Navigation durch Flächen

Jörg Roth

Fakultät Informatik, Technische Hochschule Nürnberg
Kesslerplatz 12, 90489 Nürnberg
Joerg.Roth@th-nuernberg.de

Abstract. Die Planung eines optimalen Weges in einer Fläche ist ein nicht-triviales Problem. Viele Ansätze zur Wegeplanung basieren auf Graphen. Diese können zwar einfach aus digitalen Wegenetzen berechnet werden, Flächen widersprechen aber dem etablierten Vorgehen. In diesem Beitrag wird ein rasterbasiertes Verfahren vorgestellt, das Flächen in eine Routenplanung integriert, dabei viele Nachteile von Rasteransätzen behebt. Der Ansatz erlaubt es, Kostenmodelle zu integrieren und verfügt über eine Bahnplanung.

Keywords: Routenplanung, A*, rasterbasierter Ansatz, Bahnplanung

1 Einleitung

Die Navigation in Straßennetzen ist ein gut erforschtes Gebiet. Basierend auf Dijkstras Algorithmus [1] oder A* [3] ist es möglich, Pfade mit minimalen Kosten effizient zu berechnen. Hierzu erforderlich ist die Modellierung der befahrbaren Umwelt als Graph mit Knoten (Kreuzungen) und Kanten (Straßenabschnitten zwischen Kreuzungen). Kantengewichte repräsentieren die zu minimierenden Kosten, z.B. die Durchfahrungszeit oder Distanz.

In den meisten Szenarien ist dieser Ansatz günstig: digitale Straßennetze aus Geodatenbeständen sind schon relativ nah an der gewünschten topologischen Darstellung [7, 8]. Die Situation ändert sich aber, sobald nicht nur ausschließlich ein linienförmiges Straßennetz vorliegt, und die Fahrbewegungen (zumindest Abschnittsweise) über Flächen möglich sind. Szenarien sind die Navigation auf großen Plätzen z.B. Parkplätzen, Brachen, Flugplätzen oder Flächen der Lagerlogistik. Bestimmte Fahrzeuge können auch auf natürlichen Flächen navigieren (Wiesen, Eisflächen). Auch fahrerlose Transportsysteme, letztlich sogar Erkundungsfahrzeuge wie der Mars Rover, haben dasselbe Problem.

Man kann das Grundproblem der Flächennavigation in drei Teilprobleme zerlegen:

- Da klassische Navigationsalgorithmen nur auf Linienobjekten operieren, müssen Flächen durch Graphen aus Knoten und Kanten repräsentiert werden. Das ist keineswegs trivial, da eine zweidimensionale Umgebung auf ein Netzwerk von eindimensionalen Strukturen abgebildet werden muss.
- Auf Flächen liegen die Fahrkosten erst einmal als Funktion der Ebene vor. Beispielsweise können unterschiedliche Bodenbeschaffenheiten definiert werden, auf denen ein Fahrzeug verschieden schnell fahren kann oder es wird die Steigung in die Kosten eingerechnet. Die zweidimensionale Darstellung der Fahrkosten muss auf den Graphen in Form von Kantengewichten abgebildet werden.
- Ein optimaler Weg durch die Fläche muss letztlich auf befahrbare Bahnen abgebildet werden. Während in Straßennetzen klar ist, dass jede gültige Abbiegung auch tatsächlich mit einem Fahrzeug gefahren werden kann, ist das bei Pfaden durch Flächen nicht immer gewährleistet.

Existierende Ansätze der Flächennavigation kann man grob in zwei Kategorien teilen. *Vektorbasierte* Ansätze legen linienförmige Strukturen über die Fläche, z.B. Voronoi-Diagramme [2, 5] oder Sichtbarkeitsgraphen [6]; danach wird der klassische Ansatz mit Linienstrukturen verfolgt. *Rasterbasierte* Ansätze legen ein hinreichend feinmaschiges Netz über die Fläche und modellieren die unendlich vielen möglichen Positionen durch endlich viele Zellen; navigiert wird dann von Zelle zu Zelle. Beide Ansätze haben Nachteile:

- Vektorbasierte Ansätze erreichen durch die Linienstrukturen nicht die gesamte Fläche, d.h. sie sind vor allem dazu geeignet, eine Fläche zu durchfahren. Darüber hinaus entstehen je nach Linienstruktur künstliche, insb. suboptimale Routen.
- Rasterbasierte Ansätze in der Grundform erlauben nur die Bewegung von Mittelpunkt zu Mittelpunkt zwischen Nachbarzellen. In der Konsequenz entstehen nur Fahrrichtungen von Vielfachen von 45°. Darüber hinaus geht bei der Abbildung der Umgebung auf ein Raster die Genauigkeit von Positionen verloren.

Ein weiterer Nachteil von rasterbasierten Ansätzen ist der Speicherbedarf: bei großen Gesamtflächen und kleinen Zellen können viele Millionen Zellen resultieren. Dieser Nachteil verliert aber in Hinblick auf aktuell verfügbare Speichergrößen, auch im Bereich mobiler Rechner, immer mehr an Bedeutung.

2 Der Ansatz

Ein neuer eigener Ansatz *GAA* (*Grid-based A* Advanced*) soll die Probleme existierender Verfahren überwinden. Als grundlegende Entscheidung wurde ein rasterba-

sierter Ansatz gewählt. Es entsteht zwar Rechenaufwand, das Raster anzulegen und zu verwalten, demgegenüber sind die geometrischen Operationen in der Regel in der Ausführung einfacher. Darüber hinaus können der Ressourcenbedarf und die Genauigkeit stufenlos über die Zellengröße angepasst werden. Bei der Verwendung von A* und einer geeigneten Schätzfunktion werden selbst bei großen Rastern akzeptable Zahlen von Knoten entwickelt. Darüber hinaus spart man sich bei einer sinnvollen Raster-Struktur die explizite Speicherung der Graph-Topologie durch eine Verzeigerung, da die Nachbarschaftsbeziehung über die Zellenindizes repräsentiert werden kann. Der resultierende Speicherbedarf liegt selbst bei Rastern mit Million von Zellen im Bereich von Megabytes – eine Größe die in den meisten Szenarien akzeptabel ist. Im Rahmen des GAA-Verfahrens gibt es folgende Festlegungen:

- Für jede Navigationsanfrage wird ein eigenes Raster angelegt und nach der Planung verworfen. Es wurde bewusst auf *inkrementelle* Verfahren [4, 9] verzichtet, da diese erfordern, dass sukzessive Anfragen sich auf dasselbe Ziel beziehen. Darüber hinaus sind inkrementelle Verfahren mit einigen der hier vorgeschlagenen Ansätze nicht vereinbar.
- Das Raster speichert nicht nur, ob eine Zelle ein Hindernis oder befahrbar ist, sondern repräsentiert auch die Kosten. Damit können beispielsweise Fahrkosten durch eine Steigung repräsentiert werden.

Weitere Nachteile von Rastern wurden mit einer Reihe von Verfahren behoben, die in den folgenden Abschnitten beschrieben werden.

2.1 Die Anfangsprojektion

Wir gehen davon aus, dass die gewünschte Zellengröße vom Verwender der Navigation vorgeben wird. Diese kann anhand von zwei Bedingungen gewählt werden:

- Die Zellen müssen so klein sein, dass die Umwelt in erforderlicher Genauigkeit als Zellen dargestellt werden kann. Dies betrifft die Granularität der Hindernisdarstellung, die Geometrie des Fahrzeuges aber auch die Auflösung der Kostenfunktion. Anders ausgedrückt: zu große Zellen dürfen nicht verhindern, dass zwischen zwei Zellen signifikante Änderungen der Belegung und Kostenfunktion repräsentiert werden können.
- Die Zellen müssen so groß sein, dass bei dem gewünschten Einsatz kein Speicher- oder Laufzeitproblem entsteht.

Für eine vorgegebene Zellengröße $(\Delta x, \Delta y)$ wird zunächst eine Abbildung von Weltkoordinaten (x, y) auf Zellenpositionen (p, q) festgelegt. Wir gehen im Folgenden von kartesischen Koordinaten aus, obwohl wir Koordinaten der Erdoberfläche verwenden, wir also genau genommen eine sphärische Geometrie zugrunde legen müs-

sen. In der Regel sind allerdings die fraglichen Flächen hinreichend klein, so dass die Erdkrümmung vernachlässigt werden kann. Ggfs. verwenden wir *ebene* Koordinaten, wie beispielsweise Gauß-Krüger.

Eine einfache Projektion lautet

$$\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \lfloor \tilde{p} \rfloor \\ \lfloor \tilde{q} \rfloor \end{pmatrix}, \begin{pmatrix} \tilde{p} \\ \tilde{q} \end{pmatrix} = S \cdot \begin{pmatrix} x \\ y \end{pmatrix} + T \ , \ S = \begin{pmatrix} 1/\Delta x & 0 \\ 0 & 1/\Delta y \end{pmatrix} \tag{1}$$

Hierbei wird durch $\lfloor \ \rfloor$ das Abrunden dargestellt. T wird so gewählt, dass alle Hindernisse, sowie Start (x_s, y_s) und Ziel (x_z, y_z) über Zellen-Indizes im geforderten Wertebereich abgebildet werden können. In der Regel bedeutet das $\min(p)=\min(q)=0$.

GAA verwendet allerdings eine modifizierte Projektion, die eine zusätzliche Rotation R einbezieht:

$$\begin{pmatrix} \tilde{p} \\ \tilde{q} \end{pmatrix} = S \cdot R \cdot \begin{pmatrix} x \\ y \end{pmatrix} + T \ , \ R = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{2}$$

R und T werden gemäß folgender Regeln bestimmt (Abb. 1):

- Start- und Zielzellen liegen auf demselben Zellenindex in y-Richtung, und jeweils auf der Zellenmitte in y-Richtung.
- Die Startposition liegt zusätzlich auf der Zellenmitte in x-Richtung.

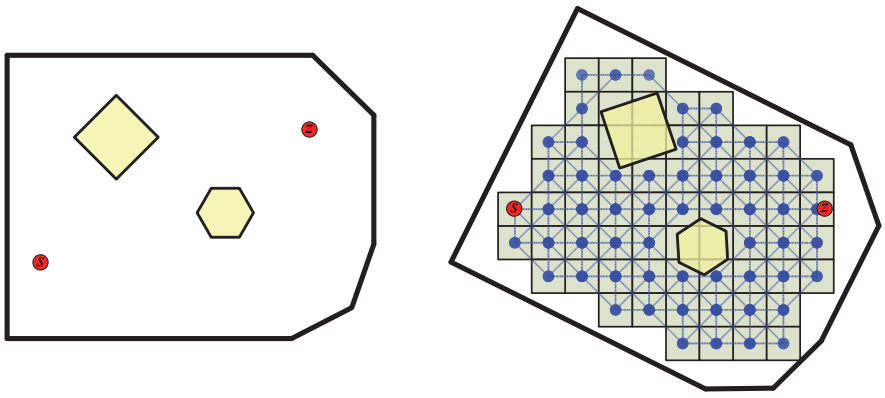

Abb. 1. Abbildung von Weltkoordinaten auf das Raster

Formal ausgedrückt

$$\tilde{q}_s = \tilde{q}_z \ , \ p_s = \tilde{p}_s - \frac{1}{2}, q_s = \tilde{q}_s - \frac{1}{2} \tag{3}$$

Diese Bedingungen reichen aus, um R und T eindeutig zu bestimmen. Die Abbildung stellt sicher, dass der Start auf einen Zellmittelpunkt abgebildet wird, d.h. exakt dar-

stellbar ist. Darüber hinaus liegen Start und Ziel nach der Projektion parallel zur x-Achse, d.h. eine direkte Verbindung (falls das die optimale Route ist), kann *exakt* über eine Folge von Zellen dargestellt werden. Damit hat man bezüglich Start und Ziel nur noch eine mögliche Ungenauigkeit: das Ziel kann innerhalb einer Zelle in x-Richtung neben dem Mittelpunkt liegen. Das muss bei dem letzten Schritt zum Ziel von den betreffenden Nachbarzellen berücksichtigt werden.

2.2 Erweitern der Nachbarschaftsbeziehung

Verbindet der Graph nur jeweils acht Nachbarn (*8-Connected-Gridworld*), erhält man als Routenrichtungen nur Vielfache von 45°; die Resultate sind daher unbefriedigend. Daher verbindet GAA nicht nur eine Zelle mit den acht Nachbarzellen, sondern kann auch mehrere direkte Nachbarn überspringen, derzeit bis zur Schrittweite fünf. Der Nachbarschaftsgraph wird dadurch zwar komplexer, da eine Zelle nicht mehr acht, sondern bis zu 120 Nachbarn hat; Algorithmen wie A* profitieren allerdings davon, da der kürzeste Weg schneller gefunden wird. Darüber hinaus können Einzelschritte von bis zu 80 verschiedenen Winkeln auftreten und nicht nur acht. Abb. 2 (links) illustriert den Sachverhalt. Um die Laufzeit zu optimieren, werden die entsprechenden Verwaltungsinformationen über vorberechnete Listen unterstützt (Abb. 2 rechts).

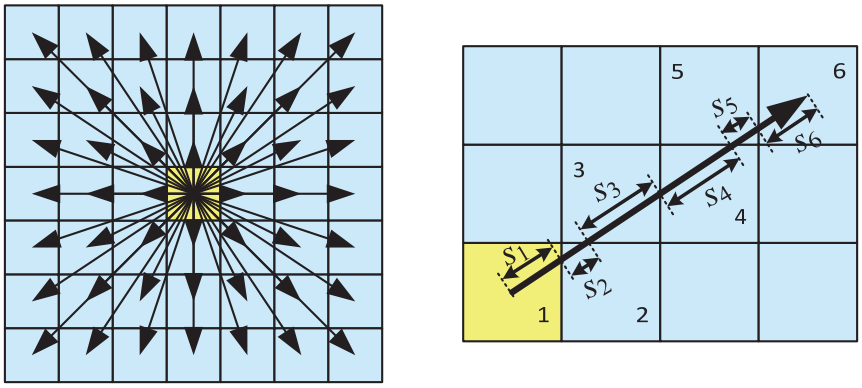

Abb. 2. Nachbarschaftsgraph der Stufe 3 (links), vorberechnete Größen (rechts)

Diese Listen enthalten:

- Die Zellen *i*, die bei der Verbindung zweier Zellen durchlaufen werden – diese Information wird benötigt, um zu erkennen, ob ein Weg überhaupt kollisionsfrei durchfahren werden kann.

- Der Anteil der Gesamtstrecke der durch eine bestimmte Zelle *i* geht, genannt s_i. Diese Information benötigt man für die Berechnung der absoluten Kosten (siehe nächster Abschnitt).

Obwohl Nachbarzellen übersprungen werden, haben lange Strecken möglicherweise immer noch überflüssige Abknickpunkte. Nachdem eine optimale Route gefunden wurde, wird daher versucht, solche Punkte zu eliminieren (Abb. 3).

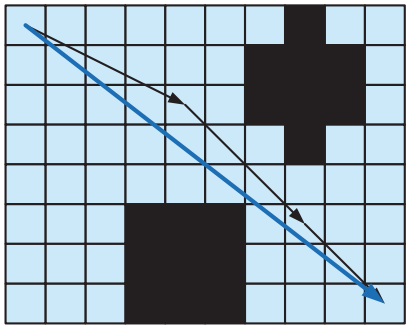

Abb. 3. Glättung einer Route

Hier wird für jede Teilfolge geprüft, ob sie durch die direkte Verbindung ersetzt werden kann. Allerdings muss geprüft werden, ob die neue Verbindung belegte Zellen durchläuft oder höhere Kosten verursacht. Um zu entscheiden, welche Variante man nimmt, werden die Kosten der direkten Verbindung zu den Kosten der Teilfolge in Relation gesetzt. Wenn das Verhältnis einen vorgegebenen Wert (z.B. 1,05) nicht überschreitet, ersetzt man die Teilfolge durch die direkte Verbindung.

2.3 Kosten

Ohne weitere Konfiguration sind die Fahrkosten nur die geometrischen Distanzen beim Durchfahren der Zellen. Damit kann man die Route mit dem kürzesten Fahrweg berechnen. Häufig sind aber komplexere Kostenfunktionen gewünscht, z.B. die Fahrzeit, die die Befahrbarkeit des Untergrundes berücksichtigt. Es gibt zwei Arten von Kosten:

- *absolute* Kosten (z.B. Bodenbeschaffenheit) werden durch Kostenfaktoren modelliert, die zur geometrische Distanz multipliziert werden;
- *differenzielle* Kosten (z.B. Steigungen) werden durch Kostenfunktionen beschrieben, die jeweils zwei Zellen (von, bis) berücksichtigen.

Der Ansatz von GAA ist, in jeder Zelle *i* pro Kostenart einen Wert einzutragen, bezeichnet mit a_i und d_i. Die absoluten Kosten eines Weges durch mehrere Zellen sind

$$K_a = \sum_{\text{durchfahrene Zellen } i} a_i \cdot s_i \tag{4}$$

wobei s_i die Länge des Abschnitts des Weges durch eine Zelle ist (wie im letzten Abschnitt beschrieben). Die differenziellen Kosten sind

$$K_d = \sum_{\text{durchfahrene Zellen } i_1 \to i_2} k(d_{i1}, d_{i2}) \tag{5}$$

Hierbei ist *k* eine Funktion, die dem System als Kostenfunktion mitgeteilt wird. Möchte man beispielsweise die Steigung in die Kosten einrechnen, würden d_i die Höhen der Zellmittelpunkte enthalten und *k* würde man wie folgt definieren:

$$k(d_{i1}, d_{i2}) = \begin{cases} d_{i2} - d_{i1} & \text{wenn } d_{i2} \geq d_{i1} \text{ (Steigung)} \\ 0 & \text{sonst (kein Vorteil durch Gefälle)} \end{cases} \tag{6}$$

Die Gesamtkosten ergeben sich als Summe von absoluten und differenziellen Kosten, ggfs. mit Gewichtungen. Die absoluten Kosten kann man in diesem Ansatz auch verwenden, um Wunschabstände zu Hindernissen einzuhalten. Die Vorgaben hierzu sind wie folgt: es gibt einen Mindestabstand und einen Wunschabstand zu Hindernissen. Der Mindestabstand *muss* eingehalten werden, der Wunschabstand *sollte* eingehalten werden, es sein denn, eine Unterschreitung (unter Berücksichtigung des Mindestabstands) führt zu einem wesentlich kürzeren Weg.

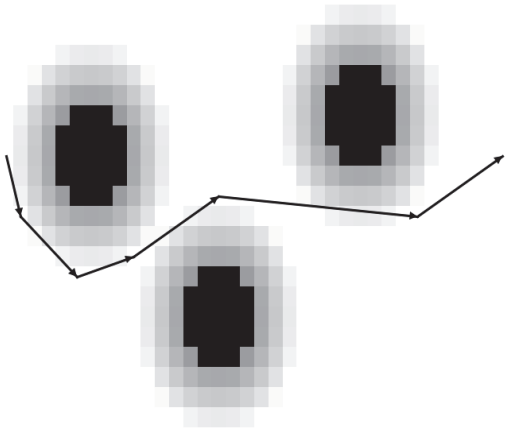

Abb. 4. Einsatz der absoluten Kosten zur Berücksichtigung von Wunschabständen

Abb. 4 illustriert den Sachverhalt. Die absoluten Kosten repräsentieren, den kürzesten Abstand zu einem der Hindernisse (dargestellt durch Grauwerte). Dadurch wird eine Route, die nahe an einem Hindernis vorbeigeht, mit höheren Kosten belastet und Routen in der Mitte zwischen zwei Hindernissen werden belohnt.

2.4 Bahnplanung

In einem letzten Schritt wird die Route in eine befahrbare Bahn überführt. Diese berücksichtigt geometrische Restriktionen des Fahrzeugs, insb. minimale Kurvenradien und Mindestabstand zu Hindernissen. Darüber hinaus soll eine Bahn nicht allzu weit von der Route entfernt sein, die durch die gradlinige Verbindung der Routenpunkte entsteht.

Wir definieren das Problem der Bahnplanung wie folgt: gegeben eine Liste von Routenpunkten, also Start, Ziel sowie alle Abknickpunkte dazwischen. Wir nehmen darüber hinaus an, dass die Ausrichtung des Fahrzeugs in der Start- und Zielposition vorgegeben ist, beispielsweise durch den Straßenverlauf der Ein- und Ausfahrten in die Fläche. Gesucht wird die Bahn, die

- alle Routenpunkte schneidet,
- die gesuchten Ausrichtungen im Start und Ziel erzeugt,
- kein Hindernis schneidet und
- von allen denkbaren Bahnen die geringsten Kosten verursacht.

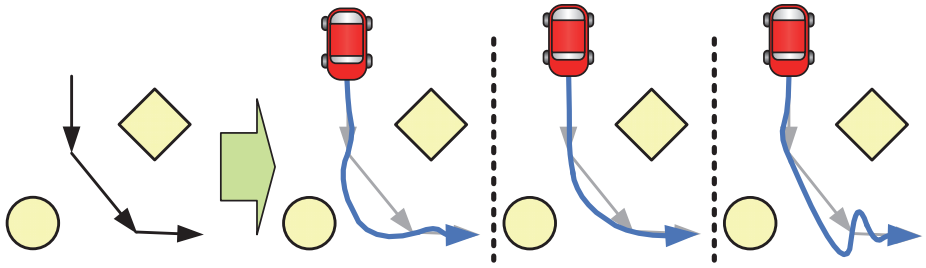

Abb. 5. Möglichkeiten für Bahnen, vier Routenpunkte zu durchfahren

Abb. 5 beschreibt ein Problem. Hier wird eine Route aus vier Routenpunkten beispielhaft durch drei Bahnen dargestellt. Alle Bahnen schneiden die Route in den gewünschten Punkten, auch sind die Ausrichtungen im Start- und Zielpunkt bei allen Bahnen wie gewünscht. Im Allgemeinen gibt es unendlich viele Bahnen, die unsere Bedingungen erfüllen. Diese Variationen entstehen durch zwei Effekte:

- Die Ausrichtungen in den Routenpunkten sind nicht vorgegeben.
- Auch bei vorgegebenen zwei Routenpunkten *mit* Ausrichtung gibt es unendlich viele Bahnen dazwischen.

Das macht es als Optimierungsproblem schwer lösbar. Der GAA-Ansatz macht daher eine Vereinfachung: von den unendlich vielen Möglichkeiten werden nur endlich viele in Erwägung gezogen.

Abb. 6 zeigt die Variationen für die Ausrichtungen in Routenpunkten. Es werden nur fünf Ausrichtungen in Betracht gezogen: die Ausrichtung durch die gradlinige Verbindung vom vorherigen (V) und zum nachfolgenden (N) Routenpunkt, der Mittelwert dieser beiden Ausrichtungen (M) sowie eine Veränderung des Mittelwerts um ±30° (M^+), (M^-).

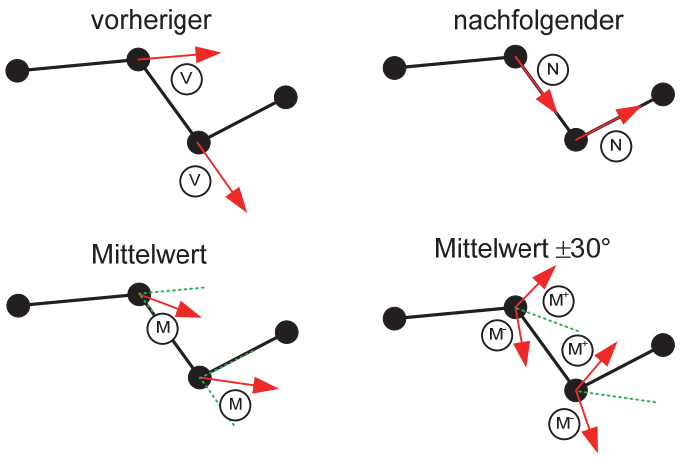

Abb. 6. Vielversprechende Ausrichtungen in Routenpunkten

Die zweite Vereinfachung betrifft die Bahnen zwischen Routenpunkten. Wir vereinfachen die Möglichkeiten, indem wir annehmen, dass diese grundsätzlich aus maximal drei Teilbahnen bestehen, wobei eine Teilbahn nur eine gradlinige Fahrt (*G*) oder eine Kurvenfahrt (*K*) mit konstantem Radius sein kann. Daraus entstehen vier mögliche Muster genannte *KG*, *GK*, *KK* und *KGK* (Abb. 7).

Davon sind die Muster *KG* und *GK* vollständig durch die Vorgaben definiert, bei den Mustern *KGK* und *KK* gibt es leider wieder unendliche viele Möglichkeiten, die sich in den Kurvenradien der beiden Kurvenfahrten unterscheiden. Damit auch hier wieder nur endlich viele vielversprechende Varianten entstehen, werden nur einige wenige Kandidaten von Kurvenradien r_1, r_2 betrachtet, z.B. $r_1/r_2 \in \{6/5, 1, 5/6\}$.

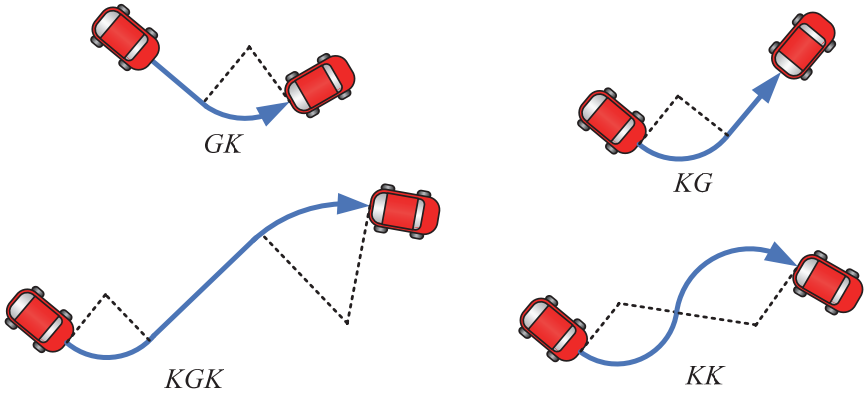

Abb. 7. Vielversprechende Bahnen zwischen zwei Routenpunkten

Die resultierenden Möglichkeiten sind immer noch zu zahlreich, um alle Permutationen auszutesten. Daher wurde ein Viterbi-Ansatz gewählt [10], der sukzessive Routenpunkte hinzunimmt und alle Möglichkeiten, die sich durch den neuen Punkt ergeben mit den bisherigen Routenmöglichkeiten kombiniert. Daraus resultiert eine überschaubare Zahl vor Variationen. Als Beispiel: bei 50 Routenpunkten entstehen nur 6000 Variationen von Bahnen, die bezüglich Kosten und Kollisionen geprüft werden müssen.

Abb. 8 zeigt einige Beispiele für die Routen- und Bahnplanung in Flächen. In allen Fällen wurden Abstandskosten definiert. Es gibt keine differenziellen Kosten.

3 Zusammenfassung und Ausblick

GAA ist ein rasterbasiertes Verfahren, das zwischen zwei beliebigen Flächenpunkten eine optimale mit dem Fahrzeug befahrbare Route berechnet, die eine flächenartige Kostenfunktion minimiert. Neben der reinen Distanz sind absolute und differenzielle Kosten möglich, die durch die Zellen modellierbar sind. Darüber hinaus können Wunschabstände zu Hindernissen berücksichtigt werden. Einige Probleme rasterbasierter Verfahren wurden behoben. Insbesondere werden die Winkel zwischen Routenpunkten nicht durch die Rasterung limitiert. Eine nachgeschaltete Bahnplanung überführt eine Route in befahrbare Bahnen. Hierbei wurde das Problem gelöst, dass es unendlich viele mögliche Bahnen gibt.

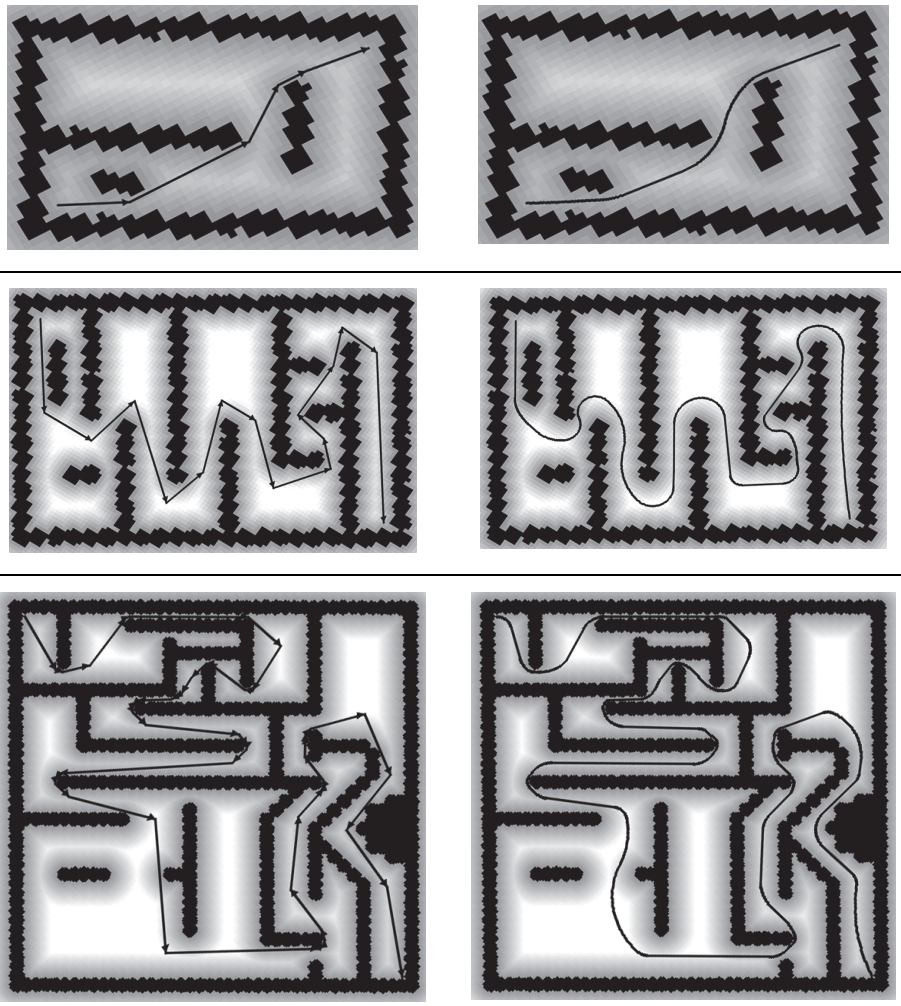

Abb. 8. Beispiel der Routenplanung (jeweils links) und der geplanten Bahnen (jeweils rechts)

Zurzeit kann die Bahnplanung fehlschlagen, obwohl die Routenplanung erfolgreich war. Das tritt dann auf, wenn eine Route prinzipiell befahrbar ist, die Ausrichtungen am Start und Ziel die Befahrung aber verhindert. Als einfaches Beispiel: Start und Ziel sind identisch, aber es ist ein Wenden von 180° gefordert. Aus der Sicht der Routenplanung ist das Problem lösbar und führt zu einer trivialen Route. Die Bahnplanung kann diese Route aber nicht in Bahnen überführen, ohne zusätzliche Routenpunkte zu integrieren. Dieses Problem ist im Moment ungelöst und ist Gegenstand zukünftiger Forschung.

Referenzen

[1] Dijkstra E. W.: A note on two problems in connexion with graphs, Numerische Mathematik. 1, 1959, 269–271 (1959)

[2] Garrido S., Moreno L., Abderrahim M., Martin F.: Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching, Proc. of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems October 9 - 15, 2006, Beijing, China (2006)

[3] Hart P. E., Nilsson N. J. and Raphael B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics SSC4 (2), 1968, 100–107 (1968)

[4] Koenig S.: Fast Replanning for Navigation in Unknown Terrain, IEEE Transactions on Robotics, Vol 21, No 3 (2005)

[5] Liebling, T., M., Lionel, P: Voronoi Diagrams and Delaunay Triangulations: Ubiquitous Siamese Twins, Documenta Mathematica. Extra Volume ISMP, 2012, 419-431 (2012)

[6] Rashid, A. T.; Ali, A. A.; Frasca, M.; and Fortuna, L.: Path planning with obstacle avoidance based on visibility binary tree algorithm. Robotics and Autonomous Systems 61(12): 1440–1449 (2013)

[7] Roth J.: Predicting Route Targets Based on Optimality Considerations, International Conference on Innovations for Community Services (I4CS), Reims (France) June 4-6, 2014, IEEE xplore, 61-68 (2014)

[8] Roth J.: Efficient Many-to-Many Path Planning and the Traveling Salesman Problem on Road Networks, KES Journal: Innovation in Knowledge-Based and Intelligent Engineering Systems, 20 (2016), IOS Press, 135–148 (2016)

[9] Stentz A.: Optimal and Efficient Path Planning for Partially Known Environments, Intelligent Unmanned Ground Vehicles, Vol. 388, Springer International Series in Engineering and Computer Science 203-220 (1997)

[10] Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In: IEEE Transactions on Information Theory. 13, Nr. 2, 1967, 260–269 (1967)