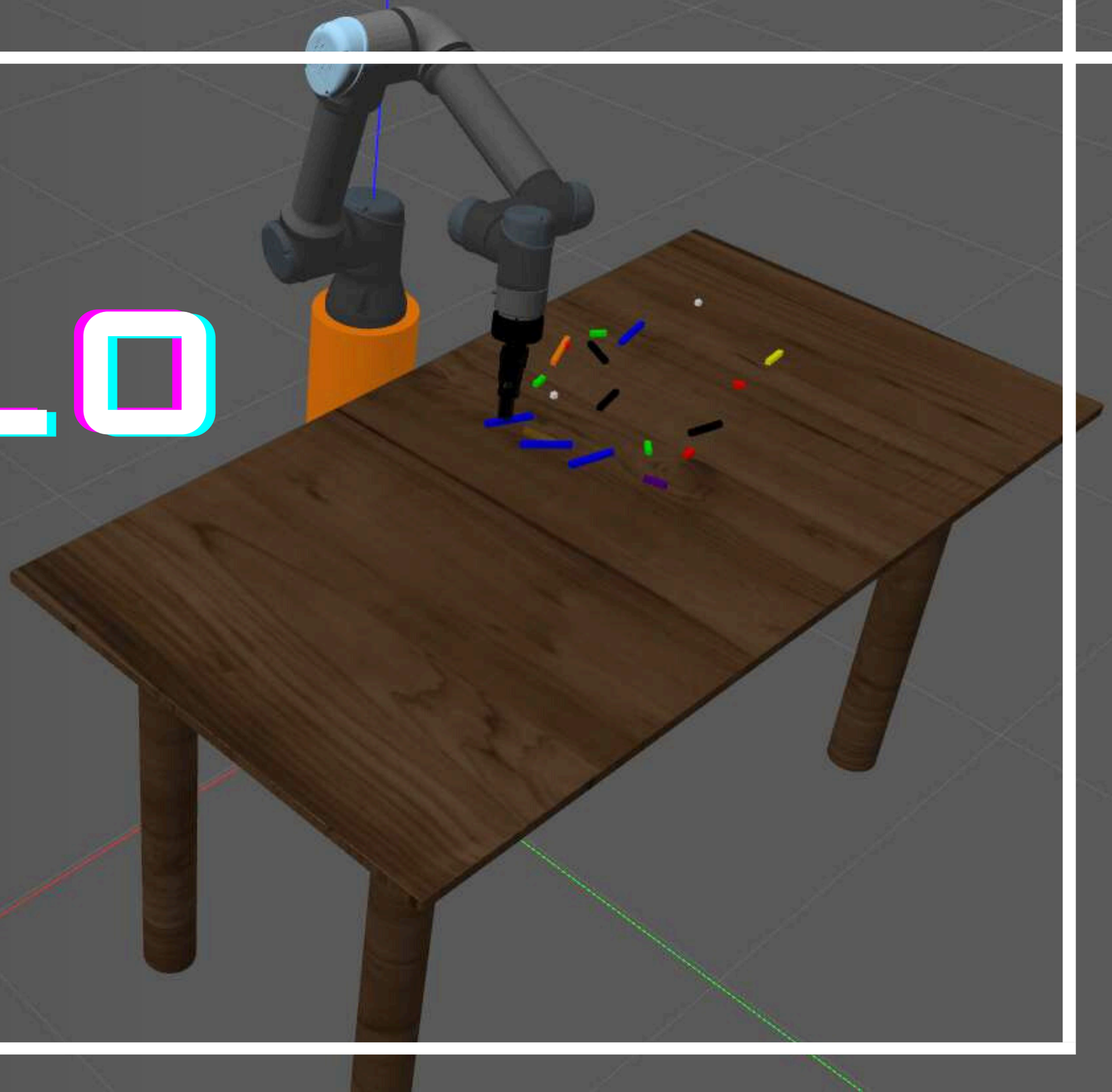
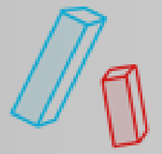
REGOLO BOT

Giulio Ganzerli, Elisa Zanni

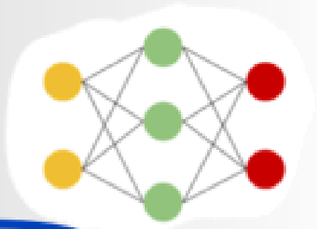


Github repo

The project



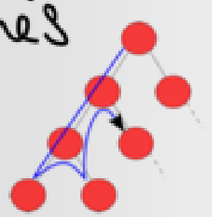
2 spawn regoli



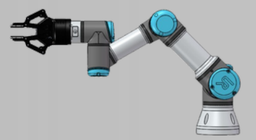
AI

3 recognize regoli with YOLO

4 algorithm for choosing the right ones



5 pick & place them in the right order



6 quit

1 input numbers

The environment

kinect
camera

UR5 robot

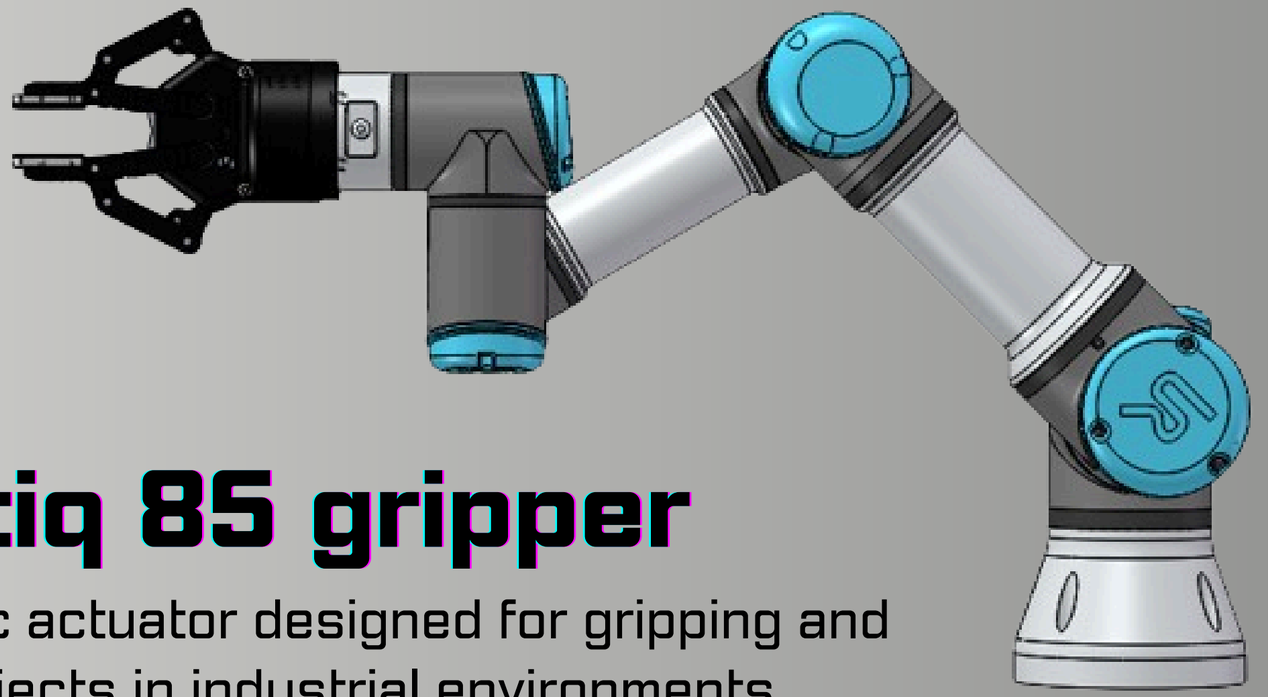
table

pedestal

table texture :

Robot UR5

A six-axis industrial robotic arm produced by Universal Robots



Robotiq 85 gripper

A specialized robotic actuator designed for gripping and manipulating objects in industrial environments

Moveit! configuration

We used Moveit! to configure the robot and the gripper with some predefined poses:

- *zero*, *home* for the arm
- *open*, *closed* for the hand

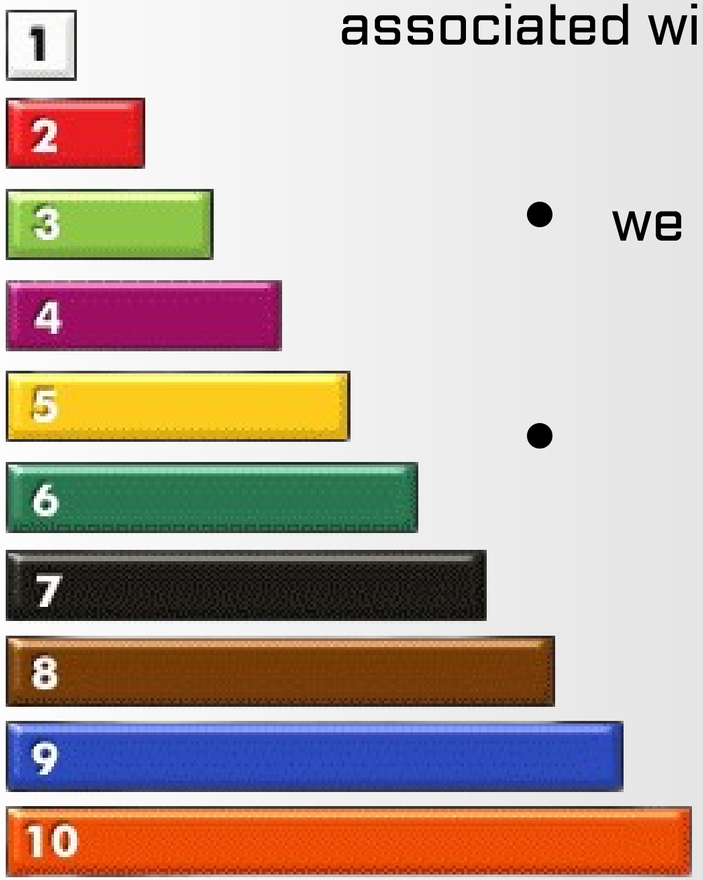
We also implemented a python interface for manipulate them

robot_interface.py

```
class Regolobot:
  constructor:
    # Initialize the moveit_commander and planning scene interface
    # Create a DisplayTrajectory ROS publisher which is used to display trajectorie
    # Create action client for the "Execute Trajectory" action server

  def set_pose(self, arg_pose_name):
    # set_named_target for the moveit_commander with the input arg_pose_name
    # Create the plan
    # Create a goal message object for the action server
    # Update the trajectory in the goal message
    # Send the goal to the action server

  def move_ee(self, x, y, z, roll=0, pitch=0, yaw=0):
    # Create a geometry_msgs.msg.Pose() with the input values
       # utilize tf.transformations.quaternion_from_euler for
       #converting the roll, pitch, yaw values
    # ...

  def is_stationary():
    return pos_error < tolerance_pos and ori_error < tolerance_ori

  def safe_move_ee():
    # allows to specify a tolerance on the position and on orientation
    # use is_stationary to check if the robot is stopped or not

  def __del__(self)
```

Math sticks

Math sticks are an educational tool with pedagogical purposes.
There are 10 of them and every color is associated with a different lenght.



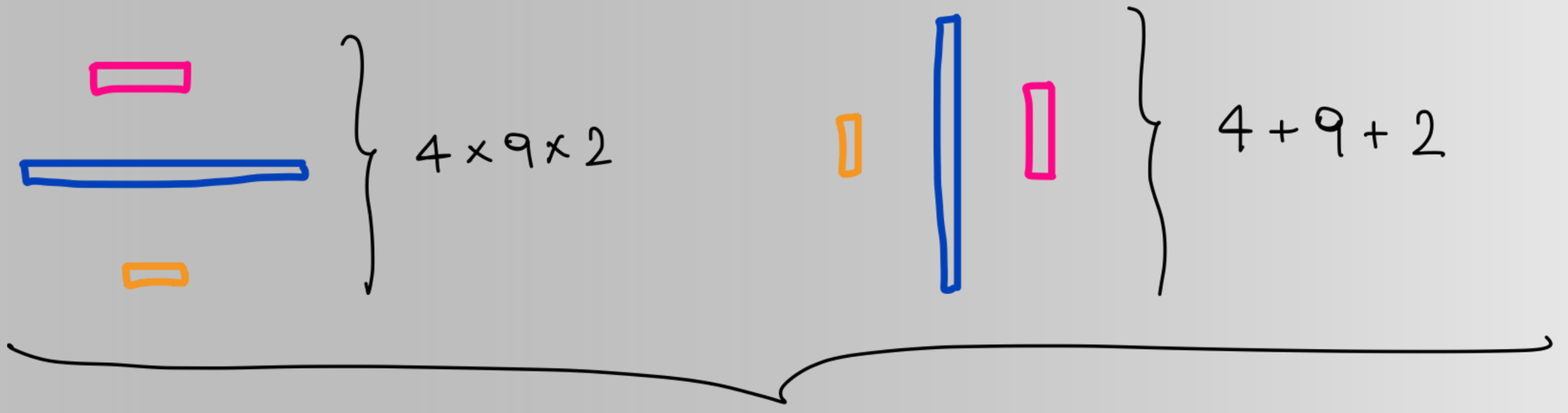
- we created the meshes using Fusion360
- we integrated them in the Gazebo environment as sdf files

```
<model name="mathstick1">
  <static>false</static>
  <link name="link">
    <collision name="collision">
      <geometry>
        <mesh>
          <uri>model://mathstick1/meshes/MathStick1.stl</uri>
          <scale>0.001 0.001 0.001</scale>
      <surface>
        <friction>
          <ode>
            <mu>1.0</mu> <!-- along the contact surface -->
            <mu2>1.0</mu2> <!-- perpendicular direction -->
      <material>
        # color
    </visual>
    <inertial>
      <mass>0.005</mass>
      <inertia>
        <ixx>0.00000008333333333333333</ixx>
        <ixy>0.0</ixy>
        <ixz>0.0</ixz>
        <iyy>0.00000008333333333333333</iyy>
        <iyz>0.0</iyz>
        <izz>0.00000008333333333333333</izz>
```

Layout concept

- multiplication: horizontal
- addition: vertical

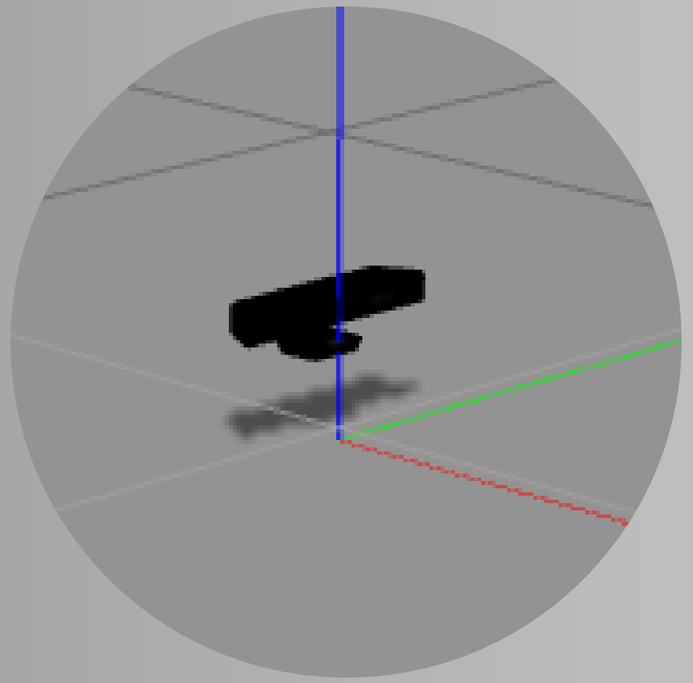
Example:



$$(4 \times 9 \times 2) \oplus 4 + 9 + 2$$

math_sticks.py

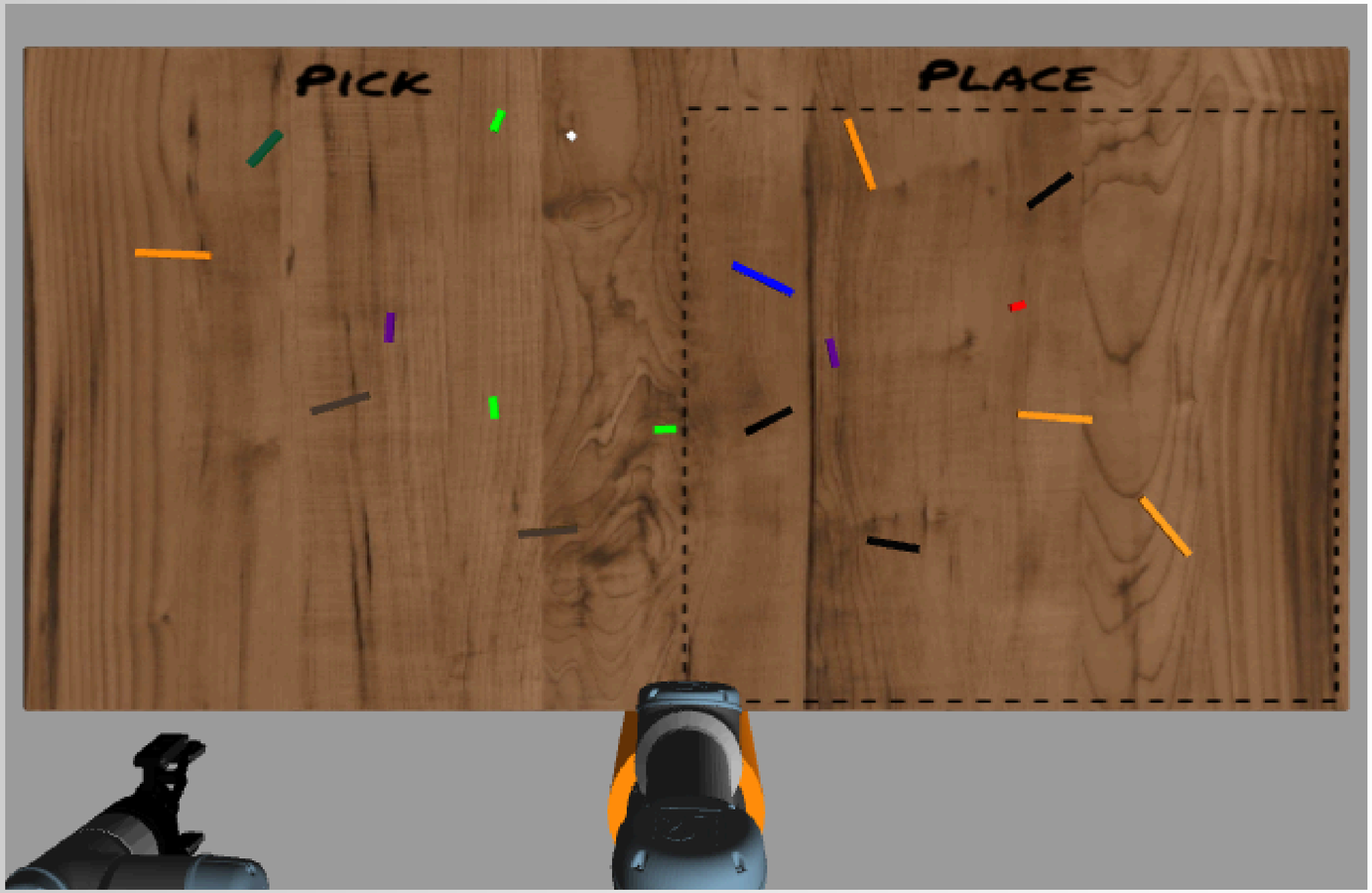
```python
class MathStick:
    def __init__(
        category,
        number,
        x, y, z,
        roll=math.pi,
        pitch=math.pi / 2,
        yaw=0,
        base_model_path=""
        )

class MathStickManager:
    def __init__(
        self,
        spawner_proxy: SpawnModel,
        delete_proxy: DeleteModel,
        categories_count: int,
        base_model_path: str,
        **kwargs,
    ):

    def generate_coordinates(self)
    def generate_angles(self)
    def collides(self, other: MathStick)
    def random(self, category=None):
    def random_within(self, r, category=None
    def spawn(self, math_stick: MathStick)
    def current_labels(self)
    def save_labels(self, output_path: Path)
    def delete(self, name: str)
    def delete_all(self)
```



Kinect camera

<plugin name="camera_plugin" filename="libgazebo_ros_openni_kinect.so">

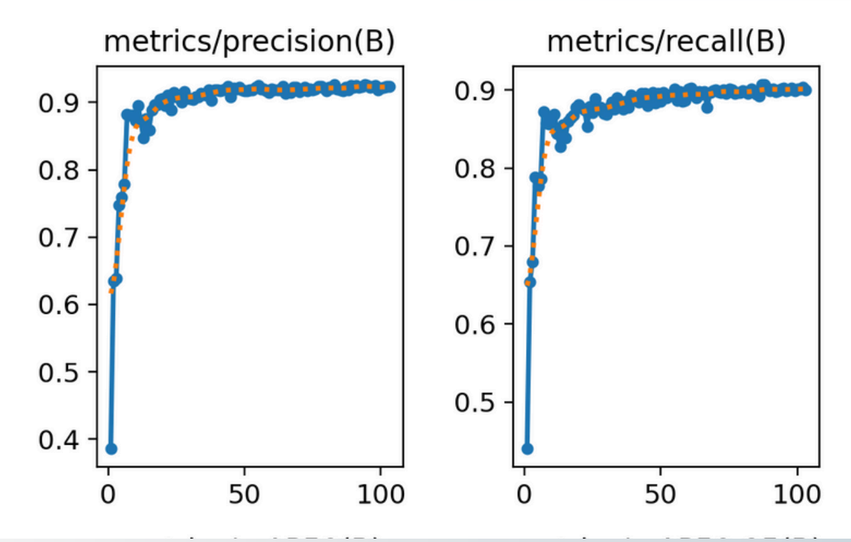
- set the camera in the Gazebo scene
- publish images and point clouds to ROS topics
- use Rviz to see them

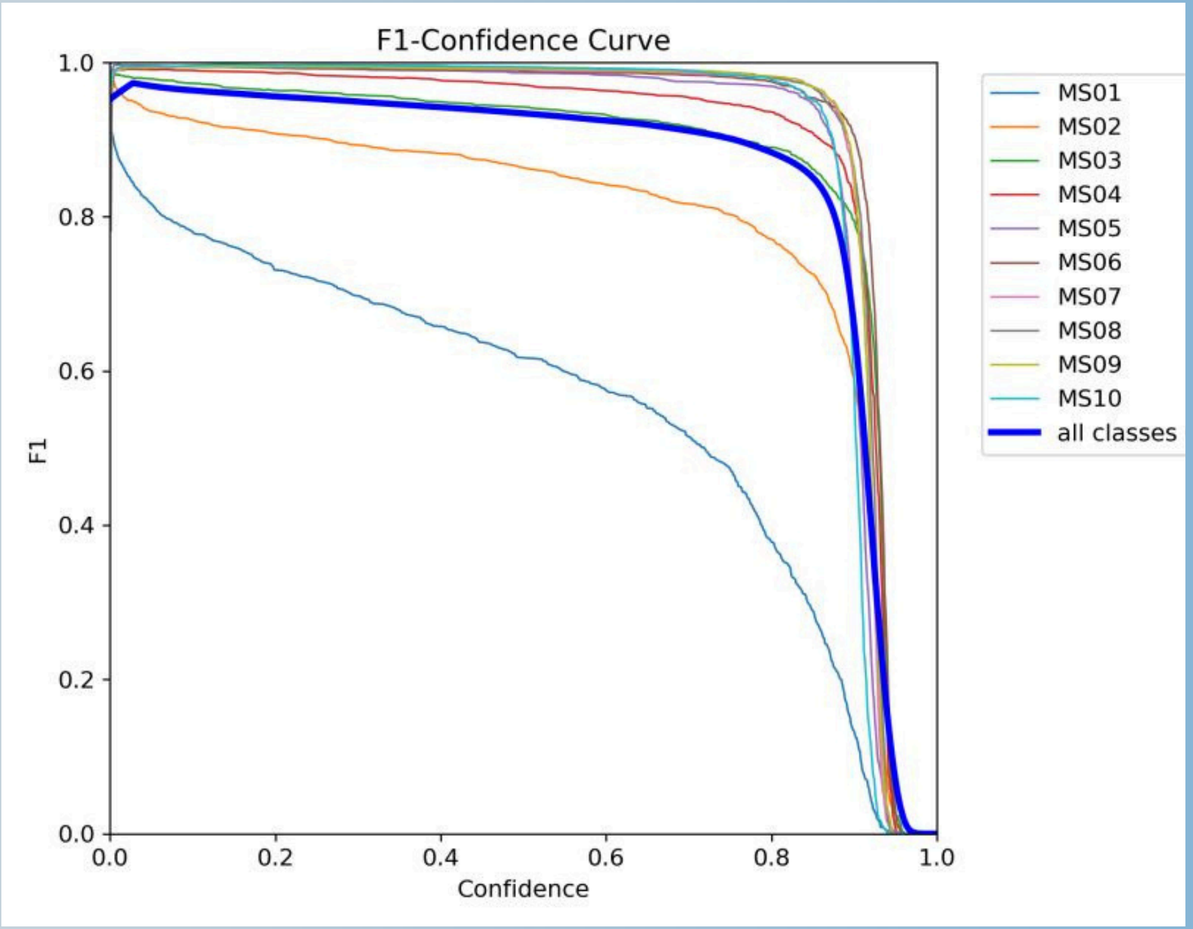


Training model

- model: YOLOv8-obb
- training: 5000 images + (2500 v2 + 2500 v3)
  - 25% validation
  - 5% test
- label format:

  math_stick_type, x1, y1, x2, y2, x3, y3, x4, y4
- *results:*



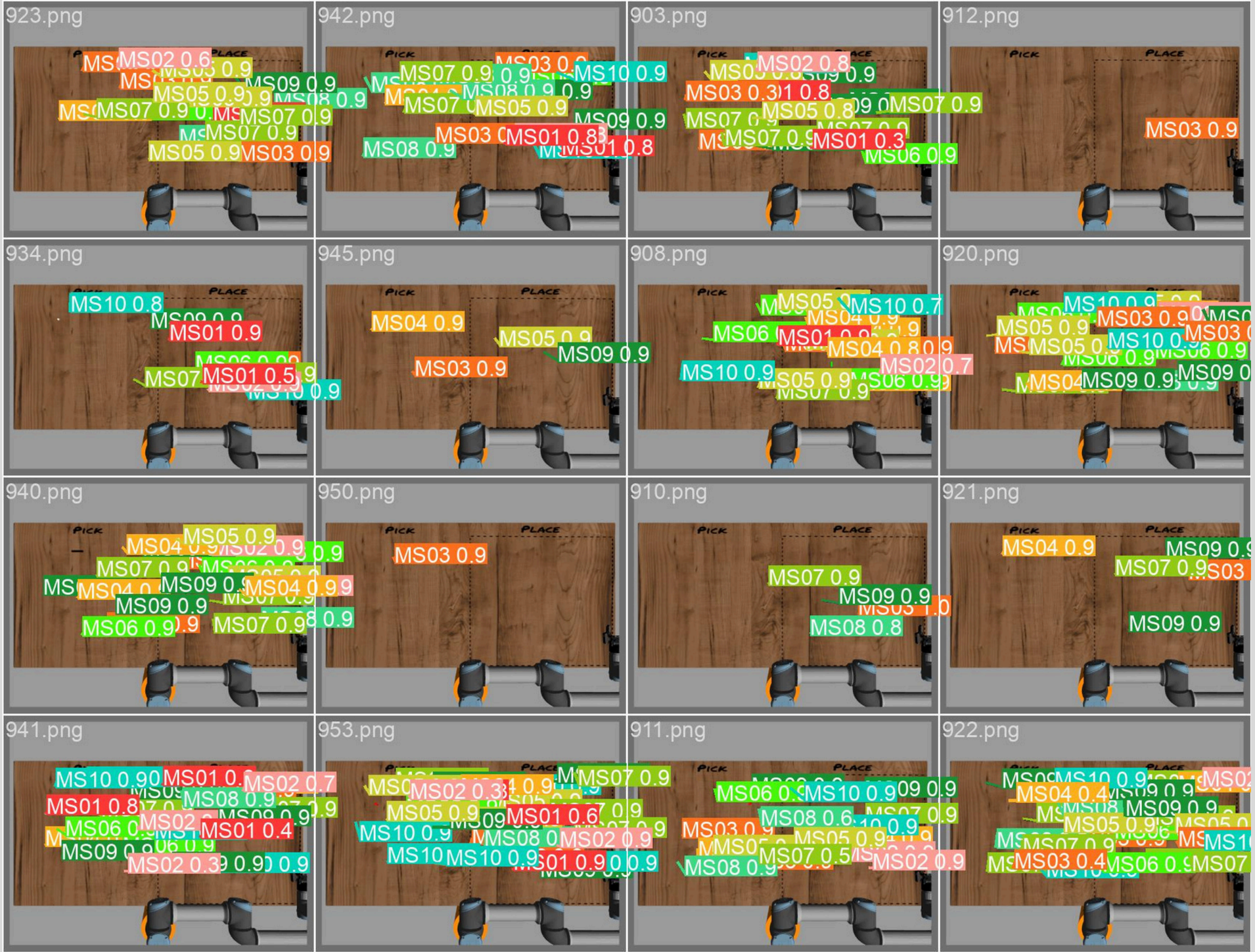



MS04 0.88
MS05 0.75
MS09 0.70
MS02 0.92
MS08 0.91
MS08 0.94
MS02 0.89
MS06 0.84


MS10 0.77
MS03 0.92
MS06 0.87
MS09 0.88
MS03 0.89
MS06 0.92
MS10 0.89

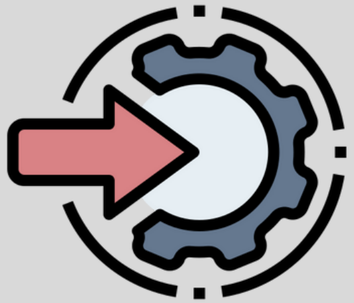

MS03 0.91
MS06 0.86
MS02 0.85
MS03 0.83
MS04 0.88
MS05 0.93
MS05 0.81
MS10 0.88
MS08 0.94


MS10 0.93
MS03 0.93
MS10 0.87
MS09 0.86
MS06 0.86
MS08 0.78
MS07 0.83
MS08 0.81
MS04 0.82
MS07 0.81
MS06 0.87
MS07 0.78

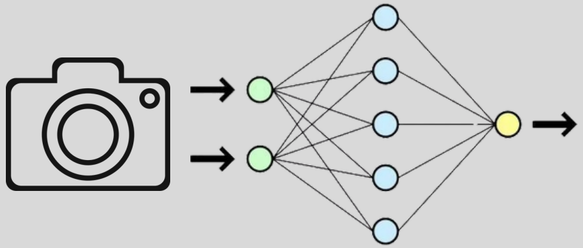

923.png
942.png
903.png
912.png
934.png
945.png
908.png
920.png
940.png
950.png
910.png
921.png
941.png
953.png
911.png
922.png

Simulation_controller

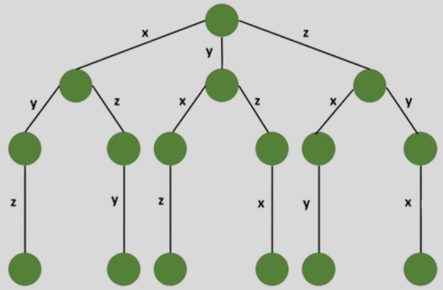
intializations



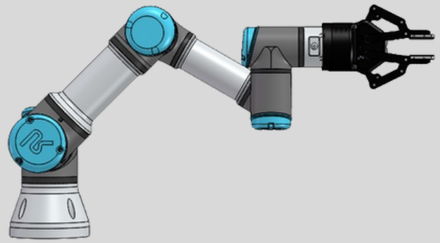
vision system



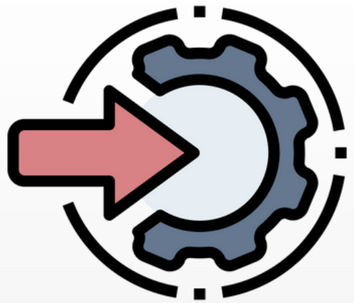
back-tracking



pick & place



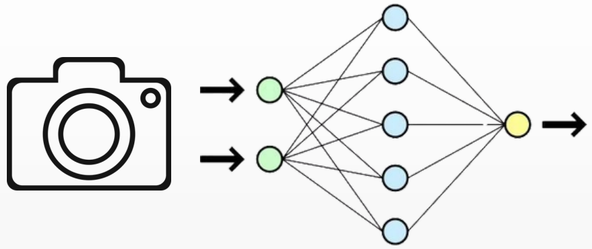
Simulation_controller



1 intializations

- initialization Robot:
  - arm = Regolobot("manipulator")
  - hand = Regolobot("gripper")
- initialization servers:
  - spawn_model_proxy = rospy.ServiceProxy("regolobot/spawn_model", SpawnModel)
  - delete_model_proxy = rospy.ServiceProxy("gazebo/delete_model", DeleteModel)
- ask inputs:
  - “Would you like to choose the math sticks to spawn?”
  - “Insert number to rapresent (target)”
- spawn_model_server spawns math sticks based on inputs messages

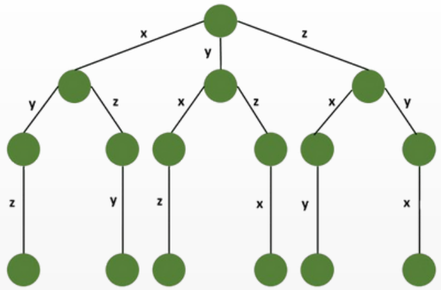
Simulation_controller



2 vision system

- Retrieve image from the kinect camera
- sends a POST request to the URL "https://regolonet.mgteam.one/predict" with the encoded image data
- The URL is associated with a **Flask server** on Giulio's pc that feed the image to the YOLO network with the pretrained weights
- It outputs the labels, translate the coordinates of Yolo images in Gazebo world coordinates and send them back as output

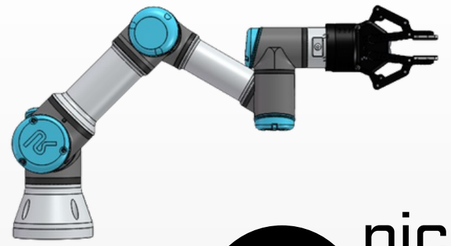
Simulation_controller



3 back-tracking

- takes the math sticks list of the ones recognized on the table & the target number
- order the list
- use back-tracking for determine a (approximate) optimal solution
  - starts from the biggest math sticks & from multiplication
  - check if the result is bigger than the target
  - if so, tries with addiction
  - check if the result is bigger than the target
  - if so, tries with smaller math sticks
- outputs a list of multiplications with the solution, if exists.

Simulation_controller



pick & place

4

- We decided a starting position (x, y)
- Determine max solution height
  - if max_height > 0.8:
    raise ValueError("The solution is too tall. It will not fit on the table.")
- Determine solution length
  - if length > 0.9:
    raise ValueError("The solution is too long. It will not fit on the table.")
- For the positioning, two innested loop:
  - the external one for addends
  - the inner one for multiplications
- 0.5 distance between each other

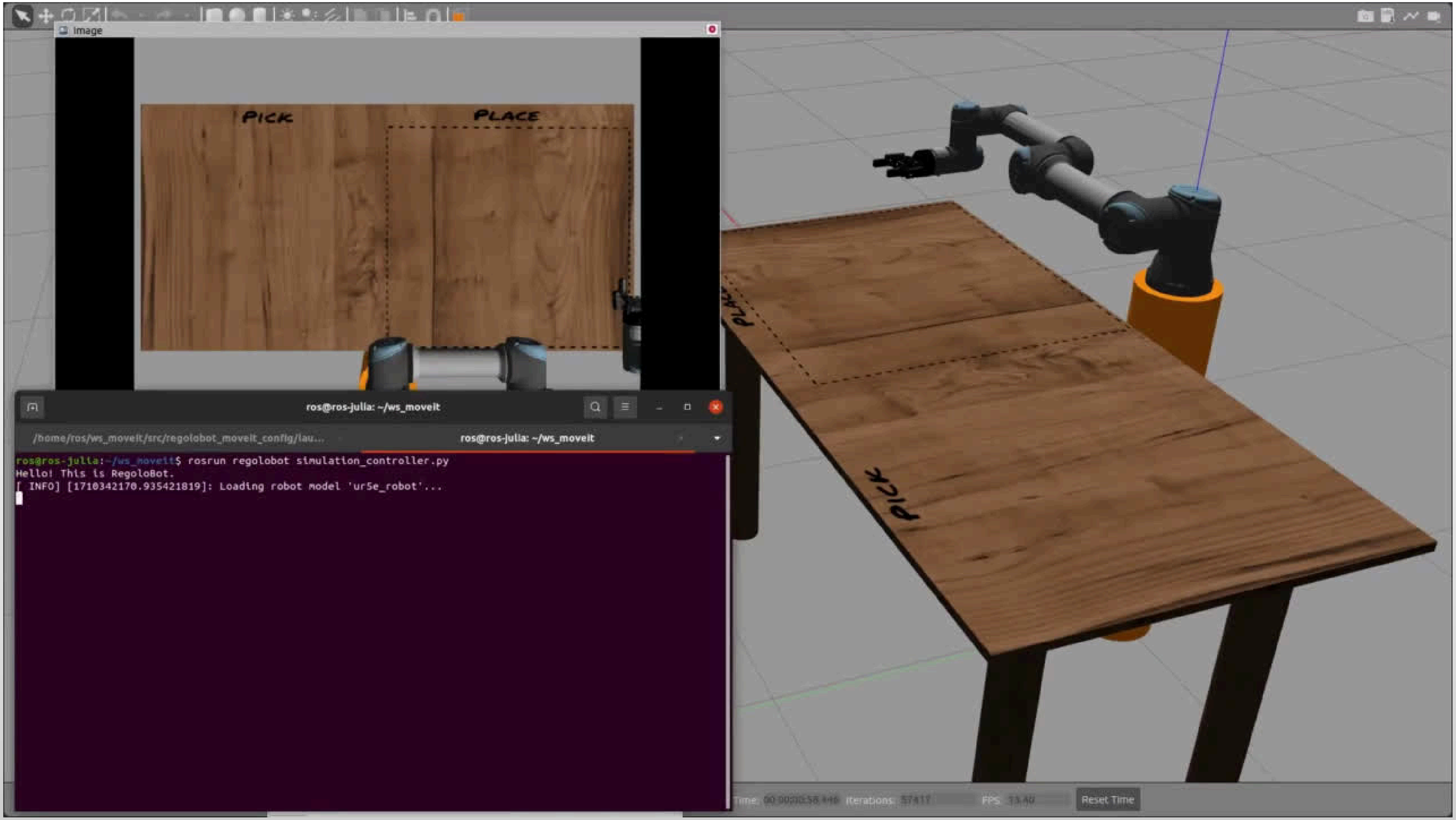


Image
PICK
PLACE
ros@ros-julia: ~/ws_moveit
/home/ros/ws_moveit/src/regolobot_moveit_config/lau...
ros@ros-julia: ~/ws_moveit
ros@ros-julia:~/ws_moveit$ rosrun regolobot simulation_controller.py
Hello! This is RegoloBot.
[INFO] [1710342170.935421819]: Loading robot model 'ur5e_robot'...
PLACE
PICK
Reset Time