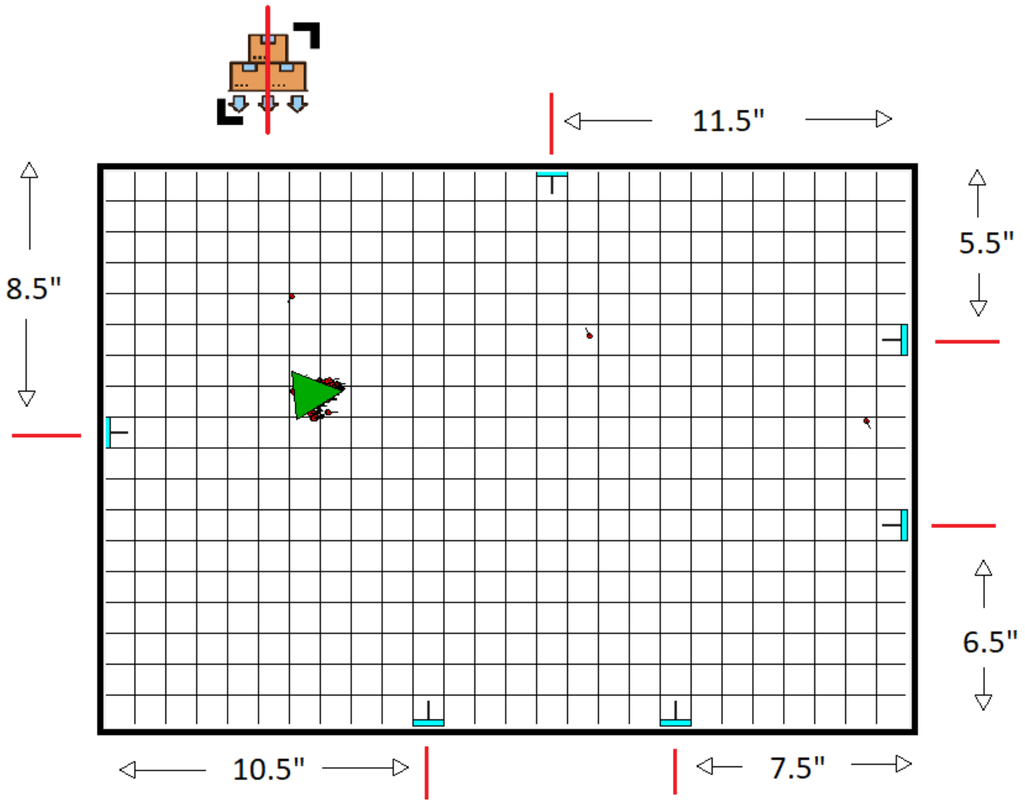
# LAB 4: PARTICLE FILTER PART II

Due: October 22nd, 11:59pm

The 4th Annual CS 3630 Skating Competition

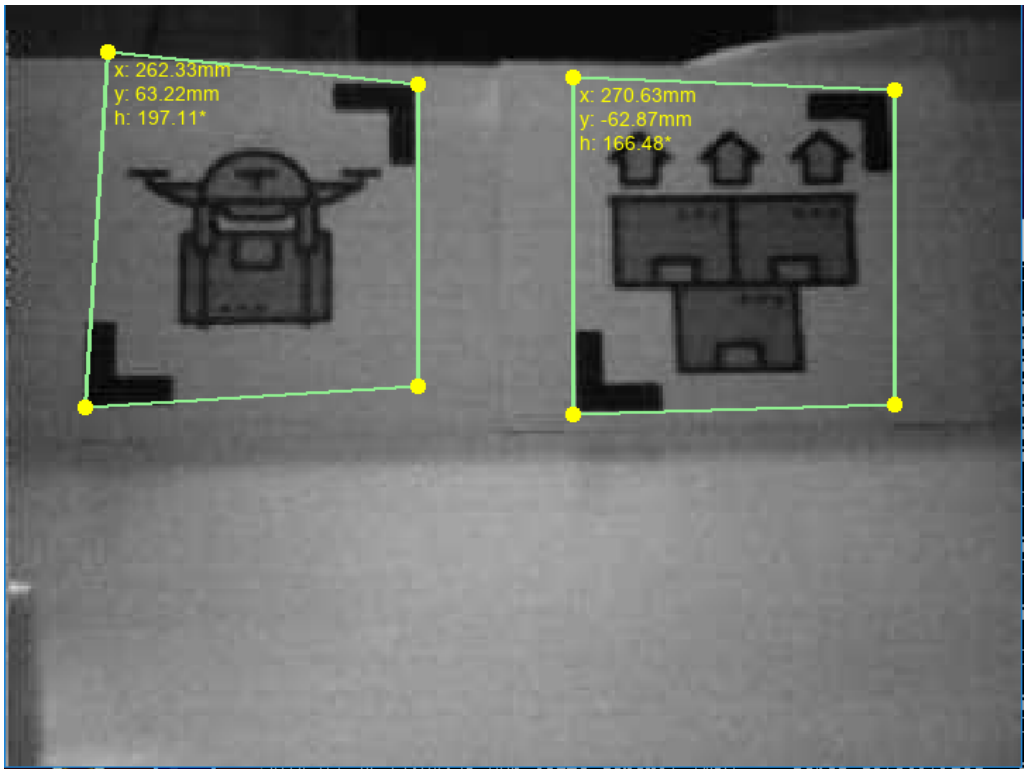
**Task:** *Welcome to the 4th annual CS3630 Skating Competition!* You're a world-renown coach who has successfully trained thousands of skater robots, and you have recently entered your top competitor, Skater-Bot, in the CS3630 Skating Competition. In Lab 3, you have implemented a particle filter, which enables you to monitor the progress of Skater-Bot. In Lab 4, you will first need to set up the arena with the localization markers. Then, you adapt your Lab 3 solution to enable Skater-Bot to skate within its arena and reach a target marker. In order to reach this goal, you must implement three functions: `marker_processing()`, `compute_odometry()`, and `run()`. Each function implementation detail can be found in the section below.

**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. We have attached a PDF file (CS3630_Symbols.pdf) in case you need to print the symbol cards. Attach the cards to your arena as shown in the figure below:



**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 or `python3 test_marker_detection_vector.py` with the Vector. Place the Cozmo/Vector 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

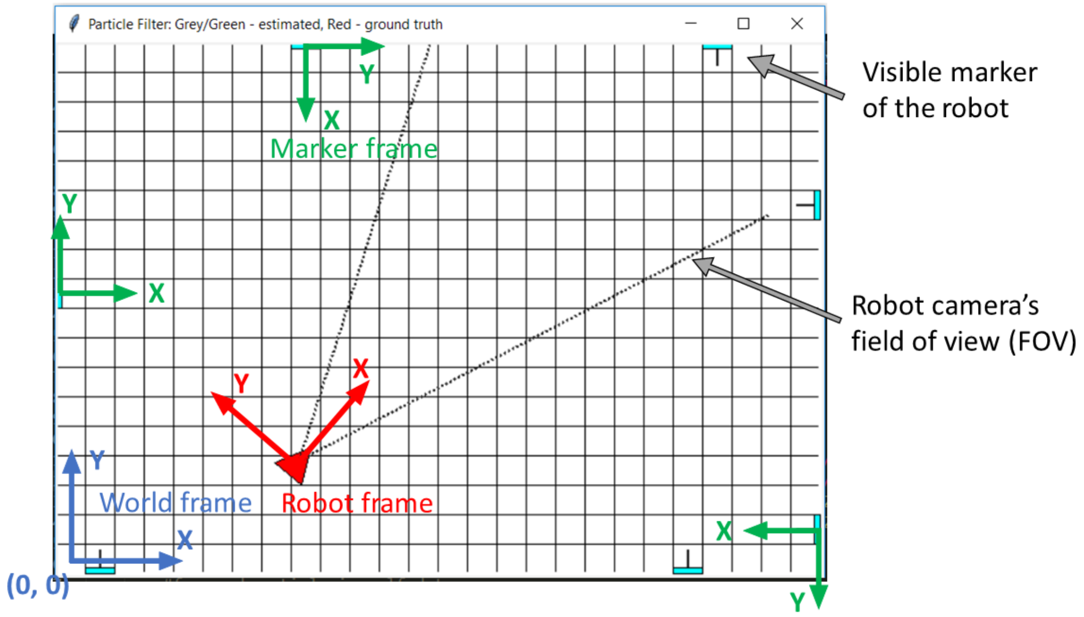
`run()`: the main processing loop, enter your code here

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. For example, you may want the robot to actively look around if the localization has not converged (i.e. global localization problem), and drive to the goal if localization has converged (i.e. position tracking problem).

- Have the robot drive to the goal, which is given as (6″, 10″, 0). Note that the goal is defined in terms of both position (x,y) = (6 inches, 10 inches) and orientation h = 0 degrees. Once there, have the robot play a happy animation and then stand still.
- Make your code robust to the "kidnapped robot problem" by resetting your localization if the robot is picked up. This should trigger 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:


Particle Filter: Grey/Green - estimated, Red - ground truth
Y
X
Marker frame
Visible marker of the robot
Y
X
Robot camera's field of view (FOV)
Y
X
Y
World frame
Robot frame
X
(0, 0)
X
Y

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

**Grading:** The assignment will be assessed based on the ***Code*** portion and the ***Video*** portion. The ***Code*** portion will be evaluated on passing all the visible and hidden test cases in Gradescope. The ***Video*** portion will be evaluated for 1) obtaining the correct robot position and orientation estimate, 2) enabling the robot to successfully drive to the goal. The ***Code*** will count as 30% of your grade, and the ***Video*** portion will count as 70% of your grade.

**Details about the *Video* portion:** You will upload a video of a single *execution run* of the robot. 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 elapse.

**Note and Suggestions:**

1. Do not create problematic situations, such as starting the robot facing a corner.
2. 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 for the ***Video*** portion will look like this:

| **Run** | **PF correctly converged** | **Reached goal** | **Reset on kidnap** | **Total** |
|---|---|---|---|---|
| **1** | /35 | /25 | /10 | /70 |

*Grading notes:*

- PF correctly converged: Full credit if the PF estimate at any point matches the real robot pose. Position should be within 3" 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. **Within the video, you need to show the moment that PF correctly converged on your computer screen as the robot is moving within the arena.**

- Reached goal: Full credit if the robot center is within 3" 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. **Within the video, you need to show your computer screen that PF distribution is uniform as you pick up the robot after it reaches a marker.**

**Submission:** Submit your `go_to_goal_cozmo/vector.py` file and `video` 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 Gradescope. If you relied significantly on any external resources to complete the lab, please reference these in the submission comments.