上海科技大学
ShanghaiTech University

# CS283: Robotics Fall 2019: Robot Arms

Sören Schwertfeger / 师泽仁

ShanghaiTech University

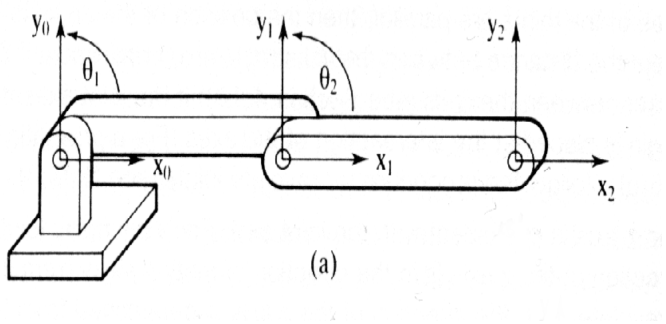
# ROBOT ARMS

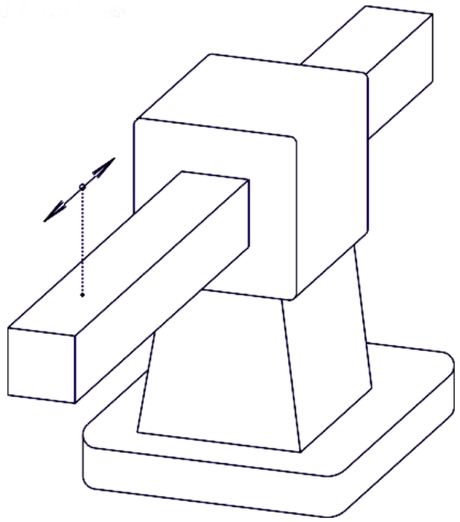
# Robot Arm

- Consists of Joints and Links …
- and a Base and a Tool (or End-Effector or Tip)

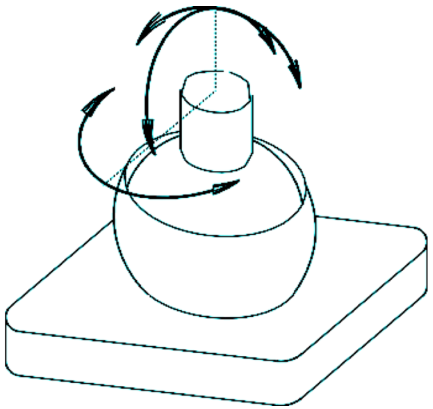
# Joints

- Revolute Joint: 1DOF



- Prismatic Joint/ Linear Joint: 1DOF



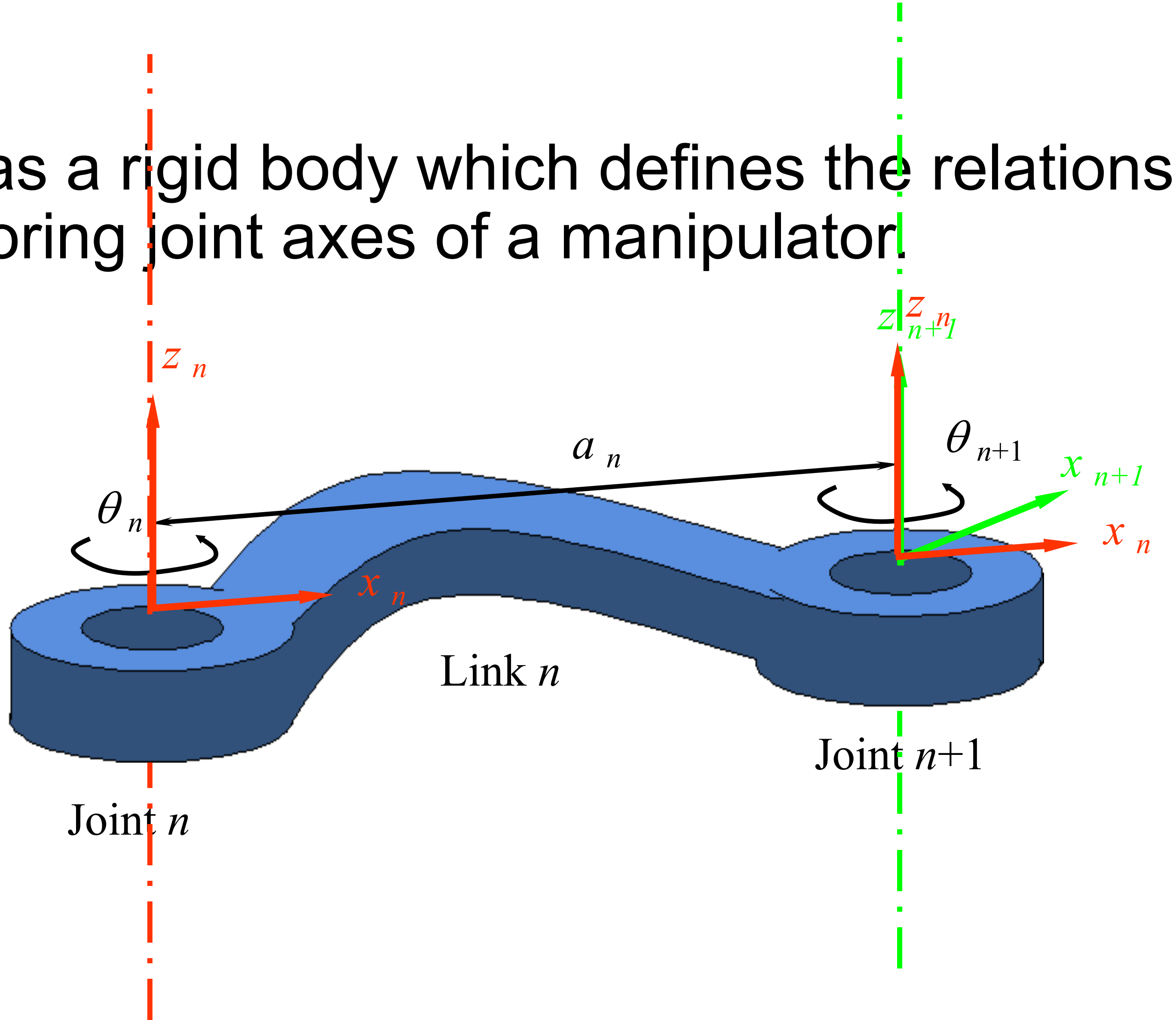
- Spherical Joint: 3DOF

# Note on Joints

- Without loss of generality, we will consider only manipulators which have joints with a single degree of freedom.
- A joint having n degrees of freedom can be modeled as n joints of one degree of freedom connected with n-1 links of zero length.

# Link

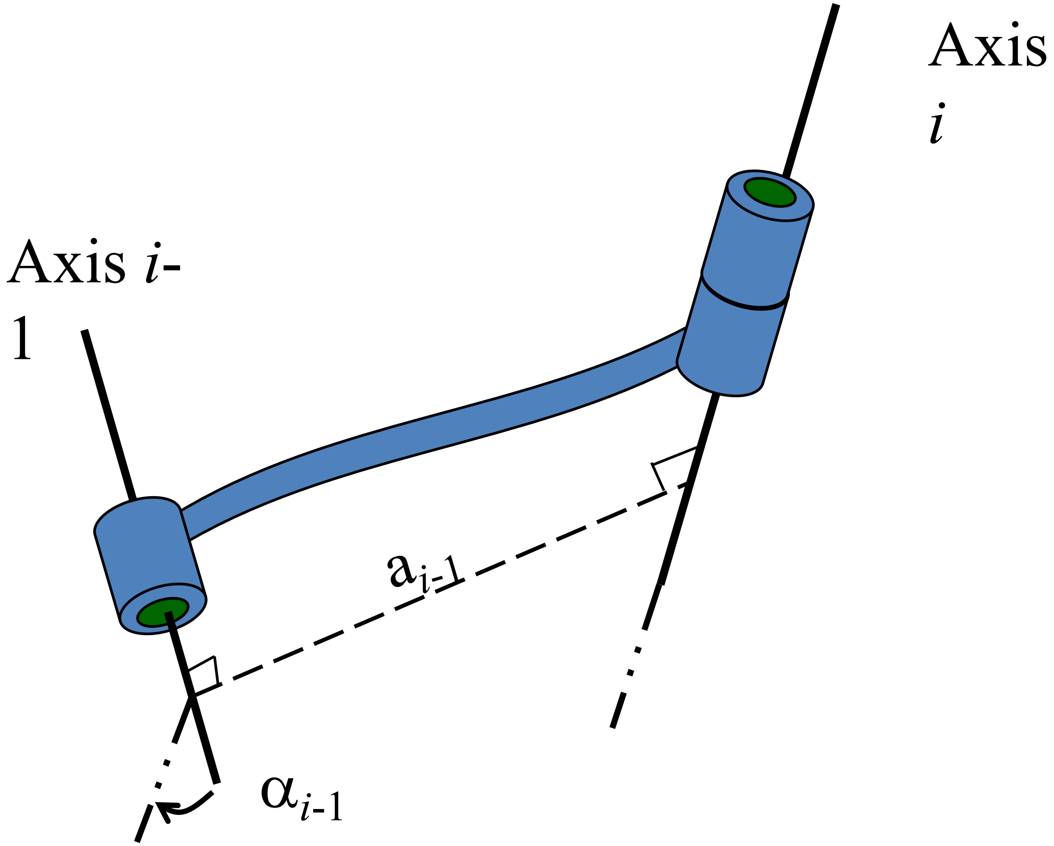
- A link is considered as a rigid body which defines the relationship between two neighboring joint axes of a manipulator.



# The Kinematics Function of a Link

- The kinematics function of a link is to maintain a fixed relationship between the two joint axes it supports.

- This relationship can be described with two parameters: the link length a, the link twist a



# Link Length

- Is measured along a line which is mutually perpendicular to both axes.

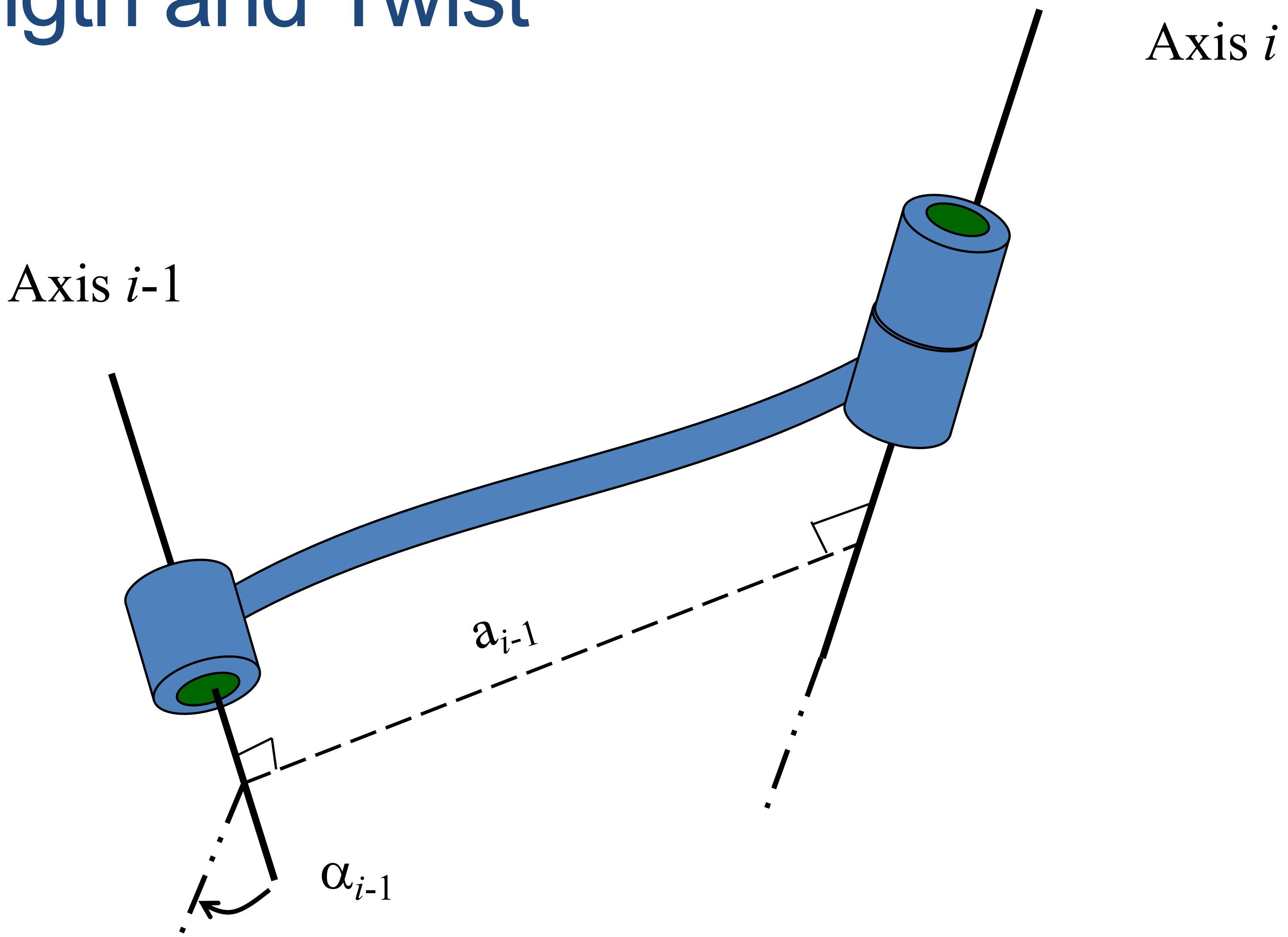
- The mutually perpendicular always exists and is unique except when both axes are parallel.

# Link Twist

- Project both axes i-1 and i onto the plane whose normal is the mutually perpendicular line, and measure the angle between them

- Right-hand coordinate system

# Link Length and Twist



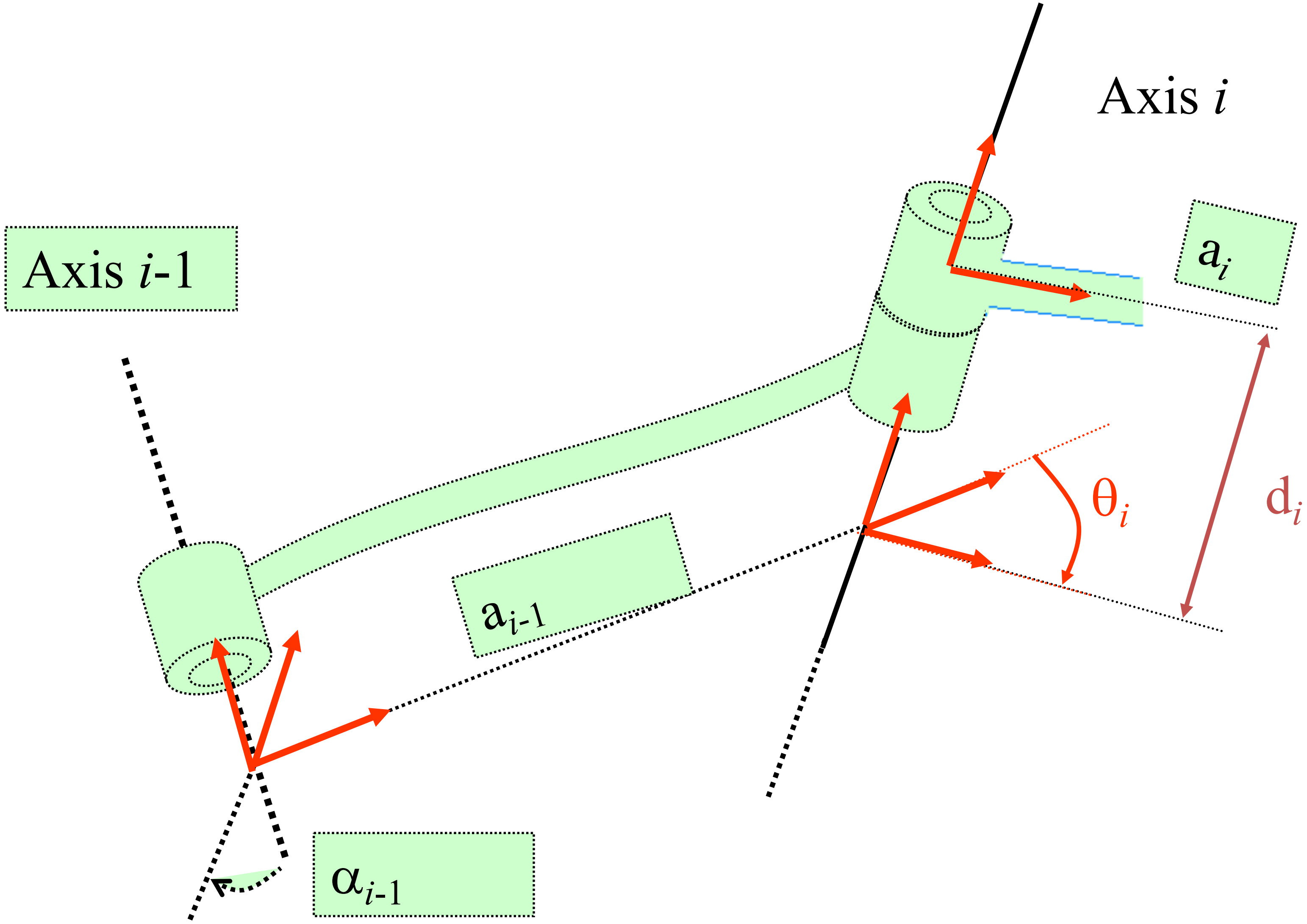
# Joint Parameters
# (the Denavit-Hartenberg Link Parameters)

A joint axis is established at the connection of two links. This joint will have two normals connected to it one for each of the links.

- The relative position of two links is called *link offset* $d_n$ which is the distance between the links (the displacement, along the joint axes between the links).

- The *joint angle* $\theta_n$ between the normals is measured in a plane normal to the joint axis.

# Link and Joint Parameters



# Link and Joint Parameters

4 parameters are associated with each link. You can align the two axis using these parameters.

- Link parameters:

$a_n$ the length of the link.

$\alpha_n$ the twist angle between the joint axes.

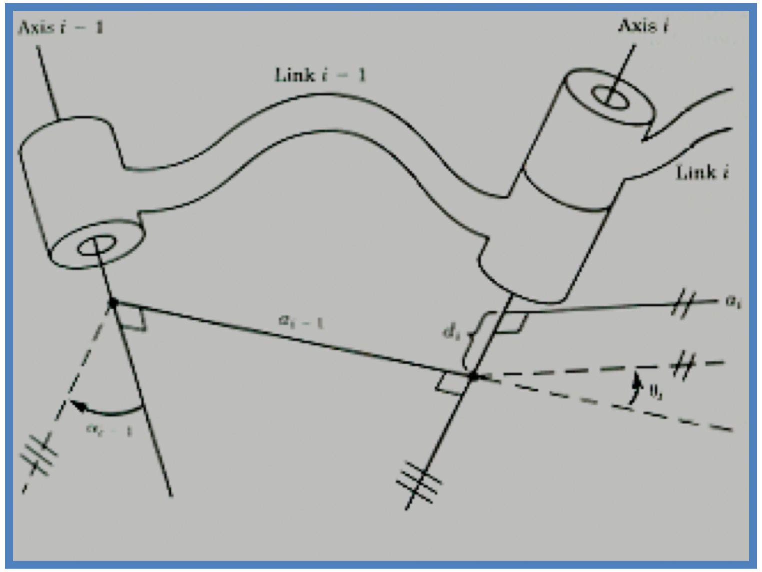
- Joint parameters:

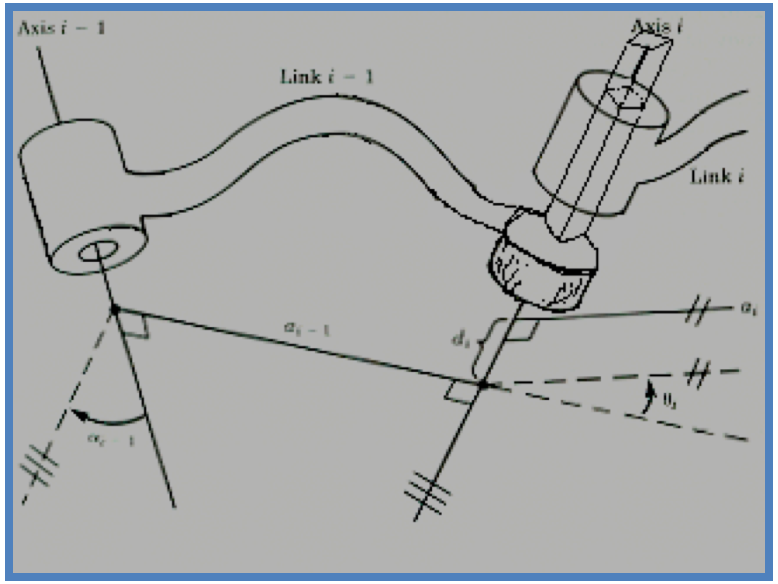
$\theta_n$ the angle between the links.

$d_n$ the distance between the links

# Link Connection Description:

For Revolute Joints: a, $\alpha$, and d. are all fixed, then "$\theta_i$" is the. Joint Variable.



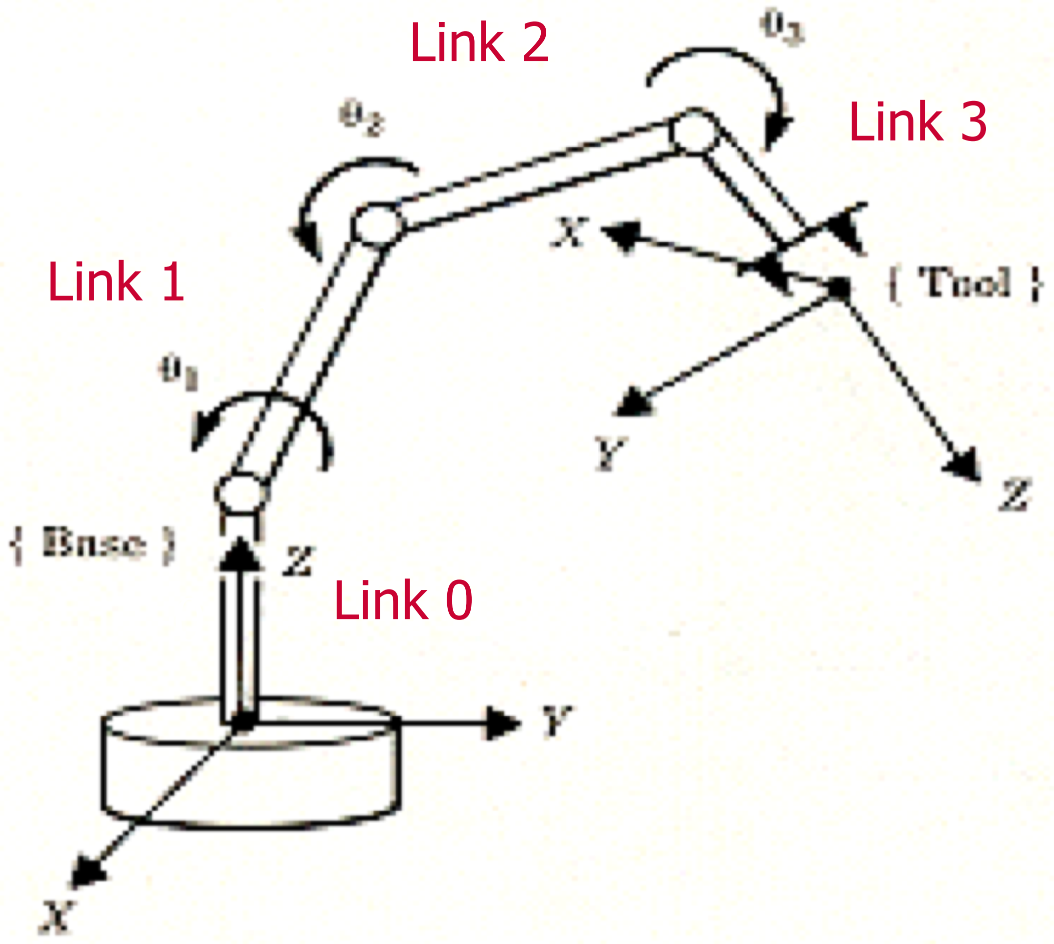




For Prismatic Joints: a, $\alpha$, and $\theta$. are all fixed, then "$d_i$" is the. Joint Variable.

These four parameters: (Link-Length $a_{i-1}$), (Link-Twist $\alpha_{i-1}$), (Link-Offset $d_i$), (Joint-Angle $\theta_i$) are known as the *Denavit-Hartenberg Link Parameters*.

# Links Numbering Convention

| | |
|---|---|
| **Base of the arm:** | **Link-0** |
| **1st moving link:** | **Link-1** |
| **.** | **.** |
| **.** | **.** |
| **.** | **.** |
| **Last moving link:** | **Link-n** |




A 3-DOF Manipulator Arm

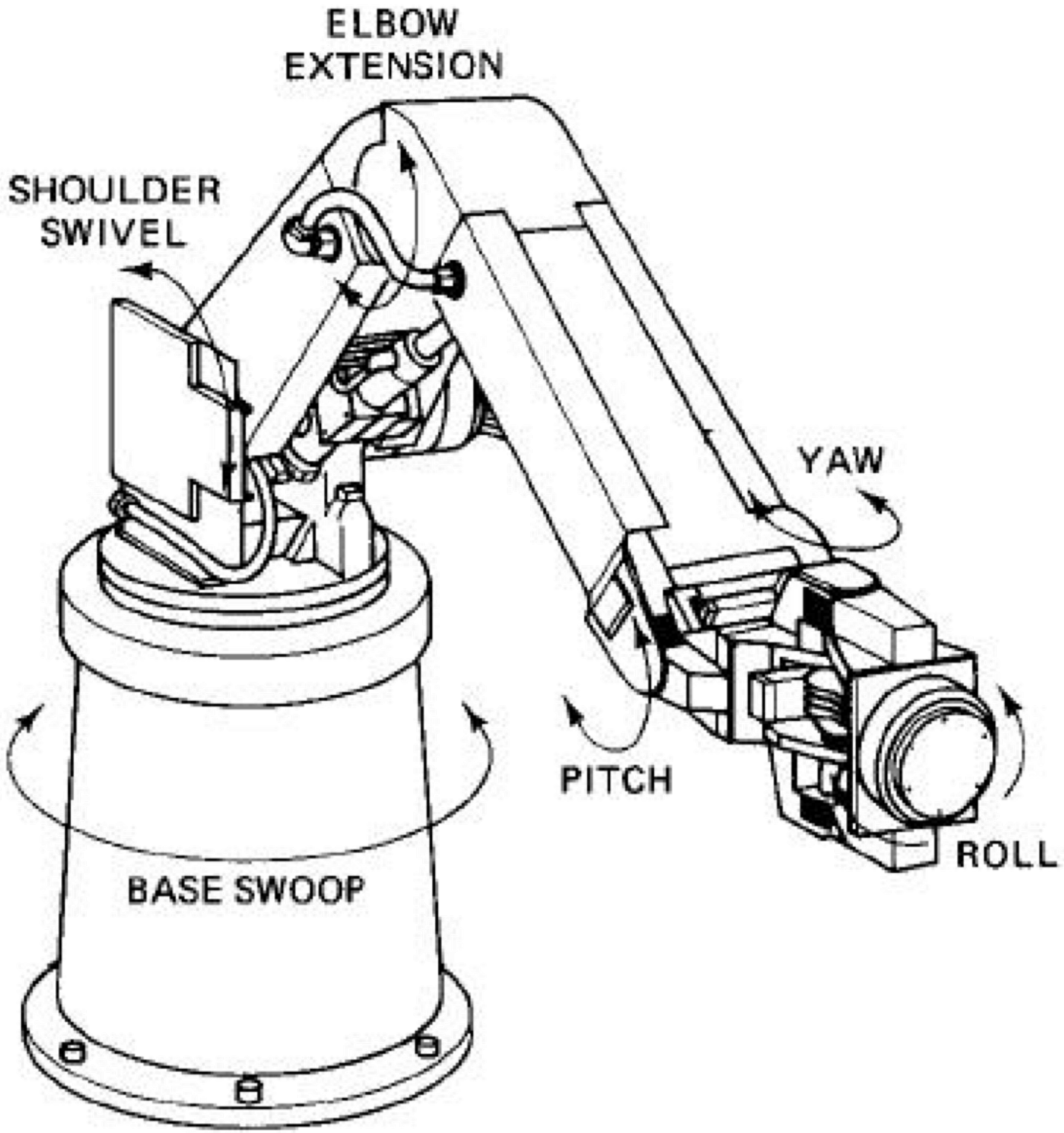
# First and Last Links in the Chain

- $a_{0}= \alpha_{n}=0.0$
- $\alpha_{0}= \alpha_{n}=0.0$
- *If joint 1 is revolute:* $d_{0}= 0$ *and* $\theta_1$ *is arbitrary*
- *If joint 1 is prismatic:* $d_{0}=$ *arbitrary and* $\theta_1 = 0$

# Robot Specifications

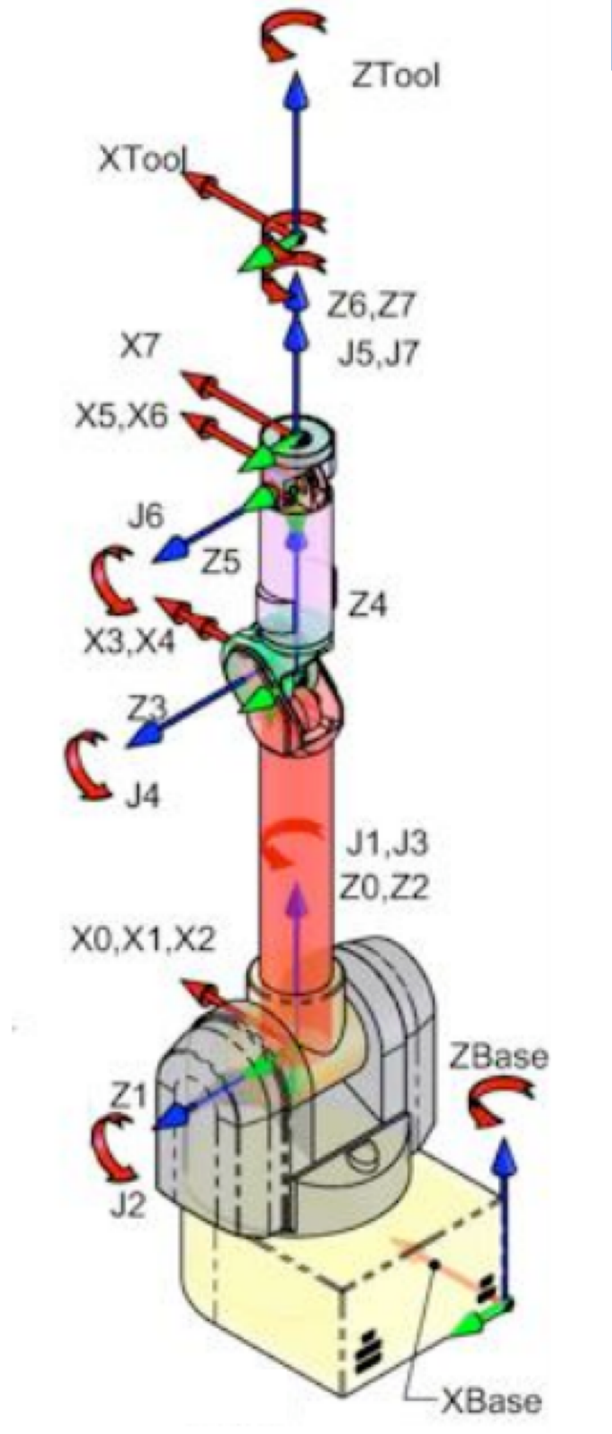
**Number of axes**

- Major axes, (1-3) => position the wrist
- Minor axes, (4-6) => orient the tool
- Redundant, (7-n) => reaching around obstacles, avoiding undesirable configuration



# Frames

- Choose the base and tool coordinate frame
  - Make your life easy!
- Several conventions
  - Denavit Hartenberg (DH), modified DH, Hayati, etc.



# KINEMATICS

# Kinematics

## Forward Kinematics (angles to position)

(it is straight-forward -> easy)

What you are given: The length of each link
The angle of each joint

What you can find: The position of any point (i.e. it's (x, y, z) coordinates)

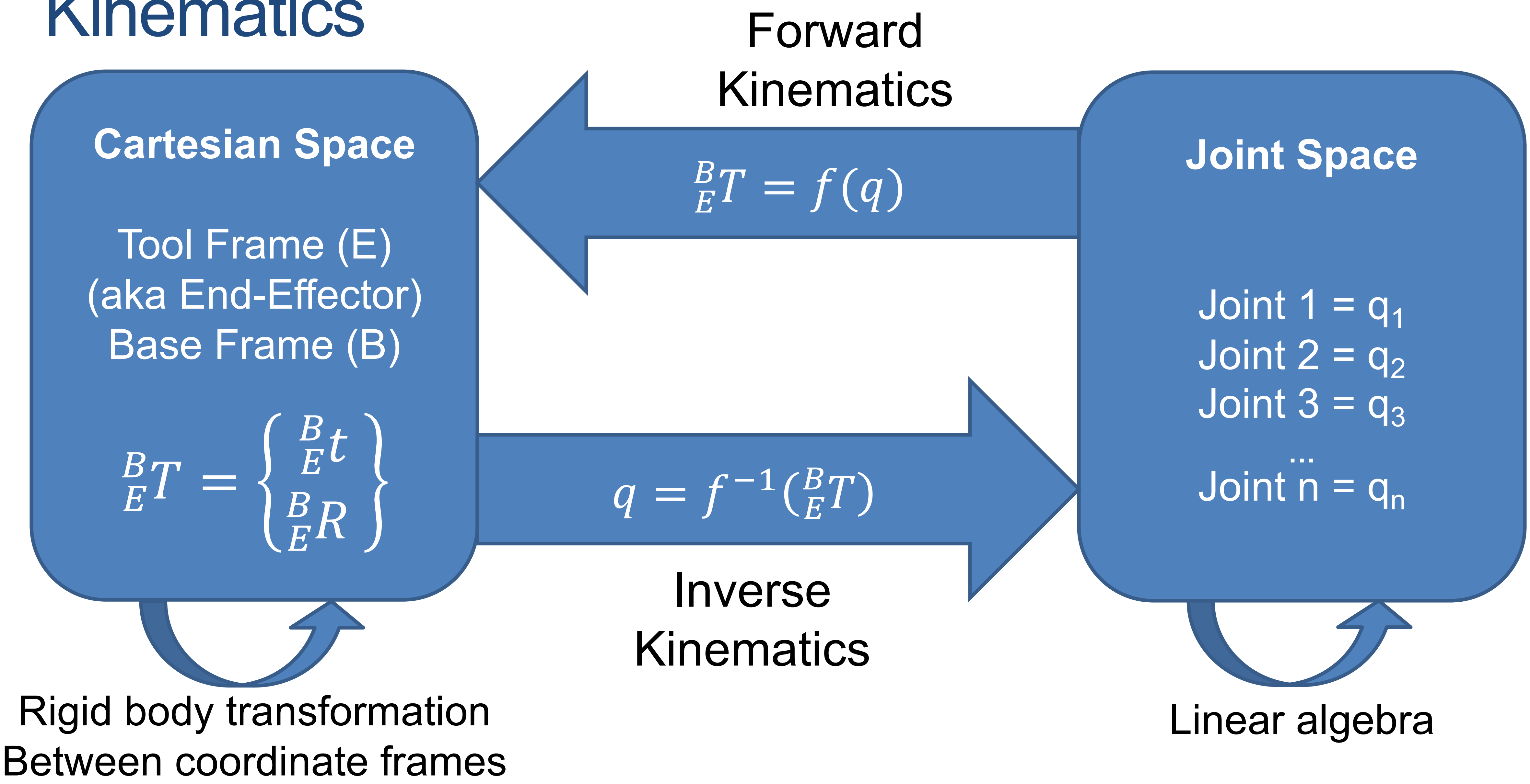
## Inverse Kinematics (position to angles)

(more difficult)

What you are given: The length of each link
The position of some point on the robot

What you can find: The angles of each joint needed to obtain that position

# Kinematics


Forward
Kinematics
$^B_E T = f(q)$
Cartesian Space
Tool Frame (E)
(aka End-Effector)
Base Frame (B)
$^B_E T = \left\{ \begin{matrix} ^B_E t \\ ^B_E R \end{matrix} \right\}$
Joint Space
Joint 1 = $q_1$
Joint 2 = $q_2$
Joint 3 = $q_3$
...
Joint n = $q_n$
$q = f^{-1}(^B_E T)$
Inverse
Kinematics
Rigid body transformation
Between coordinate frames
Linear algebra

# Kinematics: Velocities

**Cartesian Space**
Tool Frame (E)
(aka End-Effector)
Base Frame (B)

$${}^{B}_{E}V = \begin{Bmatrix} {}^{B}_{E}v \\ {}^{B}_{E}w \end{Bmatrix}$$

v: linear velocity
w: angular velocity

Jacobian

$${}^{B}_{E}V = J(q)\dot{q}$$

$$\dot{q} = J^{-1}(q)\ {}^{B}_{E}V$$

Inverse Jacobian

**Joint Space**

Joint 1 = $\dot{q}_1$
Joint 2 = $\dot{q}_2$
Joint 3 = $\dot{q}_3$
...
Joint n = $\dot{q}_n$

Rigid body transformation
Between coordinate frames

Linear algebra

# INVERSE KINEMATICS (IK)

# Inverse Kinematics (IK)

- Given end effector position, compute required joint angles
- In simple case, analytic solution exists
  - Use trig, geometry, and algebra to solve
- Generally (more DOF) difficult
  - Use Newton's method
  - Often more than one solution exist!

- Analytic solution of 2-link inverse kinematics

$$x^2 + y^2 = a_1^2 + a_2^2 - 2a_1a_2\cos(\pi - \theta_2)$$

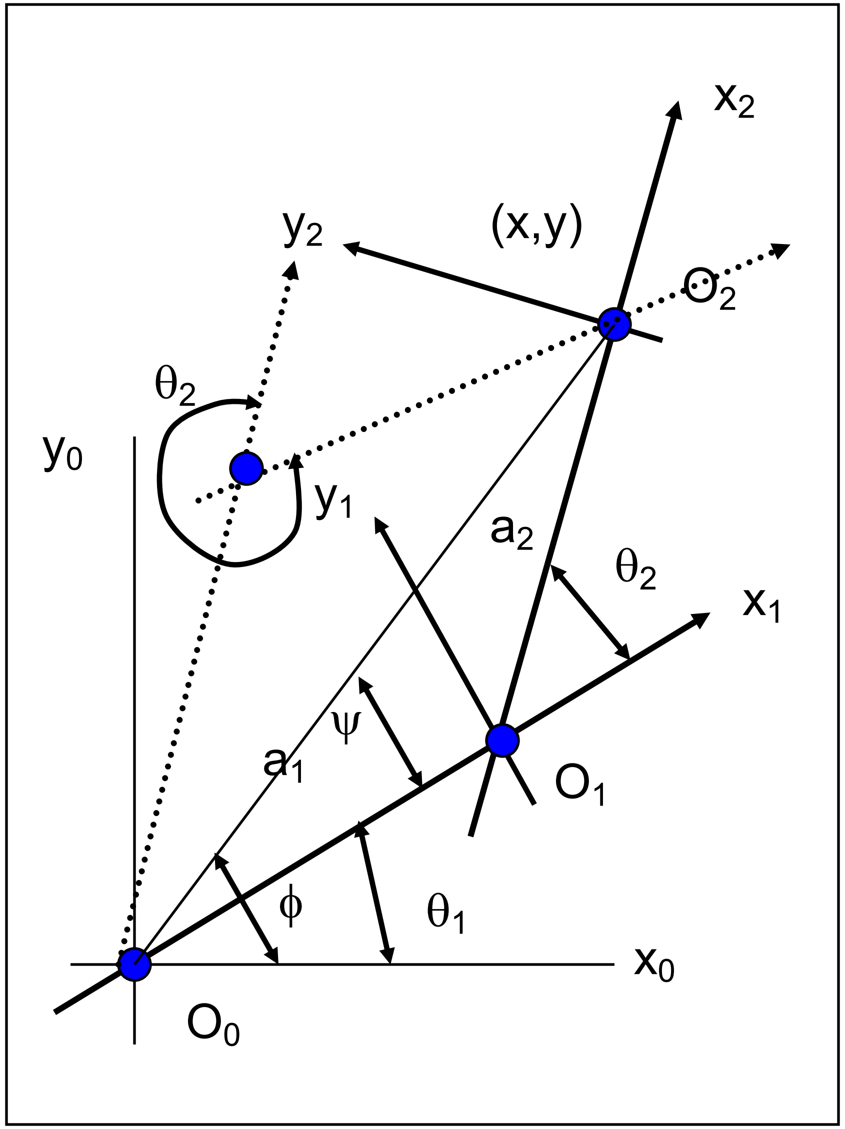
$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

for greater accuracy

$$\tan^2\frac{\theta_2}{2} = \frac{1-\cos\theta}{1+\cos\theta} = \frac{2a_1a_2 - x^2 - y^2 + a_1^2 + a_2^2}{2a_1a_2 + x^2 + y^2 - a_1^2 - a_2^2}$$

$$= \frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}$$

$$\theta_2 = \pm 2\tan^{-1}\sqrt{\frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}}$$




- Two solutions: elbow up & elbow down

# Iterative IK Solutions

- Frequently analytic solution is infeasible
- Use Jacobian
- Derivative of function output relative to each of its inputs
- If y is function of three inputs and one output

$$y = f(x_1, x_2, x_3)$$

$$\delta y = \frac{\delta f}{\partial x_1} \cdot \delta x_1 + \frac{\delta f}{\partial x_2} \cdot \delta x_2 + \frac{\delta f}{\partial x_3} \cdot \delta x_3$$

- Represent Jacobian J(X) as a 1x3 matrix of partial derivatives

# Jacobian

- In another situation,
  end effector has 6 DOFs and
  robotic arm has 6 DOFs

- $f(x_1, \ldots, x_6) = (x, y, z, r, p, y)$

- Therefore J(X) = 6x6 matrix

$$\begin{bmatrix} \frac{\partial f_x}{\partial x_1} & \frac{\partial f_y}{\partial x_1} & \frac{\partial f_z}{\partial x_1} & \frac{\partial f_r}{\partial x_1} & \frac{\partial f_p}{\partial x_1} & \frac{\partial f_y}{\partial x_1} \\ \frac{\partial f_x}{\partial x_2} & & & & & \\ \frac{\partial f_x}{\partial x_3} & & & & & \\ \frac{\partial f_x}{\partial x_4} & & & & & \\ \frac{\partial f_x}{\partial x_5} & & & & & \\ \frac{\partial f_x}{\partial x_6} & & & & & \end{bmatrix}$$

# Jacobian

- Relates velocities in parameter space to velocities of outputs

$$\dot{Y} = J(X) \cdot \dot{X}$$

- If we know $Y_{current}$ and $Y_{desired}$, then we subtract to compute $Y_{dot}$
- Invert Jacobian and solve for $X_{dot}$

# ADMIN

# Presentation

- Choose one paper from ICRA or IROS which is relevant to your project!
  - ICRA: https://ieeexplore.ieee.org/xpl/conhome/1000639/all-proceedings
  - IROS: https://ieeexplore.ieee.org/xpl/conhome/1000393/all-proceedings
  - Only full papers (6 or more pages) are allowed; no workshop papers

- Present the paper as if it were your own work!
- Front page: Name of the Paper; Full citation of the paper; **Your name in Pinyin; Your email address**
- Last slide: **ONE** slide about how this paper is relevant to your project.
- Your presentation has to be professional – not cute…
- Submit pdf or ppt to the paper repository till Monday, Oct 21 22:00 to repo! Late submissions (or if you come with the ppt/ pdf to the presentation time) will receive a flat 33% loss of points!
- Presentations in 4 slots – most likely in the evenings of Oct 22-25.
- 10 minute presentation plus 1 minute project relevance plus 3 minutes questions
  - Do not rush your presentation! Better present less items more slowly!
  - 10 minute presentation => 5 – max. 10 slides
  - Maybe have a slide towards the end that you can skip if you run out of time.
  - Give a test presentation to your friends beforehand!
- Finish early for practicing – don't learn by heart.

# Scoring of the Presentation

- 10 %: Your basic understanding/ knowledge about the paper you present
- 20 %: Presentation timing (plus or minus one minute is ok) – no rushing – good speed!
- 10 %: Correct written English in presentation:
  - No complete sentences, no grammatical or spelling mistakes
- 10 %: Good structure of presentation:
  - Depends on the type of paper, how much time you have, how long you need to present the main achievement.
  - For example: outline, introduction/ motivation, **problem statement**, state of the art, **approach**, experiments, **results**, **conclusion**, outlook
- 20 %: Clarity of written presentation
- 10 %: Good presentation style:
  - Interact with audience: look at the whole room (not just your slides, notes, or the back of the room)
  - Present the paper – do not read (or repeat the learned) speech from a prepared text
  - Use the presentation as visual aid – not as your tele-prompter to read from
  - Move your body – do not stand frozen at one place
- 10 %: Answering the questions
  - Questions have to be asked and answered in English – Chinese can be used for clarification
- 10 %: Asking questions to other students!
- **Not** scored: Your English skill

# Project Proposal (1)

- **Title:** Find a nice, catchy title for your project
- **Abstract**: A short abstract/ summary what the project is about
- **Introduction**: general description & Motivation
- **State of the Art**: Literature & open-source-ROS packages
- Per team member:
  - present and cite three papers with just three or four sentences
  - present in more detail one further paper relevant to your project. Describe it with at least 1/3rd of a page.
  - present in detail one open source ROS package relevant to your project. At least 1/3rd of a page
  - => about one page per team member – => 3 pages for 3 person team
- 

# Project Proposal (2)

- **System Description**
- **System Evaluation**: Describe how you want to test your system.
  - Experiments & how to measure their success
- **Work Plan**: Define some mile stones.
  - Possible phases: Algorithm design, implementation, testing, evaluation, documentation – some of those things can also happen in a loop (iteration).
  - Deliverables of Project:
    - Proposal (this document)
    - Mid-term report
    - Final demo
    - Final Report
    - Website
- **Conclusions**: Short summary and conclusions

# Project Proposal (3)

- Important dates:
- **Oct 17st, 22:00: due date for the proposal**
- **December 30th – Jan 08th (tbd): due date for the final report.**

- Parts of proposal go into the final project report.
- Please don’t forget to take **pictures and videos** when testing your system!

- In English! Using LaTeX!
- Put sources and PDF in git.

- **Additional task:** In glit/ gitlab: “Readme.txt” with:
- Team Name and Members; email addresses
- Documentation and how to’s regarding your project.

# PLANNING

# Kinematic Problems for Manipulation

- Reliably position the tip - go from one position to another position
- Don't hit anything, avoid obstacles
- Make smooth motions
  - at reasonable speeds and
  - at reasonable accelerations
- Adjust to changing conditions -
  - i.e. when something is picked up *respond to the change in weight*

# Planning Problem

- (Arm) Pose: Set of join values
- (Arm) Trajectory:
  - Given a start pose and an end pose
  - A list of intermediate poses
  - That should be reached one after the other
  - With associated (desired) velocities and accelerations (maxima)
    - Without time (without velocity and acceleration): path! So:
    - Path: poses;   Trajectory: poses with speeds (and maybe accelerations)
- Constrains:
  - Don’t collide with yourself
  - Don’t collide with anything else (except: fingers with the object to manipulate!!!)
  - Additional possible constrains:
    - Maximum joint velocities or accelerations
    - Keep global orientation of a joint (often end-effector) within certain boundaries

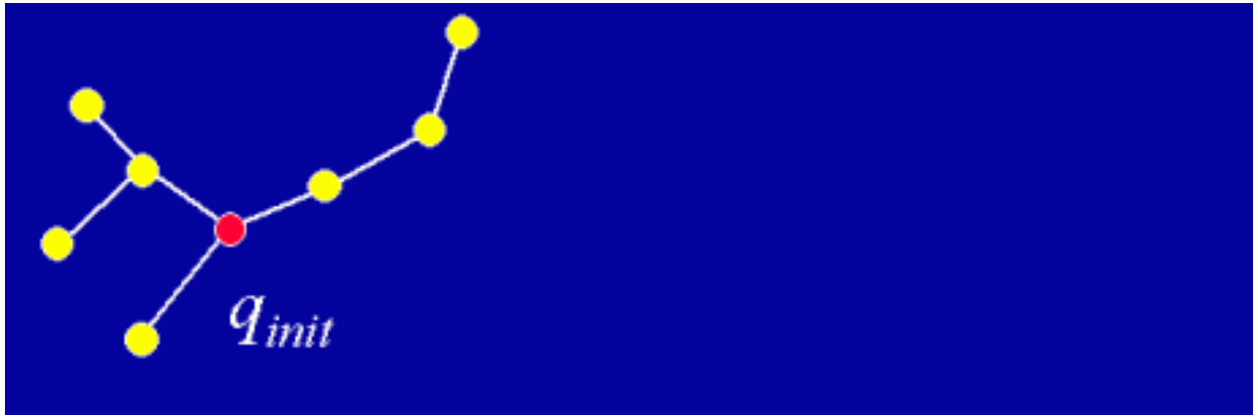
# Planning Problem cont.

- Often the goal specified in Cartesian space (not joint space)
- => use IK to get joint space
- => often multiple (even infinitely many) solutions
  - Which one select for planning?
  - Plan for several solutions and select best!?

# RRT

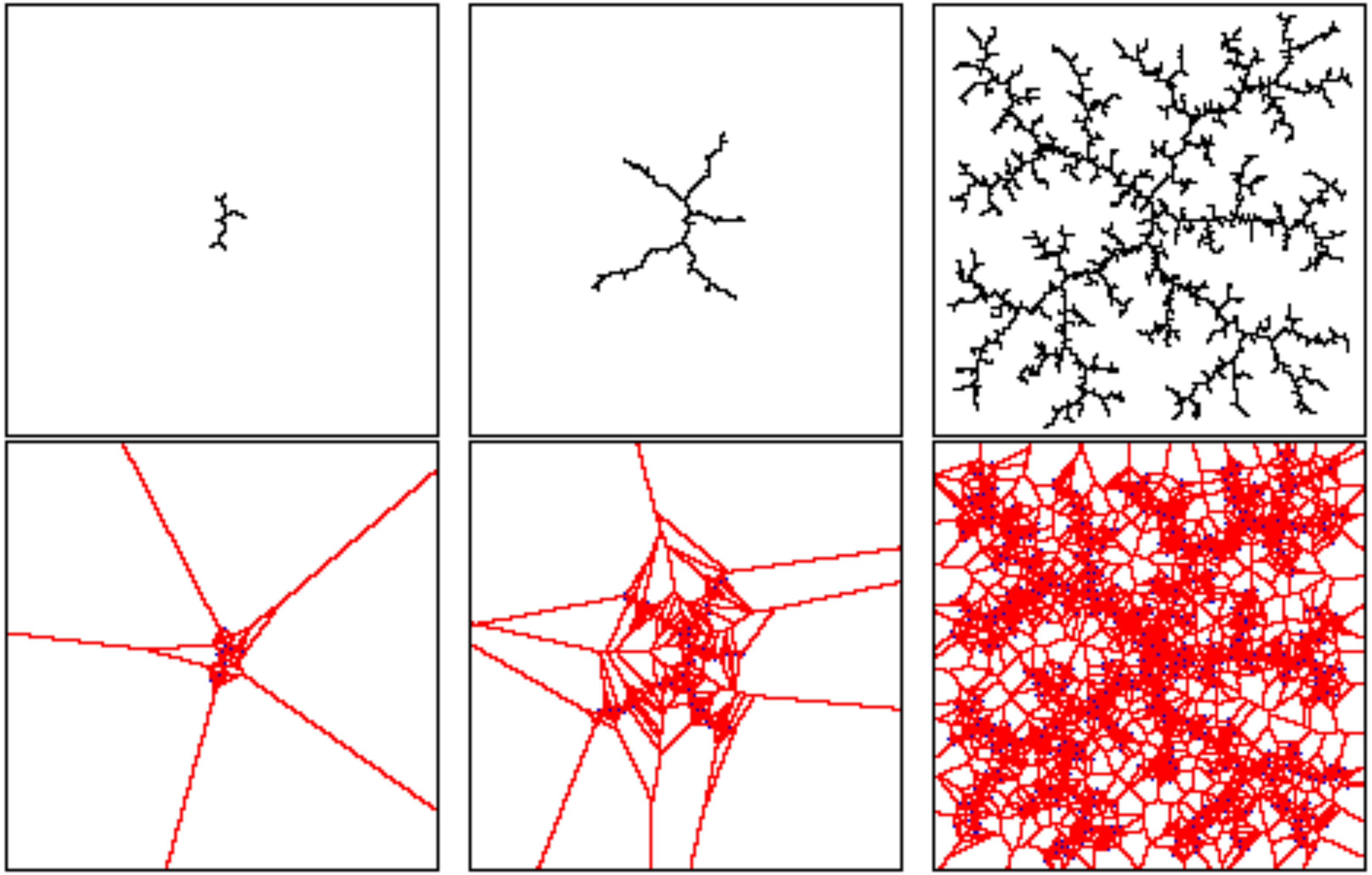
BUILD_RRT($q_{init}$)
- $\mathcal{T}$.init($q_{init}$);
- **for** $k = 1$ **to** $K$ **do**
  - $q_{rand} \leftarrow$ RANDOM_CONFIG();
  - EXTEND($\mathcal{T}$, $q_{rand}$);
- Return $\mathcal{T}$

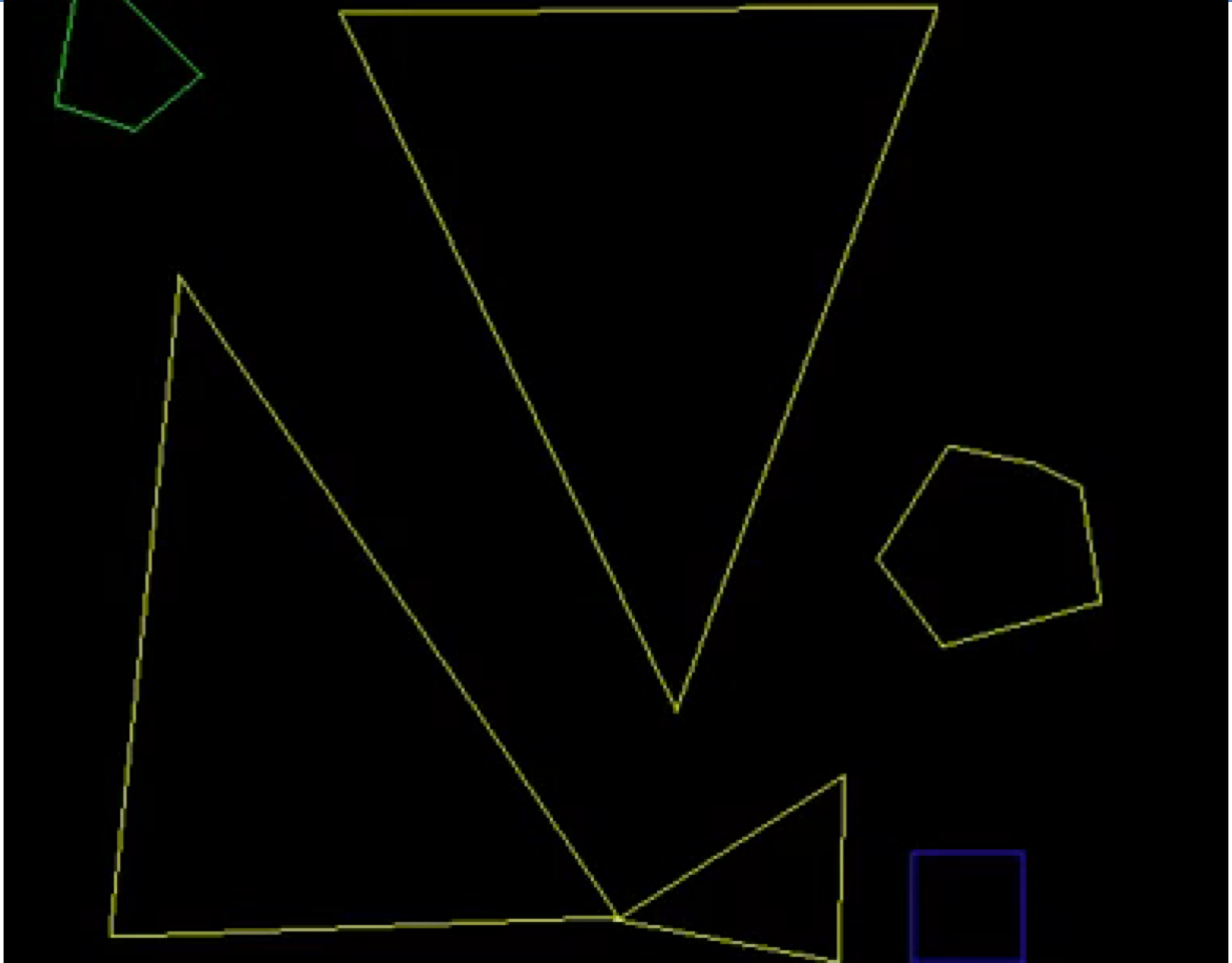
EXTEND($\mathcal{T}$, $q$)
- $q_{near} \leftarrow$ NEAREST_NEIGHBOR($q$, $\mathcal{T}$);
- **if** NEW_CONFIG($q, q_{near}, q_{new}$) **then**
  - $\mathcal{T}$.add_vertex($q_{new}$);
  - $\mathcal{T}$.add_edge($q_{near}, q_{new}$);
  - **if** $q_{new} = q$ **then**
    - Return *Reached*;
  - **else**
    - Return *Advanced*;
- Return *Trapped*;

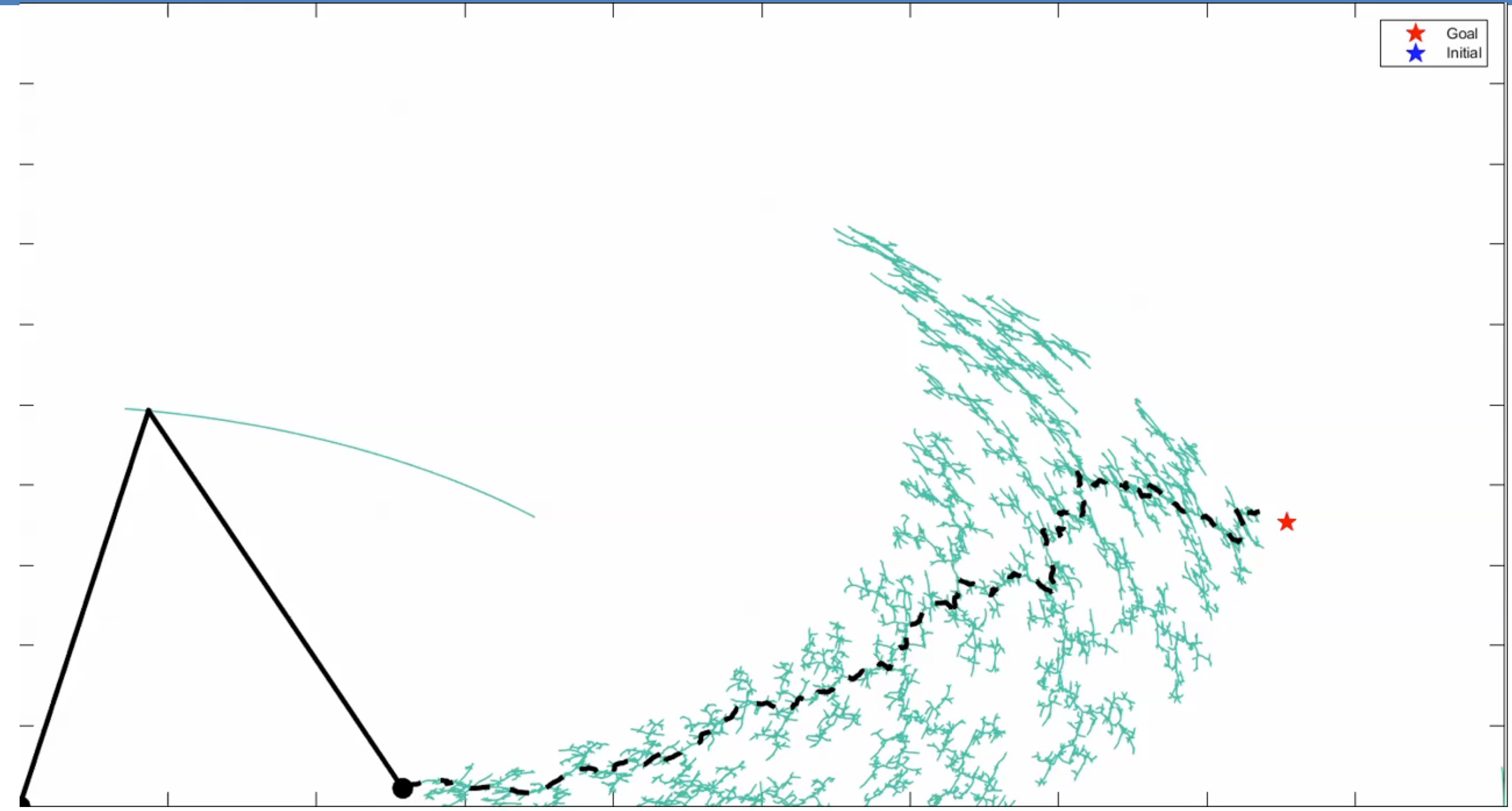


# Why are RRT's rapidly exploring?

The probability of a node to be selected for expansion is proportional to the area of its Voronoi region



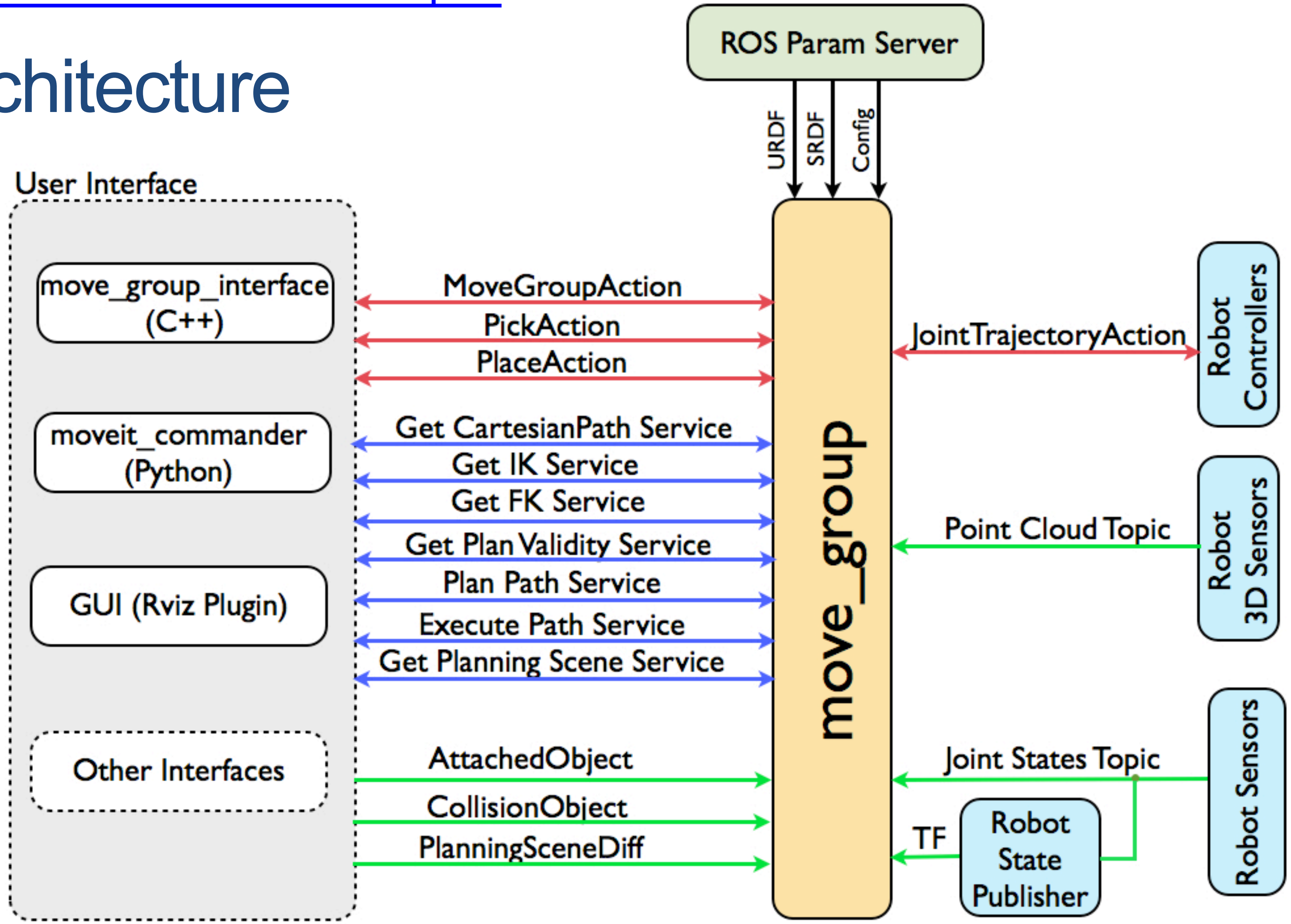



Goal
Initial

# MOVEIT

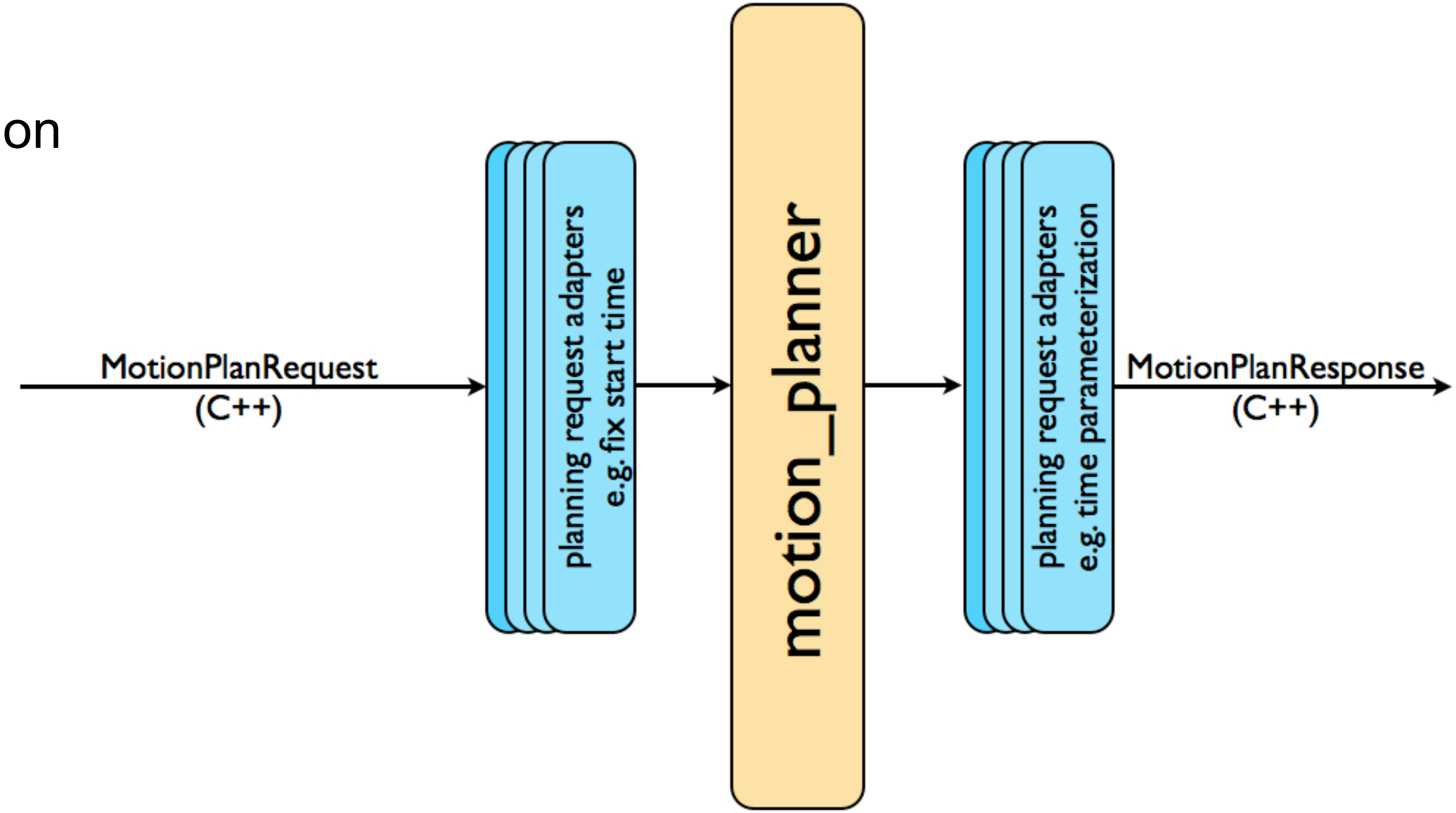
http://moveit.ros.org/documentation/concepts/

# System Architecture

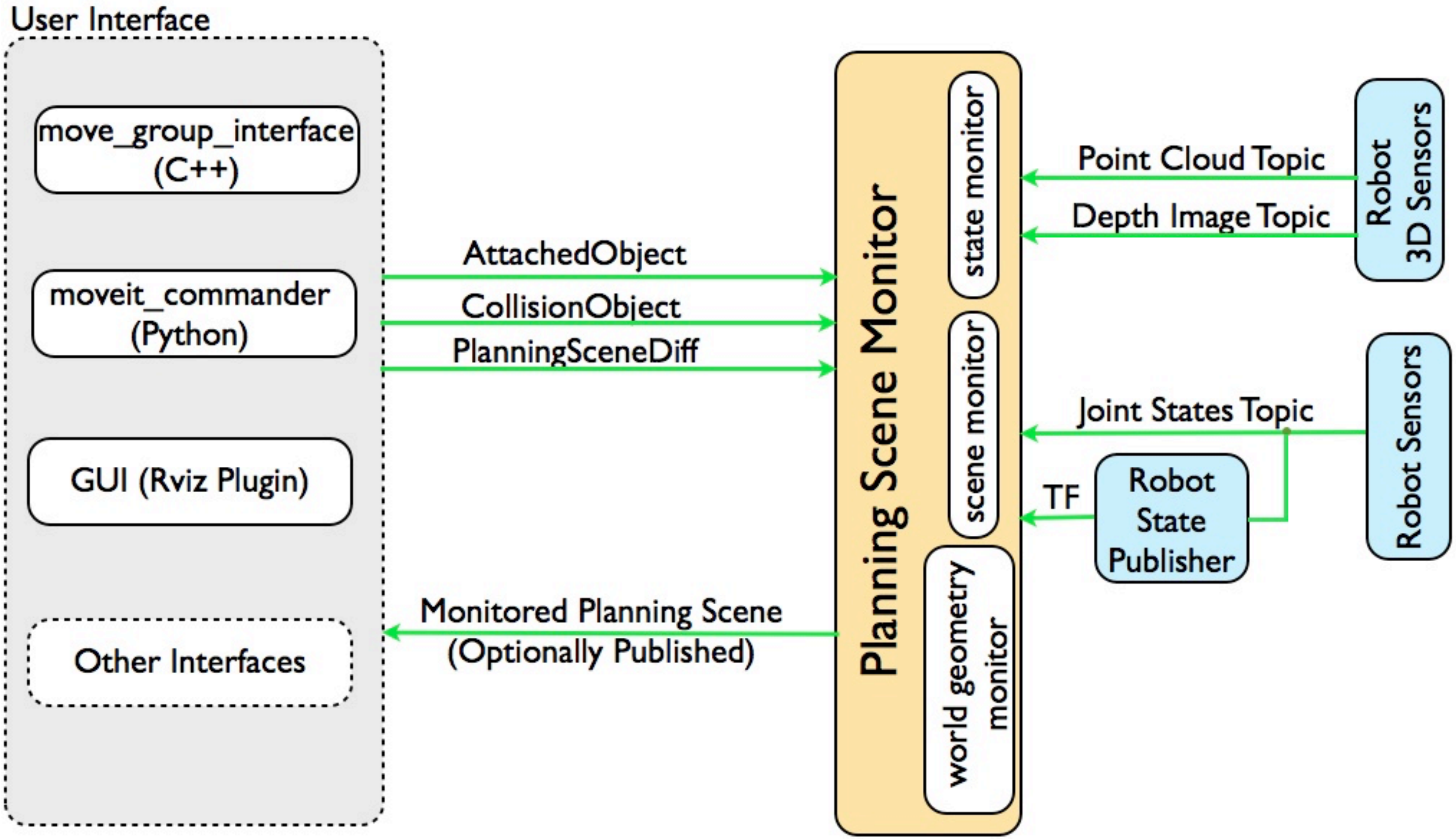


# Motion Planning

- Mainly:
- OMPL (Open Motion Planning Library)

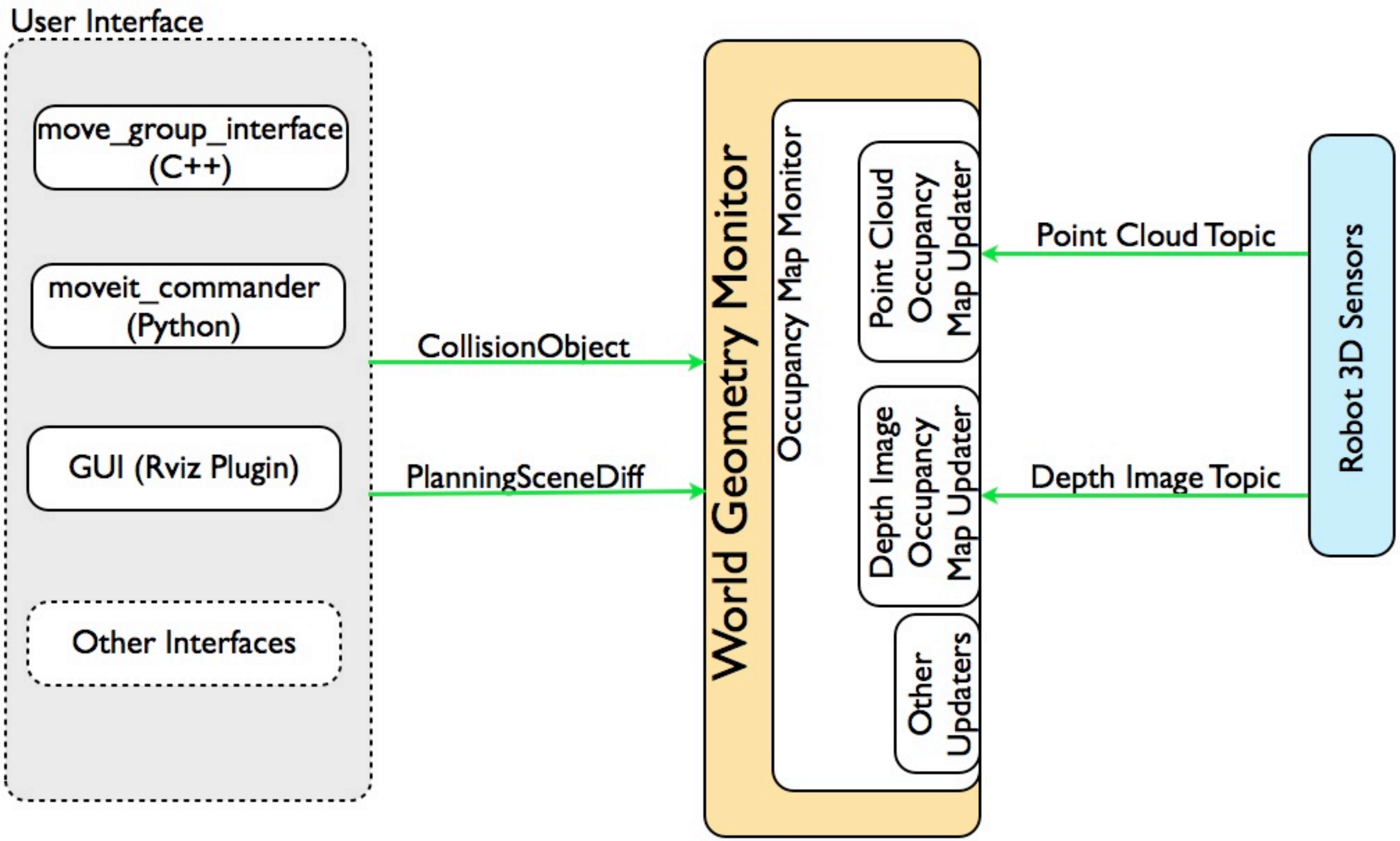


# Planning Scene



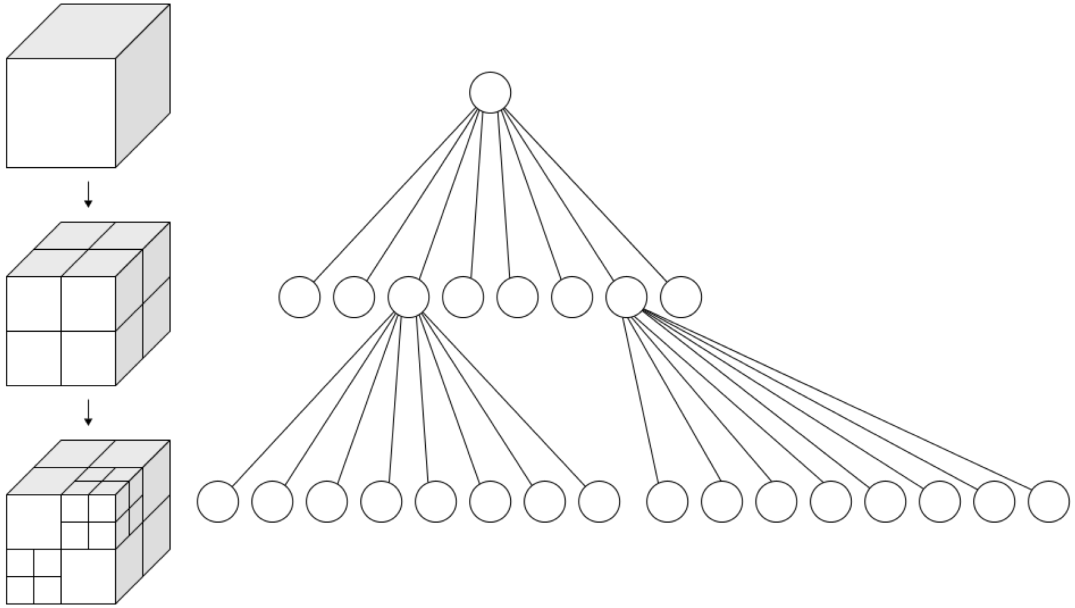
# 3D Perception

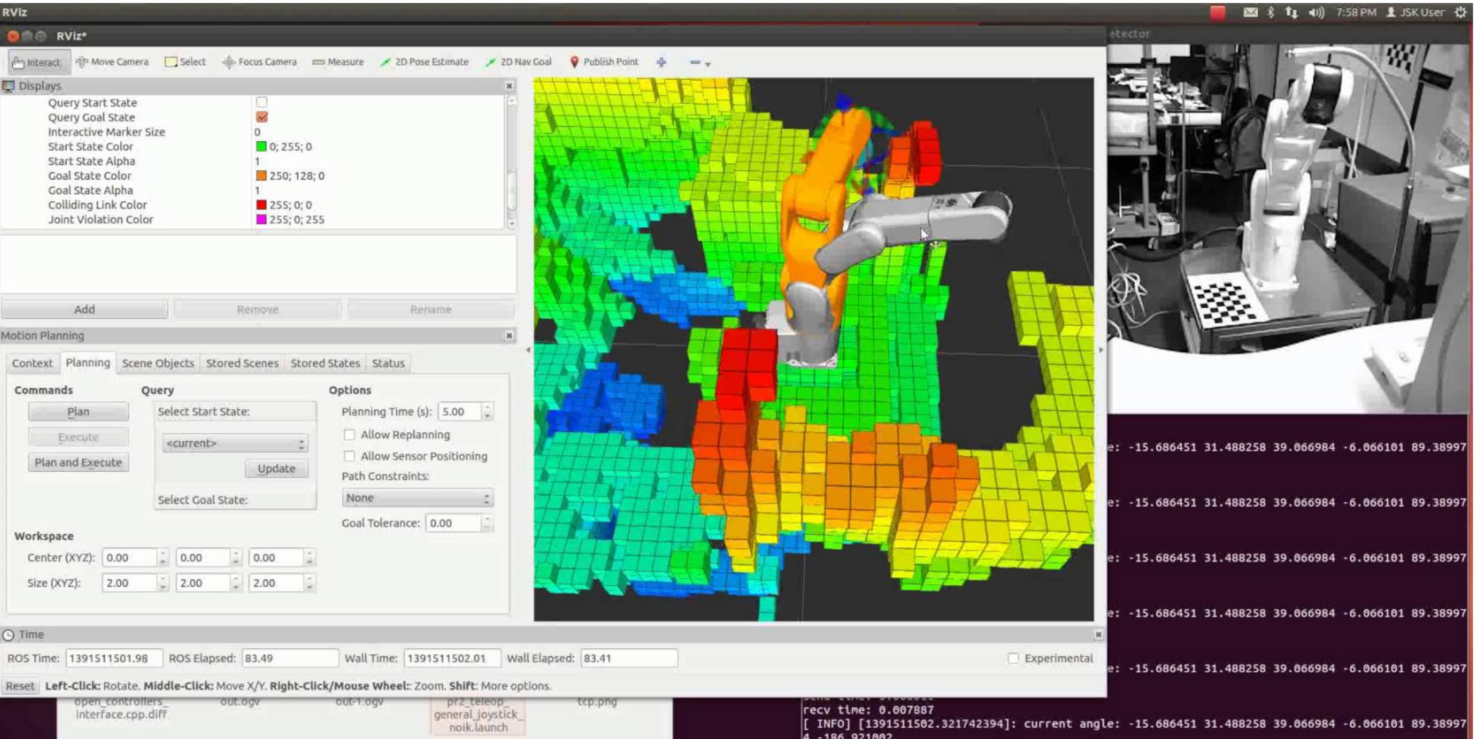
- Octomap



# Octomap / Octree

- Depth sensor (usually Kinect)
- http://wiki.ros.org/octomap



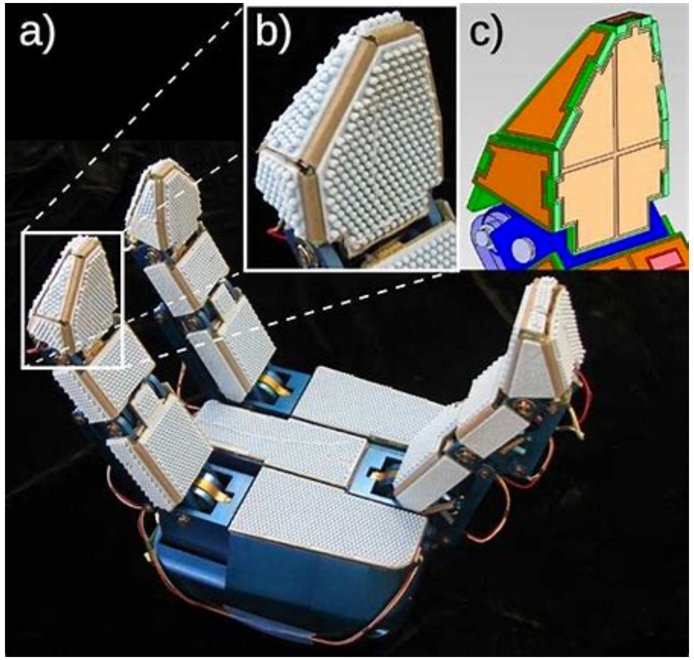


# Grasp an Object: Steps

1. Startup robot and sensors
2. Detect object & its pose
3. Select grasping points on the object
4. Scan the scene and environment (for collision checking later)
5. Use IK to check if grasping point can be reached – checks for collisions – may try thousands of possibilities (before concluding that there is always a collision)
6. Use motion planning to plan from current pose to goal pose: Lots of collision checks! Might realize that it is impossible after a long time
7. Execute that trajectory: Check if we reached the intermediate pose (within the time constraint) and command the next
8. Controller: take dynamics into account to move to the next intermediate pose
9. Once goal is reached close fingers.
10. Check if object is in fingers
11. Add the object to the collision description of the robot
12. Plan the path to the goal pose…

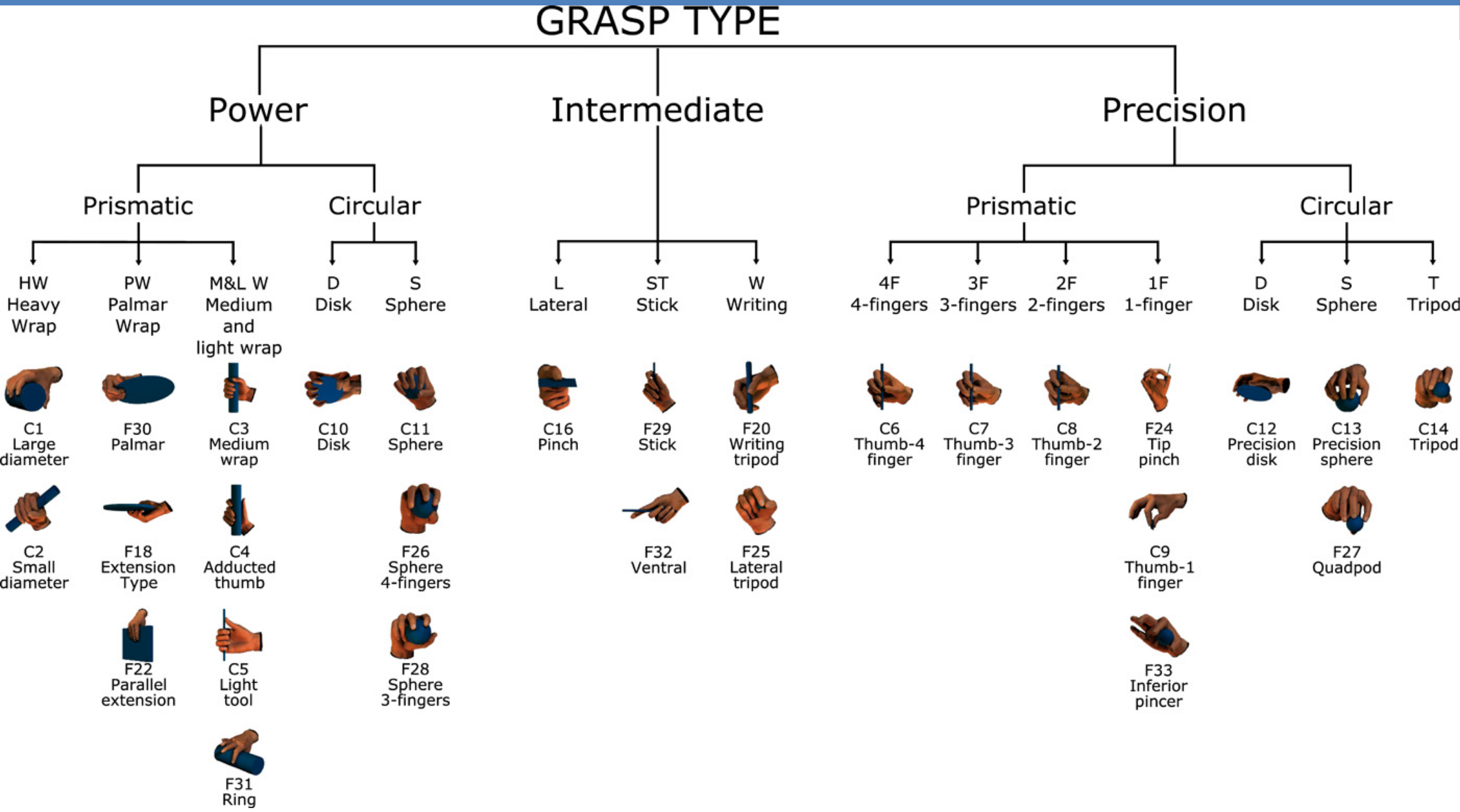
# Grasping …

- Gripper: Parallel; 3 Finger; Hand; Suction
- Force Sensors
- Tactile Sensors
- Grasp Types
- Grasp Planning




*"Biologically inspired tactile classification of object-hand and object-world interactions"*, by B. Heyneman and M. R. Cutkosky; Stanford University; Robio 2012

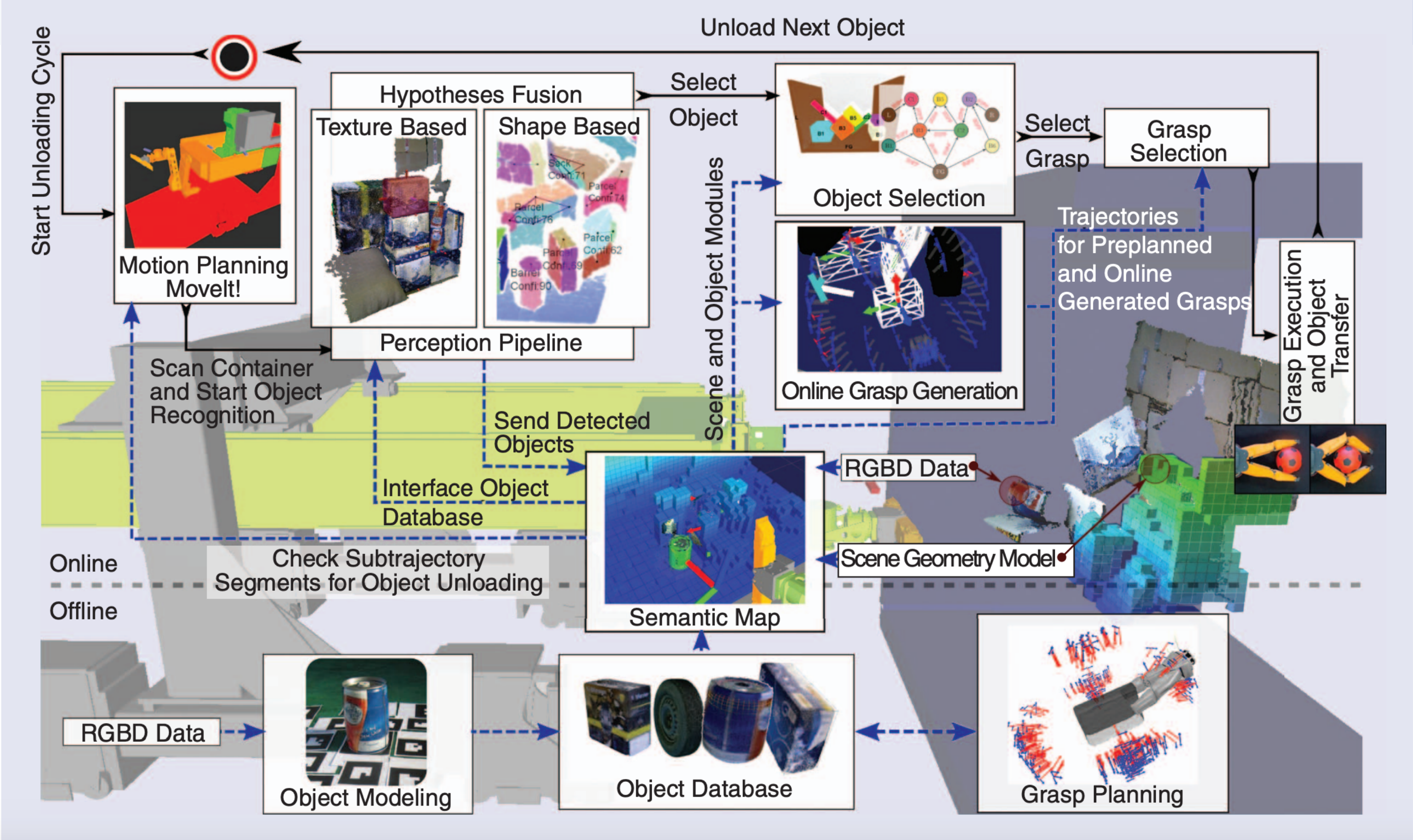
Citation for image on page 47: T. Stoyanov *et al.*, "No More Heavy Lifting: Robotic Solutions to the Container Unloading Problem," in *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 94-106, Dec. 2016.


GRASP TYPE
Power
Intermediate
Precision
Prismatic
Circular
Prismatic
Circular
HW Heavy Wrap
PW Palmar Wrap
M&L W Medium and light wrap
D Disk
S Sphere
L Lateral
ST Stick
W Writing
4F 4-fingers
3F 3-fingers
2F 2-fingers
1F 1-finger
D Disk
S Sphere
T Tripod
C1 Large diameter
C2 Small diameter
F30 Palmar
F18 Extension Type
F22 Parallel extension
C3 Medium wrap
C4 Adducted thumb
C5 Light tool
F31 Ring
C10 Disk
C11 Sphere
F26 Sphere 4-fingers
F28 Sphere 3-fingers
C16 Pinch
F29 Stick
F32 Ventral
F20 Writing tripod
F25 Lateral tripod
C6 Thumb-4 finger
C7 Thumb-3 finger
C8 Thumb-2 finger
F24 Tip pinch
C9 Thumb-1 finger
F33 Inferior pincer
C12 Precision disk
C13 Precision sphere
F27 Quadpod
C14 Tripod

Lu, Qingkai & Hermans, Tucker. (2019). Modeling Grasp Type Improves Learning-Based Grasp Planning.

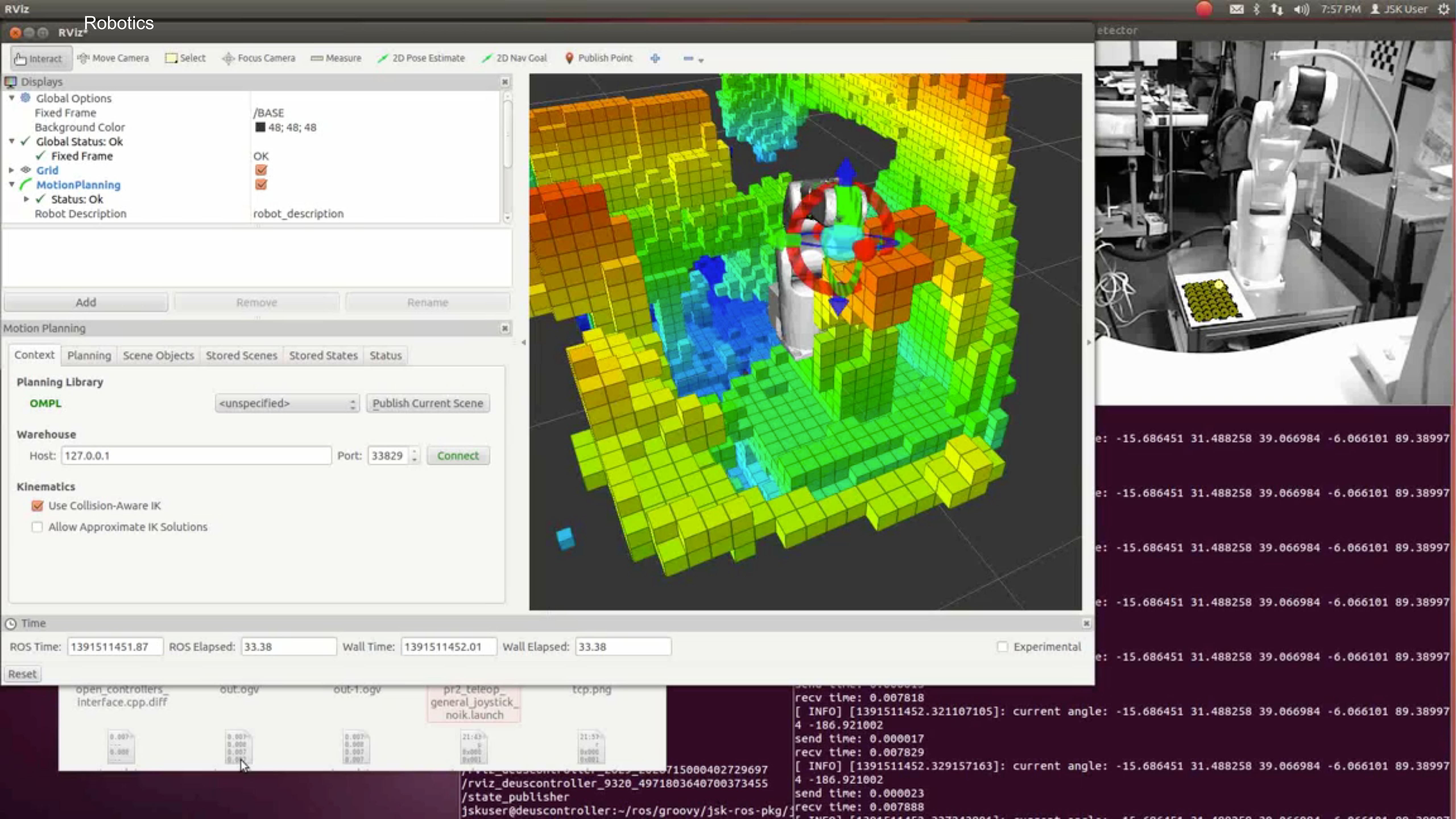
Precision

Power



Unload Next Object
Start Unloading Cycle
Hypotheses Fusion
Texture Based
Shape Based
Select Object
Object Selection
Select Grasp
Grasp Selection
Motion Planning MoveIt!
Perception Pipeline
Scene and Object Modules
Online Grasp Generation
Trajectories for Preplanned and Online Generated Grasps
Grasp Execution and Object Transfer
Scan Container and Start Object Recognition
Send Detected Objects
Interface Object Database
RGBD Data
Semantic Map
Scene Geometry Model
Online
Check Subtrajectory Segments for Object Unloading
Offline
RGBD Data
Object Modeling
Object Database
Grasp Planning

# Robotics


RViz
RViz*
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
/BASE
Background Color
48; 48; 48
Global Status: Ok
Fixed Frame
OK
Grid
MotionPlanning
Status: Ok
Robot Description
robot_description
Add
Remove
Rename
Motion Planning
Context
Planning
Scene Objects
Stored Scenes
Stored States
Status
Planning Library
OMPL
<unspecified>
Publish Current Scene
Warehouse
Host: 127.0.0.1
Port: 33829
Connect
Kinematics
Use Collision-Aware IK
Allow Approximate IK Solutions
Time
ROS Time: 1391511451.87
ROS Elapsed: 33.38
Wall Time: 1391511452.01
Wall Elapsed: 33.38
Experimental
Reset
7:57 PM
JSK User
recv time: 0.007818
[ INFO] [1391511452.321107105]: current angle: -15.686451 31.488258 39.066984 -6.066101 89.38997
4 -186.921002
send time: 0.000017
recv time: 0.007829
[ INFO] [1391511452.329157163]: current angle: -15.686451 31.488258 39.066984 -6.066101 89.38997
4 -186.921002
send time: 0.000023
recv time: 0.007888
/rviz_deuscontroller_9320_4971803640700373455
/state_publisher
jskuser@deuscontroller:~/ros/groovy/jsk-ros-pkg/
open_controllers_interface.cpp.diff
out.ogv
out-1.ogv
pr2_teleop_general_joystick_noik.launch
tcp.png