Multi-Arm Control in MoveIt!

Dave Coleman, CU Boulder

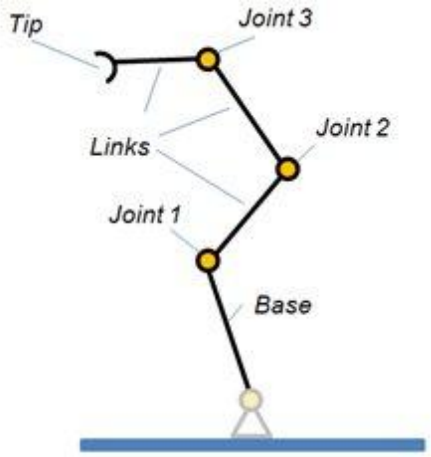
14 June, 2016

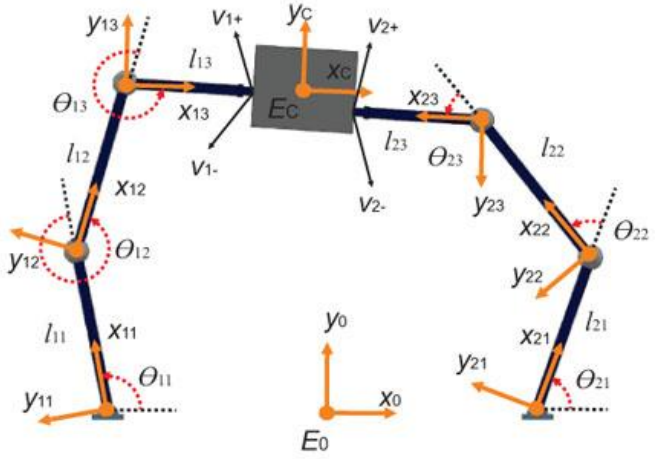
Immediate Desires

- Quickly generate smooth collision free motion plans for multiple kinematic chains
- Accommodate alternating closed loop kinematic chains
  - Holding a tray with two arms
  - Two legs attached to each other via a ground surface
- Solve inverse kinematics using overlapping joints such as torso twist and pitch
- Manipulate objects being held in a secondary arm

Multi-Arm Inverse Kinematics

- Current Capabilities
  - Can solve for multiple kinematic chains
    - Implementation: `RobotState.setFromIK(tips, poses)`
  - Uses KDL, IKFast, etc and iteratively solves subgroups independently, only checking for overlapping collisions
  - Kinematic chains *cannot* have overlapping joints
    - i.e. a torso rotation (Motoman dual arm) or torso lift (PR2)



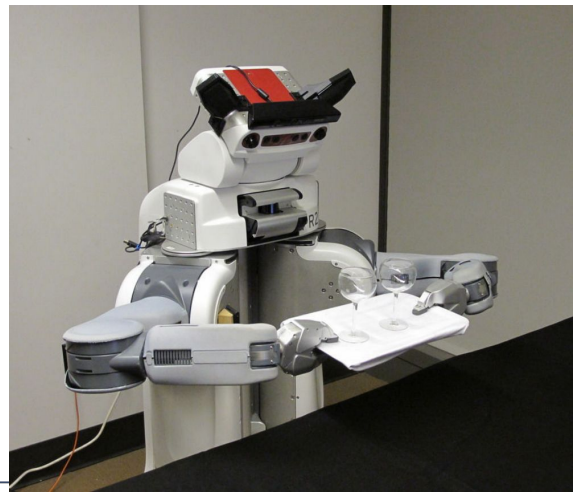


Solving Multiple Chains

- Needs IK solver plugin for solving multiple chains at once
  - with possibly overlapping joints
- Recently added multiple tip support
  - Would like to cleanup KinematicsBase API
- Experimental *moveit_whole_body_ik* solver
  - Creates Jacobian matrix that represents both arms, legs, and confounding torso joints
  - Not in finished state
- Creating or integrating other IK solvers would be desirable
  - *Stack of Tasks*, CNRS-LAAS group

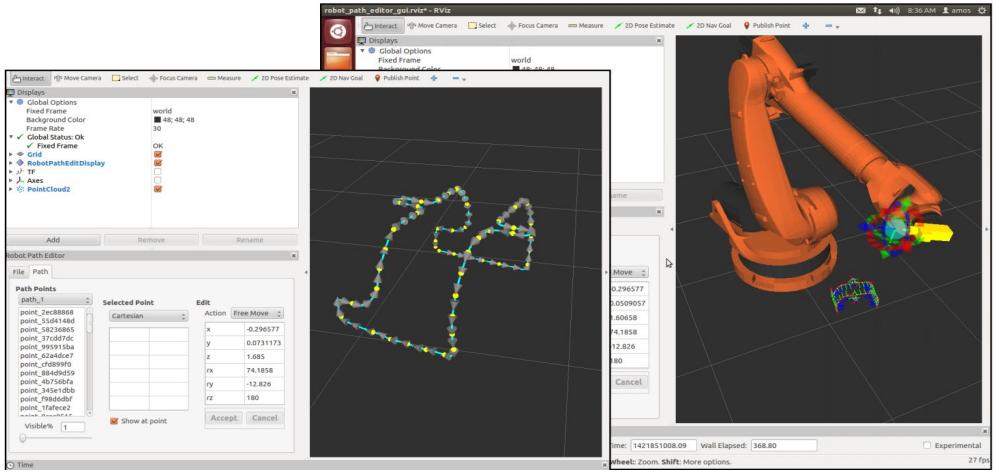
Multi-Arm Motion Planning

- Current multi-arm planning capabilities
  - Method 1: Joint space
    - Randomly samples all joints and checks for collision
    - Computationally intractable to plan with
      - Arms attached to each other / attached to a common object
      - Orientation constraints
  - Method 2: Work space
    - Randomly samples a Cartesian pose of object being manipulated, solve each arm using IK
    - Much slower because of required IK calls
    - Can enforce orientation constraints
      - Don't spill water glass



Partially Supported Capabilities

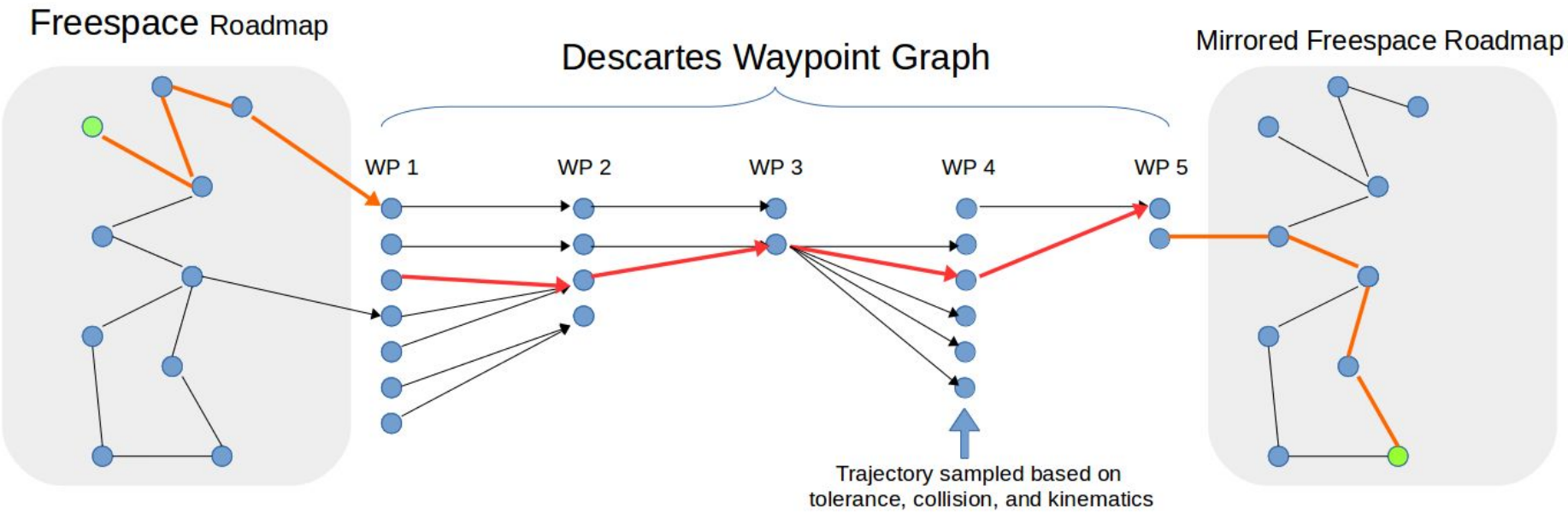
- Inputting exact Cartesian path and solving required joint positions
  - Implementation: `RobotState.computeCartesianPath()`
  - Does not handle getting stuck in local minima
  - Cannot handle underconstrained parts
    - Free z-axis
    - Tolerances for roll/pitch/yaw



Integrating Descartes

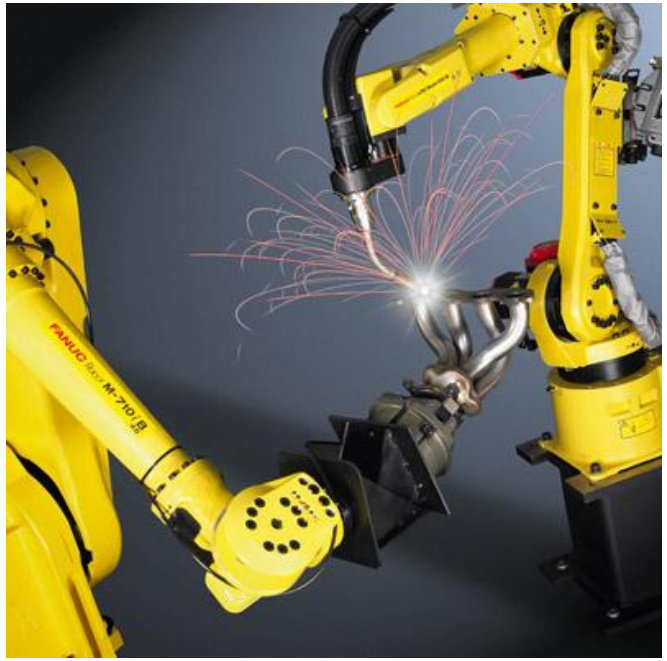
- My current project with SwRI: integrate MoveIt! and Descartes
  - Descartes: planner for under-defined Cartesian trajectories
- Project Goal: plan from

  current state → free space path→ Cartesian path →
  free space path → goal state
  - Approach: create one large configuration space with an additional discrete task dimension
  - Requires multi-modal / hybrid planning
- Naive approach: can get stuck at in local minimum (see figure)



Multi-Arm Motion Planning

- Planning a path along a workpiece being held in a secondary arm
  - Additive or subtractive operation
  - Capability to pickup part with one arm and work on it with second
- Treat the two arms as one long kinematic chain with the workpiece as the base frame
- Currently not supported in MoveIt!



Final Thoughts

- Planning in dual arm configuration spaces is hard for sampling-based algorithms
  - 14 DOFs results in very un-smooth trajectories with MoveIt! currently
  - Perhaps planning from experience can help
- Need better GUIs for inputting
  - under constrained Cartesian paths
  - manipulation tasks
  - dual arm tasks
- Questions?

Contact Information

Dave Coleman

PhD Candidate, Correll Lab

Univ of Colorado, Boulder
430 UCB
Boulder, CO 80309
U.S.A.

Phone: 251-463-2345
Email: dave@dav.ee

http://correll.cs.colorado.edu/