
string[] joint_names
trajectory_msgs/msg/JointTrajectoryPoint[] points
```

Trajectory Point

```
double[] positions
double[] velocities
double[] accelerations
double[] effort
builtin_interfaces/msg/Duration time_from_start
```

Chomp trajectory

```
class ChompTrajectory
{
public:
...
    ChompTrajectory(const moveit::core::RobotModelConstPtr& robot_model,
    double& operator()(size_t traj_point, size_t joint);
...
private:
...
    void init();
    std::string planning_group_name_;
    Eigen::MatrixXd trajectory_;
...
};
```

Goals

Define class `GSpline` type that is capable of

- Representing Solutions of the generalized Minimum-X problem: a *convex* combination of norms of derivatives

$$\min \int_{t_0}^{t_f} \alpha_1 \left\| \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} \right\|^2 + \alpha_2 \left\| \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} \right\|^2 + \cdots + \alpha_1 \left\| \frac{\mathrm{d}^k\mathbf{q}}{\mathrm{d}t^k} \right\|^2 \mathrm{d}t$$

$$\mathbf{q}(t_i) = \mathbf{w}_i \quad t_i \text{ is unknown}$$

- Representing transformations
  - Derivatives
  - Linear operations
  - Linear Scaling
- Available in  and 
- Remark: This is not an alternative other general optimization tools like CASADI.

Achieving Our Goal Through Variational Principles

We call a `GSpline` a solution of the Euler-Lagrange equations of our cost

$$-\alpha_1 \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} + \alpha_2 \frac{\mathrm{d}^4\mathbf{q}}{\mathrm{d}t^4} - \cdots + (-1)^k \alpha_k \frac{\mathrm{d}^{2k}\mathbf{q}}{\mathrm{d}t^{2k}} = 0 \quad \text{a.e.}$$

Any `GSpline` Have the following properties

- Analytical Consistency

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k} : \mathrm{GSpline} \longrightarrow \mathrm{GSpline}$$

- Algebraic Consistency I

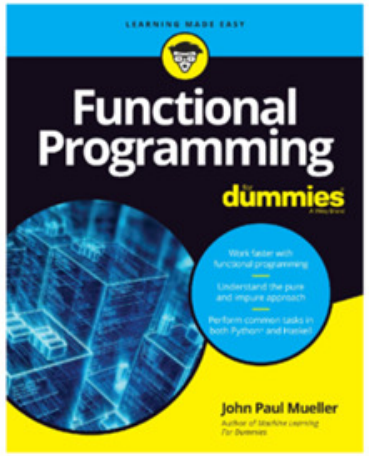
$$+ : \mathrm{GSpline} \times \mathrm{GSpline} \longrightarrow \mathrm{GSpline}$$
$$\cdot : \mathbb{R} \times \mathrm{GSpline} \longrightarrow \mathrm{GSpline}$$

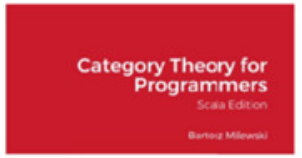
- Algebraic Consistency II

$$[t \to \sigma t] : \mathrm{GSpline} \longrightarrow \mathrm{GSpline}$$

Goals: Monoid property

- We desire to achieve the monoid property on the derivative, scaling and addition/scalar multiplication operations. These operation will become a monoid under the composition.
- Composition is a fundamental tool in programming, the Monoid structure is the most fundamental compositional structure



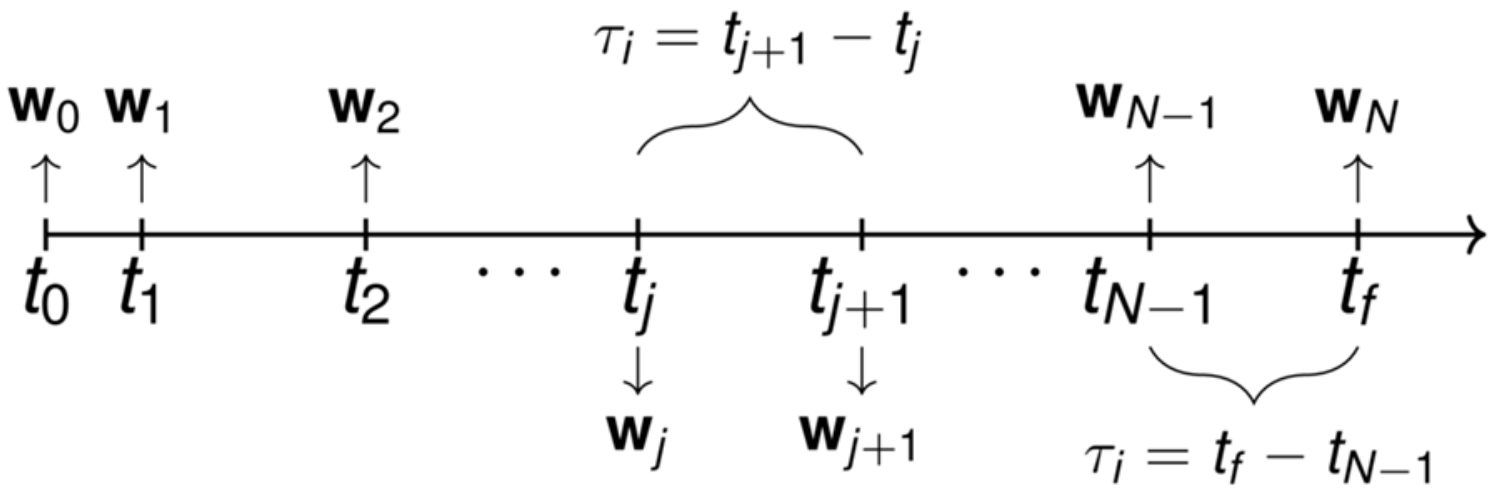


Class Design: Solution Representation

The solution of the corresponding Euler-Lagrange equations

$$-\alpha_1 \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} + \alpha_2 \frac{\mathrm{d}^4\mathbf{q}}{\mathrm{d}t^4} - \cdots + (-1)^k \alpha_k \frac{\mathrm{d}^{2k}\mathbf{q}}{\mathrm{d}t^{2k}} = 0 \quad \text{a.e.}$$

Unknows:
t_j and τ_j

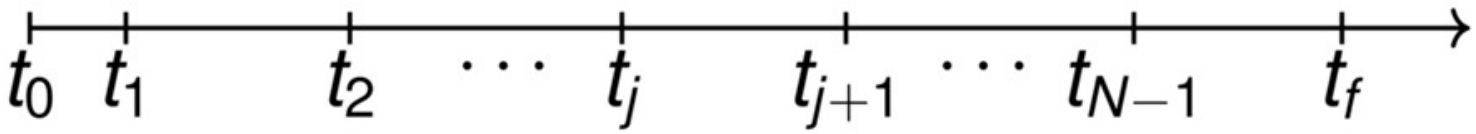


Class Design: Solution Representation

The solution of the corresponding Euler-Lagrange equations

$$-\alpha_1 \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} + \alpha_2 \frac{\mathrm{d}^4\mathbf{q}}{\mathrm{d}t^4} - \cdots + (-1)^k \alpha_k \frac{\mathrm{d}^{2k}\mathbf{q}}{\mathrm{d}t^{2k}} = 0 \text{ a.e.}$$

$$\mathbf{q}(t) = \sum_{i=1}^{2k} \mathbf{a}_i{}^j B_i(t)$$



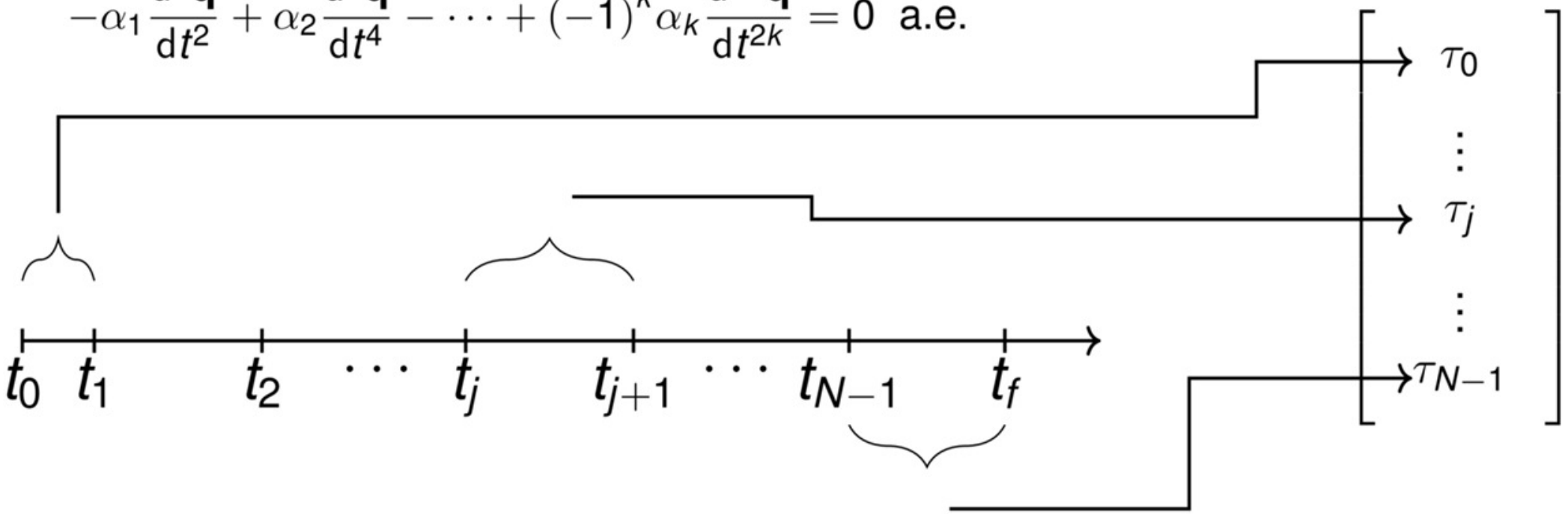
$$\mathbf{q}(t) = \sum_{i=1}^{2k} \mathbf{a}_i{}^1 B_i(t) \qquad \mathbf{q}(t) = \sum_{i=1}^{2k} \mathbf{a}_i{}^1 B_i(t)$$

$$\begin{bmatrix} \vdots \\ \mathbf{a}_0^j \\ \vdots \\ \mathbf{a}_{2k}^j \\ \vdots \\ \mathbf{a}_0^{N-1} \\ \vdots \\ \mathbf{a}_{2k}^0 \end{bmatrix}$$

Class Design: Solution Representation

The solution of the corresponding Euler-Lagrange equations

$$-\alpha_1 \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} + \alpha_2 \frac{\mathrm{d}^4\mathbf{q}}{\mathrm{d}t^4} - \cdots + (-1)^k \alpha_k \frac{\mathrm{d}^{2k}\mathbf{q}}{\mathrm{d}t^{2k}} = 0 \text{ a.e.}$$



Class Design: Solution Representation

The solution of the corresponding Euler-Lagrange equations

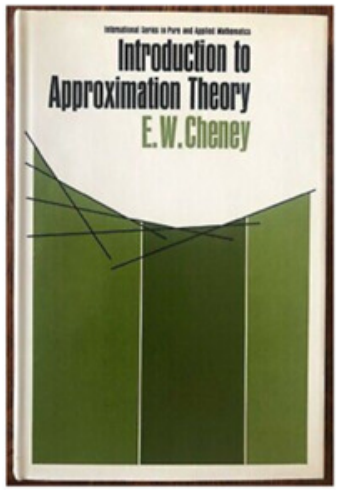
$$-\alpha_1 \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} + \alpha_2 \frac{\mathrm{d}^4\mathbf{q}}{\mathrm{d}t^4} - \cdots + (-1)^k \alpha_k \frac{\mathrm{d}^{2k}\mathbf{q}}{\mathrm{d}t^{2k}} = 0 \text{ a.e.}$$

```
class GSpline{
    Eigen::VectorXd coefficients_;
    Eigen::VectorXd time_interval_lengths_;
    Basis basis_;
}
```

$\longrightarrow [\mathbf{a}_0^1 \ \cdots \ \mathbf{a}_{2k}^1 \cdots]$

$\longrightarrow [\tau_1 \ \cdots \ \tau_N]$

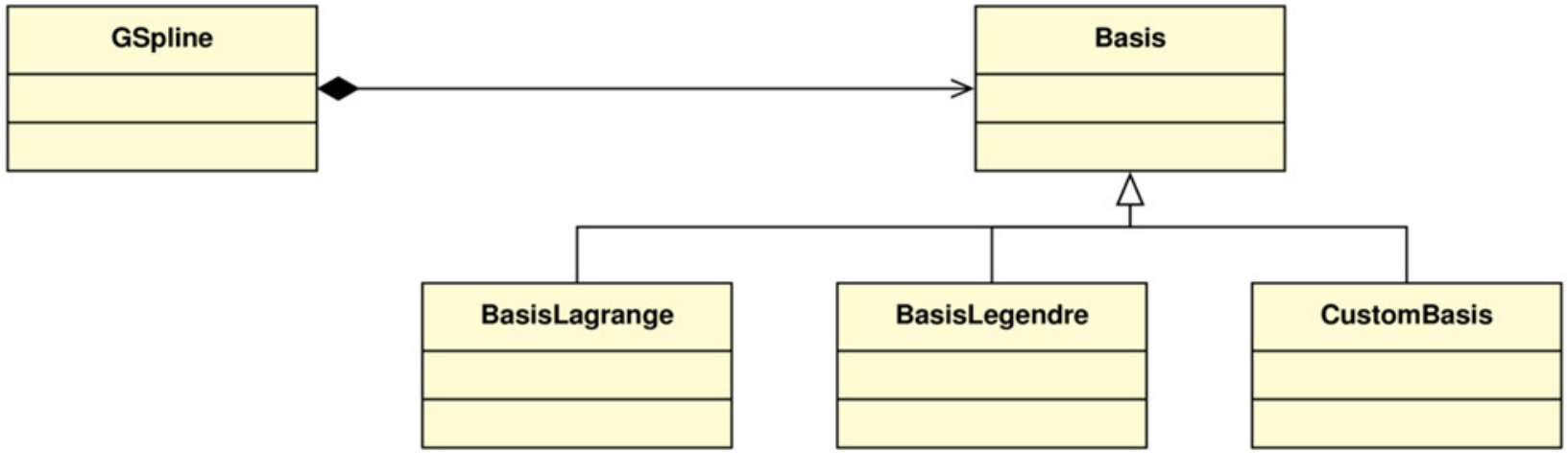
$\longrightarrow [B_i, \cdots, B_{2k}]$



Class Design: Overview

```cpp
class GSpline {
private:
    Eigen::VectorXd time_interval_lengths_;
    Eigen::VectorXd coefficients_;
    double initial_time_;
    std::unique_ptr<Basis> basis_;
public:
    GSpline linear_scaling_new_execution_time(double _T) const;
    GSpline operator+(const GSpline& that);
    GSpline operator*(double num);
    GSpline derivative(std::size_t deg);
    bool operator()(double time) const;
};
```

```cpp
class Basis {
protected:
    Eigen::VectorXd parameters_float_;
    Eigen::VectorXi parameters_int_;
public:
    virtual void eval_on_window(
                        ..., Eigen::VectorXd&_buff) const = 0;
    virtual void eval_derivative_on_window(
                        ..., Eigen::VectorXd& _buff) const = 0;
    virtual void eval_derivative_wrt_tau_on_window(
                        ..., Eigen::VectorXd& _buff) const = 0;
    virtual std::unique_ptr<Basis> clone() const = 0;
};
```



Class Design: Bases Implemented

- Polynomial Basis solution of

$$\min \int_{t_0}^{t_f} \left\| \frac{\mathrm{d}^k \mathbf{q}}{\mathrm{d}t^k} \right\|^2 \mathrm{d}t \tag{1}$$

  - `basis::BasisLegendre`
  - `basis::BasisLagrange`: Gauss-Radau and Gauss-Lobatto

- Non Polynomial basis

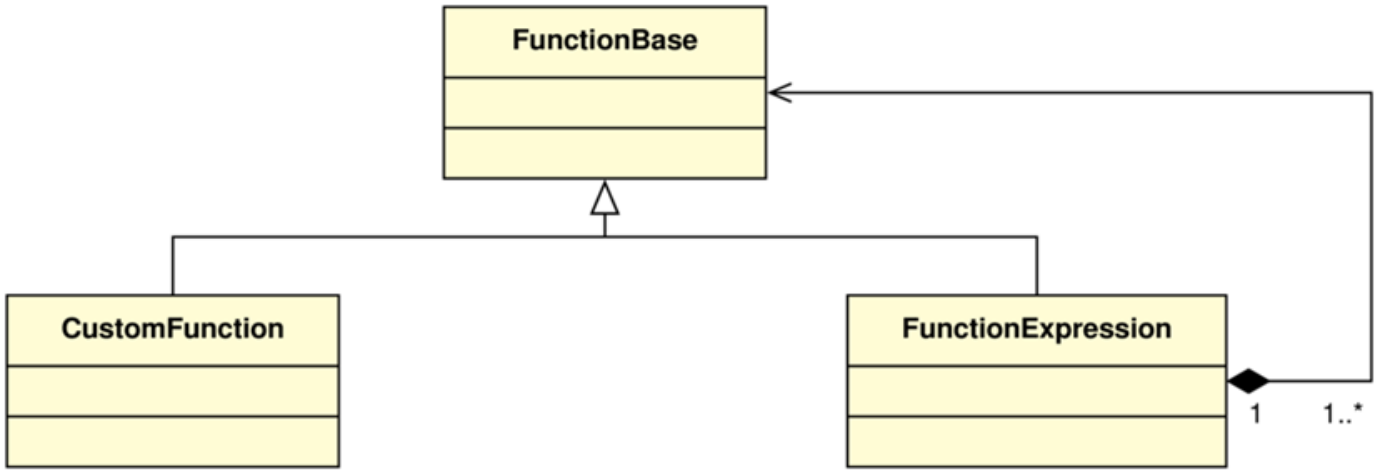
$$\min \int_{t_0}^{t_f} \alpha \left\| \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} \right\|^2 + (\alpha - 1) \left\| \frac{\mathrm{d}^3 \mathbf{q}}{\mathrm{d}t^3} \right\|^2 \mathrm{d}t \tag{2}$$

  - `basis::Basis0101`

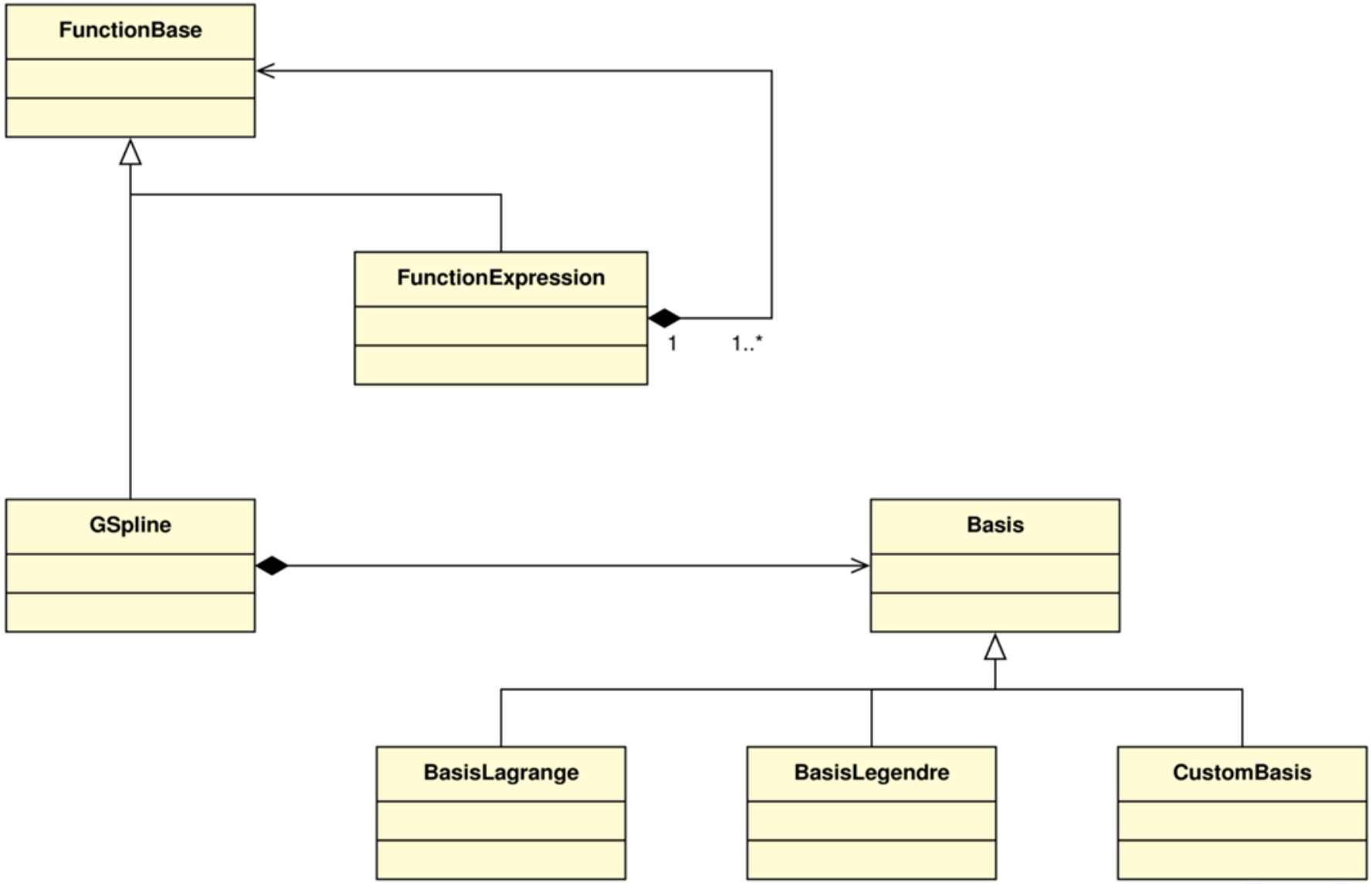
Design: General operations

To work with **trajectories that are not gsplines** we provide interface to construct general differentiable expressions. For example:

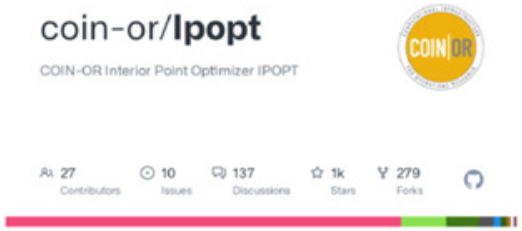
- Scalar `GSplines` multiplied by a vector `GSpline` $q \cdot \mathbf{p}$
- The norm of a derivative $\|\ddot{\mathbf{q}}\|^2$
- Composition of `GSplines` $\mathbf{q} \circ s$



Design: General Overview



Out of the Box Optimization

- 

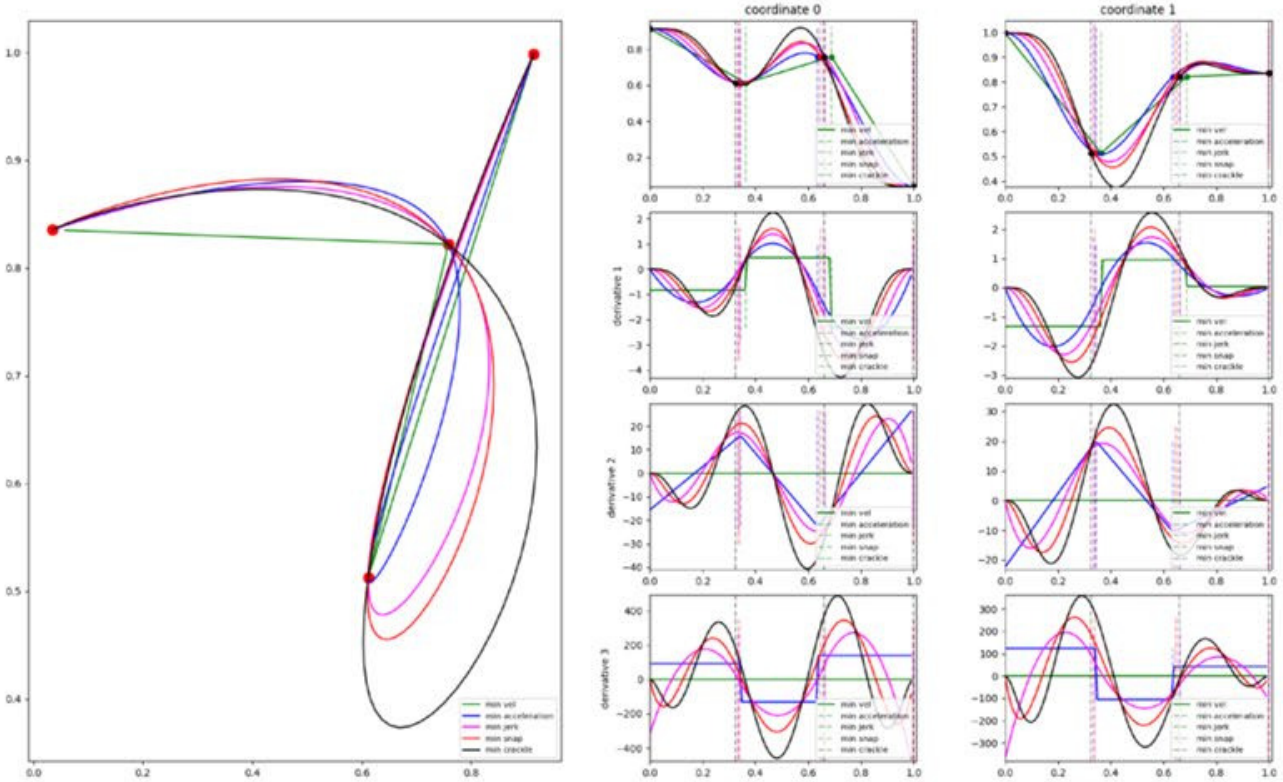
- IFOPT optimizer interface

https://github.com/ethz-adrl/ifopt

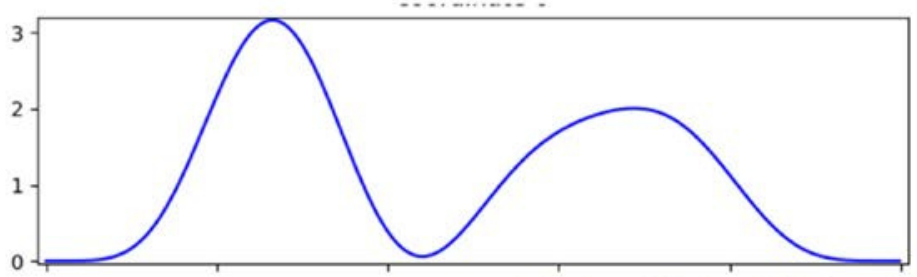
- We provide custom cost functions and constraints to build
- Minimum-X Trajectories
- Minimum Time smooth and path consistent collision avoidance

Optimization: Trajectory Generation

```
waypoints = np.random.rand(4, 2)
straigh_lines = broken_lines_path(waypoints)
min_acc = minimum_acceleration_path(waypoints)
min_jerk = minimum_jerk_path(waypoints)
min_snap = minimum_snap_path(waypoints)
min_crackle = minimum_crackle_path(waypoints)
plot2d_compare([straigh_lines, min_acc, min_jerk, min_snap, m
            'green', 'blue', 'magenta', 'red', 'black'],
            ['min_vel', 'min_acceleration', 'min_jerk',
            'min_snap', 'min_crackle'])
time_1 = 0.1
time_2 = 0.3
values_at_times = cq(np.array([time_1, time_2]))
vel_norm = min_jerk.deriv(1).dot(min_jerk.deriv(1))
plot(vel_norm)
```



$\|\dot{q}\|$

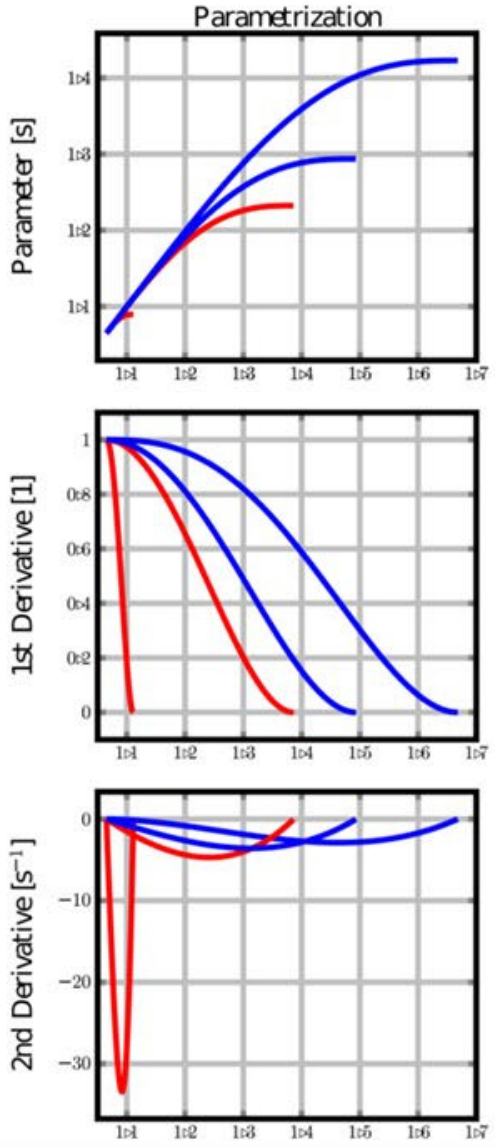


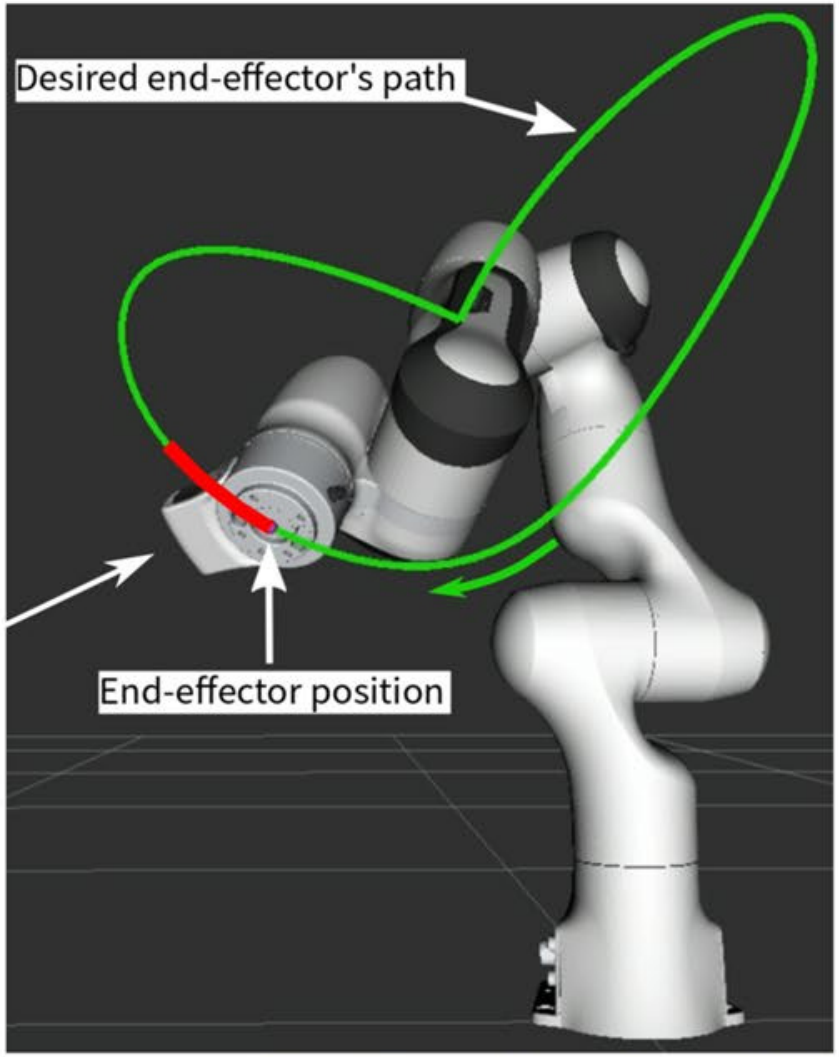
Optimization: Parametrization Generation.

Minimum time stop for collision avoidance

https://github.com/rafaelrojasmiliani/opstop_cpp

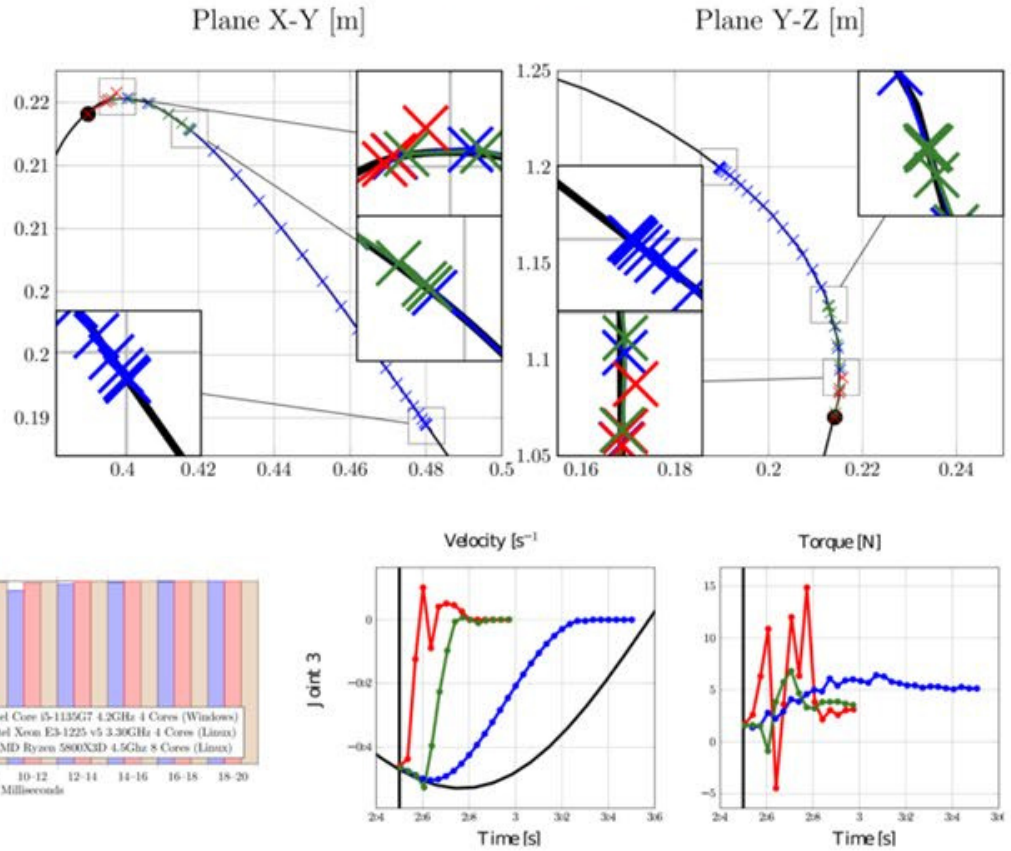
```
import gsplines
import gsplines.plot as gplot
import numpy as np
import opstop
from gsplines.optimization import minimum_jerk_path
model_file = 'path_to_urdf_robot_description'
dim = 7   # number of joints of the robot
# Generate a random numpy array of wayponts
# (each row is a waypoint in R^n)
number_of_waypoints = 5
waypoints = np.random.rand(number_of_waypoints, dim)
# The a minimum jerk trajectory with execution time of 5s
trj = minimum_jerk_path(waypoints)
# Select the time to stop as the 60% of the time.
stop_ti = trj.get_domain()[1]*0.6
# Get a parametrization that minimizes the time and
does not
# allow an increment in the acceleration larger than 50%
optimal_parametrization = \
    opstop.minimum_time_bounded_acceleration(
    trj, stop_ti, 1.5, str(model_file), 8)
# Obtain the stopping trajectory
stop_trj = trj.compose(optimal_parametrization)
# Plot the nominal and the stopping trajectory
gplot.plot_compare([stop_trj, trj], ['red', 'blue'], [
                'Emergency_Stop_Trajectory',
                'Original_Trajectory'],
                _show=True, _up_to_deriv=2)
```





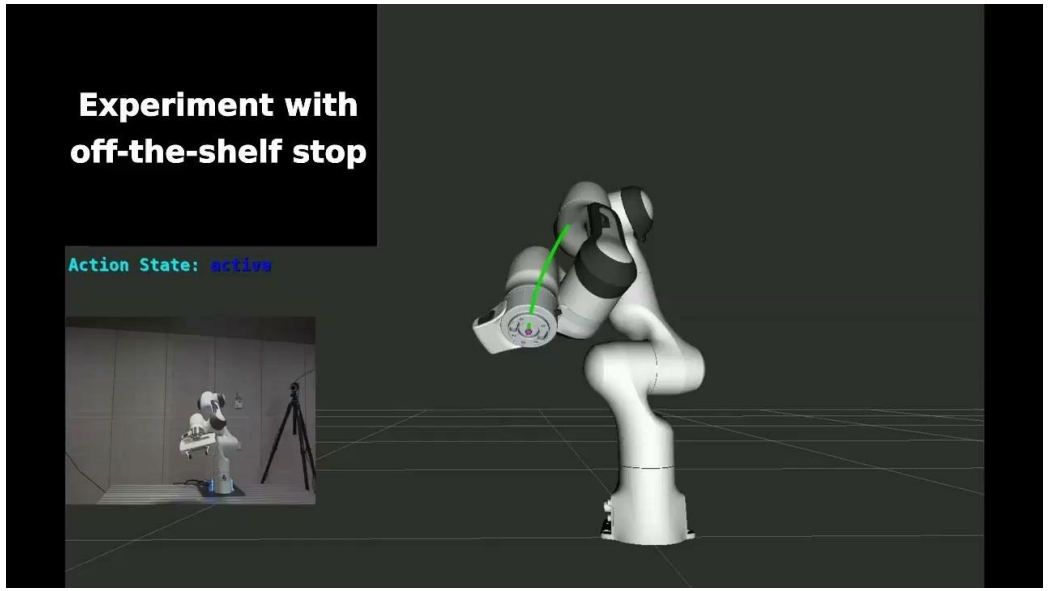
Optimization: Minimum Time Stop.

Comparison with ros_control default stop



Online Computation of Time-Optimization-Based, Smooth and Path-Consistent Stop Trajectories for Robots

Rafael A. Rojas [1,2,*], Andrea Giusti [2] and Renato Vidoni [1]

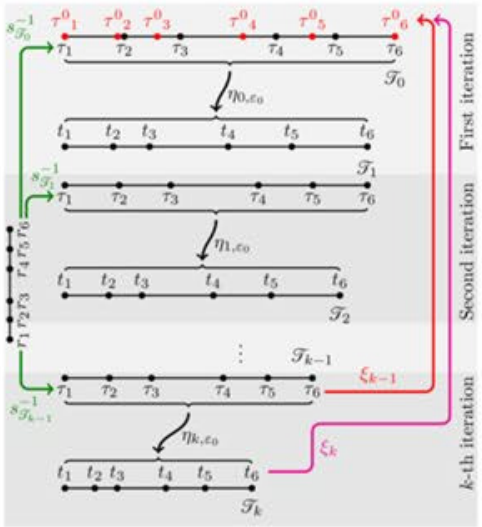


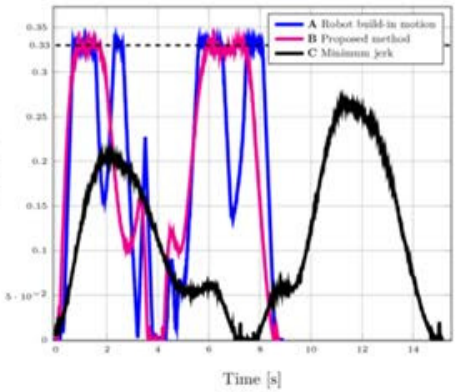
Optimization: Minimum Time parametrization.

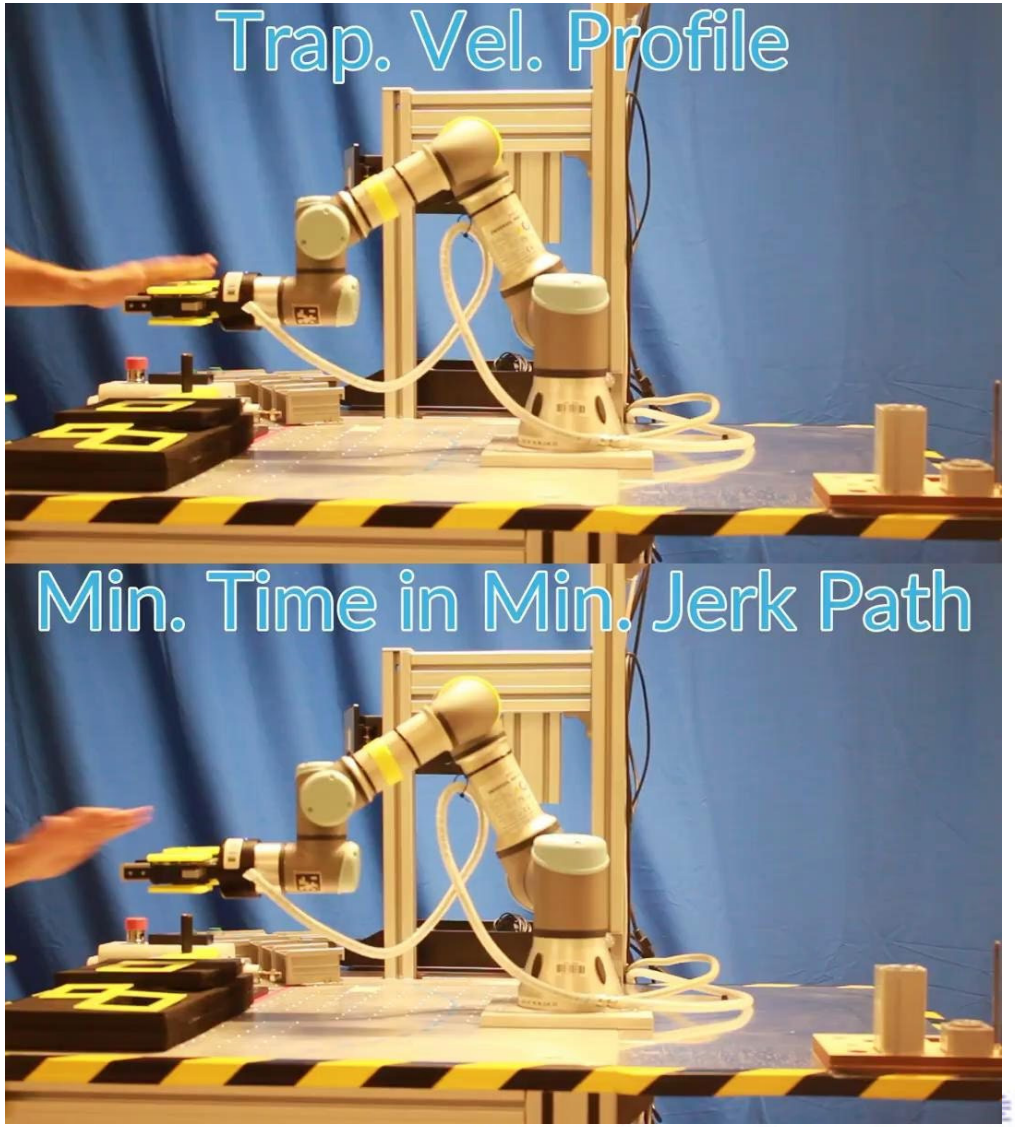
Fast and Smooth: Min. Time on a GSpline

Designing Fast and Smooth Trajectories in Collaborative Workstations

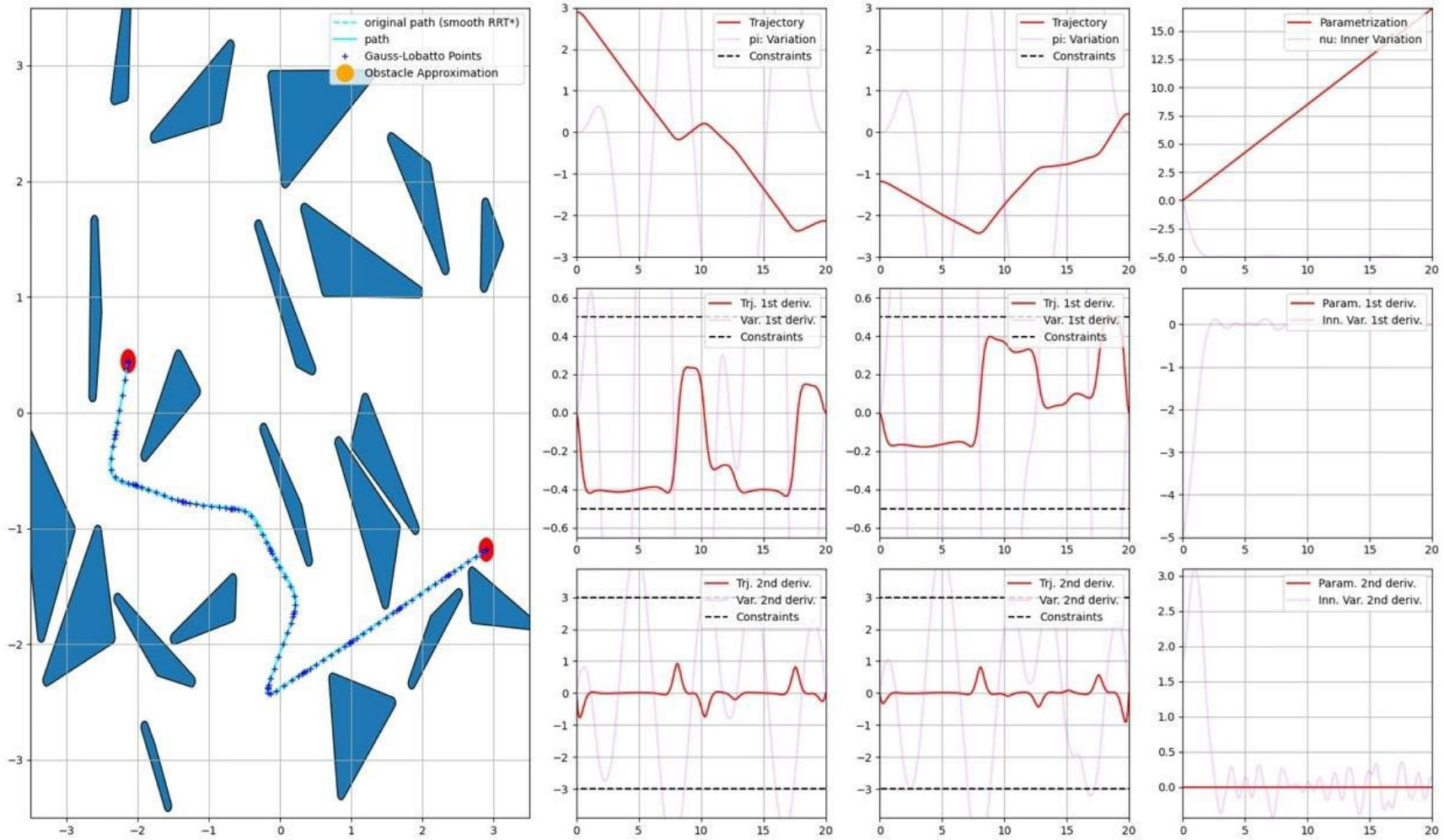
Rafael A. Rojas[1], Renato Vidoni[1]







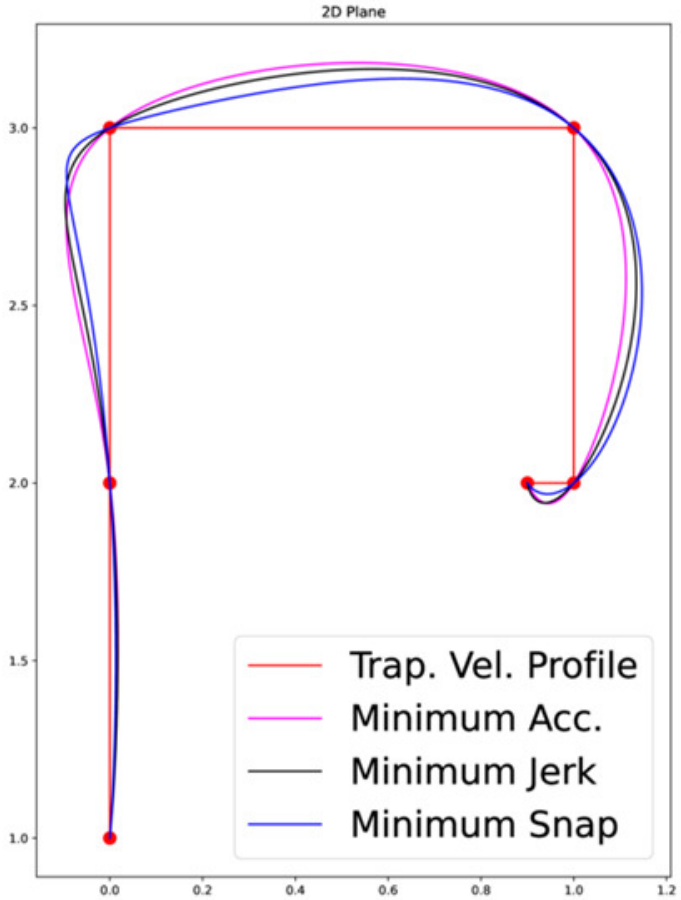
Optimization: Minimum Time Trajectory Planning.

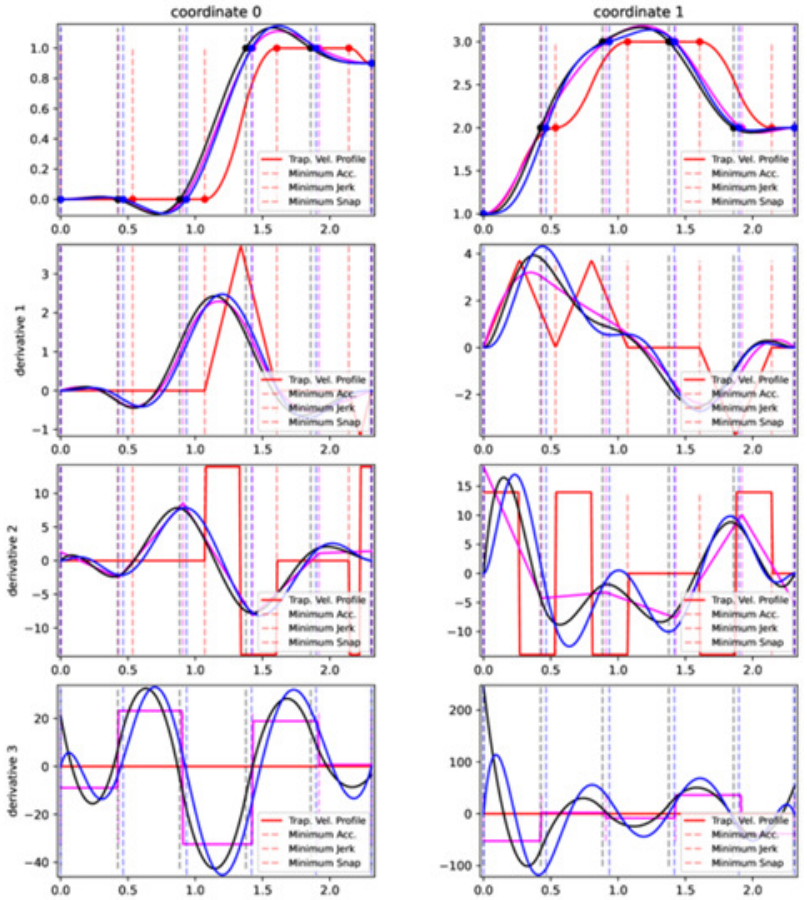


$$
\begin{aligned}
&\min \nu(T_k) \\
\text{s.t.} \quad &\mathscr{G}(\xi_k, \mathbf{p}_k) + \mathrm{d}^* \mathscr{F}'_{(\mathrm{Id}, \mathbf{q}_k)}(\nu, \boldsymbol{\pi}) \leq 0 \\
&\nu(0) = 0 \\
&\nu \in C^{k_c}([0, T_k], \mathbb{R}) \\
&\boldsymbol{\pi} \in C^{k_c}([0, T_k], \mathbb{R}^n),
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{p}_{k+1} &= \mathbf{p}_k + \varepsilon \boldsymbol{\pi}^\star \circ \eta_k \\
\xi_{k+1} &= \xi_k \circ \eta_\varepsilon^{\star -1}
\end{aligned}
$$

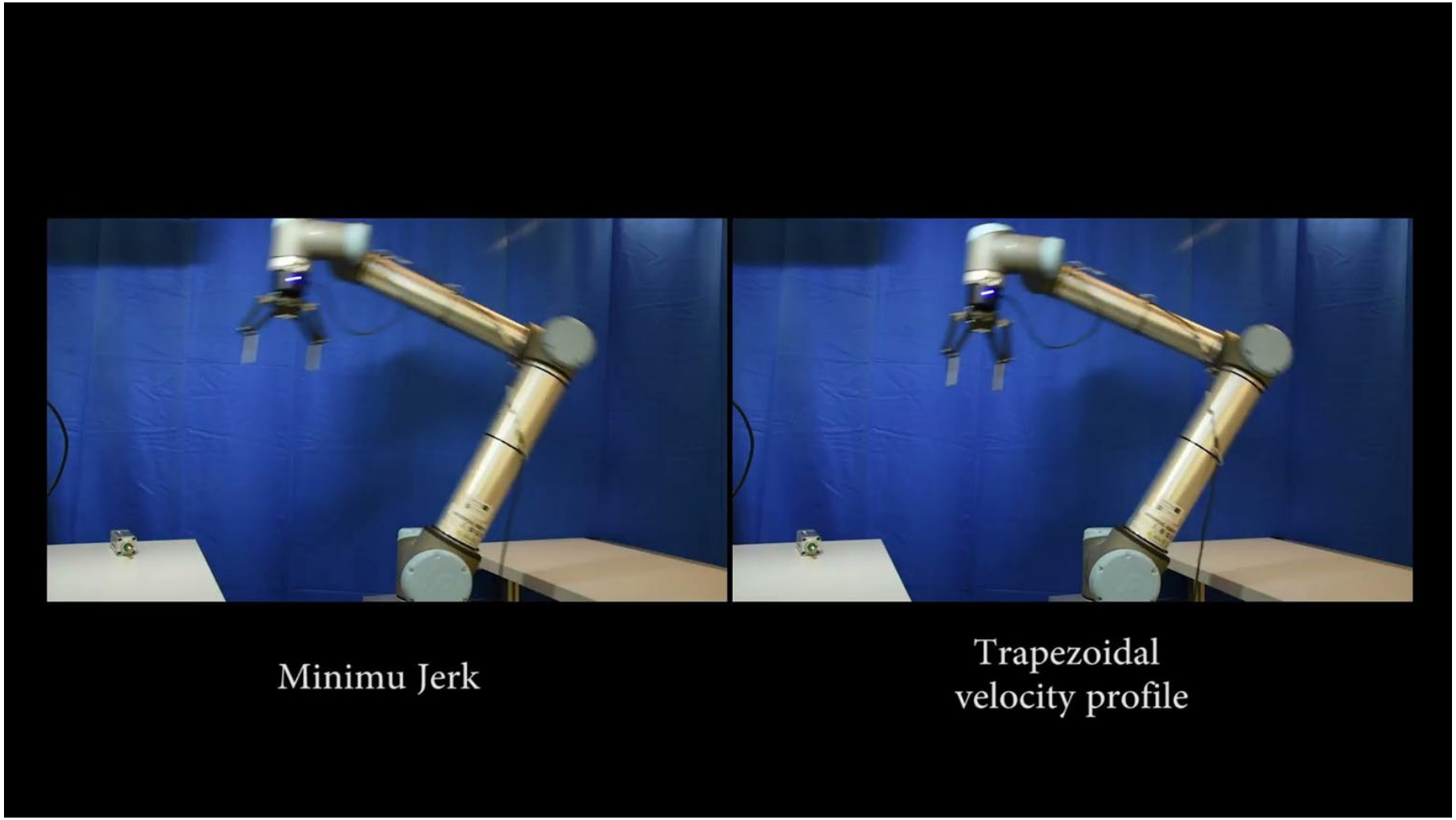
Comparison with KDL Symmetric Velocity profiles





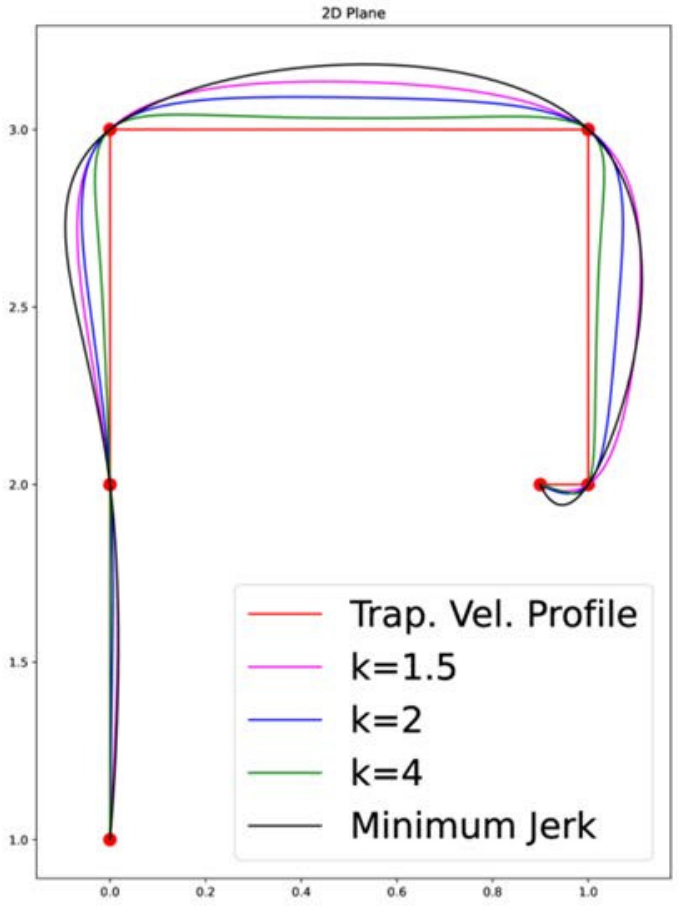
Remark: Minimum-X trajectories present large overshoots. As the arc-lenght is larger, this trajectores are slower

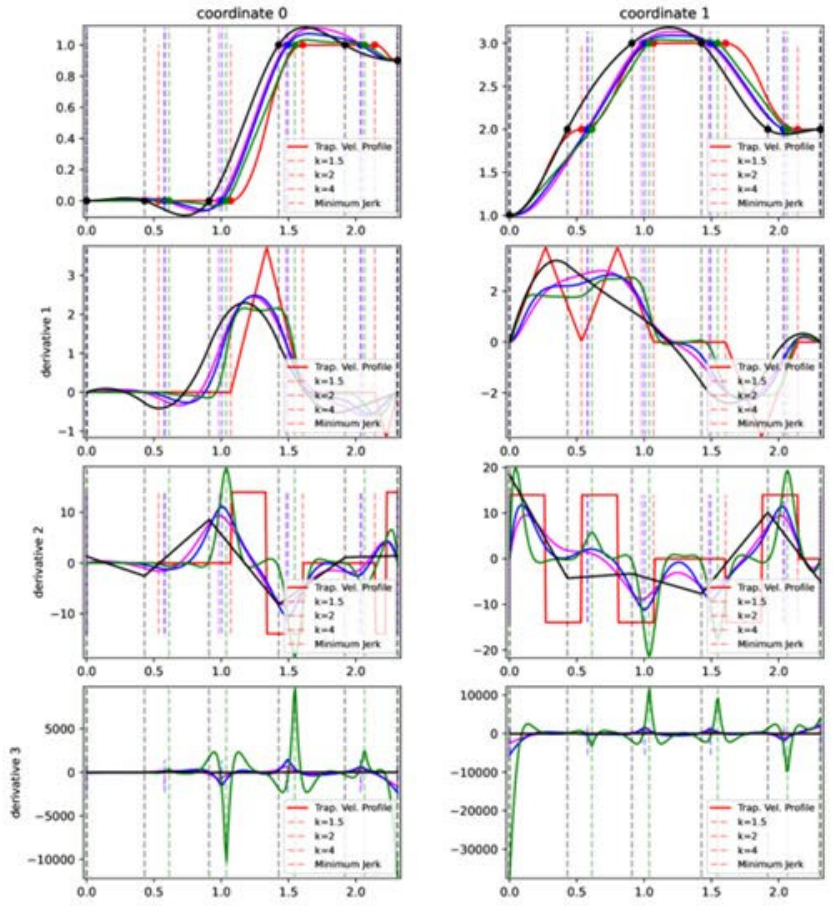
Comparison with KDL Symmetric Velocity profiles



Comparison with KDL Symmetric Velocity profiles

Overshoot reductions





$$\min \int_{t_0}^{t_f} \alpha \left\| \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} \right\|^2 + (\alpha - 1) \left\| \frac{\mathrm{d}^3\mathbf{q}}{\mathrm{d}t^3} \right\|^2 \mathrm{d}t$$

Combining safety and speed in collaborative assembly systems – An approach to time optimal trajectories for collaborative robots

Rafael A. Rojas*, Manuel A. Ruiz Garcia, Luca Gualtieri, Erwin Rauch

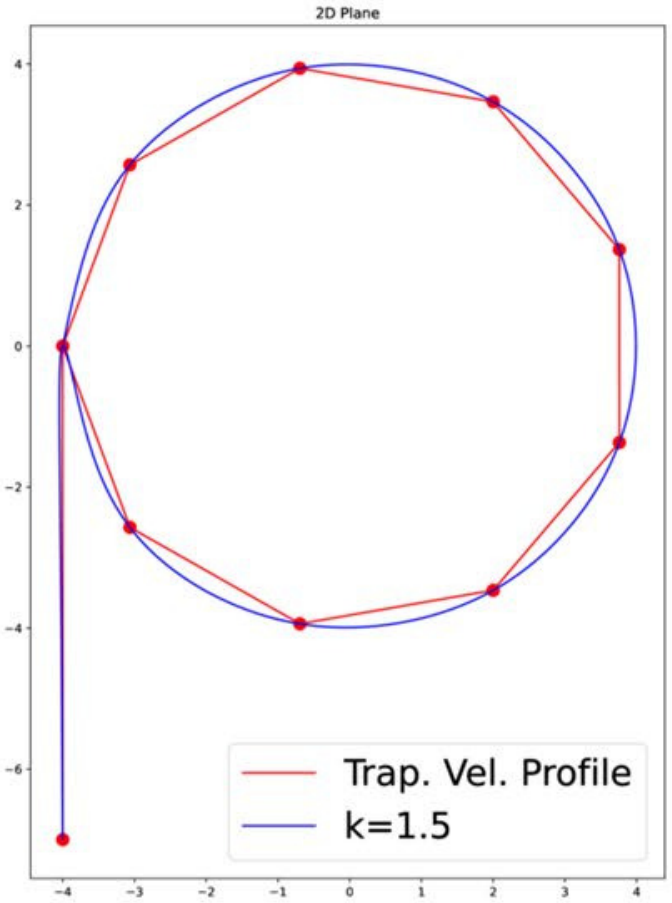
Free University of Bozen-Bolzano, Faculty of Science and Technology, Piazza Università 1, Bozen-Bolzano 39100, Italy

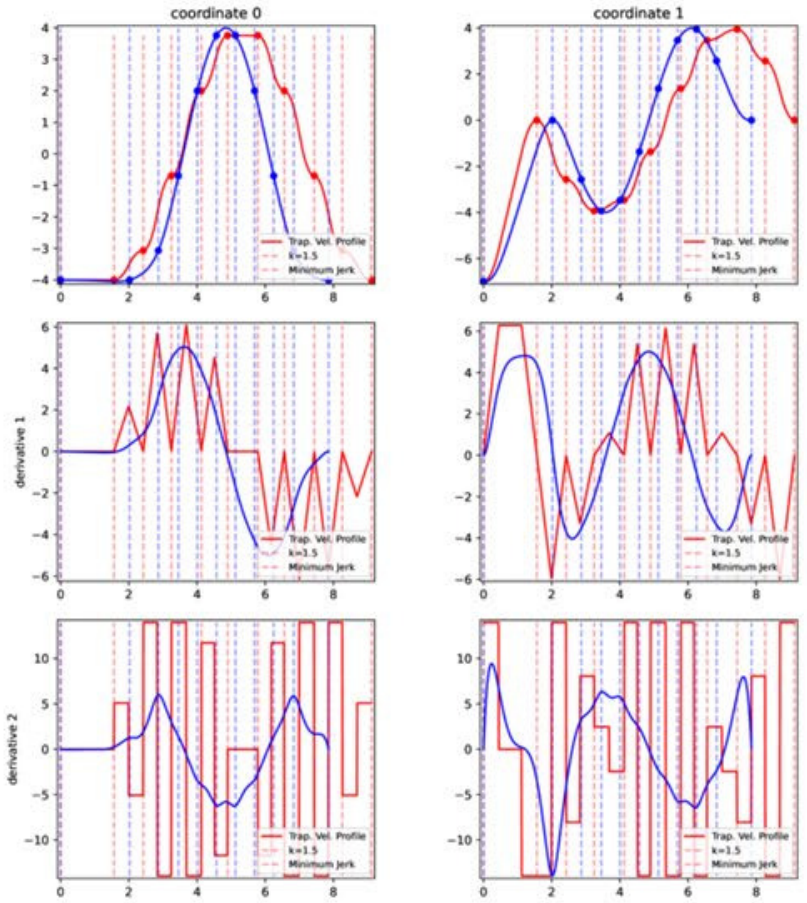
Path shape parameter:

$$k = T \frac{\sqrt{2}}{4N} \sqrt[4]{\frac{\alpha}{1-\alpha}}.$$

Comparison with KDL Symmetric Velocity profiles

Speed comparison





$$\min \int_{t_0}^{t_f} \alpha \left\| \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} \right\|^2 + (\alpha - 1) \left\| \frac{\mathrm{d}^3\mathbf{q}}{\mathrm{d}t^3} \right\|^2 \mathrm{d}t$$

- 14% Faster
- No blend radius

$$\sup_{t \in [0,T]} \|\mathbf{e}(t)\|_1 \leq T \sqrt{\int_0^T \left\| \dot{\mathbf{q}}_1(s) - \dot{\mathbf{q}}_\alpha(s) \right\|^2 \mathrm{d}s}.$$

Integration with ROS

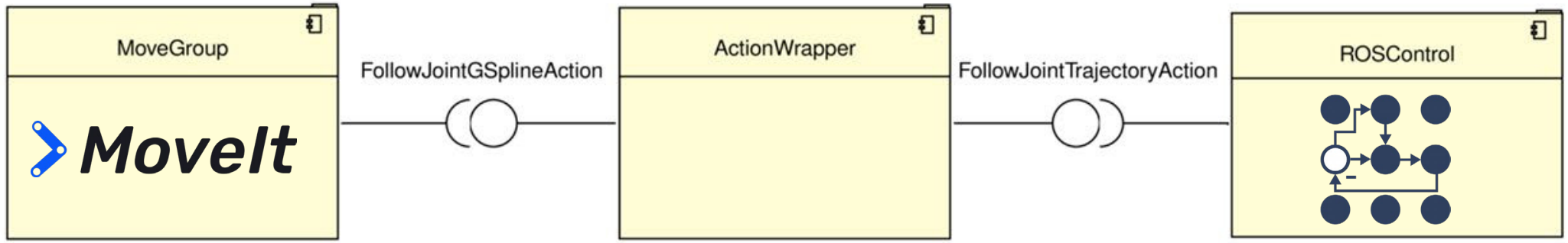
- Custom **messages** for `GSpline` objects: Trajectories, Basis.
- **Conversions** from `GSpline` into `trajectory_msgs::JointTrajectory` and `control_msgs::FollowJointTrajectoryGoal`
- Custom rqt plot tool with `rqt` plugin based on the amazing `rqt_joint_trajectory_plot`
- Moveit Planner adapter `gsplines_moveit/MinimumSobolevSeminormAdapter`, can be used inside `ompl` pipeline

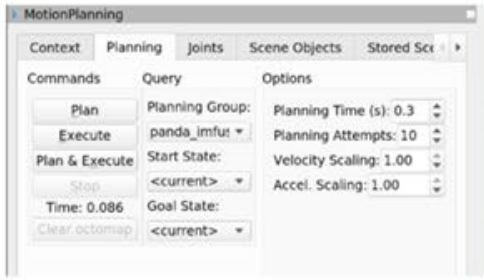
```
gsplines_moveit/MinimumSobolevSeminormAdapter
default_planner_request_adapters/FixWorkspaceBounds
default_planner_request_adapters/FixStartStateBounds
default_planner_request_adapters/FixStartStateCollision
default_planner_request_adapters/FixStartStatePathConstraints
```

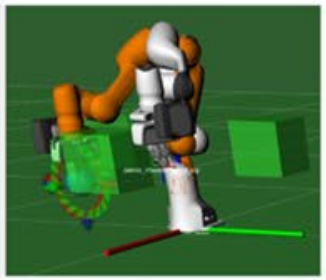
- Controller Manager `gsplines_moveit/GSplinesControllerManager`

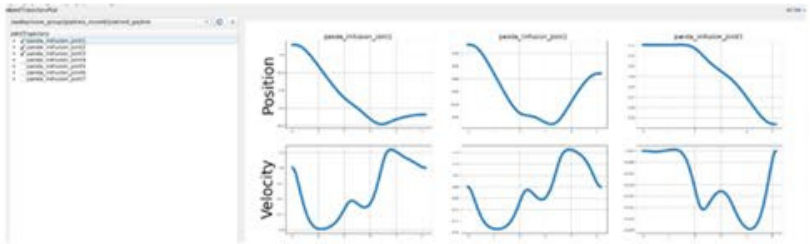
Moveit Integration

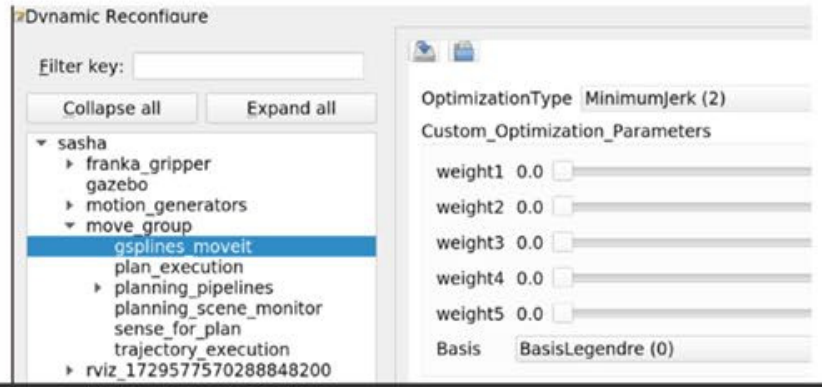
Architecture





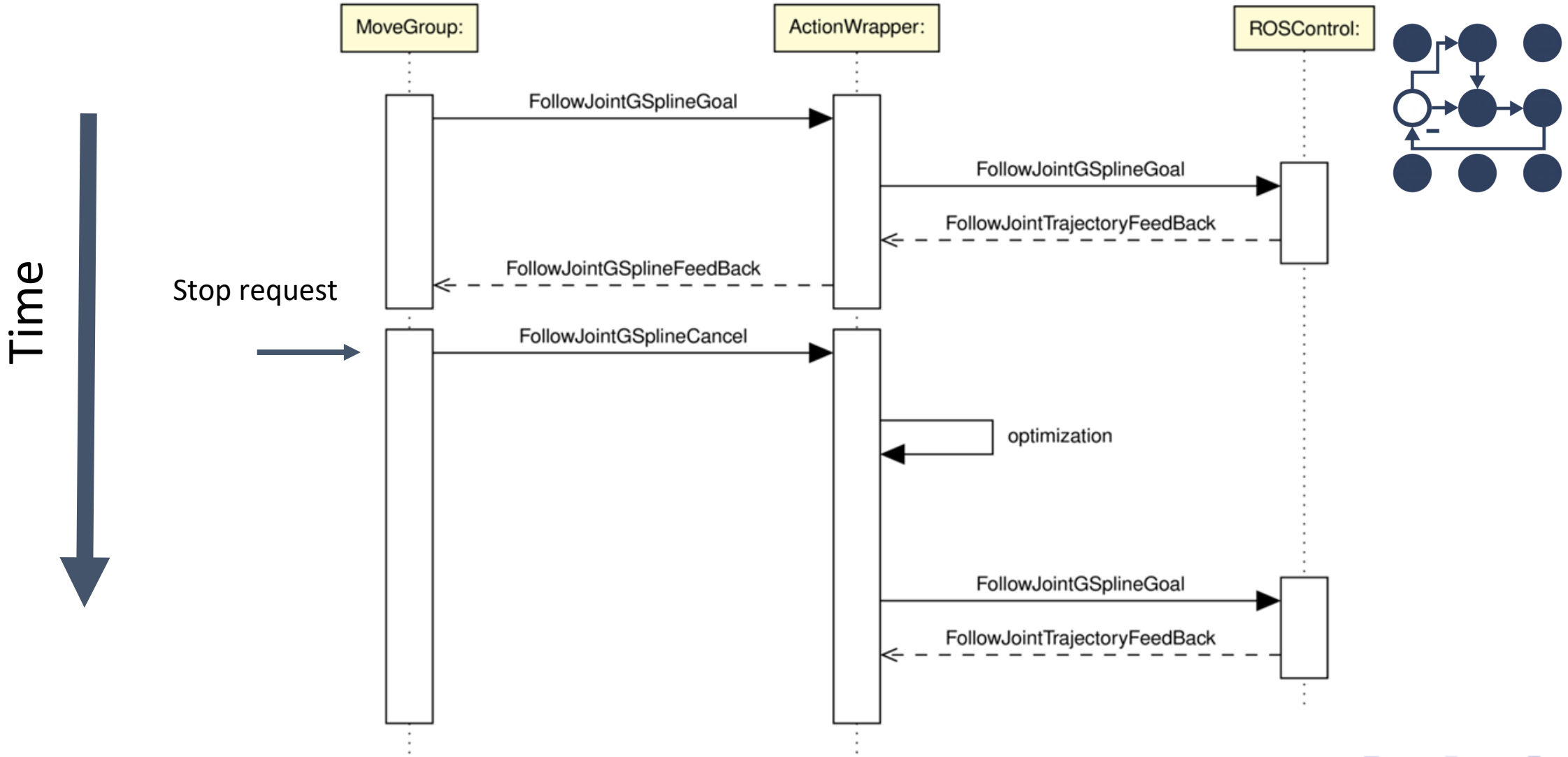






OpStop Integration

Opstop



SASHA-OR project

Situation Aware Sterile Handling Arm for the OR

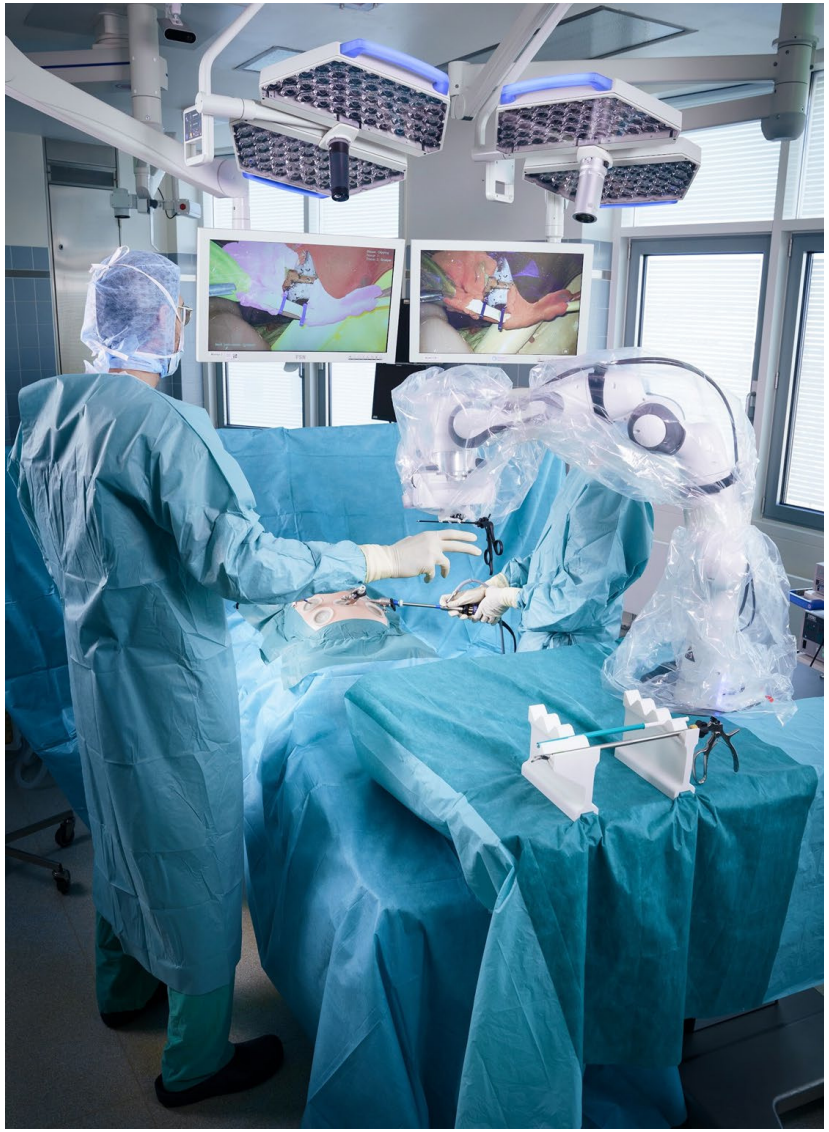
- **Robotic scrub nurse**
- Handling laparoscopic surgical instruments on demand

SASHA-OR project

Situation Aware Sterile Handling Arm for the OR

We are interested

- Evaluate cognitive ergonomics
  - Cognitive workload
  - Human perception of the motion
  - Human attention management
  - Trust and reliance
  - Motion predicability



Future work

- Implementation of trajectories on $SE(3)$ (Cartesian, Position + Orientation)
- Monadic interface to build generic cost/constraints function for optimization on manifolds. Almost ready in ImFusion motion planning API.
- Minimum Time `GSpline` parametrization: already published but must be cleaned WIP.
- Implementing Basis for

$$\min \int_{t_0}^{t_f} \alpha_1 \left\| \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} \right\|^2 + \alpha_2 \left\| \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}t^2} \right\|^2 + \alpha_3 \left\| \frac{\mathrm{d}^3\mathbf{q}}{\mathrm{d}t^3} \right\|^2 \mathrm{d}t$$

Conclusions

- We have demystified Minimum-X trajectories: They may be inefficient
- However we can design weight-cost functions to improve these trajectories
- A `GSpline` may be used to build custom gradient base optimization methods
- The `GSplines` library provides out of the box ROS-ready
  - Trajectory generation
  - Minimum time stop
  - General expressions constructions
  - Plot tool

ImFusion

Thank You!