CS-475 Assignment 3
Particle Filter

Michael Maravgakis
maravgakis@csd.uoc.gr

Release date: 24/03/2022
Deadline: 04/04/2022

1 Overview

In this assignment you will learn how to localize your robot by using the Particle Filter algorithm. Particle filter can work even if the noise is not gaussian (unlike KF and EKF) but it is more computationally demanding. In the previous assignments you used a GPS module and odometry to estimate the state of the robot with respect to the world frame. Now, you are provided with a custom made room and a drone that can navigate inside by using the teleop node. Instead of GPS you will use the provided height map of the room and displacement measurements of the drone.

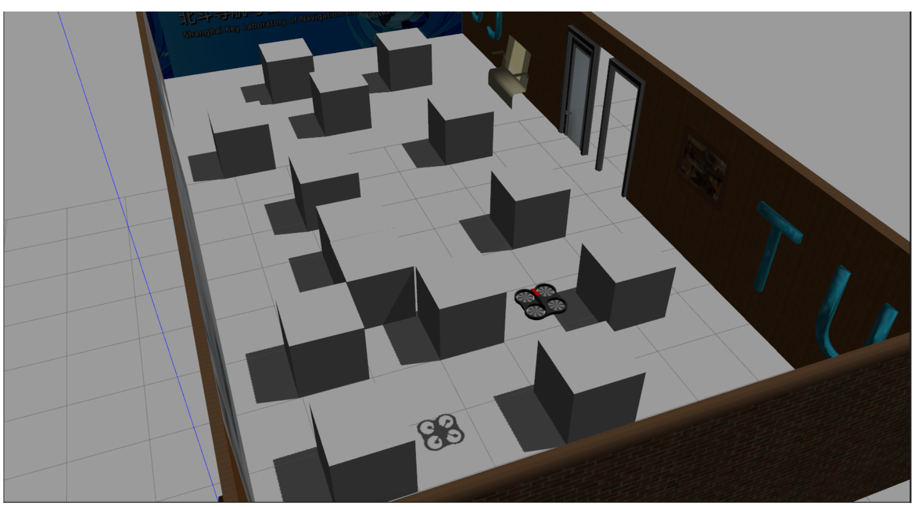

Figure 1: Custom workspace + drone

2 Installation

In your .zip file, you will find the `assign3/` package Copy and paste it inside a catkin workspace and then download the following package and compile:
`$ git clone https://github.com/tahsinkose/sjtu-drone.git`
Use `$ rospack profile` to update ros file system in order to see the new packages. You also might need to `$ chmod +x particles.py` in order to make your node executable. To check if everything is fine, run:

```
$ roslaunch assign3 drone.launch
```

Gazebo and Rviz windows will open (give it some time for the first run) and you should see the drone inside the custom room from figure 1 (drone spawns at the corner). You will probably notice a bunch of errors at the terminal (missing model.config), just ignore them if the world and drone spawn correctly. For this assignment you will need both Rviz and Gazebo to work simultaneously. Gazebo will be used for navigation and Rviz for visualization of your results. Finally to open the teleop and move the drone around the map run:

```
$ rosrun sjtu_drone drone_keyboard
```

The window from figure 2 will pop up. Navigate around after you take off with Z.

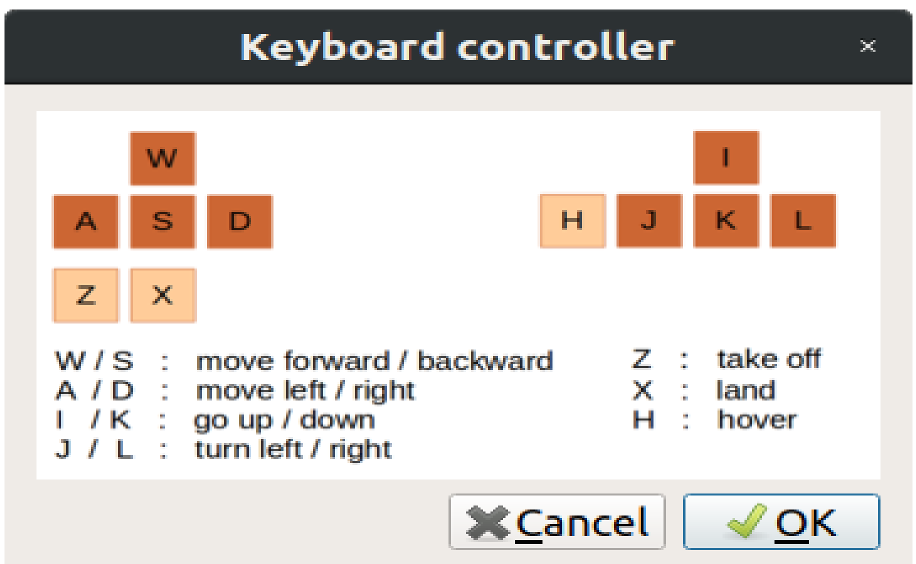

Figure 2: Teleop controls

Now that everything is set, you can move on to your implementation.

3 Information and measurements

You will have to fill the code inside `particles.py` in order to implement the particle filter. The goal is your particles to estimate the 2D position (x,y) of your drone inside the room. The message that will be used in order to create, update and visualize your particles is `geometry_msgs/PoseArray`, keep z=1 and [qx,qy,qz,qw] = [0,0,0,1], only update x and y. Also make sure that your PoseArray message has always the attribute `msg.header.frame_id = 'map'`. The size of the room is (8m,15m). The measurements that are provided and can be utilized to create your algorithm are:

Height map

The `height_map` of the room contains the height of a specific point inside the room. So, `height_map[x_i,y_i] = Height_of_that_point`. Be careful, `x_i`, `y_i` are indices inside the grid that represent the positions of x and y in meters. The resolution of the grid is something you will find useful in order to convert meters to grid cells. The resolution is 0.01, e.g. the point (5,3) in meters is (500,300) inside the height map. In the following image I've made a visualization of the height map of the room to make things more clear. The bottom right

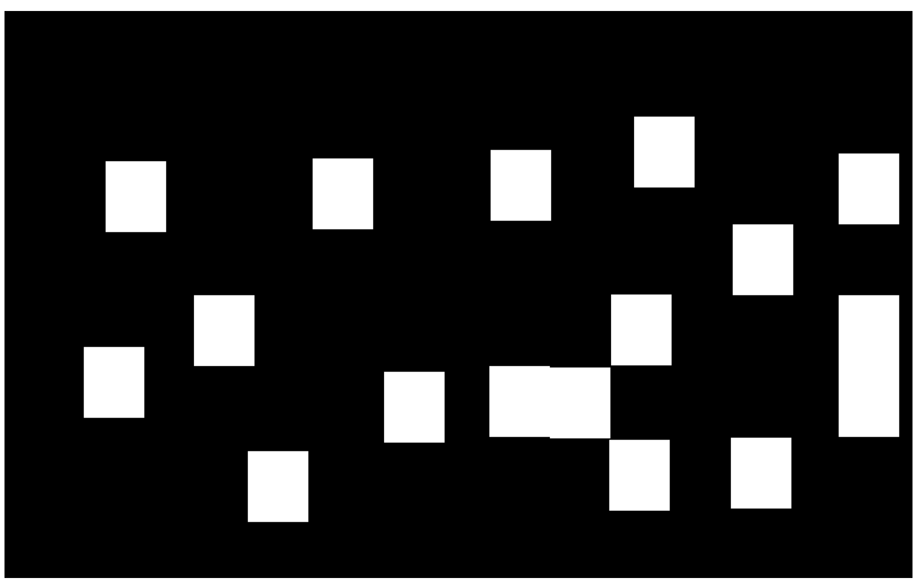

Figure 3: Height map

corner can be considered as (0,0). X-axis runs from bottom to top and Y-axis from left to right. The white pixels represent the distance from ground not equal to zero. Although the room contains only cubical boxes, note that the problem is not binary (height either one or zero), I've encoded the height from the center of the box to the edges to decrease. This information will not affect you, it was done in order to avoid binary weights. You can imagine each box to be a mountain with its center being the top.

Displacement

You are provided with 2 displacement measurements namely `dx,dy`. These represent the difference between the previous and the current position of the drone at x and y axis respectively.

Range finder

The `range_finder` variable contains the measurement of a radar that can estimate (with noise) the vertical distance below the drone. Lets say that the drone flies above a box at height 6m from the ground. The height of the box is 1m, so the range finder will give you a measurement around 5m.

Actual height

The `actual_height` variable contains information regarding the vertical distance between the drone and the ground in meters.

4 Implementation

This time you will have to code most of the algorithm yourselves. You can modify the template all you want, create functions declare new variables etc. The steps that you need to take are the following:

1. Initialize particles: Firstly, define your message type that will represent the particles. Then, randomly distribute all the particles inside the room. (tip: You can use random.uniform(low_lim,high_lim)

2. Initialize weights

3. (Loop starts)

4. Sample particles: From the already existing particles distribution, pick randomly N of them. (tip: displace them a bit e.g. -0.1,+0.1 from their current position)

5. Update particles: Use your displacement measurements to update the position of all your particles. In case that one of new particle's position gets out of the bounds of the room you can assign to it random values.

6. Calculate weights: You'll need to find a metric that evaluates and assigns a weight to all particles given the current measurements and the height map.
7. Resample particles: You can use the low variance resampling algorithm to eliminate some of the lower weight particles. (Probalistic Robotics book)

You will need to fill the gaps ***, but not with single liners. They are there as a guide on what to implement at this point. Try to define functions and keep your code clean and commended.

5 Results

If you've implemented the algorithm correctly you will need to navigate to the room for a while in order for the algorithm to converge (not too long). In the following figure you can see that although the algorithm has not converged yet, is on the right way since it has eliminated all possible positions except the top of the boxes. This is the initial output of the algorithm if the drone is sitting on top of a box when you run the algorithm (after a few iterations):

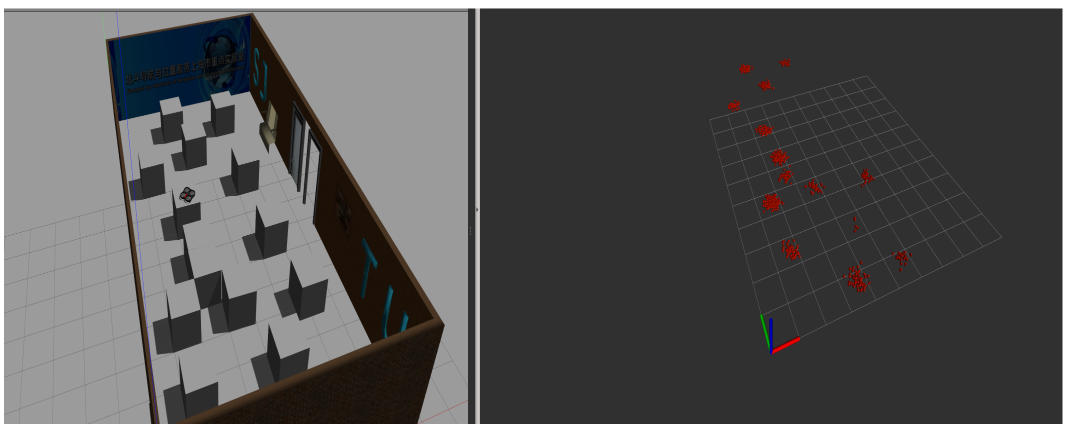

Figure 4: Initial output

If you move the robot around, the particles that can not fit will be eliminated sooner or later and the expected result during navigation will be:

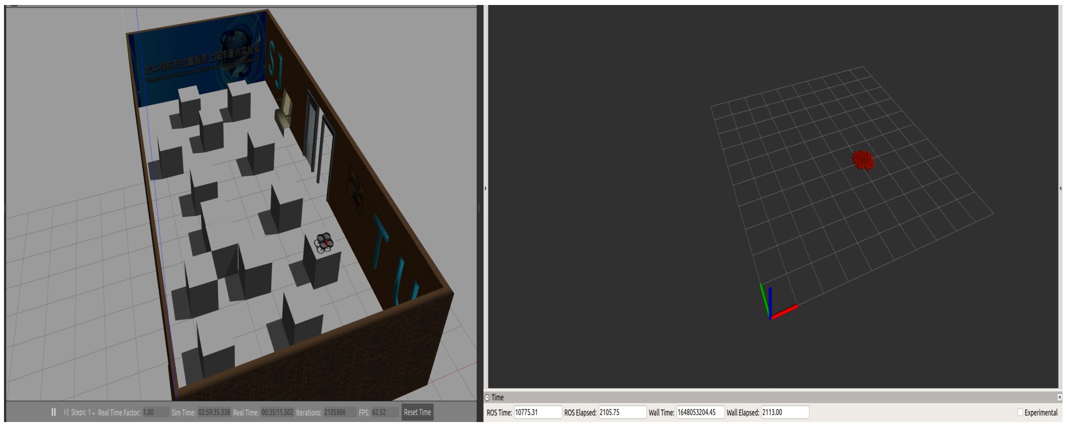

Figure 5: Converged particle filter

6 Submission

Sent your node (`particles.py`) attached via email at: **maravgakis@csd.uoc.gr** with subject `"[CS-475] Assignment 3 submission"` Don't forget to mention your name and registration number. The deadline is at **04/04/2022 23:59**