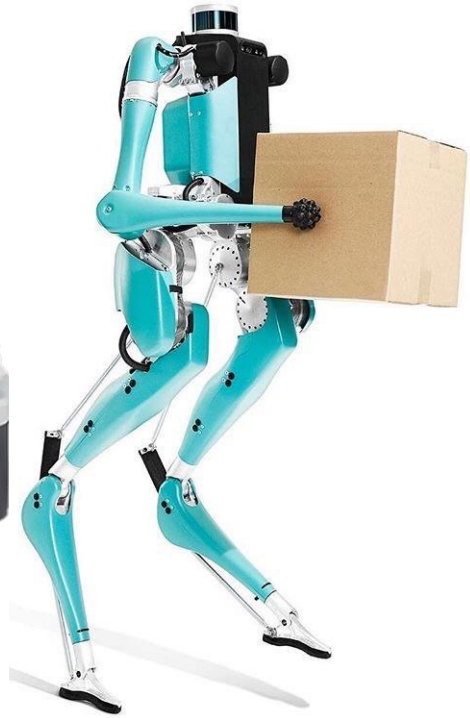
Armlab Introduction

Lecture 1

Fall 2022

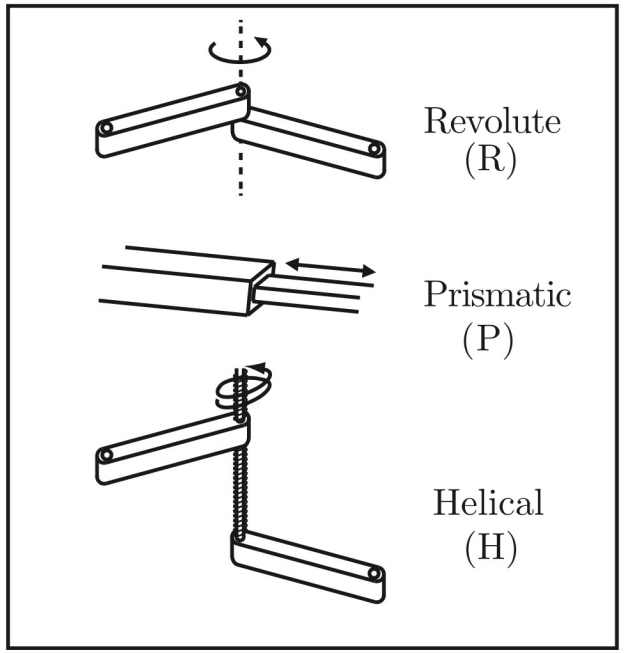
M | ROBOTICS

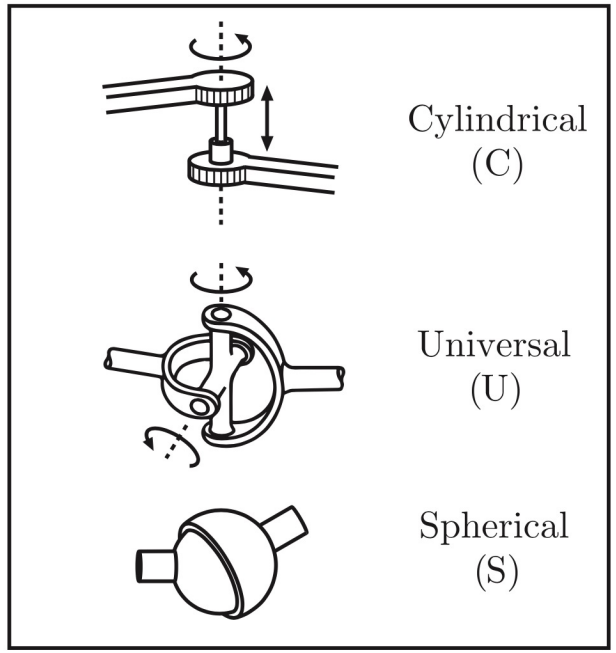
Manipulators

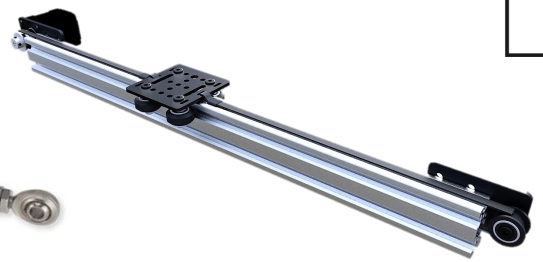


Joints

- Two basic types of actuated joints
- Rotational → Revolute
- Linear → Prisma tic
- Other types can be composed
- Spherical, Helical, etc.



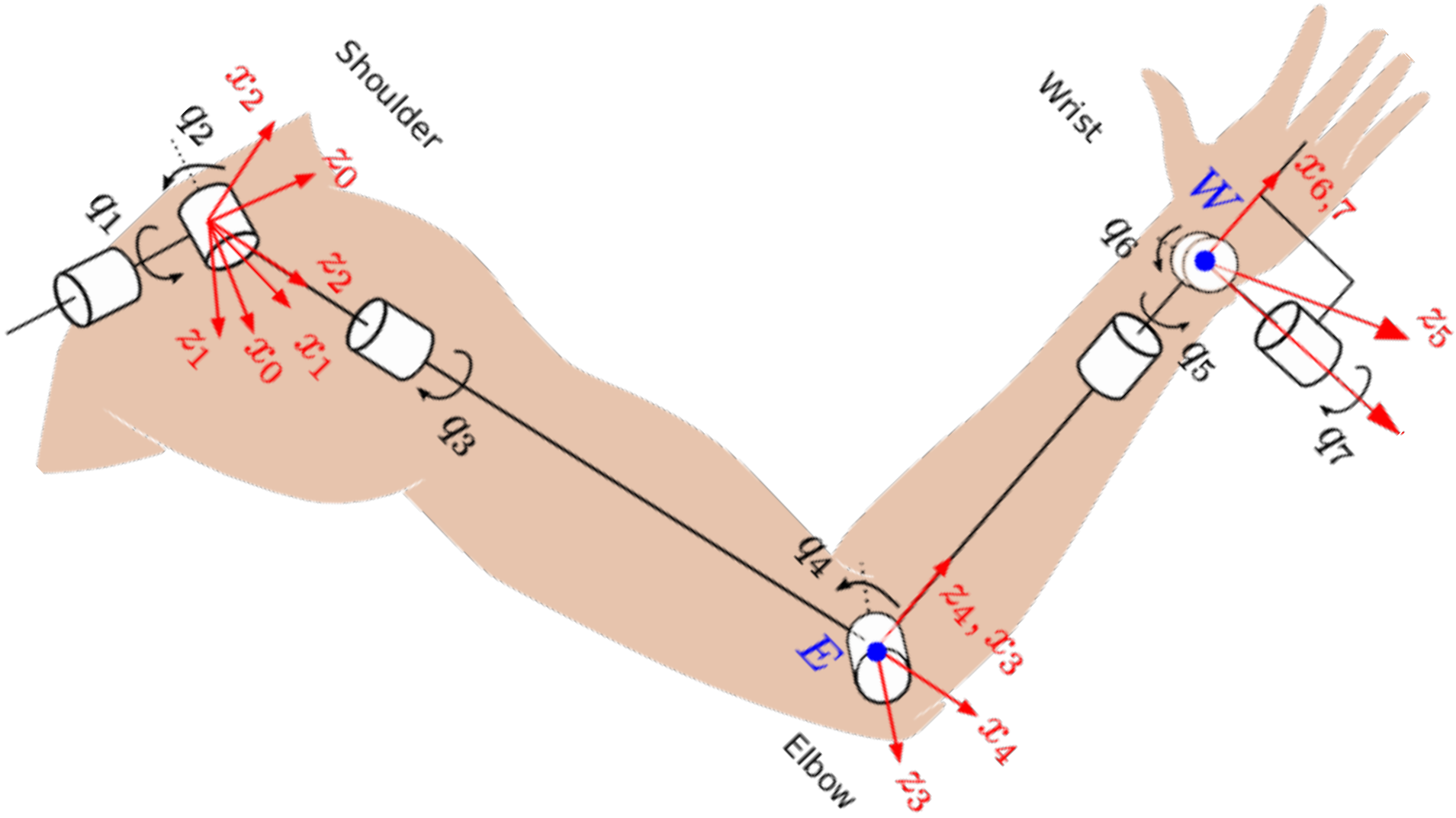




Degrees of Freedom

- An object has n degrees of freedom if it can be minimally specified by n parameters
- A rigid object on a plane (2D) has 3 DOF
  - 2 for position (x, y), 1 for orientation θ.
- A rigid object in space (3D) has 6 DOF
  - 3 for position (x, y, z), 3 for orientation (θ, ϕ, ψ).
- The number of DOF of a manipulator is equal to the dimension of the configuration space.
- A manipulator with more than 6DOF is kinematically redundant

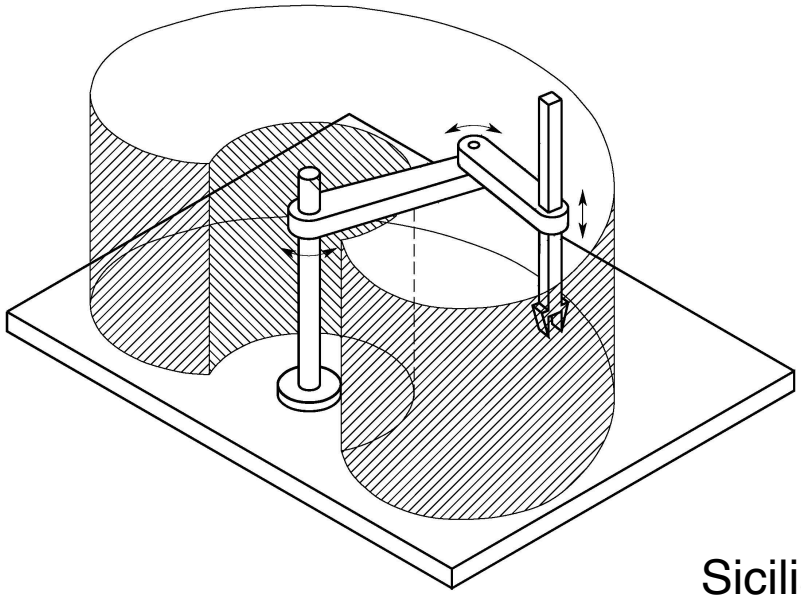
Human arm


Shoulder
Elbow
Wrist
W
E
q1
q2
q3
q4
q5
q6
q7
z0
x0
z1
x1
z2
x2
z3
x3
z4, x4
z5
x6, z6
z7

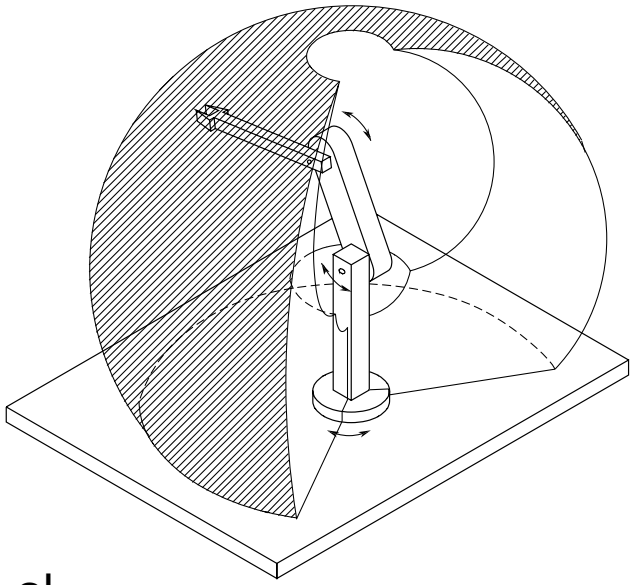
Workspace

- Total volume swept out by the end effector as the manipulator executes all possible motions
  - Reachable workspace: points that can be reached by the manipulator
  - Dexterous workspace: points that can be reached with arbitrary orientation

Epson

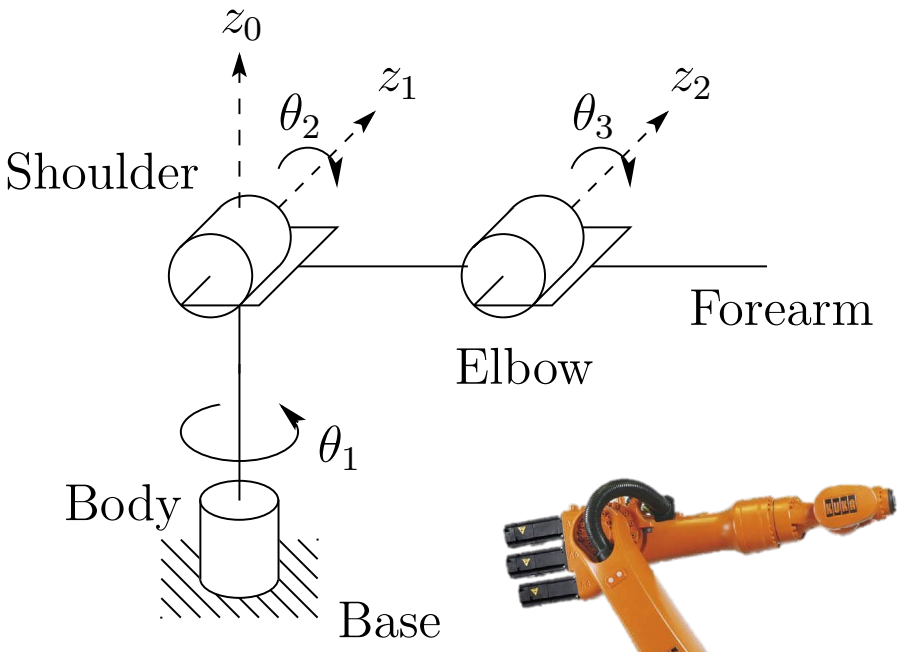


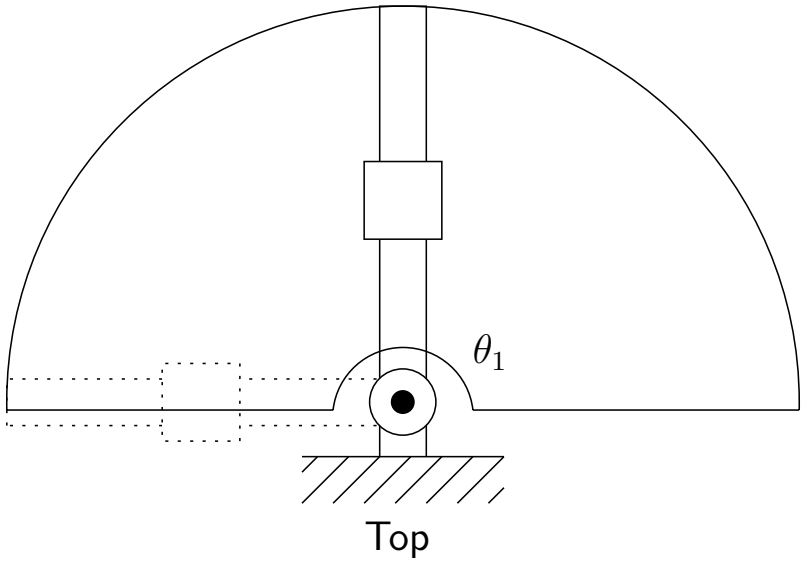
Siciliano et. al.

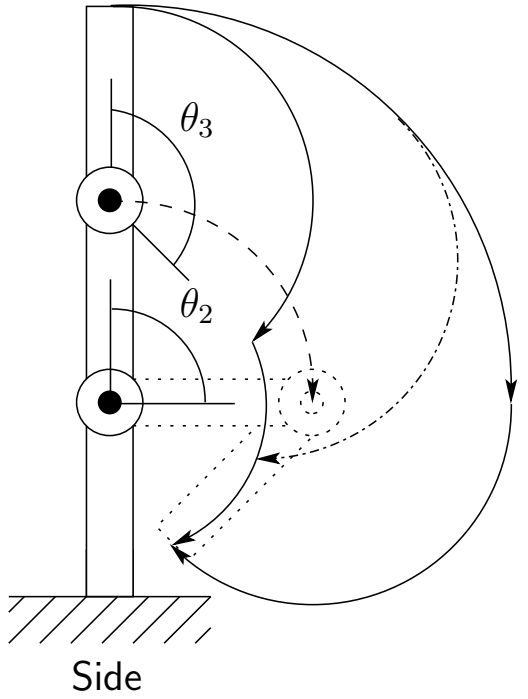


KUKA

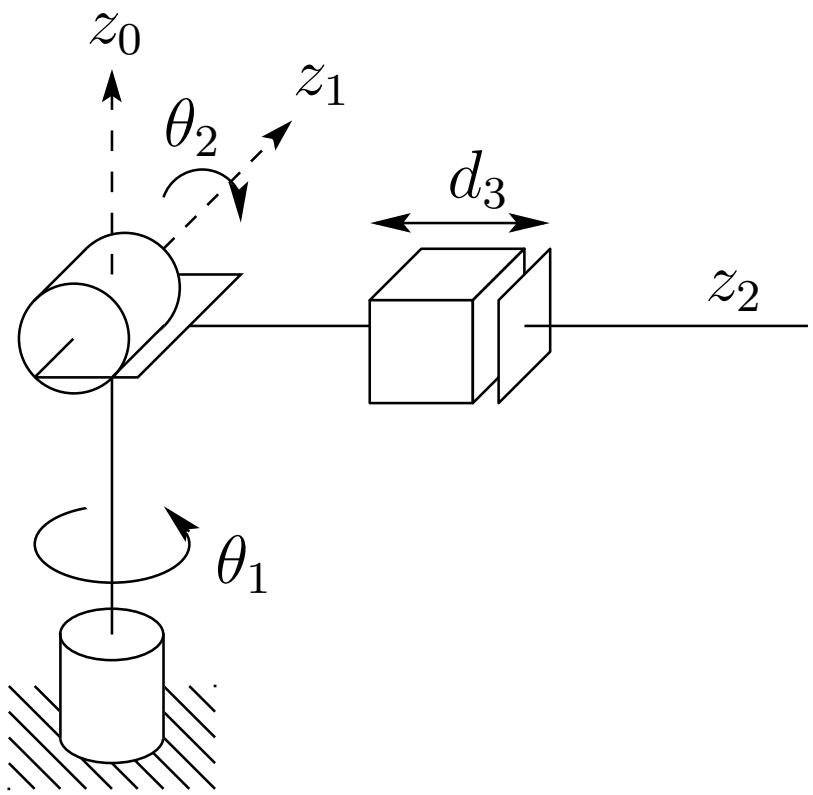
Common Robots - Articulated

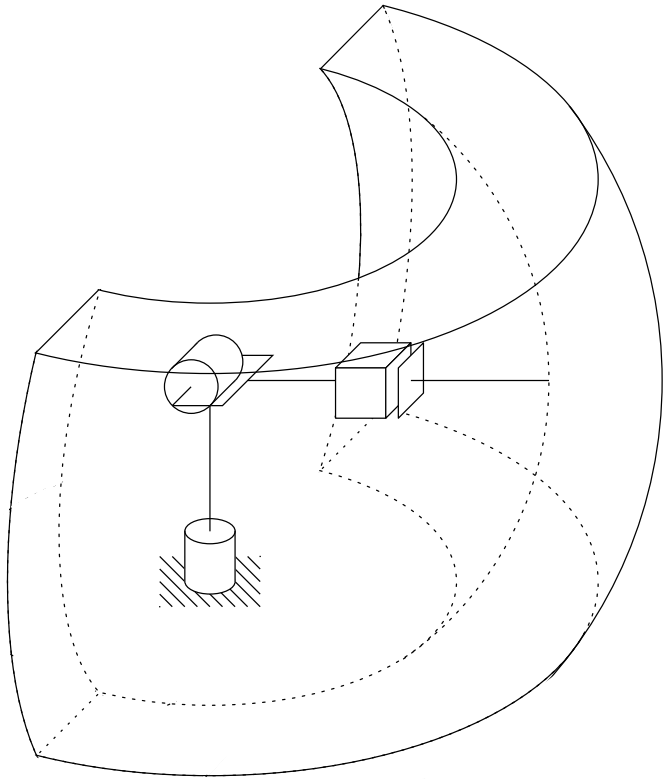






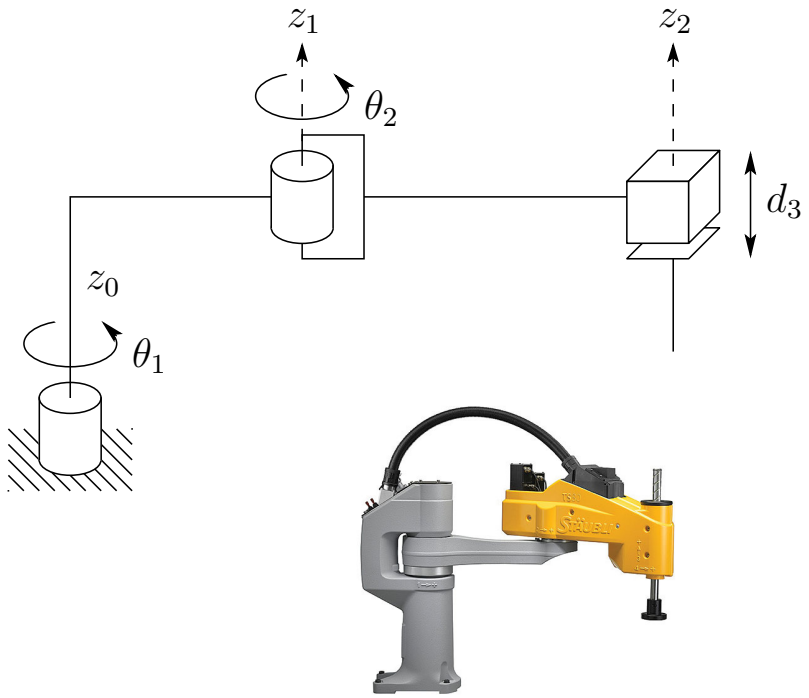
Common Robots - Spherical

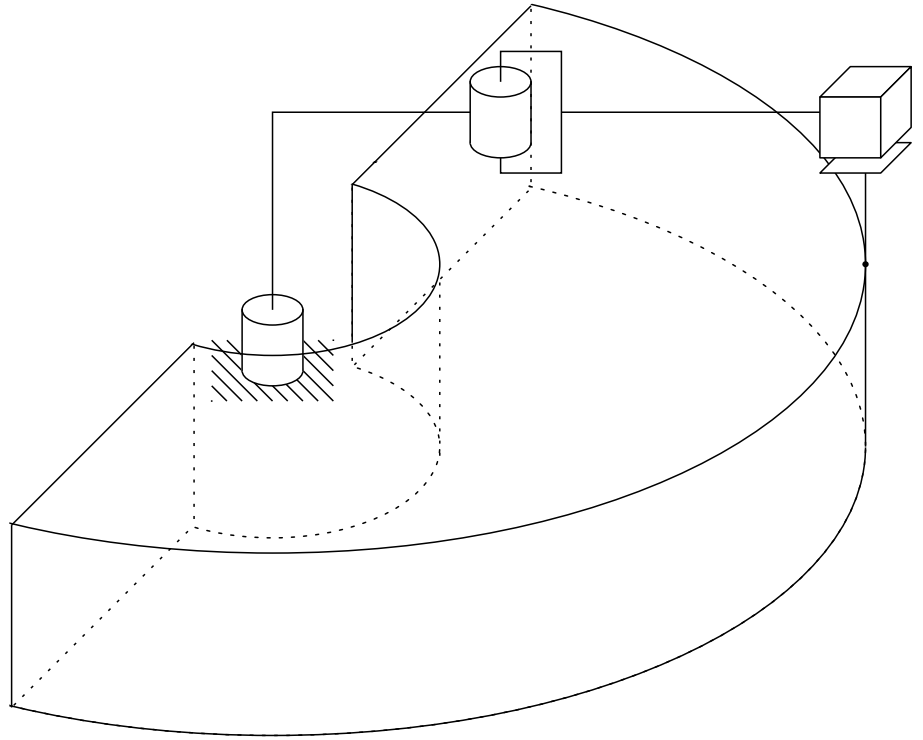




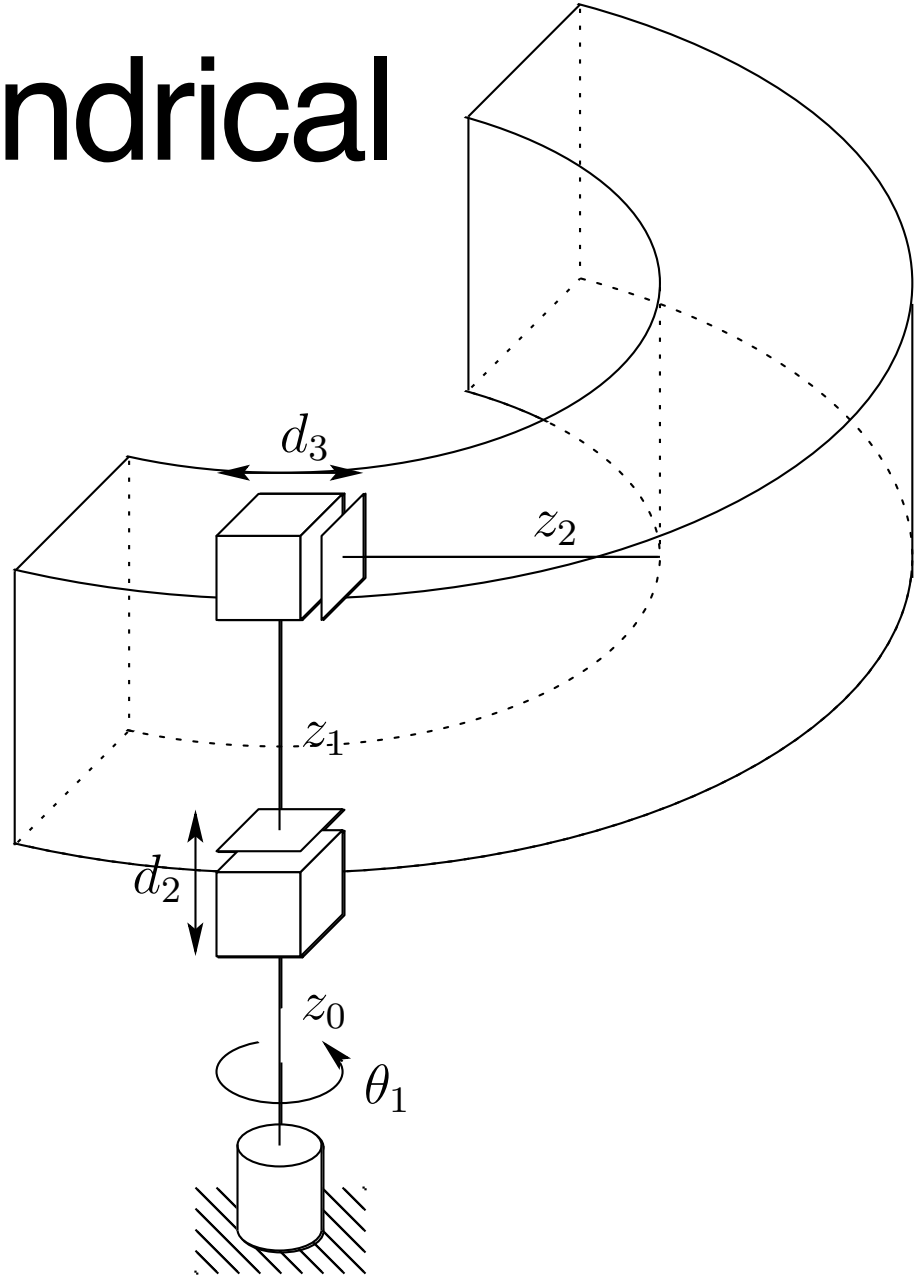
Common Robots - SCARA

- Selective Compliant Articulated Robot for Assembly

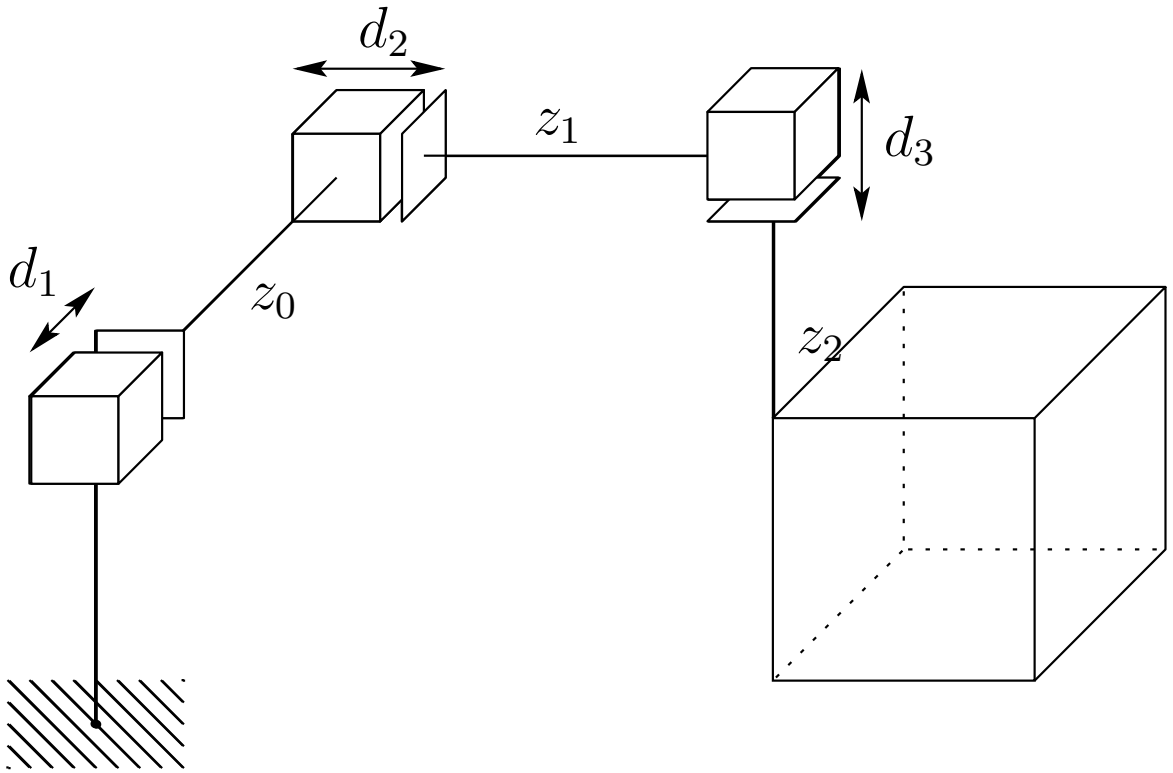




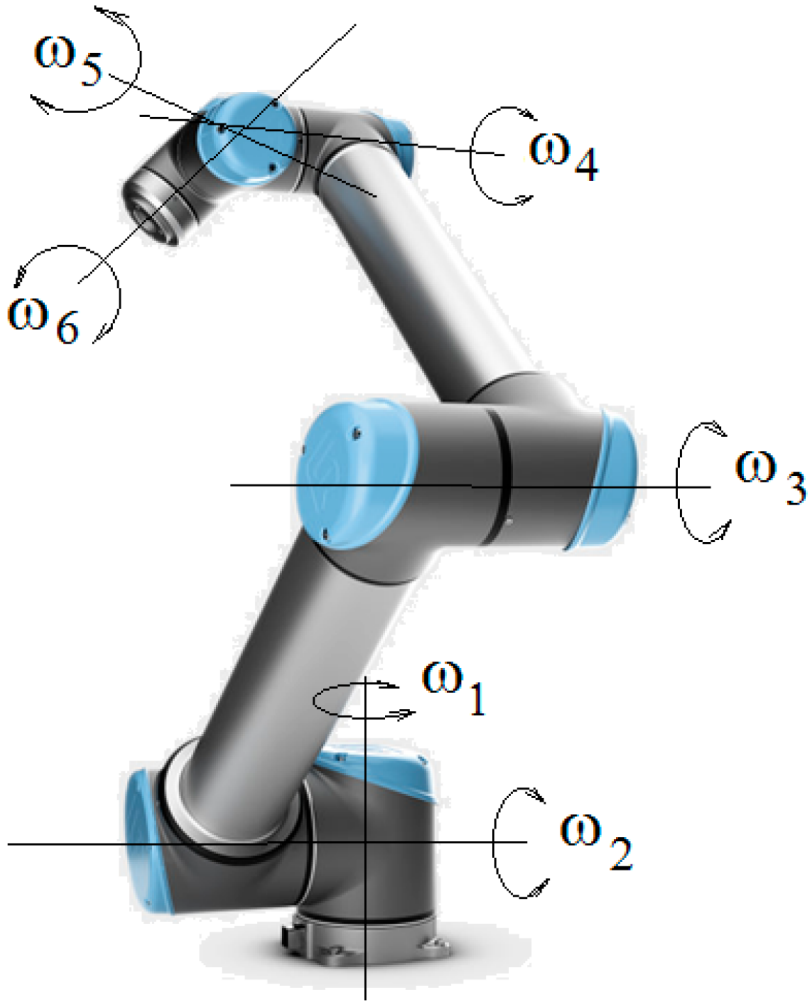
Cylindrical

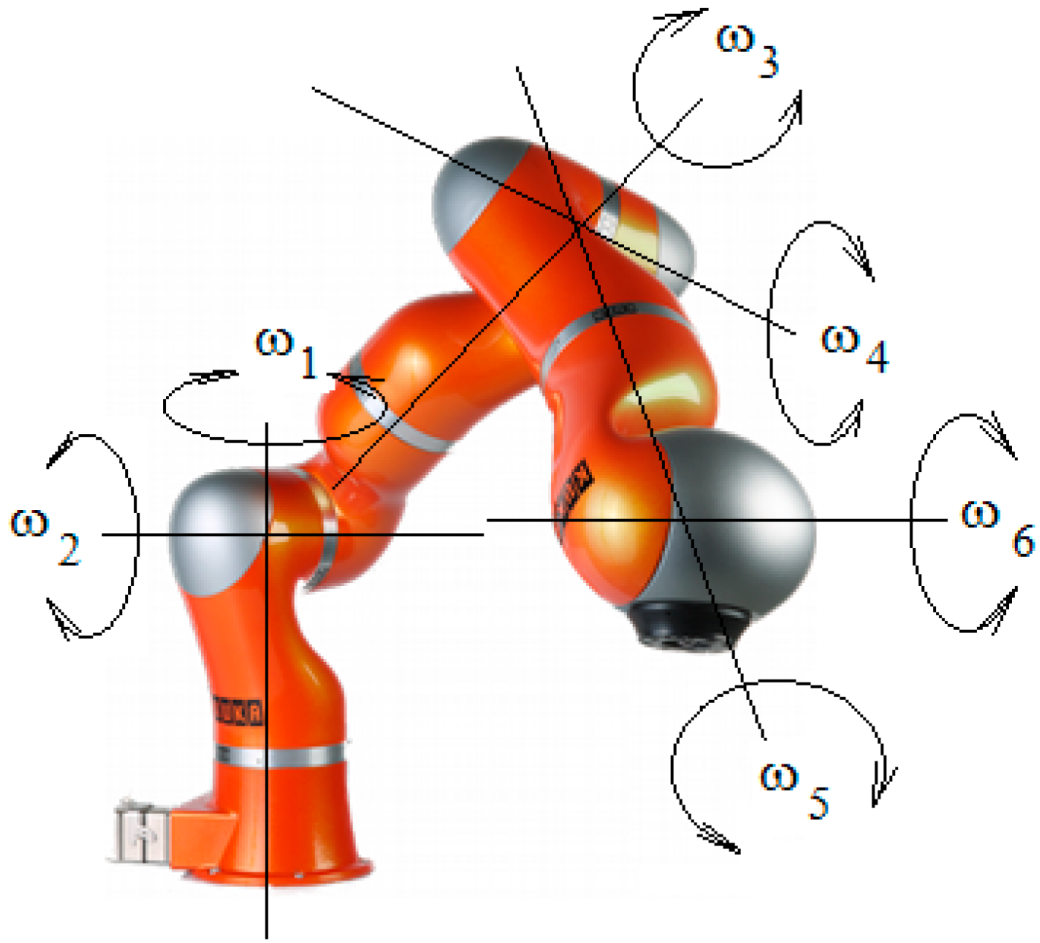


Common Robots - Cartesian



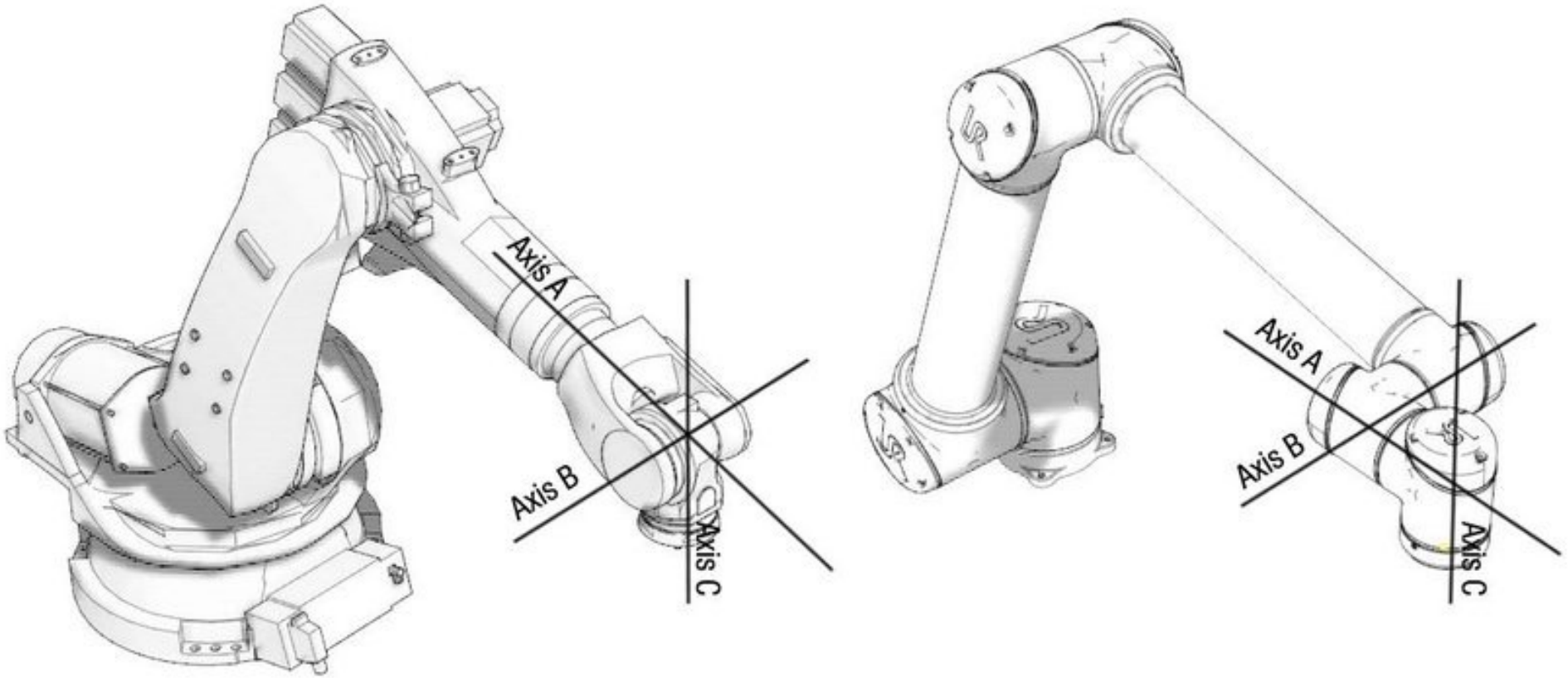
Collaborative Robot Arms





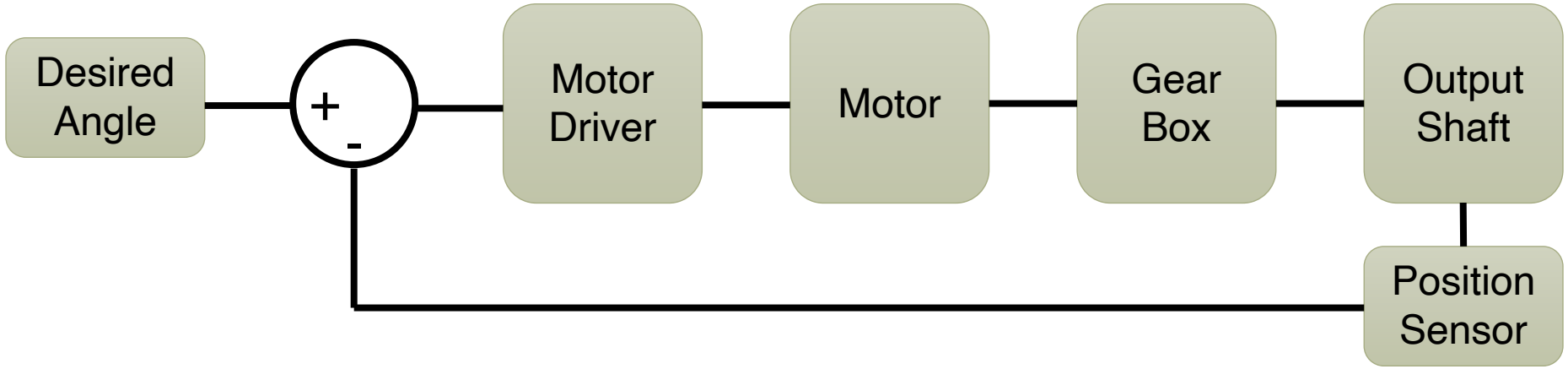
Spherical Wrists

- 3DOF wrist gives orientation
- Decoupled from position



Servo Motor

- A servo measures the absolute rotational position of the output shaft
- Uses a feedback loop to control position of the motor
- Advanced servos have more advanced feedback loops to control angular velocity and angular acceleration.



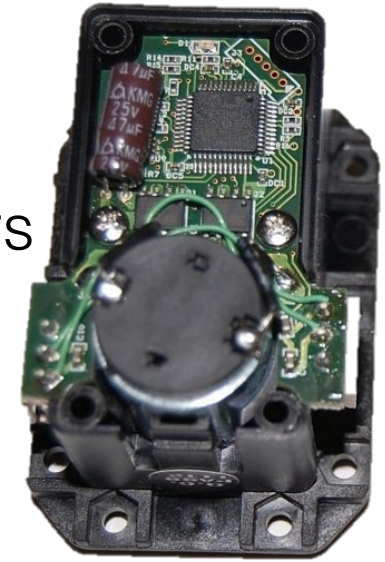
Servo Motor Internals

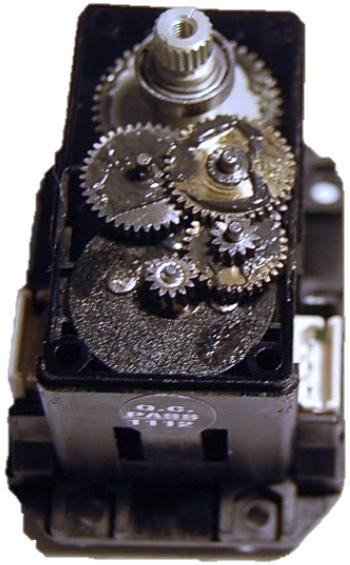
ARM M3 Processor

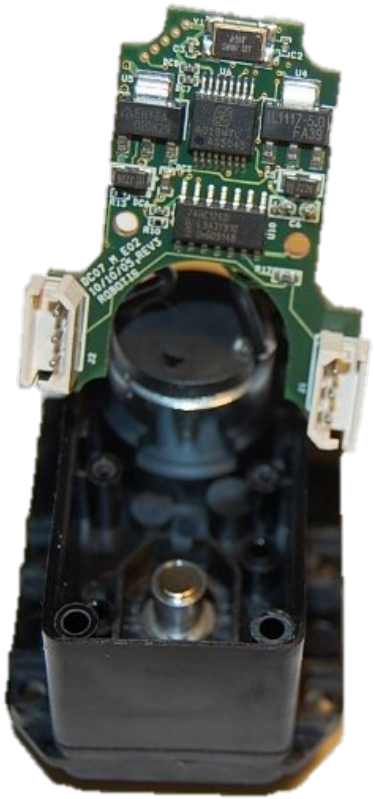
Output shaft

Magnetic Encoder

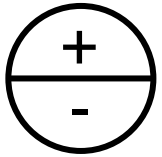
Motor Drivers







Magnet



Maxon DC motor

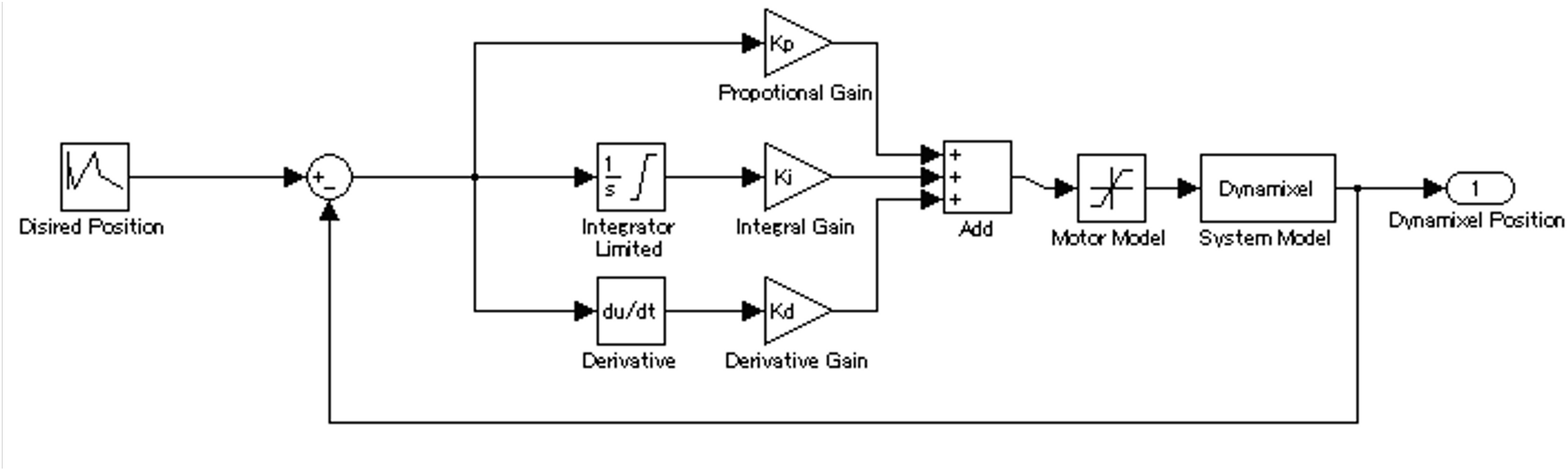
Gear box (193:1)

Dynamixel Motors – XL, XM

	XL320	XL430	XM430
			
Stall Torque	0.39 N.m	1.4 N.m	4.1 N.m
No Load Speed	114 rpm	57 rpm	46 rpm

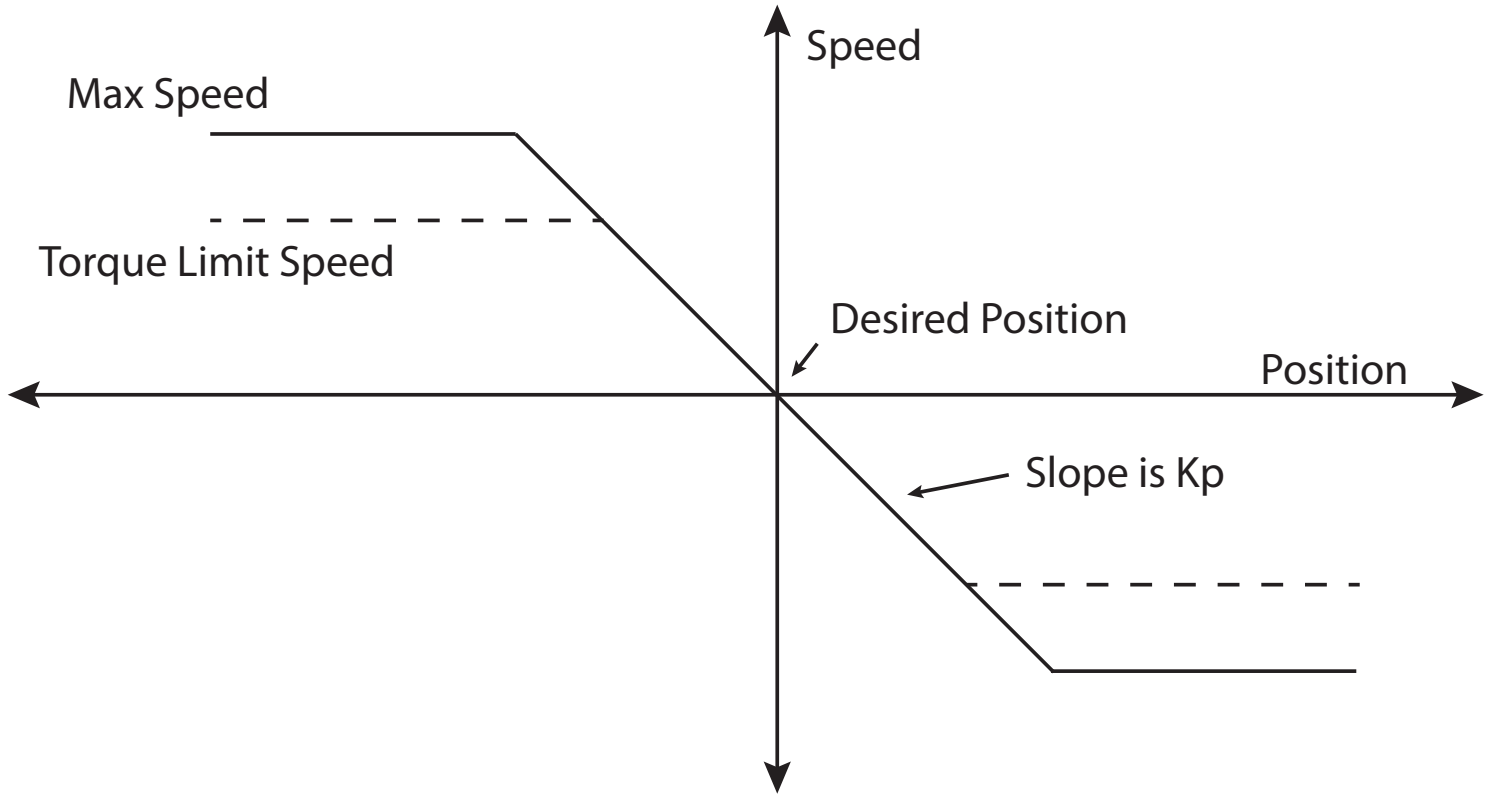
Position Controller

- Optional PID controller
- Default control is proportional control only



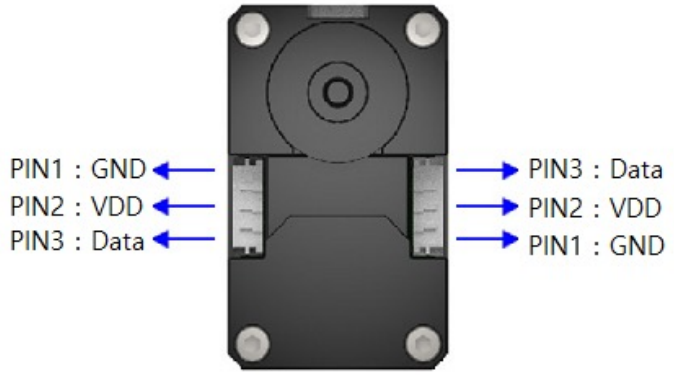
Control Inputs

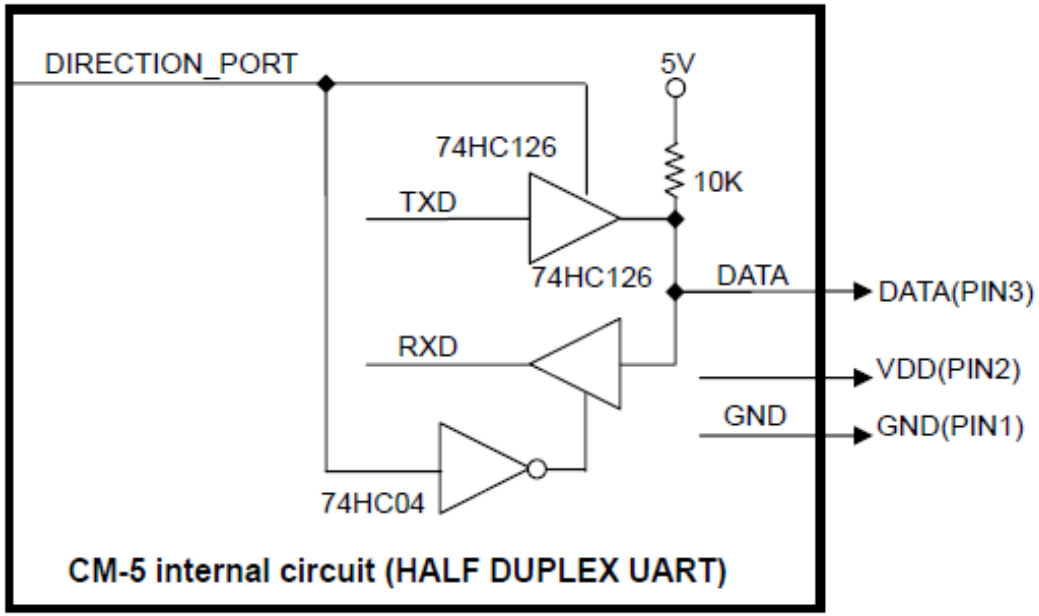
- Desired angle
- Desired (max) speed
- Max torque



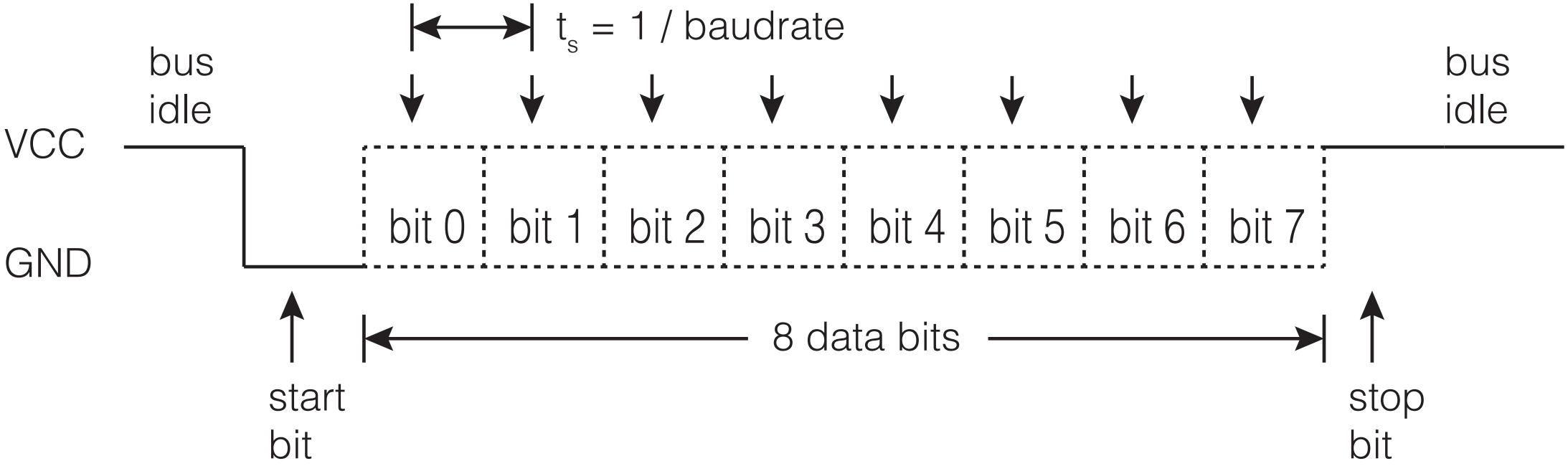
Dynamixel Serial Interface

- Dynamixel use a standard asynchronous serial protocol
- Sender & Receiver agree on transmission rate (Baud rate)
- This protocol is similar to RS-232, but is half-duplexed and uses a single wire for both transmit (TX) and receive (RX)
- Uses fixed rate of 1Mbaud, 8 bit transmissions, 1 stop bit, no parity bit





Asynchronous Serial Communications



Dynamixel Serial Messages Protocols

1.0

Command: | 0XFF | 0XFF | ID | LENGTH | INSTRUCTION | PARAMETER1 | ··· | PARAMETER N | CHECK SUM |

Status: | 0XFF | 0XFF | ID | LENGTH | ERROR | PARAMETER1 | PARAMETER2 | ··· | PARAMETER N | CHECK SUM |

2.0

Command:

Header			Reserved	Packet ID	Packet Length		Instruction	Parameter			16bit CRC	
0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	Instruction	Parameter1	…	ParameterN	CRC_L	CRC_H

Status:

Header			Reserved	Packet ID	Packet Length		Instruction	Error	Parameter			16bit CRC	
0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	0x55	ERROR	Param1	…	ParamN	CRC_L	CRC_H

Information on Dynamixel: http://emanual.robotis.com

Resources

- Data / Info:
  - https://emanual.robotis.com/
- Configuration & Testing:
  - Dynamixel Wizard 2.0

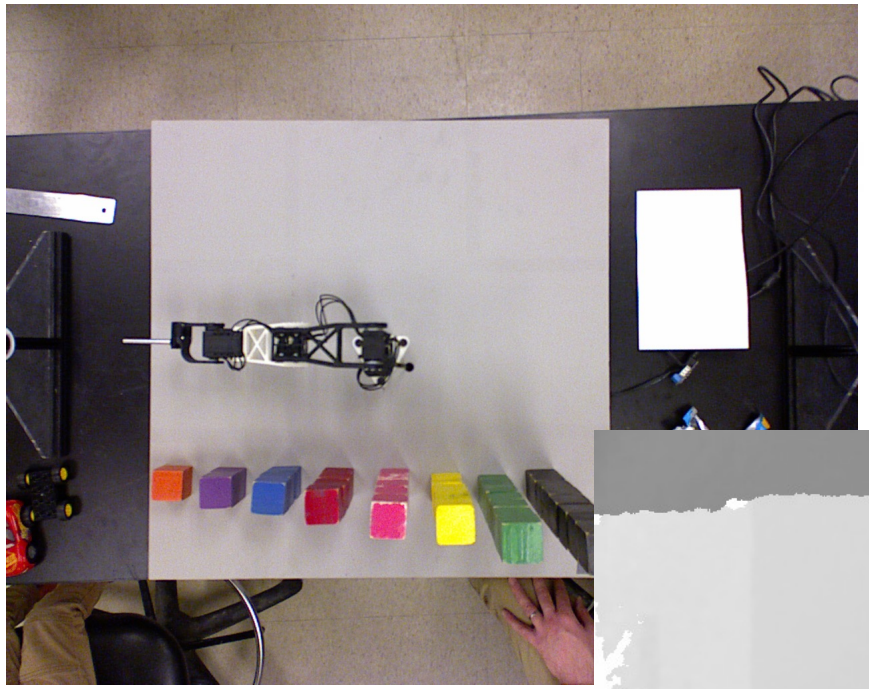
RX200 Arm

- 7 Dynamixel servos
- 5 DOF
- 55cm reach
- ±2.5mm accuracy
- ROS driver
- Python interface for position control

Depth Camera

Intel RealSense Lidar Camera L515

- Up to 1024 × 768 active TOF depth resolution
- Up to 1920 × 1080 RGB resolution
- Depth Diagonal Field of View over 70° × 55°
- 30 FPS depth & RGB streaming
- Range 0.25m to 9m
- Inertial Measurement Unit (IMU) for 6 degrees of freedom (6DoF) data



Overview of Armlab

- Kinematics
  - Find pose given angles and angles given pose
- Trajectory Following
  - Given set of configurations, smoothly move between them
- Computer Vision
  - Detect & differentiate blocks, find 3D location in workspace
- Trajectory Planning
  - Plan a path to pick up and place blocks

Intro to Armlab Code

Lecture 2

Fall 2022

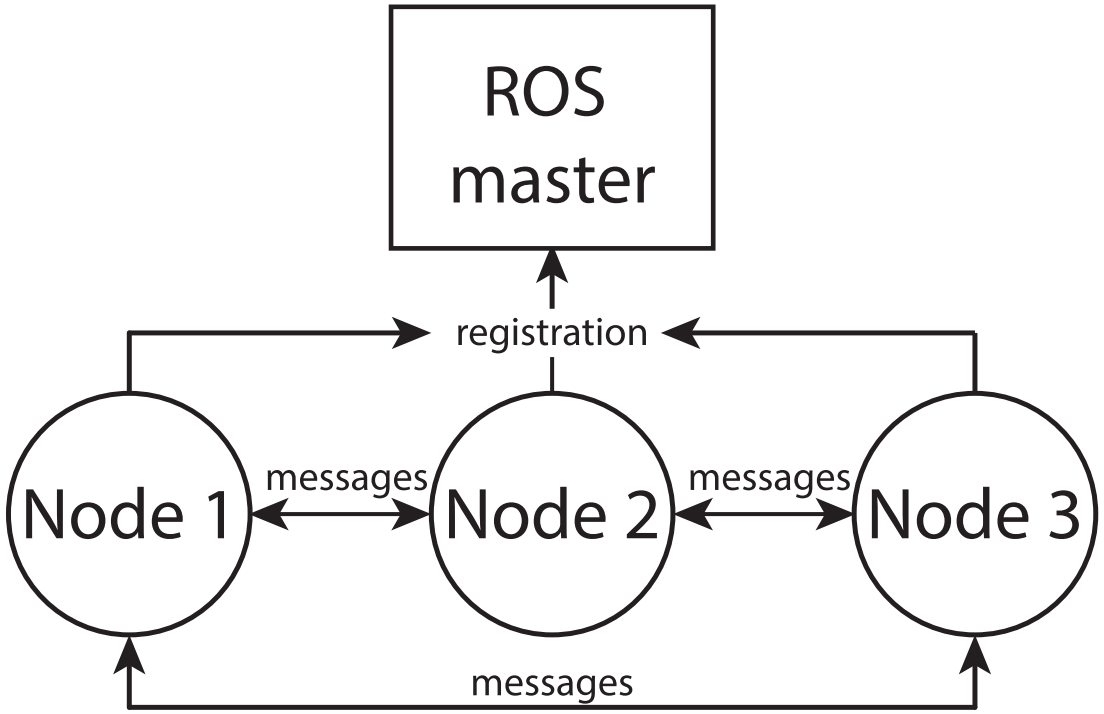
M | ROBOTICS

ROS Core Concepts

- Nodes
- Master
- Parameter Server
- Messages
- Topics
- Services
- Bags
- Documentation: http://wiki.ros.org/

Nodes

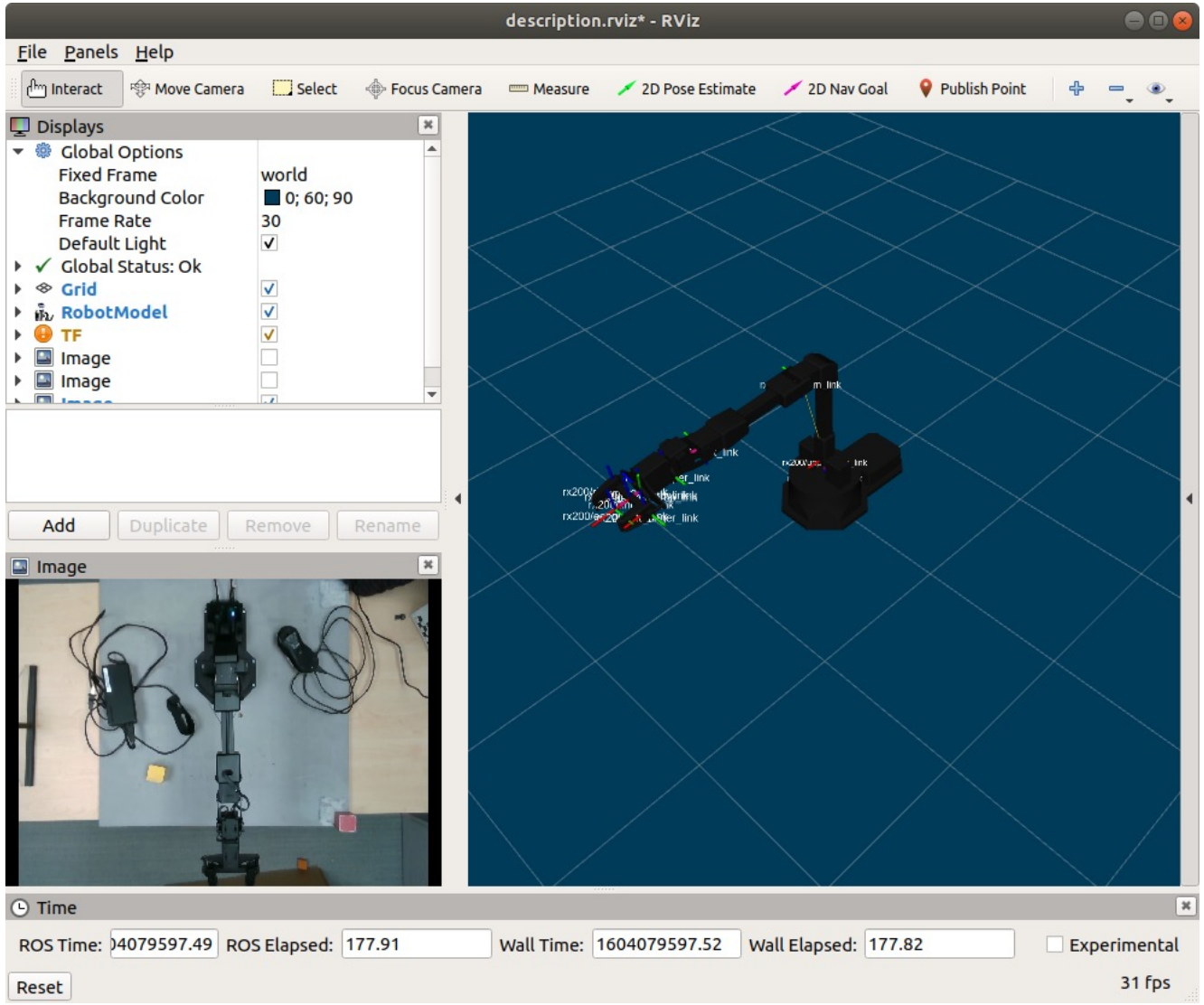
- A node is a single executable
- Hardware driver / application / visualization /
- Nodes communicate with messages
- Nodes register with rosmaster, a central marshalling service



Topics and Messages

- Publish/Subscribe model
- Like channels & messages in LCM
- Strongly typed
- Standard & Custom messages
- Uses TCP transport and central marshalling service roscore
- Example: receive joint postions from arm
  - Subscribe to **/rx200/joint_states**
  - get messages of type **sensor_msgs/JointState.msg**

RViz



- Visualizes some messages
- Model of robot arm automatically launched

Services

- Request/Reply interactions
- defined by a pair of message structures: one for the request and one for the reply
- Example: turn on/off torque for robot arm servos

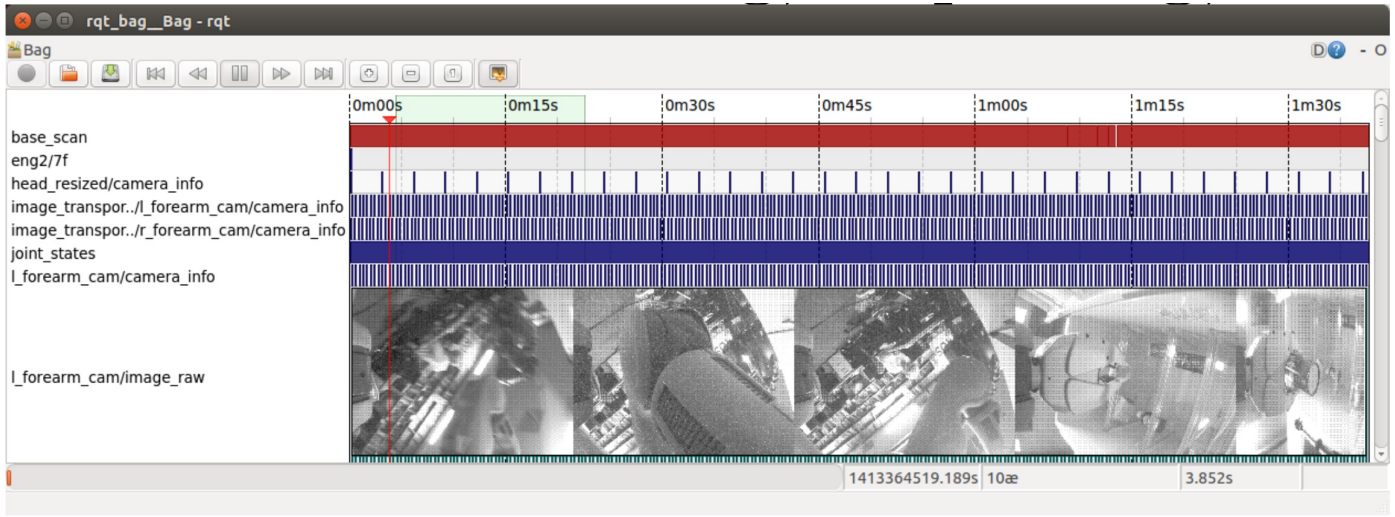
rostopic

- Get info about a topic and publish messages on a topic

Command	
$rostopic list	List active topics
$rosnode echo /topic	Prints messages of the topic to the screen
$rostopic info /topic	Print information about a topic
$rostopic type /topic	Prints the type of messages the topic publishes
$rostopic pub /topic type args	Publishes data to a topic

rosbag

- rosbag record –a
- rosbag play <your bagfile>
- Also rqt_bag



Control Station

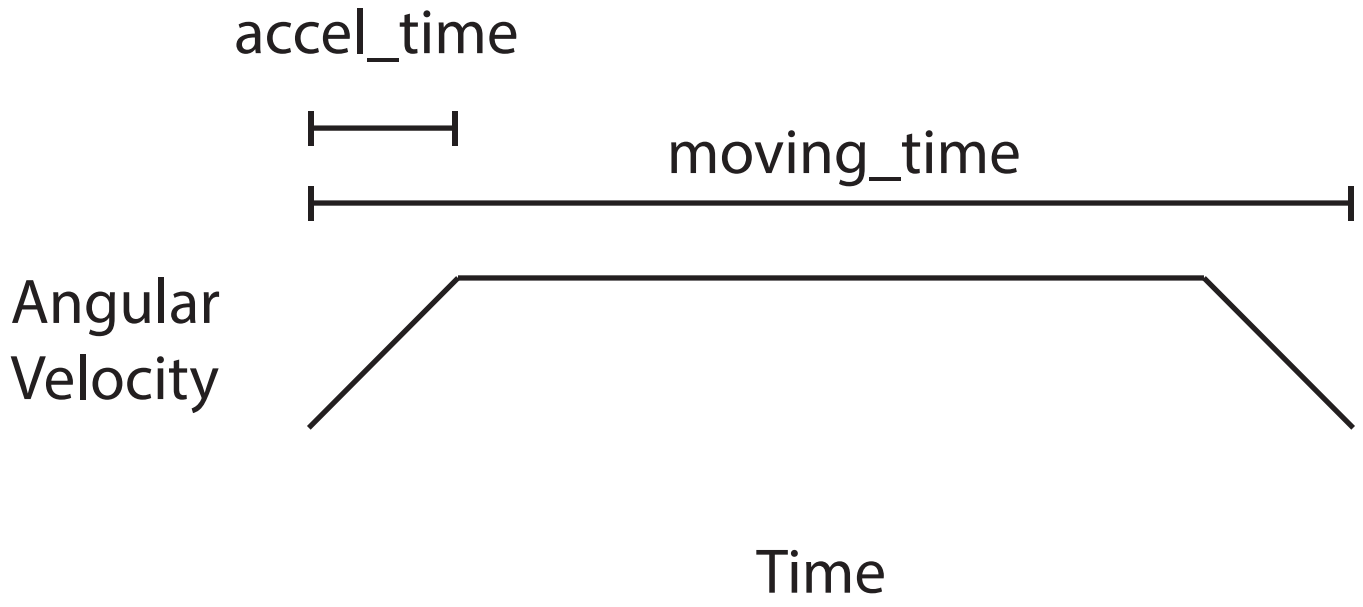
- pyQT4 GUI
- Control RX200 Arm
- Subscribes ROS messages for camera & arm
- Can modify with QTCreator
- Compile python gui code with pyuic4.

interbotix_sdk

- Driver communicates with servo motors, sends commands & gets feedback
- We will use time-base profile & pwm gripper mode
- Launch with launch_armlab.py or
- `roslaunch interbotix_sdk arm_run.launch robot_name:=rx200 use_time_based_profile:=true gripper_operating_mode:=pwm`

Time Profiles

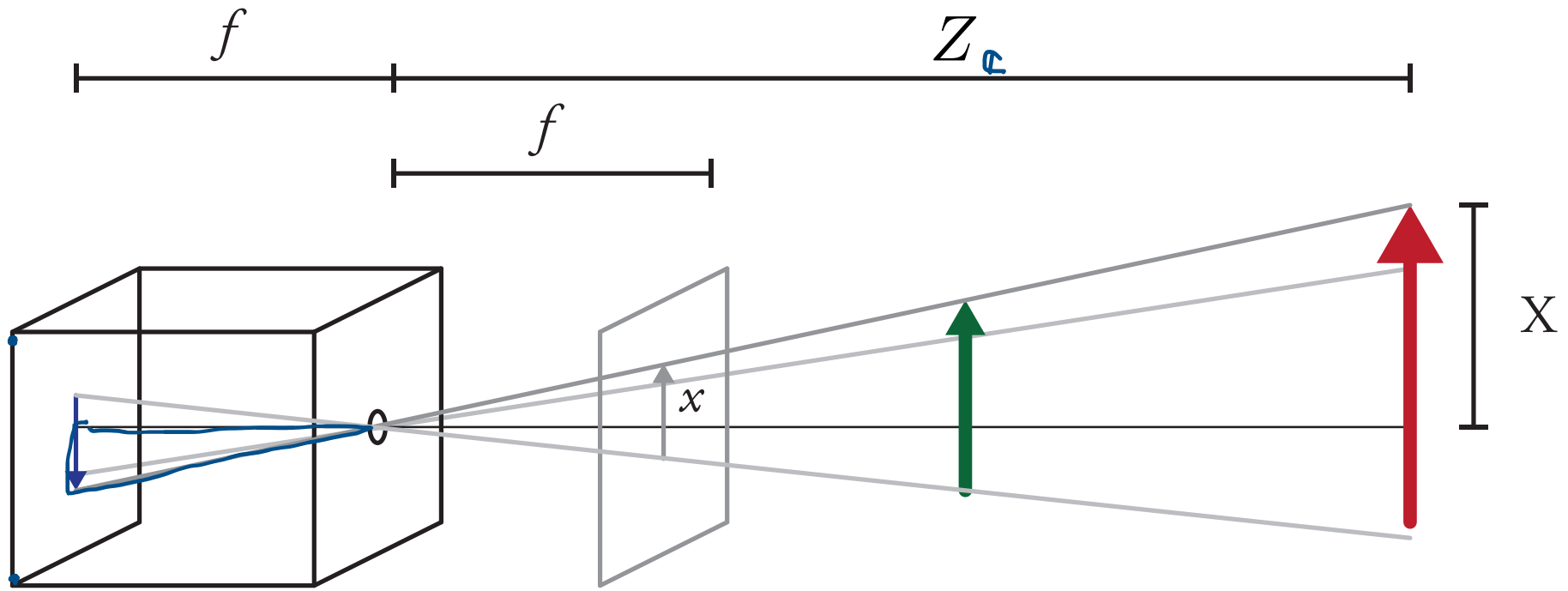
- Servos have built-in trajectory control
- Trapezoidal velocity profile
- `set_joint_positions(angles, moving_time = X, accel_time=Y)`



realsense2_camera

- Publishes RGB and depth images to /camera/* topics
- We will use registered depth frame
- Launch with launch_armplab.py or
- `roslaunch realsense2_camera rs_camera.launch align_depth:=true`
- Reconfigure gains/White Balance with `rqt_configure`

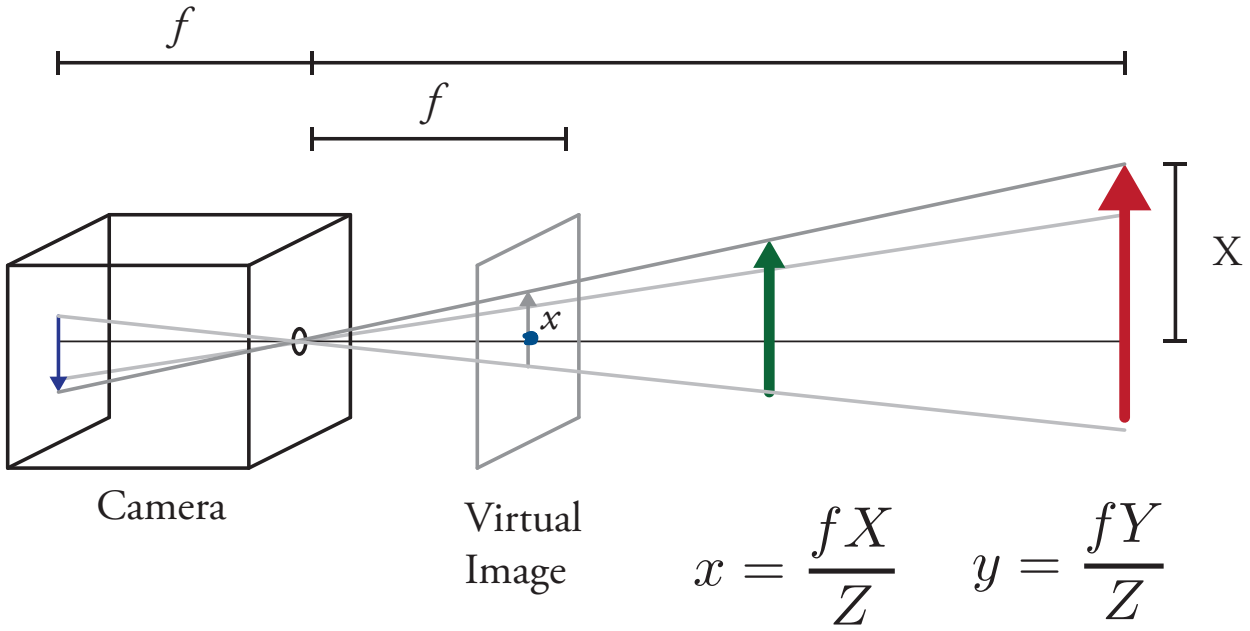
Pinhole camera model



$$\frac{x}{f} = \frac{X}{Z}$$

$$x = \frac{fX}{Z} \qquad y = \frac{fY}{Z}$$

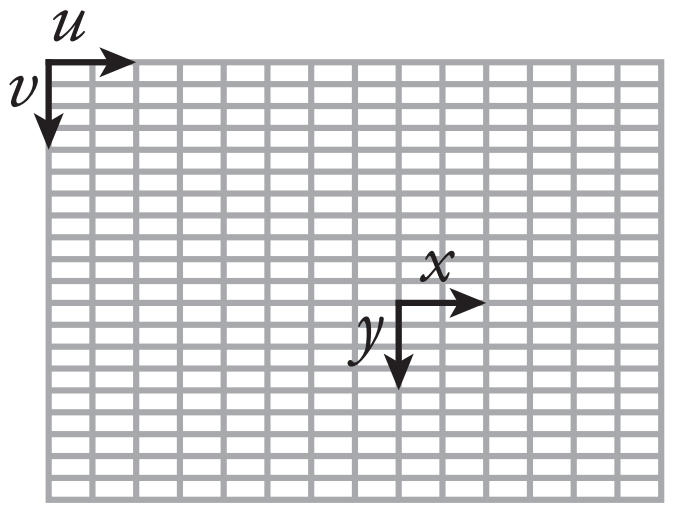
Pnhole model, matrix form



$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \\ 1 \end{bmatrix}$$

Other Considerations…

- Optical axis not centered on sensor
- Pixel coordinate system origin in corner of sensor
- Images have discrete pixels
- Pixels might not be square
- Don’t know pixel size



Camera intrinsic matrix

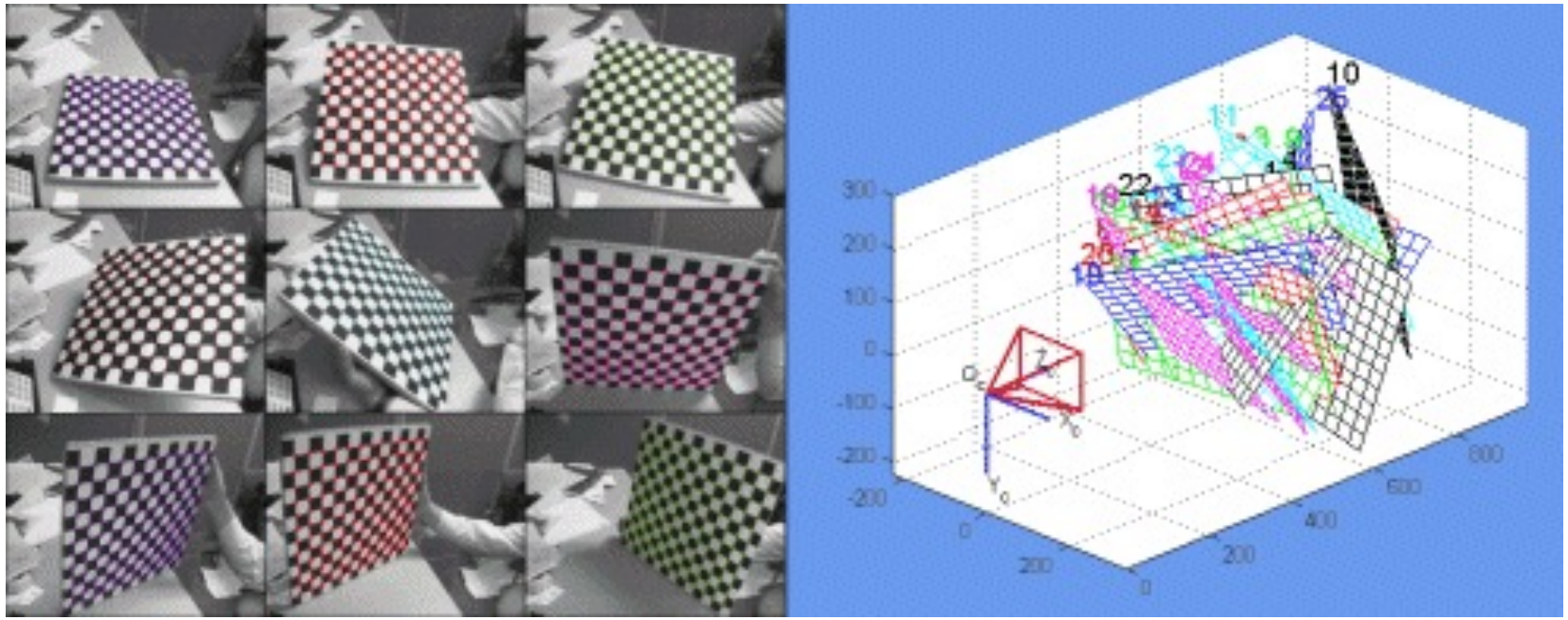
- Transforms Camera Frame to Image Frame
- Contains sensor scale information (meters to pixels)
- Depends on geometry of camera, physical characteristics of sensor, details of the image digitization

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_c} \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & | & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

Finding the intrinsic matrix

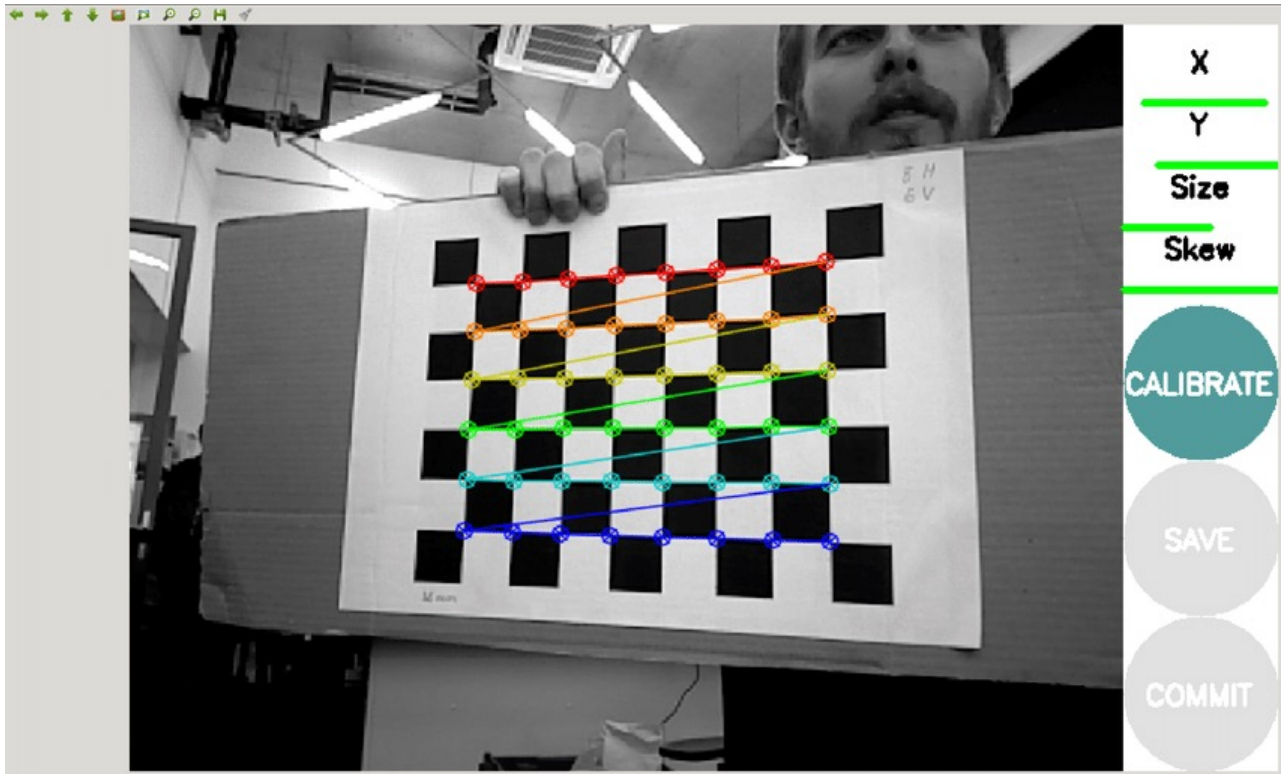
- Must perform camera calibration
- Collect many images of known points and compute intrinsic matrix that minimizes reprojection error



Jean-Yves Bouguet

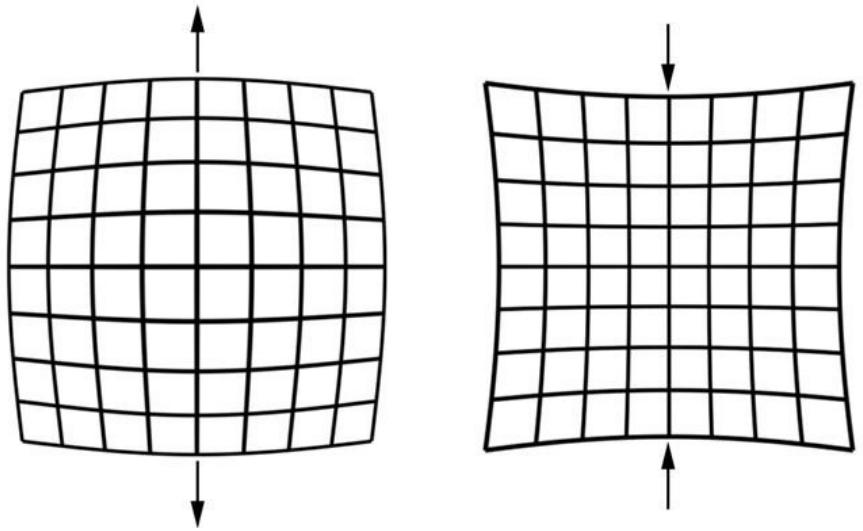
camera_calibration ROS node

- Pre-installed ROS node will find camera parameters
- Move checkerboard around until you have good coverage of workspace volume



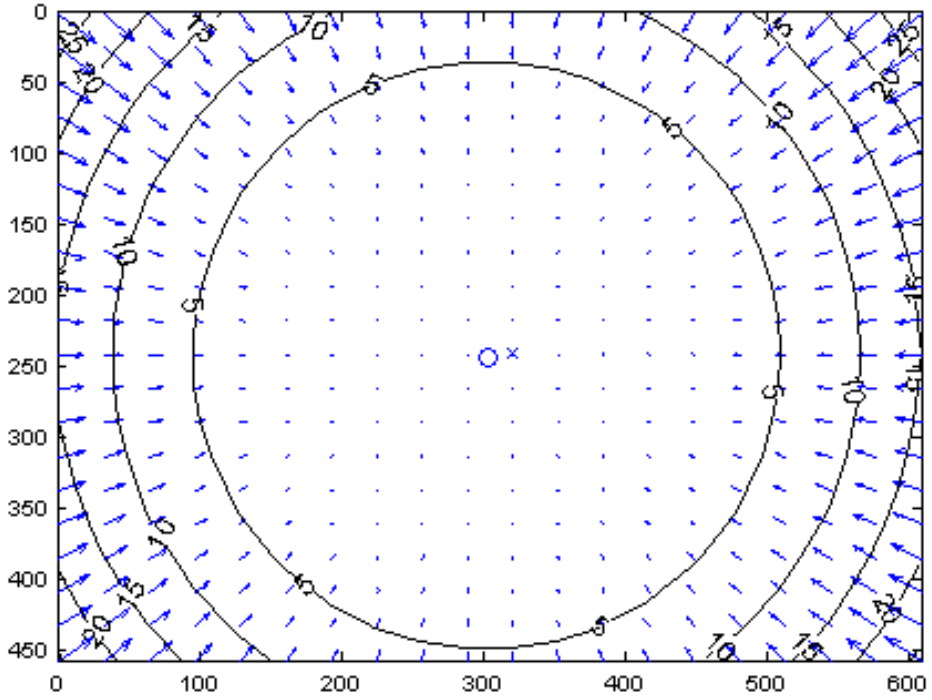
Lens Distortion

- Real cameras can have distortion
- Can correct with polynomial model



Barrel distortion

Pincushion distortion



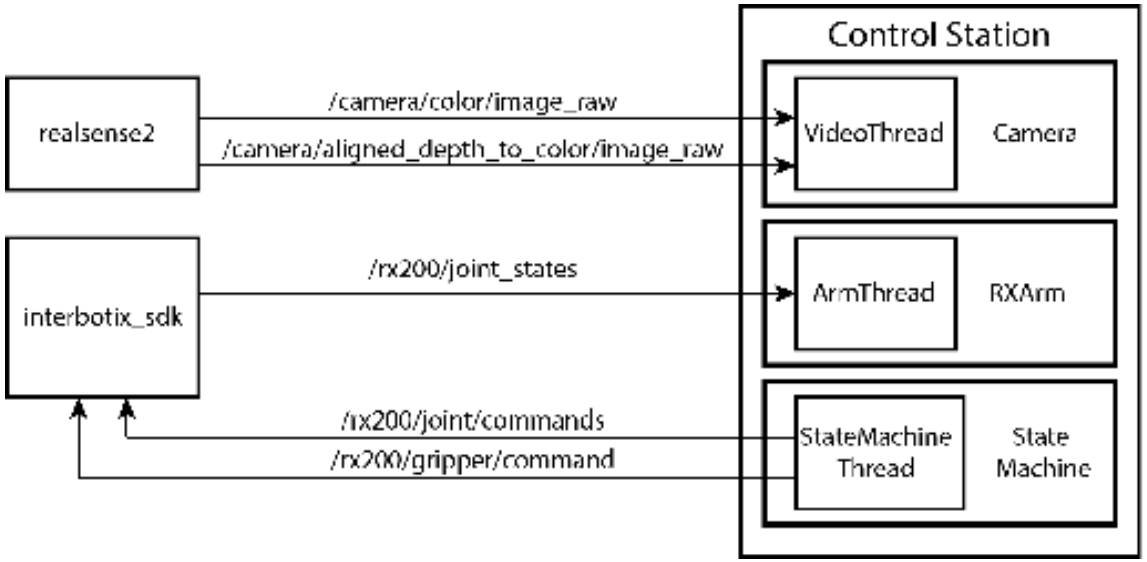
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_d \\ v_d \end{bmatrix} (1 + k_1 r^2 + k_2 r^4)$$

ROS /camera_info

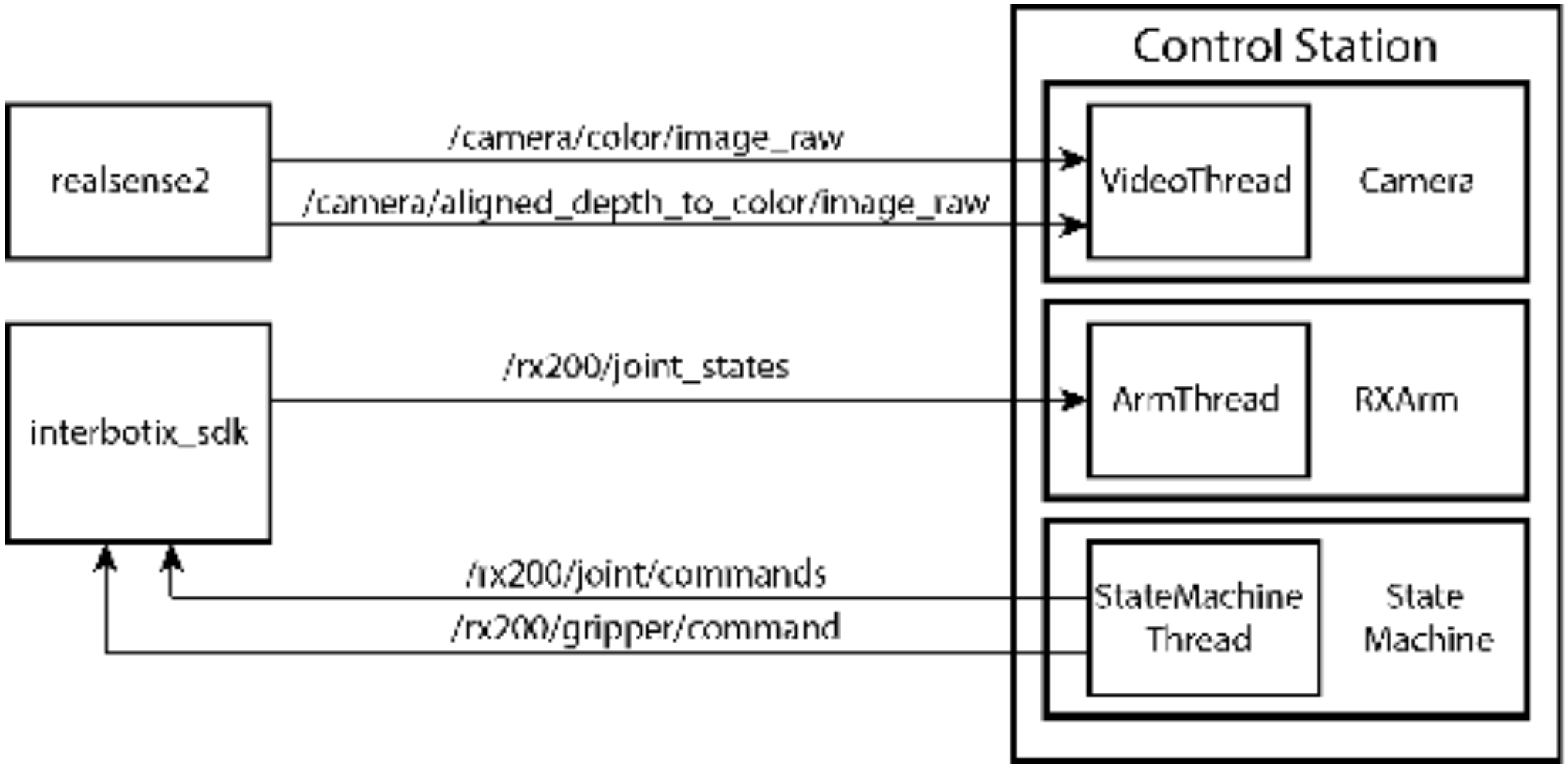
- The Realsense node publishes the intrinsic matric on the camera_info topic.
- This intrinsic matrix comes from factory calibration.
- Factory calibration is typically slightly different than we can measure with checkerboard in lab.
- Does not include distortion parameters, but these may be minimal.

armlab

- control_station.py
  - GUI Class
    - RXArm Class
      - ArmThread
      - Kinematics
    - Camera Class
      - VideoThread
      - CV Functions
    - StateMachine Class
      - StateMachineThread



armlab


Control Station
realsense2
/camera/color/image_raw
/camera/aligned_depth_to_color/image_raw
VideoThread
Camera
interbotix_sdk
/rx200/joint_states
ArmThread
RXArm
/rx200/joint/commands
/rx200/gripper/command
StateMachine Thread
State Machine

armlab folders

- Can run camera.py and rxarm.py to test independently
- Other test scripts are included /test for you to edit
- /util contains camera calibration script (needs to be updated)
- /data contains config files you will populate for kinematics
- /dynamixel contains a copy of the serial comms lib for…
- reboot_all_servos script can reboot arm remotely (in a pinch)

Getting Started

- Get the code
- Get the arm moving
- Read the code to understand whats going on
- Write a simple waypoint follower
- Find the camera intrinsic matrix
- Implement “Teach and Repeat” functionality