IOWA STATE UNIVERSITY

Department of Computer Science

Optimal sampling-based motion planning of a Dubins car

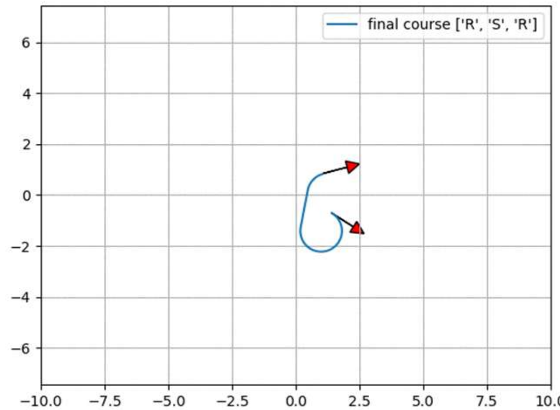
Course project

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

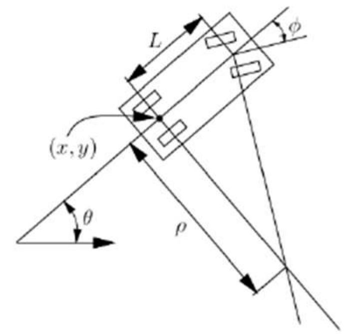
Team member: Mohammad Hashemi

05/03/2023

Problem formulation



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



- Simple car model
- Dubins conditions
- Possible path will be at least one of these six types: RSR, RSL, LSR, LSL, RLR, LRL
- World space: R^2
- Configuration space: R^2*S
- 2D action vector: (u_s, u_ϕ)
- $|u_s| \leq 1$ to neglect acceleration/dynamics
- Semi-alg obstacles (semi-circles)

Approach

- Comparison of sampling-based motion planning algorithms
- PRM vs PRM*: radius/k is a function of vertices cardinality (correlated with the sample dispersion)
- $k(n) := k_{PRM} \log(n = |V|)\,;\, k_{PRM} = 2e$

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: Processing	Time Complexity: Query	Space Complexity
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)

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

Algorithm 4: PRM*

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

Karaman & Frazzoli, 2011, International Journal of Robotics Research.

Approach (cont.)

- RRT vs PRM: better for single-query because of less memory and extension of constraints
- RRT vs RRT*: radius/k is a function of vertices cardinality; (i) create edge between new sample and tree along min cost path (ii) rewire to maintain min cost
- $k(n) := k_{RRG} \log(n = |V|)\,;\, k_{RRG} = 2e$

Algorithm 3: RRT

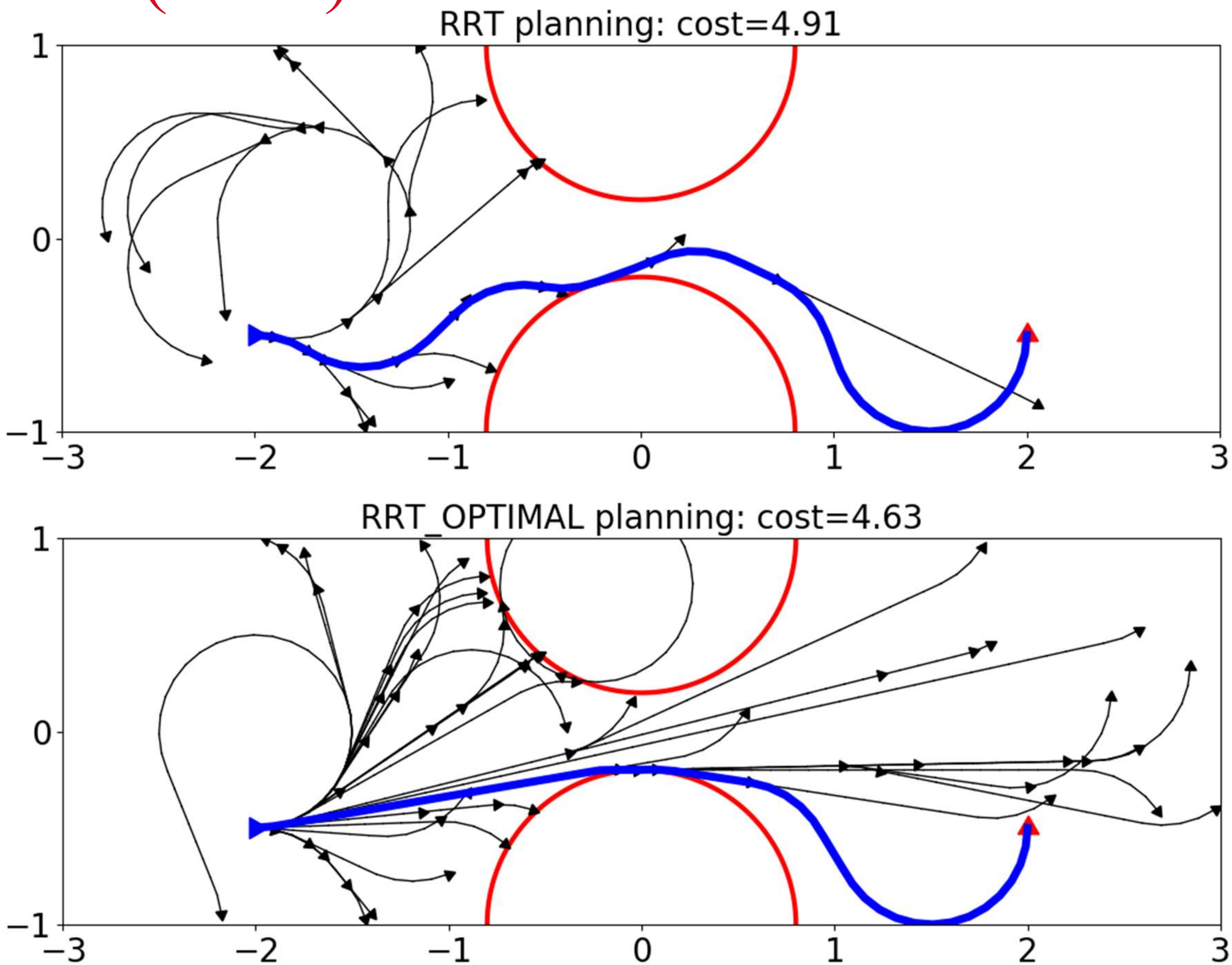
$V \leftarrow \{x_{\text{init}}\};\ E \leftarrow \emptyset;$
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow$ `SampleFree`$_i$;
  $x_{\text{nearest}} \leftarrow$ `Nearest`$(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow$ `Steer`$(x_{\text{nearest}}, x_{\text{rand}})$;
  if `ObtacleFree`$(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $V \leftarrow V \cup \{x_{\text{new}}\};\ E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}})\}$;
return $G = (V, E)$;

Algorithm 6: RRT*

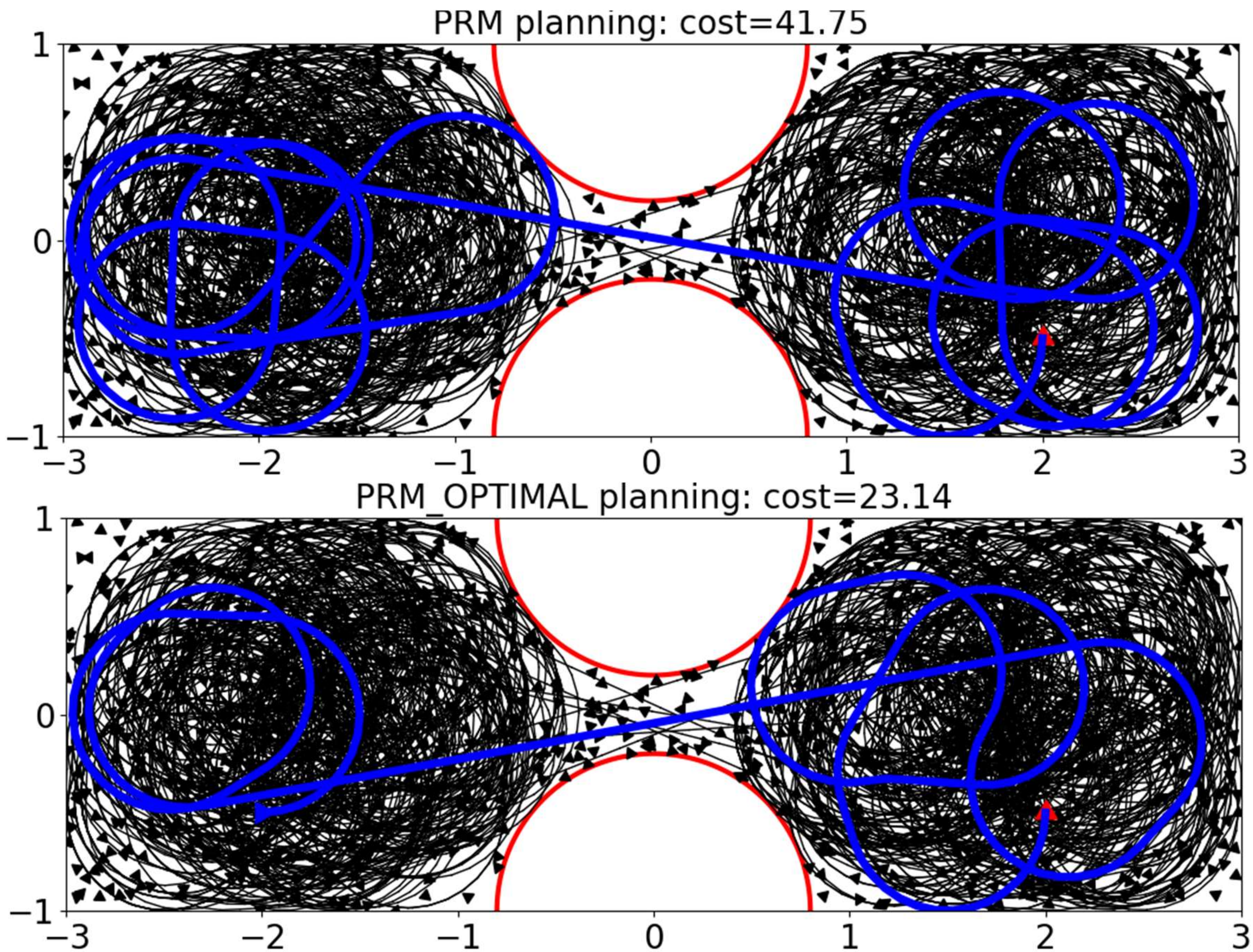
$V \leftarrow \{x_{\text{init}}\};\ E \leftarrow \emptyset;$
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow$ `SampleFree`$_i$;
  $x_{\text{nearest}} \leftarrow$ `Nearest`$(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow$ `Steer`$(x_{\text{nearest}}, x_{\text{rand}})$;
  if `ObtacleFree`$(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow$ `Near`$(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\};$
    $x_{\min} \leftarrow x_{\text{nearest}};\ c_{\min} \leftarrow$ `Cost`$(x_{\text{nearest}}) + c($`Line`$(x_{\text{nearest}}, x_{\text{new}}));$
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if `CollisionFree`$(x_{\text{near}}, x_{\text{new}}) \wedge$ `Cost`$(x_{\text{near}}) + c($`Line`$(x_{\text{near}}, x_{\text{new}})) < c_{\min}$ **then**
        $x_{\min} \leftarrow x_{\text{near}};\ c_{\min} \leftarrow$ `Cost`$(x_{\text{near}}) + c($`Line`$(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\min}, x_{\text{new}})\};$
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if `CollisionFree`$(x_{\text{new}}, x_{\text{near}}) \wedge$ `Cost`$(x_{\text{new}}) + c($`Line`$(x_{\text{new}}, x_{\text{near}})) <$ `Cost`(x_{near})
      then $x_{\text{parent}} \leftarrow$ `Parent`(x_{near});
        $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

Karaman & Frazzoli, 2011, International Journal of Robotics Research.

Results (RRT)


RRT planning: cost=4.91
RRT_OPTIMAL planning: cost=4.63
1
0
−1
−3
−2
−1
0
1
2
3

Results (PRM)



PRM planning: cost=41.75
PRM_OPTIMAL planning: cost=23.14
1
0
−1
−3
−2
−1
0
1
2
3

Conclusions

- Implemented k-RRT* and k-PRM*
- Applied the algs to a Dubins car motion planning problem
- Optimal cost plans found in the experiment
- PRM problems
- Future work/improvements:
  - radius check instead of k-nearest methods
  - bi-directional trees in RRT*
  - more accurate (polygons) and efficient (van der Corput sequence) collision check