A Multi-Heuristic Search-based Motion Planning for Autonomous Parking

Bhargav Adabala[1], **Zlatan Ajanovic**[1]
[1] Virtual Vehicle Research GmbH,
Inffeldgasse 21a,
8010 Graz, Austria
zlatan.ajanovic@v2c2.at

Abstract

Planning is a crucial component of autonomous vehicle control. It is responsible for finding a collision-free sequence of states that take the vehicle towards its goal. In unstructured environments like parking lots or construction sites, due to the large search-space and kinodynamic constraints of the vehicle, real-time planning is challenging. Several state-of-the-art solutions utilize heuristic search-based planning algorithms. However, they heavily rely on the quality of the single heuristic function used to guide the search, and they are not capable to achieve reasonable performance, resulting in unnecessary delays in the response of the vehicle. This work solves the planning problem by adopting a Multi-Heuristic Search approach, that enables the use of multiple heuristic functions and their advantages to capture different complexities of a given search space. Based on our knowledge, this approach was not used for this domain so far. For this purpose, multiple admissible and non-admissible heuristic functions are defined, original Multi-Heuristic A* Search was extended for bidirectional use and dealing with hybrid continuous-discrete search space and a mechanism for adapting scale of motion primitives is introduced. To demonstrate the advantage, Multi-Heuristic A* algorithm is benchmarked against a very popular heuristic search-based algorithm, Hybrid A*. The Multi-Heuristic A* algorithm outperformed Hybrid A* in terms of computation efficiency and motion plan (path) quality.

Introduction

Robot motion planning problems can be elegantly formulated as a path planning in higher-dimensional configuration space (Lozano-Pérez and Wesley 1979). However, finding a solution is computationally challenging due to a large continuous search space and kinodynamic constraints. The sampling-based approaches for motion planning have been extensively studied in robotics (Kingston, Moll, and Kavraki 2018). Instead of explicitly constructing the collision-free configuration space, which is time-consuming to compute, these algorithms probe the free space and search with a sampling strategy. The algorithms stop when a path connecting the initial and final poses is found. According to the sampling type, the sampling-based path finding algorithms can be classified into two categories:random-sampling-based algorithms and orderly-sampling-based algorithms.

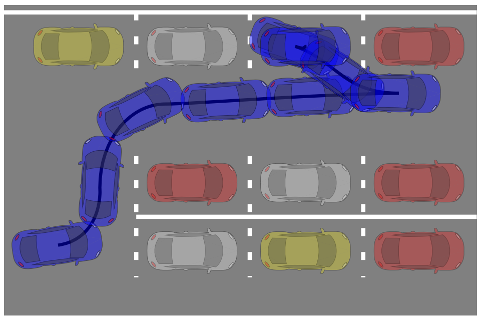

Figure 1: Motion Planning for Autonomous Parking.

The most popular random-sampling-based algorithms is the Rapidly-exploring Random Tree (RRT) (LaValle 1998). RRT can be considered as a special case of Monte Carlo Tree Search (MCTS) (Browne et al. 2012). The most notable orderly sampling-based algorithm is A* (Hart, Nilsson, and Raphael 1968) including many of its extensions. It was initially developed to plan a path for the Shakey robot and it was further generalised and used for many different domains since then.

The orderly sampling-based algorithms tend to be more efficient than random sampling-based planners, especially when the dimensions of the state spaces are fewer than six (LaValle, Branicky, and Lindemann 2004). Random sampling-based planners inherently come with the disadvantages of being highly non-deterministic and converging towards solutions that are far from the optimum and suffer from bug trap problems (Karaman and Frazzoli 2010). The variants of RRT such as RRT* also provide comparable solutions, but they are computationally expensive and the efficiency depends on the size of the search space (S. Karaman et al. 2011). Furthermore, if no collision-free path to the goal exists, orderly sampling-based algorithms can report this failure much more quickly than random sampling-based ones. However, original A* deals with discrete state-space and requires discretization of the configuration space. Continuous state-space and kinodynamic constraints can be satisfied by constructing lattice in the form of a regular grid (M. Pivtoraiko and A. Kelly 2005). Another approach is to use motion primitives (Frazzoli, Dahleh, and Feron 2002). To avoid the problem of rounding states generated using motion primitives to the grid, Hybrid-State A* (Dolgov et al. 2008) might be used.

Sampling-based motion planning approaches were extensively used for autonomous vehicle path planning in unstructured environments during the 2007 DARPA Urban Challenge, both A*-based (Likhachev and Ferguson 2009), (Dolgov et al. 2008) and RRT-based (Kuwata et al. 2009). Several extensions for these approaches have been introduced in recent years, namely A*-based (Nemec et al. 2019), RRT*-based (Banzhaf et al. 2017b) and optimization-based (Zhang, Liniger, and Borrelli 2020). An overview of recent developments and open challenges is presented in (Banzhaf et al. 2017a).

Besides motion planning for wheeled vehicles in unstructured environments (i.e. autonomous parking), different variants of search-based planning were recently used for planning footsteps for humanoid robots (Ranganeni et al. 2020), robot manipulation (Mandalika, Salzman, and Srinivasa 2018), underwater vehicles (Youakim et al. 2020), the aggressive flight of UAVs (Liu et al. 2018), as well as for special use-cases in automated driving such as energy-efficient driving (Ajanović, Stolz, and Horn 2018), driving in complex urban environments (Ajanović et al. 2018) and performance driving including drifting maneuvers (Ajanović et al. 2019).

This paper presents a motion planning approach for wheeled vehicles in unstructured environments based on Multi-Heuristic Search (Aine et al. 2016).

Based on our knowledge, this is the first application of a Multi-Heuristic planning approach for motion planning in autonomous parking scenarios.

This is achieved by using a combination of geometric and orderly sampling-based approaches in order to achieve maximum coverage of the complexities within the search space. For this purpose, two heuristic functions are defined to get an accurate estimate of the cost-to-go and prune the unnecessary search nodes for faster path computation. The geometric approach is used to solve the simplified problem (without obstacles) by modelling the physical constraints of the vehicle and is used as a second heuristic function in a Multi-Heuristic A* algorithm, the first being the path length to the destination while considering obstacles but neglecting some physical constraints like turning radius. With this approach, both non-holonomic and holonomic constraints are combined to provide an optimal solution to the motion planning problem.

Two approaches for the solution have been developed, one using *FORWARD SEARCH* and the other using *BI-DIRECTIONAL SEARCH*. Additionally, adaptive motion primitive arc length was developed to avoid the search getting stuck in depression regions indefinitely.

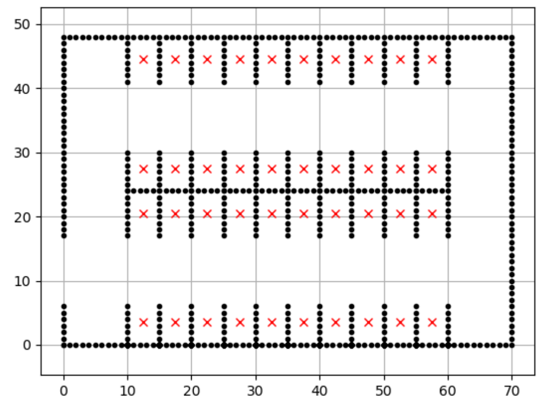

Figure 2: Parking with perpendicular parking slots

Autonomous Parking as Motion Planning Problem

The autonomous parking problem tackled in this work represents fully observable problem where intelligent infrastructure provides a connected vehicle with information about the structure and area of the parking lot. The driver enters the parking and drives the vehicle into a designated drop off area. The autonomous parking algorithm overtakes a control and guides and maneuvers the vehicle into the assigned parking slot automatically based on the information from the infrastructure. This concept was demonstrated by Bosch and Daimler as Automated Valet Parking (AVP) (Becker et al. 2014).

The planning problem this work aims to solve can be stated as follows:

”Find a solution in real-time that autonomously navigates a non-holonomic vehicle without any collisions, from a given start position to a desired goal position within the parking layout based on the input of a two-dimensional obstacle map, or report the non-existence of such a solution.”

To fully define the problem, environment (obstacles), the vehicle model and Key Performance Indicators (KPIs) must be defined.

Environment

Figure 2, shows an example of the parking layout structure which is considered for the motion planning problem of autonomous parking. There might be two most prominent parking orientation, perpendicular or parallel. However, the algorithm should be general to work on different parking arrangements. Each parking slot is enumerated with a parking slot ID. The number of parking slots and dimensions of the open space is configurable. The red cross symbols in the figure highlight the goal position for each parking slot within the layout. The entry point to the parking lot is fixed at $(x_s, y_s, \theta_s) = (0, 10, 0)$. The parking position for the respective parking slots ID is read from a pre-computed map. The computation of goal position within each parking slot

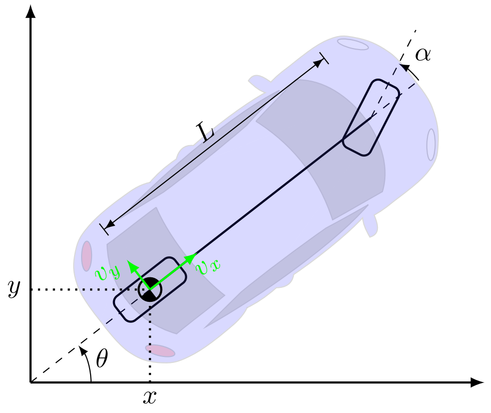

Figure 3: Single-track vehicle model.

is based on the vehicle dimensions as the control reference would be different for the individual vehicle due to differences in length and width.

Vehicle Model

Due to vehicle geometry and physics, there are constraints on the vehicle motion that restrict the allowable velocities. The first order constraints, that consider the first derivative of the position (velocity), are often called *kinematic* constraints. Including the dynamics of a vehicle results in second-order differential constraints, which allows the modelling of acceleration. The planning with such models is called *kinodynamic* planning. As the focus of this work is on low-velocity parking maneuvers, higher-order constraints are not included, only kinematic constraints are considered.

A simple yet useful model of a car is the *single track model*, also known as the *bicycle model*, shown in Figure 3. It does not consider dynamics, but it is useful to model lower velocity driving. Consider the case where a car with wheelbase L moves forward with velocity v_x with a steering angle α and assuming no wheel slip, then the car will move along a circle with radius R. The kinematic constraints can then be derived by trigonometry. Let **x** = (x, y, θ) denote the configuration state of the car-like robot, where x and y denote the position and θ the heading of the car. Vehicle motion is constrained to $\dot{x}/\dot{y} = \tan(\theta)$, which together with the constraint $R = L/\tan(\alpha)$ gives the following first-order differential constraints:

$$\dot{x} = v_x \cos(\theta) \tag{1}$$

$$\dot{y} = v_x \sin(\theta) \tag{2}$$

$$\dot{\theta} = \frac{v_x}{L} \tan(\alpha) \tag{3}$$

Setting the maximum steering angle $|\alpha| \leq \alpha_{\max}$ results in a minimum turning radius $R_{\min}$. The model clearly represents the non-holonomic behaviour as it is impossible to move sideways without violating the no-slip condition. Restricting the allowed velocities and steering angles to the finite set $\mathcal{U}_{v_x} = \{0, 1\}$ and $\mathcal{U}_\alpha = \{-\alpha_{\max}, 0, \alpha_{\max}\}$ results in the Dubins car that can only stop and move forward at unit speed (Dubins 1957) and setting it to $\mathcal{U}_{v_x} = \{-1, 0, 1\}$ results in the Reed-Shepp car that can also reverse at a unit speed (Reeds and Shepp 1990). Even though these models are very simplified they have efficient analytic solutions for optimal paths between any two states that can be useful for designing heuristic functions for search-based motion planners.

A time discretized single-track model is easily obtained using Euler forward or higher-order methods. By integrating the differential equations forward in time, a simulated path or trajectory resulting from a given input can be obtained and used to construct motion segments within a planning framework.

Collision Detection

To avoid collisions with obstacles in the environment, vehicle geometry should be considered. The geometry of a car, approximate rectangular shape, can be reasonably approximated with overlapping circular disks. This simplifies the collision detection significantly as it is sufficient to check if the obstacles fall within the boundaries of the disk, which is represented by the radii of the disks. The problem of covering rectangles with equal-sized disks in an optimal way has been studied in the literature (Melissen and Schuur 2000).

As stated in (Ziegler and Stiller 2010), a rectangle of length l and width w can be covered by n circles of radius r calculated as:

$$r = \sqrt{\frac{l^2}{n^2} + \frac{w^2}{4}} \tag{4}$$

placed at a distance of d calculated as:

$$d = 2\sqrt{r^2 - \frac{w^2}{4}} \tag{5}$$

In practical applications, the above approximation may lead to under-utilization of available collision-free space especially in case of environments where tight or narrow maneuvering is required, e.g. parking lots. Besides, the stated approach assumes the control reference to be at the center of the rectangular shape which is not true in the case of car-like robots which are mostly either front-wheel or rear-wheel driven.

In this work, an approach proposed in (Ziegler and Stiller 2010) is adapted to fit a practical application of a car-like robot. The bounding disks can be arranged as shown in Figure 4. By this method, the bounding disks are arranged compactly to fit the geometry of the vehicle allowing better utilization of the free space even for tight maneuvers in narrow spaces. Moreover, all the calculations are based on the standard dimensions available from any production car design and parametrizing the same makes the approach generic for any vehicle under consideration.

KPI Definition for Benchmark

To compare the performance and quality of solutions generated by motion planning algorithms, the following Key Performance Indicators (KPIs) have been used.

- **Performance Paramters**

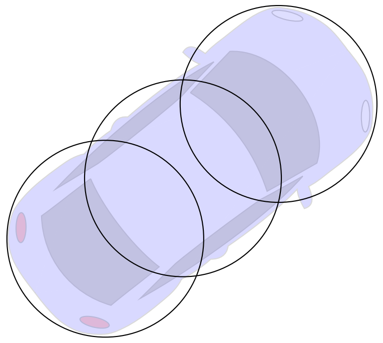

Figure 4: 3 Disk approximation for vehicle geometry.

- *Number of Expanded States*: For a given configuration space, the number of expanded states reflects the guidance power of the heuristic functions in pruning the unwanted branches of the search. The lesser the number of expanded states, the better the heuristic.
- *Execution Time*: The execution time depends on the implementation of the vehicle model, the definition of motion primitives and the algorithm itself. It is a measure of the time that the algorithm needs to return a solution using the defined attribute functions.
- *Number of Iterations*: The iteration counter quantifies how quickly the algorithm converges to either finding a solution or reporting that there exists no solution.

• **Solution Path Quality Parameters**

- *Path Length*: The path length is computed as the accumulated sum of Euclidean distance between two points on the final trajectory. It quantifies the efficiency of the generated solution as shorter path lengths are preferred.
- *Reverse Path Length*: The reverse path length indicates the quality of the algorithm to foresee a wrong branch. The longer reverse path length indicates that the vehicle had to move a lot in a backward direction in order to correct its path or in some cases the algorithm prefers to move the vehicle more in a backward direction rather than in forward. In any case, longer reverse path lengths are not preferred.
- *Direction Changes*: Each direction change during driving indicates a stop-and-go situation, which will be annoying for a human driver. Even though it is an autonomous vehicle the quality solution shall be close to an experienced human driver, i.e., avoid multiple direction changes.

Motion Planning Approach

The motion planning approach presented in this paper is based on Multi-Heuristic A* search, extended with a concept from hybrid A* Algorithm, employed in a bi-directional search fashion. Kinodynamic feasibility of the solution is provided by motion primitives based on the vehicle model. Several admissible heuristic functions enable efficient optimal planning.

Hybrid A* Algorithm

The Hybrid A* algorithm was developed as a pratical path-planning algorithm that can generate smooth paths for an autonomous vehicle operating in an unstructured environment and used in the DARPA Urban Challenge by the Stanford University team (Dolgov et al. 2008). The hybrid A* algorithm is based on the A* algorithm, with the key difference being, that state transitions occur in continuous rather than in a discrete space. By considering the non-holonomic constraints of the robotic vehicle, the algorithm generates feasible transitions which can be executed by the actuator module.

The three dimensional state space $\mathcal{X}$ (represented by x, y position and θ heading angle of the vehicle) is associated with a discrete grid of reasonable resolution such that each continuous state is associated with some grid cell to enable the use of discrete search algorithm. Continuous states are rounded to the grid for association in order to prune the branches, by keeping only the best trajectory coming to the grid cell. The expansion still uses the actual continuous value that is not rounded to the grid. Similar to the original A*, if the current state being expanded is not the goal state, new successors are generated for all possible actions $u \in \mathcal{U}(\mathbf{x})$. The *cost-to-come* is only computed for successor states that are not in the CLOSED list. If the state is not in the OPEN list it is directly pushed to the OPEN list. If the state is already in the OPEN list, and the *cost-to-come* is smaller than the cost for a state with the same index that is in the OPEN list then the pointer to the parent, the *cost-to-come* and the *cost-to-go* are updated. After that, the key is decreased using the newly computed cost.

Multi Heuristic A* Algorithm

The performance of A* algorithm depends on the quality of the heuristic function used to guide the search. It is hard to design a single heuristic function that captures all the complexities of the problem. Furthermore, it is hard to ensure that heuristics are admissible (provide lower bounds on the cost-to-go) and consistent, which is necessary for A*-like search to provide guarantees on completeness and bounds on sub-optimality.

In (Röger and Helmert 2010) authors introduced an approach of alternation between different heuristic functions for satisficing (i.e. non-optimal) planning. Multi-Heuristic A* (MHA*) (Aine et al. 2016) overcomes the dependency on a single heuristic function in optimal planning too. MHA* can use multiple inadmissible heuristic functions in addition to a single consistent heuristic simultaneously to search in a way that preserves guarantees on completeness and bounds on sub-optimality. This enables to effectively combine the guiding powers of different heuristic functions and simplifies dramatically the process of designing heuristic functions by a user because these functions no longer need to be admissible or consistent (Aine et al. 2016).

MHA* has two variants: Independent Multi-Heuristic A* (IMHA*) which uses independent cost-to-come and cost-to-go values for each search, and Shared Multi-Heuristic A* (SMHA*) which uses different cost-to-go values but a single cost-to-come value for all the searches. With this shared

Algorithm 1: Bi-Directional Multi-Heuristic Search

```
1  function $SMHA^*$ ($x_\mathrm{I}$, $X_\mathrm{G}$, $\mathcal{O}$, $h_i$):
2      return Collision free trajectory from $x_\mathrm{I}$ to $x \in X_\mathrm{G}$
3  function $CombinePath$ (($x_\mathrm{start}, \dots, x'_\mathrm{f}$),($x''_\mathrm{f}, \dots, x_\mathrm{G}$)):
       /* Use an analytic function to connect paths */
4      return Combined trajectory from $x_\mathrm{start}$ to $x_\mathrm{goal}$
5  begin
       /* Forward Search */
6      $X_\mathrm{G} \leftarrow \{x \mid \|x - x_\mathrm{G}\| \le d_\mathrm{fw1}\}$;
7      $(x_\mathrm{start}, \dots, x'_\mathrm{f}) \leftarrow SMHA^*(x_\mathrm{start}, X_\mathrm{G}, \mathcal{O}, h_i)$;
       /* Reassign Start and Goal Positions */
8      $x'_\mathrm{start} \leftarrow x_\mathrm{goal}$;
9      $X'_\mathrm{G} \leftarrow \{x \mid \|x - x'_\mathrm{f}\| \le d_\mathrm{fw2}\}$;
       /* Backward Search */
10     $(x_\mathrm{G}, \dots, x''_\mathrm{f}) \leftarrow SMHA^*(x'_\mathrm{start}, X'_G, \mathcal{O}, h_i)$;
       /* Combine Paths */
11     $path \leftarrow CombinePath((x_\mathrm{start}, \dots, x'_\mathrm{f}), (x''_\mathrm{f}, \dots, x_\mathrm{G}))$;
12 return trajectory
```

approach, SMHA* can guarantee the sub-optimality bounds with at most two expansions per state. In addition, SMHA* is potentially more powerful than IMHA* in avoiding depression regions as it can use a combination of partial paths found by different searches to reach the goal (Aine et al. 2016).

In SMHA* approach the optimal path for a given state is shared among all the searches so that if a better path to a state is discovered by any of the searches, the information is updated in all the priority queues. This allows the algorithm to expand each state at most twice, which significantly improves the computational time.

Our Planning Algorithm

The presented planning algorithm is based on MHA* Search with features of Hybrid A* search and adaptive motion primitives. It uses MHA* in a forward and backward manner as shown in Algorithm 1. For this purpose, multiple admissible and non-admissible heuristic functions are defined.

The framework has three main functions *SharedMultiHeuristicA**, *GeneratePath* and *CombinePath*.

*SharedMultiHeuristicA**: This function searches the configuration space according to the algorithm defined in (Aine et al. 2016). It is expanded with hybrid A* features and uses motion primitives. It is used in both forward and backward steps.

GeneratePath: This function executes the search using *SharedMultiHeuristicA** while continuously checking if the *EucledianDistance* from the current state to the goal state is greater than a configurable parameter d_fw. If the search has reached the closest state defined by d_fw then the function returns path from start position to the closest point to the goal.

CombinePath: Due to the discretization of the continuous space, an analytic function is required to combine the two paths generated by *GeneratePath* function. In this work, we used Reeds-Shepp curves for this purpose.

Bi-Directional Search The presented algorithm is using the *Bi-Directional Search*, by searching for the path in two steps. In the first step, the search expands in the forward direction (towards the goal) from the start state to reach the state close to the goal position. In the second step, the search proceeds backwards from the goal position towards the closest point reached by the forward search. The solution paths generated by forward search step and backward search step are then joined by the analytical expansion using Reeds-Sheep curves.

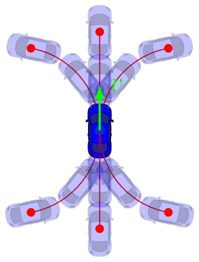
Figure 5: Motion Primitives.

Motion Primitives The motion primitives refer to the motion sequence that is triggered by an action request and correspond to a basic move that is possible by the vehicle, sampled from a continuous control space. In this work, motion primitives are generated by applying one of the six control actions defined by combinations of $\mathcal{U}_{v_x} = \{-1, 1\}$ and $\mathcal{U}_\alpha = \{-\alpha_\mathrm{max}, 0, \alpha_\mathrm{max}\}$. These represent, maximum steering left while driving in forward direction, no steering while driving in the forward direction, maximum steering right while driving in the forward direction, maximum steering left while driving in the backward direction, no steering while driving in the backward direction, maximum steering right while driving in the backward direction. Finer resolution is also possible, however that increases the branching factor and computational complexity.

As shown in Figure 5, each of these control actions is applied for a certain amount of time, resulting in an arc of a circle with a lower bound turning radius R_min. This will ensure that the resulting paths are always drivable, as the actual vehicle model is used to expand the state, even though they might result in excessive steering actions. An ***adaptive sizing of motion primitives*** is applied, wherein the arc length used for the execution of motion primitives is adapted dynamically to adjust to the environment. A shorter arc length is used near to obstacles and a longer arc length in free space. This approach improves the maneuverability in tight spaces. Using a shorter length in all cases promises higher levels of resolution completeness, as the likelihood to reach each state is increasing but reduces the computational efficiency.

Heuristics A heuristic function h is used to estimate the cost needed to travel from some state x to the goal state x_g (cost-to-go). As it is shown in (Hart, Nilsson, and Raphael 1968), if the heuristic function is underestimating the optimal cost-to-go, A* search provides the optimal solution. For the shortest path search, the usual heuristic function is the Euclidean distance.

In general, SMHA* algorithm supports n number of

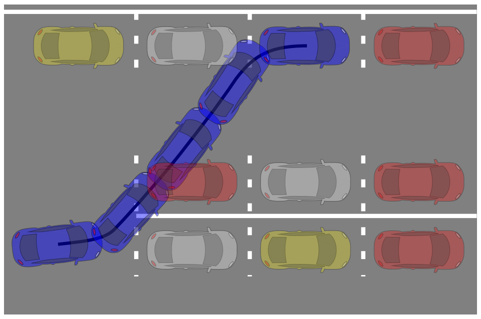

Figure 6: Heuristic function 1: non-holonomic without obstacles (Reeds-Shepp Curve).

heuristics with $n > 1$. In this work, to restrict the complexity and to be comparable with Hybrid A* which is used as a reference for benchmark, two heuristic functions have been used. The two heuristics capture different aspects of the problem as explained in sections below.

Non-Holonomic without Obstacles This heuristic function takes into account non-holonomic constraints of the vehicle while neglecting the influence of the environment (obstacles). The most suitable candidate functions are either Dubins or Reeds-Shepp curves. These curves are the paths of minimal length with an upper bound curvature for the forward, and combined forward and backward driving car respectively. We choose the Reeds-Shepp curves since in parking maneuvers it is important that the car can move in both forward and backward direction. These curves are computationally inexpensive to compute as they are based on an analytic solution. As shown in Figure 6, this heuristic takes into account the current heading as well as the turning radius, that ensures that the vehicle approaches the goal with the appropriate heading. This is especially important when the car gets closer to the goal. Given that Reeds-Shepp curves are minimal, this heuristic is clearly admissible.

Holonomic with Obstacles This heuristic function neglects the characteristics of the vehicle and only accounts for obstacles. The estimate is based on the shortest distance between the goal state and the state currently being expanded. This distance is determined using the standard Dijkstra search in two dimensions (x and y position). As the search is 2D and assumes the object under control is holonomic, the path is not smooth. The search is performed backwards, it uses the initial state of the SMHA* as the goal state, and the goal state of the SMHA* search as the start state to generate the heuristic cost. The closed list of the Dijkstra search stores all shortest distances to the goal and guides the vehicle away from dead ends and around obstacles. Since this heuristic function does not depend on any runtime sensor information, it can be fully pre-computed offline and used as a lookup table or simply translated and rotated to match the current goal instead of initiating a new search while SMHA* progresses.

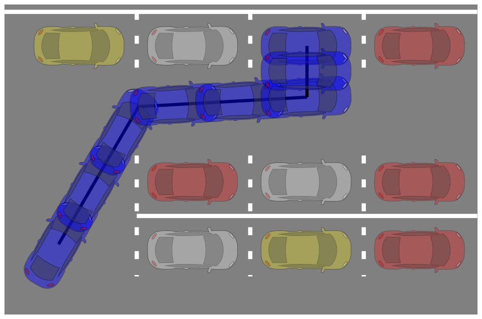

Figure 7: Heuristic function 2: holonomic with obstacles.

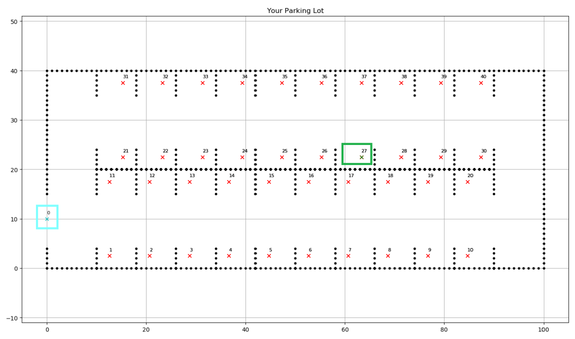

Figure 8: Parking Lot with Start Position (Cyan) and Goal Position (Green).

Simulation results

In order to benchmark the performance of the solution developed using SMHA*, we chose the Hybrid A* as a reference. As discussed earlier, Hybrid A* gives a comparable reference as it is also based on the orderly sampling approach and also uses two heuristics to guide the search. The key difference is that in the Hybrid A* approach, the maximum of both of the heuristics results is considered to update the priority queue while in SMHA* the heuristics are iteratively computed and both can update the priority queue.

The use cases chosen for the simulation depict common situations encountered in a parking lot such as *Entering Parking Lot* and *Exiting Parking Lot*. The simulations are performed by executing the Hybrid A* and SMHA* algorithm back-to-back to compare KPIs of the generated solution.

In the rest of the section, an elaborate analysis of the simulation results and solution paths that are generated for the use case *Entering Parking Lot* using *Bi-directional Search* for parallel parking configuration is presented. Figure 8 depicts the selected start and goal position (Parking Slot ID: 27) on the parking layout.

First, the configuration space is explored using the 2D Dijkstra search to generate the cost-to-go map as shown in Figure 9. The cost-to-go map shows in a color scale distance from the goal pose considering obstacles, but neglecting non-holonomic constraints. The results of this step are stored in a look-up and represents the *Holonomic with Ob-*

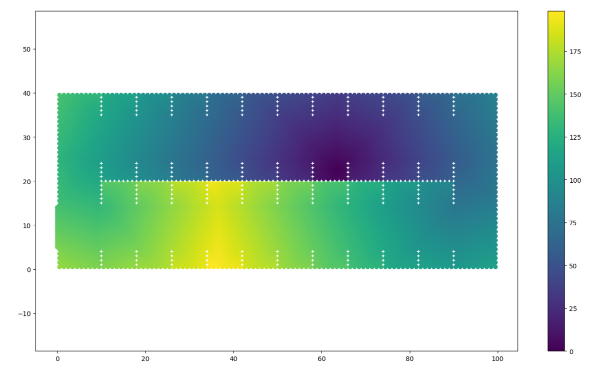
Figure 9: Cost-to-go map generated by 2D Dijkstra search.

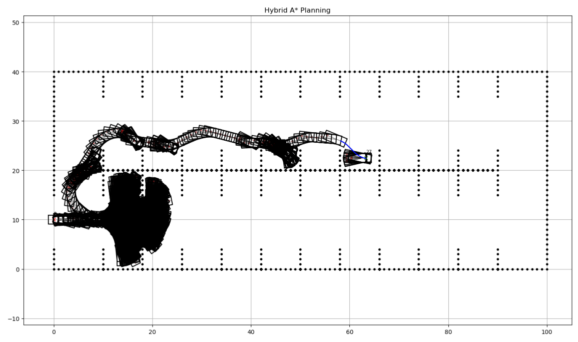

Figure 10: State Expansion by Hybrid A*.

stacles heuristic explained earlier.

Figure 10 depicts the state expansion pattern of the Hybrid A* algorithm. The algorithm has searched the area around the start position with a bias towards the goal position even though the solution path lies in another direction. The heuristic strongly guides the search towards the shortest path as far as possible within the obstacle free area. As the search explores, it expands all states with lower costs that can lead to the shortest path until it reaches a point where the heuristic cost of expanding the points which do not lead to the shortest path has a lower cost to reach the goal. As a result of this poor pruning of the unwanted branches effectively, the planner expands several states around the start position before it realizes the optimal direction of the path and leads to poor timing performance. The final path generated as seen in Figure 11 has many orientation changes in the *Forward Search* step and maneuvering step into the parking slot is determined by the backward search. The solution path is smooth, but the direction of orientation is reversed for the most part of the path which is not optimal.

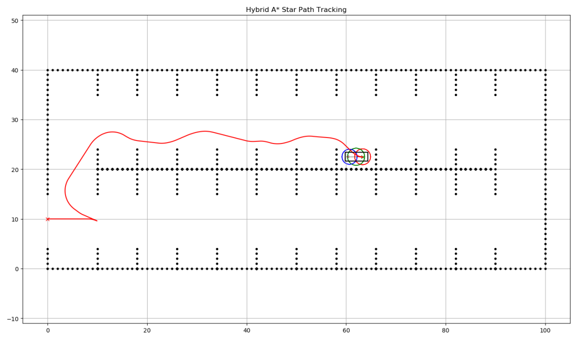

Figure 11: Path generated by Hybrid A*.

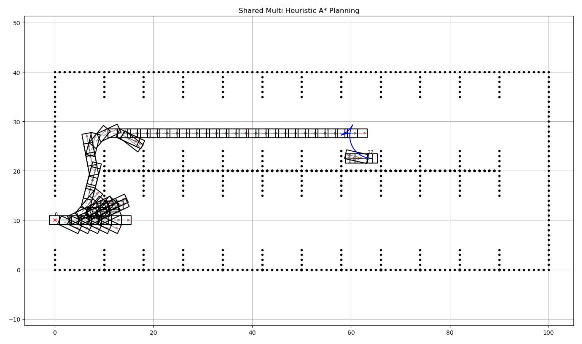

Figure 12: State Expansion by SMHA*.

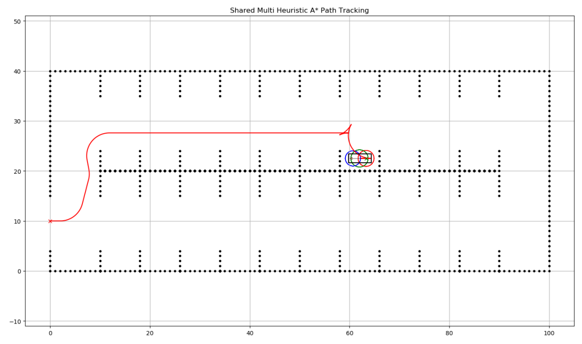

Figure 13: Path generated by SMHA*

Figure 12 depicts the state expansion pattern of SMH A* algorithm. Similar to Hybrid A*, the algorithm has searched the area around start position with a bias towards goal position even though the solution path lies in another direction. But, the multi-heuristic approach, quickly balances the bias towards finding the shortest path to finding a feasible path considering the obstacles. In addition, due to the mutually informed independent search by respective heuristics, the states that are expanded by one heuristic function are not expanded by other heuristic functions. As a result, the algorithm could prune the unwanted branches and realize the optimal direction of the path much faster compared to Hybrid A*.

As seen in Figure 13, the path is smooth and the direction of orientation is in the forward direction for the most part of the path which is preferred.

As seen from KPI values tabulated in Table 1, the Hybrid A* algorithm expands significantly more states and uses more time to generate the solution path compared to SMHA* approach. Even though the heuristics used are the same, the mutually informed independent search of SMHA*

KPI	Hybrid A*	SMHA*
Number of Expanded States	2457	73
Execution Time (s)	47.8	11.51
Path Length (m)	90.3	90.58
Reverse Path Length (m)	7.29	3.29
Direction Changes	4	4
Number of Iterations	23633	73

Table 1: Entering parking lot with Bi-Directional Search - KPI Comparision.

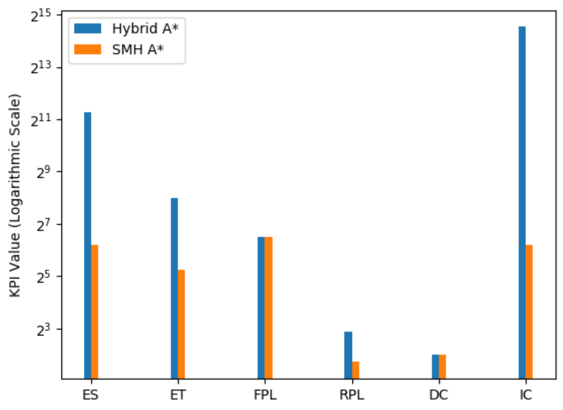

Figure 14: Entering Parking Lot with Bi-Directional Search - KPI Comparision (Expanded States (ES), Execution Time (ET), Forward Path Length (FPL), Reverse Path Length (RPL), Direction Changes (DC), Iteration Count (IC)).

prunes the unwanted branches much more significantly allowing faster convergence towards the solution and improved execution time.

To give a comprehensive performance comparison of both the algorithms for the use case *Entering Parking Lot* for parallel parking lot layout using *Bi-Directional Search*, a full simulation run through all the parking slots is executed. In this simulation mode, each parking slot ID is selected as a goal position sequentially and the back-to-back run of Hybrid A* and SMHA* algorithm is performed.

As observed from Figure 15, SMHA* outperforms the Hybrid A* algorithm in terms of both performance and solution path quality parameters. With the multi heuristic approach, the average execution time to generate the solution path is reduced by **81%**, which is a significant improvement demonstrating the potential of multi heuristic approach to solve the given planning problem.

Conclusion

The work focused on providing a Multi-Heuristic search based approach to solve the motion planning problem for autonomous parking. To benchmark the results obtained, a state-of-the-art planning algorithm Hybrid A* was chosen as reference.

As the environment for the given use case involves only low speed maneuvering, a Single Track Bicycle Model which reflects the kinematics of the vehicle was used as a

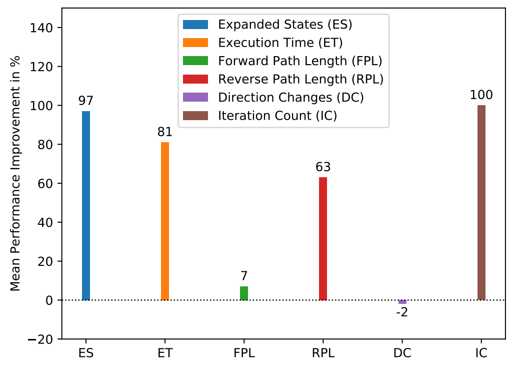

Figure 15: Entering Parking Lot with Bi-Directional Search - Mean Performance Improvement.

motion model. The model emulates the non-holonomic nature of the vehicle in all stages of the algorithm, motion primitives (node expansion) and heuristic estimates. Thus, the paths generated are always driveable.

A collision check algorithm was implemented based on the Multi-Disk Decomposition of the bounded volume. The algorithm was parameterized based on the vehicle geometry making it a generic solution to fit with any vehicle type and independent of the environment.

The path planning problem was solved using the Shared Multi-Heuristic A* approach in which two heuristic functions were implemented with a round robin scheduling. The respective heuristic searches share the current path obtained to a state. The heuristics were defined to capture the non-holonomic and holonomic constraints of the vehicle. Two solution approaches, *Forward Search* and *Bi-Directional Search* were developed.

The SMH A* algorithm solved the motion planning problem elegantly and outperformed Hybrid A* with respect to response time and path quality of the generated solution path. The KPI comparison clearly indicates that SMHA* is an ideal candidate for motion planning in slow speed driving in unstructured environments applications like autonomous valet parking.

Acknowledgments

VIRTUAL VEHICLE Research Center is funded within the COMET Competence Centers for Excellent Technologies programme by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry of Science, Research and Economy (BMWFW), the Austrian Research Promotion Agency (FFG), the province of Styria and the Styrian Business Promotion Agency (SFG). The COMET programme is administrated by FFG.

References

Aine, S.; Swaminathan, S.; Narayanan, V.; Hwang, V.; and Likhachev, M. 2016. Multi-Heuristic A*. *The International Journal of Robotics Research* 35(1-3):224–243.

Ajanović, Z.; Lacevic, B.; Shyrokau, B.; Stolz, M.; and Horn, M. 2018. Search-Based Optimal Motion Planning for Automated Driving. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4523–4530. IEEE.

Ajanović, Z.; Regolin, E.; Stettinger, G.; Horn, M.; and Ferrara, A. 2019. Search-Based Motion Planning for Performance Autonomous Driving. In *IAVSD Vehicles on Road and Tracks 2019*.

Ajanović, Z.; Stolz, M.; and Horn, M. 2018. A novel model-based heuristic for energy-optimal motion planning for automated driving. *IFAC-PapersOnLine* 51(9):255–260.

Banzhaf, H.; Nienhüser, D.; Knoop, S.; and Zöllner, J. M. 2017a. The future of parking: A survey on automated valet parking with an outlook on high density parking. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, 1827–1834.

Banzhaf, H.; Palmieri, L.; Nienhuser, D.; Schamm, T.; Knoop, S.; and Zollner, J. M. 2017b. Hybrid curvature steer: A novel extend function for sampling-based nonholonomic motion planning in tight environments. In Conference, I. I. T. S., ed., *IEEE ITSC 2017*, 1–8. Piscataway, NJ: IEEE.

Becker, J.; Colas, M.-B. A.; Nordbruch, S.; and Fausten, M. 2014. Bosch's vision and roadmap toward fully autonomous driving. In *Road vehicle automation*. Springer. 49–59.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Dolgov, D.; Thrun, S.; Montemerlo, M.; and Diebel, J. 2008. Practical search techniques in path planning for autonomous driving. *Ann Arbor* 1001(48105):18–80.

Dubins, L. E. 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics* 79(3):497–516.

Frazzoli, E.; Dahleh, M. A.; and Feron, E. 2002. Real-Time Motion Planning for Agile Autonomous Vehicles. *Journal of Guidance, Control, and Dynamics* 25(1):116–129.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Karaman, S., and Frazzoli, E. 2010. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI* 104(2).

Kingston, Z.; Moll, M.; and Kavraki, L. E. 2018. Sampling-Based Methods for Motion Planning with Constraints. *Annual Review of Control, Robotics, and Autonomous Systems* 1(1):159–185.

Kuwata, Y.; Karaman, S.; Teo, J.; Frazzoli, E.; How, J. P.; and Fiore, G. 2009. Real-Time Motion Planning With Applications to Autonomous Urban Driving. *IEEE Transactions on control systems technology* 17(5):1105–1118.

LaValle, S. M.; Branicky, M. S.; and Lindemann, S. R. 2004. On the Relationship between Classical Grid Search and Probabilistic Roadmaps. *The International Journal of Robotics Research* 23(7-8):673–692.

LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning.

Likhachev, M., and Ferguson, D. 2009. Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. *The International Journal of Robotics Research* 28(8):933–945.

Liu, S.; Mohta, K.; Atanasov, N.; and Kumar, V. 2018. Search-based motion planning for aggressive flight in se (3). *IEEE Robotics and Automation Letters* 3(3):2439–2446.

Lozano-Pérez, T., and Wesley, M. A. 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM* 22(10):560–570.

M. Pivtoraiko, and A. Kelly. 2005. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3231–3237.

Mandalika, A.; Salzman, O.; and Srinivasa, S. 2018. Lazy receding horizon A* for efficient path planning in graphs with expensive-to-evaluate edges. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Melissen, J. B. M., and Schuur, P. C. 2000. Covering a rectangle with six and seven circles. *Discrete Applied Mathematics* 99(1-3):149–156.

Nemec, D.; Gregor, M.; Bubeníková, E.; Hruboš, M.; and Pirník, R. 2019. Improving the Hybrid A* method for a non-holonomic wheeled robot. *International Journal of Advanced Robotic Systems* 16(1):172988141982685.

Ranganeni, V.; Chintalapudi, S.; Salzman, O.; and Likhachev, M. 2020. Effective footstep planning using homotopy-class guidance. *Artificial Intelligence* 103346.

Reeds, J., and Shepp, L. 1990. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics* 145(2):367–393.

Röger, G., and Helmert, M. 2010. The more, the merrier: Combining heuristic estimators for satisficing planning. In *Twentieth International Conference on Automated Planning and Scheduling*.

S. Karaman; M. R. Walter; A. Perez; E. Frazzoli; and S. Teller. 2011. Anytime Motion Planning using the RRT*. In *2011 IEEE International Conference on Robotics and Automation*, 1478–1483.

Youakim, D.; Cieslak, P.; Dornbush, A.; Palomer, A.; Ridao, P.; and Likhachev, M. 2020. Multirepresentation, Multiheuristic A* search-based motion planning for a free-floating underwater vehicle-manipulator system in unknown environment. *Journal of field Robotics*.

Zhang, X.; Liniger, A.; and Borrelli, F. 2020. Optimization-based collision avoidance. *IEEE Transactions on control systems technology*.

Ziegler, J., and Stiller, C. 2010. Fast collision checking for intelligent vehicle motion planning. In *2010 IEEE intelligent vehicles symposium (IV 2010), La Jolla, California, USA, 21-24 June 2010*, 518–522. Piscataway: IEEE.