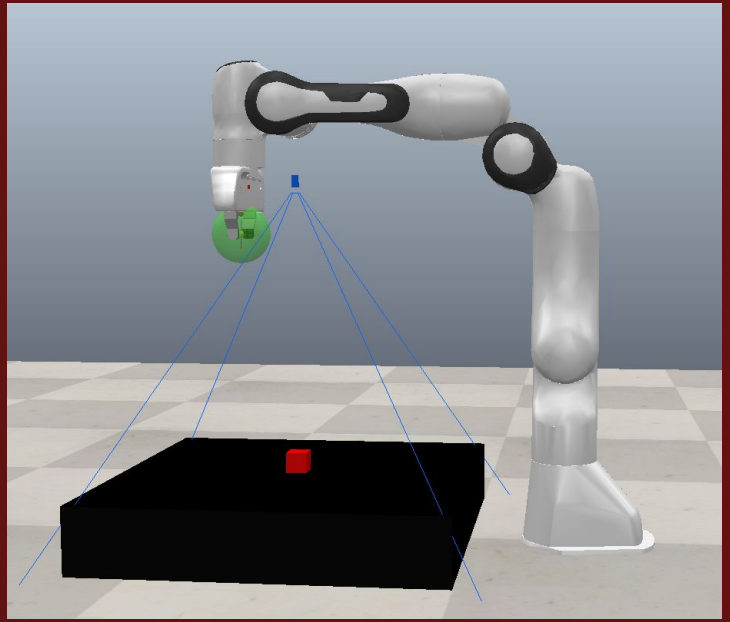
# Autonomous Object Classification and Manipulation Robotic Arm in 3D Simulation

Group 3
Kechang Wan, Xiaorong Wang, Yaodan Zhang and Tiancheng Pu

# What is it?

It is a franka robot arm in an environment that can distinguish objects of different shapes (cylinder, square, rectangle) and different heights, then pick them up to a specific location according to their shapes.

# Initial workspace



# Content

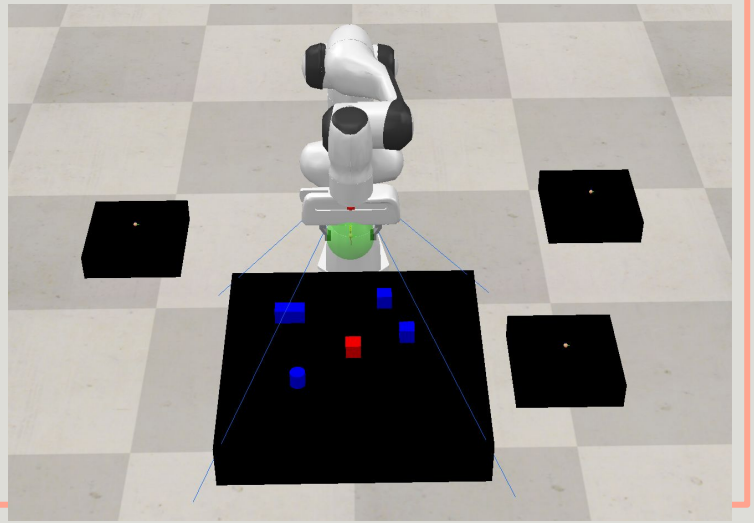
# 01 Demonstration

01. Flat surface
02. Different heights plus obstacles
03. Failure scene

# Demonstration

Scene 1

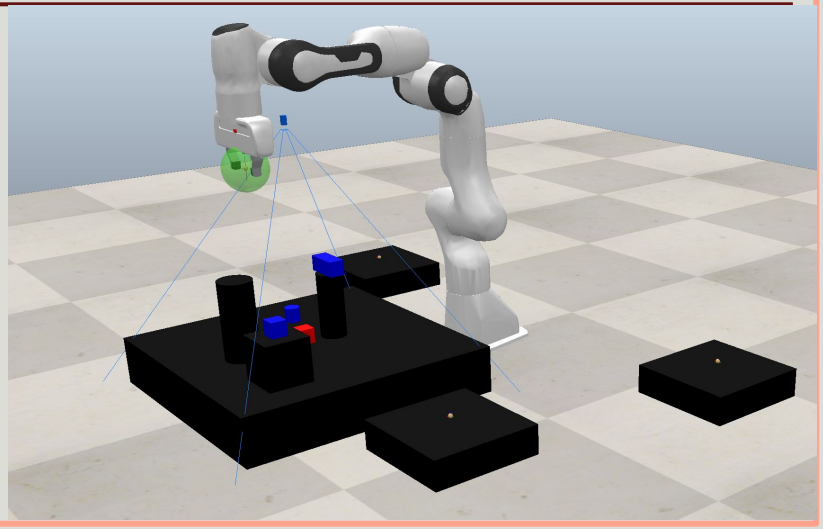
Flat surface



# Demonstration

Scene 2

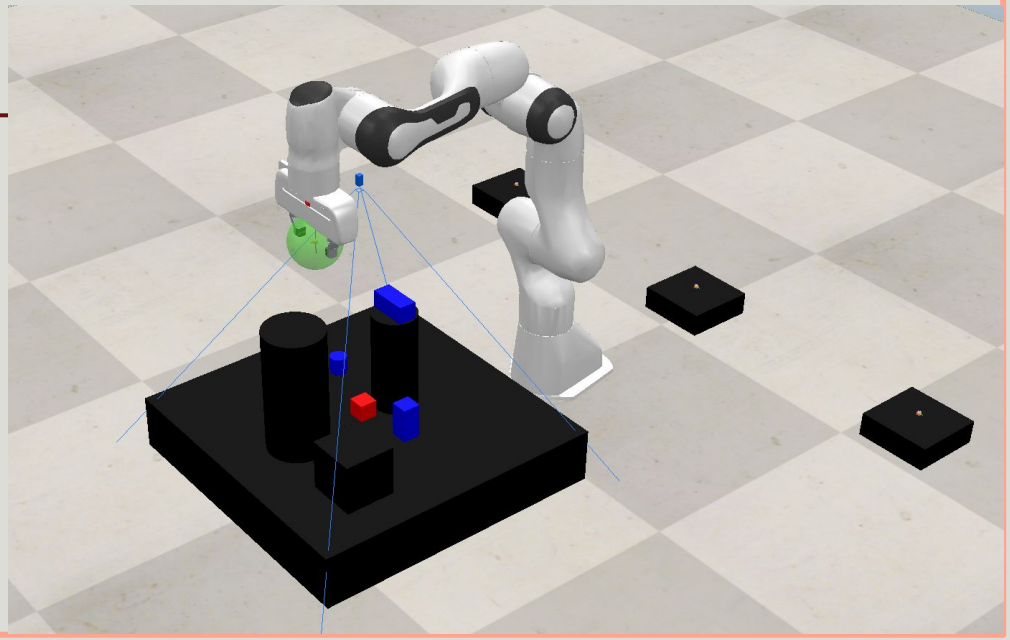
Different heights with obstacles



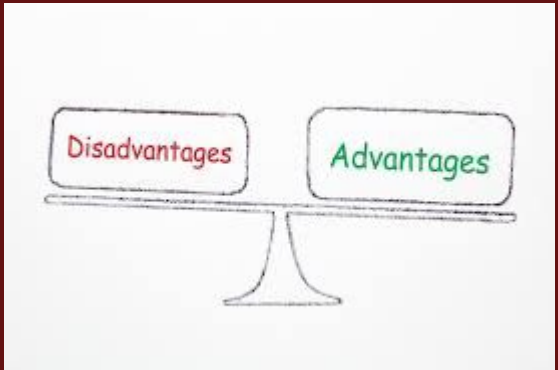
# Demonstration

Scene 3

Failure scene



# 02 Adv & Dis of Robots




01. Advantages

02. Disadvantages

# Adv & Dis of Robots


**Advantage**

1. Can recognize and distinguish shapes with high accuracy
2. Can randomize heights with minimal terrain requirements
3. Fast response time and task completion

# Adv & Dis of Robots

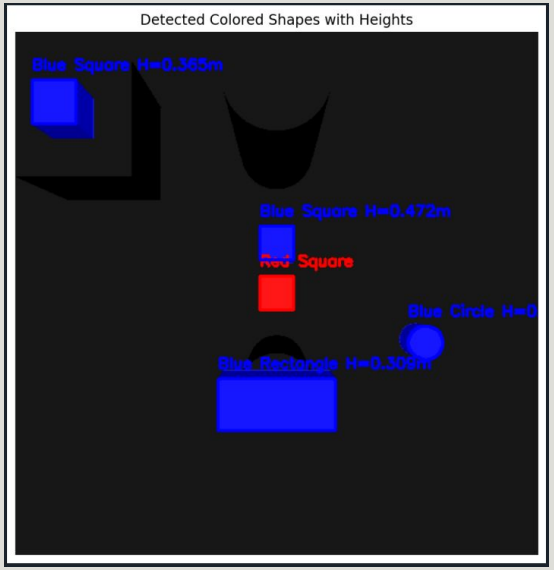
**Disadvantage**

1. Singularity
2. Must have at least one square of a different color as a reference.
3. Higher color requirements (to distinguish shadows)

# 03 algorithm

01. Segmentation
02. Find height
03. Object Grasping
04. Path Planning

# Segmentation




1. What is segmentation?
2. Image preprocessing
3. Color Segmentation
4. Contour Detection
5. Shape analysis

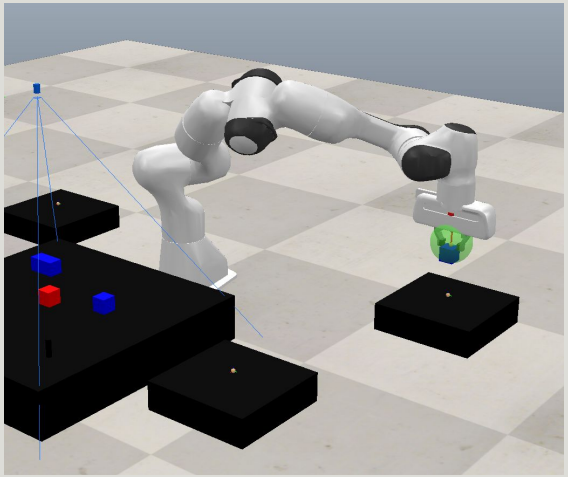
# Finding heights

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

Use references to confirm distances. There is an inverse relationship between the distance of an object from the camera and its size in the photo.

**Using equation from the class? Unfortunately, The z and X or Y in this equation need to be used in the pixel position to camera frame.**


# Object Grasping



Detection
- Uses a proximity sensor to detect objects that near the gripper

Toggle Gripper
- Uses the sim.setObjectParent function to grasp and release the object to the target drop point

# Path Planning

OMPL (Open Motion Planning Library)

*Setup_obstacle_avoidance*

- *sim.ompl.createStateSpace()*
  —to create State Space (ex.[-2, -1.0, 0] to [2, 1.0, 2])
- *sim.ompl.setCollisionPairs()*
  —to define which objects need to be avoided
- *sim.getObjectPose()*
  —Gets the target object's *start_pose* and *goal_pose*

# Path Planning

*execute_obstacle_avoidance()*

- *result, path = sim.ompl.compute(task, 20, -1, 200)*
  - —plan the path
  - —determine whether a feasible path is found
- *sim.stepSimulation()*

# 04 Application


01. Sorting toys or blocks（Lego）

02. Healthcare-Classify and sort medicines or capsules

03. Recycling and waste management

# THANKS