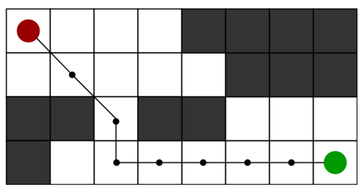
A Cost-Effective Approach to Smooth A* Path Planning for Autonomous Vehicles

Introduction

For path planning of wheeled mobile robots, grid based graph search algorithms are effective in unstructured environments.
These usually do not consider vehicle dynamics, leading to paths that have to be smoothed and may turn out to be infeasible.



Basic A*

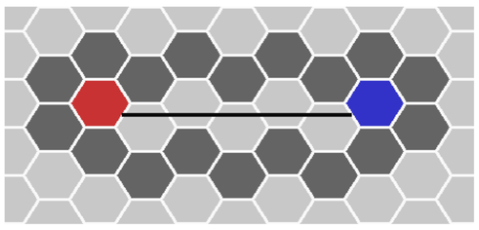


Fig. 1: Moving horizontally in a hex. grid produces a zig-zag in the grid cell formation. Nevertheless, a straight path can be planned within these grid cells.

Idea:
Adapt the A* algorithm to consider motion constraints from the beginning

We define a set of feasible motion primitives represented as sequences of grid cells. Each primitive has a specific cost, depending on the curvature of a smooth path that may be planned inside the grid cells. For example, in Fig. 1, a cell sequence for moving horizontally consists of multiple changes of directions in the grid. However, the path, can be planned straight.
For a specific cell size-to-curvature ratio, there is a unique set of feasible primitives.

Grid based path planning with limited curvature

We apply our approach on a hexagonal rather than an orthogonal grid. This way, turns in the solution path can have angles of 60° or 120°. However, depending on the cell size *rc*, a vehicle with limited turning *rmin* radius may still be unable to follow this path. Consider 3 cases:

Case 1: *rmin/ rc < 1:* the vehicle can fully turn within one cell. Any formation is feasible.

Case 2: *rmin/ rc ≤ √7 :* a full turn of the vehicle fits into 6 cells forming a circle. The grid based path is feasible if it does only contain turns of 60 deg and straight lines.

Case 3: *rmin/ rc ≤ 3.33 :* a full turn of the vehicle fits into 12 cells forming a circle. The grid based path is feasible if it includes only turns of 60°followed by a straight line or a turn of 60 deg in the other direction. Feasible paths are shown in Fig. 2.

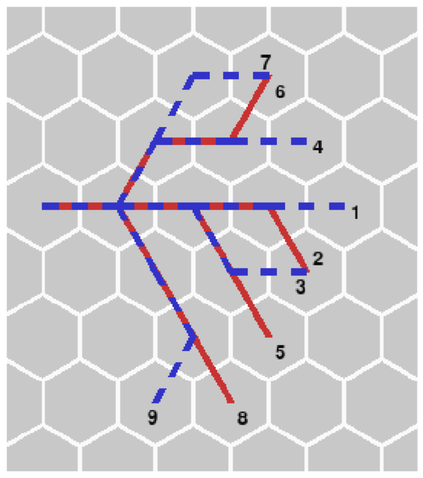


Fig. 2: Path primitives with a length of n = 5 cells representing all admissible path formations regarding the curvature constraint. Each primitive starts on the left and ends at their respective number.

Trade-off cell size:
- Large cells (Case 1) do not require restrictions on the cell formation. This, and the small number of cells, lead to low computational effort.
- Small cells (Case 3) are able to represent the environment more accurately. This is especially important for small obstacles and narrow pathways that may vanish in a coarse grid completely.

The algorithm in detail

The A* algorithm is adapted to only add cells that are feasible regarding the curvature constraint (i.e. that are part of the defined motion primitives). The adapted algorithm summarized:
Step 1: Create the "open-list" composed of all cells adjacent to the start cell. For each entry, store paths leading to the respective cell and their curvature dependent cost.
Step 2: From all paths of all cells in the open-list, select the cell with the cheapest path. This cheapest path is from now on considered a closed-path. If a cell has no more open paths, it is considered a closed cell.
Step 3: Append the selected cell to the cheapest path leading to that cell.
Step 4: Based on the cell formation of the cheapest path, determine admissibility of adjacent cells.
Step 5: Add all admissible cells to the open list. For each of these cells, the cheapest path is extended by that cell stored together with the corresponding curvature dependent cost.
Step 6: Unless one of the new cells is the targe cell, continue with step 2.
Step 7: The target cell is in the open-list with corresponding paths leading from the start cell to the target cell. Select the path with the lowest cost.

Color	Cell type
light grey	free cell
dark grey	occupied cell
red	start cell
blue	target cell
green	open cell
pink	closed cell

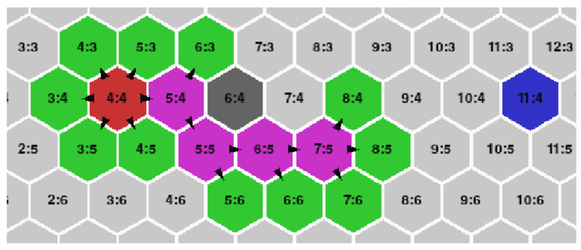


Fig. 3: Exemplary state of the modified A* algorithm. Cells are added to the open-list (green) only when the curvature constraints are satisfied.

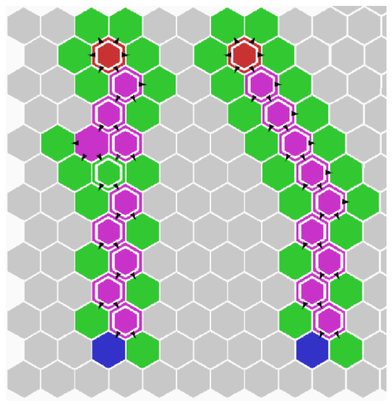


Fig. 4: Algorithm performing on an open map, planning straight lines in different angles.

Cost Function

For each open cell, a cost is computed. The cost c is [3]:

$$c = cc + cg \quad (1)$$

The cost-to-come *cc* is an estimate of the lowest cost to get from the start cell to the open cell. In our approach it is the weighted sum of the number of cells of the cheapest path $n*c$ and a cost c_K corresponding to the curvature of the path primitive leading to the open cell.

$$cc = w_n \cdot n*c + w_K \cdot c_K \quad (2)$$

For the cost c_K, for each path primitive, a fixed cost value is precomputed as described later.
The cost-to-go is approximated by the euclidean distance to the target cell.
By weighing the different parts of the cost function, the solution of the cheapest path will change and also the trade-off between exploration and exploitation can be tuned.

We demonstrate 3 variants for computing the curvature cost c_K. The resulting cost values are shown in Tab. II.

A) Ribbon
A ribbon model based path planning algorithm [12] is used to plan a smooth path within the cell area of each primitive. The implemented heuristic aims on minimizing the curvature of the planned path. The cost value c_K for each primitive is the median of the planned path curvature.

B) Adapted Ribbon
For the ribbon model based path planner, the start position and orientation is fixed, while the goal position and orientation is free. This leads to different costs when a primitive is paced in one direction or the other. Cost values are adpted to eliminate this effect.

C. Curvature Penalty
Costs are set manually for each primitive to penalize large curvature and changes of direction. The cost values of primitives 5, 7 and 9 (as labeled in Fig. 2) represent the largest curvature and where therefore set to maximum.

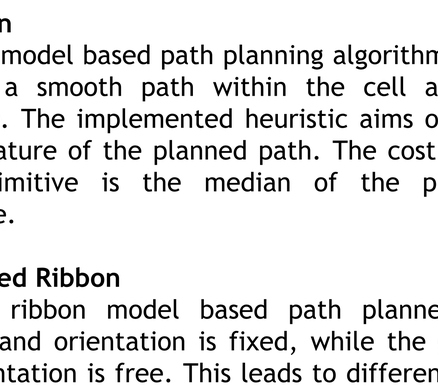


Fig. 5: To escape a narrow space while satisfying the curvature constraint, the algorithm creates a turning maneuver.

TABLE II: Curvature costs c_K for each path primitive

ID	Ribbon	Adapted Ribbon	Curvature Penalty
1	0.0	0.0	0.0
2	0.087	0.087	0.1
3	0.119	0.119	0.2
4	0.195	0.119	0.2
5	0.429	0.429	1.0
6	0.109	0.0	0.0
7	0.429	0.429	1.0
8	0.507	0.087	0.1
9	0.915	0.915	1.0

Evaluation

The proposed algorithm achieves 3 characteristics:
Straight paths in any direction: when the grid axis are aligned in a different direction than the shortest path, the resulting grid solution involves changes of direction and the shortest path by means of number of grid cells is not unique. Nevertheless, the proposed algorithm finds the cell formations where a straight path can be planned. In Fig. 4 the algorithm planning different angled straight lines in an open map is shown.
Maximum curvature constraint: additionally to the cost function, the maximum curvature is a hard constraint. Also in presence of sharp edges, the planned path will follow the spedified minimum turning radius. An example is shown in Fig. 5.
Tuning behavior: By setting different weights in the cost function, the path can be tuned for short length, small curvature or small number of changes of direction. In Fig. 7 the results of 3 different curvature cost variants are shown.

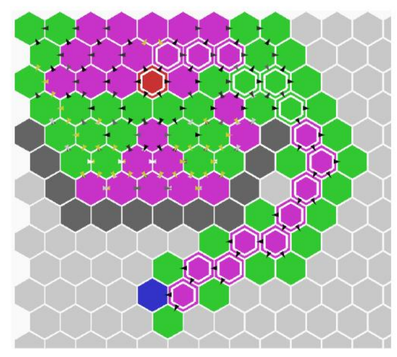
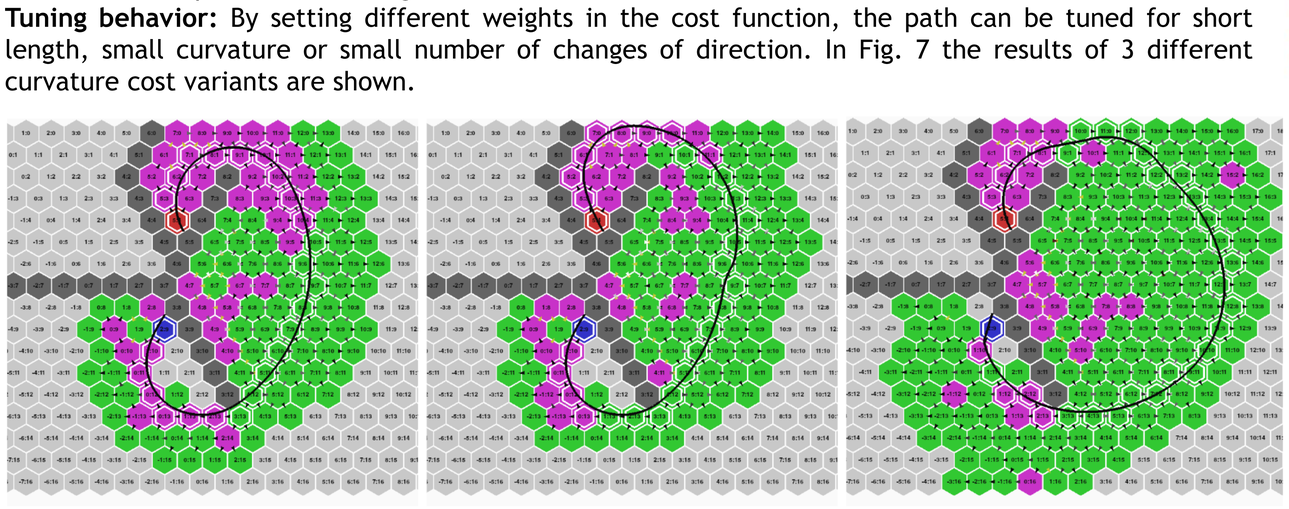


Fig. 6: When the path directed to the target cell is blocked, the number of iterations increases as multiple paths are planned in each cell.



(a) Ribbon based cost (Sec. IV-A) (b) Adapted ribbon based cost (Sec. IV-B) (c) Manual cost distribution (Sec. IV-C)

Fig. 7: Resulting grid based paths and smooth paths with different curvature cost distributions.

Conclusion

The A* algorithm is widely used for path planning in unstructured environments. To handle car-like vehicles and dynamic constraints, a two-stage approach can be effective: first, a grid-based path is planned, then a smooth path is generated. Challenges include grid accuracy and ensuring a smooth path within grid constraints.
The authors propose modifying A* by adding curvature constraints and a cost function for path smoothness. This guarantees drivable paths, even with small grid cells, and adapts well to various grid orientations. Though more computationally complex, the method effectively produces smooth, feasible paths in diverse environments.

Virtual Vehicle Research GmbH has received funding within COMET Competence Centers for Excellent Technologies from BMK, BMAW, the Province of Styria and the Styrian Business Promotion Agency (SFG).
The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

Federal Ministry Republic of Austria Climate Action, Environment, Energy, Mobility,
Federal Ministry Republic of Austria Labour and Economy
FFG Promoting Innovation.
Das Land Steiermark Wirtschaft, Tourismus, Wissenschaft und Forschung
SFG