E7 LAMP Experiment Manual

Florence Tsang Date: Dec 20, 2019

Could be done with 2 people, but 3 is ideal. One for videography, another 2 for controlling the robot and troubleshooting.

1 Purpose of Redo

- Get better video (following the robot)
- Get better data with a more stable program, there shouldn't be as many localization errors

2 Materials

The Robohub should contain almost everything listed. Using a decently fast laptop for connecting to the Jackal would be ideal. The E7 space will need to be booked beforehand before it can be used.
For building the maze:

- cloth (they're precut and should create the maze specified)
- twine
- cloth scissors (if changes need to be made)
- clothespins
- stanchions
- 15+ chairs
- cardboard boxes
- measuring tape

Other stuff:

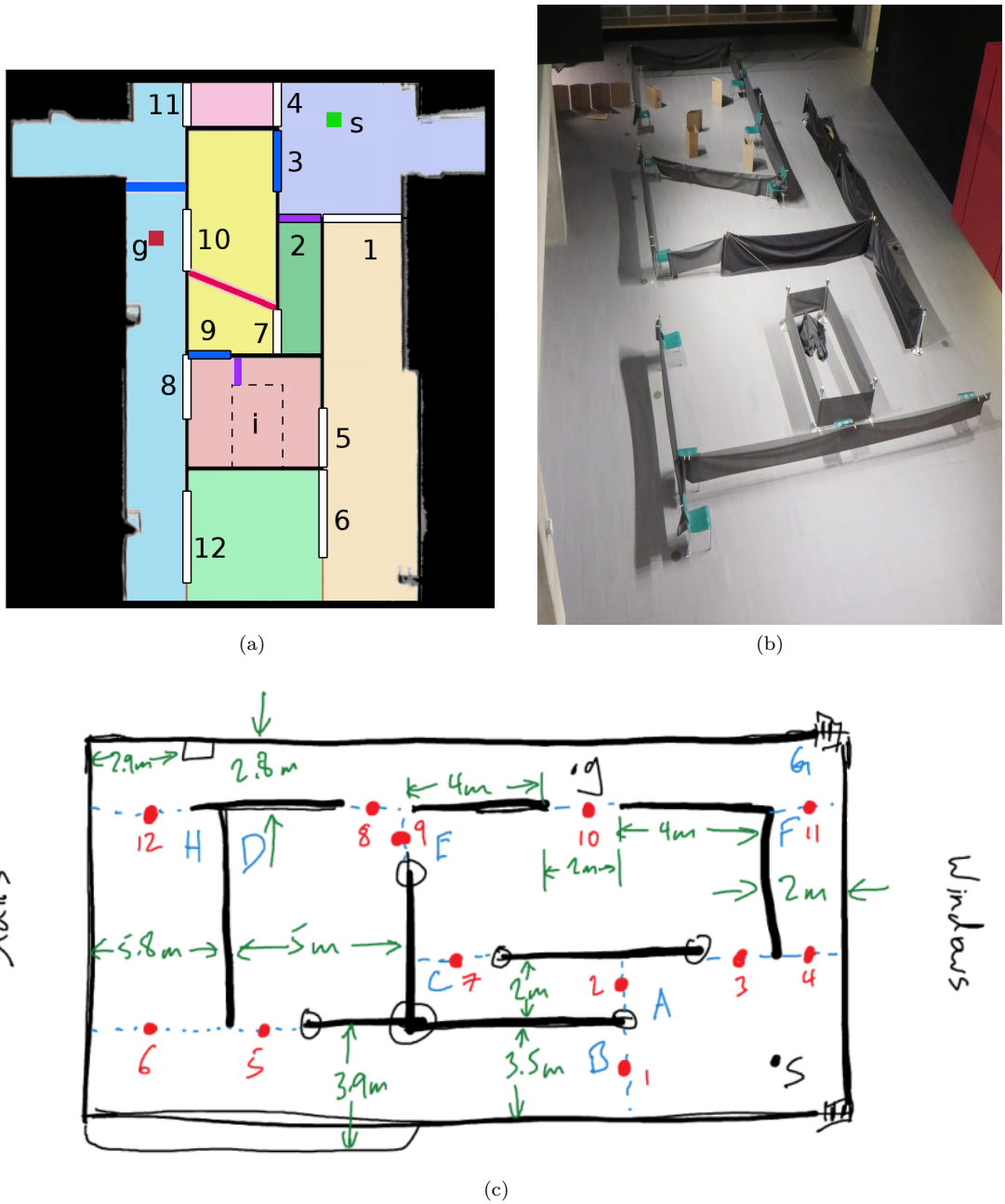
- camera + tripod, smartphone works too
- Jackal
  Charge it the night before testing.
- Fully charged spare battery for Jackal
- Charger for Jackal
- PS4 controller (charge it before testing, they aren't being charged in storage)
- Power bar
- micro USB to USB charging cable + wall adaptor would be useful for recharging the controller/smartphone/camera

Rental information:

- Kari Griffith (kgriffiths@uwaterloo.ca) - space rental
- Bookings Coordinator Donna Schell (dschell@uwaterloo.ca) - stanchion/chair rental. Only 6 stanchions can be rented.
- There's a little glass room on the 2nd floor of E5 that contains 6 stanchions, they're not as heavy as the rented ones though. The room should be unlocked.

I set my laptop as a WiFi hotspot and connected the Jackal to my WiFi to control it. Maybe they'll have a WiFi access point available for you instead.

3 Setup



(a) (b)

(c)

Figure 1: The portals should be $\geq$ 3m to minimize incorrect edge state.

There was also a long strip of cloth to cut off the LiDAR before the stairs at the bottom of the light blue, green, and orange submaps. I found this reduced localization drift. Another suggestion is to find something to create a 'wall' on the windows side. The sitting area there is around the height where the Jackal's LiDAR registers an obstacle, and I think that is the cause of some of the drift in the localization. Alternatively you could lower the height at which the sensor registers an obstacle in the cost map parameters. Make sure the cloth walls are still being registered as obstacles.

All the obstacles were a strip of cloth that could be removed from the portals, with the exception of:

- i - originally used 4 stanchions with cloth wrapped around it, might be easier with a large table
- dark blue obstacle that blocks edge (11,g), I used cardboard boxes

4 Execution

1. Check that there's an 'e7' folder in the 'maps' folder of the ROS package. The following should also be in the folder:
   - base.pgm
   - base.yaml
   - e7_tgraph.yaml
   - e7_polygons.csv
     Extra files are ok.
2. Set the robot as close to the starting position as possible. A little error is ok.
3. In 'nodes\planner_node', check that:
   - The MAP constant is set to 'e7'.
   - The option **sim=False** when calling **Lamp()**.
4. In 'nodes\edge_observer_node', check that the MAP constant is set to 'e7'.
5. In finish_task() of the high level planner, select the policies that will be executed for each task of the trial. I only did naive and offline policy. Keep in mind the robot must go from start to goal for each policy execution. So if using the above two policies for 10 tasks, the robot must be moved back 20 times. Here's an explanation of each policy:
   - online - selects the next best action during task execution, does not pre-calculate the policy
   - offline - fully calculates an RPP policy prior to task execution
   - openloop - uses optimistic policy on hybrid map
   - naive - only use navigation stack, doesn't use LAMP framework

   The policies can be selected by setting the **next_mode** variable in each **if(policy==...)** block.
   If you are not using the online policy, in **start_task()** set the obstacles to be reset when **mode**==(desired start policy).
   If you are not using the naive policy, in **finish_task()**, set the task counter to be incremented when **mode**==(desired end policy).
6. Prepare screen recorder to record rviz for analysis later. I used Kazam.
7. Run **roslaunch lamp experiment.launch**. This launches the high level planner and move_base.
8. Run **roslaunch lamp observer.launch** in a separate terminal. This launches the edge observer.
9. In the terminal where you launched experiment.launch, follow the instructions until the trial is over (i.e. all the tasks have been executed).
   - Jackal must be moved back to the start after every policy execution.
   - To select obstacles at the beginning of every NEW task, I used a random number generator to randomly select a number between 1-10 for each obstacle set (table 1) before manually setting up the selected obstacles.
10. Rename the ../results/lrpp-results.dat file after every trial. Otherwise it won't get overwritten, but the results of the next trial will be appended to the end of the previous trial.

Note: The high level planning node stops the task execution if the robot moves more than 5m in a single time step (aka AMCL returns a pose that's significantly different due to some error). Robot will have to be moved back to the start position before restarting the task with the current policy.

Obstacles	Probability
dark blue	0.6
purple	0.3
pink	0.9
i	1.0

Table 1: Probabilities of obstacles appearing in the environment.

5 More Pictures to Help with Maze Setup