Progress Review 8

Individual lab report – 07 | February 10, 2016

TEAM DAEDALUS

Submitted by:

Mohak Bhardwaj

Team Members:
Dorothy Kirlew
Richa Varma
Shivam Gautam

Pranav Maheshwari

•

1. Individual Progress

The following responsibilities were taken up by me:

- Implementing dynamic ranking of spots based on a policy
- Implementing a ROS node to publish most optimal spot calculated from occupancy data relayed by the communication system.

DYNAMIC RANKING OF SPOTS

One of the most important aspects of the project is to be able to test different policies for scheduling of cars in the parking lot. Many different metrics are possible for deciding the most optimal spot for the car that enters the parking lot. My responsibility as the task owner for multi-agent planning was to implement a system for dynamically allocating optimal spots to different car.

As specified in ILR06, the metric being used for calculating the most optimal parking spot is the minimum number of steps that it would take the car the move from the parking spot to the exit taking obstacles and other parking spots into account. One step is defined as moving from one grid point to the next in the layout of the parking lot (see Figure 1).

In order to implement such a routine, a dynamic programming based approach was taken. Python was used to implement the script. Starting recursively from the exit grid cell, the penalty function or value is calculated for every grid cell recursively. The penalties for cells that are obstacles(containing infrastructure or other known obstacles) are set to a very high value. The penalty of a cell, basically tells us how bad it is for someone to be in that state inside the parking lot with respect to the current policy being implemented i.e. number of steps to the exit.

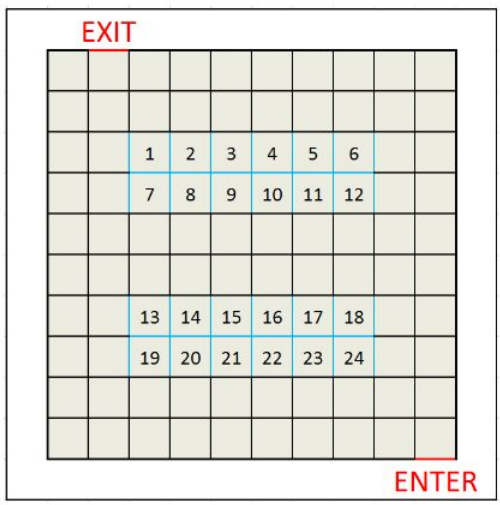

Figure 1: Grid layout of parking lot design indicating parking spot with numbers (Collaborator: Dorothy)

The recursive algorithm works as follows:

1. The motion primitives of the robot are set as : [up, down, left, right] This is so because our robot has a differential drive mechanism allowing it to turn around its axis and move to any of the adjoining cells. Diagonal motion is not being considered as of now.
2. An open list of states is maintained containing the current states under consideration. A closed list of states contains information about the cells that have already been visited and obstacles. A list of cells that are parking spots is also maintained.
3. Initially the open list contains only the exit spot and is given a value of 1. Based on the motion primitives, the children cells of the exit cell are opened and assigned a value according to the formulation:

$$V(s) = V(parent) + cost$$

4. The cost of moving from one cell to another is kept as 1 unit. Also, once the children of a cell are added to the open list, that cell is popped from the open list and added to closed in order to avoid regressive steps.
5. Another constraint that has been added to the formulation is with regards to adjacent parking spots. Initially, the separation between the spots were considered as an obstacle cell, but this led to penalizing spots along the breadth which are actually at a better position than some spots along the height of the parking lot. In order to remove the obstacle cells, I had to add an extra constraint that parking spots will not have adjacent parking spots as their children. This is important in order to keep the scenario realistic because moving from one parking spot to next is not a valid step.

The output of the program is a 2D array with the penalty function of each grid cell in the corresponding indexed location. Figure 2 shows the value of the grid cells calculated from the algorithm. As can be seen from the output, the value for obstacle cells is 99, extremely high s compared to the other cells. Also, on inspection it can be seen that the value for every state is consistent with what we would logically attribute to that state.

Using the value of each state and the indices of the parking spots, the value for every parking spot can be stored and retrieved as a list.

Although the program is currently being used to pre-compute the value of each parking-spot, it can if need be utilized for calculating parking spot values in real-time for more complicated policies.

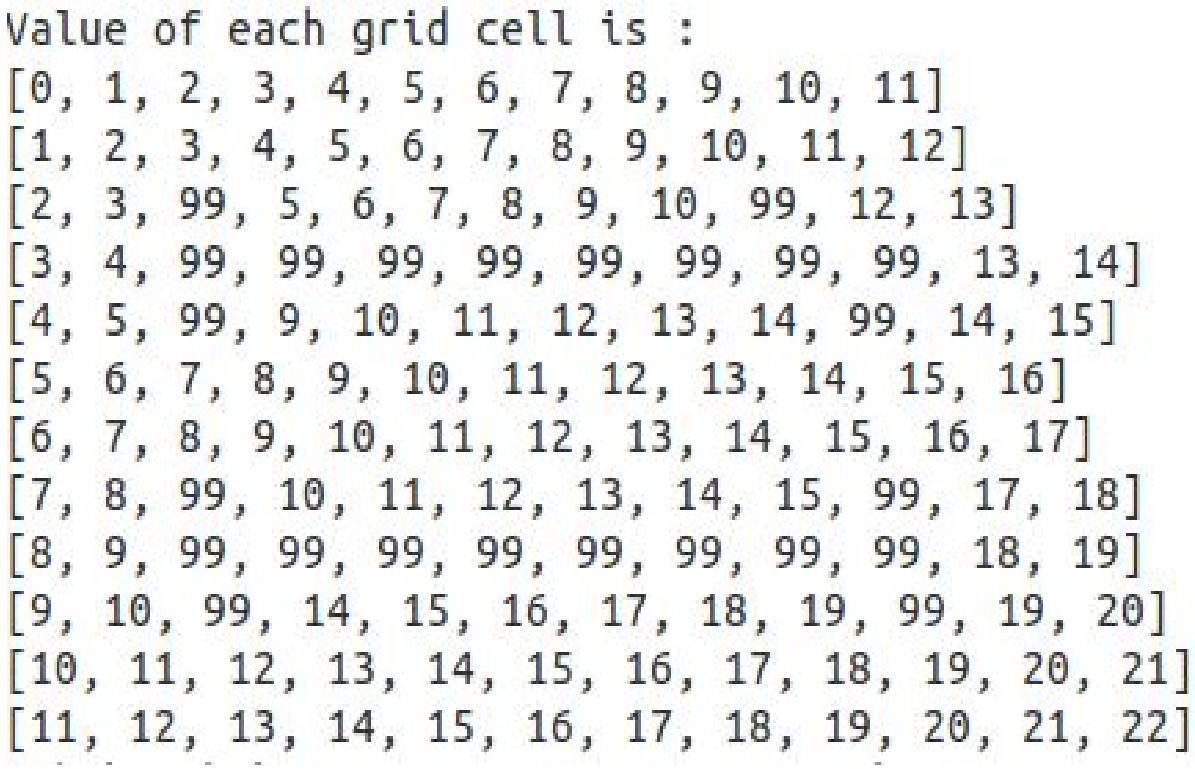

Figure 2: Value of each grid cell corresponding to the cells in the parking lot layout.

ROS IMPLEMENTATION

The above program was used to compute the values of different parking spots which were then stored in the form of a look-up list. This was used as the pre-fed knowledge for the ROS node responsible for spitting out the most optimal spot based on input from the communication subsystem. The algorithm for the node is as follows:

1. Read input from communication system in the form of a list of 24 bits (0 = EMPTY 1 + FULL) from a ROS Topic (currently toy input)
2. The indices of the list correspond to fixed parking spots. The list is used to check whether a parking spot is empty or full.
3. The parking spots are stored as a heap so that it easy to retrieve the most optimal spot based on value.
4. The most optimal and empty spot is published on a ROS Topic where it can be read by the locomotion subsystem to act as a waypoint.

Figure 3 shows the output of the node for a given summy input. It can be compared easily with the values grd and layout of parking lot to verify that the spot returned is infact the most optimal spot according to closest to exit policy.

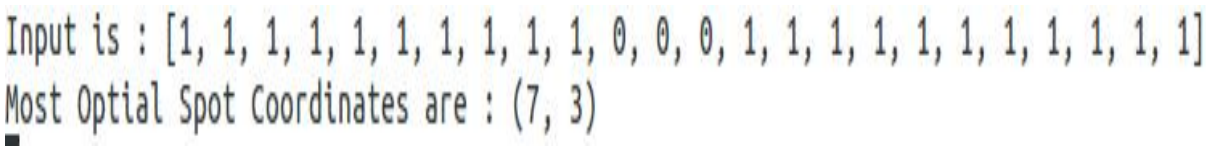

Figure 3: Output of optimal spot node based on shown input list

2. Challenges

The challenges that I had to face were:

1. Hardware issues with the mobile platform involving electronics and depth camera that delayed my work on mapping beyond its schedule.
2. The optimal spot algorithm is one of the most critical aspects of the project and had to be made applicable in different scenarios.
3. Delay in arrival of the second platform which made keeping up with the schedule extremely difficult.

3. Teamwork

The other members of my team were working on the following subsystems:

1. Shivam: He worked on developing the emergency node using arduino and interfacing it with the Parallax Laser Range Finder in addition to calibration and testing. He also worked on CAD and 3D printing of the servo mounts.
2. Pranav: He worked on developing waypoint following code using ROS Navigation stack. He also worked on developing the visualization tool for emulating different scenarios in the parking lot.
3. Dorothy: She worked along with Richa on the developing the protocol for the communication subsystem and implementing it using 3 different XBees. This was followed by extensive testing.
4. Richa: She worked along with Dorothy on completing the communication subsystem using multiple XBees and testing.

4. Plans

Now that the second mobile platform has arrived the first major task for me is to create a map of the surroundings and form guidelines for developing the parking lot. This would also enable Pranav to continue his work with the locomotion subsystem and test emergency stop along with Shivam. Shivam would work towards completing the perception subsystem and integrating it with the mobile platform. The communication subsystem is complete and will now be integrating with different subsystems. I will be working with Richa and Dorothy on integrating the multi-agent planner with the communication subsystem. Pranav will work

closely with them in order to integrate the visualization tool with the communication and with me to integrate the multi-agent planner and locomotion.