Robotic manipulator on TB2
How to control a manipulator
1) Introduction to robotic manipulators

- What is robotic manipulator?
  - Part of a robot that is intended to manipulate with objects.
  - Robotic manipulators vary in number of joints, in shape, end effector...

- Joint
  - Connection of two rigid (or floppy) parts of manipulator.

- Body
  - Rigid or floppy part of manipulator – not joint.

- Kinematic chain
  - Interconnection of bodies and joints.
  - Open vs. Closed kinematic chain.

- Degree of Freedom (DOF)
  - “Number of ways” in which a kinematic chain can move.
  - DOF of manipulator is a sum of DOFs of all bodies of the manipulator.
2) Components of arm control stack

- Hardware drivers (drivers for servo drives)
- Kinematics model
- Collision model
- Dynamics model
- IK solver
- Trajectory planner
- Grasping planner
2) Hardware drivers, kinematics model, collision model

- Hardware drivers (drivers for servo drives)
  - Provide interface for low level servo drives control to a high level system.

- Kinematics model
  - Used for Forward kinematics
  - Computation of end effector position according to joint coordinates
  - Contains also joint limits

- Collision model
  - Simplified model of arm - usually boxes or cylinders only.
  - Speeds up collision computations.
  - Is usually larger then a real arm - safety threshold.
2) Dynamics model

- Necessary for predict behavior of a real arm.
- Includes:
  - Mass
  - Servo drives torques
  - Friction of joints
  - Resistance of environment (water)
- Dynamic model might dynamically change
  - For example by grasping a heavy object.
- Essential for operation safety of real manipulators.
- Many ways of identifying parameters of dynamic model of a real system.
2) IK solver

- A mathematical apparatus that computes joint coordinates from end effector position.
- In many cases there is infinite amount of solutions.
- Typically realized as a set of differential equations which has to be solved.

Two ways how to solve IK problem:

- Analytic way:
  - Very fast.
  - Requires analytic solution of a set of differential equations.

- Numerical way:
  - Universal.
  - Does not need any "intelligence".
  - Complexity of computation grows exponentially with increasing number of DOFs.

Problem with singularities

- Limit positions (like straighten arm).
- Problem of precision of value representation on a computer.
2) Trajectory planner

- Plans joint moves from a current positions to make them reach desired positions.
- Responsible for avoiding obstacles.
- Has to deal with dynamics of an arm.
- Usually two phases:
  - Trajectory planning
  - Trajectory smoothing
- Computation of a plan is often very resource demanding.
  - We have to often choose between quality of trajectory and speed of computation.
2) Grasping planner

- Plans movements of the arm end effector to grasp given object
- Uses previous types of planners
- Usually requires some additional informations about object being grasped
  - For example from which way is it possible to grasp, now intensive force should we grasp with, ...
- Some objects also limit manipulator movements after grasping
  - Like a glass of water for example
- In case of heavy objects we have to deal with mass of the object when planning arm movements
3) Arm hardware

- Smart Arm AX 18
  - Manufactured by CrustCrowler
- 4 DOF, rotational joints
- Reach: cca 40cm
- Load: cca 500g max
- Repeatability: 2.5mm
- Power consumption: up to 60W (9.6V/6A)
- 7 Dynamixel servo drives
- Feedback of joint positions (angles), servo drive temperature, overload
4) Implementations of TB2 arm controller

- Currently 2 implementations
  - OMPL based
  - OpenRAVE based
- Each implementation has particular advantages.
- Both implementations are still under development.
- Both implementations share the servo and arm drivers.
- Driver for servo drives
  - ROS stack - dynamixel_motors
- Driver for arm we are using
  - ROS package 'btb_dynamixel'
  - Created by Zdeněk Materna
  - Controls also Kinect servo
  - Extension of 'smart_arm_controller' package
5) Models for kinematics, collision, dynamics

- Kinematics and collision model are part of official ROS support for Smart Arm
- URDF (XACRO) and Extended COLLADA file format
  - Originally 'smart_arm_description' package
  - Currently 'btb_description' package
  - Currently extended for dynamics informations (experimental)
- Files:
  
  'smart_arm.xacro' - the robot model
  
  'smart_arm_robot.xacro' - definition of robot instance and interconnection to surrounding world
  
  'smart_arm_materials.xacro' - definition of material used on the arm model
5) OMPL based arm control

• Based on a part of ROS called *Arm navigation stack*  
  (On-line documentation: [http://www.ros.org/wiki/move_arm](http://www.ros.org/wiki/move_arm))
• Created by Zdeněk Materna
• Provides a standard interface for manipulators under ROS
• Supports:
  - Forward kinematics
  - Inverse kinematics
  - Trajectory planning
  - Collision avoidance according to collision model
  - Gripper control
  - Simulation
• Currently some issues
5) OMPL based arm control: Communication with controller (1/2)

• Basic communication is based on actions:
  - move_arm/goal (move_arm_msgs/MoveArmGoal)
    • A goal for move_arm to pursue in the world.
  - move_arm/cancel (actionlib_msgs/GoalID)
    • A request to cancel a specific goal.

• Configuration of the arm control stack:
  - ROS Parameter server is used
    ~move_arm_frequency (double, default: 50.0)
    ~group (string, default: "")
    ~ik_allowed_time (double, default: 2.0)
    ~trajectory_filter_allowed_time (double, default: 2.0)
    ~controller_action_name (string, default: /r_arm_controller/joint_trajectory_action/)
5) OMPL based arm control: Communication with controller (2/2)

- **Feedback:**
  - Based on ROS Topics
    - `move_arm/feedback (move_arm_msgs/MoveArmFeedback)`
      - The current state that the internal move arm state machine is in
      - Time to completion
    - `move_arm/status (actionlib_msgs/GoalStatusArray)`
      - Provides status information on the goals that are sent to the move_arm action.
    - `move_arm/result (move_arm_msgs/MoveArmActionResult)`

- **Internal messages used also for communication between components of the arm control stack:**
  - Defined in package 'move_arm_msgs' in Electric.
  - In Fuerte the package is called 'arm_navigation_msgs'.
6) OpenRAVE based arm control

- Created by Radim Luža
- Not as good integration to ROS as previous solution
- Currently incomplete
  - Missing a standard ROS interface
  - Wrist and gripper control is not completed
- Environment for the arm is separated from the rest of robot
  - OpenRAVE simulator is used independent on the rest of the robot
  - Surroundings of the arm is simulated separately
  - Currently robot parts surrounding the arm are node modeled yet
6) OpenRAVE based arm control: OpenRAVE description

- Toolkit for controlling the robot manipulators
- Founded by Rosen Diankov as his PhD. thesis
- Automates usage of analytic IK solvers
- Autogeneration of IK solvers is based on “templates”
  - Different kinds of generalized analytic solvers
  - They are parametrized by particular manipulator model
- How it works?
  - First we create a model of manipulator
    - (extended COLLADA 3.0 file format)
  - We manually choose a generalized solver
  - Solver hast to be “applicable” on the arm model – it means it has at least a number of DOF as solver needs
  - Then we applicate solver – we choose which DOFs we wish to solve
  - OpenRAVE autogenerates an anylitic solver as a code of a C library
    - Library is portable and without special dependencies
  - Then we can use generated solver – directly or through OpenRAVE toolkit
6) OpenRAVE based arm control: OpenRAVE as a manipulator control stack

- OpenRAVE supports:
  - Forward kinematics
  - Inverse kinematics
  - Trajectory planning
    - Based on BiRRT sampling algorithm from RRT algorithm family
  - Grasping planning
  - Simulation

- Supported programming languages
  - C++
  - Python

- Plug-in architecture
  - Easy adding interface to a new hardware

- Database of robot models
  http://openrave.org/docs/latest_stable/ikfast/robots/
7) OpenRAVE based arm control: Demonstration

- First we need to start servo drives controllers:
  
  
  $ rosrun btb_dynamixel smart_arm.launch

- Now we should see topics for servo drive controllers
  
  $ rostopic list | grep dynamixel

- We should see following topics (and others):

  /dynamixel/elbow_flex_controller/command
  /dynamixel/elbow_flex_controller/state
  /dynamixel/left_finger_controller/command
  /dynamixel/left_finger_controller/state
  /dynamixel/right_finger_controller/command
  /dynamixel/right_finger_controller/state
  /dynamixel/shoulder_pan_controller/command
  /dynamixel/shoulder_pan_controller/state
  /dynamixel/shoulder_pitch_controller/command
  /dynamixel/shoulder_pitch_controller/state
  /dynamixel/wrist_roll_controller/command
  /dynamixel/wrist_roll_controller/state
7) OpenRAVE based arm control: Demonstration

- For each joint there is a 'command' topic and a 'state' topic
  - 'command' topic - for sending commands for the joint
  - 'state' topic – for receiving feedback from arm joints

- We can set angle for a joint:
  
  $ rostopic pub -1 /dynamixel/shoulder_pan_controller/command std_msgs/Float64 -- 0.1

- We can watch state of the joint:
  
  $ rostopic echo /dynamixel/shoulder_pan_controller/state
7) OpenRAVE based arm control: Demonstration

• Starting OpenRAVE arm control stack:
  - Arm servo drives has to be connected and working - otherwise the arm control stack will not start.
    $ roslaunch btb_manipulator_openrave turtlebot_arm_control.launch

• Moving arm to some endpoint position:
  $ rostopic pub -1 /arm_control btb_manipulator_openrave/TB_arm_control -- {0.30,-0.15,0.20,1.2,0,0,5}

• Adding obstacle to environment model for the arm control:
  $ rostopic pub -1 /arm_obstacle_control btb_manipulator_openrave/TB_arm_obstacle_control -- '{ID: 5, x: 0.2, y: 0.3, z: 0.25, objtype: 1, x_dim: 0.1, y_dim: 0.1, z_dim: 0.1, a: 0, b: 0, c: 0, operation: 1}'

• Communication channels:
  - Arm control topic:
    /arm_control (type btb_manipulator_openrave/TB_arm_control)
  - Topic for feedback:
    /arm_state (type btb_manipulator_openrave/TB_arm_state)
  - Obstacle management topic:
    /arm_obstacle_control (type btb_manipulator_openrave/TB_arm_obstacle_control)
Thank you for your attention