
Assignment 1: Modular Sense-Plan-Act approach for Vision-based manipulation

EN.601.498 / EN.601.698 · Hands-on Robot Learning · Fall 2026

Weight: 10% · In-class demonstration: ~~October 29, 2026~~ (during the second half of the lecture)
November 5, 2026

Objective

Program the SO-101 arm to pick up a red cube and place it in a blue tray. For this assignment, you will use only the follower arm. The exact cube and tray will be distributed by the course staff starting next week (please expect announcements on how to pick up the materials). Teams should build a modular ‘sense-plan-act’ program that does not use a learned policy for action generation. Instead, the system must have a clearly identifiable ‘sense’ (perception) component, a ‘plan’ (planning) component, and an ‘act’ (low-level action generation / control) component. The system must use only RGB images (and additionally robot proprioception) to locate the cube and tray, plan the arm's movements, and complete the task autonomously. This assignment gives you practice programming the robot's actions directly. In (the first part of) Assignment 2, you will teach the robot the same task through demonstrations.

Example workflow

The task follows these basic steps:

1. **Find the objects:** Use the camera image to determine where the red cube and blue tray are (hints: consider blob/color detection, or other object detection; you are allowed to use learning for just the perception module, although the course staff expect the task to be reasonably well-solved without the use of any learning).
2. **Plan the movements:** Decide where the arm should move, to pick up the cube and place it in the tray.
3. **Pick up the cube:** (name says it all; execute the pick up cube action) teams must implement an explicit output variable (could be a print statement or a text or display graphic) that indicates their system understands that the cube has been picked up.
4. **Place the cube:** Once cube is picked up, it should be placed onto the tray. Teams must implement an explicit output variable (again, a print statement or other display text / graphic) to indicate their system recognizes the task as being complete

Task and some helpful hints

- **Success criteria:** At the start of a trial, the team members (or course staff) will place the cube well-within the robot's workspace. A trial is considered successful if the cube is successfully placed into the tray, and the gripper retracts away.
- **Sensing:** We expect most teams to rely solely on the wrist camera for this assignment (and teams that experiment with a multi-camera setup will quickly appreciate as to why). Use of another external (RGB-only) camera is allowed, but the system must still be able to complete the task if the course staff occludes/covers the external camera (we assume that the wrist camera will always be unobstructed). We strongly recommend teams attempt this task with just the wrist camera, for these reasons. For this assignment, depth measurements, depth-camera streams, and stereo-derived

or learned depth estimates are not permitted. That said, teams may want to consider estimating object pose from RGB images, calibration, and assume known object geometry. Teams are allowed to use fiducial markers if needed and augment the robot workspace. Note that, if using such fiducial markers, teams will want to spend some time before the demo calibrating / setting up the robot workspace.

- **Computation:** The entire system must run solely on a standard laptop CPU. GPU-usage or remote-server inference is not permitted for this assignment.
- **Environment:** You may use a small number of visual markers, such as ArUco or AprilTag markers to help determine where the camera is relative to the robot workspace. You may also assume a known tabletop height or similar measurements. Explain where you place any markers, what calibration is needed, where objects can be placed, and any requirements for the background or lighting. Do not alter the supplied objects in ways that change the manipulation task.
- **Other helpful tips:** Start out small – we recommend debugging with simple test cases first (i.e., cube is close enough to the gripper). We also recommend testing out each subtask in isolation first (e.g., one completely plausible decomposition is goto cube, pickup cube, move cube to tray, place cube in tray). Test your system multiple times with the cube and tray in different positions. For each trial, record the starting setup, whether it succeeded, how long it took, and what went wrong if it failed. Report how many trials you ran and which positions you tested; a single successful video is not enough to show that the system works reliably. During the in-class demonstration, course staff may move the objects within the conditions your team has specified to check how well the system works.

What to submit

Each team should submit:

1. **A report of at most one page:** Explain how your program works, what conditions it needs, how well it performed in testing, and what went wrong. Include a photo or sketch of your setup.
2. **Test results and videos:** Include a table of your test results and videos showing at least 10 tests. The tests must span varying degrees of complexity. Be sure to include failure results. At least 2 of the 10 submitted tests must showcase failures.
3. **Team contributions and AI use:** Briefly describe what each member did and which AI tools you used for which tasks. (3-5 sentences is okay).

Awards and criteria

- **Least-strings-attached award:** For a system that works reliably with the fewest requirements for object placement or the surrounding environment, with both objects within the arm's reach.
- **Best-performing system award:** For a system that completes the task successfully and efficiently.

These awards are primarily for recognition. They are highly correlated with the grade teams would earn on this assignment.

AI policy: You may use AI tools for coding, debugging, analysis, and writing. Check their outputs, cite any external work you use, and make sure you understand everything you submit.