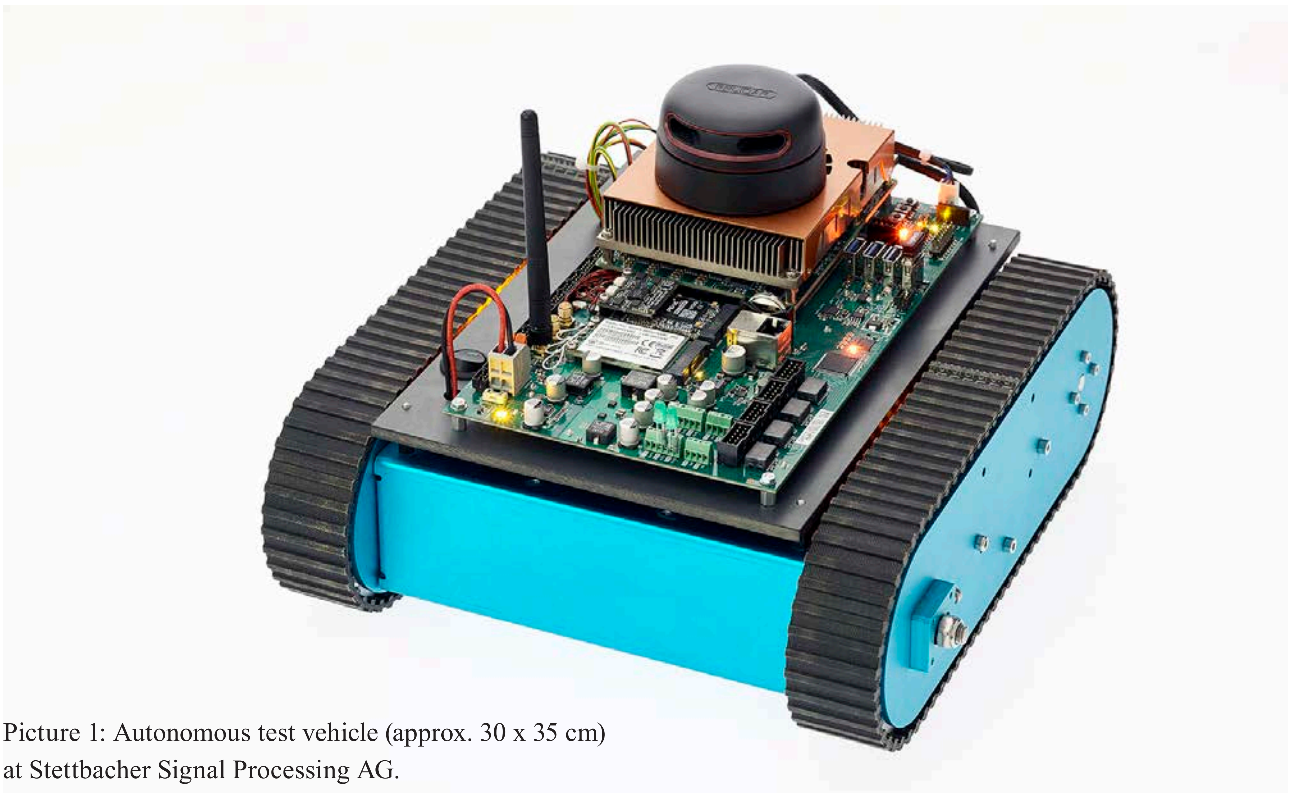
STETTBACHER SIGNAL PROCESSING
beyond Limits

www.stettbacher.ch

Autonomous Systems

How Are They Finding Their Way?

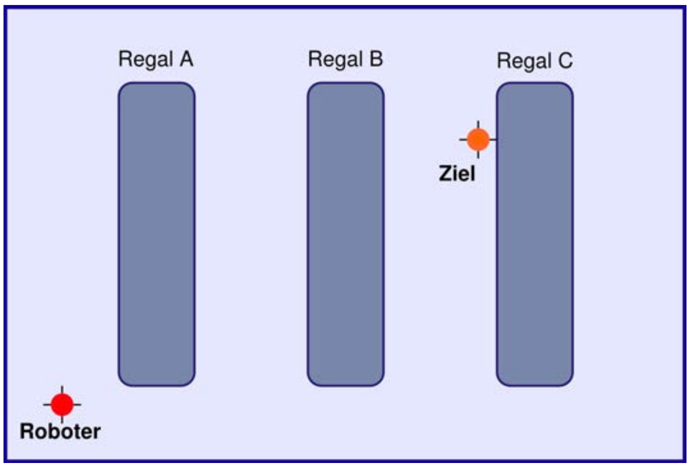
Reports of successes and setbacks in connection with self-driving passenger cars, buses, trains and other autonomous systems are currently overflowing in magazines and daily newspapers. In the shadow of these much-appreciated engineering achievements. However, a multitude of less spectacular systems have long since emerged, such that move autonomously and independently search for the right path, avoid obstacles, and so on. The most important aspects of autonomous driving are highlighted here from a technical perspective, especially finding the way.


Picture 1: Autonomous test vehicle (approx. 30 x 35 cm) at Stettbacher Signal Processing AG.

In order for a vehicle to drive autonomously, some core functionalities with corresponding algorithms are necessary: Determining the current position and orientation, calculating a path from the current location to a destination point, following that path to the destination, dealing with unexpected obstacles, and, of course, driving and controlling the motors and steering accordingly. Clearly, determining one›s position is of primary importance. In many applications, a single position sensor (for example, a GPS receiver outdoors or UWB indoors) alone is not sufficient to provide a satisfactory, sufficiently dynamic and precise position estimate. Instead, other quantities are measured and used, for example the speed and acceleration of the vehicle, the compass direction and its change, etc. Using sensor fusion (for example through a Kalman filter), a joint estimate is generated from these. In addition, it is assumed that an autonomous vehicle is able to detect obstacles in the environment in order to react accordingly, for example by stopping or avoiding them. For such anti-collision systems, one also does not normally want to rely on a single sensor. An ultrasonic system is not suitable for long distances, radar can miss very small obstacles such as a wire mesh, a laser distance measurement system (lidar) weakens in rain, and an optical system with cameras cannot achieve good results in fog or direct sunlight. The weaknesses and strengths of the individual sensors can be combined into a robust and reliable overall system using appropriate algorithms. Another important basic function of an autonomous vehicle is, of course, approaching a target. For most industrial applications, it is not sufficient to move in the direction of the target point and avoid obstacles on the way. Instead of this rather random strategy, it is often desired to find the (or an) optimal path from one›s position to the destination, taking known obstacles into account. The so-called Path Planner is responsible for this.

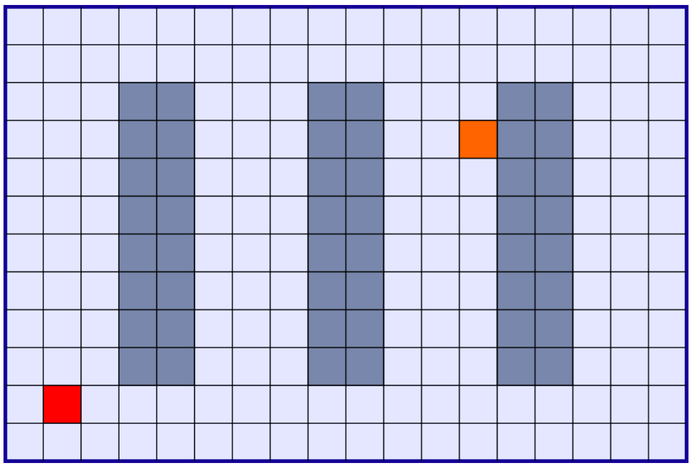
Elementary Path Planner

Its task is therefore to calculate an optimal route from a starting point to the destination. What is optimal is defined by a certain criterion, for example, that the route should have minimum gradients, minimum fuel consumption, no too tight curves, or something similar. Often, however, the shortest route is simply sought. Known obstacles are marked on a map and made available to the algorithm. To illustrate this, imagine a warehouse in which an autonomous robot is moving. The robot is supposed to drive on the shortest way to a certain place at shelf C. The robot is supposed to find the shortest way to the shelf C. A corresponding map is shown in Picture 2. Note that the shelves represent obstacles.



Picture 2: Floor Plan of a Warehouse with Shelves.

One strategy to solve this problem starts from a discretized map (see
Picture 3). Here, a grid of cells is placed over the map. Then a weighting function is defined. This function is then used to iteratively calculate some sort of cost value for each cell. Each cost number is noted directly in the respective cell. If we want to find the shortest way from the starting point to the destination, we choose as cost of a field sensibly just the number of fields which have to be crossed to get from the starting point to the field. We count the distance between two horizontally or vertically adjacent fields as one unit. For diagonally adjacent fields, we count the distance as 1.4 units, which is a simple approximation to the Euclidean distance (the square root of two).


Picture 3: Discretized map of the warehouse

To calculate the cost function for all cells to the destination, proceed as follows:
1. all drivable cells are initialized to the cost value infinite.
2. all accessible cells are marked as not yet visited (white).
3. the start cell receives the cost value 0.

Repetitively, the costs of the nodes are reduced according to the following scheme.

4. select among the not yet visited cells the one with the lowest cost value and mark it temporarily (yellow). If there are several candidates with the same lowest cost value, select one of them.
5. Calculate the cost function for all immediate neighbor cells that have not yet been visited (temporarily green).
6. For each neighbor cell, compare the new cost value with the existing one and take the profound one of the two.
7. Mark the current cell as visited (blue).
8. Repeat points 4 to 7 until the target cell is marked as visited.

Now the shortest path can be read out.

9. select the cell at the destination as the starting point.
10. Among the neighboring cells marked as visited, select the cell with the smallest cost value as successor in the path. Mark it (red border). If there are several candidates with the same lowest cost value, select one of them.
11. Repeat point 10 for each successor cell until the starting point is reached.

By the way, the algorithm just described is not new. It was published by Edsger Dijkstra in 1959, when there was hardly any talk of autonomous vehicles, and is now part of modern graph theory.

In pictures 4 through 12, the method is applied to our example map. Picture 4 shows the initialized map. The starting point is marked as selected (yellow). The cost values of its neighbors (green) are reduced in the first iteration. Picture 5 shows the new cost values.

This completes the first iteration; the starting cell is marked as visited (blue). Now the next cell is selected, namely one that is not yet visited and of it the one that currently has the smallest cost value. Picture 6 shows that the cell to the left of the starting point was selected. There would have been three more candidates with the same cost value. The choice is random. Again, the green boxes denote those that were updated in the second iteration. Picture 7 shows the state after the third iteration. The following pictures 8 and 9 show how the blue area of the fields already visited slowly spreads in all directions. In picture 10, the target cell is reached and in picture 11, the target is marked as visited. This is the end of the algorithm. In picture 12 the searched path is marked. It follows backwards from the target to the start, along the largest gradient.

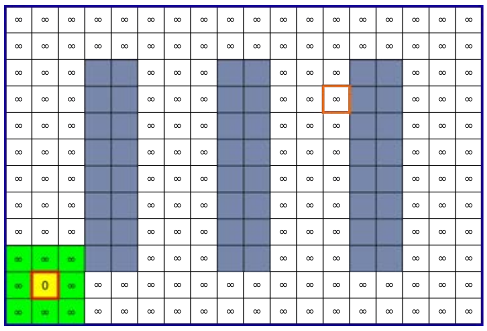
Corrections

Although this works wonderfully, Dijkstra›s algorithm has a problem: depending on the resolution and size of the map, the method can require an extremely high number of iterations and is consequently poorly suited for real-time systems. For this reason, there are several variations of it.

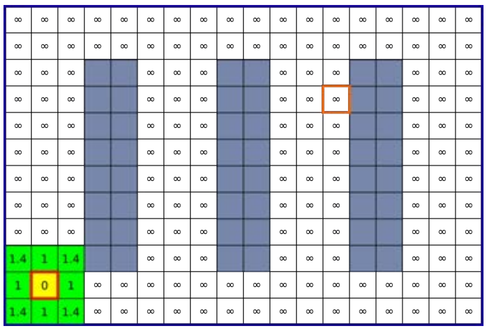
The so-called A* method primarily improves the efficiency. For this purpose, the cost function (distance of a field from the starting point) is extended by a heuristic estimate of the distance of the field to the target. The two distances are added together. This results in the cost value of cells leading away from the target increasing and consequently not being pursued immediately. Only when the seemingly direct paths do not lead to the target, the algorithm resorts to these cells.
The D* algorithm is particularly useful for autonomous systems. It does not start at the starting point and search for the shortest path to the goal, but the other way around: it searches for the shortest path from the goal to the starting point. In essence, it uses the A* method. If the starting point changes because the vehicle moves and suddenly an obstacle occurs that was not known before, the entire route does not have to be recalculated. Instead, the cost map is only partially dated in the vicinity of the obstacle. From the obstacle to the destination, the cost values are retained. This allows quick rescheduling and reaction to unforeseen changes in the map.

Conclusion

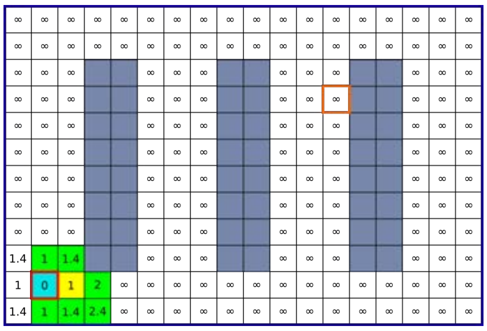
Path planning is indispensable for autonomous systems and is an essential component of intelligent vehicles. Especially in comparison to systems without this capability, which often appear to be wandering aimlessly, it quickly becomes clear how important such a procedure is in practice. However, the computational effort is not to be neglected. Moreover, the Path Planner has to cope with unexpected obstacles or changes in the map. Different variants of the classical Dijkstra algorithm take these needs into account.


Picture 4: Initialized map. The starting point has been selected (yellow) and its neighbors are marked (green).

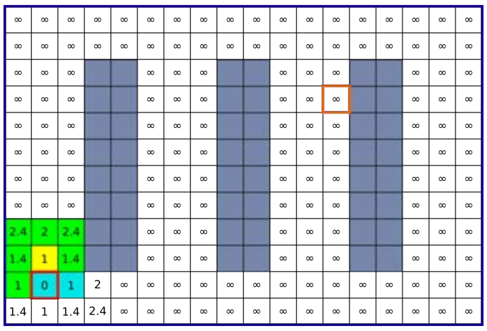
Picture 5: Map after the first iteration. The cost values of the neighbors were calculated and updated.



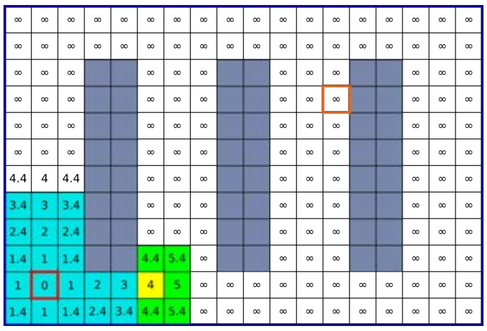
Picture 6: Map after the second iteration with cost values dated on.



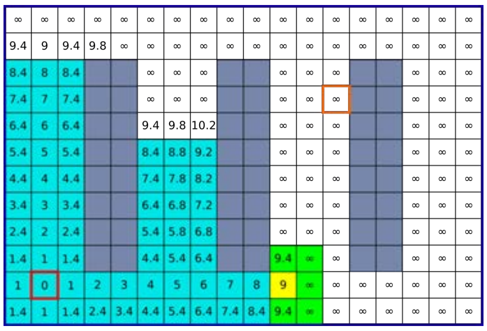
Picture 7: Map after the third iteration with cost values dated on.



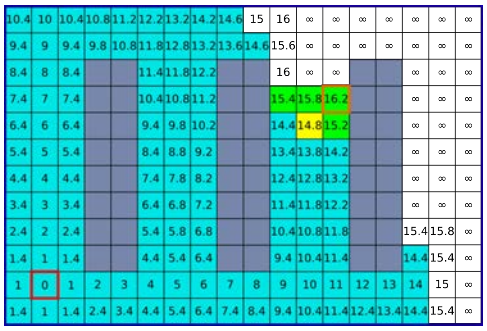
Picture 8: Map after 20 iterations.



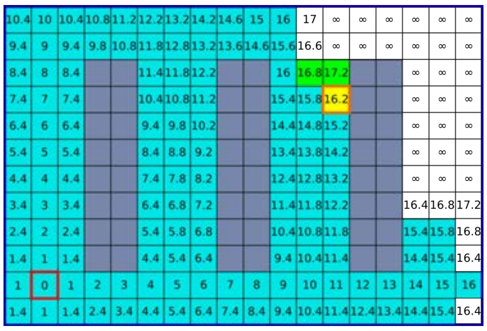
Picture 9: Map after 60 iterations.



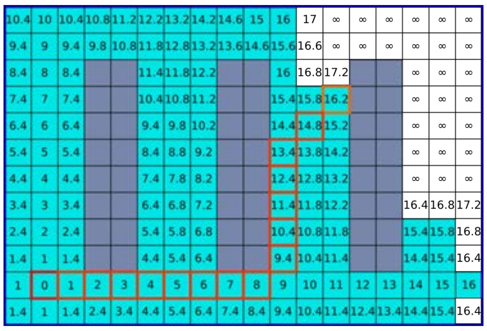
Picture 10: Map after 117 iterations. The procedure reaches the goal.



Picture 11: Map after 131 iterations. The target is now immediately marked as visited.



Picture 12: The target has been marked as visited. The algorithm has reached the end. The path is read out.



Individuelle Lösungen

Stettbacher Signal Processing AG bietet seit 20 Jahren F+E Dienstleistungen an für anspruchsvolle Projekte in den Bereichen elektronische Mess-, Steuer-, Regelungs-, Antriebs- und Kommunikationstechnik für industrielle Analytik, Qualitätssicherung, Medizin, Pharma, Verteidigung und Training. Die Firma setzt die O-3000 Kameras in eigenen Projekten ein und vertreibt sie erfolgreich auf dem Markt.

Stettbacher Signal Processing AG

dsp@stettbacher.ch
www.stettbacher.ch
+41 43 299 57 23

Neugutstrasse 54
CH-8600 Dübendorf