# CMSC477: Robotics Perception and Planning
# Project 2: Sharing is Caring

Prof. Yiannis Aloimonos, Chahat Deep Singh, Levi Burner, Botao He
March 9th, 2023

Due: March 30th, 2023

**We think this project is challenging, we encourage you to try and finish before the deadline.**

## Introduction:

In this project, you will program two robots to autonomously pick up, pass, and place a stack of LEGO's. One robot must remain in the left half of the arena and the second must remain in the right half. The object's initial location will be in the left half and the target location will be in the right half. An animated description is given below:

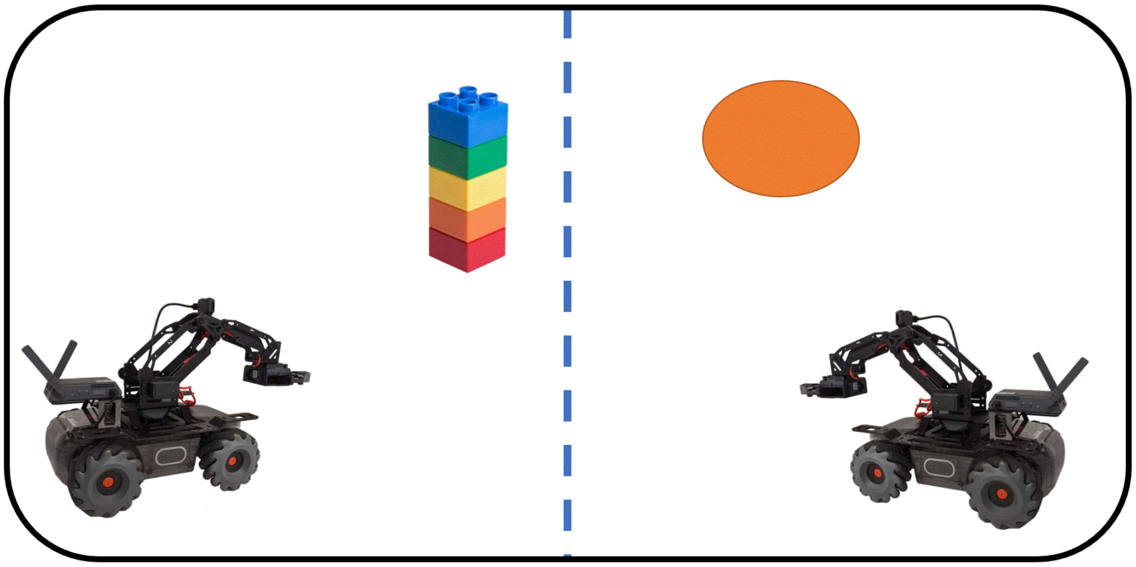

Figure 1: Animation of the pick, pass, place goal for Project 2.

**Requirements:**

- The control sequence cannot be predetermined
  - E.g. the starting locations of everything should be able to be changed.
- Do not use AprilTags for the final demonstration
  - You can use AprilTags before the final demo to help "bootstrap" the system
- The LEGO's should not touch the floor between being picked up and placed in the final destination.

## Suggestions:

You are free to accomplish this project in the manner of your choosing. However, we make the following suggestions:

- Detect a bounding box around the target object (either the other robot or the LEGO's) using the machine learning algorithm YOLOv8
  - You will need to partially retrain YOLOv8 to detect the other robot and the stack of LEGO's as discussed in class on March 9th
    - A tutorial slide deck will be posted over the weekend
  - Because the machine learning algorithm is computationally intensive, you will likely want to use one laptop per robot for this project
  - See the appendix of this document for an example script that will run a pre-trained model
- Use the size of the box to determine the target objects approximate 3D position
  - The most reliable way to do this is likely by using the height of the bounding box in pixels and fitting a linear models to determine the distance
  - You may want to use some averaging or low pass filtering to limit noise
- Adapt the control scheme from Project 1 to control the XY position and heading
  - Use the blue line in the center of the arena to get an approximate heading estimate and align the headings of the two robots.
  - You will need a simple search procedure to find the LEGOs and ending location
- Use HSV color thresholding to find pixels on the blue line or the orange end position.
- Devise a method of synchronizing the grasping of the two robots, e.g. wifi messaging, physical gestures, vibration sensing, etc. This is a great area to be innovative.
  - A good library for messaging is ZeroMQ, see the example here: [https://zeromq.org/languages/python/](https://zeromq.org/languages/python/)

## Grading:

- Successfully pick up the object 10 points
- Successfully pass the object: 60 points
- Successfully place the object: 10 points
- Report: 20 points
- Bonus: Accomplish the goal with a decentralized algorithm (no digital communication between the algorithms running each robot): 10 points

## Submission Guidelines:

Please submit a video and PDF report. The video should show your team's demonstration and any relevant animated visualizations. The report should have the following sections:

- Introduction
- Block Diagram
- Link to and description of demonstration video (Google Drive or YouTube)
- Methodology
- Results
- Conclusion
- Appendix A: Code Listing

## Collaboration Policy:

You can discuss the assignment with any number of people. But the report and code you turn in MUST be original to your team. Plagiarism is strictly prohibited. A plagiarism checker will be used to check your submission. Please make sure to cite any references from papers, websites, or any other student's work you might have referred to.

## Appendix:

To get the machine learning algorithm running using the image from the robot:

- Install the Ultralytics YOLO algorithm package: **“pip install ultralytics”**
    - If you have a laptop with an NVIDIA GPU you can run the machine learning algorithm much faster by installing CUDA enabled PyTorch
- Run the following code:

```python
from ultralytics import YOLO
import cv2
import time
from robomaster import robot
from robomaster import camera

model = YOLO("yolov8n.pt")

# Use vid instead of ep_camera to use your laptop's webcam
# vid = cv2.VideoCapture(0)

ep_robot = robot.Robot()
ep_robot.initialize(conn_type="ap")
ep_camera = ep_robot.camera
ep_camera.start_video_stream(display=False, resolution=camera.STREAM_360P)

while True:
    # ret, frame = vid.read()
    frame = ep_camera.read_cv2_image(strategy="newest", timeout=0.5)
    if frame is not None:
        start = time.time()
        if model.predictor:
            model.predictor.args.verbose = False
        results = model.predict(source=frame, show=True)
        end = time.time()
        print(1.0 / (end-start))
```