Sachin Chitta, Ioan Sucan, Acorn Pooley

with contributions from: Dave Coleman, Suat Gedikli, Mario Prats, Matei Ciocarlie, Kaijen Hsiao, Jon Binney, Adam Leeper, Julius Kammerl, David Gossow, Vincent Rabaud, Dave Hershberger and the ROS and PR2 communities
What is MoveIt!

- MoveIt! - Software for building mobile manipulation applications
  - Motion Planning, Kinematics, Collision Checking integrated with Perception, Grasping, Control and Navigation for Mobile Manipulation
Motivation

• Build state of the art software platform for robotics applications and research

• “Simple things should be easy”
  ❖ Provide out-of-the-box experience
    • easy to setup with new robots - Setup Assistant
    • Easy to use APIs - C++ and Python

• “Allow users to dive deeper to address harder problems”
  ❖ Flexible platform - easy to add new components

• Performance
  ❖ design for high performance
Motivation

• Developing new higher-level capabilities is time-consuming
  ✷ building every capability from scratch is a waste of effort

• Environment awareness critical for new applications
  ✷ Integrated 3D perception can provide situational awareness, improving safety, especially in human-robot collaborative tasks

• Motion Planning important for dynamic changing environments
  ✷ essential for maintaining safety and performance in human-robot collaborative tasks

• Constrained manipulation tasks are hard to solve
  ✷ increasingly complex types of constraints need to be handled
MoveIt!: Evolution

• MoveIt! has evolved from the arm_navigation and grasping pipeline set of stacks
  - essentially a rewrite from scratch
  - ROS API almost the same
  - incorporates lessons learnt
Mobile Manipulation: State of the art

Chitta, Jones, Ciocarlie, Hsiao, Sucan, 2011
Robots Using Our Software
Application - ROS-Industrial

Same Software Running on Industrial Hardware

*collaboration between Willow Garage, SWRI and Yaskawa
http://www.rosindustrial.org
MoveIt!
Application - ROS-Industrial

Rob@Work running MoveIt!
MoveIt!

- Technical Capabilities
  - Collision Checking: fast and flexible
  - Integrated Kinematics
  - Motion Planning
    - fast, good quality paths
    - kinematic constraints
  - Integrated Perception for Environment Representation
  - Standardized Interfaces to Controllers
  - Execution and Monitoring
  - Kinematic Analysis
MoveIt!
Collision Checking

● **FCL - Flexible Collision Library***
  - parallelizable collision checking
  - Maximum about 2-3,000 full body collision checks for the PR2 per second
    - with realtime sensor data
  - + high fidelity mesh model

● **Proximity Collision Detection**
  - Uses 3D distance transform to determine distance to nearest obstacle and gradient
  - + very fast - 40 to 80,000 collision checks per second for the full body of the PR2
  - - not as accurate

*Pan, Sucan, Chitta, Manocha - 2012*
Motion Planning

- Plugin interface for planners
- Integration with robots through MoveIt!
- Automatically configured using the MoveIt! Setup Assistant
  - Sampling based planners (OMPL)  [*http://ompl.kavrakilab.org*]
  - Search Based Planning Library (SBPL)  [*http://www.ros.org/wiki/sbpl*]

*OMPL - Sucan, Moll, Kavraki (Rice University), SBPL - Max Likhachev (CMU)*
Easy Setup and Configuration
Easy Setup and Configuration

![Diagram of robot setup]

- **Optimize Self-Collision Checking**:
  - The default self-collision matrix generator will search for pairs of links on the robot that can safely be disabled from collision checking, decreasing motion planning processing time. These pairs of links 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. Higher densities require more computation time.

<table>
<thead>
<tr>
<th>Link A</th>
<th>Link B</th>
<th>Disabled</th>
<th>Reason To Disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>back_wall1</td>
<td>front_wall1</td>
<td></td>
<td>Adjacent Links</td>
</tr>
<tr>
<td>back_wall1</td>
<td>left_wall1</td>
<td></td>
<td>Adjacent Links</td>
</tr>
<tr>
<td>back_wall1</td>
<td>right_wall1</td>
<td></td>
<td>Adjacent Links</td>
</tr>
<tr>
<td>back_wall1</td>
<td>sia20d_arm_base_link</td>
<td></td>
<td>Adjacent Links</td>
</tr>
<tr>
<td>back_wall1</td>
<td>sia20d_arm_kinect_frame</td>
<td></td>
<td>Never In Collision</td>
</tr>
<tr>
<td>back_wall1</td>
<td>sia20d_arm_kinect_ir_link</td>
<td></td>
<td>Never In Collision</td>
</tr>
<tr>
<td>back_wall1</td>
<td>sia20d_arm_kinect_keepout</td>
<td></td>
<td>Never In Collision</td>
</tr>
<tr>
<td>back_wall1</td>
<td>sia20d_arm_kinect_link</td>
<td></td>
<td>Never In Collision</td>
</tr>
<tr>
<td>back_wall1</td>
<td>sia20d_arm_kinect_s</td>
<td></td>
<td>Never In Collision</td>
</tr>
<tr>
<td>back_wall1</td>
<td>table_base1</td>
<td></td>
<td>Adjacent Links</td>
</tr>
<tr>
<td>front_wall1</td>
<td>sia20d_arm_base_link</td>
<td></td>
<td>Adjacent Links</td>
</tr>
<tr>
<td>front_wall1</td>
<td>sia20d_arm_kinect_frame</td>
<td></td>
<td>Collision by Default</td>
</tr>
<tr>
<td>front_wall1</td>
<td>sia20d_arm_kinect_ir_link</td>
<td></td>
<td>Collision by Default</td>
</tr>
<tr>
<td>front_wall1</td>
<td>sia20d_arm_kinect_keepout</td>
<td></td>
<td>Collision by Default</td>
</tr>
<tr>
<td>front_wall1</td>
<td>sia20d_arm_kinect_link</td>
<td></td>
<td>Collision by Default</td>
</tr>
<tr>
<td>front_wall1</td>
<td>sia20d_arm_kinect_s</td>
<td></td>
<td>Collision by Default</td>
</tr>
</tbody>
</table>

- **Precompute allowed contact points**

- **Regenerate Default Collision Matrix**

---

**Willow Garage**
Easy Setup and Configuration

**Robot Poses**
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like folded arms.

<table>
<thead>
<tr>
<th>Pose Name</th>
<th>Group Name</th>
</tr>
</thead>
</table>

Adding robot poses
Joint configurations can be optionally saved under specified names.
Easy Setup and Configuration
Easy Setup and Configuration

Live Demo by Dave Coleman later today!!
Integrating Perception

Representation of perceived world
Only ground is visible

Octomap: octomap.sf.net
Point Cloud Library: pointclouds.org
Reactive Motion

* Live Demo later!!
Constraint Representation

- Joint Constraints
- Position Constraints
- Orientation Constraints
- Visibility Constraints
Orientation Constraints
Applications: Benchmarking

- Industrial Environment
- Kitchen Environment
- Narrow Passage Environment
- Tabletop Environment

* Cohen,Sucan,Chitta, “A Generic Infrastructure for Benchmarking Motion Planners”, IROS 2012, Portugal
Benchmarking

More in Talk by Ryan Luna and Ioan Sucan (later today)
Applications: Kinematic Workspace Analysis

• Robot Design Evaluation
• Robot Workspace Placement
Kinematic Workspace Analysis

![Diagram of robot in a workspace]

Reachable in collision-free way (% of total)

- Countertop
- Floor and wall
- Fridge
- Pantry shelves
- Shelf with bins, roomy
- Mean

- PR2
- Kuka LWR
- UR5
MoveIt!

- Applications - Pick and Place
  - Integrated Grasping, Planning, Perception and Execution
User API

• Really simple API (e.g. moving an arm):

```cpp
move_group_interface::MoveGroup group("arm");
group.setRandomTarget();
group.move();
```
MoveIt!

- New Architecture (different from arm navigation)
  - Minimize transport and messaging overhead - Single process for planning and perception, shares environment representation (planning scene) vs. multiple ROS nodes each performing individual functions
  - Computation - Core capabilities (e.g. motion planning, kinematics, etc.) are setup in C++ libraries
  - Communication and Configuration through ROS
  - Emphasis on speed and efficiency – parallelize collision checking, kinematics, etc.
MoveIt!

- Capabilities (differences to arm navigation)
  - Collision Checking
    - Parallelizable
    - can switch between different types of collision checkers
    - cleaner C++ interface
  - Motion Planning
    - plugin based C++ interface (in addition to ROS interface)
    - Parallelizable
    - planning pipeline includes trajectory smoothing
Highlights

• Technical

- Performance
  ❖ Single process sharing environment representation
  ❖ Parallelizable collision checking and kinematics
  ❖ Parallelizable pick and place (upcoming capability)

- Integrated Perception for Environment Representation
  ❖ Can incorporate any source of point clouds
  ❖ Fast self-filtering and environment representation

- Reactive Motion Planning
  ❖ Safer operation in collaborative environments
Highlights

• User Friendly
  ❖ Easy configuration for new robots
  ❖ Graphical User Interfaces
  ❖ Better Visualization and Introspection
  ❖ Easy to use C++ API
  ❖ Python bindings
Highlights

- Integrated Applications
  - Collision-free Motion Planning and Execution
  - Kinematic Analysis/ Reachability Analysis
  - Benchmarking

- More applications in development ...
  - Pick and Place - more about this in afternoon session
Documentation - Wiki

Movelt!

Movelt is a new software framework for motion planning in ROS. Movelt is a core part of ROS and will replace the Arm Navigation packages. Movelt is not fully ready yet. We will send out an email on ros-users once it is ready for use. In the meantime have a look at these videos to see how Movelt works.
Github Repository
Issue Tracking
Community

- [moveit-users@googlegroups.com](mailto:moveit-users@googlegroups.com) - questions related to how you can use MoveIt!
Where are we going?

- MoveIt!
  ✓ what does it take to use MoveIt! in products
  ✓ interest in enterprise level, supported versions of MoveIt! and associated capabilities in ROS?
More information

Email: moveit@willowgarage.com