Sampling-based Motion Planning

Fast algorithms, probabilistically complete
Approximate connectivity of the state space
Ignore optimality
Diffusion Trees

Iteratively grow a tree of motions

Examples include RRT, EST, SBL, etc...
Implemented in ROS (ompl package)

- Kinematic planning:
  - RRT (Rapidly-exploring Random Trees)
  - LazyRRT (Lazy RRT)
  - EST (Expansive Space Trees)
  - SBL (Single-query Bi-directional Lazy tree planner)
  - pSBL (Parallel SBL)
  - pRRT (Parallel RRT)
  - KPIECE (Kinematic Planning by Interior-Exterior Cell Exploration)
  - LBKPIECE (Lazy Bi-Directional KPIECE)

- Planning with differential constraints:
  - RRT
  - KPIECE

- Other tools:
  - GAIK (Genetic Algorithm for Inverse Kinematics)
Other Needed Components

- Robot Self Filter
  - Remove points from sensor data that correspond to hits on the links the robot can see
  - Remove shadow points caused by seen links

- Collision Map
  - As the arm moves occlusions can be produced
  - Occluded information should not be forgotten
  - [http://www.ros.org/wiki/collision_map](http://www.ros.org/wiki/collision_map)

- Planning Environment
  - Brings all other components together
  - [http://www.ros.org/wiki/planning_environment](http://www.ros.org/wiki/planning_environment)
Interface to Arm Motion Planning

- Nodes that offer motion planning services
  - ompl_planning, sbpl_arm_planner (Ben), chomp_motion_planner (Mrinal)

- The move_arm package
  - Receives user requests that define desired arm positions (constraints)
  - Requests motion plans from a loaded planner (common ROS interface for all types of planners is available)
  - Monitors paths as they are being executed (requests new paths if current ones become invalid)
  - Switched between different planners if needed (not visible to the user)
What we can do now

- Plan among obstacles
  - Even slowly moving obstacles

- Account for path constraints
  - Not spilling odwalla bottles

- Update the kinematic chain of the robot on-the-fly
  - So we avoid collisions between grasped objects and the environment

- Move to positions that cause the arm to touch objects in the environment
  - Useful for grasping

- Safe tele-operation
Next Step: Planning with Dynamics

- Try using physics simulation
  - Account for friction, gravity, limits in forces
  - Needed for more accurate simulation

\[
f(q_t, u_t) = q_{t+1}
\]
Thank you!

(videos and demo in the green room)
In ROS Packages

- Perception
  - robot_self_filter
  - collision_map
- Robot model
  - planning_models
- Collision checking
  - collision_space
- Motion planning
  - planning_environment
  - ompl, ompl_ros, ompl_search, ompl_planning
  - move_arm