Inhaltsverzeichnis

1 Einleitung 2

2 Datenaufnahme 3

2.1 Untersuchte Schiffe 3

2.2 Rahmenbedingungen an Bord 6

2.3 Fahrzeugtypen 7

2.4 Fahrverhalten 8

2.4.1 Allgemeines 8

2.4.2 Auswertung 9

2.4.3 Fahrzeuglängen 10

2.4.4 Parkabstände 10

2.4.5 Begleiteter Verkehr 11

2.4.6 Nicht begleiteter Verkehr 13

2.5 Messfehler 14

3 Modellentwicklung 15

3.1 Anforderungen 15

3.2 Bestehende Modelle 15

3.3 Implementierung 16

3.3.1 Zusammenfassung 16

3.3.2 Routen 16

3.3.3 Fahrzeuge 18

3.3.4 Simulationsablauf 19

3.3.5 Beladungsalgorithmus 20

3.3.6 Abstandberechnung 22

3.3.7 Simulation der Fahrzeugbewegung 23

3.3.8 Erstellung einer Simulation 24

4 Programmstruktur 25

4.1 Klassen 25

4.1.1 TRoute 25

4.1.2 TVehicle und TTrailer 25

4.1.3 TRandom 26

4.1.4 TDXFHandler 26

4.1.5 TForm 26

5 Ergebnisse 27

5.1 Zusammenfassung 27

5.2 Kalibirierung und Vergleich mit der Scandlines-Fähre 27

5.3 Vergleich von Entladetaktiken 28

5.4 Ausblick 29

6 Literatur 30

7 Danksagung 31

Anhang 32

1 Einleitung

Während des Entwurfs von RoRo-Fähren ist es kaum möglich, die Beladedauern vorher zu sagen. Diese sind für den Reeder besonders dann interessant, wenn eine Fähre zwei Häfen in kurzer Taktzeit miteinander verbinden soll. Um zwischen den höheren Baukosten einer Doppelendfähre und den längeren Umschlagszeiten einer Fähre mit nur einer Laderampe abwägen zu können, ist zum Beispiel die Be- und Entladedauer von signifikanter Bedeutung. Doch nicht nur solch grundsätzliche Entscheidungen bezüglich des Schiffstyps, sondern auch Detailfragen, wie zum Beispiel die Anordnung einer Rampe im Schiff den Ladevorgang beeinflusst, können ohne eine vorherige Analyse mittels einer Simulation nur sehr grob abgeschätzt werden.

Um eine Aussagen über die Ladevorgänge treffen zu können, wurde von der Howaldtwerke Deutsche Werft AG (*HDW*) [1] in Zusammenarbeit mit der TraffGo GmbH (*TraffGo*) [2] das Projekt SimRoP (*Simulation of RoRo Processes*) gestartet. Das Ziel des Projektes ist die neue Entwicklung einer Verkehrssimulation zur Untersuchung von Ladevorgängen. Ziel des Projekts war die Beobachtung realer RoRo-Ladevorgänge auf Fähren, um eine Grundlage sowohl für das Fahrverhalten als auch für die Beladungstrategie zu finden. Anhand der gewonnen Daten sollte ein Modell zur Simulation des Ladeverkehrs entwickelt und in einem Programm umgesetzt werden.

Bestehende Straßenverkehrssimulationen wurden für Autobahn- und Stadtverkehre entwickelt und können damit die Komplexität eines Ladeprozesses nicht ausreichend detailliert abbilden. Außerdem war eine Benutzerumgebung mit sehr großer Flexibilität gefordert, die es ermöglicht, schnell und einfach verschiedene Geometrien einzubinden und die Simulationsumgebung zu modellieren.

2 Datenaufnahme

Grundlegend für die Modellbildung sind Kenntnisse über den Ladevorgang an Bord sowie Daten des Fahrverhaltens der Fahrzeuge bzw. der Fahrer. Da für das Fahrverhalten von Fahrzeugen im Ladeprozess (kleine Geschwindigkeiten) keine Informationen vorlagen, mussten Messungen der Verladevorgänge durchgeführt werden. Anhand der Messungen sollten die verschiedenen Fahrzeugtypen im Hinblick auf die Modellbildung kategorisiert und auf ihr unterschiedliches Fahrverhalten hin untersucht werden. Die währned der Datenaufnahme mit den Lademeistern an Bord geführten Interviews ergaben nicht nur Aufschluss über die Vorgehensweise und Taktik, sondern auch über die Einschränkungen und Randbedingungen beim Ladungswechsel.

Zur Untersuchung wurden an markanten Punkten Kameras aufgestellt, die den Verkehrsfluss filmten. Anhand der Videos konnten Daten wie Fahrzeuggeschwindigkeiten und Fahrzeugabstände ausgewertet werden. Eine genaue Beschreibung der Vorgehensweise erfolgt in Kapitel 2.4.2.

Neben den Geschwindigkeiten wurden auch die Abstände der parkenden Fahrzeuge sowie die Längen der Pkw gemessen. Eine detaillierte Beschreibung der Messungen ist in den Beschreibungen zu den Messungen [3],[4] und[5] zu finden.

2.1 Untersuchte Schiffe

Die Untersuchungen fanden auf drei Schiffen statt, um einen Überblick über verschiedene Verladesysteme zu erhalten. Alle Fähren besitzen eine Rampe sowohl im Bug- als auch im Heckbereich, so dass meist ein Wendevorgang im Schiff entfällt.

a) Stena Line – *Stena Germanica*

Abb. 2-1: Die *Stena Germanica* der Stena Line (Foto: Stena Line).

Die RoPax-Fähre *Stena Germanica* der Stena Line Reederei [6] pendelt täglich zwischen Göteborg und Kiel. Für die Beladung stehen die Decks 3, 4, 5 und 6 zur Verfügung, davon sind die beiden Decks 4 und 6 in Segmente aufgeteilte Hängedecks,

deren Segmente je nach Bedarf einzeln herabgelassen werden. Bei herabgelassenen Hängedecks reicht die Deckshöhe auf allen Decks nur für Pkw aus. Werden diese nicht eingesetzt ist die Höhe unter ihnen für Lkw ausreichend. Da die Lkw auf Deck 3 gestaut werden, kommt das Hängedeck 4 meist nicht zum Einsatz. Die Be- und Entladung der unteren beiden Decks in Kiel erfolgt über eine Rampe im Bug, die oberen Decks werden über eine seitliche Luke im Heck entladen. Beim Ladungswechsel in Göteborg verlassen die Fahrzeuge das untere Deck über eine Heckrampe, die oberen beiden Decks werden über eine seitliche Luke im Bugbereich entladen.

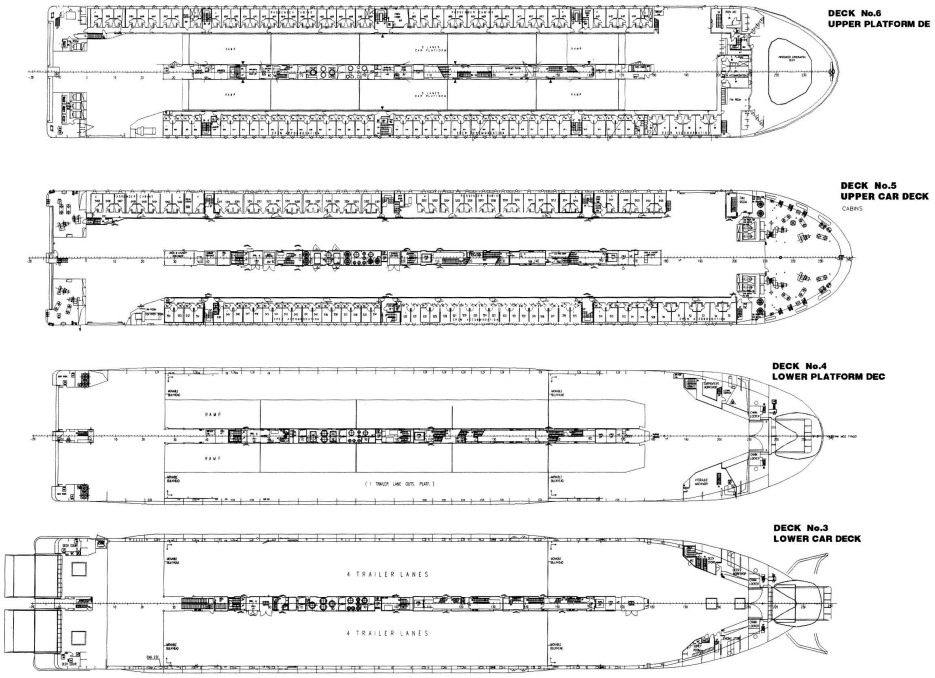

Abb. 2-2: Generalplan der *Stena Germanica* (Quelle: Stena Line).

b) Scandlines – *Schleswig Holstein*

Abb. 2-3: Die Doppelendfähre *Schleswig-Holstein*. (Foto: Scandlines).

Scandlines [7] bedient mit mehreren baugleichen Doppelendfähren den Verkehr zwischen Puttgarden und Rødby. Die untersuchte Fähre *Schleswig Holstein* besitzt zwei Fahrzeugdecks. Das obere ist auf Grund der Höhe und der Fahrspurbreite nur für Pkw geeignet, während das untere Deck von hohen Fahrzeugen und Fahrzeugen

mit Anhängern genutzt wird. Doppelendfähren besitzen Rampen an Bug und Heck, so dass die Fahrzeuge während des Be- und Entladens nicht wenden müssen. Dadurch erfolgt der Ladungswechsel innerhalb weniger Minuten.

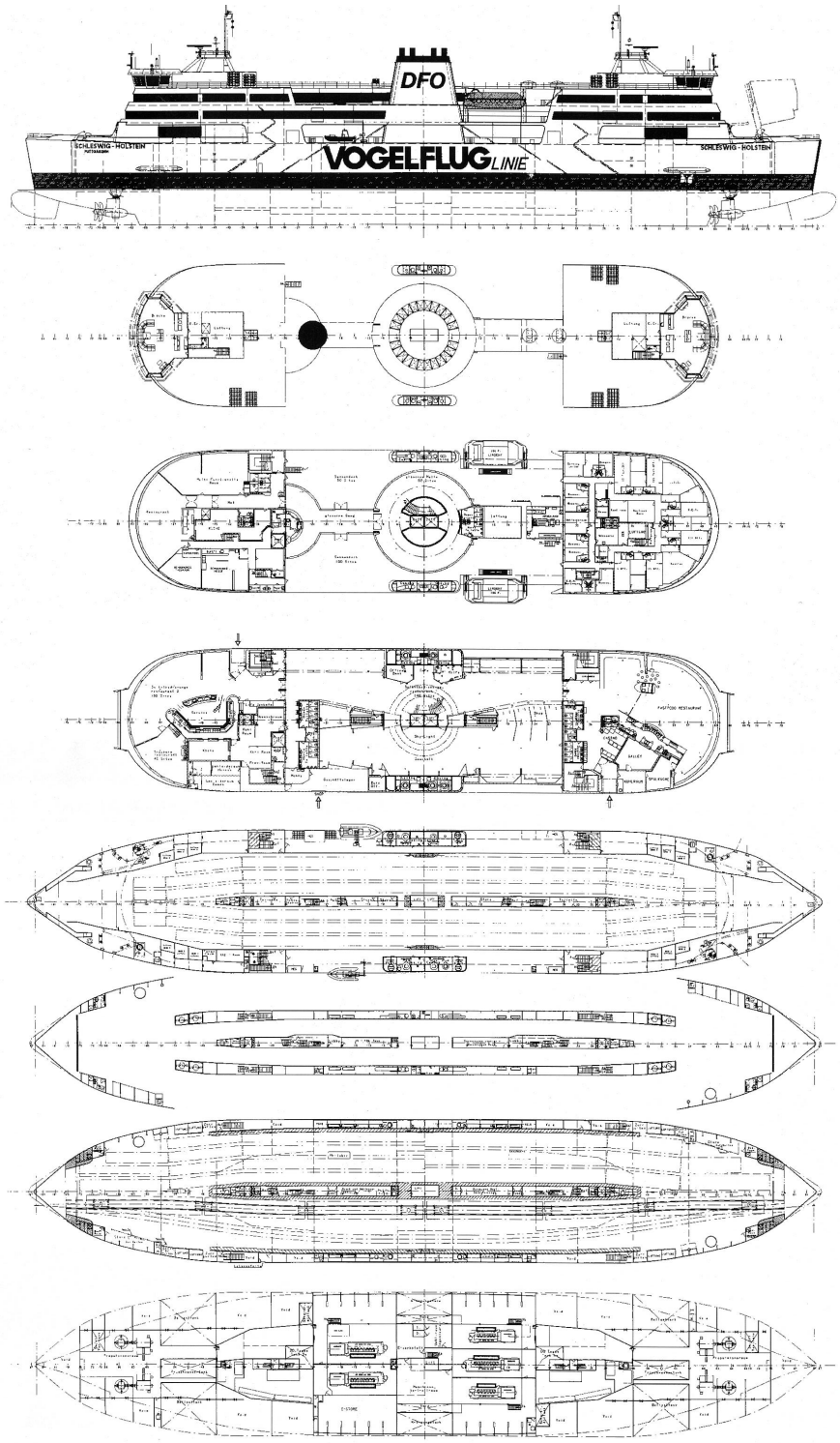

Abb. 2-4: Generalplan der *Schleswig-Holstein* (Quelle: Scandlines).

c) Superfast – *Superfast 8*

Abb. 2-5: Die Ro-Pax-Fähre *Superfast 8*. (Foto: Superfast).

Die *Superfast 8* fährt im RoPax-Verkehr zwischen Rostock und Hanko in Finnland. Sie wird über zwei Rampen im Bug und drei Rampen im Heck be- und entladen. Bei der Beladung stehen zwei Decks für Lkw und zwei Decks für Pkw zur Verfügung. Die Pkw-Decks liegen unter dem Schottendeck und werden über eine Rampe im Schiff vom unteren Lkw-Deck aus erreicht. Da die Rampe beim Beladen vom Heck aus befahren wird, müssen alle Pkw beim Ladungswechsel über den Bug im Schiff wenden.

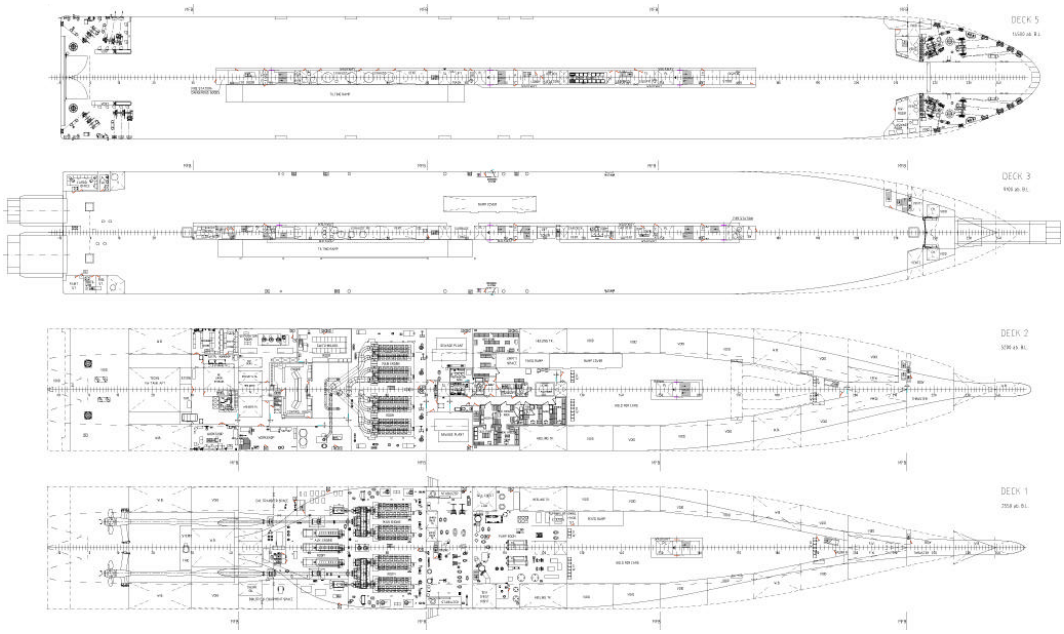

Abb. 2-6: Generalplan *Superfast 8* der Decks 1,2,3 und 5 (Quelle: HDW).

2.2 Rahmenbedingungen an Bord

Generell sind bei der Beladung von RoRo-Schiffen mehrere Rahmenbedingungen einzuhalten. Neben der Spurbreite und der Höhe der Decks ist bei der Beladung des Schiffs auf die Stabilität des Schiffes sowohl im Endzustand, als auch während der Verladung zu achten. So darf ein bestimmter Krängungswinkel aufgrund der Rampentorsion nicht überschritten werden. Zum Teil kann dies zwar durch eine Krän-

gungsanlage kompensiert werden, die Lademeister wollen sich allerdings nicht auf technische Hilfen verlassen und steuern die Beladung so, dass der Krängungswinkel kleiner als zwei Grad gehalten wird.

Um einen möglichst hohe Auslastung der Fähre zu erreichen, ist es wichtig eine gute Verschachtelung der Fahrzeuge zu erreichen. Dazu gehört auch ein möglichst kleiner Parkabstand zwischen den Fahrzeugen. Dies erfordert allerdings die tatkräftige Unterstützung der Einweiser.

Die Decks an Bord von RoRo-/ RoPax-Schiffen unterscheiden sich in der lichten Höhe und in der Spurbreite. Um dies variabel zu gestalten, gibt es Hängedecks, die nach je nach Bedarf herabgelassen werden können. Diese fungieren gleichzeitig als Rampen. Im Ladungsvorgang ist das Verstellen der Decks jedoch recht zeitaufwändig.

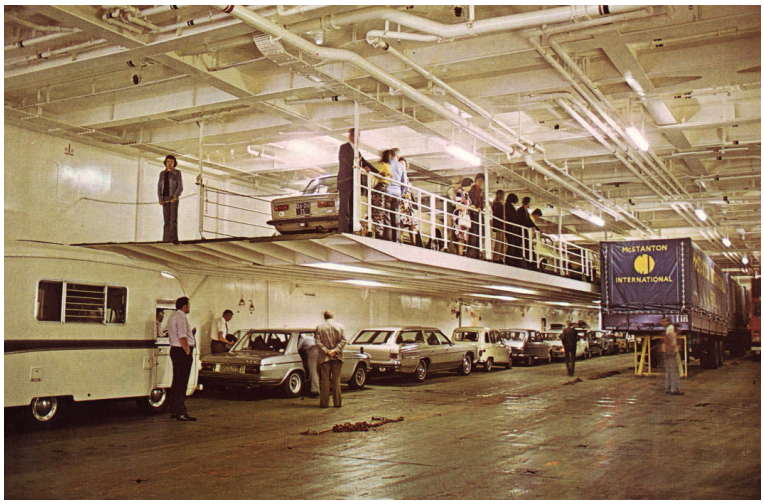

Abb. 2-7: Foto eines Hängedecks (Quelle Mc Gregor).

Beim Laden kommen zwei verschiedene Rampentypen zum Einsatz. Zum einen horizontale Rampen auf denen die Fahrzeuge ins Schiff gelangen und zum anderen solche auf dnen die Fahrzeuge das Deck wechseln können. Diese sind entweder verfahrbar, um auch darunter Fahrzeuge stauen zu können, oder sie sind fest eingebaut und werden mit einem Lukendeckel verschlossen.

2.3 Fahrzeugtypen

Die Fahrzeuge auf einer Fähre lassen sich in zwei Hauptgruppen unterteilen: Begleiteten Verkehr, auch Selbstfahrer genannt und unbegleiteten Verkehr. Mit begleitetem Verkehr sind alle Fahrzeuge gemeint, die von einem Fahrer gesteuert werden, der selber auf der Fähre mitreist. Bei unbegleitetem Verkehr, beispielsweise der Auflieger eines Sattelschleppers, wird der Auflieger von einer Zugmaschiene (Tugmaster) an Bord des Schiffes gefahren, dort abgestellt und verzurrt. Anschließend verlässt die Zugmaschine wieder das Schiff. Die beiden Hauptgruppen unterscheiden sich zum einen darin, dass beim nicht begleiteten Verkehr während eines Be- oder Entladevorgangs bidirektionaler Verkehr erfolgt, also Fahrzeugbegegnungen im Schiff stattfinden können. Im Gegensatz dazu fahren die Selbstfahrer hintereinander und immer in einer Richtung ins Schiff. Zum anderen entstehen durch das Rangieren und Laschen der Auflieger Rüstzeiten die den Ladeverkehr aufhalten.

Zu den unbegleiteten Fahrzeugen gehören fast ausschließlich die Auflieger von Sattelschleppern, die meist Container transportieren oder einen geschlossenen Aufbau haben. Vereinzelt sind auch normale Pkw-Anhänger zu finden.

Die Gruppe der Selbstfahrer kann wiederum in mehrere Untergruppen aufgeteilt werden. Maßgebliches Kriterium ist hier die Höhe des Fahrzeugs, da diese auf Grund der Deckshöhe eingeschränkt ist. Es ergeben sich folgende vier Gruppen:

- Pkw mit eine Höhe kleiner als 2 m
- Wohnmobile / Kleinbusse und Transporter
- Lkw / Sattelschlepper und Reisebusse
- Kleinfahrzeuge wie Motorräder, Fahrräder etc.

Pkw mit Anhängern wurden je nach den Bedingungen auf dem Schiff (wie Rampen und Spurbreiten) zu der ersten oder zweiten Gruppe gezählt.

2.4 Fahrverhalten

2.4.1 Allgemeines

Das Fahrverhalten unterscheidet sich nicht nur zwischen den einzelnen Fahrzeugtypen. So beschleunigen Lkw nicht unbedingt langsamer als Pkw, während des recht langsamen Be- und Entladens. Es zeigten sich nicht allein Unterschiede bei den Fahrzeugtypen, sondern es konnten auch Unterschiede auf Grund der Streckenführung festgestellt werden. Die Ausführung der Rampen beispielsweise ist maßgeblich für die dortige maximale Geschwindigkeit der Fahrzeuge und ist zum Teil ein stark einschränkendes Nadelöhr während der Beladung. Bei den Doppelendfähren der Scandlines-Reederei ist die Beladerampe landseitig installiert und schließt eben mit der Fahrbahn im Schiff ab. Die Fahrzeuge werden bei dieser Ausführung nicht in ihrer Fahrgeschwindigkeit eingeschränkt, sondern beschleunigen schon weit vor dem Verlassen der Fähre. Zusätzlich passt die landseitig installierte Rampe immer ihre Höhe an das Schiff an, so dass keine starken Höhenunterschiede auftreten. Bei der *Germanica* verlassen die Fahrzeuge der oberen beiden Decks nach einer 90° Wende das Schiff. Dies führt bei machen Fahrern zu einem unbeabsichtigtem Spurwechsel wodurch anschließend ein kurzer Stau beim Entladen entstehen kann. Die Rampe, über die das Schiff verlassen wird, hat mehrere starke Knicke, die bei der zum Teil zu vorsichtigen Fahrweise der Fahrer den Verkehr stark verlangsamen. Lange Wohnmobilgespanne kommen nur sehr zögerlich und nur mit Aufsetzten des Anhängers über die Rampe.

2.4.2 Auswertung

Ausgewertet wurden die maximalen Geschwindigkeiten der Fahrzeuge, sowie das Abstandverhalten der Fahrzeuge in Abhängigkeit von ihrer Geschwindigkeit.

Die Vorgehensweise bei den Messungen ist nachfolgend exemplarisch an Hand der Doppelendfähre der Scandlines-Reederei dargestellt. Die Ladevorgänge wurden mit mehreren Videokameras gleichzeitig gefilmt. Eine war im Rampenbereich der Fähre positioniert, eine zweite im Treppenaufgang des Centercasings und eine dritte am Ende des Centercasings (siehe Abb. 2-8). Die Kameras filmten meist quer zur Fahrbahn, um eine Geschwindigkeitsmessung zu ermöglichen.

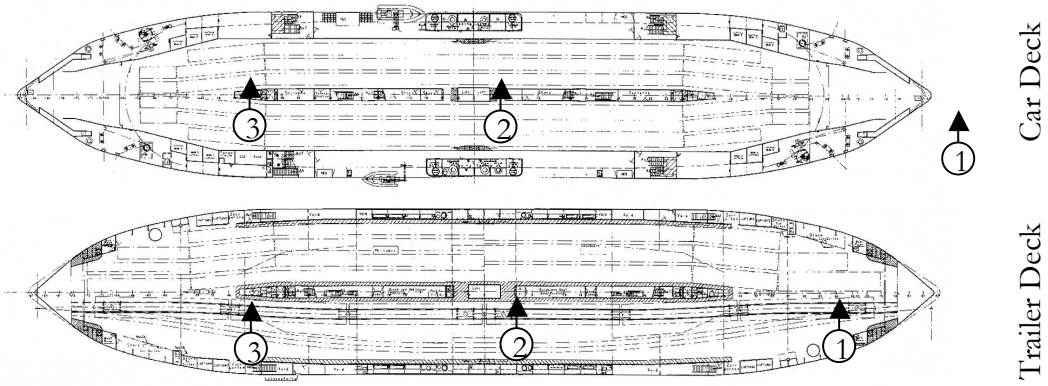

Abb. 2-8: Die beobachteten Decks und die Kamerapositionen.

Für die Messung der Geschwindigkeit wurden in den Film Markierungen mit definierten Abständen projiziert (siehe Abb. **2-9**).

Anhand der Zeitpunkte zu denen die Markierungen von den Autos überfahren wurden, konnte die Geschwindigkeit berechnet werden.

$$v = \frac{d}{\Delta t} \qquad (1)$$

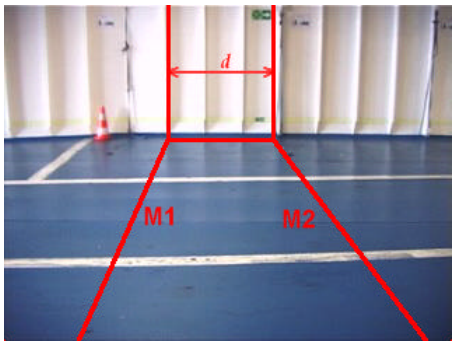

Abb. 2-9: Die projizierten Linien zur Geschwindigkeitsbestimmung.

Dies wurde für die Fahrzeugfront und das Fahrzeugheck durchgeführt. Die Fahrzeuggeschwindigkeit wurde anschließend aus beiden Werten gemittelt.

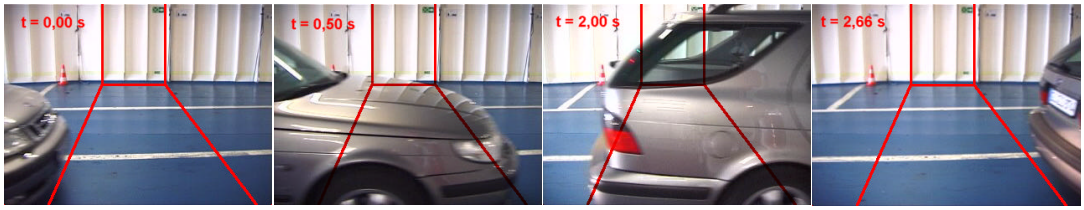

Abb. 2-10: Bestimmung der Fahrgeschwindigkeit anhand der virtuellen Linien.

In dem gezeigten Beispiel (Abb. 2-10) beträgt der Abstand zwischen den Linien d = 2,4 m. Bei einem Zeitintervall von ? t_1 = 0,5 s und ? t_1 = 0,66 s beträgt die gemittelte Geschwindigkeit 4,32 s.

2.4.3 Fahrzeuglängen

Während der Überfahrt wurden auf der *Schleswig-Holstein* sowohl die Fahrzeuglängen, als auch die Abstände zwischen den Fahrzeugen gemessen. In der statistischen Verteilung der Pkw-Längen (siehe Abb. 2-11) sind zwei Maxima zu erkennen. Das kleinere Maximum umfasst Fahrzeuge mit einer Länge von ca. μ_1 = 4 m, wie sie zum Beispiel in der Golf Klasse zu finden sind. Die größeren Pkw haben eine durchschnittliche Länge von μ_2 = 4,55 m. Das Verhältnis von kürzeren zu längeren Fahrzeugen beträgt etwa 1:3. Die Standardabweichungen betragen s_1 = 0,1 und s_2 = 0,16.

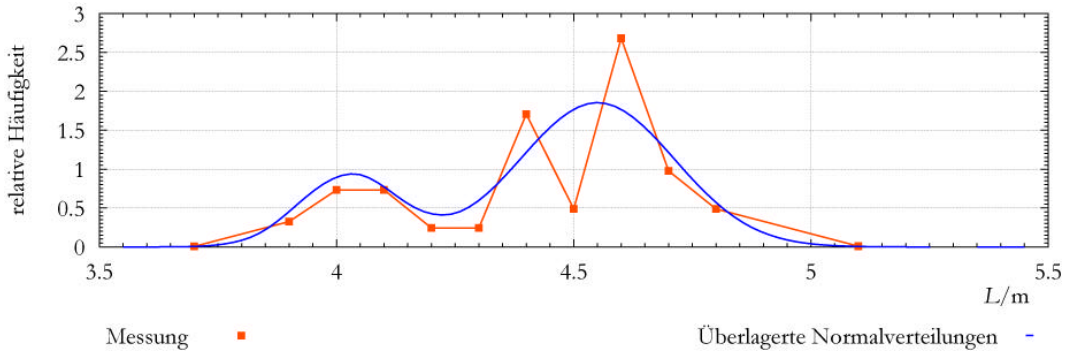

Abb. 2-11: Die relative Häufigkeitsdichte der gemessenen Pkw-Längen.

Wohnmobile/ Kleinbusse und Transporter: Während der Messungen wurden nur 9 Fahrzeuge dieser Kategorie angetroffen. Diese hatten eine Länge von 5,8 m – 6,7 m.

Lkw/ Sattelschlepper und Reisebusse: Um eine sichere Aussage über die statistische Verteilung der Fahrzeuglängen zu treffen waren zu wenige Fahrzeuge dieser Kategorie an Bord.

2.4.4 Parkabstände

Die gemessenen Abstände zwischen den parkenden Fahrzeugen weisen eine Log-Normalverteilung auf (siehe **Abb. 2-12**). Eine Begründung dafür kann darin liegen, dass viele Sinneseindrücke logarithmisch wahrgenommen werden [9]. Beispielsweise wird ein Geräuschpegel logarithmisch zum Schalldruck wahrgenommen. Ähnliches kann auch für das Sicherheitsempfinden beim Einparken gelten.

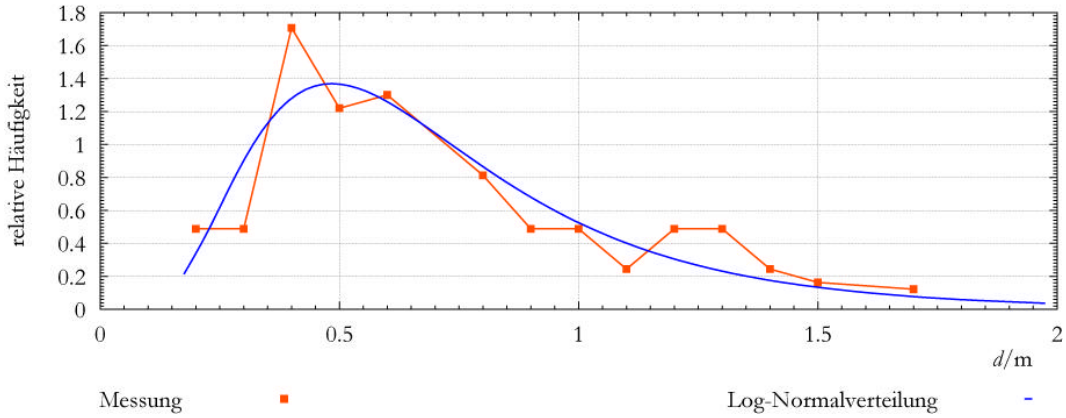

Abb. 2-12: Die relativen Häufigkeitsdichten der gemessenen Parklückenlängen.

Der Mittelwert der Log-Normalverteilung beträgt e^{μ} = 0,67 m bei einer Standardabweichung von s = 0,536.

Das Parken fand hier ohne Hilfe von Einweisern statt. Bei einer Aufsicht durch Einweiser ist der Abstand zwischen den Fahrzeugen deutlich geringer.

2.4.5 Begleiteter Verkehr

a) Pkw mit eine Höhe kleiner als 2 m

In der Darstellung (Abb. 2-13) sind die Fahrzeuggeschwindigkeiten den Abständen vor dem Fahrzeug gegenübergestellt. Eine Geschwindigkeitserhöhung geht somit mit einer Abstandsvergrößerung einher. Setzt man für die Streuung der Abstände bei gleicher Geschwindigkeit wie für die Parkabstände eine Log-Normalverteilung an, so stellt die Gerade

$$v = 0{,}357\,\tfrac{1}{s} \cdot (d - 3{,}3m) \qquad (2)$$

in einer Näherung den Mittelwert dar.

Bei einer Geschwindigkeit von 0 m/s entspricht das aus Gl. (2) ermittelte d nicht dem Mittelwert der Parkabstände. Dies liegt daran, dass ein Fahrer meist erst losfährt, wenn sein Vorgänger losgefahren ist. Aufgrund der Reaktionsdauer hat sich dann bereits ein größerer Abstand als während des Parkens gebildet.

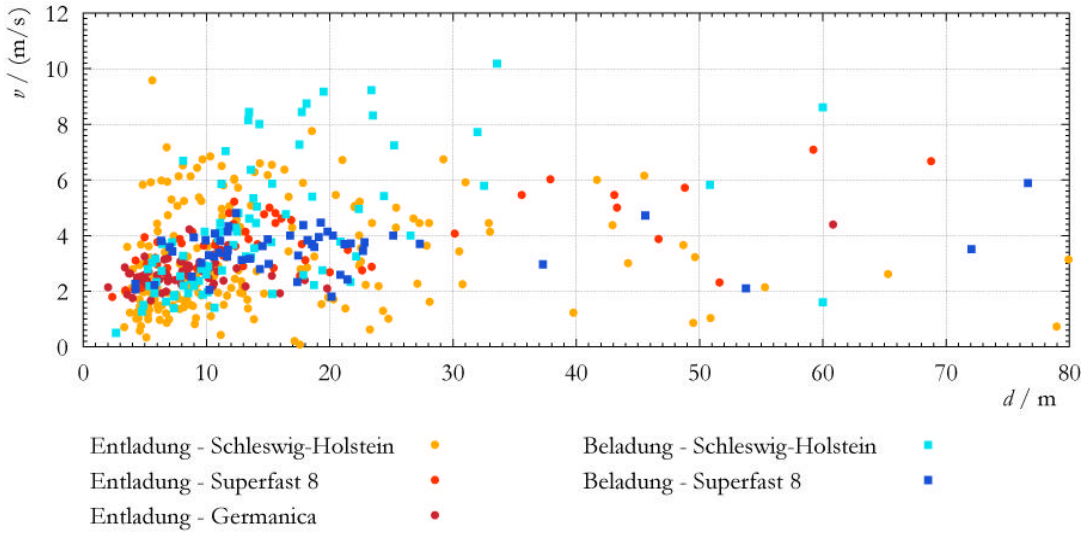

Abb. 2-13: Die ermittelten Pkw-Fahrtgeschwindigkeiten in Abhängigkeit vom Fahrzeugabstandes während der Be- und Entladung.

Betrachtet man nicht den Abstand selbst, sondern das Zeitintervall ($t_{safe} = d/v$), das zwischen zwei Fahrzeugen liegt und trägt ihn über der Geschwindigkeit auf, erhält man eine detailliertere Betrachtung der Fahrtzustände (siehe Abb. 2-14). Ebenfalls unter der Annahme, dass die Zeiten Log-Normalverteilt sind, erhält man eine Mittelwertkurve für t_{safe}, die ab einer Geschwindigkeit von 3 m/s nahezu konstant bei 3 s liegt. Der starke Abfall von t_{safe}, bei Geschwindigkeiten kleiner 3 m/s weist ebenfalls auf das verzögerte Anfahren der Pkw hin.

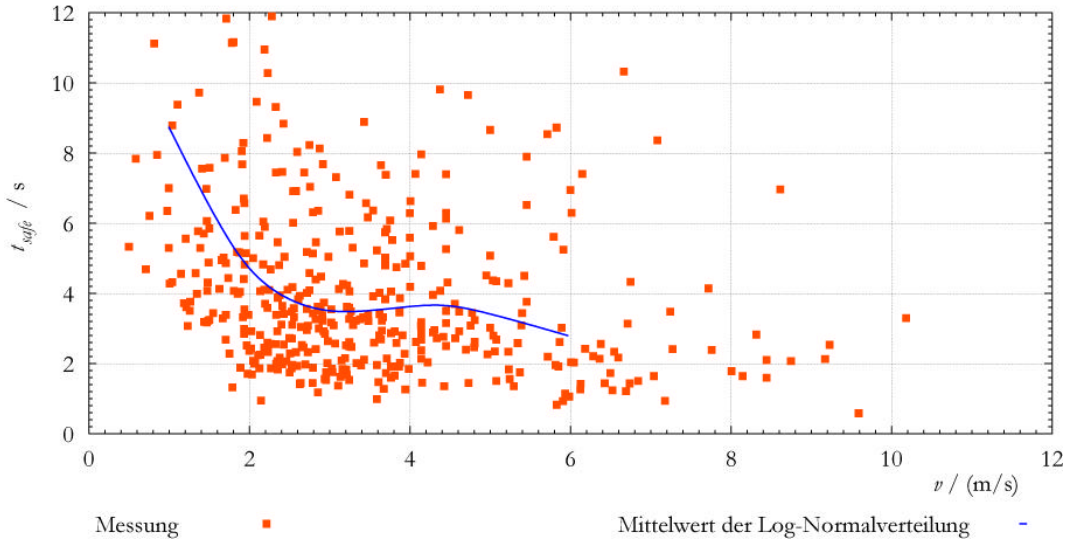

Abb. 2-14: Zeitliches Abstandverhalten.

Anhand der Pkw-Beladung der *Superfast 8* lässt sich zeigen, dass die maximale Geschwindigkeit der Fahrzeuge stark von den äußeren Randbedingungen abhängt (siehe **Abb. 2-15** und **Abb. 2-16**). Da die Fähre über den Bug beladen wird, und die Rampe zu den Autodecks vom Heck aus zu befahren ist, müssen die Fahrzeuge im Heck wenden.

Am Ende der langen schmalen Bugrampe erreichen die Fahrzeuge eine Maximalgeschwindigkeit von 4,5 m/s. Nach der Rampe beschleunigen die Fahrer und knapp 25 m dahinter wird die größte, im Schiff gemessene Geschwindigkeit von ca. 6 m/s erreicht. Im Wendebereich im Heck sinkt die maximale Geschwindigkeit auf 2,5 m/s, diese bleibt bis zur Rampe im Schiff etwa konstant. Da beim Beladen ein Stau im Schiff entsteht, werden an der Rampe und im Wendebereich auch sehr kleine Geschwindigkeiten um 0,5 m/s gemessen.

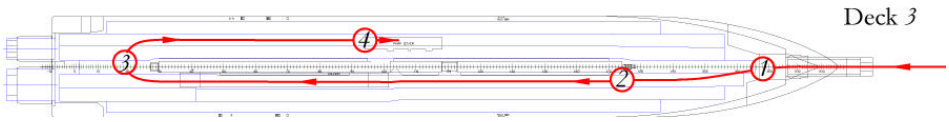

Abb. 2-15: Messpunkte auf Deck 3.

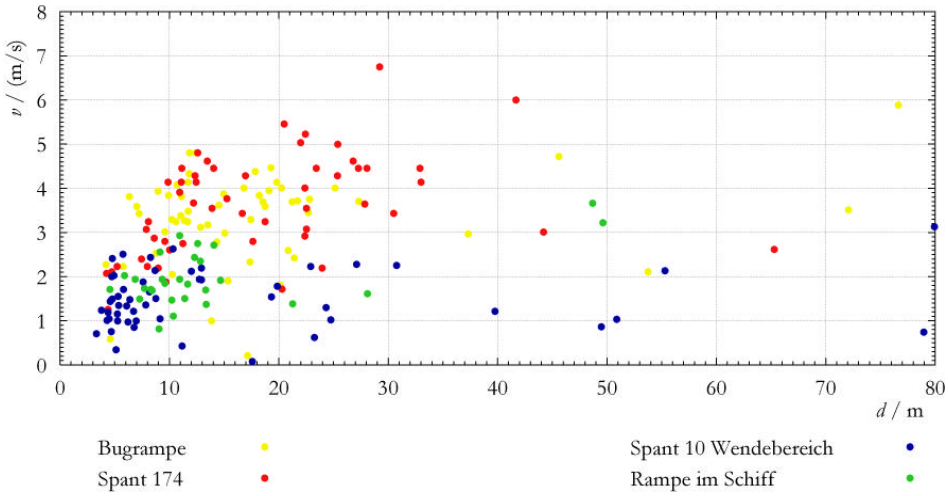

Abb. 2-16: Maximale Geschwindigkeit an verschiedenen Messpunkten.

b) Lkw / Sattelschlepper und Reisebusse

Die auf der *Superfast 8* und der *Germanica* ermittelte Maximalgeschwindigkeit der Lkw ist mit ca. 2,0 m/s deutlich geringer als bei den Pkw. Nur einzelne Ausreißer fahren schneller. Die wenigen Lkw, die auf der *Schleswig-Holstein* gemessen wurden, fuhren alle etwa doppelt so schnell. Da die Geschwindigkeiten alle an der Rampe gemessen wurden, kann die hohe Geschwindigkeit in der Geometrie der Öffnung am Bug begründet sein. Zwar reichte die Höhe bei allen Fähren aus, aber die Pforte im Bug der *Schleswig-Holstein* machte den größten Eindruck und die Rampe war absolut stufenlos.

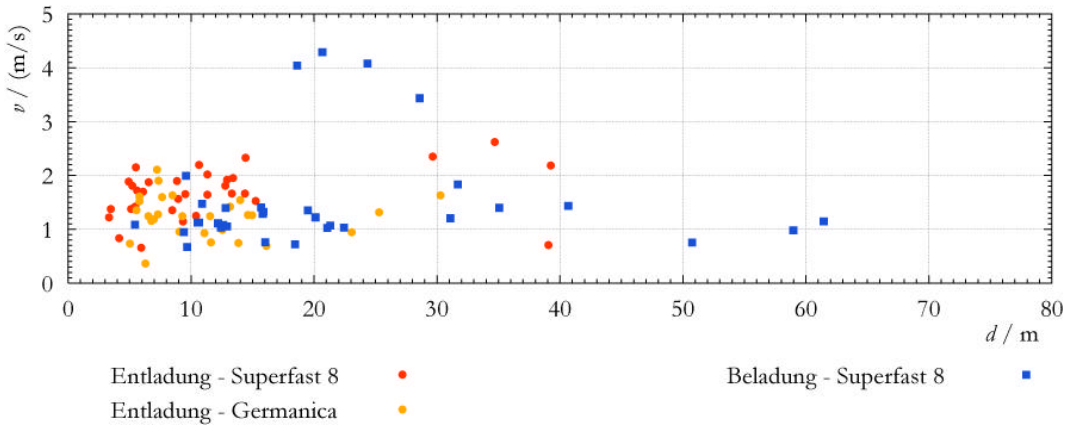

Abb. 2-17: Die ermittelten Lkw-Fahrtgeschwindigkeiten in Abhängigkeit des Fahrzeugabstandes während der Be- und Entladung.

2.4.6 Nicht begleiteter Verkehr

Das Verladen der Trailer wird von erfahrenem Personal durchgeführt, die schneller fahren als die Fahrer von Lkw. Im beladenen Zustand fahren die Zugmaschinen mit Anhänger 3 m/s, unbeladen erreichen durchschnittlich 4,15 m/s.

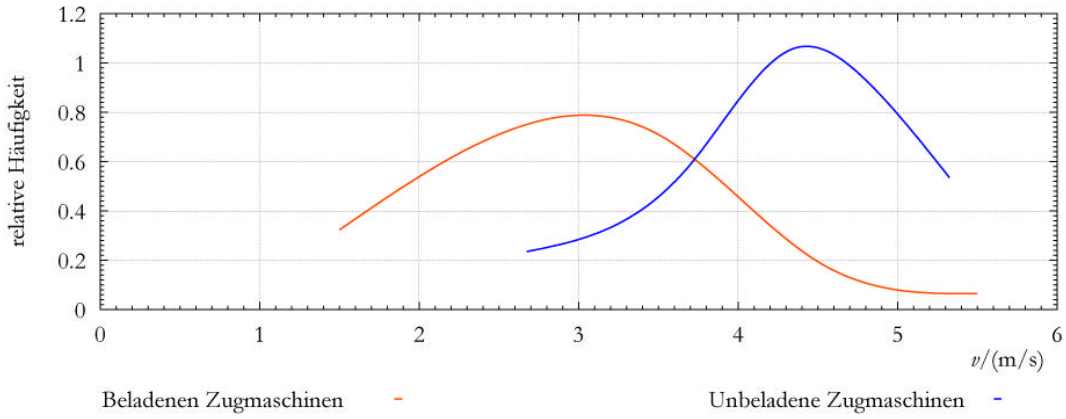

Abb. 2-18: Geschwindigkeitsverteilung der Zugmaschinen.

Beim Verladen kommen neben der Fahrzeit noch eine Rangierdauer und eine Anpick- bzw. Abstelldauer hinzu. Für das Anpicken oder Abstellen werden etwa 30 s benötigt. Das Rangieren der Trailer beim Beladen dauert etwa 100 s, beim Entladen entfällt das Rangieren weitgehend.

2.5 Messfehler

Mögliche Fehlerquellen bei der Untersuchung beruhen auf der Auswertung anhand von Videofilmen. Die im Film eingeblendeten Linien sind an Objekten wie Spanten und Markierung ausgerichtet. Für eine gute Analyse sind die genauen Abmessungen nötig. Eine Verzerrung durch die Kameralinse erschwert die Konstruktion der Linien. Aufgrund der engen Situation an Bord passieren die Fahrzeuge die Kamera so dicht, dass eine Zuordnung zwischen Position des Fahrzeugs und der Linie schwer fällt. Ein weiterer Fehler liegt in der zeitlichen Auflösung der Filme, was bei hohen Geschwindigkeiten zu Fehlern führen kann.

3 Modellentwicklung

3.1 Anforderungen

Das Modell wurde mit dem Ziel entwickelt unterschiedliche Geometrien möglichst einfach in eine Simulation einzubinden und dabei den Ladeverkehr, wie er auf einer Fähre üblich ist, abzubilden. Das bedeutet, dass mit Rücksicht auf das jeweilige Verhalten verschiedener Verkehrstypen simuliert wird.

Des weiteren wird von dem Modell gefordert, dass es ein selbständiges Beladen durchführt, das unfallfrei abläuft und keine nicht auflösbaren Blockierungen der Fahrzeuge entstehen lässt.

3.2 Bestehende Modelle

Es gibt zahlreiche Modelle zur Simulation des Fahrverhaltens von Fahrzeugen. Diesen fehlt jedoch die Anbindung an variable Geometrien, auch eine selbständige Koordinierung der Routenwahl fehlt. Die Modelle für die Fahrzeugbewegung seien hier nur kurz erwähnt. Die Quelle [10] liefert eine detailliertere weiterführende Beschreibung aller gängigen Modelle.

Grob können sie in folgende Kategorien aufgeteilt werden. In den makroskopischen Modellen wird ausgehend von fluiddynamischen Ansätzen der Verkehrsfluss untersucht und simuliert. Für eine detaillierte Analyse, wie sie hier gefordert ist, reicht dies jedoch nicht aus. Mikroskopische Modelle bilden jedes Fahrzeug einzeln ab. Durch Regeln wird das Fahrverhalten eines Fahrzeugs definiert, was bei der Simulation automatisch zu Interaktionen mit den anderen Fahrzeug führt. Die Stecken, auf denen sich die Fahrzeuge bewegen, können entweder kontinuierlich sein, oder bei einem Zellularen Automaten in einzelne Abschnitte unterteilt sein, bei denen die Fahrzeuge von Zelle zu Zelle springen. Zellular Automaten haben einen Geschwindigkeitsvorteil in der Simulationzeit und werden häufig bei Simulationen von Verkehren mit hoher Geschwindigkeit, wie auf Autobahnen, eingesetzt. Üblicherweise sind die Zellen 7,5 m lang, was dem Platzbedarf eines Autos entspricht. Um die kleinen Geschwindigkeiten des Ladeprozesses detailliert abzubilden, müssen in einem Zellularautomaten die Zellen deutlich kleiner sein als 7,5 m, wodurch sein Geschwindigkeitsvorteil verloren geht.

Dem gegenüber stehen kontinuierliche Modelle, bei denen sich ein Fahrzeug an einer beliebigen Stelle auf einer Strecke befindet. Dabei kann auch die Geschwindigkeit stetig sein, was bei den Zellularautomaten nicht gegeben ist, da die Fahrzeuge immer ganze Zellen weit springen.

Zu Beginn der Implementierung stand noch nicht fest welche Parameter nötig sind, um den Be- und Entladeverkehr exakt abzubilden. Um keine Einschränkungen aufgrund der Zellengröße zu erhalten und der Vorteil in der Geschwindigkeit der Simulation gering erschien, wurde hier ein kontinuierliches Modell verwendet. Gewählt wurde ein von Krauss, Wagner und Gawron [10]vorgeschlagenes Modell, auf das im Abschnitt 3.3.7 näher eingegangen wird.

3.3 Implementierung

3.3.1 Zusammenfassung

Zusammenfassend lässt sich das Modell wie folgt beschreiben: Für die Fahrwege der Fahrzeuge wird vor der Simulation ein Wegenetz festgelegt, dem die Fahrzeuge folgen. Das Wegenetz wird anhand einer CAD-Zeichnung erstellt. Die Fahrzeuge suchen selbständig eine Route in dem Wegenetz und fahren diese entsprechend ihres Fahrverhaltens und eines implementierten Bewegungsalgorithmus ab. In den Bewegungsalgorithmus fließt eine Trödelwahrscheinlichkeit ein, die das stochastische Verhalten der Fahrzeuge erfasst. Daher wird die Simulation für ein untersuchtes Wegenetz mehrmals durchlaufen, um eine Aussage über die Gesamtbelade- oder Entladedauer zu gewinnen.

3.3.2 Routen

Für die Darstellung der Fahrwege wird das Wegenetz in einzelne Streckenabschnitte geteilt. Eine Route ist ein solcher Streckenabschnitt, für den bestimmte Eigenschaften gelten, die das Lade- und Fahrverhalten beeinflussen. Jede Route kann sich in mehrere Nachfolgerouten aufteilen, wie es beispielsweise beim Beladen einer Fähre geschieht. Um mehrere Routen zusammenzuführen, haben mehrere Routen eine andere Route gemeinsam als Nachfolgeroute. Im Falle des Zusammenführens darf jede der zusammengeführten Route jedoch nur diese eine Nachfolgeroute besitzen.

Die Routen werden anhand eines Polygonzugs definiert. Jede Route besitzt einen Ursprung in einem globalen, kartesischen Koordinatensystem. Von dort aus ist jede Gerade des Polygonzugs relativ zu der vorhergehenden Gerade in einem Polarkoordinatensystem gespeichert (siehe Abb. 3-1). Die Position eines Fahrzeugs auf der Route wird in einer Gleitkommazahl gespeichert. Der ganzzahlige Anteil i der Position gibt an, dass sich das Fahrzeug auf der i+1 ten Strecke im Polygonzug befindet. Der Nachkommaanteil gibt den Anteil des auf diesem Abschnitts zurückgelegten Wegs an. Ist die Position negativ wartet das Fahrzeug am Routenanfang auf das Losfahren.

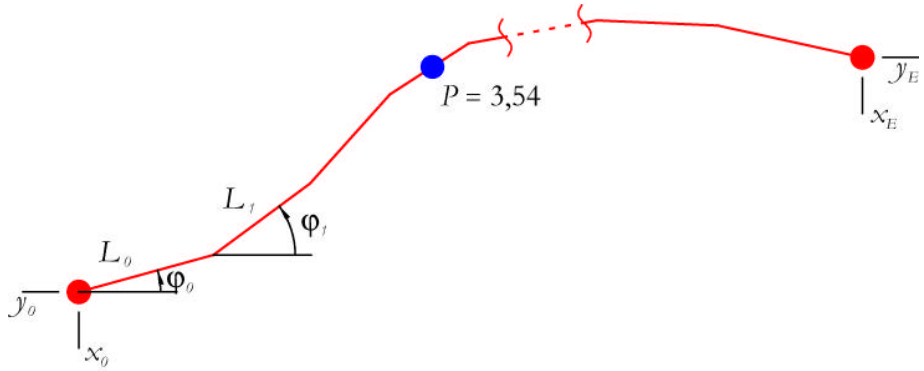

Abb. 3-1: Definition der Routen (rote Linie) und der Position eines Fahrzeugs (blauer Punkt).

Mehrere Eigenschaften der Routen regeln das Fahrverhalten und das Vorgehen beim Verladen der Fahrzeuge:

- Parkverbot: Ist diese Eigenschaft gesetzt, dürfen auf dieser Route keine Fahrzeuge abgestellt werden. Am Ende der Simulation sind diese Routen leer.
- Überstand erlauben: Fahrzeuge dürfen auf einer Route parken, auch wenn sie länger sind, als der noch freie Platz der Route.
- V_{max}: Jeder Route wird eine Höchstgeschwindigkeit zugeordnet. Beim Übergang zwischen den Routen wird die Maximalgeschwindigkeit entsprechend dem Bremsvermögen eines Fahrzeugs geglättet.
- Routeneintritt eines Fahrzeugs mitloggen: Wechselt ein Fahrzeug die Route wird der Zeitpunkt, die Fahrzeuggeschwindigkeit des Fahrzeugs und der Abstand zum Vordermann notiert.
- Priorität der Nachfolger: Um festzulegen, in welcher Reihenfolge die Anschlussrouten belegt werden, kann eine Priorität angegeben werden. Haben zwei Anschlussrouten die gleiche Priorität, werden Fahrzeuge immer zu der momentan leereren Route (auf der noch mehr freier Platz für Fahrzeuge zur Verfügung stehtmehr Platz bis zum nächsten Fahrzeug) weitergeleitet.
- Priorität beim Zusammenführen von Routen: Ähnliches gilt beim Zusammenführen der Routen. Vorfahrt erhalten die Fahrzeuge, deren Route die höhere Priorität hat. Bei gleicher Priorität erhält das dem Knotenpunkt nähere Fahrzeug den Vorzug. Dabei muss berücksichtigt werden, dass ein fahrendes Fahrzeug nur ein bestimmtes Abbremsvermögen hat und gegebenenfalls entgegen der Priorität Vorfahrt bekommt.
- Trödelwahrscheinlichkeit: Mit dem Trödeln wird berücksichtigt, dass ein Fahrzeug nicht immer die maximal mögliche Geschwindigkeit, die der Abstand vor ihm zulässt, auch erreicht. Zufällig, mit der angegebenen Trödelwahrscheinlichkeit, reduziert das Fahrzeug seine Geschwindigkeit um einen bestimmten Betrag.
- Trödeln beim Anfahren: Wie bei den Messungen deutlich geworden ist, zögern Fahrzeuge beim Losfahren stärker als während des Fahrens. Dies wird durch eine zweite Wahrscheinlichkeit erfasst. Näheres zu den Trödelwahrscheinlichkeiten ist auf Seite 23 beschrieben.
- Sonderrouten: Als Fahrzeugquellen und Senken sind drei weitere Routeneigenschaften definiert. Ist die Senkeneigenschaft gesetzt, werden alle Fahrzeuge, die auf diese Route wechseln, gelöscht. Des weiteren stehen zwei Quelleigenschaften zur Verfügung. Routen mit dieser Eigenschaft erzeugen mittels einer einstellbaren statistischen Verteilung Fahrzeuge unterschiedlicher Fahrzeugtypen.Um Entladefälle zu simulieren, müssen die Routen zunächst mit Fahrzeugen initialisiert werden. Der Unterschied zwischen den beiden Eigenschaften besteht darin, dass die eine nur während der Initialisie-

rung eine bestimmte Anzahl von Fahrzeugen erzeugt und die andere während der Simulation so viele Fahrzeuge erzeugt, wie auf den Routen untergebracht werden können.

- Abbruchbedingung auswerten: Für den Abbruch der Simulation werden mehrere Kriterien ausgewertet. Das Auswerten kann mit dieser Eigenschaft auf bestimmte Routen beschränkt werden.

3.3.3 Fahrzeuge

Um eine exakte Abbildung der Größen und Verhältnisse zu erreichen, werden die Fahrzeuge nicht durch ein Stück belegte Strecke einer Route abstrahiert, wie es in vielen anderen Simulationen geschieht. Da besonders lange Fahrzeuge in Kurven weit ausscheren, kommt es bei einer einfachen Abbildung zu Kollision zwischen den Fahrzeugen. Um dies zu verhindern, werden alle Fahrzeuge achsweise aufgelöst. Die Mitte der Vorderachse ist ein sich bewegender Punkt, an dem alle weiteren Achsen frei drehbar angehängt sind. Jeder Anhang besteht aus einem Achsmittelpunkt mit einer bestimmten Deichsellänge davor und einer definierten Kupplungslänge dahinter. Am Ende der Kupplung können weitere Anhänge folgen.

Das Fahrverhalten der Fahrzeuge richtet sich nach folgenden Eigenschaften:

- Maximale positive und negative Beschleunigung: Über die Werte der maximalen Beschleunigungen wird das Abstandverhalten während des Fahrens beeinflusst.
- Maximale Geschwindigkeit: Neben einer maximal zulässigen Geschwindigkeit auf einer Strecke hat jedes Fahrzeug eine maximale Geschwindigkeit, die nicht überschritten wird.
- Parkabstand: Für jedes Fahrzeug wird ein Abstand vorgegeben, den das Fahrzeug beim Parken einhält.

Die Fahrzeuge werden in der Simulation auf die Achsmitten mit bestimmten Abständen dazwischen reduziert.

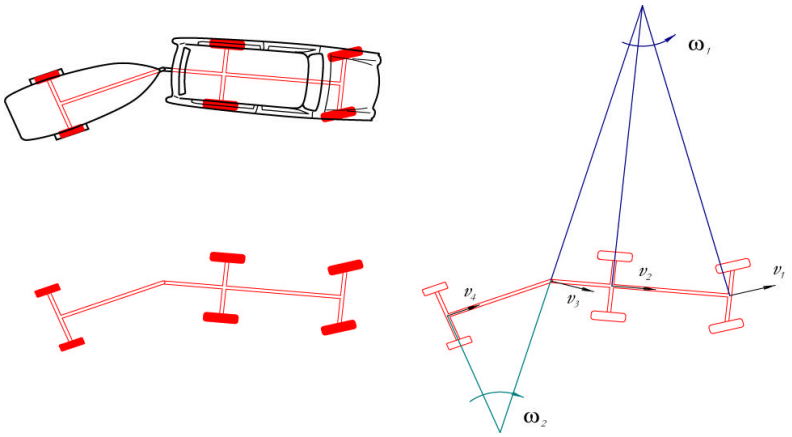

Abb. 3-2: Darstellung der Fahrzeuge.

Für die Vorderachse wird der Bewegungsvektor (v_0, a_0) angegeben. Die Bewegung der folgenden Anhänge wird gemäß kinematischer Gleichungen berechnet. Um eine

schnelle Berechnung der Bewegung zuzulassen, gelten die Bewegungsgleichungen näherungsweise für kleine Drehwinkel und kleine Zeitintervalle.

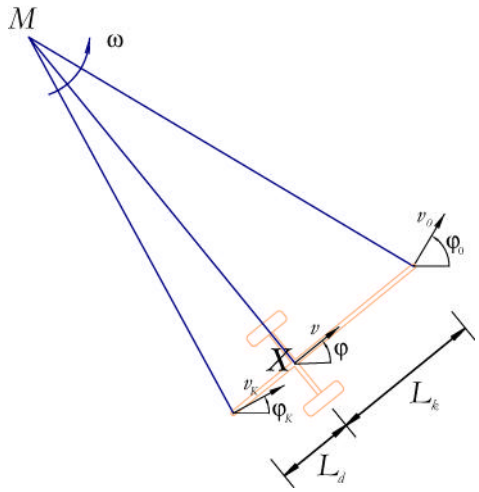

Abb. 3-3: Drehung eines Anhangs um den Momentanpol *M*.

$$v^{+} = \cos(j_0 - j^{-}) \cdot v_0 \tag{3}$$

$$X^{+} = X^{-} + \begin{pmatrix} \cos(a^{-}) v^{+} \Delta t \\ \sin(a^{-}) v^{+} \Delta t \end{pmatrix} \tag{4}$$

$$a^{+} = a^{-} + \tfrac{v_0}{L_d} \sin(a_0 - a^{-}) \Delta t \tag{5}$$

Die mit $^{+}$/ $^{-}$ gekennzeichneten Werte stehen für den Wert nach/vor dem zu berechnenden Zeitschritt.

3.3.4 Simulationsablauf

Da die Simulation auf statistischen Berechnungen beruht, muss ein Ladungswechsel mehrfach simuliert werden, so dass am Ende der Simulation ein Mittelwert für eine Beladungsdauer ausgegeben wird. Deshalb ist es erforderlich, die gesamte Simulation in mehrere Zyklen zu unterteilen.

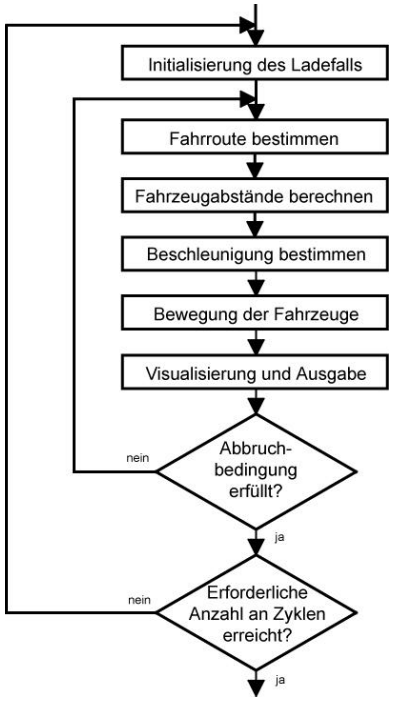

Abb. 3-4: Programmablaufdiagramm.

Die Simulation eines Zeitschrittes läuft in mehreren Schritten ab. Zunächst wird entsprechend eines Beladungsalgorithmus (siehe Abschnitt 3.3.5) jeweils für das erste Fahrzeug einer Route ein Weg für das Fahrzeug gesucht, auf dem es abgestellt werden kann. Dabei ist darauf zu achten, dass das Beladen unfallfrei abläuft und keine Blockierungen während des Beladens entstehen. Zudem soll möglichst aller zur Verfügung stehende Parkraum genutzt werden.

Anschließend können die Abstände zwischen den Fahrzeugen berechnet werden, die für eine Bestimmung der Geschwindigkeit notwendig sind (siehe dazu Abschnitt 3.3.6 und Abschnitt 3.3.7). Nun entscheiden der Abstand und die Fahrzeugeigenschaften über ein Abbremsen oder Beschleunigen des Fahrzeugs. Mit der berechneten Beschleunigung werden die Fahrzeuge einen Zeitschritt lang bewegt und auf dem Bildschirm ausgegeben. Zum Ende eines jeden Simulationszeitschritts werden die zu loggenden Ereignisse ausgegeben und die Abbruchkriterien überprüft.

3.3.5 Beladungsalgorithmus

In jedem Zeitschritt wird zunächst für das erste Fahrzeug einer Route ein Weg gesucht auf dem es Parken darf, das heißt die Parkroute muss die Eigenschaft „Parken" besitzen. Ist ein Platz für ein Fahrzeug gefunden, wird auf der gefundenen Parkroute ein Stück entsprechend der Fahrzeuglänge mitsamt des Parkabstands reserviert. Der reservierte Platz auf der bisherigen Route wurde vor der Suche nach einem Parkplatz freigegeben. Wurde kein neuer Parkplatz gefunden, wird der alte Zustand wieder hergestellt. Das Fahrzeug speichert nun den Weg, wie es den ihm zugewiesenen Parkplatz erreicht.

Die Suche geschieht rekursiv im nachfolgenden Routenbaum. Bei jedem Aufruf der Rekursion wird unterschieden, ob es sich bei den nachfolgenden Routen um eine Zusammenführung von mehreren Routen, oder um eine Aufteilung (Aufsplittung) der Route handelt. Wird die Route aufgesplittet, ist zunächst die Priorität der Nachfolgerouten wichtig. Die Nachfolgerrouten werden nacheinander so lange nach einem Parkplatz durchsucht, bis ein Platz für das Fahrzeug gefunden wird. Dabei wird mit der Route mit der höchsten Priorität begonnen. Besitzen mehrere der Nachfolgerouten die gleiche Priorität, wird die Rekursion nicht nach dem Auffinden eines Parkplatzes abgebrochen, sondern die Suche für alle Routen der gleichen Priorität fortgesetzt. Anschließend wird die Route, auf der am meisten Platz ist, ausgewählt. Dadurch ist ein gleichmäßiges Befüllen mehrer Routen möglich.

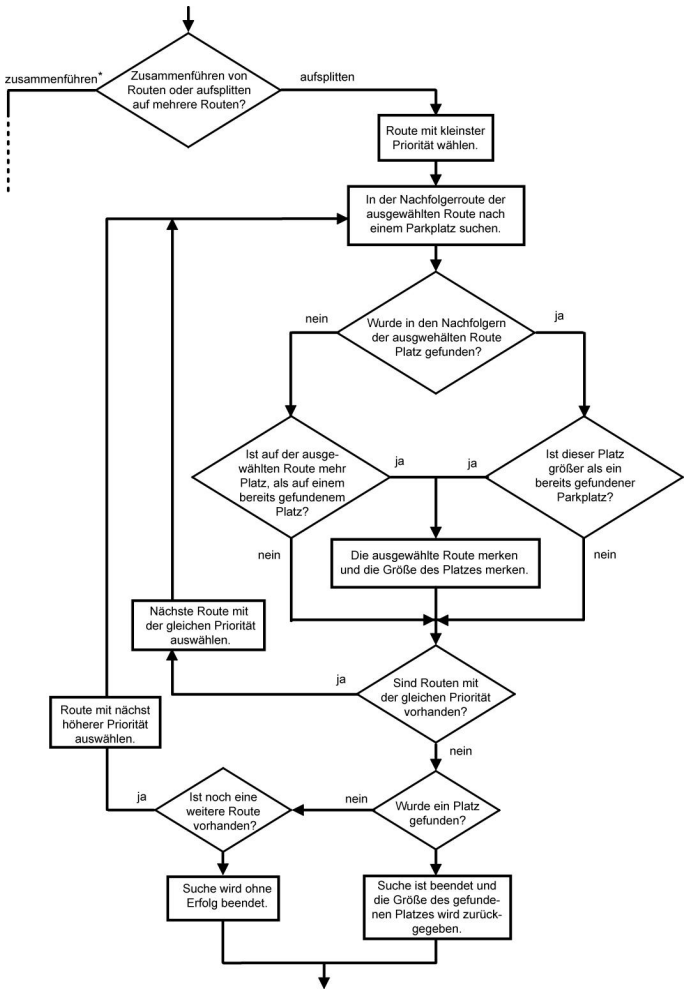

Abb. 3-5: Aufsplitten von Routen. *) Das Zusammenführen ist in Abb. 3-6 dargestellt.

Das Zusammenführen erfordert mehr Aufwand. Zunächst gilt, dass immer nur ein Fahrzeug in der Simulation auf seinem Weg zum Parkplatz die Route, auf der Zusammengeführt wird, nutzen darf. Dies ist notwendig, um eine Kollision von gleichzeitig eintreffenden Fahrzeugen zu vermeiden. Daher speichert jede Route von welchem Fahrzeug sie als nächstes genutzt wird. Ist eine Route bei der Suche eines Parkplatzes bereits in Nutzung, entscheidet zunächst die Prioritäten der Vorgängerrouten welchem Fahrzeug der Vorzug geben wird. Sind die Prioritäten gleich, wird geprüft, welches Fahrzeug der Route näher ist. Dem Fahrzeug, das bisher die Route nutzte, wird nur dann die Vorfahrt entzogen, wenn es auch bis zum Punkt der Zusammenführung einen neuen Parkplatz findet und bis dahin mit seinem Bremsvermögen stehen bleiben kann.

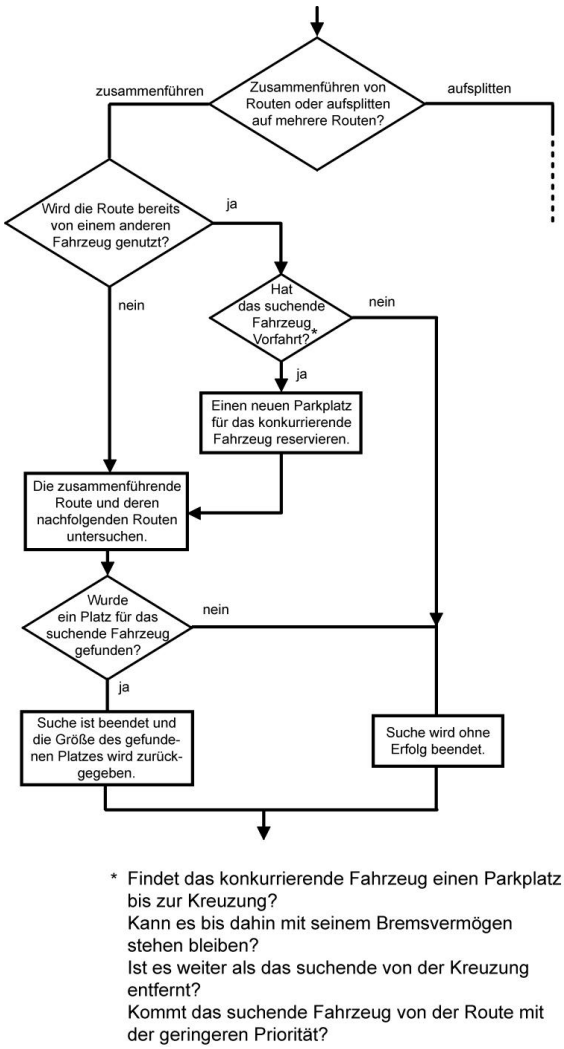

Abb. 3-6: Zusammenführen von Routen.

Das Überprüfen, ob ein Platz für das Fahrzeug vorhanden ist, läuft in mehreren Schritten ab. Ist auf der Route noch kein Parkplatz reserviert worden, wird die nächste Stufe der Rekursion aufgerufen, also die nachfolgenden Routen dieser Route untersucht. War die Suche nicht erfolgreich, wird die Route selbst überprüft. Wenn ausreichend Platz vorhanden ist, wird die Route als Parkroute für das Fahrzeug vorgemerkt. Dies geschieht auch, wenn nicht genug Platz vorhanden ist, aber die Eigenschaft »*Überstände erlauben*« gesetzt ist.

Das Reservieren des Platzes kann auf drei unterschiedliche Arten erfolgen. Passt das Fahrzeug ganz auf die Route, wird die Zahl »*Speicher_B*« um die Länge des Fahrzeugs und des Parkabstandes erhöht. Ragt das Fahrzeug über das Routenende hinaus, ist »*Speicher_A*« gleich der Länge, die durch das Fahrzeug auf der Route belegt. Steht ein Fahrzeug nur mit seinem Heck auf der Route wird dieser Teil in »*Speicher_C*« reserviert. Die freie noch beparkbare Länge ergibt sich damit aus der Differenz der Länge und der Summe der gespeicherten Fahrzeuglängen.

3.3.6 Abstandberechnung

Der Berechnung der Lücke vor einem Fahrzeug liegt die Länge der Bahnlinie d_1 zu dem vorrausfahrenden Fahrzeug zugrunde. Um das Ausscheren des Hecks in Kur-

ven zu berücksichtigen, fließt bei einem kleinen Abstand der direkte Abstand d_2 zwischen den Fahrzeugen mit ein. Die genaue Berechnung ist in Abb. 3-7 dargestellt.

Falls $d_1 < L_p+L$:

$$d = (1 - \frac{d_1}{L + L_p})d_1 + \frac{d_2}{L + L_p} d_2 \qquad (6)$$

sonst:

$$d = d_1 \qquad (7)$$

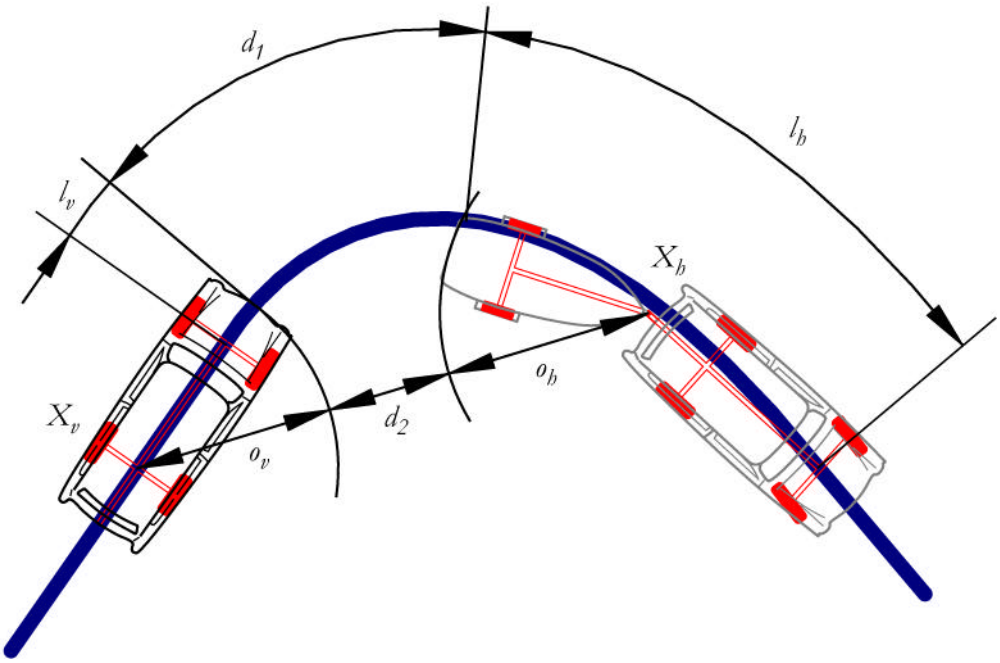

Abb. 3-7: Berechnung der Fahrzeugabstände.

3.3.7 Simulation der Fahrzeugbewegung

In der Simulation wird ein leicht verändertes, an die Umgebung angepasstes Modell von Krauss, Wagner und Gawron [11] verwendet. Dem Modell liegt die Annahme zugrunde, dass ein Fahrzeug ein bestimmtes Beschleunigungs- *(a_{max})* und Bremsvermögen *(a_{min})* hat. Für jeden Zeitschritt findet für alle Fahrzeuge gleichzeitig ein sogenanntes paralleles Update statt. Ausgehend von der Geschwindigkeit des Vordermanns und des Abstands *(d)* zu diesem wird eine eigene maximale Geschwindigkeit berechnet, mit der das Fahrzeug unfallfrei fahren kann. Der Bremsweg des Vorgängers (indiziert mit dem Index *p*) hat die Länge:

$$B_p = \frac{v_p^2}{2a_{\min,p}} \qquad (8)$$

Da das Fahrzeug beim Parken den Abstand d_0 einhalten soll, ergibt sich für das Fahrzeug eine maximale Geschwindigkeit von:

$$v_{safe} = \sqrt{2a'_{\min}(B_p + d + d_0)} \qquad (9)$$

Diese Maximalgeschwindigkeit kann durch eine zulässige Höchstgeschwindigkeit auf der Strecke und durch eine vom Fahrzeug bedingte Maximalgeschwindigkeit eingeschränkt werden.

$$v_{des} = \min(v_{safe}, v_{\max,Strecke}, v_{\max,Fahrzeug}) \qquad (10)$$

Die nun gewählte Geschwindigkeit, ist die, die das Fahrzeug nun annehmen soll, aus ihr wird eine Bescheunigung berechnet, die dann tatsächlich auf das Fahrzeug wirkt. Die so ermittelte Beschleunigung a_{soll} wird nun mit einer bestimmten Wahrscheinlichkeit P_{dawdle} herabgesetzt. Dies geschieht, um das stochastische Trödeln der Fahrer mit zu berücksichtigen. Bei abbremsenden Fahrzeugen wird zufällig ein Wert zwischen a_{soll} und a_{min}, bei beschleunigenden Fahrzeugen zwischen a_{soll} und 0 gewählt. Bei der Bestimmung von v_{safe} fließt $a'_{\min} = 0{,}5a_{\min}$ ein, um ein zufälliges, schärferes Abbremsen zu ermöglichen.

Stehende Fahrzeuge werden von einer zweiten Wahrscheinlichkeit beeinflusst. Für diese ist eine slow-to-start Regel [12] implementiert. Mit einer Wahrscheinlichkeit von P_{sts} ist die Beschleunigung der stehenden Fahrzeuge im nächsten Zeitschritt 0. Diese Regel wurde implementiert, um den sonst zu schnellen Verkehrsabfluss aus einem Stau zu reduzieren.

3.3.8 Erstellung einer Simulation

Zur Erstellung einer Simulation wird die Geometrie der Routen über eine DXF-Schnittstelle importiert. Es werden alle Linien und Polylinien aus der Datei gelesen und dargestellt. Anschließend werden Routen anhand der dargestellten Linien und Polylinien definiert. Mit der Maus können einzelne Linien selektiert werden, die an die Route angehängt werden. Die Orientierung des zuletzt gewählten Linienzugs kann durch einen Klick mit der rechten Maustaste umgekehrt werden.

Nun müssen die Eigenschaften jeder Route und der Routenbaum definiert werden. Dazu können eine oder mehrere Routen in der Liste *»Routes«* markiert, und die dazu gehörigen Eigenschaften gesetzt werden. Die nachfolgenden Routen können in der Liste *»FollowingRoutes«* ausgewählt werden.

Zum Start der Simulation müssen noch Angaben zur Anzahl der Durchläufe und der Art der Simulation gemacht werden. Handelt es sich um eine Entladung müssen zunächst alle Routen initialisiert werden. Alle während der Simulation erzeugten Zufallszahlen werden von einem Zufallsgenerator erzeugt, der bei einer bestimmten Initialisierung immer dieselben Zufallszahlen erzeugt. Diese Initialisierung kann auch vor der Simulation vorgenommen werden.

4 Programmstruktur

4.1 Klassen

Alle im Programm verwendeten Instanzen der Klassen TRoute, TVehicle, TRandom, TDXFHandler werden von dem Objekt TForm gesteuert. TForm stellt zudem das Interface für den Benutzer dar, mit dem er eine Simulation vorbereiten und starten kann. Die Funktionen und Eigenschaften der Klassen sind im Anhang aufgeführt.

4.1.1 TRoute

Das Objekt TRoute speichert den Verlauf und die Eigenschaften eines Streckenabschnittes. Hier sind die Algorithmen zur Routensuche und Geschwindigkeitsberechnung implementiert. Des weiteren erfolgt hier das Zeichnen der Route auf einem `Canvas`. Mehrere Funktionen des Objekts unterstützen das Erstellen der Route.

4.1.2 TVehicle und TTrailer

Die Klassen TVehicle und TTrailer stellen gemeinsam die Fahrzeuge dar. Die Fahrzeuge werden von außen nur über Instanzen der Klasse TVehicle angesprochen. TVehicle repräsentiert nur die Vorderachse des Fahrzeugs und dessen Fahrverhalten. Das Aussehen des Fahrzeugs legt TTrailer fest. Ein TTrailer-Objekt stellt den Anhang hinter der Vorderachse dar. Er besteht aus einer Deichsel einer Achse und deren Abstand zur Kupplung, an die ein weiteres TTrailer-Objekt angehängt werden kann. Die Bewegung der einzelnen Fahrzeuganhänge wird von einer in TTrailer eingebundenen Funktion durchgeführt.

Die tatsächlich verwendeten Objekte für die Anhänge sind von TTrailer abgeleitet und sind speziell für jeden Fahrzeugtyp angepasst. Von TTrailer sind TAuto, TZugmaschine, TAuflieger und TBoot abgeleitet. Diese besitzen keine neuen Eigenschaften sondern nur einen eigenen Konstruktor und eine eigene Paint-Anweisung. Direkte Anhänge an ein TVehicle-Objekt können vom Typ TAuto oder TZugmaschine sein. An diese können wiederum Objekte vom Typ TBoot oder TAuflieger angehängt werden.

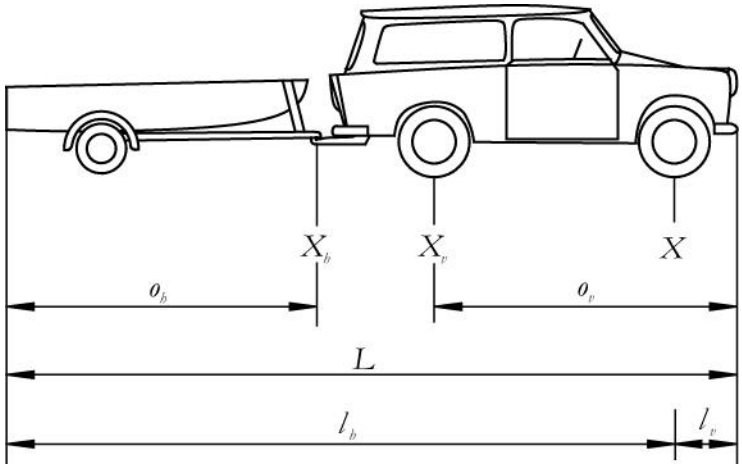

Abb. 4-1: Fahrzeuggeometrie.

4.1.3 TRandom

Alle Objekte, die statistisch verteilte Daten benötigen, verwenden die selbe Instanz eines TRandom-Objekts, um Zufallgrößen zu erzeugen. TRandom fasst mehrere C-Algorithmen für Zufallszahlen zusammen. Dies bietet sich an, da nun die Initialisierung der Zufallsgeneratoren nur noch in einer Klasse stattfindet.

4.1.4 TDXFHandler

Die Geometrischen Daten der Routen werden durch Linien und Polylinien einer DXF-Zeichnung definiert. Die Zeichnung wird von dem Objekt TDXFHandler eingelesen. Des weiteren können alle eingelesenen Daten auf einem `Canvas` dargestellt werden. Die Daten können auf eine Koordinate hin untersucht werden. Dabei wird das der Koordinate nächste Objekt gefunden und zurückgegeben.

4.1.5 TForm

TForm stellt zum einen die grafische Oberfläche des Programms dar, zum anderen verknüpft es alle Klassen und koordiniert die Simulation. Die Ausgangsdaten der Simulationen können in einem eigenen Dateiformat gespeichert und wieder geladen werden.

5 Ergebnisse

5.1 Zusammenfassung

Mit der entwickelten Simulation ist es möglich, verschiedene Verkehrsnetze einzulesen und zu simulieren. Dabei kann das Fahrverhalten der verschiedenen Fahrzeuge berücksichtigt werden. Zur Kalibrierung und Validierung des Fahrverhaltens werden die Messungen auf den Fähren herangezogen.

5.2 Kalibirierung und Vergleich mit der Scandlines-Fähre

Zur Kalibrierung der Parameter wurden die Pkw-Decks der Scandlines-Fähre gewählt, da hier die einfließenden Faktoren leicht voneinander getrennt werden können. Eingestellt wurden die Beschleunigungswerte der Pkw und die Wahrscheinlichkeiten P_{dawdle} und P_{sts}. Als Randbedingungen wurden die Maximalgeschwindigkeiten auf den Routen gesetzt, wie sie auch in der Messung zu finden waren.

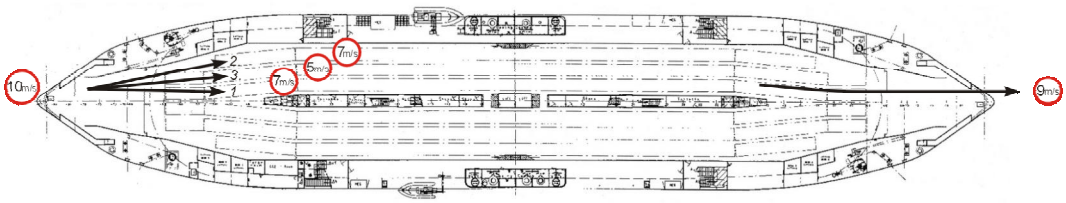

Abb. 5-1: Maximale Geschwindigkeiten in der Simulation.

Im Beladungsfall wurde zunächst die innere, anschließend die äußere und zuletzt die mittlere Spur beladen. Da das Entladen spurweise stattfand und so keine Interaktion zwischen den Spuren auftrat, wurde für die Kalibrierung nur die Entladung der inneren Spur simuliert. In der Simulation wurde die Geschwindigkeit und der Abstand vor den Fahrzeugen an den gleichen Stellen wie im Schiff ausgegeben.

Je größer das Bremsvermögen der Fahrzeuge ist, umso kleiner wird bei gleicher Geschwindigkeit der zeitlichen Abstand zwischen den Fahrzeugen. Bei einer größeren Maximalbeschleunigung bleibt das Verhältnis von Geschwindigkeit zu zeitlichem Abstand konstant, die maximale Geschwindigkeit wird jedoch erst später erreicht. Eine gute Übereinstimmung mit der Messung wird bei einer maximalen Beschleunigung von 1 m/s^2 und einer Bremsbeschleunigung von -2 m/s^2 erreicht. Die Trödelwahrscheinlichkeiten betragen dabei P_{dawdle} = 50 % und P_{sts} = 70 %.

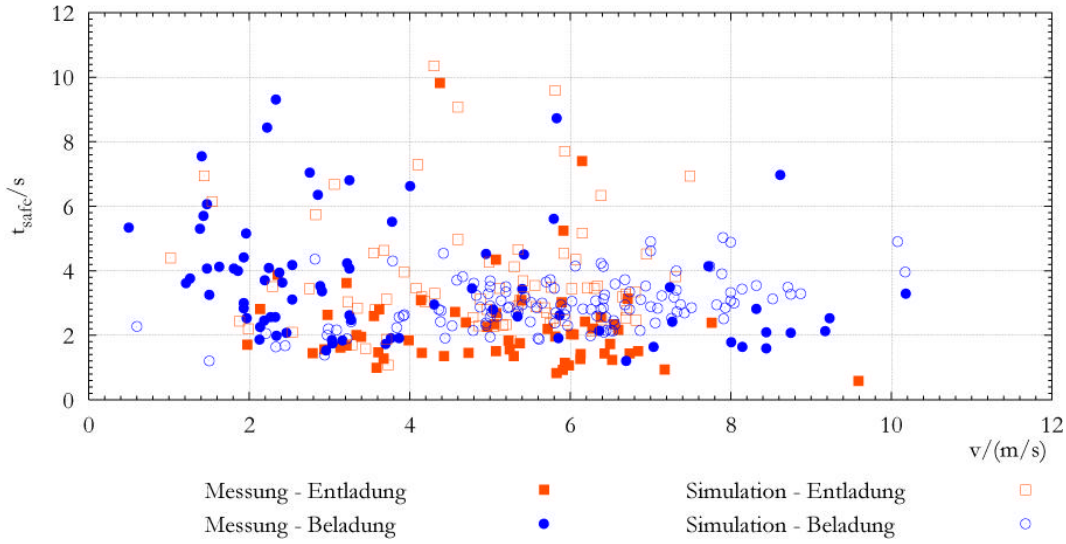

Abb. 5-2: Vergleich von Messung und Simulation.

In Abb. 5-2 sind in blau die gemessenen Fahrzeuge bei der Beladung und in rot bei der Entladung dargestellt. Die ausgefüllten Punkte wurden auf dem Schiff gemessen, die übrigen wurden von der Simulation ausgegeben. Die Punkte der Messung liegen im gleichen Bereich wie die der Simulation, sowohl bei der Be- als auch bei der Entladung. Insgesamt stimmt die Simulation gut mit der Messung überein. Auch ein Vergleich der Gesamtbeladezeiten liefert gute Ergebnisse. In der Simulation werden durchschnittlich 279 s benötigt, um die Fähre zu beladen. Während der Messungen wurden auf der Scandlinesfähre Beladezeiten von 241 s, 266 s und 300 s gemessen. Diese liegen alle in dem von der Simulation ermittelten Intervall der längesten und schnellsten Ladedauer zwischen 235 s und 350 s.

5.3 Vergleich von Entladetaktiken

Ausgehend von der in der Kalibrierung gewonnnenen Einstellung, wurden nun exemplarisch zwei verschiedene Möglichkeiten der Zusammenführung von Routen untersucht. Im Folgenden ist eine Reißverschlusszusammenführung (also zwei Routen mit gleicher Priorität) mit einer Zusammenführung verglichen, in der einer Route der Vorzug gegeben wird.

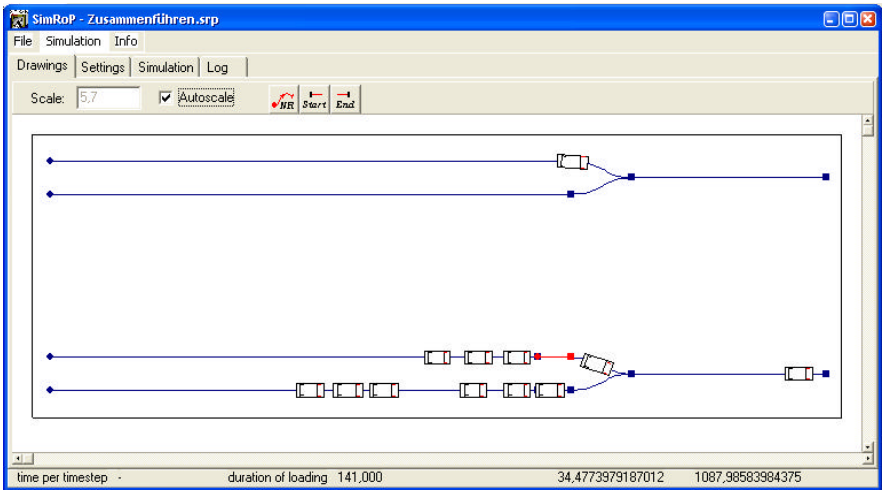

Abb. 5-3: Vergleich der Entladetaktiken.

Das Diagramm zeigt die Zeitpunkte zu denen die Fahrzeuge die Zusammenführung erreichen. Es ist zu erkennen, dass der Fluß (in Fahrzeuge pro Zeit) bei der Entladung der beiden Spuren nacheinander größer ist als bei dem Reißverschlussverfahren. Die Stufe bei der spurweisen Entladung nach ca. 90 s entsteht bei dem Wechsel zwischen der ersten und zweiten Spur. Dagegen blockieren sich die Fahrzeuge im Reißverschlussverkehr während des Einfädelns, was zu kleineren Geschwindigkeiten und so zu einem kleineren Fluß führt.

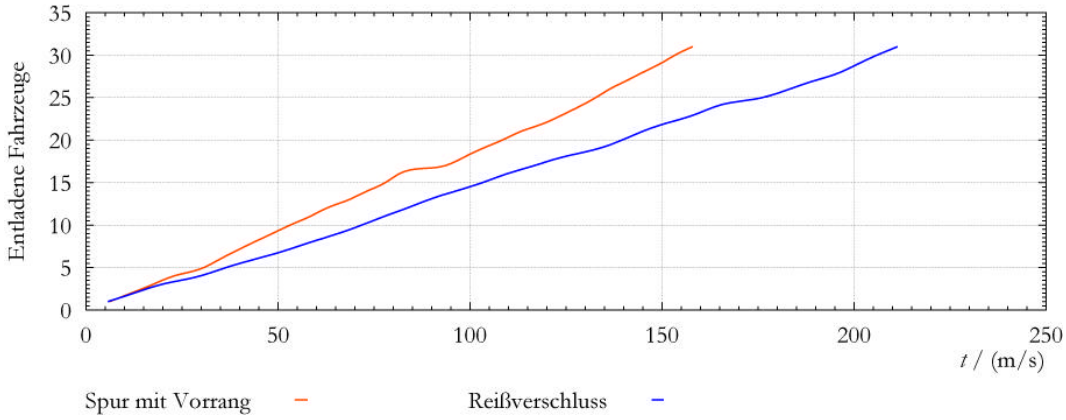

Abb. 5-4: Entladezeiten.

5.4 Ausblick

Geometrien und Abläufe, die nicht durch die Routenverknüpfung „Aufsplitten“ oder „Zusammenführen“ und die Eigenschaften „Parken“ und „Parkverbot“ regelbar sind, erfordern eine Erweiterung des Modells. Das selbständige Rangieren mehrerer Fahrzeuge auf einem Deck beispielsweise, kann nicht durch diese Regeln simuliert werden. Auch der nichtbegleitete Verkehr stellt eine Schwierigkeit dar, nicht nur weil hier die gleiche Route aus verschiedenen Richtungen benutzt wird. Auch das Rangieren des Trailers und das Abstellen des Trailers, was eine Veränderung des eigentlichen Fahrzeugs zur Folge hat, ist ein weiteres Problem.

Weitere komplexe Geometrien sind Verkehrsnetze, wie sie in Parkhäusern zu finden sind. Hier gibt es Abläufe, wie rückwärts Ein- oder Ausparken, die die Eigenschaften des Wegenetzes kurzzeitig beeinflussen. Hierin steckt auch eine Möglichkeit zur Lösung dieser komplexeren Abläufe. Ereignisbezogen müssen die Eigenschaften der angrenzenden Routen verändert werden. Um ein Rangieren zu ermöglichen, könnten die Nachfolger einer Route umgeändert oder deren Eigenschaften wie „Parken“ oder „V_{max}“ kurzzeitig geändert werden. Auch die Einführung einer Totzeit für das Verfahren einer Rampe oder das Abstellen eines Trailers erscheint sinnvoll. Mit der Erweiterung geht allerdings auch ein nicht unerheblicher Modellieraufwand einher, der bei großen Parkflächen erheblich sein kann. Das stellt zudem große Herausforderungen an einen Preprocessor. Durch kleine Änderungen an den Regeln der aufsplittenden Routenverbindungen könnte das Modell auch dazu genutzt werden Stadtverkehre zu simulieren. Hier steht bei der Auswahl der Nachfolgeroute nicht wie bei den Ladeprozessen die Suche nach einem Parkplatz, sondern eher eine Gewichtung der Anschlüsse nach deren Nutzung im Vordergrund.

6 Literatur

[1] Hompage der Howaldtwerke Deutsche Werft AG (*HDW*)
www.hdw.de

[2] Homepage der TraffGo GmbH (*TraffGo*)
www.traffgo.com

[3] Jochen Hundemer, *Ladeanalyse StenaLine*, SimRoP-Projekt

[4] Jochen Hundemer, *Ladeanalyse Scandlines*, SimRoP-Projekt

[5] Jochen Hundemer, *RoRo Loading Analyses Superfast*, SimRoP-Projekt

[6] Homepage der StenaLine
www.stenaline.de

[7] Homepage von Scandlines
www.scandlines.de

[8] Homepage von Superfast
www.superfast.com

[9] Ehrenstein, Walter: Fechnersches Gesetz, in Lexikon der Psychologie in 5 Bänden. Band 2, Heidelberg, Berlin:Spektrum, Akad. Verl. 2001

[10] D. Chowdhury, L. Santen and A. Schadschneider, *Statistical Reviews of Vehicular Traffic and Some Related Systems*, Institut für Theoretische Physik, Universität zu Köln.

[11] Krauss, P. Wagner and C. Gawron, Phys. Rev. E54, 3707(1996); 55, 5597 (1997)

[12] S.C. Benjamin, N.F. Johnson and P.M Hui, J. Phys. A: Math. Gen. 29 (1996) 3119–3127.

7 Danksagung

Mein Dank gilt den Initiatoren dieser Studienarbeit HDW und TraffGo. HDW möchte ich für den Kontakt zu den Reedern und die schiffbauliche Betreuung danken. Bei TraffGo habe ich jederzeit offene Gesprächspartner zur Diskussion der Algorithmen und Vorgehensweise gefunden. Darüberhinaus möchte ich mich bei den Reedereien Stena-Lines, Scandlines und Superfast und deren Mitarbeitern bedanken, die uns bei den Messungen an Bord ihrer Schiffe zuvorkommend untestützt haben.

Anhang

TRoute

Strukturen

```
typedef struct {
     double phi;
     float L;
} TSECTION
```

Aus Elementen der Struktur `TSECTION` setzt sich der Polygonenzug der Route zusammen.

```
enum TRouteTactics{ RTfirst,
     RTNoParking,
     RTAllowOverhang,
     RTisOpenEnd,
     RTisStartRoute_always,
     RTisStartRoute_init,
     RTLoadDuringInit,
     RTLogVER,
     RTSumSpeed,
     RTlast}
```

Die Menge TRouteTactics enthält Taktiken, die den Ladevorgang einer Route beeinflussen. Diese sind im Kapitel 3.3.2 erläuatert.

Eigenschaften

`AnsiString Caption`	Name der Route.
`float x0, y0, xe, ye`	Anfangs- und Endpunkt.
`Float L`	Länge der Route.
`TSECTION *Lane`	Liste der Linienabschnitte.
`Float nSections`	Anzahl der in `Lane` gespeicherten Linienabschnitte.
`Set tactics`	`tactics` speichert einige Optionen, die das Verhalten der Route beschreiben. Das Set ist vom Typ `TRouteTactics` Zur Beschreibung der einzelnen Eigenschaften siehe S. 16.
`int Priority` `EnterPriority`	Priorität mit der die Route beim Aufsplitten gewählt wird. Vorfahrtspriorität beim zusammenführen der Routen. Höchste Priorität `Priority = 0`, geringere Priorität `Priority > 0`
`TRoute **follow` `*ParentRoute`	Zeiger auf die Liste der Nachfolgerouten. Zeiger auf die Vorgängerroute.
`int nfollows` `nparents`	Anzahl der Folgerouten. Anzahl der Vorgängerrouten.

`TVehicle **Vehicles`
`*LastStored,`
`*FirstStored`
`*UsedBy`
	Zeiger auf die Liste der Fahrzeuge einer Route.
	Wenn ein Fahrzeug am Routenende übersteht zeigt `LastStored` auf dieses Fahrzeug.
	Wenn ein Fahrzeug am Routenanfang übersteht zeigt `FirstStored` auf dieses Fahrzeug.
	Wenn mehrere Routen zu dieser Route zusammengeführt werden zeigt `UsedBy` auf das Fahrzeug, das diese Route benutzen will, sich aber noch auf einer andern Route befindet.

`int nVehicles`	Anzahl der Fahrzeuge.
`TRandom *GetRandom`	Zeiger auf das Zufallsgeneratorobjekt.

`float P_car,`
`P_cartrailer`
`P_trucktrailer`
	Mit einer Wahrscheinlichkeit von `P_car` wird ein Fahrzeug vom Typ `vtPkw` sonst vom Typ `vtLkw` erzeugt.
	`P_cartrailer` und `P_trucktrailer` geben ob, ein Pkw oder Lkw einen Anhänger hat oder nicht. Die Wahrscheinlichkeiten haben nur wenn in `tactics` die Eigenschaft `RTisStartRoute_init` oder `RTisStartRoute_allways` gesetzt ist.

`float P_dawdle` `P_sts`	Wahrscheinlichkeit mit der ein Fahrzeug auf einer Strecke Trödelt. Wahrscheinlichkeit mit der ein stehendes Fahrzeug nicht anfährt.
`float Vmax`	Auf der Route maximal zugelassene Geschwindigkeit

`float StoreA`
`StoreB`
`StoreC`
	`StoreA` speichert; wenn das letzte Fahrzeug auf der Route übersteht, die durch das Fahrzeug auf dieser Route belegte Strecke.
	`StoreB` speichert die Summe von V->L+V->PD von allen auf der Route parkenden Fahrzeugen.
	`StoreC` speichert, wenn das erste Fahrzeug auf der Route übersteht, die durch das Fahrzeug auf dieser Route belegte Strecke.

`float FreeLength`	Ist die nicht für Fahrzeuge reservierte Länge einer Route und wird von der Funktion `FL` berechnet.

Funktionen

`TRoute (void)`	Im Konstruktor werden alle Zeiger `NULL` gesetzt und besimmte Initialisierungswerte gesetzt.
`~TRoute (void)`	Im Destruktor werden der reservierte Speicher freigegeben.

`float Distance (int Vehicle1, int Vehicle2)`	Berechnet den Bahnlinien abstand zwischen zwei Fahrzeugen.
`void CalcDistances (void)`	Bestimmt mit der Funktion Distance den Abstand vor allen Fahrzeugen die sich auf der Route befinden.
`void CalcAcceleration (float dt)`	Berechnet die Beschleunigung der Fahrzeuge für den nächsten Zeitschritt der Länge `dt`.
`void ChangePositions (float dt)`	Bewegt alle Fahrzeuge der Route mit der aktuellen Geschwindigkeit um den Zeitschritt `dt` weiter.
`void GetParking (bool init)`	Sucht für das erste Fahrzeug der Route einen neuen Parkplatz. Dafür wird zunächst mit `FreeSpace` der reservierte Platz freigegeben, dann sucht `CheckRoute` nach einem Parkplatz, der zum Schluss von `StoreVehicle` reserviert wird. Ist `init == true` wird die Simulation gerade initalisiert und die Suche nach einem Parkplatz wird nur in Routen durchgeführt bei den `RTLoadDuringInitialisation` gesetzt ist.
`float CheckRoutes (TVehicle *V, int tiefe, bool init)`	Sucht rekursiv im nachfolgenden Routenbaum nach einem Parkplatz für das Fahrzeug `V`.
`void StoreVehicle (TVehicle *V)`	Reserviert mit der Funktion `ReserveSpace` einen Parkplatz für das Fahrzeug `V` und verknüpft die Eigenschaften `UsedBy` und `ParentRoute`.
`Void ReserveSpace (TVehicle *V, bool refit)`	Reserviert in den Variblen `StoreA, StoreB, StoreC` Platz für das Fahrzeug `V`. Verwaltet zudem die Variablen `LastStored, FirstStored`.
`void FreeSpace (TVehicle *V, bool refit)`	Gibt den von `ReserveSpace` reservierten Platz wieder frei und löst die daraus entstandenen Verknüpfungen wieder auf. Die Variable `refit` gibt an, ob anschließend die Funktion `CheckFreeLength` ausgeführt werden soll.

`void CheckFreeLength (void)`	Überprüft, ob durch die Änderung der Überstände Platz frei geworden, und passt die weiteren Überstände neu an.
`void SetJumpedFalse (void)`	Fahrzeuge, die nach einem Zeitschritt die Route wechseln, erhalten den Stempel `isjumped`, dieser wird nach dem Zeitschritt durch die Funktion wieder gelöscht.
`void MoeXY (int Vehicle, float s,float dt)`	Bewegt das i-te Fahrzeug um das Wegstück s die in der Zeit dt zurückgelegt werden soll.
`void PaintRoute (Tcanvas *C, float X0, float Y0, float Scale, Tcolor all, Tcolor last)`	Stellt die Route auf dem Canvas `C` in dem Koordinatensystem `X0`, `Y0` und dem Saclierungsfaktor `Scale` dar. Die zur letzte Gruppe zusammengefassten Linien werden mit der Farbe `last` die andern mit der Farbe `all` gezeichnet.
`bool AddVehicle (TvehicleType car)`	Erzeugt auf der Route ein neues Fahzeug vom Typ car.
`void AddVehicle (TVehicle *Vehicle, float dx, floa dy, double dphi)`	Fügt der Route ein bereits vorhandenes Fahrzeuug der Route hinzu und verschiebt dess Koordinaten um dx,dy und dphi.
`void DeleteVehicle (int V)`	Löscht das i-te Fahrzeug einer Route.

`void AppendSection (float L, double phi)`
`void AppendSecPoint (float x, float y)`
`void SetGroupmark (void)`
`void ReverseGroup (void)`
`void DeleteGroup (void)`
	Die Funktionen werden für die Erstellung der Route benötigt. Mit den Append-Funktionen werden einzelne Abschnitte angehängt. Einzelne Anschnitte der Route können zu einer Gruppe zusammengefasst werden. Mit ReverseGroup wird die oreintierung der letzten Gruppe geändert. Die letzte Gruppe wird mit DeleteGroup gelöscht.

`void AddFollwer (TRoute *follower)`
`void RemoveFollower (TRoute *follower)`
`void RemoveAllFollowers (void)`
	Nachfolgeroute werden mit den Funkioen dem Zeiger *follow hinzugefügt oder daraus entfernt.

`void SetPriority (int prio)` `bool ChangePriority (TRoute R, int prio)`	Die Eigenschaft `Priority` wird mittels der beiden Funktionen verändert, und die Elemente der List `*follow` mit aufsteigender `Priority` sortiert.
`void SetToOpenendRoute (int count)`	Initialisiert eine Senkenroute.

`void SetToStartRoute (int count)`	Initialisiert eine Quellroute.

TVehicle und TTrailer

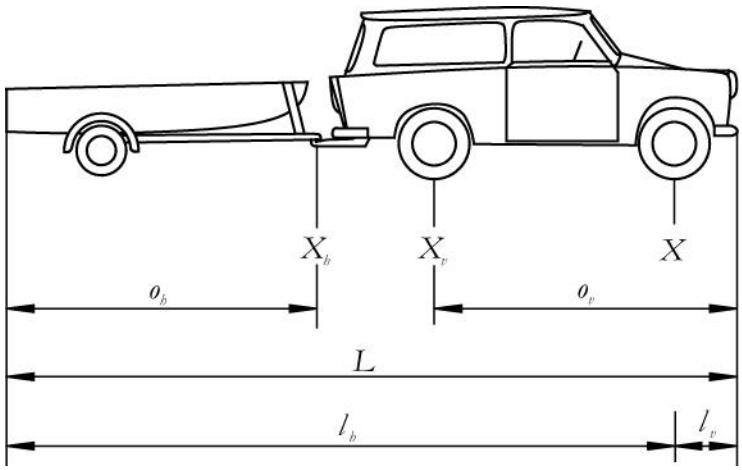

Abb. 7-1: Fahrzeuggeometrie.

Allgemein:

Strukturen

`enum TVehicleType{ vtNone, vtLKW, vtPKW}`	Legt den Typ des ersten Anhangs fest.
`enum TApendageType{ atNone, atAuflieger, atBoot}`	Legt den Type eines weiteren Anhangs fest.
`typedef struct { float x, y, v; double alpha; }TVehicleCOORD;`	Beschreibt eine Fahrzeugkoordinate mit Ort, momentaner Geschwindigkeit und Richtung.

TVehicle

Eigenschaften

`float Position`	Speichert die aktuelle Position eines Fahrzeugs auf der Route.

`float L`
`float lv, lh, oh, ov`
`TVehicleCOORD x, xv, xh`
	Die Längen und Koordinaten des Fahrzeuganhänge werden durch Funktionen berechnet. Siehe dazu Abb. 4-1.

`float V,Vmax`	Aktuelle und maximale Geschwindigkeit des Fahrzeugs. Vmax verweist auf eine in den Anhängen gespeicherte Variable.
`float a,acc,dcc`	Aktuelle, maximale und minimale Fahrzeugbeschleunigung.
`float PD`	Abstand zum Vordermann beim Parken.
`float d`	Abstand zum nächsten Fahrzeug.
`TTrailer *trailer`	Zeiger auf die Anhänge des Fahrzeugs.
`TVehicle *Vp`	Zeiger auf das vorrausfahrende Fahrzeug.
`TRoute *ParkRoute, *way[250]`	`ParkRoute` ist ein Zeiger auf die Route, die das Fahrzeug zum Parken nutzen kann. `way` speichert den Weg des Fahrzeugs dorthin. Das Fahrzeug befindet sich immer auf der Route `way[0]`.
`TRandom *GetRandom`	Zeiger auf das Zufallsgeneratorobjekt.

Funktionen

`TVehicle (TvehicleType VT,`
`  float x0, float y0,`
`  double phi, TRandom *GRnd)`
	Im Kontruktor werden Startwerte für jedes Fahrzeug übergeben, anhand derer das Fahrzeug initialisiert wird.

`~TVehicle (void)`	Im Destruktor wird der reservierte Speicher freigegeben.
`void Move (float dx,float` `  dy,float dt)`	Bewegt ein Fahrzeug samt seinen Anhängen um das Stück dx, dy.

`void PaintVehicle (Tcanvas *C,`
`  float x0, float y0,`
`  float Scale)`
	Zeichnet die Vorderräder und ruft die Paint Anweisungen der Anhänge auf.

`void AddTrailer` `  (TApendageType at)`	Fügt einen Anhang an ein bestehendes Fahrzeug an.
`void CalcL (void)`	Berechnet die Geomietrie des Fahrzeugs.
`void SetToStartOfRoute`	Verschiebt den Anhang um `dx`, `dy` und

`(float dx0, float dy, double dphi)`	dreht es um den Winkel `dphi`.

TTrailer

Eigenschaften

`float Vmax`	Maximale Geschwindigkeit des Fahrzeugs. Wird vom TVehicle-Objekt aufgerufen.
`TVehicleCOORD x, xh`	`x` ist Position des Anhangs. Wenn der Anhang der letzte Anhang eines Fahrzeugs ist gibt er die Koordinate seiner Aufhängung zurück, ansonsten verweist er auf seinen eigenen Anhang.
`TTrailer *trailer`	Zeiger auf weitere Anhänge.
`virtual char* type (void) = 0`	Zeiger auf den Namen des Fahrzeugtyps.

Funktionen

`~TTrailer (void)`	Destruktor.
`void Move (float v,double alpha, float dt)`	Bewegt den Anhang und ruft die `Move`-Anweisung der weiteren Anhänge auf.
`virtual void PaintTrailer (TCanvas *C, float x0, float y0, float Scale) = 0`	Zeichnet den Anhang und ruft die `PaintTrailer`-Anweisungen der weiteren Anhänge auf.
`void AddTrailer (TApendageType at)`	Fügt dem Fahrzeug einen weiteren Anhang an.
`void SetToStartOfRoute (float x0, float y0, double dphi)`	Verschiebt den Anhang um `dx`, `dy` und dreht es um den Winkel `dphi`.

Von TTrailer sind TAuto, TZugmaschine, TAuflieger und TBoot abgeleitet. Diese besitzen keine neuen Eigenschaften sondern nur einen eigenen Konstruktor und eine eigene Paint-Anweisung.

TRandom

Eigenschaften

`float LogBasis`	Alle Log-normalvereitelten Zufallszahlen werden zur Basis `LogBasis` brechnet.

Funktionen

`TRandom ()`	Der Konstruktor initialisiert die Zufallsgeneratoren
`void SetSeed (int seed1)`	Initialisiert die Zufallsgeneratoren mit `seed1`.
`int intrand (int num)`	Gibt eine gleichverteilte Integergröße zwischen 0 und `num`-1 zurück.
`float floatrand (float min, float max)`	Gibt eine gleichverteilte Floatgröße zwischen min und max zurück.
`bool Event (float propability)`	Gibt mit einer Wahrscheinlichkeit von `propability` `true` sonst `false` zurück.
`float Gauss (float Mean, float StdDev)`	Gibt eine Normalverteilte Floatgröße zurück.
`float LogNorm(float Mean, float StdDev)`	Berechnet eine Log-normalverteilte Floatgröße zur Basis `LogBasis`.

TDXFHandler

Strukturen

`typedef struct { float x, y, z; } TDXFPoint`	Struktur für eine Koordinate.
`typedef struct { TDXFPoint P1, P2; TColor Color; } TDXFLine;`	Struktur für eine Linie
`typedef struct { TDXFPoint *Points; TColor Color; int nP; } TDXFPolyline;`	Struktur für eine Polylinie
`typedef struct{ int type; void *Pointer; } TDXFStructure;`	Die Struktur enthält einen Zeiger auf eine Struktur vom Typ TDXFLine oder TDXFPolyline. type gibt anwelchen
	Gibt einen Zeiger auf ein TDXFLine,TDXFPolyline... Element zurück0:

	kein Element1:TDXFLine2:TDXFPolyline

Eigenschaften

`AnsiString Name`	Name der DXF-Datei.
`float xmax, ymax, zmax, xmin, ymin, zmin;`	Maximal und Minimalwerte der DXF-Objekte.

Funktionen

`bool open (char *fname)`	Lädt eine DXF-Datei.
`void PaintBitmap (Graphics::Tbitmap *Bmp, float Scale, float x0, float y0)`	Zeichnet die gelesenen Linien und Polylinen auf ein `Bitmap`.
`Int GetStructure(float x, float y, float margin, TDXFStructure* S)`	Gibt das der Koordinate (x,y) nächste Objekt zurück, wenn es von dieser nicht weiter als `margin` entfernt ist.