Optimal sampling-based motion planning of a Dubins car

Course Project
COM S 576 Motion Strategy Algorithms and Applications by Dr. Nok

Team member: Mohammad Hashemi

05/05/2023

Introduction and problem formulation

In this project, motion planning with differential constraints using sampling-based algorithm is explored using simple models and algorithms due to the limited time and scope of the project. It is crucial to study such motion planning problems because many real-world robotics problems involve continuous state spaces and feedback control systems to follow a given path considering the world uncertainties and dynamics of the robot, and considering the differential constraints arising from the robot's kinematics and dynamics in the motion planning is an advantageous approach due to its better compliance with the natural motions (LaValle, 2006). Here, a simple car model with Dubins constraints, a simple setup of the world and the obstacles as defined homework 4 of this course, and some variations of optimal sampling-based algorithms, specifically, k-RRT* and k-PRM* (Karaman et al., 2011) are considered. From here to the end of this report, the paper refers to the main reference of this project (Karaman et al., 2011), unless otherwise is explicitly stated. The formal problem formulation is as follows.

The robot is a rigid body moving in the plane with configuration or C-space $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}$ and a configuration $q = (x, y, \theta)$, as depicted by figure 1.

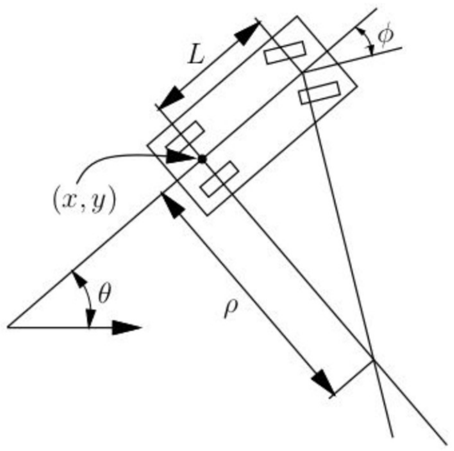


Figure 1. A simple car model schematic.

The origin of the body frame is at the center of the rear axle. The x-axis points along the main axis of the car. Wheels cannot slide sideways, so it is an underactuated system due to less action variables than degrees of freedom and a Nonholonomic one due to differential constraints that cannot be completely integrated, i.e., the configurations are not restricted to a lower dimensional subspace of $\boldsymbol{\mathcal{C}}$. In a small time interval, the car must move approximately in the direction that the rear wheels are pointing. The configuration transition equation is given by $\dot{q} = f(q, u)$, where $u \in U(q)$ can be interpreted as an abstract action vector and U(q) is assumed to be fixed for all q, i.e., $U(q) = U$. Suppose that the speed s and the steering angle ϕ are directly specified by the action variables u_s and u_ϕ, respectively. Therefore, the action vector is $u = (u_s, u_\phi)$, and the following differential constraints should be satisfied:

$$\begin{aligned}\dot{x} &= u_s \cos\theta \\ \dot{y} &= u_s \sin\theta \\ \dot{\theta} &= \frac{u_s}{L} \tan u_\phi\end{aligned}$$

The maximum steering angle of a simple car should be $\phi_{max} < \pi/2$ due to physical constraints. Consequently, we require that $|\phi| \leq \phi_{max}$, and the minimum turning radius is $\rho_{min} = L \,/\, \tan(\phi_{\max})$. Furthermore, to safely neglect the dynamics, e.g., the dependency between maximum steering angle and speed and having infinite acceleration or sudden changes in the speed, $|u_s| \leq 1$. The obstacles are semi-algebraic objects, specifically semi-circles in the experiments. A Dubins car constrain the set of possible actions such that $U = \{0, 1\} \times (-\phi_{max}, \phi_{ma}\ )$. For the Dubins car, a distance function can be defined based on the length of the shortest path between two configurations. The shortest path for the Dubins car can always be expressed as a combination of no more than 3 motion primitives: R or turn as sharply as possible to the right, L or turn as sharply as possible to the left, and S or drive straight. An optimal path can always be characterized by one of the six types: RSR, RSL, LSR, LSL, RLR, LRL. Note that the distance function violates the symmetry axiom and is not continuous.

In summary, the problem is a sample of Piano Mover's problem with differential constraints. There is a world $\mathcal{W} \subset \mathbb{R}^2$, a semi-algebraic robot $\mathcal{A} \subset \mathcal{W}$, a semi-algebraic obstacles region $\mathcal{O} \subset \mathcal{W}$, and the configuration space $\mathcal{C}$. An unbounded time interval $T = [0, \infty)$ is available, and time is not considered explicitly in the solution. The state space X is a smooth manifold. Let $\kappa: X \to \mathcal{C}$ denote a function that returns the configuration $q \in \mathcal{C}$ associated with $x \in X$. An obstacle region X_{obs} is defined for the state space, and $X_{free} = X \backslash X_{obs}$. For each state $x \in X$, a bounded action space $U(x) \subseteq \mathbb{R}^m \cup \{u_T\}$ where m is the number of action variables, and u_T represents the final action (causing no change in the state). A system is specified using the state transition equation $\dot{x} = f(x, u)$, defined for every x ∈ X and u ∈U(x) such that $f(x, u_T) = 0$. The initial state is $x_I \in X_{free}$, and the goal region $X_G \in X_{free}$. Since optimal sampling-based algorithms are explored in this project, a probabilistically complete and asymptotically optimal algorithm is desired such that it computes an action trajectory $\tilde{u}: T \to U$, for which the state trajectory satisfies $x(0) = x_I$, there exists some $t^* > 0$ for which $u(t^*) = u_T$ and $x(t^*) \in X_G$, and $\forall t < t^*: x(t) \in X_{free}$ (feasibility), as well as satisfying the asymptotic optimality condition as defined in the paper and based on the Dubins distance function. The exact characteristics and geometrical features of the problem studied in this project are given by those of homework 5 as shown in summary in figure 2.

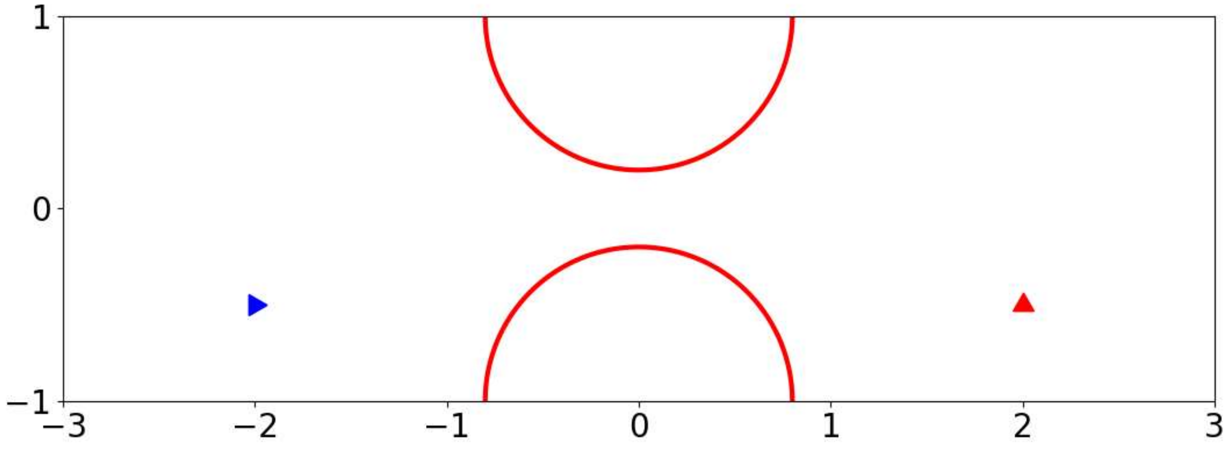


Figure 2. The world and obstacles (the red half-circles are the obstacles, and the blue and red arrows represent the initial and goal configurations, respectively).

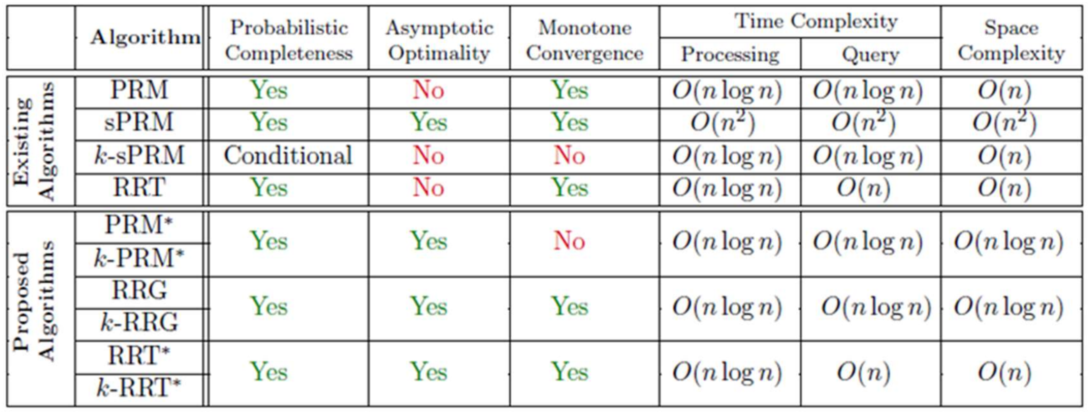
Methods

Based on the paper, two optimal sampling-based algorithms, k-RRT* and k-PRM*, are implemented by expanding the solution codes provided in homework 5. In both, for the variable k in each iteration, the following function is considered:

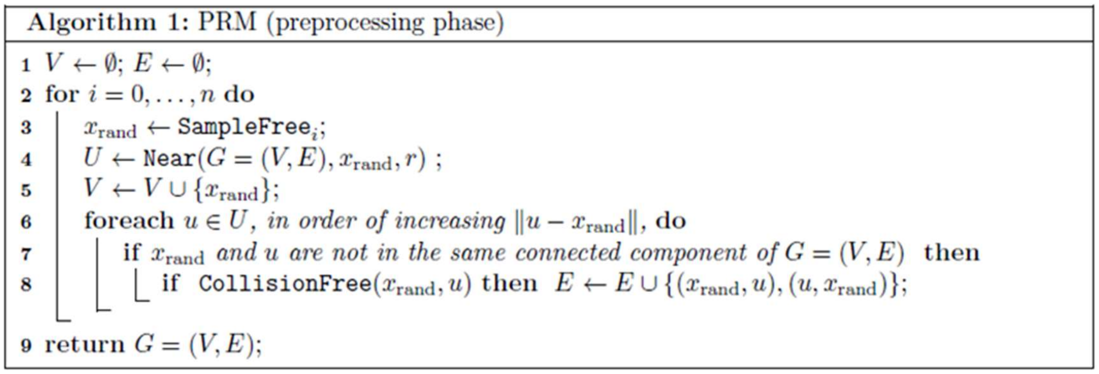
$$k(n) := k_{RRG}\log(n = |V|)\,;\, k_{RRG} = 2e$$

The explored algorithms are as follow:

Table 1: Summary of results. Time and space complexity are expressed as a function of the number of samples n, for a fixed environment.

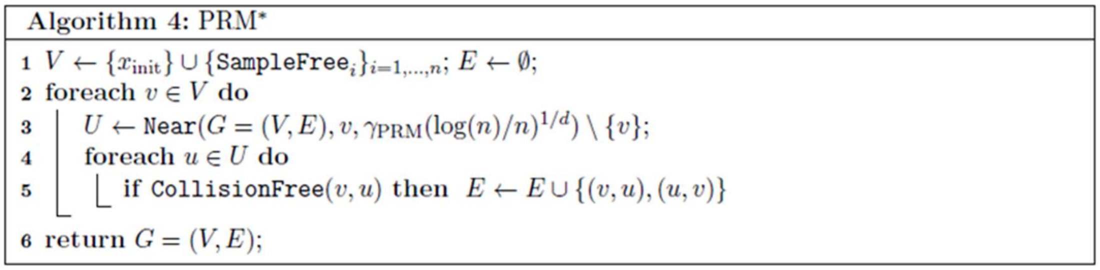


	Algorithm	Probabilistic Completeness	Asymptotic Optimality	Monotone Convergence	Time Complexity		Space Complexity
					Processing	Query	
Existing Algorithms	PRM	Yes	No	Yes	$O(n\log n)$	$O(n\log n)$	$O(n)$
	sPRM	Yes	Yes	Yes	$O(n^2)$	$O(n^2)$	$O(n^2)$
	k-sPRM	Conditional	No	No	$O(n\log n)$	$O(n\log n)$	$O(n)$
	RRT	Yes	No	Yes	$O(n\log n)$	$O(n)$	$O(n)$
Proposed Algorithms	PRM*, k-PRM*	Yes	Yes	No	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
	RRG, k-RRG	Yes	Yes	Yes	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$
	RRT*, k-RRT*	Yes	Yes	Yes	$O(n\log n)$	$O(n)$	$O(n)$



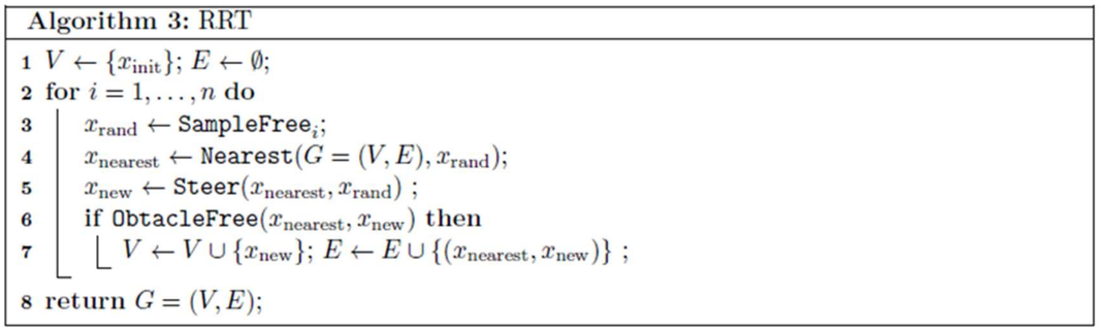
Algorithm 1: PRM (preprocessing phase)

```
1 $V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
2 for $i = 0, \ldots, n$ do
3     $x_{\mathrm{rand}} \leftarrow$ SampleFree$_i$;
4     $U \leftarrow$ Near$(G = (V, E), x_{\mathrm{rand}}, r)$ ;
5     $V \leftarrow V \cup \{x_{\mathrm{rand}}\}$;
6     foreach $u \in U$, in order of increasing $\|u - x_{\mathrm{rand}}\|$, do
7         if $x_{\mathrm{rand}}$ and $u$ are not in the same connected component of $G = (V, E)$ then
8             if CollisionFree$(x_{\mathrm{rand}}, u)$ then $E \leftarrow E \cup \{(x_{\mathrm{rand}}, u), (u, x_{\mathrm{rand}})\}$;
9 return $G = (V, E)$;
```



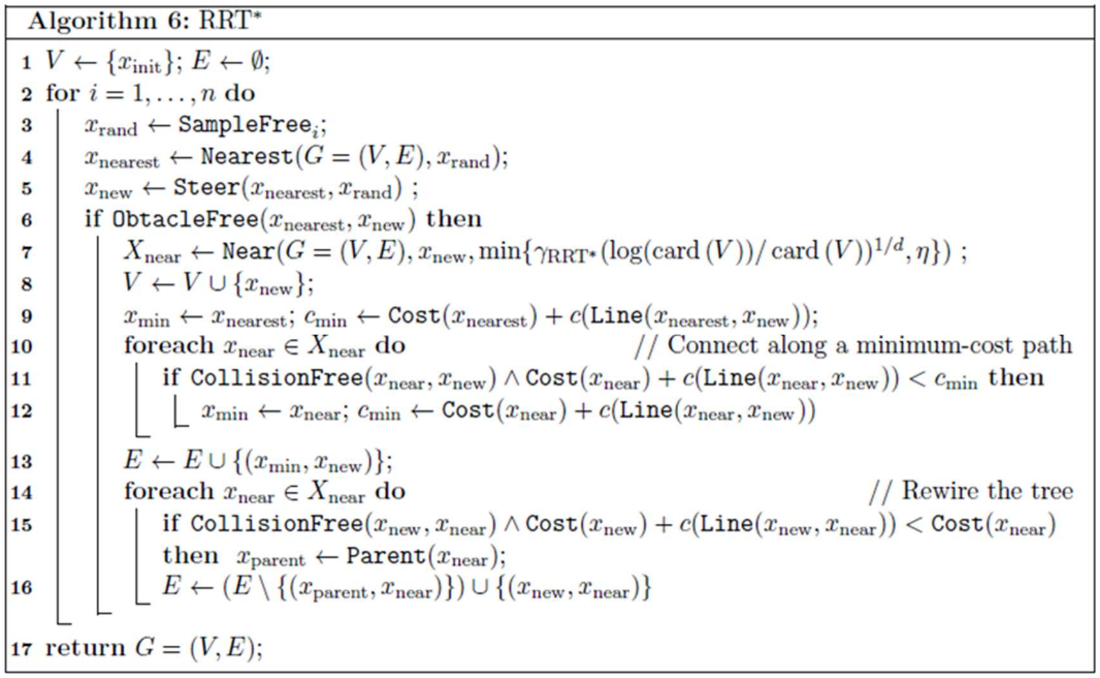
Algorithm 4: PRM*

```
1 $V \leftarrow \{x_{\mathrm{init}}\} \cup \{$SampleFree$_i\}_{i=1,\ldots,n}$; $E \leftarrow \emptyset$;
2 foreach $v \in V$ do
3     $U \leftarrow$ Near$(G = (V, E), v, \gamma_{\mathrm{PRM}}(\log(n)/n)^{1/d}) \setminus \{v\}$;
4     foreach $u \in U$ do
5         if CollisionFree$(v, u)$ then $E \leftarrow E \cup \{(v, u), (u, v)\}$
6 return $G = (V, E)$;
```



Algorithm 3: RRT

```
1 $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
2 for $i = 1, \dots, n$ do
3     $x_{\text{rand}} \leftarrow$ SampleFree$_i$;
4     $x_{\text{nearest}} \leftarrow$ Nearest$(G = (V, E), x_{\text{rand}})$;
5     $x_{\text{new}} \leftarrow$ Steer$(x_{\text{nearest}}, x_{\text{rand}})$ ;
6     if ObtacleFree$(x_{\text{nearest}}, x_{\text{new}})$ then
7         $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$ ;
8 return $G = (V, E)$;
```



Algorithm 6: RRT*

```
1  $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
2  for $i = 1, \dots, n$ do
3      $x_{\text{rand}} \leftarrow$ SampleFree$_i$;
4      $x_{\text{nearest}} \leftarrow$ Nearest$(G = (V, E), x_{\text{rand}})$;
5      $x_{\text{new}} \leftarrow$ Steer$(x_{\text{nearest}}, x_{\text{rand}})$ ;
6      if ObtacleFree$(x_{\text{nearest}}, x_{\text{new}})$ then
7          $X_{\text{near}} \leftarrow$ Near$(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$ ;
8          $V \leftarrow V \cup \{x_{\text{new}}\}$;
9          $x_{\min} \leftarrow x_{\text{nearest}}$; $c_{\min} \leftarrow$ Cost$(x_{\text{nearest}}) + c($Line$(x_{\text{nearest}}, x_{\text{new}}))$;
10         foreach $x_{\text{near}} \in X_{\text{near}}$ do          // Connect along a minimum-cost path
11             if CollisionFree$(x_{\text{near}}, x_{\text{new}}) \wedge$ Cost$(x_{\text{near}}) + c($Line$(x_{\text{near}}, x_{\text{new}})) < c_{\min}$ then
12                 $x_{\min} \leftarrow x_{\text{near}}$; $c_{\min} \leftarrow$ Cost$(x_{\text{near}}) + c($Line$(x_{\text{near}}, x_{\text{new}}))$
13         $E \leftarrow E \cup \{(x_{\min}, x_{\text{new}})\}$;
14         foreach $x_{\text{near}} \in X_{\text{near}}$ do          // Rewire the tree
15             if CollisionFree$(x_{\text{new}}, x_{\text{near}}) \wedge$ Cost$(x_{\text{new}}) + c($Line$(x_{\text{new}}, x_{\text{near}})) <$ Cost$(x_{\text{near}})$
               then $x_{\text{parent}} \leftarrow$ Parent$(x_{\text{near}})$;
16                 $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
17 return $G = (V, E)$;
```

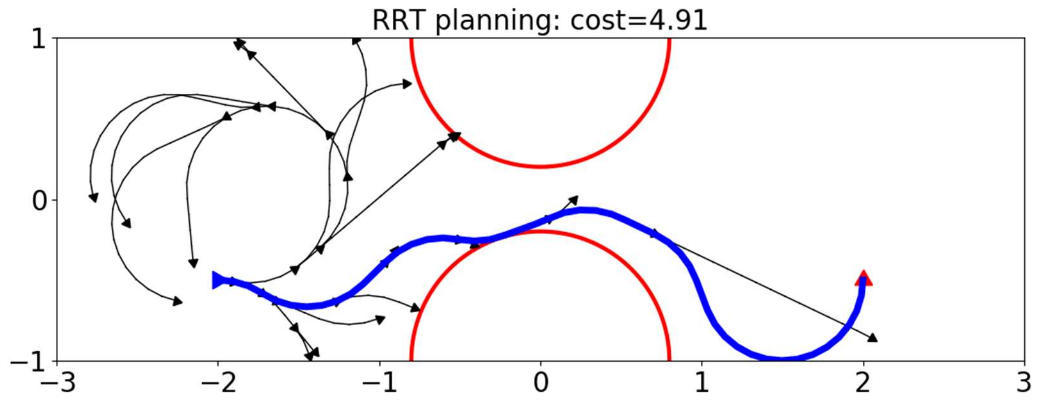
For more details and mathematical proofs, the reader is referred to the paper.

Results

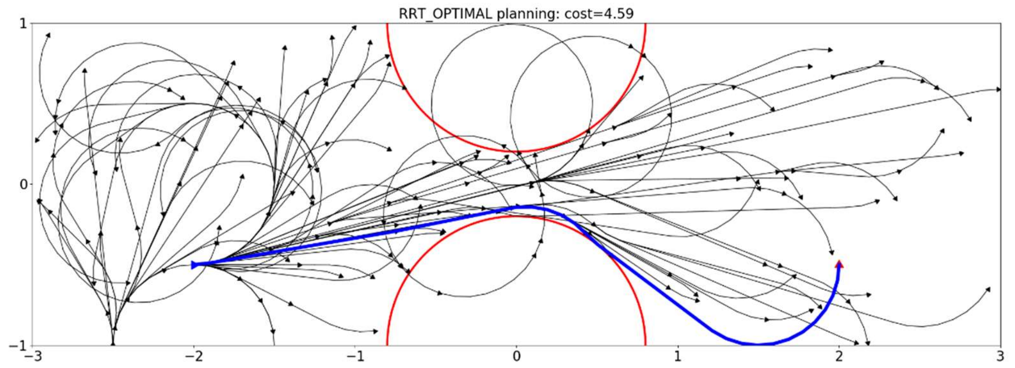
The numerical results are as follows:

PRM*	Cost				avg	min
50	12.4	19.56	13.15	15.76	15.2175	12.4
100	11.07	16.02	14.03	10.78	12.975	10.78
200	13.95	11.04	11.39	10.96	11.835	10.96
RRT*	Cost				avg	min
50	4.66	4.68	4.76	4.66	4.69	4.66
100	4.59	4.58	4.64	4.76	4.6425	4.58
200	4.61	4.6	4.63	4.59	4.6075	4.59

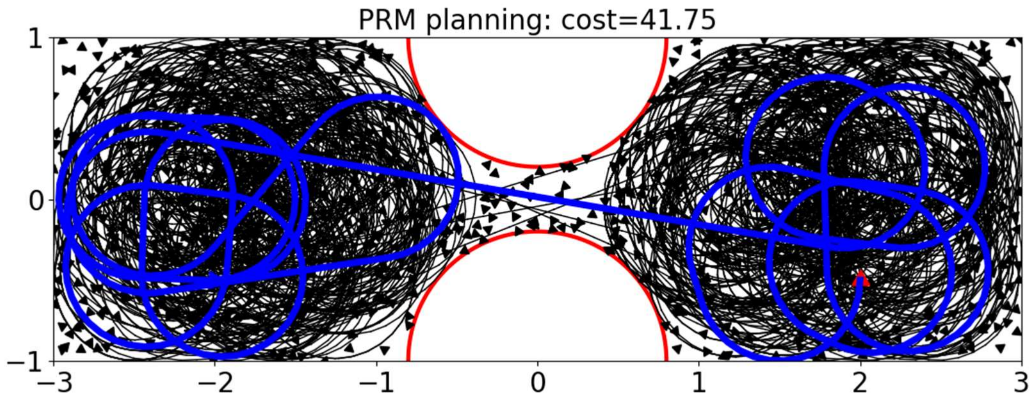
A sample solution of RRT:



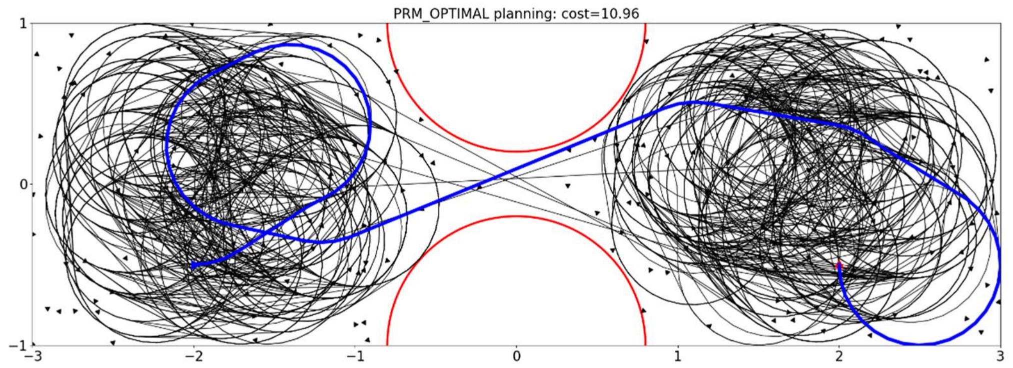
A sample solution of RRT*:



A sample solution of PRM:



A sample solution of PRM*:



Discussions and conclusions

In conclusion, k-RRT* and k-PRM* have been implemented in Python for this course project. To test their performance, characteristics, and verify the programs, they have been applied to a specific problem of sampling-based motion planning of a Dubins car with assumptions and limitations discussed in the "Introduction" section. According to the experimental results, optimal cost plans can be found in an asymptotic manner in these sampling-based algorithms, and their solutions are probably more efficient in terms of the cost function in comparable scenarios, i.e., fixed conditions such as constant number of vertices and constant pseudo-random samples of the configuration space. Comparing PRM and RRT in general, RRTs (variants of RRT including the simple RRT and k-RRT*) are more efficient and better suited for single-query problems as they can find solutions faster than their PRM counterparts, based on this project experiments, despite having the same asymptotic higher bounds on the time complexities, and having a clear advantage in terms of memory cost and query time complexity ($O(n)$ vs $O(nlog(n))$).

To further improve and complete this study, other variations of the sampling-based algorithms can be considered and compared for the specific case study of a Dubins car, for instance, RRT* and PRM* based on the radius check for finding near neighbors as proposed in the paper, as well as more complicated or realistic models of a simple car, for example, higher-order differential constraints for the acceleration or Reeds-Shepp Car model. It can be hypothesized that more efficient solutions, in terms of the average number of samples or iterations they take to find a feasible path given a fixed cost threshold, might be achieved through bi-directional trees in RRT* instead of a single tree grown from the initial or a goal configuration. Due to the limited time and scope of this project, an easy setup and model of the world, the obstacles, and the robot or car have been considered. However, provided more time and effort, one can expand this framework to more realistic problems. For instance, a polygon model for the car can be considered instead of a simple dimension-less point for the collision check, a more challenging setup of the world and obstacles is given such that the shortest path between the initial and goal configurations is more challenging and tortuous, and more efficient collision-check algorithms can be implemented by checking the discretized vertices of an edge according to van der Corput sequence instead of monotonic sequences from 0 to 1 (0 meaning the first vertex of the edge and 1 meaning the second vertex of the edge given a local path which simulate the edge in the configuration space).

References

Karaman, S., robotics, E. F.-T. international journal of, & 2011, undefined. (2011). Sampling-based algorithms for optimal motion planning. *Journals.Sagepub.Com*, *30*(7), 846–894. https://doi.org/10.1177/0278364911406761

LaValle, S. M. (2006). Planning algorithms. *Planning Algorithms*, *9780521862059*, 1–826. https://doi.org/10.1017/CBO9780511546877