Heuristic RRT-based approaches to improve planning in obstacle-populated environments

Planning and Reasoning
Prof. Paolo Liberatore

Lorenzo Nicoletti
1797464
Sapienza Università di Roma

February 25, 2022

Abstract

The main goal of the proposed project is to build an «Open Motion Planning Library (OMPL)»([4])-based framework that compares different planning algorithms in environments containing obstacles, walls and obstructions and to highlight how the introduction of an heuristic can improve the quality of the solutions and optimize the available resources. The study is focused on a subset of planners belonging to the RRT ([1]) family; in particular, the analysis involves and compares the performances of a plain RRT, its optimized version known as RRT introduced by [2] and InformedRRT* ([3]), an expansion of the previous algorithm that is combined with an admissible heuristic to speed up the search for a solution. The algorithms will be used with a simple robotic architecture and the related results will be explained in a quantitative and a qualitative fashion to better underline the advantages and the limitations of each of the aforementioned planning techniques.*

I. Introduction

The following first section of the report is devoted to an overview of the adopted robot and the addressed planning methods from a theoretical point of view, it will provide a comprehensive description of the robotic mobile base used in the experiments and an explanation of the algorithms to introduce the reader in the topic.

i. Robot description

As sketched in Figure 1, the structure of the used mobile base is a simple omni-directional planar disk robot, which means that the robotic agent has no constraints on the maneuver direction and its workspace is limited to the $\mathbb{R}^2$ plane. Moreover, the robot is equipped with a tick on its surface to monitor the rotation of the base while moving; consequently, it defines an angle that corresponds to the orientation of the robot body that is added to its set of variables (even though it has no dynamic or kinematic meaning, since it is not constraining the motion of the agent). The robot is supposed to have no bounds on velocities and accelerations.

The configuration of the robot, namely the minimal set of parameters that allows to identify the posture of the robot in its workspace, is then defined as:

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \tag{1}$$

where x and y indicate the Cartesian position of the center point of the disk robot and θ is the orientation angle defined by the tick.

Consequently, the 3-dimensional configuration space $\mathcal{C}$, defined as the set of all possible values of the configuration of the robot, is represented as:

$$\mathcal{C} = \mathbb{R}^2 \times SO(2) = SE(2), \tag{2}$$

where:

$-SO(2)$ is the space of planar rotations;

$-SE(2)$, known as *Special Euclidean* space, is realized through the superposition of the Euclidean and the planar rotation spaces.

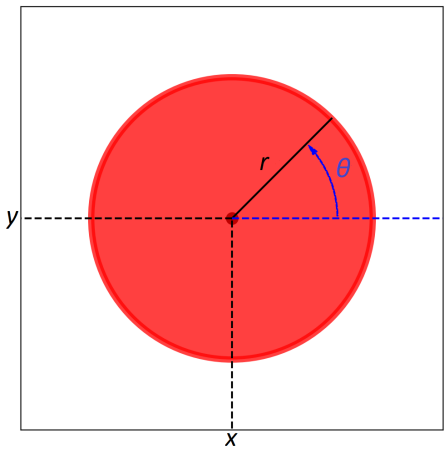


Figure 1: *The considered robotic mobile base with the configuration variables.*

The last parameter depicted in the figure above corresponds to the radius of the circular-shaped robot structure: this constant hyperparameter has been fixed to 0.2 meter and will be used to determine the distance of the disk to the obstacles and, consequently, to detect collisions between the robot and the environment.

ii. Basics of Heuristic (or Informed) Search

In general, a motion planning problem may be characterized by different paths that connect the start state q_s with the goal state q_g. Even though each of them represents a suitable solution to solve the prefixed task, such paths can differ in terms of length, power consumption, time and/or space complexity, memory storage and so on; consequently, the solution that better cares about these restrictions is often the preferred option to be outputted by the planning algorithm.

To overcome such limitations and provide feasible but also better solutions, the planning procedure can sometimes be combined with an **heuristic**, a suggestion that the algorithm uses to proceed the search and direct the tree expansion towards the most promising direction. The heuristic is basically an estimate of the distance to the goal and can be exploited in such a way that the planning algorithm selects the node n that optimizes

$$\min_{n \in SearchThree} d(n) + h(n), \tag{3}$$

where:

$-d(n)$ is the actual distance from the start node to n;

$-h(n)$ is the *estimated* distance from n to the goal.

It is immediate to understand how the most appropriate choice for $h(\cdot)$ requires particular attentions since it can hugely influence the effectiveness of the planning algorithm.

In the literature, several properties have been assigned to an heuristic to identify its quality and reliability. One of the most important characteristics is represented by the *admissibility* of the heuristic, namely the capability to never overestimate the real cost $h^*(\cdot)$ to reach the goal:

$$\forall n \in SearchTree, h(n) \leq h^*(n) \tag{4}$$

This property has been carefully considered in the implementation of the informed search in the project and will be discussed in detail in the next sections.

Finally, the most relevant aspect while evaluating results will be specifically to analyse the benefits introduced by the heuristic and to support the thesis of this project about the improvements derived with an informed-searching approach if compared to an non-informed one.

iii. Planning techniques

Thre different RRT-based planning algorithms have been exploited in the experimental session of the project.

iii.1 Rapidly-Exploring Random Tree (RRT)

Rapidly-Exploring Random Tree (RRT)([1]) is a probabilistic sampling-based planning method, namely the algorithm aims to build a roadmap of nodes from the start state to the goal by randomly sampling in the configuration space $\mathcal{C}$. Even though the procedure is simple, RRT is attested to be very efficient and performing in every kind of scenario, it is *probabilistically complete*, which means it is able to find a solution with a probability that goes to 1, and *single-query*, so it must be re-executed every time start and/or goal states are changed.

The basic iteration to build the search tree T_s rooted at q_s is to generate a vertex q_{rand} in $\mathcal{C}$ with a uniform probability distribution and search the nearest configuration q_{near} among the visited nodes in T_s. Then, a new node q_{new} is picked up at a certain distance δ from q_{near} in the direction of q_{rand} and, if no collisions are detected for q_{new} and for the segment connecting q_{near} to q_{new}, the node is safely added to T_s. This routine is graphically described in Figure 2, where the tree expansion process is carried out for a single iteration in an environment with simple obstacles.

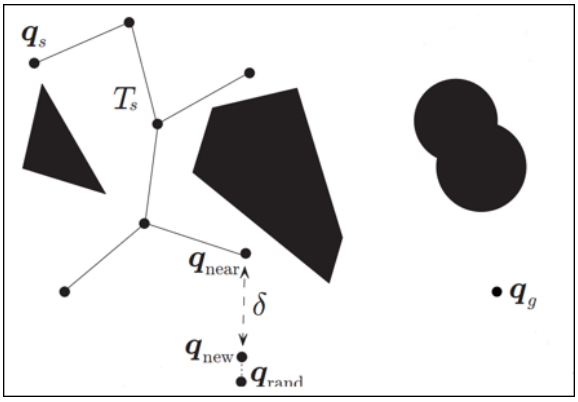


Figure 2: *Example of RRT sampling/expansion procedure.*

Thanks to its high computational efficiency, RRT is certified to be very fast while converging to a solution; moreover, it is capable to quickly cover unexplored areas in $\mathcal{C}$ even if it contains narrow passages between obstacles or difficult-to-reach regions.

On the other hand, a relevant drawback of the algorithm is that it is not *asymptotically optimal*, i.e., the computed solution cannot be proven to be the optimal one. This can be inevitably a problem in resource-limited applications; it is even more accentuated by the fact that plain RRT cannot be combined with an heuristic, the number of visited nodes remains relatively high and, consequently, the amount of required memory needs to be sufficiently large.

iii.2 RRT* for optimal planning

RRT* is an optimized version of RRT, developed by [2] to solve the aforementioned limitations of its plain version. The algorithm is, in fact, capable to find and deliver the shortest possible paths to the goal thanks to two key additions with respect to RRT:

1. RRT* can record the distance between each visited vertex in the search tree and its parent through a cost function defined for every node. When the closest node is found in the graph, a neighborhood of vertices within a fixed radius from the new node are examined and, if a neighbor node has a cheaper cost, it replaces the previously added node.

2. RRT* introduces a "rewiring" operation while building the tree. In particular, after a vertex has been connected to its cheapest neighbor as described in the point above, the neighbors are again examined by checking if a rewiring to the newly added vertex will make their cost decrease. If the cost indeed decreases, the neighbor node is rewired to the newly added one; otherwise, no modifications are performed. This feature makes the paths smoother if compared to the trajectories tracked by the simple RRT.

Except for these two new operations, the functioning of the algorithm is the same of RRT: the sampling of q_{rand} and the selection of q_{near} and q_{new} are similar as before. Then, q_{new} is subjected to the two examinations described above in order to optimize the cost function and attempt to provide cheaper solutions at each iteration.

As mentioned before, RRT* is an optimal planning algorithm[1], which means that it is able to optimize a specific criterion and output the best possible solution according to such objective. In the experimental section of this project, RRT* has been combined with two different optimization objectives (or their weighted combination):

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[2] 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 node n, the cost of n is represented as:

$$c_{length}(n) = c_{length}(n.parent) + \|n - n.parent\| \tag{5}$$

2. **Minimum 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 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 configuration q can be defined as:

$$\gamma(q) = \min_{s \in \partial C_{free}} \|q - s\| \tag{6}$$

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

$$c_{clearance}(n) = c_{clearance}(n.parent) + \gamma(q) \tag{7}$$

[1] in particular, *asymptotically optimal.*

[2] 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 configuration spaces involving rotation angles, since it is a reasonable approximation only for «local» motions. In this case, the assumption is empirically found to be valid because no inconsistencies have been detected during the experiments.

Finally, the optimization problem can be simply summarized as:

$$\min_{n \in SearchTree} c_{length}(n) \tag{8}$$

or

$$\max_{n \in SearchTree} c_{clearance}(n) \tag{9}$$

according to the requested objective.

Optionally, RRT* supports weighted combination of criteria as well: exploiting the fact that

$$\max_{n \in SearchTree} c_{clearance}(n) \equiv \min_{n \in SearchTree} \frac{1}{c_{clearance}(n)} = \min_{n \in SearchTree} c'_{clearance}(n) \tag{10}$$

the compound cost function to be minimized is defined as:

$$c(n) = \lambda_1 c_{length}(n) + \lambda_2 c'_{clearance}(n), \tag{11}$$

where λ_1 and λ_2 represent the weighting factors that measure the influence of each objective in the optimization procedure.

iii.3 InformedRRT* for informed optimal planning

InformedRRT*, firstly introduced by [3], is the core of the project because of its capability to exploit the benefits of an heuristic to perform informed search. The algorithm is basically built on top of RRT* with the additive characteristic that uses heuristics to only consider a sub-problem that could provide a better solution. Thanks to this relaxation, the search is limited to this specific sub-problem by pruning the graph, generating samples only in this sub-problem and, when available, using the measure of this sub-problem to calculate the connection terms to expand the exploration while iterating over the RRT-based pipeline.

According to the authors, InformedRRT* is an extension of RRT* that demonstrates how informed sets can be used to improve asymptotically optimal planning. Since the search is focused only to the relative sub-problem, the algorithm increases the likelihood of sampling states that can improve the solution and increases the convergence rate towards the optimum.

More specifically, at each iteration, InformedRRT* calculates the current best solution from the vertices in the goal region. This defines an informed set of nodes that is used to both sample q_{rand} and prune the graph. This process continues for as long as time allows or until a suitable solution is found.

InformedRRT* retains the probabilistic completeness and asymptotically optimality of RRT*.

At this point, it is clear how the most innovative aspect of this planning algorithm is the usage of an *informed state sampler*, which is responsible to guide the search tree expansion towards the direction suggested by the adopted heuristic. Since the execution of the planner is integrated with an optimization objective, the choice of the heuristic can be safely performed accordingly to the criterion to minimize/maximize. This project is, in fact, based on the assumption that the optimization objective and the heuristic can be grouped together in order to merge optimal planning and informed search and to leverage the advantages of both problem instances. The proposed integration is indeed feasible thanks to the fact that the criteria defined by Equation 5 and Equation 7 already imply admissible heuristics[3] and satisfy the condition at Equation 4.

Therefore, InformedRRT* selects the node n that optimizes (minimizes/maximizes according to the adopted criterion)

$$d(n) + h(n) = d(n) + c(n), \tag{12}$$

given that $c(\cdot)$ represents the requested objective cost function in the optimal planning problem.

[3]This implication can be easily derived by the fact that both objectives are based on the Euclidean distance that is intrinsically defined as the shortest displacement between two points in the plane and, consequently, it will never overestimate the actual distance that can be even larger due to the presence of obstacles and walls.

II. Implementation

As already mentioned, the project is entirely based on OMPL ([4]). It is a motion planning software package which contains the implementation of several 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 the design choices and implementation using OMPL-functionalities.

i. Instantiating the robot and its state space

OMPL already provides simple state spaces (for instance, $\mathbb{R}^n$, $SO(2)$, $SO(3)$, $SE(2)$ and $SE(3)$), therefore, the creation of the needed configuration space defined in Equation 2 was immediate. In particular, OMPL's $SE(2)$ is built as a compound space through the combination of $\mathbb{R}^2$ with weight 1 and $SO(2)$ with weight 0.5, where the weighting factors are used in the computation of the distance between states (consequently, θ has lower influence with respect to x and y, as desired).

Since the robot has no kinematic constraints, its creation is straightforward: it is modeled as a simple point robot by considering the center of the disk and the calculation of distances between this point and obstacles are offset by its radius r.

ii. Creating the scenarios and the visualization tools

Three different simulation scenarios have been created for the experiments following an «increasing-difficulty» criterion in terms of available free space, maneuver complexity and obstacle-population quantity. Since OMPL does not support direct visualization, the project implements a "plotting" routine to compensate such absence: when a solution in `.txt` format is outputted by the planner, it is parsed by a `Python` script to create a video simulation of the robot path in the environment by merging together all the "frame-wise" images.

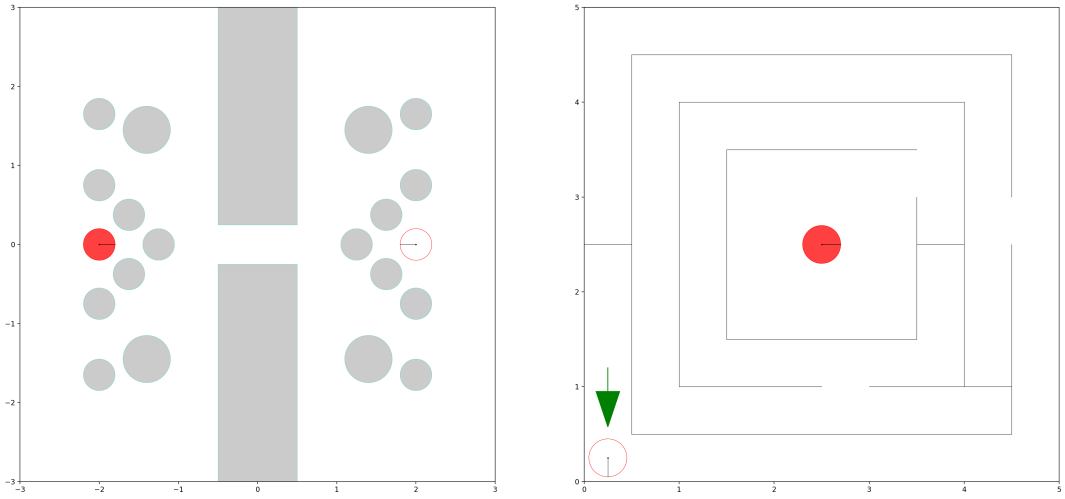
The adopted scenarios are very different and peculiar, each of them has been designed to test and evaluate specific properties and characteristics of the planning algorithms that will be later analyzed in the next section of the report. The environments, shown in Figure 3, are:

a **Obstacles avoidance**, it is the first and the simplest scenario representing an obstacle-populated environment: circular-shaped obstacles at a different location and with varying radius have been carefully placed in the two large areas that are divided by a narrow passage in the middle (Figure 3a);

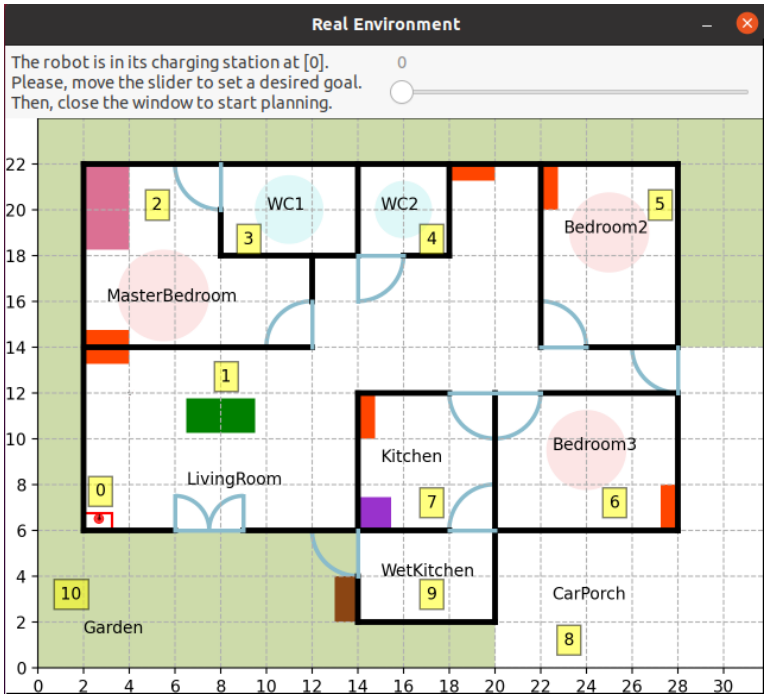
b **Maze**, it is an environment created as a tricky labyrinth with lots of dead-end corridors to test the ability of the planner of backtracking when useless tree branches are found during the search. The maze exit is indicated by the green arrow at the bottom left corner (Figure 3b);

c **Real-world test**, it is a realistic environment representing a house with several rooms and obstacles contained therein where the robot starts moving from its "charging station" at $[0]$ and attempts to move towards the user-chosen goal state. The main aspect of this final scenario is to study the behaviour of the robot in a close-to-truth world (Figure 3c).

The environmental parameters are contained in Table 1 and specify q_s and q_g for each scenario together with the boundaries of the $\mathbb{R}^2$ domain. It's interesting to notice the dimension of the *Real-world test* environment, that is definitively larger than the other twos because it actually represents a realistic house area. Moreover, the choice of the goal state is left to the user that is free to select a desired destination by specifying the $[id]$ of the room to reach.



(a) *Obstacles avoidance* **(b)** *Maze*



(c) *Real-world test*

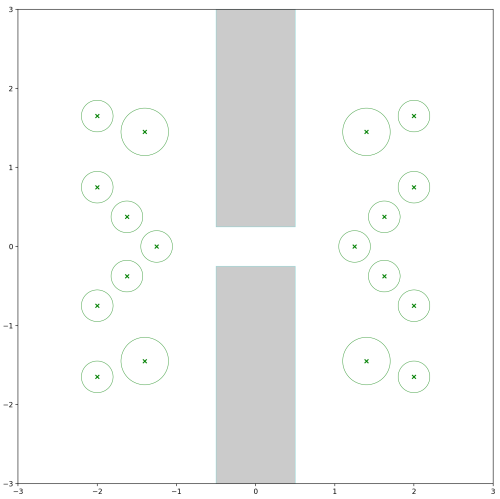
Figure 3: *Scenarios for the experiments: the «filled» and «empty» robots in each environment represent the start and the goal states, respectively. In "Real-word test", a fixed goal state is missing since the choice is left to the user by moving the slider in the GUI at will.*

Environment	Boundaries		q_s			q_g		
	$[x_l, x_h]$	$[y_l, y_h]$	x	y	θ	x	y	θ
Obstacles avoidance	$[-3, 3]$	$[-3, 3]$	-2	0	0	2	0	$-\pi$
Maze	$[0, 5]$	$[0, 5]$	2.5	2.5	0	0.25	0.25	$-\frac{\pi}{2}$
Real-world test	$[0, 32]$	$[0, 24]$	2.675	6.5	$\frac{\pi}{2}$	tbd	tbd	tbd

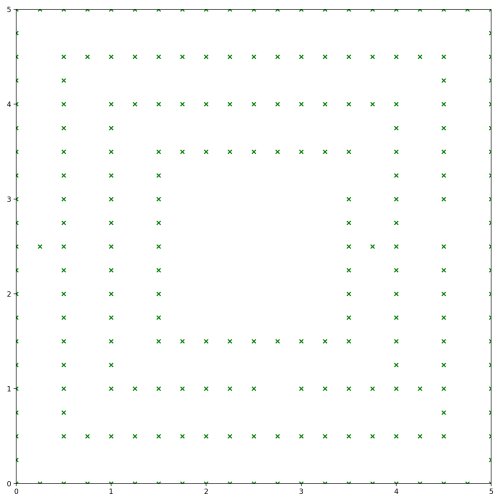
Table 1: ***Environmental parameters:*** *state space bounds for the Cartesian components, initial and goal configuration.*

iii. Setting up state validity and collision checking

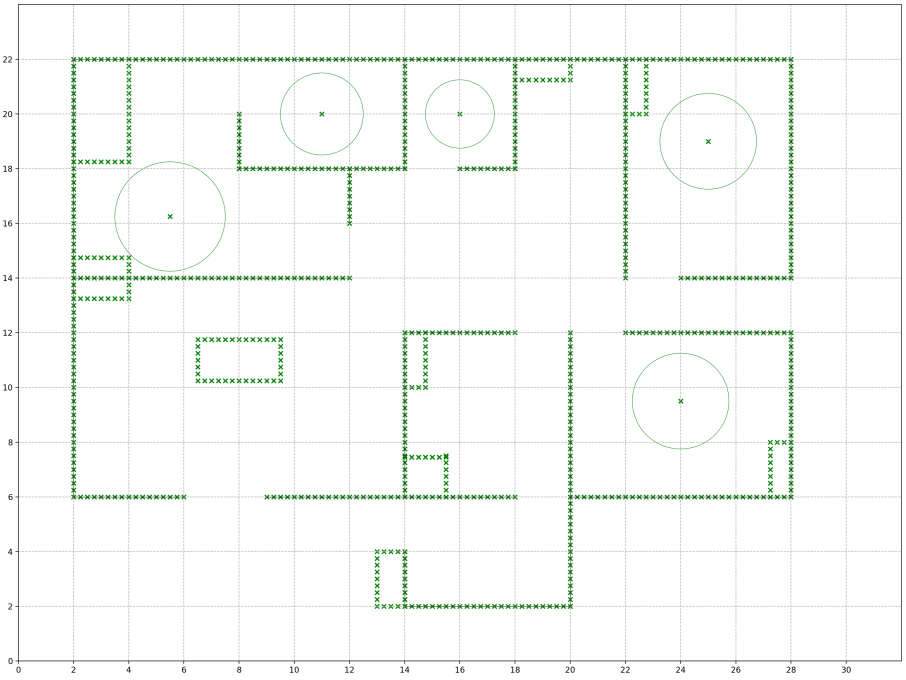
One of the aforementioned limitations of OMPL was the unavailability of a pre-implemented collision-checking routine. Therefore, this component should be customized. In the proposed approach, the obstacles are modeled through a discrete and finite number of representative *control points* along the perimeter of the obstructions, as shown in Figure 4. 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].



(a) *Obstacles avoidance*



(b) *Maze*



(c) *Real-world test*

Figure 4: *Control points for each environment defined by the green circles and/or crosses along the obstacles body.*

[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).

While circular-shaped obstacles are easily "represented" by a single control point located at their center and by their radius to define the circumference, the walls and square-shaped obstructions require a relevant number of points to be checked for a collision. In particular, *Obstacles avoidance* contains 18 control points (Figure 4a), one per each circle, while the more complex *Maze* and *Real-world test* need respectively 225 and 834 sufficiently-near control points in the environment (Figure 4b and Figure 4c).

Consequently, the state validity checking role simply becomes an evaluation of the minimum distance between the two closest control points: one of the disk robot and the other of the nearest obstacle.

It is important to notice that the same reasoning is inherited by the minimum clearance maximization objective in the optimal planning problem with (Informed)RRT*: the clearance to be optimized is actually the minimum distance computed by the state validity while checking for collisions.

III. Experiments

In this section, experiments and results will be analyzed and evaluated in a quantitative but also qualitative fashion. Data have been collected through benchmarks, an OMPL tool to re-execute an experiment for a desired amount of N runs and to store common properties such as length and clearance of the paths or time needed to find an exact solution. Each planning algorithm has been executed for a prefixed amount of time: if no solutions are derived within this time limit, an approximate (not precisely reaching the goal) solution is returned. The *accuracy rate*, defined as the ratio of number of exact solutions over the total number of runs, will be a relevant metric when the complexity of the scenario will strongly influence the effectiveness of the planners.

i. Obstacle avoidance

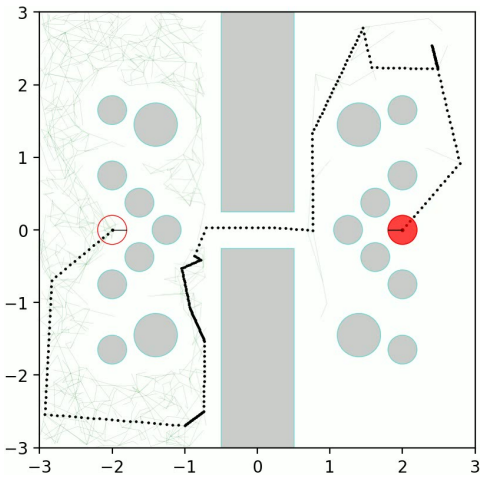
The *obstacle avoidance* scenario is designed in order to highlight the influence that an heuristic/optimization objective can have while planning and building a solution path. Each experiment performed in such environment has been repeated for $N = 100$ times with a time limit of 30 seconds and the related quantitative results are attached in Figure 9 and Figure 10 and reported in Table 2 and Table 3. In particular, a first set of runs has involved RRT and the two optimal planners, RRT* and InformedRRT*, combined with path length minimization; while a second branch of experiments has differed in the usage of minimum clearance maximization as optimizing criterion.

Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	100	1550	-	1550	27.1	-	27.1	**0.03**
RRT* - Le	100	1421	-	**1421**	16.7	-	16.7	0.21
InformedRRT* - Le	100	1496	-	1496	16.6	-	**16.6**	**0.19**

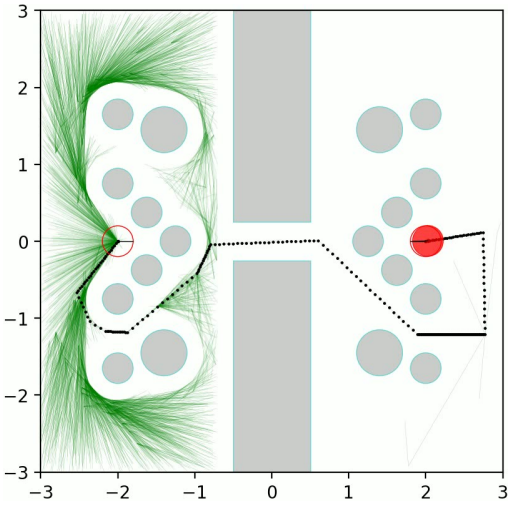
Table 2: ***Obstacles avoidance with path length minimization quantitative evaluation****: average number of states, solution length and execution time. The % column indicates the accuracy rate.*

Planner	%	Avg number of states			Avg clearance			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	**100**	1425	-	1425	0.32	-	0.32	**0.02**
RRT* - Cl	98	1370	3712	1417	0.43	0.41	0.43	**9.33**
InformedRRT* - Cl	**99**	327	4008	**364**	0.43	0.47	**0.43**	10.75

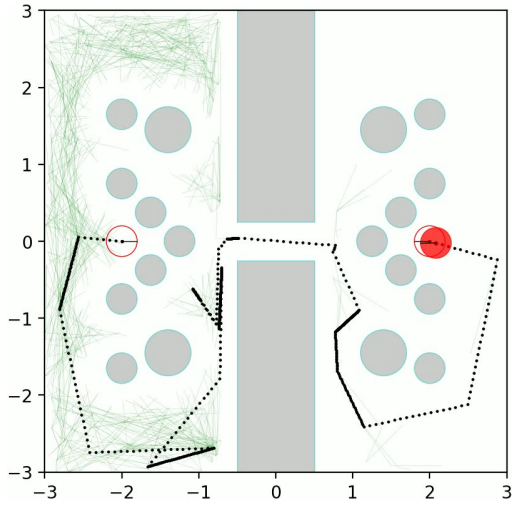
Table 3: ***Obstacles avoidance with minimum clearance maximization quantitative evaluation****: average number of states, solution clearance and execution time. The % column indicates the accuracy rate.*



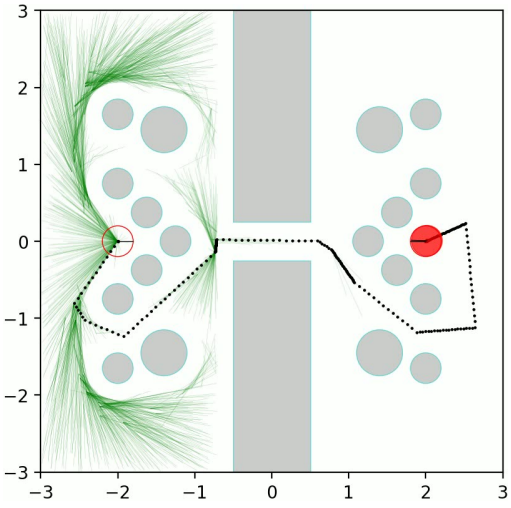
(a) *RRT*



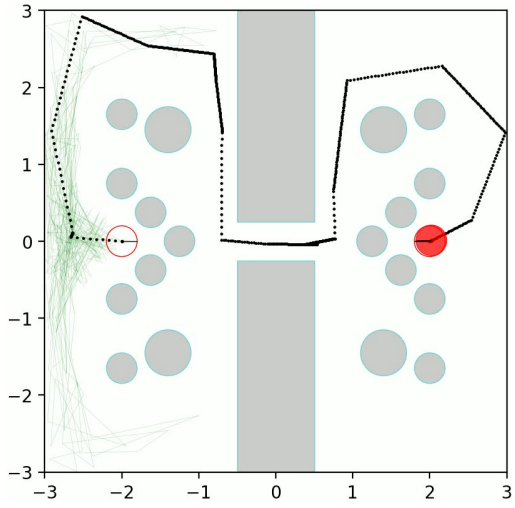
(b) *RRT* with path length minimization*



(c) *RRT* with minimum clearance maximization*



(d) *InformedRRT* with path length minimization*



(e) *InformedRRT* with minimum clearance mazimization*

Figure 5: ***Solution samples for the «Obstacles avoidance» scenario with different planning algorithms and objectives.***

Due to the simplicity of the environment, the accuracy rate is always very high: only few experimental instances has reported approximated solutions when optimal planners were combined with the more articulated clearance cost function. Moreover, even though RRT is extremely faster with respect to the others because it is not combined with a function to optimize, its reliability is very poor given the fact that it usually outputs longer and more dangerous solutions if compared to the shorter and safer paths tracked by RRT* and InformedRRT*.

In general, the introduction of an heuristic in this scenario makes InformedRRT* the most performing algorithm with slightly higher achievements in terms of average length and clearance. The greatest success of this method is given by its incredible small amount of stored nodes in the search tree when trying to maximize the distance from the obstacles: the 364 average number of states in the graph is even four times lower than the other approaches. These improvements are graphically reflected in the aggregated plots of Figure 9b, Figure 10a and Figure 10b as well, where the correspondent distributions highlight the described advantages of the heuristic technique.

The underlined superiority of InformedRRT* is perceived also from a qualitative point of view thanks to the solution samples grouped in Figure 5: the related paths are typically smoother than the other trajectories and the search trees are less dense with respect to the other optimal planner, RRT*.

Moreover, the reported results are perfectly explaining the varying behavior of the robot path according to the adopted objective: if minimizing the length, the disk robot goes through some dangerous and narrow corridors between obstacles (Figure 5b and Figure 5d); on the other hand, when safe maneuvers are preferred, the robotic agent usually takes the longer route to circumnavigate the obstructions and avoid the risk of collisions (Figure 5c and Figure 5e).

ii. Maze

The second environment, *Maze*, has been devised to monitor the capability of the algorithms to backtrack when a corridor is attested to be dead-ended. The scenario has registered a difficulty for the optimal planners with clearance maximization to reach a solution because of the excessive narrowness of the labyrinth; therefore, RRT* and InformedRRT* have been combined only with length minimization.

The experimental session was composed by $N = 25$ runs for each algorithm with an execution limit of 120 seconds. This longer time interval has allowed to register all and only exact solution paths. The correspondent quantitative results are reported in Table 4 and shown in Figure 11.

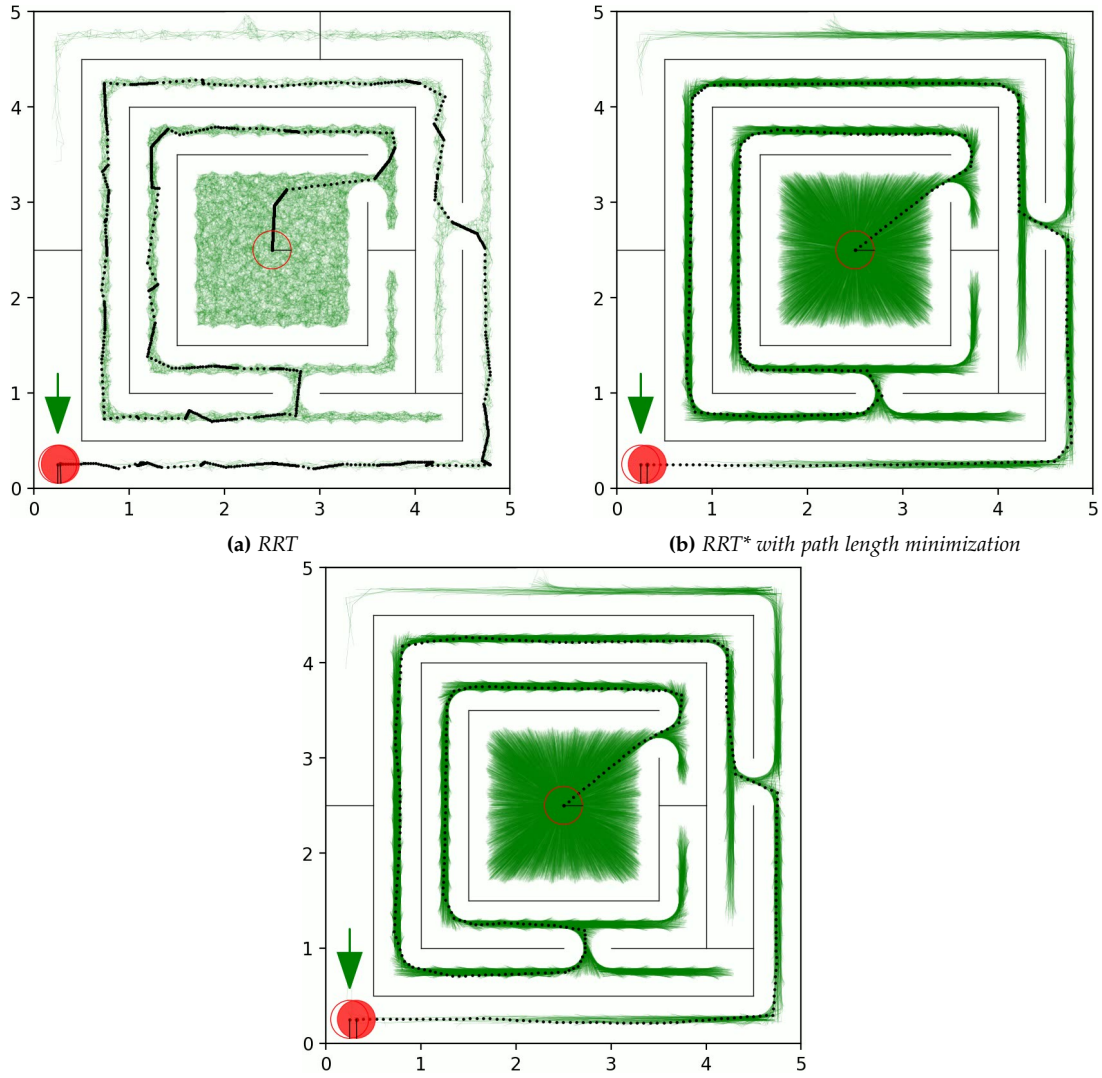
Planner	%	Avg number of states			Avg length			Avg time
		exact	approx.	total	exact	approx.	total	
RRT	100	45130	-	**45130**	50.3	-	50.3	**5.18**
RRT* - Le	100	46059	-	46059	26.6	-	26.6	102.92
InformedRRT* - Le	100	46057	-	**46057**	26.5	-	**26.5**	**102.10**

Table 4: ***Maze quantitative evaluation****: average number of states, solution length and execution time. The % column indicates the accuracy rate.*

Similarly to the previous case, RRT has recorded the shortest execution time due to the fact that it is not constrained in optimizing an objective function. Nevertheless, it is characterized by a doubled average path length, factor implying the uncertainty and the incapability of this plain algorithm to prevent the disk robot from entering dead-end areas. On the other hand, optimal planning is attested to be an efficient method to overcome such limitation and provide the shortest possible path to leave the maze: RRT* and its informed version registered similar performances, where the latter certified slight improvements in the number of nodes, path length and execution time. In general, both approaches are reliable and trustworthy and no relevant differences are detected with an heuristic search.

Analogous conclusions can be derived by observing the qualitative sampled solutions in Figure 6: the path tracked by RRT is clearly characterized by several uncertain motions; while

with the other two optimal methods, the disk robot smoothly reaches the goal state located at the exit of the maze.



(a) *RRT*

(b) *RRT* with path length minimization*

(c) *InformedRRT* with path length minimization*

Figure 6: ***Solution samples for the «Maze» scenario with different planning algorithms.***

iii. Real-world test

The final environment has been designed to analyse the behaviour of the disk robot in a realistic environment: a house organized in several rooms with domestic appliances, beds, tables and other items commonly found in a habitation. The choice of the adopted planning algorithms has been performed according to this aim. Since previous evaluations have highlighted the high unreliability of the plain RRT, the experiments have involved only the two optimal planners, RRT* and InformedRRT*. Moreover, the optimization objective/heuristic to improve planning performances has been modified in order to include simultaneously all the advantages from both path length minimization and minimum clearance maximization by exploiting a mixed criterion where the two previous objectives have been merged together, as defined by Equation 11 with $\lambda_1 = \lambda_2 = 5$ to equally balance the importance of both metrics.

The set of experiments in this scenario was organized in $N = 15$ runs for each planner with a prefixed time limit of 150 seconds. The related quantitative results are grouped in Table 5 and shown in Figure 12, the goal state has been set at the correspondence of the id [6]: *"Go to Bedroom3"*.

Planner	%	Avg number of states			Avg length			Avg clearance			Avg time
		exact	approx.	total	exact	approx.	total	exact	approx.	total	
RRT*	53	28	427	214	34.9	28.3	31.8	0.97	1.18	**1.07**	**0.94**
InfRRT*	**60**	35	383	**174**	34.7	28.7	**31.3**	0.92	1.05	0.97	2.53

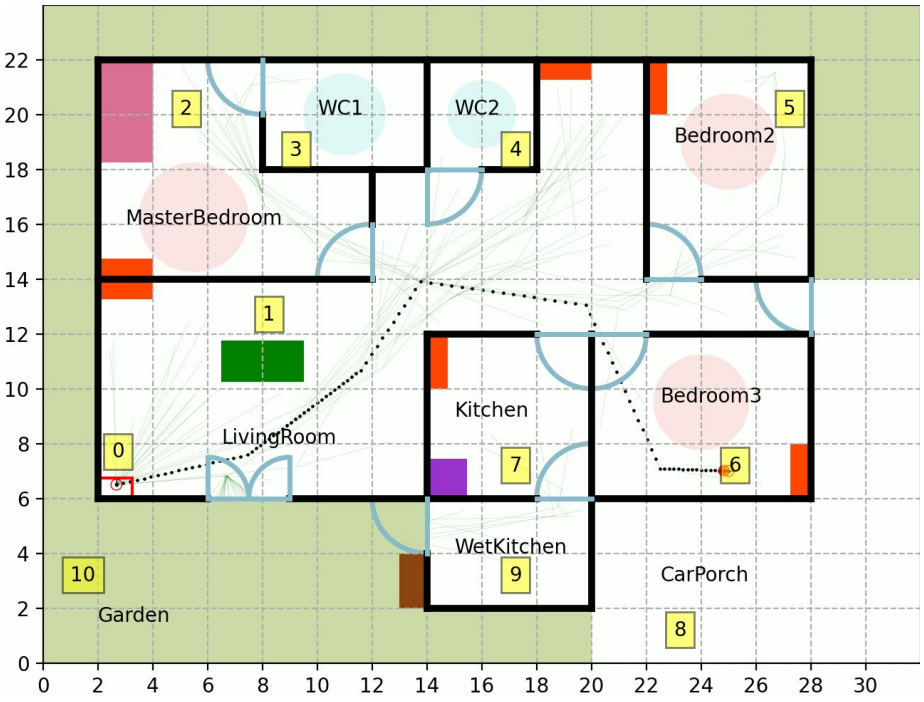
Table 5: ***Real-world test quantitative evaluation****: average number of states, solution length, solution clearance and execution time. The % column indicates the accuracy rate.*

Since the dimensions of the environment are intentionally larger if compared to the diameter of the disk robot and, consequently, the size of the available free space may confuse the planner while approaching to the goal, the accuracy rate registered a relevant decrease in terms of number of exact solutions found. However, the 60% probability of success for InformedRRT* makes this algorithm more efficient than standard RRT*. Even if the latter reported a shorter time needed to solve the task and a slightly larger clearance, the introduction of the combined heuristic has allowed InformedRRT* to minimize path length while definitely reducing the number of visited nodes (and consequently the required memory storage) thanks to its sub-problem relaxation. The property of considering only a portion of the problem instance has guaranteed overall better performances and InformedRRT* was certified as the most performing algorithm in this challenging and articulated scenario as well.

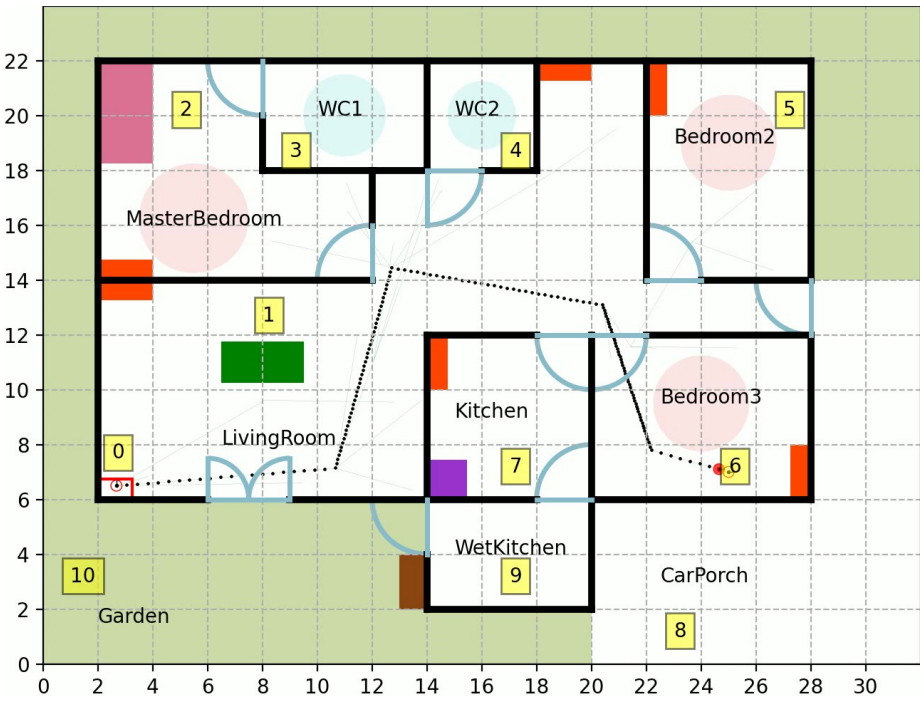
Qualitative results are displayed in Figure 7 with goal id again set to [6] and in Figure 8 with goal id set to [3]: *"Go to WC1"*. In general, both planners have outputted very similar smooth and feasible solution paths, where the combined length minimization and clearance maximization is perceived when the robot is immediately directed to the destination but simultaneously steers away from obstacles and walls. The main advantage of InformedRRT* is clearly evident by focusing on the growth of the search trees: in both cases, RRT* required a preliminary exploration of distant-to-the-goal rooms such as *MasterBedroom* and *WC1* in Figure 7a or *WetKitchen* and *CarPorch* in Figure 8a; on the other hand, the search trees expanded by InformedRRT* are shallower and, therefore, the solution goal states are more easily reached.

IV. Conclusions

The main goal of the project was to compare different planning algorithms to enhance the benefits of the heuristic search and to integrate the application into the OMPL software package. The proposed 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 the implementation, by highlighting its strengths and limitations. Motion and optimal motion planning problems have been addressed and solved in increasing-complexity environments and the peculiarities of each algorithm have been exhaustively analyzed, while proceeding in the dissertation. As expected, the most satisfactory results were related to the introduction of admissible heuristics in the system, an estimation of the distance to the goal to move the search towards more promising directions. The informed search, in fact, has allowed to achieve better solutions, attested to be robust and reliable achievements. All the experiments were commented with accompanying plots and tables in a quantitative and qualitative way. Finally, the proposed results were studied in terms of the adopted evaluation metrics and the overall outcomes of the project were coherent with the introductory objectives of the proposed work.

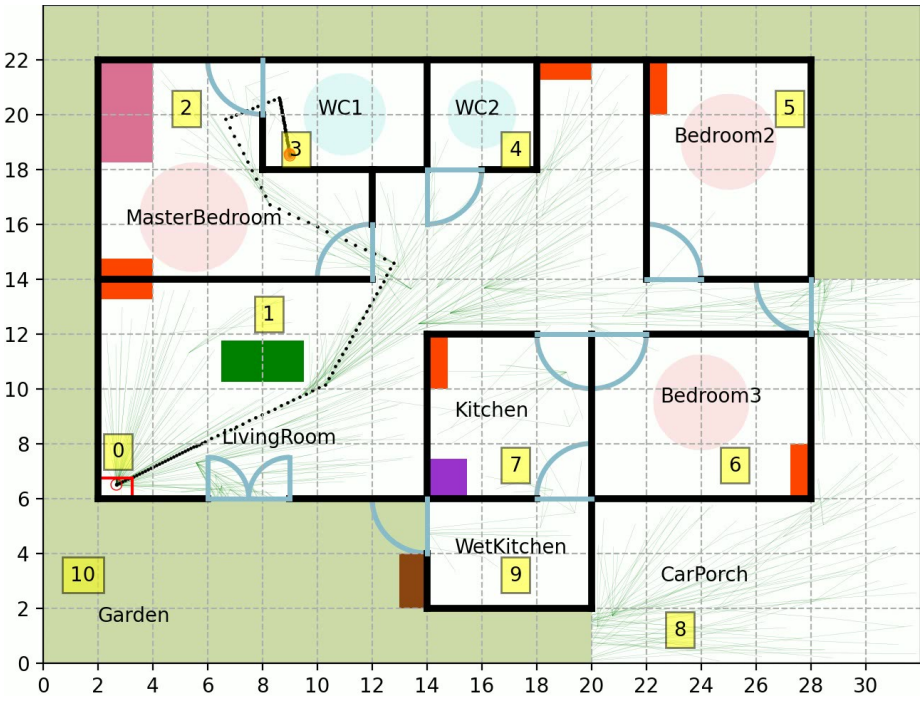


(a) *RRT**

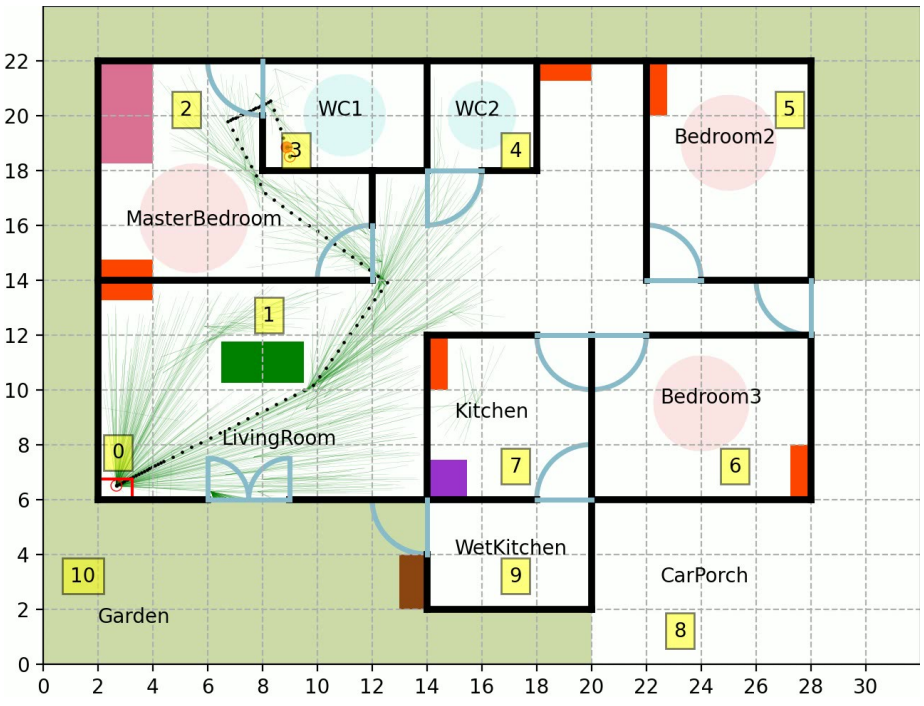


(b) *InformedRRT**

Figure 7: ***Solution samples for the «Real-world test» scenario with different planning algorithms and goal state set to*** [6].



(a) *RRT**



(b) *InformedRRT**

Figure 8: ***Solution samples for the «Real-world test» scenario with different planning algorithms and goal state set to*** [3].

References

[1] *J. J. Kuffner and S. M. LaValle*

"RRT-connect: An efficient approach to single-query path planning," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings, 2000, pp. 995-1001 vol.2*

[2] *S. Karaman and E. Frazzoli*

"Sampling-based Algorithms for Optimal Motion Planning", *International Journal of Robotics Research, 2011*

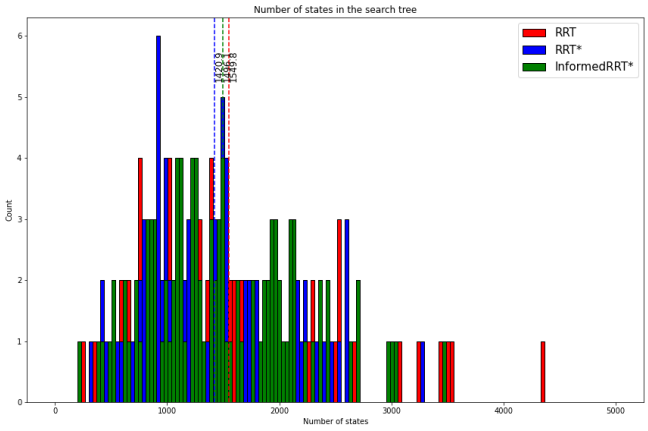
[3] *J. D. Gammell, T. D. Barfoot and S. S. Srinivasa*

"Informed Sampling for Asymptotically Optimal Path Planning", *IEEE Transactions on Robotics, vol. 34, no. 4, pp. 966-984, Aug. 2018*

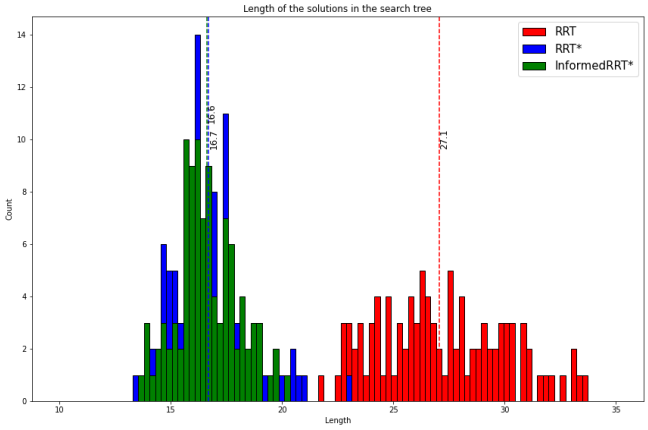
[4] Open Motion Planning Library. *http://ompl.kavrakilab.org/*

Appendix

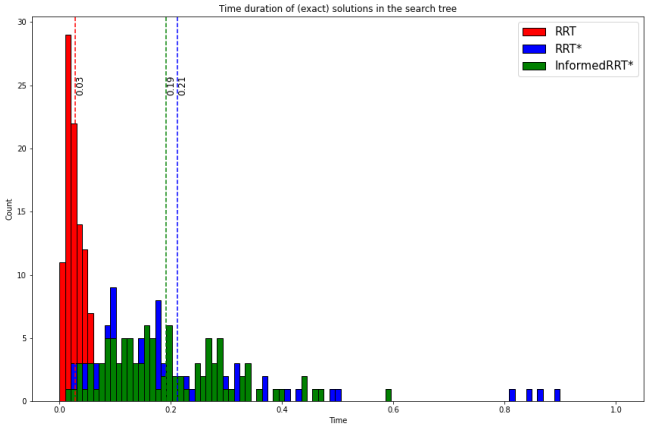
A. Obstacles avoidance: Path Length Minimization



(a)



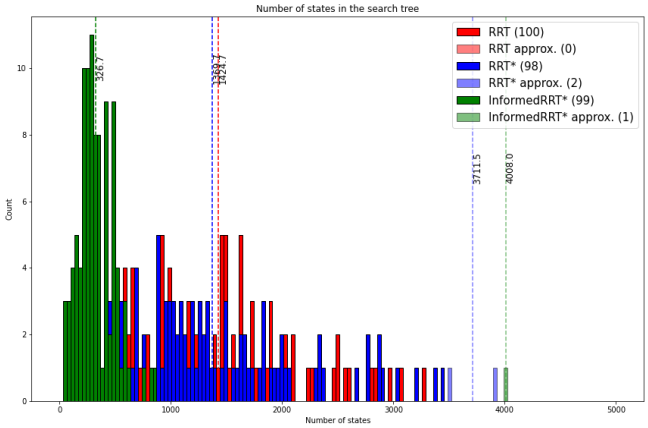
(b)



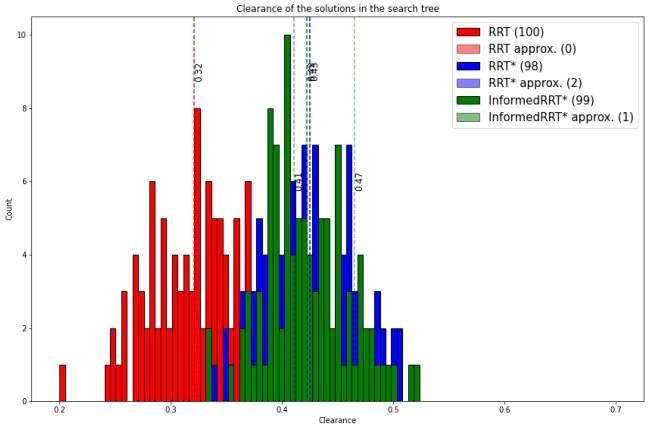
(c)

Figure 9: ***Obstacles avoidance histograms with path length minimization:*** *for number of states in search tree, path length and execution time.*

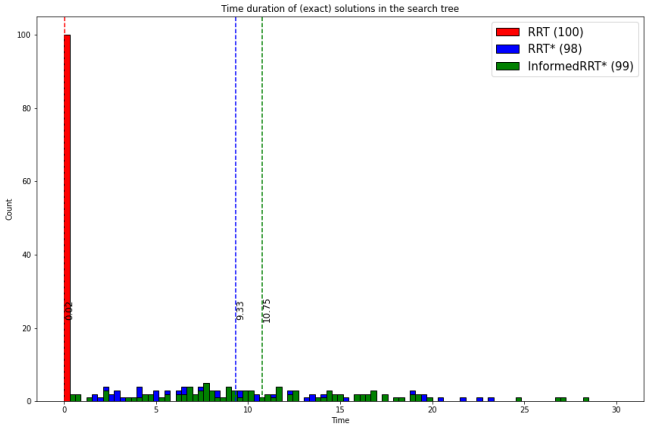
B. Obstacles avoidance: Minimum Clearance Maximization



(a)



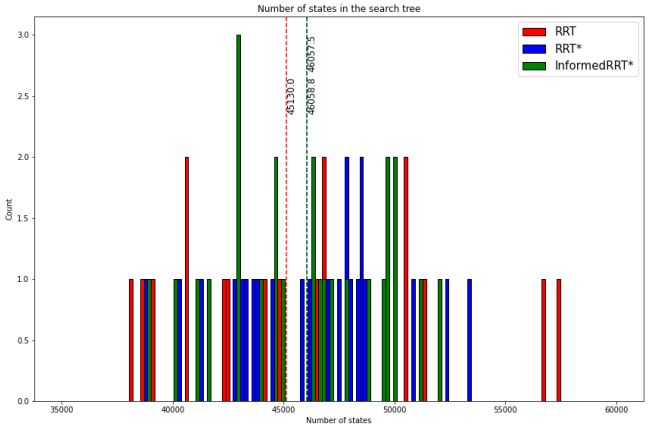
(b)



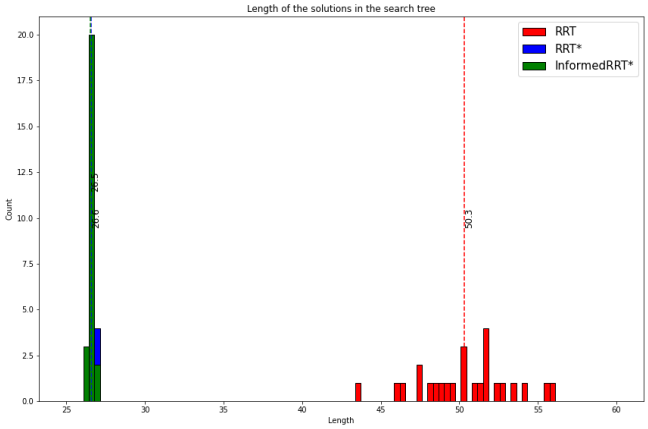
(c)

Figure 10: ***Obstacles avoidance histograms with minimum clearance maximization:*** *for number of states in search tree, clearance and execution time.*

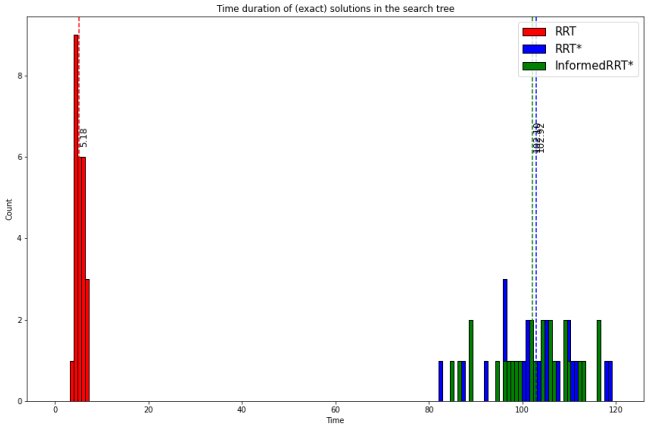
C. Maze



(a)



(b)



(c)

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

D. Real-world test

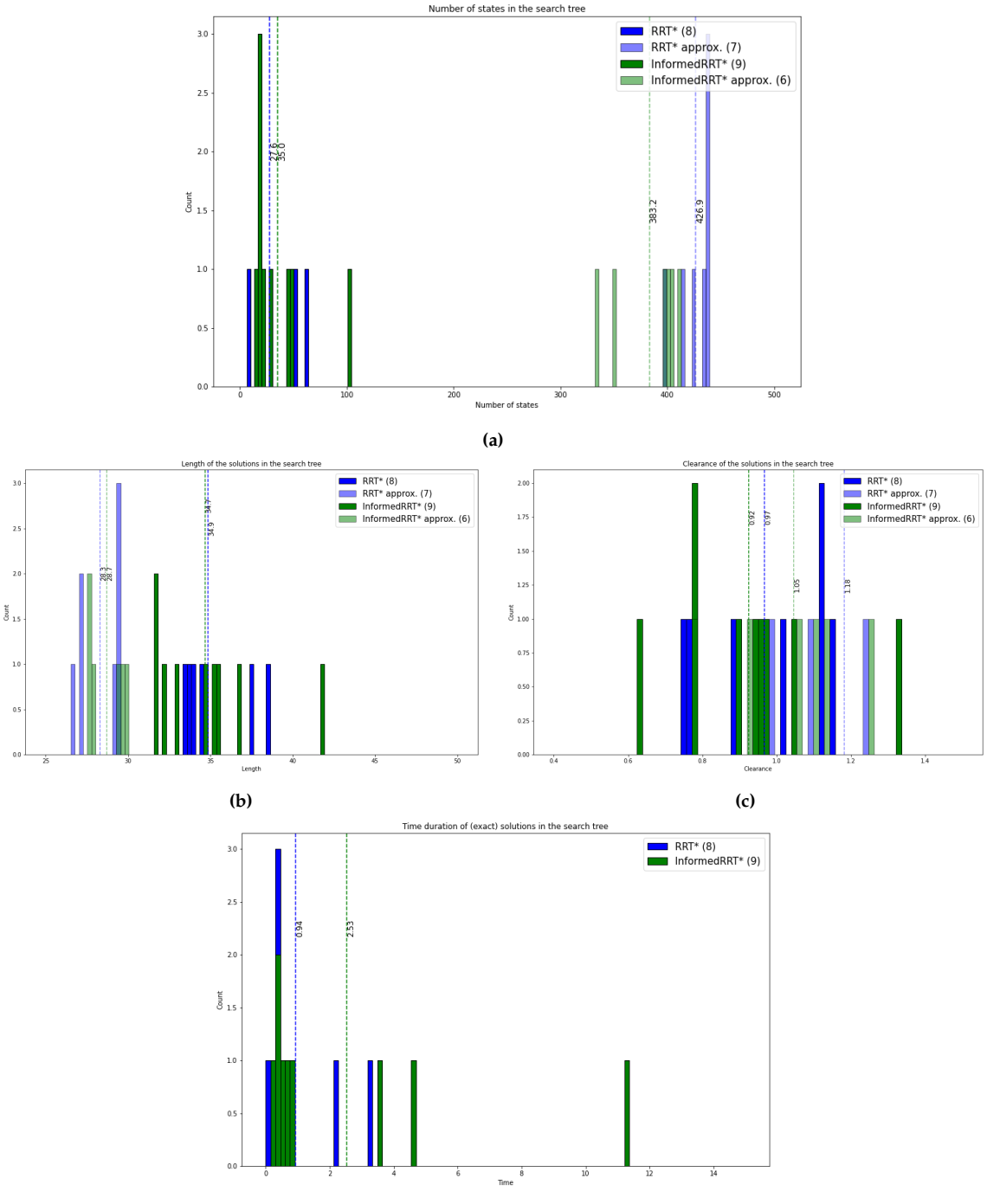


Figure 12: ***Real-world test histograms:*** *for number of states in search tree, path length, clearance and execution time.*