Task Space, Workspace and Introduction to ROS

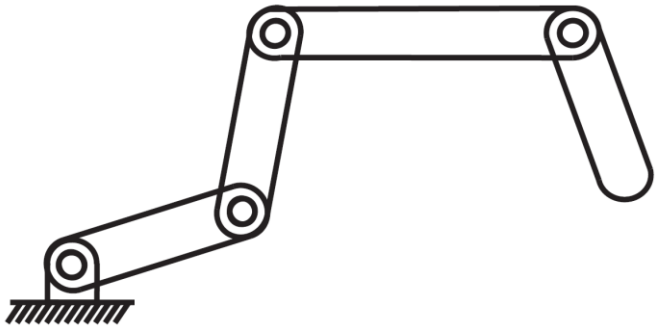
CS 6301 Special Topics: Introduction to Robot Manipulation and Navigation

Professor Yu Xiang

The University of Texas at Dallas

Configuration Space of a Robot

- The configuration of a robot is a complete specification of the position of every point of the robot.
- The minimum number n of real-valued coordinates needed to represent the configuration is the number of degrees of freedom (DOF) of the robot.
- The n-dimensional space containing all possible configurations of the robot is called the configuration space (C-space).
- The configuration of a robot is represented by a point in its C-space.



- 4 revolute joints
- 4 DOFs

Task Space

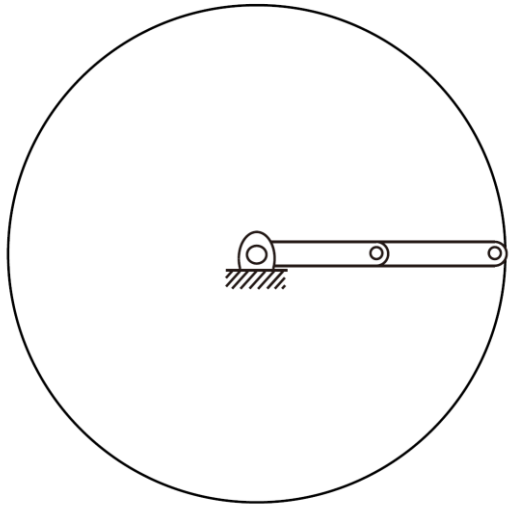
- The task space is a space in which the robot's task can be naturally expressed

- Task examples
  - Draw on a piece of paper: $\mathbb{R}^2$
  - Manipulate a rigid body: C-space of the rigid body

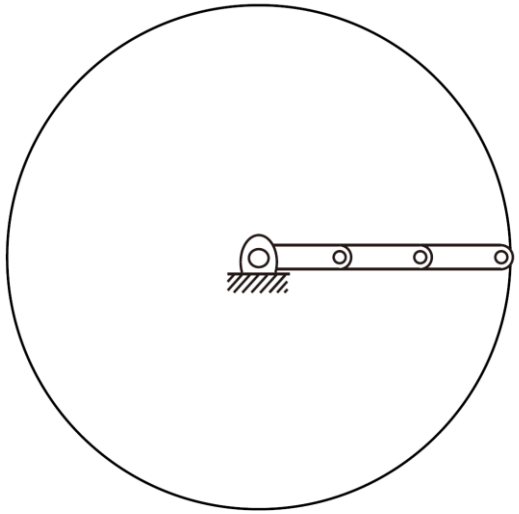
- Task space is driven by the task, independently of the robot

Workspace

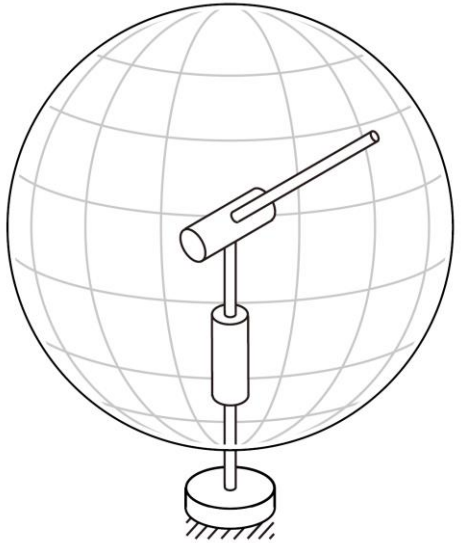
- The workspace is a specification of the configurations that the end-effector of the robot can reach.
- Depends on the robot structure, independent of the task



a planar 2R open chain

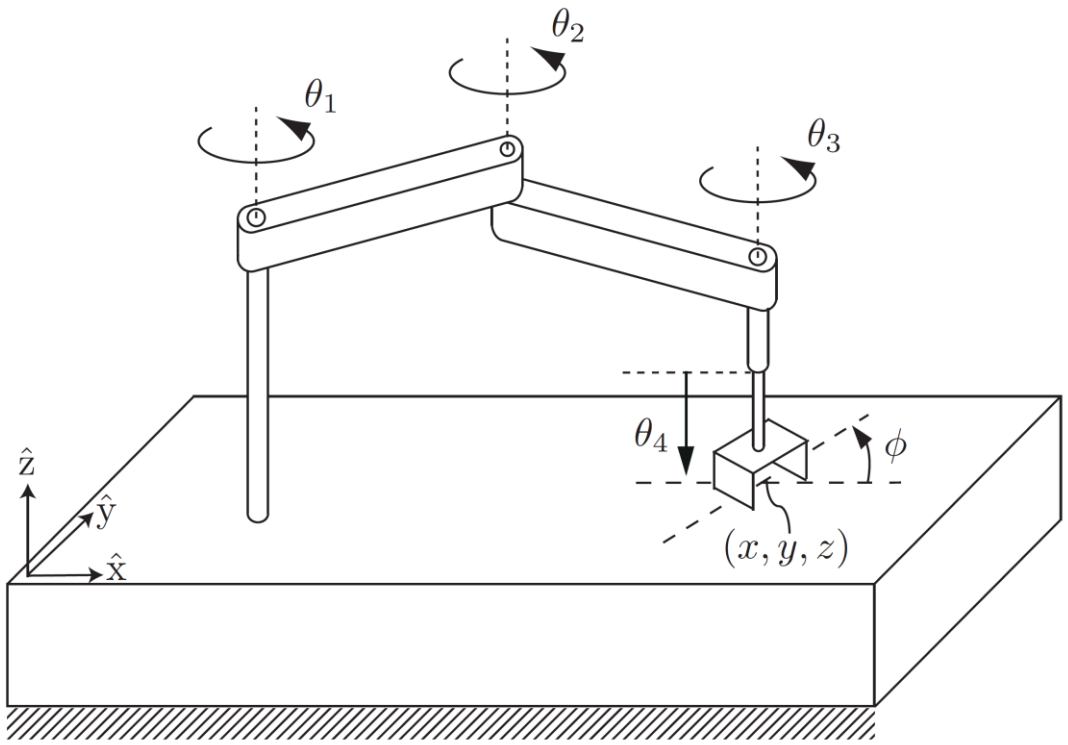


a planar 3R open chain



a spherical 2R open chain

SCARA Robot



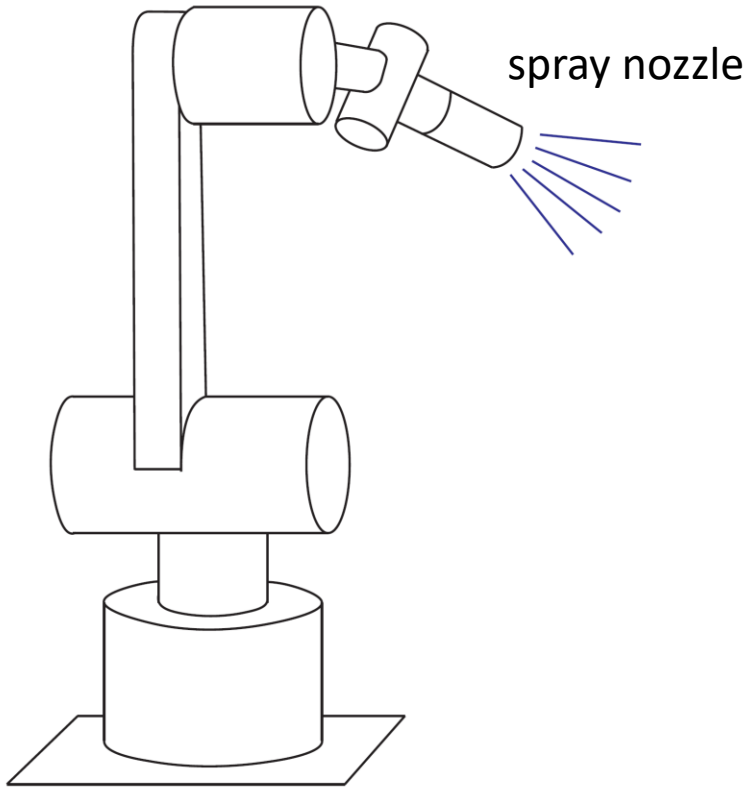
- End-effector configuration

  (x, y, z, ϕ)

- Task space $\mathbb{R}^3 \times S^1$

- Workspace
  - Reachable (x, y, z, ϕ)

A 6R Robot



A spray-painting robot

- End-effector configuration

$$(x, y, z) \quad (\theta, \phi)$$

Cartesian position of the nozzle

Spherical coordinates to describe the direction in which the nozzle is pointing

- Task space $\mathbb{R}^3 \times S^2$
- Workspace
  - Reachable

$$(x, y, z) \quad (\theta, \phi)$$

https://www.rnaautomation.com/case-study/robotic-spray-booth/

Robot Programming

- Sensing
  - How to receive data from sensors on the robot?
  - RGB image, depth image, lidar scan, odometry, joint state
- Computation
  - Use the sensor data for computation
  - Object recognition, motion planning, compute control command, etc.
- Control
  - How to send the control command to the robot?

Robot Operating System (ROS)

- ROS is a set of software libraries and tools that can be used to build robot applications
  - Drivers, algorithms, developer tools, etc.
- Goal of ROS: support code reuse in robotics research and development
- Operating systems: Unix-based platforms (Ubuntu)

https://www.ros.org/
https://wiki.ros.org/

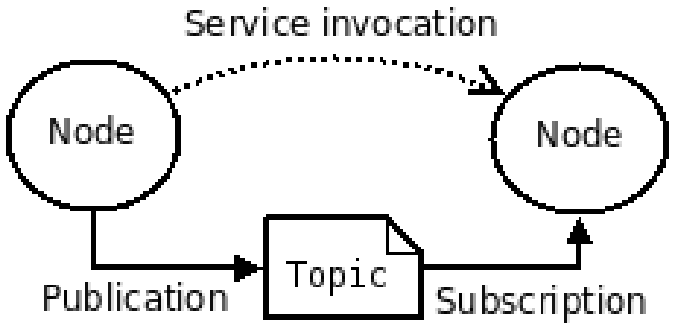
Robot Operating System (ROS)

https://www.ros.org/

ROS Computation Graph

- The computation graph is the peer-to-peer network of ROS processes that are processing data together

- Computation graph concepts
  - Nodes: processes that perform computation
  - ROS Master: provides name registration and lookup, nodes can find each other via ROS master
  - Messages: nodes communicate by passing messages, a data structure with type fields (integer, floating, arrays, etc.)



ROS Message Example

File: sensor_msgs/Image.msg

Raw Message Definition

```
# This message contains an uncompressed image
# (0, 0) is at top-left corner of image
#

Header header        # Header timestamp should be acquisition time of image
                     # Header frame_id should be optical frame of camera
                     # origin of frame should be optical center of camera
                     # +x should point to the right in the image
                     # +y should point down in the image
                     # +z should point into to plane of the image
                     # If the frame_id here and the frame_id of the CameraInfo
                     # message associated with the image conflict
                     # the behavior is undefined

uint32 height         # image height, that is, number of rows
uint32 width          # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding       # Encoding of pixels -- channel meaning, ordering, size
                      # taken from the list of strings in include/sensor_msgs/image_encodings.h

uint8 is_bigendian    # is this data bigendian?
uint32 step           # Full row length in bytes
uint8[] data          # actual matrix data, size is (step * rows)
```

std_msgs/Header Message

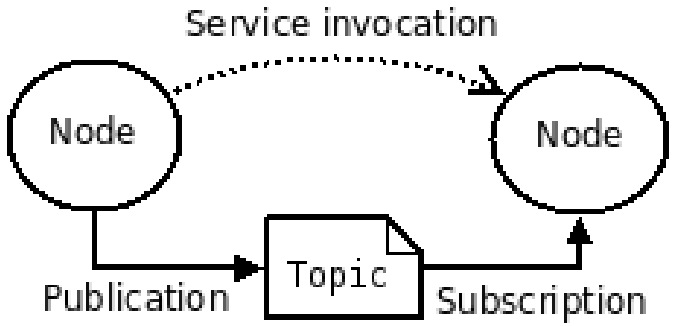
File: std_msgs/Header.msg

Raw Message Definition

```
# Standard metadata for higher-level stamped data types.
# This is generally used to communicate timestamped data
# in a particular coordinate frame.
#
# sequence ID: consecutively increasing ID
uint32 seq
#Two-integer timestamp that is expressed as:
# * stamp.sec: seconds (stamp_secs) since epoch (in Python the variable is called 'secs')
# * stamp.nsec: nanoseconds since stamp_secs (in Python the variable is called 'nsecs')
# time-handling sugar is provided by the client library
time stamp
#Frame this data is associated with
string frame_id
```

ROS Computation Graph

- Topics: a node publishes messages to a topic. The topic is the name to identify the content of the message



Node: Head camera

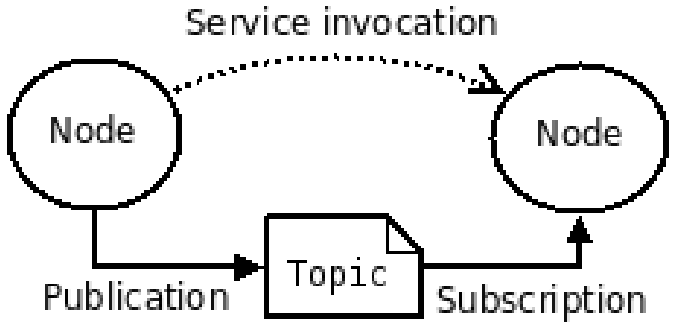
Node: Object detection

Topic: /rgb_image

Message: sensor_msgs/Image

ROS Computation Graph

- Service: request and reply interactions



Service: `FollowJointTrajectory`

Node: Motion Planner

File: control_msgs/FollowJointTrajectoryGoal.msg

```
# ====== DO NOT MODIFY! AUTOGENERATED FROM AN ACTION DEFINITION ======
# The joint trajectory to follow
trajectory_msgs/JointTrajectory trajectory

# Tolerances for the trajectory.  If the measured joint values fall
# outside the tolerances the trajectory goal is aborted.  Any
# tolerances that are not specified (by being omitted or set to 0) are
# set to the defaults for the action server (often taken from the
# parameter server).

# Tolerances applied to the joints as the trajectory is executed.  If
# violated, the goal aborts with error_code set to
# PATH_TOLERANCE_VIOLATED.
JointTolerance[] path_tolerance

# To report success, the joints must be within goal_tolerance of the
# final trajectory value.  The goal must be achieved by time the
# trajectory ends plus goal_time_tolerance.  (goal_time_tolerance
# allows some leeway in time, so that the trajectory goal can still
# succeed even if the joints reach the goal some time after the
# precise end time of the trajectory).
#
# If the joints are not within goal_tolerance after "trajectory finish
# time" + goal_time_tolerance, the goal aborts with error_code set to
# GOAL_TOLERANCE_VIOLATED
JointTolerance[] goal_tolerance
duration goal_time_tolerance
```

Node: Arm Controller

File: control_msgs/FollowJointTrajectoryAction.msg

Raw Message Definition

```
# ====== DO NOT MODIFY! AUTOGENERATED FROM AN ACTION DEFINITION ======

FollowJointTrajectoryActionGoal action_goal
FollowJointTrajectoryActionResult action_result
FollowJointTrajectoryActionFeedback action_feedback
```

Compact Message Definition

```
control_msgs/FollowJointTrajectoryActionGoal action_goal
control_msgs/FollowJointTrajectoryActionResult action_result
control_msgs/FollowJointTrajectoryActionFeedback action_feedback
```

ROS Computation Graph

- ROS bags
  - A format for saving and playing back ROS message data
  - We can save sensor data into a ros bag, and use it for development

```
rosbag record --duration=30 --output-name=/tmp/mybagfile.bag \
    /topic1 /topic2 /topic3
```

Docker

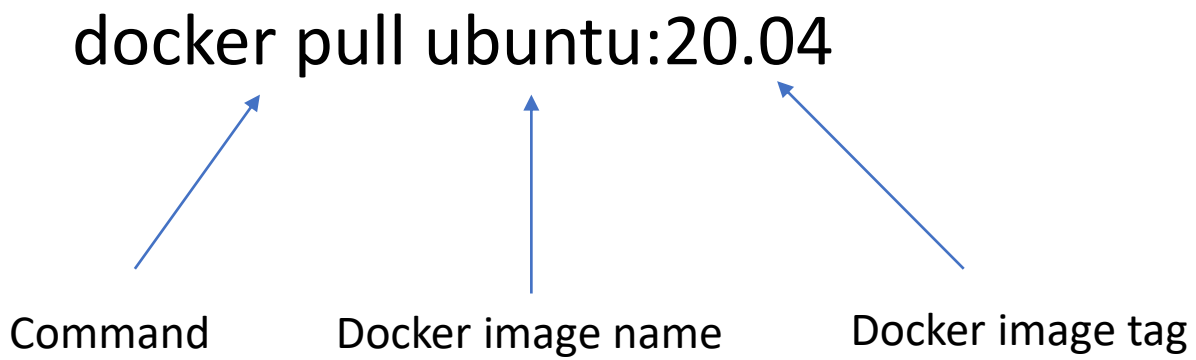
- An open platform that enables you to separate your applications from your infrastructure

- Container
  - A lightweight environment that contains everything to run an application
  - A container is a runnable instance of an image

- Image
  - A read-only template with instructions for creating a docker container

Ubuntu in Docker

- Download the ubuntu docker image
  https://hub.docker.com/_/ubuntu



Docker

docker run -i -t ubuntu:20.04 /bin/bash

- Run an **ubuntu container**
- You need to have an **ubuntu image** locally, if not, the command will pull an ubuntu image as by docker pull ubuntu
- Docker creates a new container as though you had run docker container create
- Docker starts the container and execute /bin/bash
- -i, -t the container is running interactively and attached to your terminal
- When exit, the container stops but is not removed

ROS in Docker

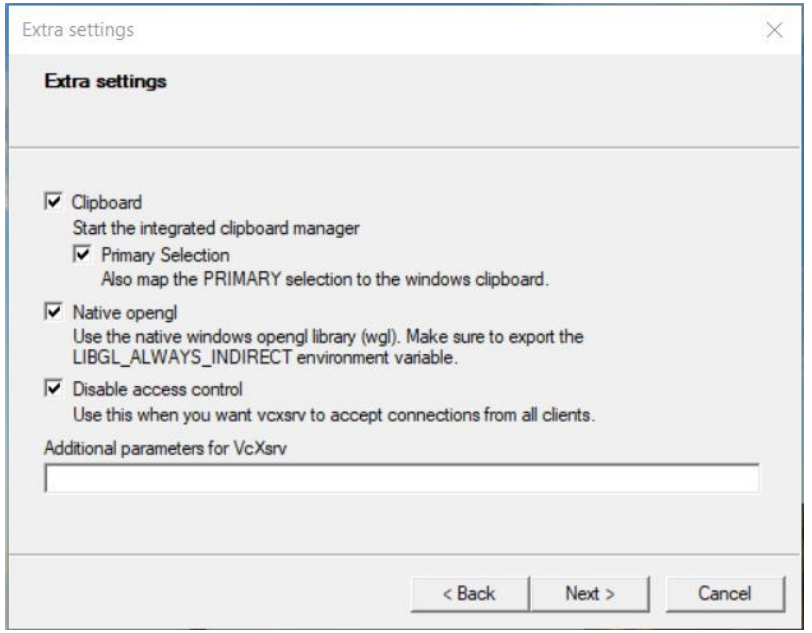
- Install Docker Desktop https://docs.docker.com/get-docker/
- Start the Docker Desktop
- Ubuntu images https://hub.docker.com/_/ubuntu
- Run command “docker run –i –t ubuntu:20.04 /bin/bash”
- No need to use sudo in docker, do an “apt update” first
- Install ROS http://wiki.ros.org/noetic/Installation/Ubuntu
- Install terminator https://manpages.ubuntu.com/manpages/bionic/en/man1/terminator.1.html

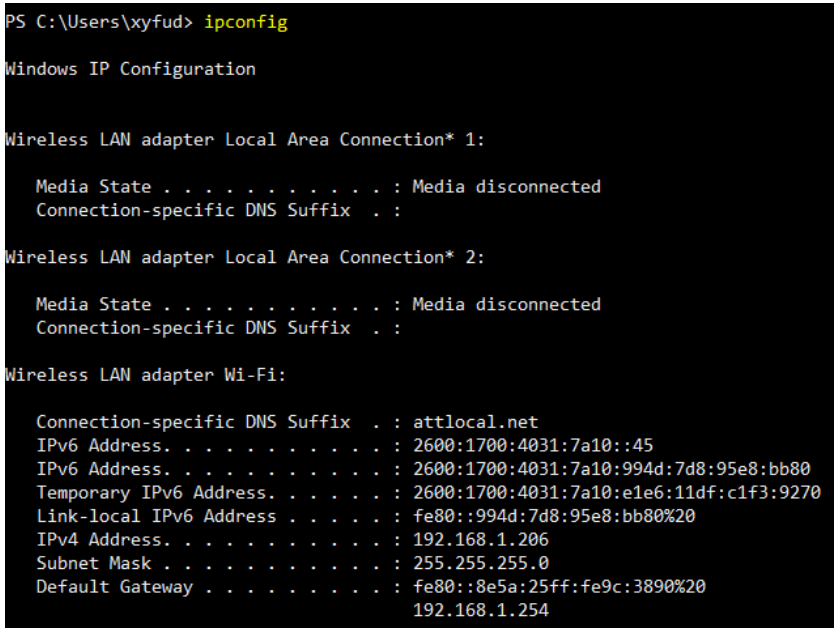
ROS in Docker

- Install X server
  - Windows: VcXsrv Windows X Server https://sourceforge.net/projects/vcxsrv/
  - Mac: Xquartz https://www.xquartz.org/

- Start the X server
- Check IP address
- In Ubuntu terminal

Export DISPLAY=my_ip:0.0





https://medium.com/@potatowagon/how-to-use-gui-apps-in-linux-docker-container-from-windows-host-485d3e1c64a3

ROS in Docker

- Test ROS installation

- In one terminator terminal, start roscore
  - source /opt/ros/noetic/setup.bash
  - roscore

- In another terminator terminal, start rviz
  - source /opt/ros/noetic/setup.bash
  - rosrun rviz rviz

Commit Your Docker Image

- After you exit the docker container
- Run the command “docker container list -a” to see all the containers. Find the container ID of the latest one
- Run the command “docker container commit <CONTAINER_ID>”
- Run the command “docker image list -a” to see the latest image ID
- Run the command “docker image tag <IMAGE_ID> TAG”. Give a name to this image such as “ubuntu:ros”

ROS in Docker

- After install all needed packages, exit
- docker container commit CONTAINER_ID
- docker image tag <IMAGE_ID> TAG

- Useful commands
  - docker container list –a
  - docker image list -a

- The new tagged image will have all the installed packages

Summary

- Task space
- Workspace
- ROS
- Docker

Further Reading

- Chapter 2 in Kevin M. Lynch and Frank C. Park. Modern Robotics: Mechanics, Planning, and Control. 1st Edition, 2017 http://hades.mech.northwestern.edu/images/7/7f/MR.pdf

- ROS wiki https://wiki.ros.org/

- Docker document https://docs.docker.com/get-started/overview/