ROS Industrial

邱强 （qqfly）

上海交通大学

2017.7 ROS Summer School

提纲

- 背景
- 工业机器人（理论篇）
  - 运动学
  - 碰撞检测
  - 运动规划
- 工业机器人（ROS篇）
  - ROS-I 简介
  - URDF
  - KDL, TRAC_IK, IKFast
  - MoveIt!

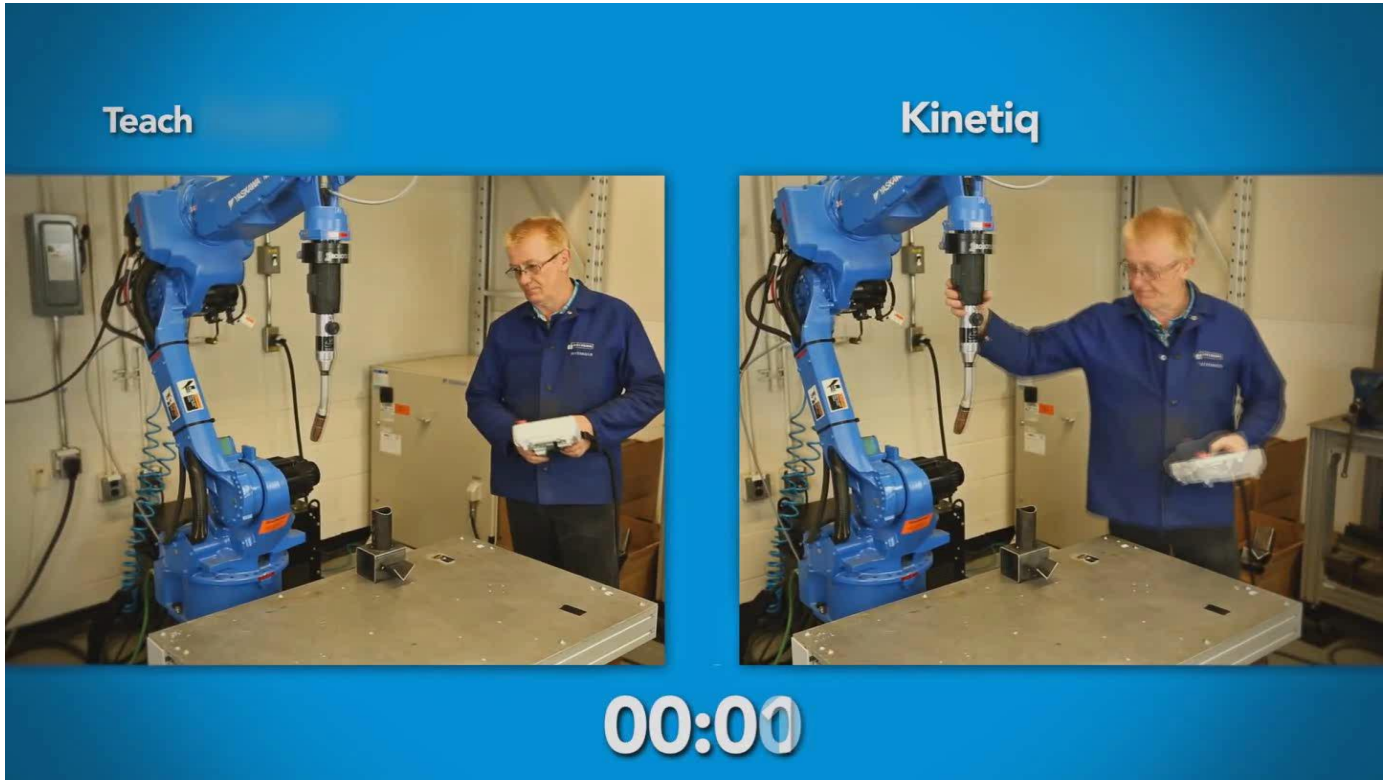
背 景

背景

- 工业机器人很不好用！
  - 编程语言，接口不统一

背景

- 工业机器人很不好用！
  - 智能化水平低 → 人工示教



背景

- 工业机器人
  - 主要用在汽车行业
  - 难以推广到 3C 等行业

- 研究机构
  - ROS让很多机器人变得好用

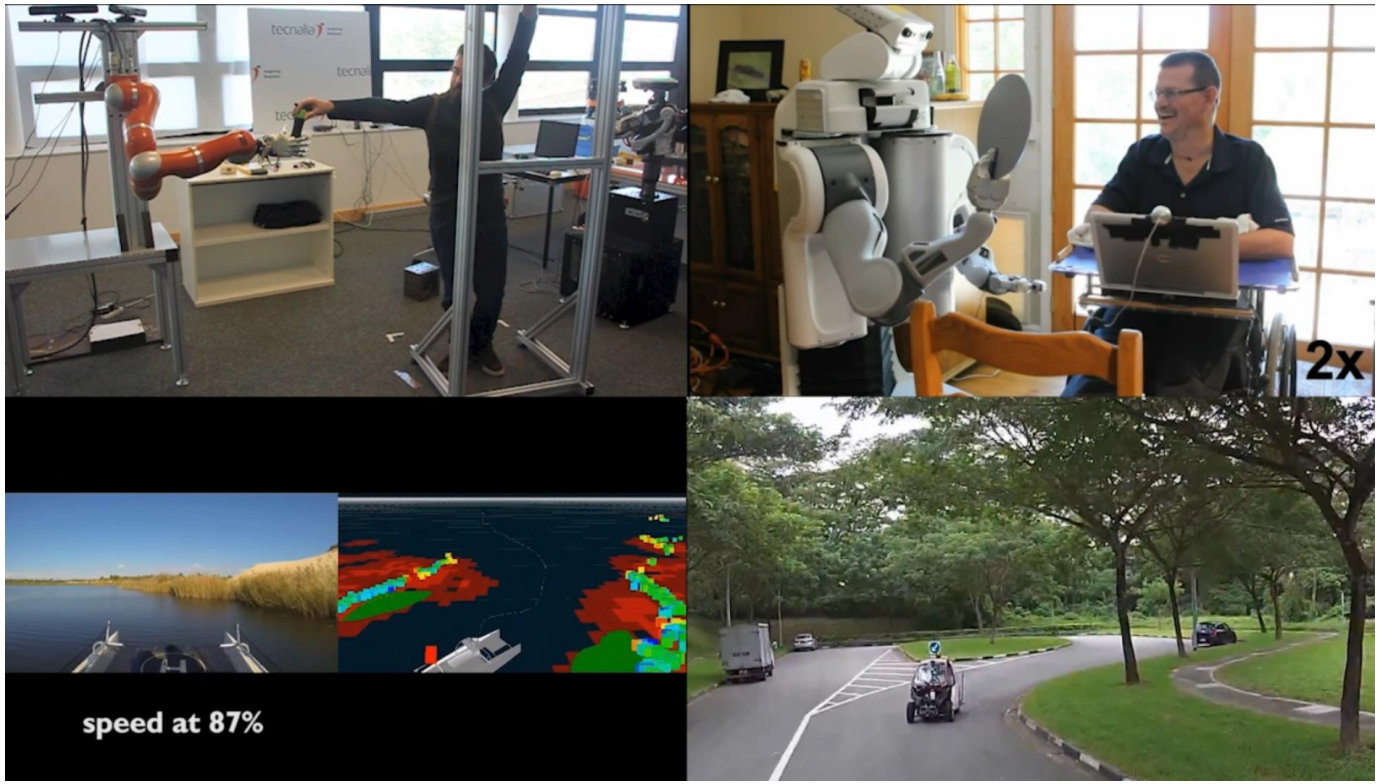
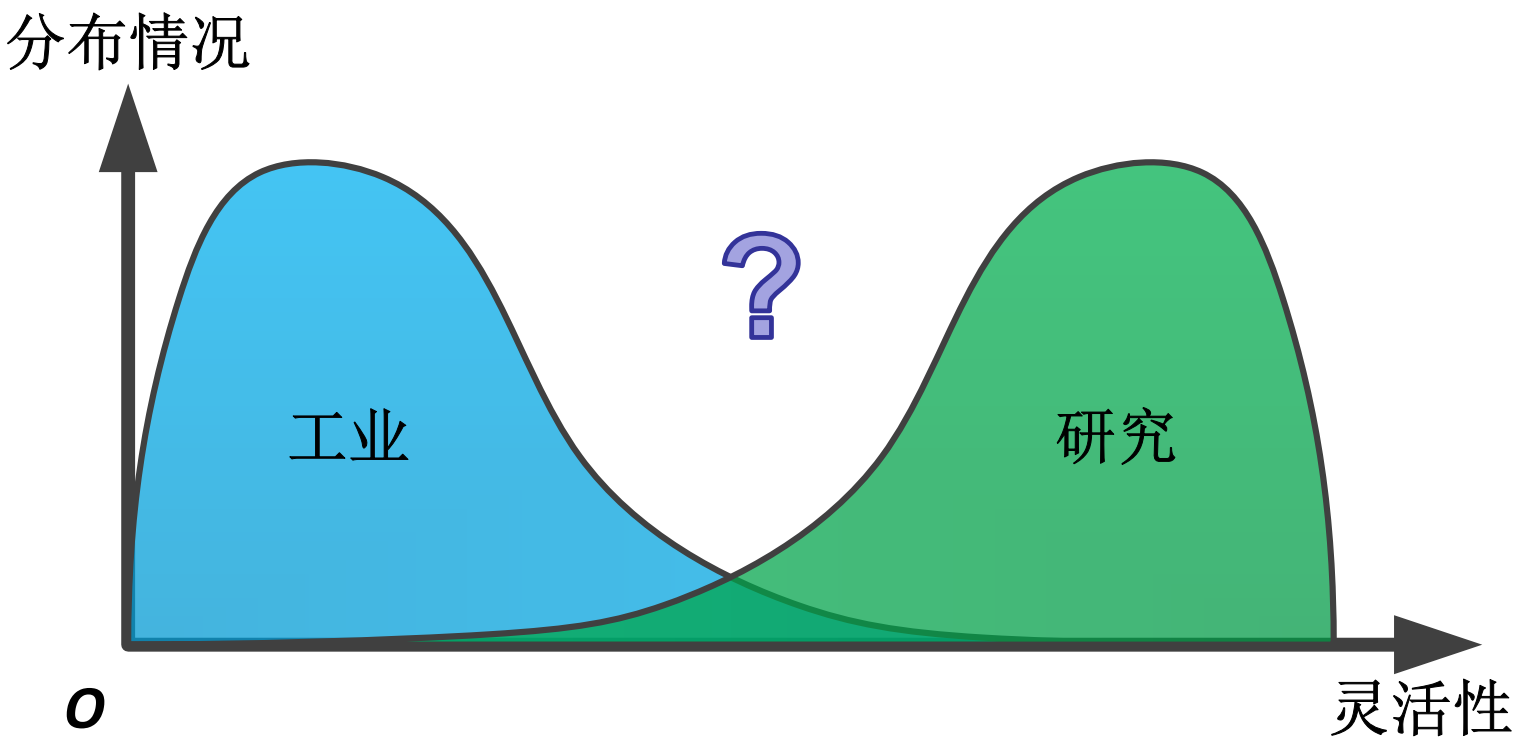
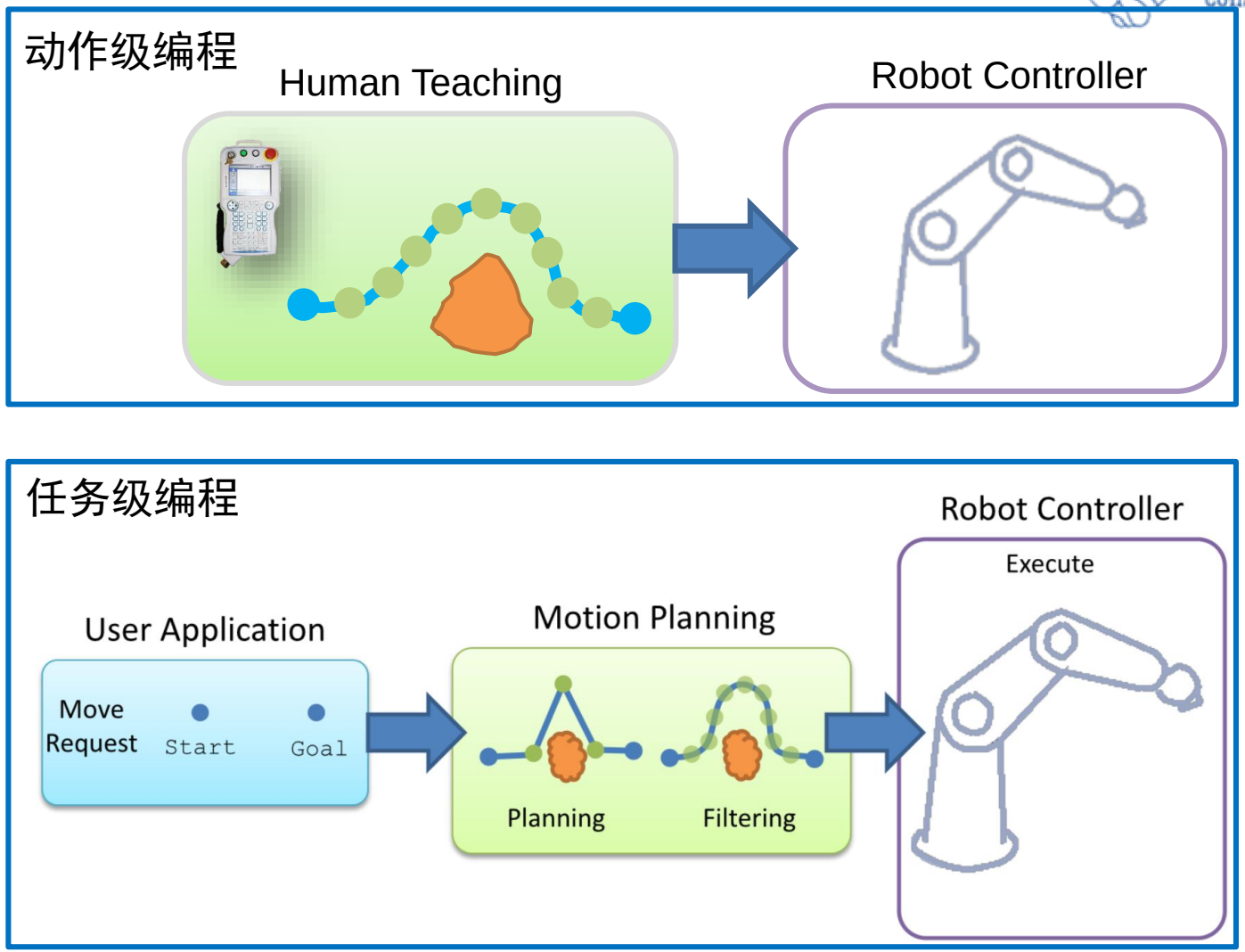


Image from: ROS 8 Year Montage

背景

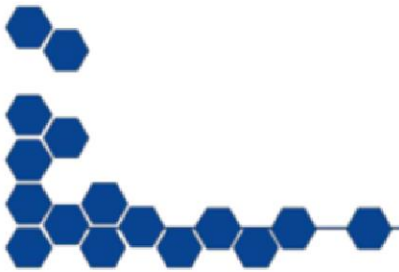


背景


动作级编程
Human Teaching
Robot Controller
任务级编程
Robot Controller
Execute
User Application
Motion Planning
Move
Request
Start
Goal
Planning
Filtering

工业机器人

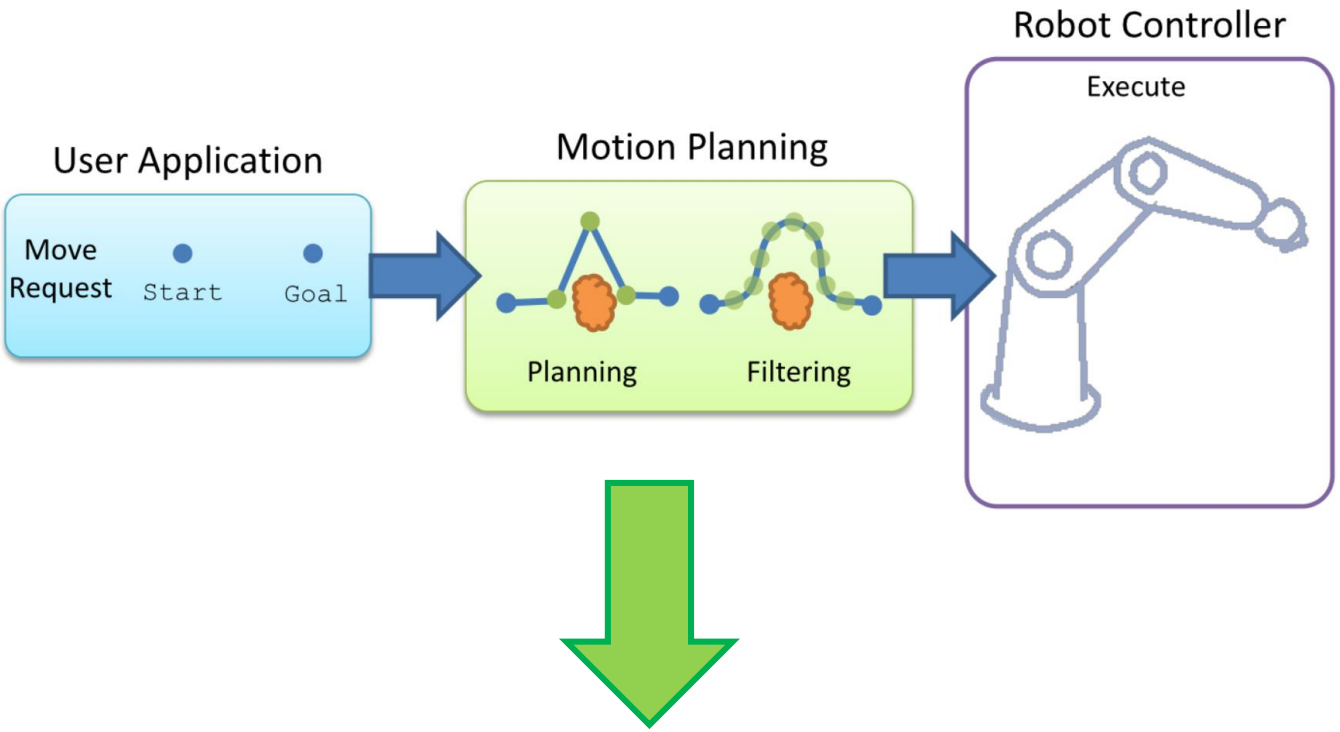
（理论篇）



ROS 不是万能的！

ROS 不是万能的！

ROS 不是万能的！

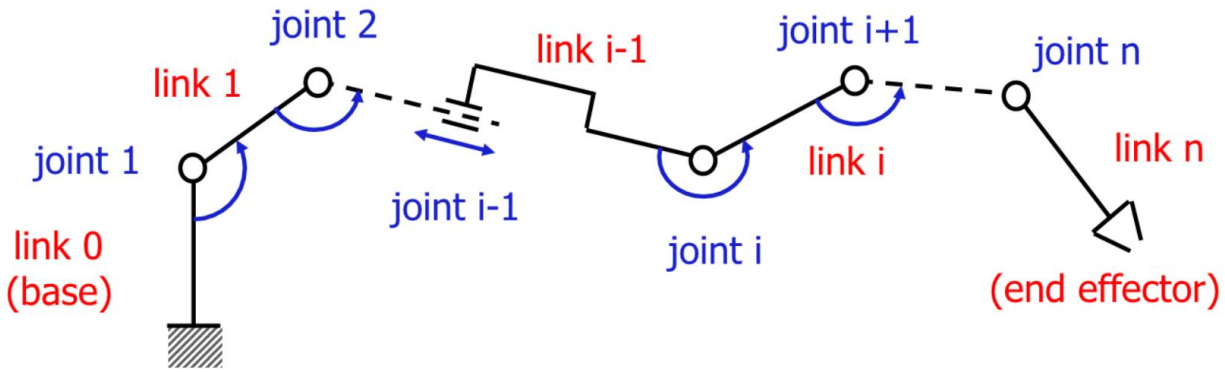


运动学
Kinematics

碰撞检测
Collision Checking

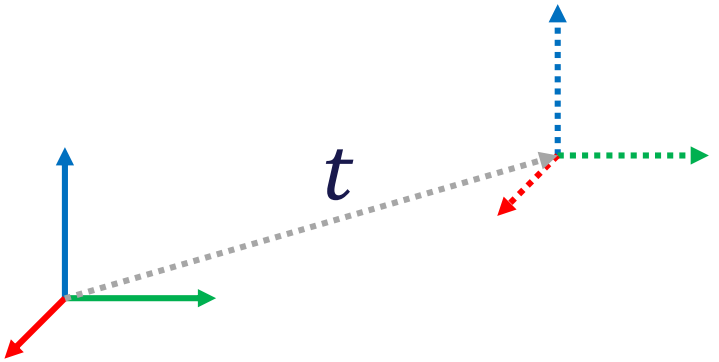
运动规划
Motion Planning

- 运动学 kinematics
  - 研究机器人几何方面的属性
  - 正运动学
    - 给定每个关节角度，计算末端位姿 $x = f(q)$
  - 逆运动学
    - 给定末端位姿，计算关节角度 $q = f^{-1}(x)$



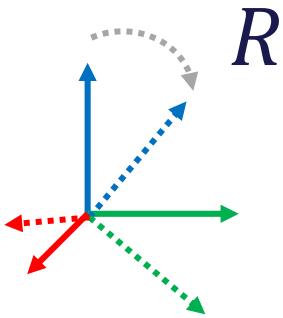
- 空间变换
  - 平移

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}' = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_0 + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$



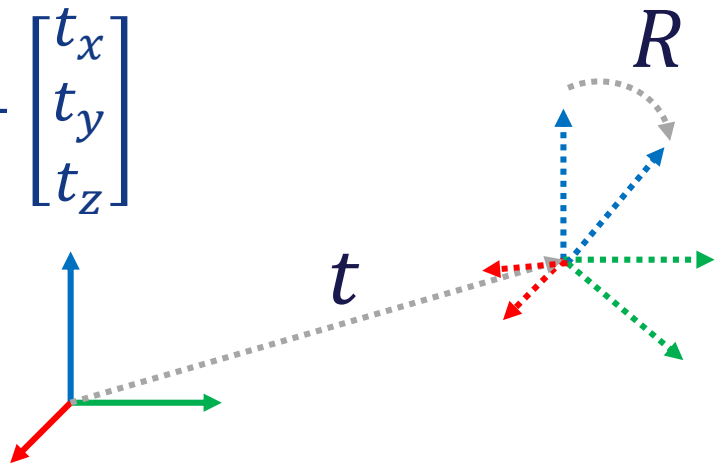
  - 旋转

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}' = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_0$$



- 空间变换
  - 平移+旋转

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}' = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_0 + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$



  - 齐次变换

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}' = \left[\begin{array}{ccc:c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_0 \Rightarrow x' = Tx_0$$

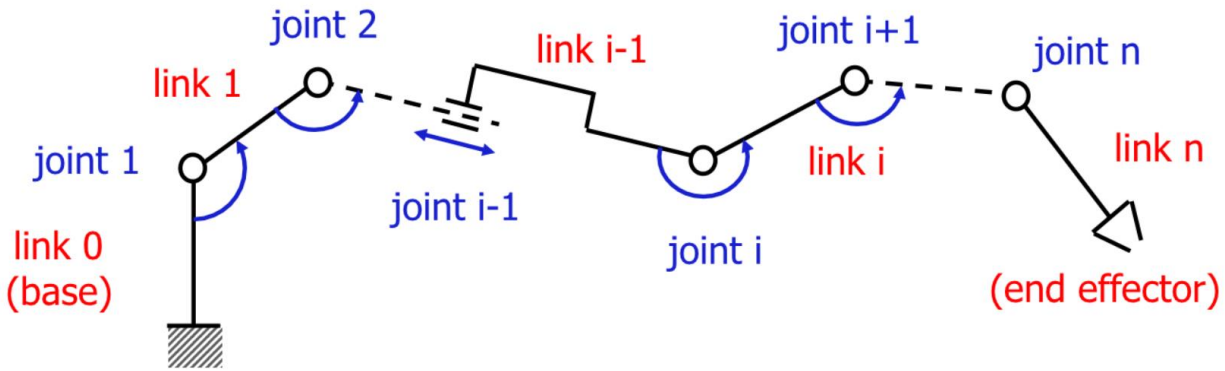
- 齐次变换

$$ {}^{0}_{n}T = {}^{0}_{n-1}T \cdot {}^{n-1}_{n}T $$

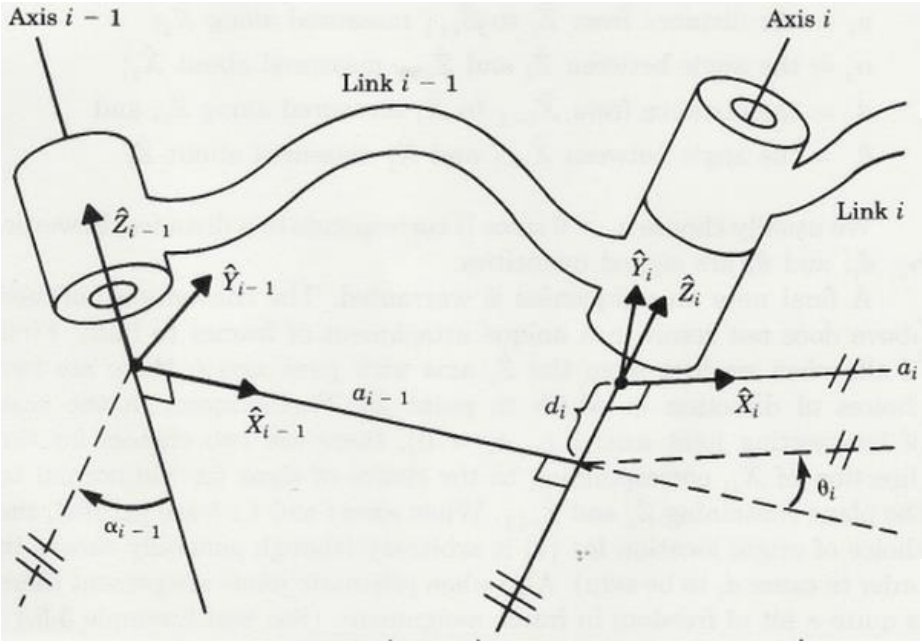
$$ {}^{0}_{n}T = {}^{0}_{n-2}T \cdot {}^{n-2}_{n-1}T \cdot {}^{n-1}_{n}T $$

$$ \dots $$

$$ {}^{0}_{n}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot \dots \cdot {}^{n-2}_{n-1}T \cdot {}^{n-1}_{n}T $$



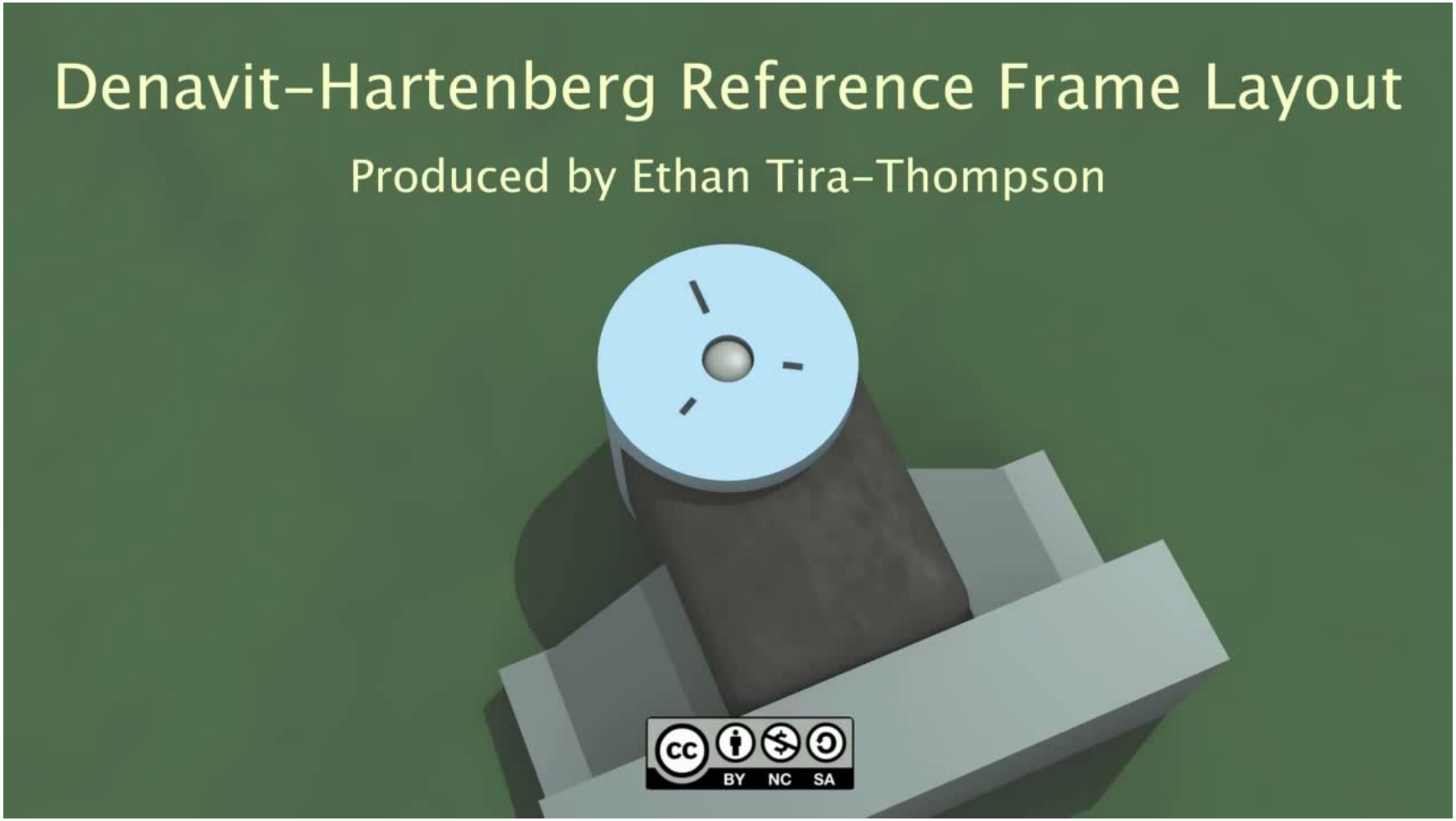
- DH 坐标



a_i = the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$;
α_i = the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$;
d_i = the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$; and
θ_i = the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$.

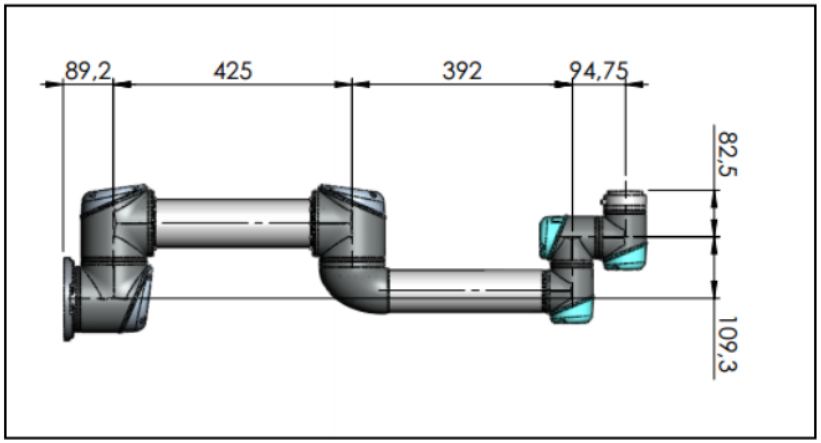
$$
{}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

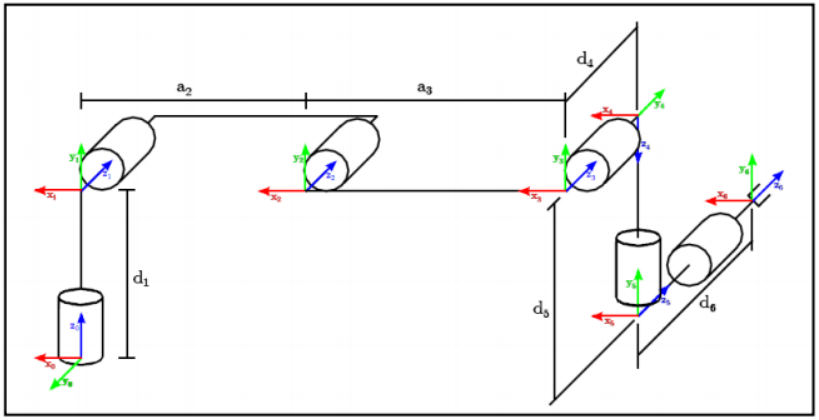
出处：Craig, John J. ***Introduction to robotics: mechanics and control***. Vol. 3. Upper Saddle River: Pearson Prentice Hall, 2005.



Video: Denavit-Hartenberg Reference Frame Layout

- 正运动学 Forward Kinematics





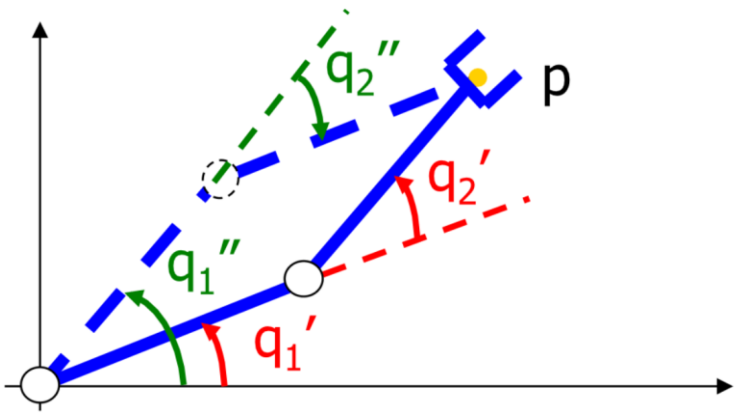
i	d	a	α	θ
0	-	0	0	-
1	d_1	0	$\pi/2$	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	d_4	0	$\pi/2$	θ_4
5	d_5	0	$-\pi/2$	θ_5
6	d_6	-	-	θ_6

$$ {}^{0}_{n}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot \ldots \cdot {}^{n-2}_{n-1}T \cdot {}^{n-1}_{n}T $$

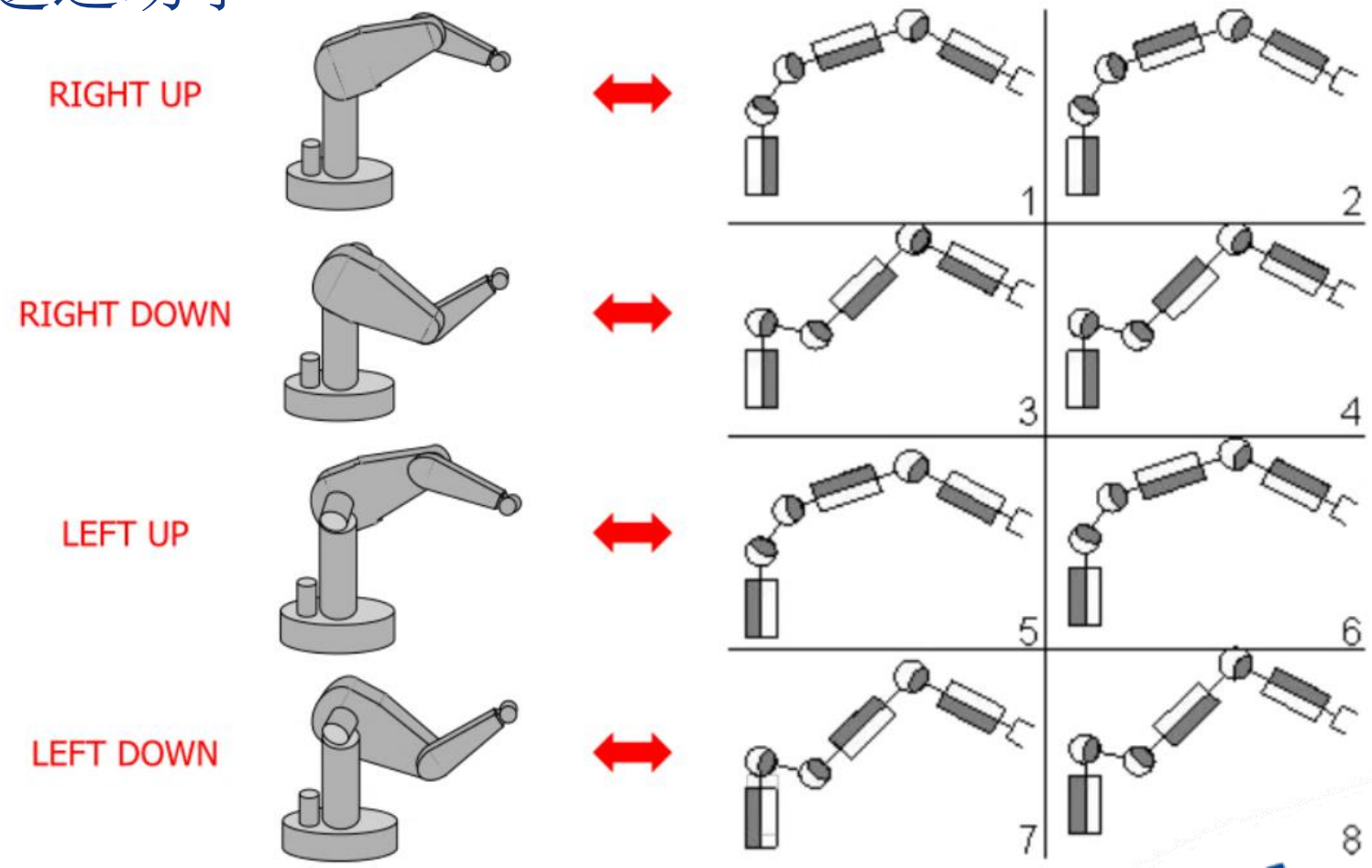
- 逆运动学 Inverse Kinematics
  - 已知末端位姿，求各关节角度

$$ {}^{0}_{n}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \Longrightarrow \theta_1, \theta_2, \theta_3, \cdots, \theta_n $$

  - 存在多解



- 逆运动学



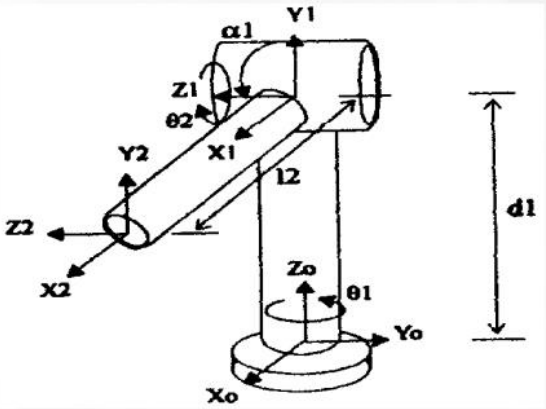
- 逆运动学
  - 解析解 Analytical Solution
    - 通过代数或几何方法进行求解。

$$T=\begin{bmatrix}\cos\theta_1\cos\theta_2 & -\cos\theta_1\sin\theta_2 & \sin\theta_1 & l_2\cos\theta_1\cos\theta_2\\ \sin\theta_1\cos\theta_2 & -\sin\theta_1\sin\theta_2 & -\cos\theta_1 & l_2\sin\theta_1\cos\theta_2\\ \sin\theta_2 & \cos\theta_2 & 0 & d_1+l_2\sin\theta_2\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}n_x & o_x & a_x & p_x\\ n_y & o_y & a_y & p_y\\ n_z & o_z & a_z & p_z\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- Select p_x e p_y

$$\frac{p_y}{p_x}=\frac{l_2\cdot\text{sen}(\theta_1)\cdot\cos(\theta_2)}{l_2\cdot\cos(\theta_1)\cdot\cos(\theta_2)}\Rightarrow\theta_1=\tan^{-1}\left(\frac{p_y}{p_x}\right)$$

- Pre-multiply by $(A_{0,1}{}^{0})^{-1}$



$$\frac{p_x\cdot\cos(\theta_1)+p_y\cdot\sin(\theta_1)}{p_z-d_1}=\frac{l_2\cdot\cos(\theta_2)}{l_2\cdot\sin(\theta_2)}\Rightarrow\theta_2=\tan^{-1}\left(\frac{p_z-d_1}{p_x\cdot\cos(\theta_1)+p_y\cdot\sin(\theta_1)}\right)$$

- 逆运动学
  - 数值解 Numerical Solution
    - 通过优化迭代的方法进行求解

$$q = f^{-1}(x)$$

$$\min_{q} e \,, where\ e = \left(x - f(q)\right)^2$$

- 逆运动学
  - 数值解

$$J_r = \frac{\partial f_r(q)}{\partial q} \qquad \dot{x} = J_r \dot{q}$$

- Newton method (here for m=n)
  - $r_d = f_r(q) = f_r(q^k) + J_r(q^k)\,(q - q^k) + o(\|q - q^k\|^2)$ ← neglected

$$q^{k+1} = q^k + J_r^{-1}(q^k)\,[r_d - f_r(q^k)]$$

- Gradient method (max descent)
  - minimize the error function

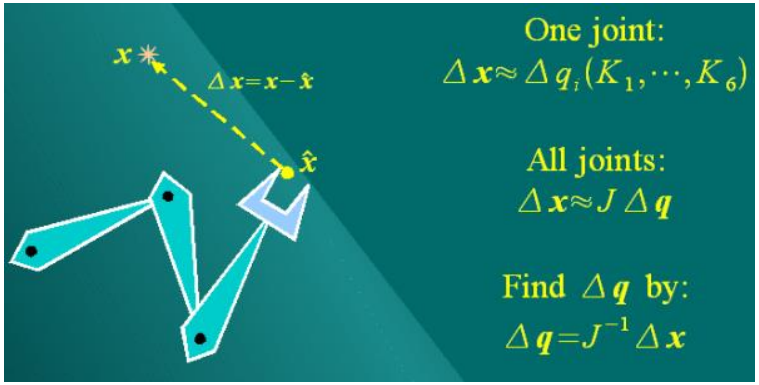
$$H(q) = \tfrac{1}{2}\,\|r_d - f_r(q)\|^2 = \tfrac{1}{2}\,[r_d - f_r(q)]^T\,[r_d - f_r(q)]$$

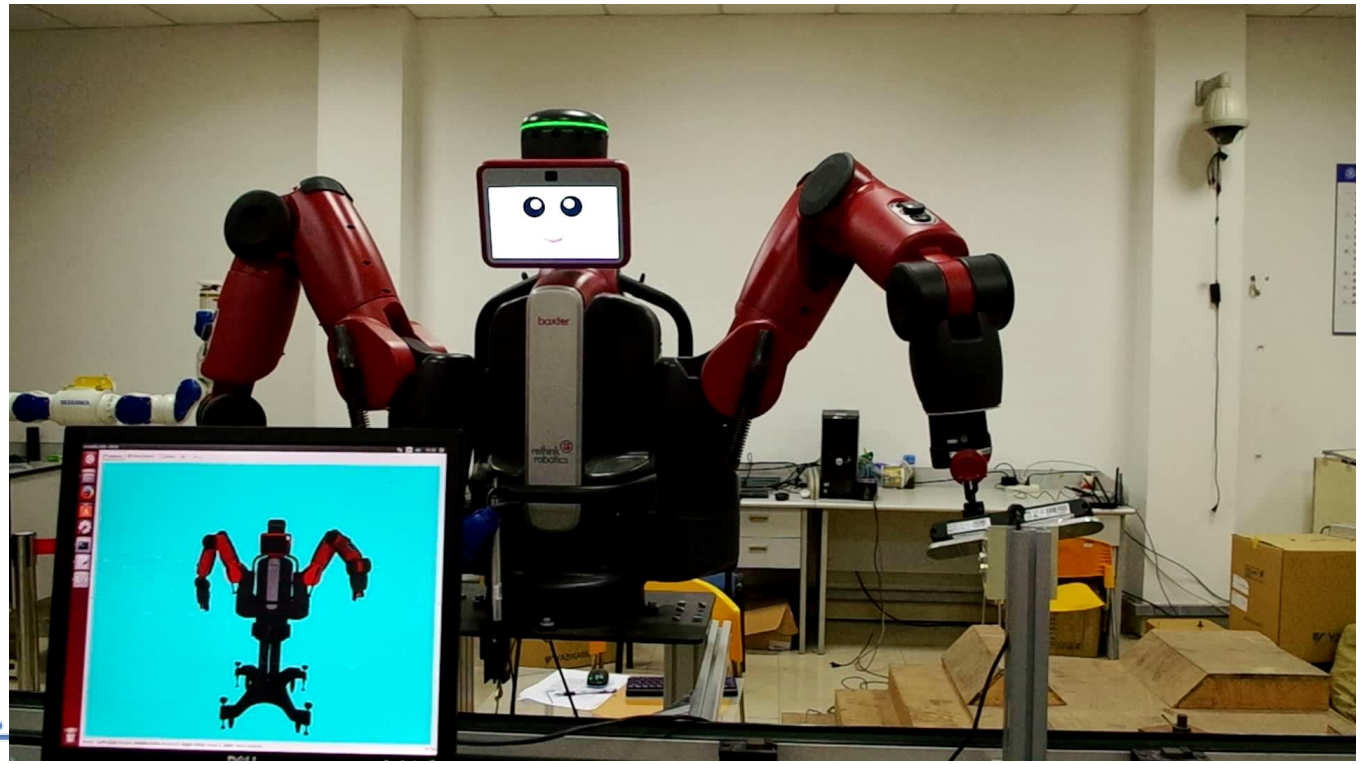
$$q^{k+1} = q^k - \alpha\,\nabla_q H(q^k)$$

from $\nabla H(q) = -\,J_r^T(q)\,[r_d - f_r(q)]$, we get

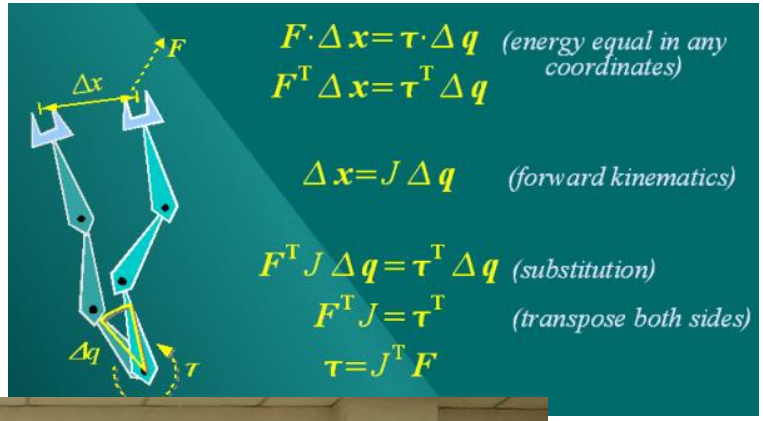
$$q^{k+1} = q^k + \alpha\,J_r^T(q^k)\,[r_d - f_r(q^k)]$$

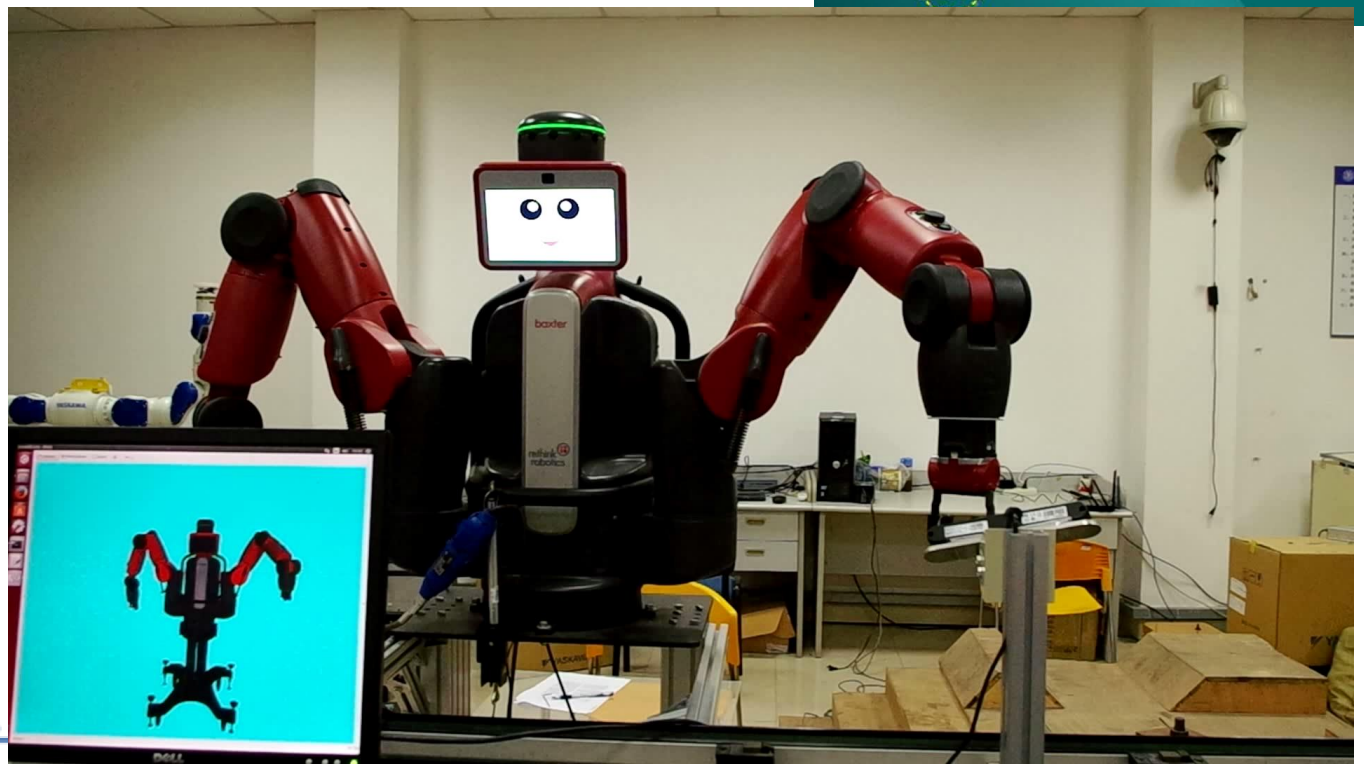
- 逆运动学
  - 逆雅克比法
    - 朝着目标点运动





- 逆运动学
  - 雅克比转置法
    - 朝着目标点拖动



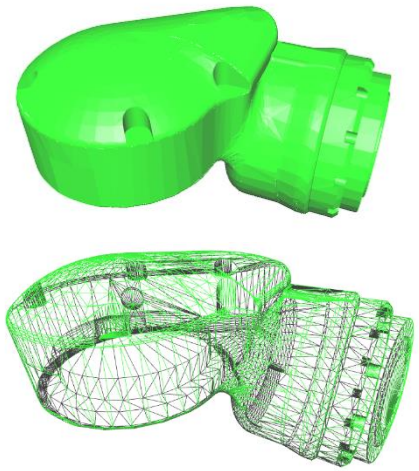


- 解析解
  - 可以获得所有解
  - 求解速度快
  - 完美，如果能找到
  - 不具有通用性
  - 某些构型不易求解
  - 无法求解冗余构型
- 数值解
  - 适用于所有串联机器人
  - 可以采用通用求解框架
  - 可以求解冗余构型
  - 每次只能获得一组解
  - 求解速度较慢，~1000倍

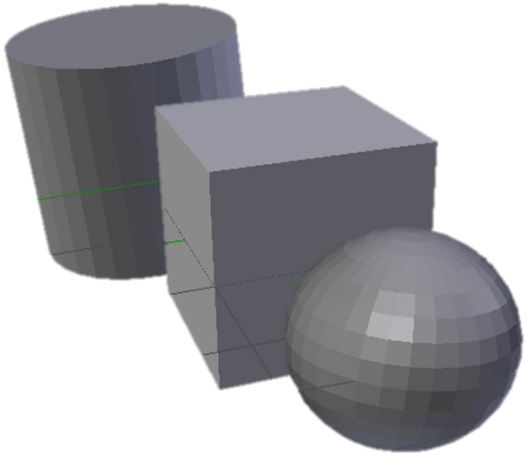
- 碰撞检测 Collision Checking
  - 检测机器人与环境是否发生碰撞
  - 计算机器人与障碍物之间的距离

  - 步骤：
    - 几何描述
    - 位置更新
    - 距离计算
    - 其他

- 几何描述



CAD mesh

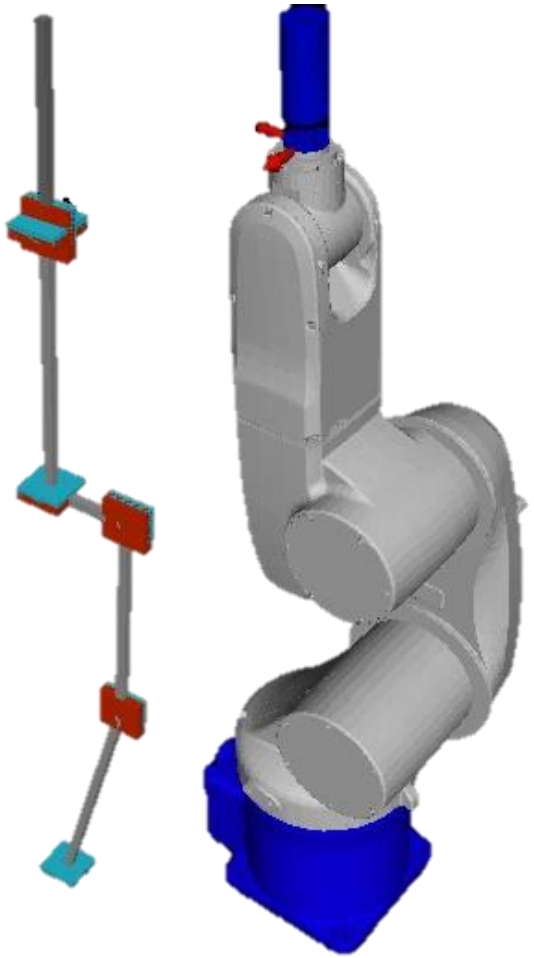


Simple shape

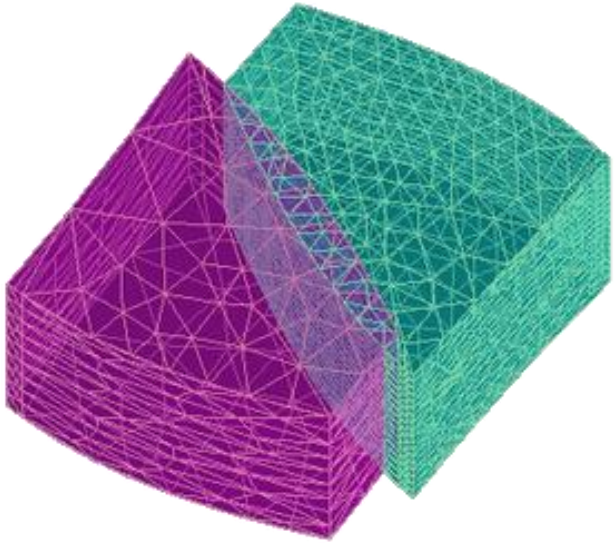
Point cloud

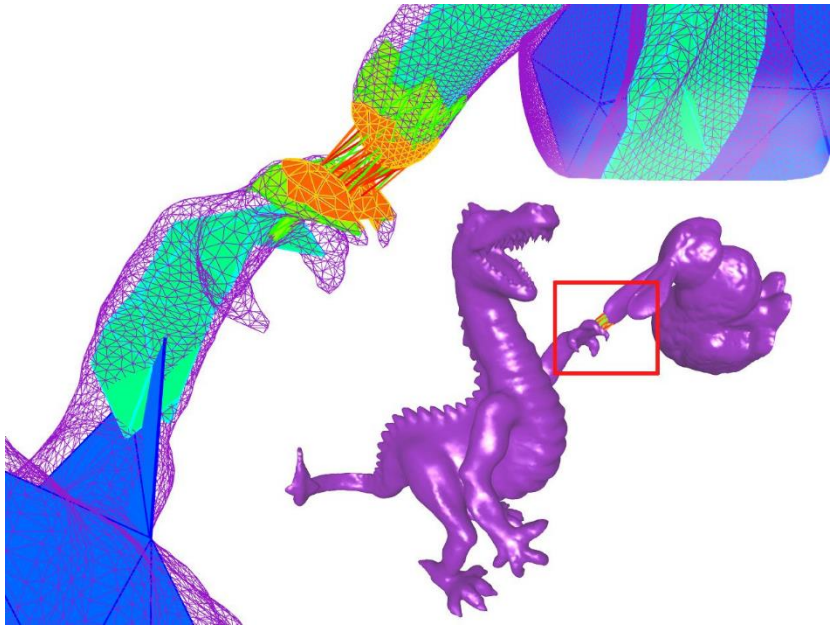
- 位置更新
  - 获得关节角度
  - 为每个模块计算运动学正解
  - 通过对应的空间变换（运动学正解）更新模型文件空间位置

$$ {}^{0}_{i}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot \dots \cdot {}^{i-2}_{i-1}T \cdot {}^{i-1}_{i}T $$



- 距离计算
  - Gilbert–Johnson–Keerthi distance algorithm
  - axis-aligned bounding box (AABB)

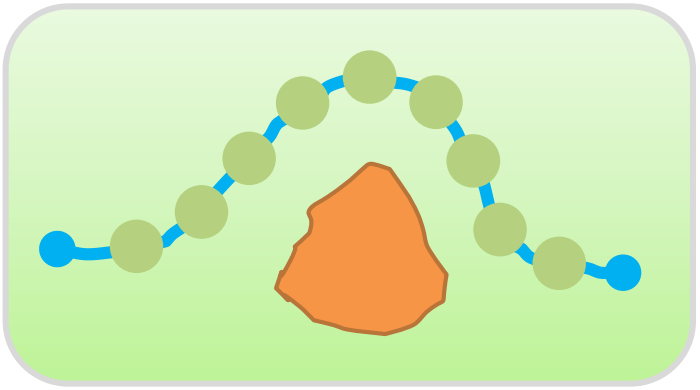




- 其他
  - 自身碰撞

	l_1	l_2	l_3	l_4	l_5	l_6
l_1		0	1	1	1	1
l_2	0		0	1	1	1
l_3	1	0		0	1	1
l_4	1	1	0		0	1
l_5	1	1	1	0		0
l_6	1	1	1	1	0	

- 运动规划 Motion Planning
  - 在给定环境中，指定机器人起点与终点，计算出连接起点与终点，并满足一定约束条件（如避障）的轨迹。



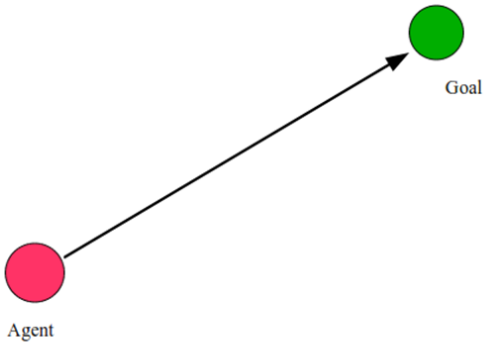
模型
- 运动学
- 3D模型
- 环境模型
- …

目标
- 到达终点
- 避开障碍物
- 减小路径长度
- …

路径
- 一序列状态
- 控制策略
- …

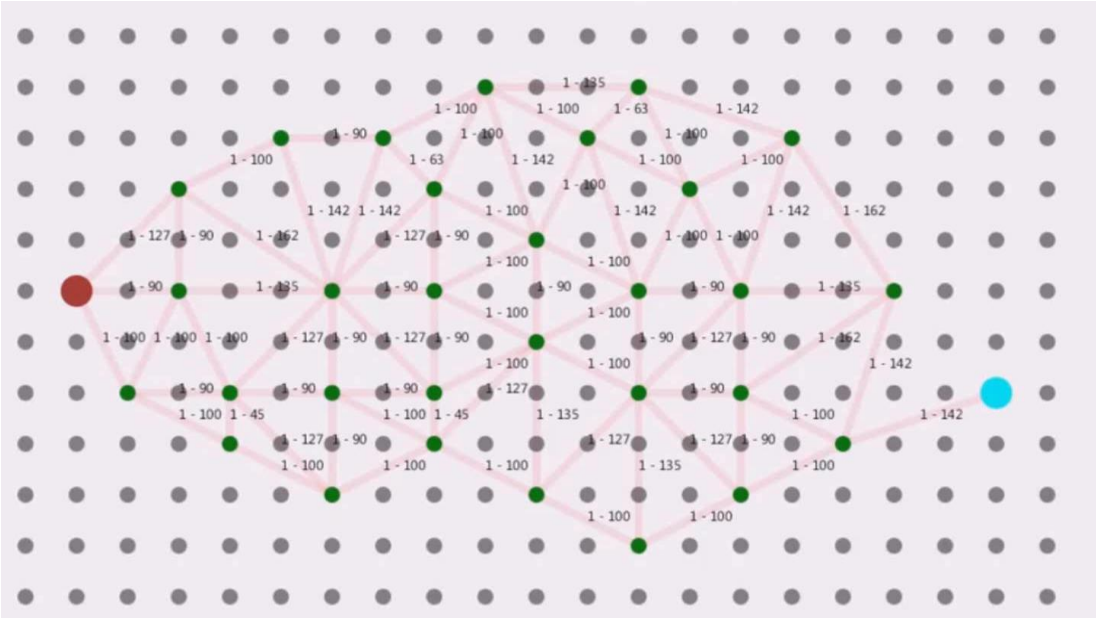
- 规划器评价标准
  - Optimality（最优性）: 路径最短、规划速度最快等。
  - Complete（完备性）:在有限时间内解决所有有解问题。
- 先从2D，点状机器人（point agent）的问题入手。

- Walk To
  - 直接朝着目标走，直到到达目标点为止。
  - 很多 RPG 游戏就采用了这种简单的算法
  - 最优性，但不完备



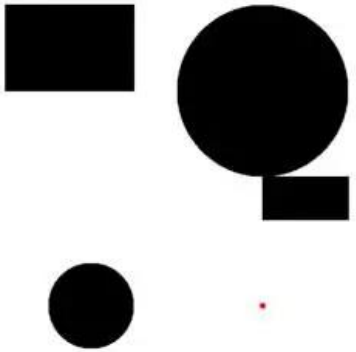
- 优化算法（蚁群等）
  - 类似最优控制
  - 大部分情况下效果不错，但复杂问题很容易陷入局部极值
  - 不完备也不最优

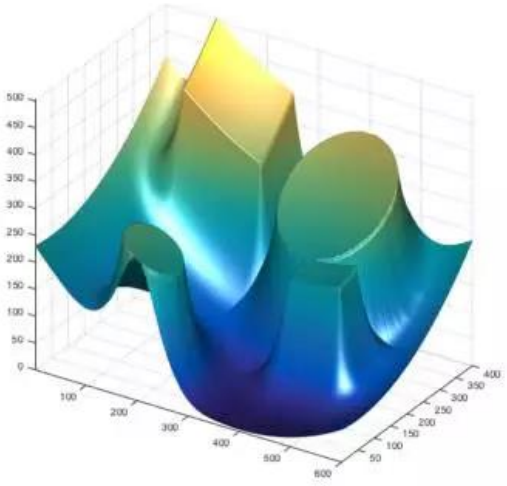
$$
\begin{aligned}
\min_{u,x} \quad & (x_T - x_G)^\top (x_T - x_G) \\
\text{s.t.} \quad & x_{t+1} = f(x_t, u_t) \quad \forall t \\
& u_t \in \mathcal{U}_t \\
& x_t \in \mathcal{X}_t \\
& x_0 = x_S
\end{aligned}
$$

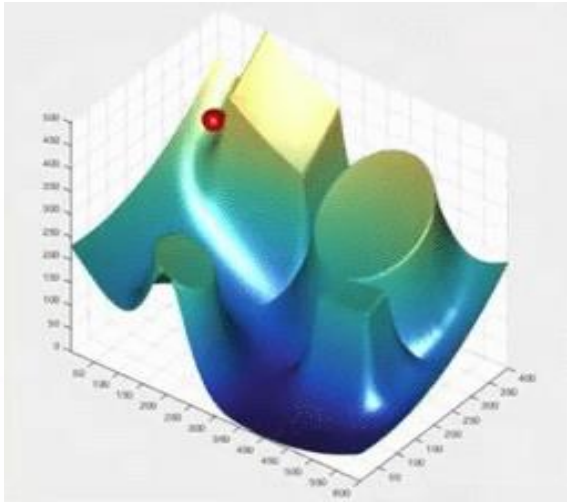


- 人工势场
  - 在障碍物周围建立排斥势场
  - 从起点到终点构建吸引势场
  - 采用梯度下降等方式求解

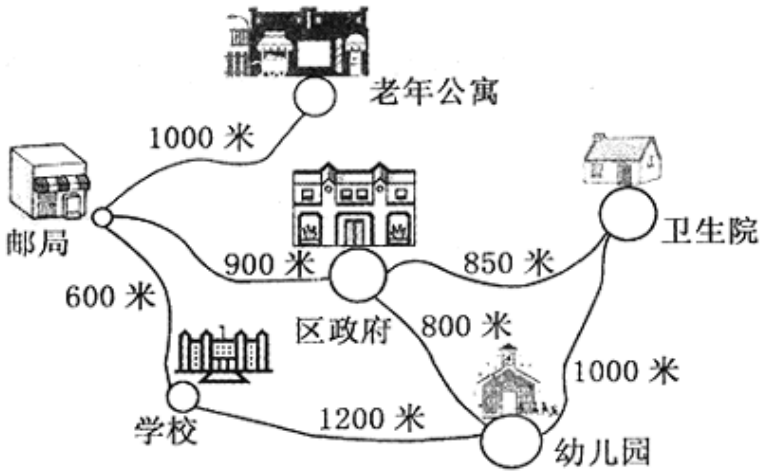
- 容易实现、效果很好
- 可以与控制结合
- 可能陷入局部极值
- 不完备且不最优

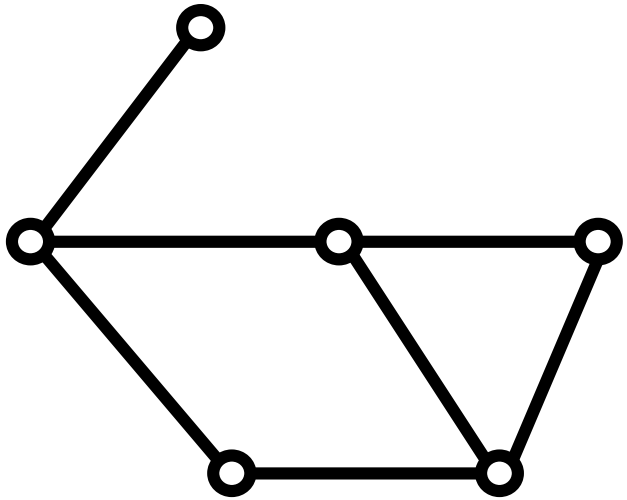






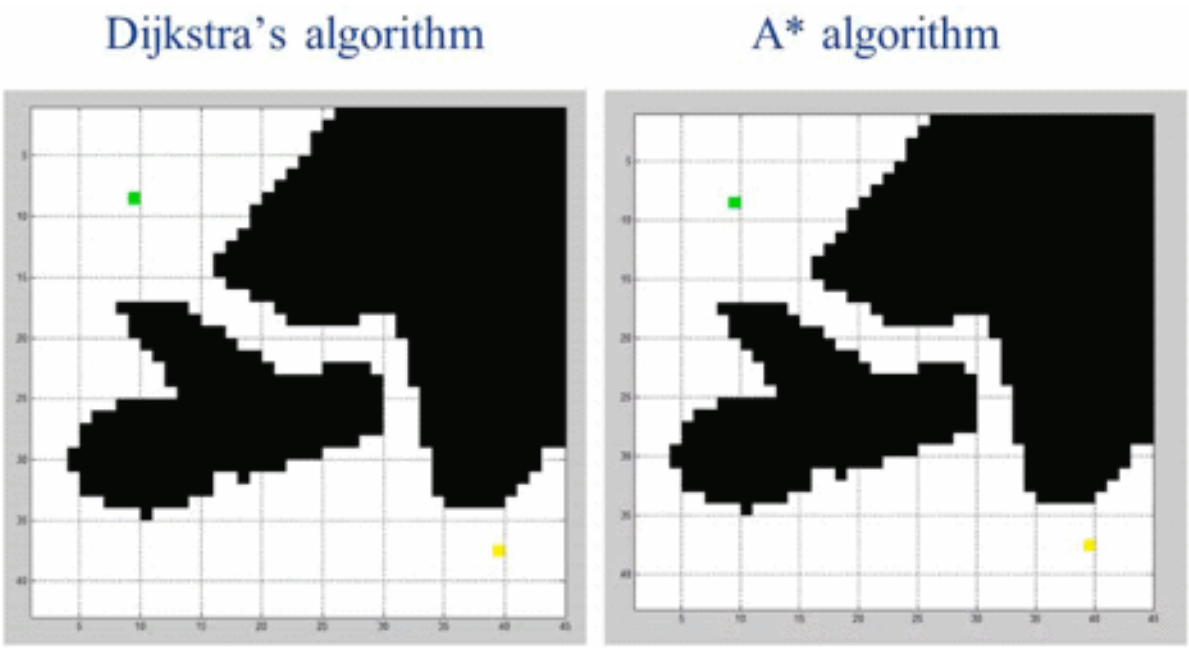
- 图搜索算法
  - 将问题描述成图（节点+边）
  - 用图搜索算法解决问题





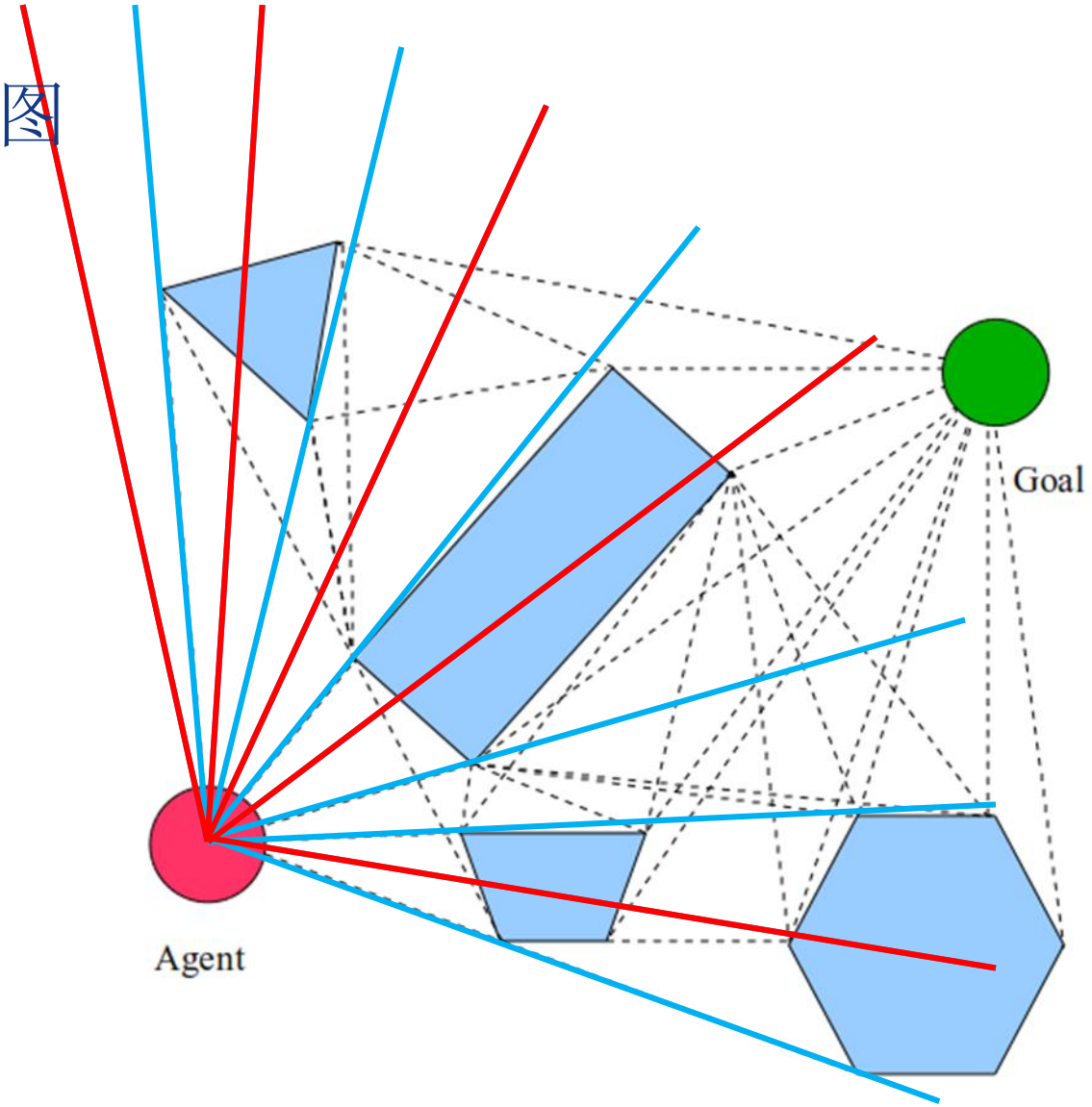
- 图搜索算法
  - 将问题描述成图（节点+边）
  - 用图搜索算法解决问题
  - Dijkstra、A*
  - 在给定的图中完备且最优

- **如何建立图？**

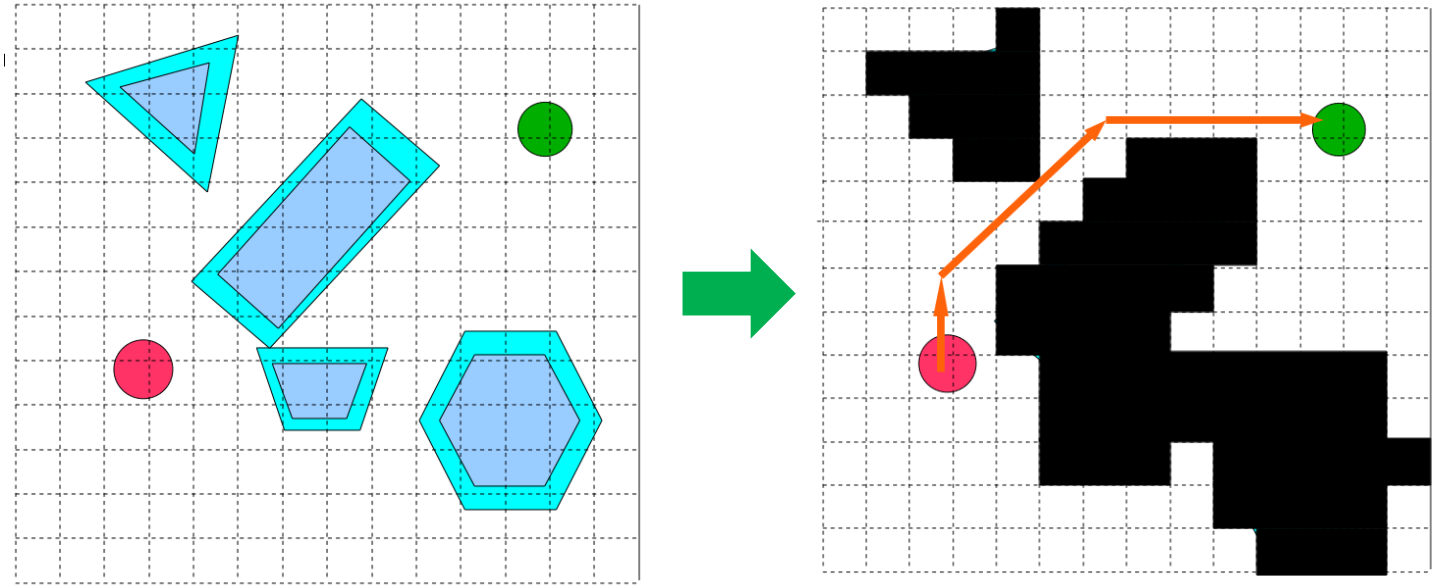


- 可视图（Visibility Graph）
  - 用封闭多面体描述障碍物
  - 利用障碍物顶点间的连线构建一个图（graph），之后用图搜索算法求解
  - 站在某个顶点上，环绕四周，把你能看到（无障碍物）的顶点连接起来
  - 完备且最优

- 可视图

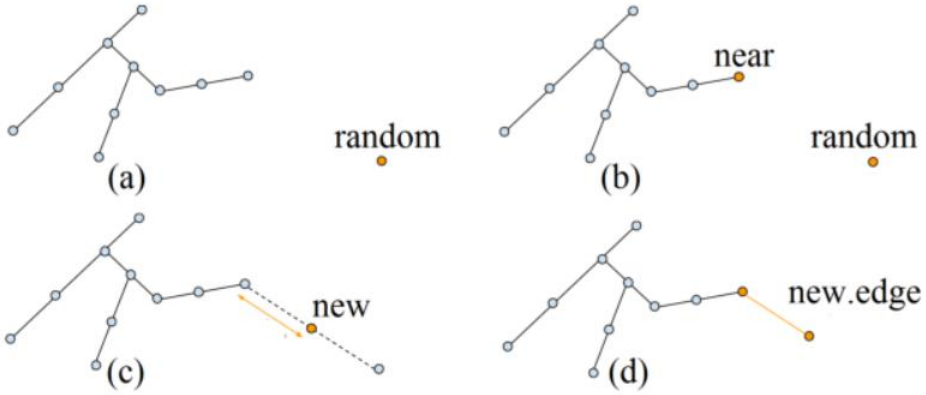


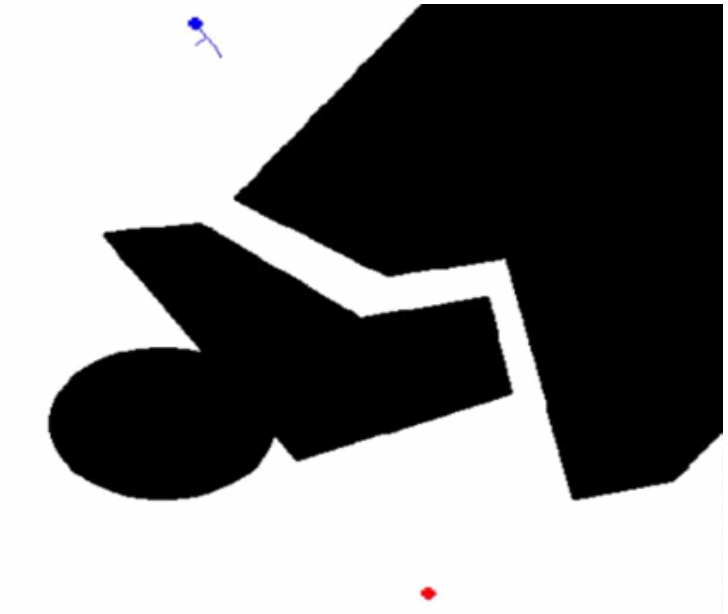
- 栅格化（Cell Decomposition）
  - 按一定分辨率将地图进行网格划分
  - 用四连通或八连通规则建立网格图
  - 分辨率完备（Resolution Complete）且最优



- 随机路图法
  - PRM（Probabilistic Road Maps）
  - 通过随机采样选取不碰撞的点
  - 两点连接采用简单的局部规划器
    - 如 Walk to 算法
  - 将起止点连入路图
  - 用图搜索求解
  - 概率完备且不最优

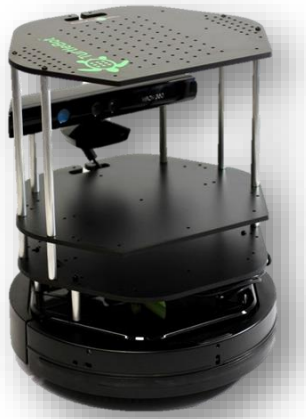
- 快速扩展随机树法
  - RRT（Randomly Exploring Randomized Trees）
  - 基于树状结构的搜索算法
  - 概率完备且不最优



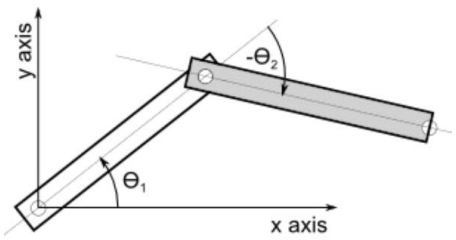


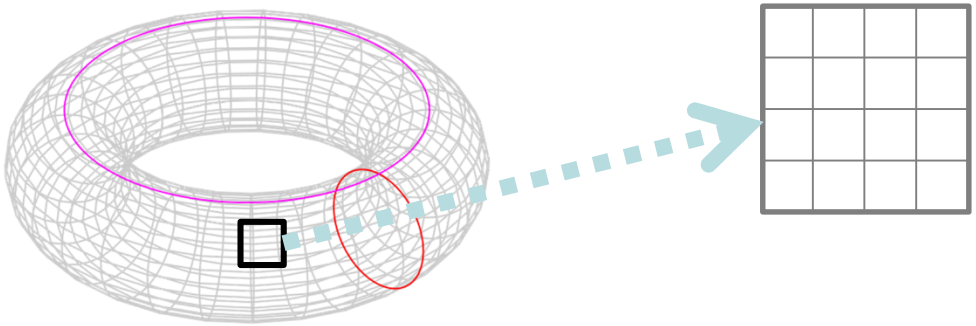
- 拓展到实际机器人
  - 机器人不再是一个点，需要将机器人的体积考虑在内
  - 机器人的自由度更高，原本的算法是否都还可用？

- C 空间（理论基础）
  - 构形空间，Configuration Space
  - 用向量描述机器人的构形
  - 在 C 空间内，机器人是一个点
    - 平面机器人$(x, y, \theta)^T$
    - 七轴机器人$(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)^T$

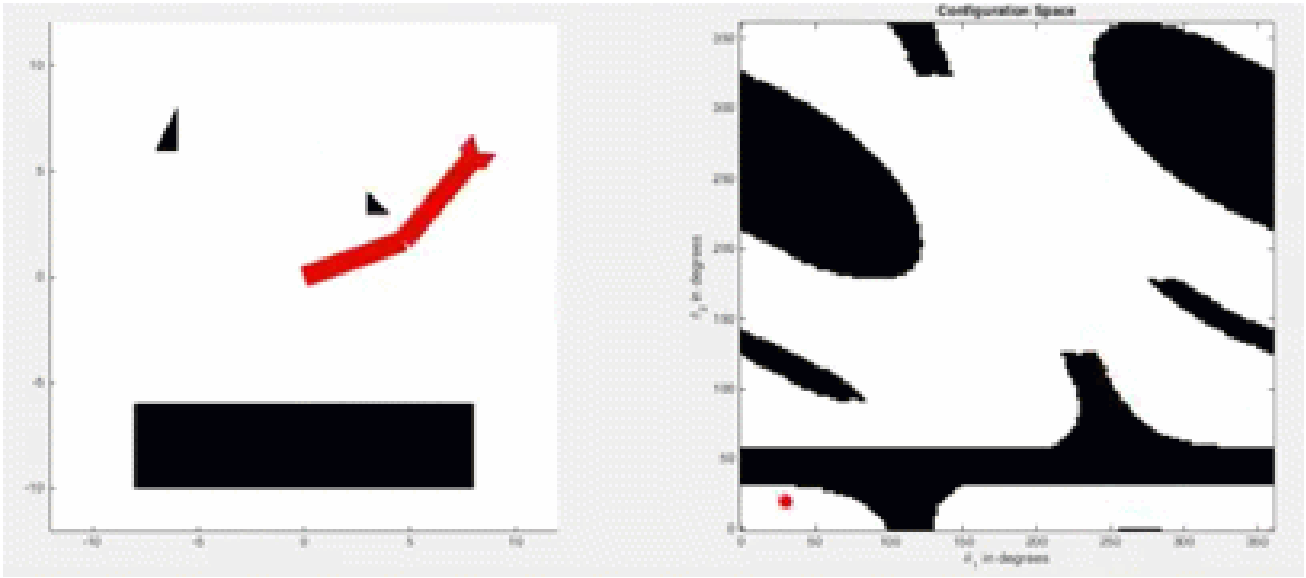


- C 空间（理论基础）
  - C 空间拓扑性质与笛卡尔坐标系下的情况不同
    - 二自由度机械臂的C空间是一个圆环面
  - 大部分机构（连续旋转关节、平动关节等）形成的构形空间均是**微分流形**，任一点的邻域均与欧式空间**同态**



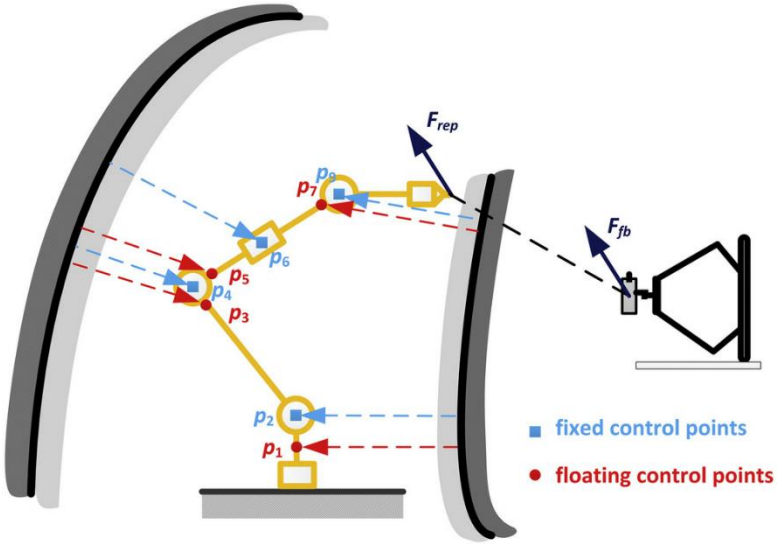


- C 空间（理论基础）
  - 微分流形：大部分算法效果与在笛卡尔坐标下效果相同



- 高维度
  - 蚁群等优化算法：收敛慢，更多局部极值点
  - 可视图法：在高维空间中，算法不成立
  - 栅格法：
    - 理论上可行
    - 但会计算量太大
    - 对于一个六自由度机械臂，我们按照6°分辨率（已经是很低的分辨率了）划分网格，那么将会产生$60^6 = 4.67 \times 10^{10}$个网格，单是对每个网格进行碰撞检测（如果碰撞检测速度为0.1ms），就需要1296小时。
    - 一般在高于三维的问题上不使用该方法。

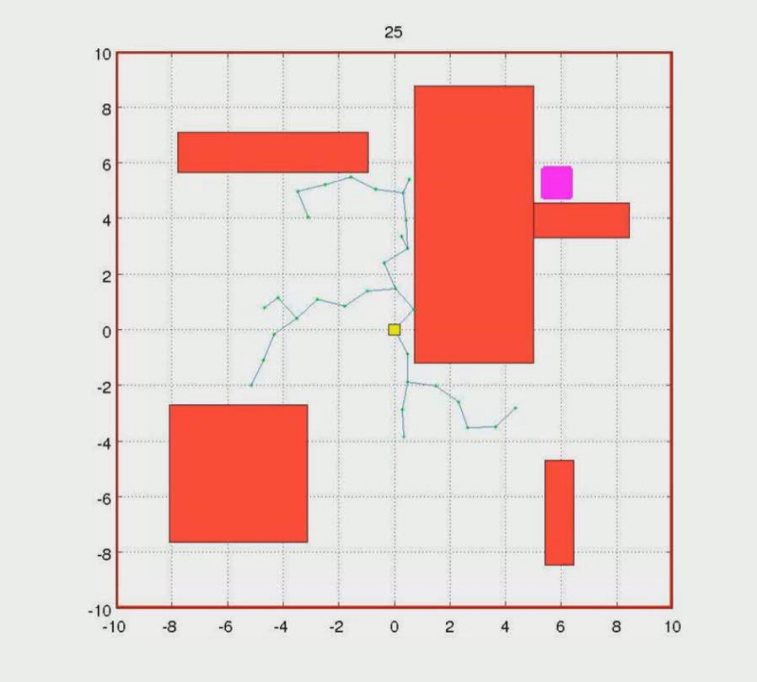
- 人工势场
  - 在 C 空间内建立势场不方便
  - 只对个别控制点进行计算，折算到每个关节上
  - 不完备且不最优，但对于简单的问题很实用

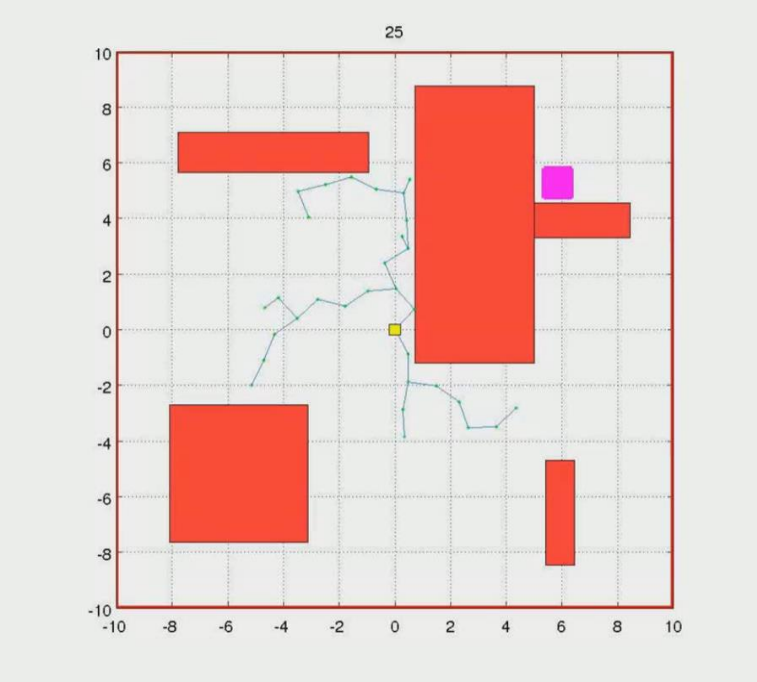


- PRM 和 RRT
  - 不需要知道 C 空间的具体情况，只对**随机采样点**进行碰撞检测（判断是否在 C 空间的可行区域内）
  - 两点之间采用简单的局部规划器（如 Walk to）进行连接
  - PRM：获得一个**图**，采用图搜索算法求解
  - RRT：获得一个连接到终点的**树**，反向搜索即可
  - 在高维空间内可行，概率完备且不最优
- 现状：
  - 主要使用 RRT 和 PRM 等 Sampling-based methods
  - 这些算法计算的结果一般需要进行后处理（smoothing 等）

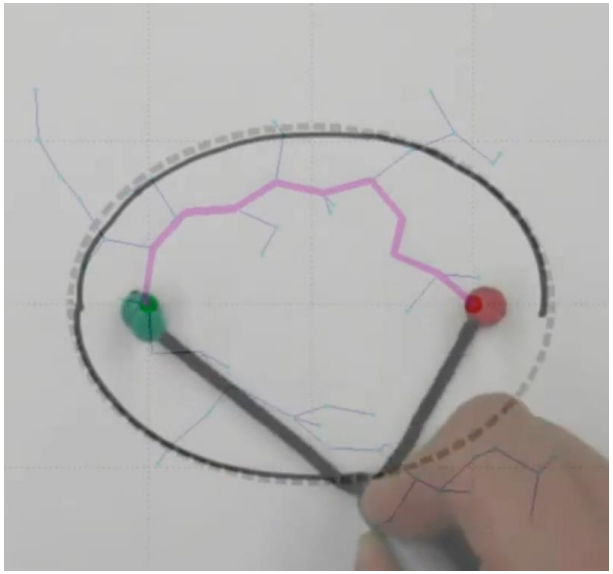
- RRT 和 PRM 变种
  - C 空间
  - 随机采样（各种采样算法 T-RRT）
  - 有效性判断（如碰撞检测算法 AABB、减少碰撞检测 Lazy-RRT）
  - 局部规划器连接（各种连接方法、重新连接 RRT*，PRM*）
  - …

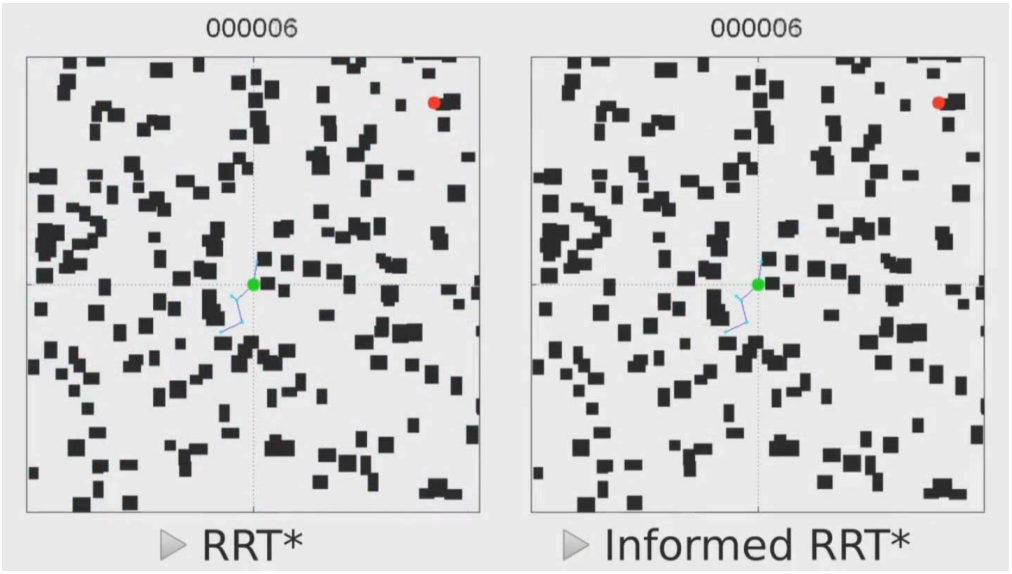
- RRT*
  - 渐进最优


25
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10 -8 -6 -4 -2 0 2 4 6 8 10


25
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10 -8 -6 -4 -2 0 2 4 6 8 10

- Informed RRT*
  - 先验知识→只在sub-problem下采样





运动规划

Video: 机器人运动规划集锦

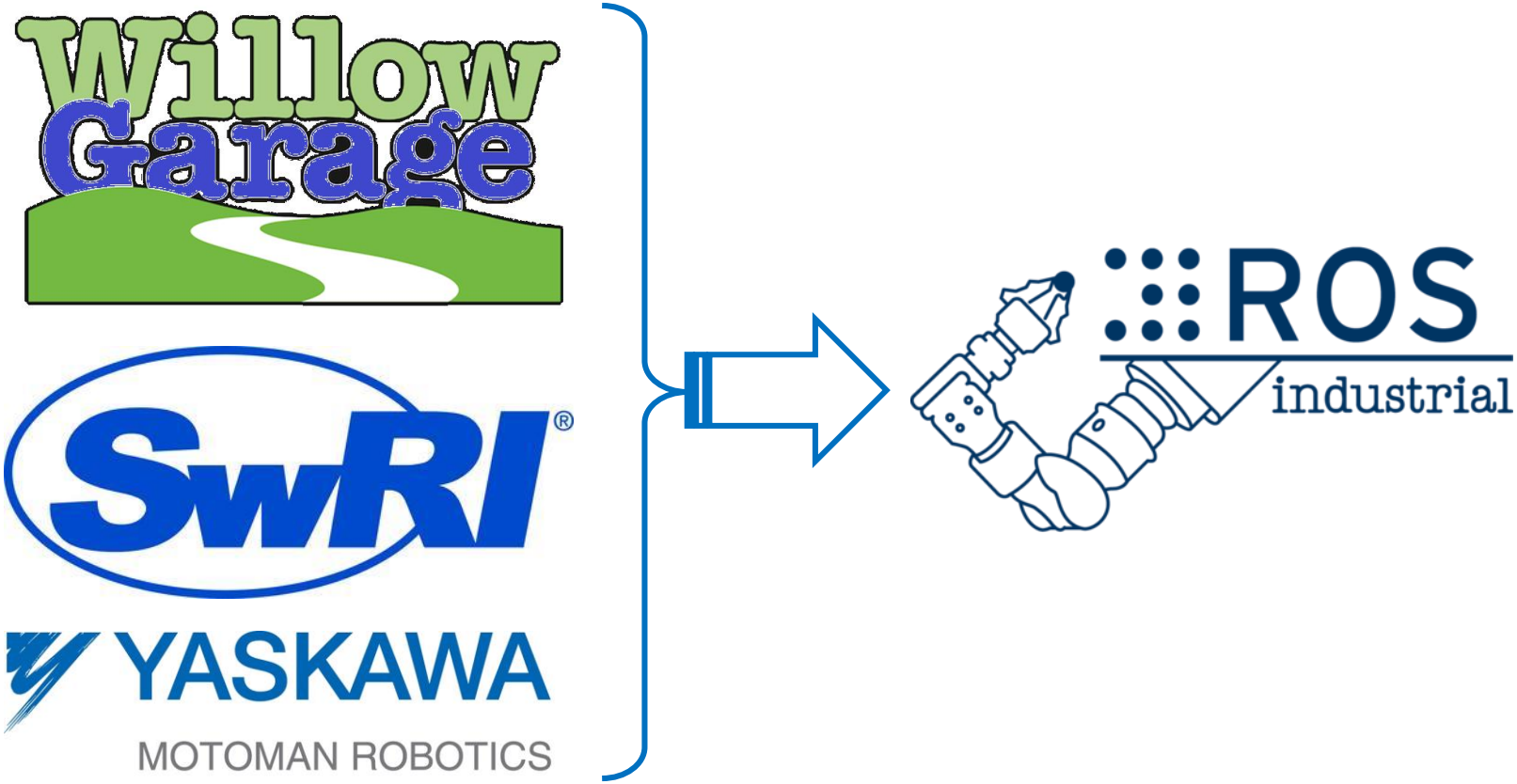
- 理论现状
  - 从运动学规划角度
  - 给定**足够多**的时间，一定能够最优且完备地求解到轨迹
- 从理论的角度而言，这个问题已经**解决**了
- 但从实用的角度而言，如何在有限时间内获得尽可能最优的路径，尚未解决。

- 推荐阅读
  - 机器人学
    - Craig, John J. ***Introduction to robotics: mechanics and control***. Vol. 3. Upper Saddle River: Pearson Prentice Hall, 2005.
  - 碰撞检测
    - Pan J, Chitta S, Manocha D. **FCL: A general purpose library for collision and proximity queries**[C]//Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012: 3859-3866.
  - 运动规划
    - 《运动规划 | 简介篇》
    - Choset, Howie M. ***Principles of robot motion: theory, algorithms, and implementation***. MIT press, 2005.

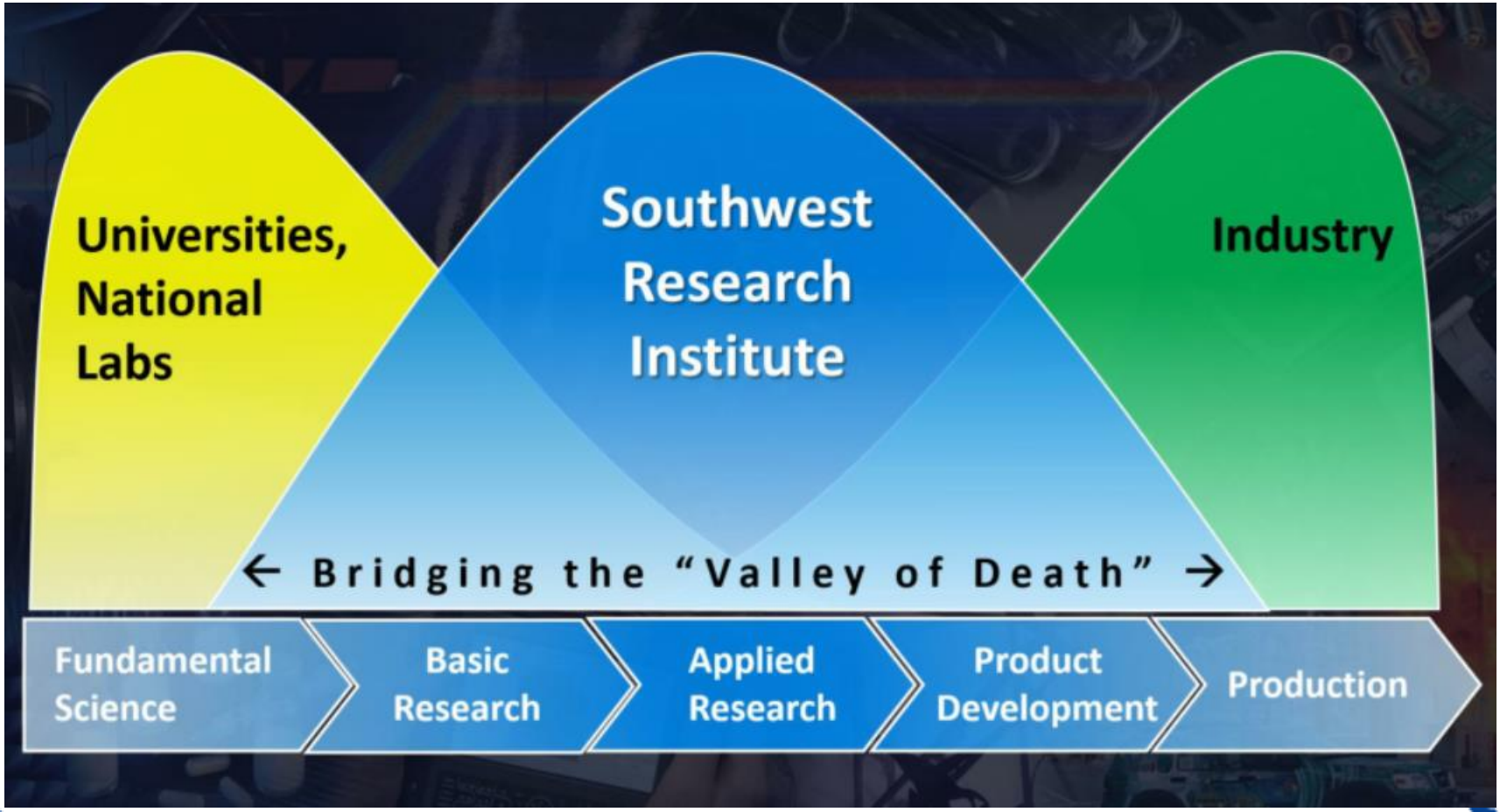
工业机器人

（ROS篇）

ROS-I


Willow Garage
SwRI
YASKAWA
MOTOMAN ROBOTICS
ROS
industrial

ROS-I


Universities, National Labs
Southwest Research Institute
Industry
← Bridging the "Valley of Death" →
Fundamental Science
Basic Research
Applied Research
Product Development
Production

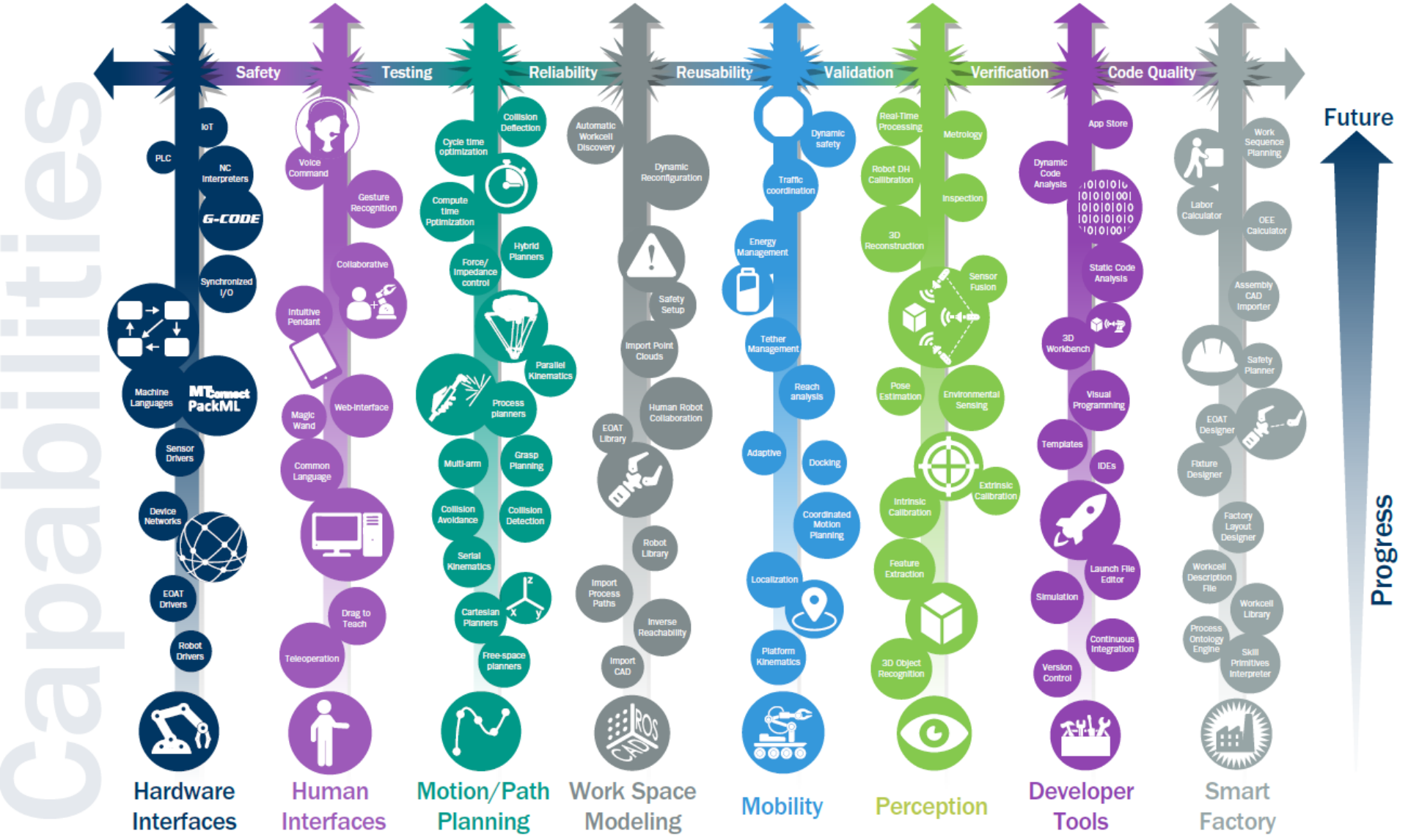
ROS-I

Shaun Edwards

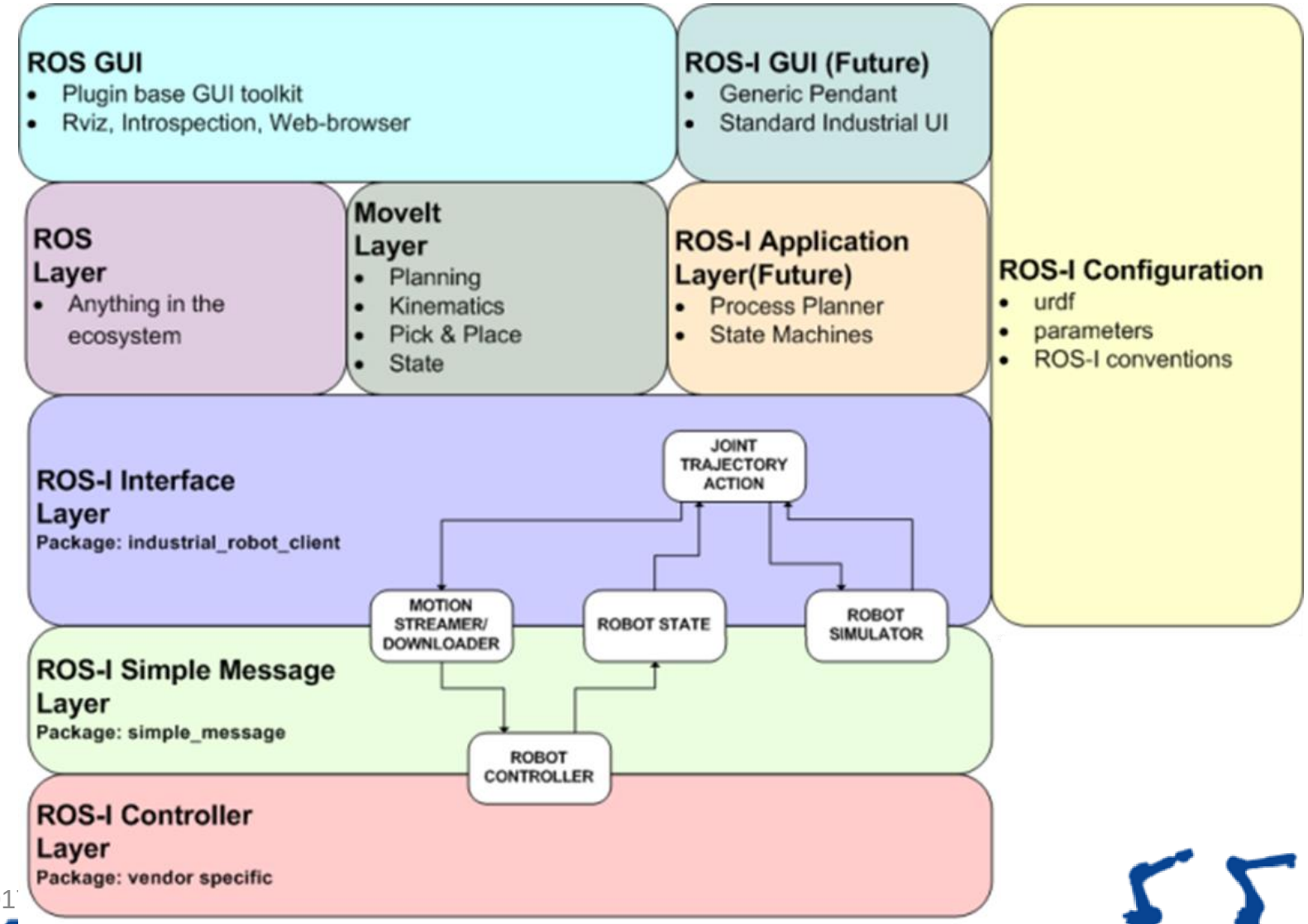
- *Founder and Lead Architect of ROS-Industrial Software*
- Co-founder at PlusOne Robotics
- Visiting researcher at Willow Garage.

ROS-I

Roadmap


Capabilities
Safety
Testing
Reliability
Reusability
Validation
Verification
Code Quality
Future
Progress
IoT
PLC
NC Interpreters
G-CODE
Synchronized I/O
Machine Languages
MTConnect PackML
Sensor Drivers
Device Networks
EOAT Drivers
Robot Drivers
Hardware Interfaces
Voice Command
Gesture Recognition
Collaborative
Intuitive Pendant
Magic Wand
Web-Interface
Common Language
Drag to Teach
Teleoperation
Human Interfaces
Collision Deflection
Cycle time optimization
Compute time Optimization
Hybrid Planners
Force/ Impedance control
Parallel Kinematics
Process planners
Multi-arm
Grasp Planning
Collision Avoidance
Collision Detection
Serial Kinematics
Cartesian Planners
Free-space planners
Motion/Path Planning
Automatic Workcell Discovery
Dynamic Reconfiguration
Safety Setup
Import Point Clouds
Human Robot Collaboration
EOAT Library
Robot Library
Import Process Paths
Inverse Reachability
Import CAD
Work Space Modeling
Dynamic safety
Traffic coordination
Energy Management
Tether Management
Reach analysis
Adaptive
Docking
Coordinated Motion Planning
Localization
Platform Kinematics
Mobility
Real-Time Processing
Metrology
Robot DH Calibration
Inspection
3D Reconstruction
Sensor Fusion
Pose Estimation
Environmental Sensing
Intrinsic Calibration
Extrinsic Calibration
Feature Extraction
3D Object Recognition
Perception
App Store
Dynamic Code Analysis
Static Code Analysis
3D Workbench
Visual Programming
Templates
IDEs
Launch File Editor
Simulation
Continuous Integration
Version Control
Developer Tools
Work Sequence Planning
Labor Calculator
OEE Calculator
Assembly CAD Importer
Safety Planner
EOAT Designer
Fixture Designer
Factory Layout Designer
Workcell Description File
Workcell Library
Process Ontology Engine
Skill Primitives Interpreter
Smart Factory

Architecture

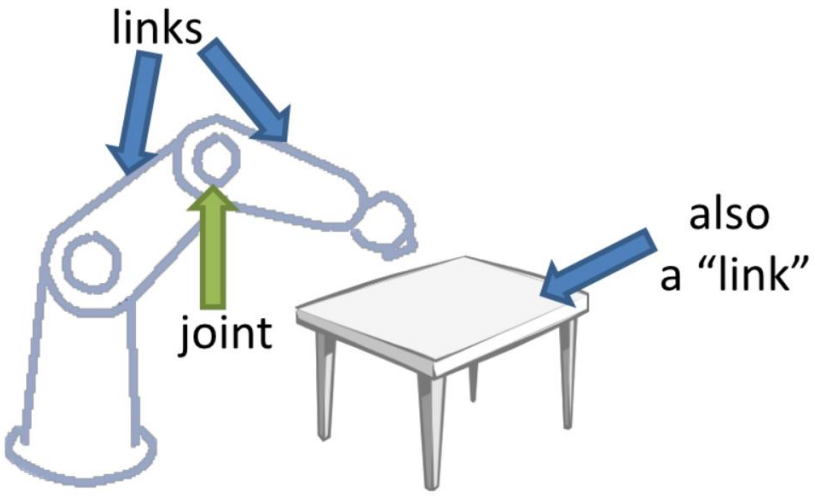


ROS-I

理论	ROS
运动学正解	URDF+TF
逆解（数值解）	KDL, TRAC_IK
逆解（解析解）	IKFast
碰撞检测	FCL
运动规划	MoveIt!

URDF

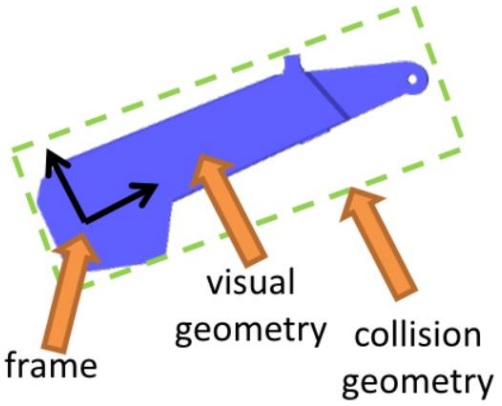
- URDF
  - Unified Robot Description Format
  - 通过 XML 格式描述机器人（和工作空间）的属性
    - Links: 几何信息、坐标信息等
    - Joints: 定义连杆间的关系



URDF

- 连杆 Link
  - 每个 link 对应一个 TF frame
  - 包含可视化/碰撞模型

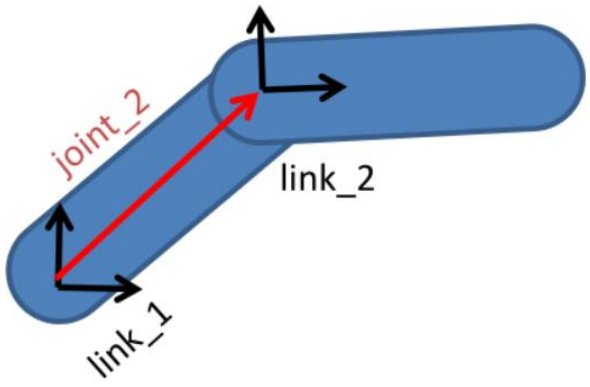
```
<link name="link_4">
   <visual>
        <geometry>
            <mesh filename="link_4.stl"/>
        </geometry>
        <origin xyz="0 0 0" rpy="0 0 0" />
   </visual>
   <collision>
        <geometry>
            <cylinder length="0.5" radius="0.1"/>
        </geometry>
        <origin xyz="0 0 -0.05" rpy="0 0 0" />
   </collision>
</link>
```



URDF

- 关节 Joint
  - 定义相邻连杆的关系
    - 类型：fixed, free, linear, rotary
  - 定义运动轴
  - 定义关节约束（位置、速度等）

```
<joint name="joint_2" type="revolute">
    <parent link="link_1"/>
    <child link="link_2"/>
    <origin xyz="0.2 0.2 0" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-3.14" upper="3.14" velocity="1.0"/>
</joint>
```



URDF

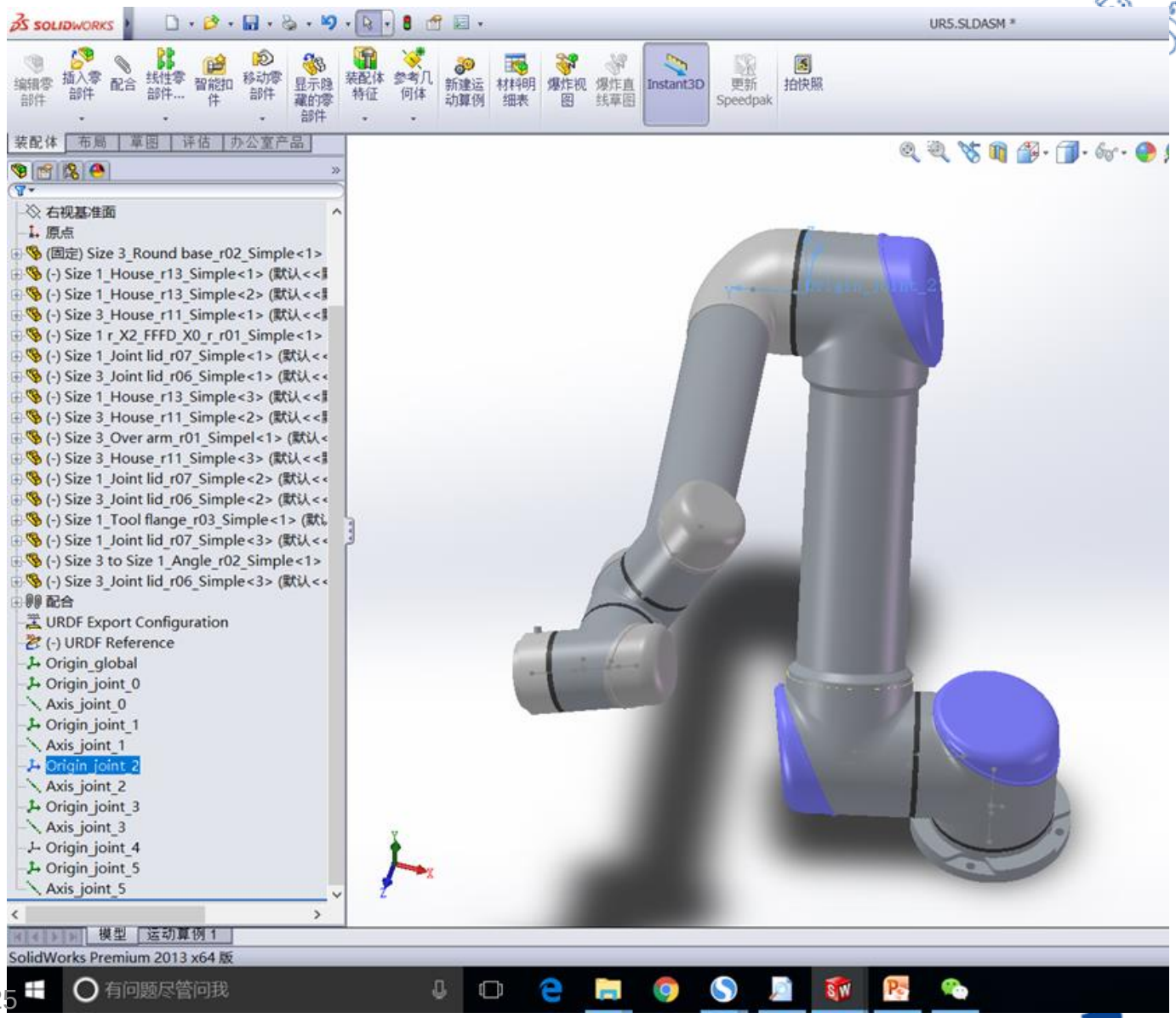
- URDF
  - 可以用 xacro 文件精简 URDF 文件（略）
  - 添加动力学属性、关节控制器等信息后，可以在 Gazebo 中使用。

- 看看 UR5 的 URDF 文件
  - ur_description
    - urdf
      - ur5.urdf.xacro

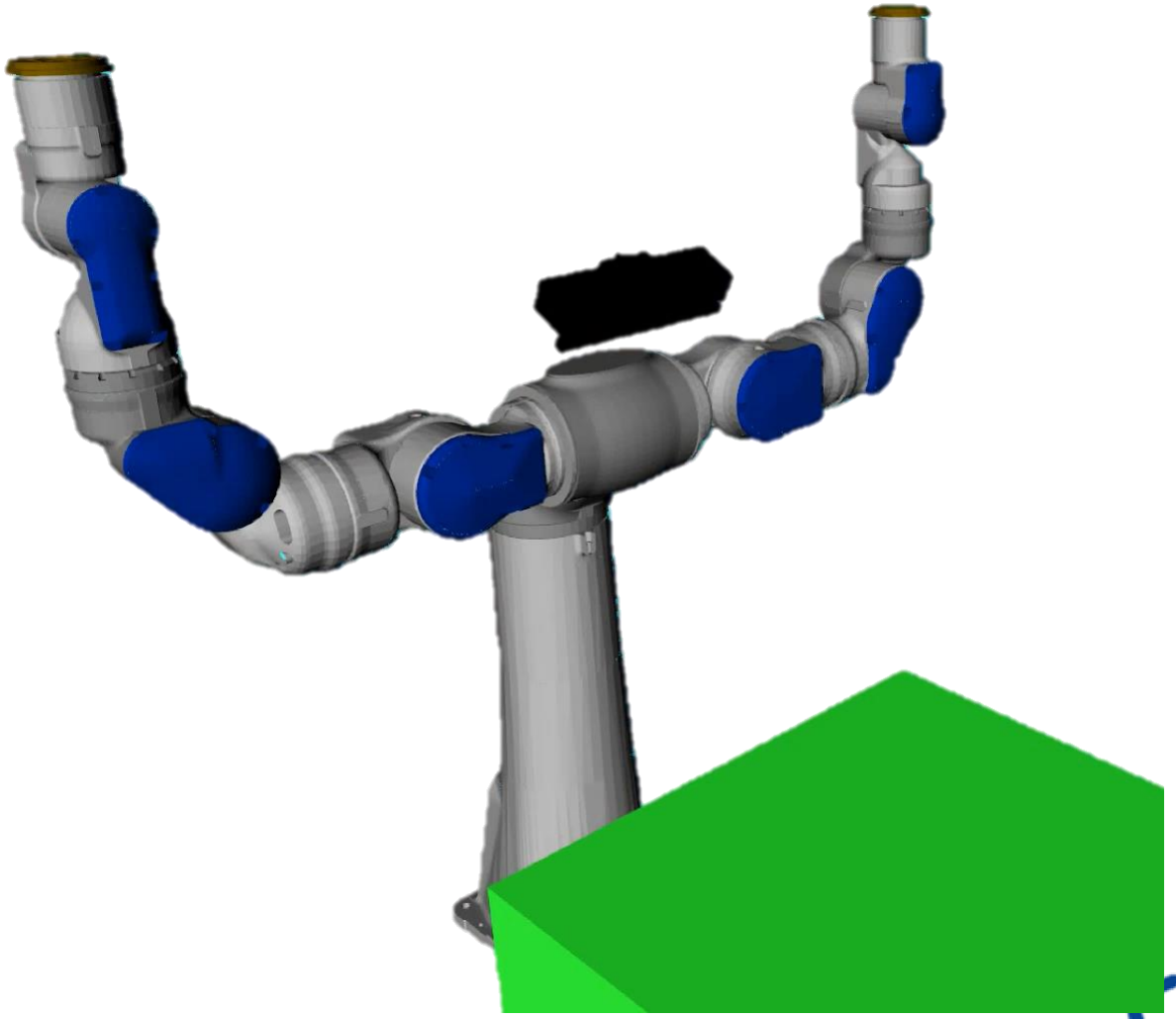
URDF

- 如何获得 URDF 文件
  - 自己手写
    - 推荐做一遍
  - 使用 SolidWorks 转 URDF 的插件
    - sw_urdf_exporter
    - 在 SW 中先定义好每个连杆坐标系
    - 在生成的 URDF 中增加简化的碰撞模型
    - 在处理稍微复杂点的机器人时，可能会出问题
    - 参考教程：《如何建立URDF》
  - 与实际机器人校验
    - 相对位置、关节极限、转动方向等

URDF

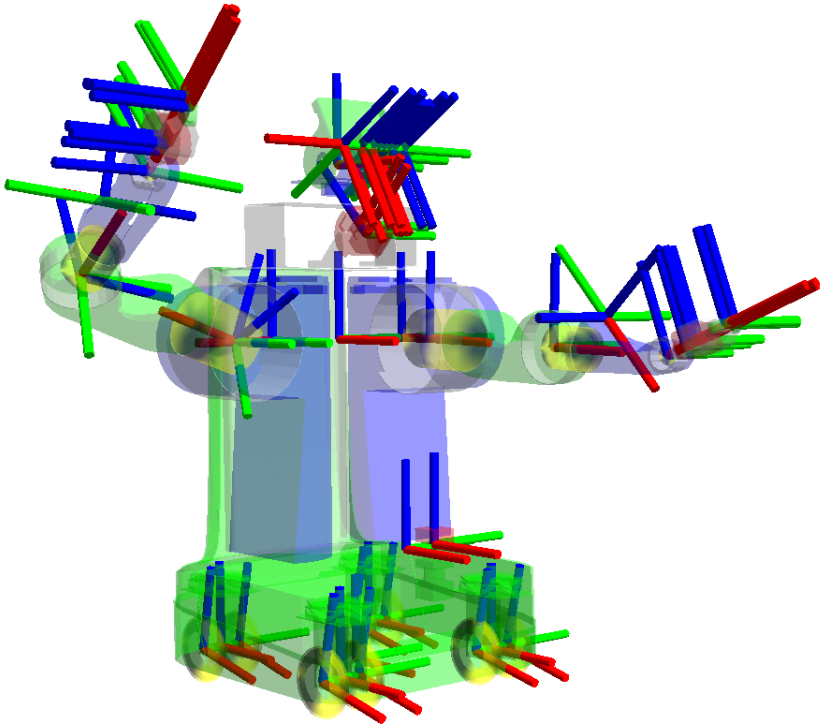

SOLIDWORKS
UR5.SLDASM *
Instant3D
右视基准面
原点
(固定) Size 3_Round base_r02_Simple<1>
(-) Size 1_House_r13_Simple<1>
(-) Size 1_House_r13_Simple<2>
(-) Size 3_House_r11_Simple<1>
(-) Size 1 r_X2_FFFD_X0 r_r01_Simple<1>
(-) Size 1_Joint lid_r07_Simple<1>
(-) Size 3_Joint lid_r06_Simple<1>
(-) Size 1_House_r13_Simple<3>
(-) Size 3_House_r11_Simple<2>
(-) Size 3_Over arm_r01_Simpel<1>
(-) Size 3_House_r11_Simple<3>
(-) Size 1_Joint lid_r07_Simple<2>
(-) Size 3_Joint lid_r06_Simple<2>
(-) Size 1_Tool flange_r03_Simple<1>
(-) Size 1_Joint lid_r07_Simple<3>
(-) Size 3 to Size 1_Angle_r02_Simple<1>
(-) Size 3_Joint lid_r06_Simple<3>
配合
URDF Export Configuration
(-) URDF Reference
Origin_global
Origin_joint_0
Axis_joint_0
Origin_joint_1
Axis_joint_1
Origin_joint_2
Axis_joint_2
Origin_joint_3
Axis_joint_3
Origin_joint_4
Origin_joint_5
Axis_joint_5
模型
运动算例 1
SolidWorks Premium 2013 x64 版
有问题尽管问我

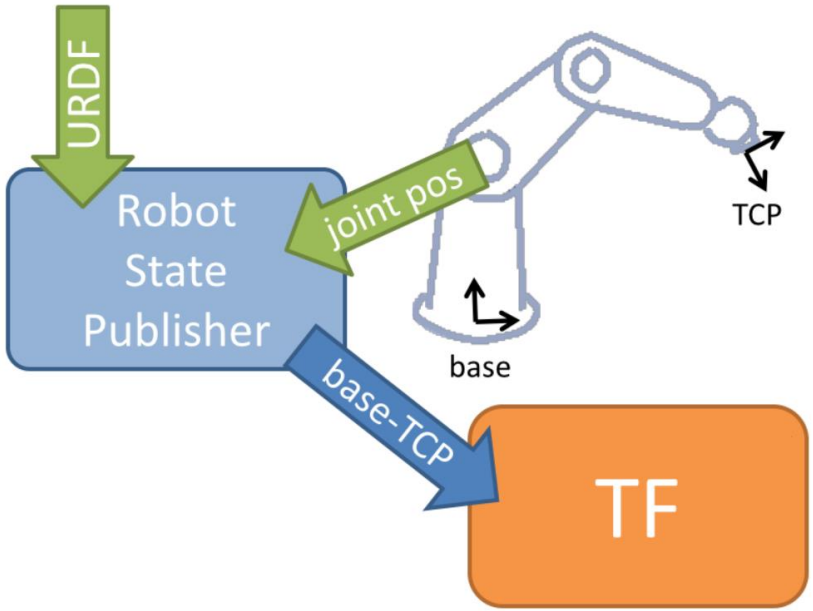
URDF



URDF

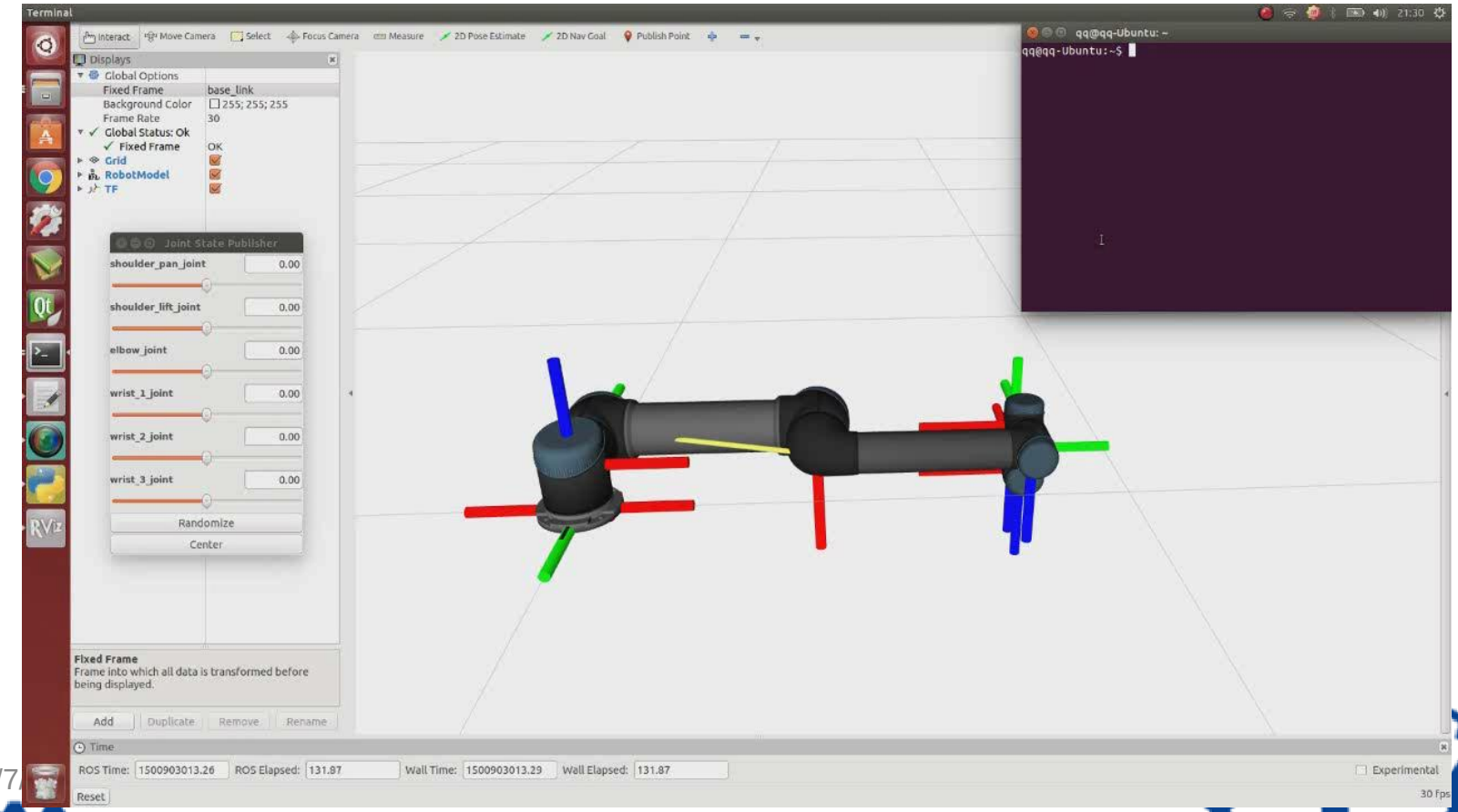
- TF





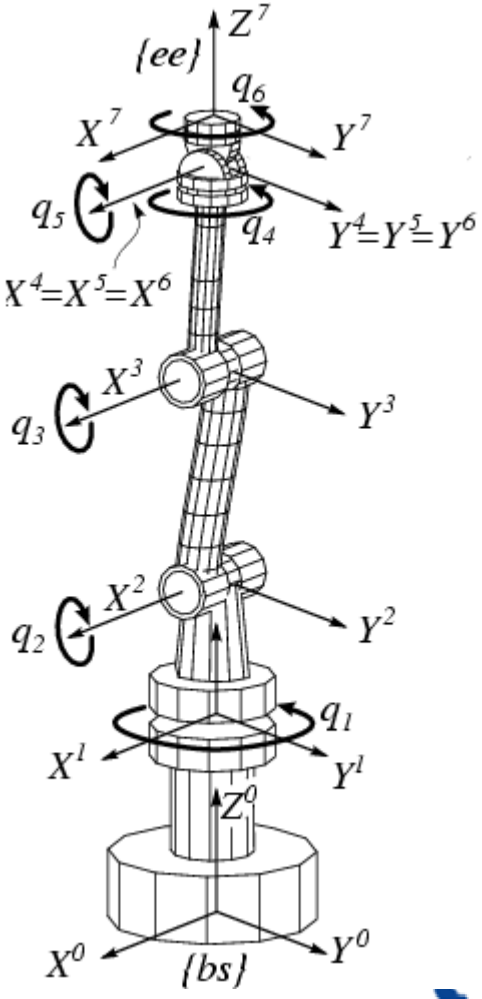
URDF

- TF
  - /base_link → /tool0　　运动学正解

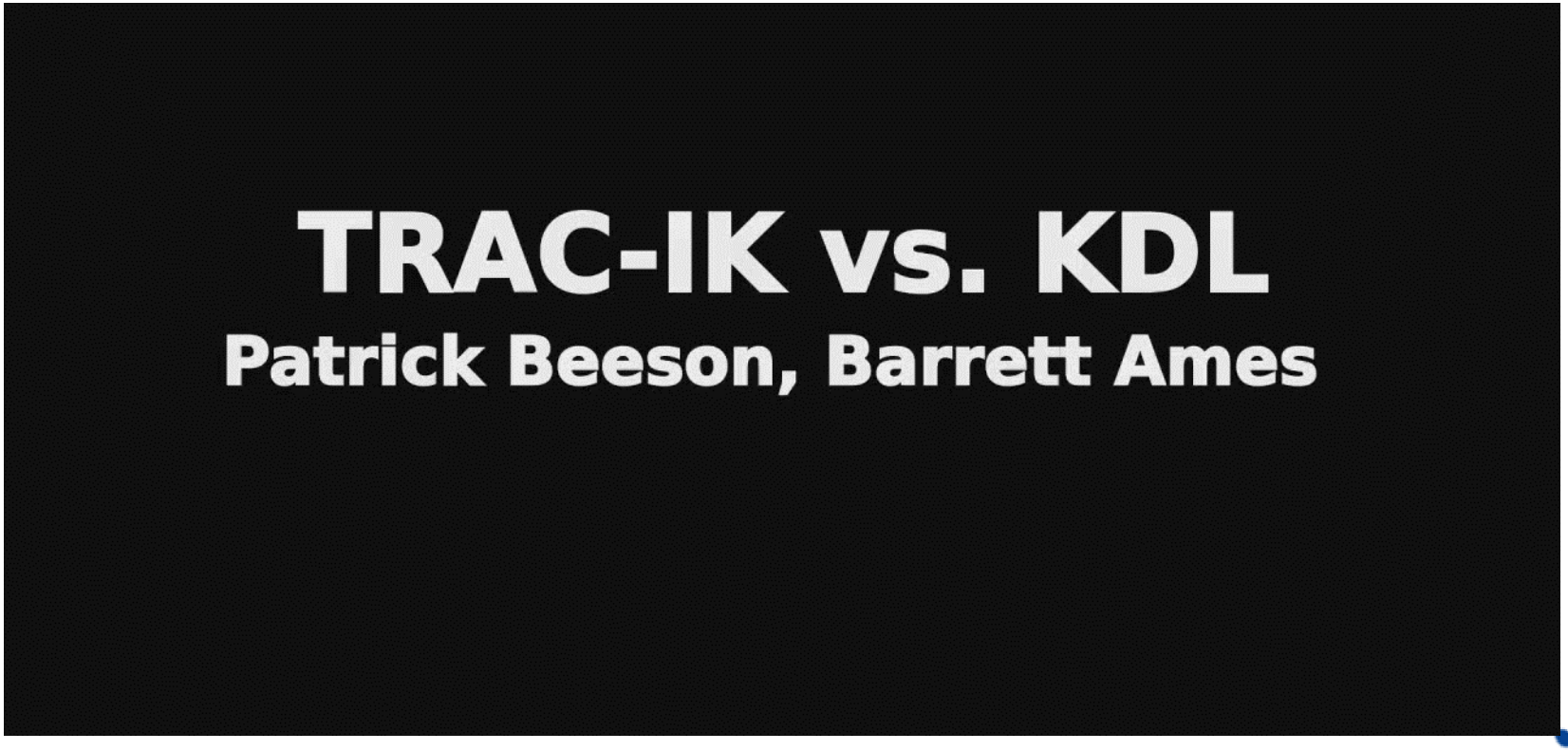


KDL

- KDL
  - Orocos: **K**inematics and **D**ynamics **L**ibrary
  - 逆雅克比数值解；
  - 用于计算 KDL tree 的运动学
    - 正解
    - 逆解
    - 雅克比
    - …
  - kdl_parser: 将 URDF 转换成 KDL tree



- TRAC-IK
  - TRAC-IK handles joint-limited chains better than KDL

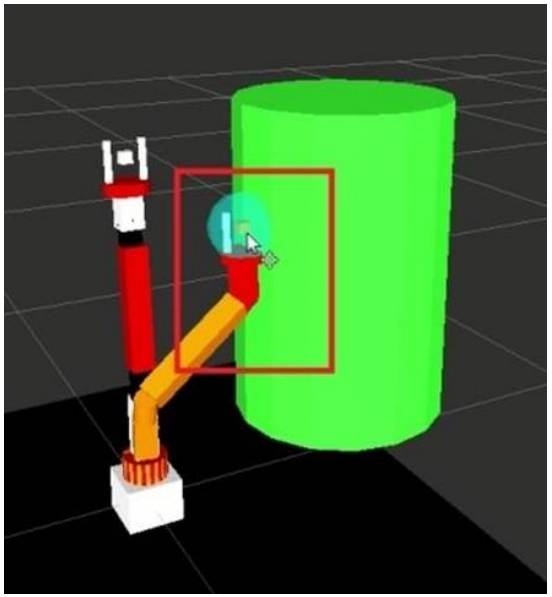


IKFast

- IKFast
  - OpenRave
  - 6 dof and 7 dof
  - 逆解解析解！ ~1000倍
  - 生成 C++ file
  - 参考教程： Generate IKFast Plugin Tutorial
  - 原理：
    - Diankov, Rosen. "*Automated construction of robotic manipulation programs*." (2010).

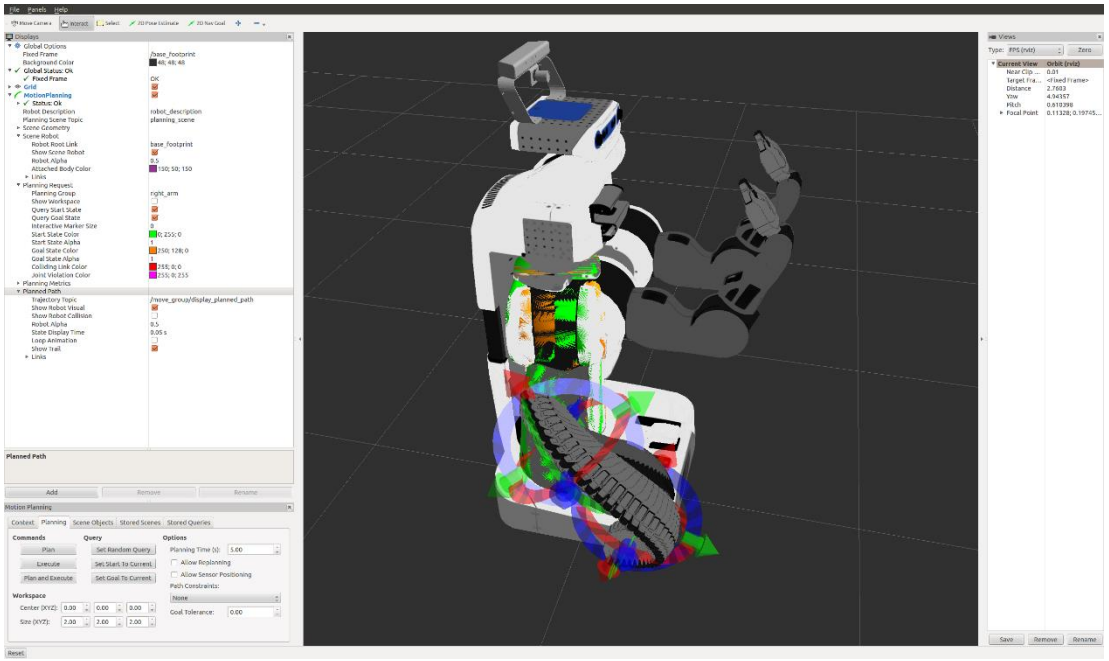
FCL

- FCL
  - Flexible-Collision-Library
  - 作者: Jia Pan （潘佳）
  - 物体:
    - Simple shapes: box, sphere, cylinder, …
    - Mesh: STL, dae
    - Point cloud
    - Octree
  - 功能：
    - 碰撞检测
    - 距离计算
    - 接触信息

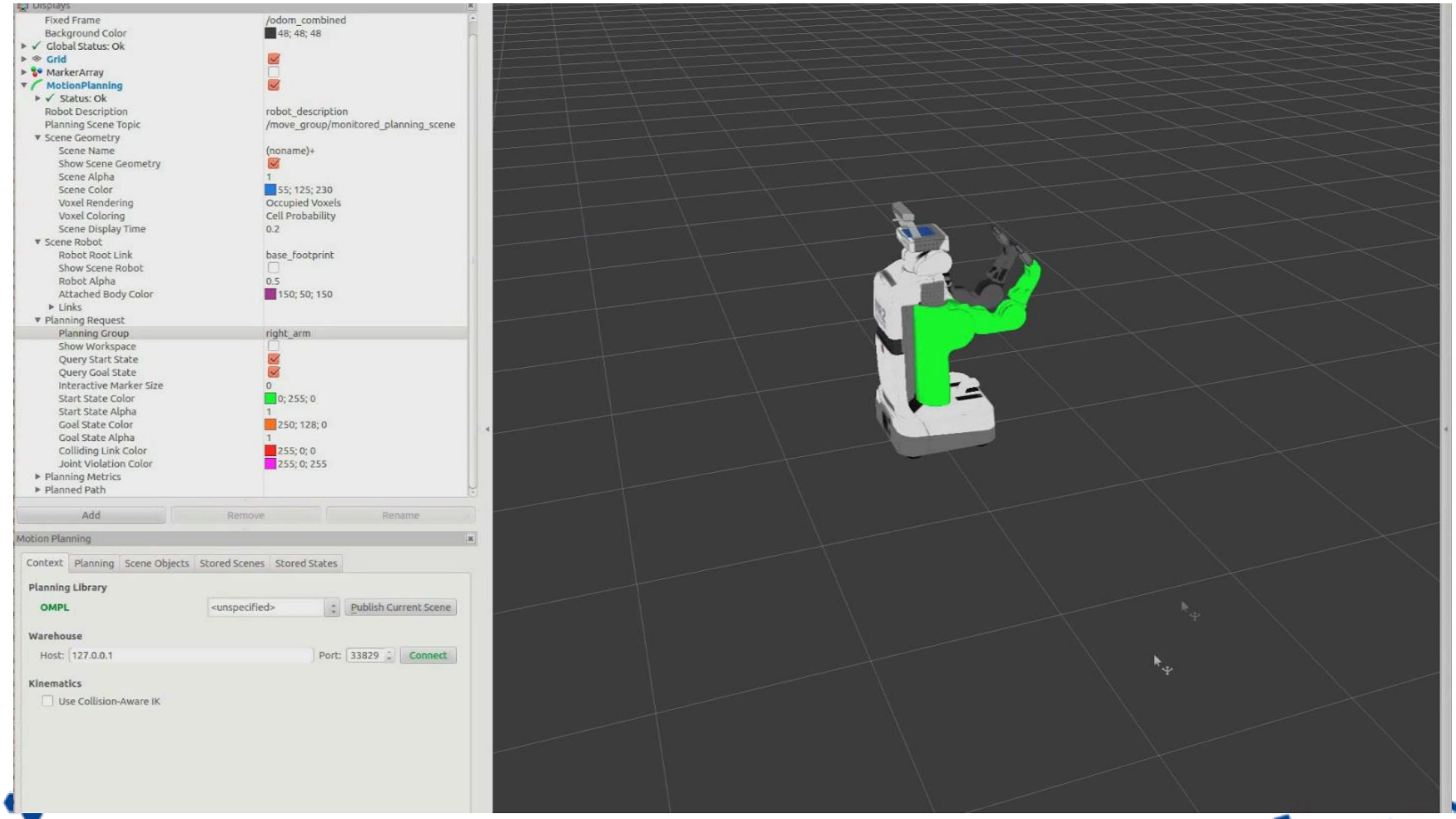


MoveIt!

- MoveIt!
  - 它是目前最先进的移动操作机器人软件，整合了最先进的运动规划、操作、3D感知、运动学、控制与导航算法。为这方面的开发人员提供了一个十分便利的开发平台。

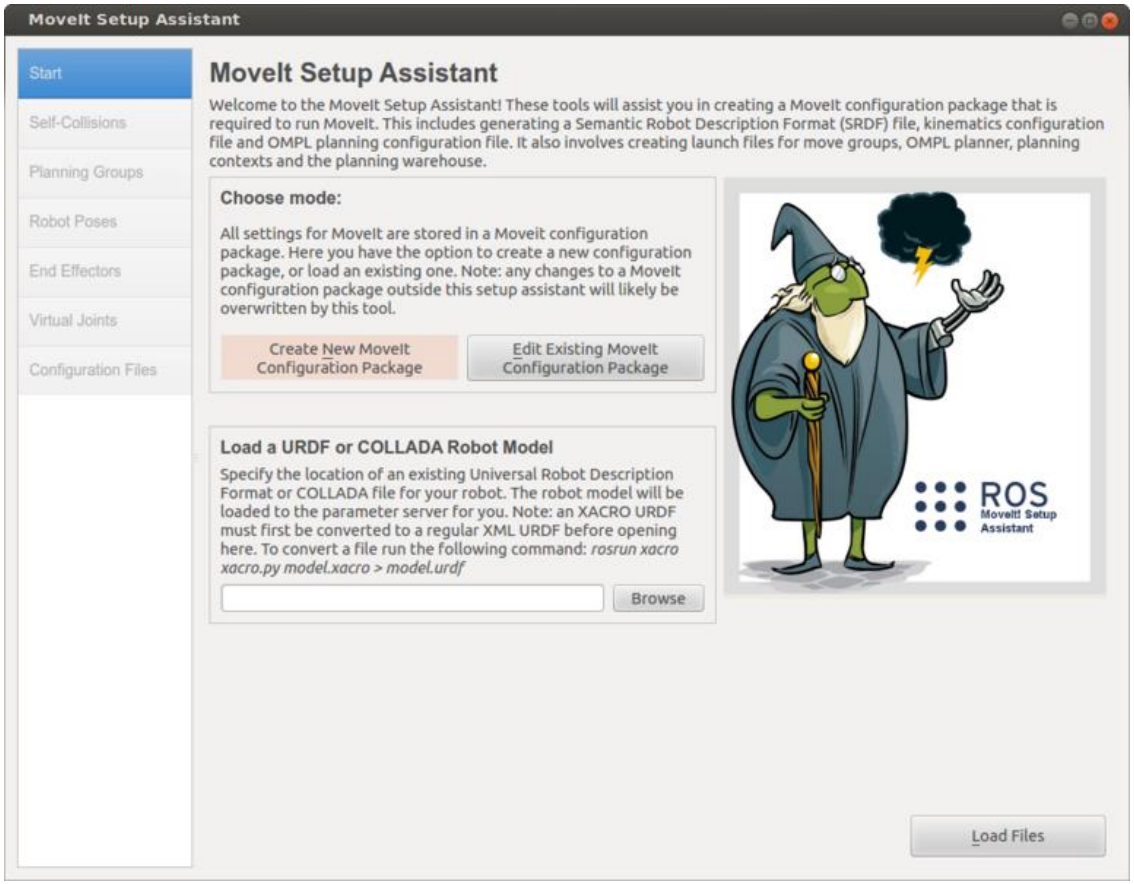


MoveIt!


Displays
Fixed Frame /odom_combined
Background Color 48; 48; 48
Global Status: Ok
Grid
MarkerArray
MotionPlanning
Status: Ok
Robot Description robot_description
Planning Scene Topic /move_group/monitored_planning_scene
Scene Geometry
Scene Name (noname)+
Show Scene Geometry
Scene Alpha 1
Scene Color 55; 125; 230
Voxel Rendering Occupied Voxels
Voxel Coloring Cell Probability
Scene Display Time 0.2
Scene Robot
Robot Root Link base_footprint
Show Scene Robot
Robot Alpha 0.5
Attached Body Color 150; 50; 150
Links
Planning Request
Planning Group right_arm
Show Workspace
Query Start State
Query Goal State
Interactive Marker Size 0
Start State Color 0; 255; 0
Start State Alpha 1
Goal State Color 250; 128; 0
Goal State Alpha 1
Colliding Link Color 255; 0; 0
Joint Violation Color 255; 0; 255
Planning Metrics
Planned Path
Add
Remove
Rename
Motion Planning
Context
Planning
Scene Objects
Stored Scenes
Stored States
Planning Library
OMPL
<unspecified>
Publish Current Scene
Warehouse
Host: 127.0.0.1
Port: 33829
Connect
Kinematics
Use Collision-Aware IK

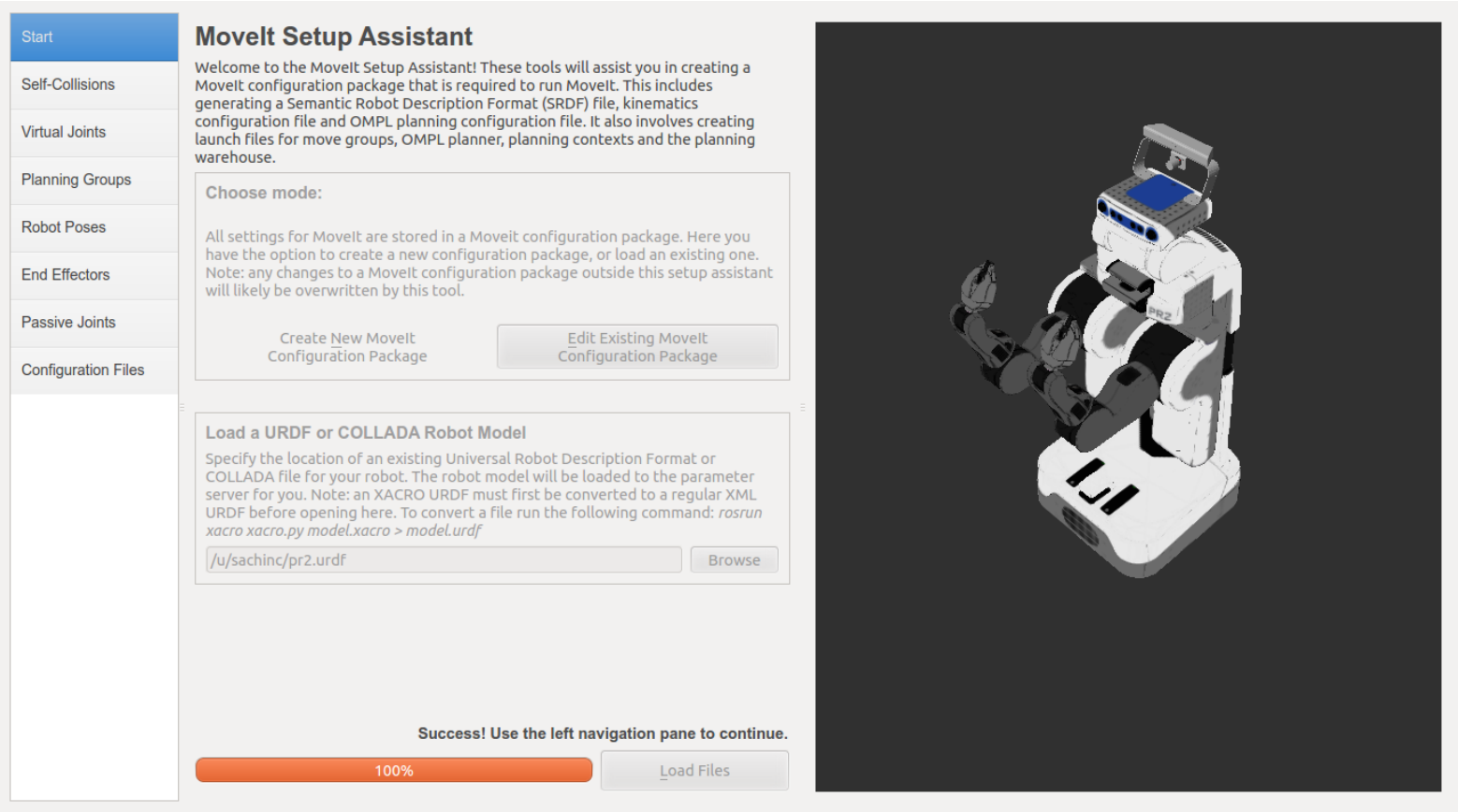
MoveIt!

- 打开 moveit_setup_assistant
  - roslaunch moveit_setup_assistant setup_assistant.launch



MoveIt!

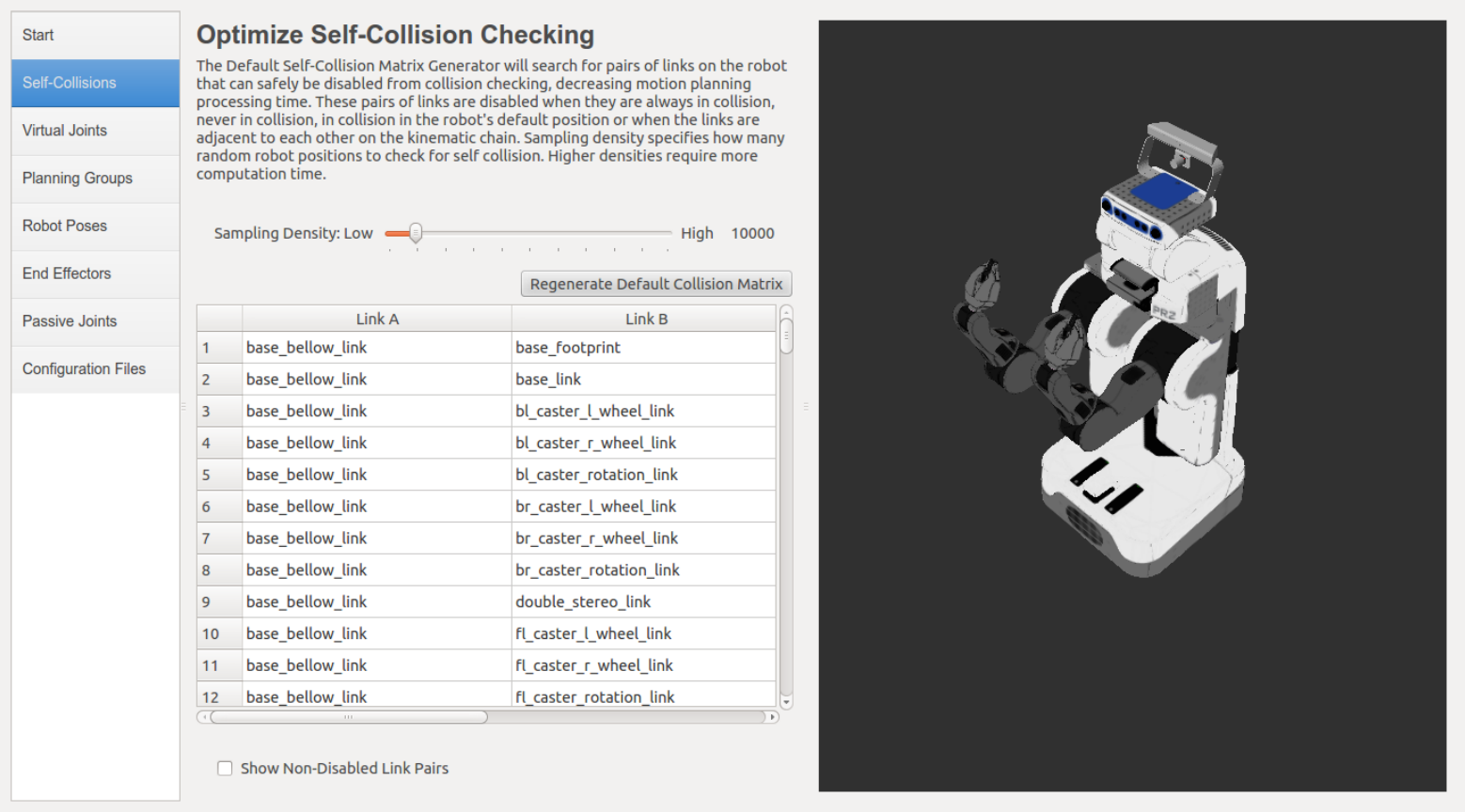
- 选择机器人的URDF文件



MoveIt!

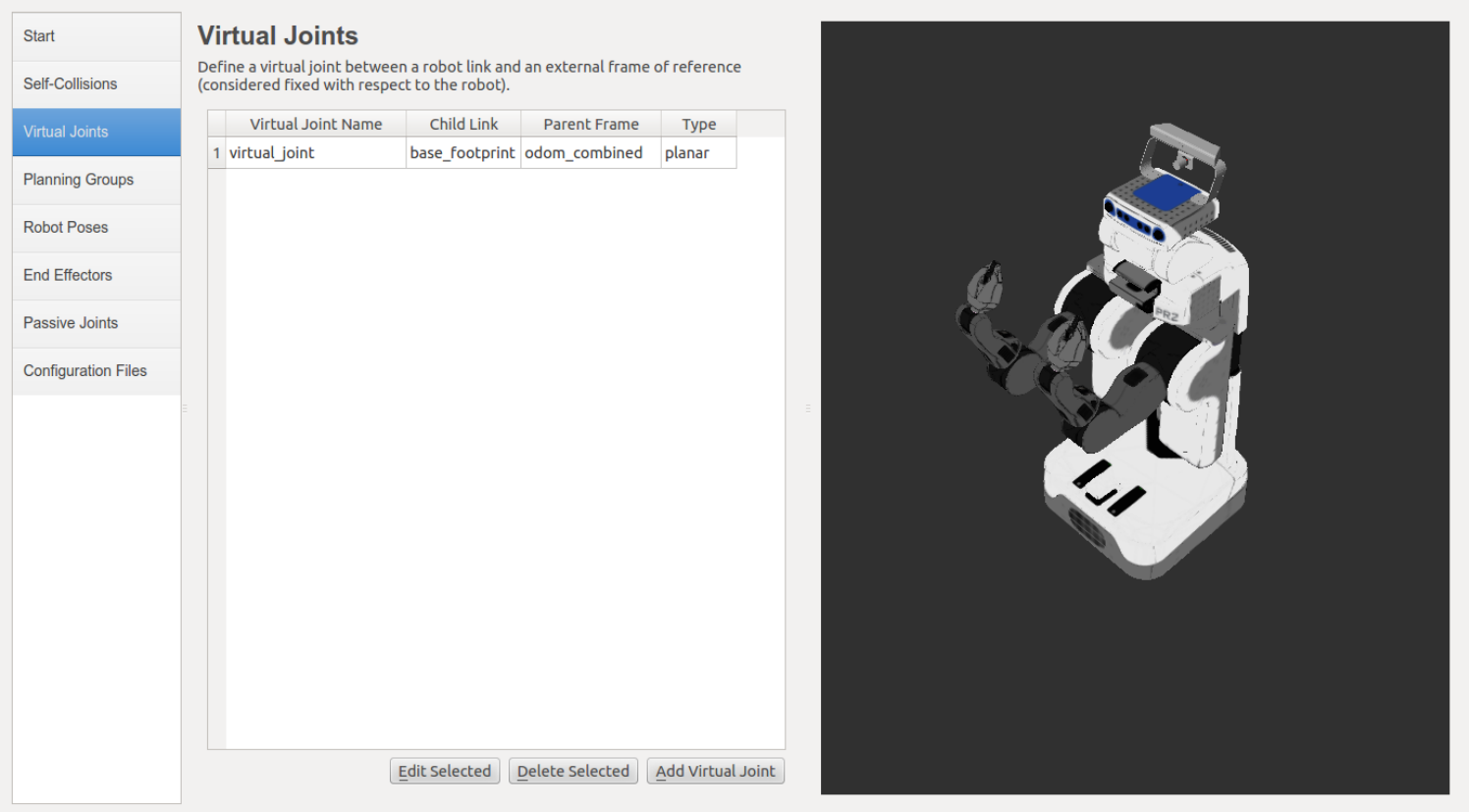
- 生成自身碰撞矩阵

	l_1	l_2	l_3	l_4	l_5	l_6
l_1		0	1	1	1	1
l_2	0		0	1	1	1
l_3	1	0		0	1	1
l_4	1	1	0		0	1
l_5	1	1	1	0		0
l_6	1	1	1	1	0	



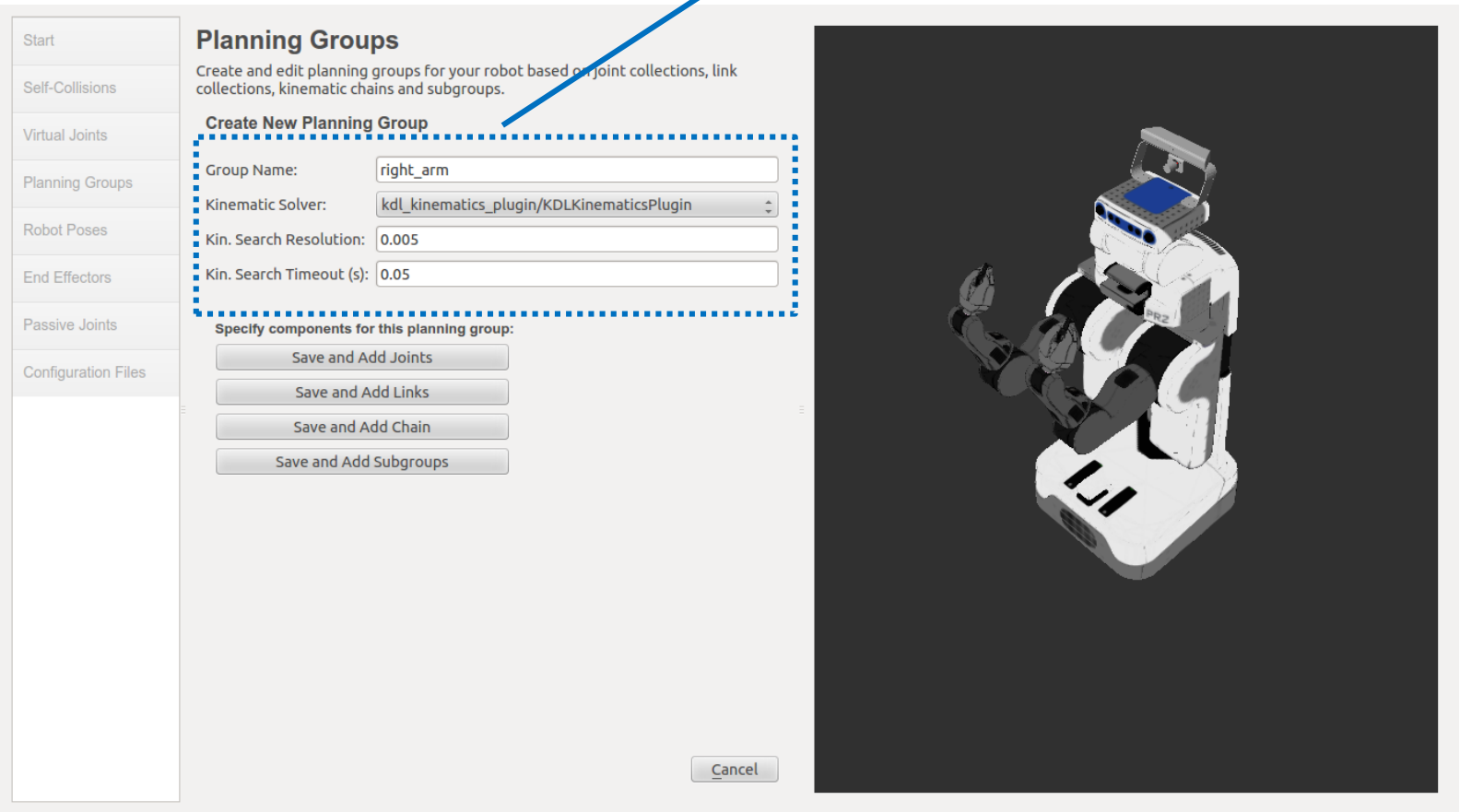
MoveIt!

- 添加虚拟关节
  - 机器人与世界坐标的关系

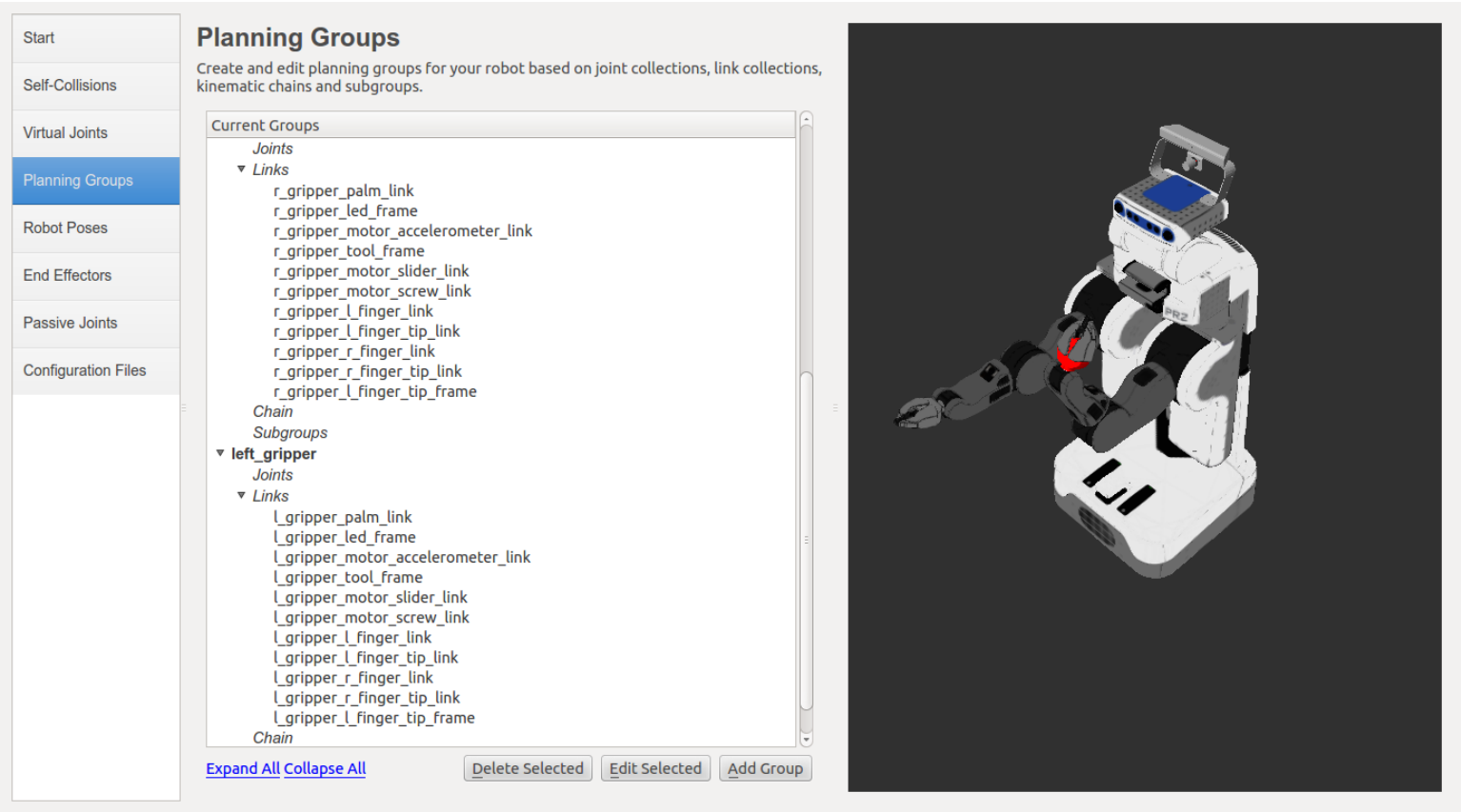


MoveIt!

- 设置 planning groups

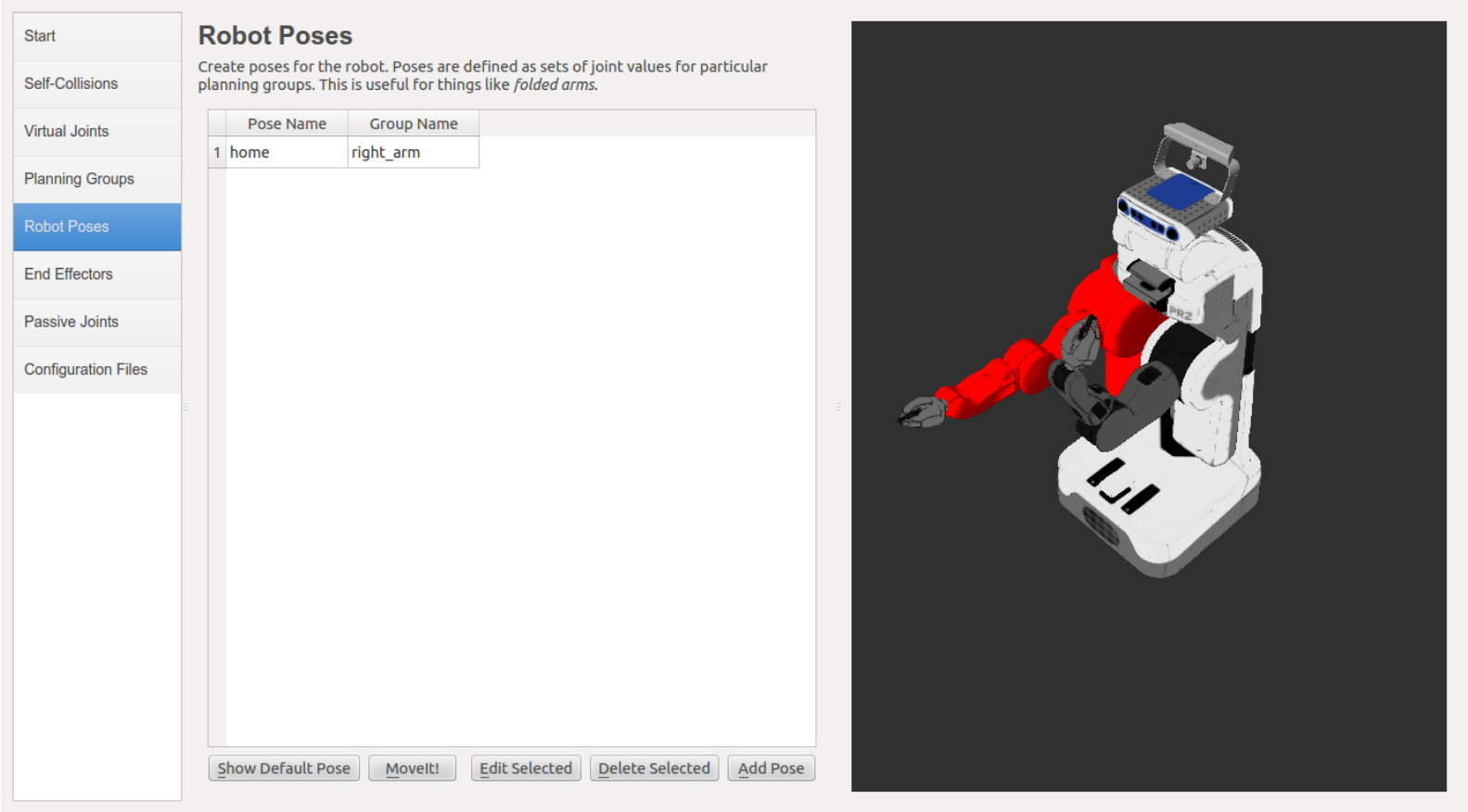


- 设置 planning groups



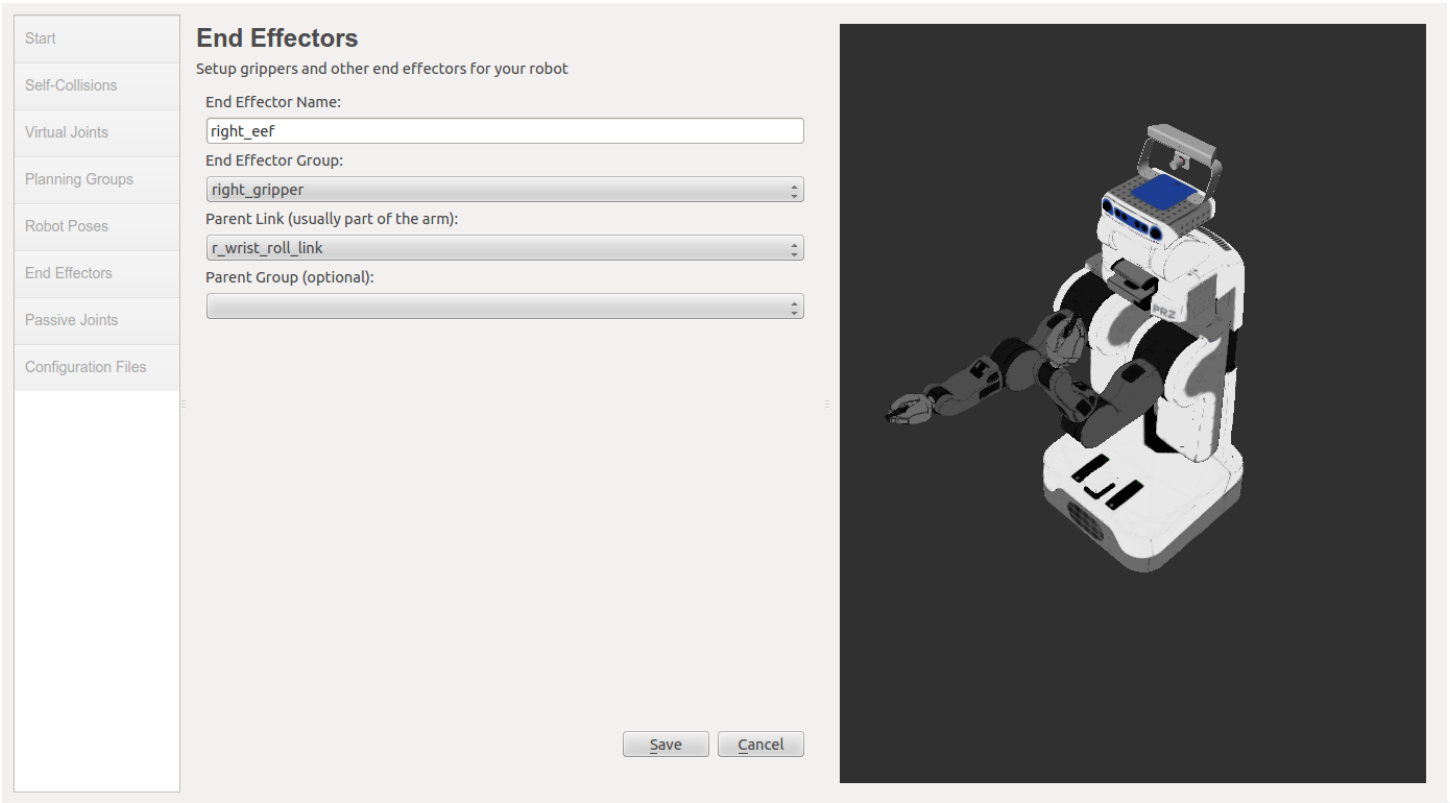
MoveIt!

- 设置默认姿势



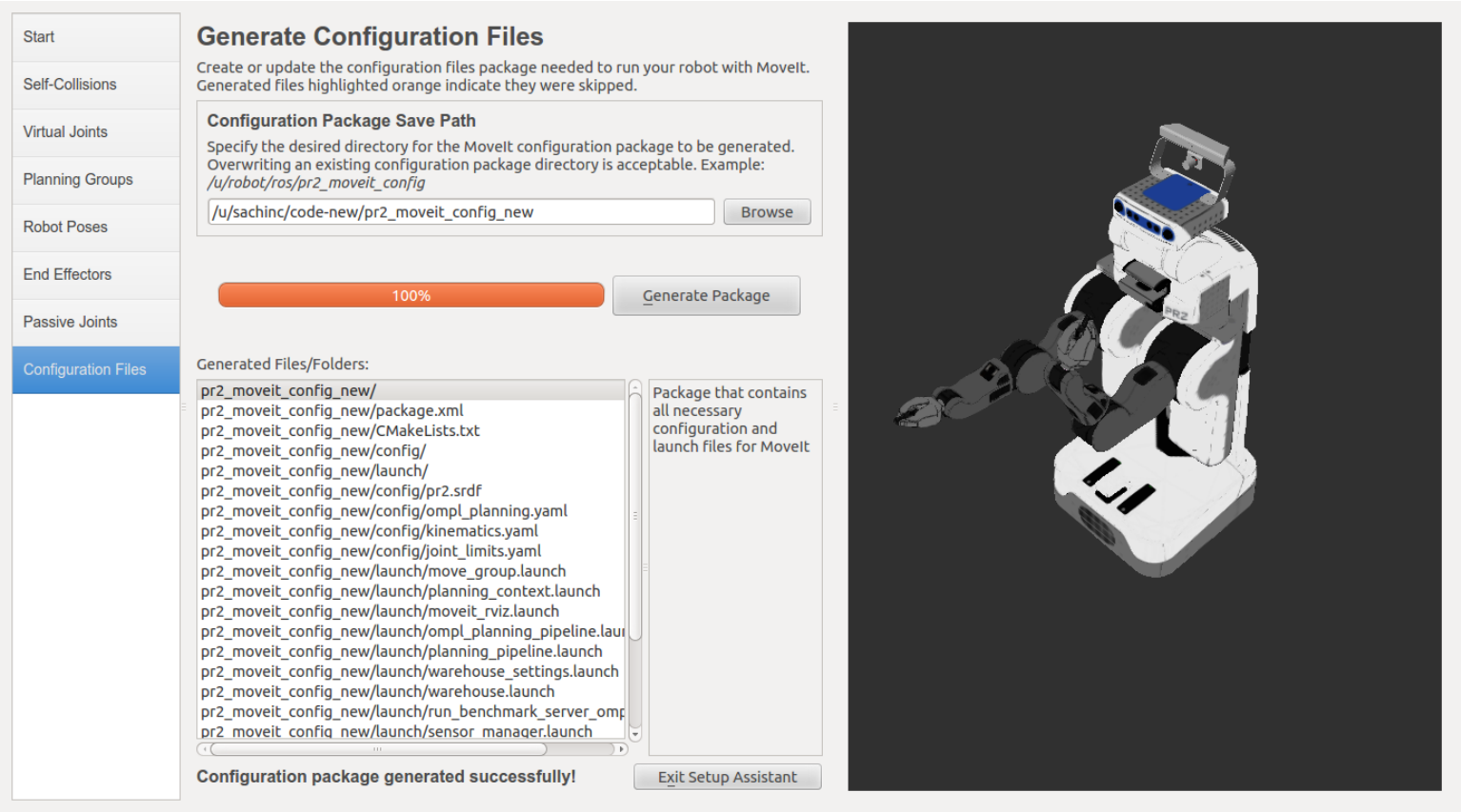
MoveIt!

- 设置末端执行器
  - 未设置好，Rviz 中无可拖动 Marker

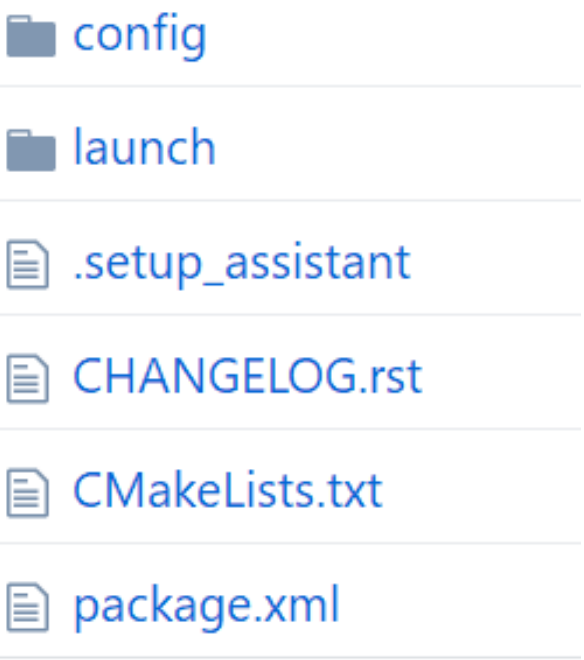


MoveIt!

- 生成 ROS package



- ROS package


config
launch
.setup_assistant
CHANGELOG.rst
CMakeLists.txt
package.xml

MoveIt!

- ROS package
  - config
    - ***.srdf**
    - 记录 planning group, end_effector, pose 等信息
    - Collision Matrix: （临时添加几个连杆，可以直接在这里修改）

```
<!--DISABLE COLLISIONS: By default it is assumed that any link of the robot could potentiall
<disable_collisions link1="base_link" link2="shoulder_link" reason="Adjacent" />
<disable_collisions link1="ee_link" link2="wrist_1_link" reason="Never" />
<disable_collisions link1="ee_link" link2="wrist_2_link" reason="Never" />
<disable_collisions link1="ee_link" link2="wrist_3_link" reason="Adjacent" />
<disable_collisions link1="forearm_link" link2="upper_arm_link" reason="Adjacent" />
<disable_collisions link1="forearm_link" link2="wrist_1_link" reason="Adjacent" />
<disable_collisions link1="shoulder_link" link2="upper_arm_link" reason="Adjacent" />
<disable_collisions link1="wrist_1_link" link2="wrist_2_link" reason="Adjacent" />
<disable_collisions link1="wrist_1_link" link2="wrist_3_link" reason="Never" />
<disable_collisions link1="wrist_2_link" link2="wrist_3_link" reason="Adjacent" />
```

- ROS package
  - config
    - **ompl_planning.yaml**
    - OMPL中的规划算法
    - 规划算法的参数选择
    - 每个 planning group 单独配置

```
  PRMkConfigDefault:
    type: geometric::PRM
    max_nearest_neighbors: 10
  PRMstarkConfigDefault:
    type: geometric::PRMstar
manipulator:
  planner_configs:
    - SBLkConfigDefault
    - ESTkConfigDefault
    - LBKPIECEkConfigDefault
    - BKPIECEkConfigDefault
    - KPIECEkConfigDefault
    - RRTkConfigDefault
    - RRTConnectkConfigDefault
    - RRTstarkConfigDefault
    - TRRTkConfigDefault
    - PRMkConfigDefault
    - PRMstarkConfigDefault
```

- ROS package
  - config
    - **joint_limits.yaml**
    - 配置每个关节的速度/加速度约束，用于轨迹插补

```
joint_limits:
  shoulder_pan_joint:
    has_velocity_limits: true
    max_velocity: 3.15
    has_acceleration_limits: true
    max_acceleration: 3.15
  shoulder_lift_joint:
    has_velocity_limits: true
    max_velocity: 3.15
    has_acceleration_limits: true
    max_acceleration: 3.15
```

MoveIt!

- ROS package
  - config
    - **kinematics.yaml**
    - 逆解求解器选择与参数配置
    - 直接修改可以切换不同的求解器
      - KDL，TRAC_IK，IKFast

```
manipulator:
  kinematics_solver: kdl_kinematics_plugin/KDLKinematicsPlugin
  kinematics_solver_search_resolution: 0.005
  kinematics_solver_timeout: 0.005
  kinematics_solver_attempts: 3
```

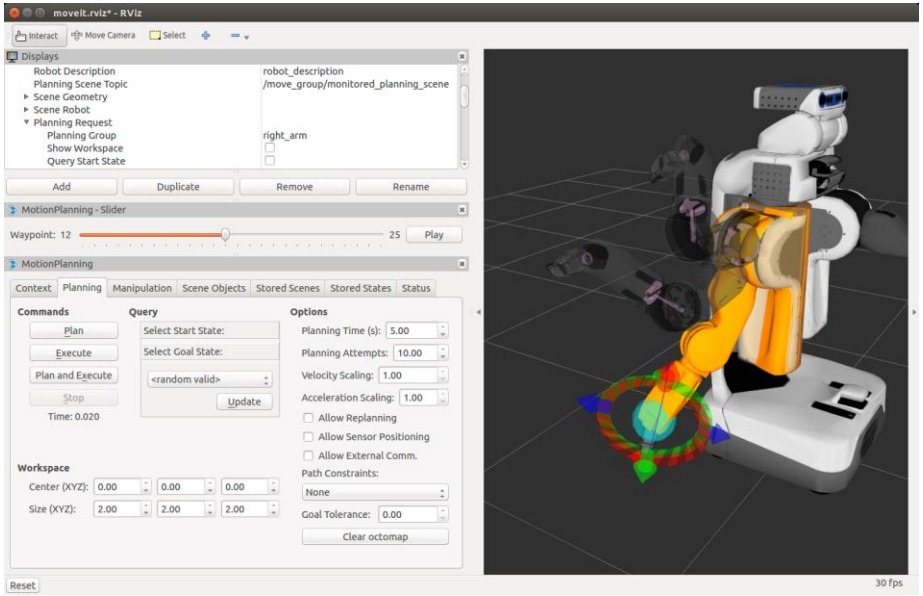
- ROS package
  - config
    - **fake_controllers.yaml**
    - 不连接真实机器人时，fake controller的配置
    - 如需要连接真实机器人，需要新建一个controllers.yaml
    - moveit_simple_controller_manager 会根据controllers.yaml 的信息生成相应的 action

```
controller_list:
  - name: ""
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint
```

MoveIt!

- ROS package
  - launch
    - 启动 MoveIt 的 launch 文件。
    - 不连接真实机器人的情况下，可以运行：

      roslaunch [robot]_moveit_config demo.launch



- 如果需要连接实际机器人
  - 准备好机器人 ROS_driver
    - Industrial_Robot_Driver_Spec
    - /joint_states: 发布关节角度
    - /joint_trajectory_action: 接收规划结果
    - /joint_path_command: 将轨迹下发给机器人（TCP/IP）
  - 配置 controllers.yaml 文件
    - Action 属性与 ROS_driver 相同

并联机构

- 并联结构
  - 工业机器人
  - URDF 不支持并联（闭环）结构

并联机构

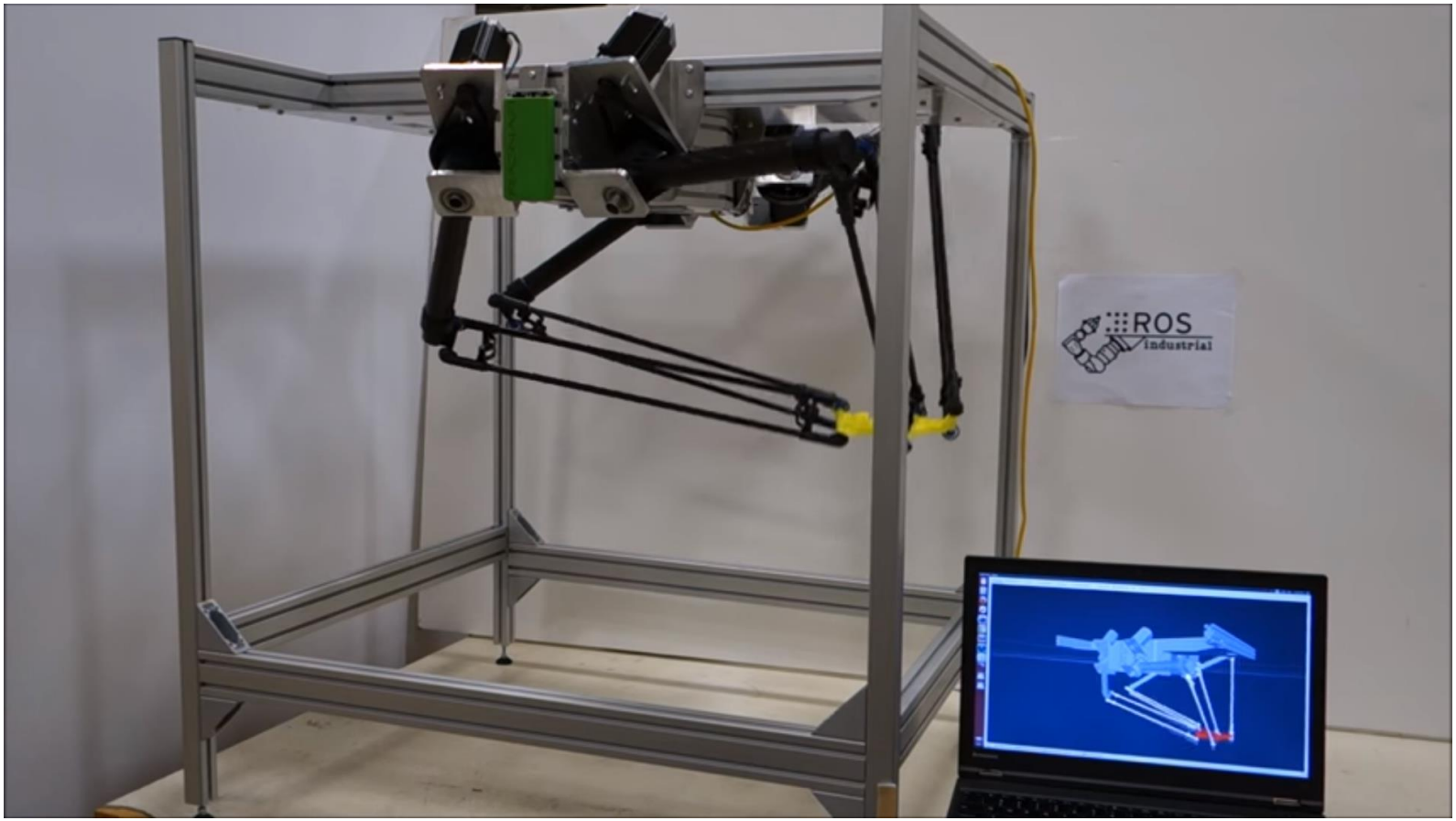
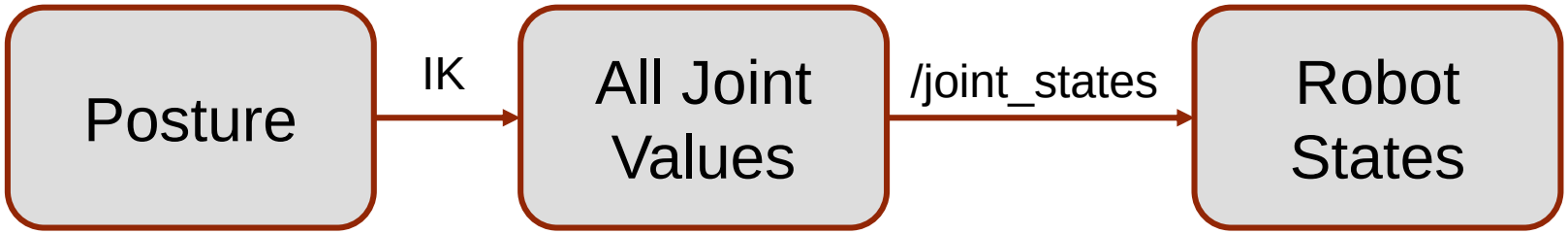


Image from: [Blue WorkForce Ragnar ROS-I Driver]

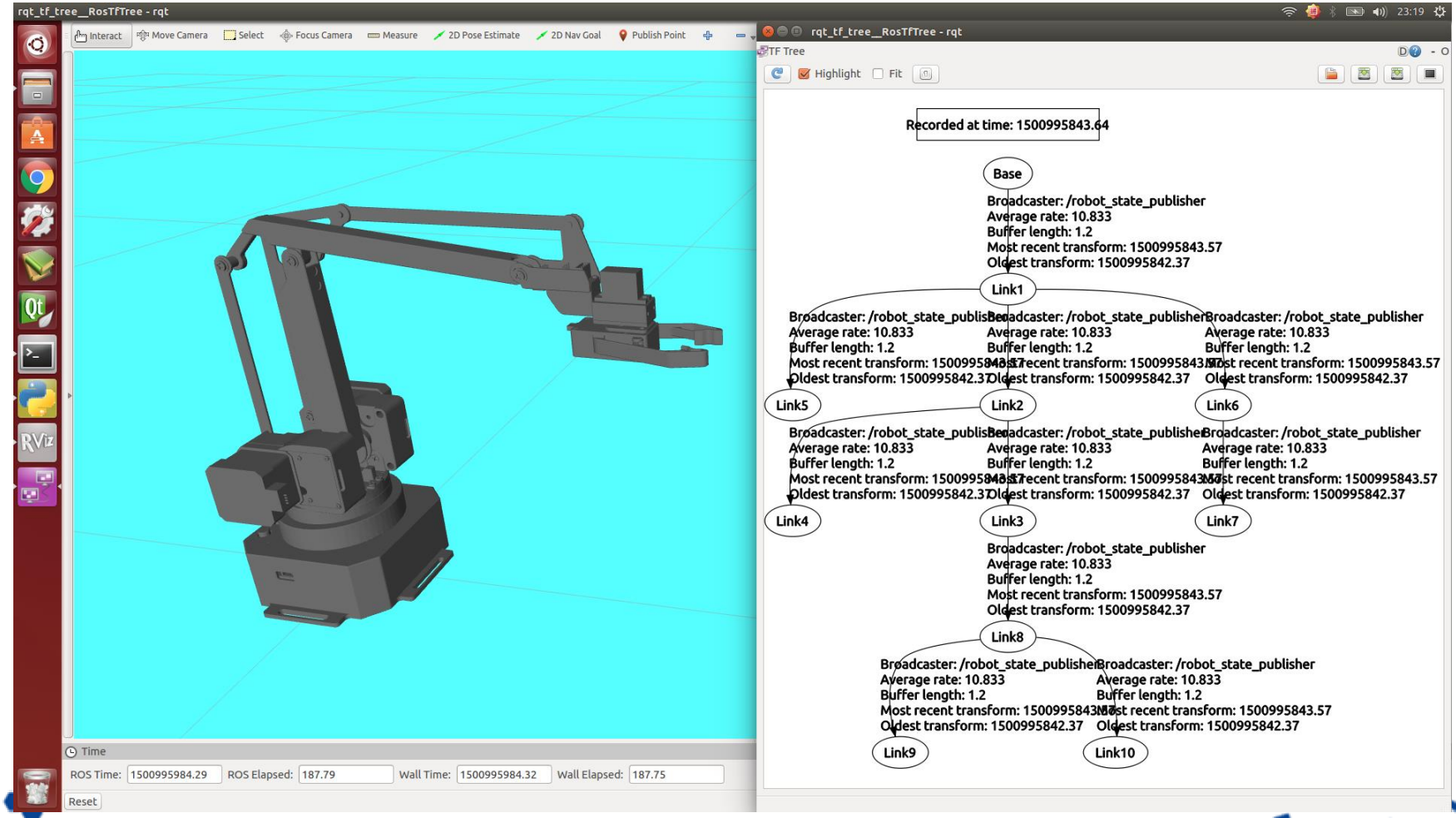
- 并联结构
  - 按照串联结构建立 URDF
  - 接收末端位姿
  - 利用逆解程序将末端位姿转换成 URDF 中每个关节角度，并利用 /joint_states 发布。



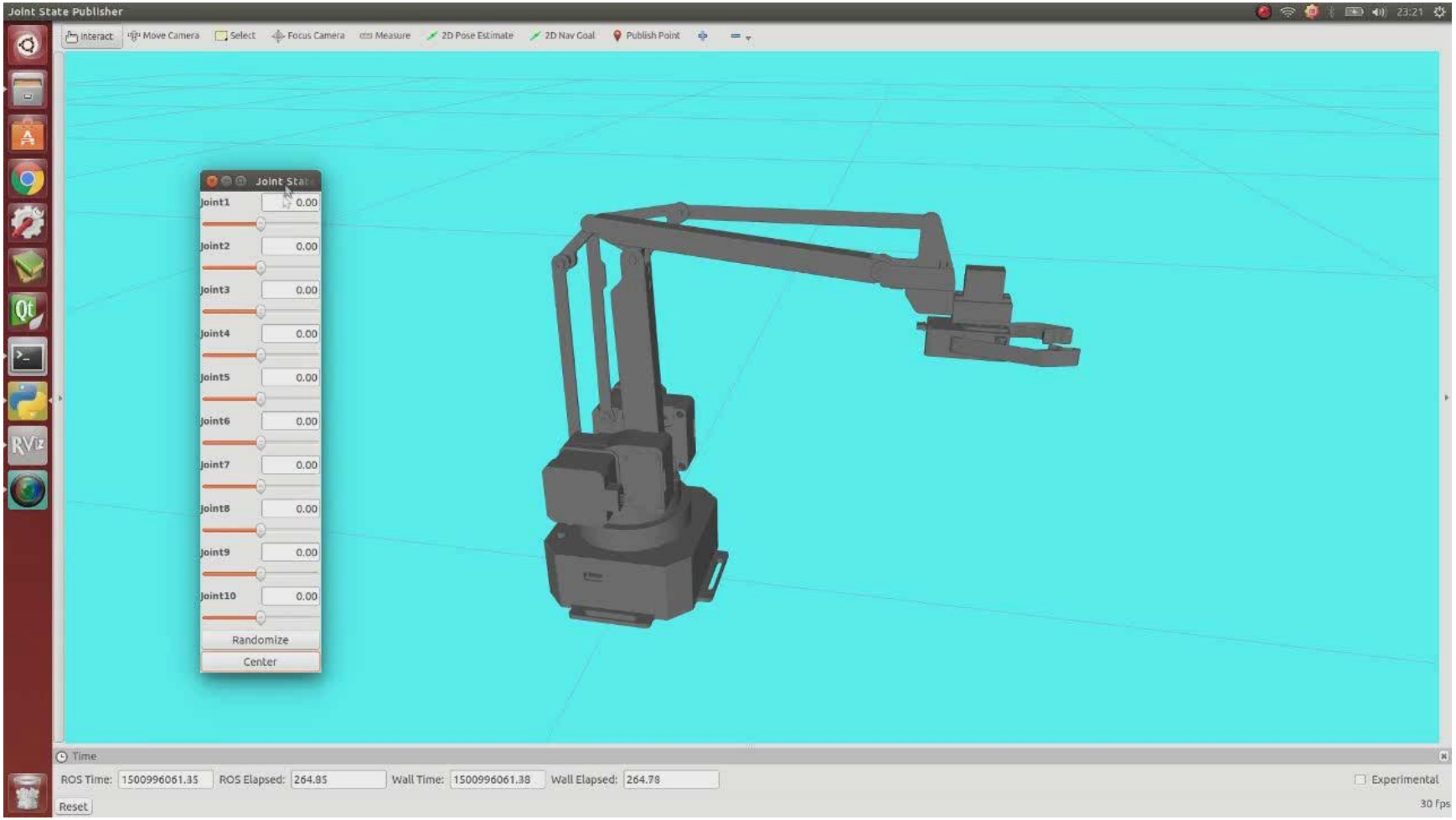
- uArm - SwiftPro

15:30 -- 17:30	桌面机械臂uArm	张哲明 UFACTORY公司 研发副总

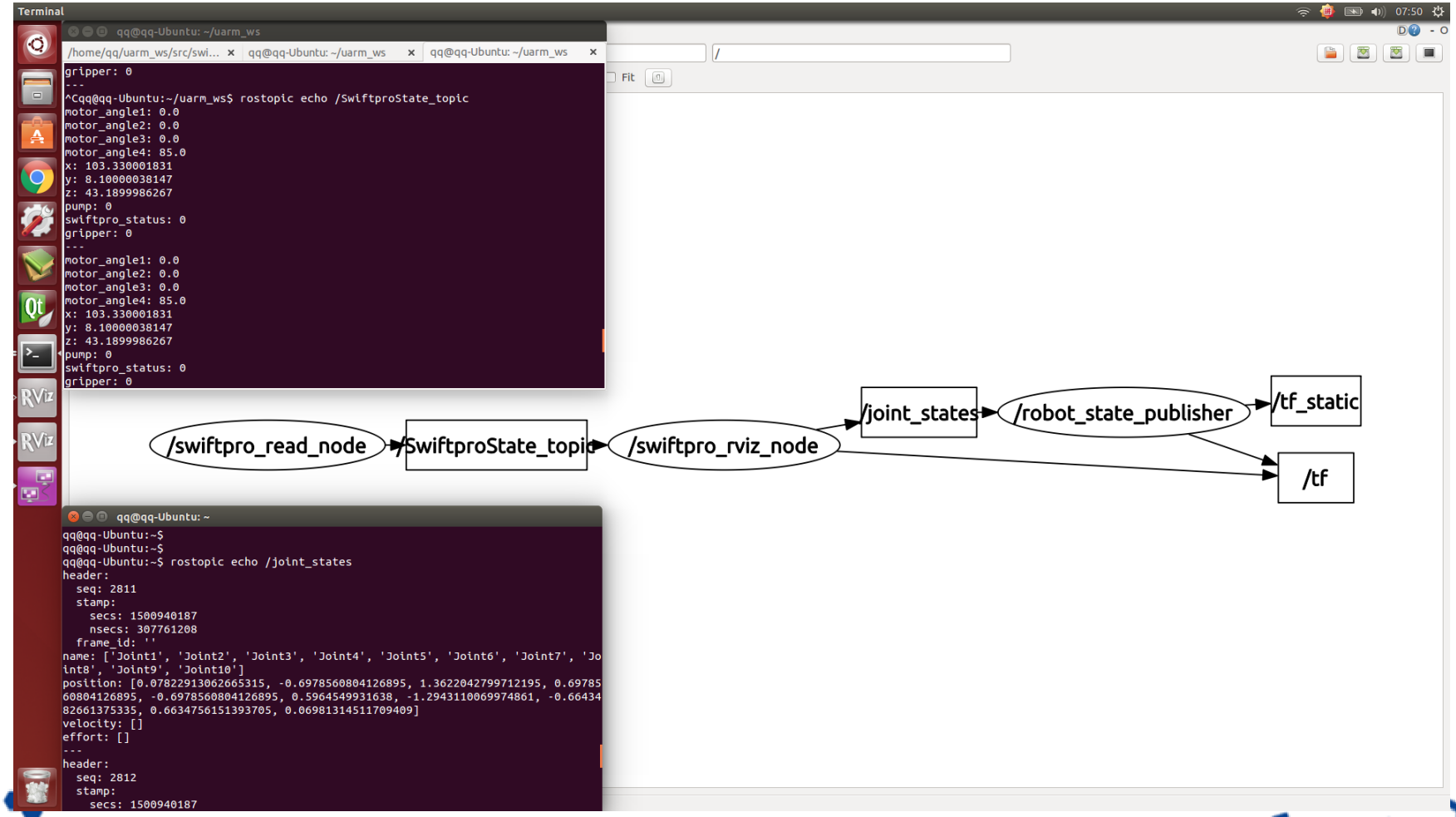
并联结构


rqt_tf_tree__RosTfTree - rqt
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Time
ROS Time: 1500995984.29
ROS Elapsed: 187.79
Wall Time: 1500995984.32
Wall Elapsed: 187.75
Reset
rqt_tf_tree__RosTfTree - rqt
TF Tree
Highlight
Fit
Recorded at time: 1500995843.64
Base
Broadcaster: /robot_state_publisher
Average rate: 10.833
Buffer length: 1.2
Most recent transform: 1500995843.57
Oldest transform: 1500995842.37
Link1
Link5
Link2
Link6
Link4
Link3
Link7
Link8
Link9
Link10

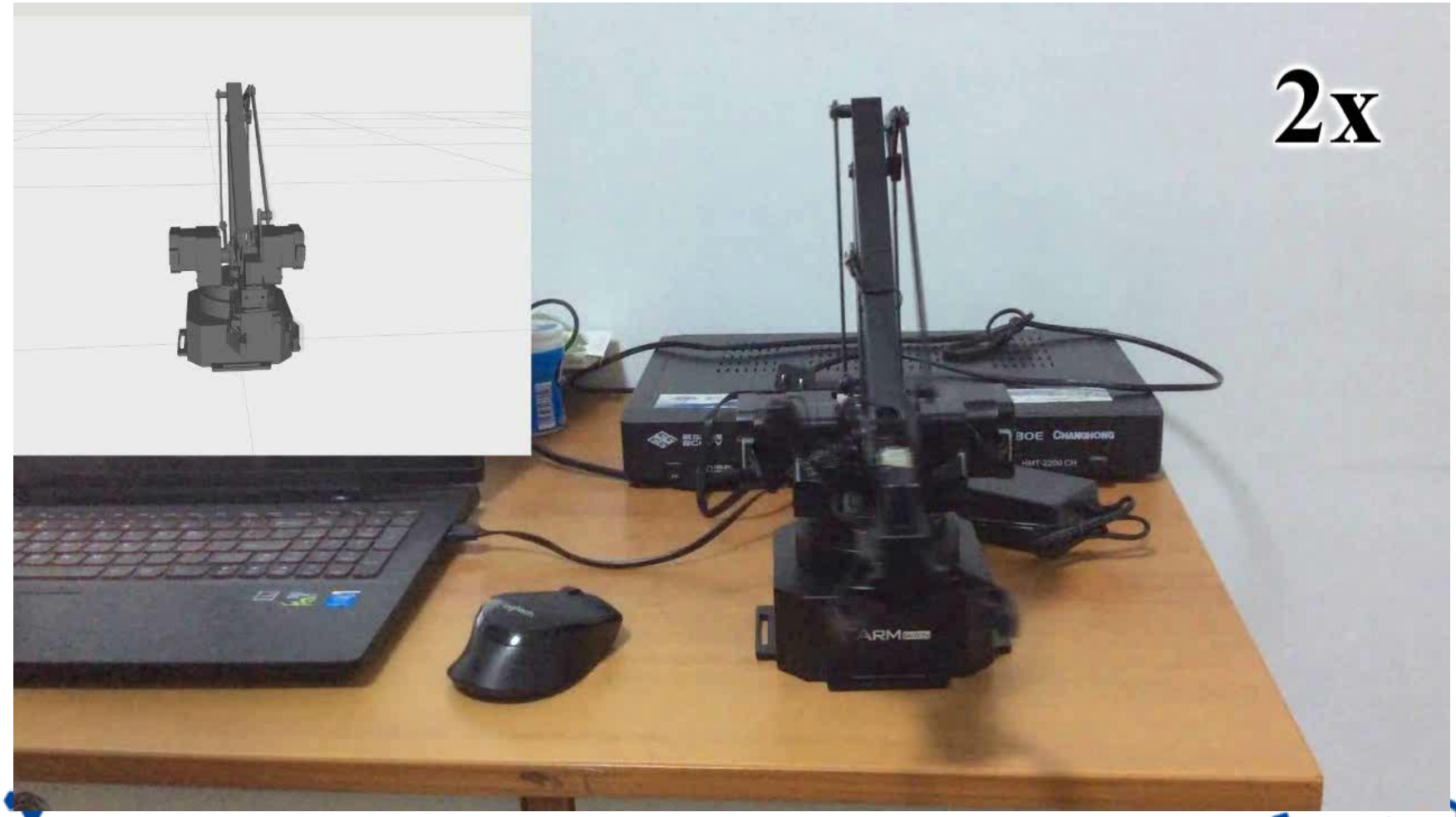
并联结构


Joint State Publisher
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Joint1 0.00
Joint2 0.00
Joint3 0.00
Joint4 0.00
Joint5 0.00
Joint6 0.00
Joint7 0.00
Joint8 0.00
Joint9 0.00
Joint10 0.00
Randomize
Center
Time
ROS Time: 1500996061.35
ROS Elapsed: 264.85
Wall Time: 1500996061.38
Wall Elapsed: 264.78
Experimental
Reset
30 fps

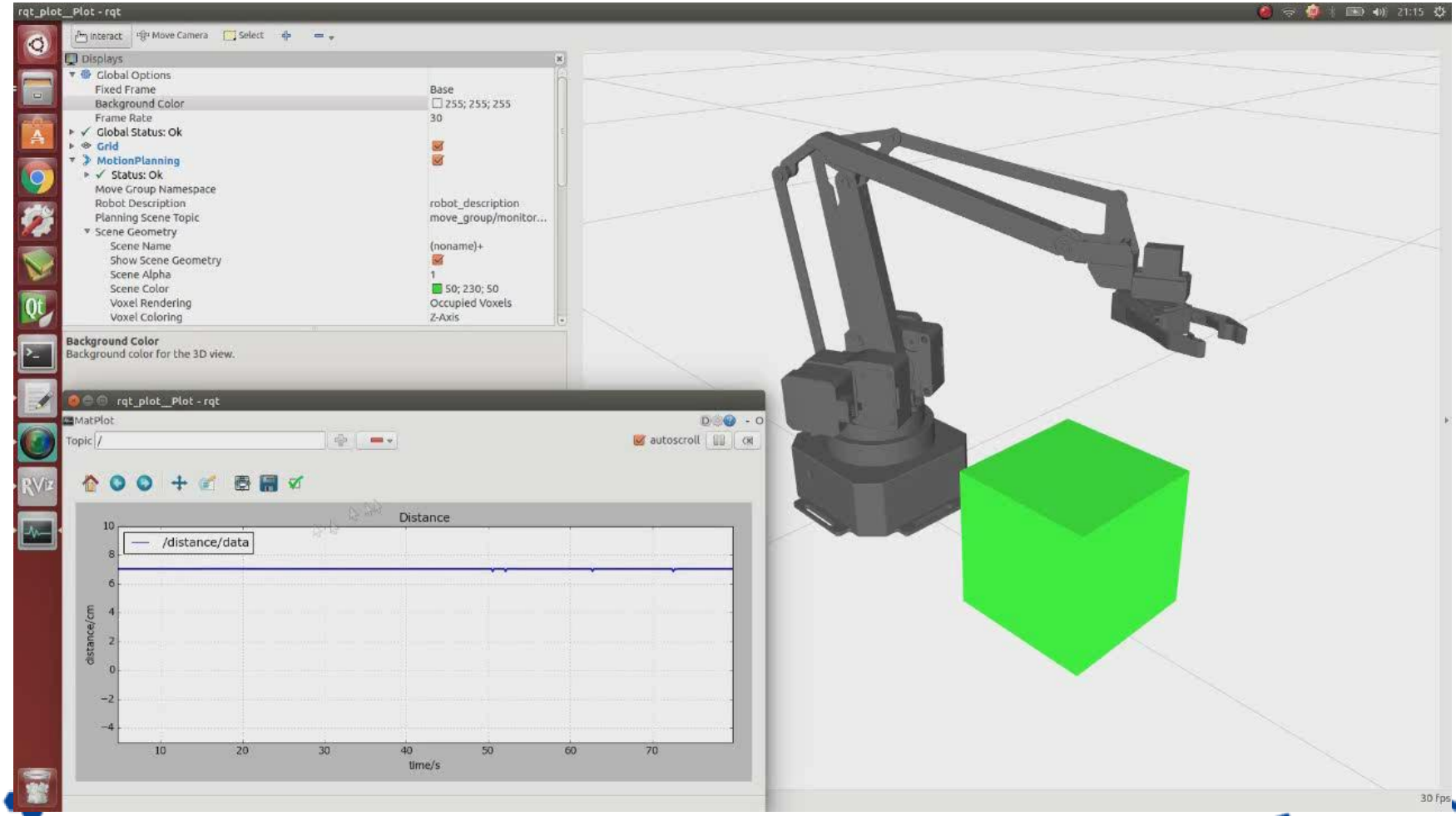
并联结构


Terminal
qq@qq-Ubuntu: ~/uarm_ws
/home/qq/uarm_ws/src/swi...
qq@qq-Ubuntu: ~/uarm_ws
qq@qq-Ubuntu: ~/uarm_ws
gripper: 0
---
^Cqq@qq-Ubuntu:~/uarm_ws$ rostopic echo /SwiftproState_topic
motor_angle1: 0.0
motor_angle2: 0.0
motor_angle3: 0.0
motor_angle4: 85.0
x: 103.330001831
y: 8.10000038147
z: 43.1899986267
pump: 0
swiftpro_status: 0
gripper: 0
---
motor_angle1: 0.0
motor_angle2: 0.0
motor_angle3: 0.0
motor_angle4: 85.0
x: 103.330001831
y: 8.10000038147
z: 43.1899986267
pump: 0
swiftpro_status: 0
gripper: 0
/swiftpro_read_node
/SwiftproState_topic
/swiftpro_rviz_node
/joint_states
/robot_state_publisher
/tf_static
/tf
qq@qq-Ubuntu: ~
qq@qq-Ubuntu:~$
qq@qq-Ubuntu:~$
qq@qq-Ubuntu:~$ rostopic echo /joint_states
header:
seq: 2811
stamp:
secs: 1500940187
nsecs: 307761208
frame_id: ''
name: ['Joint1', 'Joint2', 'Joint3', 'Joint4', 'Joint5', 'Joint6', 'Joint7', 'Jo
int8', 'Joint9', 'Joint10']
position: [0.07822913062665315, -0.6978560804126895, 1.3622042799712195, 0.69785
60804126895, -0.6978560804126895, 0.5964549931638, -1.2943110069974861, -0.66434
82661375335, 0.6634756151393705, 0.06981314511709409]
velocity: []
effort: []
---
header:
seq: 2812
stamp:
secs: 1500940187

并联结构



并联结构


rqt_plot__Plot - rqt
Interact
Move Camera
Select
Displays
Global Options
Fixed Frame
Base
Background Color
255; 255; 255
Frame Rate
30
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitor...
Scene Geometry
Scene Name
(noname)+
Show Scene Geometry
Scene Alpha
1
Scene Color
50; 230; 50
Voxel Rendering
Occupied Voxels
Voxel Coloring
Z-Axis
Background Color
Background color for the 3D view.
rqt_plot__Plot - rqt
MatPlot
Topic /
autoscroll
Distance
/distance/data
distance/cm
time/s
30 fps

并联结构

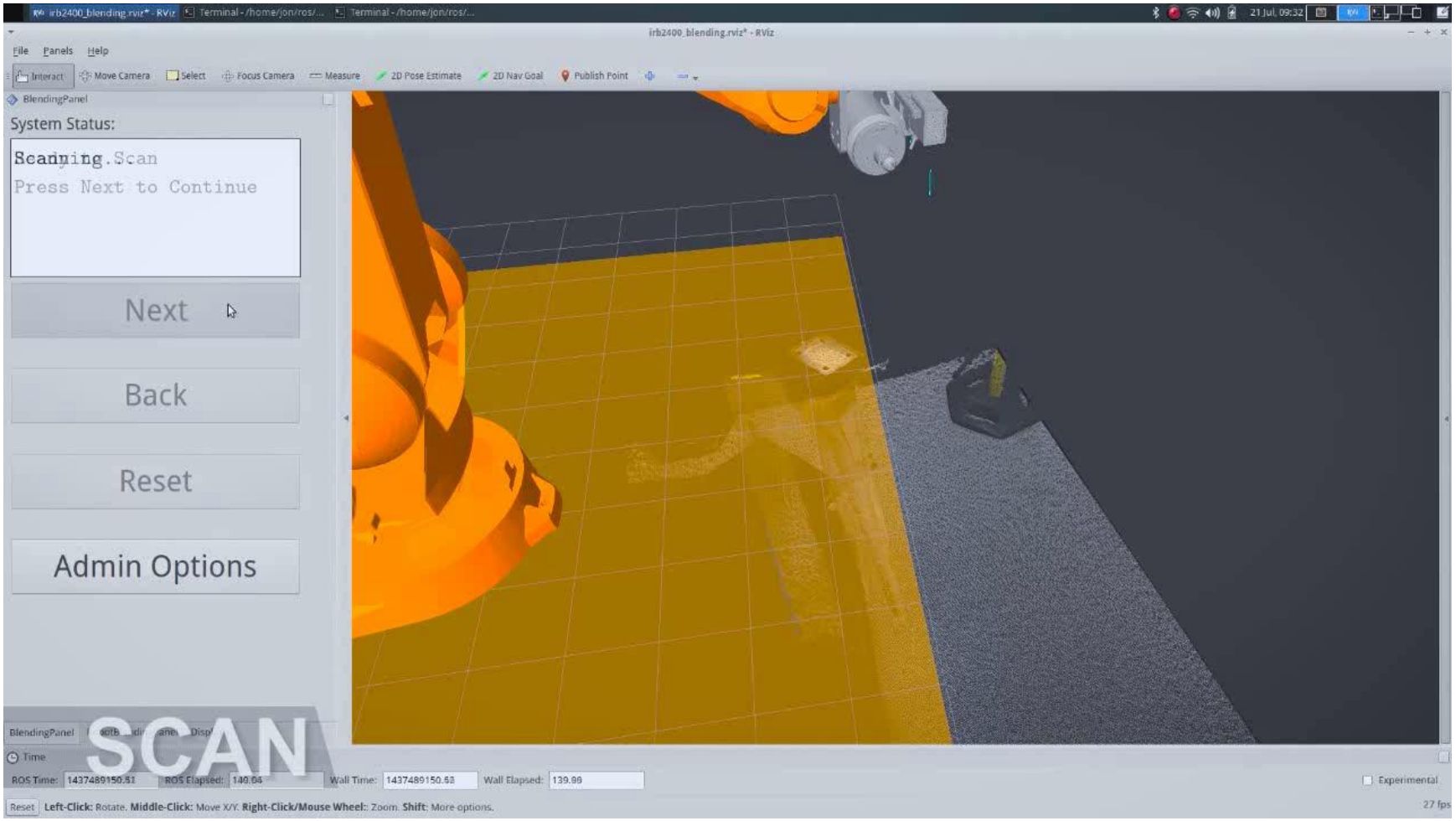
应用

- ROS-I
  - 运动学求解
  - 碰撞检测
  - 运动规划
  - …
- 应用
  - 杂乱分拣
  - 自动打磨
  - …

ROS®

Video: Robotic Product Singulation Testbed

应用

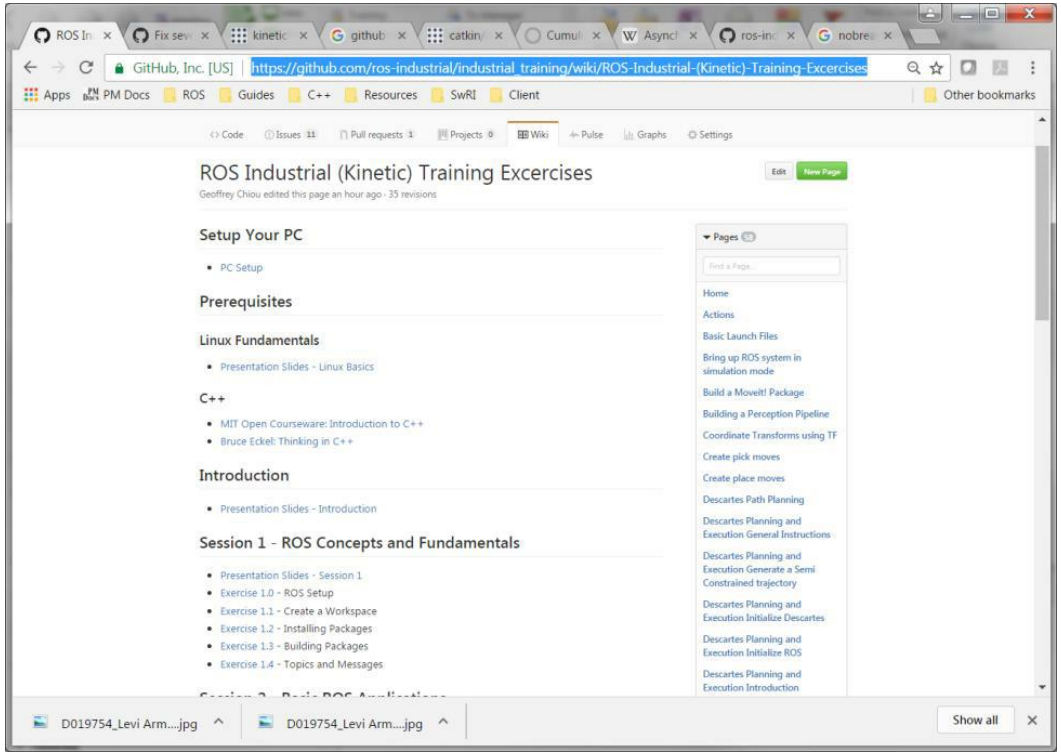


Video: [Scan-N-Plan]

应用

Video: ROS-I Five Years Montage

- 如何学习
  - ROS-I 官网教程
    - ROS-I Training Class



谢谢~