Planning Parking Maneuvers for a Car-Trailer Vehicle

Autonomous and Mobile Robotics
Prof. Giuseppe Oriolo
Supervisors: Ph.D. Tommaso Belvedere - Ph.D. Michele Cipriano

Leandro Maglianella Lorenzo Nicoletti Olga Sorokoletova
1792507 - 1797464 - 1937430
Sapienza Università di Roma

February 9, 2022

Abstract

The overall goal of this project was to provide an Open Motion Planning Library (OMPL [3])-based framework, able to perform motion planning for a car-trailer vehicle while handling the so-called jackknifing phenomenon. Three sampling-based planners have been exploited: RRT, SST with various optimization objectives and CL-RRT, where the latter was described in the accompanying paper [1] and essentially introduces a technique to stabilize backward motion primitives. Approaches have been evaluated over a series of experiments on five different environmental set-ups, including parking maneuvers addressed in [1]. Quantitative and qualitative comparative analysis of the approaches has been performed and the achievement of the stabilization for the hitch angle divergence has been validated.

I. Introduction

The current section is devoted to the detailed description of the considered motion planning and control problem of the reversing with a trailer.

i. Car-Trailer vehicle

The given nonholonomic vehicle is not a single-body, but a **car-trailer system**, composed by the car-like tractor and a single nonzero-hooked trailer, schematically sketched in Figure 1.

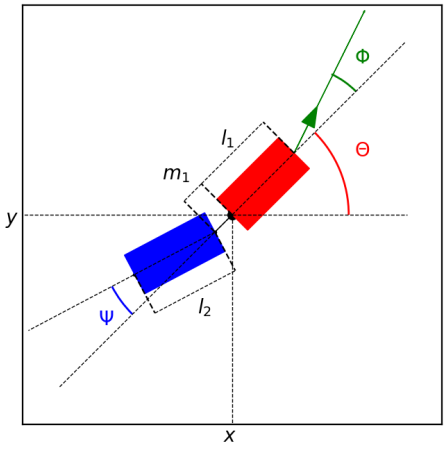


Figure 1: *The considered car-trailer vehicle.*

The configuration space of the robot is $\mathcal{C} = SE(2) \times (SO(2))^2$, and the configuration state $q \in \mathcal{C}$ is described by a vector of five generalized coordinates as:

$$q = \begin{bmatrix} x & y & \theta & \psi & \phi \end{bmatrix}^T, \tag{1}$$

with:
(x, y) – the Cartesian coordinates of the car rear-axle midpoint,
θ – the car orientation,
ϕ – the steering angle, and
ψ – the relative orientation of the trailer with respect to the car sagittal axis, which is called **hitch angle**.

The list of notations completes letting to be:
l_1, l_2 – the length of the car and the trailer, respectively, and
m_1 – the distance between the car rear axle midpoint and the hitch joint axis (since $m_1 > 0$, the nonzero-hooking can be stated).

Finally, the **kinematic model (KM)** of such vehicle is derived in the following equations:

$$\dot{x} = v \cos\theta \tag{2}$$

$$\dot{y} = v \sin\theta \tag{3}$$

$$\dot{\theta} = \frac{v \tan\phi}{l_1} \tag{4}$$

$$\dot{\psi} = -\frac{v \tan\phi}{l_1}\left(1 + \frac{m_1}{l_2}\cos\psi\right) - \frac{v \sin\psi}{l_2} \tag{5}$$

$$\dot{\phi} = \omega \tag{6}$$

where the driving velocity v and the steering velocity w represent the 2-dimensional control input.

The physical characteristics of the robot adopted for the simulations are reported in Table 1.

Parameter	Value
l_1	0.25 m
l_2	0.26 m
m_1	0.07 m
$\lvert\psi_{max}\rvert$	45°
$\lvert\phi_{max}\rvert$	30°

Table 1: *Physical characteristics of the proposed vehicle.*

ii. Instability problem

The described car-trailer system is in general unstable in **backward motion** ($v < 0$), where the so-called **«jackknifing» phenomenon**[1] issue arises: when the vehicle performs reversing (moves backwards) along the reference trajectory, **if the trajectory is not «stable»** (that is often the case, apart from some particular trajectories and initial configurations), the hitch angle ψ starts to diverge. That essentially leads to a loss of maneuverability and possibly causes a self-collision (refer to the Figure 2). Simple motion primitives will not yield stable trajectories. Therefore, backward motion for a car with the trailer, which is indeed an important part of parking maneuvers, requires a special care. To this aim, i.e. to decrease, to some extent, the hitch angle divergence issue, a CL-RRT planning approach, described in the following subsection, has been employed.

[1]The assumption we made to mathematically model this event was to bound ψ to $[-45°, +45°]$ as already mentioned in Table 1: outer hitch values will be assumed to cause jackknifing.

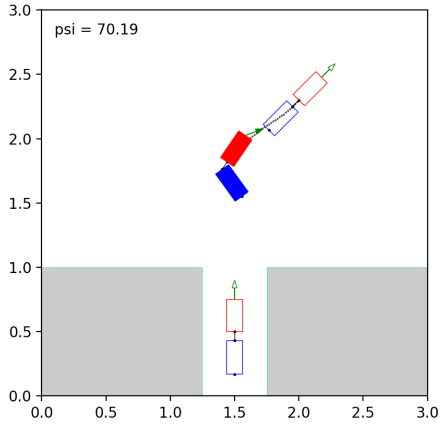


Figure 2: *Example of the situation when jackknifing occurred during reversing into the spot.*

iii. Planning techniques

Three different approaches to motion planning and control have been adapted to plan complex maneuvers in parking scenarios or challenging obstacle avoidance situations.

iii.1 Rapidly-Exploring Random Tree (RRT)

First, we concentrate on the motion planning with a plain RRT − a sampling-based probabilistically complete planner which accounts for the nonholonomic constraints by **sampling the input space and forward integrating** the system generating dynamically feasible motion primitives. In short words, RRT samples a random state q_r in the state space, then finds the state q_c among the previously visited states that is closest to q_r and expands from q_c towards q_r, until some state q_m in that direction is reached. Then q_m is added to the exploration tree.

Moreover, in the process of randomly selecting states in the state space to attempt to go towards the goal, the algorithm may in fact choose the actual goal state, if known, with some probability, which is called **goal bias**. In our approach, goal bias is used and tuned to achieve better parking accuracy (closeness of the final state to the goal state q_g) and balance the alternating processes of exploration (random q_r) and exploitation ($q_r = q_g$).

iii.2 Stable Sparse RRT (SST)

SST is an **asymptotically near-optimal** incremental modification of RRT, that **enables optimal planning**. This algorithm has been introduced in [4] as an alternative way to achieve the desirable property of asymptotic optimality for dynamic systems. SST has been proved by the authors to quickly converge to high-quality paths, while maintaining only a sparse set of sampled states, which provides computational efficiency.

From the algorithmic point of view, SST makes use of the best near selection process and applies a pruning operation to keep the number of stored nodes small. Therefore, **SST has reduced per iteration cost relative to the sub-optimal RRT** given the pruning operation, which accelerates search for the nearest neighbors.

At high level, SST follows the abstract framework, expressed in pseudo-code in algorithm 1.

For N iterations, a **selection/propagation/pruning procedure** is followed.

The selection follows the principle of the **best-first strategy** to return an existing node on the tree, $x_{selected}$. In this context, best-first means that the node $x_{selected}$ should be chosen so that the method prioritizes nodes that correspond to good quality paths, while also balancing optimization objectives. For instance, one way to achieve this in an RRT-like fashion i.e., first sample a random state x_{random} and then among all the nodes on the tree within a certain radius δ_{BN}, select the one that has the best path cost from the root.

Then *MonteCarlo_Prop* is called, which samples a random control and a random duration and then forward integrates the system dynamics.

The pruning operation should maintain nodes that locally correspond to good paths. For instance, it is possible to evaluate whether a node has the best cost in a local vicinity and prune neighbors with worse cost as long as they do not have children with good path costs in their **local neighborhood**[2]. Nodes with high path cost in a local neighborhood do not need to be considered again for propagation. Therefore, if the path $\overline{x_{selected} \rightarrow x_{new}}$ is collision-free, the new node x_{new} is evaluated on whether it is the best node in terms of path cost in a local neighborhood. If x_{new} is indeed the best, it is added to the tree and any previous node in the same local vicinity that is dominated, is pruned.

The new aspects of the approach are the following:

1. SST requires an additional input parameter δ_{BN}, used in the selection process of the *Best_First_Selection_SST* procedure;
2. SST requires an additional input parameter δ_s, used to evaluate whether a newly generated node x_{new} has locally the best path cost in the *Is_Node_Locally_the_Best_SST* procedure;
3. SST splits the nodes of the tree V into two subsets: $\mathbb{V}_{active}$ and $\mathbb{V}_{inactive}$. The nodes in $\mathbb{V}_{active}$ correspond to nodes that in a local neighborhood have the best path cost from the root. The nodes $\mathbb{V}_{inactive}$ correspond to dominated nodes in terms of path cost but have children with good path cost in their local neighborhoods and for this reason are maintained on the tree for connectivity purposes;
4. In order to define local neighborhoods, SST uses an auxiliary set of states, called "witnesses" and denoted as S. The approach maintains the following invariant with respect to S: for every witness s kept in S, a single node in the tree will represent that witness (stored in the field s.*rep* of the corresponding witness), and that node will have the best path cost from the root within a δ_s distance of the witness s.

Algorithm 1: STABLE_SPARSE_RRT ($\mathbb{X}$, $\mathbb{U}$, x_0, T_{prop}, N, δ_{BN}, δ_s)

$\mathbb{V}_{active} \leftarrow \{x_0\}, \mathbb{V}_{inactive} \leftarrow \emptyset$;
$G = \{V \leftarrow (\mathbb{V}_{active} \cup \mathbb{V}_{inactive}), \mathbb{E} \leftarrow \emptyset\}$;
$s_0 \leftarrow x_0, s_0.rep = x_0, S \leftarrow \{s_0\}$;
for N *iterations* **do**
  $x_{selected} \leftarrow$ *Best_First_Selection_SST*$(\mathbb{X}, \mathbb{V}_{active}, \delta_{BN})$;
  $x_{new} \leftarrow$ *MonteCarlo_Prop*$(x_{selected}, \mathbb{U}, T_{prop})$;
  if *CollisionFree*$(\overline{x_{selected} \rightarrow x_{new}})$ **then**
    if *Is_Node_Locally_the_Best_SST*(x_{new}, S, δ_s) **then**
      $\mathbb{V}_{active} \leftarrow \mathbb{V}_{active} \cup \{x_{new}\}$;
      $\mathbb{E} \leftarrow \mathbb{E} \cup \overline{x_{selected} \rightarrow x_{new}}$;
      Prune_Dominated_Nodes_SST$(x_{new}, \mathbb{V}_{active}, \mathbb{V}_{inactive}, \mathbb{E})$;
return G;

So far with a plain RRT, the so-called problem of regular motion planning has been addressed. There exists an alternative way to state the problem, which is known as **optimal motion planning problem**. Defining an optimal motion planning problem is almost exactly the same as defining a regular motion planning problem, with two main differences: the possibility to specify an optimization objective and the need to use an optimizing planner.

SST has been mainly used in our framework with one of the following two optimization objectives (or their weighted combinations):

[2]There are many different ways to define local neighborhoods. For instance, a grid-based discretization of the space or incremental approach of defining visited regions of the state space could be defined.

1. **Path Length Minimization,** is defined as an objective which attempts to optimize the length and avoid paths containing excessive (not useful to reach the goal state) motions, but it can cause the robot to steer very close to obstacles, which can sometimes be unsafe. The mathematical formulation of the path length cost function is straightforward since it is based on the definition of the Euclidean distance[3] and is defined as a sum of the costs of the vertices along the path. Given in fact a branch of the search tree that connects the root to a vertex v, the cost of v is represented as:

$$c(v) = c(v.parent) + \|v - v.parent\| \tag{7}$$

2. **Path Clearance Maximization,** is defined as an objective which attempts to steer the robot away from obstacles to efficiently increase safety. The path clearance measure is defined as a summation of state costs along the path, where each state cost is a function of the state distance from the obstacles. In particular, defining the configuration state space $\mathcal{C}$ and its subset of collision-free states known as *free configuration space* $\mathcal{C}_{free}$, the clearance cost function for a given state q can be defined as:

$$\gamma(q) = \min_{s \in \partial C_{free}} \|q - s\| \tag{8}$$

where $\partial\mathcal{C}_{free}$ denotes the boundaries of $\mathcal{C}_{free}$ and consequently implies the presence of nearby obstacles. Therefore, given a vertex v along the path and the associated configuration q in $\mathcal{C}$, the cumulative cost function for v can be defined as:

$$c(v) = c(v.parent) + \frac{1}{\gamma(q)} \tag{9}$$

Finally, the optimization problem in both cases can be simply summarized as:

$$\min_{v \in SearchTree} c(v). \tag{10}$$

Note that, even though we have defined the clearance objective as a maximization, the criterion can be easily converted to a minimization by considering the inverse of the clearance as we did in Equation 9.

iii.3 Closed-Loop RRT (CL-RRT)

Applying the original RRT-framework to unstable system models, such as the considered car-trailer (when driving forward the system is stable but **in reverse ($v < 0$) the system becomes unstable**) faces aforementioned instability problem. To overcome this problem and **account for the off-axle hitch angle not entering the jack-knife state** when reversing as in Figure 2, CL-RRT has been developed. The novelty of CL-RRT framework consists in the introduction of a closed-loop dynamics, where after a transformation of the control input, **instead of sampling the steering velocity** ω as before, the sampled steering primitive becomes a desired **steering angle** ϕ_d. This new input is used to compute the reference angle ϕ_r, defined as:

$$\phi_r = \phi_d - K_{stab}\psi, \tag{11}$$

with $K_{stab} \geq 0$ ($K_{stab} = 0$ when sampled $v > 0$) – gain for the «stabilising» control action, i.e weighting parameter for the dependence of the referenced steering angle on the hitch angle, that stands for amount of **steering needed to counter for the divergence of the hitch angle**.

[3] Assuming two consecutive nodes in the path belong to the same small neighborhood in the configuration space. If the assumption does not hold, the usage of the Euclidean distance is not topologically correct for high-dimensional state spaces since it is a reasonable approximation only for «local» motions. In our case, the assumption is empirically found to be valid since no inconsistencies have been detected during the experiments.

Consequently, the steering velocity is altered in order to track such reference quantity by imposing:

$$\dot{\phi} = \omega = K_{reg}(\phi_r - \phi) \tag{12}$$

with $K_{reg} > 0$ – gain for the **regulation of the reference steering angle** by means of the steering velocity, i.e. weighting parameter for the simple integrator.

The cascaded control, introduced in Equation 11 and Equation 12, will guarantee **exponential convergence** of ϕ to ϕ_r (assuming a constant or slowly-varying ϕ_r, which implies a zero or a very small feedforward term $\dot{\phi}_r$).

II. Implementation

As already mentioned, the project is entirely based on OMPL [3]. It is a motion planning software which contains implementation of the state-of-the-art sampling-based algorithms. However, its advantage of being integrable into large variety of external systems comes at cost of a limitation: OMPL by intentional choice of the developers is able to perform motion planning exclusively (no additional components), which in particular means that **environment specification, collision detection and visualization are left to the user**. The library is written in the `C++` programming language and also offers `Python` bindings.

This section is devoted to a high-level explanation of our design choices and implementation using OMPL-functionality.

i. Creating the car-trailer with its state and control spaces

OMPL already provides simple state spaces (for instance, $\mathbb{R}^n$, $SO(2)$, $SO(3)$, $SE(2)$ and $SE(3)$), therefore, our custom **configuration space** of Equation 1 was easily created **by compounding** simpler sub-spaces together.

Similarly, the control inputs were defined as the **composition of two discrete control spaces**: one for each of the inputs (v and ω – for RRT and SST, or v and ϕ_d for CL-RRT). The discretization allowed us to smartly instantiate the set of motion primitives to be used as controls in the KM of Equation 2-Equation 6. This set was adapted for the type of scenario/experiment and will be further explained in the following subsections.

Finally, regarding the forward integration process, we started adopting the simple discrete-time Euler method; then, due to its unavoidable integration error, the decision to change it in favor of a more reliable and accurate **Ordinary Differential Equation (ODE)** solver (again directly provided by OMPL) has been taken. That allowed us to achieve realistic and at the same time precise solutions which will be discussed and analyzed in details in the next section, devoted to the description of the experiments.

ii. Setting up the planner with state validity and collision checking

One of the aforementioned limitations of OMPL was the unavailability of a pre-implemented **collision-checking** routine. Therefore, this component should be customized. In our approach, we define a discrete number (16) of representative **control points** along the robot body as shown in Figure 3. Consequently, the collision detector performs an iterative check for an obstacle-hit over the control set and discards invalid states while searching for a solution path[4].

An analogous design choice was made for the **collision-with-obstacles checking**: for the case of a non-free environment, a set of control points is defined not only on the vehicle but also inside the environment on the obstacles boundaries in correspondence with the nature and specific characteristics of the given environment. The state validity checking role then simply becomes an

[4]The proposed discretization of a collision-checking mechanism has been tested within a wide range of experiments and assumed to be correctly working since no malfunctioning was noticed (i.e. no invalid states were added to the search tree during its execution).

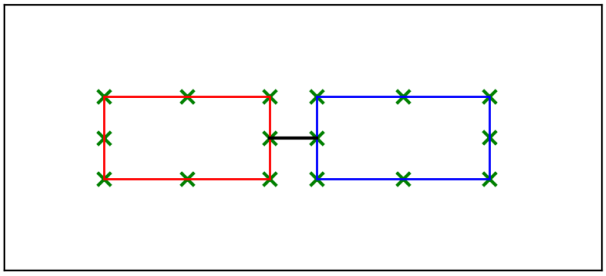
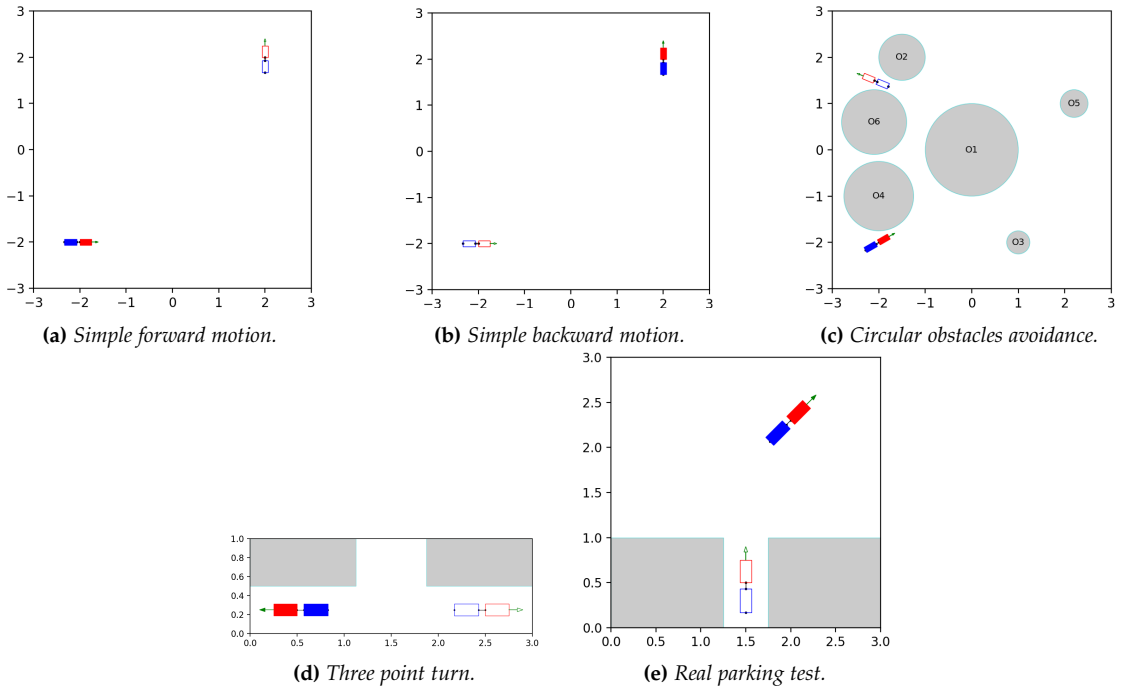


Figure 3: *Sketch of the highlighted control points along the perimeter of the car-trailer multi-body.*

evaluation of the minimum distance between the two closest control points: of the car-trailer and of the obstacle.

It is important to notice that the same reasoning is inherited by the path clearance maximization objective in SST.

iii. Defining the scenarios to test the maneuverability of the robot



(a) *Simple forward motion.* **(b)** *Simple backward motion.* **(c)** *Circular obstacles avoidance.*

(d) *Three point turn.* **(e)** *Real parking test.*

Figure 4: *Scenarios for our experiments: the «filled» and «empty» car-trailer robots in each environment represent the start and the goal states, respectively.*

Five different simulation scenarios (Figure 4) have been created for the experiments following an «increasing-difficulty» criterion in terms of available free space and maneuver complexity:

a **Simple forward motion in free space**, a square area with no obstacles or boundaries, where the maneuver should be mainly done moving forward;

b **Simple backward motion in free space**, a square area with no obstacles or boundaries, where the maneuver should be mainly done moving backward;

c **Circular obstacles avoidance**, the first scenario of an obstacles-populated environment: circular obstacles of a different location and radius;

d **Three point turn**, inspired by the [1], it is a motion in a narrow environment where an orientation inversion in a gap is required.

e **Real parking test**, inspired by the [1], it is a motion in a large environment with a very narrow parking gap.

The collision-checking routine, introduced in the previous subsection, for the case of scenarios in Figure 4c-Figure 4e uses the following finite sets of environmental control points: for circular-shaped obstructions, the points coincide with the centers of the circles (measured distance is translated by the radius), while for the other two cases with walls, we picked up sufficiently close-located points along the wall boundary in order to monitor possible collisions.

III. Experiments

The detailed information about the **specifics of the aforementioned scenarios and parameters** that were set-up for each of them, along with the **results** collected over multiple runs of the implemented software can be found in the current section. These results are expressed in terms of various **quantitative and qualitative metrics**, intended to provide a fair comparison of algorithmic properties between different planners and detect the best one of them capable of performing planning of the complicated car-trailer parking maneuvers.

Environment	Boundaries		q_s					q_g				
	$[x_l, x_h]$	$[y_l, y_h]$	x	y	θ	ψ	ϕ	x	y	θ	ψ	ϕ
Simple Forward	$[-3,3]$	$[-3,3]$	-2	-2	0	0	0	2	2	$\frac{\pi}{2}$	0	0
Simple Backward	$[-3,3]$	$[-3,3]$	2	2	$\frac{\pi}{2}$	0	0	-2	-2	0	0	0
Circular Obstacles	$[-3,3]$	$[-3,3]$	-2	-2	$\frac{\pi}{6}$	0	0	-2.1	1.5	$\frac{7\pi}{8}$	0	0
Three-point Turn	$[0,3]$	$[0,1]$	0.5	0.25	$-\pi$	0	0	2.5	0.25	0	0	0
Real Parking	$[0,3]$	$[0,3]$	2	2.3	$\frac{\pi}{4}$	0	0	1.5	0.5	$\frac{\pi}{2}$	0	0

Table 2: ***Environmental parameters:*** *state space bounds for the Cartesian components, initial and goal configuration.*

Environment	Planner	v		ω / ϕ_d		ϵ	Bias	Step	Dur.	Time
		min	max	min	max					
S. Forward	**RRT**	-0.25	0.5	-1	1	0.2	0.3	0.1	$(1,10)$	45
	SST									
S. Backward	**RRT**	-0.5	0.25	-1	1	0.3				
	SST									
Circular O-s	**SST Min**	-0.5	0.5	-1	1	0.25				
	SST Max									
3-point Turn	**RRT**	-0.5	0.5	-1	1	0.25				
	SST			-1	1					
	CL-RRT			-25	25					
Real Parking	**RRT**	-0.5	0.5	-1	1	0.25				
	SST			-1	1					
	CL-RRT			-25	25					

Table 3: ***Hyper-parameters:*** *control inputs (driving velocity v $[m/s]$ and steering velocity w $[rad/s]$ or desired steering angle ϕ_d $[^\circ]$), goal threshold ϵ, goal bias, propagation step size, control duration and maximum time limit to return an exact solution. SST Min and SST Max stand for SST with Minimum Path Length objective and SST with Maximum Path Clearance objective, respectively.*

The **environmental parameters**, containing Cartesian boundaries together with the start and goal configurations, are reported in Table 2. The set of the **best hyper-parameters** has been

achieved through the procedure of **repetitive fine-tuning** and are collected in Table 3. Each planner inside each simulated environment has been executed 50 times through the usage of benchmarks. Note that some of the hyper-parameters (e.g. control duration, propagation step and time limit) were eventually fixed at value, common for all the five scenarios, meanwhile some others (e.g. goal threshold) were further fine-tuned to account for the peculiarities of the scenario.

Moreover, we will assume that if no optimization objective is clearly specified for SST, path length minimization will be the default criterion to be optimized.

Now, with the given information about the experimental set-ups, we can proceed to the detailed exploration of the results.

i. Simple forward motion

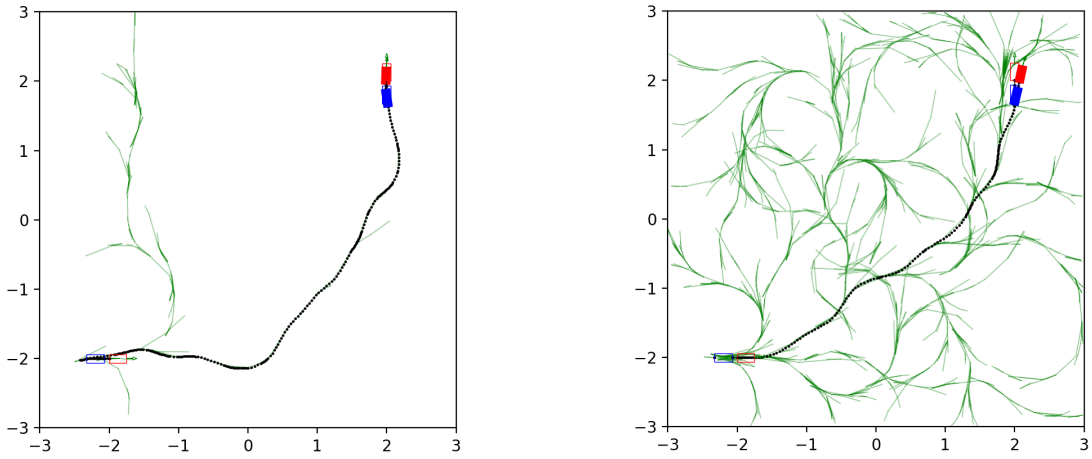
The first and simplest scenario is a *Simple forward motion* in free space and was introduced as a preliminary step before modeling parking manoeuvres to study some basic properties of two initial planners (RRT and SST) and detect the differences between them.

The quantitative results, aggregated over 50 independent runs of each planning method, are collected in Table 4 and plotted in Figure 10. From them, it can be concluded that, in average, accuracy of the RRT algorithm is similar to the one of SST as they are capable to find approximately **the same number of exact solutions**, and found solutions correspond to **the same level of complexity** in terms of their length. However, **RRT** is executed almost two times **faster**, meanwhile **SST requires less memory** for the storage of states.

Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	**58**	23948	84691	49460	24.9	26.7	25.7	**8.89**
SST	54	9456	25962	**17049**	23.6	24.8	**24.2**	14.51

Table 4: ***Simple forward motion quantitative evaluation****: average number of states, solution length and execution time for exact and approximate solutions (and their weighted average). The % column indicates the success rate.*

From a qualitative point of view, the **hypothesis about similarity of the RRT and SST is again confirmed** by looking at the sampled example of found paths and trees in Figure 5, although **SST** could be preferable, since it typically **outputs smoother path**. Note that, even though SST tree looks denser and RRT seems to be closer to the goal state, these results are not statistically meaningful and correspond to just a particular sampled example.



(a) *RRT* **(b)** *SST*

Figure 5: ***Solution samples for the Simple forward motion scenario with different planning algorithms.***

In general, such **similarities** in the performances between these two planners **can be explained by the simplicity** of the given scenario.

ii. Simple backward motion

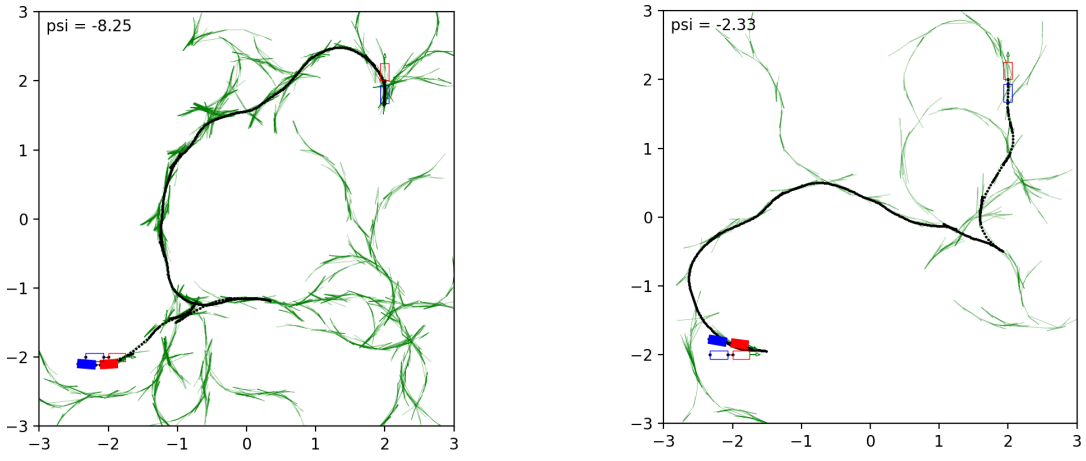
The next scenario is a *Simple backward motion*. It differs from the previous one by simply switching the start and the goal configurations and shifting controls in order to sample negative driving primitives with a higher probability than the positive ones, while the environment is still free from obstacles. The aim is to challenge the system's ability to handle the instability problem.

Aggregated over 50 runs, quantitative results are collected in Table 5 and the relative information about distributions is illustrated in Figure 11. The **success rate** (probability to find an exact solution) is again similar for RRT and SST, but noticeably **lower than in the forward case** which is conditioned by the task complexity increase. Similarly to the forward scenario, SST is characterized by a lower number of nodes thanks to its pruning operation and the correspondent distributions are analogous to the ones analyzed in the previous experiment. However, the **difference in execution times is now not so conspicuous** as before. Moreover, even without relevant differences between *Simple forward* and *Simple backward motion*, **the average path length is doubled** in the latter scenario, which is again a reasonable change due to the higher difficulty of the requested backward maneuver.

Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	**46**	31096	60313	46873	57.1	57.7	57.4	**16.93**
SST	44	8062	14662	**11758**	54.0	57.0	**55.7**	20.46

Table 5: ***Simple backward motion quantitative evaluation****: average number of states, solution length and execution time for exact and approximate solutions (and their weighted average). The % column indicates the success rate.*

Interesting details can be noticed from the sampled results compared in Figure 6: in both RRT and SST cases, the paths have a «switch point» at which the **car-trailer changes direction of motion**, alternating between forward and backward sub-motions. A monitoring of the hitch angle during these two sampled runs allows to detect that **at this point the risk of entering in a jackknifing situation** is high (Figure 15). Apart from that, SST path curve is again smoother and represents a more natural motion.



(a) *RRT* **(b)** *SST*

Figure 6: ***Solution samples for the Simple backward motion scenario with different planning algorithms.***

Overall, since neither RRT nor SST gives a significant growth or fall in performance with respect to its opponent, it cannot be detected which planner would be better for a particular reversing maneuver. Nevertheless, the important observation about the necessity to monitor and to control the hitch angle divergence has been made exactly at this stage.

iii. Circular obstacles avoidance

Leaving aside for a moment the challenge of stabilization while moving backward, in the *Circular obstacles avoidance* experiment we started introducing obstacles.

The given **environment is populated with** N **obstacles** of circular shape with different radius and centered at random positions in the workspace. To collect the information about the statistics (as usually, over 50 runs of each planner), 5 obstacles (assumed to be known), described in the Table 6 and plotted in Figure 4c, have been fixed in the environment.

Radius	**Center**
1	$(0.0, 0.0)$
0.5	$(-1.5, 2.0)$
0.25	$(1.0, -2.0)$
0.75	$(-2.0, -1.0)$
0.3	$(2.2, 1.0)$
0.7	$(-2.1, 0.6)$

Table 6: *Geometry of circular obstacles in a Obstacles avoidance scenario.*

Since the main goal for this obstacles-populated environment is to see how different optimization objectives can be exploited, the sub-optimal RRT is no more usable.

The first planner we tested in the proposed scenario is SST combined again with the path length minimization criterion. Quantitative results are reported and pictured in Table 7 and in Figure 12 respectively.

Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
SST Length	92	4301	15663	5210	34.3	47.5	35.4	9.30

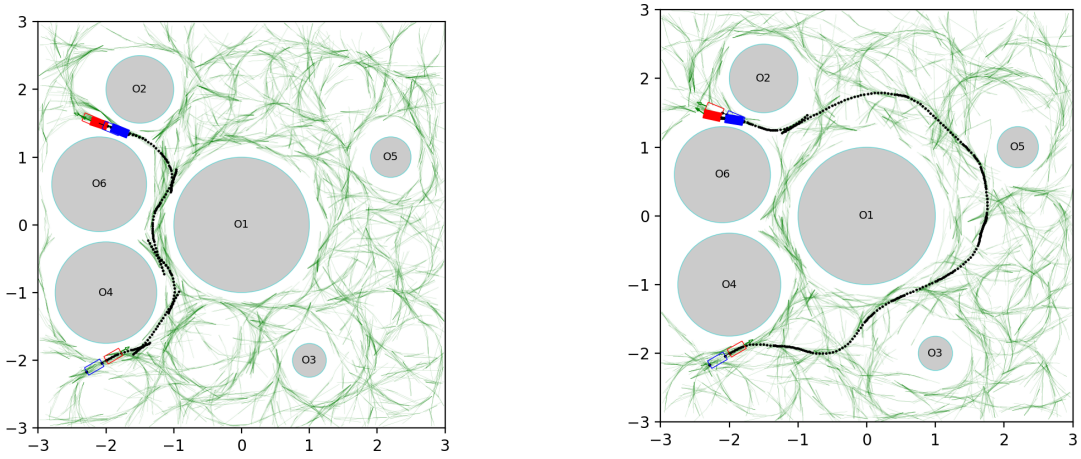
Table 7: ***Circular obstacles avoidance quantitative evaluation****: average number of states, solution length and execution time for exact and approximate solutions (and their weighted average). The % column indicates the success rate.*

Even if the introduction of obstacles limits the number of valid states in the search tree, the incredibly high success rate of this algorithm testifies how **the returned solution is almost always exact**. Moreover, the number of states, the average length and the execution time to complete the task are kept very low if compared with the analogous results of previous experiments. The correspondent distributions highlight the fact that, except for some outliers due to the randomness in the tree expansion of SST, the performances of the algorithm are very efficient in terms of memory consumption and required time (Figure 12a-Figure 12c).

The important novelty of the scenario regards the possibility to choose a different optimization objective in addition to the already discussed path length minimization. Given a non-free environment, in fact, a reasonable approach is to perform motion planning while optimizing the criterion that will keep the robot away from obstacles. Then, the task can be replaced with an optimal motion planning where **path clearance** (instead of path length) objective serves as the criterion to optimize.

«Qualitative» plots of sampled solutions (Figure 7) are indeed informative. It can be clearly verified, that **optimization of the desired criterion is achieved in both cases**: in the Path Length minimization case, the found solution leads typically through the same corridor as in Figure 7a,

while in the Path Clearance maximization case, typically through the longer route, but larger in distance from obstacles, as shown in Figure 7b.



(a) *SST Path Length* **(b)** *SST Path Clearance*

Figure 7: ***Solution samples for the Circular obstacles avoidance scenario with different optimization objectives.***

Since both objectives are extremely useful to obtain short but also safe solution paths, a **mixed** optimization objective has been used to **combine the advantages** of the two optimizing routines. In the next and final experiments, we will assume to adopt this compound criterion for SST.

iv. Three point turn

Three point turn is one of the two environments inspired by the [1] and is the first scenario proposing a realistic parking maneuver for the car-trailer. We tested our three planning algorithms – RRT, SST and CL-RRT – again over a total number of 50 runs for each. The related results are shown in Figure 13 and Table 8.

Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	76	19865	53277	27884	22.8	23.1	22.9	14.70
SST	26	777	1380	**1223**	18.1	14.4	**15.4**	20.56
CL-RRT	**92**	13084	56093	16524	24.4	21.5	24.2	**8.87**

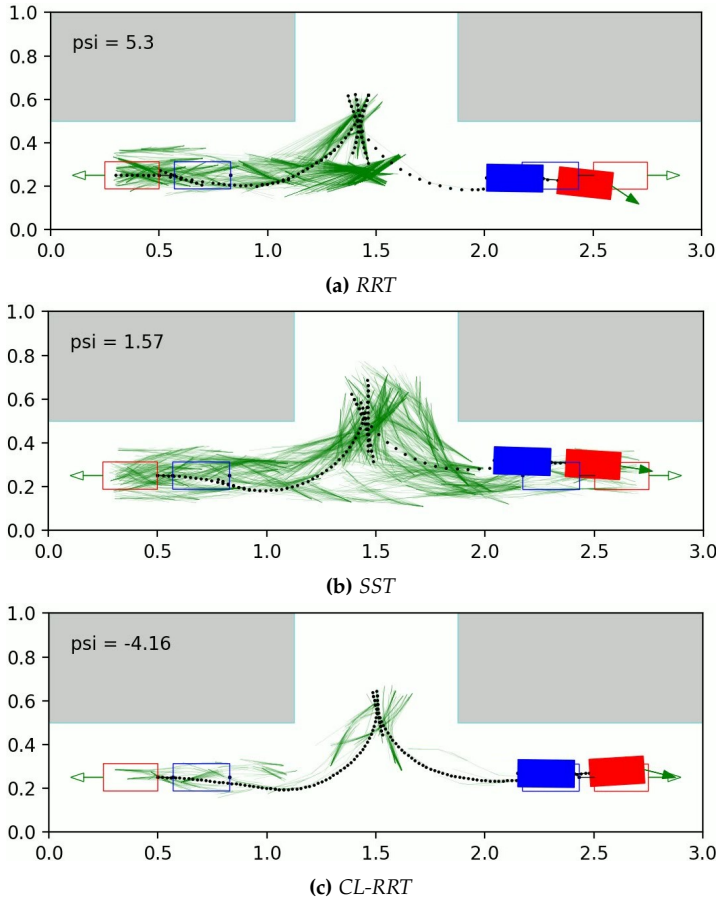
Table 8: ***Three point turn quantitative evaluation:*** *average number of states, solution length and execution time for exact and approximate solutions (and their weighted average). The % column indicates the success rate.*

Even though SST registered the best performances in terms of low number of needed states to reach the goal and short path length, the poor 26% of success rate makes this algorithm consistently unreliable for the considered type of scenario. On the other hand, RRT reported overall medium-level performances, while CL-RRT has been certified as the best performing planning algorithm with the 92% of probability to find an exact solution combined with the lowest execution time of 8.87s, which is almost half of the time spent by the other two planners.

Moreover, inspecting Figure 13, it could be noticed that the number of states in the search trees, path lengths and execution time were uniformly distributed in their intervals for RRT and CL-RRT, while SST registered a peak in correspondence to its mean.

From a qualitative point of view, some sampled results are shown in the Figure 8.

The peculiar tree structure makes this scenario even more interesting than the previous ones. Assuming that the maneuver can be split into a two-phase motion composed by a backward and a



(a) *RRT*

(b) *SST*

(c) *CL-RRT*

Figure 8: ***Solution samples for the Three point turn scenario with different planning algorithms.***

forward half, the tree expansion can be considered separated as well. RRT is characterized by a dense graph during the first part of the maneuver and a shallow directed-to-the-goal expansion during the second part, SST has large branches of the tree for both parts of the maneuver and CL-RRT efficiently reduces the number of unnecessary motions.

A final remark for the *Three point turn* experiment is related to the jackknifing risk in the backward maneuver, represented as a heatmap of the measured values for the hitch angle ψ in the Figure 16. The bright (yellow) bars on the right side of the image indicate a closeness of getting into a jack-knife state, in particular, when the car-trailer performs a right turn (highlighted interval at 40°-45°), i.e. when entering the gap. Nevertheless, given the nature of the environment, this unsafe motion is almost unavoidable in order to approach to the goal. On the other hand, all the planners have also registered a good capability to recover the safe state during the second half of the motion and efficiently reach the requested goal configuration.

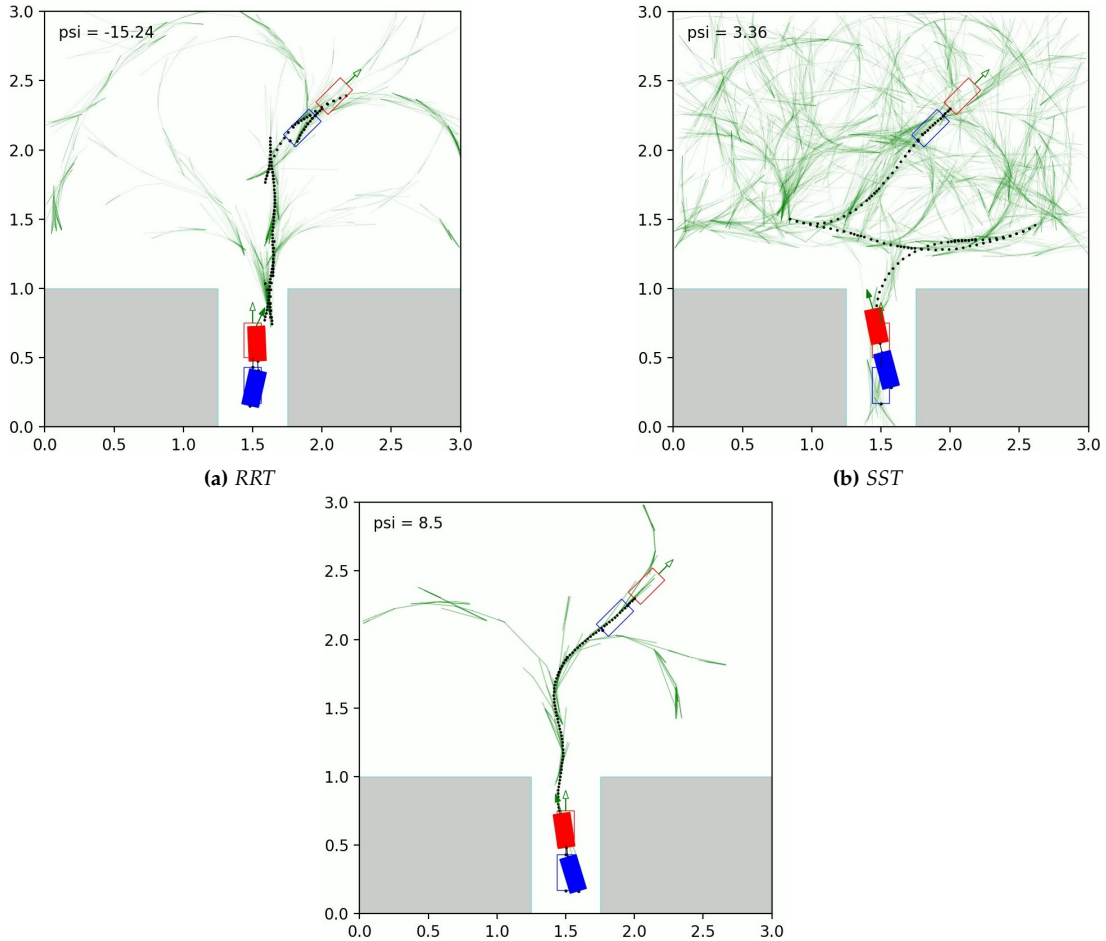
v. Real parking test

Our last proposed scenario is maybe the most difficult environment because of the narrowness of the parking area that the car-trailer should reach in a backward and, more importantly, safe motion. The quantitative evaluation can be performed through the data collected in Table 9, which testify all the issues the planning algorithms faced. The performances for plain RRT and SST are indeed extremely poor and incomparable with the relatively high 82% of accuracy reported by CL-RRT: since *Real parking test* requested an almost only-backward maneuver, the stabilization introduced by the Closed-Loop was the determinant factor to achieve reliable solution paths. Moreover, its

lowest average time needed to complete the task (10.97s) better testified the trustworthiness of the algorithm. On the other hand, the drawback of such achievements is given by the high cost in terms of memory to store the state nodes in the search graph; SST is, in fact, the only planner able to keep the number of explored states low.

Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	30	25261	61613	50707	23.2	22.3	22.6	11.43
SST	36	4173	11036	**8565**	20.8	18.7	**19.5**	14.12
CL-RRT	**82**	23266	65609	30888	26.5	31.9	27.5	**10.97**

Table 9: ***Real parking test quantitative evaluation:*** *average number of states, solution length and execution time for exact and approximate solutions (and their weighted average). The % column indicates the success rate.*



(a) *RRT* **(b)** *SST* **(c)** *CL-RRT*

Figure 9: ***Solution samples for the Real parking test scenario with different planning algorithms.***

The above results are repeated and validated by the histogram plots grouped in Figure 14 where the distribution of number of states, path lengths and execution times are graphically explained: SST registered the best performances in the first two metrics, while CL-RRT confirmed the lowest required amount of time in the last third image on the right. Overally, the latter is easily certified as the best performing algorithm in this scenario because of its high efficiency combined with the low time interval required to complete the maneuver.

The previous conclusions can be rearranged in a qualitative point of view as well, by considering the solution samples of Figure 9. The aforementioned objectives for this experiment were related not only to the smoothness of the path but also to the motion safety in terms of maximum clearance to the gap borders. Consequently, it is clear how RRT (Figure 9a) preferred to quickly reach the goal state even though its maneuver was close to the right wall of the parking area, SST (Figure 9b) aimed at optimizing the clearance when entering into the narrow corridor but with the addition of unnecessary preliminary motions, while CL-RRT (Figure 9c) represented the most promising compromise between the other two planners: a smooth solution path together with a confident and safe trajectory in the proximity of the boundaries.

The related jackknifing risk heatmaps presented in Figure 17 are self-explanatory: RRT implied the highest danger given that its interval of hitch angles registered values close to the feasible margins of $\pm 45°$, while SST and (even better) CL-RRT carefully controlled the risk of jackknifing by limiting ψ in safe ranges of values.

IV. Conclusion

The main goal of the project was to provide reliable planning algorithms to be exploited in unstable backward maneuvers and integrated in the OMPL package. Our approach composed by three different planners was tested and evaluated in a wide set of experiments and scenarios in order to fully explore the potentialities of our implementation, by highlighting its strengths and limitations. The jackknifing phenomenon was efficiently handled in increasing-complexity environments and the peculiarities of each control planning algorithm were exhaustively analyzed, while proceeding in the dissertation. As expected, the most satisfactory results were related to the introduction of the Closed-Loop in the dynamic system, a regulation routine to stabilize the diverging hitch angle. All the experiments were commented with accompanying plots and tables in a quantitative and qualitative way. Finally, the proposed results were studied and certified as robust and reliable achievements in terms of the adopted evaluation metrics and the overall outcomes of the project were coherent with our introductory objectives.

References

[1] *Evestedt N., Ljungqvist O., Axehill D.*

"Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT", *IROS 2016*

[2] *Beglini M., Lanari L., Oriolo G.*

"Anti-Jackknifing Control of Tractor-Trailer Vehicles", *2020 IEEE International Conference on Robotics and Automation (ICRA)*

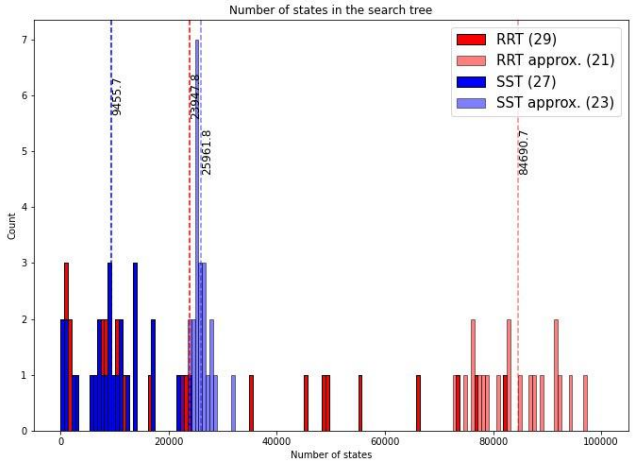
[3] Open Motion Planning Library. *http://ompl.kavrakilab.org/*

[4] *Li Y.,Littlefield Z., Bekris K. E.*

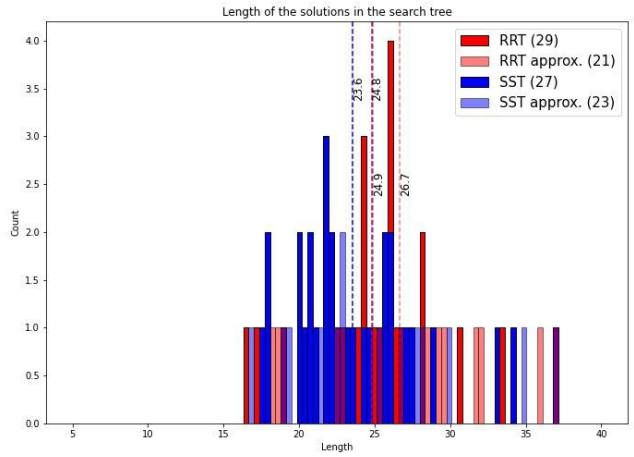
"Asymptotically Optimal Sampling-based Kinodynamic Planning", *2016*

Appendices

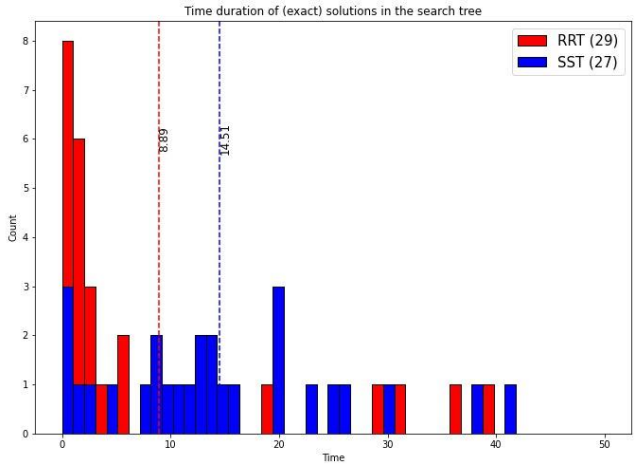
A. Experiments: Quantitative Plots/Statistics



(a)

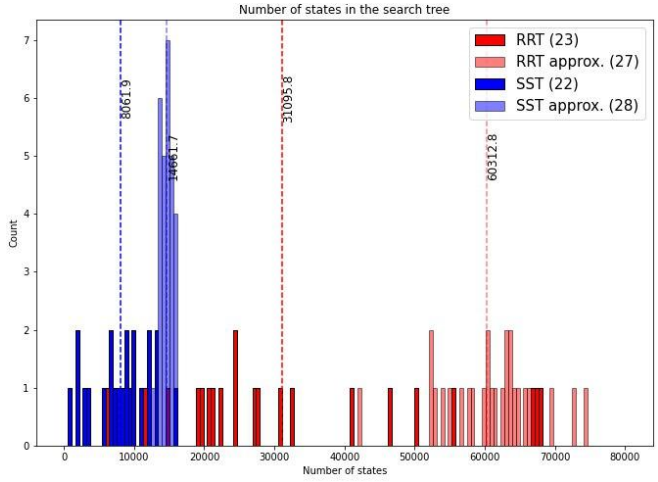


(b)

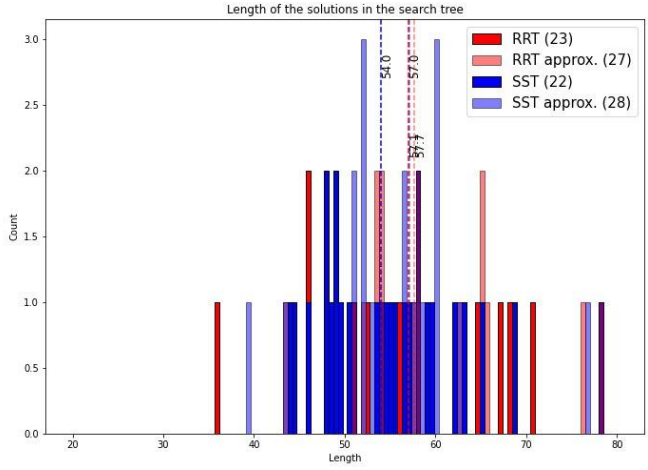


(c)

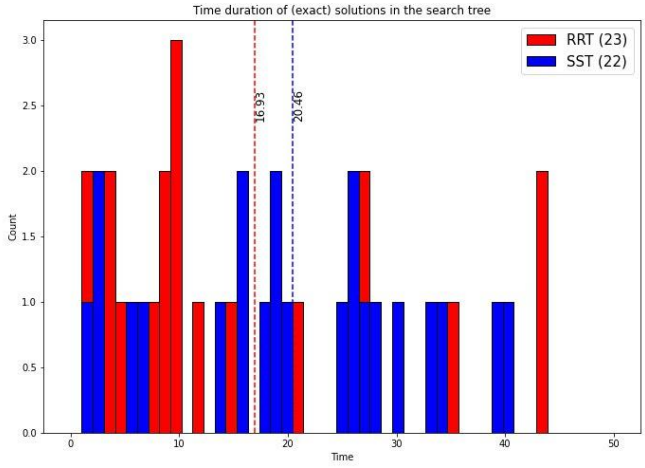
Figure 10: ***Simple forward motion histograms:*** *for number of states in search tree, path length and execution time.*



(a)

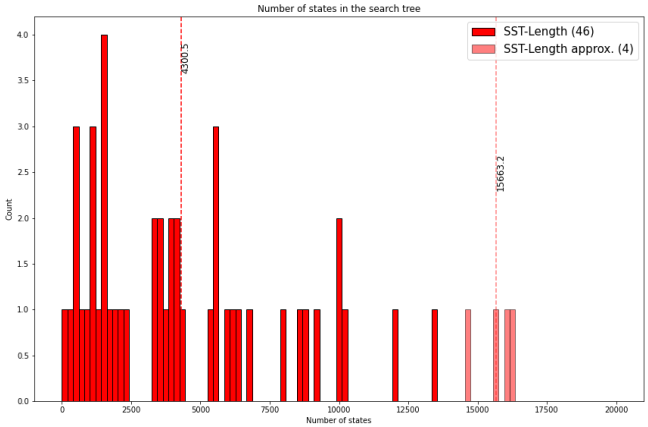


(b)

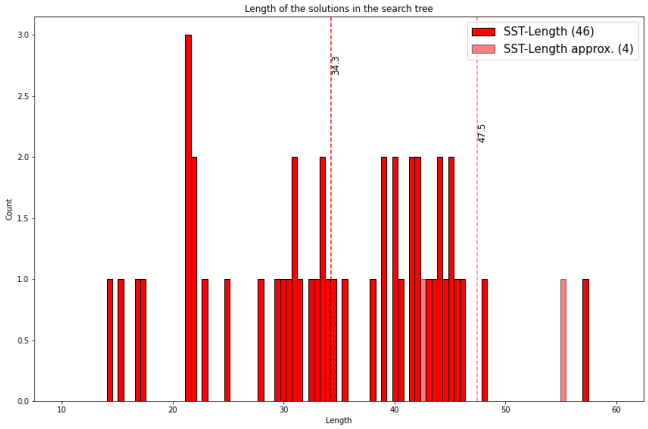


(c)

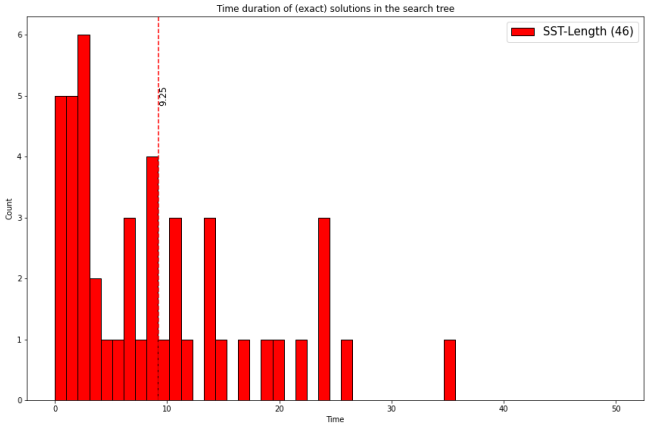
Figure 11: ***Simple backward motion histograms:*** *for number of states in search tree, path length and execution time.*



(a)

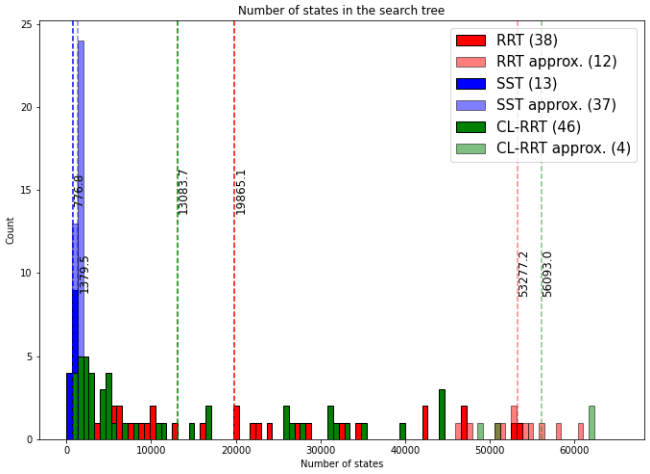


(b)

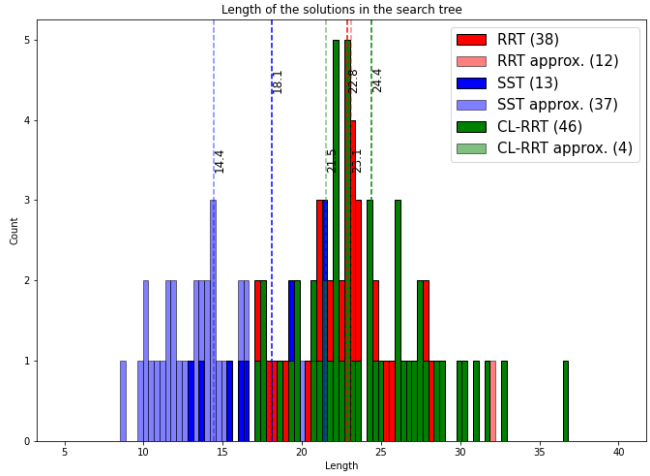


(c)

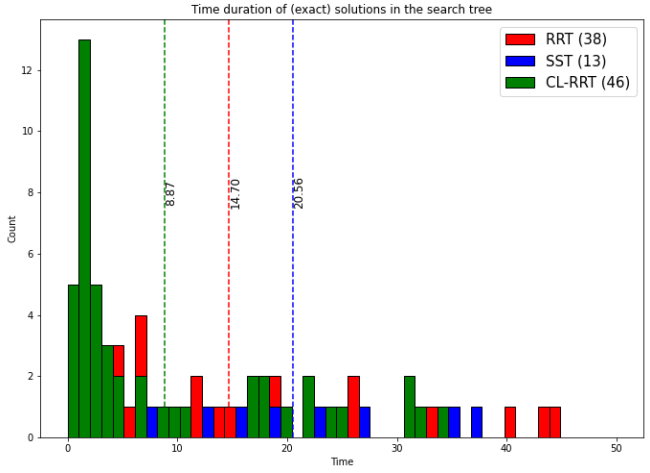
Figure 12: ***Circular obstacles avoidance histograms:*** *for number of states in search tree, path length and execution time.*



(a)

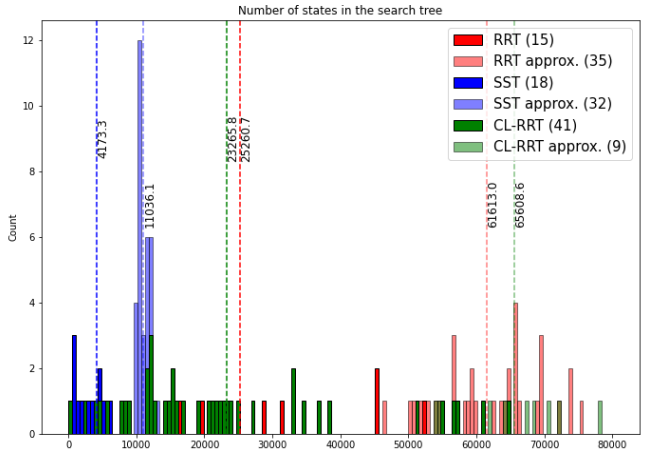


(b)

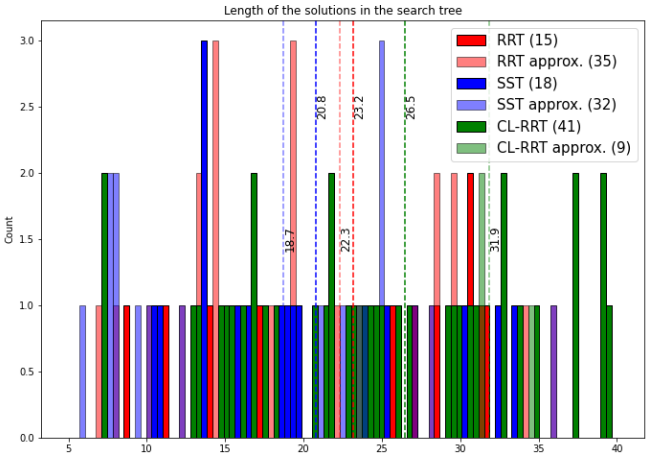


(c)

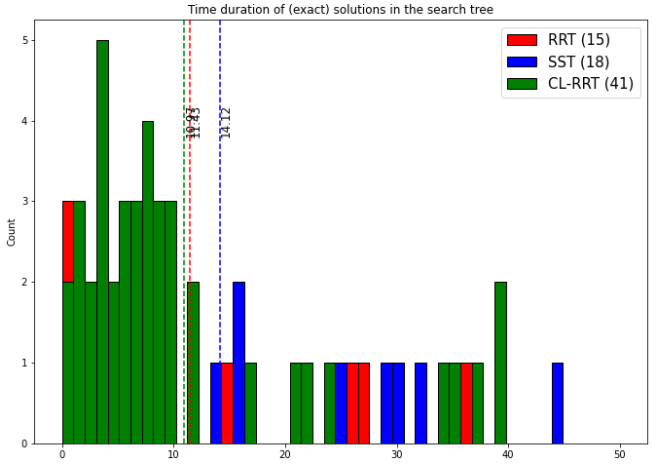
Figure 13: ***Three point turn histograms:*** *for number of states in search tree, path length and execution time.*



(a)



(b)



(c)

Figure 14: ***Real point turn histograms:*** *for number of states in search tree, path length and execution time.*

B. Experiments: Heatmaps

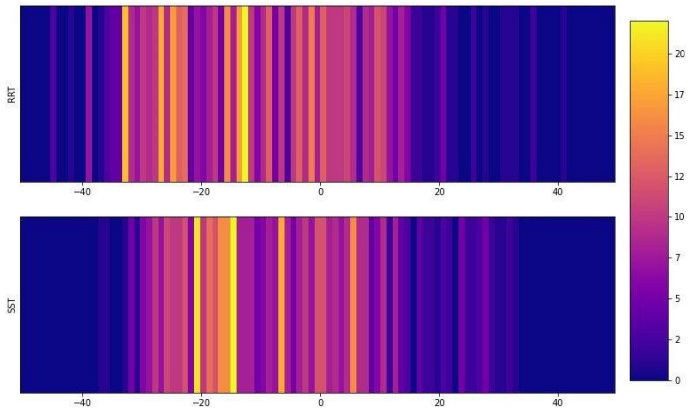


Figure 15: ***Simple backward motion experiment:*** *jackknifing risk heatmap.*

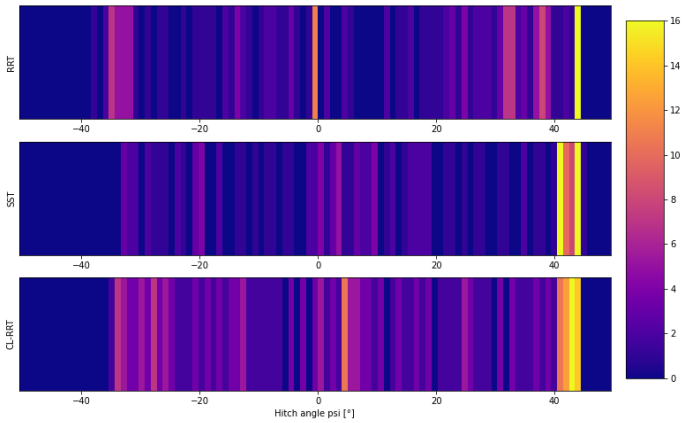


Figure 16: ***Three point turn experiment:*** *jackknifing risk heatmap.*

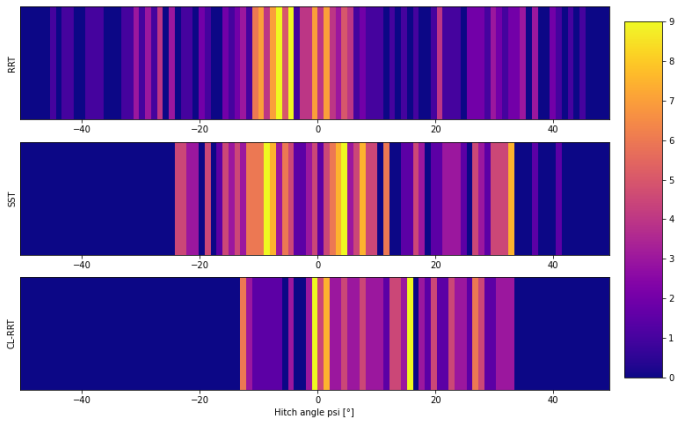


Figure 17: ***Real parking test experiment:*** *jackknifing risk heatmap.*