Project - Robot Pick and Place

ECE 275

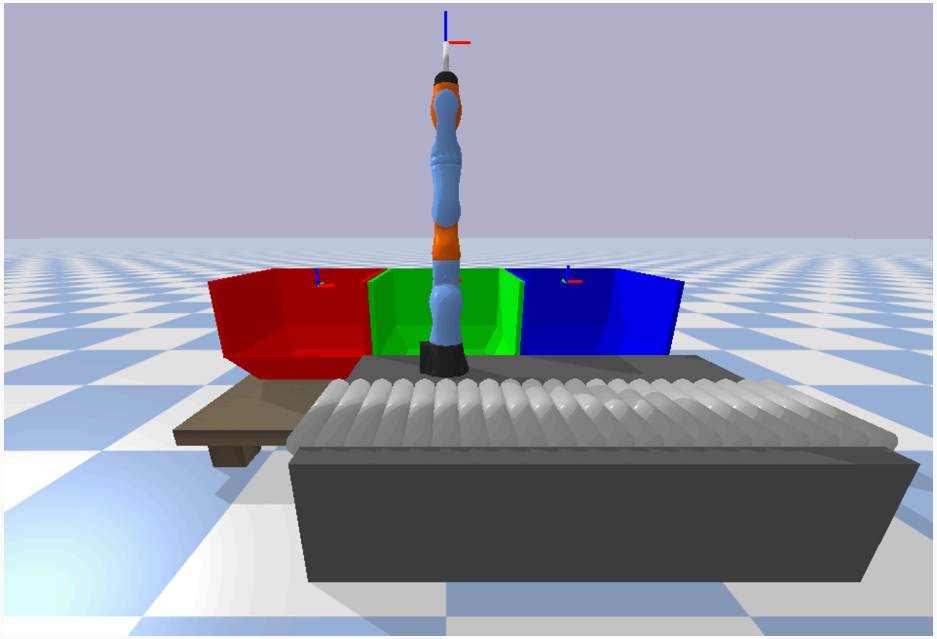
David Felipe Alvear Goyes

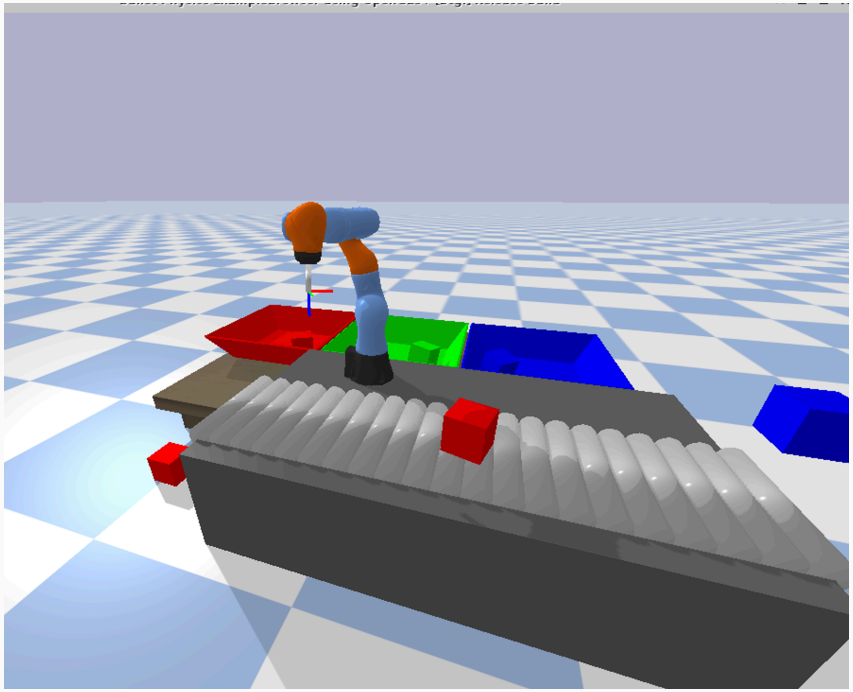
King Abdullah University of Science and Technology

Table of contents

Problem Formulation

Problem Formulation

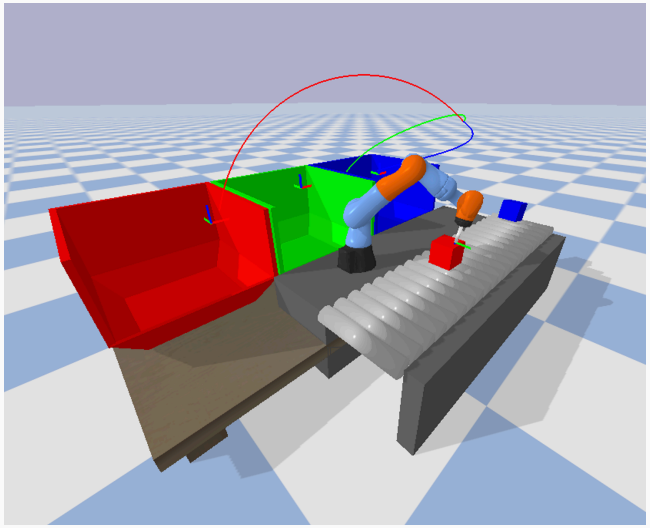




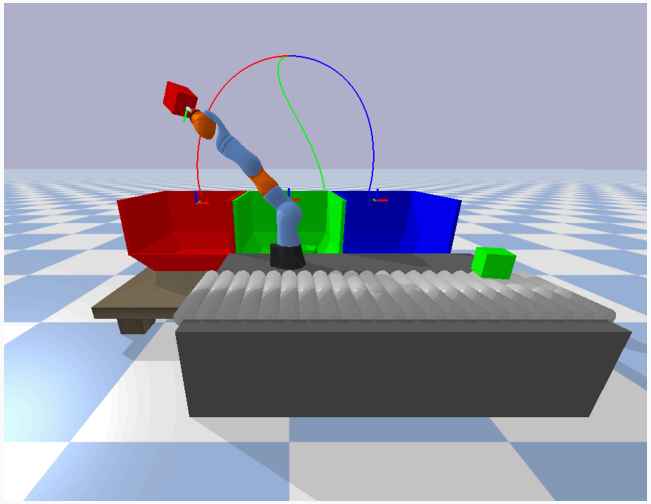
Design a control and planning algorithm for a manipulator to pick color-coded cubes on a conveyor system, and accurately place in their corresponding color-coded trays

Solution

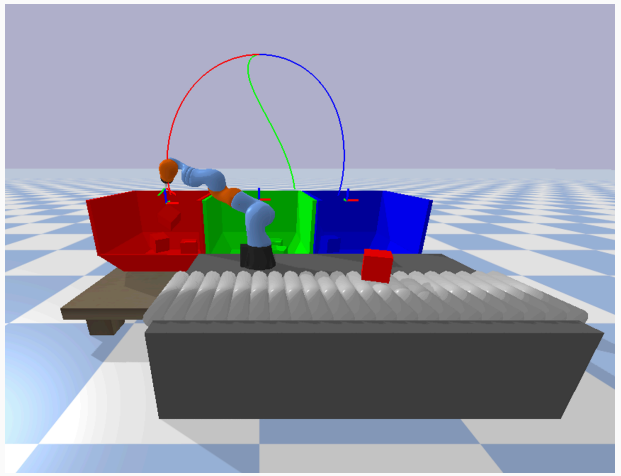
Three Scenarios



Pick up Cube



Transport Cube



Release Cube and restart

Design Requirements

- **LQR Controller**: Design a Linear Quadratic Regulator (LQR) controller to provide optimal control for the manipulator during the pick and place process.
- **PRM Trajectory Generator**: Implement a Probabilistic Roadmap (PRM) trajectory generator to efficiently plan collision-free paths for the manipulator to follow while picking and placing cubes.
- **Task Allocator**: Develop a task allocation algorithm to intelligently assign the pick and place tasks to the manipulator, considering the cube's color and drop-off locations.

LQR Controller

Dynamics of the manipulator

$$\tau = M(\theta)\bar{\tau} + V(\theta, \dot{\theta}) + G(\theta) \tag{1}$$

$$\bar{\tau} = \ddot{\theta} \tag{2}$$

Feedback linearization

$$\mathsf{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \quad \dot{\mathsf{x}} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix}$$

$$\dot{\mathsf{x}} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \mathsf{x} + \begin{bmatrix} 0 \\ I \end{bmatrix} u$$

LQR Controller

Design a LQR controller $u = Kx$, where $K = (K1, K2)$.

$$\tau = M(\theta)\bar{\tau} + V(\theta, \dot{\theta}) + G(\theta) \tag{3}$$

$$\tau = M(\theta)(K_1\theta + K_2\dot{\theta}) + V(\theta, \dot{\theta}) + G(\theta) \tag{4}$$

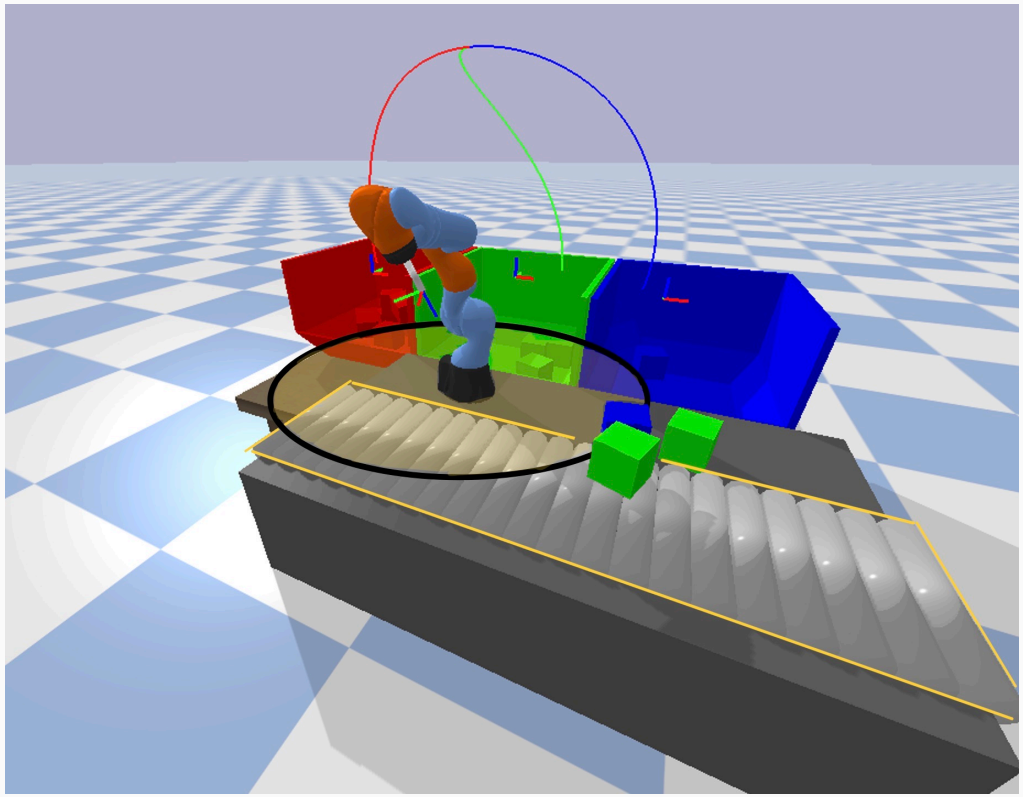
Given a reference trajectory θ_{ref}, $t > 0$. Design a controller for trajectory tracking

$$\mathbf{x}_{ref} = \begin{bmatrix} \theta_{ref} \\ \dot{\theta}_{ref} \end{bmatrix}, \quad \dot{\mathbf{x}}_{ref} = \begin{bmatrix} \dot{\theta}_{ref} \\ \ddot{\theta}_{ref} \end{bmatrix}$$

$$\dot{\mathbf{x}}_{ref} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \mathbf{x}_{ref} + \begin{bmatrix} 0 \\ I \end{bmatrix} u_{ref}$$

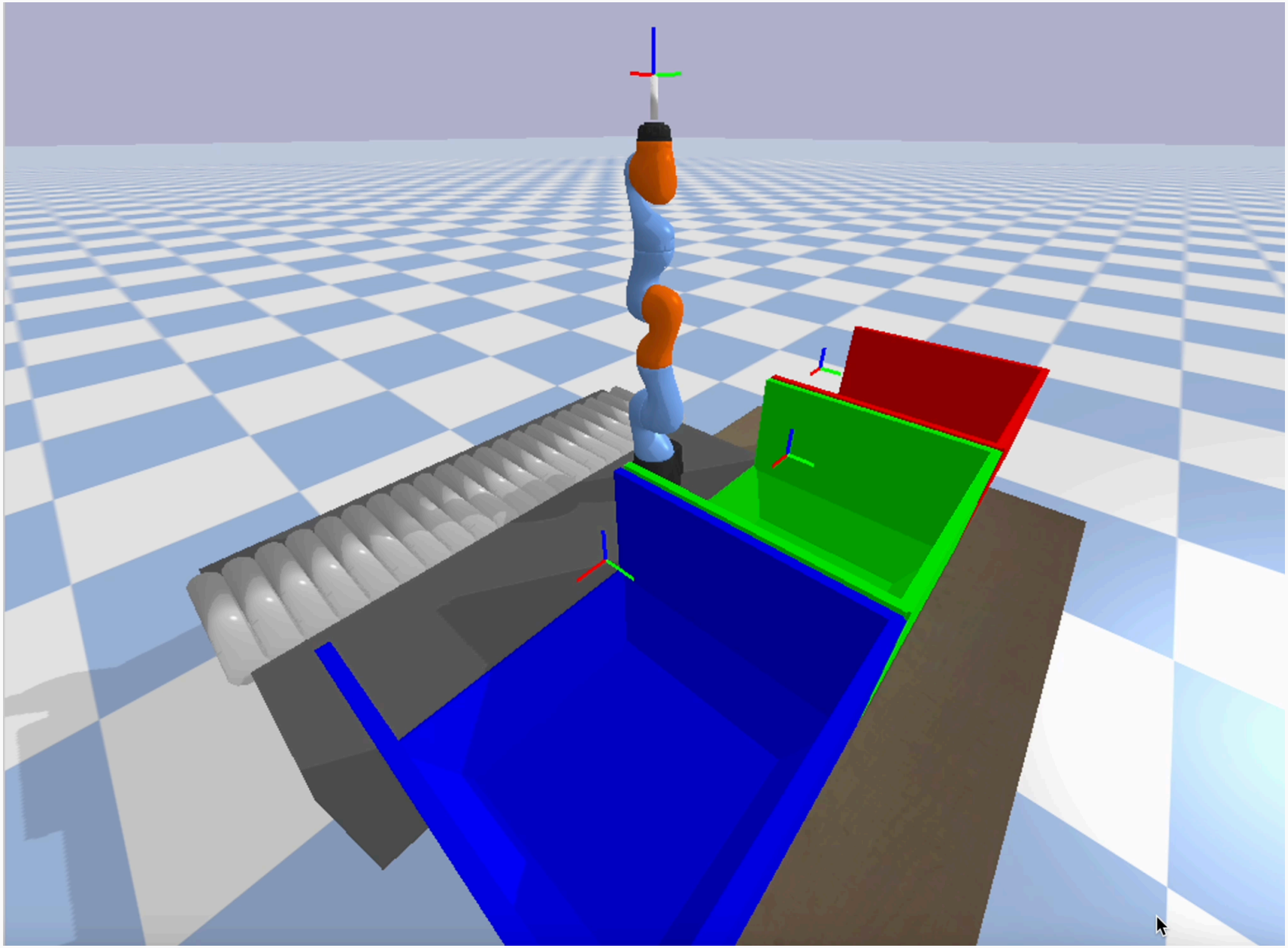
Define $\bar{x} = x - x_{ref}$ and $\bar{u} = u - u_{ref}$. Then we can design a controller using LQR. We can compute the K matrix using the **control** library in **python** and apply the control input $\bar{u} = K\bar{x} = K_1(\theta - \theta_{ref}) + K_2(\dot{\theta} - \dot{\theta}_{ref})$.

Task Allocator and Conditions

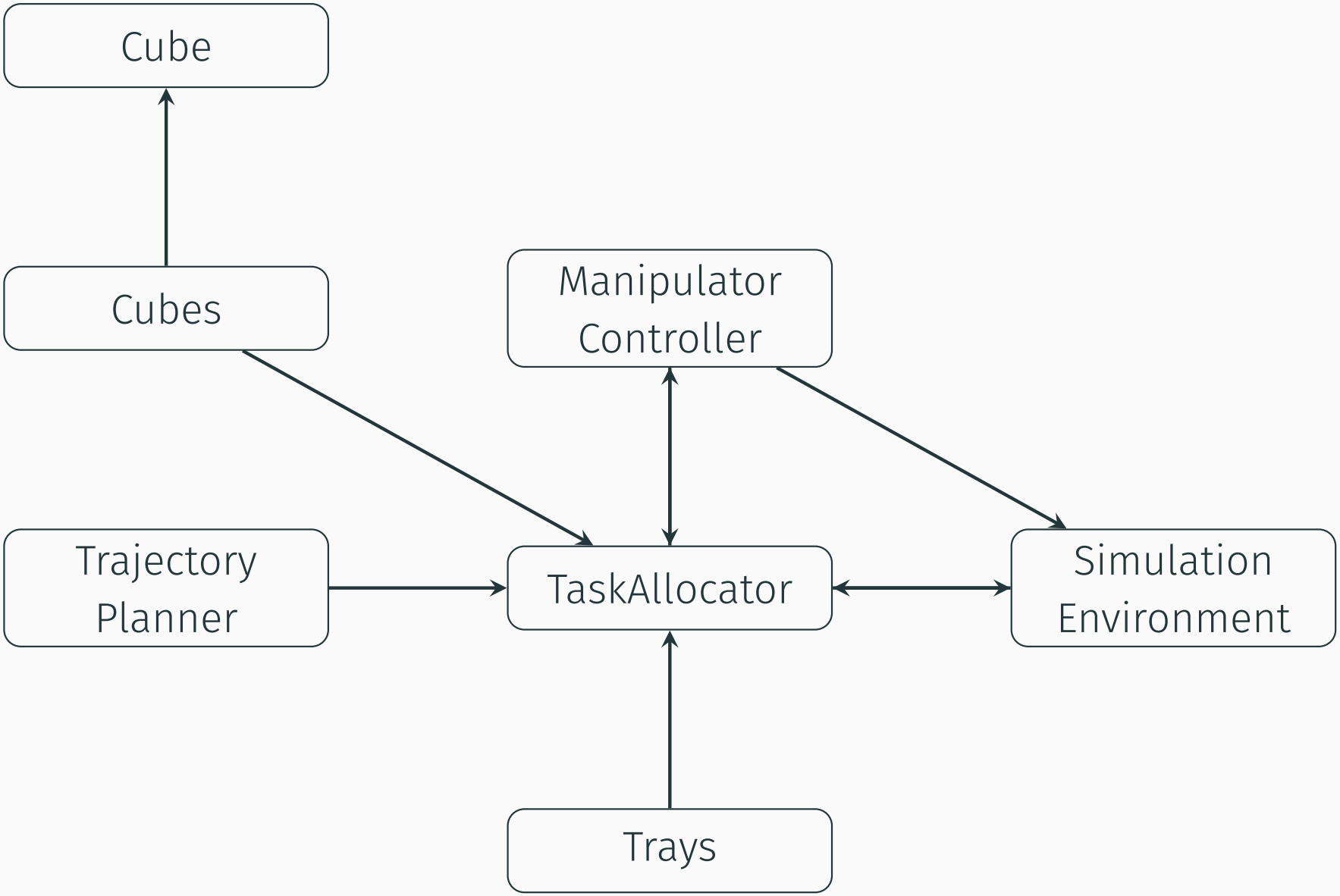


- **Efficient task management:** The Task Allocator effectively manages the allocation of tasks.
- **Prediction capabilities:** By tracking the objects' positions and predicting their future locations.
- **Adaptive to simulation state:** The Task Allocator continuously updates based on the state of the simulation.
- **Flexible trajectory planning:** Integration of waypoint pre-defined or PRM trajectories.

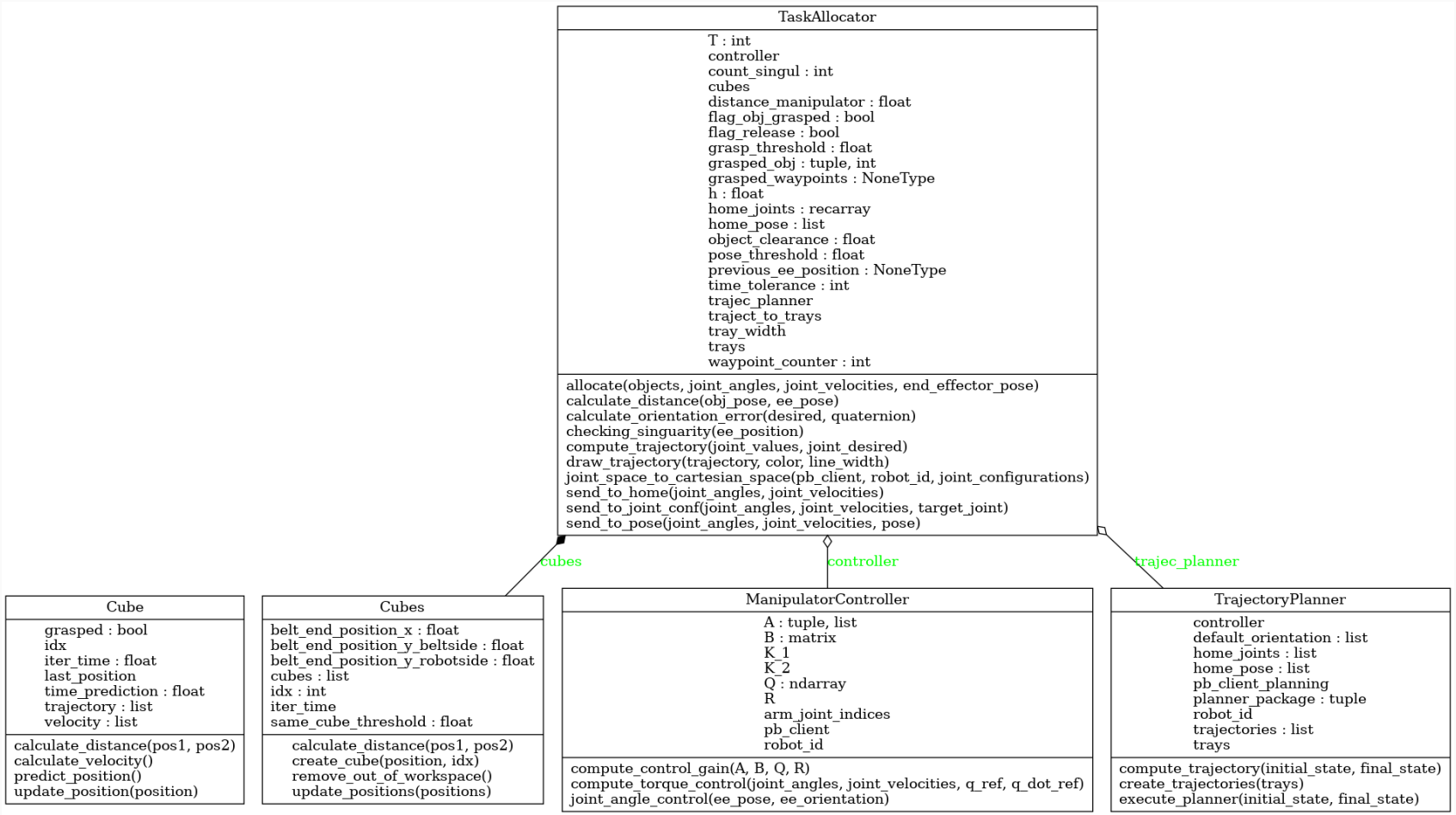
Planner for Collision-Free Trajectories



Solution Architecture



Class Diagram



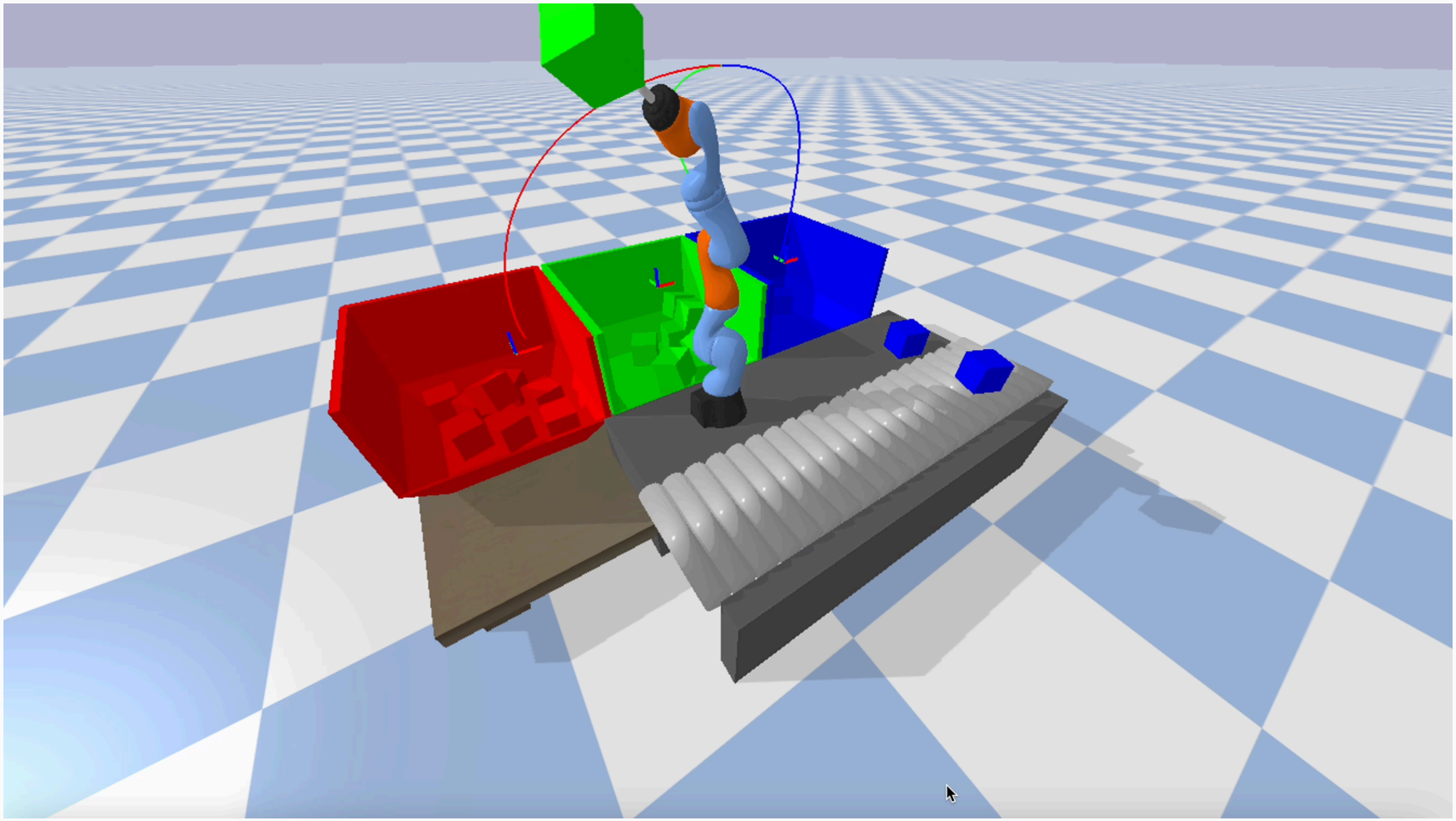
Simplified Pseudocode

Algorithm 1 Simplified Allocate Function

```
1: procedure ALLOCATE(objects, joint_angles, joint_velocities, end_effector_pose)
2:     cubes.update_and_remove()
3:     if not grasping_object then
4:         if length(objects) > 0 then
5:             handle_object_approaching()
6:         else
7:             send_to_home()
8:         end if
9:     else
10:        if last_waypoint_reached and object_released then
11:            reset_grasping_state()
12:        end if
13:        handle_grasped_object_movement()
14:    end if
15:    return torques, q_ref
```

Simulation Results and Discussion

Simulation Results



Discussion

- LQR Controller time response.
- Performance depending on the defined home position.
- PRM pre-computed trajectories to perform monotonous tasks.

Thank you, Questions?