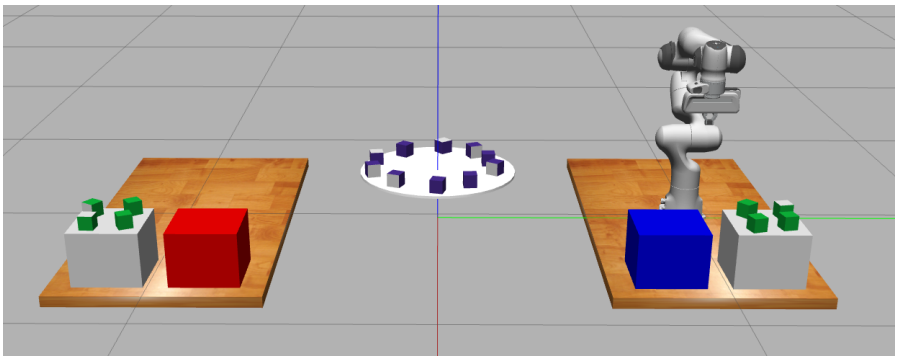
Final Project: Pick and Place Challenge

Project Proposal Due: 11/19/21 @ 11:59pm
In-Class Presentations: 12/07/21 and 12/09/21
Final Demo/Competition: To Be Announced
Report Due: 12/10/21 *(No-Penalty Extension until 12/14/21 @ 11:59pm)*



MEAM520 culminates in a final project which will allow you to leverage all the concepts and skills you've learned throughout the semester, applying them to perform the practical task of picking up objects in your environment and placing them where they need to go! Your task is to develop a robust and reliable solution which can acquire blocks (either stationary or in motion) and stack them on a goal platform, right-side up, in as tall a tower as possible. Your deliverables for this assignment are:

1. A **Project Proposal** describing your preliminary ideas for how you will tackle the task, what challenges your foresee, and what ideas you wish to explore in your solution
2. An **In-Class Presentation** which describes the approach you took to solving this challenge, any outstanding shortcomings of your approach, and some preliminary results of your testing
3. Participation in the **MEAM520 Final Demo/Competition**, in which your robot will go head-to-head with that of other teams in a shared pick-and-place environment using the real Panda arms. This demonstration and competition serves as an extended opportunity to test the performance and reliability of your solution as well as its ability to adapt to changes in the environment. The top-scoring teams will also receive a small amount of extra credit!
4. A **Final Report** which presents in detail the project goals, the approach you took to solving this challenge, and your evaluation and analysis of its successes and shortcomings

You will also submit your code as supplementary material, but it will not be graded against an autograder. You will submit all materials through Gradescope. **All students must work in a team of 4 students (except for a small number of teams due to class size). These team size requirements are strict. Please check out Piazza to help with team formation.**

Most testing for this project will be done in a specially crafted Gazebo environment, but you will have some opportunities to test your solution on the real robots before the Final Demo/Competition (subject to the time and availability constraints). The report and code are due Dec 10 due to University policy, but **a no-penalty extension will automatically be granted to all students until Dec 14 at 11:59pm. After this time, late submissions will be accepted per the standard late submission policy, but they will be penalized by** 25% **for each partial or full day they are submitted past the time of the no-penalty extension.** After the late deadline, no further assignments may be submitted; post a private message on Piazza *before the (no-penalty extension) deadline* to request an extension if you need one due to a special situation such as illness.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. **If you make use of outside sources, you are expected to cite them in your report.** When you get stuck, post a question on Piazza or go to office hours!

1 Rules and Specifications

Below we describe the structure of the Pick and Place Challenge, from the basic ground rules to the ways you can score points. Your task is to maximize the score your robot can reliably achieve during a round of the game!

1.1 Ground Rules

1. **Students are required to work in teams of four.** You are welcome to form your own team, and can also post on Piazza to look for other teammates. If you are having trouble forming a team, please reach out to the teaching team on Piazza. A small number of 3-person teams will be necessary, but must obtain permission to do so in advance.

2. Each match will consist of two teams (designated Red Team and Blue Team) competing head to head in a shared environment, shown above.

   (a) The base of each robot is located `.978m` from the origin of the world frame, on the world's `y` axis. The axes of the base frame of each team's robot's is parallel to the world frame's axes. The robots are positioned such that their workspaces just barely do not overlap.

   (b) **No point on the robot (except the base) is ever permitted to go below an invisible horizontal planar barrier with an altitude of `.200m` in the world frame.** This will be enforced in software in the real-world environment.

   The static environment will not change, but the arrangement of scoreable objects will be randomized before the start of each match, and your opponent will also have access to the shared dynamic blocks.

3. **Matches will last for 3 minutes.** Teams will activate their robot at the start whistle, and their robots will be halted with the Software Stop at the final whistle.

4. **Teams must only interact with the robots through a single ArmController object for their robot.** Teams are forbidden from directly sending or receiving data over any ROS topics or services, in order to create a level playing field for all.

5. **Teams must operate the physical robot in a safe manner *at all times*.** The judgment of whether a robot is operating safely (i.e. without erratic motions that could harm the robot, people, or the environment) is solely up to the teaching team. **If at any point during testing or demo/competition, the robot is deemed to be operating in an unsafe manner, the teaching team will not hesitate to Software Stop the robot, *without restarting the match*.** Accordingly, teams should make sure their robot operates in a way that is deliberate and can be deemed perceptibly safe to a reasonable observer.

1.2 Scoring Points

1. Scoreable objects are `50mm×50mm×50mm` wooden blocks, which come in two varieties:

   (a) **Static Blocks:** These objects are randomly dispersed on a stationary platform with its surface `.200m` above the world's `x-y` plane, with its center `1.147m` from the world frame `x` axis and `.562m` from the world frame `y` axis.

   (b) **Dynamic Blocks:** These objects are arranged around the edge of a rotating turntable with radius `.305m`, centered at the world frame origin with its surface `.200m` above the world's `x-y` plane.

   All scoreable objects have one side that is white (in simulation), corresponding to the `+z` axis of that object's coordinate frame.

2. During the match, teams will manipulate objects to move them onto the goal platform matching their team's color (Red/Blue), located `.809m` from the world frame `x` axis and `.562m` from the world frame `y` axis. This goal platform seen from above is `.250m` × `.250m`, with its surface `.200m` above the world's `x-y` plane.

3. Each individual scoreable object supported by a team's goal platform will receive points according to the formula `Points = Value × (Altitude + SideBonus)`, where:

   (a) `Altitude` is the distance from the center of the object to the surface of the goal platform in millimeters

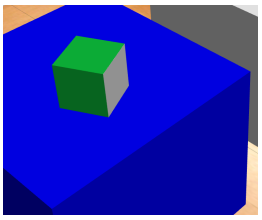
   (b) `SideBonus` is `50` if the white side of the object is pointing upward, `0` otherwise.

(c) `Value` is `1` for Static Blocks and `2` for Dynamic Blocks.

The team's final score is the sum of the point value of each scoreable object.

4. **All points will be scored after the match concludes, so that only the final state of the blocks determines the awarded points.** Below are some example final configurations of blocks and their corresponding point values.

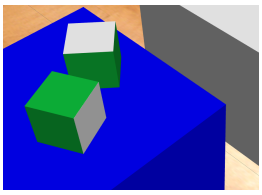
Example 1
static block, no bonus



individual blocks:
`1 × (25 + 0) = 25`

Total Points: `25`

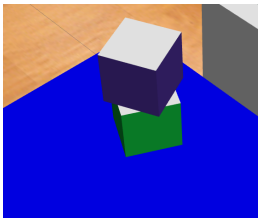
Example 2
two static blocks side by side, one bonus



individual blocks:
`1 × (25 + 0) = 25`
`1 × (25 + 50) = 75`

Total Points: `100`

Example 3
dynamic block on top of static block, both bonus



individual blocks:
`1 × (25 + 50) = 75`
`2 × (75 + 50) = 250`

Total Points: `325`

1.3 Tournament

1. The exact tournament schedule will be announced via Piazza. The tournament will consist of two phases:

   (a) **Qualification Rounds** *(5 matches per division, each team plays 2 times)*

      i. The class will be divided into 4 divisions, each containing 5 teams.
      ii. Each team will play against two other randomly assigned teams in their division
      iii. The two teams with the best record will advance, determined successively by:
         - Number of wins
         - Total points scored by the team
         - Total points scored by the team's opponents in each round

   (b) **Single Elimination** *(7 matches total)*:

      i. The remaining 8 teams will go head to head, with only the winner of each match advancing to the next round
      ii. The last Panda standing is the champion!

2. In the event of a tie, the winner of a match will be determined successively by

   - `Altitude` of the highest scored block
   - Number of Dynamic blocks scored
   - Sum of `SideBonus` over all scored blocks
   - Coin flip, rematch, or TA judgment as a last resort

2 Project Proposal

The goal of the Project Proposal is to get you started early thinking about your solution and working with your team.

Note: Unlike the Prelabs, this assignment should be completed with your team, not individually. Submitting the Project Proposal with your team constitutes the official formation of your team of four students. If you would like to register a team of 3, please seek advanced approval from the teaching staff via Piazza as soon as possible, since we have limited slots for 3-person teams due to the number of students in the class.

Your project proposal should have the following information:

- **Goal:** Summarize in 2–3 sentences what you want to accomplish
- **Approach:** Describe in 2–3 sentences what techniques you plan to use
- **Anticipated Challenges:** Describe in 2–3 sentences what parts you think will be harder than others.

The project proposal is a completion grade worth 5 pts on Canvas. Some topics you *might* wish to consider include (but are certainly not limited to):

1. Will will your overall strategy balance performance, implementation difficulty, and the actions of your opponent?
2. Given desired manipulation of a block, how can you plan a robot motion which achieves it? How might you handle acquiring blocks that are in motion?
3. Will algorithms you implemented during the semester (e.g. IK, RRT) perform quickly and robustly enough for use in your project? How could you improve their robustness?
4. What other extensions on the basic functionality you've already implemented might be necessary to perform these tasks?
5. How will you address the challenges introduced by the gap between simulation and reality?

3 Simulation Development and In-Lab Testing

We will adopt our usual workflow, where you develop your solution in the simulated environment and later test it on the real robot. The Final Demo/Competition will take place with two real robots.

3.1 Simulation Setup

As usual, update your own private fork of the `meam520_labs` repo for the Final Project:

```
$ cd ~/meam520_ws/src/meam520_labs/
$ git pull upstream master
```

You will also need to install a new ROS package:

```
sudo apt install ros-noetic-velocity-controllers
```

To launch the simulated testing environment, with randomized block arrangement, run **either** of the below commands to test as that team.

```
$ roslaunch meam520_labs final.launch team:=red
$ roslaunch meam520_labs final.launch team:=blue
```

The script with which you will control the robot is located at

```
~/meam520_ws/src/meam520_labs/labs/final/final.py
```

and contains some starter code which sets up the arm controller, automatically determines the team you are playing, and waits for you to press enter to begin.

3.2 Controlling the Robot

Important: To avoid self-collision and collision with tables, in this project the **ONLY** commands you are allowed to use to control the robot are the following "safe" methods of the `ArmController` class, which function similarly to their non-"safe" counterparts but with additional checks against environmental collision, etc.

- `safe_move_to_position(q)`

  Under the hood, this command is calling a trajectory planner which moves the arm from one position to another position along a linear path through configuration space. For usage examples, you can consult: Lab 0, Lab 1, Lab 3, Lab 4, and Robot Lab 1.

- `safe_set_joint_positions_velocities(q, qd)`

  This function **directly** updates the setpoint of the robot's joint controllers to track a desired position and velocity. Because of this, it must only be used with **smoothly** varying inputs, so that the commanded setpoint is never far away from the current value. **Note:** it is totally reasonable to send very small position updates along with a commanded velocity of zero. If you send a command which differs too much from the current state, your command will be rejected. You can consult `follow.py` in Lab 2 for usage examples.

- `exec_gripper_command(pos, force)`

  This command directs the gripper to close to a width of `pos` meters, and apply up to `force` newtons to do so.

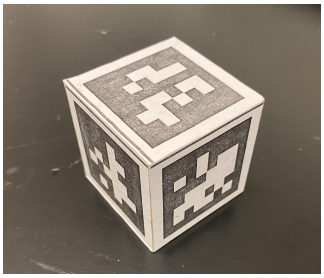


Figure 1: A scoreable block covered in AprilTags

3.3 Detecting Blocks

In order to manipulate the blocks in the environment, you must first know where they are located! To make this possible, each block in the real-world environment will have a unique AprilTags (visual fiducials) on each face of the block. To be clear - all static blocks use the same 6 tags, and all dynamic blocks use another set of 6 tags, but no two faces of a single block have the same tag on them.

The vision which tracks these tags in the real world consists of an overhead camera mounted on 80x20 aluminum extrusion. The teaching staff has calibrated the camera, and is using OpenCV to estimate the position and orientation of each **tag** visible to the camera. In particular, the system determines the homogeneous transformation **from the tag coordinate frame to the camera coordinate frame**. In simulation, we use the ground truth pose of every block to generate virtual tag detections. We limit the returned detections only to block faces which are pointing toward the camera (within an angular tolerance) and which are within the field of the camera (another angular tolerance).

In Fig. 2, we see a schematic of the blocks, along with the coordinate systems of the tags that will be attached to each face. Likewise, in 3, we see images from the vision system, detecting the tags on faces of a Dynamic Block. Your task is to estimate the actual pose of the block, or whatever information you need to manipulate the blocks to score points. **The key observation to make is that due to the consistent arrangement of tags, given the actual pose of a single tag, it is possible to estimate the full pose of the block as a whole.**

It's vital to keep in mind that while the simulation data is "ground truth", meaning it has virtually no noise, the real data generated by the vision system in performing AprilTag detection can be **extremely** noisy, and will likely not be very usable in its raw format. This is particularly true of the *depth* (distance away from the camera) and the *orientation*, while much less true of the *bearing* (a unit vector pointing from the camera in the direction of the block). We encourage you to ask what *priors* (structural information about the environment) you can leverage to improve your estimates. For a block sitting on the platform or turntable, ask yourself questions like:

- What orientations are even permissible?
- What information does the height of the table give me?
- How does the block pose evolve (or not) over time?

Using these observations will help you produce much more accurate block pose estimates.

In addition to the AprilTags on the faces of the blocks, there is also another tag, `tag0`, which is affixed to the surface of the table, with its coordinate frame parallel to the robot's based coordinate frame, and its origin translated `.500m` in the `-x` direction (behind the robot). Because the poses of detected tags are estimated in the camera frame, you will need to make use of this additional tag to estimate the pose of the objects relative to the robot.

To read detections from the (real or simulated) vision system, use the following:

```
from core.interfaces import ObjectDetector
detector = ObjectDetector()
for (name, pose) in detector.get_detections():
     ... # do something
```

This method returns all AprilTags found in the most recent frame. This example is also provided in `final.py`.

Note: The camera cannot see through the arm, so if the arm is occluding tags, the camera will not detect them. Occlusions by the arm or other blocks are not modeled in simulation.

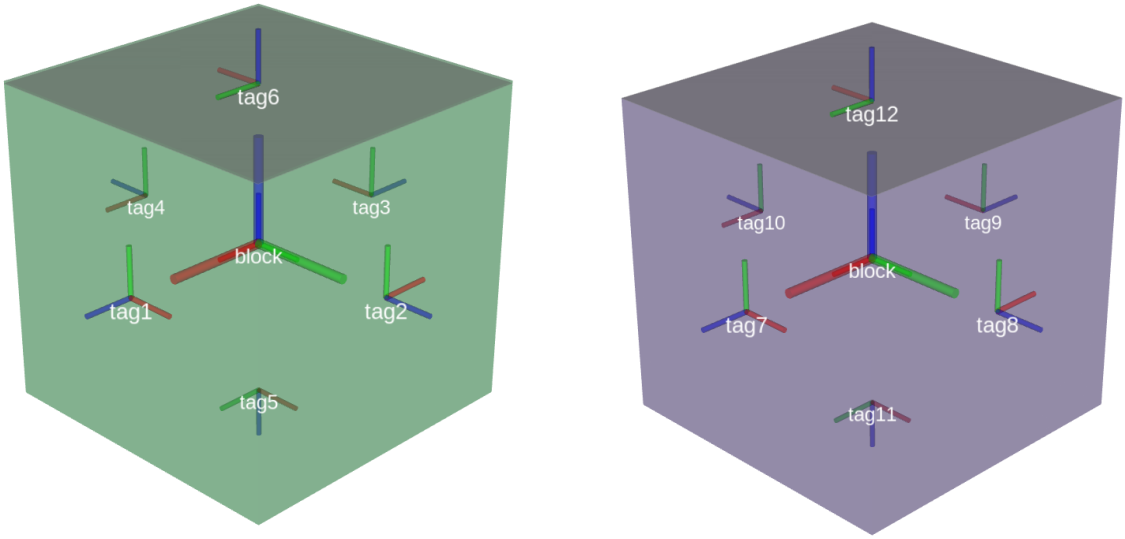


Figure 2: These figures show the arrangement of AprilTags on scoreable objects. All Static Blocks have the same tags attached with the same orientations on the same side of the block, relative to the block coordinate frame. The same is true for all Dynamic Blocks. In this figure, $XYZ \leftrightarrow RGB$, so both blocks have their `+z` axis pointing upward (i.e. in the `SideBonus` configuration, a Static Block will have `tag6` on top, while a dynamic block will have `tag12` on top).

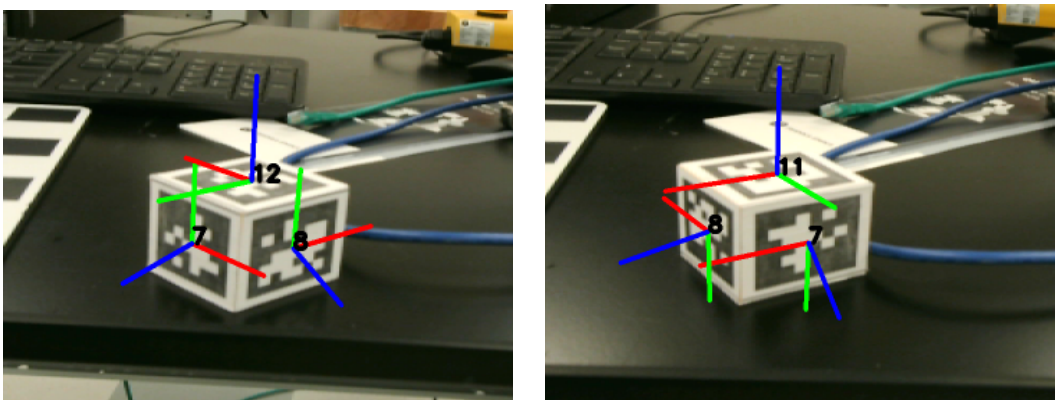


Figure 3: Here, the vision system detects the tags on three sides of a Dynamic Block in each photo. You can verify that the various tag coordinate frames are consistent with the schematic above.

3.4 Timing

Because your simulation does not run with a Real-Time Factor of exactly 1, it's important that any time-based reasoning your code performs uses the current ROS time, not the Wall time (i.e. the time that a clock on the wall of the lab would read). Check out the function below that we provide for this purpose.

```
from core.utils import time_in_seconds
...
t = time_in_seconds() # matches ROS time
```

That way, in simulation `t` will match the Gazebo time, while on hardware it will match the robot time i.e. realtime.

3.5 Transferring Your Code

We will use Google Drive to allow each team to quickly transfer their `lib` and `final` folders to the lab computers during the demo/competition (and testing). Stay tuned for more details on Piazza on this front.

3.6 Testing on Hardware

We anticipate the bulk of your development and preliminary testing will happen in simulation. Once you are relatively confident your code works in simulation, it is our hope that you and your team will reserve time in lab to do additional testing to practice your strategies on the robots. Note: while in lab you saw things work in simulation and then work the same in the real world. During this challenge, there are additional sources of noise which may impact the performance of your method on the hardware (e.g. noise in the perception). So it will be important to practice on the hardware.

Every member of your team must have completed Robot Lab 0 and Robot Lab 1 to test on the robot, and at least 2 members of your team must be present to operate the robot. We will expect that you and your group mates respect all of the safety instructions covered. Each group will get one warning if safety rules are broken. The second violation will result in you and your team being asked to leave no matter where you are in the reservation time.

We will ask that you use the same pipeline on the robots that you used in the robot labs: **Test in simulation, and then test on the robot.** Because there will be no way for the teaching staff to verify your code ahead of time as we did with your earlier lab submissions, we will ask that you show the TA your code working as expected in simulation on the lab computer. Once you have an ok from the TA you are good to run tests on the robot. It will be important for the team member manning the software stop to be vigilant and stop at any time they are concerned something may not be working on the robot as they expect.

You and your team will be allowed **four 30 minute reservations a week**. This lab time is yours, and if you want to discuss your strategy with a TA or test on the robot you are welcome to. We will be strict about following reservations and kindly request your cooperation in promptly wrapping up before the end of your slot.

4 In-Class Presentation

The goal of the in-class presentation is to give each team the opportunity to describe and share their work with the rest of the class. Similar to the proposal, this assignment should be completed with your team, not individually. Your in-class presentation should cover the following topics:

- **Methodology (3 points):** What are the component technologies you plan to employ for the final project? How are the component technologies integrated? What are the design choices that went into these decisions? What is the architecture of your proposed planning and control framework?
- **Experimental Evaluation Plan (4 points):** What is your test and evaluation plan (this should consist of a combination of simulation tests as well as physical tests with the robot)? What is the setup and the procedures you are using to test and evaluate the various subcomponents and the integrated system? How are these tests designed? What are the metrics you are using to evaluate success?
- **Results and Discussions (3 points):** Report any preliminary results and analysis you have done and how these preliminary results will impact your work forward.

The project presentation is worth 10 points. Presentations will be scored by your peers as well as the teaching team. Your team's final presentation grade will be determined by a weighted average of the two (60% teaching team and 40% peers).

5 Report

You will submit a written report describing your work in this project. **The report must be no more than 12 pages in Times New Roman 11pt font with 1-in margins, not including references.** You are free to use whatever typesetting workflow you are comfortable with, but we highly recommend using LaTeX. Handwritten submissions will not be accepted. If you need to sketch some diagrams by hand, this is okay.

5.1 Report Rubric

The report will be graded on a more holistic rubric than previous reports, given the wide variation which will be present in students' solutions. Regardless, your report should be thorough, addressing in detail all aspects of the task at hand, your approach, your implementation, the testing process, and analysis of the results. **The gold standard is that another team who completed the prior labs in the course would be able to reproduce your approach in the final project from your report alone.**

1. **Completeness** *(5 pts)*: Did the report address all relevant questions with no obvious holes?
2. **Method** *(10 pts)*: Was the approach technically sound and reproducible? Was it complete and free of error or bias?
3. **Evaluation** *(10 pts)*: Were all relevant results reported? Are the cases chosen sufficient to demonstrate advantages and limitations?
4. **Analysis** *(10 pts)*: Was the analysis complete, free of error, and based on data/observations?
5. **Lessons Learned** *(5 pts)*: What are some insights that you gained? What worked well, what did not? Why?
6. **Clarity** *(10 pts)*: Was the report clear and organized?

We wish to emphasize that the report is the **primary deliverable** for the Final Project, since your performance in the Final Demo/Competition does not directly impact your grade, rather serving as an opportunity to evaluate the performance of your solution in a practical setting.

5.2 Submission via Gradescope

- **Group Submission:** Each group should only submit once to Gradescope per Gradescope assignment. While going through the submission please ensure that you **add your group members to the submission**. Failure to do so could impact the grades of all parties involved.
- Please submit a PDF of your report to the Report assignment, and your code to the Code assignment (solely as supplementary material).