Winning the Amazon Picking Challenge 2016

Team Delft – Motion Module and Lessons Learned

Warehouse Picking Automation Workshop (WPAW 2017)
Mukunda Bharatheesha
29.05.2017
## Introduction

<table>
<thead>
<tr>
<th>Stowing Challenge</th>
<th>Picking Challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delft - 214</td>
<td>Delft – 105 (00:00:30)</td>
</tr>
<tr>
<td>NimbRo - 186</td>
<td>PFN – 105 (00:01:07)</td>
</tr>
<tr>
<td>MIT - 164</td>
<td>NimbRo - 97</td>
</tr>
</tbody>
</table>
Contents

• Team Delft APC motion module with MoveIt! [2]
• MoveIt! Specific lessons learnt
• General lessons learnt
• Concluding remarks
Robot Setup

Selected configuration

roslaunch moveit_workspace_analysis workspace_analysis.launch
roslaunch moveit_workspace_analysis reader.launch
Basic Software Architecture

High level task manager

Sense
- Object Recognition
- Object Pose Estimation
- Bin Pose Estimation

Plan
- Grasp Synthesis
- Manipulation Planning

Act
- Coarse Motion
- Fine Motion
- I/O
Motion Module – Robot motion

Robot Motions

- Offline
  - Coarse Motions
    - Free (joint) space motion planning with RRT-Connect.
      (Trajectory Cache)
  - Fine Motions
    - Cartesian path planning to achieve final grasp pose.

- Online

- robot_state::RobotState APIs for setting various “MasterPoses”.
- planning_interface::MoveGroup APIs for planning.
Motion Module – I/O

• “Event” based I/O
  • Key waypoints - “approach”, “contact”, “lift”, “retreat”
  • I/O actions - “vacuum on”, “suction on”, “front suction activate”, …

• Reliable trajectory tracking
  • Track trajectory waypoints (distance to waypoints).
  • Stable tracking regardless of other tasks.
  • Dedicated callback queue and background spinner.
Lessons Learnt
1. Collision checking redundancy

“Gap” in octomap due to reflections
Lessons Learnt

1. Collision checking redundancy

- Bin collision models

- Collision checking with “new” meshes in the planning scene
  - Non-reproducible but random failures in collision checking.

- Add sanity checks for “impossible” motions.
Lessons Learnt
2. Octomap – Planning Scene

- “Static” point cloud
- “You Only Look Once”
- Simulated point cloud stream using a topic.

```
MoveGroup::getCurrentState() takes long time to return while using octomaps
```

- Use `robot_state::JointModelGroup` to avoid delays when real robot state is different from “test” robot state during online planning.
- Cause: Planning scene monitor prioritises servicing octomap updates(*)
Lessons Learnt
3. Octomap clearance

• “Static” point cloud
  • One point cloud snapshot instead of a point cloud “live feed”.
  • Simulated point cloud stream using a topic.

• `/clearOctomap()` - Inconsistency with clearing the current octomap.
  • Publish “out of range” valued point cloud.
  • Octomap always cleared on service call.
Lessons Learnt

Assumptions

- Optimal planners
  - Euclidean distance NOT default.

- Fake components
  - Clarity in interface definitions.
  - “Easy” integration.

- Writing “json” files correctly
  - Especially when pick/stow attempt fails.

- Robot Driver*
  - Sanity checks.

“Assumptions bear the roots of all disasters!”
Concluding Remarks

20 days before the challenge!
(No stowing yet!)
Concluding Remarks

- Stowing completed in almost half of allotted time.

- More than 16 picks performed in the given time (including “move jobs”)

- Not moving like Icebergs anymore!
Acknowledgements

The work leading to these results has also received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 609206.
Thank you for your attention!
References


[2] moveit.ros.org


Lessons Learnt
4. Trajectory stitching

- Combine fine motions with coarse motions
- IK solution need not necessarily match with coarse motion start.
- RRT-Connect from last IK point to “MasterPose”.

- Trac-IK with “distance” setting
- Default is “speed” setting.

- Combine multiple motion segments in joint space.
- Re-(time) parameterize the “stitched” motions in time
  \( \text{iptp.computeTimeStamps()} \).