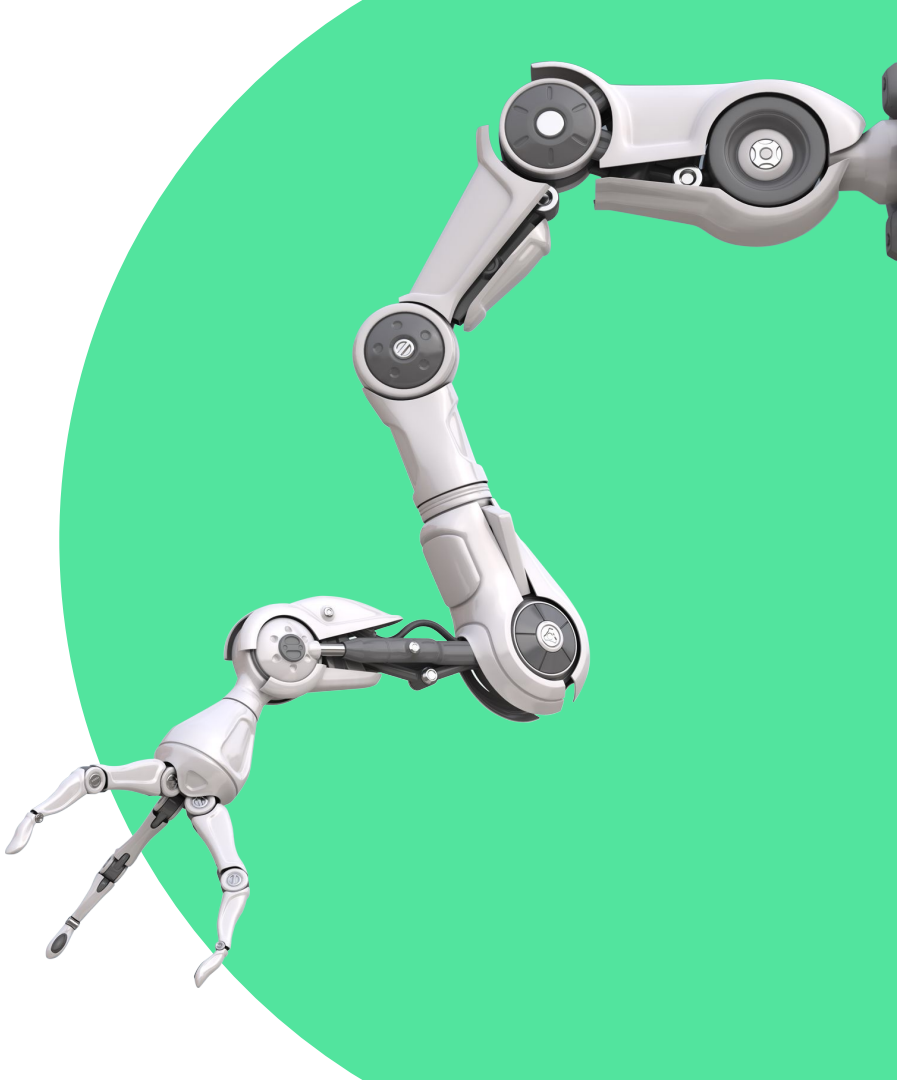
MoveIt for ROS 2

ROS 2 TSC - December 2020



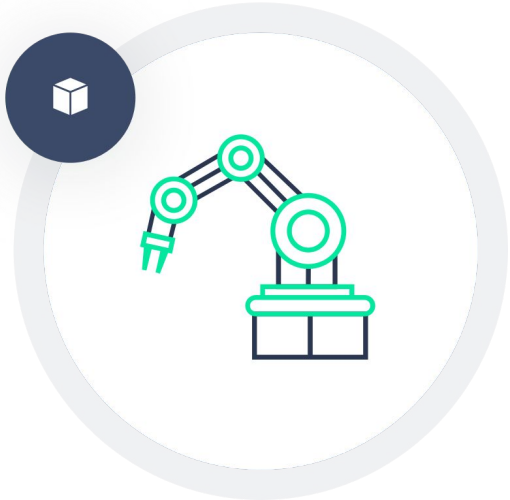
PICKNIK

Henning Kayser, MS
PickNik Robotics
henningkayser

Outline

PICKNIK

- Roadmap Status Update
- Hybrid Planning
- Migration Challenges
- ROS 2 Learnings
- Future Plans



PICKNIK

Roadmap Status Update

MoveIt2

Milestone 1	Milestone 2	Milestone 3
Straight Port to ROS 2	**Realtime Support**	**Fully Leverage ROS 2**
Fully migrate existing MoveIt packages to ROS 2	Reactive, closed-loop control to sensor input	Lifecycle management of MoveIt nodes
Wrap up Acutronic's work porting core MoveIt functionality	Visual servoing, octomap updates	Deterministic startup, reset, & shutdown sequences
Leverage ROS 2:	Preempt motion if new collision detected	Leverage ROS2 component nodes
Build system (ament), middleware, logging, parameters	Seperate global and local planner (hybrid planning)	Ability to run MoveIt as single or multi-process
Cleanup MoveIt 2 codebase	Global planner (full collision checking): 30hz	Replace pluginlib with components
	Local planner (IK-based, field-based): 300hz	Cleanup API
	Zero-memory copy integration to controllers (ros_control)	More generic and standalone interfaces
	Tighter integration to ros_control	
	Integrate pilz_industrial_motion	

Milestone 1 - Full Migration

PICKNIK

“Straight Port to ROS 2”

Progress: 94% - 58 of 62 targets ported

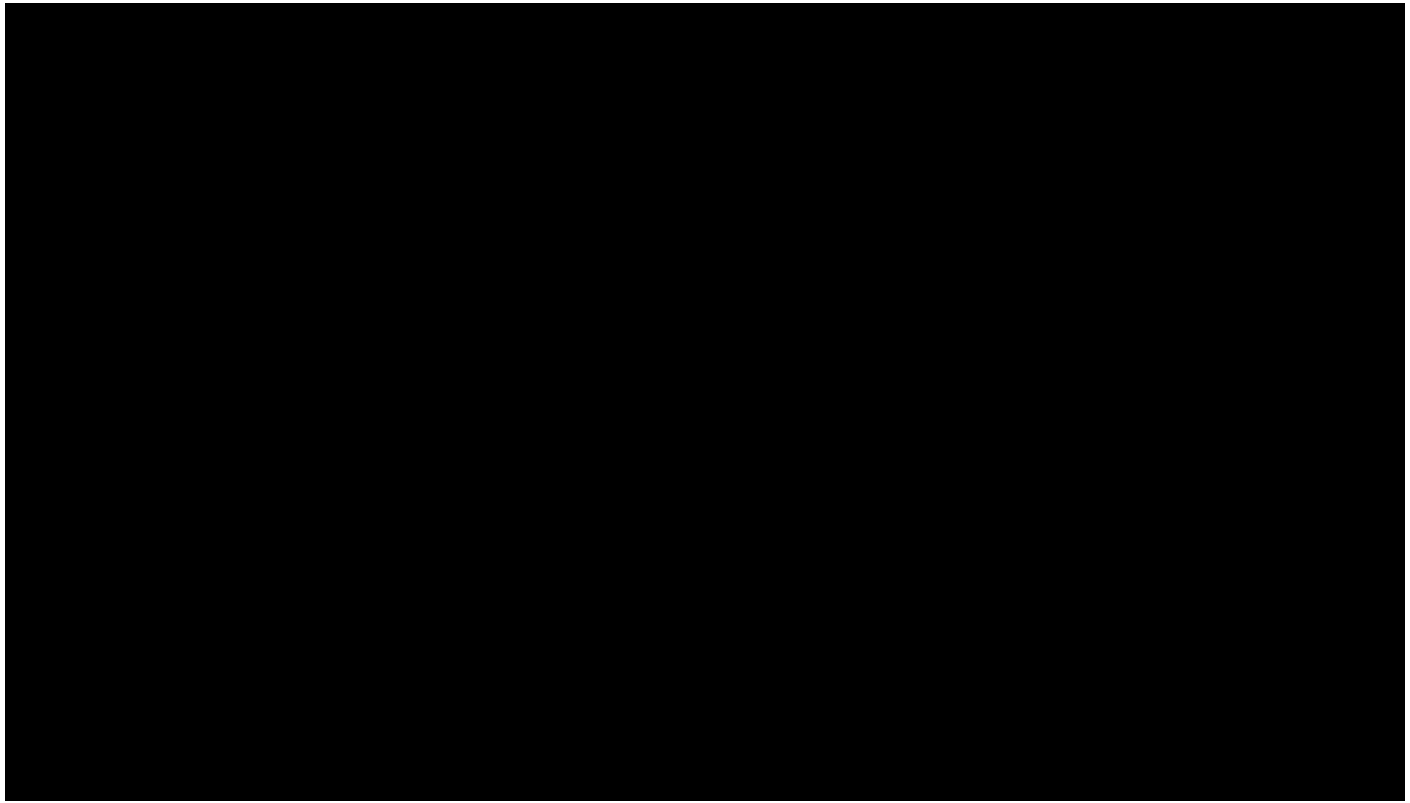
Incomplete: Bullet, TrajOpt, MoveIt Setup Assistant, Perception

Demos: MoveGroup, MoveItCpp, MoveIt Servo

Milestone 1 - Full Migration - MoveGroup

PICKNIK

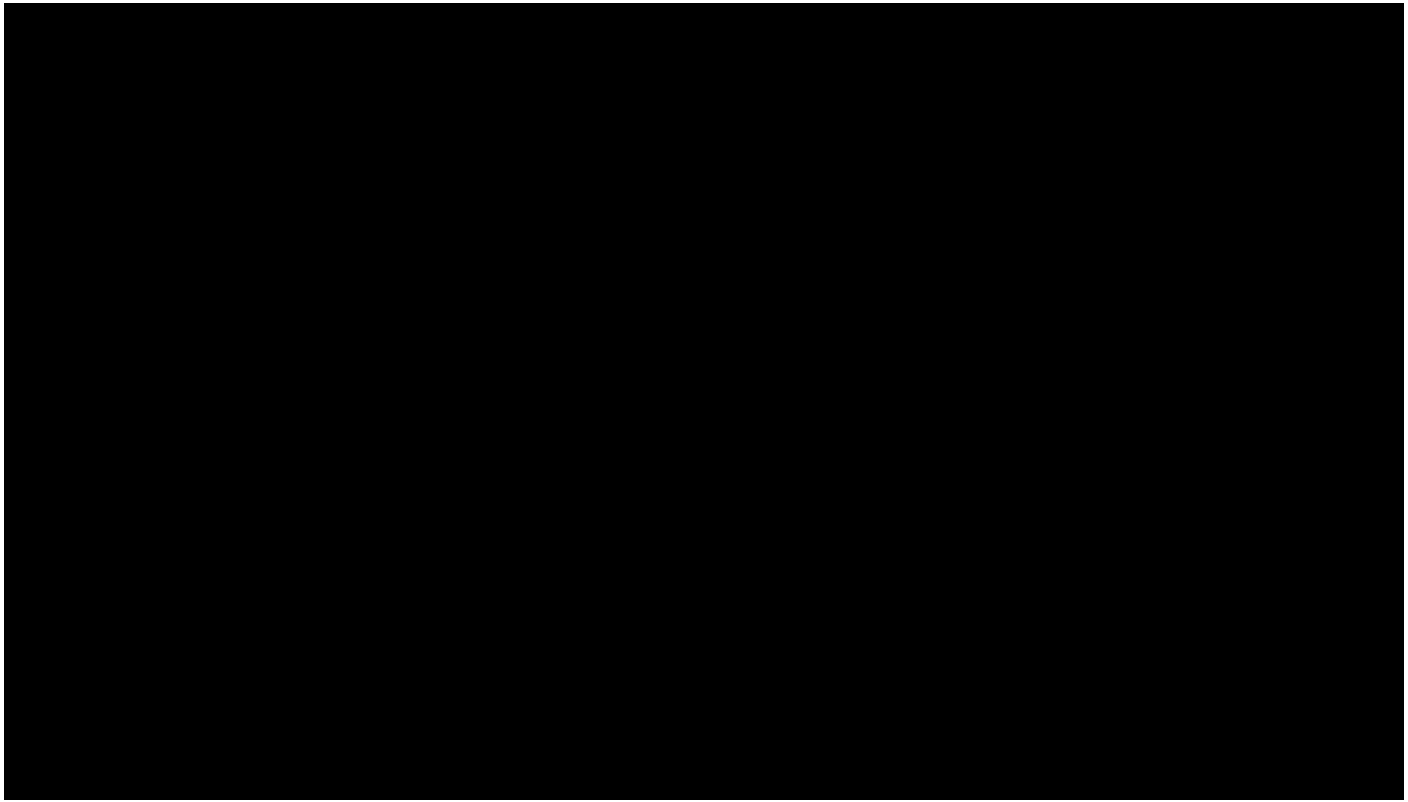
Demo URL: https://github.com/ros-planning/moveit2/tree/main/moveit_demo_nodes/run_move_group



Milestone 1 - Full Migration - MoveItCpp

PICKNIK

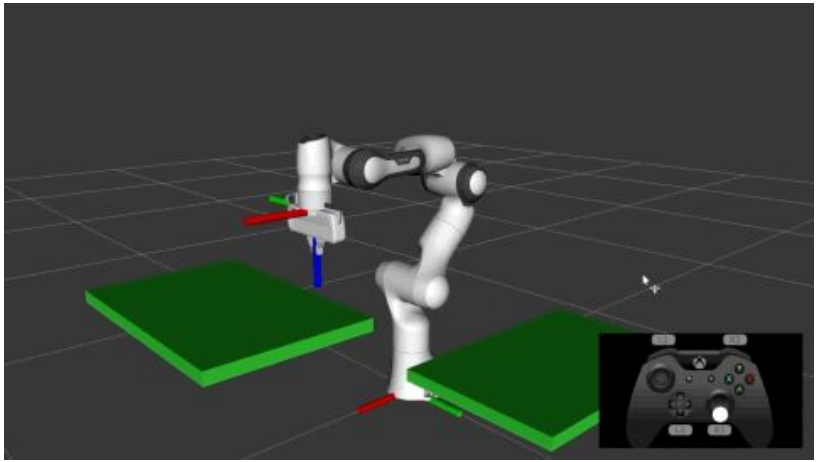
Demo URL: https://github.com/ros-planning/moveit2/tree/main/moveit_demo_nodes/run_moveit_cpp



MoveIt Servo

PICKNIK

- Joint/Velocity-streaming controller, inverse Jacobian method
- Input message allows wide range of input devices
- Checks for joint limits, collision, singularity safety



PICKNIK

Hybrid Planning

PICKNIK

2. Separate Global/Local Planner (Hybrid Planning)

R&D Student Intern: **Sebastian Jahr**

(Karlsruhe Institute of Technology)

Project Status

- Initial research completed
- Working on architecture design
- Selecting & Testing Planner Candidates
- Completion planned for end of **January 2021**

Realtime Support
Reactive, closed-loop control to sensor input
Visual servoing, faster octomap updates
Preempt motion if new collision detected
Seperate global and local planner (hybrid planning)
Global planner (full collision checking): ~10hz
Local planner (IK-based, field-based): ~300hz
Zero-memory copy integration to controllers (ros_control)
Tighter integration to ros_control
Integrate pilz_industrial_motion

ROSin

Current approach: Sense-Plan-Act

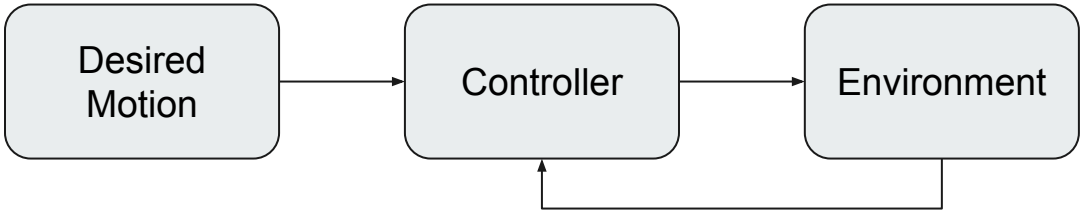
Strength

- Can find a global solution in workspaces with complex geometries

Weakness

- Copes bad with uncertainties and (fast) changing environment

Local reactive control



Strength

- Reacts immediately to changes in the environment

Weakness

- Gets stuck in local minimum

Hybrid Planning

PICKNIK

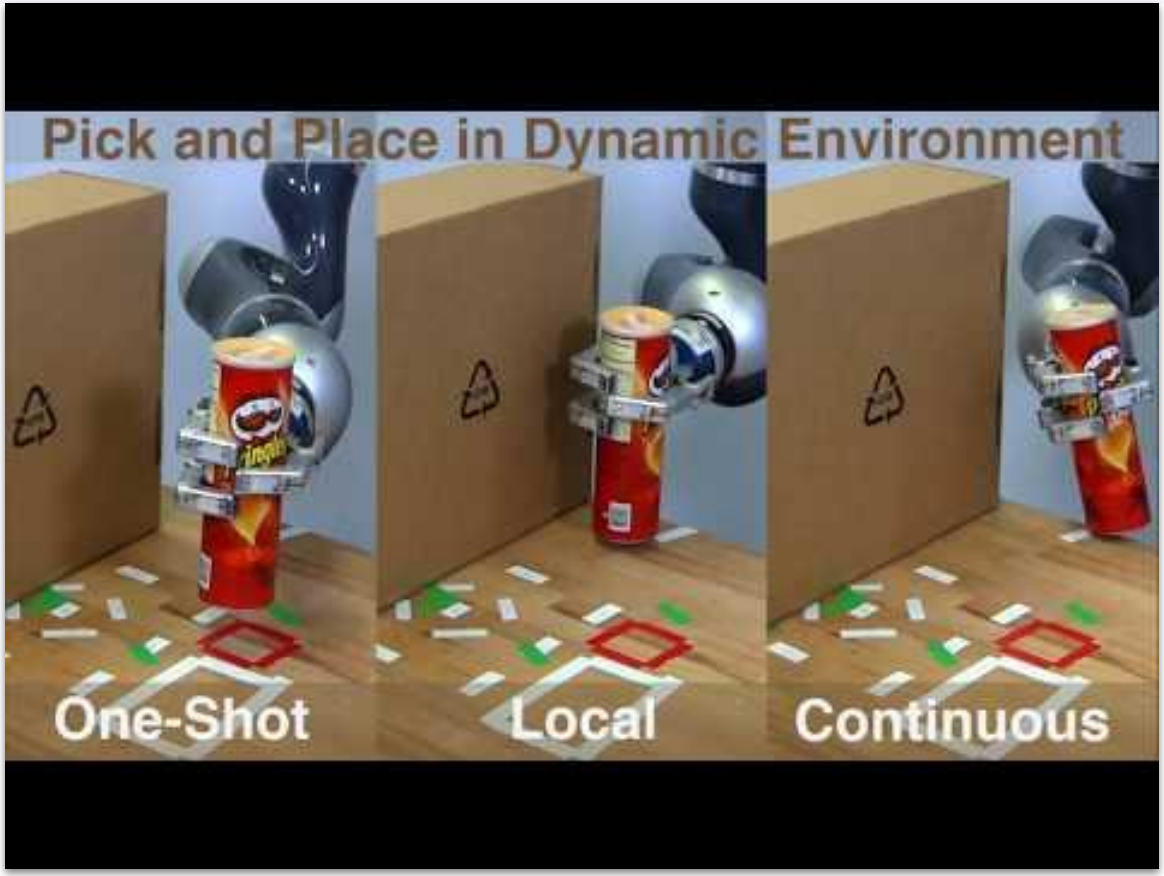
Goal: Execute adaptive and reactive motions using global/local planning

Adaptive Motion - *Drawing on a chalkboard*

- Global planner defines the motion required for drawing the letters
- Local planner follows motion while controlling for force, smoothness, etc..

Reactive Motion - *Steering around a new collision object in the scene*

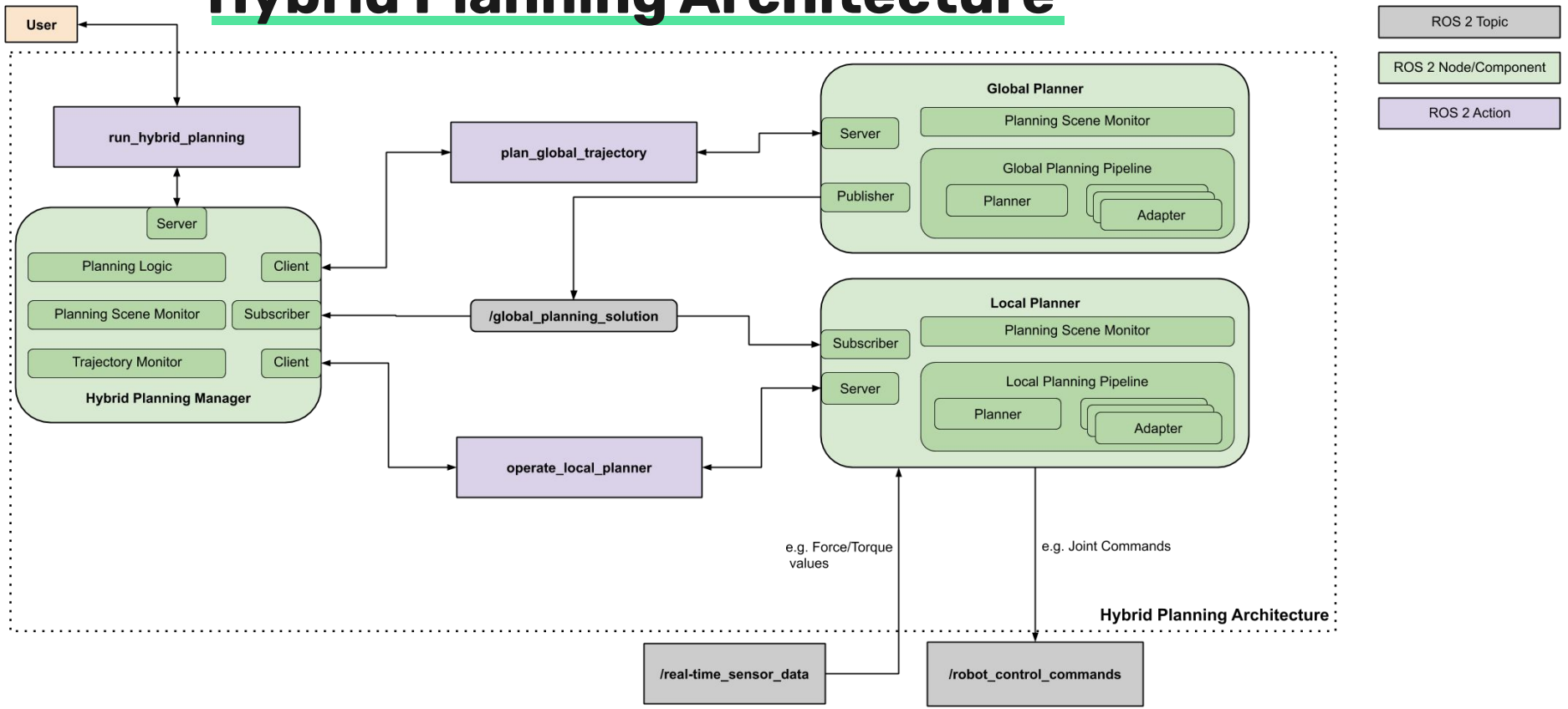
- Global planner used for fixing invalidated trajectories
- Local planner allows “keeping clear” from objects using field-based distance minimization



From: https://am.is.tuebingen.mpg.de/publications/2017_rss_system

PICKNIK

Hybrid Planning Architecture



Message Performance Requirements

PICKNIK

Communication use cases:

- Controller commands - High frequency, low bandwith, realtime-safe
- Point cloud data - High message size, possibly zero-copy comm
- Scene updates - Synchronized processing vs. shared access
- Plan action feedback - Possibly high-frequent events

We’re still looking into exact specs and bottlenecks before optimizing DDS.

PICKNIK

Migration Challenges

ROS 2 - Migration Challenges - **CallbackQueue**

PICKNIK

Use case: PlanningSceneMonitor provides safe concurrent read/write access to unique planning scene via ROS service and topic interfaces

ROS 1

- Requests are handled in dedicated CallbackQueue and processed in a serialized event loop
- CallbackQueue is managed by separate NodeHandle and AsyncSpinner

ROS 2

- AsyncSpinner is replaced with SingleThreadedExecutor
- NodeHandle is replaced with private node instance at subnamespace “*_private”

Possible improvements:

- Implement CallbackQueue behavior similar to the CallbackGroup API (https://github.com/ros2/rclcpp/issues/1287)

ROSin

ROS 2 - Migration Challenges - **XmlRpc**

PICKNIK

Use case: Custom OMPL planner identifiers are mapped to algorithm config in yaml file

ROS 1

- Parameters can be structured as generic dictionary map with unknown keys
- Parameter groups can be parsed to arbitrary structs

ROS 2

- All parameters should be declared in advance, unknown parameters are discouraged
- Workaround:
  - Define separate parameter key list and declare unknown params at runtime
  - Specify explicit struct types in config instead of using implicit conventions

ROS 2 - Migration Challenges - **DynamicReconfigure**

PICKNIK

Use case: Update specific node parameters from remote interfaces

ROS 1

- Classes can register a predefined config for updating the internal parameter state
- Remote nodes can update and apply new parameters at runtime

ROS 2

- Config changes are handled using the parameter callback API
- Parameter types need to be filtered, validated and applied for each class instance

ROSin

ROS 2 - Migration Challenges - **Launch & Config structure**

PICKNIK

Use case: MoveIt setup packages generated by the MoveIt Setup Assistant templates

ROS 1

- Launch structure is composed of nested XML files for enabling specific MoveIt components
- Each MoveIt component has it's own XML file that loads a set of rosaparams from a YAML
- Many parameters are global and used by different nodes, i.e. "robot_description"

ROS 2

- Nodes are configured as LaunchDescription instances in a single python file
- Parameters are read from YAML files and passed to nodes where required
- Launch files are not composable (yet)
- Still a lot of redundancy in MoveIt's launch files

PICKNIK

ROS 2 Learnings

PICKNIK

Overall great focus on code quality standards and best practices

- Improved and concise build tools: colcon, ament, vcstool
- Outstanding linter support:
  - ament_lint_[cmake, cpplint, copyright, clang_format, pep ...]
  - Good linter defaults useful for package standardization
  - Perfectly set up for extensibility
- Very clean Modern C++ implementations and features:
  - Consistent support for handling async calls with std::future
  - Flexible callback types enabling simple lambda implementations
  - Parameter templates provide control over declared value types and definitions

ROS 2 - Pain Points

PICKNIK

Still an early-stage framework that lacks the long term usage from ROS 1

- Lack of documentation and examples for advanced usage
  - Unclear behavior needs to be looked up in code
  - Missing best practices for Python launch files and parameter loading
- Several breaking API changes in upstream dependencies (probably less with future releases)
- Some callback function types don't work with clang-tidy
- No apparent consensus of proper time source usage in standard packages
- Several features are WIP and require workarounds

PICKNIK

Future Plans

Future Plan

PICKNIK

Releases

- New Foxy Debian Release every 6 weeks **now**
  - Frequent syncs with MoveIt 1
- Switching main branch to Rolling Ridley **Q1 2021**
  - Foxy continued in release branch
- Windows & OSX support - CI enabled **Q1 2021**
- Galactic release **Q3 2021**

Upcoming Work

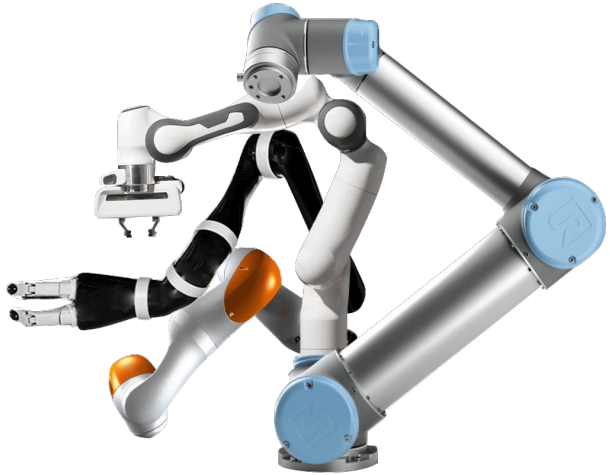
- Port MoveIt ecosystem: MTC, moveit_calibration, …
- Redesign MoveIt config, launch files, setup assistant
- New long term feature roadmap in Q1 2021

Hardware Integration Challenges

PICKNIK

“Chicken and Egg” Problem:

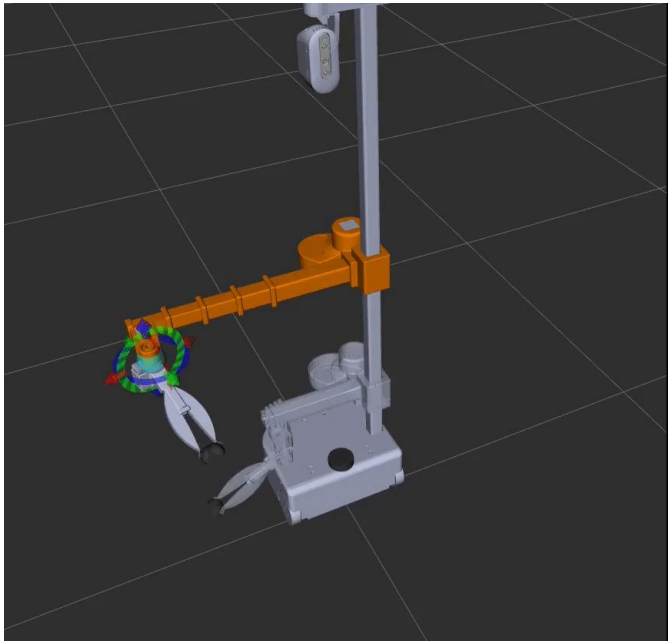
- ROS 2 user adoption is driven by hardware support
- Broad hardware support requires user adoption



PickNik is working on multiple hardware integration efforts...

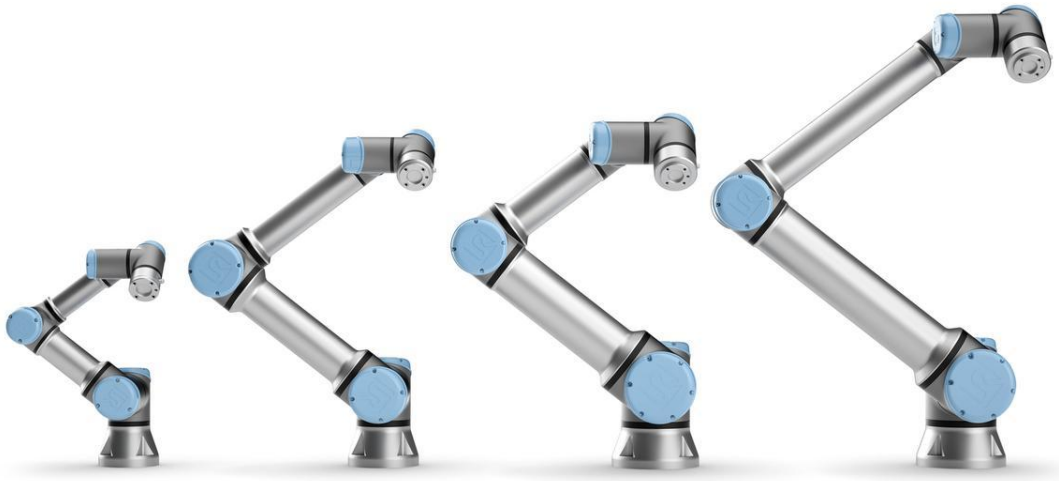
PICKNIK

Hello Robot - “Stretch”



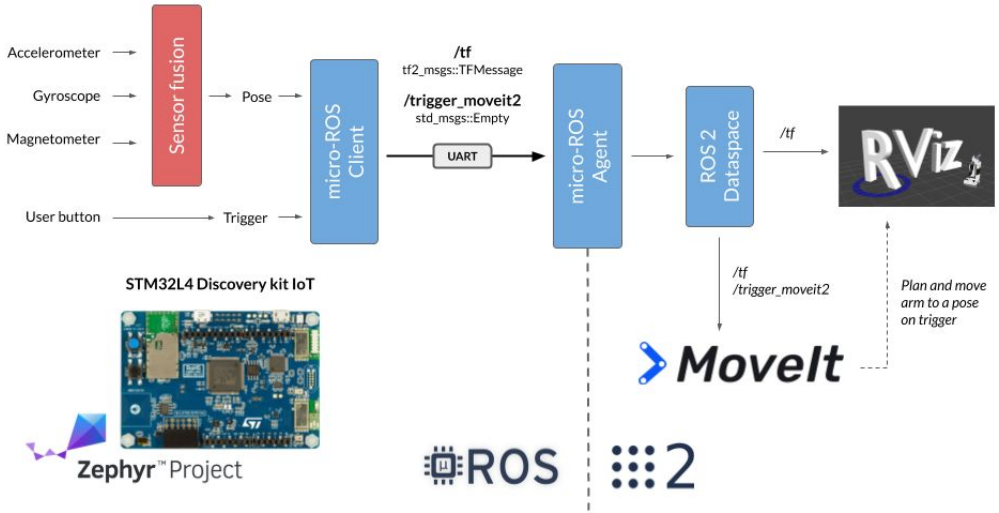
PICKNIK

Universal Robots - ROS 2 driver



eProsima - micro-ROS sensor integration

PICKNIK



See: https://discourse.ros.org/t/micro-ros-meets-moveit/16836

PICKNIK

Get Involved

https://github.com/ros-planning/moveit2

Many approaches:

- Adding New Features
- Helping with MoveIt 2 Roadmap
- Financial contributions via code sprints and grants
- Enhancing Documentation
- Reporting & Fixing Bugs

MoveIt2

PICKNIK

Thanks!