PROGRESS REVIEW 12: ILR11

Richa Varma
Team I
April 13, 2016
Team members: Dorothy Kirlew, Mohak Bhardwaj, Pranav Maheshwari, Shivam Gautam

1. Individual Progress:

For the progress review on April 11, Dorothy and I worked together to achieve our stretch goal of implementing a global multi-agent planner for multiple simultaneously moving vehicles in the lot. I also completed the task of getting our second platform fully functional.

Multi-agent planner : Modified A*

Until now, due to scope limitations, we assumed that there is only one moving vehicle in the lot at a time. Thus, selecting an optimal parking spot was purely based on the ranking of spots on their distance from the exit.
Moving towards scenarios where multiple moving vehicles are present in the lot at a given time, required a means to convey this information to other vehicles through our communication system. It also required a multi-agent planner that uses this information to modify goal selection and path planning. To this end, modifications were made to the XBee messages to communicate vehicle motion state as well as waypoints in case of a moving vehicle. A new planner was implemented using modified A* algorithm to select a parking spot and the best route to reach it in case of these dynamic scenarios. Dorothy and I spent some time to systematically plan the modifications that needed to be made at the communication end and at the planner end.

1. Xbee messages were modified to include a motion field. The vehicle dictionary now includes a tuple for each vehicle ID, i.e (spot id, motion).
2. The dictionary also contains a list of waypoints (obtained from A* planner).
3. If a return command is received, the planner uses the spot id ‘25’ as the goal (ID of the exit) and calls the A* planner to give waypoints to the exit using the normal heuristic cost to the exit without taking moving vehicles into account.
4. If a park command is received, the planner takes in the top 10 available spots and runs A* on them to find the minimum cost path based on a cost map that includes inflated costs for the paths of moving vehicles in the lot. The spot (goal) that has the minimum cost associated with it is chosen as the optimal spot by that vehicle. The planner also returns a list of waypoints for the vehicle to navigate to this chosen spot.

1. A* implementation

Dorothy and I worked on implementing the A* based multi-agent planner.
The gcost is traditionally the cost of reaching a particular node in a graph from the start position. We modified this definition to fit the context of our system. For our gcost map, we used a map of 1s as there are 4 possible directions of movement (forward, backward, left and right) and moving from one grid cell to the next incurs a cost of 1. We also introduced corner costs so that out of two equal length paths, the algorithm would chose a path that has less number of turns. This gcost map is dynamic in the sense that it accounts for moving vehicles in the lot. Based on the waypoints of the moving vehicle, the incoming vehicle inflates the cost of the entire path on its gcost map. This ensures that the new vehicle will chose a spot and navigate to it without any path crossing with the moving vehicle.

For the heuristic cost, we used a precomputed cost map based on distance from the exit. Top 10 spots are chosen based on their ranking based on this measure and A* is run for these 10 spots. The best spot is the one that has the minimum cost path associated with it (from the entry spot).

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2

Figure 1. gcost map with inflated costs to account for motion (Courtesy : Dorothy)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36

Figure 2. Precomputed hcost map (Courtesy: Dorothy)

2. Xbee message format modifications

```
{100: [1, 1], 101: [0, 0]}
updated waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
heard something from master:
parked
original dict
{100: [1, 1], 101: [0, 0]}
original waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
heard parked
Sending PARKED message b/c ROS parked status received
sending PARKED message from vehicle 100 to XBees
updated dict
{100: [1, 0], 101: [0, 0]}
updated waypoints
{}
received message
original dict
{100: [1, 0], 101: [0, 0]}
original waypoints
{}
I received UPDATE from  101
updated dict
{100: [1, 0], 101: [2, 1]}
updated waypoints
{101: [[18, 18], [18, 2], [6, 2], [6, 4]]}
```

```
updated dict
{100: [1, 0], 101: [0, 0]}
updated waypoints
{}
heard something from master:
park
original dict
{100: [1, 0], 101: [0, 0]}
original waypoints
{}
heard park
the spot_options:[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
]
waiting for planner response
Sending UPDATE because optimal spot was selected
sending UPDATE message from vehicle 101 to XBees
updated dict
{100: [1, 0], 101: [2, 1]}
updated waypoints
{101: [[18, 18], [18, 2], [6, 2], [6, 4]]}
```

Figure 2. Scenario: Vcl 1 parked in spot 1, does not send waypoints. Vcl 2 selects spot 2 and broadcasts waypoints

```
sending INTRO message from vehicle 100 to XBees
updated dict
{100: [1, 1]}
updated waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
received message
original dict
{100: [1, 1]}
original waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
I received UPDATE from  101
updated dict
{100: [1, 1], 101: [0, 0]}
updated waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
received message
original dict
{100: [1, 1], 101: [0, 0]}
original waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
I received UPDATE from  101
updated dict
{100: [1, 1], 101: [7, 1]}
updated waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]], 101: [[18, 18], [16, 18], [16, 8],
, 8], [4, 6]]}
```

```
updated dict
{100: [1, 1], 101: [0, 0]}
updated waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
heard something from master:
park
original dict
{100: [1, 1], 101: [0, 0]}
original waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]]}
heard park
the spot_options:[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
]
waiting for planner response
Sending UPDATE because optimal spot was selected
sending UPDATE message from vehicle 101 to XBees
updated dict
{100: [1, 1], 101: [7, 1]}
updated waypoints
{100: [[18, 18], [18, 2], [4, 2], [4, 4]], 101: [[18, 18], [16, 18], [16, 8], [4, 8], [4, 6]]}
```

Figure 3. Scenario 2: Vcl 1 parking in spot 1, broadcasts waypoints. Vcl 2 uses modified costs, choses spot 7, broadcasts waypoints

In order to include a motion flag and waypoint list to the Xbee messages as planned, Dorothy modified the code to incorporate these changes. A vehicle only broadcasts its waypoints when it is moving (parking or returning). Similarly, it deletes the waypoints of all parked vehicles from its dictionary. This scheme allows us to use the collaborative network to share waypoints.

2. Challenges

It took some time to come up with the A* formulation for our setup. It was important to choose the correct heuristic and gcost in order to find the optimal spot and path. We were finally able to come up with a solution that gave us desired results. No other major challenges were faced during implementation. On the platform end, I faced some power issues that were raising errors and inhibiting motion of the platform. I consulted documentation to sort out these issues and the platform is now in good working condition.

3. Teamwork

Pranav, Mohak, and Shivam worked on the simulation environment and integrated it to show 100 vehicles parking in the environment avoiding collisions. They also worked on navigation with the physical platform and repeated testing of the parking routine. Dorothy and I worked together in implementing the multi-agent planner for our stretch goal.

4. Future Plans

Pranav, Mohak, and Shivam will put finishing touches on the simulation environment and introduce statistics to show how our system is more efficient that normal parking without collaboration. Dorothy and I will integrate the UI with the new A* planner in order to simulate more complex environments. We will also be integrating the planner with navigation on the physical platform. We have received our third and final platform and will set that up to perform in the SVE.