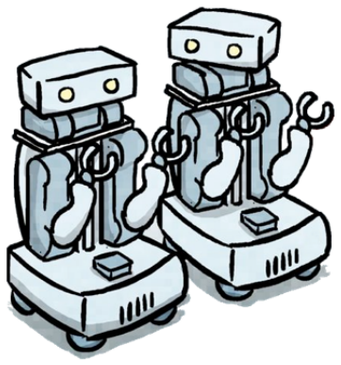


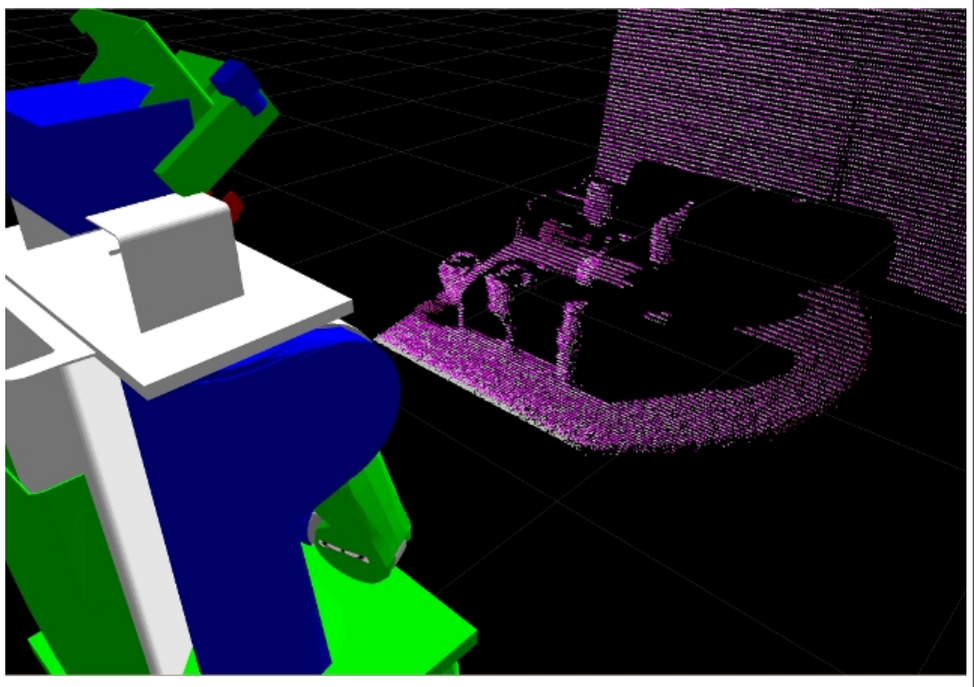
The Complete Story:

Calibration, Perception and Execution

Sachin Chitta

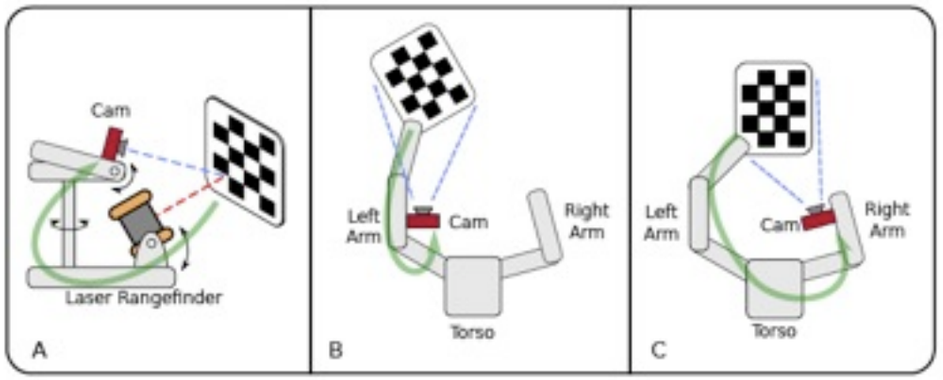
Sensing

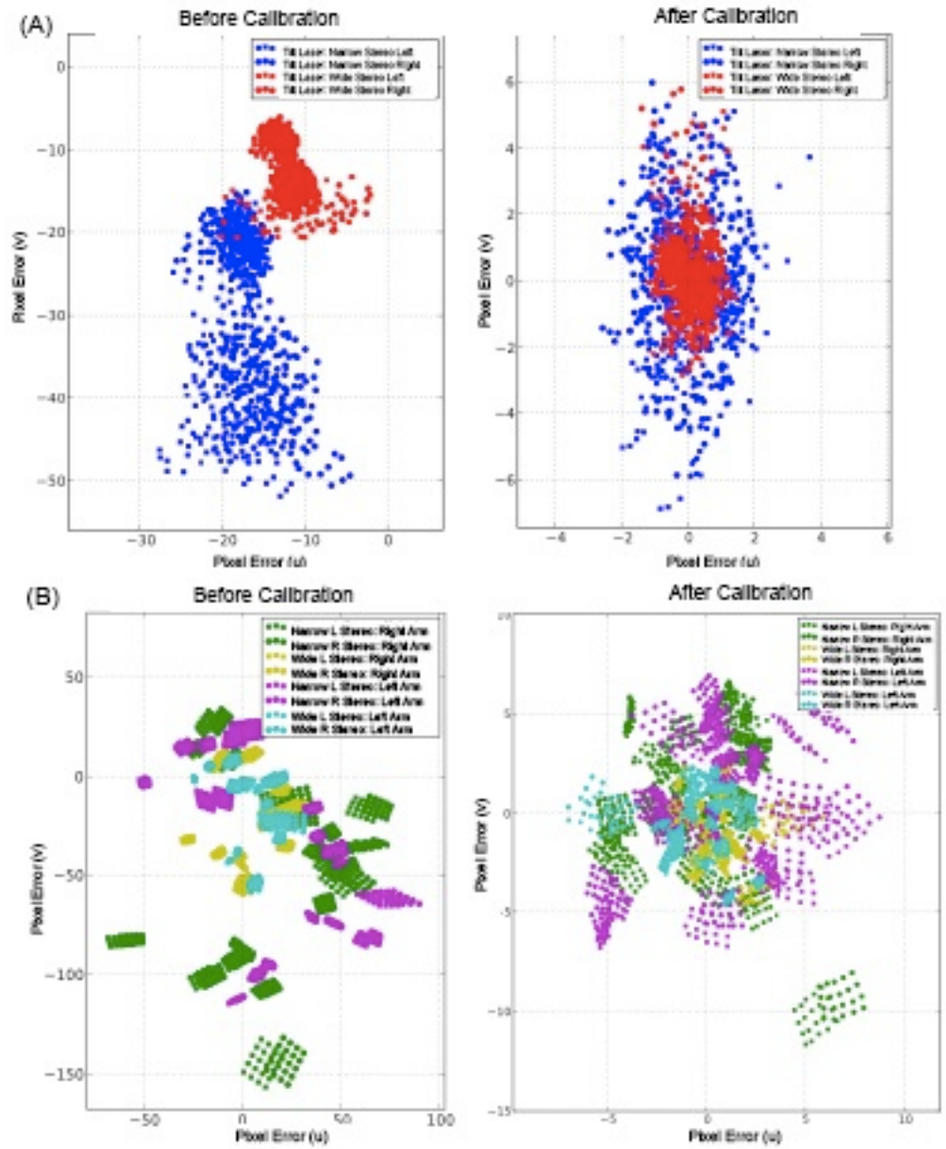
- Sensor input
  - point cloud (from stereo, lasers, etc.)
    - ❖ collection of 3D points
    - ❖ can annotate other information
      - ✓ RGB color, intensity values
    - ❖ stereo (20-25,000 points)
    - ❖ laser sensor (5-7,000 points)



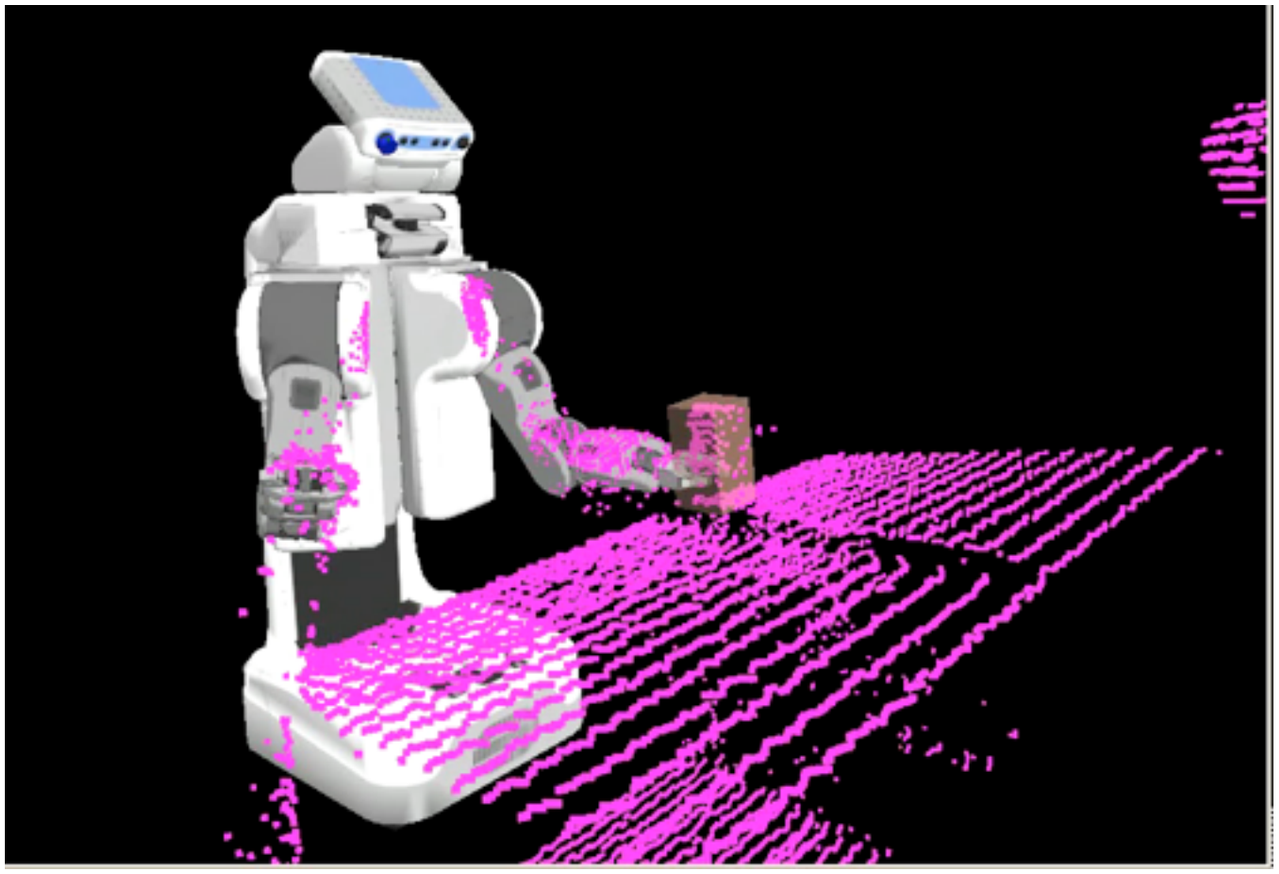
ROS - Calibration

- http://www.ros.org/wiki/pr2_calibration
  - calibrating the full chain





Sensing Pipeline



- Shadow filtering
  - ❖ filter out noisy sensor data
- Self-filtering
  - ❖ filter out body parts and attached objects
- Final filtered scans
  - ❖ less noisy + do not contain body parts

Sensing Pipeline

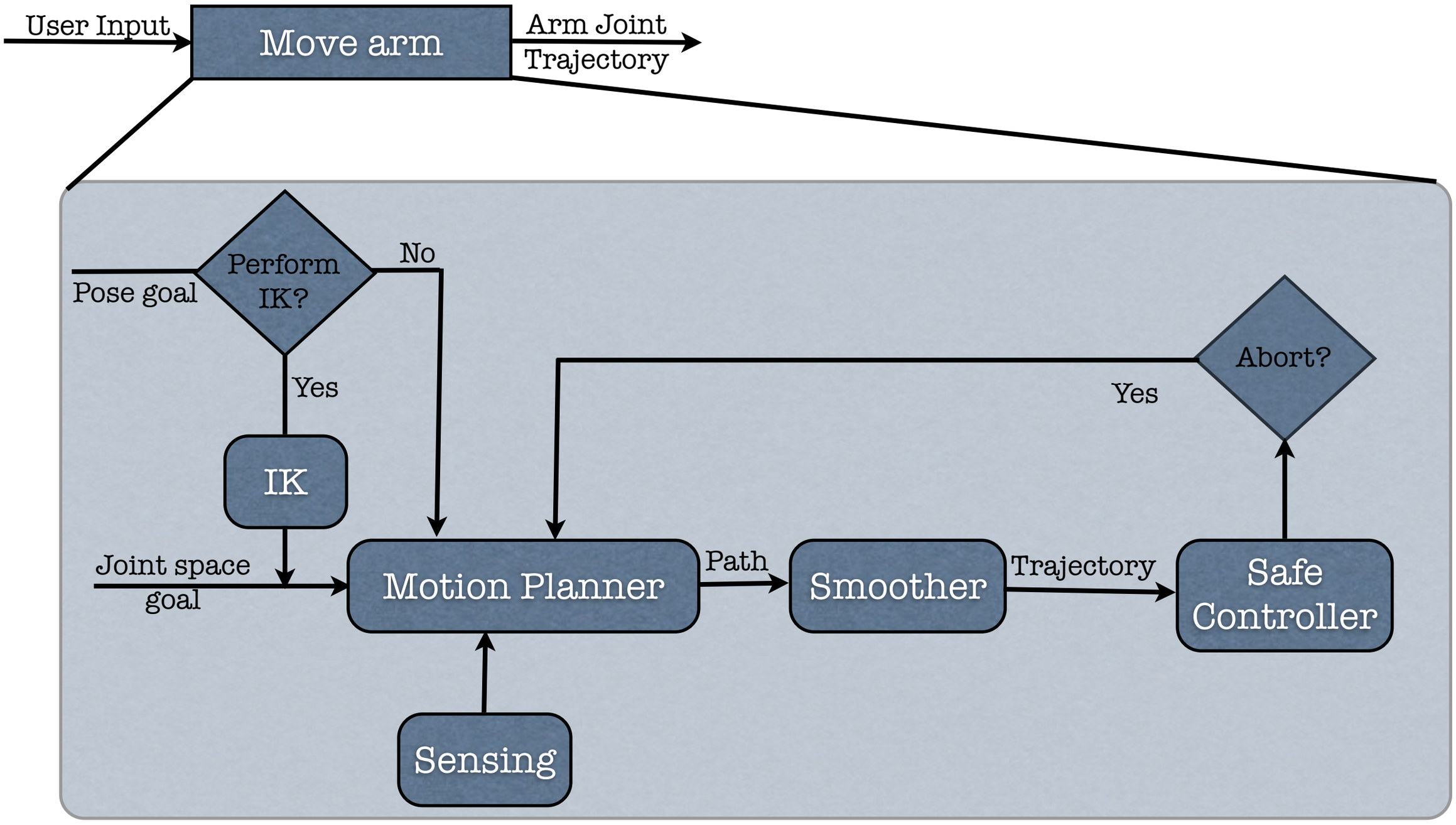
- 3D pipeline
  - ROS PointCloud2 message forms the basis of perception pipeline
  - Point clouds from multiple sensors can be integrated
  - Self filtering is important
    - ✓ good calibration is essential for self-filtering
    - ✓ good synchronization of sensing data with proprioceptive information, e.g. joint data, is essential
- You will need to implement this for your own robot
  - you can use the tools/components we provide
  - a fair bit of configuration/tweaking will be required
    - ✓ not fully automated yet (Sorry!)

Sensing Pipeline

- Object recognition as input
  - ❖ main input to motion planning pipeline is through addition of a “Collision Object”
    - ✓ encode object information using ROS CollisionObject message
    - ✓ can represent simple geometric primitives or mesh objects

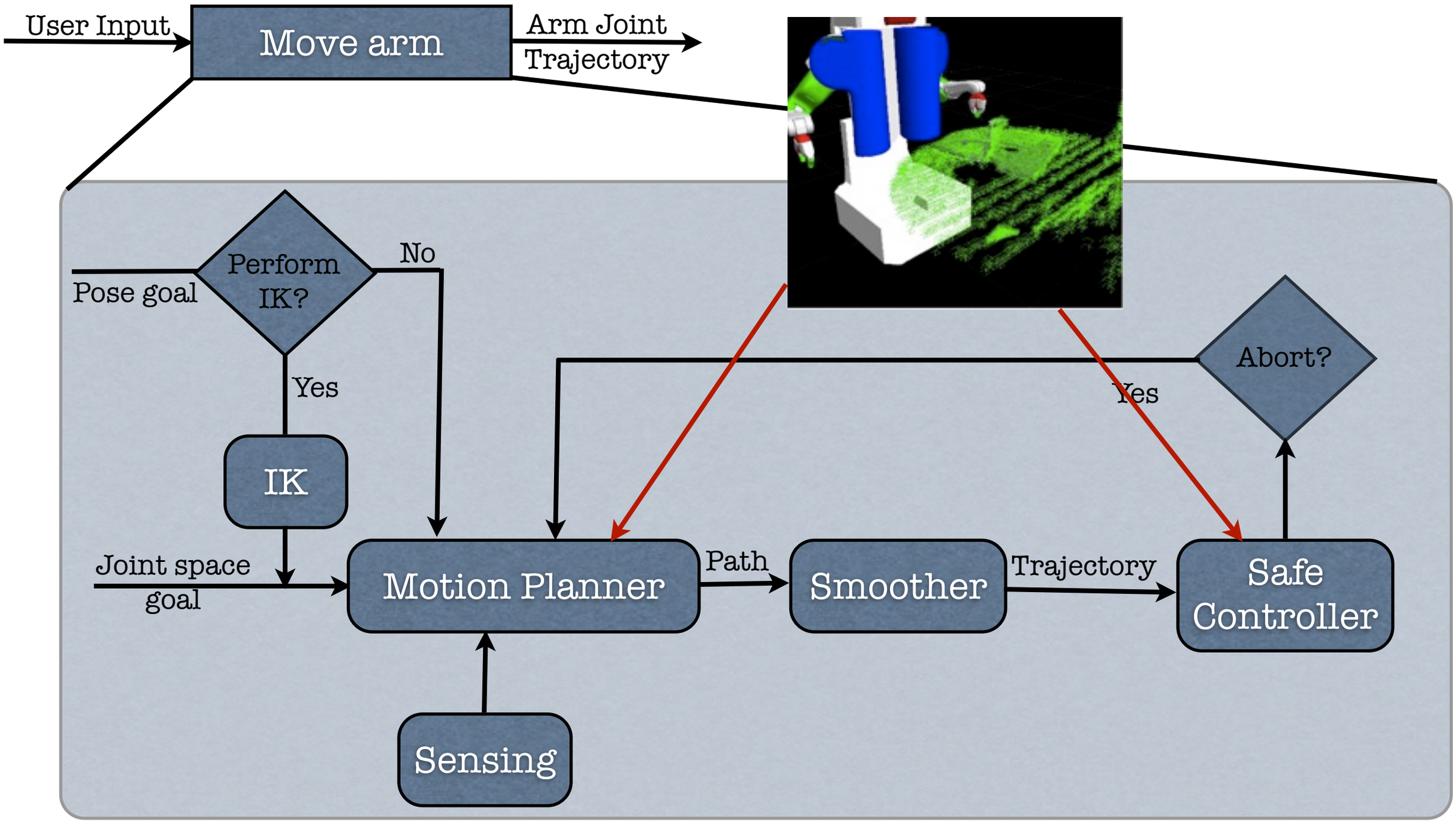
http://www.ros.org/doc/electric/api/arm_navigation_msgs/html/msg/CollisionObject.html

Execution



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

Execution


User Input
Move arm
Arm Joint Trajectory
Perform IK?
No
Pose goal
Yes
IK
Joint space goal
Motion Planner
Path
Smoother
Trajectory
Safe Controller
Abort?
Yes
Sensing

Configuration

- move_<group_name> files are generated for you
  - you need to connect them up with your robot
    - controller interface is most important configuration
    - connect to controller using “action” interface
- check your joint limits file
    - you need to specify correct acceleration, velocity constraints

Control

- Action Interface
  - ROS FollowJointTrajectory “action” interface must be implemented on your robot
  - allows for the following capabilities
    - ✓ follow trajectories specified using ROS JointTrajectory message
      - trajectories consist of waypoints with position, velocity (optional) and acceleration(optional)
    - ✓ queue trajectories for future execution (according to timestamp)
    - ✓ preempt trajectories
    - ✓ abort trajectories

http://www.ros.org/doc/electric/api/control_msgs/html/msg/FollowJointTrajectoryAction.html

Thank you!

More resources:

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