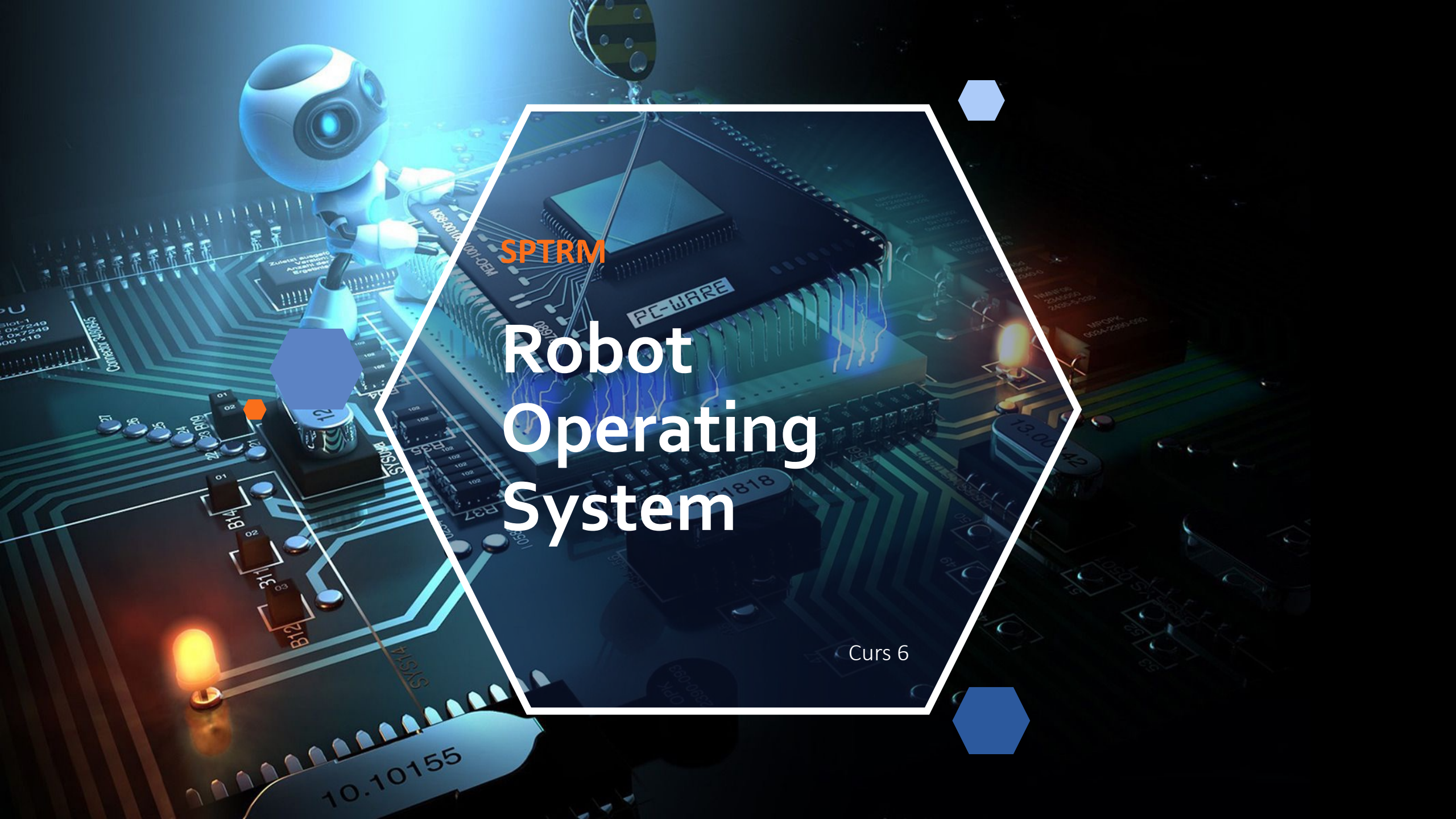


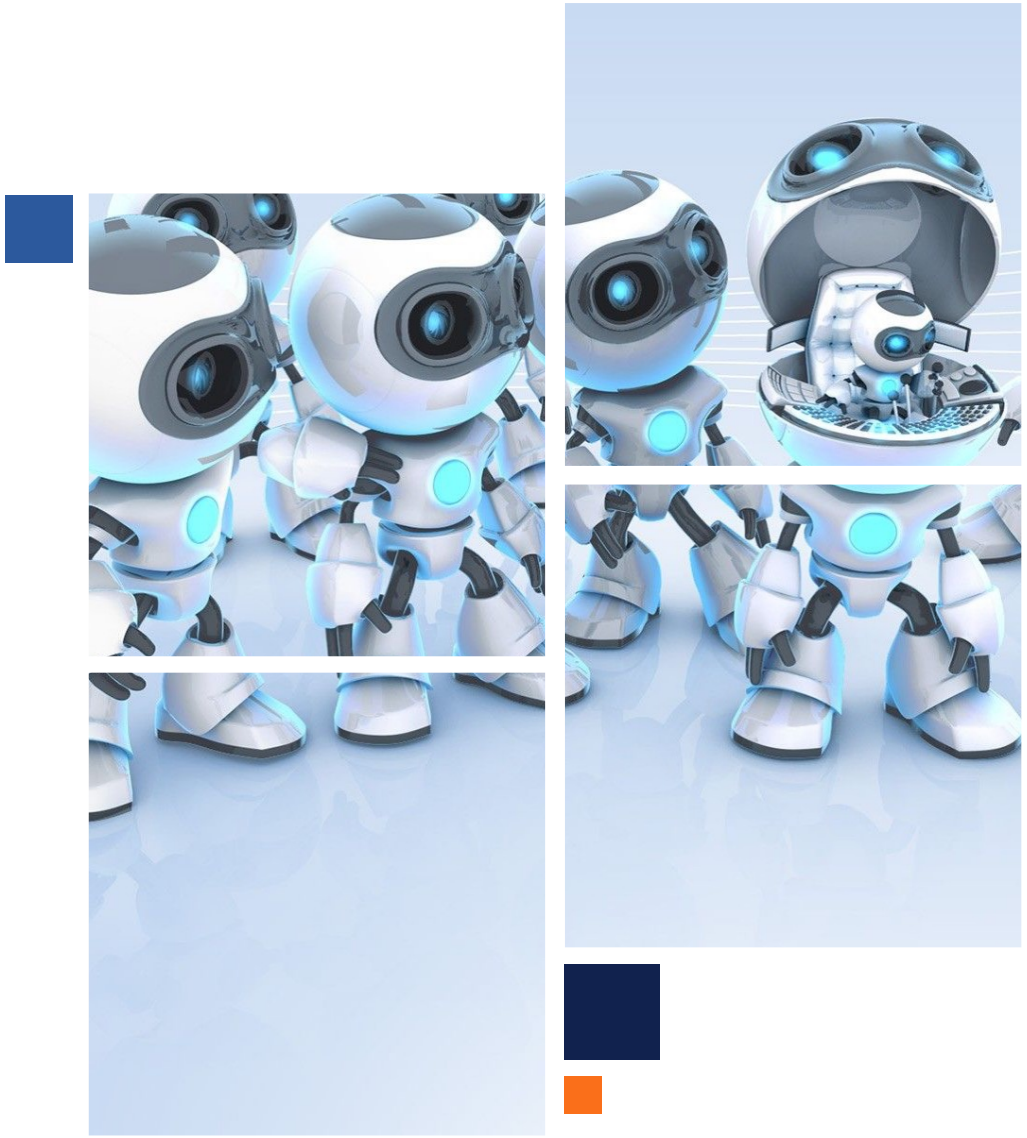
SPTRM

Robot Operating System

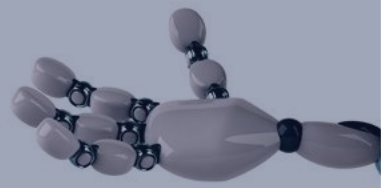
Curs 6

Agenda

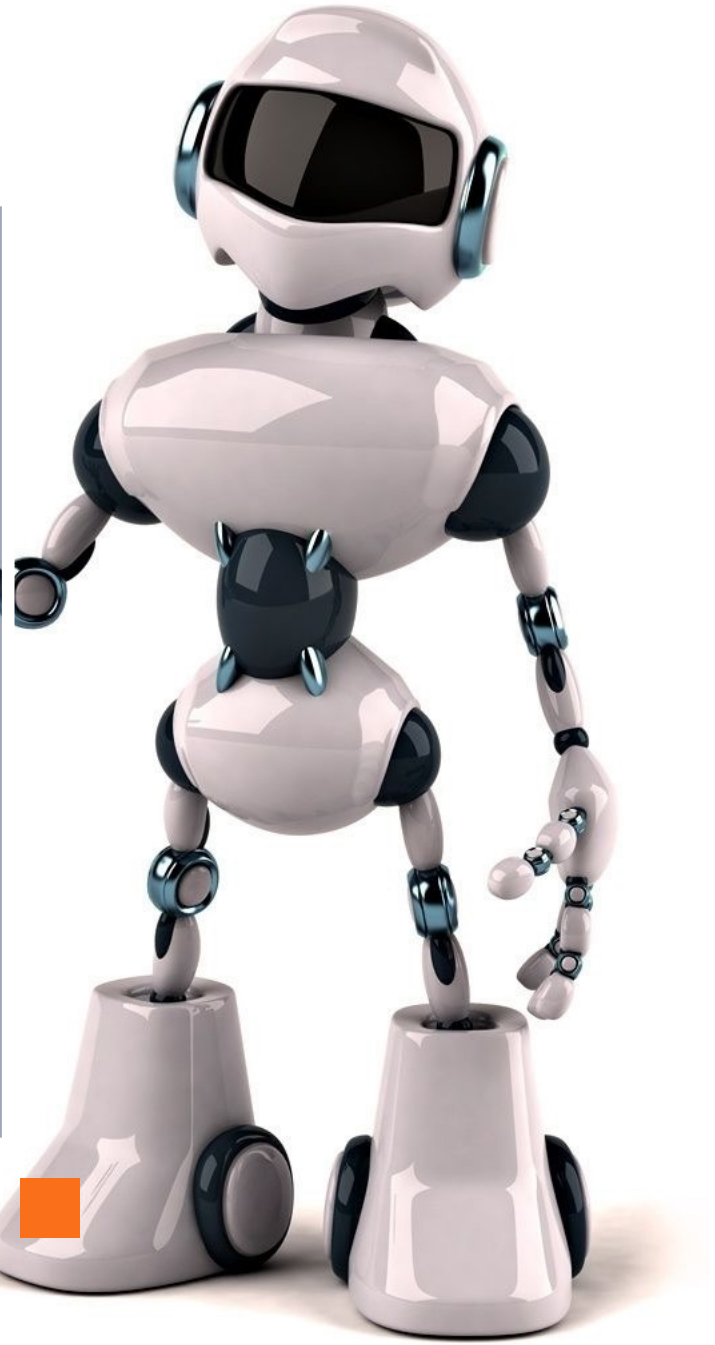
- Recap
- Controlling a robot
  - Manipulator. End effector
  - Motion planning
  - IK, FK
  - Grasp planning
- MoveIt!
  - Collision detection
  - Planning Scene
  - Setup Assistant Tool



Recap



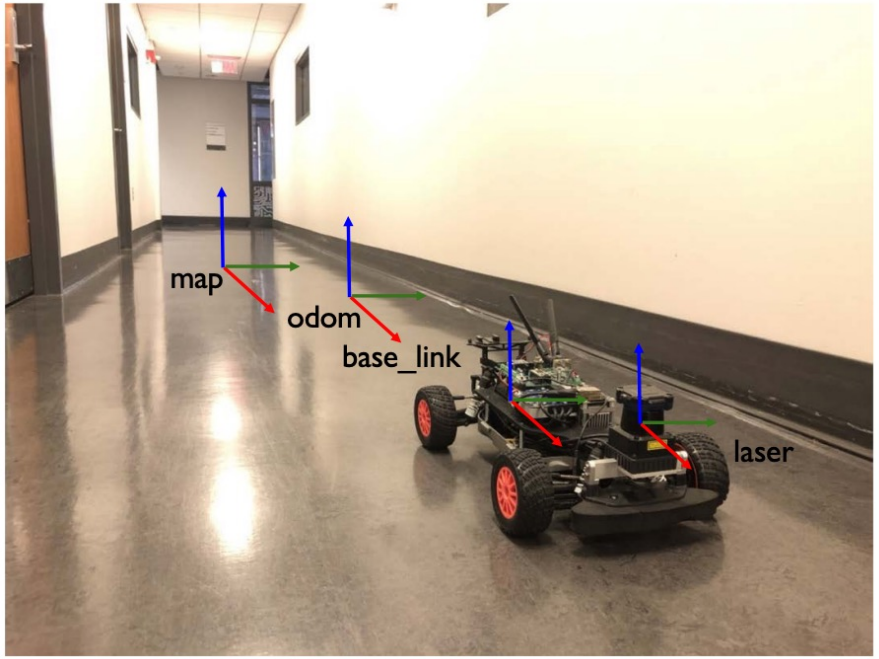
From the last episode…



ROS Coordinate Frames

- Usually in ROS one can find 4 coordinate frames
  - Map frame
  - Robot Base (base_link) frame
  - Tool frame
  - Odometry frame
- Odometry = distance measurement technique for vehicles and pedestrians

RGB → XYZ

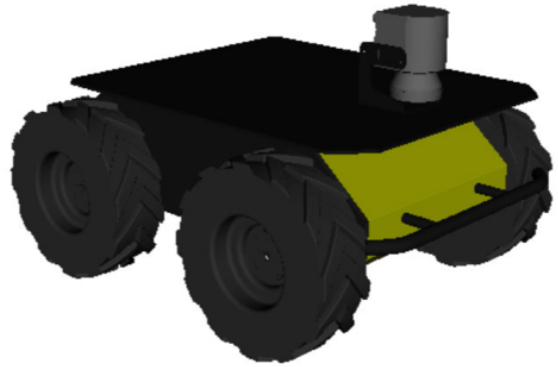


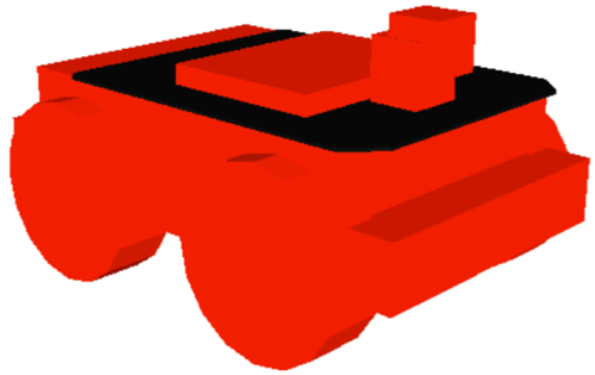
Robot models

- Unified Robot Description Format (URDF)

  http://wiki.ros.org/urdf
- XML file for representing a robot model
- Kinematic and dynamic description
- Visual representation (mesh)
- Collision model (primitives)
- URDF generation can be be scripted with XACRO

  http://wiki.ros.org/xacro





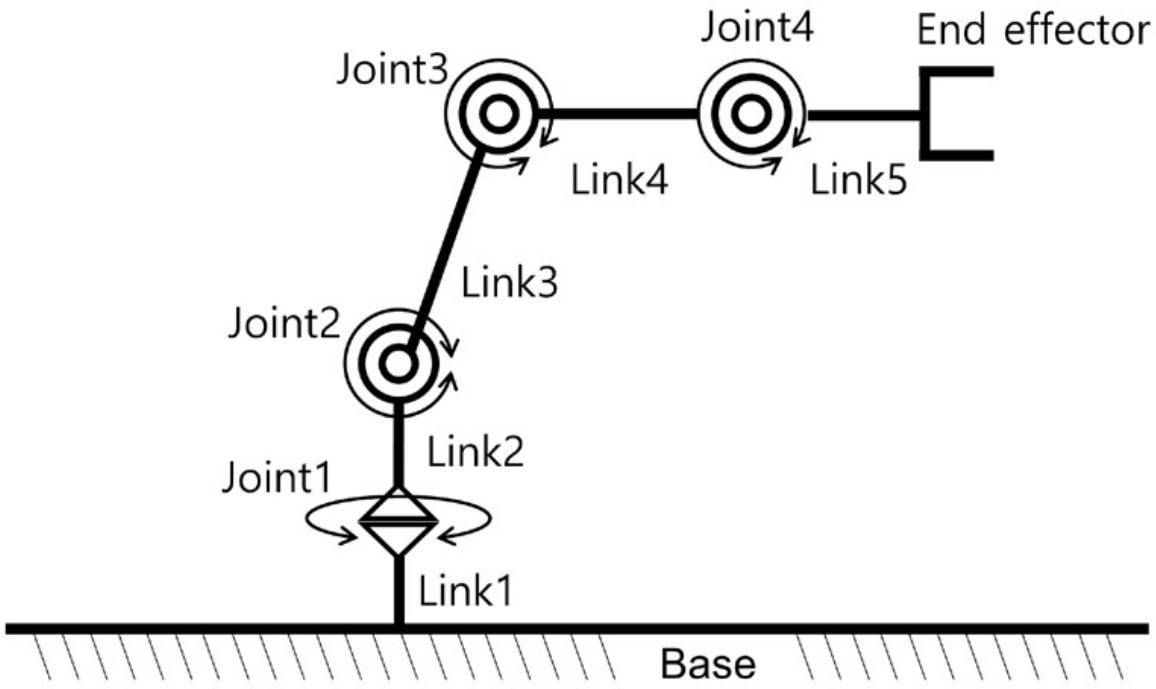
Controlling a robot

Robot Manipulation. Motion planning

Introduction

- **Manipulator** = robot created to perform repetitive tasks
- Usually used in factories on assembly lines
- Medical robots (surgeries)
- Hazardous environments (radioactive / medical)
- **End effectors**
  - defines the function performed by the manipulator
  - different types: grippers, welders, drills

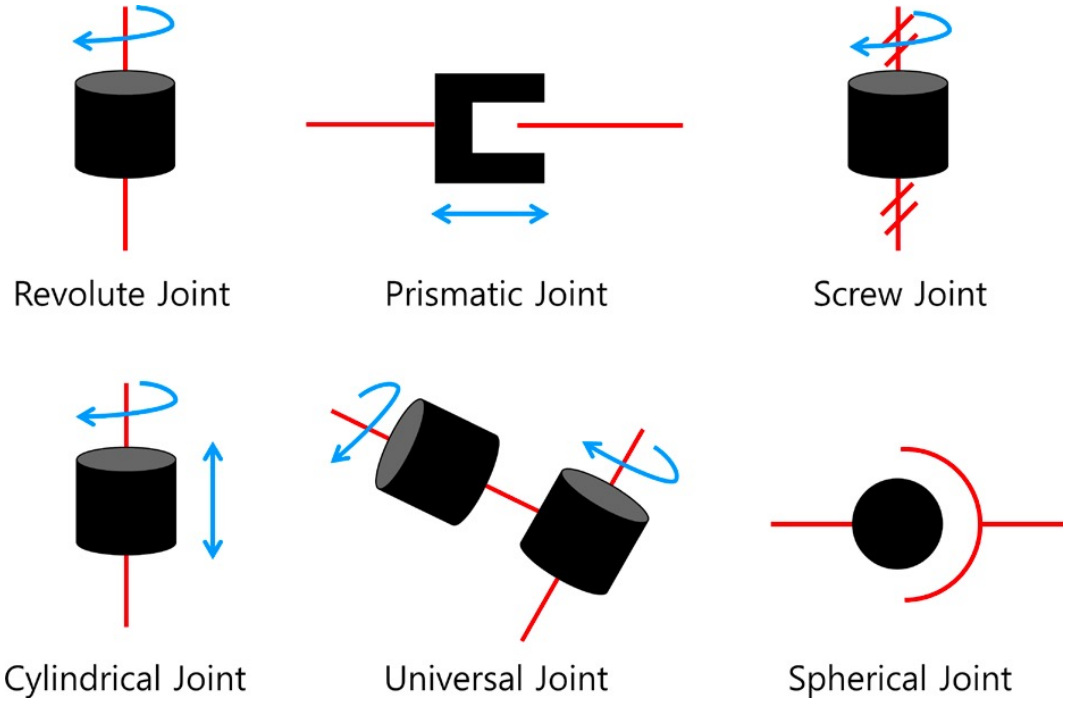
Basic structure



- Links
- Joints
- End effector (Tool)

Links and Joints

- Used to position the end effector
- Define the working space of the manipulator
- Links - multiple lengths
- Joints - different types



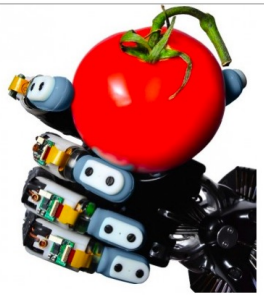
End effectors

- Grippers
  - Most common used for research
  - Perform a grasp action
  - Consists of fingers (rigid or mini-manipulators)
  - Electrical or pneumatically driven

Parallel jaw gripper

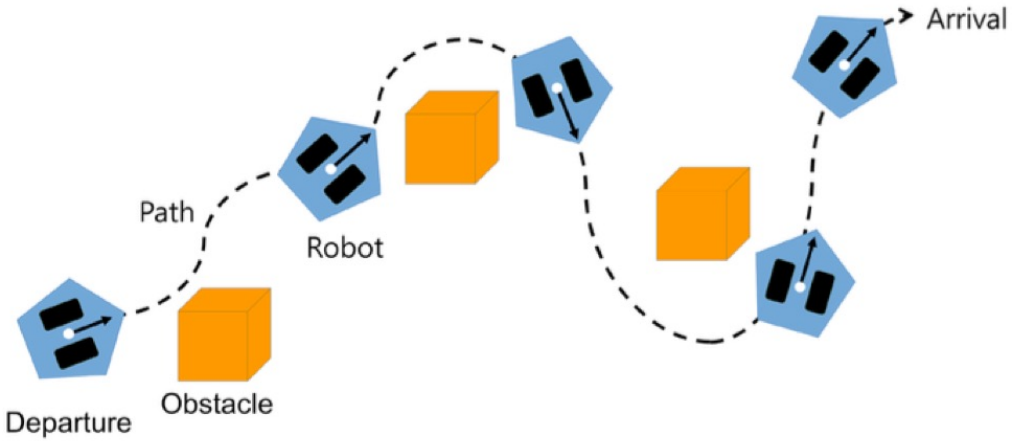
Barrett hand

Biomimetic hands

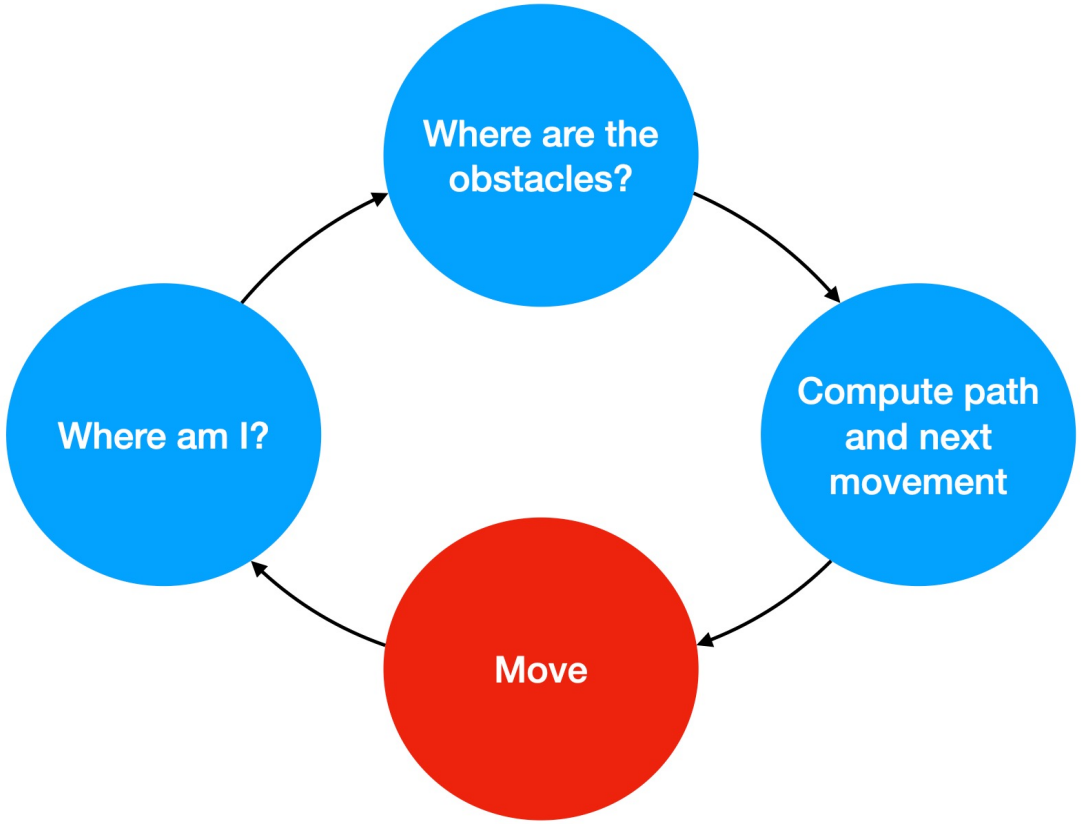


Motion planning. Navigation

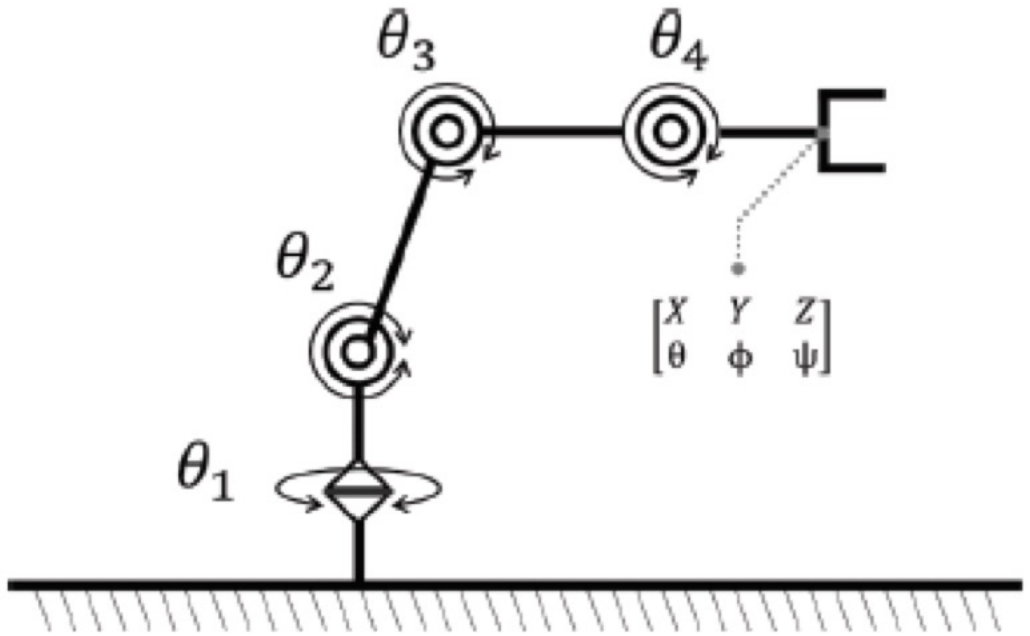
- **Motion planning** is a technique that finds the optimum path that moves the robot gradually from initial pose to goal pose without collisions with the world or itself
- Inputs
  - Initial pose of the robot
  - Goal pose of the robot
  - Geometrical description of the world
- **Navigation** is the movement of the robot from the current position to a defined destination, preferably using an optimized route



Continuous problem



Motion planning



- Constraints
  - **Position constraints** = restrict the position of a link
  - **Orientation constraints** = restrict the orientation of a link
  - **Visibility constraints** = restrict a point of a link to be visible (sensor viewable area)
  - **Joint constraints** = restrict joint limits
  - **Other constraints** = context dependent user specified constraints
- **Goal** => generate the suitable trajectory (in joint space) from the motion planner which obeys all the constraints
- Joint Space Control $\theta_1 \ \theta_2 \ \theta_3 \ \theta_4$
- Tool (/Task) Space Control $\begin{bmatrix} X & Y & Z \\ \theta & \phi & \psi \end{bmatrix}$

Robot pose

- pose = position (x, y, z) + orientation (w)
- outdoor: GPS, DGPS (Differential GPS)
- indoor: dead reckoning (DR) method
  - starts from a fixed known point
  - calculates the new position based on data recorded by the robot (direction, speed, etc)
  - increased accuracy by using inertial information

Forward Kinematics (FK)

- **Forward kinematics** refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameter

- Used in robotics, computer games, animations

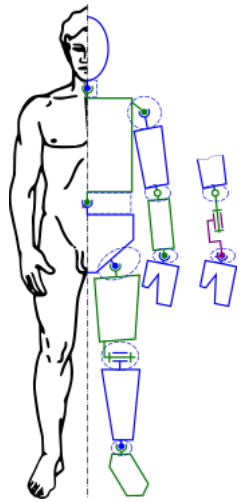
- **Kinematics equations of the serial chain** = sequence of transformations obtained by using a rigid transformation [Z] to characterize the relative movement allowed at each joint and separate rigid transformation [X] to define the dimensions of each link

$$[T] = [Z_1][X_1][Z_2][X_2]\dots[X_{n-1}][Z_n],$$

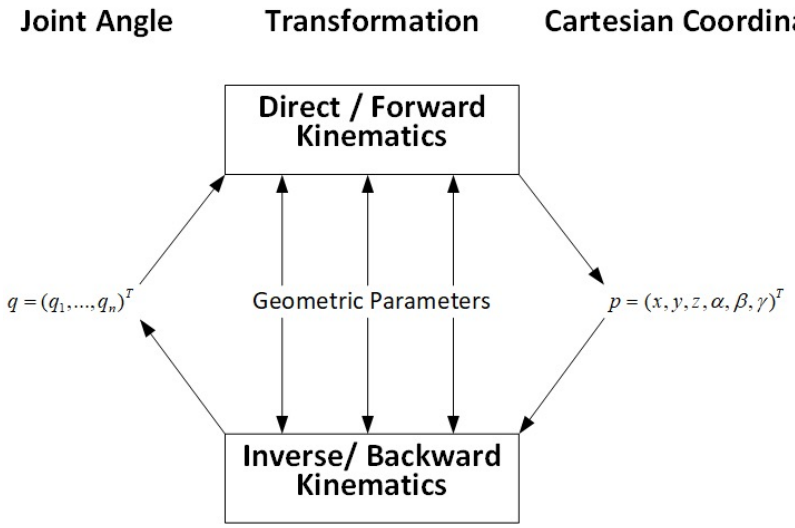
where [T] is the transformation locating the end-link, joint matrices [Z] and link matrices [X]

Inverse Kinematics (IK)

- **Inverse kinematics** is the mathematical process of calculating the variable joint parameters needed to place the end of a kinematic chain, such as a robot manipulator or animation character's skeleton, in a given position and orientation relative to the start of the chain
- Much more complicate than FK
- Can be used also to recover the movements of an object in the world from some other data (e.g. film) => movie animation
- Determines the joint parameters that provide a desired configuration (position and rotation) for the robot’s end-effector
- Goal for the motion planning
- Different types of algorithms:
  - Analytic approach (IKFast): generate a solver
  - Numerical approach: Jacobian inverse technique
  - Heuristic methods: Cyclic Coordinate Descent (CCD), Forward And Backward Reaching Inverse Kinematics (FABRIK)

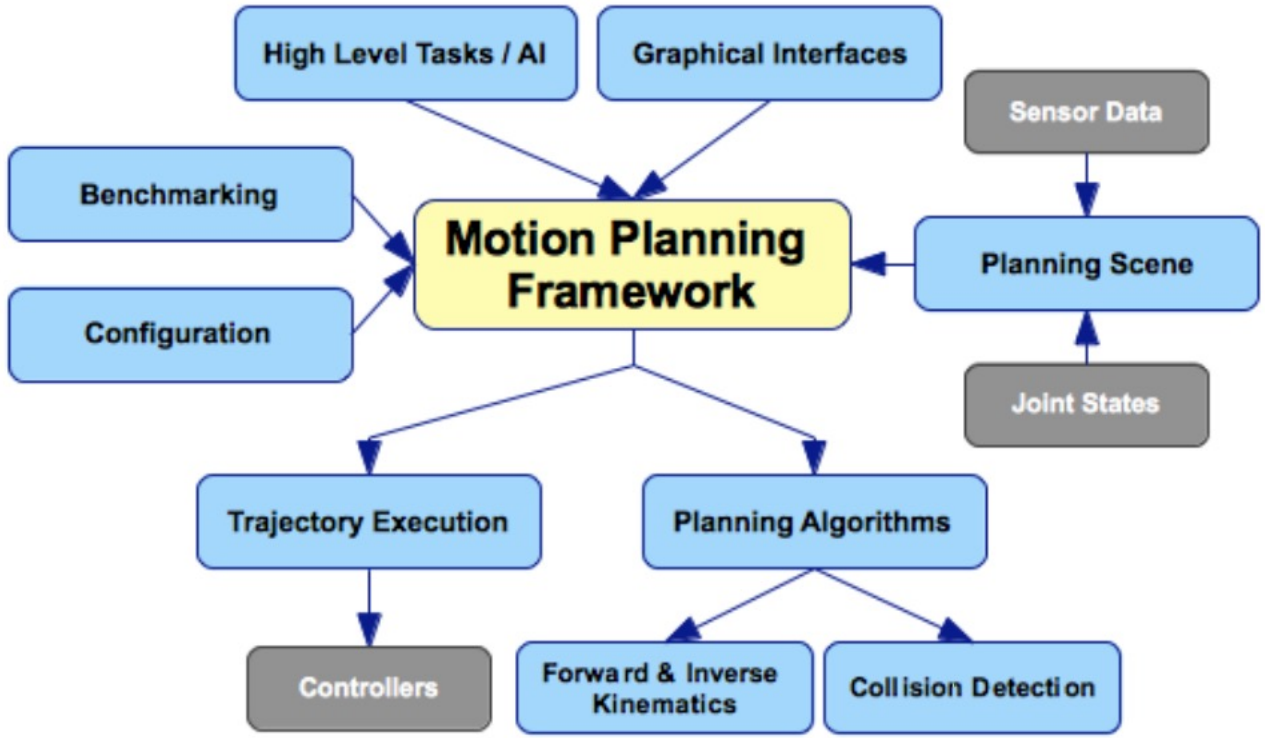


FK vs IK



Forward Kinematics	Inverse Kinematics
Compute tool position based on the angle of rotation of each joint	Compute the angle of rotation of each joint in order for the tool to be in a given position
$\begin{matrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\end{matrix} \rightarrow \begin{bmatrix} X & Y & Z \\ \theta & \phi & \psi \end{bmatrix}$	$\begin{bmatrix} X & Y & Z \\ \theta & \phi & \psi \end{bmatrix} \rightarrow \begin{matrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\end{matrix}$
Single solution	Mutiple or no solution
Simple algorithm (matrix multiplication)	Intense computing (complexity depends on the number of joints)

Motion planning framework



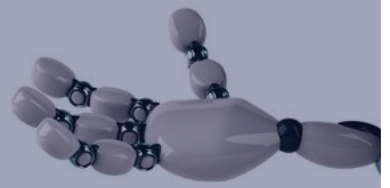
Grasp planning

- Detect possible grasps based on object dimensions and function to be performed
- Objects are modelled as a set of shape primitives (spheres, cylinders, cones and boxes)
- Planning for each finger depending on the grasp type
- Collision is good! - in case of grasping, collision means contact with the object
- Usually the final position is located inside the object
- **GraspIt!** = developed by Columbia University Robotics Group and similar to MoveIt!

Grasp taxonomy

	Power						Intermediate			Precision				
Opp:	Palm			Pad			Side			Pad				Side
VF:	3-5	2-5		2	2-3	2-4	2-5	2	3	3-4	2	2-3	2-4	2-5	3
Thumb Abducted		1. Large Diameter 3. Medium Wrap	2. Small Diameter 10. Power Disk 11. Power Sphere	31. Ring	28. Sphere 3 Finger	18. Extension Type 26. Sphere 4 Finger	19. Distal Type	23. Adduction Grip		21. Tripod Variation	9. Palmar Pinch 24. Tip Pinch 33. Inferior Pincer	8. Prismatic 2 Finger 14. Tripod	7. Prismatic 3 Finger 27. Quadpod	6. Prismatic 4 Finger 12. Precision Disk 13. Precision Sphere 22. Parallel Extension	20. Writing Tripod
Adducted Thumb	17. Index Finger Extension	4. Adducted Thumb 15. Fixed Hook	5. Light Tool 30. Palmar					16. Lateral 29. Stick 32. Ventral	25. Lateral Tripod		Non-Prehensile 34. Lift 35. Push				

Motion planning in ROS



MoveIt!

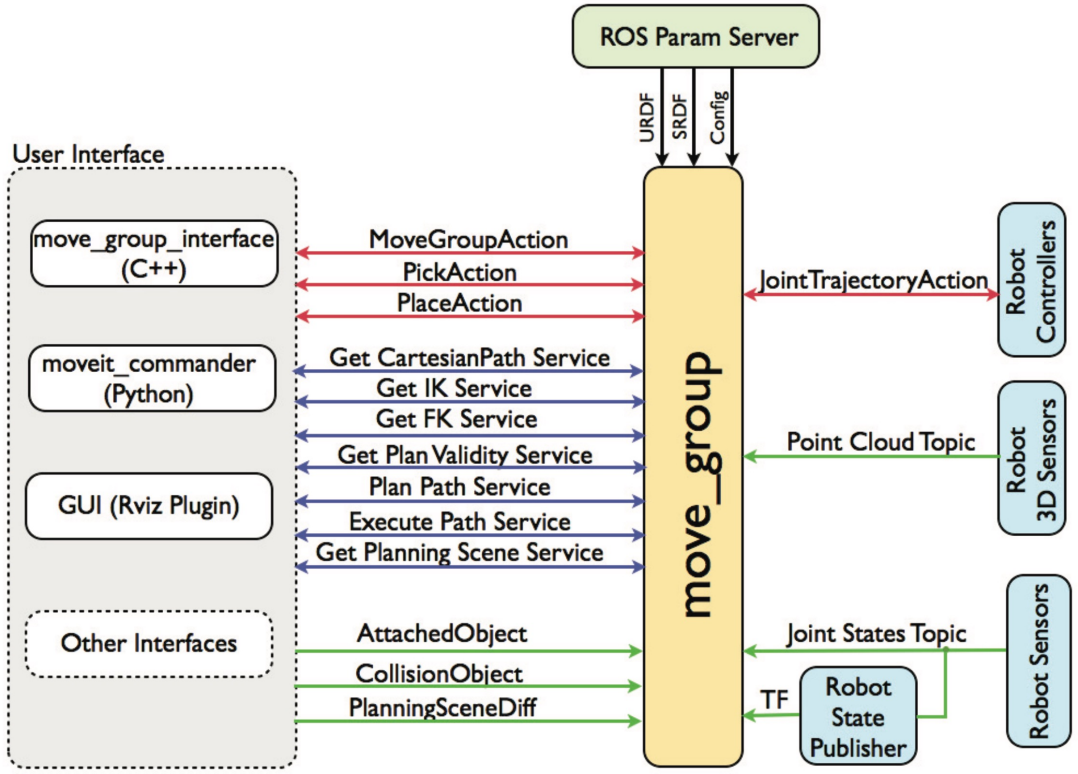
MoveIt!

- **MoveIt!** - set of packages and tools for doing mobile manipulation in ROS
- Supports multiple robots
- Supports multiple actions and scenarios:
  - Pick and place
  - Grasping
  - Motion planning using invers kinematics

- http://moveit.ros.org

- State-of-the-art software for
  - Motion planning
  - Manipulation
  - 3D perception
  - Kinematics
  - collision checking
  - control
  - Navigation
- Vizualization
  - Dedicated GUI
  - RViz plugin

MoveIt! architecture

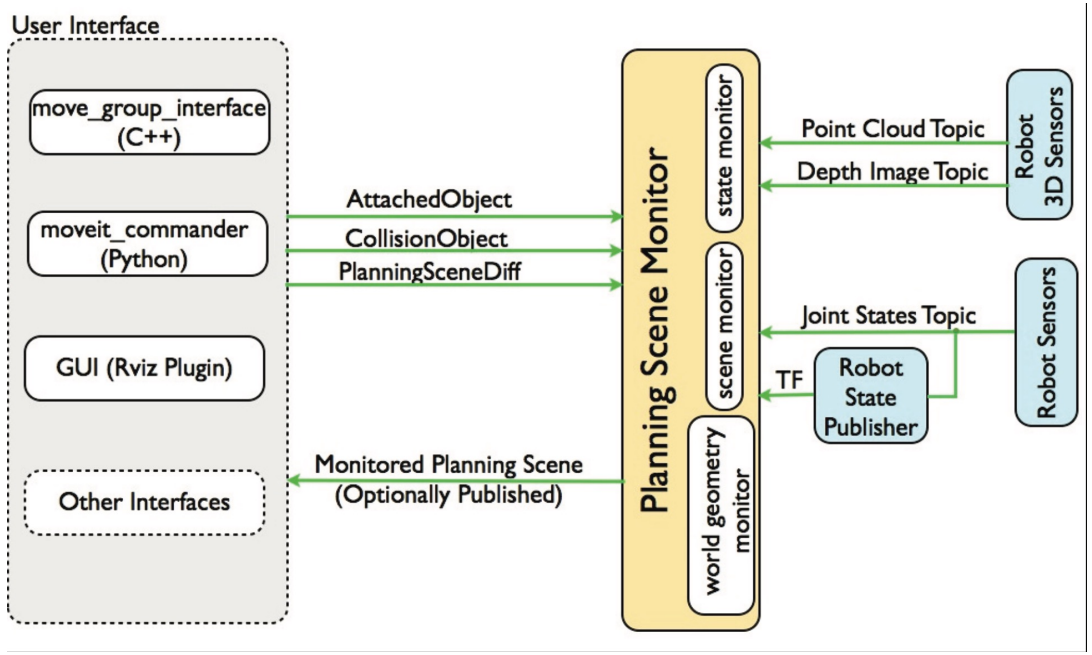


- move_group
  - the heart of MoveIt!
  - integrates the various components of the robot
  - delivers actions/services according to the user's needs
  - collects robot information via topics and services:
    - point clouds
    - joint state
    - transform frames (TF)
    - URDF, SRDF and other configuration files
  - uses plugins for
    - kinematics solvers (FK, IK)
    - motion planning
    - action tracking

MoveIt! motion planning

- Motion planers can be integrated as plugins
- Default motion planner – OMPL (http://ompl.kavrakilab.org)
- Start the motion planning request by specifying the planning requirements (e.g. pick and place operation)
- Request adapters
  - preprocess the request (e.g. correct a small violation in the joint states)
  - postprocess the response (e.g. convert to time-parametrized trajectory)

MoveIt! planning scene



- Planing scene represents the world around the robot and stores also the state of the robot itself
- Planning scene monitor
  - Reads the joint_states from the robot => state monitor
  - Reads the sensor information => scene monitor
  - Reads the world geometry => world geometry monitor
- Octomap
  - Outcome of the planning scene monitor
  - 3D representation of the environment

MoveIt! kinematics handling

- Great flexibility by using robot plugins to switch IK algorithms
- Default IK solver => numerical jacobian-based solver
- Analytic solvers are faster than numerical ones (**IKFast** package)
- Depends hugely on the number of degree of freedoms (DOFs)
- FK are directly integrated in MoveIt!

MoveIt! collision checking

- Flexible Collision Library (FCL) package => used to find collisions inside a planning scene
  - Meshes
  - Primitive shapes (boxes, cylinders, cones, spheres)
  - Octomap

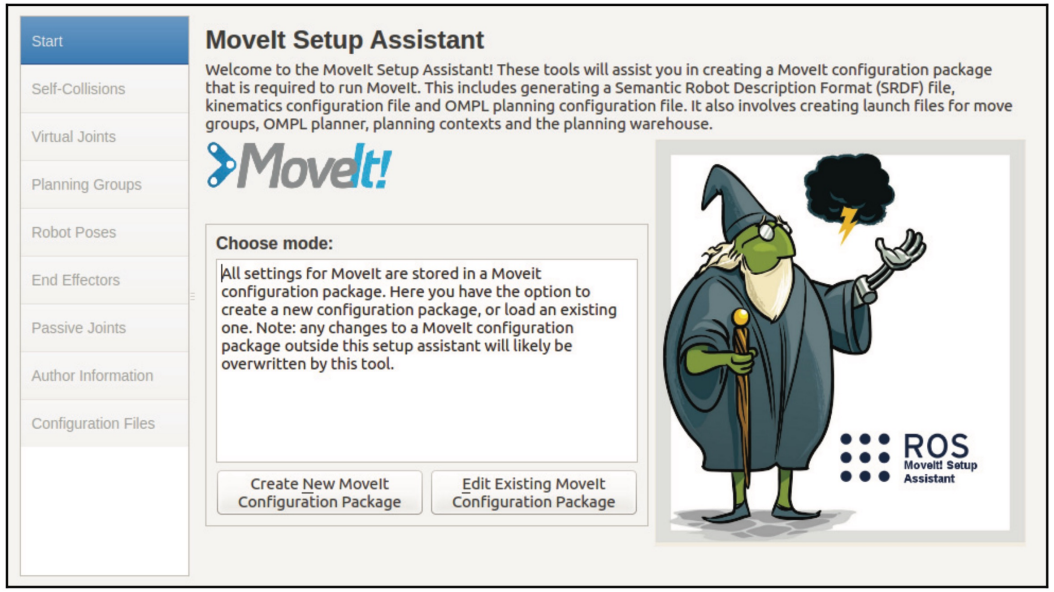
- Collision checking is one of the computationally expensive tasks during motion planning

- Allowed Collision Matrix (ACM) reduces the computation complexity by defining if a collision needs to be checked for two pairs of bodies (0 – yes, 1 – no), e.g. if 2 objects are far a way so no need to check collision

MoveIt! Setup Assistant tool

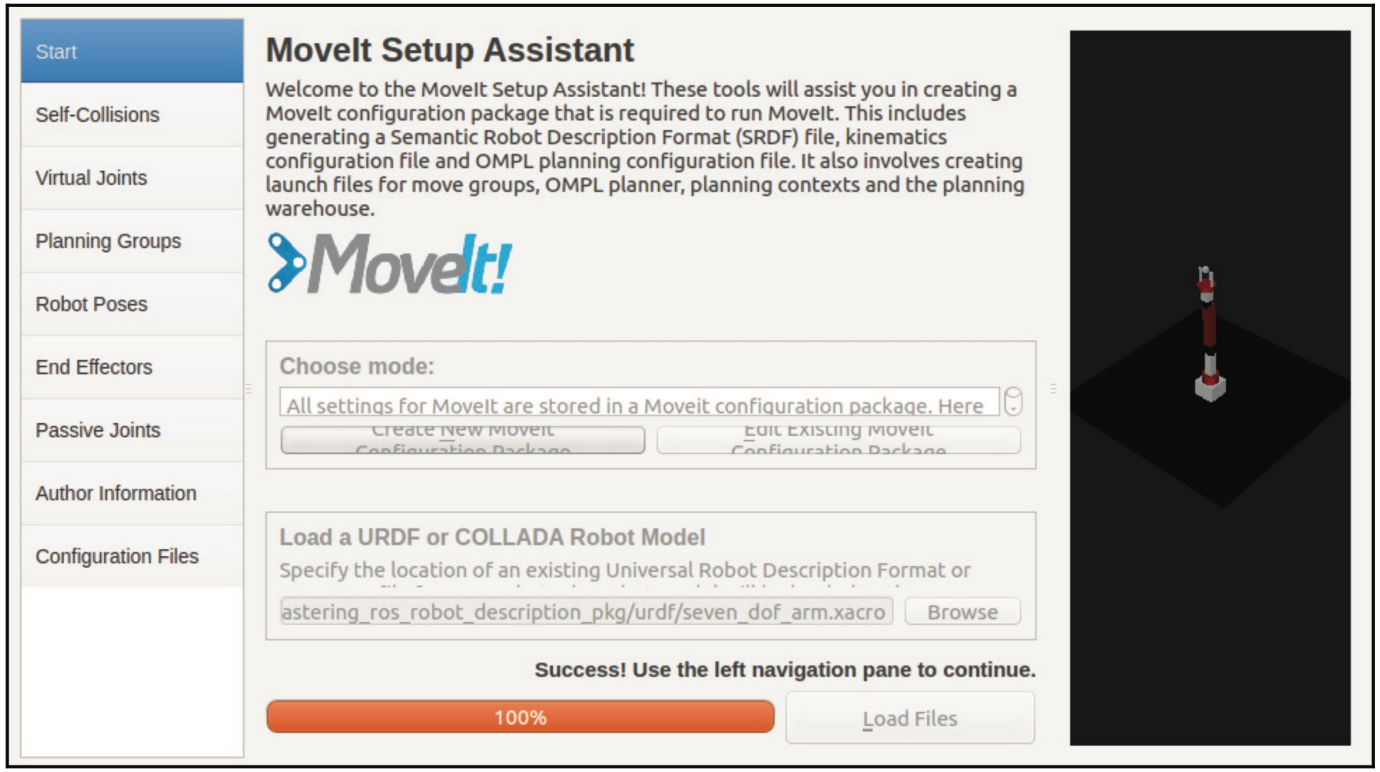
- GUI for configuring any robot in MoveIt!
- Generates the SRDF, configuration files, launch files and scripts from the URDF
- Encapsulated all information into configuration packages that can be edited afterwards

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```



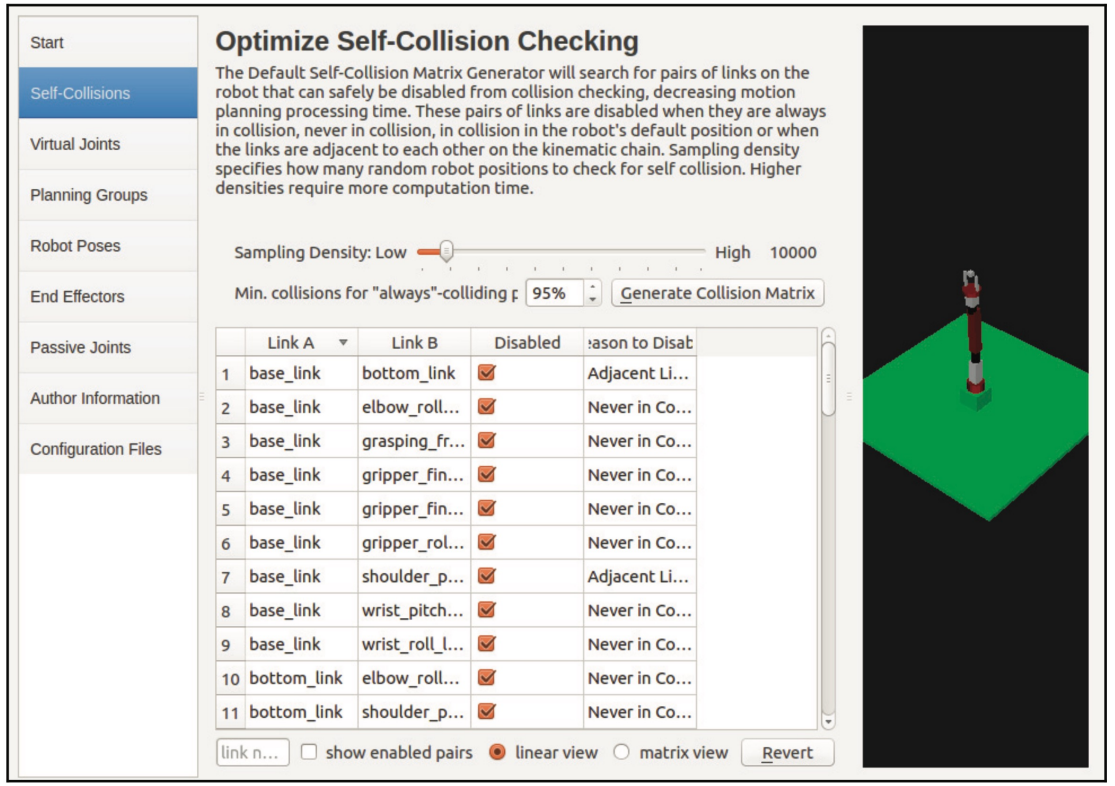
1. Loading the URDF file

- First step asks for the URDF model
- xacro can also be used
- If the model is successfully parsed, it will be displayed in the window



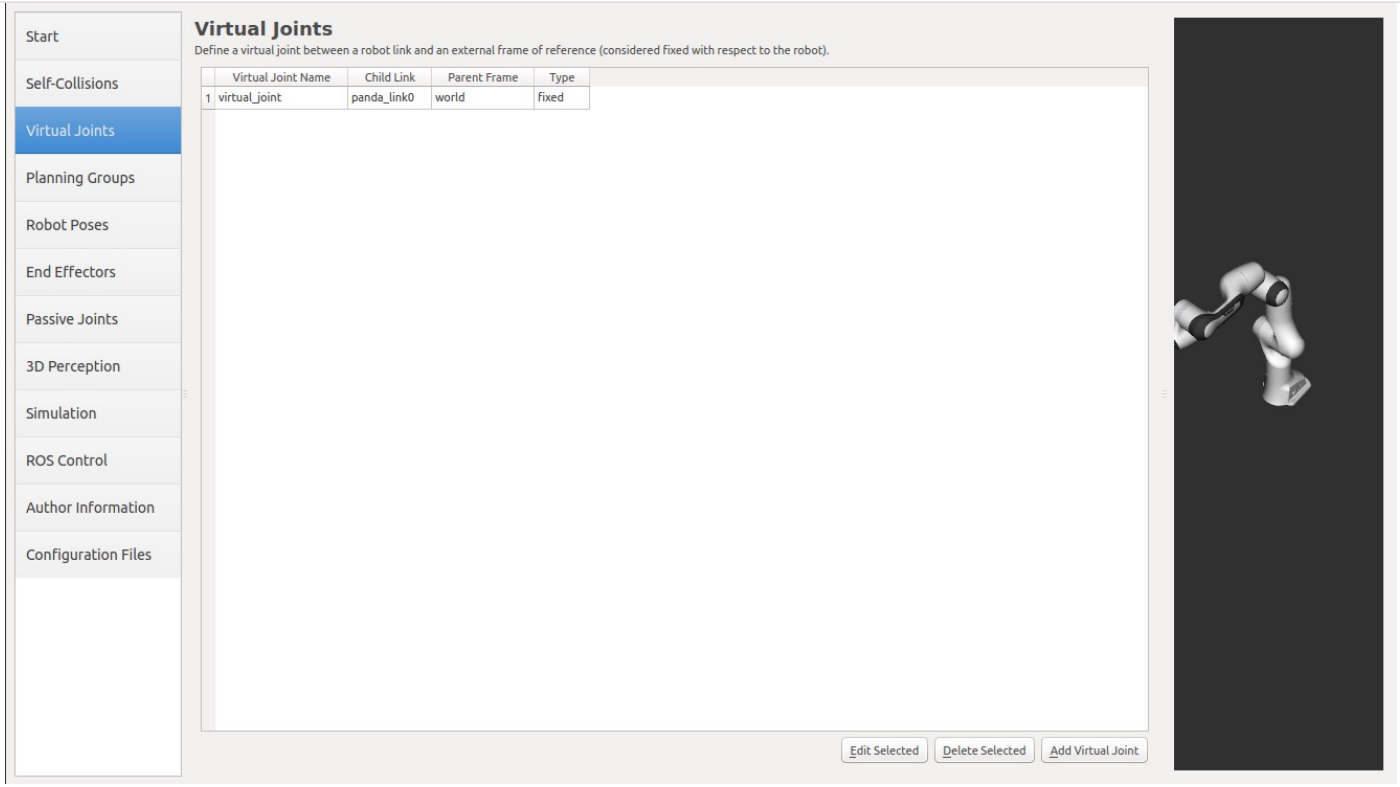
2. Generating the self-collision matrix

- Searches for pairs of links for which collision checking is disabled
- Analyzes each link pair and marks them as: Always in collision, Never in collision, Default in collision, Adjacent links, Sometimes in collision
- The sampling density is the number of random positions to check for self-collision



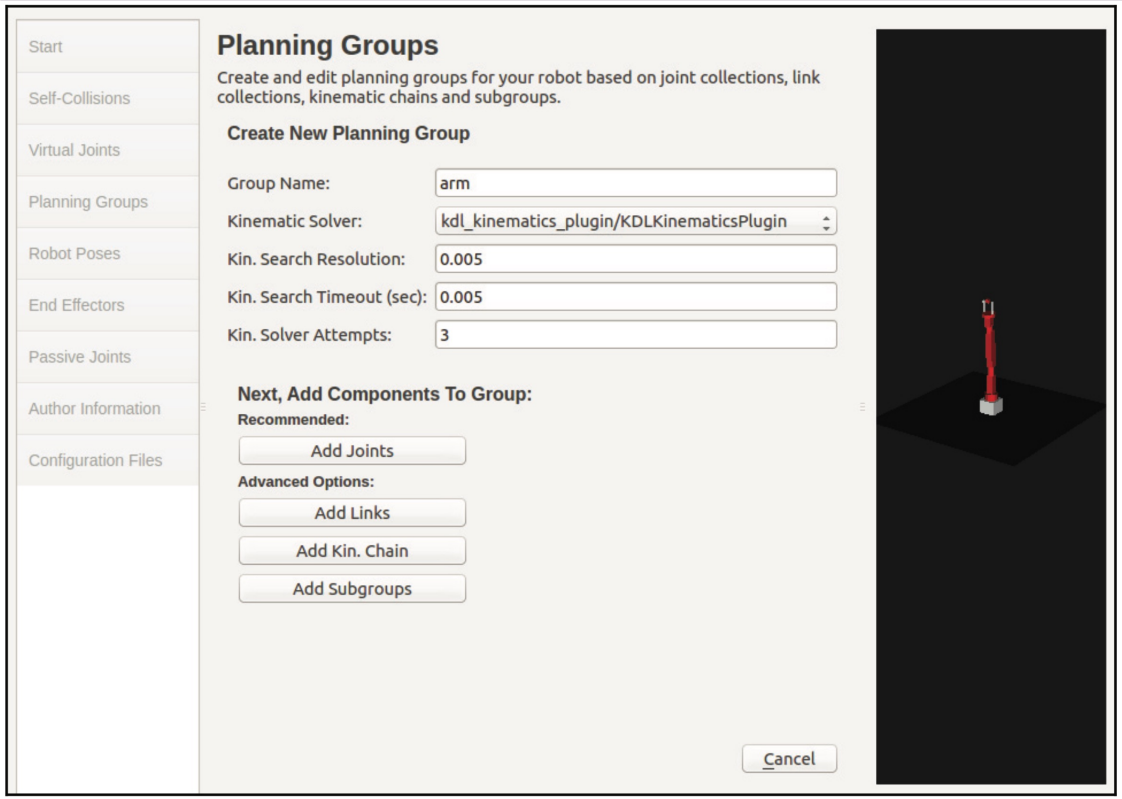
3. Adding virtual joints

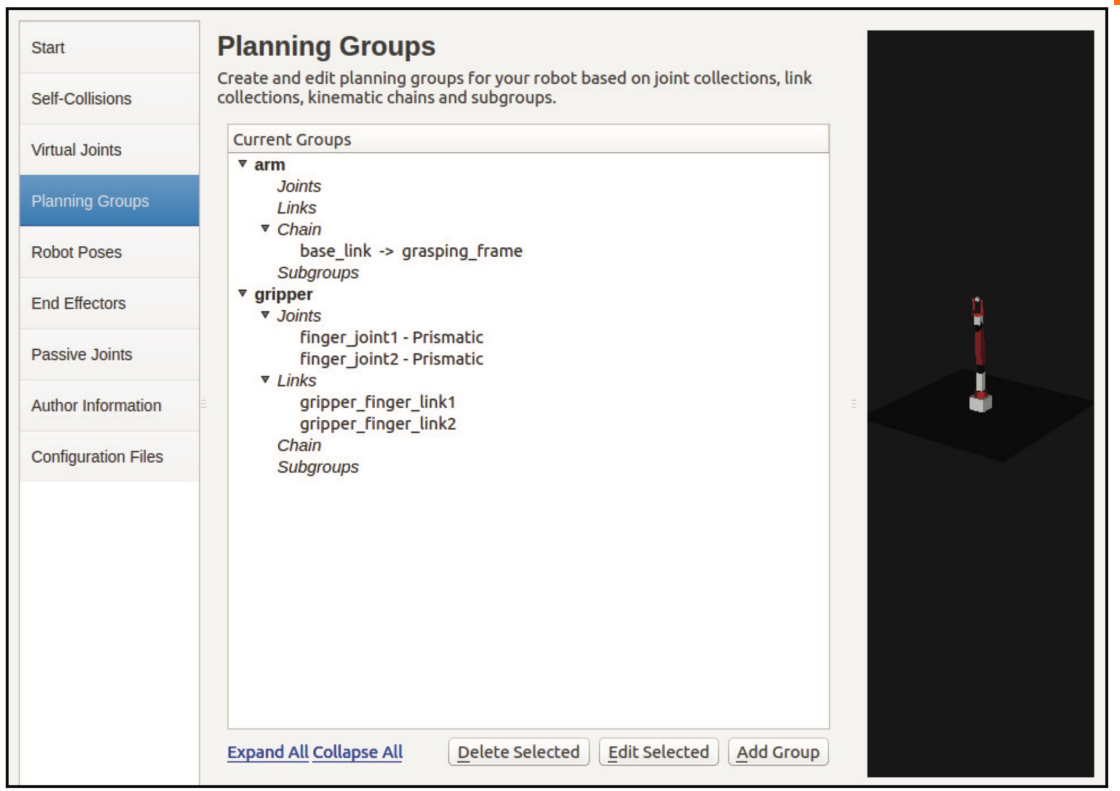
- Virtual joints attach the robot to the world
- They are not mandatory for a static robot which does not move
- Required when the base position of the robot is not fixed
- Defined with respect to the odometry frame (odom)



4. Adding planning groups

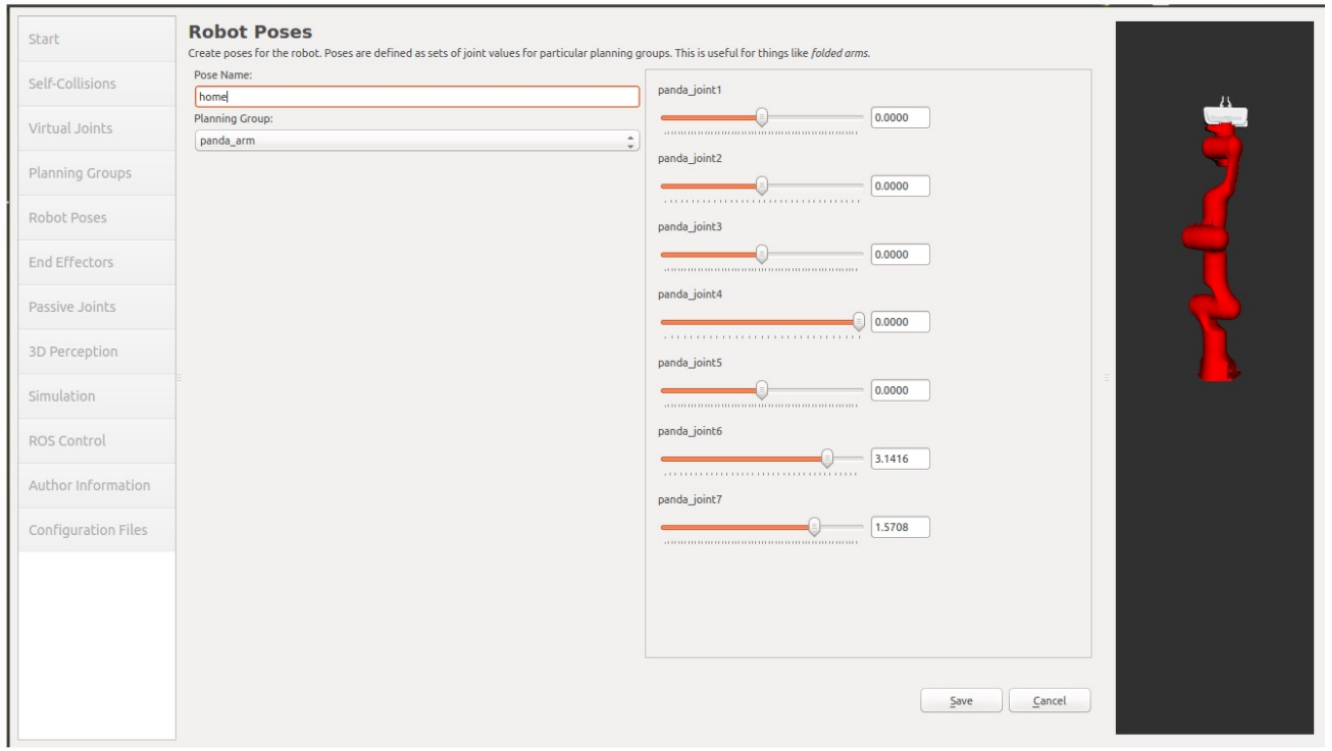
- A **planning group** is a group of joints/links in a robotic arm which plans together to achieve a goal position of a link or the end effector
- Multiple groups can be created (e.g.: one for the arm and one for the gripper)
- Define the kinematic chain
- Choose kinematic solver





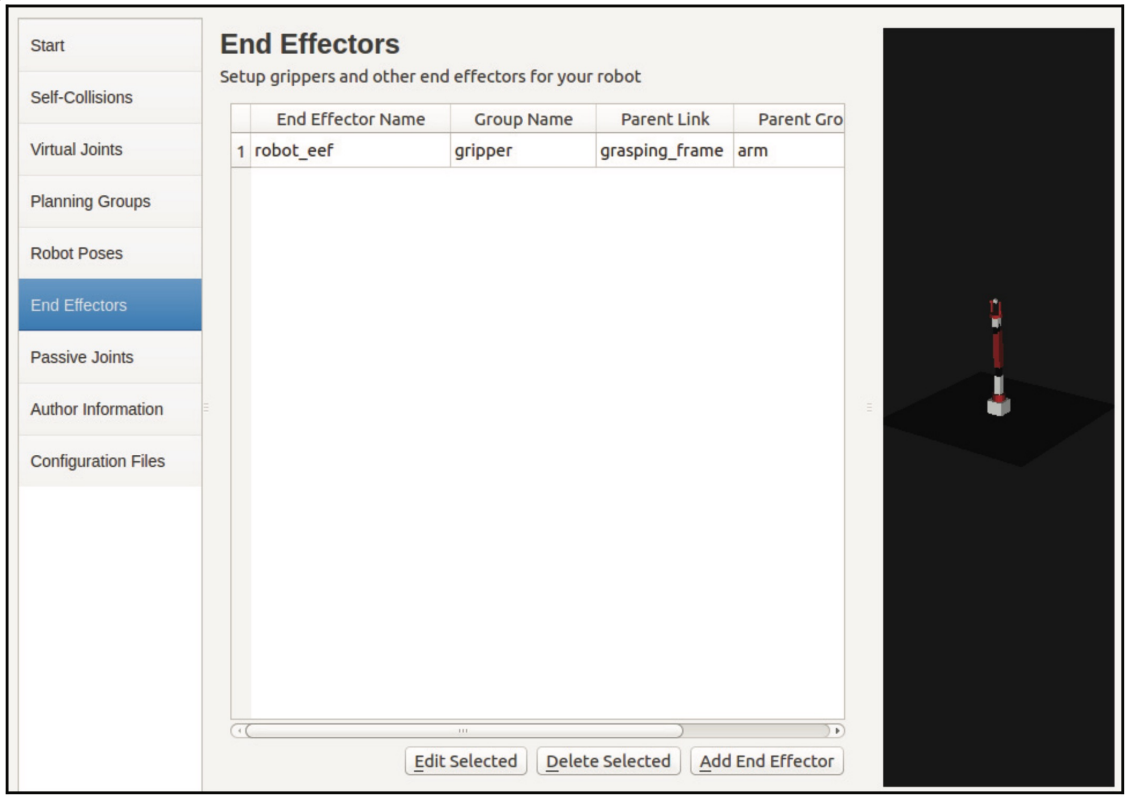
5. Adding robot poses

- Add certain fixed poses in the robot configuration (e.g. home pose)
- Can be called directly



6. Setting up the robot end effector

- Name the robot end effector and assign the end-effector group, the parent link, and the parent group
- Multiple end effectors can be defined

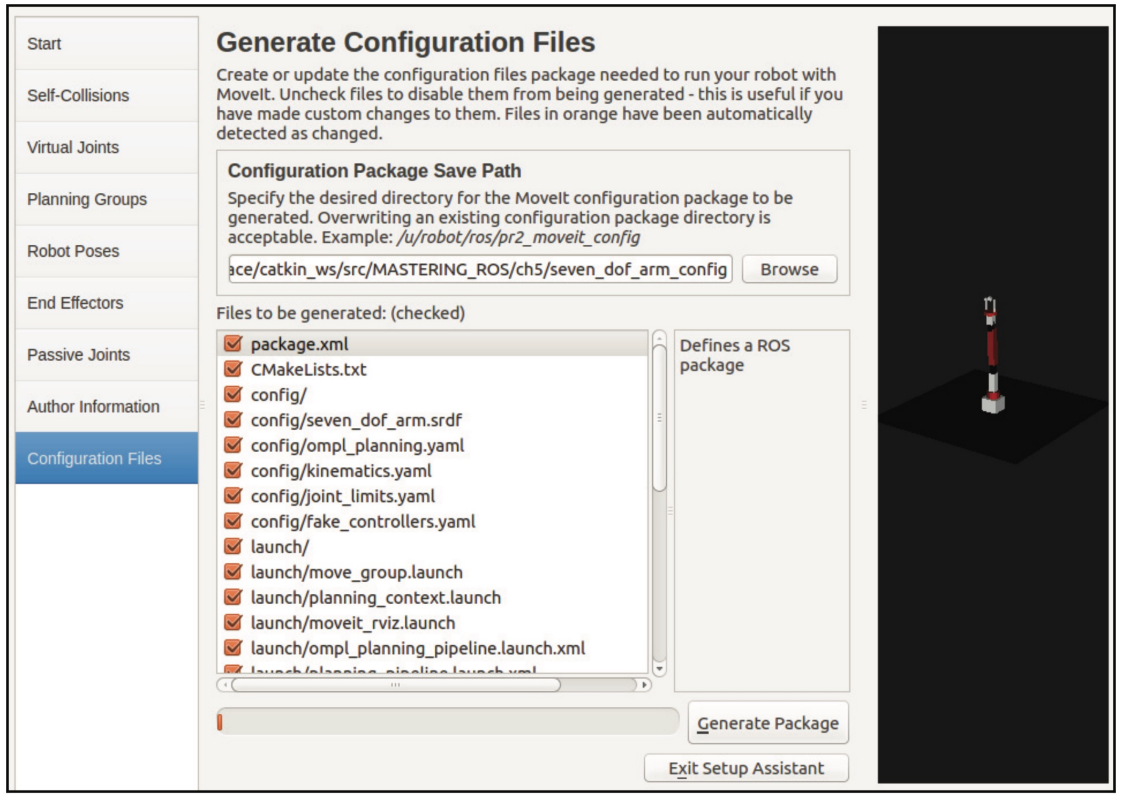


6. Other configurations

- Passive joints
  - the joints do not have any actuators (e.g. caster wheels)
  - This tells the planners that they cannot (kinematically) plan for these joints because they can’t be directly controlled
- 3D Perception
  - set the parameters of the YAML configuration file for configuring the 3D sensors
- Gazebo simulation
  - simulate the robot in Gazebo
  - generate a new Gazebo compatible urdf (if needed)
- ROS Control
  - set of packages that include controller interfaces, controller managers, transmissions and hardware_interfaces
  - auto generate simulated controllers to actuate the robot joints
- Author Information
  - author information required by Catkin for publishing purposes

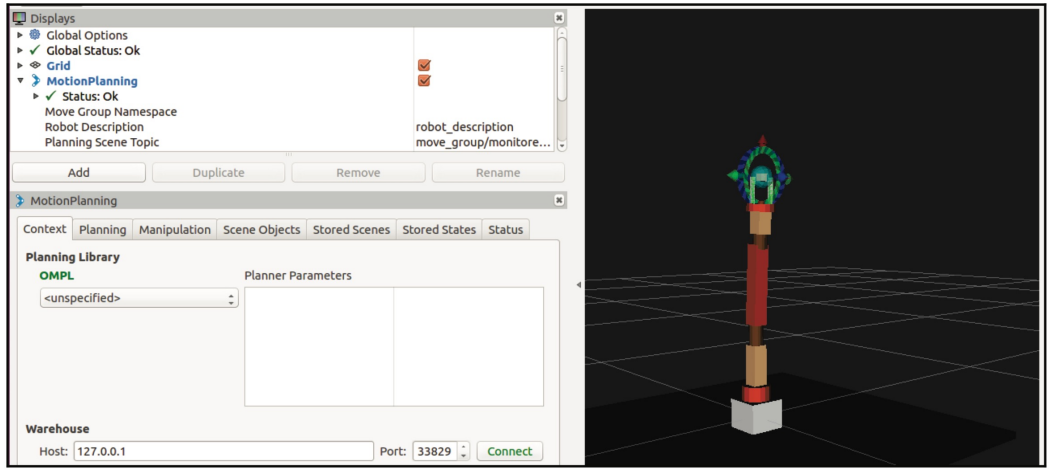
8. Generating configuration files

- generate a configuration package which contains the file needed to interface MoveIt!

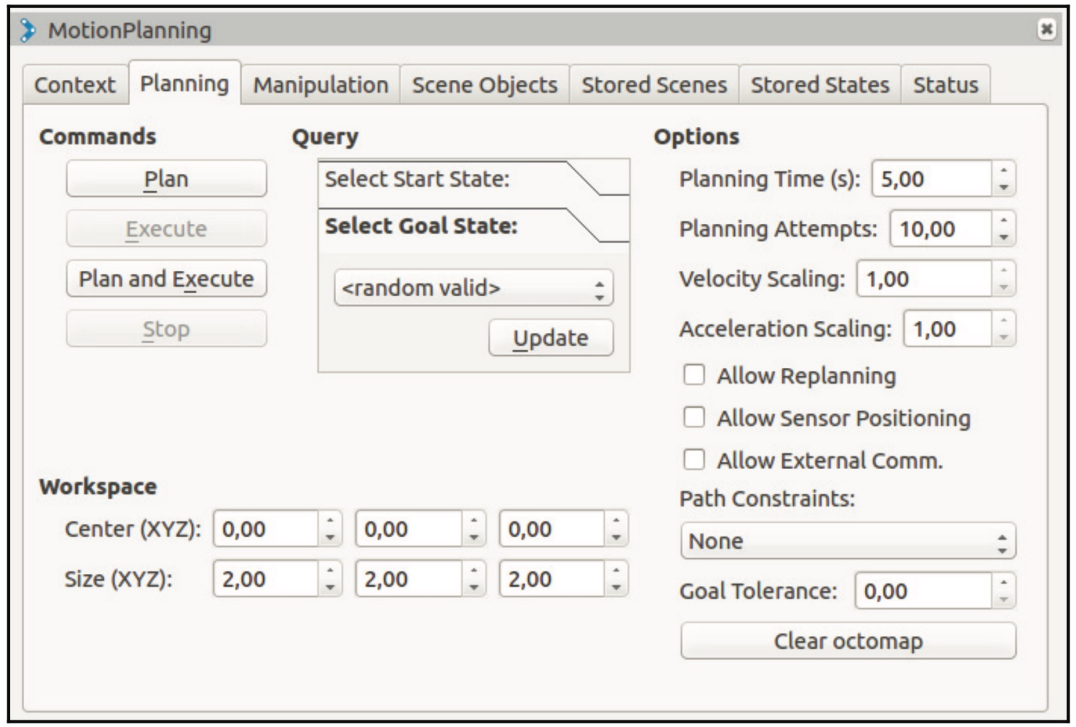


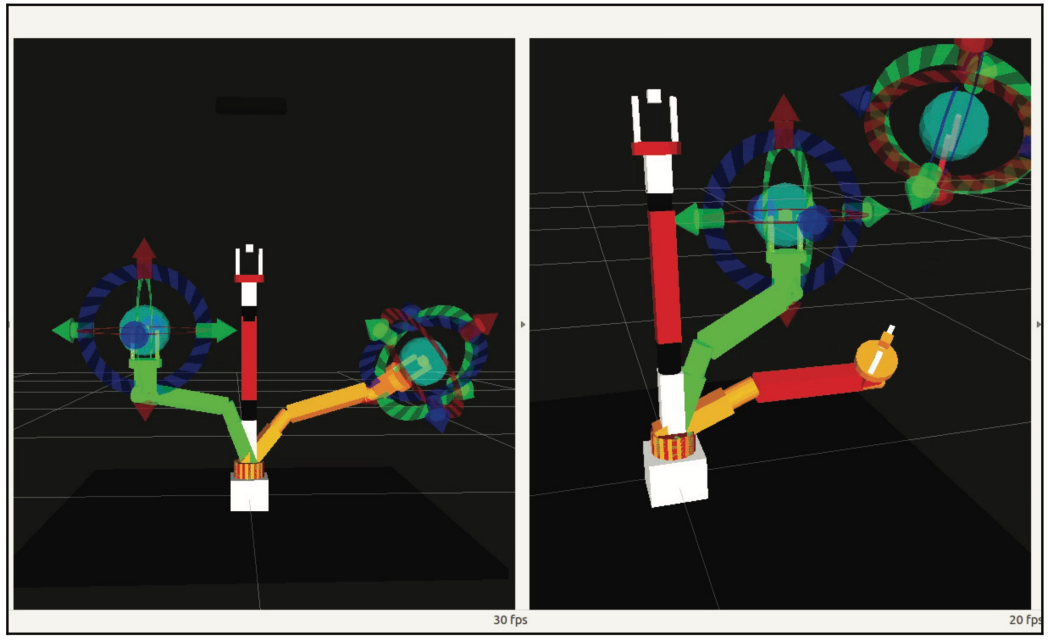
RViz plugin

- Create new planning scenes
- Generate motions plans
- Add new objects
- Visualize planning output
- Directly interact with visualized robot

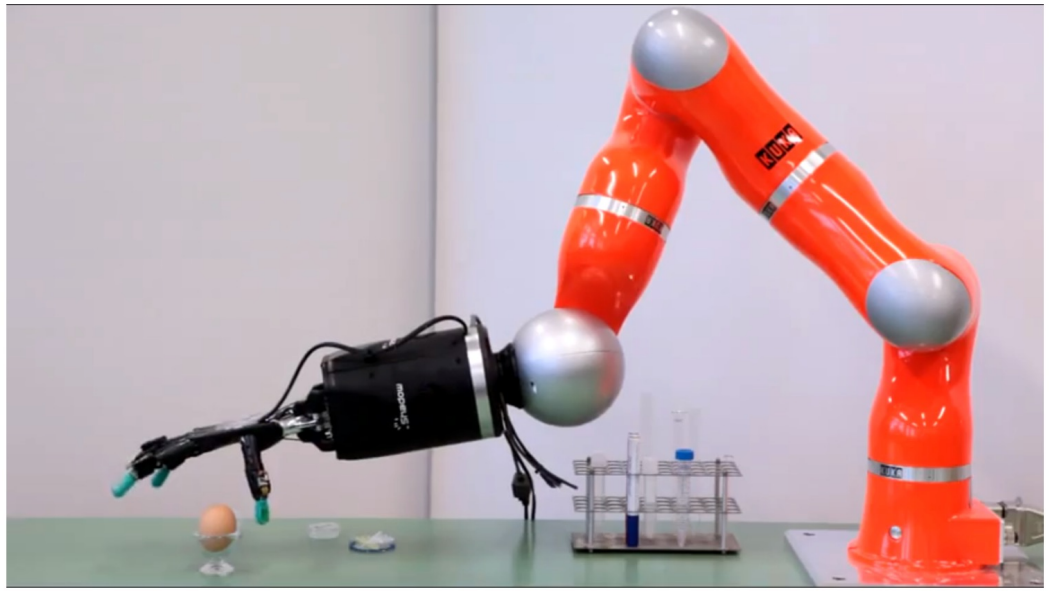


RViz Motion planning

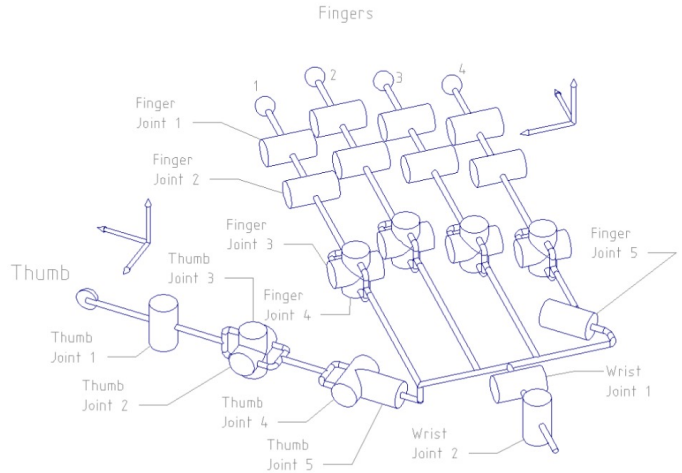




Experimental setup

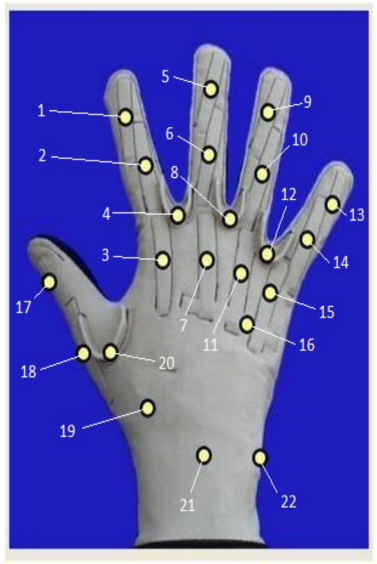


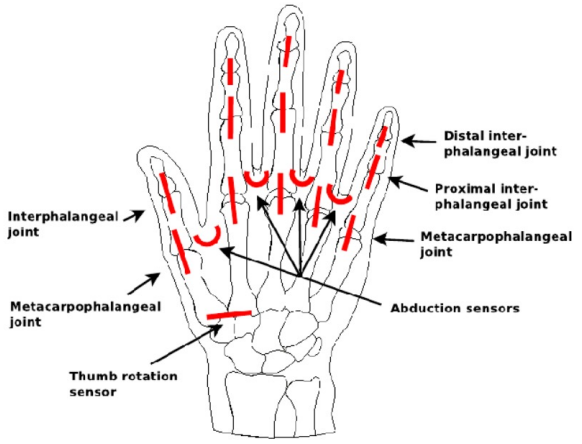
- Kuka Lightweight Robot (LWR 4)
  - 7 degrees of freedom (DOF)
  - weighs only 16 kg
- Shadow Dexterous Hand C6M2
  - 20 degrees of freedom (DOF)
  - controlled using tendons



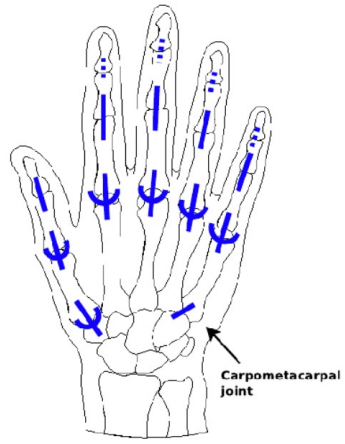
Telemanipulation

CyberGlove II





Cyber-glove sensors



Shadow-hand joints

Thank you

Question Time