Lab 3 – Dubins trajectories

REPLAN team

Tuesday 11th March, 2025

Contents

compiled at: 11/03/2025, 08:08

The idea

Using the mathematical model of the Dubins car, we build the Dubins trajectories to achieve a desired configuration.

1 Theoretical background

We consider the simplified "Dubins car" which cannot slip laterally ("sideslip") and which can go only forward [1]:

$$\begin{cases} \dot{x} &= u_V \cos\theta, \\ \dot{y} &= u_V \sin\theta, \\ \dot{\theta} &= \dfrac{u_V}{L} \tan u_\theta, \end{cases} \quad \xrightarrow{\text{simplified}} \quad \begin{cases} \dot{x} = & \cos\theta \\ \dot{y} = & \sin\theta \\ \dot{\theta} = & u \end{cases} \tag{1}$$

The car is driven by $u_V \in \{-1, 1\}$ and $u \in (-\phi_{\max}, \phi_{\max})$. The state vector is given by position (x and y) and orientation (θ). The dynamics impose limitations:

- The car cannot turn "in place"; it has a maximal turn angle $\phi_{\max} < \pi/2$, or, equivalently stated, a minimum turn radius $\rho_{\min} = L/\tan\phi_{\max}$;
- The model is nonholonomic because there appear differential constraints which cannot be integrated away, in our case:

$$-\dot{x}\sin\phi + \dot{y}\cos\phi = 0.$$

We are interested in tracing a path between two arbitrary configurations (configuration = combination of position and orientation of the vehicle):

$$(x(t_i), y(t_i), \phi(t_i)) \qquad \longrightarrow \qquad (x(t_f), y(t_f), \phi(t_f)) \tag{2}$$

It is known that, when minimizing the path length,

$$L(\tilde{q}, \tilde{u}) = \int_{t_i}^{t_f} \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} dt \tag{3}$$

for any pair of configurations there are only 6 optimal possibilities for tracing the Dubins trajectory:

$$\{LRL, RLR, LSL, LSR, RSL, RSR\}, \tag{4}$$

where, L – left, R – right și S – straight. Hence, the path between two successive configurations is uniquely defined by the word chosen from (4) and the length of each component (angle for L, R, or distance for S).

2 Implementing a Dubins trajectory in the plan

There are many libraries that generate Dubins paths, for example, https://github.com/fgabbert/dubins_py which has the major advantage that it is "simple": there are no dependencies, it can be used regardless of the platform.

For illustration, this is the code snippet that calculates the parameters of an LSL path:

```python
def dubinsLSL(alpha, beta, d):
    tmp0        = d + math.sin(alpha) - math.sin(beta)
    tmp1        = math.atan2((math.cos(beta)-math.cos(alpha)),tmp0)
    p_squared = 2 + d*d - (2*math.cos(alpha-beta)) + (2*d*(math.sin(alpha)-math.sin(beta)))
    if p_squared<0:
        print('No LSL Path')
        p=-1
        q=-1
        t=-1
    else:
        t           = (tmp1-alpha) % (2*math.pi)
        p           = math.sqrt(p_squared)
        q           = (beta - tmp1) % (2*math.pi)
    return t, p, q
```

The implementation is based on the article [2] and the parameters that appear in the code snippet respect the relation (the fragment computes the path length for the LSL word):

$$L_q(S_p(L_t(0,0,\alpha))) = (d,0,\beta),$$

which says that, starting from a configuration $(0,0,\alpha)$ we arrive at $(d,0,\beta)$ ← various changes of coordinates and assumptions are made to arrive at these particular forms. The implementation of the code is based on solving the equations:

$$\begin{aligned}
p\cos(\alpha+t) - \sin\alpha + \sin\beta &= d\\
p\sin(\alpha+t) + \cos\alpha - \cos\beta &= 0\\
\alpha + t + q &= \beta[mod\, 2\pi]
\end{aligned}$$

whose solution is given as:

$$\begin{aligned}
t &= -\alpha + \arctan\frac{\cos\beta-\cos\alpha}{d+\sin\alpha-\sin\beta}[mod\, 2\pi]\\
p &= \sqrt{2+d^2-2\cos(\alpha-\beta)+2d(\sin\alpha-\sin\beta)}\\
q &= \beta - \arctan\frac{\cos\beta-\cos\alpha}{d+\sin\alpha-\sin\beta}[mod\, 2\pi]
\end{aligned}$$

Then, the length of the path is

$$\mathcal{L}_{\text{LSL}} = -\alpha + \beta + p_{\text{LSL}}.$$

3 Proposed exercises

Using and modifying the available code, solve the following exercises.

Exercise 1. Consider a list of $N \geq 10$ waypoints whose configuration and order of passing are known. Implement the following:

i) Compute and plot the overall Dubins trajectory (stitching together all consecutive segments).

ii) Plot the total path length as a function of ρ_{min}, the minimum turn radius.

Exercise 2. Consider a list of three waypoints given as

$$(x_1, y_1, \theta_1), \quad (x_2, y_2, \star), \quad (x_3, y_3, \theta_3)$$

where the orientation of the middle waypoint is left undefined ($\theta_2 = \star$). Implement the following:

i) Compute and plot the resulted trajectories for a range of values of $\theta_2 \in (-\pi/2, \pi/2)$.

ii) Plot the total path length as a function of θ_2. Do you observe a change in active words?

iii) Repeat the procedure but for a different combination of initial and final way-point.

Exercise 3. Consider a list of $N \geq 10$ waypoints whose configuration is fully-known and where the first and last waypoint are known. Construct the shortest path passing through all of them. **Indication**: You can construct an adjacency graph where the weight of the edge connecting two WPs is given by the Dubins trajectory. Then, you may solve the exercise as a "traveling salesman problem".

Exercise 4. Consider the decision table for the shortest path

	1	2	3	4
1	RSL	if $S_{12} < 0$ then RSR if $S_{12} > 0$ then RSL	if $S_{13} < 0$ then RSR if $S_{13} > 0$ then LSR	if $S^1_{14} > 0$ then LSR if $S^2_{14} > 0$ then RSL, else RSR
2	if $S_{21} < 0$ then LSL if $S_{21} > 0$ then RSL	if $S^1_{22} < 0$ then LSL if $S^1_{22} > 0$ then RSL if $S^2_{22} < 0$ then RSR if $S^2_{22} > 0$ then RSL	RSR	if $S_{24} < 0$ then RSR if $S_{24} > 0$ then RSL
3	if $S_{31} < 0$ then LSL if $S_{31} > 0$ then LSR	LSL	if $S^1_{33} < 0$ then RSR if $S^1_{33} > 0$ then LSR if $S^2_{33} < 0$ then LSL if $S^2_{33} > 0$ then LSR	if $S_{34} < 0$ then RSR if $S_{34} > 0$ then LSR
4	if $S^1_{41} > 0$ then RSL if $S^2_{41} > 0$ then LSR, else LSL	if $S_{42} < 0$ then LSL if $S_{42} > 0$ then RSL	if $S_{43} < 0$ then LSL if $S_{43} > 0$ then LSR	LSR

with

$$S_{12} = p_{RSR} - p_{RSL} - 2(q_{RSL} - \pi), \qquad S_{13} = t_{RSR} - \pi$$
$$S^1_{14} = t_{RSR} - \pi, \qquad S^2_{14} = q_{RSR} - \pi$$
$$S_{21} = p_{LSL} - p_{RSL} - 2(t_{RSL} - \pi) \qquad S^1_{22} = p_{LSL} - p_{RSL} - 2(t_{RSL} - \pi)$$
$$S^2_{22} = p_{LSL} - p_{RSL} - 2(q_{RSL} - \pi) \qquad S_{24} = q_{RSR} - \pi$$

$$
\begin{aligned}
S_{31} &= q_{LSL} - \pi, & S_{33}^1 &= p_{RSR} - p_{LSR} - 2(t_{LSR} - \pi) \\
S_{33}^2 &= p_{RSR} - p_{LSL} - 2(q_{LSR} - \pi), & S_{34} &= p_{RSR} - p_{LSR} - 2(t_{LSR} - \pi) \\
S_{41}^1 &= t_{LSL} - \pi, & S_{41}^2 &= q_{LSL} - \pi \\
S_{42} &= t_{LSL} - \pi, & S_{43} &= p_{LSL} - p_{LSR} - 2(q_{LSR} - \pi)
\end{aligned}
$$

Sample the space of $(\alpha, \beta) \in (-\pi, \pi)^2$ with a uniform grid (for example, with 100×100 points) and, using the table and associated switching functions, determine the active regions (by example, color "blue" for all points that correspond to RSL).

References

[1] Steven LaValle. *Planning Algorithms / Motion Planning*. Cambridge university press, 2006. URL: https://lavalle.pl/planning/ (visited on 02/04/2025).

[2] Andrei M. Shkel and Vladimir Lumelsky. "Classification of the Dubins set". In: *Robotics and Autonomous Systems* 34.4 (2001). Publisher: Elsevier, pp. 179–202. URL: https://www.sciencedirect.com/science/article/pii/S0921889000001275 (visited on 03/09/2025).