POLE ROBOTIQUE P19 PROJECT S8

Tossing Robot

Daniel Stulberg Huf
Emilio Hajjar
Luan Rocha do Amaral

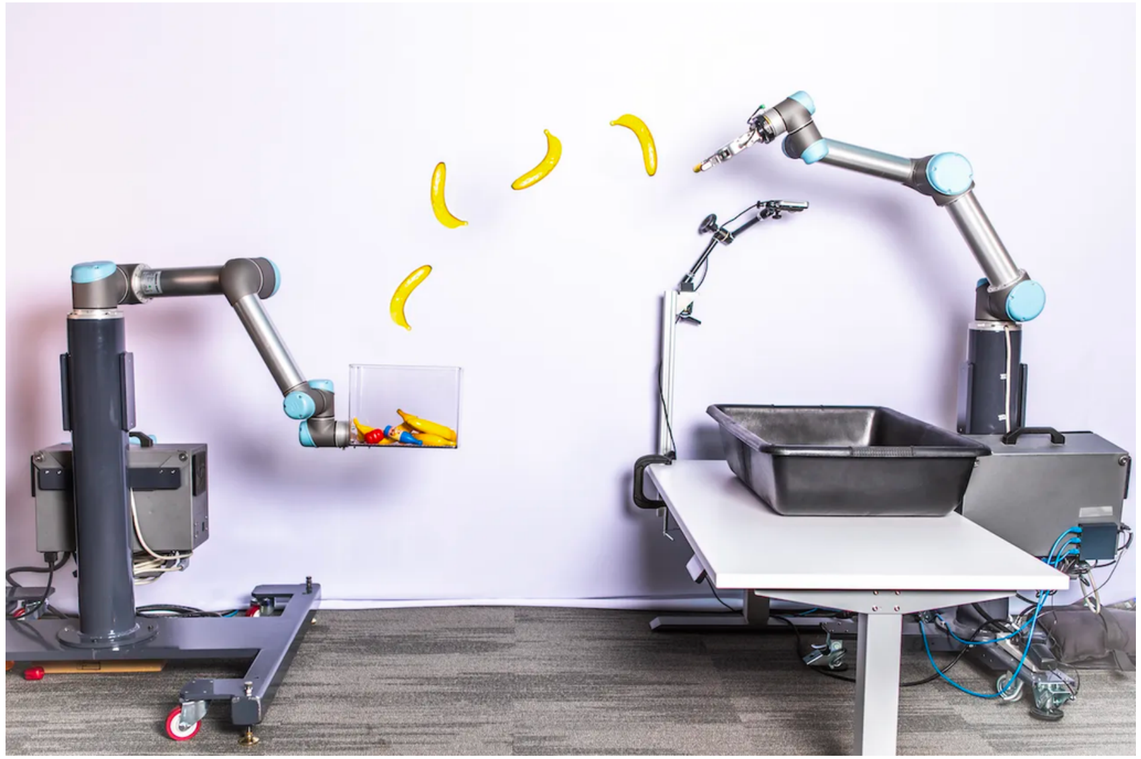
Marina Daumas Carneiro
Tomas Gonzalez Villarroel
Yuanhao Dong

Sumary

- Problem Statement
- Computer Vision
  - ArUco Markers for distance estimation
  - Cup Detection using AI
  - 3D Vision
- Control
  - Mathematical Modelling
  - Controller Design
  - Trajectory generation
- System
  - Simulation Environment
  - Subsystems and communication

Tossing-ball Robot

- Automating human tasks
- Interest for Industrial Applications



22/23 S8: Challenges

- Remote Work -> working with simulations and single camera
- 2D-3D Vision Performance improvements
- Trajectory optimization

- 3 main indicators:

1. Speed / Accuracy
2. Robustness
3. Precision

Computer Vision Techniques for Cup Distance Estimation

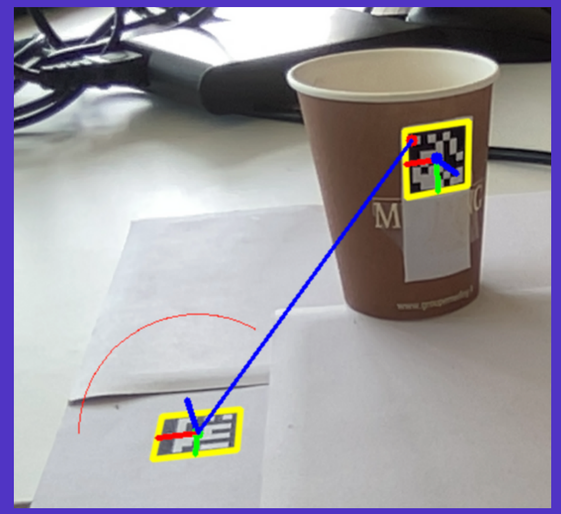
Division of CV tasks

- Remote Challenge
- 3D Vision Performance improvements

Cup Detection using a Monocular Camera

ArUco Markers

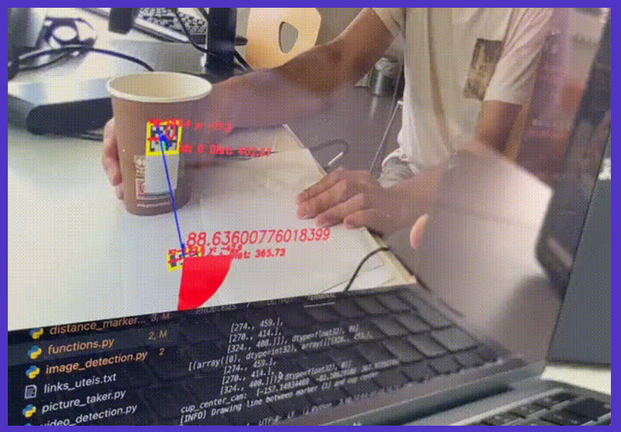
- Allow precise location and pose estimation of objects
- Estimate the Cup's center using perspective transformations



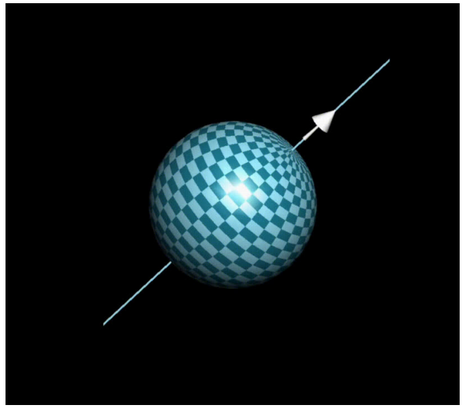
Cup Detection using a Monocular Camera

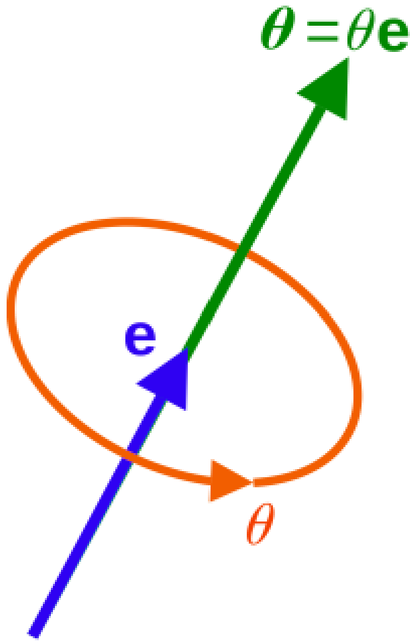
Pipeline

- Calibrate camera
- Detect markers
- Use perspective transformations to estimate the cup center
- Get distance and angle from the arm marker to the center of the cup



Perspective Transformation using ArUco Markers





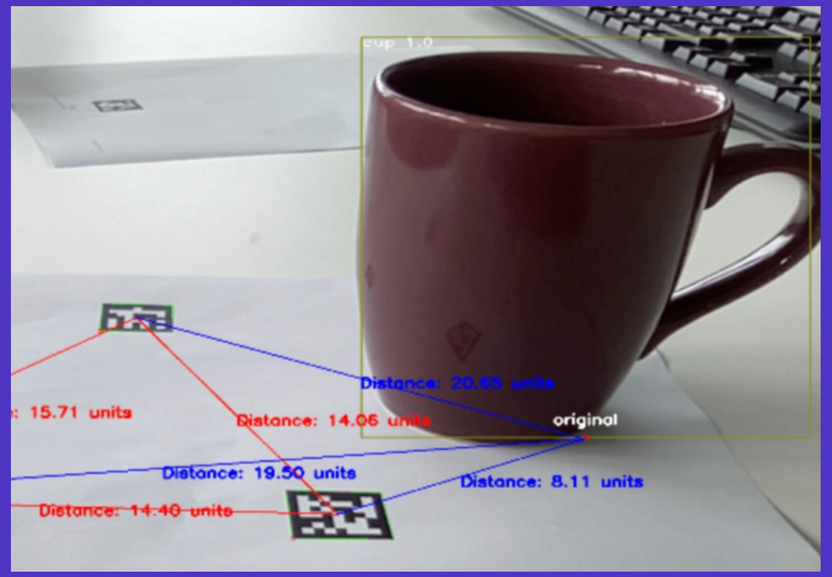
Camera-Marker vector

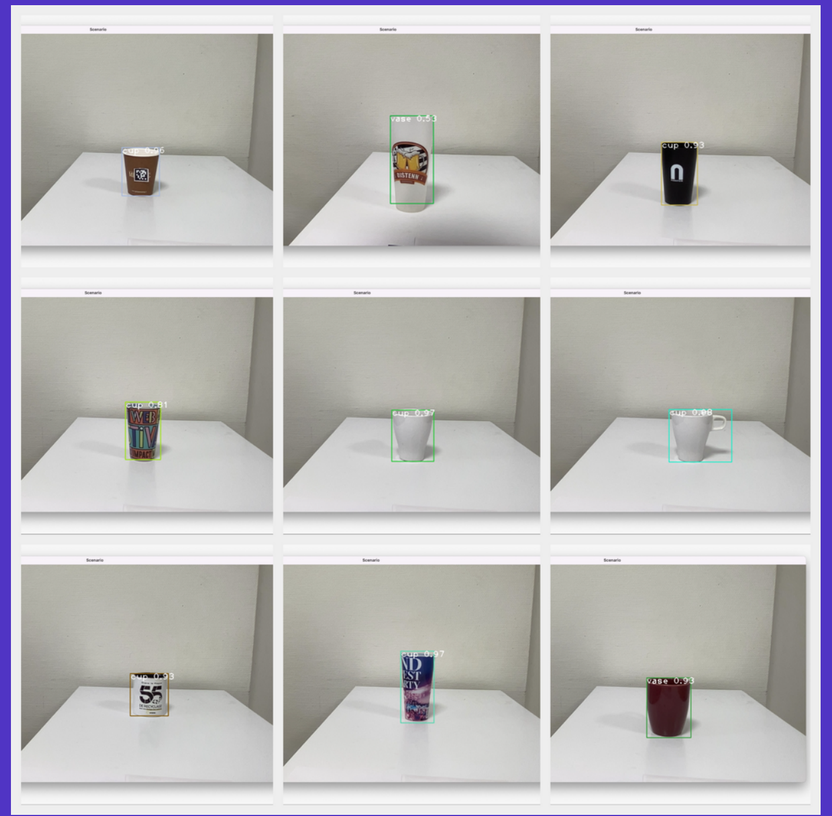
$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{cam} = \begin{bmatrix} {}^{cam}\mathrm{R}_{tag} & {}^{cam}\mathrm{t}_{tag} \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{tag}$$

Perspective Transformation between Camera and Marker

Cup Detection using YOLO Algorithm

- Allows for more freedom in the cup choice
- Combination of calculations (ArUco Marker + Bounding Box location)
- Worse performance due to depth ratio calculation
- Slower real-time performance (~4FPS vs ~29FPS)

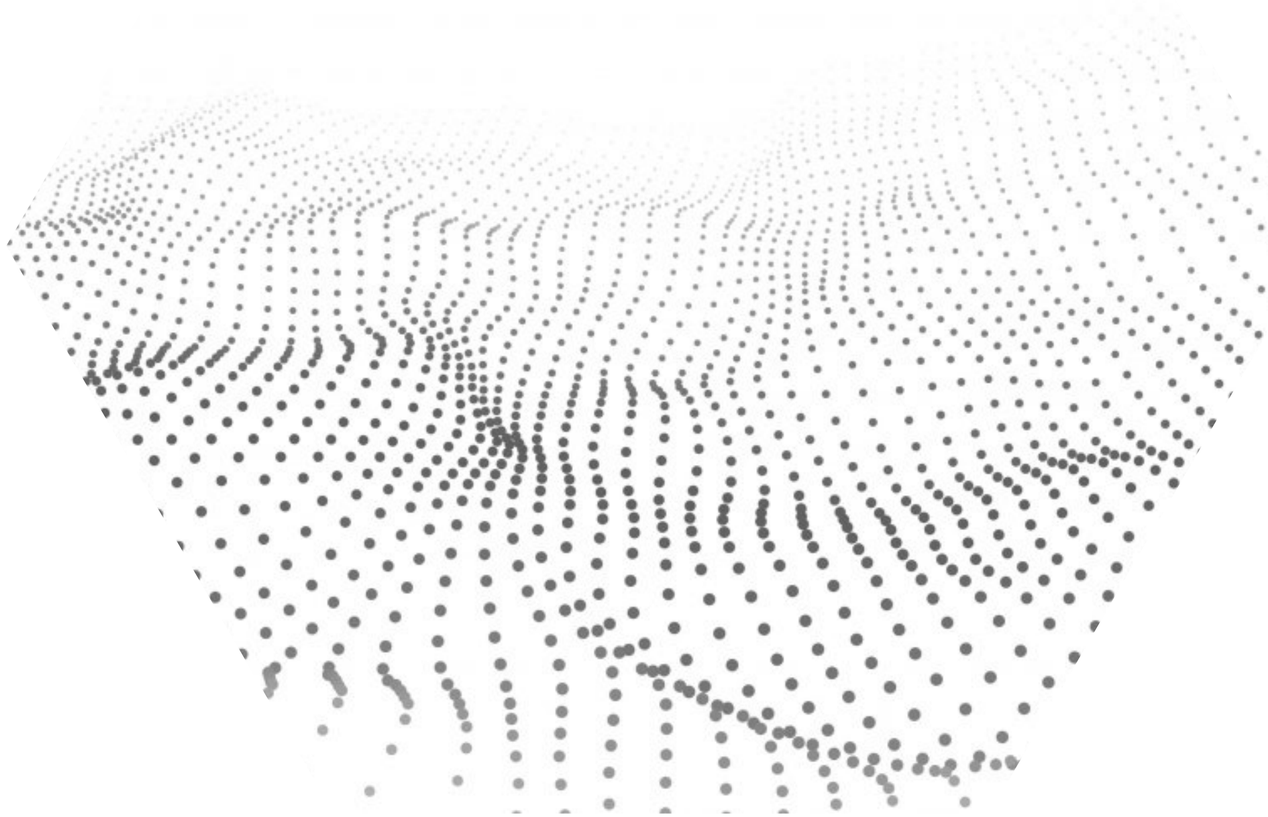




3D vision

Precisely locate the objects with depth camera

- 3D point cloud acquisition
- Object segmentation
- Position estimation



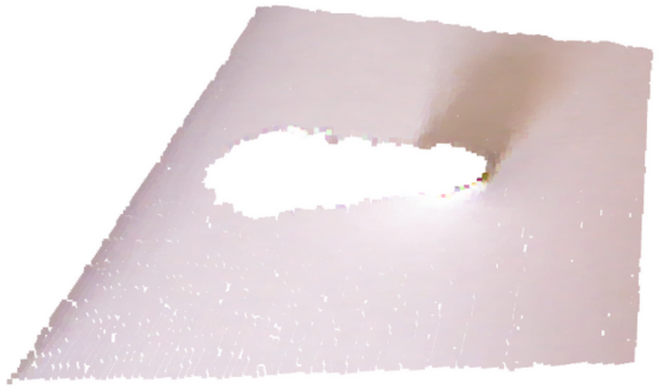
3D point cloud acquisition

3D point cloud of the scene

3D point cloud of the table

Object segmentation

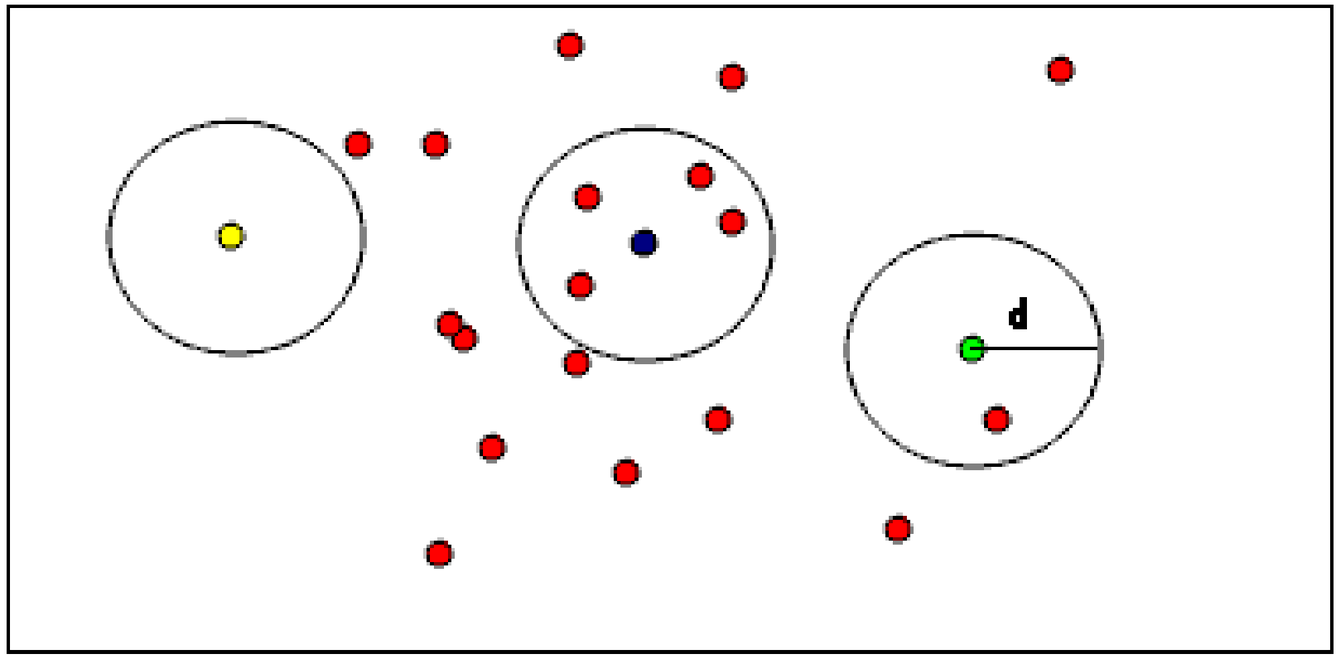
Using RANSAC(RANdom SAmple Consensus) algorithm to remove plane



Noise reduction

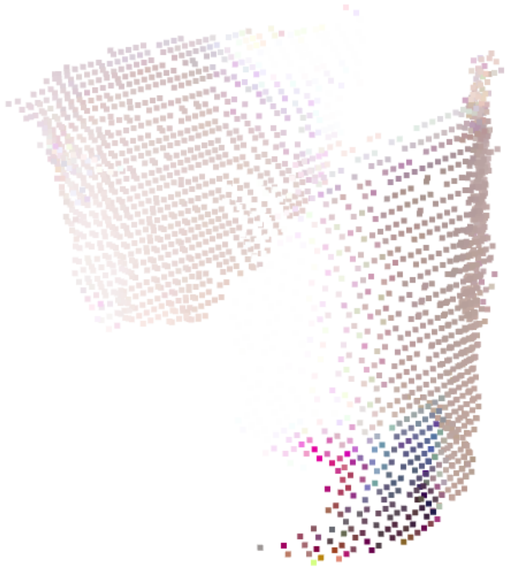
We delete the outlier points using radius filter

Filter(pcd, radius, num_neighbors)



Noise reduction

We delete the outlier points using radius filter

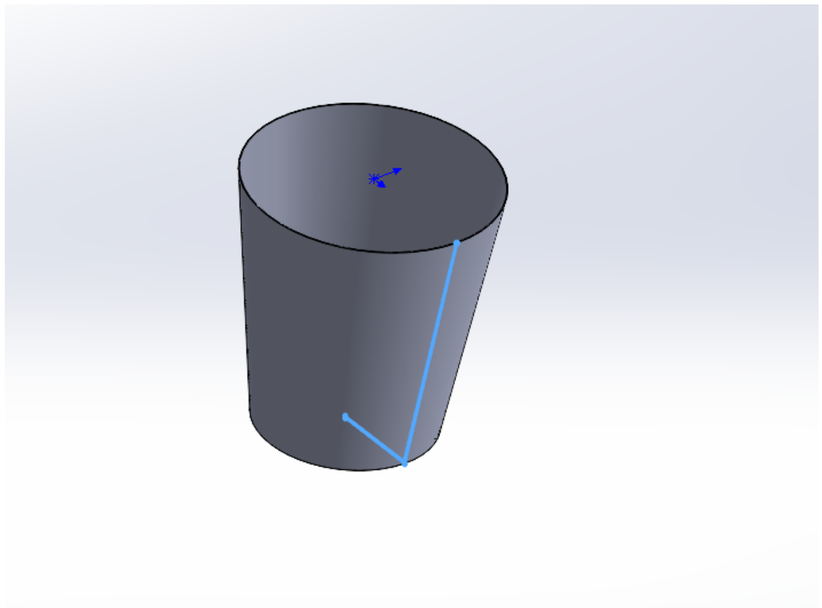


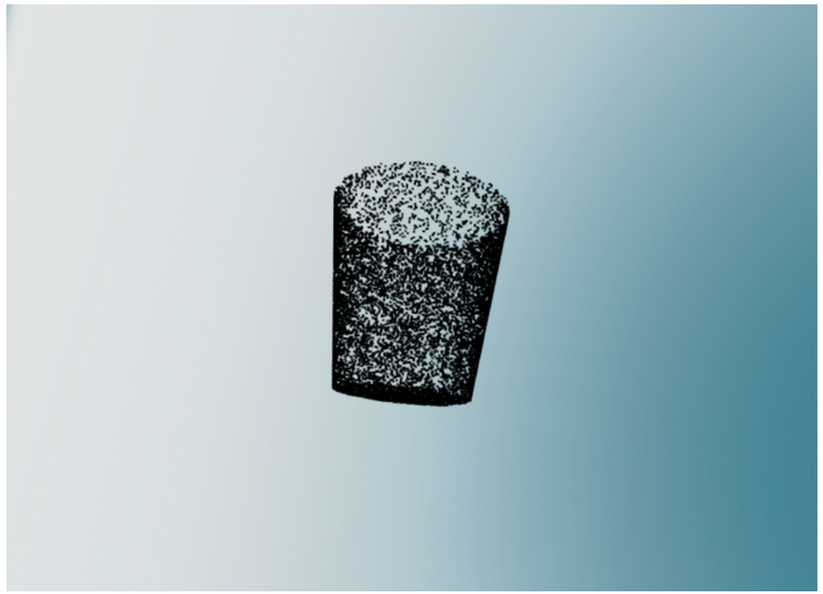
Now we have the point cloud of the cup, how to get the position of its center?

Position estimation

Idea: use a cup model to fit these points

$$X_{object} = RX_{model} + t$$



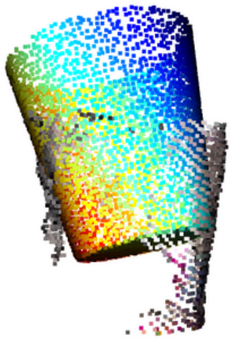


Position estimation

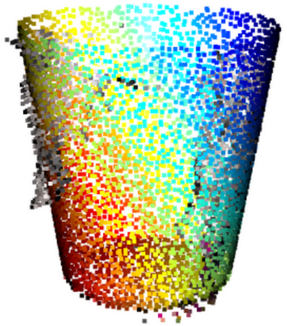
Registration - combination of RANSAC and ICP algorithms

- Firsr apply RANSAC (RANdom SAmple Consensus) algorithm for coarse matching
- Then apply ICP (Iterative Closest Point) algorithm for fine matching

Before



After RANSAC



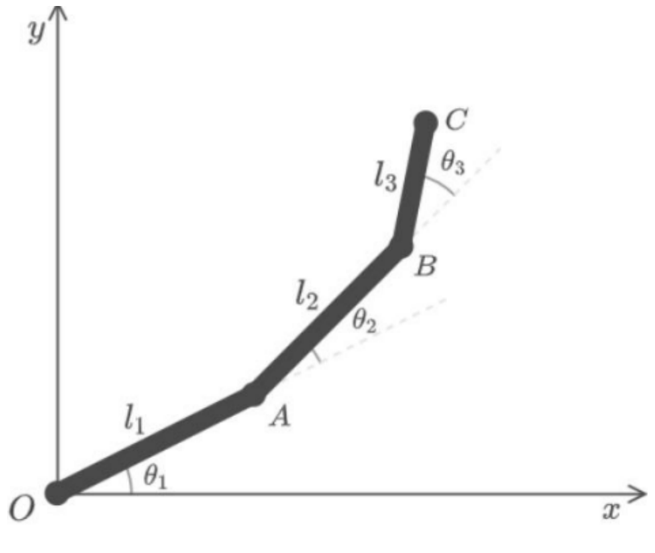
After ICP

Position estimation

Results

When cup is 1m away, the distance error is within 5mm and angular error is within 5°

Mathematical Model



Direct Kinematic Model:

Objective:
Express the Cartesian position of the gripper of the robot according to the angles of the joints of the robot.

$$x = l_1.\cos(\theta_1) + l_2.\cos(\theta_1 + \theta_2) + l_3.\cos(\theta_1 + \theta_2 + \theta_3)$$
$$y = l_1.\sin(\theta_1) + l_2.\sin(\theta_1 + \theta_2) + l_3.\sin(\theta_1 + \theta_2 + \theta_3)$$

Inverse Kinematic Model

Objective:
Express the Cartesian position of the gripper of the robot according to the angles of the joints of the robot

$$\theta_2 = \cos^{-1}\left(\frac{({x_B}^2 + {y_B}^2) - (l_1^2 + l_2^2)}{2.l_1.l_2}\right)$$
$$\theta_1 = \sin^{-1}\left(\frac{(l_1 + l_2\cos(\theta_2))y_B - (l_2\sin(\theta_2))x_B}{x_B^2 + y_B^2}\right)$$
$$\theta_3 = \varphi - \theta_1 - \theta_2$$

Mathematical Model

Dynamic Model

Lagrangian: $L = K - P$

Kinetic energy

$$K(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\, \dot{\boldsymbol{q}}^T\, \boldsymbol{M}(\boldsymbol{q})\, \dot{\boldsymbol{q}}$$

$M(\boldsymbol{q})$ is the $n \times n$ *inertia matrix* which is symmetric and positive definite.

Potential energy

$$P(\boldsymbol{q})$$

The dynamical model is obtained using **Lagrange equations**:

$$\frac{d}{dt}\left(\frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = \boldsymbol{\tau}$$

$$\ddot{q} = M^{-1}(q).\,(\tau - \dot{q}.\,C(q, \dot{q}) - G(q))$$

$q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$: Vector of joint coordinates.

$M(q) \in R^{3x3}$: Inertia Matrix.

$\dot{q}.\,C(q, \dot{q}) \in R^3$: Vector of Coriolis and Centrifugal Forces.

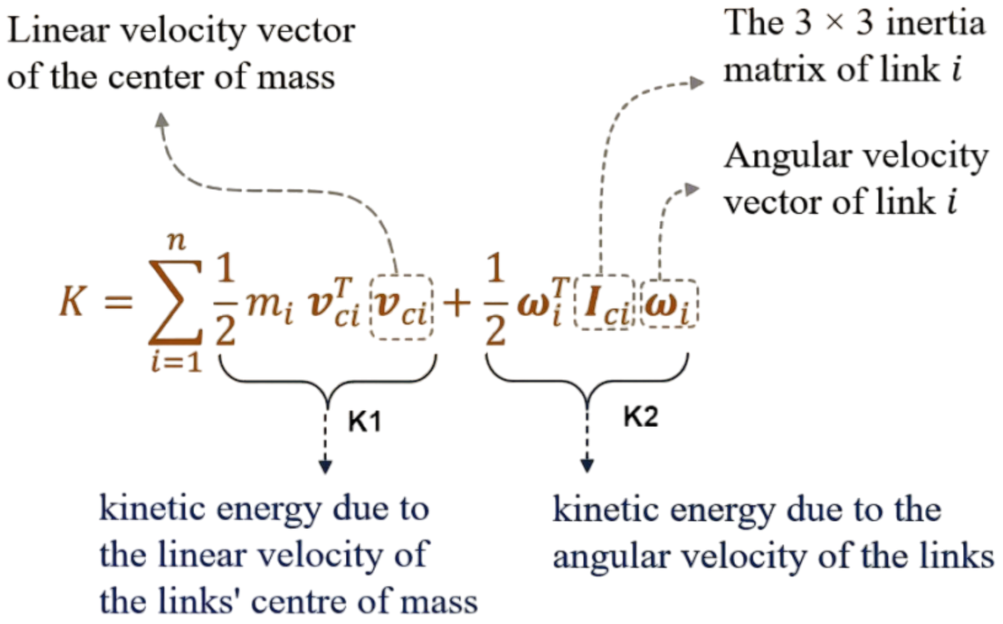
$G(q) \in R^3$: Vector of Gravity Forces.

$\tau = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}$: Vector of Joint Torques.

Mathematical Model

Dynamic Model

The overall kinetic energy of the manipulator with n links is:



$$K(q,\dot{q}) = \frac{1}{2}\dot{q}^t . M(q) . \dot{q}$$

Extensive Calculations lead to M(q).

$$M(q) = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{12} & M_{22} & M_{23} \\ M_{13} & M_{23} & M_{33} \end{pmatrix}$$

Symmetric Definite Positive Matrix

$$\boldsymbol{M(q)\ddot{q} + \underbrace{\dot{M}(q)\dot{q} - \frac{1}{2}\frac{\partial}{\partial q}(\dot{q}^T M(q)\dot{q})}_{C(q,\dot{q})\dot{q}} + \underbrace{\frac{\partial P(q)}{\partial q}}_{g(q)} = \tau}$$

D(q,q̇)

Mathematical Model

Dynamic Model

01

We calculate the potential energy associated to the masses m1, m2 and m3 and then add them up to find the total potential energy of the system.

02

Computing G for various angular positions using Matlab, we noticed that the gravitational matrix reached its maximum value when the arm was fully extended (i.e q1 = q2 = q3 = 0).

03

we incorporated a friction term into the Lagrange equation to account for the joint friction. This term is proportional to the absolute value of the joint velocity and the sign of the velocity. This is because the friction force acts in the opposite direction to the joint velocity.

$$G(q) = \frac{\delta}{\delta q} P(q) = \begin{pmatrix} \frac{\delta P}{\delta q_1} \\ \frac{\delta P}{\delta q_2} \\ \frac{\delta P}{\delta q_3} \end{pmatrix}$$

Controller Design

Linearization

$$\ddot{q} = M^{-1}(q).(\tau - \dot{q}.C(q,\dot{q}) - G(q) - Ff(q,\dot{q}))$$

$$U = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}$$

$$X = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix}$$

$$Y = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

$$\dot{X} = AX + BU$$

$$Y = CX + DU$$

With:

$$A = \begin{pmatrix} \frac{\delta f_1}{\delta q_1} & \cdots & \frac{\delta f_1}{\delta q_6} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_6}{\delta q_1} & \cdots & \frac{\delta f_6}{\delta q_6} \end{pmatrix}_{U_e, X_e}$$

$$B = \begin{pmatrix} \frac{\delta f_1}{\delta \tau_1} & \cdots & \frac{\delta f_1}{\delta \tau_3} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_6}{\delta \tau_1} & \cdots & \frac{\delta f_6}{\delta \tau_3} \end{pmatrix}_{U_e, X_e}$$

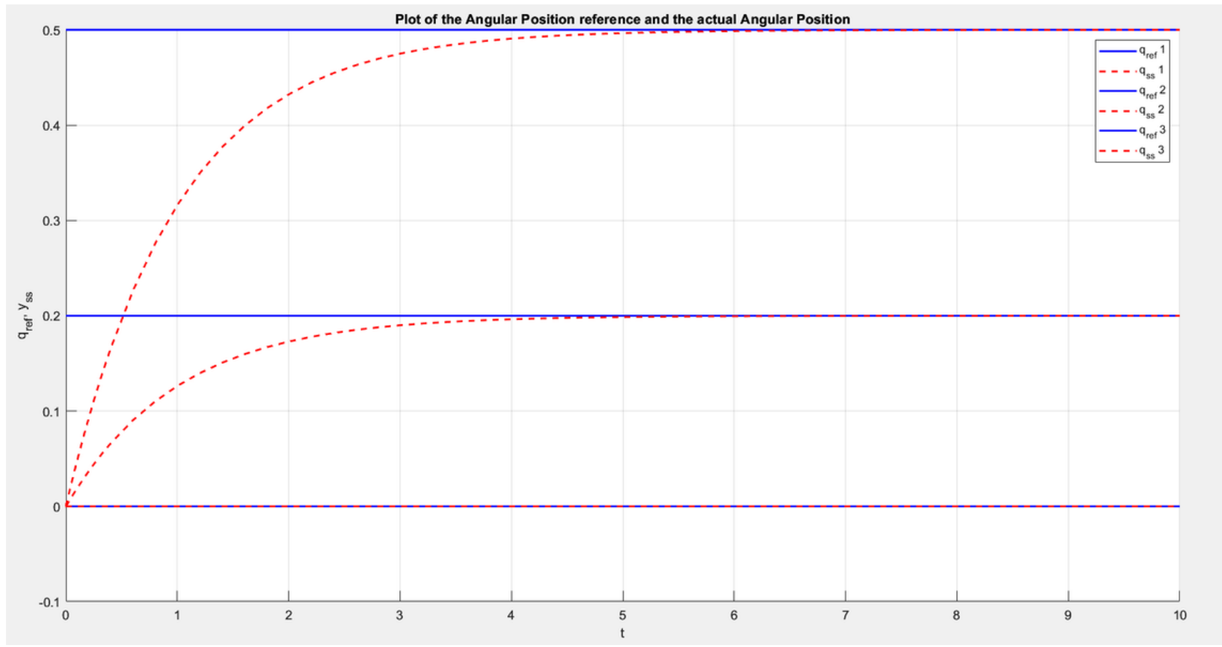
$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \mathbf{0}_{3\times 3}$$

Controller Design

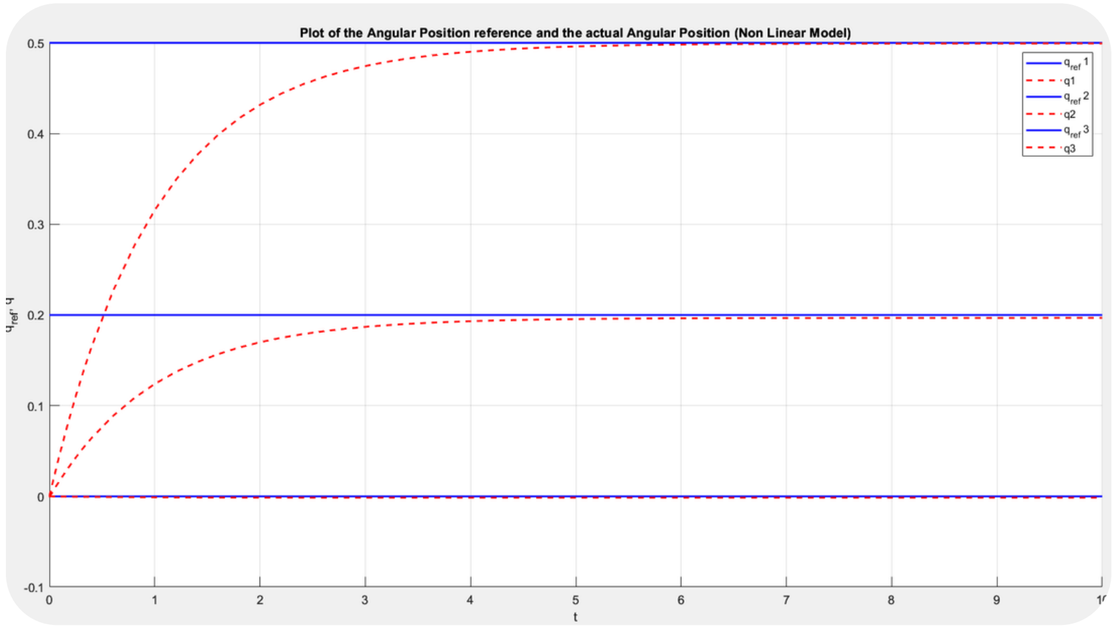
Control on Linear System

- LQR Controller
- Robust, Optimal
- Accurate trajectory tracking

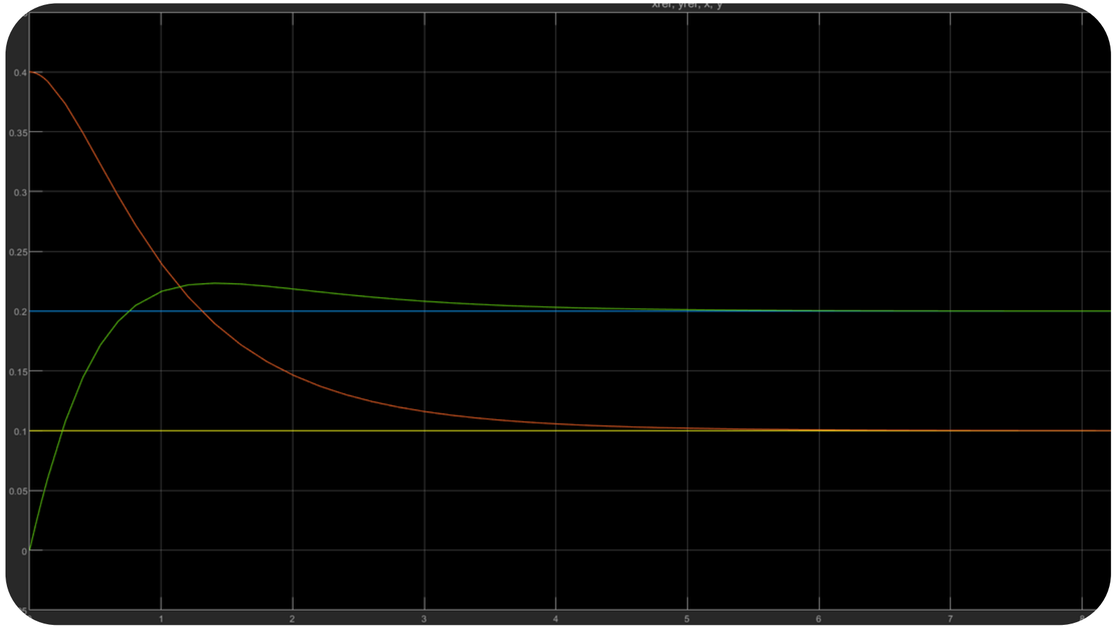


Controller Design

Angular Position Control



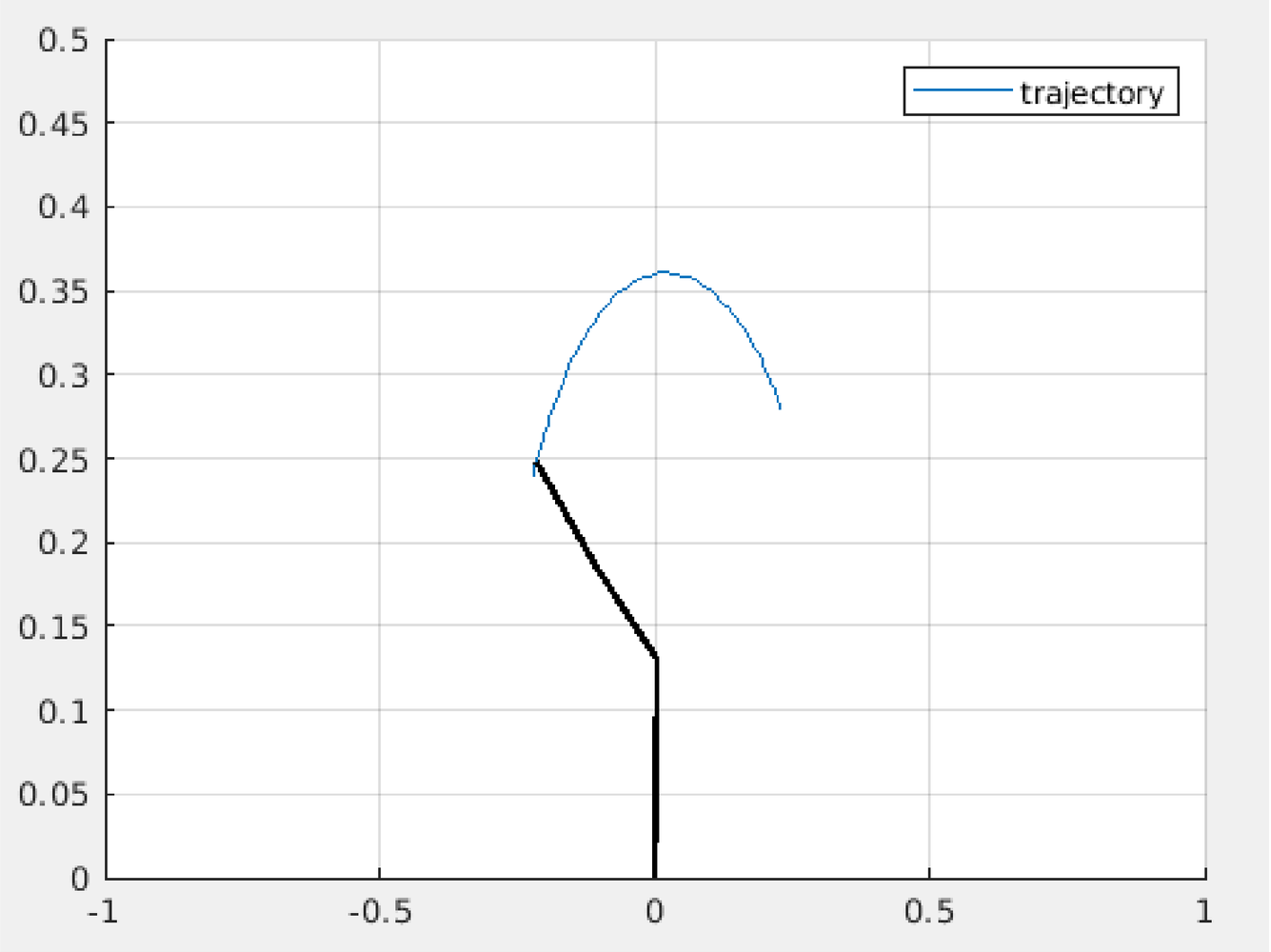
Cartesian Position Control



Control on Non Linear System

TRAJECTORY GENERATION

- Gives the angle and angular velocity of the joints to the controller;
- Endless possibilities;
- Position and velocity of the throw.



TRAJECTORY GENERATION

Developed an optimization algorithm to define the position and the velocity of the throw.

Throw equations

$$E = m \cdot g \cdot p_y + \frac{1}{2} m \cdot (v_x^2 + v_y^2)$$

$$t_{end} = \frac{v_y + \sqrt{v_y + 2 \cdot g \cdot p_y}}{g}$$

$$v_x = \frac{x_{end} - p_x}{t_{end}}$$

Cost Function

$$J = g \cdot p_y + \frac{1}{2}(v_x^2 + v_y^2)$$

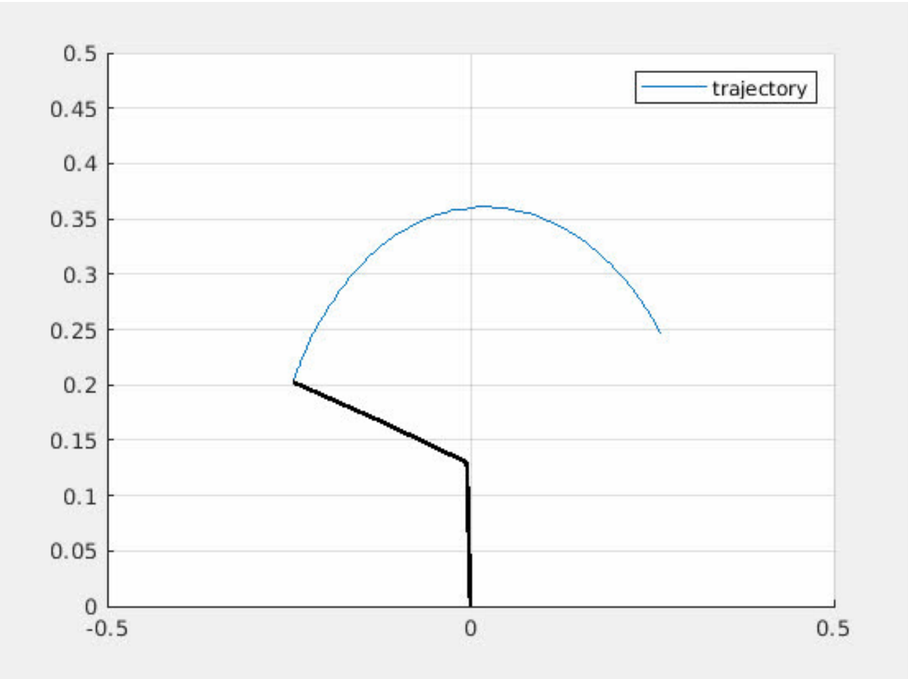
Constraints

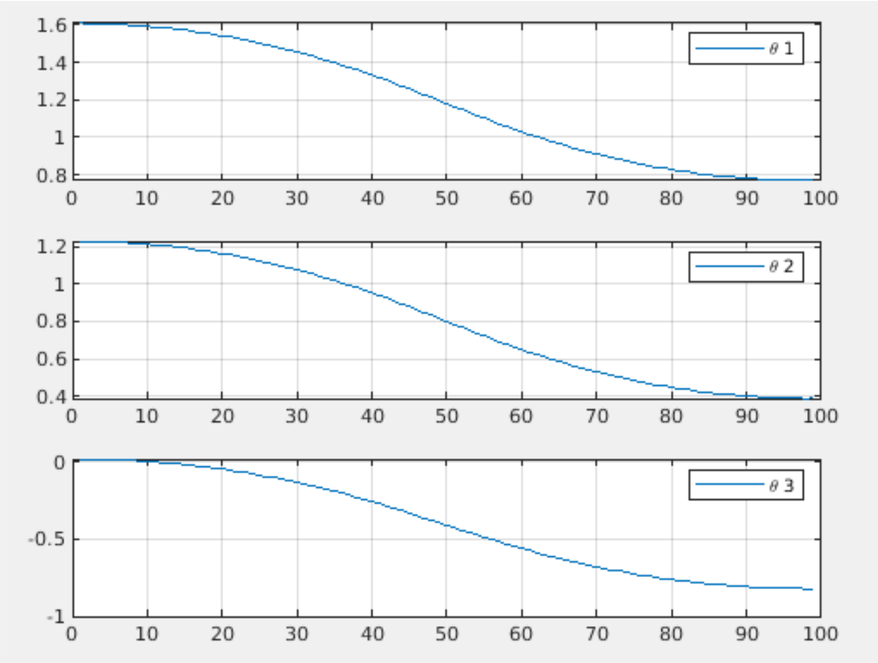
$$\begin{bmatrix} -(p_x^2 + p_y^2) + L_{min}^2 \\ p_x^2 + p_y^2 - L_{max}^2 \\ -(v_x^2 + v_y^2) \\ v_x^2 + v_y^2 - V_{max}^2 \end{bmatrix} \leq 0$$

$$\left[p_x \cdot v_x + p_y \cdot v_y \right] = 0$$

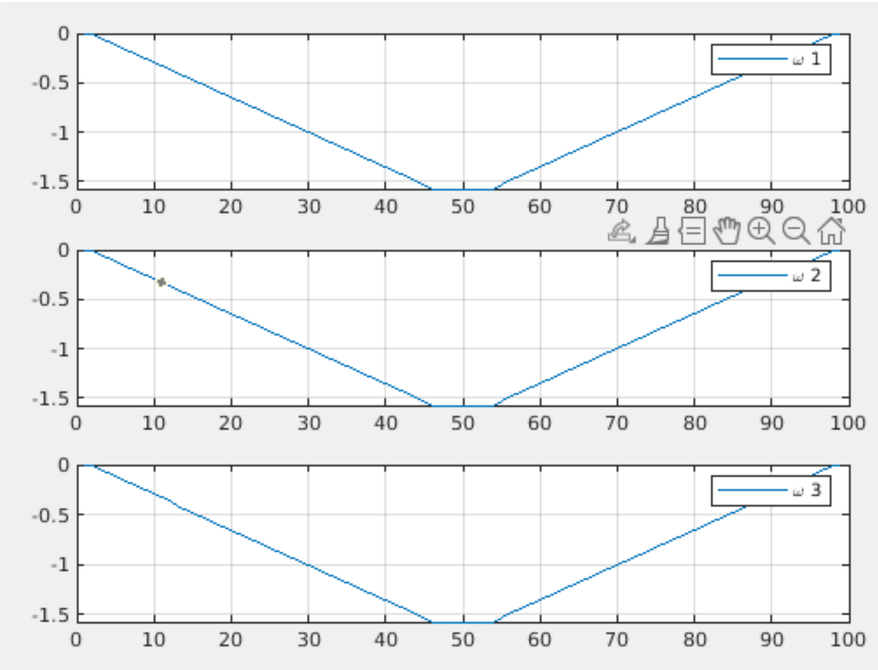
TRAJECTORY GENERATION

With the informations of the throw we create the trajectory using the trapezoidal profile





Angular position



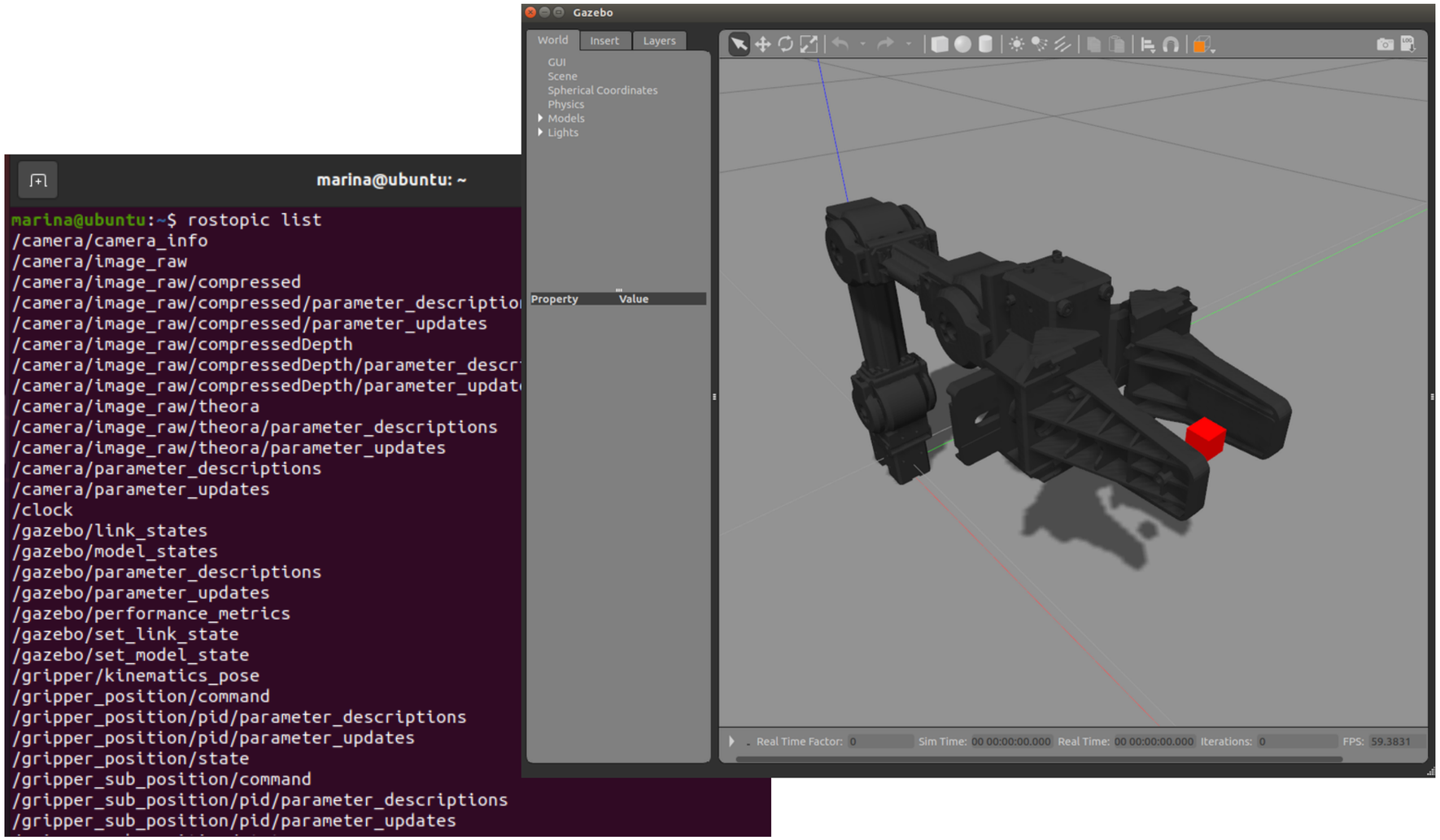
Angular velocity

System

- **Simulation environment**
  - Gazebo model
  - ROS distribution
- **Subsystems and Communication**

Simulation Environment:

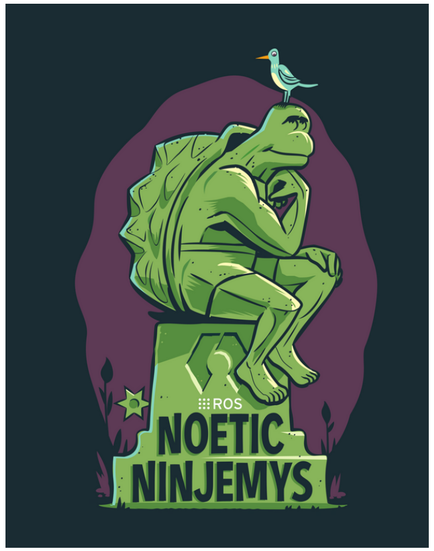
Gazebo + OpenMANIPULATOR-X Robotis model



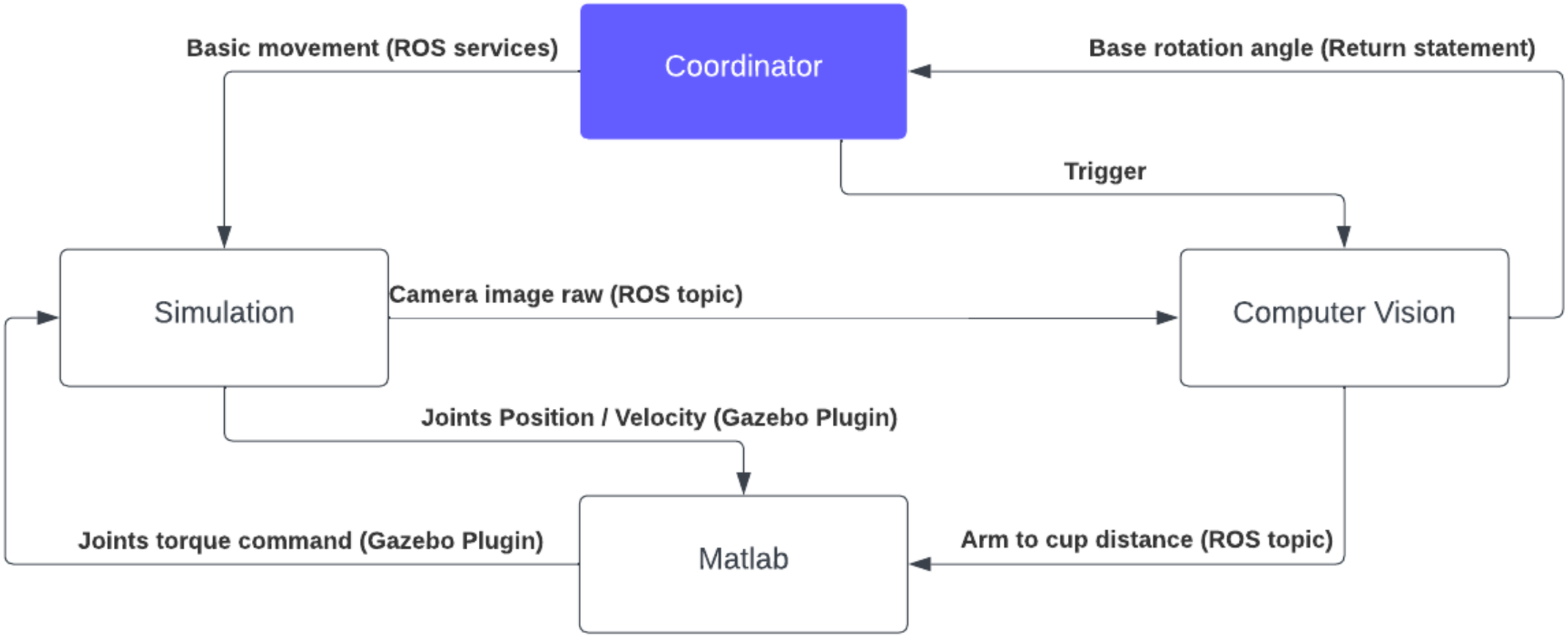
Robot Operating System (ROS)

Why ROS Noetic Ninjemys?

- **Robotis model is only compatible with ROS 1 distributions.**
- **Grater variety of ressources.**
- **Negative aspect: Not compatible with the work of previous groups.**



Subsystems


Coordinator
Basic movement (ROS services)
Base rotation angle (Return statement)
Trigger
Simulation
Camera image raw (ROS topic)
Computer Vision
Joints Position / Velocity (Gazebo Plugin)
Joints torque command (Gazebo Plugin)
Matlab
Arm to cup distance (ROS topic)

Coordinator

Main subsystem. Contains a loop to go through all the steps of a throw.

01 Generate scene

02 Trigger computer vision algorithm

03 Move the robotic arm to grab the ball and go to throwing position

04 Receive user input after the throw

05 Automatically prepare the objects for next throw

06 Repeat the loop from actions 02 to 06

Computer vision

Everything related to visual

Communicates with Simulation and Matlab using ROS topics.

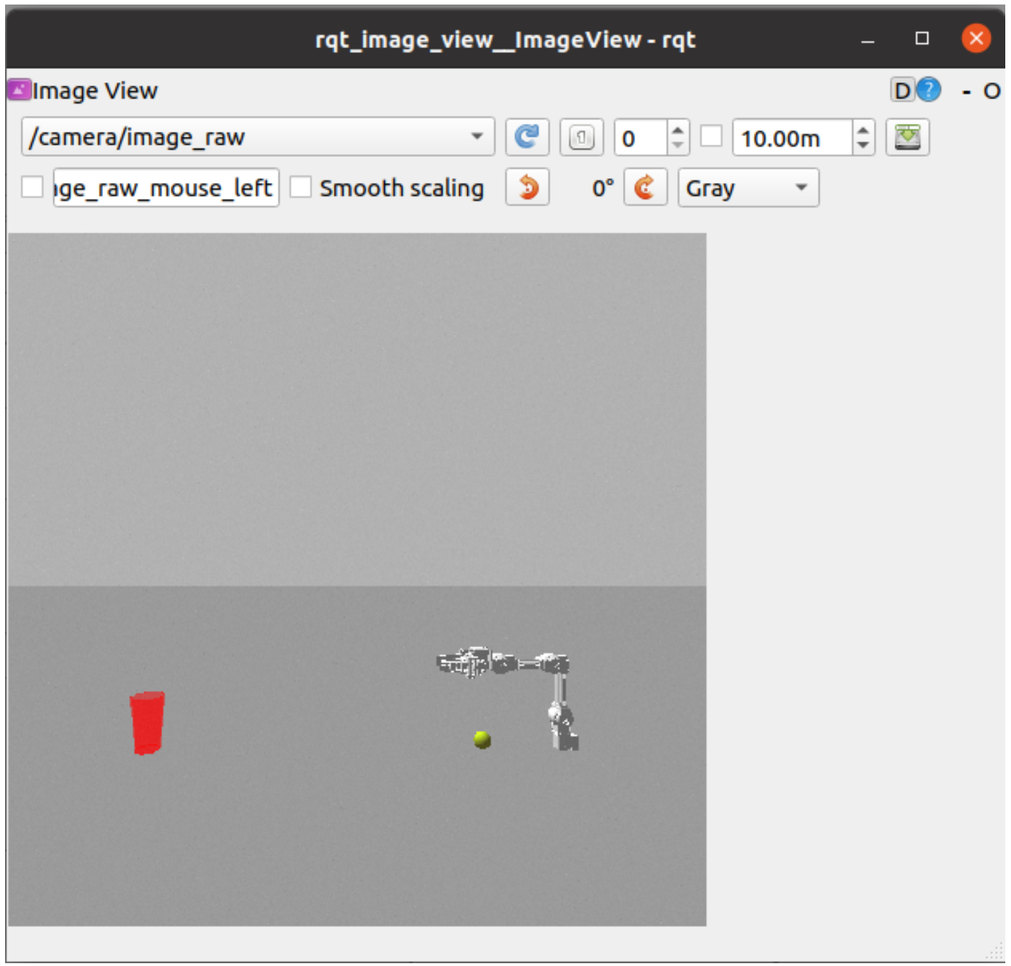
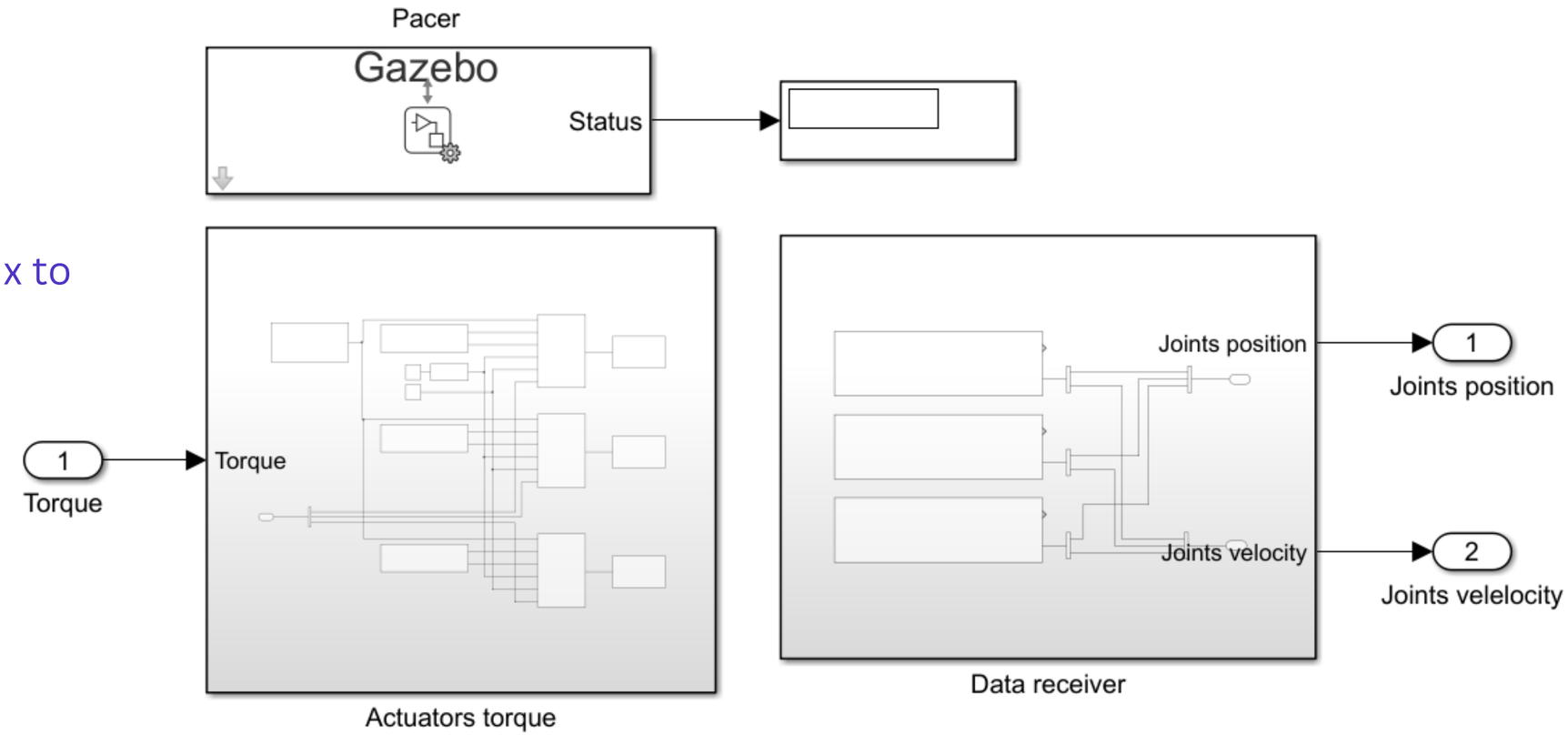


Image received from /camera/image_raw topic

Matlab

Contains everything related to control, trajectory generation and mathematical model.

Uses Windows OS instead of Linux to increase performance.



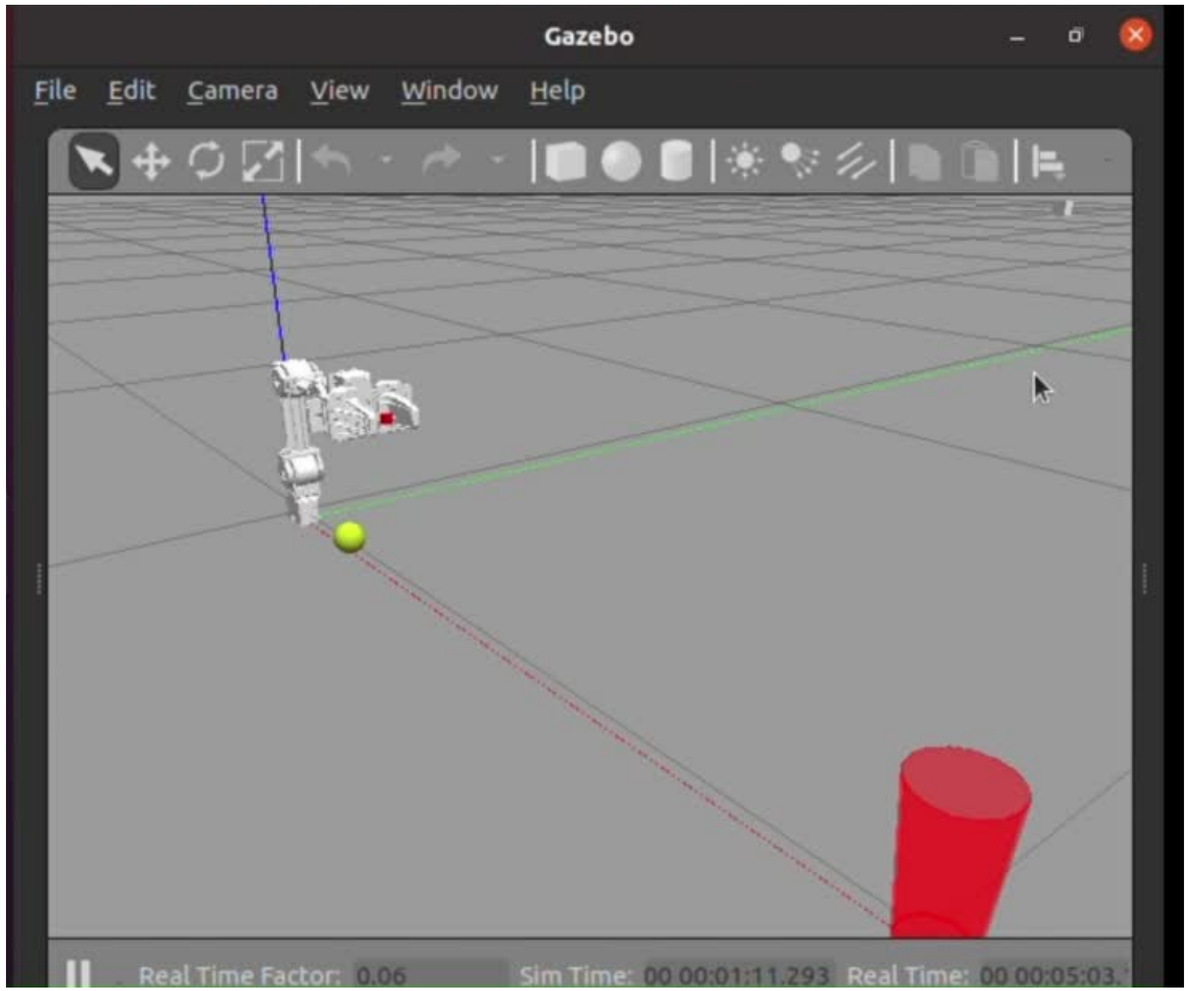
Blocks used to exchange information Gazebo x Simulink

Simulation

Models the behaviour of the system.
Is in direct communication with all other subsystems.

Empty world with:

- Robotic arm
- Camera
- Ball
- Target cup
- ArUco markers yet to be integrated



Gazebo simulation gripper test.

Conclusions & Future Work

01 We have a connected simulation system able to get the image from the simulator, process it, generate the trajectory an control it;

02 For next projects, It is needed to adapt the mathematical model control with gazebo and improve the trajectory generation algorithm;

03 Since the simulation is available, use Neural networks must be a great form to learn all the physics and control of the task;

04 We could use an AI model to estimate the cup depth.