Flight camp assignment

The time has now come for you to show that you understood something from the tutorials Part 1-3 and can use your knowledge in practice and that you are ready to take the leap to working with the real system.

Your task is to to make a smaller version of what you will do in the project. You will work only in simulation and making assumptions that are likely not to hold in the real world. Your task is to design and implement three nodes that can

1. Take the ArUco marker detections, transform them into the map frame in a node and publish a tf from map to /aruco/detectedX where X corresponds to the id of the marker.
   - The camera is mounted 0.01m forward and 0.02m up with respect to the Crazyflie.
   - It is enough if you can do this for one marker (ie so that you can hard code X) but cooler if you can do it for all X.
   - You should also be able to display this at the same time as the known pose of the markers (that you looked at in **Part 2 (https://kth.instructure.com/courses/17743/pages/flight-camp-part-2)**) in Rviz.
   - You should be able to comment on any differences between these frames.
2. Make the drone move between a set of at least three poses defined in the map frame. These poses could, e.g.,
   - be hard coded poses, maybe such that the drone actually flies through the gates?
   - be the marker poses (read from the json file like was shown in the tutorial). In this case you might need to edit the map file to remove some of the gate walls to avoid collisions.
3. Detect at least one object in the environment (other than marker).
   - The minimum requirement is that you can generate a message of some sort when the object is in the image.
   - Even cooler is if you can also tell where it is in the image.
   - Super cool you can tell roughly where in the world it is, based on the image data.

Guidelines

- What you did in the tutorial (part 1-3) is part of the assignment so you should be able to show that you undertand and can use this as well
- You should write your own code and be able to explain and motivate all parts.
  - You can discuss solutions with others but not exchange code.
- You should also be able to discuss what parts of the design/implementation would work in the real world and not and why / why not.
- You should be able to run your code starting from an empty terminal. Having a checklist for this is OK but we want to see that you actually can run the system yourself.