Jusuk Lee
Department of Mechanical and Aerospace Engineering
Seoul National University, Seoul

April 11, 2024

HOMEWORK — 1

You can check the code below.
`https://github.com/JU-SUK/motion_planning-24spring`

1 Consider the 2-D environment. Find the shortest path A(x,y)=(3,15) to B(x,y)=(23,1).

가정:
Robot can move to neighboring 8 cells, and the cost is 1 for each horizontal or vertical movement, and 1.4 for diagonal movement.

Grid 기반의 경로를 찾기 위해 A^* 알고리즘을 활용했다. 아래 그림은 A^*을 이용하여 최단 거리를 찾았을 때의 결과이다. 시작 지점은 녹색, 도착 지점은 빨간색, 장애물은 검은색, 경로는 파란색을 이용하여 표시했다.

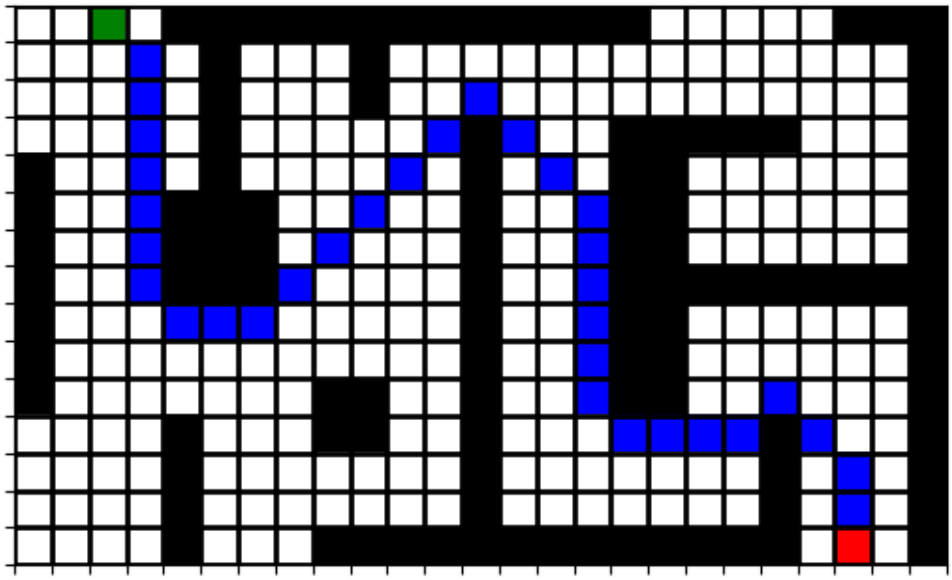

Figure 1: A^* 알고리즘 결과

2 Neglect the grid lines. Find the trajectory from A to B for the following system

$$\dot{x} = vcos\theta \tag{1-1}$$

$$\dot{y} = vsin\theta \tag{1-2}$$

$$\dot{\theta} = u_1 \tag{1-3}$$

$$\dot{v} = u_2 \tag{1-4}$$

(a) Use RRT and shortest path search

가정:
Dubins car model을 사용하기 위해 차량이 등속도로 움직인다는 추가적인 가정을 도입했다.

내용:
더 효율적인 경로를 찾기 위해서 near neighbor search와 rewiring이 추가된 RRT^*를 이용했다. 주어진 node와 가장 근접한 node의 index를 찾을 때, 기존의 유클리드 거리 상 가장 가까운 node를 찾는 방식 대신에 dubins car model을 고려하여 이동 거리가 가장 작은 node를 찾도록 했다.

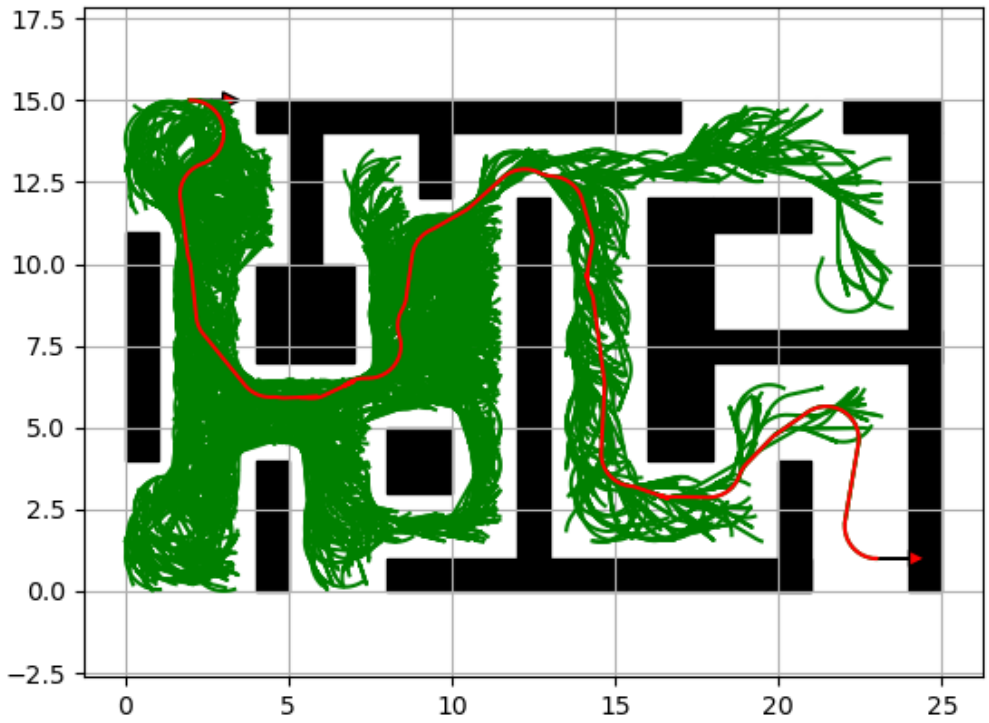

Figure 2: RRT^* 알고리즘 결과

(b) Use the potential function method. (You may show only partial result for this)

가정:
(a)번과 동일하게 dubins car model을 이용하기 위해 차량의 속도는 일정하다.

내용:
아래 이미지는 artificial potential field 알고리즘을 사용했을 때, local minima에 빠지는 것을 보여주는 결과이다. 배경은 potential을 의미한다. 차량이 목적지와 인력이 작용하고 장애물들과는 척력이 작용할 때, 두 개의 힘의 합력으로 차량의 움직임이 결정된다. 아래 이미지에서 볼 수 있듯이, 장애물들간의 거리가 좁은 bottleneck으로 지나가지 못하기에 최종 목적지에 도달하지 못했다.

Dubins car constraint를 고려한 dynamics는 아래와 같다.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta + \omega t) \\ \sin(\theta + \omega t) \\ \omega \end{bmatrix} \tag{1-5}$$

여기서 yaw의 최대각도를 40도로 제한한다. 따라서 시간에 따른 미분 방정식은 아래와 같다.

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} \frac{\sin(\theta_f + \omega\Delta t) - \sin(\theta_i)}{\omega} \\ \frac{\cos(\theta_i) - \cos(\theta_f + \omega\Delta t)}{\omega} \\ \omega\Delta t \end{bmatrix} \tag{1-6}$$

사전에 potential을 계산하지 않고, 로봇이 현재 상태에서 Δ t만큼의 시간 안에 도달할 수 있는 지점들의 potential을 계산하고 그 중에서 가장 낮은 것을 선택하도록 하였다.

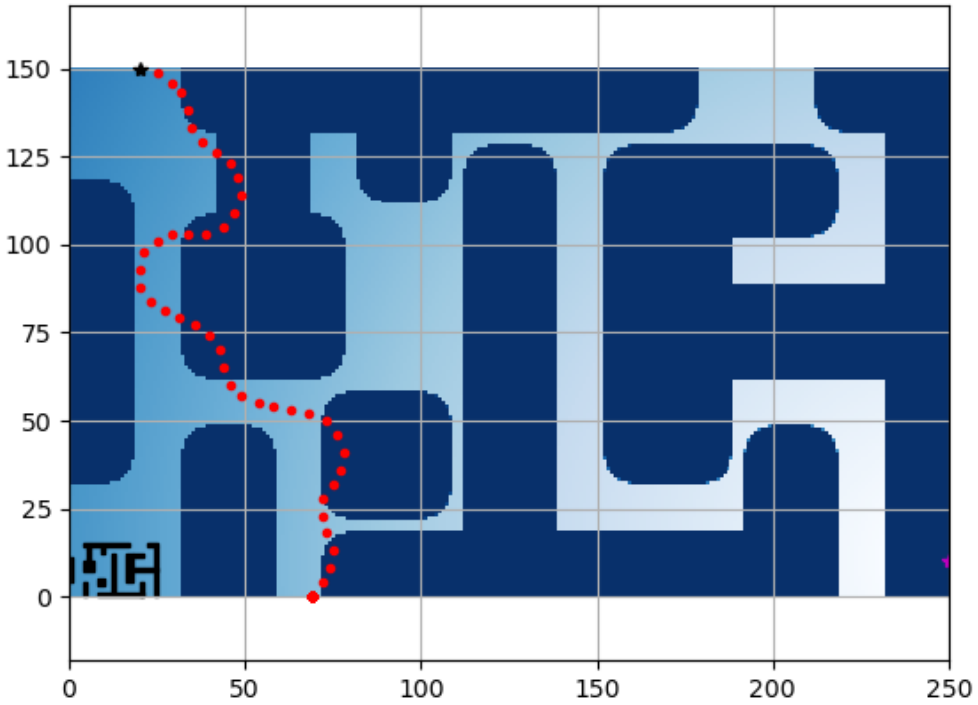

Figure 3: Artificial Potential Field 알고리즘 결과

3 Try to track the shortest path computed in Prob. 1 with the car in Prob. 2. Show the actual resulting trajectory of the car

Dubins car constraint dynamics를 고려하지 않고 shortest path를 구한 Prob. 1과 contraint를 고려하고 shortest path를 구한 Prob. 2-(a)의 결과를 비교하였다.

앞서 언급했듯이, Prob. 1은 A^*알고리즘을 통해서 구하였고, Prob. 2-(a)는 RRT^*알고리즘을 통해서 구하였다. 아래 2개 그림을 비교한 결과를 보면, non-holonomic한 car constraint로 인해 둘의 경로가 상이한 것을 알 수 있다.

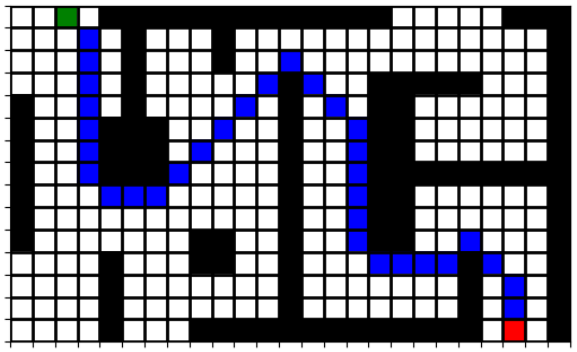

Figure 4: Dubins car dynamics를 고려하지 않은 최단 경로

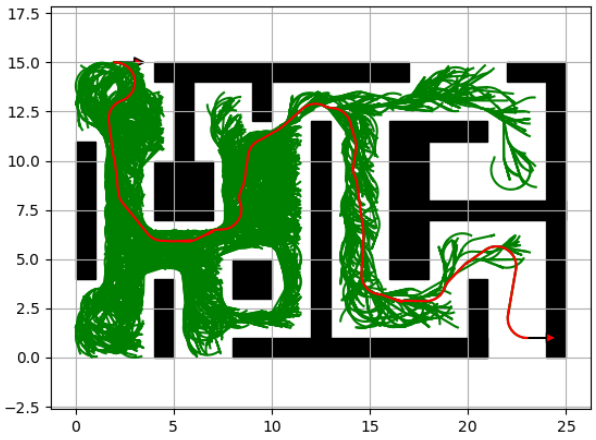

Figure 5: Dubins car dynamics를 고려한 최단 경로

4 Muti-agent path planning

가정:
차량이 위, 아래, 좌, 우, 정지로 총 5가지 움직임이 가능하다고 추가적으로 가정한다.

내용:
2개의 에이전트가 충돌없이 효율적인 경로를 갖기 위해서, A^* 알고리즘과 Conflict Based Scheduling 알고리즘을 병합했다. 기본적으로 A^* 알고리즘을 통해서 움직이지만 에이전트가 반경 6m 이내에서 장애물 없이 발견하게 된다면 centralized multi-agent path planning 알고리즘 중 하나인 conflict based scheduling 알고리즘으로 변경되게 하였다.
Conflict Based Scheduling은 2단계(high-level, low-level)의 검색으로 진행된다. High-level의 검색은 contraint tree(CT)에서 진행되며, 이 트리의 노드에는 에이전트 별 시간, 위치에 대한 조건이 포함되어 있다. 각 노드에서는 제한 사항을 지키면서 모든 에이전트의 경로를 계획하게 된다. A^*와 CBS의 가장 큰 차이점은 실행시간이다. A^*는 에이전트 수의 지수적으로 탐색해야 할 트리가 증가하지만 CBS는 문제 해결 중 마주치는 conflict 수의 지수적으로 탐색해야할 트리가 증가한다.
Conflict based scheduling에서 constraint 검사하는 부분에선 vertex constraint와 edge contraint 2가지를 고려했다. 이를 통해 장애물 및 에이전트간의 충돌 없이 효과적으로 경로를 계획할 수 있게 된다.
마지막으로, 에이전트 사이의 거리가 6m 이내이지만 장애물에 가려져서 보이지 않는 경우는 Bresenham's line algorithm을 사용하여 판단하였다. 에이전트 주변 3m 이내의 그리드들과 전부 직선을 이은 후, 직선 위에 장애물이 있는지 판단하였다.

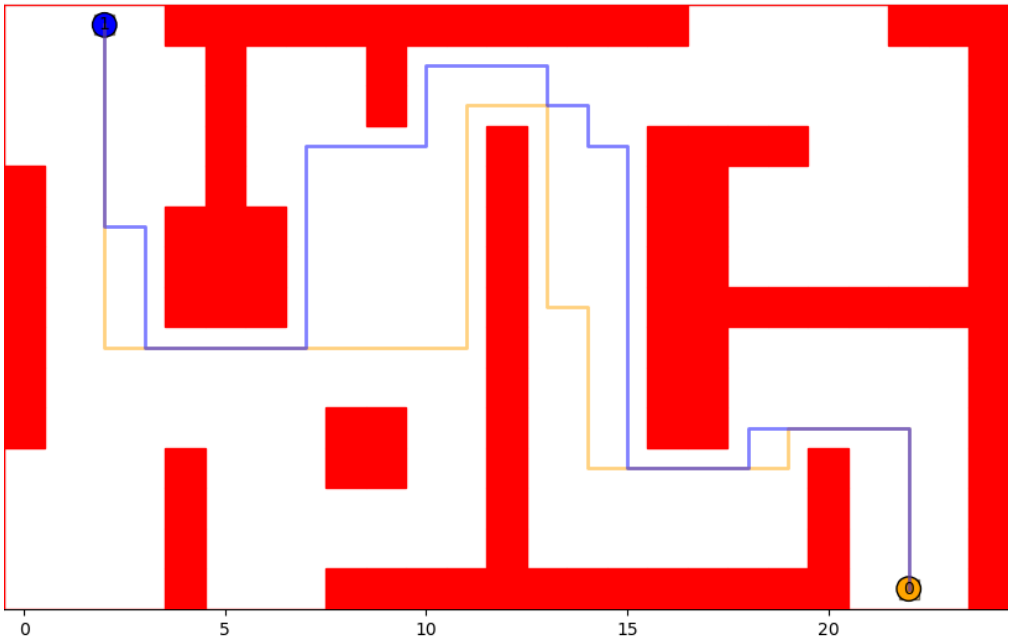

Figure 6: A^* 와 Conflict Based Search 결합 알고리즘 결과

Submitted by Jusuk Lee on April 11, 2024.