Einsatz von graphbasierten Ansätzen in einer mikroskopischen Personenstromsimulation für die Wegewahl der Fußgänger

Angelika Kneidl, André Borrmann
Lehrstuhl Computation in Engineering, Technische Universität München, Arcisstraße 21, 80290 München
kneidl@bv.tum.de
Dirk Hartmann
Siemens AG, CT T DE TC3, 80200 München, Germany

Kurzfassung: In diesem Beitrag befassen wir uns mit dem Einsatz von Graphen zur Navigation von Fußgängern in Personenstromsimulationen. Dabei integrieren wir einen Sichtbarkeitsgraphen in ein diskretes mikroskopisches Simulationsmodell. In diesem Modell wird auf unterster Ebene die Bewegung von Fußgängern als Resultat des Einwirkens abstoßender Kräfte modelliert, die durch andere Fußgänger und Hindernisse hervorgerufen werden. In einer darüber liegenden Modellebene wird ein Graph eingeführt, mit dessen Hilfe einzelne Fußgänger von der Quelle zum Ziel über mehrere Zwischenziele hinweg navigieren können. Wir stellen unterschiedliche Verfahren zur automatisierten Erzeugung des Graphen vor. Ein Vergleich der kürzesten Wege in den so konstruierten Graphen rundet den Beitrag ab.

1 Einleitung

Ziel des Einsatzes von Fußgängersimulationen ist es unter anderem, realistische Evakuierungszeiten zu berechnen sowie mögliche Konfliktpunkte bzw. Engstellen in Gebäuden/Umgebungen zu identifizieren und daraus optimale Evakuierungswege zu ermitteln. Im Rahmen des Forschungsprojekts REPKA1[1] (Regionale Evakuierung, Planung, Kontrolle und Anpassung) liegt der Fokus auf der Entwicklung eines Trainingssimulators, der zur Schulung von Sicherheitskräften dienen soll, die bei Großveranstaltungen eingesetzt werden. Betrachtetes Anwendungsszenario ist die regionale Evakuierung des Betzenberg-Stadions in Kaiserslautern.

Eine wesentliche Anforderung des Projektes liegt darin, eine hohe Anzahl von Personen (~ 50.000 Personen) in einem großen Gebiet (bis zu 1 km^2) schneller als in Echtzeit zu simulieren. Um dieser Anforderung gerecht zu werden, basiert unser Simulationsmodell auf einem diskreten, mikroskopischen Ansatz (Schadschneider et al. 2009). Die Navigation der Fußgänger wird hierbei entweder durch Potentialfeld-basierte Algorithmen (Burstedde et al. 2001; Kretz 2009) oder graphbasierte Algorithmen realisiert (Li & Höcker 2009).

Bei ausschließlich Potentialfeld-basierten Ansätzen stellen komplexe Geometrien eine Herausforderung dar. Durch Überlagerung der einzelnen Potentiale können lokale Minima auftreten, in die Fußgänger hineinlaufen und nicht wieder herausfinden (bspw.

[1] Gefördert vom Bundesministerium für Bildung und Forschung (BMBF)

ein U-förmiges Hindernis, auf das die Personen zusteuern). Eine Möglichkeit, die Potentiale entsprechend zu berechnen, ist in (Hartmann 2010) beschrieben.
Graphbasierte Ansätze hingegen können die gegenseitige Beeinflussung der Fußgänger nicht abbilden, da hier Fußgänger nicht als Individuen betrachtet werden, sondern vielmehr eine ganzheitliche Sicht umgesetzt wird.
Wir wollen in unserem Beitrag einen Ansatz vorstellen, der beide Ansätze kombiniert; dabei bildet das Kräftemodell die Beeinflussung der Fußgänger untereinander ab, die Navigation der Fußgänger wird mittels einem graphbasierten Ansatz realisiert.

2 Beschreibung des Simulationsmodells

In unserem Simulationsmodell wird der Raum mit einem hexagonalen Zellgitter diskretisiert, auf dem ein Zustandsautomat (zellulärer Automat) definiert ist (Liepert, Borrmann & Klein 2008; Klein, Köster & Meister 2010). Das Verhalten der Fußgänger wird auf diesem Zellgitter mittels eines Potentialmodells abgebildet. Dieses Potentialmodell setzt sich aus verschiedenen Potentialen zusammen, von denen die Fußgänger beeinflusst werden. Jeder Fußgänger steuert auf ein bestimmtes Ziel zu (Zielpotential) und wird auf dem Weg dorthin von Hindernissen (Hindernispotential) und anderen Fußgängern (Fußgängerpotential) beeinflusst. Diese drei Kräfte werden überlagert; auf dem resultierenden Potentialfeld bewegen sich die Fußgänger entlang des Gradienten zu ihrem Ziel. Da der Gradient eines Potentialfeldes im Allgemeinen eine Kraft darstellt, ist somit eine direkte Analogie zu Kräftemodellen gegeben.
Um komplexen Geometrien zu begegnen und das „Hängenbleiben“ von Fußgänger in lokalen Minima zu vermeiden, verwenden wir einen Sichtbarkeitsgraphen (de Berg, van Kreveld & Overmars 2000) zur Navigation, anhand dessen sich die Fußgänger orientieren können. Damit können wir das Zielpotential mit einer einfachen euklidischen Metrik abbilden und durch das Platzieren von Zwischenzielen in Sichtweite zueinander auf das Hindernispotential verzichten. Darüber hinaus verwenden wir den Graphen, um eine Reihe von Verhaltensmodellen zu implementieren, die die unterschiedliche Ortskenntnis verschiedener Personengruppen berücksichtigen (z.B. „Kürzester Weg“, „Heuristik“, „Wegesuche“).

3 Automatische Grapherzeugung

Jedes Szenario in unserem Simulator besteht aus ein oder mehreren Quellen und Zielen, sowie unterschiedlichen Arten von Hindernissen wie Wänden oder Polygonen (z.B. Schreibtische in Gebäuden etc.), um die die Fußgänger auf ihrem Weg zum Ziel herumlaufen. Aus der gegebenen Geometrie erzeugen wir automatisiert den Sichtbarkeitsgraphen in mehreren Schritten. Diese sind im Folgenden beschrieben.

3.1 Platzieren von Orientierungspunkten

Im ersten Schritt werden an jeder konvexen Ecke eines Hindernisses sogenannte Orientierungspunkte gesetzt, die später als Zwischenziele dienen. Der Abstand zur Ecke wird dabei so gewählt, dass künstliche Staus an Ecken vermieden werden. Zusätzlich muss die Bedingung erfüllt sein, dass zwischen der Ecke des Hindernisses und dem Orientierungspunkt eine Sichtverbindung besteht. Dazu muss überprüft werden, ob ein Hindernis oder eine Wand zwischen der Ecke und dem Orientierungspunkt liegt. Falls dies der Fall ist, wird der Punkt entsprechend näher an der Ecke platziert. Dies geschieht, indem der Abstand zwischen Ecke und schneidendem Hindernis gemessen und auf halber Strecke der Punkt neu gesetzt wird (siehe Abbildung 1).

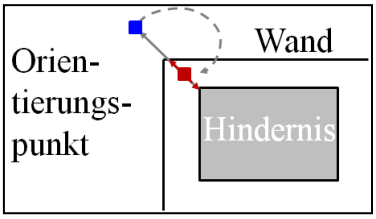


Abbildung 1 Verschieben von Orientierungspunkten

Um zu vermeiden, dass zwei benachbarte Orientierungspunkte zu nah beieinander liegen, werden die betroffenen Punkte in einem neuen, gemeinsamen Punkt zusammengefasst. Ein Beispiel hierfür ist in Abbildung 2 zu sehen.

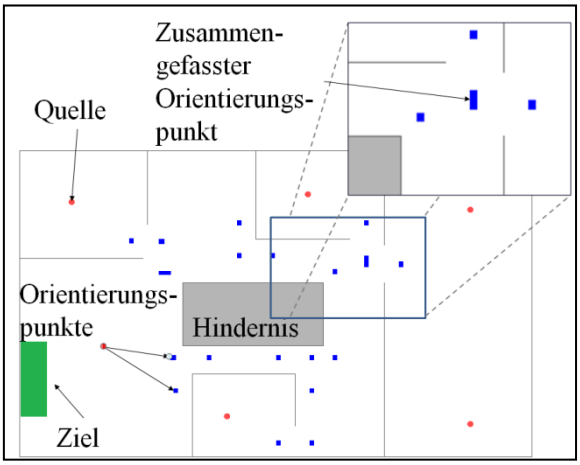


Abbildung 2 Zusammenführen von benachbarten Orientierungspunkten

Die resultierenden Orientierungspunkte sowie alle Quellen und Ziele entsprechen den Knoten des Sichtbarkeitsgraphen.

3.2 Verbinden der Punkte und Erzeugen des Graphs

Die wie oben beschrieben erzeugten Punkte werden nun durch Kanten miteinander verbunden. Dabei muss für jede Kante die Bedingung erfüllt sein, dass kein Hindernis geschnitten wird. Da wir richtungsbezogene Kanten benötigen, erzeugen wir einen gerichteten Graph.

3.2.1 Sichtbarkeitsgraph

Im einfachsten Fall werden jeweils zwei Punkte miteinander verbunden, die zueinander in Sichtlinie liegen. Dabei fügen wir zwischen zwei Knoten jeweils in beide Richtungen eine Kante ein; Ausnahme bilden Quellen und Ziele. Der so erzeugte Graph ist in Abbildung 5 oben links dargestellt.

Wie aus der Abbildung ersichtlich, ist der Graph sehr dicht. Viele Kanten sind überflüssig, da sie keinen Mehrwert bezüglich des kürzesten Weges liefern.

3.2.2 Kantenreduktion durch Winkelüberprüfung

Um diese überflüssigen Kanten zu eliminieren, wenden wir folgendes Verfahren an: Wir suchen von der Quelle zum Ziel anhand einer Breitensuche den Graphen ab. Dabei überprüfen wir für jeden gefundenen Knoten die ausgehenden Kanten. Da wir die ausgehenden Kanten nach Winkel sortiert abspeichern, können wir jeweils zwei benachbarte Kanten miteinander vergleichen.
Wenn nun der Winkel zwischen zwei Kanten einen gewissen Grenzwert ε unterschreitet, wird überprüft, ob eine dritte Kante existiert, die die beiden Endpunkte der ausgehenden Kante verbindet. Falls eine solche Kante existiert, werden die restlichen Winkel des aufspannenden Dreiecks untersucht: Falls zwei Winkel des Dreiecks kleiner als ε sind, wird die längste der drei Kanten gelöscht. Ein Beispiel ist in

Abbildung 3 gegeben.

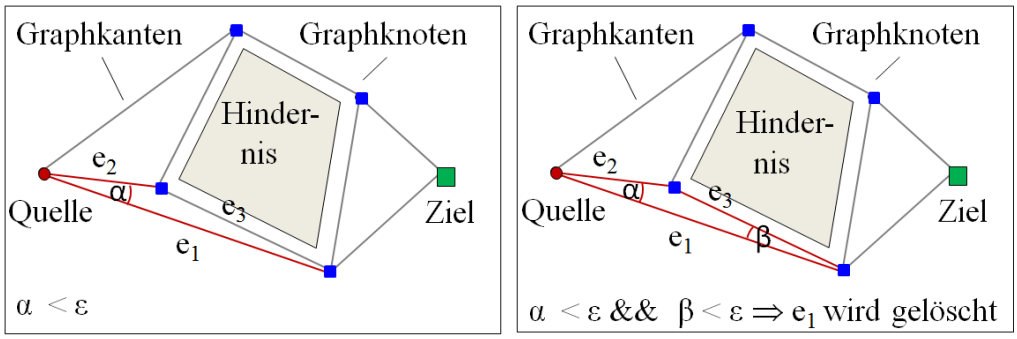


Abbildung 3 Kantenreduktion mittels Winkelüberprüfung

In unserem Fall haben wir ε = π/10 gesetzt. Mit dieser Festlegung wollen wir sicherstellen, dass die gelöschte Kante nicht wesentlich kürzer ist als die verbleibenden zwei Kanten; sich somit die Weglänge des kürzesten Weges nicht wesentlich verlängert.

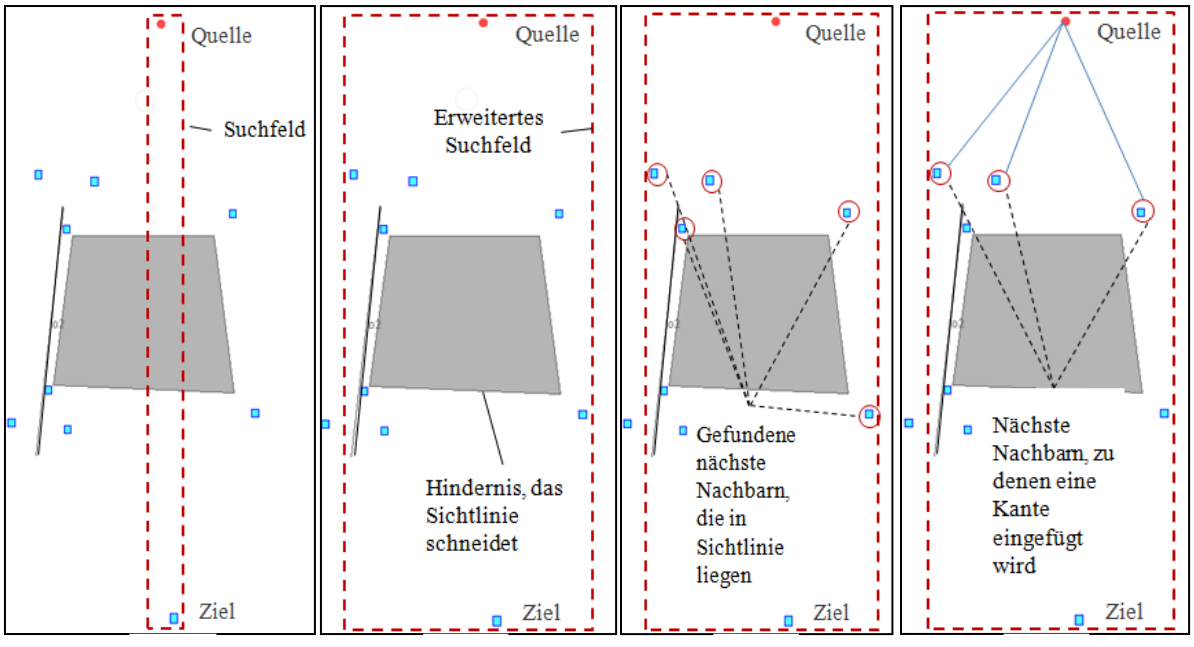
Mittels dieser Vorgehensweise können bis zu 40 Prozent der Kanten eliminiert werden wie in Abbildung 5 oben rechts dargestellt.

3.2.3 Grapherzeugung mit Hilfe einer räumlichen Datenstruktur

Der Graph weist trotz dieser Kantenelimination eine sehr hohe Dichte auf und enthält viele Kanten, die nicht relevant für Wege zwischen Quelle und Ziel sind. Daher verfolgen wir im Weiteren die Strategie, nur solche Knoten miteinander zu verbinden, die von der Quelle aus gesehen in Richtung Ziel liegen.

Um dies zu erreichen, führen wir eine räumliche Datenstruktur ein, in der wir die Knoten räumlich sortiert abspeichern. Wir verwenden einen R*-Baum (Beckmann et al. 1990). Diese Struktur erlaubt zum einen, unterschiedliche Arten von Objekten (Punkte und Polygone) abzuspeichern und zum anderen wird der Baum dynamisch bezüglich der räumlichen Verteilung der verwalteten Objekte generiert. Somit speichert der R*-Baum nur Regionen ab, in denen auch Punkte liegen. Mittels dieser Voraussetzungen können räumliche Zugriffsmethoden sehr effizient durchgeführt werden – in unserem Fall die Suche nach den nächsten Nachbarn.

Um die relevanten Kanten zu finden, gehen wir wie folgt vor: Für jedes Ziel eines Szenarios starten wir mit einem beliebigen Knoten (im Beispiel die Quelle) und definieren uns ein Suchfeld (Abbildung 4 (a)). Im zweiten Schritt wird überprüft, ob Hindernisse auf der Sichtlinie zwischen untersuchten Knoten und dem Ziel liegen. Falls ja, wird das Suchfeld entsprechend erweitert (Abbildung 4 (b)). Nun suchen wir innerhalb des Suchfelds nach den nächsten Nachbarn (Abbildung 4 (c)). Wir verbinden die nächsten Nachbarn mit dem Knoten (Abbildung 4(d)), falls kein Hindernis auf der Sichtlinie liegt. Insgesamt werden Kanten zu den drei nächsten Nachbarn (falls vorhanden) eingefügt. Diese Schritte wiederholen wir für jeden Knoten im Graph in Richtung jedes Ziels.



(a) (b) (c) (d)

Abbildung 4 Algorithmus zum Finden der nächsten Nachbarn: (a) Definition des Suchfelds zwischen Quelle und Ziel, (b) Erweiterung des Suchfeldes unter Berücksichtigung schneidender Hindernisse, (c) Ermitteln

der nächsten Nachbarn innerhalb des Suchfeldes, (d) Verbinden des Knotens mit den nächsten drei Nachbarn

Der resultierende Graph ist in Abbildung 5 unten links zu sehen. Da auch hier noch Kanten existieren können, die ein Dreieck mit sehr spitzen Winkeln aufspannen, können wir hier ebenfalls das in 3.2.2 beschriebene Verfahren zur Kantenelimination anwenden. Der resultierende Graph ist in Abbildung 5 unten rechts dargestellt.

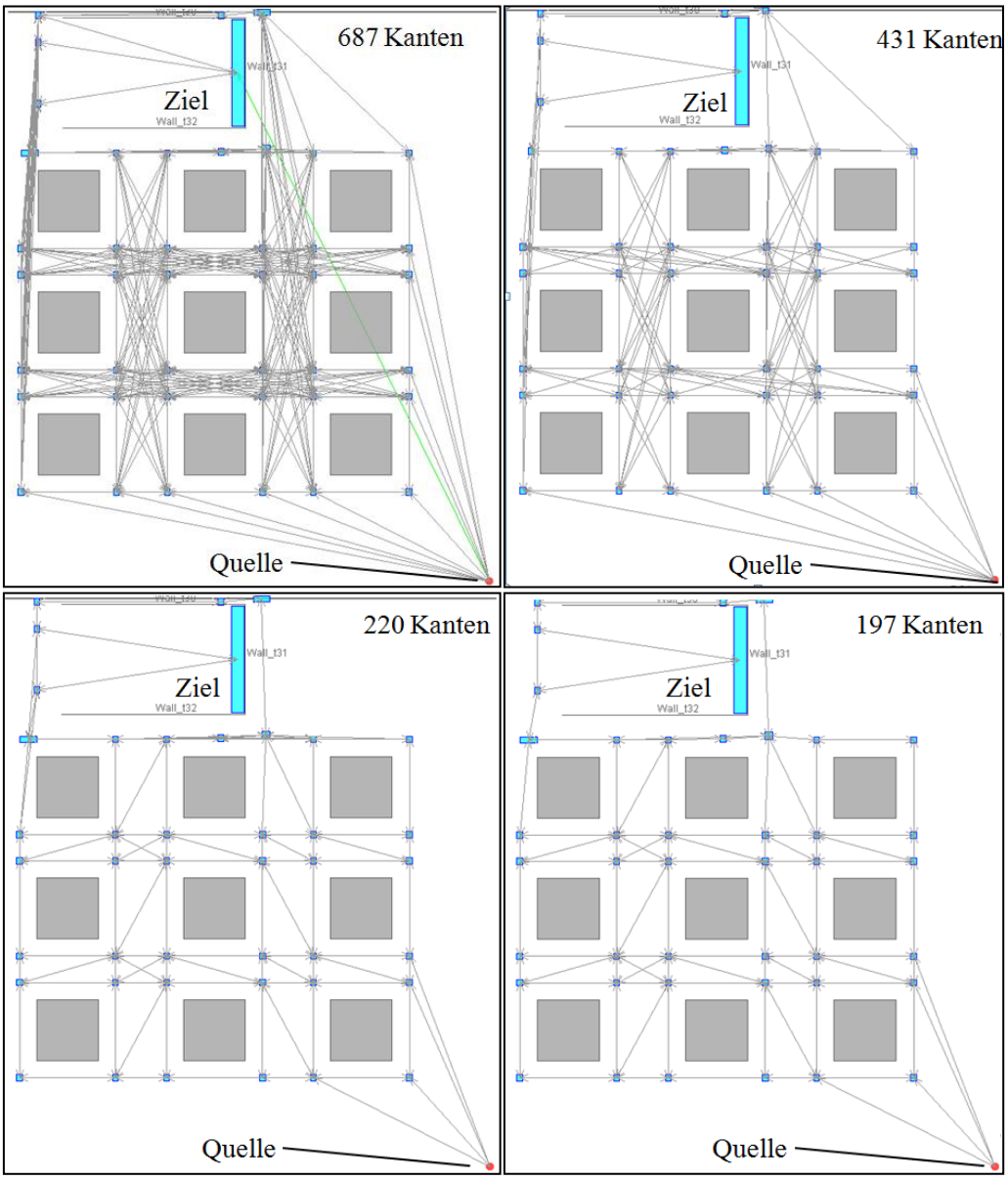


Abbildung 5 Anzahl der Kanten nach Konstruktion des Graphen mit unterschiedlichen Methoden: Sichtbarkeitsgraph (oben links), Sichtbarkeitsgraph mit Kantenreduktion (oben rechts),Sichtbarkeitsgraph mit räumlicher Datenstruktur (unten links) und mit räumlicher Datenstruktur sowie Kantenreduktion (unten rechts)

4 Vergleich der unterschiedlichen Graphen anhand der kürzesten Wege

Die mit den unterschiedlichen Verfahren erzeugten Graphen unterscheiden sich bezüglich ihrer Kantenanzahl wesentlich. Dies wirkt sich in starken Maß auf die Laufzeit des Personenstromsimulators aus, da in unserem Ansatz nicht nur zu Beginn sondern auch während der Simulation immer wieder nach kürzesten Wegen gesucht wird. Im einfachsten Fall und ohne Kantenreduktion zählen wir in unserem Beispielszenario 687 Kanten, die wir um gut 71 Prozent auf 197 Kanten reduzieren konnten (vgl. Tabelle 1). Es bleibt noch zu zeigen, dass die kürzesten Wege des so ausgedünnten Graphen nur unwesentlich länger sind als die im ursprünglichen Graphen.

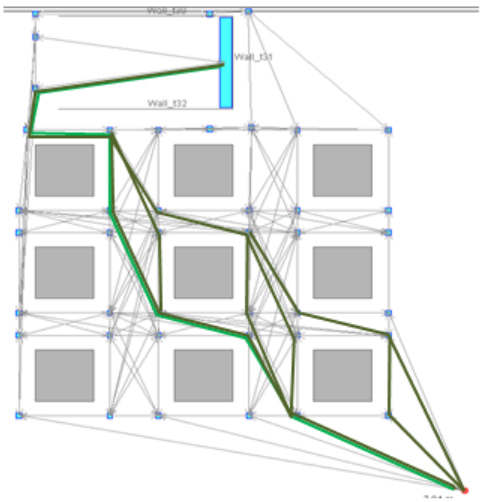
Dazu suchen wir mit Hilfe des Dijkstra-Algorithmus (Dijkstra 1959) nach den zehn kürzesten Wegen und vergleichen die gefundenen Wege bezüglich ihrer Länge.

In Tabelle 1 ist das Ergebnis der Suche nach dem jeweils kürzesten Weg aufgelistet. Es ist ersichtlich, dass die kürzesten Wege im ausgedünnten Graph um maximal 3,86 Prozent länger werden als in dem sehr dichten Graph bei einer Kantenreduktionsrate von gut 71 Prozent.

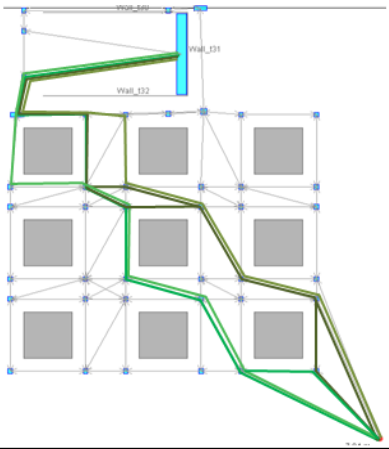
Tabelle 1: Vergleich des kürzesten Weges in den unterschiedlichen Graphen

Art des Graphen	Länge des kürzesten Weges	Prozentuale Veränderung der Länge	Prozentuale Veränderung der Kantenanzahl
Sichtbarkeitsgraph	78,33 m	100,00 %	100,00 %
Sichtbarkeitsgraph mit Ausdünnung	78,33 m	100,00 %	62,73 %
Sichtbarkeitsgraph mit räumlicher Datenstruktur	81,36 m	103,86 %	32,02 %
Sichtbarkeitsgraph mit räumlicher Datenstruktur und Ausdünnung	81,36 m	103,86 %	28,67 %

Um die räumliche Lage der gefundenen kürzesten Wege zu vergleichen, sind in Abbildung 6 jeweils die zehn kürzesten Wege dargestellt. Auf der linken Seite für den Sichtbarkeitsgraphen ohne räumlicher Datenstruktur, rechts für den Graphen, der mittels räumlicher Datenstrukturen erzeugt wurden. In beiden Graphen wurde eine Kantenreduktion durch Winkelprüfung durchgeführt. Es ist ersichtlich, dass die kürzesten Wege räumlich sehr ähnlich liegen. Da die Navigation der Fußgänger zwischen den Zwischenzielen auf der Ebene des zellulären Automaten erfolgt und auch von der Beeinflussung anderer Fußgänger abhängt, wird sich die zurückgelegte Wegstrecke der Fußgänger trotz der unterschiedlichen Graphen kaum unterscheiden.



Sichtbarkeitsgraph mit Ausdünnung	Länge des Weges (m)
Kürzester Weg	78,33
2. kürzester Weg	78,73
3. kürzester Weg	79,49
4.-7. kürzester Weg	79,66
8.-10. kürzester Weg	79,87



Sichtbarkeitsgraph mit räumlicher Datenstruktur und Ausdünnung	Länge des Weges (m)
1. kürzester Weg	81,36
2.-3. kürzester Weg	81,46
4. kürzester Weg	81,69
5. kürzester Weg	82,10
6. kürzester Weg	82,20
7. kürzester Weg	82,43
8. kürzester Weg	82,65
10. kürzester Weg	82,86

Abbildung 6 Vergleich der kürzesten Wege im Sichtbarkeitsgraphen ohne räumliche Datenstruktur (links) und mit räumlicher Datenstruktur (rechts)

5 Zusammenfassung und Ausblick

Wir haben in unserem Beitrag Ansätze zur automatischen Erzeugung eines Sichtbarkeitsgraphen aus der Geometrie eines Szenarios präsentiert. Diesen Graphen können wir dazu nutzen, die Fußgänger von ihrer Quelle zur ihrem Ziel auch in komplexen Geometrien zu navigieren. Neben der konventionellen Methode der Grapherzeugung haben wir eine weitere Methode vorgestellt, die mittels einer räumlichen Datenstruktur richtungsbezogenen Kanten findet und somit überflüssige Kanten vermeidet. Dabei unterscheidet sich die Länge der kürzesten Wege in den verschiedenen Graphen um einen geringen Prozentsatz, wobei die Kantenanzahl sehr deutlich reduziert werden kann.

Um eine bessere Annäherung an den kürzesten Weg zu bekommen, werden wir im nächsten Schritt die Erzeugung des Graphen mit räumlicher Datenstruktur optimieren. Dabei werden wir nicht – wie bisher – starr jeweils eine Kante zu den nächsten drei Nachbarn einfügen, sondern mit Hilfe von Sichtkegeln die nächsten Nachbarn suchen. Sobald ein nächster Nachbar gefunden wurde, wird ein Sichtkegel ausgehend vom untersuchten Knoten aufgespannt, innerhalb dessen kein weiterer Punkt mehr verbunden wird. Somit haben wir zum einen den Vorteil, dass wir alle Richtungen abdecken können, zum anderen sparen wir uns den nächsten Schritt der Ausdünnung, da durch Vorgabe der Sichtkegel Kanten mit Winkeln kleiner als ein unterer Grenzwert nicht erzeugt werden.

6 Referenzen

Beckmann, N, Kriegel, H, Schneider, R & Seeger, B 1990, 'The R*-tree: an efficient and robust access method for points and rectangles', *SIGMOD Rec.*, vol. 19, no. 2, pp. 322-331.
Berg, M de, van Kreveld, M & Overmars, M 2000, *Computational geometry*. *Algorithms and applications*. Chapter 15: Visibility Graphs, Springer, Berlin, pp. 307–317.

Burstedde, C, Klauck, K, Schadschneider, A & Zittartz, J 2001, 'Simulation of pedestrian dynamics using a two-dimensional cellular automaton', *Physica A: Statistical Mechanics and its Applications*, vol. 295, 3-4, pp. 507-525.

Dijkstra, EW 1959, 'A note on two problems in connexion with graphs', *Numer. Math.*, vol. 1, no. 1, p. 269.

Hartmann, D 2010, 'Adaptive pedestrian dynamics based on geodesics', *New Journal of Physics*, vol. 12, no. 4, p. 43032.

Klein, W, Köster, G & Meister, A 2010, 'Towards The Calibration of Pedestrian Stream Models'. *8th Int. Conf.on Parallel Processing and Applied Mathematics*, eds J Weglarz & B Wyrzykowski R. and Szymanski, Springer, Berlin

Kretz, T 2009, 'Pedestrian traffic: on the quickest path', *J. Stat. Mech.*, vol. 2009, no. 03, pp. P03012.

Li, Y & Höcker, M 2009, 'Heuristische Navigationsalgorithmen für Fußgägnersimulationen'. *Forum Bauinformatik 2009. 23. bis 25. September 2009, Universität Karlsruhe (TH)*, ed P von Both, Universitätsverl., Karlsruhe, pp. 25–36.

Liepert, T, Borrmann, A & Klein, W (eds.) 2008, *Ein Mulit-Speed-Personenstromsimulator auf Basis eines zellularen Automaten /// Forum Bauinformatik 2008. Junge Wissenschaftler forschen ; Tagungsband*, Techn. Univ. Inst. für Bauinformatik, Dresden.

Schadschneider, A, Klingsch, W, Kluepfel, H, Kretz, T, Rogsch, C & Seyfried, A 2009, 'Evacuation dynamics: Empirical results, modeling and applications', *Encyclopedia of Complexity and System Science*, pp. 3142–3176.