Strengths, Weaknesses, and Developer Insights

ROSCon Hamburg 2015

Dave Coleman
dave@dav.ee
Overview

Establish Credibility 1min
Background of MoveIt! 5min
What it's done well 5min
Typical use patterns 5min
Demystifying complexity 5min
Amazon Picking Challenge 5min
Where MoveIt! needs improvement 4min
Future Roadmap 10min
Q&A 5min
Establish Credibility 1min

- PhD Student At CU Boulder with Nikolaus Correll
- Interned with E. Gil Jones & IoanSucan at Willow Garage
  - Created Setup Assistant
- Have used and contributed to MoveIt! since before it was released
- Am a MoveIt! maintainer
- Have contributed to OMPL and many other ROS packages
Background of MoveIt! 5min

Easy to use framework for motion planning, manipulation, 3D perception, kinematics, control and navigation

- Created at Willow Garage by Ioan Sucan, Sachin Chitta, many others
- Collaboration between many organizations
- Predecessor: arm_navigation announced in March 2010
- 31 contributors to moveit_core
- Written in C++ with Python bindings
- [https://github.com/ros-planning/](https://github.com/ros-planning/)
The First Movelt!
Community Meeting
(Online)

240 people registered, 150 who attended
It's popular.

- #3 Package in ROS (ROS survey)
- 700 Members on Mailing list
- Number of installations 2015: 10,089
- ICRA 2015
  - 11 Papers cited/used MoveIt!
- IROS 2015
  - 5 Papers cited/used MoveIt!
- Has been run on over 65 robots worldwide
Community

Movelt! Mailing List Membership

Movelt! Code Contributors

Total Posts on Movelt! Mailing List

Motion Planning Library Code Contributors
Exciting Developments

- Benchmarking getting rewrite
  - Mark Moll / Kavraki Lab
- STOMP being revived
  - Jorge Nicho / SwRI
- Descartes Cartesian Planner
  - Shaun Edwards / SwRI
- Collision detection plugin
  - Michael Ferguson / Fetch
- New release maintenance manager
  - Michael Ferguson / Fetch
What it's done well 5min
Setup Assistant

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.

Sampling Density: Low — High

<table>
<thead>
<tr>
<th>Link A</th>
<th>Link B</th>
<th>Disabled</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>backpack</td>
<td>base</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>baseplate</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>body_cover</td>
<td></td>
<td>Collision by</td>
</tr>
<tr>
<td>backpack</td>
<td>left_shoulder_pitch</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>left_shoulder_roll</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>neck_base</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>neck_lower_pitch</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>neck_roll</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>right_shoulder_pitch</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>right_shoulder_roll</td>
<td></td>
<td>Never in Co</td>
</tr>
<tr>
<td>backpack</td>
<td>waist_center</td>
<td></td>
<td>Adjacent Lli</td>
</tr>
<tr>
<td>base</td>
<td>baseplate</td>
<td></td>
<td>Adjacent Lli</td>
</tr>
</tbody>
</table>

Regenerate Default Collision Matrix

Show Non-Disabled Link Pairs

Min. collisions for "always"-colliding pairs: 95%
Rviz Motion Planning Plugin
Comparing research libraries

- **OMPL**
  Open Motion Planning Library
- **SBPL**
  Search Based Planning Library
- **CHOMP**
  Covariant Hamiltonian Optimization
- **STOMP**
  Stochastic Trajectory Optimization
- **FCL**
  Fast Collision Checking Library
- **PCD**
  Proximity Collision Detection
- **IKFast**
  Analytical Inverse Kinematics Solver
- **KDL**
  Kinematics Dynamics Library - Inverse Kinematics
- **Octomap**
  3D occupancy grid mapping

Robot Agnostic
Flexibility (but also complexity)

- Can Handle:
  - Groups of joints
  - Multivariable joints
  - Mimic joints

- Notions of:
  - Cartesian-Space Planning
  - Joint-Space Planning
  - Orientation Constraints
  - Visibility Constraints
Typical Use Patterns  5min
Rviz Motion Planning Plugin
Known commands:

help                show this screen
id|which            display the name of the group that is operated on
load [<file>]       load a set of interpreted commands from a file
save [<file>]       save the currently known variables as a set of commands
use <name>          switch to using the group named <name> (and load it if necessary)
use|groups          show the group names that are already loaded
vars                display the names of the known states
show                display the names and values of the known states
show <name>         display the value of a state
record <name>       record the current joint values under the name <name>
delete <name>       forget the joint values under the name <name>
current             show the current state of the active group
constrain <name>    use the constraint <name> as a path constraint
constrain           clear path constraints
eef                 print the name of the end effector attached to the current group
go <name>           plan and execute a motion to the state <name>
go <dir> <dx>       plan and execute a motion in direction up|down|left|right|forward|backward
rand                plan and execute a motion to a random state
plan <name>         plan a motion to the state <name>
execute             execute a previously computed motion plan
rotate <x> <y> <z>  plan and execute a motion to a specified orientation (about the X,Y,Z axes)
tolerance           show the tolerance for reaching the goal region
tolerance <val>     set the tolerance for reaching the goal region
wait <dt>           sleep for <dt> seconds
x = y               assign the value of y to x
x[idx] = val        assign a value to dimension idx of x
x = [v1 v2...]      assign a vector of values to x
trace <on|off>      enable/disable replanning or looking around
ground              add a ground plane to the planning scene
allow replanning <true|false> enable/disable replanning
allow looking <true|false> enable/disable looking around

> a = current
> go rand
> wait 5
> plan a
move_group Python Interface

```python
import moveit_commander
import geometry_msgs.msg

# Initialize MoveGroupCommander
group = moveit_commander.MoveGroupCommander("left_arm")

# Create a pose target
pose_target = geometry_msgs.msg.Pose()
pose_target.orientation.w = 1.0
pose_target.position.x = 0.7
pose_target.position.y = -0.05
pose_target.position.z = 1.1

group.set_pose_target(pose_target)

# Plan the move
plan1 = group.plan()
```
moveit::planning_interface::MoveGroup group("right_arm");

geometry_msgs::Pose target_pose;
target_pose.orientation.w = 1.0;
target_pose.position.x = 0.28;
target_pose.position.y = -0.7;
target_pose.position.z = 1.0;
group.setPoseTarget(target_pose);

moveit::planning_interface::MoveGroup::Plan my_plan;
bool success = group.plan(my_plan);
ROS Services & Actions

User Interface

move_group_interface (C++)
moveit_commander (Python)
GUI (Rviz Plugin)
Other Interfaces

move_group

ROS Param Server
URDF
SRDF
Config

JointTrajectoryAction
Point Cloud Topic
Joint States Topic
TF

Robot Controllers
Robot 3D Sensors
Robot Sensors

MoveGroupAction
PickAction
PlaceAction
Get CartesianPath Service
Get IK Service
Get FK Service
Get Plan Validity Service
Plan Path Service
Execute Path Service
Get Planning Scene Service
AttachedObject
CollisionObject
PlanningSceneDiff

Robot State Publisher
Pro-Tip: Use C++ classes individually

```cpp
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
Demystifying Complexity 5min

- Motion Planning Framework
  - High Level Tasks / AI
  - Graphical Interfaces
  - Benchmarking
  - Configuration
  - Trajectory Execution
  - Planning Algorithms
    - Controllers
    - Forward & Inverse Kinematics
    - Collision Detection
  - Sensor Data
    - Planning Scene
    - Joint States
Many Plugins
Planning Scene Monitor

User Interface

- `move_group_interface` (C++)
- `moveit_commander` (Python)
- GUI (Rviz Plugin)
- Other Interfaces

Planning Scene Monitor

- AttachedObject
- CollisionObject
- PlanningSceneDiff

- Point Cloud Topic
- Depth Image Topic
- Joint States Topic
- TF

Robot State Publisher

Robot Sensors

Robot 3D Sensors

Monitored Planning Scene (Optionally Published)
<table>
<thead>
<tr>
<th>Planners</th>
<th>OMPL</th>
<th>SBPL</th>
<th>CHOMP/STOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lydia Kavraki's lab</td>
<td>Maxim Likhachev's lab</td>
<td>Kalakrishnan et al</td>
<td></td>
</tr>
<tr>
<td>Sampling-based planners</td>
<td>Graph-based planners</td>
<td>Optimization-based planner that generates smooth well behaved collision free motion paths in reasonable time</td>
<td></td>
</tr>
<tr>
<td>Stochastic</td>
<td>Deterministic</td>
<td>Can incorporate additional objective functions - collision avoidance and smoothness</td>
<td></td>
</tr>
<tr>
<td>Probabilistically complete</td>
<td>Resolution complete</td>
<td>CHOMP being resurrected by ROS Industrial group</td>
<td></td>
</tr>
<tr>
<td>Typically no optimality guarantees</td>
<td>Optimality guarantees</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computationally fast</td>
<td>Requires pre-processing phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>More reliable runtime for real-world applications</td>
<td>Computationally expensive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Many variants of algorithms available</td>
<td>More reliable solutions for real-world applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Renewed work from Michael Ferguson</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experience Planners
Planner Request Adapters

- **AddTimeParameterization**
  - Modifies the timestamps of a kinematic (position-based) trajectory to respect velocity and acceleration constraints
  - Uses iterative parabolic time parameterization
- **FixWorkspaceBounds**
  - If no minimum workspace bounds is specified, sets to a default
- **FixStartStateBounds**
  - Tweaks joints to not be outside joint limits
  - Accounts for floating point and encoder noise
- **FixStartStateCollision**
  - Tweaks start state to not be in collision with environment
  - Creates a new planning request with modified start state
- **FixStartStatePathConstraints**
  - Plans separate path from invalid start state to valid start state

*Adapts research theory to real world hardware*
IK Solvers

- **KDL**
  - Kinematics Dynamics Library, OROCOS
- **IKFast**
  - OpenRave Analytical
- **Robot-specific custom solvers**
  - PR2
Amazon Picking Challenge 5min
Workspace Analysis

Baxter Parallel Electric Gripper

Yale OpenHand 3-Finger Gripper
Kinova Jaco2 + 1m Vertical Gantry
Challenge Takeaways

- Simplest possible grasping → suction
- Low cost hardware → visual servoing
- Reduce calibration needs
- 2 mobile bases won → larger workspace
- Slim arms → better reachability
- Good visualizations → introspection and development
- Perception and manipulation teams must work closely
- Test whole system working together often

Movelt! used by at least 10 teams:
  - PickNik, Z.U.N., University of Washington, Team IntBot,
    NUS_SMART_HAND, Team Applied Robotics, Team WPI,
    University of Alberta Team, Plocka Packa, Team CVAP

None of the winning teams used Movelt!
Team RBO - 1st Place
TU Berlin
148 points

Barrett WAM arm
- Backdrivability key to skillful interactions with the environment

Nomadic XR4000 mobile base
- Omnidirectional / holonomic
- Very large workspace

Did not rely on motion planning

Hybrid automaton composed of sequences of controllers with sensor-based transitions.

Vacuum attachment tool with suction cup drilled into side of fender

“Simple but robust” RGB object recognition algorithm
Team MIT - 2nd Place
MIT
88 points

ABB 1600ID
- Sub-millimeter precision
- Internal canals for cables

Custom dual-purpose end effector
- Aviation-grade aluminum
- Spatula-like finger nail
- Suction also

Used MIT Drake (Locomotion Group at MIT) for motion planning

Automatically chooses which motion primitive to use based on dynamics simulator
- grasp, suck, scoop, toppling, push-rotate

Kinect2 cameras mounted on frame, Realsense on arm

Outsourced perception to a robotics startup - Caspen Robotics
Team Grizzly - 3rd Place
Oakland University
w/Dataspeed Inc.
35 points

- Rethink Baxter
- Custom Mobile Base
- Yale OpenHand
- Suction gripper
- Kinect2 on Head

Custom Cartesian motion planning algorithm accepted position and orientation commands from the perception system
Where Movelt! needs improvement 4min
(and where you can help!)
Motion Planner Reliability

- Sometimes fails with difficult to understand explanations
- Sometimes generates very suboptimal paths

Solutions:
- Hybridize several planning attempts (threads)
- Plan with cost functions, e.g. RRT*, PRM*
- Increase the time MoveIt! spends on smoothing paths
- For some applications, planners other than OMPL's defaults are better
- Improve user feedback to diagnose setup issues
  - Check your joint space is parameterized correctly (<2pi)
  - Introspection tools
Obstacle Clearance

Can generate plans that come very close to obstacles

Solutions:
- Add out of box support for biasing trajectories away from obstacles
- Cost-based OMPL, STOMP, CHOMP
Grasping Support

Difficult to generate grasps in MoveIt!

Solution:

- Provide default URDF + SRDF compatible grasp generator
- Clearer documentation on how to integrate third party grasping pipeline
Documentation

- Need more exhaustive documentation from community support (*you!*)
- Tutorials for how to use Movelt! beyond quick start demo
- Make it easier for our many users to contribute back
Community meeting's end of year goals

- Integrate better support for humanoid kinematics
- Integrate benchmarks updates
- Resurrect support for other types of planners (STOMP)
Visual Servoing Support

- Once a trajectory is planned, no easy way to integrate visual or tactile feedback

**Solution:**

- Position/pose-based visual servoing (PBVS)
- Hooks to modify plan based on alignment of target object
- Ability to add meta-data to trajectories indicating when to use VS, what objects to track
- Requires much tighter coupling with controllers, planners, and perception system
Planning with Behaviors

- Free Space Planning
- OMPL
- Cartesian Straight Line + Visual Servoing + Impedance Control
- Grasp Tool
- Cartesian Straight Line + Impedance Control
- Semi-constrained trajectory using Descartes
- Free Space Planning
- OMPL
- Cartesian Straight Line + Impedance Control
Sense-Plan-Act & ROS Control

- Faster connection for streaming commands
- Integrate ros_control with Setup Assistant
- Rename MoveIt ControllerManager to ControllerInterface
- More advanced plugin than SimpleControllerManager
- Switching controllers
Affordance Templates

Human in the loop tools for high level commands such as more sophisticated interactive markers
Calibration

- Integrate better calibration packages
  - ROS Industrial - `industrial_calibration`
  - Fetch Robotics - `robot_calibration`
- More clearly document how this should be integrated
Stability vs. Progress

- MoveIt! needs to stay current
- Other motion planning frameworks are very capable
  - OpenRave, MIT Drake, MuJuCo + Whole Body Planning, etc

Distributed Software Collaboration Is Hard

- Currently we have 50 open pull requests
- Need continuous integration badly
- Need more simulation tests

If there are good features worth upgrading to, breaking changes are tolerable.
Proposal: Consolidate to One Repo

Inconsistent moveit_ros and moveit_msgs with Indigo?
#615

Closed RashmicaG opened this issue 2 days ago · 3 comments

dg-shadow commented 23 days ago

Extends #594 to allow connection to remote warehouse db.

Goes with ros-planning/warehouse_ros#24

TheBrewCrew commented on Aug 14

This enables the changes in ros-planning/moveit_core#255 by adding a ROS param
(/move_group/octomap_diff) to enable change detection on the Octomap. This pull request should build
without ros-planning/moveit_core#255 but will not do anything until it is merged.
Too many repos to keep synced

moveit_core
moveit_ros
moveit_planners
moveit_docs
moveit_msgs
moveit_robots
moveit_ikfast
moveit_commander
moveit_kinematic_tests
moveit_advanced
moveit_setup_assistant
moveit_metapackages
moveit_plugins
moveit_resources
moveit_pr2

packages not yet in rospkg group:
moveit_benchmarks
moveit_visual_tools
moveit_simple_grasps
moveit_python
moveit_web
moveit_whole_body_ik
industrial_moveit

Plus many ros-planning packages not prefixed with moveit_*
Overall

- Movelt! is awesome
- Successful because it is easy for beginners
- Needs many more features and improvements
- Stability (stagnation) should not be the #1 focus
- Please contribute!
Do you like to Movelt Movelt?

Thanks to Mike Ferguson, Sachin Chitta, Ioan Sucan, Shaun Edwards, Jon Bohren, Conor Brew, Acorn Pooley, Dave Hershberger, Chris Lewis, Jorge Nicho, Ben Chretien, Adolfo Rodriguez, Kei Okada, Stefan Kohlbrecher, and many more...