Aniket Sanjay Gujarathi
1007629979

Assignment 1

AER 1516
March 2, 2022

Problem 1

The bug2 algorithm works on the principle of motion-to-goal and boundary-following. During motion-to-goal, the robot moves towards the goal on the line joining the start point and the goal point (m-line) until it encounters an obstacle. If the robot encounters an obstacle, it circumnavigates the obstacle until it reaches a new point on the m-line closer to the goal than the initial point of contact of the obstacle. Then the robot proceeds towards the goal exhibiting the same behaviour till it reaches the goal.
Suppose for bug2 the line from q_{start} to q_{end} intersects the i^{th} obstacle n_i times. Then there are at most n_i leave points for this obstacle, since the robot may only leave the obstacle when it returns to a point on the m-line. As shown in Fig. 1 half of the leave points are not valid leave points as they may crash into an obstacle when moving towards the goal. Hence, in the worst case the robot will traverse nearly the entire perimeter, p_i, of the obstacle for each leave point. So the bug will never travel a distance greater than :

$$d_{tot} = d_{goal} + \frac{1}{2}\sum_{i=1}^{M} n_i p_i$$

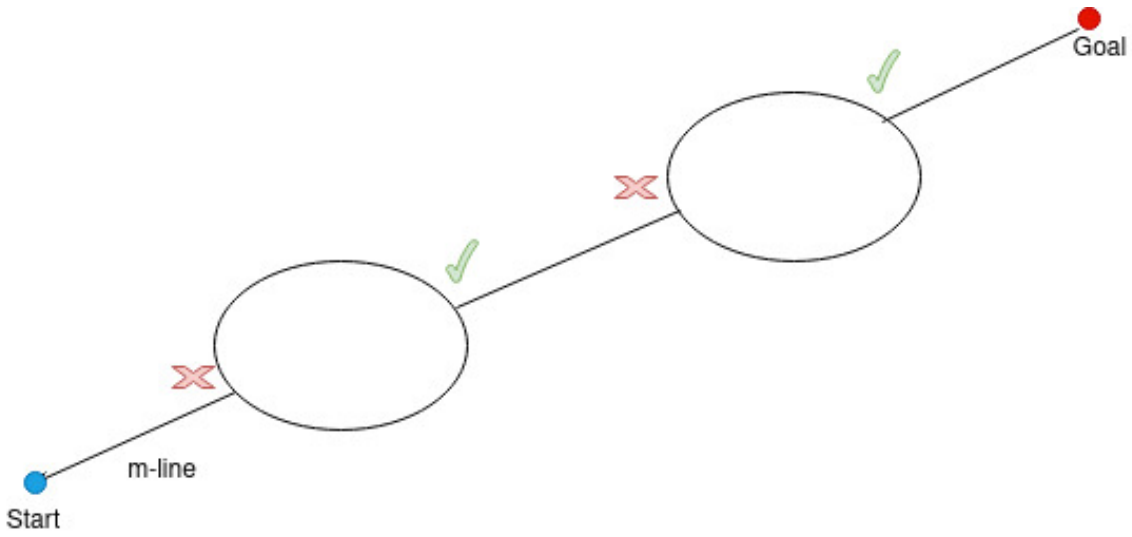

Figure 1: Simple Example for Bug2

Problem 2

Let us expand the Z-Y-X Euler angle sequence into the rotation matrix as:

$$C_{0,3} = C_{0,1}C_{1,2}C_{2,3}$$

$$C_{0,3} = \begin{bmatrix} cos(\alpha)cos(\beta) & cos(\alpha)sin(\beta)sin(\gamma) - sin(\alpha)cos(\gamma) & cos(\alpha)sin(\beta)cos(\gamma) + sin(\alpha)sin(\gamma) \\ sin(\alpha)cos(\beta) & sin(\alpha)sin(\beta)sin(\gamma) + cos(\alpha)cos(\gamma) & sin(\alpha)sin(\beta)cos(\gamma) - cos(\alpha)sin(\gamma) \\ -sin(\beta) & cos(\beta)sin(\gamma) & cos(\beta)cos(\gamma) \end{bmatrix}$$

We generally find the angles α and γ respectively from the first column and the last row (by dividing and taking the arctangent). When $\beta = \frac{\pi}{2}$ or $-\frac{\pi}{2}$, the entries used to compute the α and γ angles will also become zero (let us consider when $\beta = \frac{\pi}{2}$):

$$C_{0,3} = \begin{bmatrix} 0 & cos(\alpha)sin(\gamma) - sin(\alpha)cos(\gamma) & cos(\alpha)cos(\gamma) + sin(\alpha)sin(\gamma) \\ 0 & sin(\alpha)sin(\gamma) + cos(\alpha)cos(\gamma) & sin(\alpha)cos(\gamma) - cos(\alpha)sin(\gamma) \\ -1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & sin(\gamma - \alpha) & cos(\alpha - \gamma) \\ 0 & cos(\alpha - \gamma) & sin(\alpha - \gamma) \\ -1 & 0 & 0 \end{bmatrix}$$

Hence, the entries to compute the other angles become zero and hence they cannot be recovered uniquely. This is also called as Gimbal Lock or singularity. Due to gimbal lock, one degree of freedom is lost - in this case the ability to roll.

Problem 3

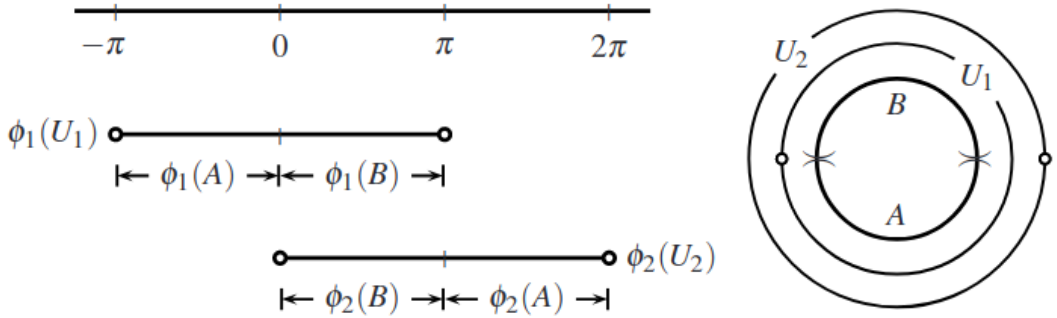

Figure 2: Atlas on the unit circle [1]. ($\phi 1 = \phi$, $\phi 2 = \psi$, $U1 = U$, $U2 = V$)

As shown in Fig. 2, the unit circle $S^1 = (x, y)|x^2 + y^2 = 1$ can be covered by the following

atlas :

$$U = S^1 \setminus (-1, 0),\ \phi = arctan(\frac{y}{x})\ with\ -\pi < \phi(x, y) < \pi$$
$$V = S^1 \setminus (1, 0),\ \psi = arctan(\frac{y}{x})\ with\ 0 < \psi(x, y) < 2\pi$$

The domain union is $U \cup V = S^1$. As we need the domains to be open sets, U covers S^1 but excludes the point $(1, 0)$ and similarly for V excluding the point $(-1, 0)$. By defining the atlas as above, we can cover entire S^1.
To show that the map $\phi \circ \psi^{-1}$ is a diffeomorphism, we can define $\phi(x, y)$ as the unique $-\pi < \theta < \pi$ such that $(x, y) = (cos(\theta), sin(\theta))$.

$$\phi \circ \psi^{-1}(\theta) = \begin{cases} \theta, & \text{if } 0 < \theta < \pi \\ \theta + 2\pi, & -\pi < \theta < 0 \end{cases} \qquad \psi \circ \phi^{-1}(\theta) = \begin{cases} \theta, & \text{if } 0 < \theta < \pi \\ \theta - 2\pi, & \pi < \theta < 2\pi \end{cases}$$

Hence smooth and bijective maps exist between U and V, thus the charts are diffeomorphic to each other.

Problem 4

The number of degrees of freedom of a system is the dimension of the configuration space.

1. Two mobile robots rotating and translating in a plane:
   For two mobile robots in a plane, every robot has 3 degrees of freedom - 2 for translation and 1 for rotation. Hence, the dimension of the configuration space is 6.

2. Two mobile robots tied together by a rope rotating and translating in the plane:
   Considering the system of the robot with a rope, we can select n points of the rope (given that the rope can slack). To define the configuration of the robots with the rope, we would need n points as well as the orientation of the robots. Hence, the dimension of the configuration space will be $N + 2$, wherein N defines the configuration of the points on the rope (including the end points of the rope where the robots are tied) and 2 represents the orientation of the robots.

3. A train on train tracks, including the wheel angles? (The wheels roll without slipping):
   A train on train tracks has 1 DOF in translation. As we know the distance travelled and the radius of the wheels, we can figure out the orientation of the wheels with the information. As the wheels are constrained to move together, if we know the configuration of one set of wheels, we automatically have the knowledge of the other side. Hence, the dimension of the configuration space is 1.

4. Your legs as you pedal a bicycle (remaining seated with feet fixed to the pedals):
   A leg on the bicycle can only rotate in the frame of reference of the bicycle. The legs

on a bicycle have only 1 DOF. This is because when one leg is at a position on the pedal, the other leg is constrained by its motion and must be at a fixed position on the other end. Hence, the dimension of the configuration space is 1.

Problem 5

For the problem:

1. $W = R^2$ is the workspace
2. $WO \subset W$ denotes the workspace obstacles.
3. $A \subset W$ is the robot in the given workspace
4. $q \in C$ denotes the configuration
5. CO denotes the configuration obstacles

To Prove: $C(WO_i \cup WO_j) = CO_i \cup CO_j$
Proof:

$$\begin{aligned} RHS &= CO_i \cup CO_j \\ &= \{q \in C | A(q) \cap WO_i \neq \emptyset\} \cup \{q \in C | A(q) \cap WO_j \neq \emptyset\} \\ &= \{q \in C | (A(q) \cap WO_i) \cup (A(q) \cap WO_j) \neq \emptyset\} \\ &= \{q \in C | A(q) \cap (WO_i \cup WO_j) \neq \emptyset\} \\ &= C(WO_i \cup WO_j) \\ &= LHS \end{aligned}$$

Problem 6

Yes, the wavefront planner determines the shortest path, but at the cost of coming very close to the obstacles. The wavefront planner starts by assigning the goal cell in the grid map as 2 and assigning the neighbouring pixels with an increment of 1 till the start pixel of the robot is reached. The planner performs a breadth-first iteration of the graph to assign the values. The planner then determines the path via gradient descent on the grid starting from the start position.
The wavefront planner can be considered as a special case of the Dijkstra's algorithm that optimizes the number of stages to reach the goal. Due to the assignment of values to each pixel in the grid, we assure that the cost associated with each wavefront is monotonically increasing till the start position is reached. Due to this assignment, the wavefront planner essentially

forms a potential function on the grid which has only one local minima. Boundedness of the free space (and hence the discretization) and continuity of the distance function ensure that construction of the wave front guarantees that there will always be a neighboring pixel whose value is one less than that of the current pixel and that this procedure forms a path in the grid to the goal, i.e., to the pixel whose value is two. Hence, the gradient descent algorithm is able to find the shortest path as every traversal through the wavefront decreases the cost to goal till the goal is reached. The wavefront planner is using the Manhattan Distance, as all the points on the same wavefront have the same manhattan distance to the goal.

References

[1] Loring W. Tu. *An Introduction to Manifolds*. https://doi.org/10.1007/978-1-4419-7400-6. 2011.