STETTBACHER SIGNAL PROCESSING
beyond Limits

www.stettbacher.ch

Autonome Systeme

Wie finden die eigenlich den Weg?

In Zeitschriften und Tageszeitungen überschlagen sich derzeit Meldungen über Erfolge und Rückschläge im Zusammenhang mit selbstfahrenden Personenwagen, Bussen, Bahnen und anderen autonomen Systemen. Im Schatten dieser viel gewürdigten Ingenieursleistungen ist aber längst eine Fülle von weniger spektakulären Systemen entstanden, die sich autonom bewegen und selbständig den richtigen Weg suchen, Hindernissen ausweichen, usw. Die wichtigsten Aspekte des autonomen Fahrens werden hier aus technischer Sicht beleuchtet, insbesonders die Wegfindung.

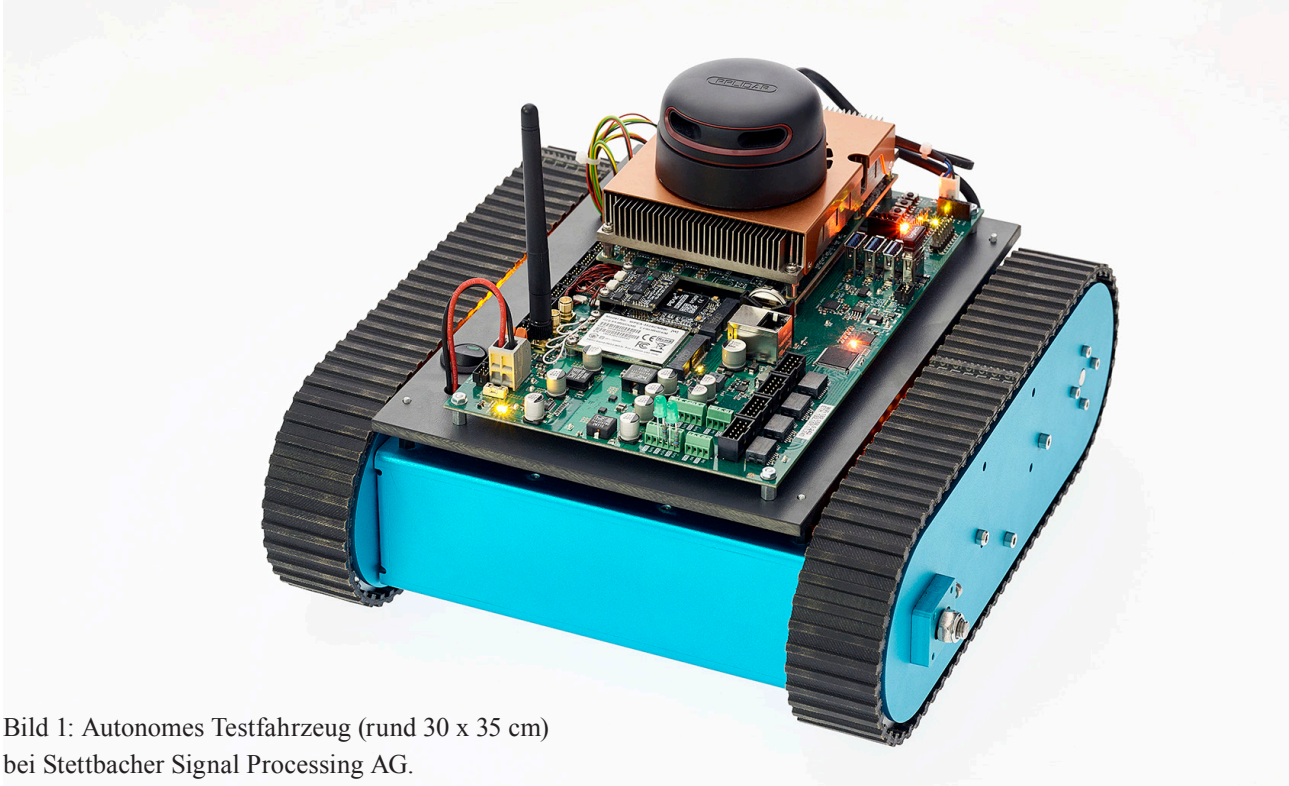


Bild 1: Autonomes Testfahrzeug (rund 30 x 35 cm)
bei Stettbacher Signal Processing AG.

Damit ein Fahrzeug autonom fahren kann, sind einige Kernfunktionalitäten mit entsprechender Algorithmik notwendig: Die Bestimmung der aktuellen Position und Ausrichtung, das Berechnen eines Weges vom gegenwärtigen Ort zu einem Zielpunkt, das Verfolgen dieses Weges zum Ziel, der Umgang mit unerwarteten Hindernissen und natürlich die entsprechende Ansteuerung und Regelung der Motoren und der Lenkung. Es ist klar, dass die Ermittlung der eigenen Position von primärer Bedeutung ist. In vielen Anwendungen reicht ein einzelner Positionssensor (zum Beispiel ein GPS-Empfänger im Freien oder UWB in Räumen) alleine nicht aus, um eine befriedigende, genügend dynamische und präzise Positionsschätzung zu erhalten. Statt dessen werden weitere Grössen gemessen und benutzt, zum Beispiel die Geschwindigkeit und Beschleunigung des Fahrzeugs, die Himmelsrichtung und deren Änderung usw. Mittels Sensor Fusion (zum Beispiel durch

ein Kalman Filter) wird daraus eine gemeinsame Schätzung erzeugt. Zudem wird vorausgesetzt, dass ein autonomes Fahrzeug in der Lage ist, Hindernisse in der Umgebung zu erfassen, um darauf entsprechend zu reagieren, zum Beispiel indem es stoppt oder ihnen ausweicht. Bei solchen Anti-Kollisionssystemen will man sich im Normalfall auch nicht auf einen einzelnen Sensor verlassen. Ein Ultraschallsystem ist nicht geeignet für lange Distanzen, Radar kann sehr kleine Hindernisse, etwa einen Maschendraht, übersehen, ein Laserdistanzmesssystem (Lidar) schwächelt bei Regen und ein optisches System mit Kameras kann bei Nebel oder direkter Sonneneinstrahlung keine guten Resultate erzielen. Die Schwächen und Stärken der einzelnen Sensoren können durch geeignete Algorithmik zu einem robusten und zuverlässigen Gesamtsystem vereint werden. Eine weitere wichtige Grundfunktion eines autonomen Fahrzeugs ist natürlich das Anfahren eines Ziels. Für die meisten industriellen Anwendungen genügt es nicht, sich in Richtung des Zielpunktes zu bewegen und unterwegs Hindernissen auszuweichen. Statt dieser eher zufälligen Strategie ist es oft erwünscht, von der eigenen Position zum Ziel, unter Berücksichtigung von bekannten Hindernissen, den (oder einen) optimalen Weg zu finden. Der sogenannte Path Planner ist dafür verantwortlich.

Elementarer Path Planner

Seine Aufgabe besteht also darin, von einem Startpunkt aus, einen optimalen Weg zum Ziel zu berechnen. Was optimal ist, definiert ein bestimmtes Kriterium, zum Beispiel indem der Weg minimale Steigungen haben soll, minimalen Treibstoffverbrauch verursacht, keine zu engen Kurven enthält, oder etwas Ähnliches. Oft jedoch ist einfach der kürzeste Weg gesucht. Bekannte Hindernisse werden in einer Karte markiert und dem Algorithmus zur Verfügung gestellt. Zur Illustration stelle man sich eine Lagerhalle vor, in der sich ein autonomer Roboter bewegt. Der soll nun auf dem kürzesten Weg zu einer gewissen Stelle bei Regal C fahren. Eine entsprechende Karte ist in Bild 2 dargestellt. Beachte, dass die Regale Hindernisse darstellen.

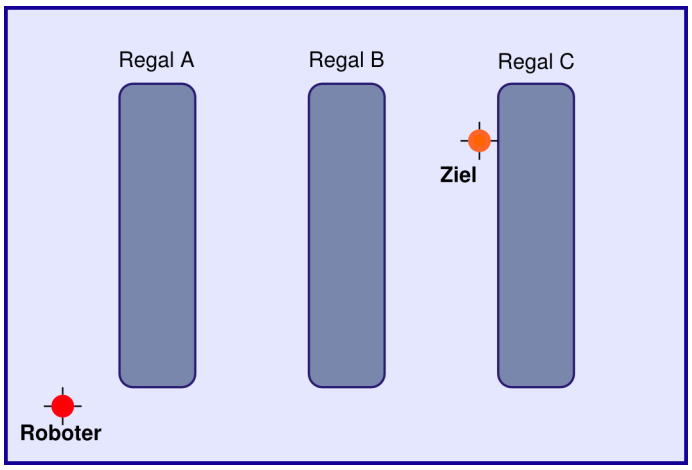


Bild 2: Grundriss einer Lagerhalle mit Regalen.

Eine Strategie, um dieses Problem zu lösen, geht von einer diskretisierten Karte aus (siehe Bild 3). Dabei wird ein Gitter mit Zellen über die Karte gelegt. Dann wird eine Bewertungsfunktion definiert. Mit Hilfe dieser Funktion wird anschliessend für jede Zelle iterativ eine Art Kostenwert errechnet. Jede Kostenzahl wird direkt in die jeweils betreffende Zelle notiert. Wenn wir den kürzesten Weg vom Startpunkt zum Ziel finden wollen, so wählen wir als Kosten eines Feldes sinnvollerweise gerade die Anzahl Felder, die durchgequert werden müssen, um vom Startpunkt zum Feld zu gelangen. Die Distanz zwischen zwei horizontal oder vertikal benachbarten Feldern zählen wir als eine Einheit. Bei diagonal benachbarten Feldern zählen wir die Distanz als 1.4 Einheiten, was eine einfache Annäherung an die euklidische Distanz ist (die Quadratwurzel aus zwei).

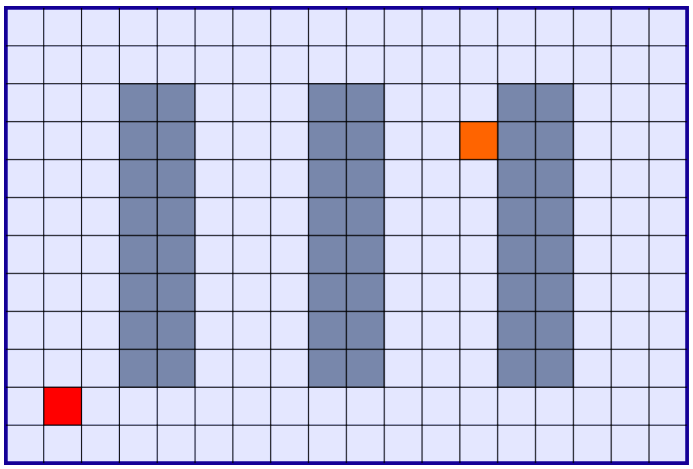


Bild 3: Diskretisierte Karte der Lagerhalle.

Um für alle Zellen bis zum Ziel die Kostenfunktion zu berechnen, geht man folgendermassen vor:

1. Es werden alle befahrbaren Zellen auf den Kostenwert unendlich initialisiert.
2. Alle befahrbaren Zellen werden als noch nicht besucht markiert (weiss).
3. Die Startzelle erhält den Kostenwert 0.

Iterativ werden nun die Kosten der Knotenpunkte nach folgendem Schema reduziert:

4. Wähle unter den noch nicht besuchten Zellen jene mit dem tiefsten Kostenwert aus und markiere sie temporär (gelb). Sollte es mehrere Kandidaten mit dem selben tiefsten Kostenwert geben, so wähle einen davon aus.
5. Berechne die Kostenfunktion für alle unmittelbaren Nachbarzellen, die noch nicht besucht wurden (temporär grün).
6. Vergleiche für jeden Nachbarn den betreffenden neuen Kostenwert mit dem schon bestehenden und übernimm den tieferen Wert von beiden.
7. Markiere die aktuelle Zelle als besucht (blau).
8. Wiederhole die Punkte 4 bis 7 solange, bis die Zielzelle als besucht markiert wurde.

Nun kann der kürzeste Weg ausgelesen werden:

9. Wähle als Ausgangspunkt die Zelle am Ziel.
10. Wähle unter den unmittelbaren Nachbarzellen, die als besucht markiert sind, jene Zelle als Nachfolger im Pfad aus, die den kleinsten Kostenwert hat. Markiere sie (roter Rand). Sollte es mehrere Kandidaten mit dem selben tiefsten Kostenwert geben, so wähle einen davon aus.
11. Wiederhole Punkt 10 für jede Nachfolgerzelle so lange, bis der Startpunkt erreicht ist.

Der eben beschriebene Algorithmus ist übrigens nicht neu. Er wurde von Edsger Dijkstra 1959 publiziert, als von autonomen Fahrzeugen noch kaum die Rede war, und ist heute Teil der modernen Graphen-Theorie.

In den Bilder 4 bis 12 wird das Verfahren auf unsere Beispielkarte angewendet. Bild 4 zeigt die initialisierte Karte. Der Startpunkt ist als ausgewählt markiert (gelb). Die Kostenwerte seiner Nachbarn (grün) werden in der ersten Iteration reduziert. In Bild 5 sind die neuen Kostenwerte eingetragen.

Damit ist die erste Iteration abgeschlossen, die Startzelle wird als besucht markiert (blau). Jetzt wird die nächste Zelle ausgewählt, nämlich eine, die noch nicht besucht ist und davon jene, die aktuell den kleinsten Kostenwert hat. Bild 6 zeigt, dass das Feld links des Startpunktes gewählt wurde. Es hätte noch drei weitere

Kandidaten mit dem selben Kostenwert gegeben. Die Wahl ist zufällig. Wiederum bezeichnen die grünen Felder jene, die in der zweiten Iteration aufdatiert wurden. Bild 7 zeigt den Zustand nach der dritten Iteration. Die folgenden Bilder 8 und 9 zeigen, wie die blaue Fläche der schon besuchten Felder sich langsam in alle Richtungen ausbreitet. In Bild 10 wird die Zielzelle erreicht und in Bild 11 wird das Ziel als besucht markiert. Damit endet der Algorithmus. In Bild 12 ist der gesuchte Weg markiert. Er folgt rückwärts vom Ziel zum Start, entlang des grössten Gradienten.

Verbesserungen

Obwohl das wunderbar funktioniert, hat Dijkstras Algorithmus ein Problem: Je nach Auflösung und Grösse der Karte kann das Verfahren eine extrem hohe Anzahl von Iterationen erfordern und ist folglich für Echtzeitsysteme schlecht geeignet. Aus diesem Grund gibt es einige Varianten davon.
Das sogenannte A*-Verfahren verbessert in erster Linie die Effizienz. Zu diesem Zweck wir die Kostenfunktion (Entfernung eines Feldes vom Startpunkt) durch eine heuristische Schätzung der Distanz des Feldes zum Ziel erweitert. Die beiden Entfernungen werden addiert. Das führt dazu, dass der Kostenwert von Zellen, die vom Ziel weg führen anwachsen und folglich nicht sofort weiter verfolgt werden. Erst wenn die scheinbar direkten Wege nicht zum Ziel führen, greift der Algorithmus auf diese Zellen zurück.
Der D*-Algorithmus ist für autonome Systeme besonders nützlich. Er beginnt nicht beim Startpunkt und sucht den kürzesten Weg zum Ziel, sondern umgekehrt: Er sucht den kürzesten Weg vom Ziel zum Startpunkt. Dabei verwendet er im Wesenlichen das A*-Verfahren. Verändert sich nun der Startpunkt, weil sich das Fahrzeug bewegt und tritt plötzlich ein zuvor noch nicht bekanntes Hindernis auf, so muss nicht der gesamte Weg neu berechnet werden. Statt dessen wird die Kostenkarte nur partiell in der Umgebung des Hindernisses aufdatiert. Vom Hindernis zum Ziel bleiben die Kostenwerte erhalten. Dies erlaubt ein schnelles Umplanen und Reaktion auf unvorhergesehene Veränderungen der Karte.

Fazit

Path Planning ist bei autonomen Systemen nicht wegzudenken und bildet einen essentiellen Bestandteil intelligenter Fahrzeuge. Gerade im Vergleich zu Systemen ohne diese Fähigkeit, welche oft den Anschein erwecken, als irren sie ziellos umher, wird schnell klar, wie wichtig ein solches Verfahren in der Praxis ist. Der Rechenaufwand ist indessen nicht zu vernachlässigen. Ausserdem muss der Path Planner mit unerwarteten Hindernissen oder Änderungen der Karte zurecht kommen. Verschiedene Varianten des klassichen Dijkstra-Algorithmus tragen diesen Bedürfnissen Rechnung.

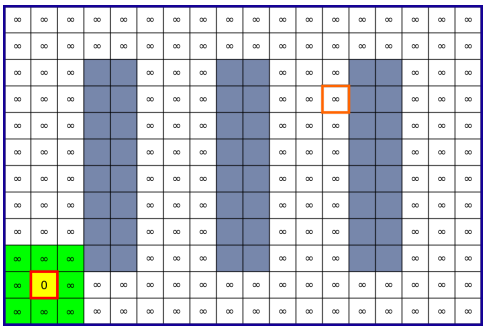


Bild 4: Initialisierte Karte. Der Startpunkt wurde ausgewählt (gelb) und seine Nachbarn sind markiert (grün).

Bild 5: Karte nach der erste Iteration. Die Kostenwerte der Nachbarn wurden berechnet und aufdatiert.

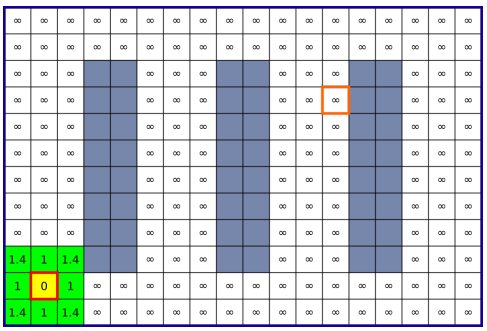


Bild 6: Karte nach der zweiten Iteration mit aufdatierten Kostenwerten.

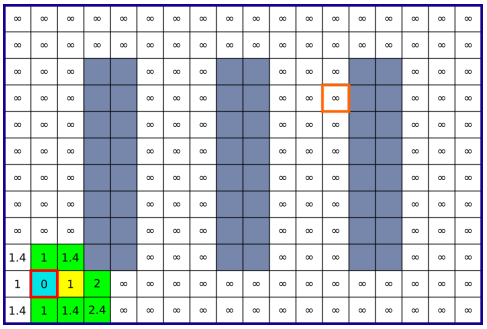


Bild 7: Karte nach der dritten Iteration mit aufdatierten Kostenwerten.

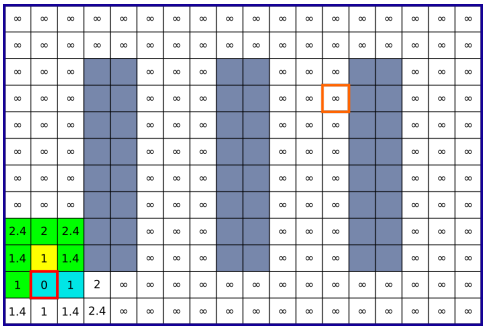


Bild 8: Karte nach 20 Iterationen.

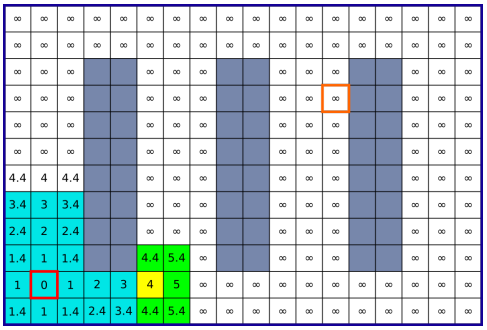


Bild 9: Karte nach 60 Iterationen.

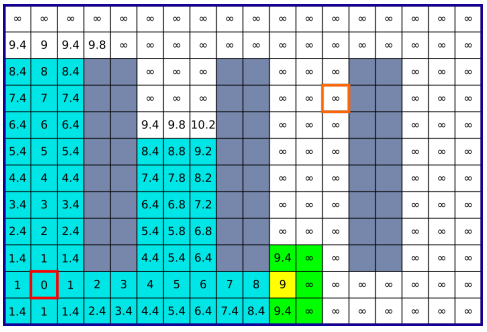


Bild 10: Karte nach 117 Iterationen. Das Verfahren erreicht das Ziel.

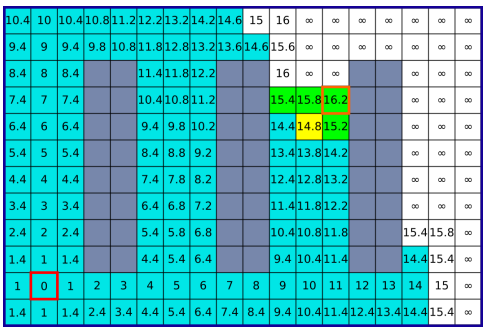


Bild 11: Karte nach 131 Iteration. Das Ziel wird nun gleich als besucht markiert.

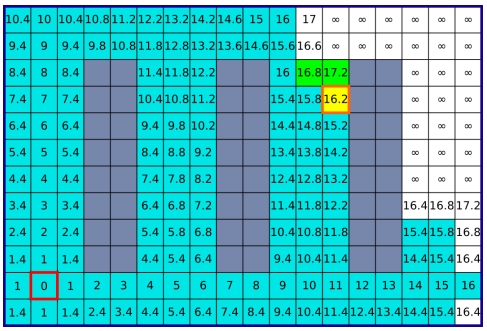
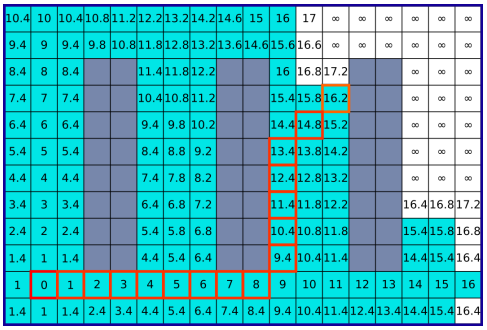


Bild 12: Das Ziel wurde als besucht markiert. Der Algorithmus hat das Ende erreicht. Der Pfad wird ausgelesen.



Stettbacher Signal Processing AG

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

Neugutstrasse 54
CH-8600 Dübendorf