1 A* start search and navigation

A* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost. It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied. At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A* selects the path that minimizes :

$$f(n) = g(n) + h(n) \tag{1}$$

$g(n)$ is true cost. We define true cost = +1 move into a neighbor cell that is unoccupied, and true cost = 1000 to move into a neighbor cell that is occupied. There are 15 landmarks,which means there are 15 occupied cells.

$h(n)$ is heuristic, applying Euclidean distance estimated distance from the current node to the end node. The obstacle true cost is much greater than heuristic value. With this heuristic, we can make sure that we can actually calculate it and its admissible. It's also very important that this heuristic is always an underestimation of the total path, as an overestimation will lead to A* searching for through nodes that may not be the 'best' in terms of f value.

$f(n)$ is the total cost. A* is the algorithm prioritizes the node with the lowest f and the 'best' chance of reaching the end.

Robot start from the original point, and use original point as the current point, this current point would have children as 8 neighbors, and add all children to the open list. Then robot need to pick a node with smallest $f(n)$ from the open list, if more than one node have same smallest value of $f(n)$, then robot would pick a node which has smaller $g(n)$ and add this node to the close list, and delete this node from the open list. Also robot would use this node to update current node, and use the same method to keep update the current node until it match with the goal. The last step to trace back its parents to find the path.

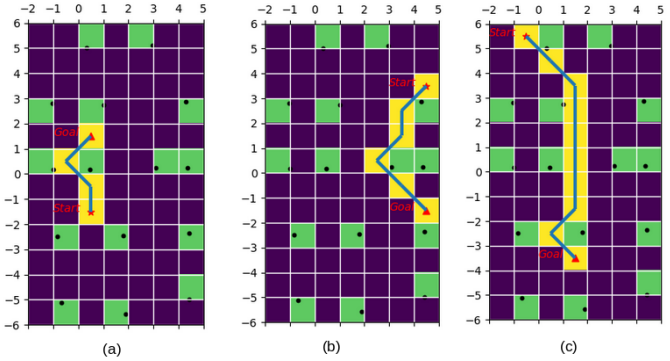

Figure 1: Plan path base on A* algorithm (a):S =[0.5,-1.5],G = [0.5,1.5], (b): s=[4.5,3.5], G =[4.5,-1.5], (c):S = [-0.5, 5.5],G = [1.5, -3.5]

Online refers that the robot can only see its immediate neighbors and has no knowledge about where obstacles are in the beginning.As robot moves and encounters new obstacles, it will update

the map of obstacles, after that, robot will have to re-plan accordingly. Robot can only expend notes from its immediate neighbors.

Online A* code: when robot at the start point, robot using question 2 code to find the best cost node which has smallest $f(n)$ among its neighbors. Then robot use this best node to replace its start point, which means robot then would start from this best node, then go through question 2 code again to find next best cost among its neighbor and update new start point. robot would keep doing this method until its new start point match its goal. All start points would be the path of Online A* code. So unlike Question 2, the open list for the online A* would only contain the 8 neighbors of current start point.

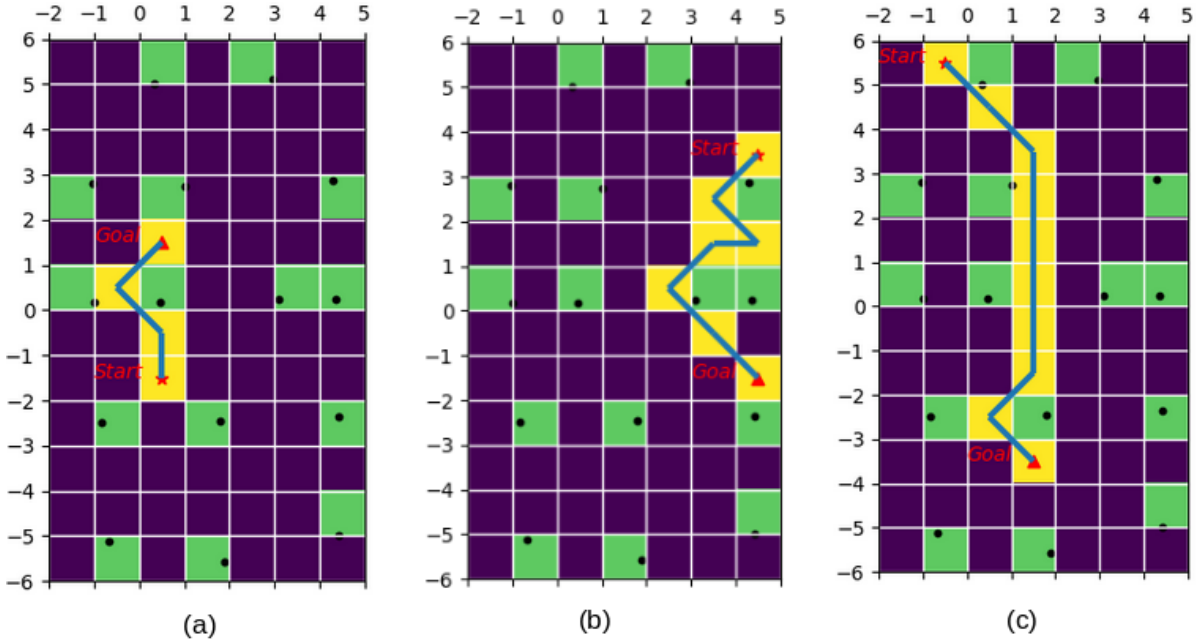

Figure 2: Plan path base on online A* algorithm (a):S =[0.5,-1.5],G = [0.5,1.5], (b): s=[4.5,3.5], G =[4.5,-1.5], (c):S = [-0.5, 5.5],G = [1.5, -3.5]

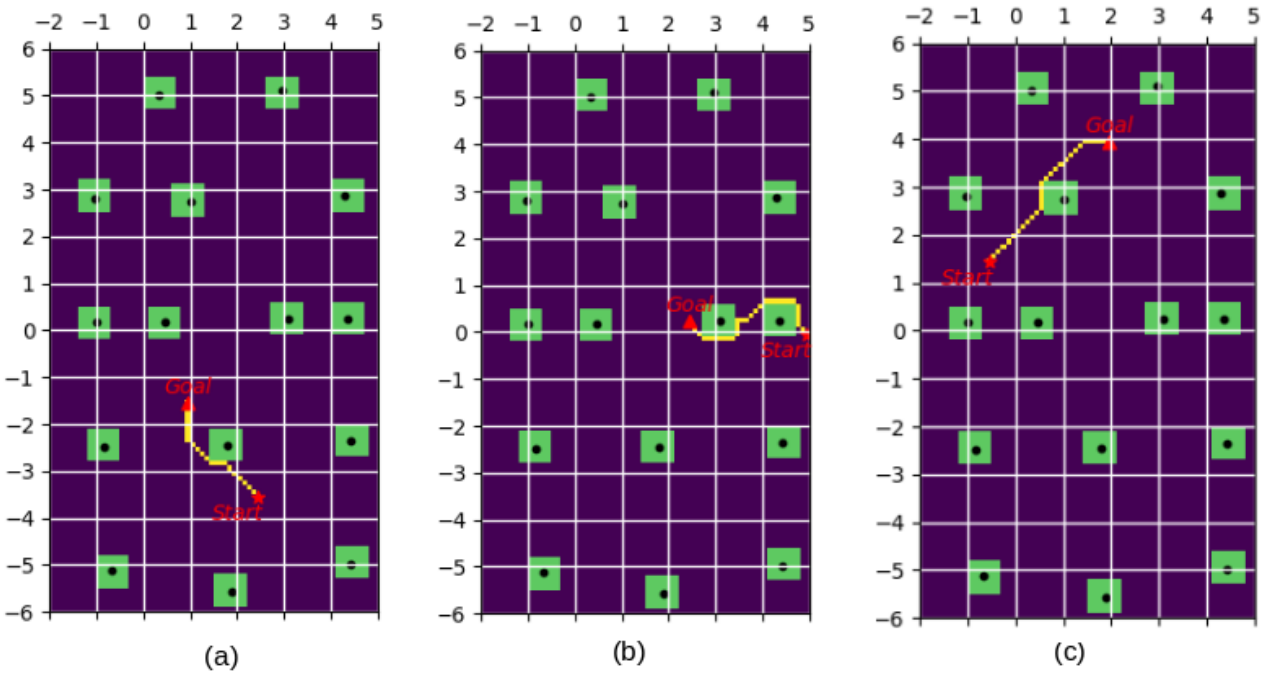

Figure 3: Plan path base on online A* algorithm (a):S =[2.45,-3.55],G = [0.95,-1.55], (b): s=[4.95,-0.05], G =[2.45,0.25], (c):S = [-0.55, 1.45],G = [1.95, 3.95]

B(Figure 3) shows on the robot path from the start path to the end path, robot need to move along the node. Given the robot current 2-D position and heading $[x, y, \theta]$,(θ is in range [0,2pi]),and the Target position [x_T,y_T] it is needed to design a inverse kinematics which out put linear velocity and angular velocity $[v, \omega]$ The robot walk between two node are designed to do the rotation motion first then do the linear motion. And the robot velocity at each node is 0.

The rotational motion:

$$\phi = tan^{-1}(x_T, y_T) - theta \tag{2}$$

If $\pi > \phi > 0$, robot need go clockwise

If $2\pi > \phi > \pi$, robot need go counterclockwise

If $-\pi < \phi < 0$, robot need go clockwise,

If $-2\pi < \phi < -\pi$.robot need go counterclockwise.

since $\dot{\omega}_{max} = 0.288$, robot plan to increase the speed of rotation with max acceleration, with lowest acceleration, and keep the max speed for 0.1 second if necessary and then decrease sped of rotation with max acceleration.

time of rotational motion:

$$T_a = 2 \times \sqrt{\frac{\theta}{0.288}} \tag{3}$$

The rotational motion:round t with 1 decimal

$$T_a = round(T_a, 1) \tag{4}$$

angular velocity: when $0 < t < \frac{T_a}{2}$

$$\omega = 0.288\times \tag{5}$$

angular velocity:

$$\omega_{max} = 0.288\frac{T_b}{2} \tag{6}$$

angular velocity:$T_a > t > \frac{T_a}{2}$

$$\omega = \omega_m ax - 0.288 \times t, \tag{7}$$

since $\dot{v}_{max} = 5.579$,, robot plan to increase the linear speed with max acceleration, then keep the max speed for 0.1 second if necessary and then decrease linear speed to 0 with max acceleration.

Linear distance between two node

$$S = \sqrt{(x_T - x)^2 + (y_T - y)^2)} \tag{8}$$

Total time of linear motion

$$T_b = 2 \times \sqrt{\frac{S}{5.579}} \tag{9}$$

Round $T_b with 1 decimal T_b = round(T_b, 1)(10)$
Linear velocity: when $0 < t < \frac{T_b}{2}$

$$v = 5.579 \times t \tag{11}$$

linear velocity:

$$v_{max} = 5.579 \times \frac{T_b}{2} \tag{12}$$

linear velocity:when $T_b > t > \frac{T_b}{2}$

$$v = v_{max} - 5.579 \times t \tag{13}$$

linear velocity:when $T_b > t > \frac{T_b}{2}$

$$v = v_{max} - 5.579 \times t \tag{14}$$

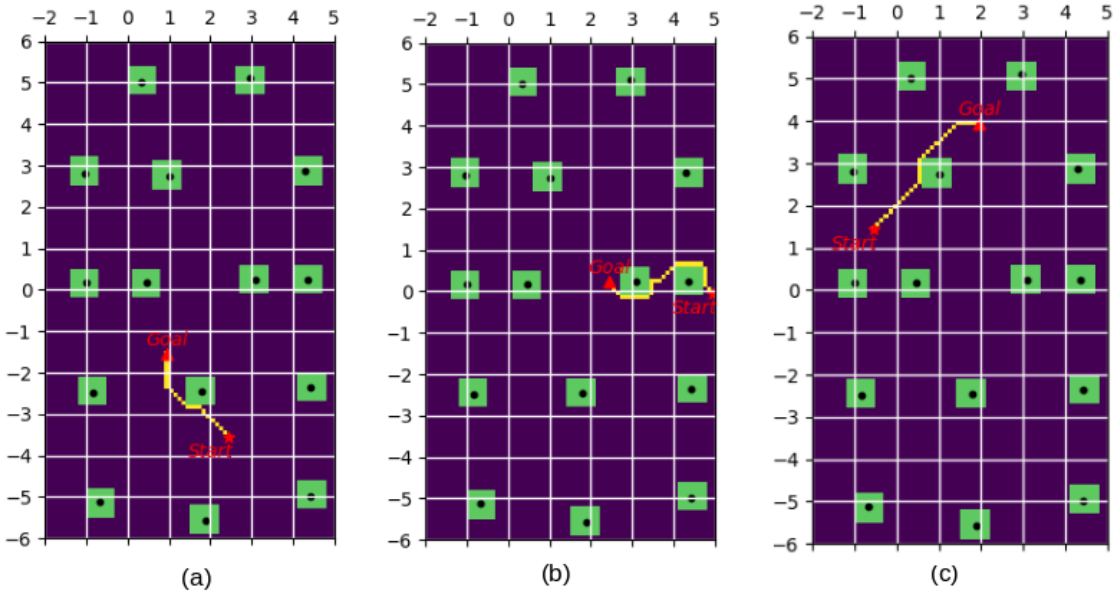

Figure 4: Plan path base on online A* algorithm (a):S =[2.45,-3.55],G = [0.95,-1.55], (b): s=[4.95,-0.05], G =[2.45,0.25], (c):S = [-0.55, 1.45],G = [1.95, 3.95]

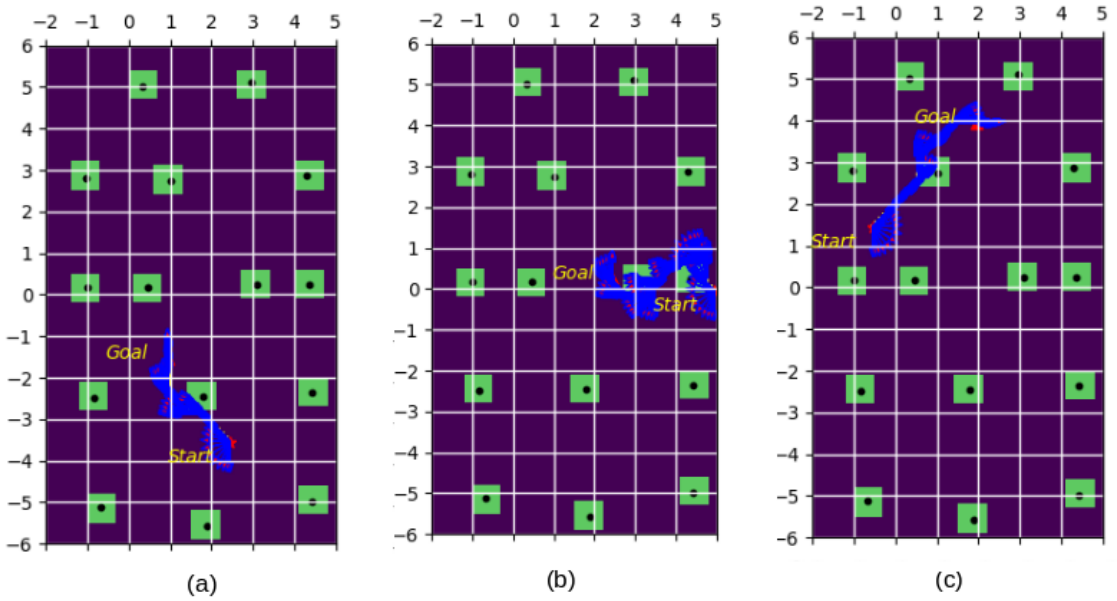

Figure 5: Robot path base on online A* algorithm (a):S =[2.45,-3.55],G = [0.95,-1.55], (b): s=[4.95,-0.05], G =[2.45,0.25], (c):S = [-0.55, 1.45],G = [1.95, 3.95]

From the figure, we can see robot is almost adhered with path planed by A*search, and it optimize rotation direction by choosing its counter-clock wise or clock wise direction. However, sometimes robot's heading is hard matching exactly target heading, it could cause problem of robot

getting close to the goal, but its hard to match with goal.

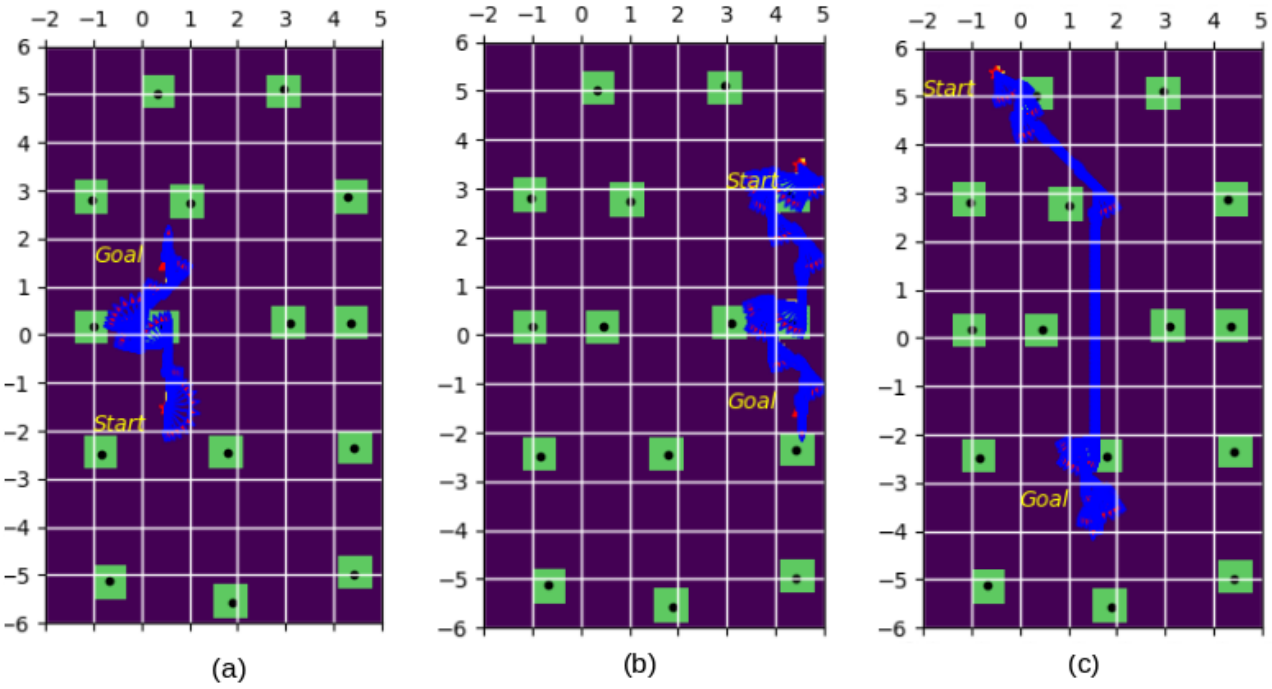

Figure 6: Plan path base on online A* algorithm (a):S =[0.5,-1.5],G = [0.5,1.5], (b): s=[4.5,3.5], G =[4.5,-1.5], (c):S = [-0.5, 5.5],G = [1.5, -3.5]

From the figure 6, Sometimes, robot heading is hard to map the exactly target, we can see path is getting longer than A*star path, since sometimes robot cannot get exactly pre-designed node, so robot need to re-plan the path, which means that path is much longer than A* path. The advantage is robot path can be change if the environment get change.