Zellulärer Zustandsautomat

3. Projekt zu Modellierung und Simulation

Daniel Graf, Dimitrie Diez, Arne Schöntag, Peter Müller

Inhaltsverzeichnis

1 Einführung

Im Zuge der ersten Studienarbeit wurde das Laufverhalten von Probanden in der Ebene und auf der Treppe untersucht. Basierend auf den Erkenntnissen bezüglich der individuellen Wunschgeschwindigkeiten werden in dieser Studienarbeit Personenbewegungen in der Ebene simuliert. Mit Hilfe solcher Simulationen können, beispielsweise im Zuge von Gebäudeplanungen unterschiedliche Gebäude- und Raumgestaltungen simuliert und hinsichtlich schnellster Räumungen im Krisenfall optimiert werden.

2 Beschreibung des Modells

Für die Simulation der Personenbewegungen wird ein zellulärer Zustandsautomat implementiert. Jede Zelle kann entweder leer oder durch eine Person oder ein Hindernis besetzt sein. Sie kann außerdem ein Ziel oder eine Quelle enthalten. Ein Hindernis kann von keiner Person betreten werden. Die Personen versuchen im Laufe der Simulation von der Quelle (Startposition) zum Ziel zu gelangen. Hierbei können sie sich in der sog. Moore-Umgebung bewegen. Es kann jede Zelle betreten werden, die frei ist und eine Nachbarzelle der aktuellen Zelle ist. Als Nachbarzelle wird jede Zelle bezeichnet, welche mit der aktuellen Zelle eine Kante oder Ecke teilt.
Die Ziele werden als attraktiv für die Personen angesehen. Die Personen steigern ihren persönlichen Nutzen, je näher sie dem Ziel kommen. In der Realität fühlt sich eine Person jedoch unwohl, wenn ihr eine fremde Person zu nahe kommt. Im Modell wird dies dadurch berücksichtigt, dass sich der Nutzen einer Person verringert, wenn sie einer anderen zu nahe kommt. Um die Simulation möglichst realistisch zu gestalten, bewegen sich die Personen mit einer individuellen Wunschgeschwindigkeit. Diese wird aus den Ergebnissen der ersten Studienarbeit ermittelt.
Das erläuterte Modell gibt die einzelnen Personenbewegungen aus und visualisiert diese

zusätzlich. Die Bewegung der Personen wird in drei getrennten Simulationen mit jeweils unterschiedlichen Algorithmen berechnet. Zunächst wird für jedes Feld der negative euklidische Abstand als Zielnutzen berechnet. Jede Person versucht durch die Wahl des umliegenden Feldes mit dem maximalen Wert seinen Nutzen zu steigern.
Darüber hinaus besteht die Möglichkeit die Personenbewegung mit Hilfe des Floor-Flooding, basierend auf dem Dijkstra Algorithmus zu ermitteln. Jede Person wählt von den anliegenden Feldern das Feld, welches die geringste Anzahl an notwendigen Bewegungen bis zum Ziel hat.
Als dritte Variante erfolgt die Ermittlung der Personenbewegung ebenfalls mit Hilfe des Floor-Flooding. Als Grundlage dient hierbei jedoch die Lösung der Eikonal-Gleichung, welche die Ausbreitung einer Welle beschreibt. Hierfür wird der Fast-Marching Algorithmus verwendet.
Im Laufe der Studienarbeit werden die beschriebenen Modelle anhand unterschiedlicher Tests verglichen und ausgewertet. Eine detaillierte Erläuterung der Tests ist im folgenden Kapitel 3.13 beschrieben.

3 Anforderungen/Requirements

3.1 Allgemeines

Es soll ein zellulärer Zustandsautomat implementiert werden. Die Software soll Personenbewegungen in der Ebene simulieren. Die unterschiedlichen Szenarien (Zustand der Zellen zu Beginn der Simulation), wie beispielsweise die Position der Hindernisse werden der Software als .png Bilddatei übergeben. Diese sind im Unterkapitel 3.13 näher erläutert. Es wird davon ausgegangen, dass sich zu Beginn der Simulation alle Personen im Startbereich (es soll möglich sein mehrere Startbereiche anzulegen) befinden.

3.2 Zellen

Das gesamte Feld (zweidimensionaler Bereich indem sich die Personen bewegen können) soll in Zellen eingeteilt werden. Die Größe des Feldes und die Anzahl an Zellen sollen der Software als Parameter übergeben werden.
Jede Zelle soll quadratisch sein. Jede Zelle kann verschiedene Zustände haben. Sie kann entweder leer oder besetzt sein. Ist eine Zelle besetzt, wird noch weiter unterschieden, ob sie durch ein Hindernis, das Ziel- (Personensenke) oder Startfeld (Personenquelle) oder durch eine Person besetzt ist.

3.3 Hindernisse

Es soll davon ausgegangen werden, dass jedes Hindernis eine oder mehrere Zellen ganz belegt. Hindernisse können von den Personen nicht betreten werden. Falls sich ein

Hindernis zwischen einer Person und ihrem Ziel befindet, ist die Person gezwungen um das Hindernis herum zu gehen.
In der Realität versucht eine Person einem Hindernis möglichst frühzeitig auszuweichen. Sie senkt somit ihren Nutzen, je näher sie einem Hindernis kommt. Dies wird im weiteren Verlauf dieser Studienarbeit nicht berücksichtigt.

3.4 Ziele

Durch das entsprechende Testszenario werden ein oder mehrere Ziele definiert. Ziele werden als attraktiv für die Personen angesehen. Jede Person erhöht ihren Nutzen, wenn sie sich dem Ziel nähert. Befinden sich mehrere Ziele im Feld, wird die Wahl des Ziels für jede Person abhängig vom Betriebsmodus bestimmt.

3.5 Personen

Jede Person soll in ihrem Startfeld starten und sich im Laufe der Simulation von Zelle zu Zelle auf ihr Ziel hin bewegen und dadurch ihren persönlichen Nutzen, je näher sie dem Ziel kommt, steigern. Jede Person kann sich in der Moore-Umgebung (8 mögliche Bewegungsrichtungen) bewegen. Die möglichen Bewegungsrichtungen sind in Abbildung 1 rot markiert.

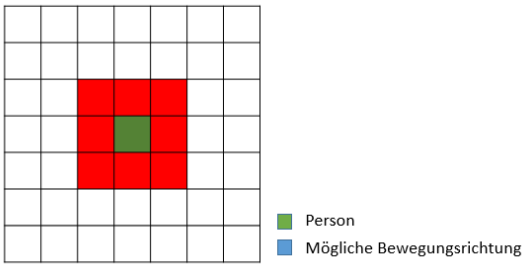

Abbildung 1: Moore-Umgebung

Die Personen können nur freie Zellen betreten. Jede Person wählt genau die Zelle als nächste Zelle, welche ihr den steilsten Nutzenanstieg ermöglicht. Der Nutzen der verschiedenen Zellen wird abhängig vom Betriebsmodus unterschiedlich berechnet. Darüber hinaus soll der Nutzen durch die Position der anderen Personen beeinflusst werden.

Hierfür wird zwischen zwei Bereichen, abhängig vom Abstand zur nächsten Person unterschieden. Beträgt der Abstand zu einer fremden Person weniger als $45cm$, wird der vertraute Bereich (intimate space) der Personen betreten. Beträgt der Abstand zwischen $45cm$ und $120cm$, wird der persönliche Bereich (personal space) der Personen betreten.

Für den vertrauten Bereich ist der Nutzenverlust (Nv(x)), abhängig vom Abstand x, nach Funktion 2 definiert. Für den persönlichen Bereich wird der Nutzenverlust (Np(x)), wiederum abhängig vom Abstand x zur anderen Person, durch die Funktion 1 definiert.

$$Np(x) = \mu * \exp(\frac{4}{(\frac{d(x)}{\delta_p * r})^2 - 1}) \tag{1}$$

$$Nv(x) = Np + \frac{\mu}{a_p} * \exp(\frac{4}{(\frac{d(x)}{\delta_v * r})^{2*b_p} - 1}) \tag{2}$$

$d(x)$ beschreibt den aktueller Abstand zur nächsten Person. Die folgende Tabelle enthält die Bedeutung der einzelnen Parameter und die entsprechenden Werte:

Parameter	Beschreibung	Wert
μ	Stärke des Nutzenverlustes	5.0
a_p	Dämpfung zwischen vertrautem und persönlichem Bereich	1.0
b_p	Stärke des Übergangs zwischen vertrautem und persönlichen Bereich	1
δ_p	persönlicher Bereich [m]	1.20
δ_v	vertrauter Bereich [m]	0.45
r	Radius einer Person [m]	0.2

Tabelle 1: Parameter und Werte für die Berechnung des Nutzenverlustes

Weitere Erläuterungen zu den Parametern, den Werten und den verwendeten Funktionen sind im Kapitel **??** beschrieben.

3.6 Geschwindigkeit der Personen

Jede Person soll eine individuelle Wunschgeschwindigkeit besitzen. Sie bewegt sich bei freier Bahn mit dieser Geschwindigkeit fort. Die Wunschgeschwindigkeiten sollen als normalverteilt angesehen werden. Als Mittelwert und Standardabweichung sollen die errechneten Werte aus der ersten Studienarbeit verwendet werden.

3.7 Zeitmodellierung

Die Simulation unterliegt keinen harten Echtzeitanforderungen. Sie soll für die gesamte Simmulationsdauer so schnell wie möglich durchgeführt werden. Es soll keine globale Uhr modelliert werden.
Die Visualisierung der Simulation soll hingegen die Bewegungen der Personen möglichst in Echtzeit darstellen. Sie unterliegt weichen Echtzeitanforderungen. Abweichungen aufgrund mangelnder Rechner-Ressourcen sind tolerierbar.

3.8 Parameter

Die Tabelle 2 zeigt alle möglichen Parameter, welche die Software steuern. Alle Parameter sind optional.

Parameter	Beschreibung	Default
–free-flow-velocity	Mittelwert der Wunschgeschwindigkeit [m/s]	1,0
–free-flow-deviation	Standardabweichung der Wunschgeschwindigkeiten [m/s]	0
–algorithm	Algorithmus (euclid, dijkstra, fast-marching)	dijkstra
–output-folder	Ordner in den der Output gespeichert wird	../data/
–cellsize	Größe der einzelnen Zellen [m]	1
–input-map	Pfad zum Bild der map (Auswahl des Testfalls)	default.png

Tabelle 2: Parameter und Defaultwerte

3.9 Berechnungen

Die Aufgabe der Software ist die Simulation von Personenbewegungen. Für möglichst genaue Zeitangaben wird die Java Klasse BigDecimal verwendet. Für die Division werden 32 Nachkommastellen berücksichtigt und als Rundungsmodus wird Half_Even verwendet, da dieser kumulative Fehler bei sich wiederholenden Berechnungen minimiert. Die anschließende Berechnung und Aufbereitung der Daten (insebesondere für den Test 3.13.6) erfolgt mit Mathematica.

3.10 Report

Alle errechneten Ergebnisse sollen über einen Report ausgeleitet werden. Der Report soll im .xml Format ausgeleitet werden um eine gute Weiterverarbeitung der Ergebnisse (Visualisierung) zu gewährleisten. Die Datei *output.xml* soll alle benötigten Informationen, wie beispielsweise die einzelnen Events der Personen, die durch den Algorithmus berechneten Entfernungen der einzelnen Zellen vom Ziel und eine Übersicht über den Zustand der einzelnen Zellen (Fieldmap), für Simulation beinhalten.
Die Wunschgeschwindigkeiten werden zufällig generiert. Diese sollen normalverteilt sein. Für eine Überprüfung der Zufallszahlgenerierung sollen die Wnschgeschwindigkeiten und die ID der entsprechenden Person im .csv Format in der Datei velocity.csv ausgegeben werden.

3.11 Berechung des Zielnutzens

Es sollen drei unterschiedliche Algorithmen für die Berechnung des Zielnutzens (Attraktivität) einer Zelle implementiert werden.

Algorithmus 1: Der Nutzen bzw. die Attraktivität jeder Zelle entspricht dem negativen euklidischen Abstand. Hierbei können keine Hindernisse umgangen werden. Jede Person versucht den Nutzen zu maximieren und wählt jeweils die Zelle mit maximalen Wert aus den umliegenden aus.

Algorithmus 2: Der Nutzen bzw. die Attraktivität jeder Zelle wird durch Floor-Flooding, basierend auf dem Dijkstra Algorithmus, berechnet. Für jede Zelle wird die geringste Anzahl an notwendigen Bewegungen bis zum Ziel ermittelt.

Algorithmus 3: Analog zum Algorithmus 2 wird auch in diesem hier der Nutzen bzw. die Attraktivität jeder Zelle mittels Floor-Flooding berechnet. Es wird jedoch der Fast-Marching Alforithmus (Lösung der Eikonal-Gleichung) als Grundlage verwendet.

3.12 Werte für die Auswertung

Alle in der Tabelle 3 aufgeführten Werte sollen auf Basis der Simulation ermittelt und ausgewertet werden. Einige Werte werden nur für bestimmte Tests benötigt.

Ermittelte Werte	Beschreibung	Wann benötigt
mittlere Geschwindigkeit	durchschnittliche Geschwindigkeit der Personen im Feld	Rimea-Test 4
Fluss (Personen/Meter/s)	Anzahl der Personen mit Geschwindigkeit $>$ 1m/s im Feld	Rimea-Test 4
Simulationsdauer	Dauer bis der Test beendet ist	Hühner Test, Evakuierung (2 Evakuierung (4

Tabelle 3: Werte für die Auswertung

3.13 Testszenarien für die Simulation

Die implementierten Algorithmen (vgl. Unterkapitel 3.11) sollen anhand unterschiedlicher Testszenarien verglichen und verifiziert werden. Die verschiedenen Szenarien sollen der Simulation als .png Datei übergeben werden.

3.13.1 Erstellung der Testszenarien

Es soll die Möglichkeit bestehen, unterschiedliche Testszenarien mittels eines Bildbearbeitungsprogrammes zu erstellen und an die Simulation (als .png Datei) zu übergeben. Bei der Erstellung ist sowohl die Anzahl der Personen, als auch die Anzahl der Hindernisse und Ziele (Personensenken) beliebig. Hinsichtlich der Farben der Objekte müssen jedoch

die in der folgenden Tabelle 4 beschriebenen Farbcodes verwendet werden. Jedes Pixel wird durch eine Zelle des Automaten repräsentiert.

Farbcode	Farbe	Bedeutung
#3F48CC	Blau	Ziel/ Personensenke
#000000	Schwarz	Ziel/ Personensenke
#FFFFFF	Weiß	Leere Zelle
#22B14C	Grün	Person

Tabelle 4: Werte für die Auswertung

3.13.2 Testszenario: Freier Fluss

Im Zuge dieses Tests soll überprüft werden, ob sich Personen bei freier Bahn (keine Hindernisse zwischen Start und Ziel) mit ihrer jeweiligen Wunschgeschwindigkeit auf das Ziel hin bewegen. Des weiteren soll überprüft werden, ob die Person den kürzesten Weg verwendet. Der Test soll mit allen 3 Betriebsmodi durchgeführt werden. Abbildung 2 stellt den Testaufbau schematisch dar.

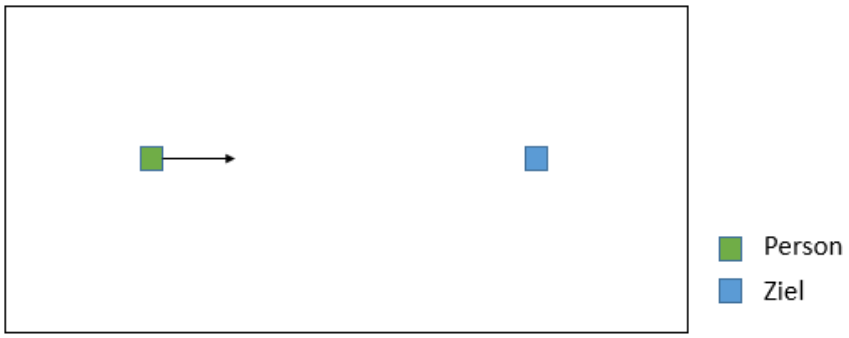

Abbildung 2: Testfall Freier Fluss (schematisch)

3.13.3 Testszenario: Hühnertest

Zwischen Start und Ziel soll ein U-förmiges Hindernis (Öffnung in Richtung des Startes) eingefügt werden. Abhängig vom Betriebsmodus wird ein unterschiedliches Ergebnis erwartet:
Betriebsmodus 1: Die Personen sollen sich auf dem kürzesten Weg auf das Ziel zubewegen. Sobald sie das Hindernis erreicht haben, sollen sie stehen bleiben. Erwartungsgemäß wird der Test nicht erfolgreich verlaufen.
Betriebsmodi 2 und 3: Die Personen sollen das Hindernis umgehen und das Ziel erreichen.

Bei einem „Feststecken im Hindernis“(wie im Betriebsmodus 1 gefordert) wird der Test als fehlgeschlagen gewertet. Abbildung 3 stellt den Testaufbau schematisch dar.

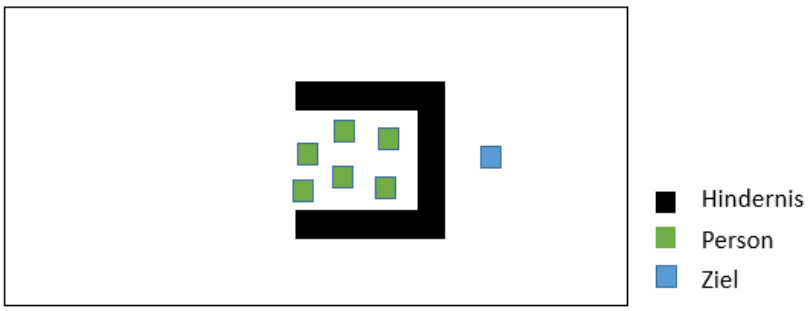

Abbildung 3: Testfall Hühnertest (schematisch)

3.13.4 Testszenario: Evakuierung eines Raumes mit 2 Türen

Die Personen befinden sich zu Beginn in einen von Hindernissen umgebenen, quadratischen Raum. An 2 Seiten sind Öffnungen platziert (freie Zellen). Hinter diesen Öffnungen befindet sich jeweils ein Ziel. Im Laufe des Tests sollen die Personen durch die beiden Engstellen zum Ziel gelangen.
Dieser Test soll mit den Betriebsmodi 2 und 3 in unterschiedlichen Versionen durchgeführt werden. Zunächst sollen sich die Türen an benachbarten Seiten befinden. Hierbei sollen sie einmal mit geringem und einmal mit sehr großem Abstand zueinander plaziert werden. Anschließend sollen die Türen an gegenüberliegenden Seiten plaziert mittig werden. Unterschiede hinsichtlich der Evakuierungsdauer sind festzuhalten. Die Abbildungen 4, 5 und 6 zeigen die unterschiedlichen Testversionen schematisch.

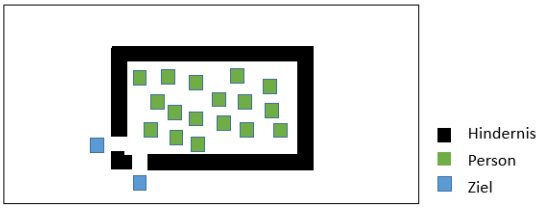

Abbildung 4: Testfall Evakuierung eines Raumes mit 2 Türen (Nebeneinander, geringer Abstand)

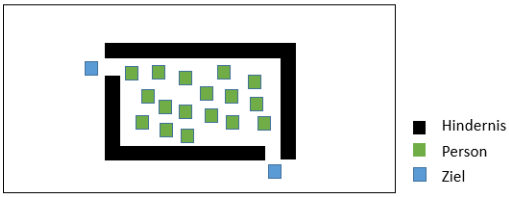

Abbildung 5: Testfall Evakuierung eines Raumes mit 2 Türen (Nebeneinander, großer Abstand)

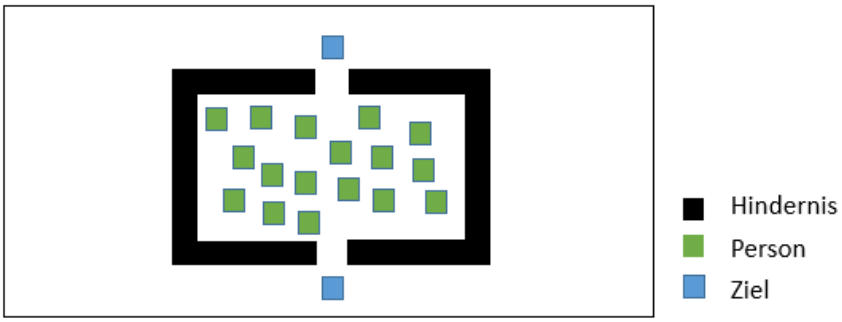

Abbildung 6: Testfall Evakuierung eines Raumes mit 2 Türen (Gegenüber)

3.13.5 Testszenario: Evakuierung eines Raumes mit 4 Türen

Dieser Testfall entspricht der in Kapitel 3.13.4 beschriebenen Evakuierung mit 2 Türen. In diesem Fall befindet sich jedoch an jeder Raumseite eine Tür. Die Türen sollen sich in einer ersten Version mittig bezogen auf die jeweilige Raumseite befinden. Darüber hinaus sollen in einer weiteren Version die Auswirkungen ermittelt werden, wenn jeweils die 2 Türen an benachbarten Seiten mit minimalen Abstand zueinander plaziert werden. Unterschiede hinsichtlich der Evakuierungsdauer sind festzuhalten. Aufgrund der Analogie zur Evakuierung mit 2 Türen wird auf eine graphische Veranschaulichung des Testaufbaus verzichtet.

3.13.6 Testszenario: Fundamentaldiagramm - Rimea-Test 4

In diesem Testfall soll ein $65m$ langer und $12m$ breiter Gang angelegt werden. Auf der einen Seite des Ganges befindet sich die Personenquelle, auf der anderen Seite die Personensenke. Im Laufe des Testes sollen die Personen den Gang entlangschreiben. Die Anzahl der Personen ist zu variieren. In einer weiteren Version des Tests sollen Personen, welche die Personensenke erreicht haben, erneut bei der Personenquelle starten (Endlosschleife). Abbildung 7 zeigt den Testaufbau schematisch.

3.13.7 Testszenario: Engstelle (unidirektional)

In diesem Testfall soll, analog zum Rimea-Test 4, ein $65m$ langer und $12m$ breiter Gang angelegt werden. Am Ende des Ganges befindet sich eine Engstelle mit $6m$ Breite und dahinter die Personensenke. Es soll mit variierender Anzahl an Personen getestet werden, wie lange es dauert bis die Engstelle durchschritten wurde. Der schematische Testaufbau ist in Abbildung 8 aufgeführt.

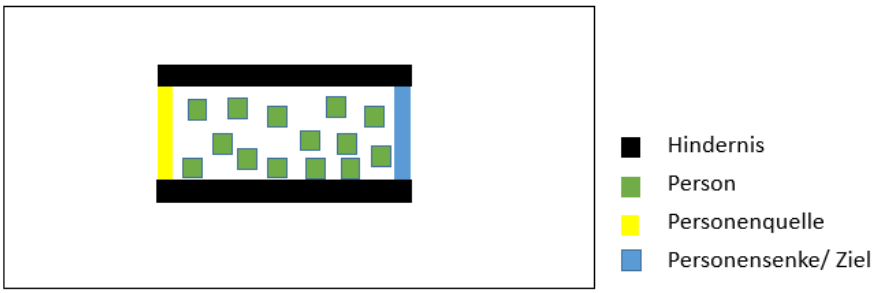

Abbildung 7: Testfall Rimea 4 (schematisch)

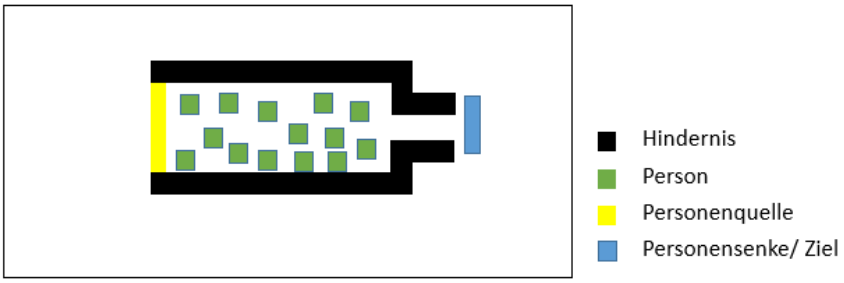

Abbildung 8: Testfall Engstelle (schematisch)

3.14 Visualisierung

Die Visualisierung soll die errechneten Personenbewegungen graphisch darstellen. Als Grundlage soll der ausgegebene Report (vgl. Abschnitt 3.10) dienen. Konkrete Anforderungen bezüglich der Visualisierung sind in Abbildung 9 abgebildet.

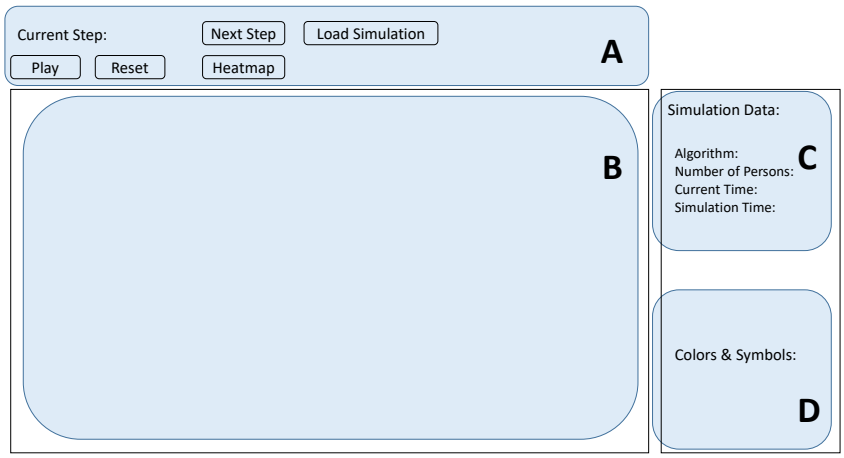

Abbildung 9: Visualisierung (schematisch)

Die Oberfläche der Visualisierung soll aus 4 verschiedenen Teilen bestehen, welche in der Abbildung durch die blauen Kästen A-D hervorgehoben wurden. Im Teil A sollen Konfigurationsmöglichkeiten für die Visualisierung aufgeführt sein. Beispielsweise soll zwischen einer normalen Darstellung der Simulation und einer Heatmap gewechselt werden können. Darüber hinaus soll die Möglichkeit bestehen, die Visualisierung zu stoppen und schrittweise fortzusetzen.
Teil B enthält die eigentliche Visualisierung der Personen, Hindernisse und Ziele. In Teil C sind alle Informationen bezüglich der Visualisierung aufgeführt und Teil D enthält eine Legende der verwendeten Symbole und Farben.

4 Softwaredesign

Der Aufbau der Anwendung wurde im Team diskutiert und anschließend mittels UML spezifiziert.

5 Softwaretest

6 Vergleich der Algorithmen

Die Algorithmen werden in Abbildung 10 in Vergleich gesetzt. Dafür wird eine Karte erstellt mit einem Raum, einer Tür und einem Hindernis vor der Tür. Die Karten werden jeweils nach 1, 20, 100, 1000 Schritten geplottet. Als letztes wird noch das Gesamtergebnis dargestellt.

6.1 Euklid

Um den Euklidischen Abstand zu berechnen, benötigt man lediglich die Position des Ziels, die Position der aktuellen Zelle und die Zellbreite. Man geht alle Zellen der Reihe nach durch und berechnet den Abstand der aktuellen Zelle zum Ziel. Die Suche nach noch nicht berechneten Zellen läuft nicht von oben nach unten sondern mit der Funktion „parallelStream()“. Diese teilt das Feld in Teilfelder auf und sucht parallel nach freien (noch nicht berechneten) Feldern. Dadurch entstehen die in Abbildung 10 sichtbaren Stufen. [1] Darüber hinaus ist zu erkennen, dass das Hindernis vor der Tür für den Euklid Algorithmus keinen Einfluss hat. Dieses Hindernis wird bei der Berechnung ignoriert.

6.2 Dijkstra

Beim Dijkstra Algorithmus geht man ausgehend vom Ziel immer ein Feld weiter und berechnet den Abstand des aktuellen Feldes anhand des Abstand des Nachbarfeldes. Dadurch wird die Karte ausgehend vom Ziel aufgebaut wie in Abbildung 10 zu sehen ist. Es ist auch zu erkennen, dass der Dijkstra Algorithmus die Abstände um das Hindernis berechnet. Das Hindernis wird also in die Berechnung einbezogen.

6.3 Fast Marching

Ähnlich wie beim Dijkstra arbeitet der Fast Marching Algorithmus vom Ziel aus. Das Zielfeld wird als Basis angenommen und die Nachbarfelder um das Ziel werden als „considered“also als „in Betracht kommend“markiert. Das in Betracht kommende Nachbarfeld mit dem besten Wert wird als „accepted“also als „akzeptiert“markiert. Danach wird für alle Nachbarn dieses akzeptieren Feldes die Entfernung mittels der Eikonalgleichung berechnet. Auch hier ist also die Berechnung abhängig von den vorher berechneten Feldern. Es ist deutlich zu erkennen, dass der Fast Marching Algorithmus eine deutlich rundere Karte erzeugt, als der Dijkstra.

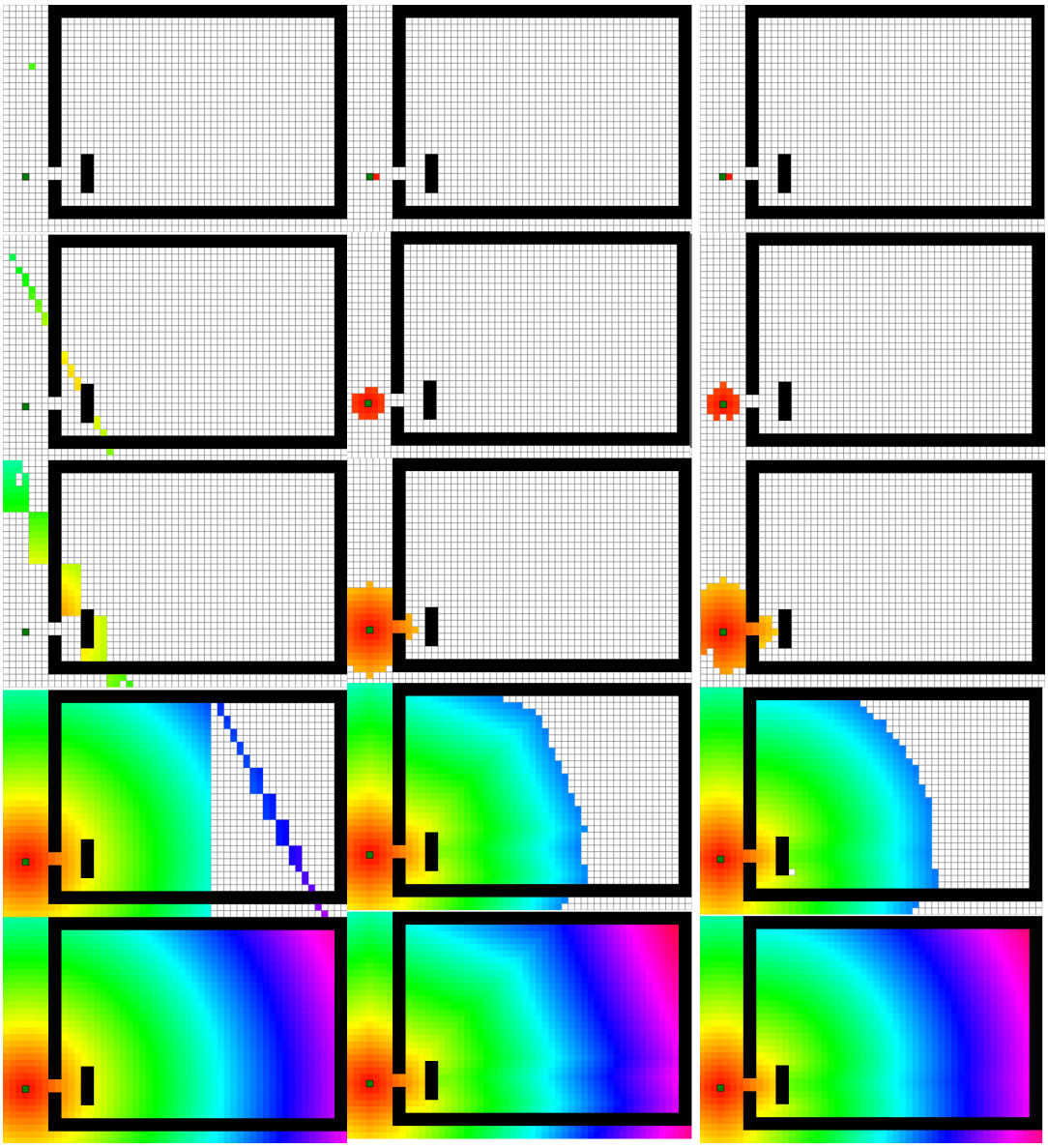

Abbildung 10: Die berechnung der Abstände mit verschiedenen Algorithmen im Vergleich. Euklid (links), Dijkstra (mitte), Fast Marching (rechts), jeweils von oben nach unten nach 1, 20, 100, 1000 Schritten und Gesamtbild

6.4 Einfluss der Zellgröße auf die Abstandsberechnung

Bei der Berechnung der Abstände hat die Wahl der Zellgröße je nach Algorithmus einen entscheidenden Einfluss auf die Ergebnisse der Simulation. Die Zellgröße gibt nicht nur die minimale Schrittweite vor, sondern hat auch Einfluss auf die Genauigkeit der Nutzenberechnungen. An dieser Stelle wird der Euklid Algorithmus dem Fast Marching Algorithmus gegenübergestellt. Um einen Vergleich darzustellen wurde ein Quadratisches Feld angelegt. Die Zellgröße des Feldes sowie die Anzahl der Felder wurden so variiert, dass die Entfernung von der rechten unteren Ecke zur linken oberen Ecke (am weitesten entfernter Punkt) immer den gleichen Abstand hat. Es wird also die Diagonale des Rechtecks berechnet.

6.4.1 Grafische Überprüfung

Die Abbildung 11 Zeigt den Vergleich beider Algorithmen bei der der Abstandsberechnung. Üblicherweise wird die Farbskala so gewählt um alle Werte vom kleinsten Abstand zum weitesten Punkt abzubilden. Um den Vergleich an dieser Stelle deutlich zu machen, wurde für beide Bilder die selbe Farbskala verwendet. Mit Bloßem Auge ist die Differenz nur schwer zu erkennen. Es ist jedoch sichtbar, dass das Feld, welches mit dem Euklid Algorithmus berechnet wurde, einen deutlicheren Bogen aufweist.

Eine Möglichkeit die Unterschiede zu visualisieren ist es, die Bilder mittels einer Differenzbildung der Beiden Bilder. Dafür wird ein Bild als Basis hergenommen und das zweite Bild als eine Ebene darüber eingefügt. Die Farben der zweiten Ebene werden von der ersten Ebene Subtrahiert. Gleichen sich die Farben beider Ebenen, so ist das Ergebnis schwarz. Dieser Vergleich macht nur Sinn, wenn für beide Darstellungen die selbe Farbskala verwendet wurde, also die Entfernungen mit dem gleichen Farbcode codiert wurden. Die Abbildung 12 zeigt die Differenzmenge der in Abbildung 11 dargestellten Karten. An diesem Bild ist deutlich zu erkennen, dass die Zellen, welche sich nur in X und nur in Y Richtung vom Ziel entfernen schwarz sind. Es gibt also keinen Unterschied zwischen dem Euklidischen Abstand und dem Abstand welcher mittels Fast Marching berechnet wurde, bei einer Betrachtung der Zellen nur in X oder nur in Y Richtung. Entfernt man sich schräg vom Ziel (Also X und Y Anteil gleichzeitig), sieht man deutlich Differenzen der beiden Algorithmen.

Um dies nochmals zu verdeutlichen werden die Algorithmen in Abbildung 13 mit einer kleineren Zellgröße verglichen. Auch hier wurde auf eine einheitliche Farbskala geachtet. Die Differenzbildung ist in Abbildung 14 zu sehen. Es ist eine Art Kegel zu erkennen. Die Kanten unten und rechts sind Schwarz, da sich die Abstände beim Euklid und Fast Marching gleichen. Näher zur Diagonalen wird das Bild heller, da es einen größeren Unterschied gibt.

6.4.2 Rechnerische Überprüfung

Eine rechnerische Überprüfung wird auf den diagonalen Abstand durchgeführt. Die Tabelle 5 zeigt die Ergebnisse dieses Experiments. Die Cellsize gibt dabei die Kantenlänge einer quadratischen Zelle an. Wichtig ist der Wert „Distance in X and Y“, welcher besagt, dass das Ziel in X und in Y Richtung immer jeweils 14 Zellen entfernt ist. Nach der Formel wurde der Abstand von Ziel zum am weitesten entfernten Punkt (also diagonal) berechnet:

$$a = \sqrt{x^2 + y^2} = \sqrt{(14\ m)^2 + (14\ m)^2} = 19,799\ m$$

Wie in der Tabelle zu sehen ist, berechnet der Euklid Algorithmus den Abstand sehr präzise. Da der Euklid Algorithmus kein „Gedächtnis“ hat also ohne Einfluss vorheriger Zellen berechnet und die oben genannte Formel anwendet ist der Algorithmus gut geeignet um die exakte Entfernung zu berechnen. Interessant in der Tabelle ist vor allem die Abweichung von 1,45 m, welche beim Fast Marching Algorithmus entsteht. Diese Abweichung entspricht einem Fehler von von 7,3 % bezogen auf die tatsächliche Entfernung. Verringert man die Zellgröße, so konvergiert der Fast Marching Algorithmus gegen den euklidischen Abstand und der Fehler gegen 0. Die prozentualen Fehler in Abhängigkeit der Zellgröße sind in Abbildung 15 dargestellt.

7 Einfluss der Zellgröße auf die maximale Dichte

Darüber hinaus hat die Zellgröße direkten Einfluss auf die Dichte $[\frac{Personen}{m^2}]$ in einem Feld. Würde man die Zellgröße von 2 m^2 wählen, erhielte man eine maximale Dichte von 0,5 $\frac{1}{m^2}$. Es ist darauf zu achten, dass die Zellgröße angemessen gewählt wird. Bei weiteren Versuchen wurde stets eine Zellbreite von 0,4 bis 0,5 m verwendet, sofern nichts anderes angegeben.

7.1 Verifikation

7.2 Validation

8 Ausblick und Fazit

Literatur

[1] Oracle Documentation. Java parallelism, https://docs.oracle.com/javase/tutorial/collections/streams/parallelism.html, 2017.

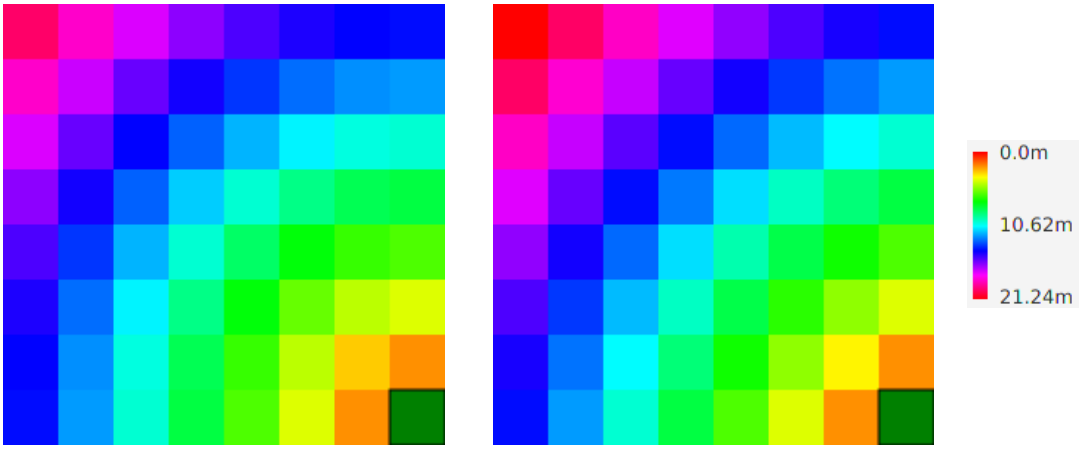

Abbildung 11: Berechnung der Abstände mittels Euklid (links) und Fast Marching (rechts) an einem Feld mit einer Zellbreite von 2 m. Dunkelgrün: Ziel, Rot: am weitesten entfernter Punkt im Feld

Cellsize [m]	2	1	0,5	0,25	0,125	0,0625	0,03125
Cells in X and Y	7	14	28	56	112	224	448
Distance in X and Y [m]	**14**	**14**	**14**	**14**	**14**	**14**	**14**
Tatsächliche Entfernung [m]	19,799	19,799	19,799	19,799	19,799	19,799	19,799
Euklid [m]	19,799	19,799	19,799	19,799	19,799	19,799	19,799
Fast Marching [m]	**21,245**	**20,717**	**20,363**	**20,137**	**19,997**	**19,913**	**19,863**
Diff [m]	1,45	0,92	0,5645	0,3380	0,1979	0,1138	0,0644
Error [%] diff/distance	**7,30**	**4,64**	**2,85**	**1,71**	**1,00**	**0,57**	**0,33**

Tabelle 5: Vergleich der Algorithmen Fast Marching und Euklid. Cellsize ist die Kantenlänge einer quadratischen Zelle in m (Zellbreite)

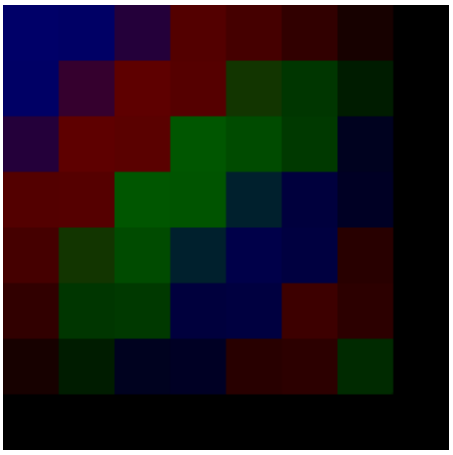

Abbildung 12: Differenzbildung der beiden Karten Euklid und Fast Marching bei einer Zellbreite von 2 m

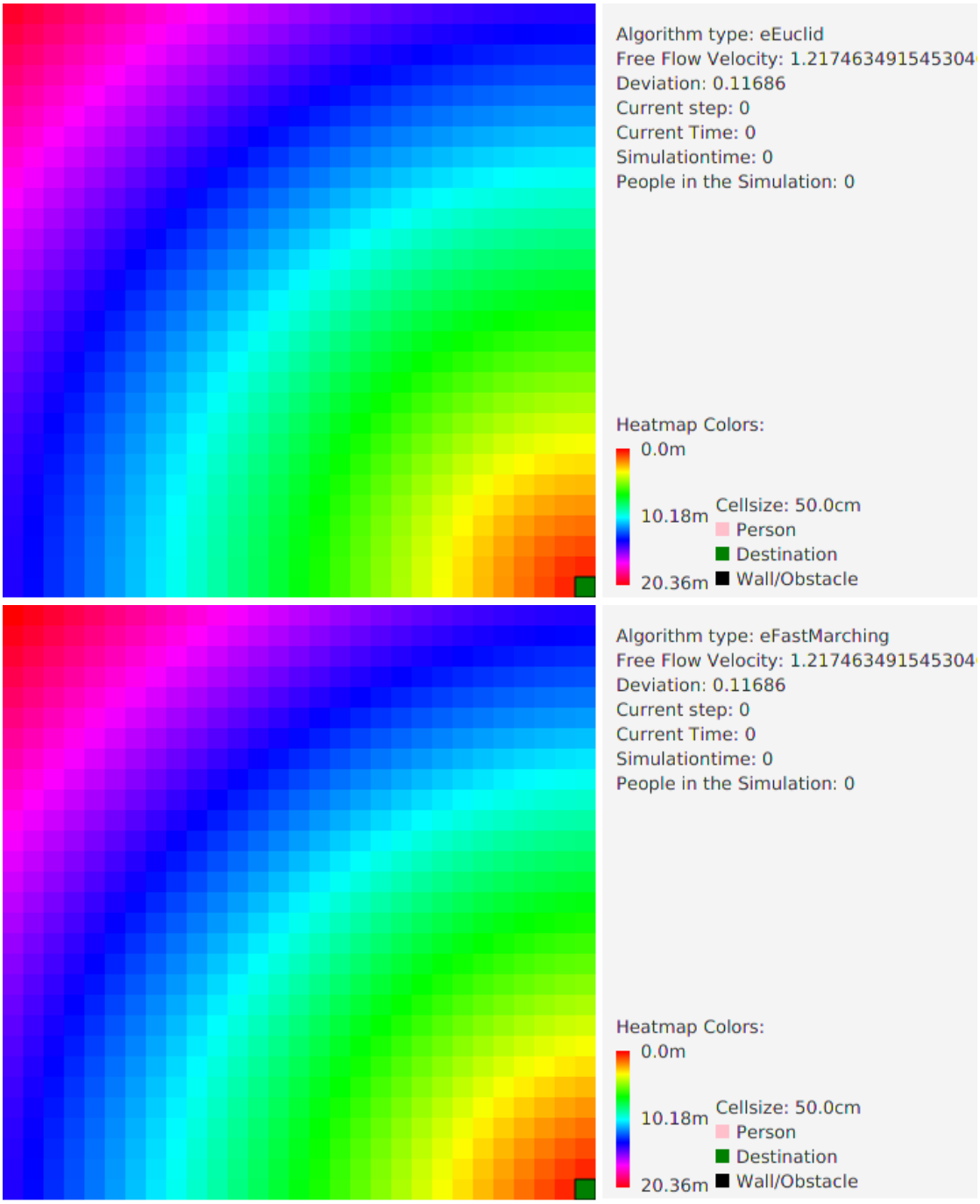

Abbildung 13: Berechnung der Abstände mittels Euklid (oben) und Fast Marching (unten) an einem Feld mit einer Zellbreite von 0,5 m

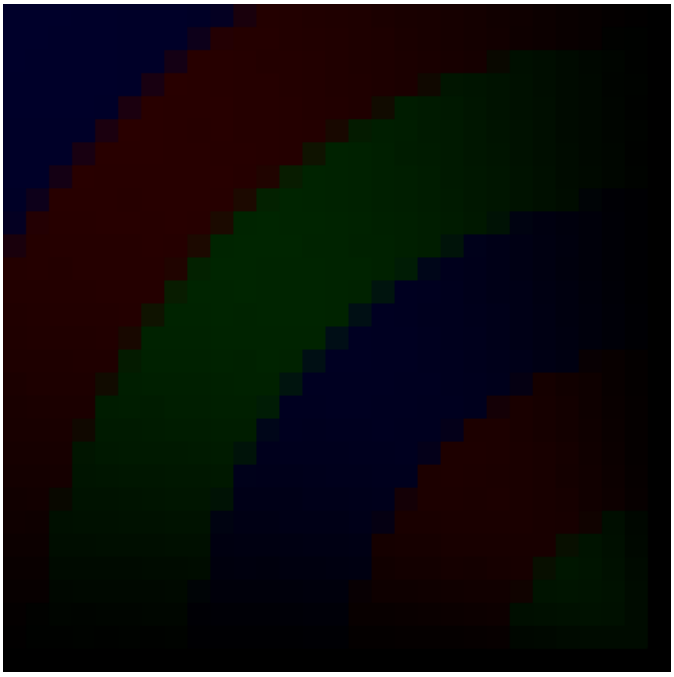

Abbildung 14: Differenzbildung der beiden Karten Euklid und Fast Marching bei einer Zellbreite von 0.5 m

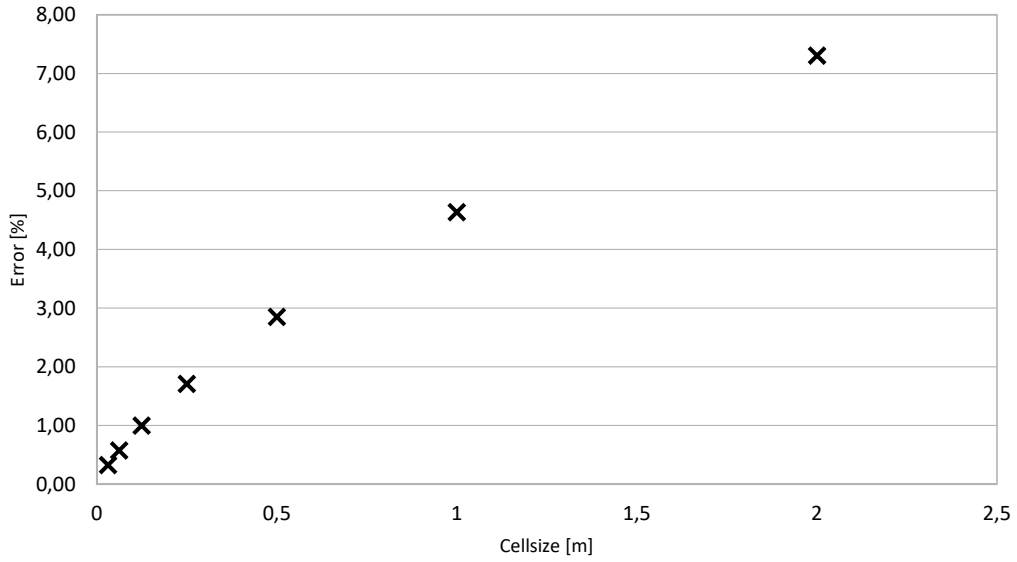

Abbildung 15: Prozentualer Fehler für die diagonale Entfernung des Fast Marching Algorithmus in Abhängigkeit der Zellbreite.