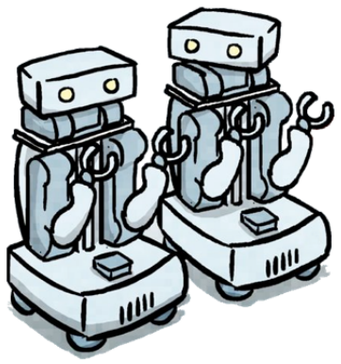


Open Source Solutions for Motion Planning

Sachin Chitta

This talk includes contributions from the entire ROS and PR2 development teams at Willow Garage (including several interns) and the PR2 Beta Program and ROS community in general.

External collaborators/contributors:
Maxim Likhachev, CMU
Lydia Kavraki, Rice University

Outline

- The PR2 robot
- ROS
  - tools for simulation
  - motion planning
    - navigation and manipulation
  - task execution
    - several examples and movies
  - applications
- Future directions

About Us

- Willow Garage - Menlo Park, CA, USA
  - 60 full-time employees

Picture courtesy: Armin Hornung, Freiburg

  - Interns and visiting researchers - Freiburg, TUM, CMU, Penn, UNC, Stanford, GaTech.....

PR2

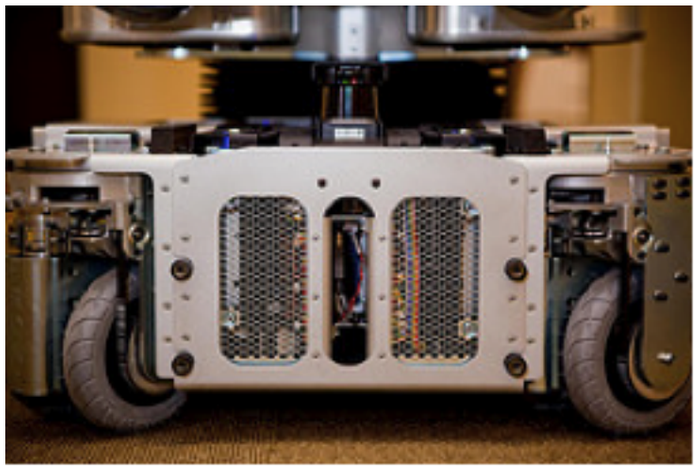
- PR2 - Personal Robotic System

PR2 Beta Program

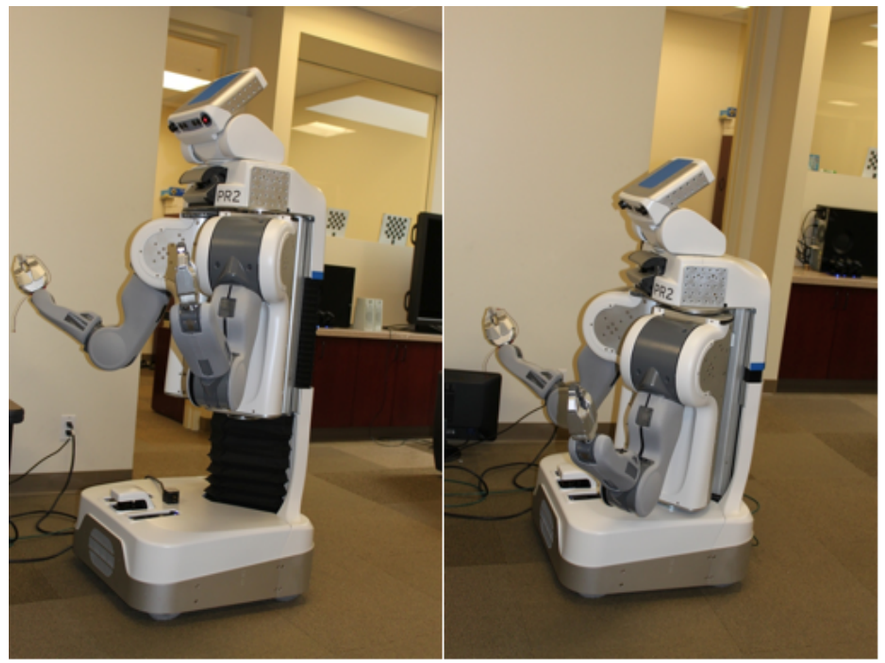
- 11 robots to universities and industry
- 2 year program - research and develop state of the art techniques for personal robotics

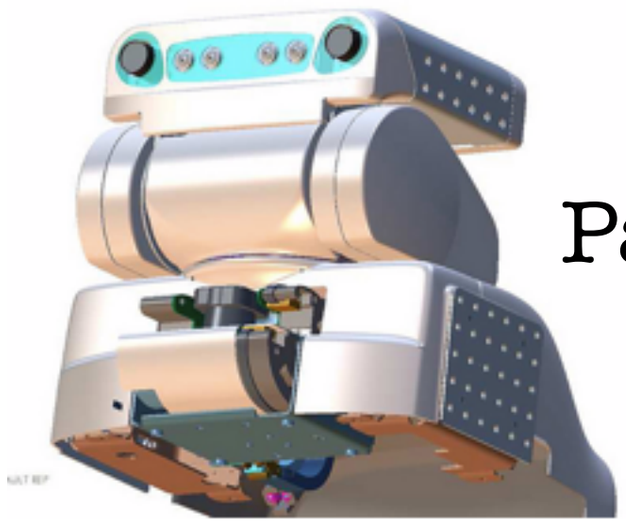
PR2

Omnidirectional base



Telescoping spine



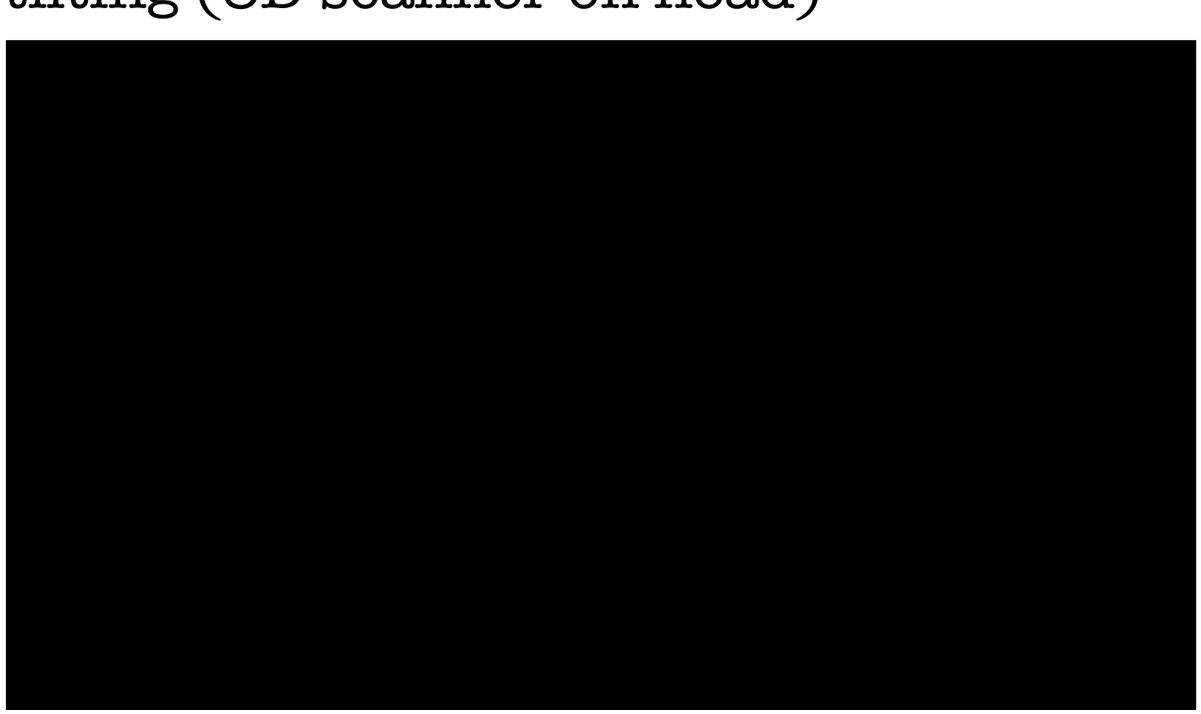


Pan and Tilt head

Willow Garage

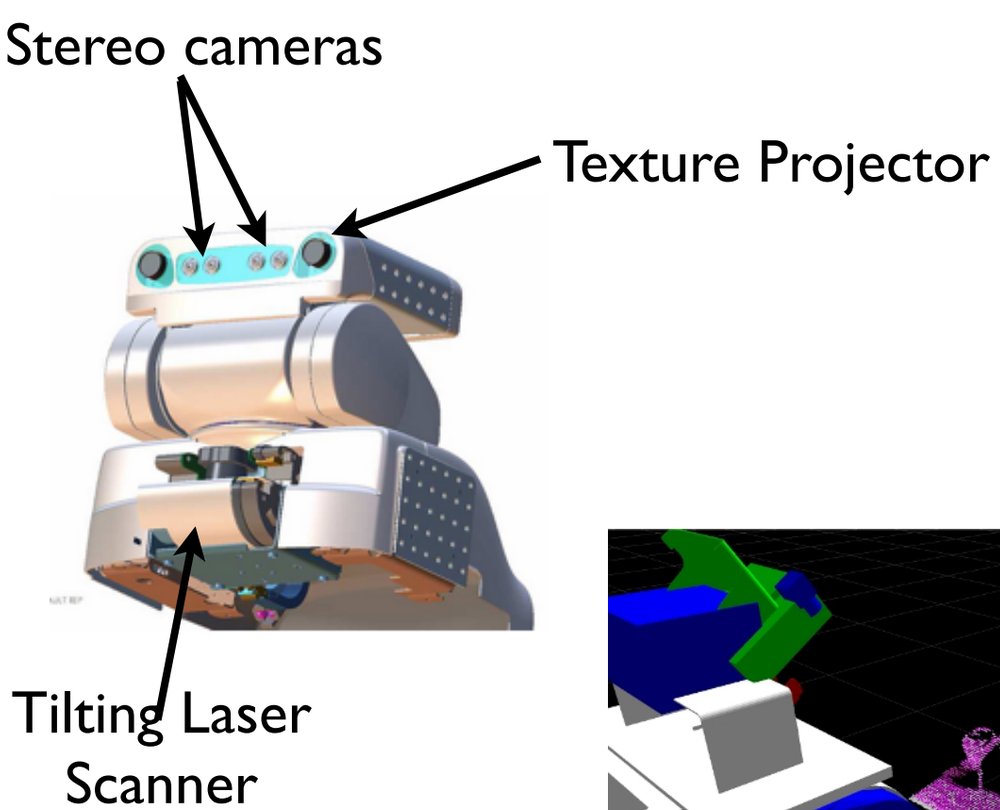
Sensing on the PR2

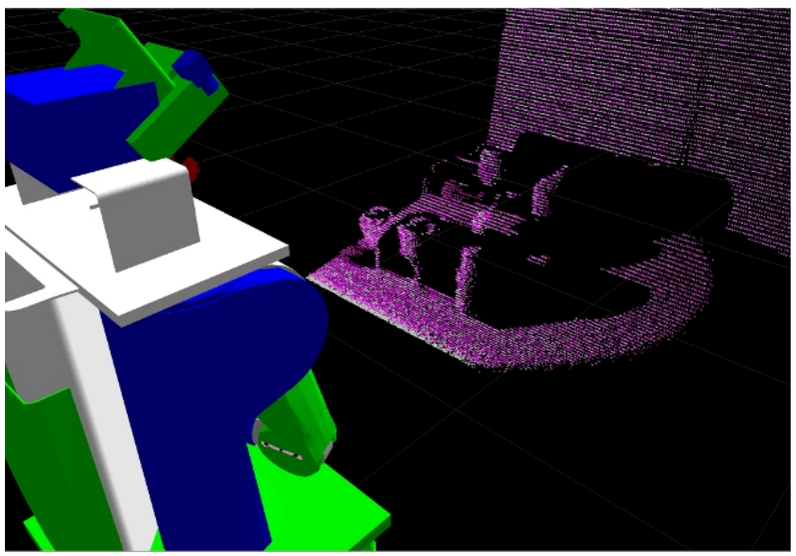
- Sensors for navigation
  - IMU
  - Laser scanner
    - base (planar)
    - tilting (3D scanner on head)

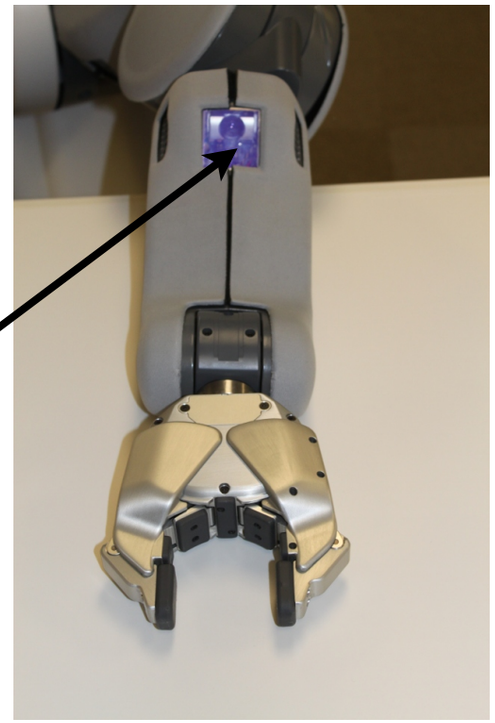


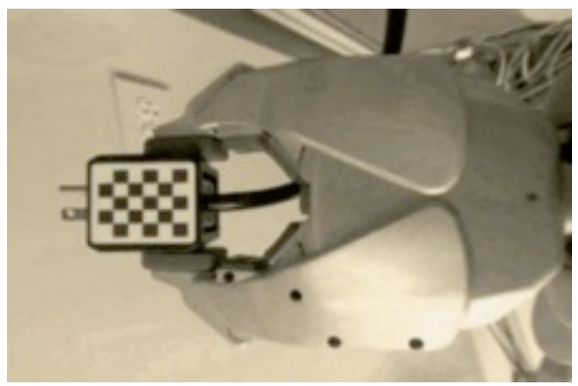
Sensing on the PR2

- Sensors for manipulation







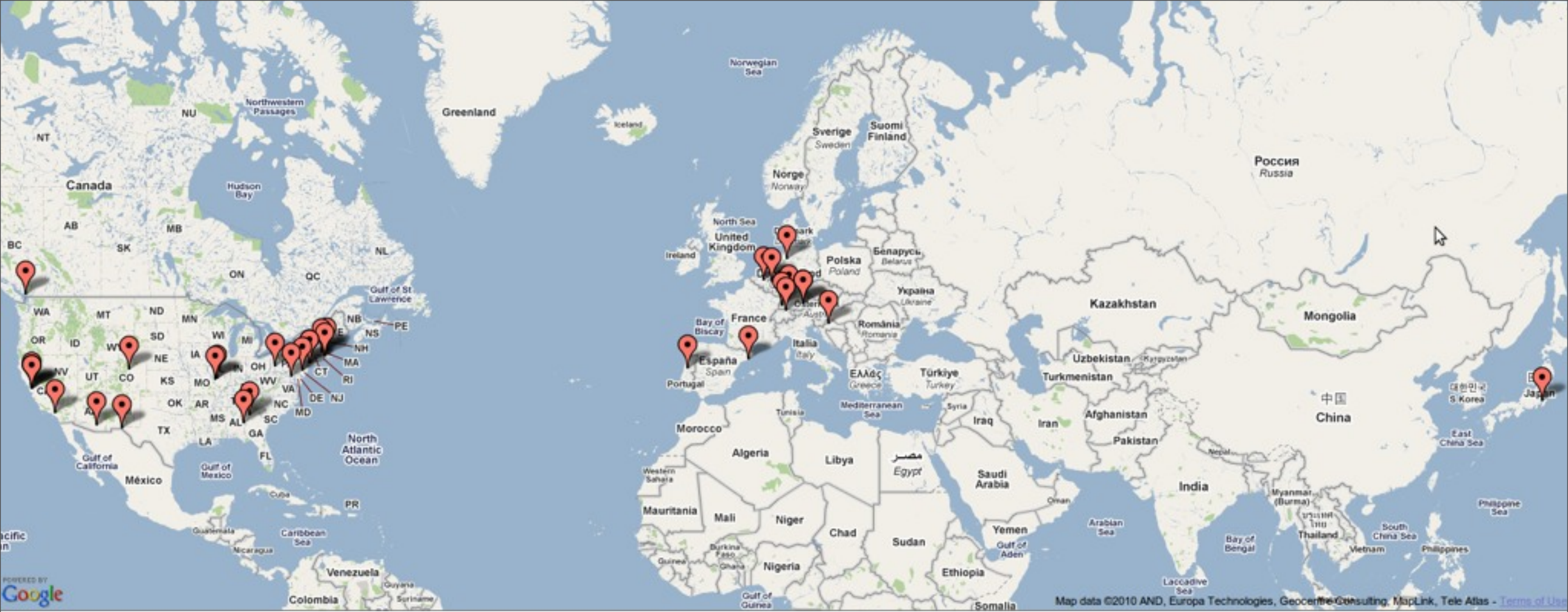


Computing power

- 2 servers
  - 8 cores each
  - 20 GB Memory
  - External hard drives for data storage

Outline

- The PR2 Robot
- ROS
  - tools for simulation
  - tools for motion planning
  - task execution
  - applications
- Future Directions



STANFORD UNIVERSITY · Colorado University of Colorado at Boulder · Willow Garage · Care-O-bot Research · Carnegie Mellon

BOSCH · Cornell University · Washington University in St. Louis · The University of Texas at Austin · Uni Freiburg · Technische Universität München

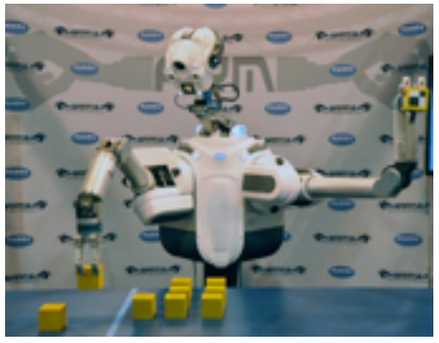
+ many more!

University of Maryland · Georgia Tech · Penn · Columbia University · Katholieke Universiteit Leuven · The City College of New York · Deutsches Forschungszentrum für Künstliche Intelligenz GmbH · Shadow Robot Company · REC

Brown · MIT · USC University of Southern California · Caltech · WPI · 東京大学 The University of Tokyo · University of California · UBC · Rice

ROS

- Number of robots with ROS on them keeps growing



Resources: http://www.ros.org/wiki/robots

ROS

- Over 100 repositories
- Over 2000 packages!!
- Lots of code reuse
  - navigation
  - localization
  - mapping
  - motion planning
  - state machines
  - perception
  - drivers

1. wg-ros-pkg
2. prairiedog (CU)
3. utexas-art-ros-pkg
4. ccny-ros-pkg
5. alufr-ros-pkg
6. ua-ros-pkg
7. umd-ros-pkg
8. dfki-sks-ros-pkg
9. iheart-ros-pkg
10. gt-ros-pkg
11. openrobotino
12. bosch-ros-pkg
13. tum-ros-pkg
14. wu-ros-pkg
15. care-o-bot
16. kul-ros-pkg
17. berkeley-ros-pkg
18. brown-ros-pkg
19. sail-ros-pkg
20. cmu-ros-pkg
21. mit-ros-pkg
22. ros-engagement (WPI)
23. mod-ros-pkg (Penn)
24. usc-ros-pkg
25. cu-ros-pkg
26. amor-ros-pkg
27. rice-ros-pkg
28. sr-ros-interface
29. jsk-ros-pkg
30. isr-uc-ros-pkg
31. auburn-automow
32. RCPRG-ros-pkg
33. aptima-ros-pkg
34. ubc-ros-pkg
35. siue-ros-pkg
36. vanadium-ros-pkg
37. hwu-osl-ros-pkg
38. seabee3-ros-pkg
39. cwru-ros-pkg
40. flyatar
41. ethz-asl
42. vmi-ros-pkg
43. ais-bonn-ros-pkg
44. pi-robot
45. stanford-wbc
46. clearpath-ros-pkg
47. uuisrc-ros-pkg
48. albany-ros-pkg
49. acin-tuwien
50. webots-ros-pkg
51. starmac-ros-pkg
52. wpi-ros-pkg
53. ssc-rovers-ros-pkg
54. csiro-asl-ros-pkg
55. tuc-ros-pkg
56. cornell-ros-pkg

ROS - Evolution

- Navigation
- Logging and playback (rosbag, rxbag)
- Image/laser pipelines
- OpenCV Integration
- Simulators (gazebo,stage)
- Visualizers (rviz, nav_view)
- Hardware drivers

- Arm navigation (initial release)
- Grasping pipeline (initial release)
- Task coordination (SMACH)
- Calibration
- Firewire drivers
- Nodelets
- Collada

- Lightweight ROS
- Point Cloud Library
- SMACH (1.0)
- OpenNI/Kinect

Willow Garage

ROS

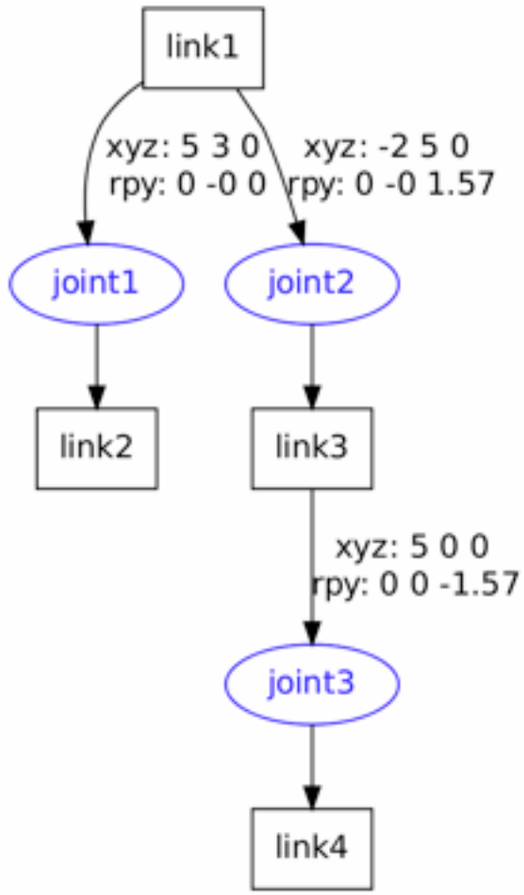
- Transport infrastructure/framework
- Tools
  - navigation
  - control
  - sensing
  - perception
  - motion planning
  - simulation

How can you start using ROS for motion planning?

Robot Description

- URDF (Universal Robot Description Format)
  - ✓ serial manipulators
  - ✓ URDFs already available for different robots

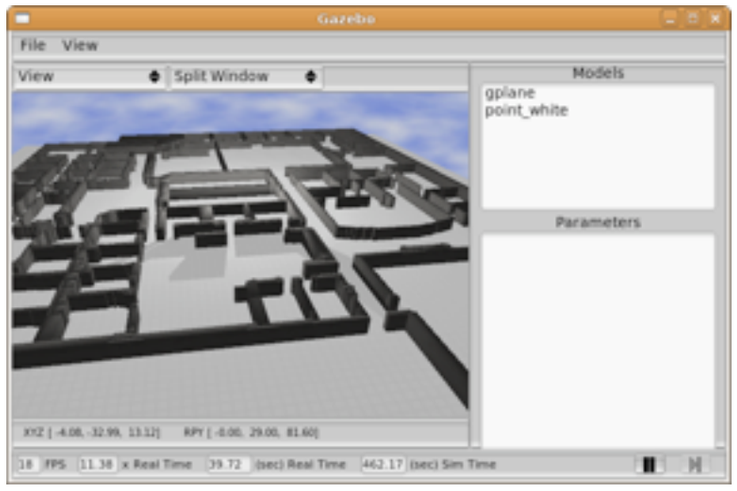
```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
    <origin xyz="5 3 0" rpy="0 0 0" />
    <axis xyz="-0.9 0.15 0" />
  </joint>

  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
    <origin xyz="-2 5 0" rpy="0 0 1.57" />
    <axis xyz="-0.707 0.707 0" />
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
    <origin xyz="5 0 0" rpy="0 0 -1.57" />
    <axis xyz="0.707 -0.707 0" />
  </joint>
</robot>
```

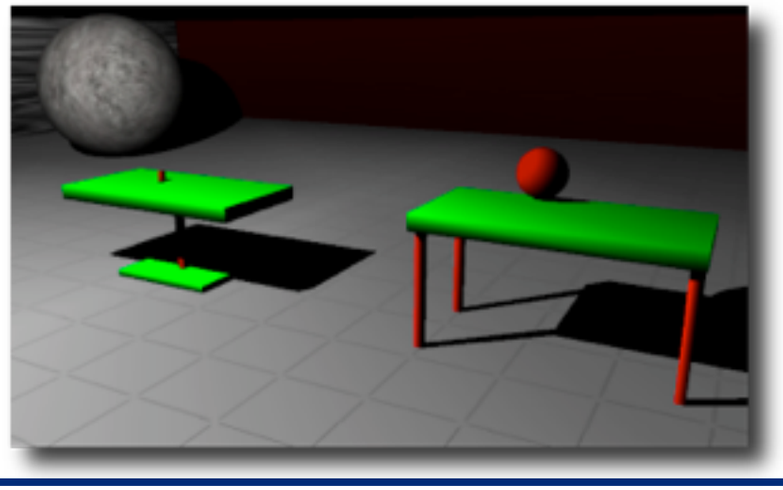


1. www.ros.org/wiki/urdf
2. www.ros.org/wiki/robots

Simulator

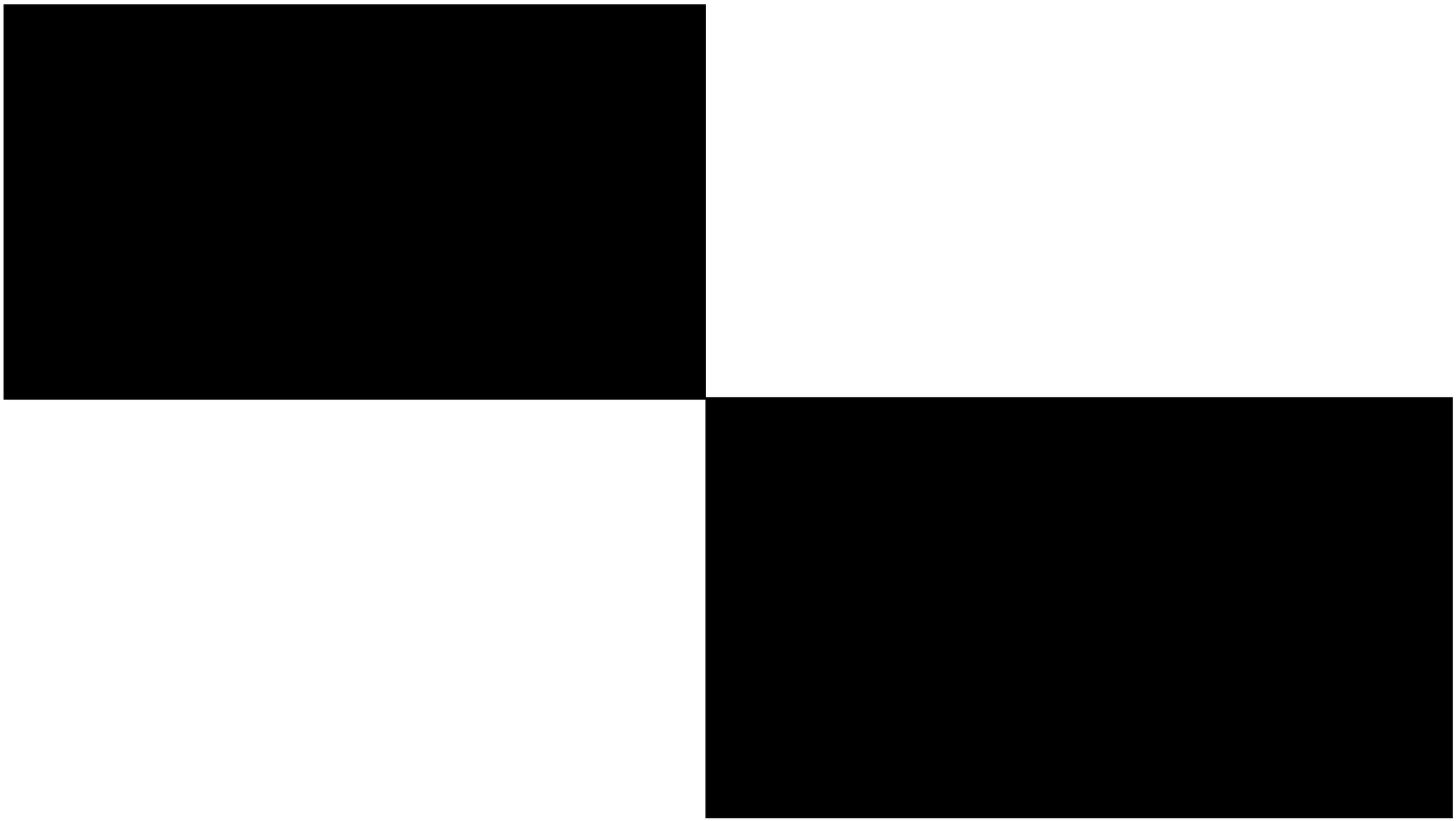
- Gazebo
  - ❖ now maintained actively by Willow Garage
  - ❖ using the URDF and ROS or through config files
    - ✓ add robots
    - ✓ add objects
    - ✓ add sensors
    - ✓ add maps





http://www.ros.org/wiki/simulator_gazebo

Simulator



Outline

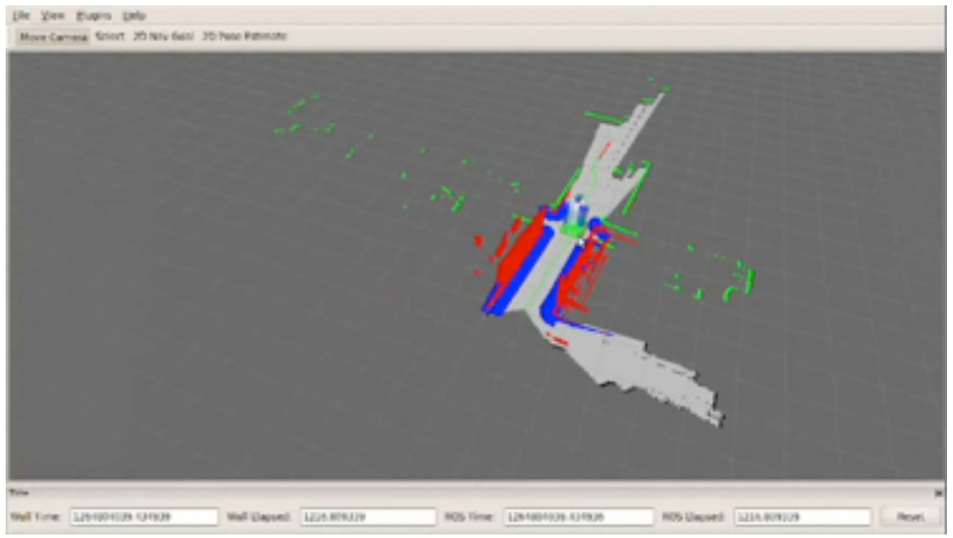
- The PR2 robot
- ROS
  - tools for simulation
  - tools for motion planning
    - navigation
  - task execution
  - applications
- Future Directions

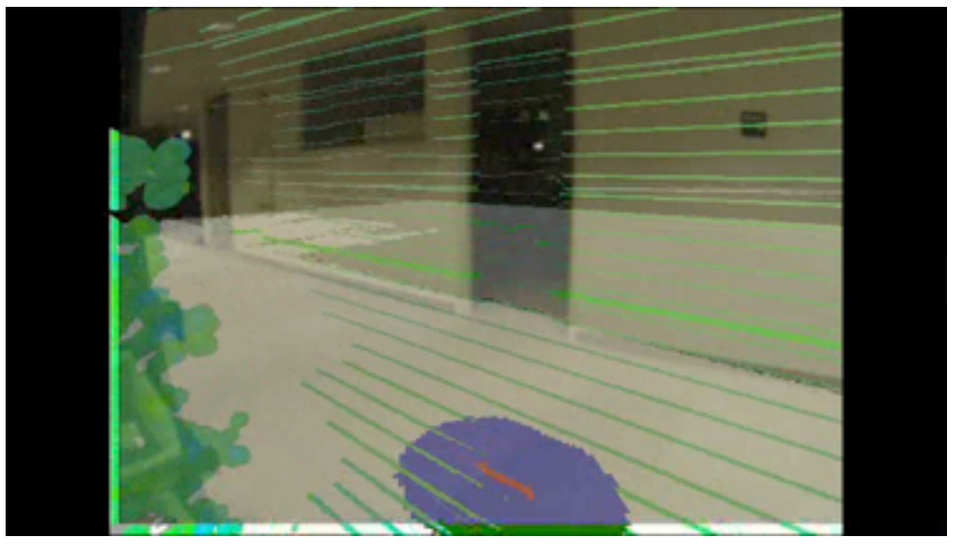
Navigation

- Need
  - ❖ representation of robot
    - ✓ footprint - 2D projection of robot onto ground plane
  - ❖ representation of environment
  - ❖ global motion planner
  - ❖ local motion planner
  - ❖ controller

Navigation

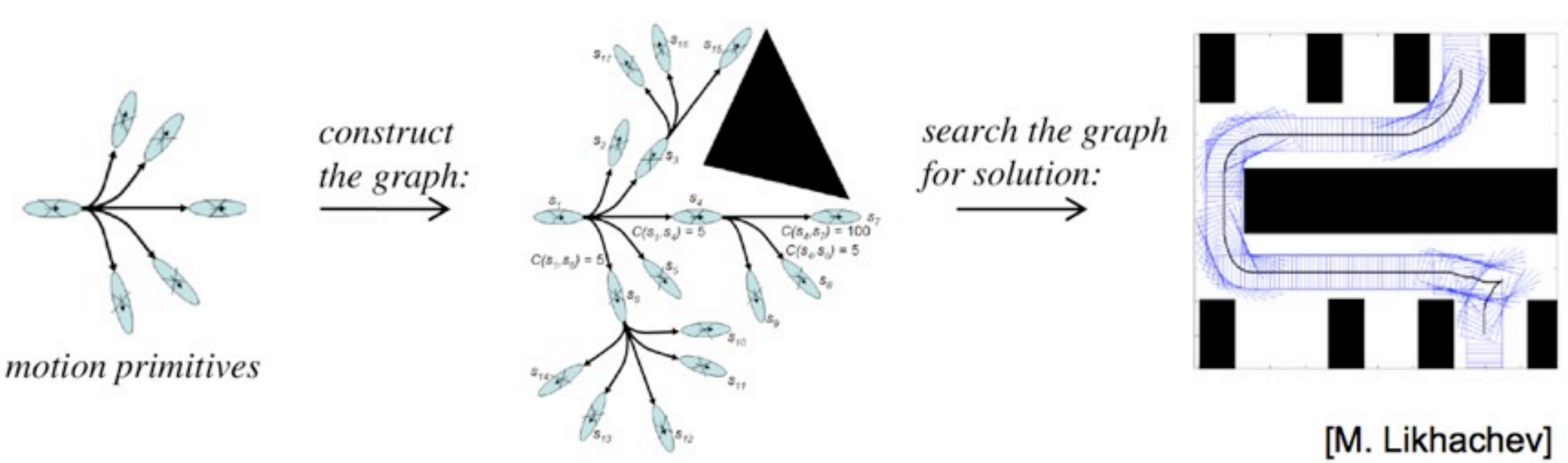
- Environment Representation
  - ❖ Dealing with 3D obstacles
  - ❖ Voxel grid (stored compactly)
- Global planner
  - ✓ Djikstra's, A*
  - ✓ plans in x,y space
  - ✓ assumes circular robot
- Local planner
  - ✓ trajectory rollout
  - ✓ lays out full footprint of the robot





Lattice based planner

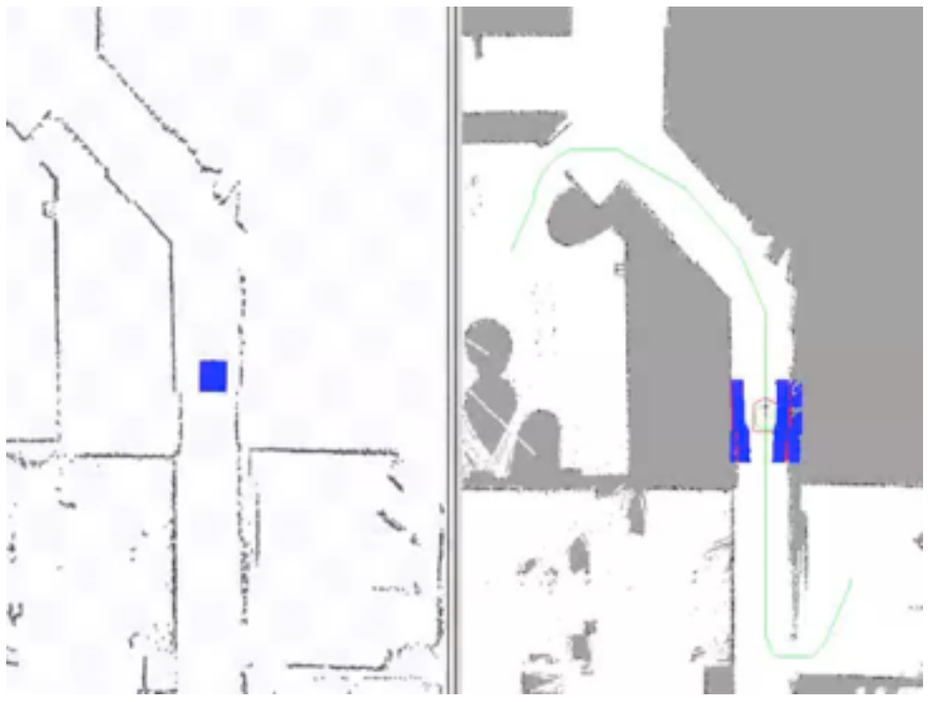
- 2D global planner can produce infeasible paths
  - nice to have better paths for local controller to follow
  - take into account constraints/dynamics in global planning as well



[M. Likhachev]

Navigation

- Discretize orientation as well
  - ability to plan in orientation as well as x,y
  - better trajectories that are easier to follow for local planner

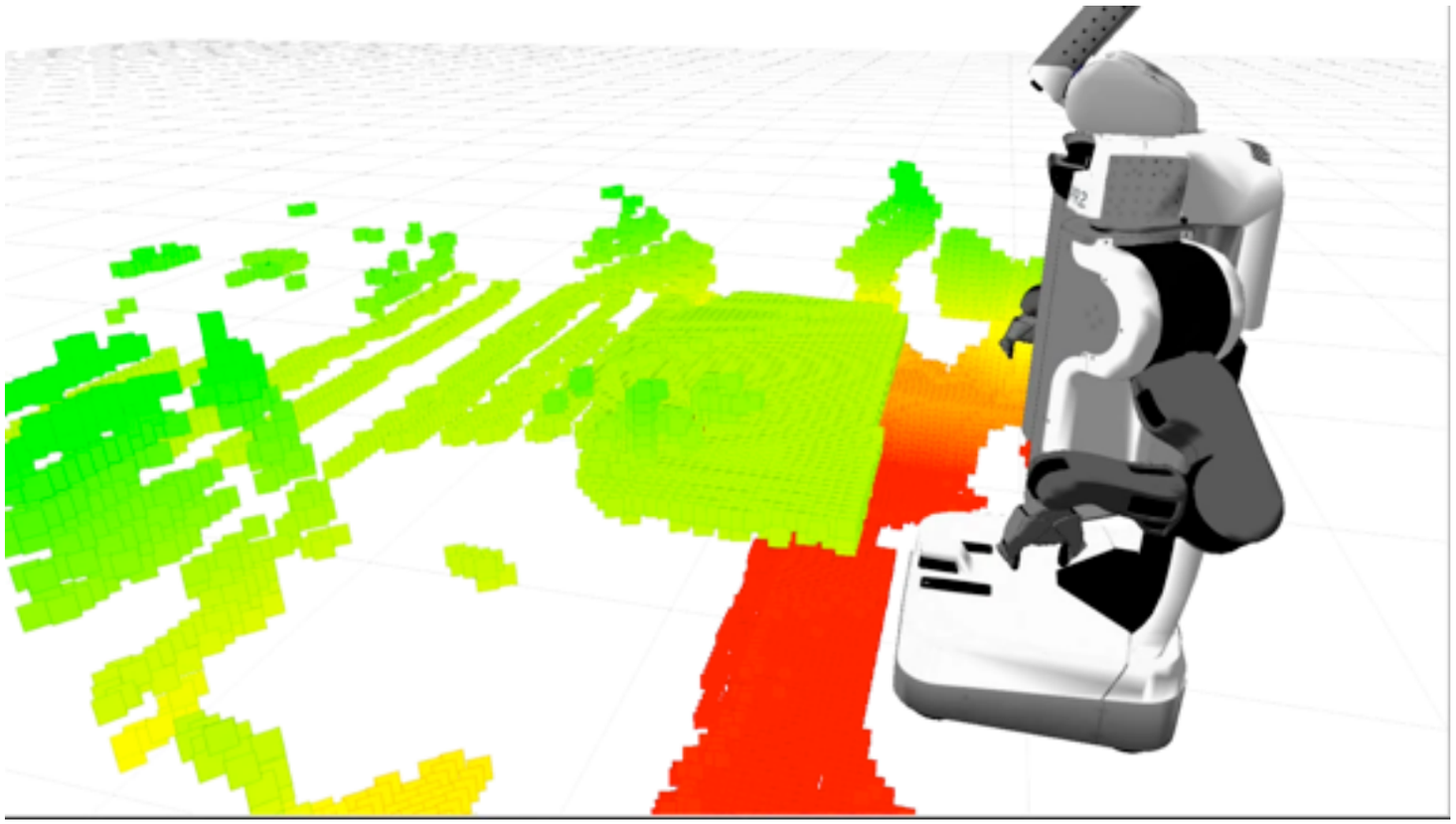


Lattice planner - http://www.ros.org/wiki/sbpl_lattice_planner
Search based planning library - http://www.ros.org/wiki/sbpl
- developed by Maxim Likhachev's group at CMU in collaboration with Willow Garage

Navigation - extension

- Used 3D data projected onto a 2D costmap
  - problems with tables, overhangs etc. in environment
  - inability to plan with arms not tucked
- Collision checking of 3D robot model against 3D model of environment
  - need compact representation of environment

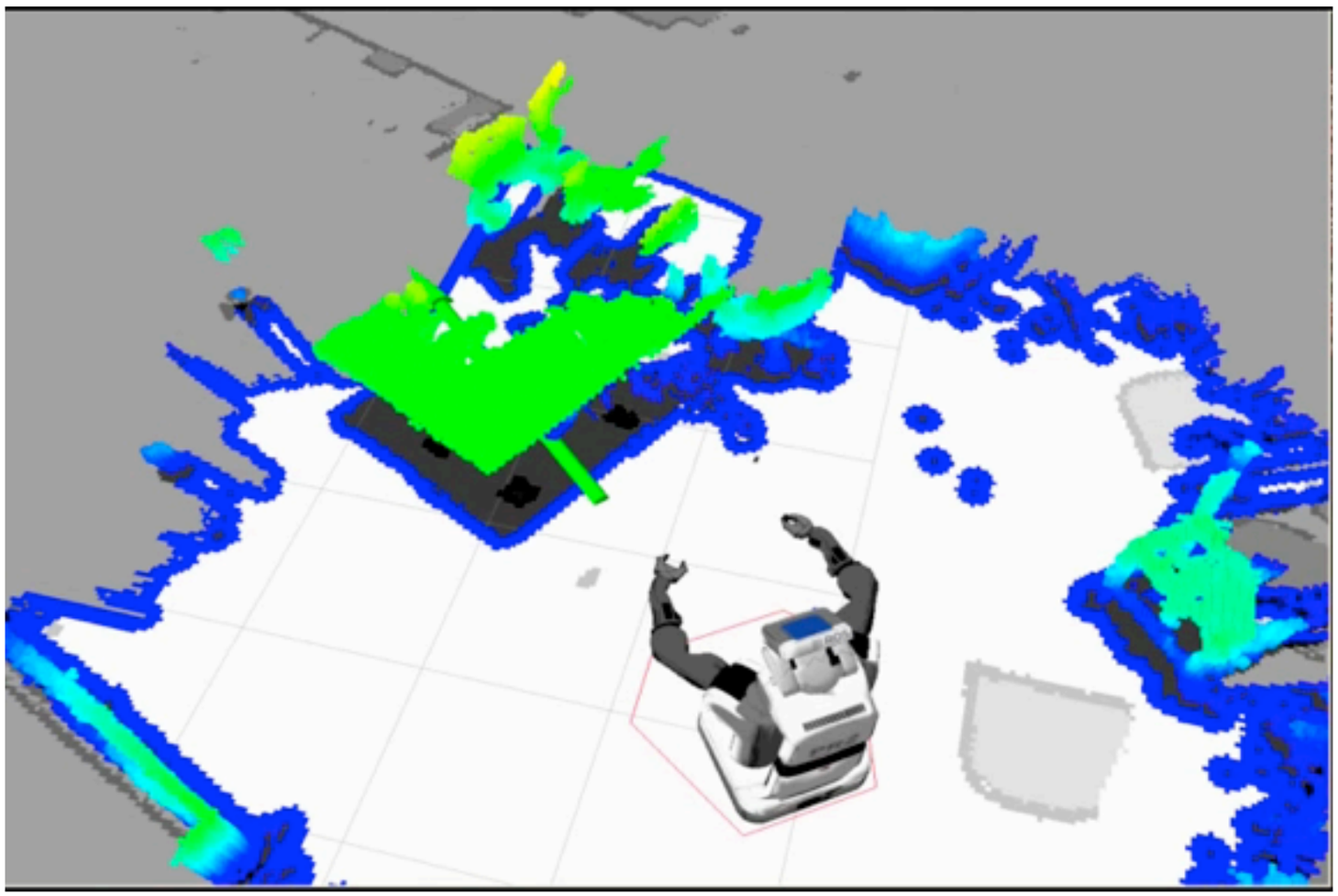
Octomap



- Octree-based representation of the environment
  - ❖ Probabilistic, flexible, and compact 3D mapping
  - ❖ Optimized for online operation

Octomap - http://octomap.sf.net
Armin Hornung, Kai Wurm, et. al., Freiburg

“3D” Navigation



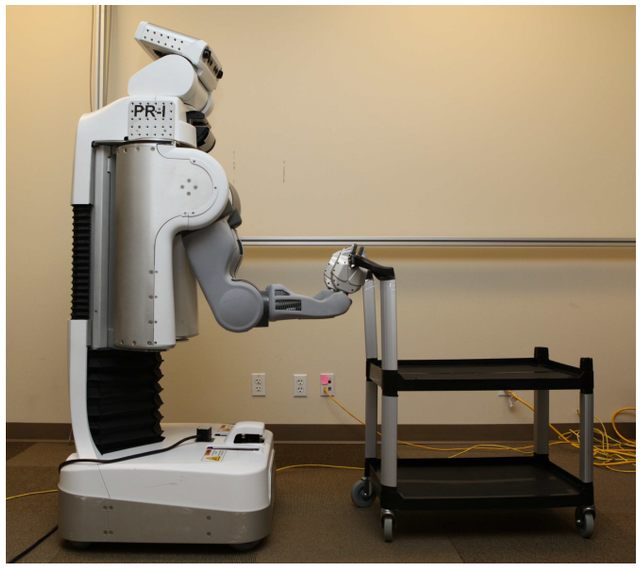
- Octomap + lattice based planner
  - ❖ full 3D collision checking only when necessary
  - ❖ allows general navigation in unstructured environments (even with arms outstretched)

Applications - Long term autonomy

- 13 days, 2 hours
- 138.9 km
- 2 manual interventions
  - robot emails for help
    - get around a chair
    - re-localization
    - (both issues resolved over the web)

Applications

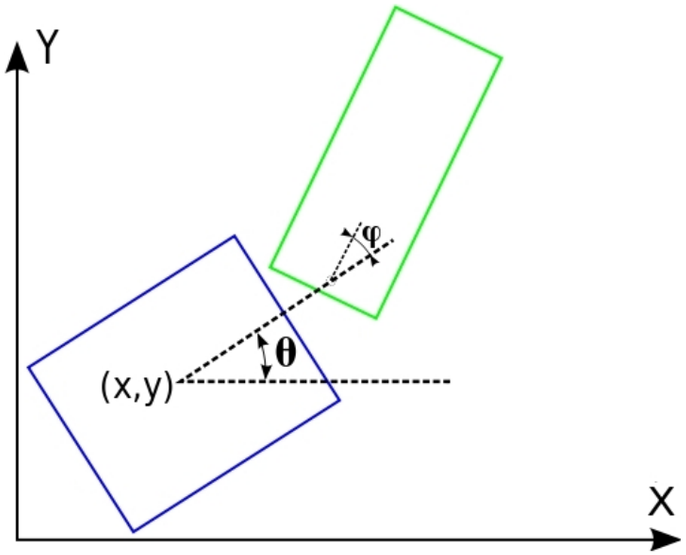
- Navigation with moveable objects
  - need to move large objects around
  - robots limited in their payload capacity
    - e.g. PR2 has a maximum 5 lb. payload
    - can carry only one object in each hand



- Example tasks
  - carry a set of dishes to the kitchen
  - arrange chairs in a room

Motion Planning

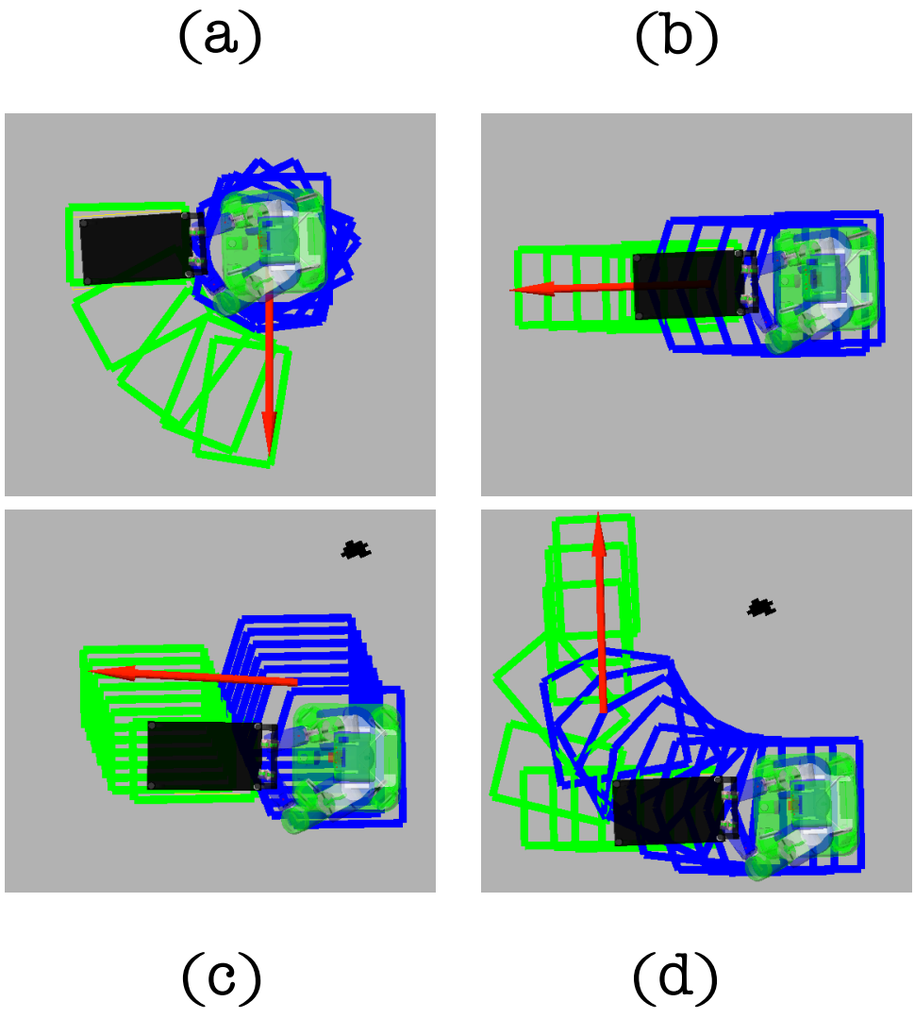
- Given a start and goal, generate a collision free path that takes the robot (and cart) from start to goal
  - State representation
    - ❖ whole body planning too expensive
    - ❖ choose to work in lower-dimensional space
      - ✓ x,y position of the robot base
      - ✓ orientation of the base
      - ✓ articulated angle for the cart (relative to robot)



  - cart motion restricted to pivoting about a center of rotation in front of robot

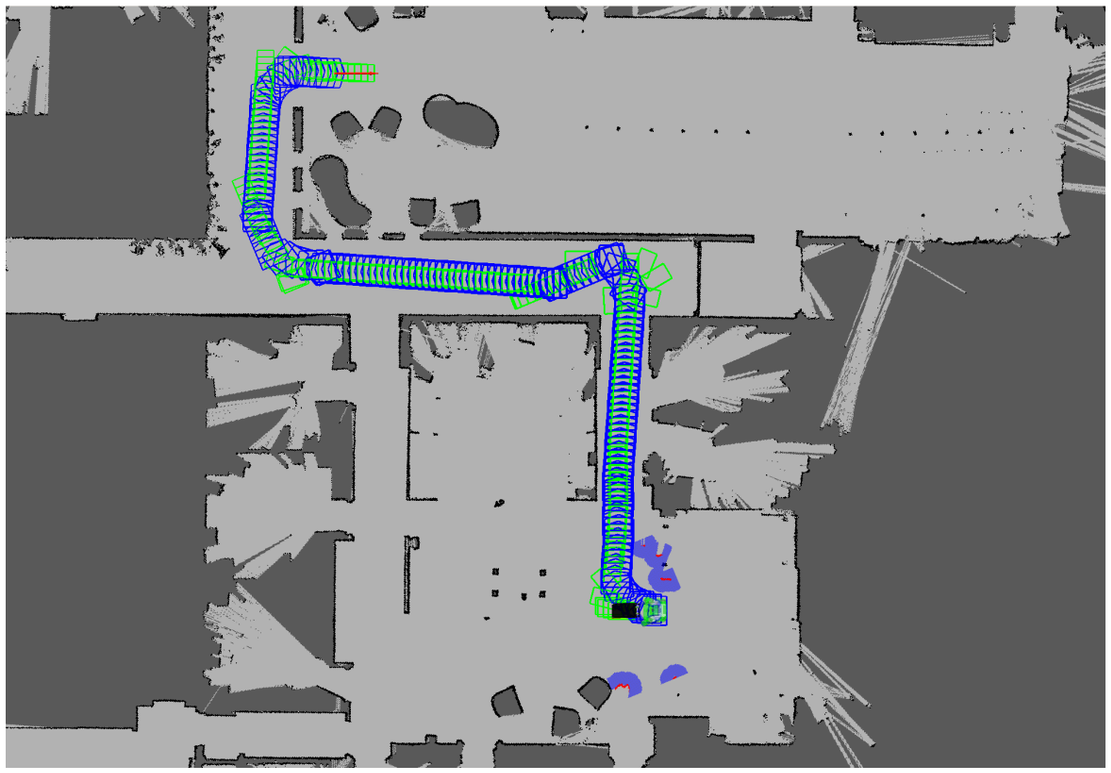
Motion Primitives

- Rotation, forward, sideways (a), (b), (c)
- Articulated motion primitive (d)
  - designed for going around 90 degree corners
  - cart angle starts and ends at 0 degrees



Experiments

- Run the robot + cart in Willow Garage office building
  - plan continuously between multiple waypoints on the map
  - map generated offline (a priori)



Experiments

Environment Uncertainty

Some surprises along the way!!

Outline

- The PR2
- ROS
  - tools for simulation
  - tools for motion planning
    - manipulation
  - task execution
  - applications
- Future Directions

Manipulation

- Motion planning
  - sensing for motion planning
    - semantic perception
  - collision environment
  - system architecture
  - motion planners
    - applications
  - safer control

Manipulation

- Motion planning
  - sensing for motion planning
    - semantic perception
  - collision environment
  - system architecture
  - motion planners
    - applications
  - safer control

Manipulation

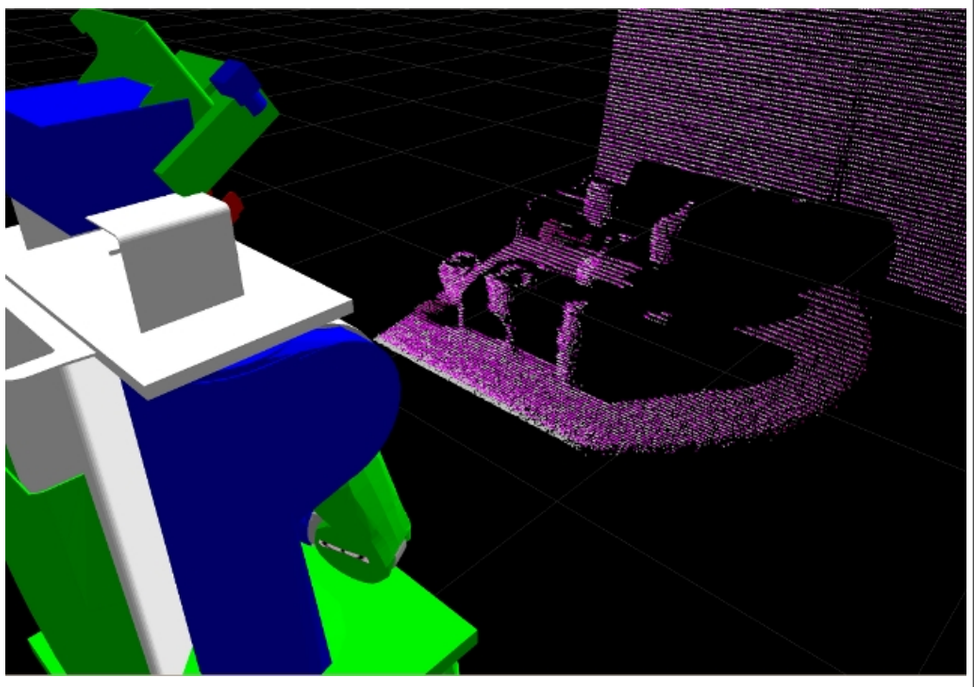
- Motion planning
  - sensing for motion planning
    - semantic perception
  - collision environment
  - kinematics
  - motion planners
  - smoothers
  - monitoring and control

Manipulation

- Motion planning
  - sensing for motion planning
    - ❖ semantic perception
  - collision environment
  - kinematics
  - motion planners
  - smoothers
  - monitoring and control

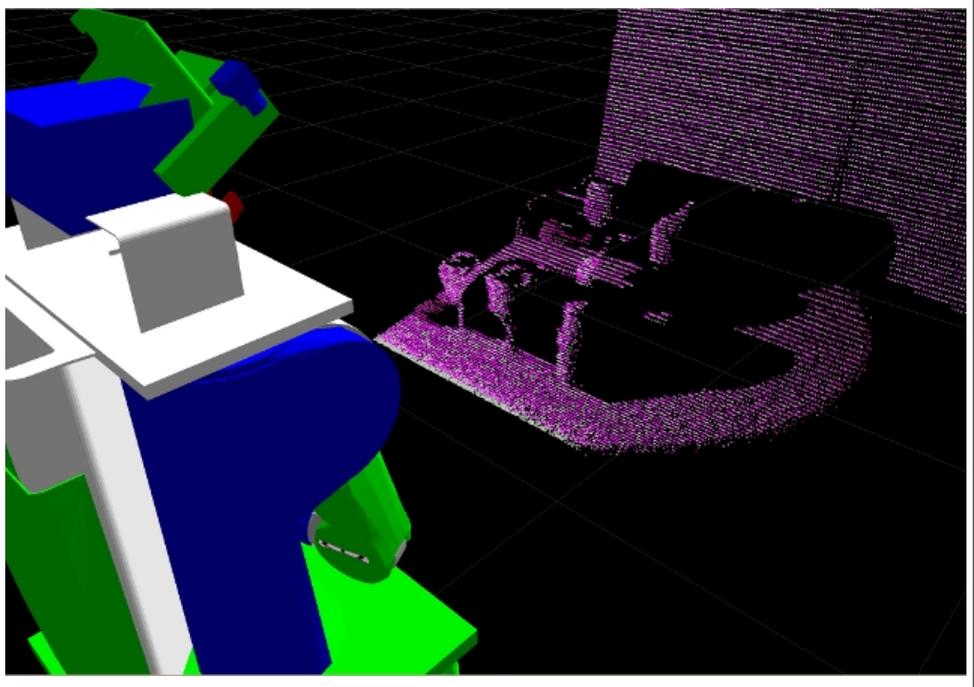
Sensing

- Essential for operation in unstructured environments
- Sensor input
  - point cloud (from stereo, lasers, etc.)
    - collection of 3D points
    - can annotate other information
      - ✓ RGB color, intensity values
    - stereo (20-25,000 points)
    - laser sensor (5-7,000 points)



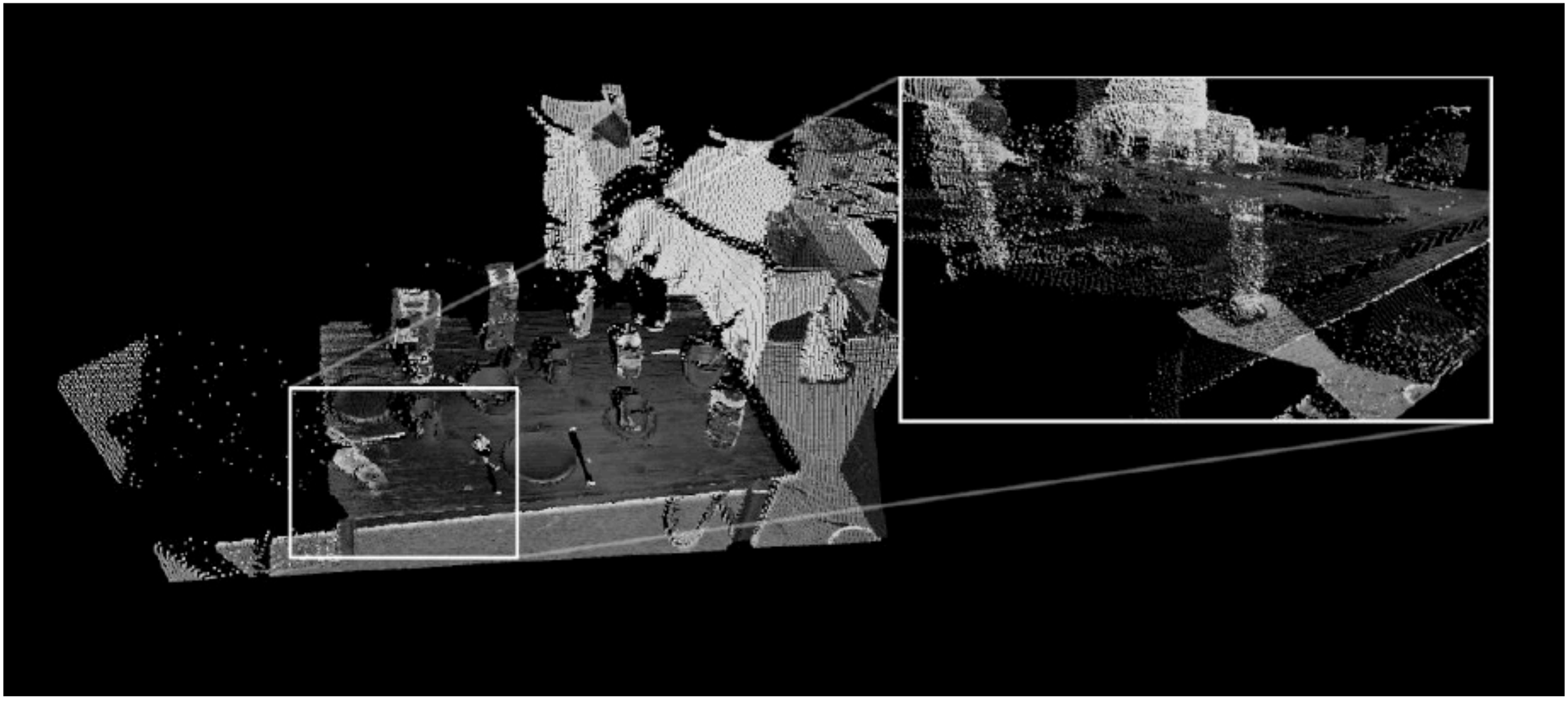
Sensing

- Essential for operation in unstructured environments
- Sensor input
  - point cloud (from stereo, lasers, etc.)
    - ❖ collection of 3D points
    - ❖ can annotate other information
      - ✓ RGB color, intensity values
    - ❖ stereo (20-25,000 points)
    - ❖ laser sensor (5-7,000 points)



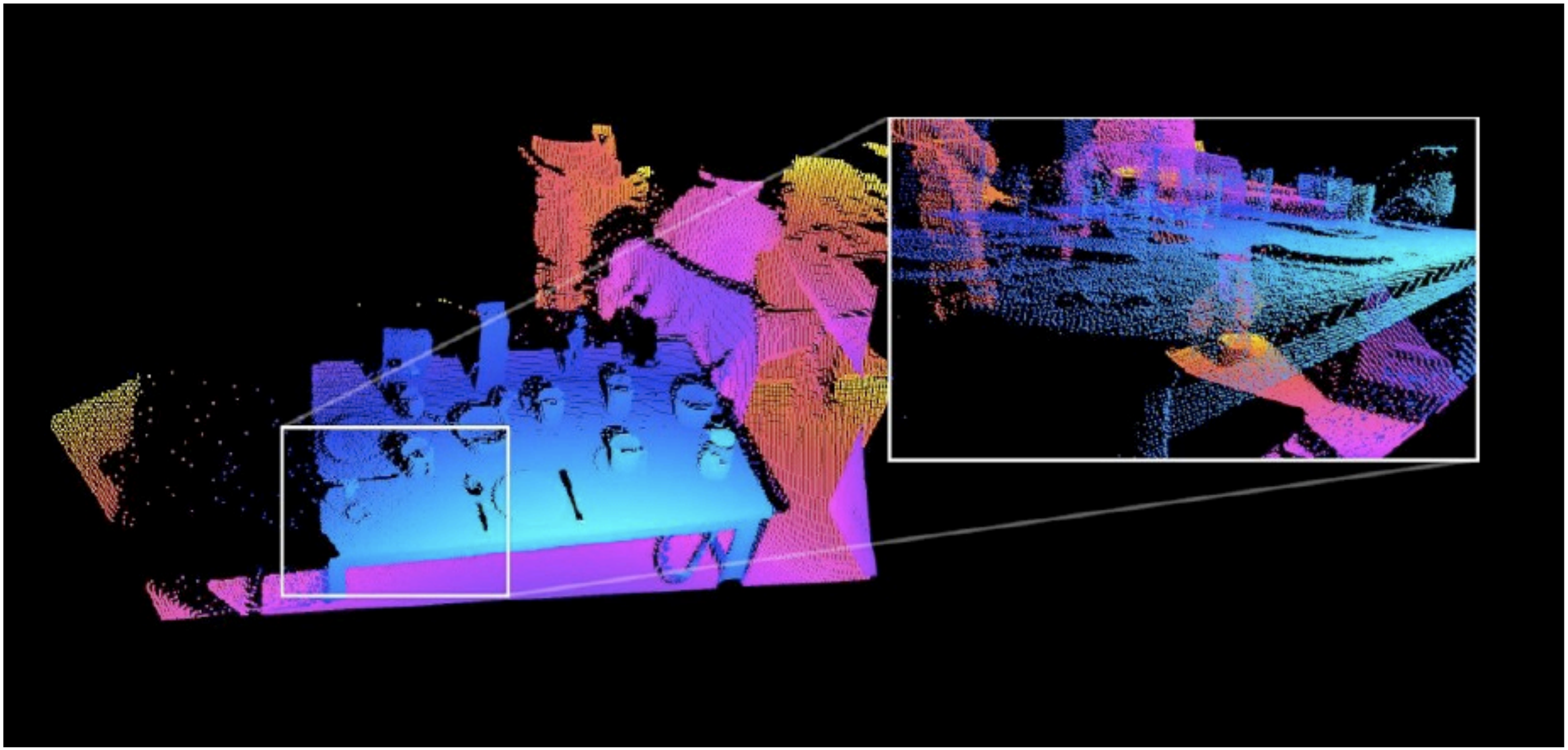
Semantic Perception

Extracting semantic information from point clouds



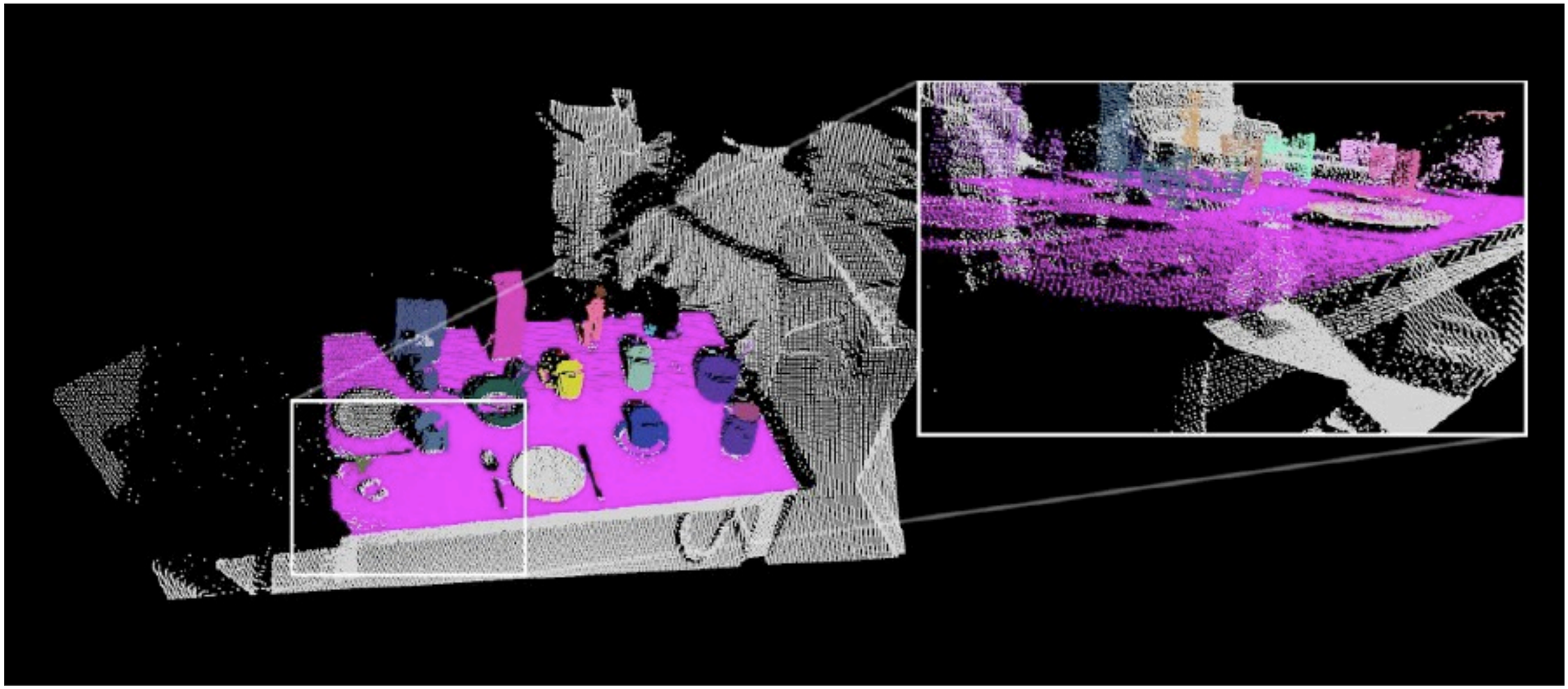
Semantic Perception

Extracting semantic information from point clouds



Semantic Perception

Extracting semantic information from point clouds

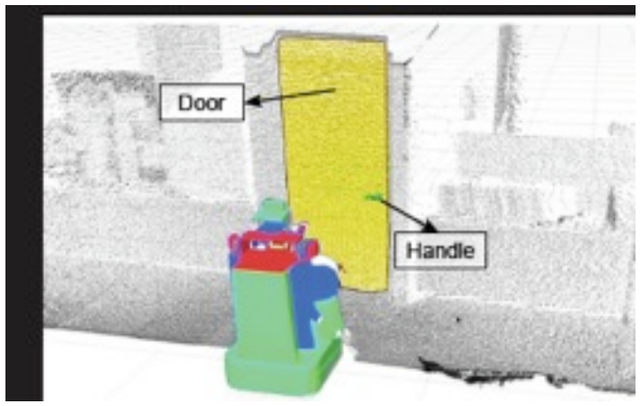


Semantic Perception

- Helps in
  - ❖ object recognition
  - ❖ object representation for collision checking
  - ❖ place recognition
  - ❖ task planning

✓ place/pickup locations for objects

✓ constrained planning

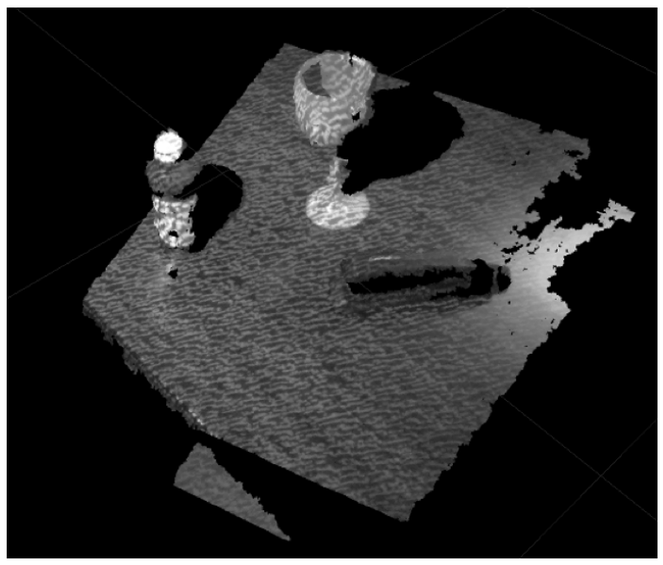


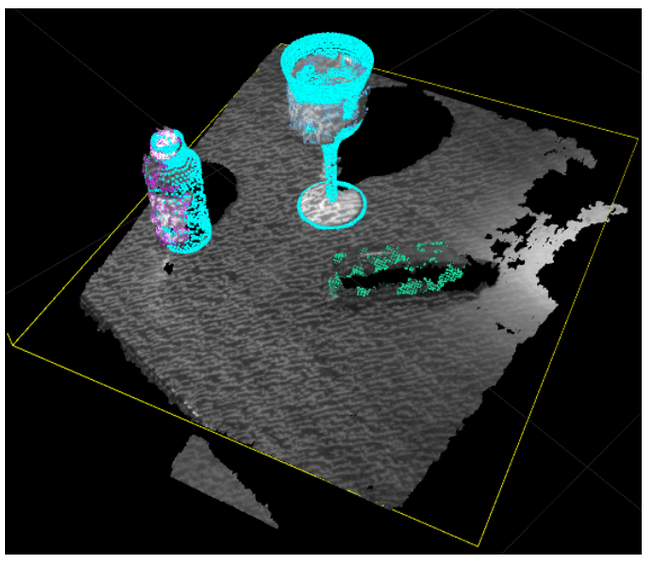
Manipulation

- Motion planning
  - sensing for motion planning
    - semantic perception
  - collision environment
  - kinematics
  - motion planners
  - smoothers
  - monitoring and control

Collision Environment

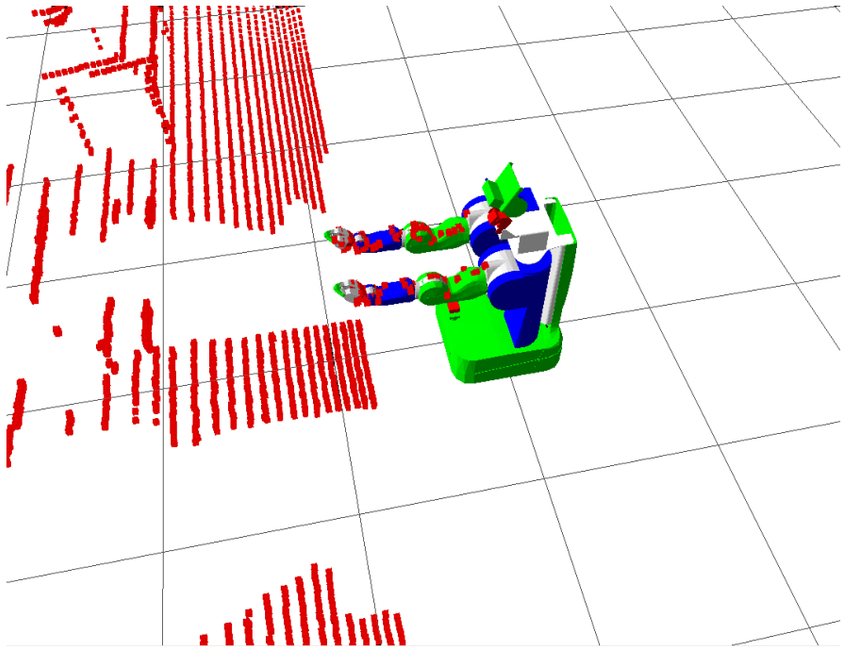
- From point clouds to collision representation
  - meshes for known objects
  - each point represented as box/sphere primitive

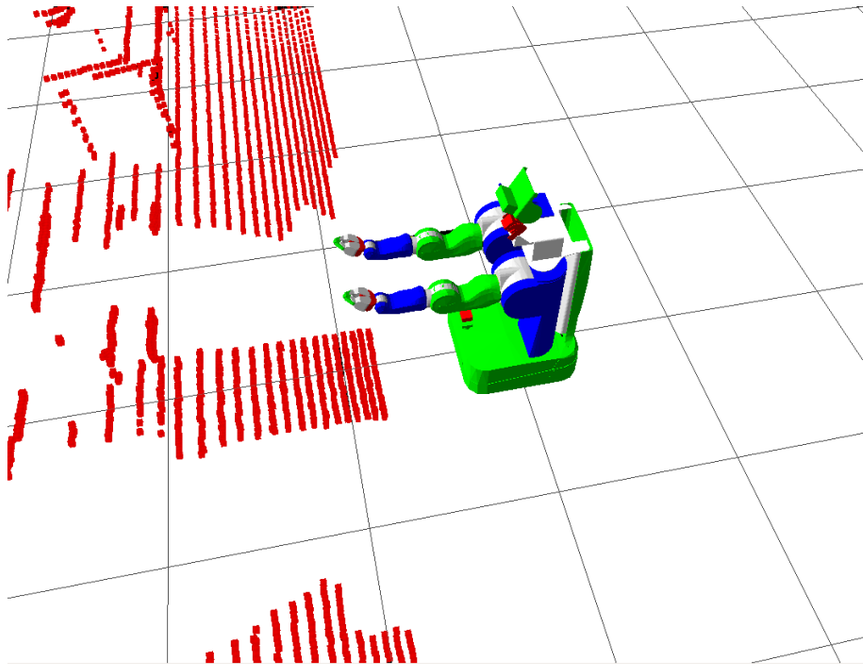




Collision Environment

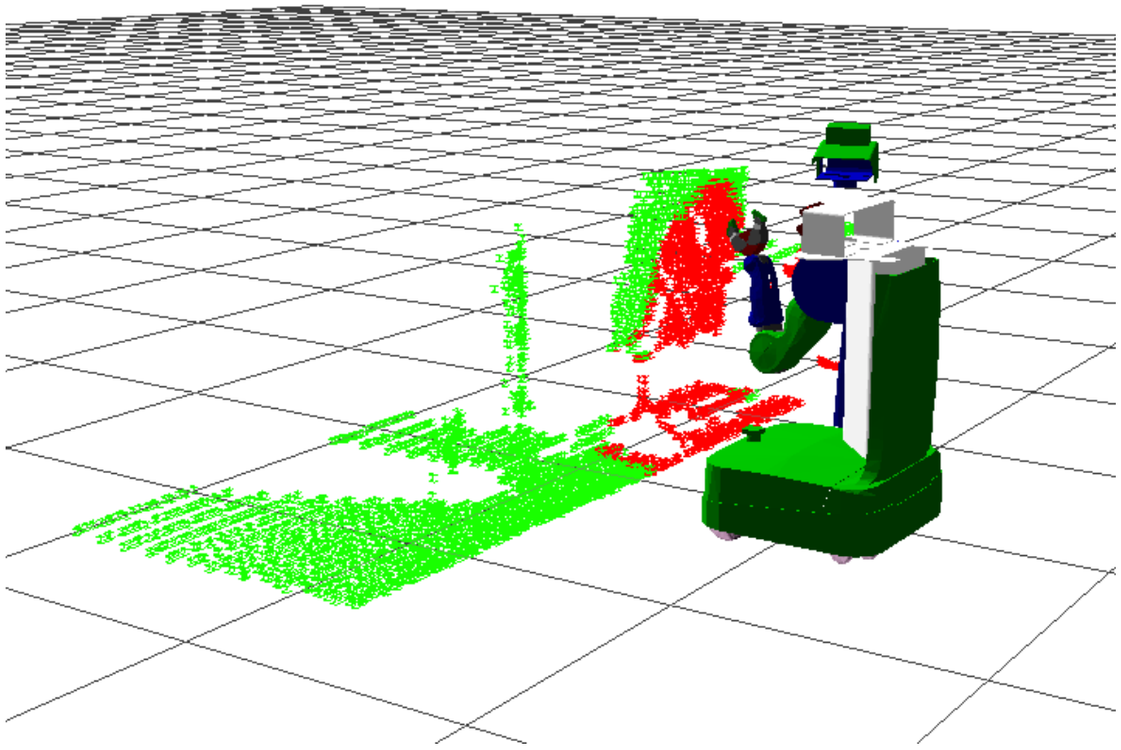
- Sensor data -> Collision map
  - 3D Occupancy grid
  - Filter robot parts out of sensor streams



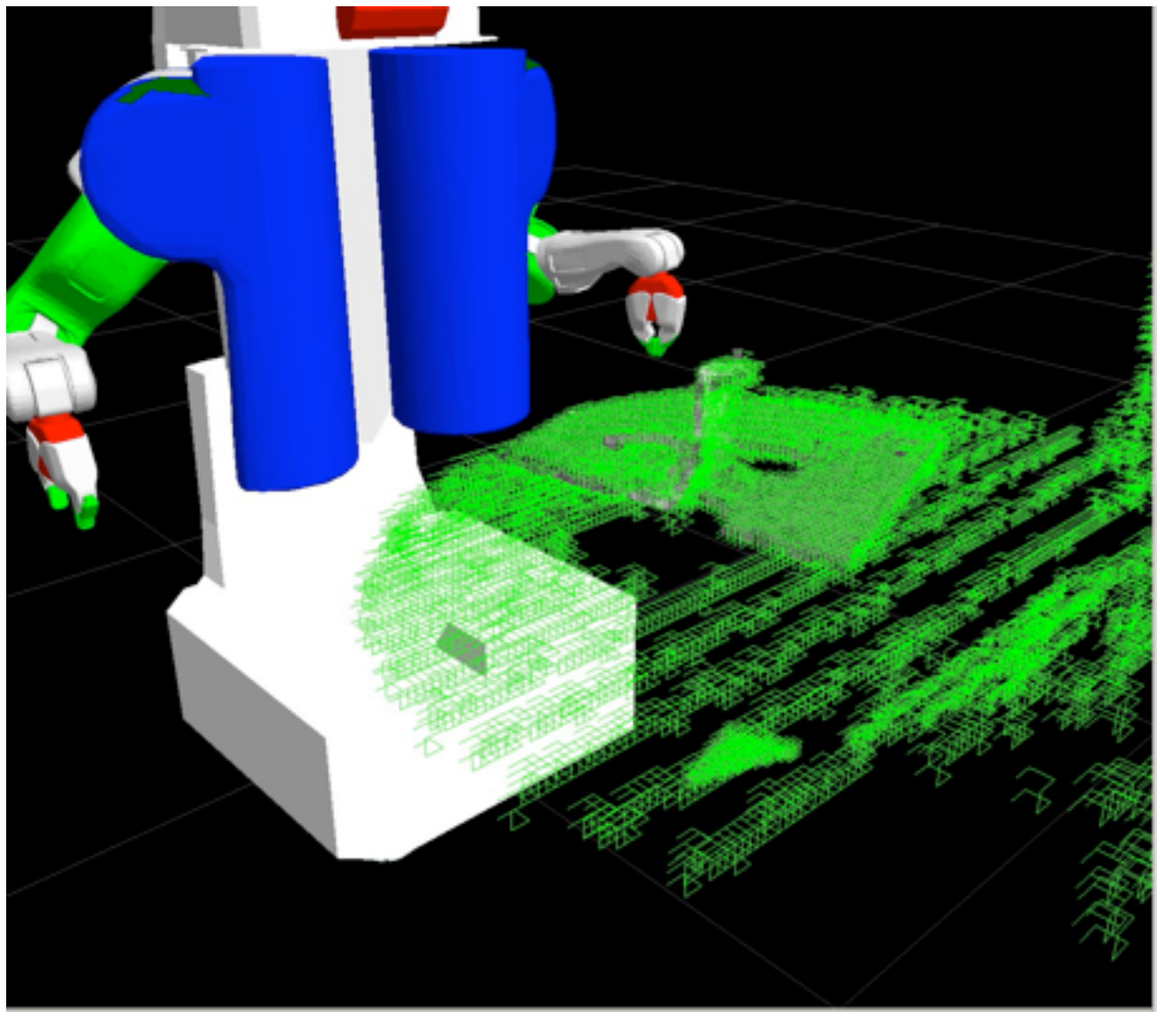


Collision Environment

- Collision map
  - account for self-occlusion



Collision Environment



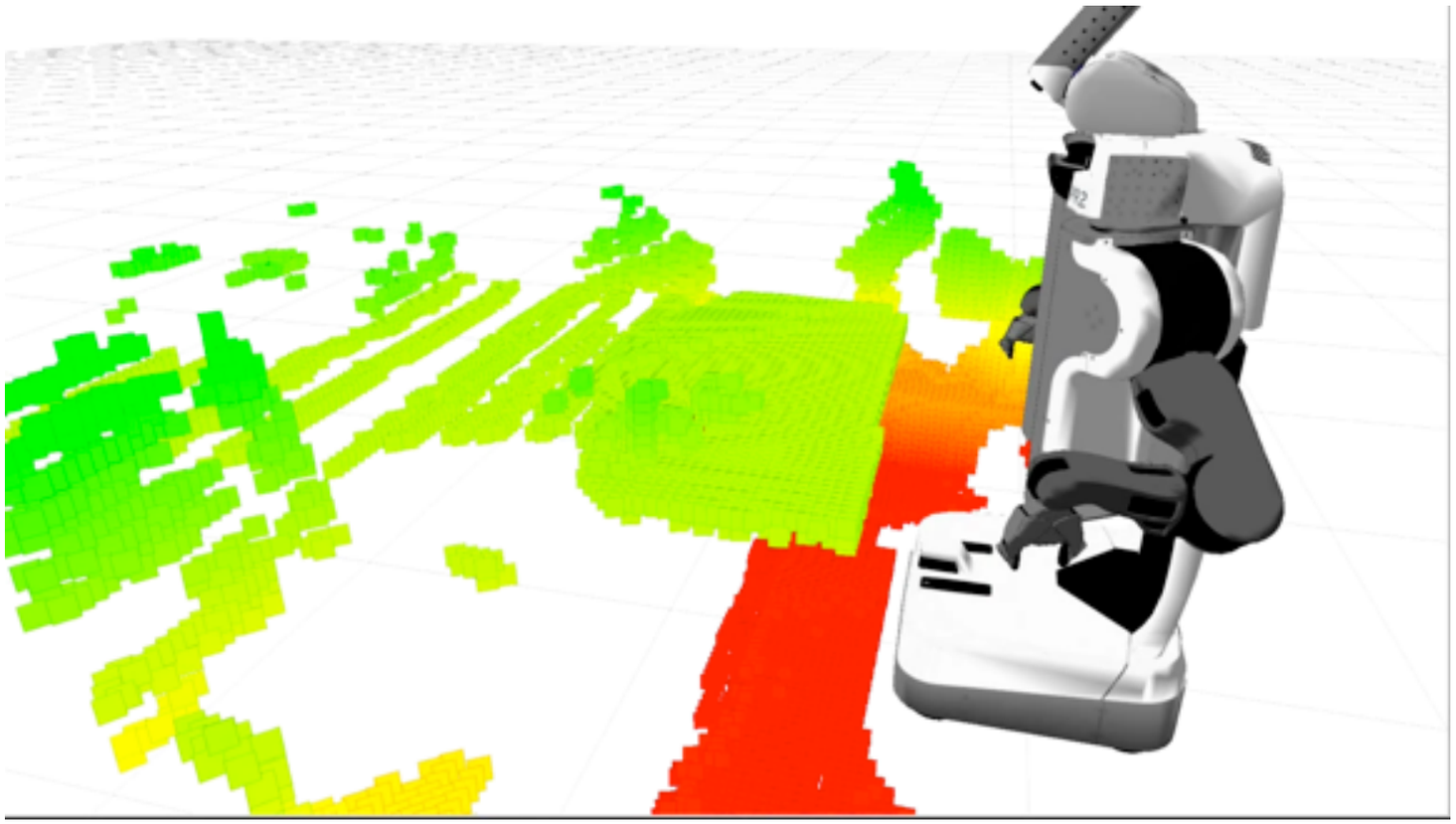
Collision map -http://www.ros.org/wiki/collision_map
Collision space - http://www.ros.org/wiki/collision_space

Collision Environment

- You can create your own environment by adding objects into the space as a
  - ✓ mesh
  - ✓ geometric primitive - boxes, cylinders, spheres

```
//add the cylinder into the collision space
  mapping_msgs::CollisionObject cylinder_object;
  cylinder_object.id = "pole";
  cylinder_object.operation.operation = mapping_msgs::CollisionObjectOperation::ADD;
  cylinder_object.header.frame_id = "base_link";
  cylinder_object.header.stamp = ros::Time::now();
  geometric_shapes_msgs::Shape object;
  object.type = geometric_shapes_msgs::Shape::CYLINDER;
  object.dimensions.resize(2);
  object.dimensions[0] = .1;
  object.dimensions[1] = .75;
  geometry_msgs::Pose pose;
  pose.position.x = .6;
  pose.position.y = -.6;
  pose.position.z = .375;
  pose.orientation.x = 0;
  pose.orientation.y = 0;
  pose.orientation.z = 0;
  pose.orientation.w = 1;
  cylinder_object.shapes.push_back(object);
  cylinder_object.poses.push_back(pose);
```

Collider



- Generate collision map for manipulation using Octomap
  - ❖ future plans to use notion of uncertainty in sensing
  - ❖ probabilistic representation possibly more robust to sensor noise, errors

Octomap - http://octomap.sf.net
Collider - http://www.ros.org/wiki/collider
Armin Hornung, Kai Wurm, et. al., Freiburg

Manipulation

- Motion planning
  - sensing for motion planning
    - semantic perception
  - collision environment
  - kinematics
  - motion planners
  - smoothers
  - monitoring and control

Kinematics

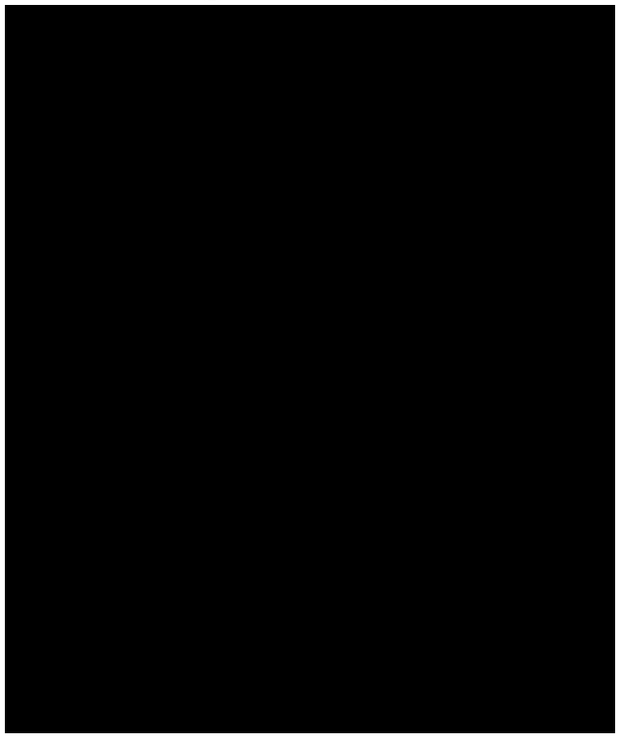
- Robot specific kinematics
  - e.g. pr2_kinematics package - fast custom solvers

- Constraint, collision aware kinematics
  - search for collision free configurations that satisfy constraints for general arms

PR2 custom kinematics -http://www.ros.org/wiki/pr2_arm_kinematics
General kinematics solvers - http://www.ros.org/wiki/arm_kinematics_constraint_aware

Kinematics

- Robot specific kinematics
  - e.g. pr2_kinematics package - fast custom solvers



- Constraint, collision aware kinematics
  - search for collision free configurations that satisfy constraints for general arms

PR2 custom kinematics -http://www.ros.org/wiki/pr2_arm_kinematics
General kinematics solvers - http://www.ros.org/wiki/arm_kinematics_constraint_aware

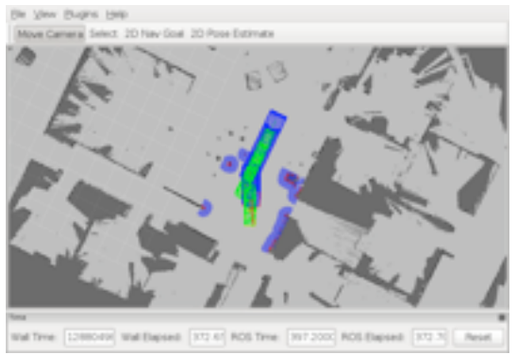
Manipulation

- Motion planning
  - sensing for motion planning
    - semantic perception
  - collision environment
  - kinematics
  - motion planners
  - smoothers
  - monitoring and control

ROS - Motion Planners

- Sampling based planners (OMPL)
  - ❖ developed by Lydia Kavraki's group at Rice University in collaboration with Willow Garage

- CHOMP*
- Search based planners (SBPL)
  - ❖ developed by Maxim Likhachev's group (now at CMU) in collaboration with Willow Garage



*Developed by Mrinal Kalakrishnan, Intern, Summer 2009 at Willow Garage based on "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning". Nathan Ratliff, Matthew Zucker, J. Andrew Bagnell and Siddhartha Srinivasa. IEEE International Conference on Robotics and Automation, May 2009.

http://www.ros.org/wiki/ompl
http://www.ros.org/wiki/ompl_ros_interface
http://www.ros.org/wiki/chomp_motion_planner
http://www.ros.org/wiki/sbpl

OMPL

- Sampling-based planners
  - RRT, RRT-Connect, EST, KPIECE, SBL and more
  - coming soon - RRT*
- Very fast planning times
- Extensive documentation
- ROS interface
  - easily go from robot description to complete planners
  - configure task space planners

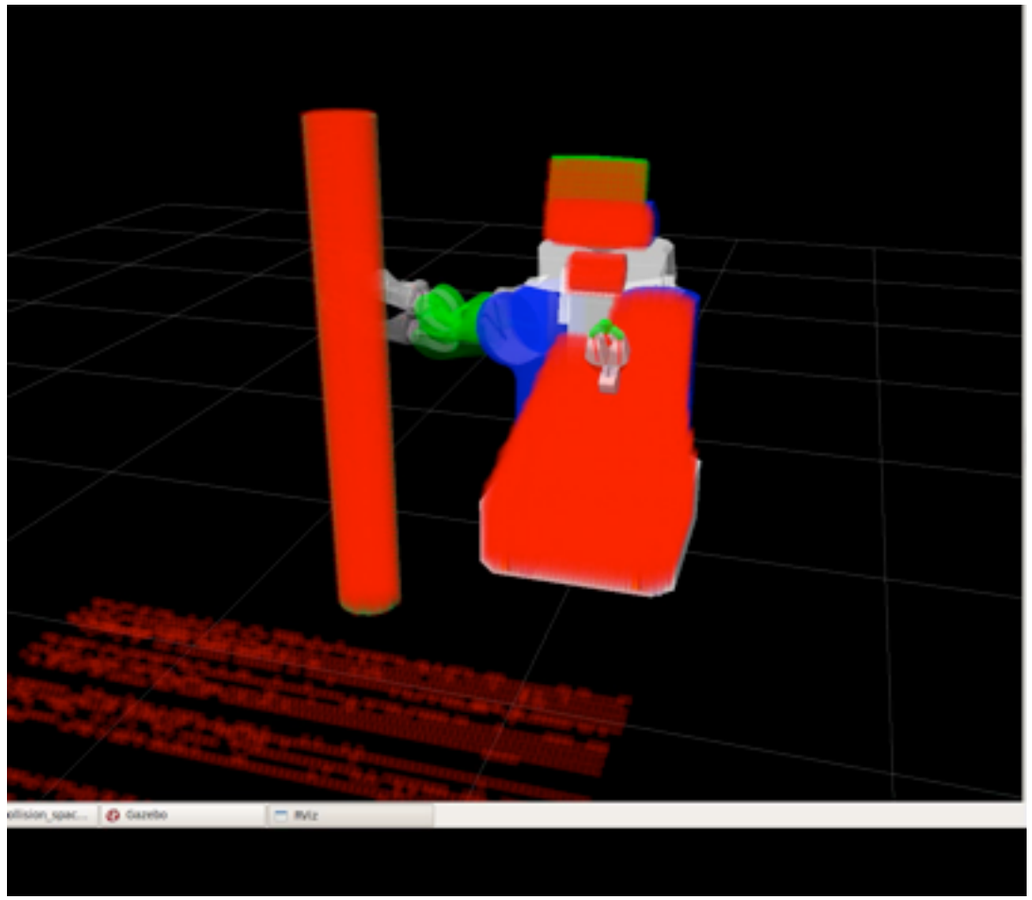
Smoothers

- Sampling based planners generate jerky paths
  - smoothing required before plans can be sent to controllers
  - cubic spline shortcuts used to smoothen paths

Cubic spline smoother - http://www.ros.org/wiki/constraint_aware_spline_smoother

CHOMP

- Covariant descent based approach - generates smoother trajectories
- Uses signed distance field to move robot away from obstacles

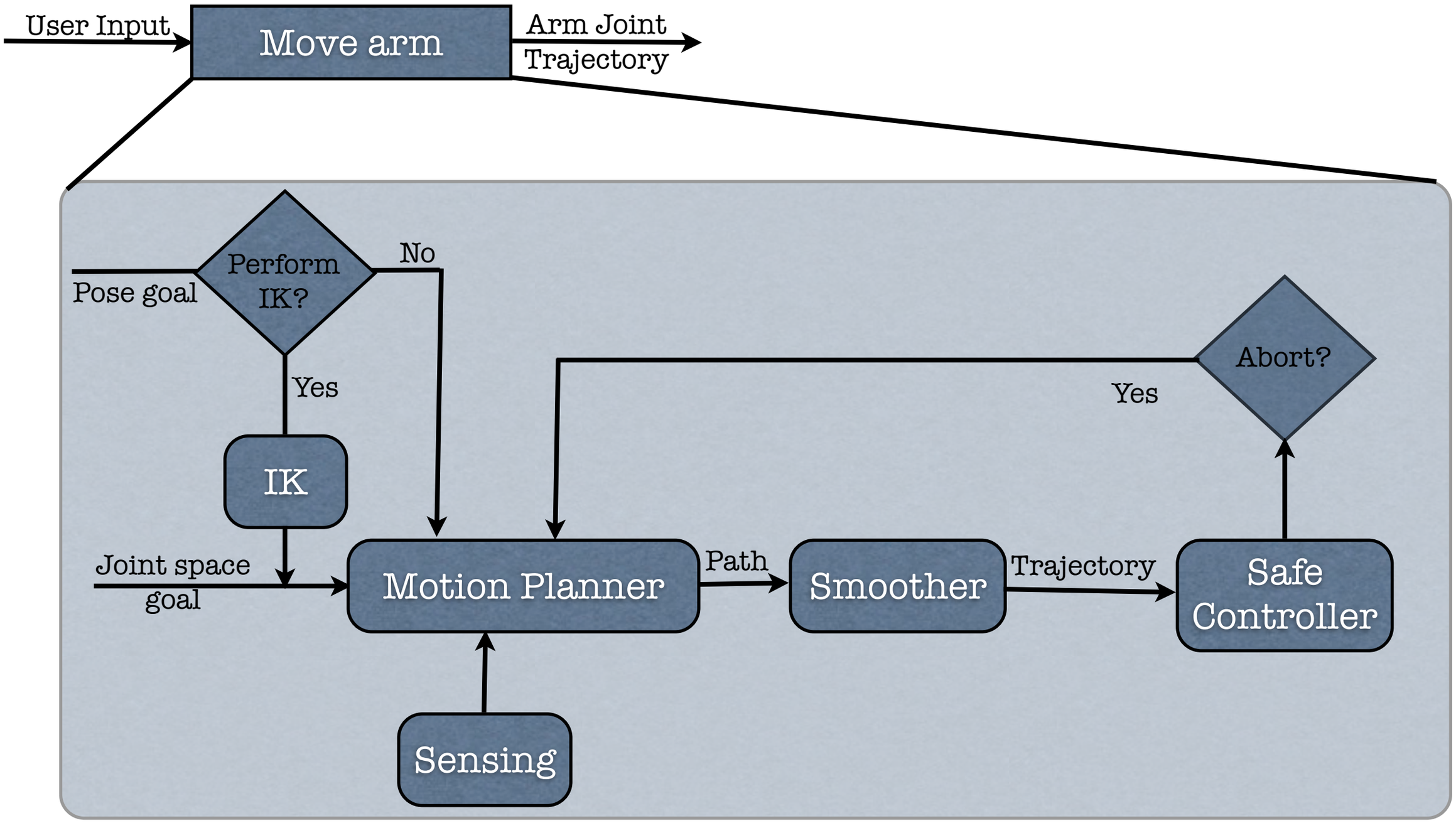


CHOMP - http://www.ros.org/wiki/chomp_motion_planner

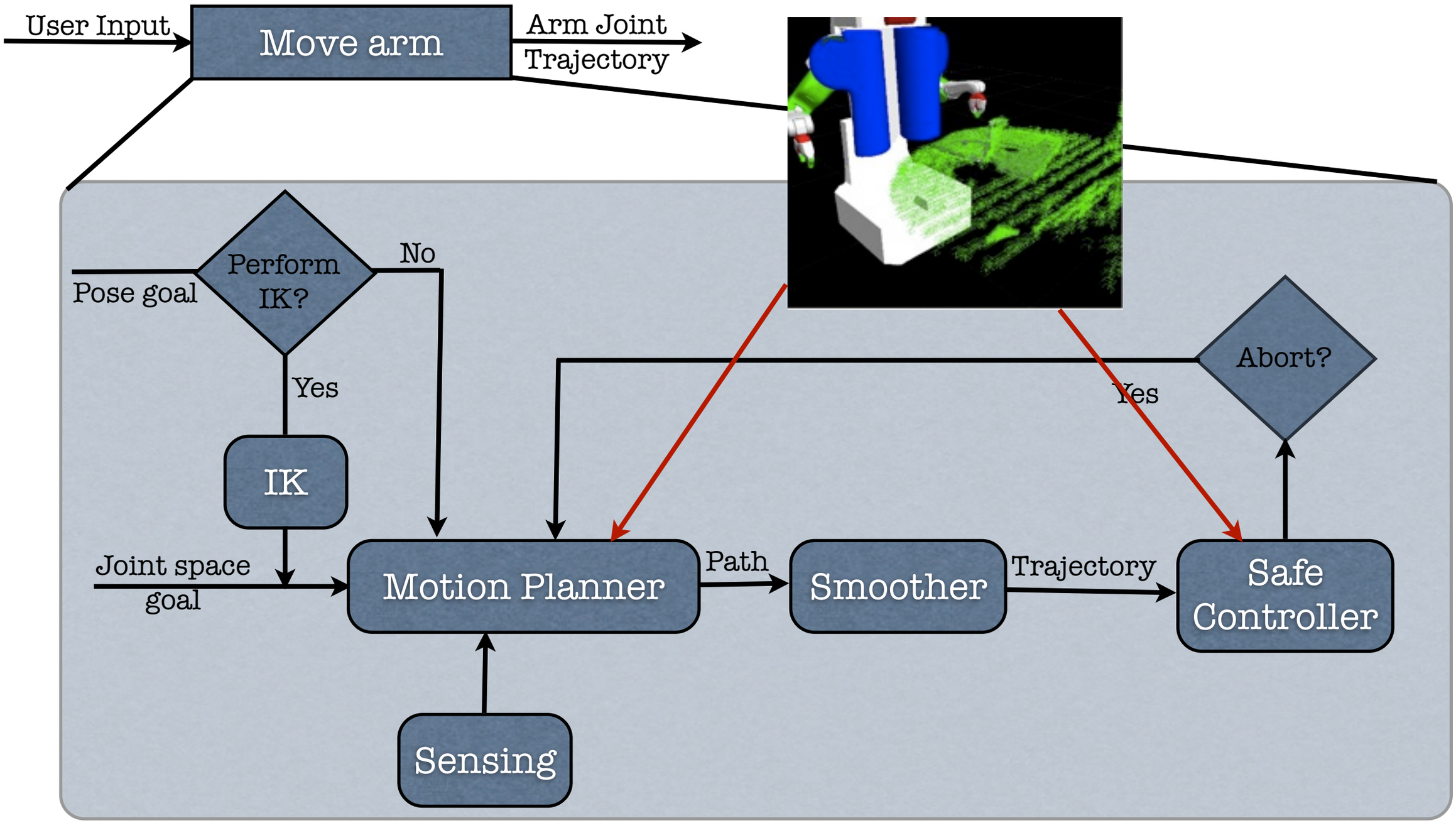
Applications for CHOMP

- Planning out of contact
  - ✓ noisy sensor data, noisy models, actual contact
  - ✓ typically achieved by jittering start state

System Architecture



System Architecture



System Architecture

- Input
  - ❖ specify goal as regions in space
    - ✓ joint space goal - nominal joint angle + tolerance
    - ✓ position goal - nominal position + region of space
    - ✓ orientation goal - nominal orientation + tolerance (in roll, pitch and yaw)

Willow Garage

Move arm -http://www.ros.org/wiki/move_arm
Interface definition -http://www.ros.org/wiki/move_arm_msgs

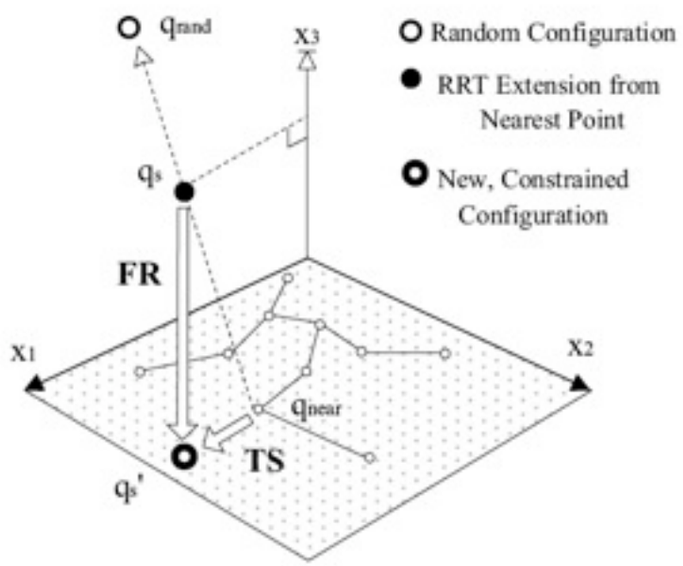
System Architecture

- Constraints
  - keep the glass of water level - less freedom in roll and pitch space
  - constraint specification:

```
orientation_constraint.header.frame_id = "torso_lift_link";
orientation_constraint.header.stamp = ros::Time::now();
orientation_constraint.link_name = "r_wrist_roll_link";
orientation_constraint.orientation.x = 0.0;
orientation_constraint.orientation.y = 0.0;
orientation_constraint.orientation.z = 0.0;
orientation_constraint.orientation.w = 1.0;
orientation_constraint.type = motion_planning_msgs::OrientationConstraint::HEADER_FRAME;
orientation_constraint.absolute_roll_tolerance = 0.1;
orientation_constraint.absolute_pitch_tolerance = 0.1;
orientation_constraint.absolute_yaw_tolerance = M_PI;
```

Dealing with constraints

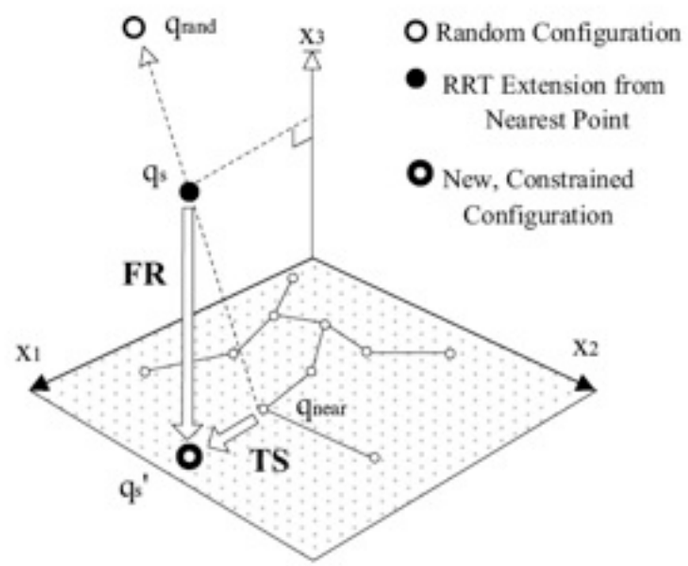
- In joint space - the constraint manifold is complicated
- sampling based planners sample in joint space
  - ✓ far away from constrained manifold
  - ✓ need to project samples back onto constraint manifold



Picture from Stilman M., "Task Constrained Motion Planning in Robot Joint Space", in IROS 2007

Dealing with constraints

- In joint space - the constraint manifold is complicated
- sampling based planners sample in joint space
  - ✓ far away from constrained manifold
  - ✓ need to project samples back onto constraint manifold



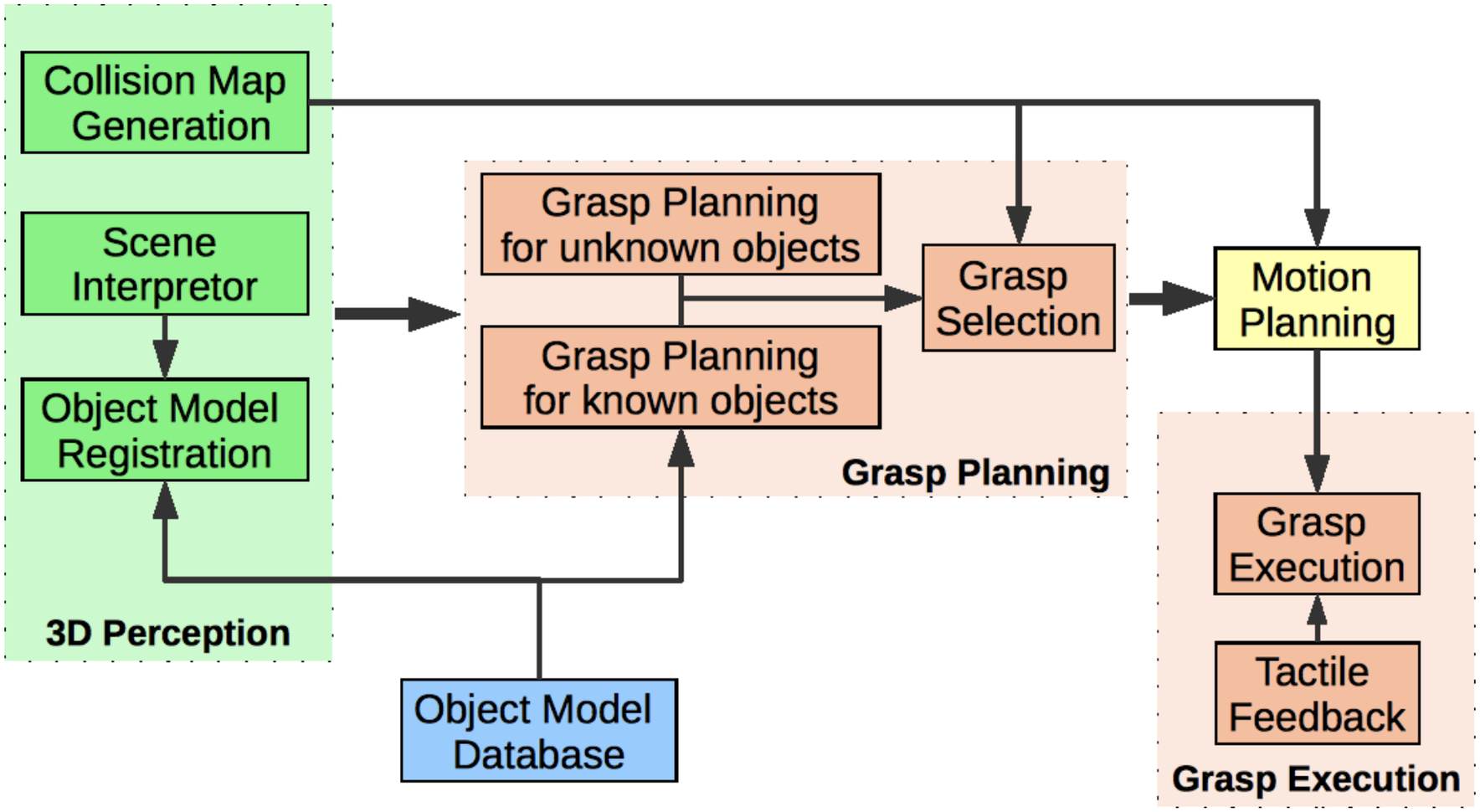
- Most constraints are expressed more naturally in cartesian space
  - ✓ e.g. holding water level => restricting roll and pitch of the end-effector

Picture from Stilman M., "Task Constrained Motion Planning in Robot Joint Space", in IROS 2007

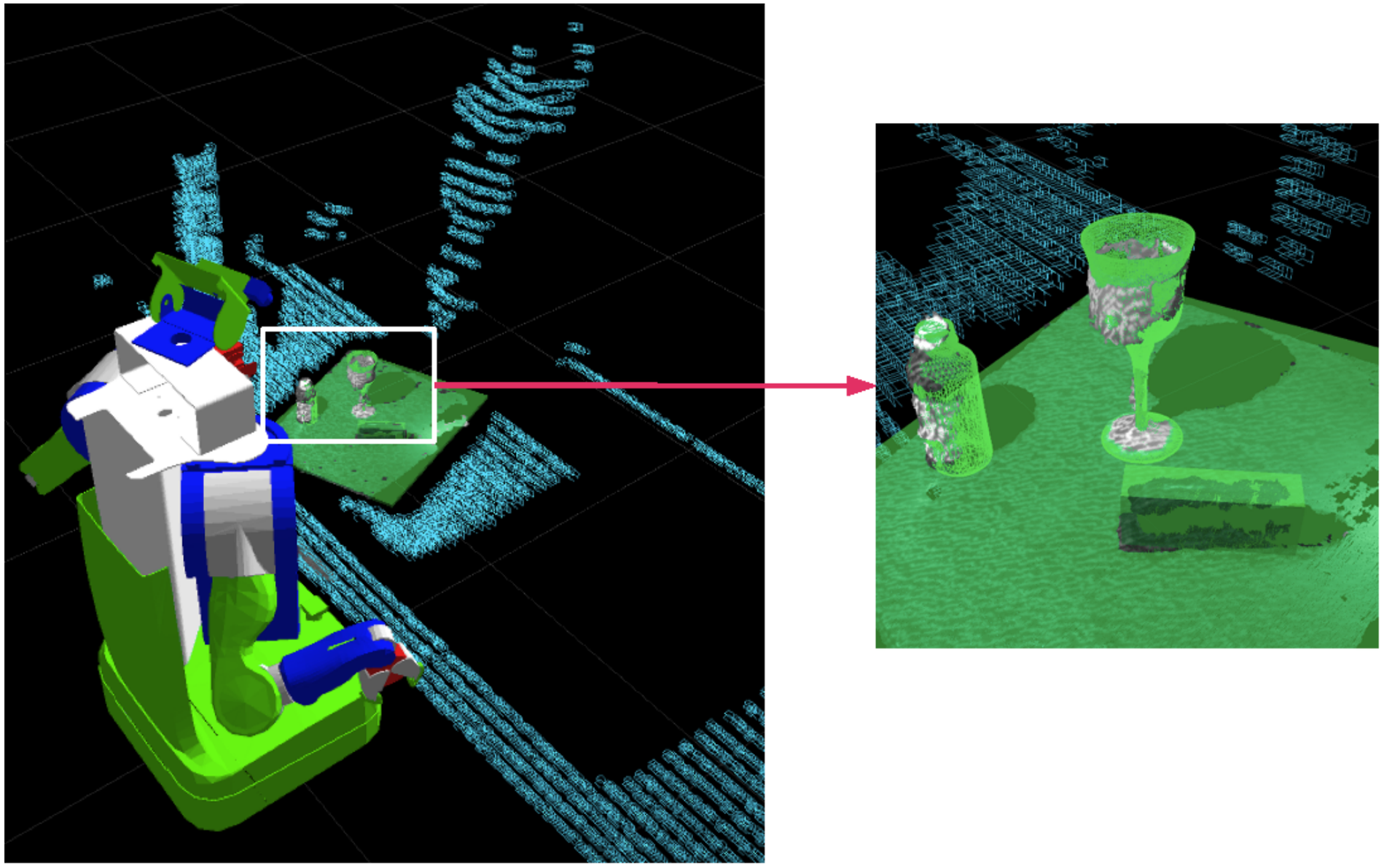
Constrained Planning

- Exploit geometry of the PR2
  - ❖ joint limits imply only 1 IK solution branch exists most of the time
  - ❖ plan in cartesian end-effector coordinates + redundancy (shoulder roll joint angle)
  - ❖ no need to do constraint projections in joint space
  - ✓ just shrink region available for sampling!

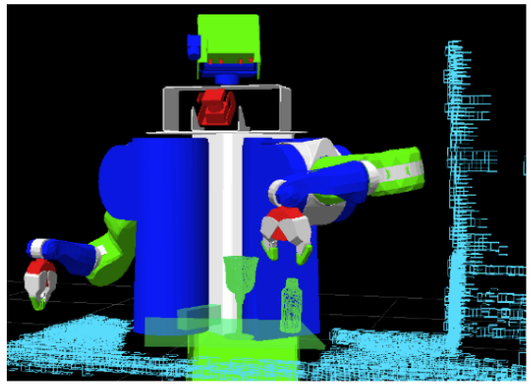
Application - Grasping

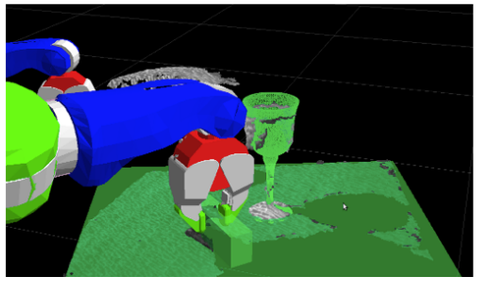


Grasping Pipeline



Arm Navigation and Manipulation

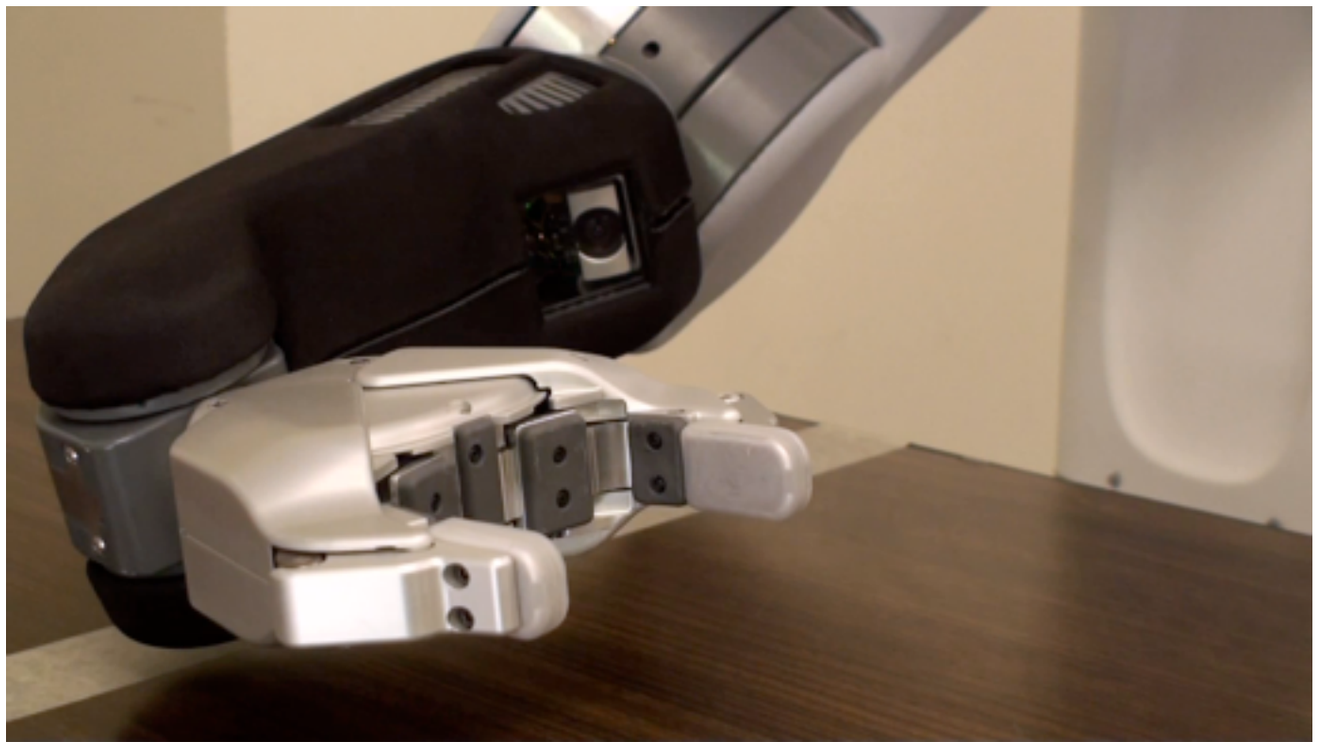




http://www.ros.org/wiki/arm_navigation
http://www.ros.org/wiki/object_manipulation
http://www.ros.org/wiki/pr2_object_manipulation

Towards a safer robot

- Detect impacts
  - accelerometer and joint signals



Joe Romano (Intern at Willow Garage, Summer 2010), Kaijen Hsiao, Gunter Neimeyer, Sachin Chitta, Katherine Kuchenbecker (Penn)

http://www.ros.org/wiki/pr2_gripper_sensor_action

Towards a safer robot

- Actively monitor motion
  - abort on possible collision

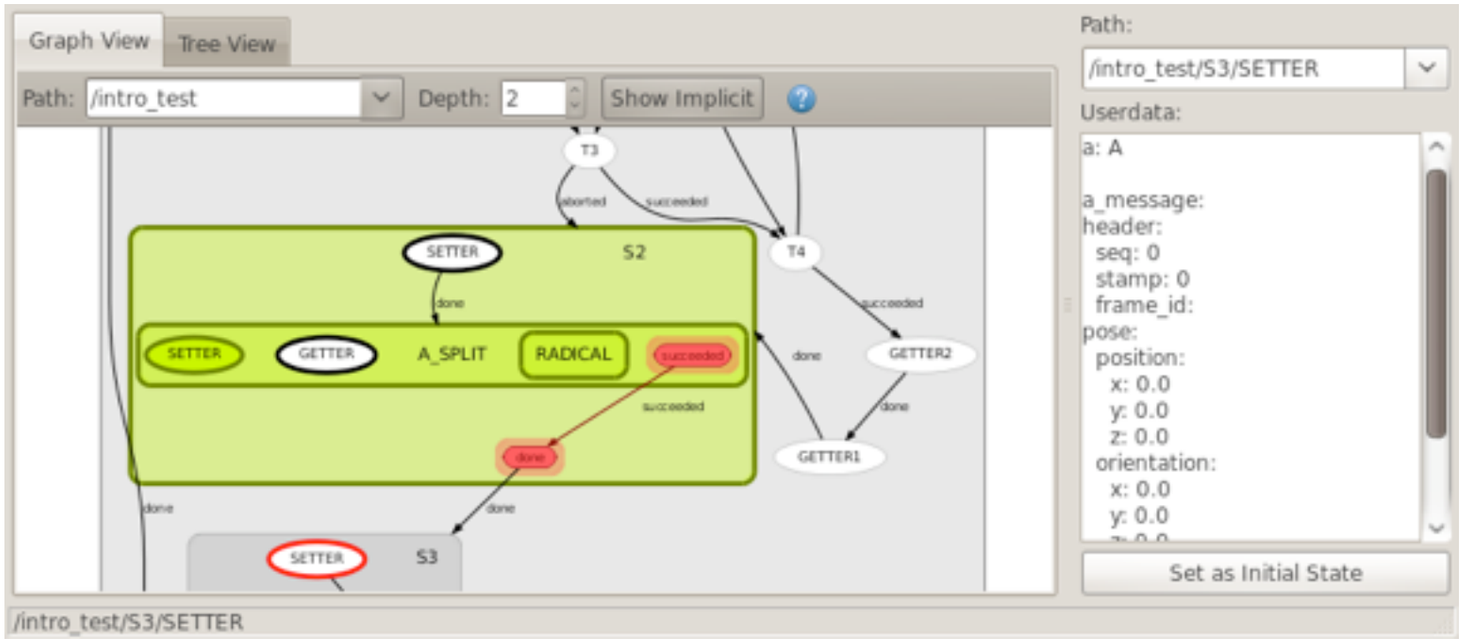
Recent Work - STOMP

"Stochastic Trajectory Optimization for Motion Planning", Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorus, Peter Pastor, Stefan Schaal, ICRA 2011

Outline

- The PR2
- ROS
  - tools for simulation
  - tools for motion planning
  - task execution
  - applications
- Future Directions

Task Execution



- SMACH
  - ❖ Task-level state machine architecture
  - ❖ State machine architecture
    - ✓ concurrence
    - ✓ synchronization
    - ✓ failure recovery
    - ✓ pre-emption

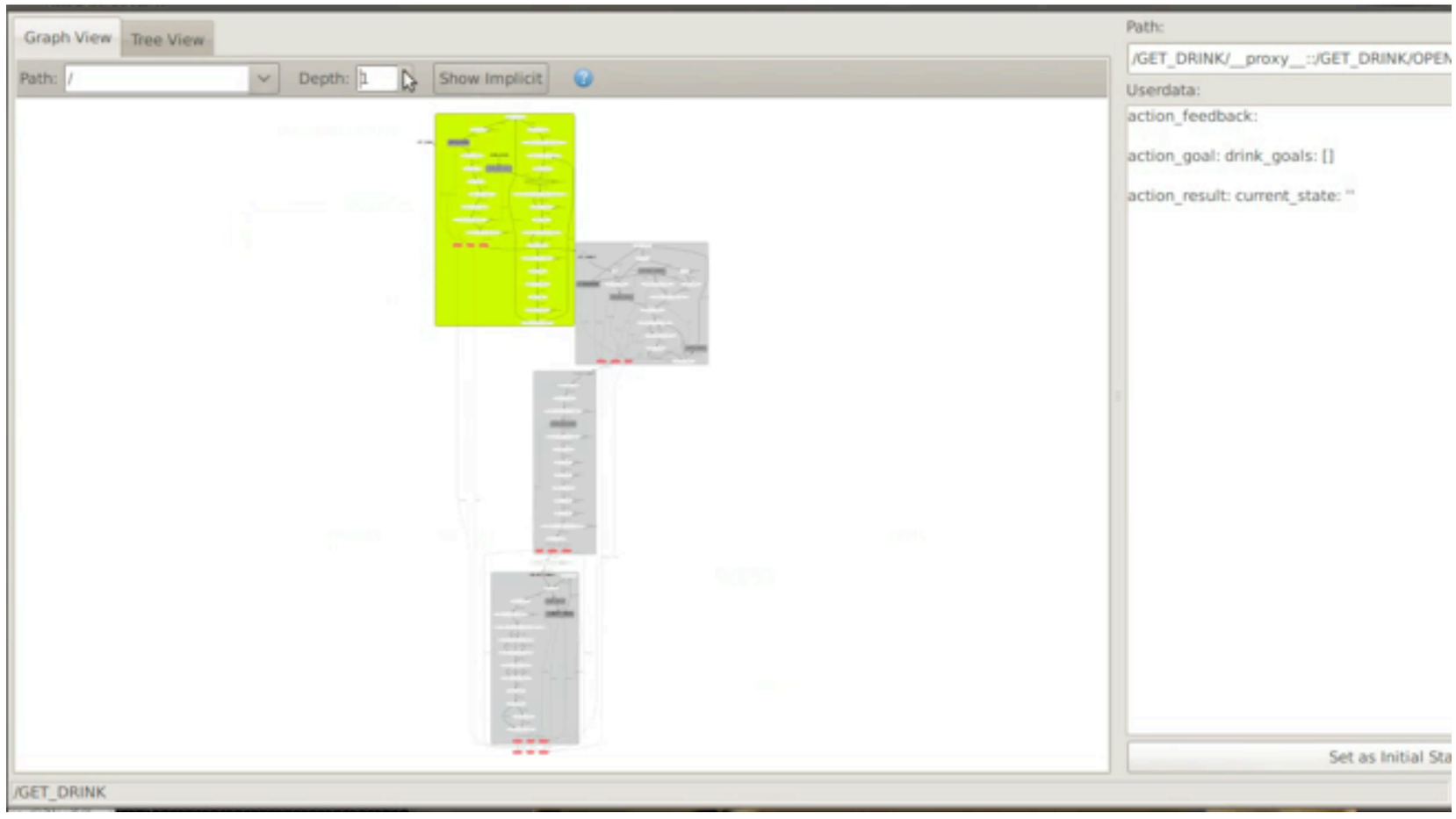
SMACH

- Meta-programming on top of Python

```
def construct_sm():
    open_door_sm = StateMachine(['succeeded','aborted','preempted'])
    with open_door_sm:
        cs0 = Concurrence(['succeeded','aborted'],'aborted')
        StateMachine.add('PREP_FOR_NAV',cs0,{'succeeded':'NAVIGATE_TO_FRIDGE'})
        with cs0:
            Concurrence.add('TUCK_FOR_NAV',
                    SimpleActionState('tuck_arms',TuckArmsAction,
                        goal = TuckArmsGoal(False,True,True)))

            Concurrence.add('LOWER_SPINE',
                    SimpleActionState('torso_controller/position_joint_action', SingleJointPositionAction,
                        goal = SingleJointPositionGoal(position=0.07)))

        Concurrence.add_outcome_map('succeeded',{'TUCK_FOR_NAV':'succeeded','LOWER_SPINE':'succeeded'})
```

SMACH

- Visualize updates in real-time



Application - Beverage Retrieval

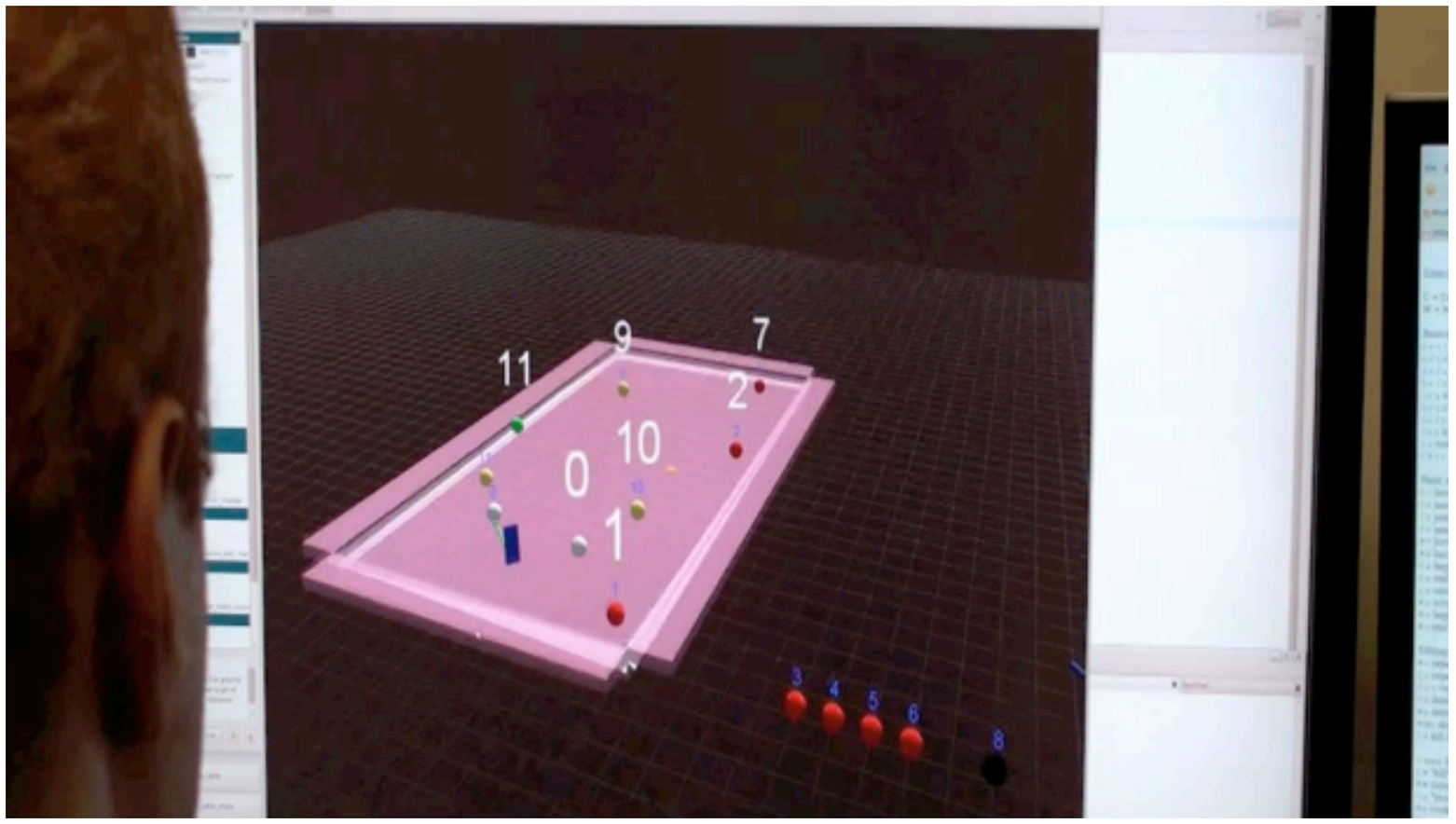
Applications - Pool

- ✓ specialized hardware
- ✓ hackathon - 1 week effort

- ✓ no motion planning for the arm
- ✓ pre-determined waypoints around table for base

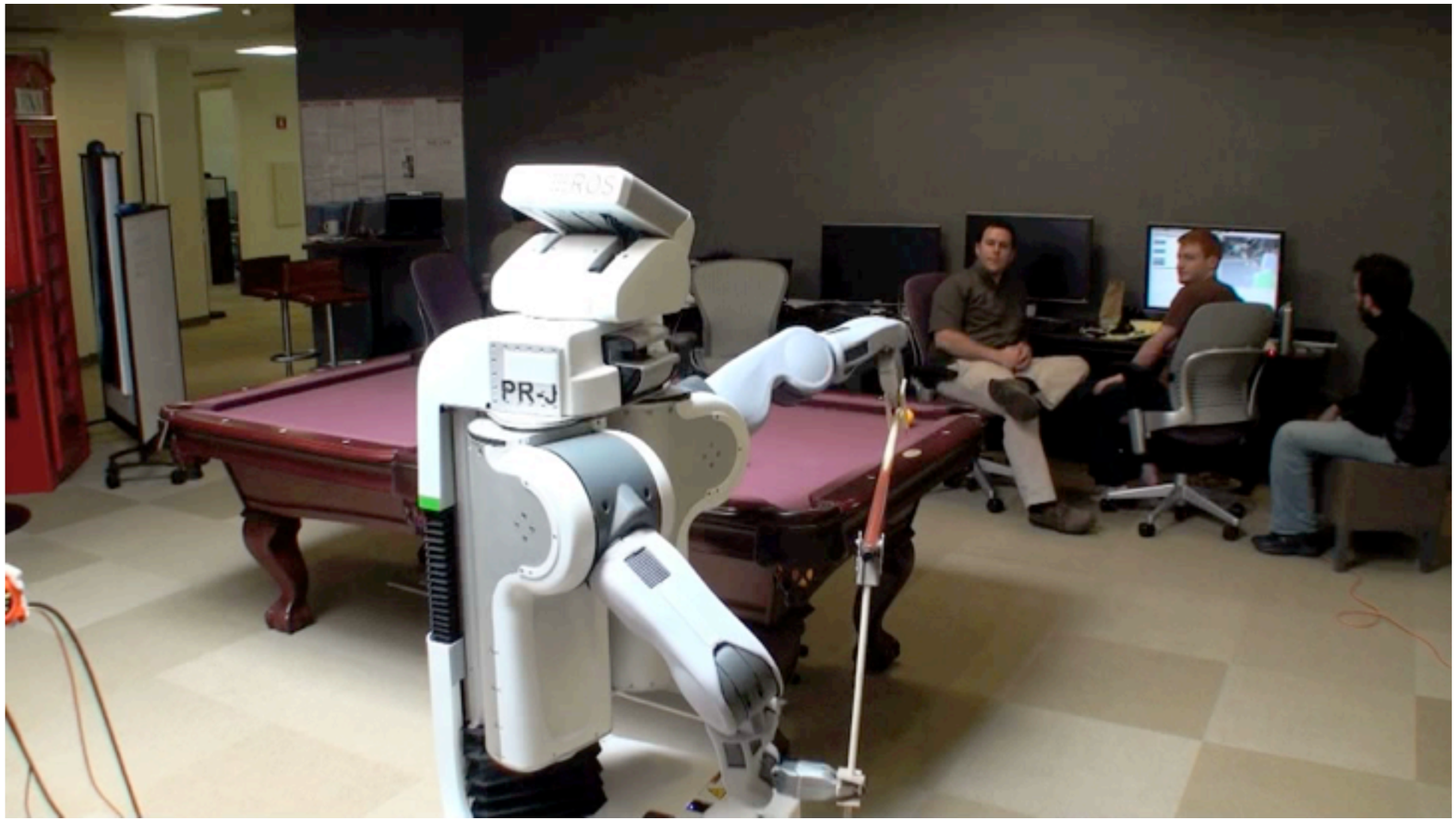
Application - Pool

- Simulation based planning*



*Open source simulator FastFiz by Alon Altman at Stanford University
http://billiards2.stanford.edu/FastFiz/main.html

Application - Pool



Application - Busbot

What's coming?

- Better tools
  - ❖ go from URDF to complete planning in a few quick steps
  - ❖ visualize everything along the way
  - ❖ interactive GUI to test out planning

Thank you!

More resources:

1. http://www.ros.org/wiki
2. http://answers.ros.org
3. ros-users mailing lists

Thank you!

Email: sachinc@willowgarage.com

This talk included contributions from the entire ROS and PR2 development teams at Willow Garage, the PR2 Beta Program and the ROS community.

External collaborators/contributors:
Lydia Kavraki, Rice University
Maxim Likhachev, CMU