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

Bhargav Adabala\textsuperscript{1}, Zlatan Ajanovic\textsuperscript{1}
\textsuperscript{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.

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-
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 overdrives 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 ID. The number of parking slots and dimensions of the open space is configurable.
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 $\alpha$ 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 $x = (x, y, \theta)$ denote the configuration state of the car-like robot, where $x$ and $y$ denote the position and $\theta$ 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:

\begin{align}
\dot{x} &= v_x \cos(\theta) \\
\dot{y} &= v_x \sin(\theta) \\
\dot{\theta} &= \frac{v_x}{L} \tan(\alpha)
\end{align}

Setting the maximum steering angle $|\alpha| \leq \alpha_{\text{max}}$ results in a minimum turning radius $R_{\text{min}}$. The model clearly represents the non-holonomic behaviour as it is impossible to move sideways without violating the non-slip condition. Restricting the allowed velocities and steering angles to the finite set $U_{v_x} = \{-1, 0, 1\}$ and $U_{\alpha} = \{-\alpha_{\text{max}}, 0, \alpha_{\text{max}}\}$ results in the Dubins car that can only stop and move forward at unit speed (Dubins 1957) and setting it to $U_{v_x} = \{-1, 0, 1\}$ 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:

\[ r = \sqrt{\frac{l^2}{n^2} + \frac{w^2}{4}} \]  

placed at a distance of $d$ calculated as:

\[ d = 2\sqrt{r^2 - \frac{w^2}{4}} \]

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 manuevering 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 Parameters**

![Figure 3: Single-track vehicle model.](image-url)
Several admissible heuristic functions enable efficient optimization. Kinodynamic feasibility of the solution is provided by motion primitives based on the vehicle model. Several admissible heuristic functions enable efficient optimization.

**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.

**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 practical 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 \( \theta \) 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 algorithms. 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 \( u \in \delta(x) \). 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
The presented algorithm is using Reeds-Shepp curves for this purpose. It is used in both forward and backward motion primitives. It uses MHA* in a forward and backward manner as shown in Algorithm 1. This function executes the search using SMHA* features and uses motion primitives. It is used in both forward and backward steps.

**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 $U_{\alpha_{\text{max}}} = \{-1, 1\}$ and $U_{\tau_{\text{min}}} = \{-\alpha_{\text{max}}, 0, \alpha_{\text{max}}\}$. 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, but 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 $R_{\text{min}}$. 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...
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.

**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 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 11: Path generated by Hybrid A*.

Figure 10: State Expansion by Hybrid A*.

Figure 12: State Expansion by SMHA*.

Figure 13: Path generated by SMHA*

**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 12 depicts the state expansion pattern of SMHA* 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*
Table 1: Entering parking lot with Bi-Directional Search - KPI Comparision.

<table>
<thead>
<tr>
<th>KPI</th>
<th>Hybrid A*</th>
<th>SMHA*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Expanded States</td>
<td>2457</td>
<td>73</td>
</tr>
<tr>
<td>Execution Time (s)</td>
<td>47.8</td>
<td>11.51</td>
</tr>
<tr>
<td>Path Length (m)</td>
<td>90.3</td>
<td>90.58</td>
</tr>
<tr>
<td>Reverse Path Length (m)</td>
<td>7.29</td>
<td>3.29</td>
</tr>
<tr>
<td>Direction Changes</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Number of Iterations</td>
<td>23633</td>
<td>73</td>
</tr>
</tbody>
</table>

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.

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.

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

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 (BMWFV), 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


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*.


