

# Movelt Overview

**Mark Moll, PhD**

Director of Research

PickNik Robotics

 mamoll



- **Movelt Background**
- **Movelt Applications: Land, Sea, Space**
- **Typical Usage Patterns**
- **Components**
- **More Resources**



# The Dream: A Multi Purpose Robot

Powered by MoveIt and ROS



# Movelt: A Hardened Motion Planning Platform

arm\_navigation



Version 1.0  
2010

Movelt!



Beta  
2013

Movelt!



Version 1.0  
2019

Movelt2



Version 2.0  
2020

Movelt3?



Version 3.0  
2022?

## Global Planners

- OMPL
- SBPL
- TrajOpt
- STOMP
- CHOMP

## Cartesian Planners

- RobotState
- Descartes
- JogArm
- PilzMotion

## Inverse Kinematics

- KDL
- IKFast
- TracIK
- LMA
- BioIK

## Grasping Libraries

- MoveIt Grasps
- Grasp Pose Detection (GPD)
- Intel OpenVino GPD

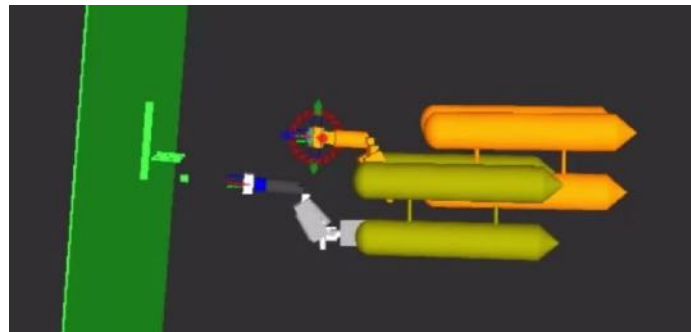
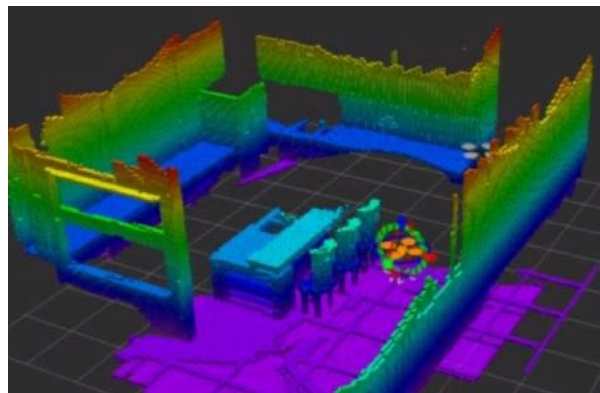
## Collision Checking

- Fast Collision Library (FCL)
- Bullet

## Perception / Octomap

- Depth Images
- Point Clouds

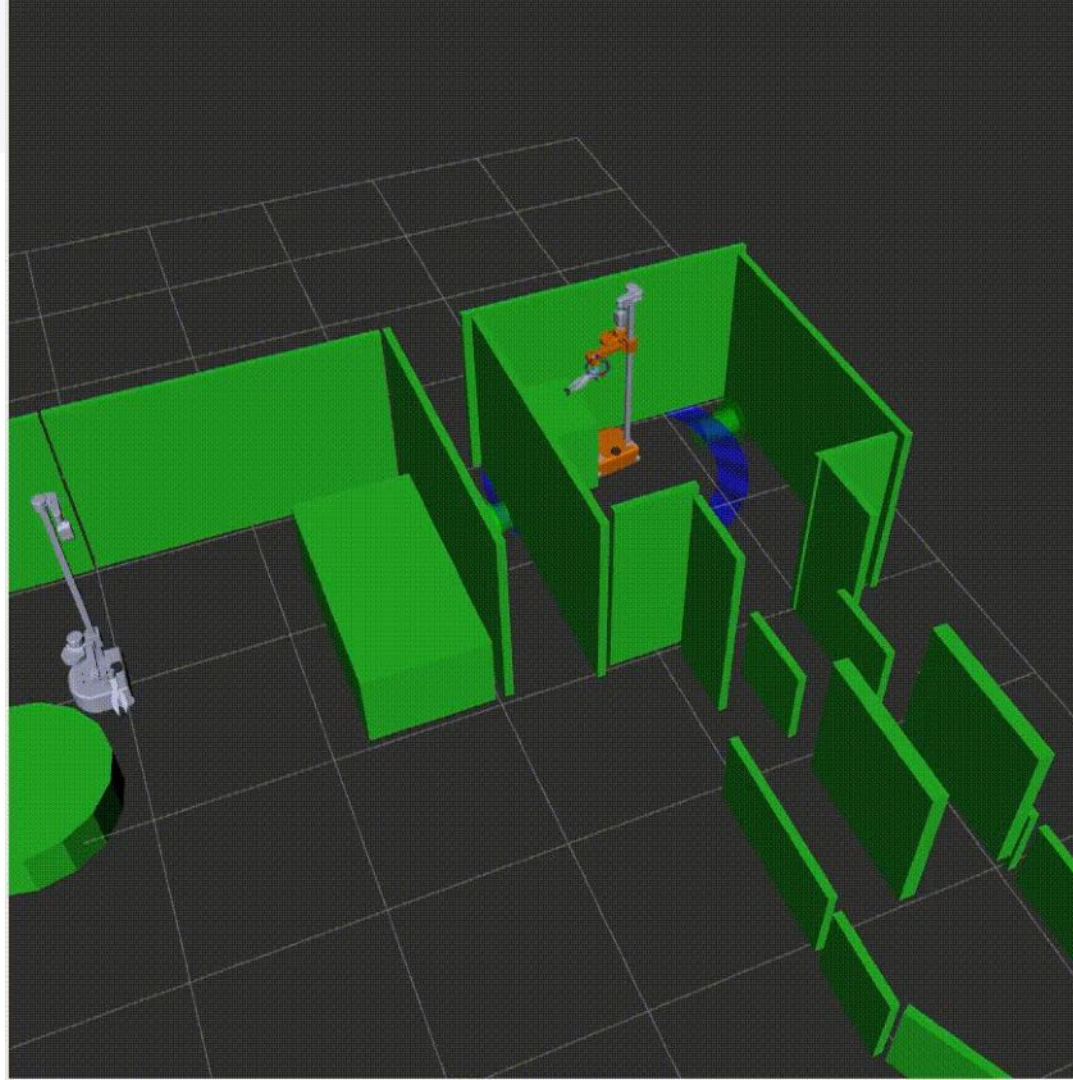
- Can handle:
  - Groups of joints
  - Multivariable joints
  - Mimic joints
  
- Notions of:
  - Cartesian-Space Planning
  - Joint-Space Planning
  - Orientation Constraints
  - Visibility Constraints



## Flexibility (but also complexity)

Differential drive treated as a special joint:

- 3 DOFs: position + orientation
- joint distance/interpolation overridden to correspond to optimal diff-drive paths
- rest of MoveIt “just works”





**152** Robot types integrated to work with Movelt

**29,843** Downloads per month of moveit\_core

**733** Academic citations of Movelt

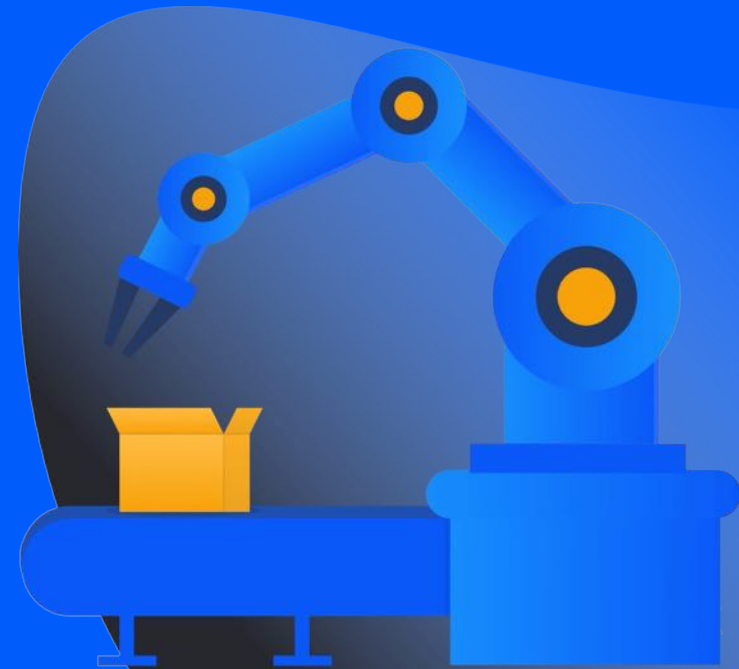
**162,630** Unique users to moveit.ros.org in 2021

**5,600** Members of Discourse, Movelt's Discussion Forum

**1,136** Github users have starred the Movelt project

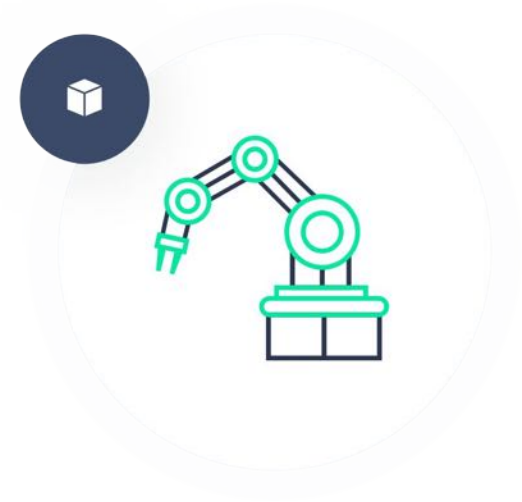
**262** Github code contributors to Movelt

**167** International attendees of 2020 MoveltWorld online event





- World MoveIt Day (annually)
- Monthly MoveIt Manipulation Working Group meetings
- ROSCON / ROS World workshops & tutorials



# Land, Sea, Space



Agriculture

## Plant Harvesting



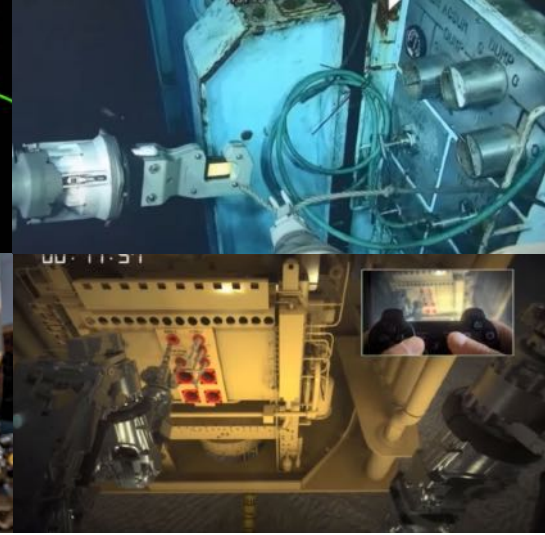
Food

## Kitchen Assistant



Logistics

## Bin Picking



ROVs

## Remotely Operated Underwater Vehicles



IVR  
Inter Vehicle  
Robotics



OSAM  
On-orbit Servicing,  
Assembly, and  
Manufacturing



Planetary Activities  
Lunar Base  
Construction

# Typical Usage Patterns

# Setup Assistant *(work in progress for MoveIt 2)*

**Optimize Self-Collision Checking**

This searches for pairs of robot links that can safely be disabled from collision checking, decreasing motion planning time. These pairs are disabled when they are always in collision, never in collision, in collision in the robot's default position, or when the links are adjacent to each other on the kinematic chain. Sampling density specifies how many random robot positions to check for self collision.

Sampling Density: Low  High 100000

Min. collisions for "always"-colliding pairs: 95%

	panda_link0	panda_link1	panda_link2	panda_link3	panda_link4	panda_link5	panda_link6	panda_link7	panda_hand	panda_leftfinger	panda_rightfinger
panda_link0		✓	✓	✓	✓						
panda_link1	✓		✓	✓	✓						
panda_link2	✓	✓		✓	✓						
panda_link3	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
panda_link4	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
panda_link5				✓	✓			✓	✓		
panda_link6				✓	✓			✓	✓	✓	✓
panda_link7				✓	✓	✓		✓	✓	✓	✓
panda_hand				✓	✓	✓	✓		✓	✓	✓

link name filter   linear view  matrix view

# Rviz Motion Planning Plugin

The screenshot displays the Rviz Motion Planning Plugin interface. The main 3D view shows a robot arm in a simulated environment with blue obstacles. The robot's joints are highlighted in green and orange, and its end effector is surrounded by a green and red striped sphere with intersecting axes, representing the current pose or a target state.

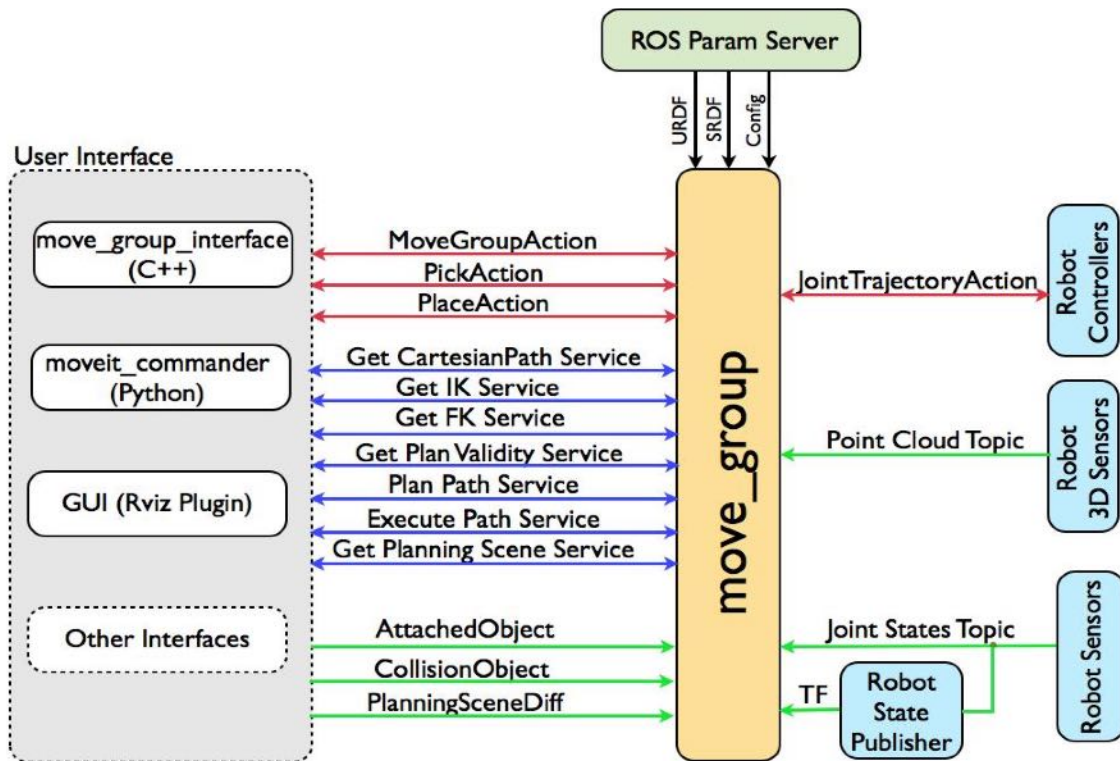
The interface includes several panels:

- Displays:** A list of visual elements with their properties. The 'Goal State Color' is currently selected, showing a color of 246; 160; 49.
- MotionPlanning:** The main control panel with tabs for Context, Planning, Manipulation, Scene Objects, Stored Scenes, Stored States, and Status. It contains:
  - Commands:** Plan, Execute, Plan and Execute, Stop.
  - Query:** Select Start State, Select Goal State, a dropdown menu (currently showing '<random valid>'), an Update button, and a Clear octomap button.
  - Options:** Planning Time (s): 5.00, Planning Attempts: 10.00, Velocity Scaling: 1.00, Acceleration Scaling: 1.00. Checkboxes for Allow Replanning, Allow Sensor Positioning, Allow External Comm., Use Collision-Aware IK (checked), and Allow Approx IK Solutions.
  - Path Constraints:** None, Goal Tolerance: 0.00.
- MotionPlanning - Trajectory Slider:** Waypoint: 0, with a slider and a Pause button.

At the bottom, a status bar provides controls: Reset, Left-Click: Rotate, Middle-Click: Move X/Y, Right-Click: Move Z, Shift: More options. The frame rate is shown as 31 fps.



# ROS Services & Actions



# Pro-Tip: Use C++ classes individually

```
robot_model_loader_.reset(new robot_model_loader::RobotModelLoader("robot_description"));
robot_model_ = robot_model_loader_->getModel();
planning_scene_.reset(new planning_scene::PlanningScene(robot_model_));
tf_.reset(new tf::TransformListener(nh_));
psm_.reset(new planning_scene_monitor::PlanningSceneMonitor(
    planning_scene_, robot_model_loader_, tf_, "my_scene"));
psm_->startStateMonitor("/joint_states", "");
psm_->startPublishingPlanningScene(planning_scene_monitor::PlanningSceneMonitor::
    UPDATE_SCENE, "my_planning_scene");
visuals_tools_.reset(new MoveItVisualTools(robot_model_, planning_scene_monitor_));
planning_pipeline_.reset(new planning_pipeline::PlanningPipeline(robot_model_nh_,
    "planning_plugin", "request_adapters"));
trajectory_execution_manager_.reset(new trajectory_execution_manager::
    TrajectoryExecutionManager(robot_model_));
```

*Best for researchers who want to modify code*

# Pro-Tip: Use MoveItCpp

Full tutorial [here](#)

```
1  static const std::string PLANNING_GROUP = "stretch_arm";
2  auto moveit_cpp_ptr = std::make_shared<moveit_cpp::MoveItCpp>(node);
3  moveit_cpp_ptr->getPlanningSceneMonitor()->providePlanningSceneService();
4  auto planning_components = std::make_shared<moveit_cpp::PlanningComponent>
   (PLANNING_GROUP, moveit_cpp_ptr);
5  planning_components->setStartStateToCurrentState();
6  planning_components->setGoal("extended");
7  auto plan_solution = planning_components->plan();
8
9  if (plan_solution)
10 {
11 | ... ..
12 }
```

# Movelt Task Constructor

`/root/ws_stretch/install/pick_place_task/share/pick_place_task/rviz/mtc.rviz* - RViz`

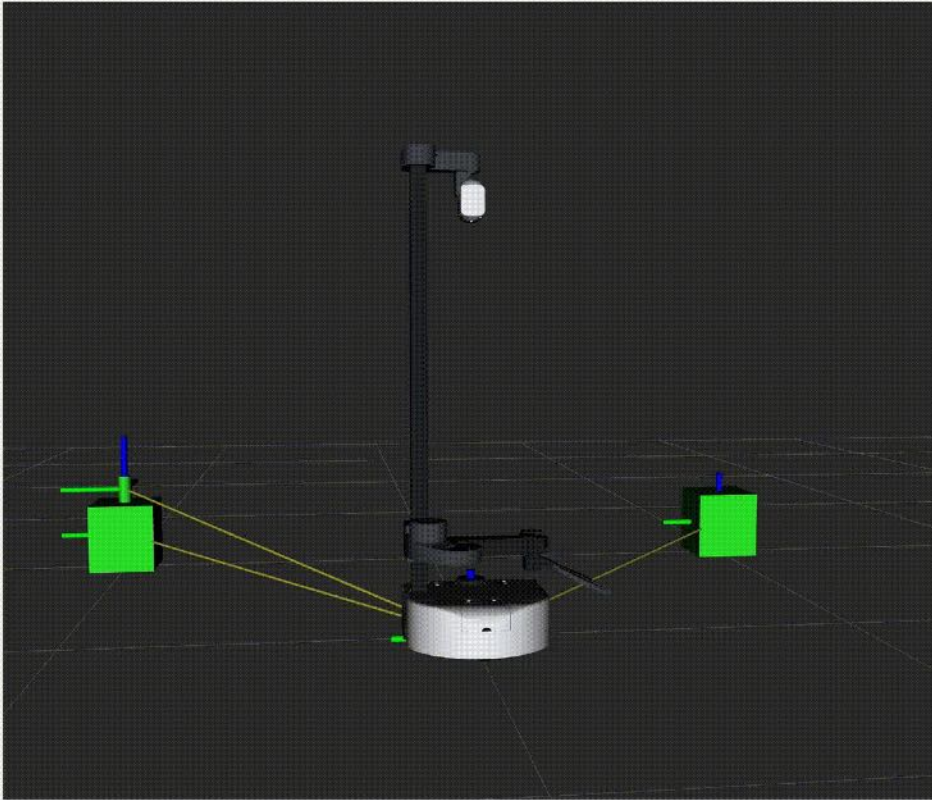
File Panels Help  
Motion Planning Tasks

Task Tree

name	time	#	cost	comment
▼ Motion Planning Tasks		5	39.4736	
▼ task pipeline	5	0.51236	445.8399	
↑ current state	1	0.07172	335.6373	
↓ open hand	1	0.0261	236.4049	
↑ move to object pose	23	0.7809	136.1323	
▼ ↑ pick object	24	0.30374		
↑ approach object	24	0.0545		
▼ ↑ grasp pose IK	61	152.6196		
↑ generate grasp pose	25	0.0002		
↓ allow collision (hand,object)	25	0.0029		
↓ close hand	25	0.3109		
↓ attach object	25	0.0022		
↓ allow collision (object,sup...)	25	0.0019		
↓ lift object	25	0.0424		
↓ forbid collision (object,surf...)	25	0.0021		
↓ move to place	5	0.2235		
▼ ↑ place object	5	0.3380		
↓ lower object	5	0.0089		
▼ ↑ place pose IK	5	22.02788		
↑ generate place pose	500	0.0077		
↓ open hand	5	0.0496		
↓ forbid collision (hand,object)	5	0.0003		
↓ detach object	5	0.0002		

Properties

eef	gripper
forwarded_properties	undefined
group	mobile_base_arm
hand	gripper
ik_frame	link_grasp_center
marker_ns	task
timeout	undefined



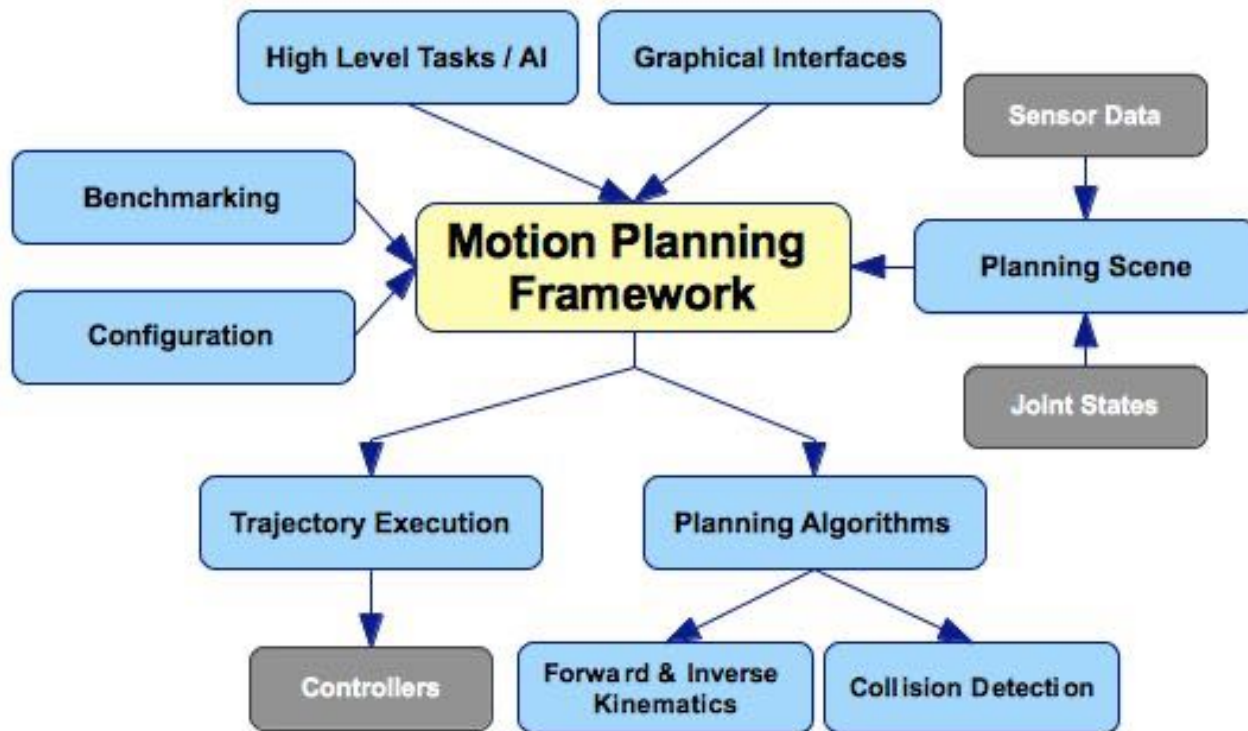
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

31 fps

PICKNIK

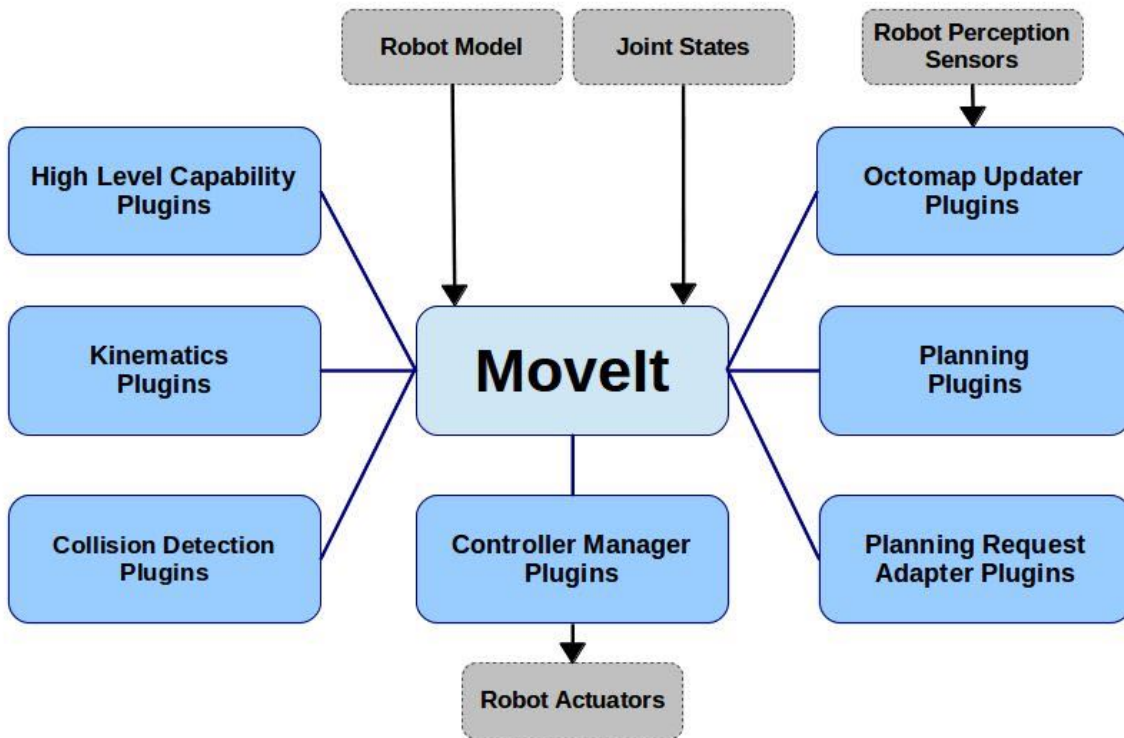
# Components

# High Level Components

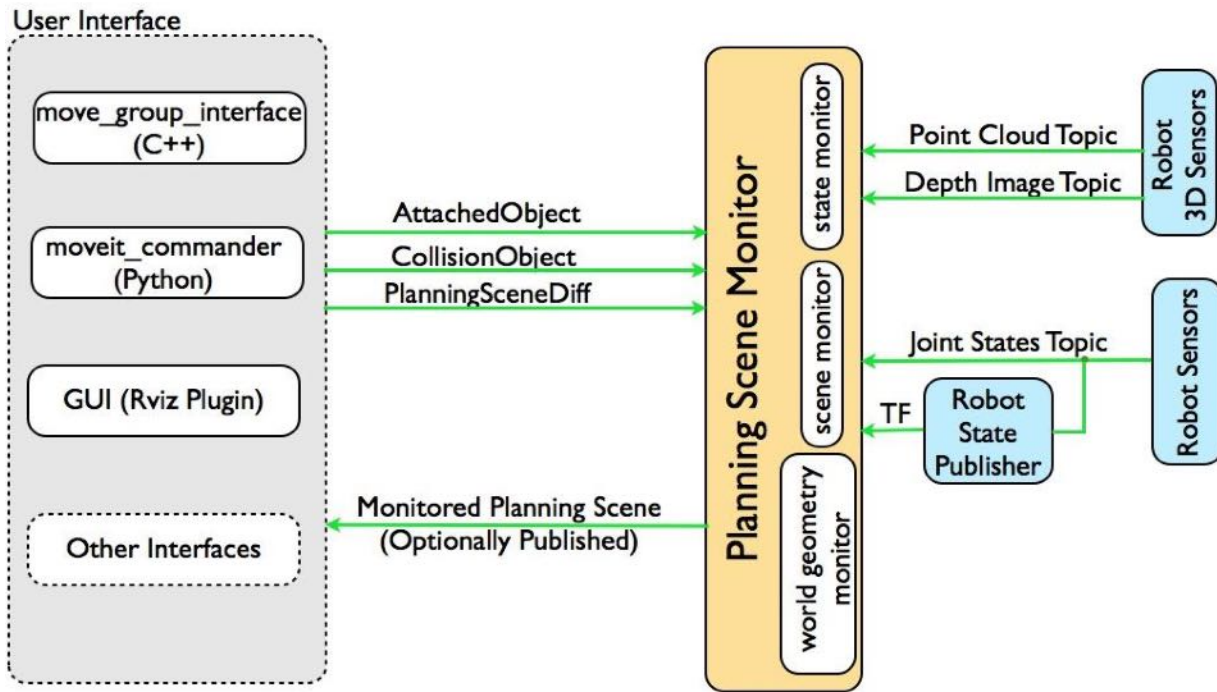


# Plugin Framework

Full details [here](#)



# Planning Scene Monitor





# Motion Planner Plugins

<b>OMPL</b>	<b>STOMP / CHOMP</b>	<b>TrajOpt</b>	<b>SBPL</b>
Kavraki Lab at Rice	Kalakrishnan et al	John Schulman... Pieter Abbeel	Maxim Likhachev's Lab at CMU
Probabilistic, Sampling-Based	Optimization-Based	Sequential Convex Optimization	Graph-Based
Probabilistically Complete	Locally Optimal	Locally Optimal	Resolution Complete
Stochastic	Deterministic	Deterministic	Deterministic
Is a library of many planners Computationally fast More reliable runtime for real-world applications Many variants of algorithms available	Generates smooth well behaved collision free motion paths in reasonable time Can incorporate additional objective functions - collision avoidance and smoothness	Currently in final development	Requires pre-processing phase Requires custom heuristics for different configuration spaces More reliable solutions for real-world applications

# Planner Request Adapters

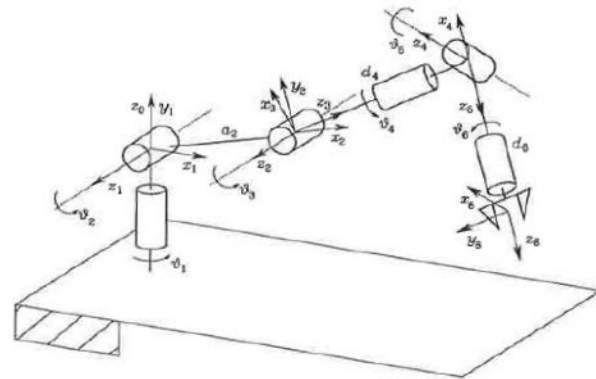
---

- Plugins for pre-/post-processing planner request
- Two types of plugins:
  - **Trajectory generation:** (re)parametrize trajectory to respect velocity, acceleration, jerk limits.
  - **Heuristic fixes for common problems:**
    - set workspace bounds if none defined
    - tweak joints in start state to be within joint limits
    - tweak start state to be collision free

*Adapts research theory to real world hardware*

# Inverse Kinematics

- [KDL](#) - x dof
  - Kinematics Dynamics Library
  - OROCOS
- [IKFast](#) - 6 or 7 dof
  - OpenRave Analytical
- [Trac-IK](#)
  - Combines KDL with Sequential Quadratic Programming approach
- [OPW Kinematics](#) - 6 dof
  - Closed form IK for parallel base, spherical wrist industrial manipulators



# Time Parameterization

---

- TOTG: Time Optimal Trajectory Generation (recommended)
- Ruckig: Only method that supports jerk limits (new!)
- ITPP: Iterative Time Path Parameterization
- TOPP: Time Optimal Path Parameterization

More info [here](#)

# Comparison of MoveIt 1 and MoveIt 2



	MoveIt 1	MoveIt 2
ROS 1 Support	✓	via <a href="#">ros1_bridge</a>
ROS 2 Support	✗	✓
Motion Planning	✓	✓
Inverse Kinematics	✓	✓
Perception	✓	✓
Grasping	✓	✓
Setup Assistant	✓	in <a href="#">development</a>
MoveIt Task Constructor	✓	<a href="#">pending</a>
Game Controller Integration for Servo	✓	✓

	MoveIt 1	MoveIt 2
Industrial Trajectory Generator	✓	<a href="#">planned</a>
<a href="#">Probabilistically complete Cartesian Planning</a>	<a href="#">Stale Patch</a>	✓
Composable Nodes	<a href="#">nodelet subsystem</a>	✓
<a href="#">Planning for Differential Drive Bases</a>	✗	✓
<a href="#">Hybrid Planning</a> (global + local planners)	✗	<a href="#">pending</a>
Based on Realtime Capable DDS Messaging	✗	✓
Native Windows Build	via <a href="#">RoboStack</a>	✓
New Feature Development by PickNik	✗	✓
Development Coordinated with ROS 2 Technical Steering Committee	✗	✓
<a href="#">Built for Industrial Security</a>	✗	✓

# More Resources



Version: MoveIt 2 - Foxy ▾

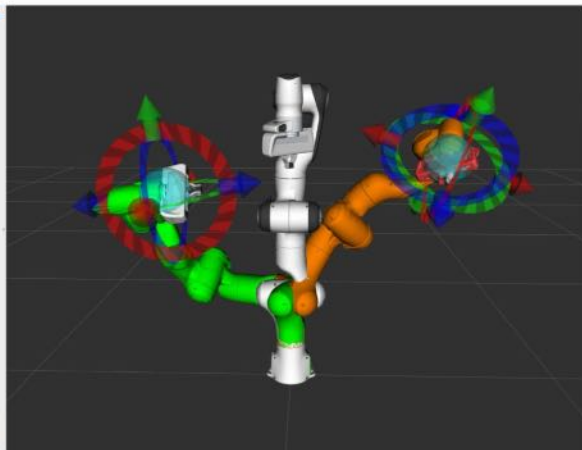
- Getting Started
- MoveIt Quickstart in RViz
- Move Group C++ Interface
- Robot Model and Robot State
- Planning Scene
- Planning Scene Monitor
- Planning Scene ROS API
- MoveItCpp Tutorial
- Planning For Differential-Drive Mobile Base and an Arm
- URDF and SRDF
- Realtime Arm Servoing

MoveIt » Tutorials » MoveIt 2 Tutorials

[Edit on GitHub](#)

## MoveIt 2 Tutorials

These tutorials will quickly get you, and your robot, using the MoveIt 2 Motion Planning Framework.



In these tutorials, the Franka Emika Panda robot is used as a quick-start demo. Alternatively, you can easily use any robot that has already been configured to work with MoveIt - check the [list of robots running MoveIt](#) to see whether MoveIt is already available for your robot. Otherwise, you can setup MoveIt to work with your custom robot in the tutorial section "Integration with a New Robot", below.

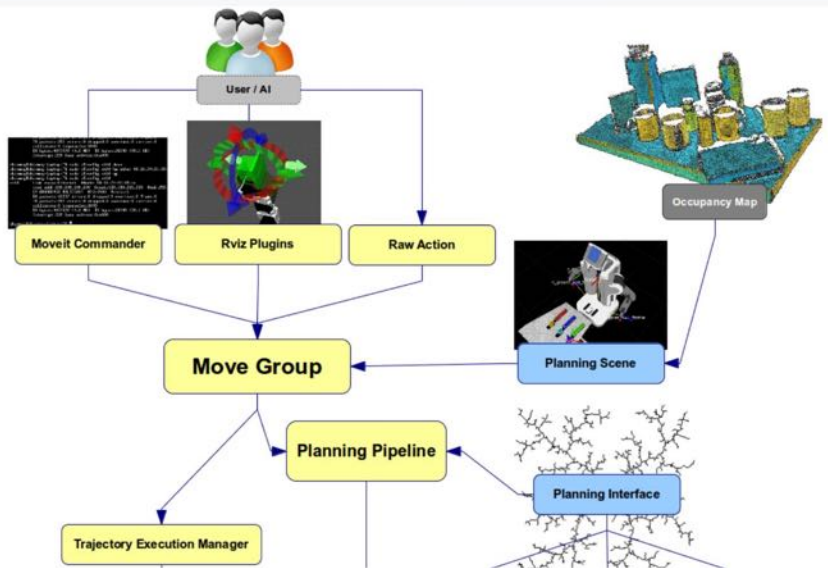
- Tutorials
- Applications
- Concepts**
- Related Projects
- Plugin Interfaces
- Planners
- Source Code & API

## Concepts

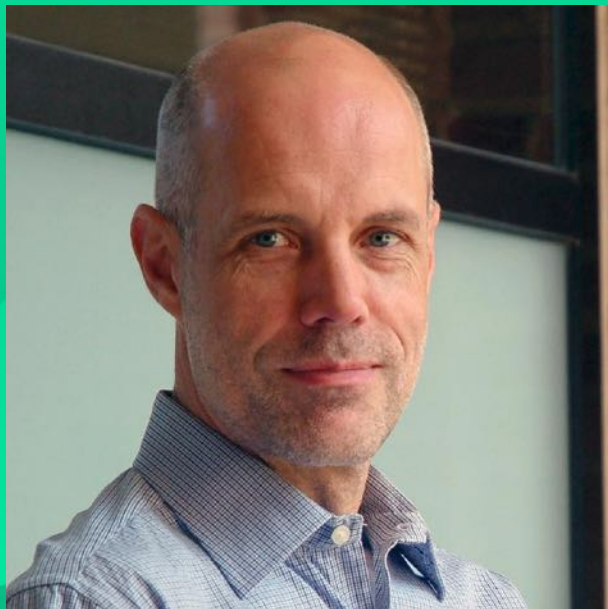
The following is an overview of how MoveIt works. For more concepts and details see the [tutorials](#) or the [developers' concepts](#).

## System Architecture

### Quick High Level Diagram







**Mark Moll**  
mark@picknik.ai

# Thanks!

**PickNik Robotics**

picknik.ai

Colorado, USA

 [@picknikrobotics](https://twitter.com/picknikrobotics)