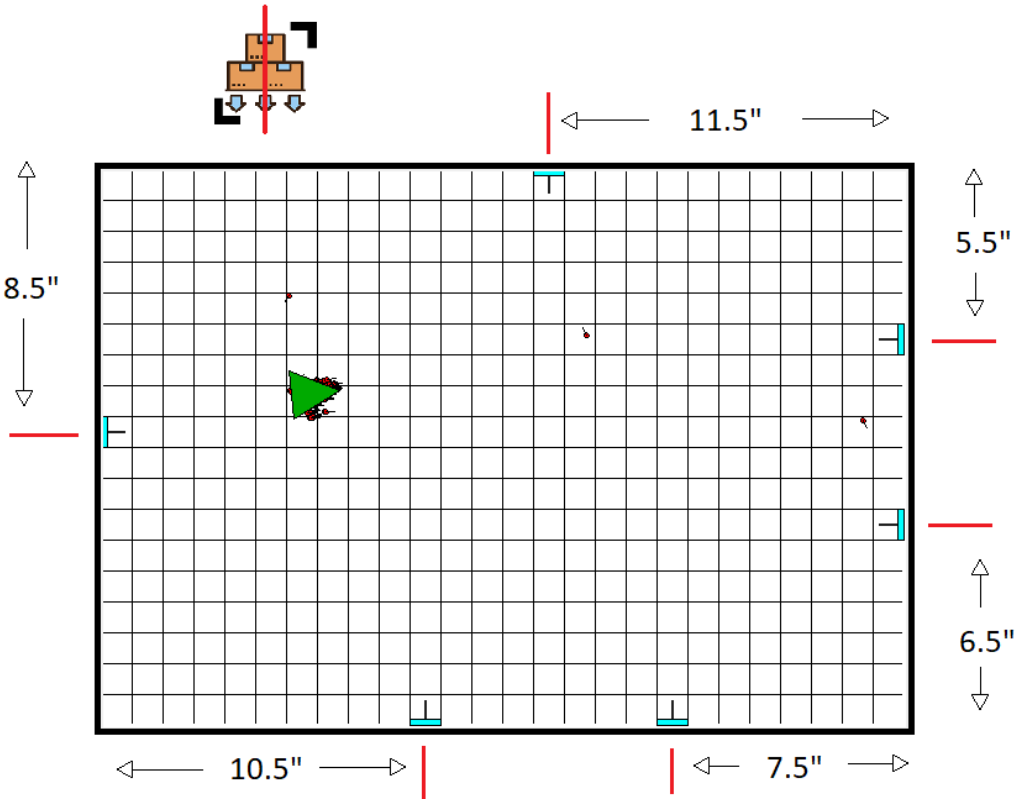
LAB 4: PARTICLE FILTER PART II

Due: November 6th, 10:45 am

In this lab, you will build on your Lab 3 solution to enable Cozmo to localize within its arena using a Particle Filter.

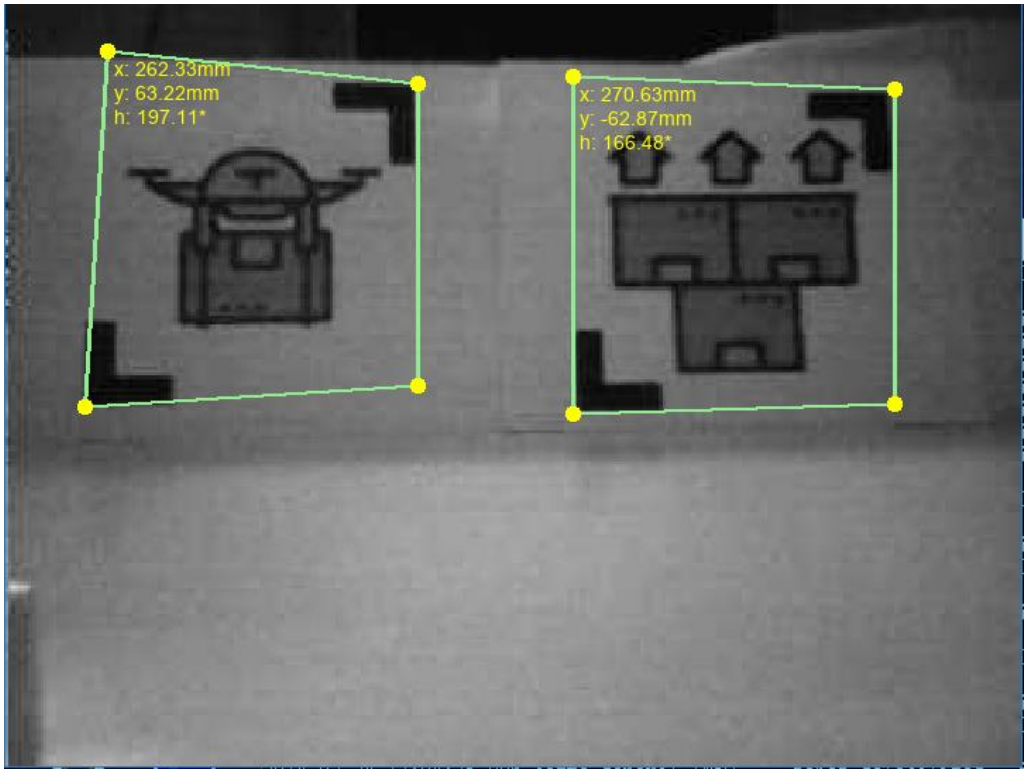
Setup: Before you begin, you will need to add localization markers to the arena. We will use the symbol cards that you already have in place of localization markers. Attach the cards to your arena as shown in the figure below (the red lines should be aligned with the center of the symbol cards):



Important note: Unlike previous labs, in this assignment we will treat all symbols as interchangeable. In other words, all the robot needs to know is that it saw "a marker/symbol", but not which one. Do not hard code the locations of the symbols into your code since we will randomize their location during grading.

Marker detection: We have provided marker detection code for your use, located in the `markers/` directory. We do not expect you to edit this code, but if you choose to do so please submit your modified version. To test the code, run `python3 test_marker_detection.py` with Cozmo on. Place the Cozmo about 10 inches from a marker; you should see the marker become highlighted in

the image window if it is recognized. Spend a few minutes experimenting with this capability to gain an understanding of the distances and orientations at which the robot is able to detect the marker.



Localization: The main component of the lab is to use your particle filter from Lab 3 to enable the robot to 1) determine where it is, and 2) use this information to go to a predefined goal location on the map. We have provided a starter file, called `go_to_goal.py`, which contains the following functions:

`marker_processing()`: waits for a camera image, runs marker detection on the image, and calculates the position and orientation of the detected markers

`compute_odometry()`: calculates the robot's odometry offset since the last measurement

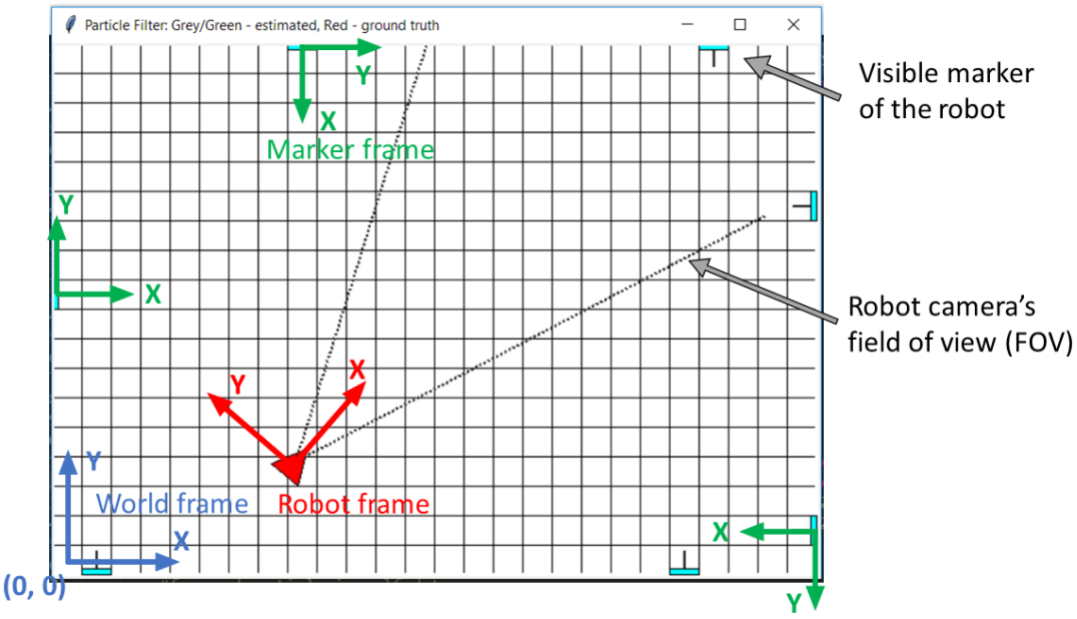
`run()`: the main processing loop, enter your code here

The main processing loop should:

- Obtain odometry information
- Obtain list of currently seen markers and their poses
- Update the particle filter using the above information
- Update the particle filter GUI (for debugging)
- Determine the robot's actions based on the current state of the localization system.
    - The robot could actively look around if the localization has not converged (i.e. global

localization problem).
    - The robot should drive to the goal if localization has converged (i.e. position tracking problem).
- Have the robot drive to the goal. Note that the goal is defined in terms of both position and orientation. We have specified the coordinates of the goal in the `goal` variable in `go_to_goal.py`. Please use this variable. Once there, have the robot play a happy animation, then stand still. Your cozmo should have reached the goal within **90** seconds.
- Make your code robust to the "kidnapped robot problem" by resetting your localization if the robot is picked up. This should be triggered both when the robot is on its way to the goal and once it has reached it (i.e. picking up the robot from the goal and placing it somewhere else should immediately cause it to try to localize itself and search for the goal again).

You are encouraged to reuse as much of your existing code as possible for this lab. The coordinate frame we are using is:



Note: this picture is just used for explaining coordinate frames. Please setup marker positions as in '**Setup**' section.

Grading: You will demo your code for grading during class on the day the assignment is due. The assignment will be evaluated for 1) obtaining the correct robot position estimate (location), 2) enabling the robot to successfully drive to the goal.

Due to the probabilistic nature of the particle filter, not every run will lead to success. To test the robustness of your code we will conduct three runs and drop the lowest performing run. In a single *execution run*, the robot will be placed at start location from which one or more markers are visible

and allowed to explore until it reaches the goal or until **90** seconds have elapsed. At a fixed period of time into the run (e.g. after 20 seconds), your robot will be "kidnapped" and placed in a new location.

If at any point your robot is stuck in a hopeless situation, you may request a "kidnapping," in which case your grader will relocate the robot to a central location in the arena with good view of the markers for a 5-point penalty.

Note that we will not purposefully create problematic situations, such as starting the robot facing a corner.

We are also aware that the marker detection code is sensitive to lighting. You may provide additional lighting, such as with a flashlight, if you feel it is necessary.

Our grading rubric will look like this:

Group #	Run	PF correctly converged	Reached goal	Reset on kidnap	Kidnap request penalty	Extra credit (1-time)	Run Subtotal	Total (sum best 2 of 3)
	1	/30	/15	/5	/-5	/5	/50	/100
	2	/30	/15	/5	/-5	/5	/50	
	3	/30	/15	/5	/-5	/5	/50	

Grading notes:

- Particle Filter (PF) correctly converged: Full credit if the PF estimate at any point matches the real robot pose. Position should be within 3 inches and angle should be within 15 degrees of real robot pose. It's fine if the PF does not stay converged, full credit will be awarded if it converges to the right pose at any time.
- Reached goal: 10 points if the robot center is within 3 inches of the goal. 5 points if the angle of the robot is within 15 degrees of the specified angle.
- Reset on kidnap: full credit if the PF resets to a uniform distribution if the robot is picked up.
- Kidnap request penalty: we record number of times the team requests additional kidnappings. Note, there is always one kidnapping that happens during each run, that one does not count as a penalty.
- Extra credit: adds 5 points to the overall lab grade (just once, not per run) if the robot acts unhappy when we pick it up for kidnapping. Any degree of unhappiness will do.

Submission: Submit only your `go_to_goal.py` file, make sure you enter the names of both partners in a comment at the top of the file. Make sure the file remains compatible with the autograder. Only one partner should upload the file to T-Square. If you relied significantly on any external resources to complete the lab, please reference these in the submission comments.